EVALUATION OF MATERIALS

Your opinion about the quality of out training materials is important to us! Please take a moment, and in the space below, tell us how you would rate the effectiveness of our materials. Whenever possible, use specific manual numbers and page numbers to note errors or areas of confusion. An instructor will collect this page from you when you complete the course. Thank you for your valuable input!

CRAY Y-MP

Hardware Training Volume

VOLUME 2

HTV-0834

This document cannot be sent outside the United States without a Validated Export License.

Cray Research, Inc. Unpublished Proprietary Information - All Rights Reserved.

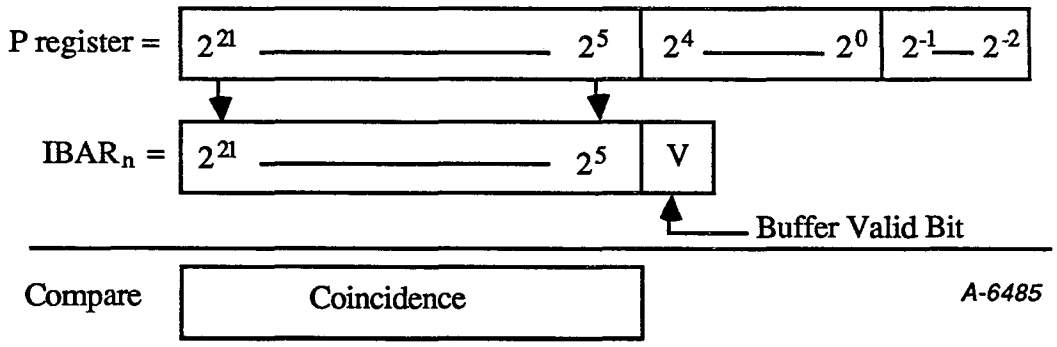
CRAY Y-MP FETCH OPERATION

The Fetch operation in the CRAY Y-MP is the process that transfers instructions from memory to the Central Processing Unit (CPU). This operation reads 32 consecutive words from memory and loads them into one of four instruction buffers. The CPU initiating the Fetch reads one word at a time from memory, competing for memory among the other CPUs. The instruction buffer is written and read separately, controlled by the (HA) option.

Fetch Sequence

An Exchange Sequence voids the instruction buffers by clearing the four IBAR buffer valid bits. This will cause Fetch to occur immediately after the exchange completes. During a Fetch, the IBAR associated with the instruction buffer being loaded receives the upper 17 bits of the P register.

The four IBARs are compared against the P register each clock period. If the upper 17 bits of the P register coincides with one of the IBARs and the valid bit is set, then an in-buffer or a coincidence condition exists and the proper instruction parcel is selected from the associated instruction buffer. If the P register does not coincide, then a no-coincidence condition exists, causing a Fetch operation. Instruction must be loaded into an instruction buffer before execution can continue.



The (YZ0) option checks for coincidence. If there is no coincidence, the (YZ) will generate a Fetch request and send the Port D reference to the (YK) option along with the address bits (2^1 - 2^4) to select the proper section and subsection. If a conflict exists among Port A - D, Port D would have the highest priority.

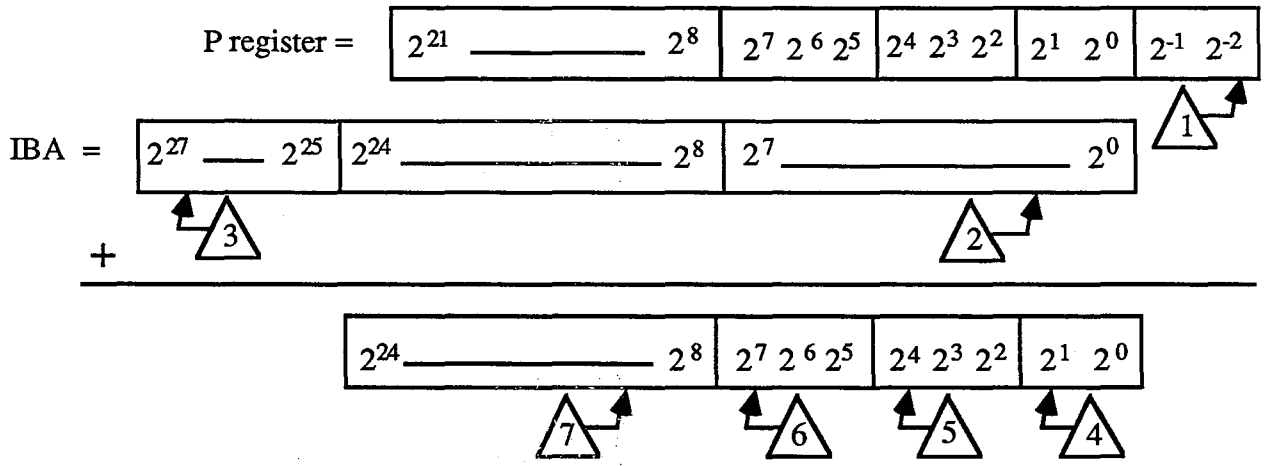
Accessing Memory Conflicts

The Fetch operation accesses the CRAY Y-MP memory for 32 consecutive CPs. However, each clock period could be held because of bank conflicts among the other CPUs. The (YK) option will check for any subsection conflicts among the other previously issued ports. If a conflict exists, the (YK) will suspend the request until the SS conflict is cleared. The (YL) option will check for any bank conflicts among the other CPUs. The CPU with the lowest lettered reference (A - H) would have the highest priority.

Hardware Trng.
YM10/25A J.E.S.

Fetch Address Generation

The (YH0) option adds P register bits (2^{21} - 2^5) to the IBA bits (2^{27} - 2^8). The Absolute Fetch address is then compared against the ILA, also held on the (YH) option, to determine if a Program Range Error exists.



A-6486

1. Parcel Select bits are not used in Memory Word address.

2. Lower 8 bits of IBA are forced to zero so Instruction Base address starts on 400_8 word boundaries.

3. Upper 3 bits are not used on the CRAY Y-MP 8/32 but will allow future expansion to 256 million words of memory.

4. Section Select bits (2^0, 2^1) select one of four sections.

5. Subsection Select bits (2^4, 2^3, 2^2) select one of eight subsections within the sections.

6. Bank Selects will select one of eight banks within a subsection.

7. Internal address and 1-bit of chip select is used to address 32 million words.

Hardware Trng.
YM10/26 J.E.S.

Instruction Buffers

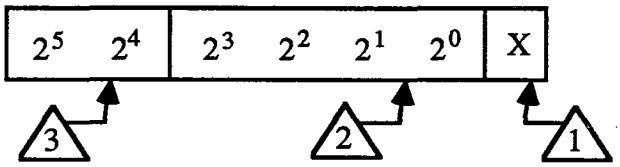
Each CPU module contains four instruction buffers each of which holds 128 consecutive 16-bit instruction parcels. The instruction buffers hold the instruction parcels and make them available to the NIP/LIP and LIP 1 registers, eight (HS) options contain the four instruction buffers which are divided by even and odd words.

The beginning instruction parcel in a buffer always has a Word address that is a multiple of 40_8. For each buffer there is an IBAR associated with it that contains this value.

Each (HS) option contains 16 bits of the 64-bit word. The 16 bits are in groups of 8 bits and the 8-bit groups are separated by 32 bits. For example, (HS0) contains bits (2^7 - 2^0) and (2^{39} - 2^{32}) of all the even words.

Loading Instruction Buffer

When a no-coincidence condition is generated on the (YZ) option, the (YZ) will generate a fetch request and set Port D active. The (YZ, YH) options will send control bits (2^4 - 2^1) to the (YK) to check for SS conflicts and also bank bits (2^7, 2^6, 2^5) to the (YL) to check for bank conflicts. The (YO, YN) option will delay the controls for the duration that the conflicts exist. Once the coast is clear, the (YN, YO) passes a Port D reference to the (YZ) option which will start a timer from CP 10 - CP 4. At CP 3, time the word which was read from memory is present on the (HS) option. The (YZ) will signal the (HA) to then supply the address to the (HS) option so the data can be loaded into the buffer. The (YZ, YH) will have, within the next clock period, generated the next request which would go through the same process as the first reference. To load the (HS) options, the (HA) will supply the (HS) with a 6-bit address (2^5 - 2^0). The 6-bit address was previously divided into even buffer or odd buffer to separate the Read and Write conflicts to the same options. The address breaks down as follows.



1. Bit position 2^0 on the (HA) is separated from the (HS) options as IB0 R/W and IB1 R/W.

2. Bit position (2^3 - 2^0) on the (HS) options identifies two, 8-bit bytes per buffer.

3. Bit position (2^5, 2^4) on the (HS) option identifies one of four instruction buffers (0 - 3).

A-6487

Hardware Tmg.
YM10/27 J.E.S.

Reading Instruction Buffer

Before the (HA) option can begin reading the instruction buffer, the Write address must be greater than or equal to the Read address, taking into account end-around. If the parcel which is to be read from the instruction buffer has been previously fetched from memory, the (HA) will supply the (HS0 - 3) if the address is even, or (HS4 - 7) if the address is odd, and in 2 CPs the (HS) option will output 4 parcels or an entire word. The (HC/HD) will select the proper parcels from 1 of 8 parcels if available from memory. To select the proper parcel, the (HC/HD) is given a 3-bit address which consists of (2^0, 2^{-1}, 2^{-2}) of the P register. The bypass path is not used on the (HS) option.

Control Dump Sequence

The Control Dump sequence is a maintenance feature which is built into the CRAY Y-MP to allow the four instruction buffers to be loaded from memory and then dumped back into memory. To enable the Control Dump (CD) the maintenance switch must be Enabled, for CD, followed by a Deadstart, and then a dead dump. The EAM bit in the Exchange Package should be Enabled. The CD sequence has bit 2^{10} forced in the address, so memory is read from 2000 - 2177, into the four instruction buffers. The instruction buffers will then be written back to memory along with the P register in locations 2200 - 2377 for IB0, 2403 - 2602 for IB1, 2606-3005 for IB2, and 3011 - 3210 for IB3. Memory can then be dumped to the MCU by using the dead dump lines. The MCU memory can then be examined to see if everything transferred correctly.

When filling the four instruction buffers, the path from memory would be the same as a normal Fetch operation. However, when reading the instruction buffers, instruction parcels LIP and LIP 1 are both selected in the (HC/HD) options and take there own path back to the (YC/YD) option. The path used is through extra fanout paths on the (VS, HZ) options. The (YC/YD) will assemble a 64-bit word to be written into memory which consists of the P register, LIP, and LIP 1.

CRAY Y-MP FETCH OPERATION (OPTIONS INVOLVED)

(HA0) Option

The (HA/HZ) options contain the P register. (HA0) contains P register bits (2^4 - 2^0, 2^{-1} - 2^{-2}) while (HB0) contains P register bits (2^{21} - 2^5). The (HA) is also responsible for supplying the IB Read address and the IB Write address. The IB Read address and Write address are divided into even and odd instruction buffers to interleave the reading and writing of the IBs. The (HA) also supplies the controls necessary for performing the Control Dump sequence.

Hardware Trng.
YM10/28 J.E.S.

(HZ0) Option

The (HZ) contains the upper P register bits (2^{21} - 2^{5}) which are loaded and compared against the IBAR each clock period to check for a no-coincidence condition.

(HS0 - HS7) Options

The (HS) options contain the storage for the instruction buffers. Each (HS) option contains 16 bits which are in two groups of 8 bits; the 8-bit groups are separated by 32 bits. The eight (HS) options are also divided into even word and odd word as they are written from memory. The bypass path on the (HS) option is not used, so it takes 2 CPs to get a parcel readout of the (HS) option. One CP to address the IB and another CP to readout the parcels, if the parcel were previously written from memory.

(YZ0) Option

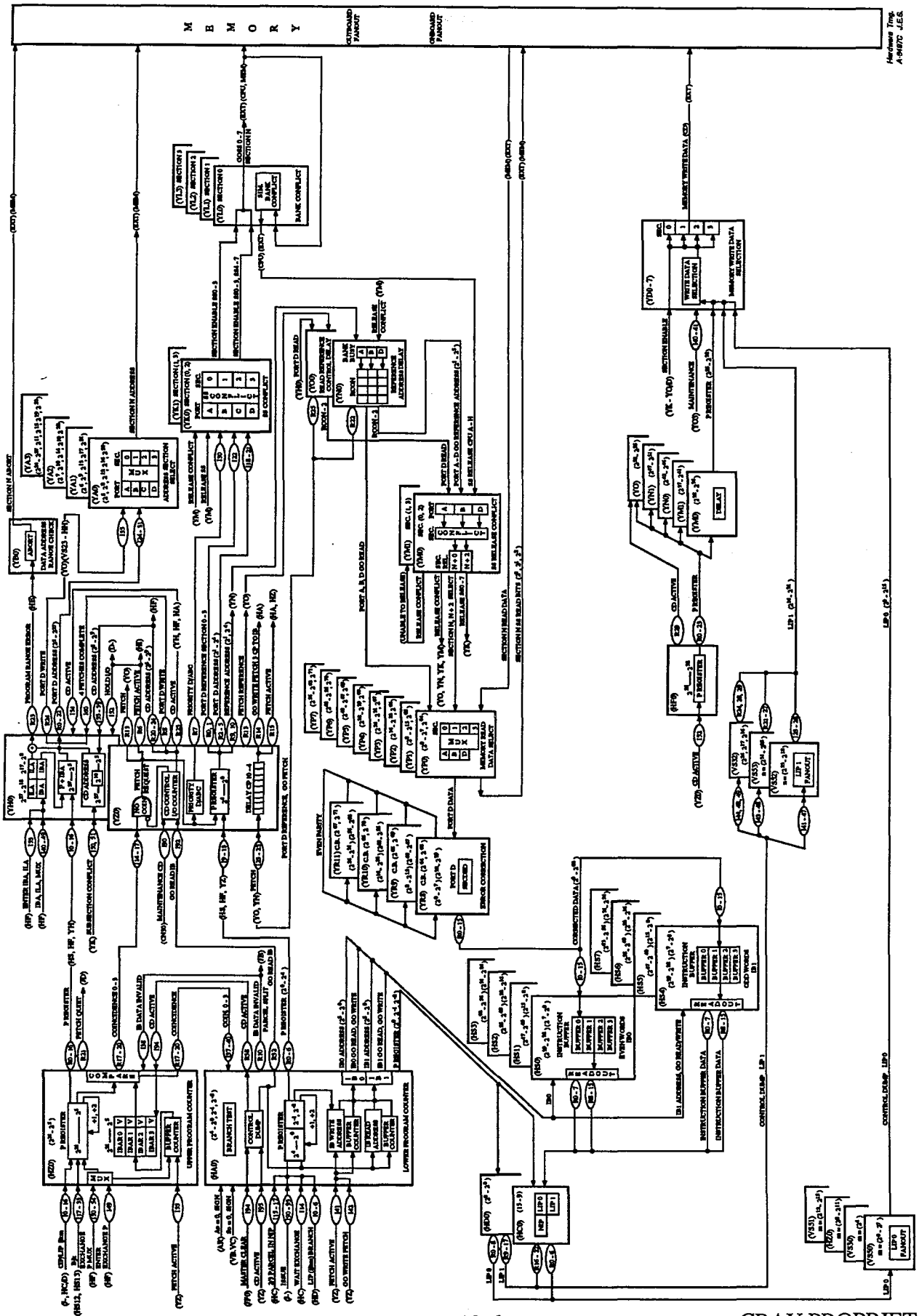
The (YZ) option checks for conflicts against the Ports A - D and allows Port D the highest priority when used by Fetch. The (YZ) contains the lower 5 bits (2^{4} - 2^{0}) of the P register which contains the control for sections and subsections. The (YZ) will check for the no-coincidence condition and generate the fetch request. The (YZ) will delay the control needed to load the data from memory into the instruction and supply these controls to the (HA) at the proper time. The (YZ) generates the initial CD control and also increments the lower address bits (2^{1} - 2^{5}) which is used to address memory during the CD sequence.

(YH0) Option

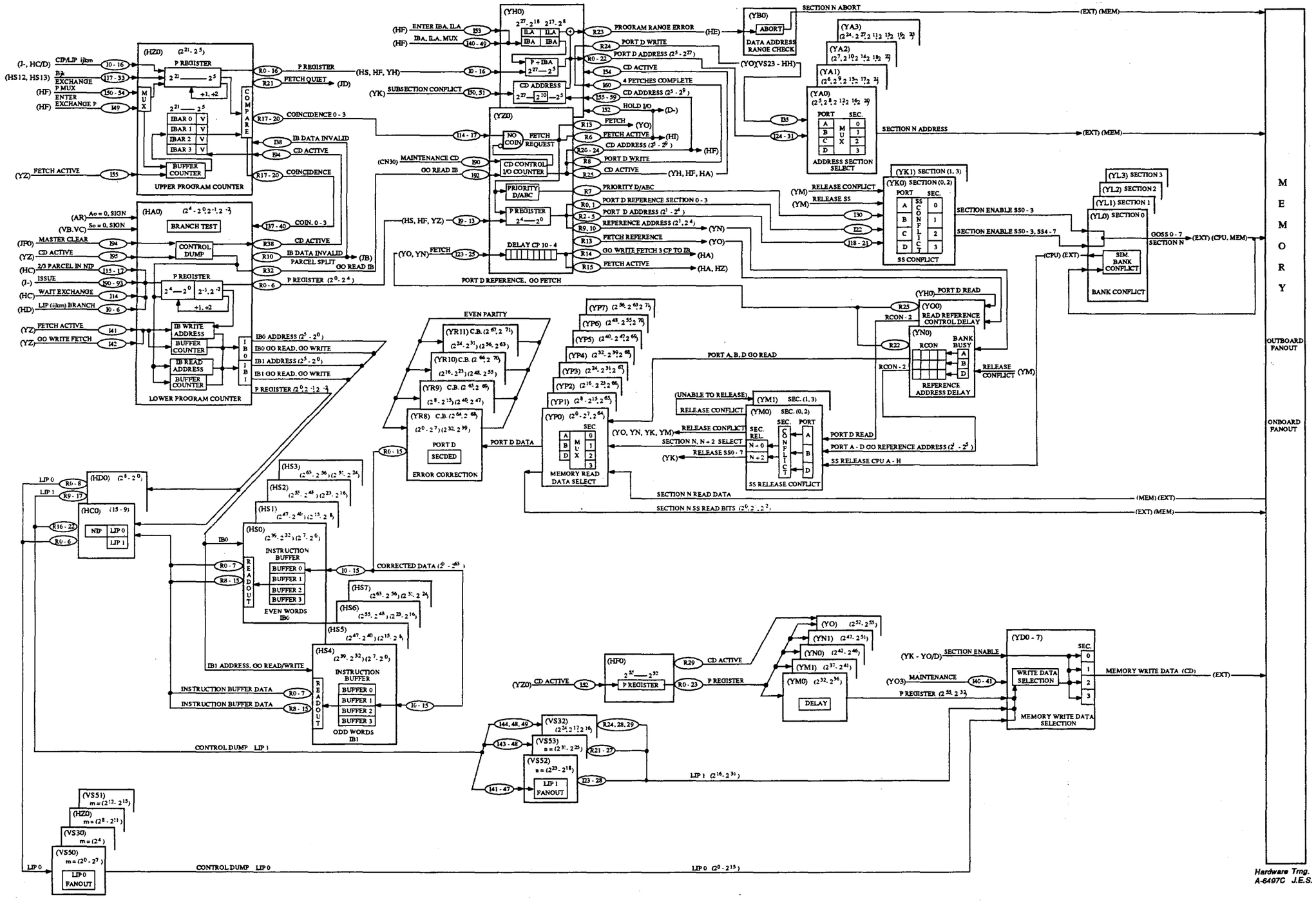
The (YH0) option will add P + IBA to create the Absolute Fetch address. The Absolute Fetch address is then checked against the ILA to see if the Fetch is within its limits, if not, the (YH) will generate a Program Range Error to the (YB) option.

(YA - YR) Option

The (YA - YR) option is similar to the operation covered in the Memory Control section of this manual. The only difference being the Fetch operation would be using Port D. Port D goes through the subsection conflict checks and bank conflict checks similar to what Port A, B, C would go through.

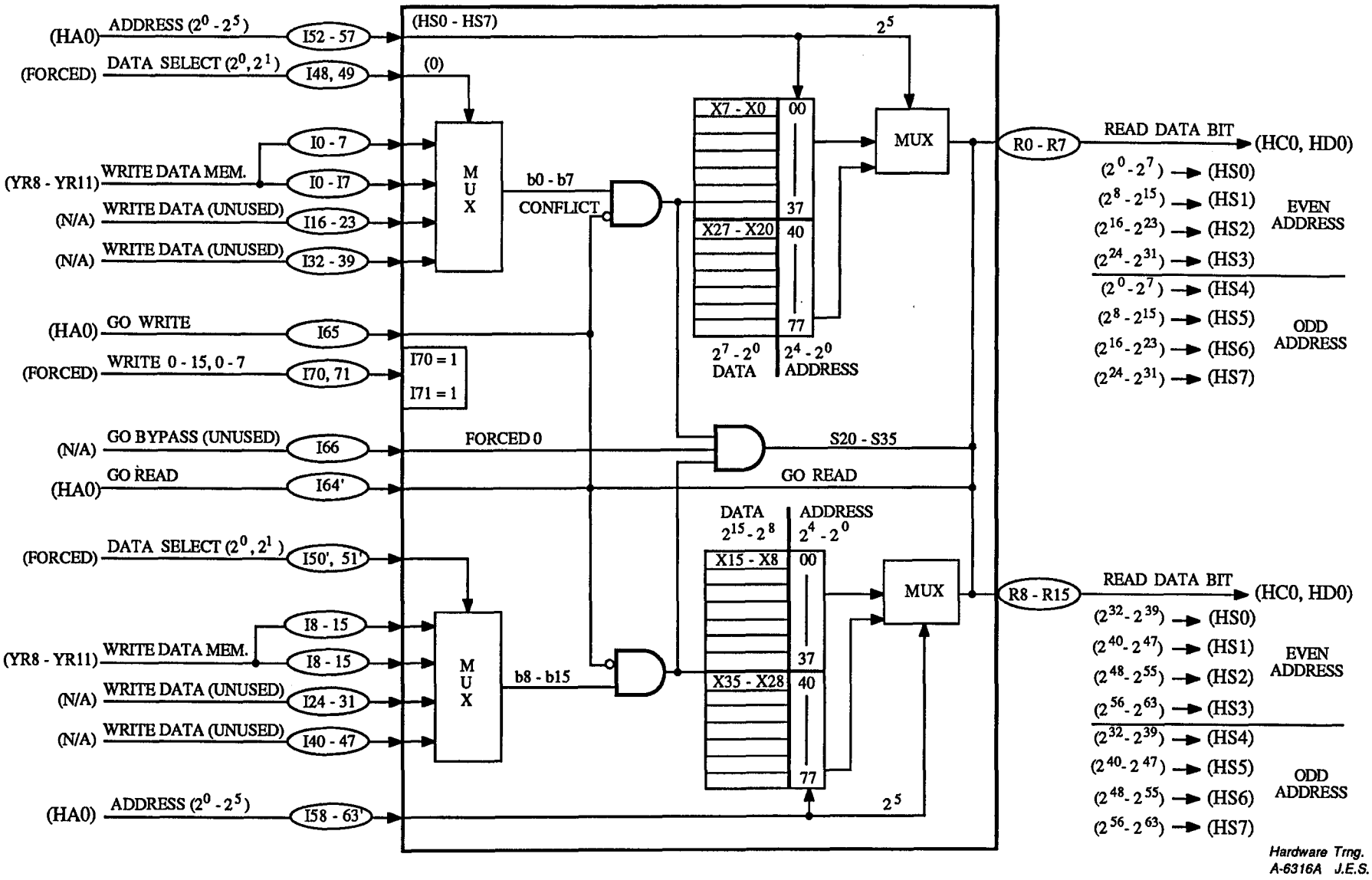


CRAY Y-MP FETCH BLOCK DIAGRAM AND CONTROL DUMP



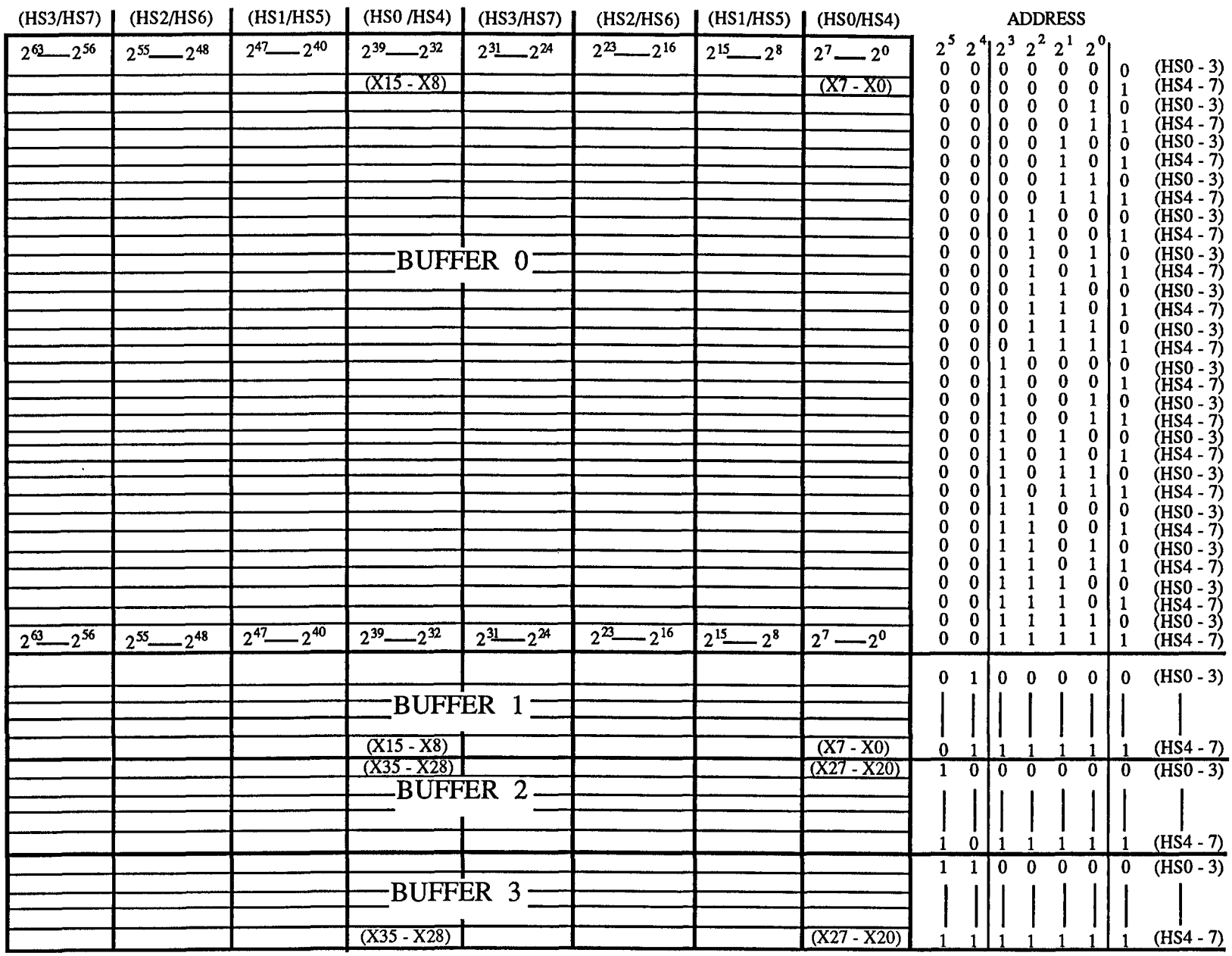
CRAY Y-MP FETCH BLOCK DIAGRAM AND CONTROL DUMP

CRAY PROPRIETARY



CRAY Y-MP (HS) BLOCK DIAGRAM
INSTRUCTION BUFFERS

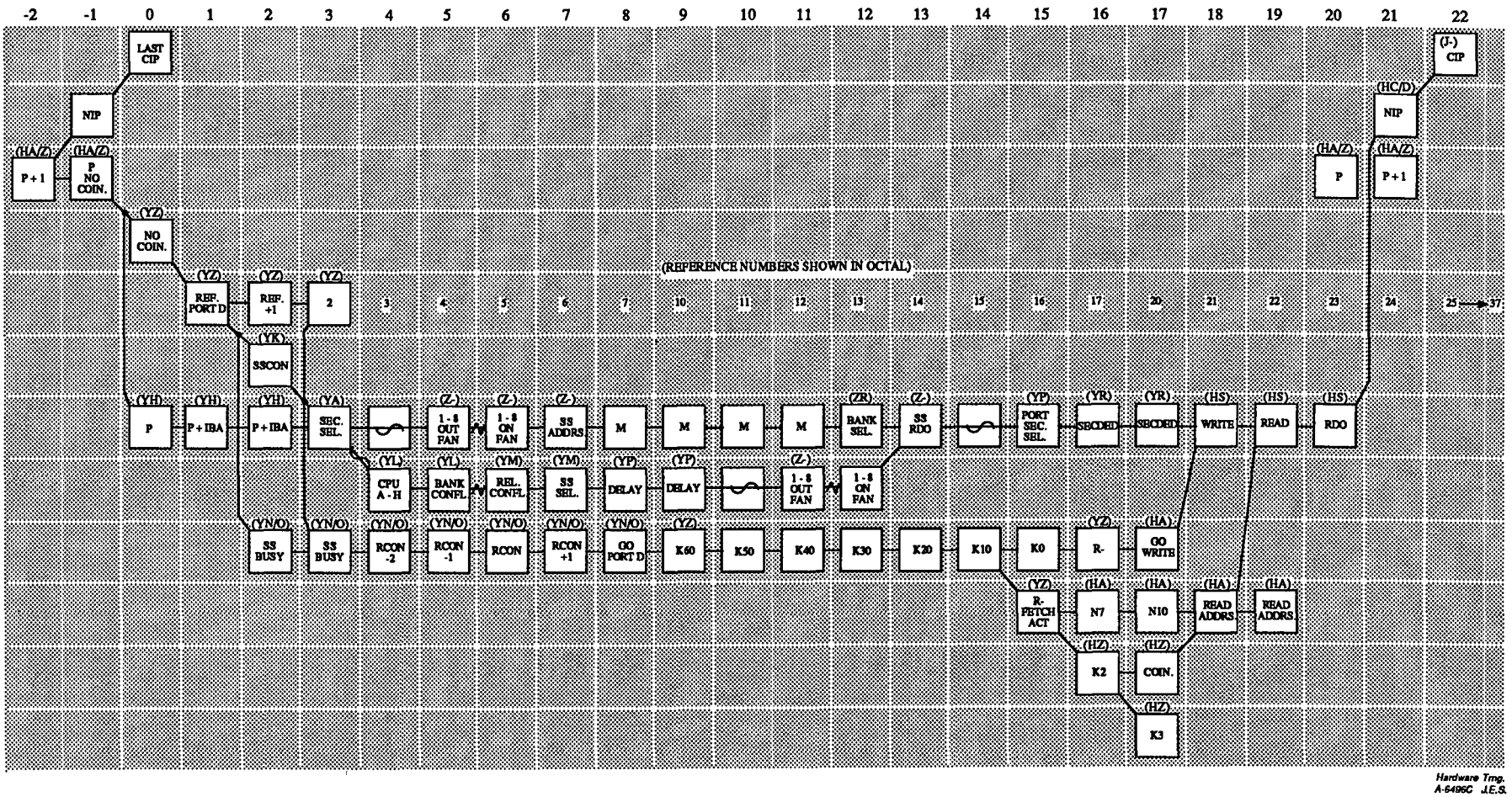
CRAY PROPRIETARY



CRAY Y-MP (HS) INSTRUCTION BUFFER FORMAT

Hardware Trng.
A-6488A J.E.S.

CRAY PROPRIETARY



CRAY Y-MP FETCH TIMING SEQUENCE

CRAY PROPRIETARY

CRAY Y-MP CONTROL DUMP

1. Enable Control Dump switch, Disable Error Correction switch

2. :DS - Deadstart

3. :DD - Dead Dump to Force Memory to I/O

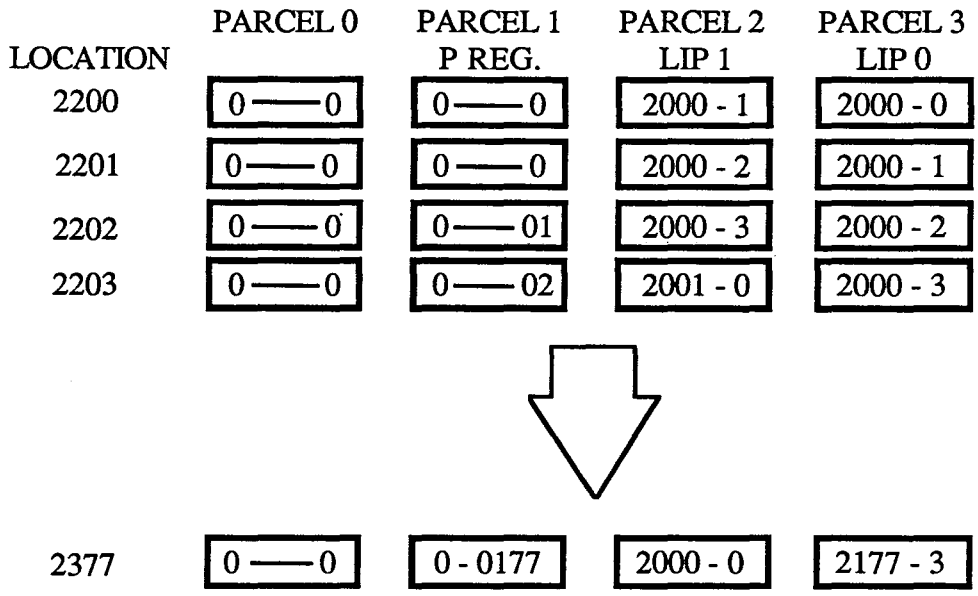
4. Set EAM - Set CRAY Y-MP mode in Initial Exchange Package

5. Dump Locations 2000 - 2177 to four instruction buffers.
   Locations 2000-2177 - Contain Memory instructions to be loaded into the four instruction buffers.

6. Four instruction buffers to memory breakdown:

   Location 2200 - 2377 = IB0
   Location 2403 - 2602 = IB1
   Location 2606 - 3005 = IB2
   Location 3011 - 3210 = IB3

7. Four instruction buffers to memory format:



```
00002000 001000 001001 001002 001003 202
00002001 001004 001005 001006 001007 050
00002002 001010 001011 001012 001013 050
00002003 001014 001015 001016 001017 202
00002004 001020 001021 001022 001023 050
00002005 001024 001025 001026 001027 202
00002006 001030 001031 001032 001033 202
00002007 001034 001035 001036 001037 050
00002010 001040 001041 001042 001043 050
00002011 001044 001045 001046 001047 202
00002012 001050 001051 001052 001053 202
00002013 001054 001055 001056 001057 050
00002014 001060 001061 001062 001063 202
00002015 001064 001065 001066 001067 050
00002016 001070 001071 001072 001073 050
00002017 001074 001075 001076 001077 202
00002020 001100 001101 001102 001103 050
00002021 001104 001105 001106 001107 202
00002022 001110 001111 001112 001113 202
00002023 001114 001115 001116 001117 050
00002024 001120 001121 001122 001123 202
00002025 001124 001125 001126 001127 050
00002026 001130 001131 001132 001133 050
00002027 001134 001135 001136 001137 202
00002030 001140 001141 001142 001143 202
00002031 001144 001145 001146 001147 050
00002032 001150 001151 001152 001153 050
00002033 001154 001155 001156 001157 202
00002034 001160 001161 001162 001163 050
00002035 001164 001165 001166 001167 202
00002036 001170 001171 001172 001173 202
00002037 001174 001175 001176 001177 050
00002040 001200 001201 001202 001203 050
00002041 001204 001205 001206 001207 202
00002042 001210 001211 001212 001213 202
00002043 001214 001215 001216 001217 050
00002044 001220 001221 001222 001223 202
00002045 001224 001225 001226 001227 050
00002046 001230 001231 001232 001233 050
00002047 001234 001235 001236 001237 202
00002050 001240 001241 001242 001243 202
00002051 001244 001245 001246 001247 050
00002052 001250 001251 001252 001253 050
00002053 001254 001255 001256 001257 202
00002054 001260 001261 001262 001263 050
00002055 001264 001265 001266 001267 202
00002056 001270 001271 001272 001273 202
00002057 001274 001275 001276 001277 050
00002060 001300 001301 001302 001303 202
00002061 001304 001305 001306 001307 050
00002062 001310 001311 001312 001313 050
00002063 001314 001315 001316 001317 202
00002064 001320 001321 001322 001323 050
00002065 001324 001325 001326 001327 202
00002066 001330 001331 001332 001333 202
00002067 001334 001335 001336 001337 050
00002070 001340 001341 001342 001343 050
00002071 001344 001345 001346 001347 202
00002072 001350 001351 001352 001353 202
00002073 001354 001355 001356 001357 050
00002074 001360 001361 001362 001363 202
00002075 001364 001365 001366 001367 050
00002076 001370 001371 001372 001373 050
```

```
00002077 001374 001375 001376 001377 202
00002100 001400 001401 001402 001403 327
00002101 001404 001405 001406 001407 175
00002102 001410 001411 001412 001413 175
00002103 001414 001415 001416 001417 327
00002104 001420 001421 001422 001423 175
00002105 001424 001425 001426 001427 327
00002106 001430 001431 001432 001433 327
00002107 001434 001435 001436 001437 175
00002110 001440 001441 001442 001443 175
00002111 001444 001445 001446 001447 327
00002112 001450 001451 001452 001453 327
00002113 001454 001455 001456 001457 175
00002114 001460 001461 001462 001463 327
00002115 001464 001465 001466 001467 175
00002116 001470 001471 001472 001473 175
00002117 001474 001475 001476 001477 327
00002120 001500 001501 001502 001503 175
00002121 001504 001505 001506 001507 327
00002122 001510 001511 001512 001513 327
00002123 001514 001515 001516 001517 175
00002124 001520 001521 001522 001523 327
00002125 001524 001525 001526 001527 175
00002126 001530 001531 001532 001533 175
00002127 001534 001535 001536 001537 327
00002130 001540 001541 001542 001543 327
00002131 001544 001545 001546 001547 175
00002132 001550 001551 001552 001553 175
00002133 001554 001555 001556 001557 327
00002134 001560 001561 001562 001563 175
00002135 001564 001565 001566 001567 327
00002136 001570 001571 001572 001573 327
00002137 001574 001575 001576 001577 175
00002140 001600 001601 001602 001603 175
00002141 001604 001605 001606 001607 327
00002142 001610 001611 001612 001613 327
00002143 001614 001615 001616 001617 175
00002144 001620 001621 001622 001623 327
00002145 001624 001625 001626 001627 175
00002146 001630 001631 001632 001633 175
00002147 001634 001635 001636 001637 327
00002150 001640 001641 001642 001643 327
00002151 001644 001645 001646 001647 175
00002152 001650 001651 001652 001653 175
00002153 001654 001655 001656 001657 327
00002154 001660 001661 001662 001663 175
00002155 001664 001665 001666 001667 327
00002156 001670 001671 001672 001673 327
00002157 001674 001675 001676 001677 175
00002160 001700 001701 001702 001703 327
00002161 001704 001705 001706 001707 175
00002162 001710 001711 001712 001713 175
00002163 001714 001715 001716 001717 327
00002164 001720 001721 001722 001723 175
00002165 001724 001725 001726 001727 327
00002166 001730 001731 001732 001733 327
00002167 001734 001735 001736 001737 175
00002170 001740 001741 001742 001743 175
00002171 001744 001745 001746 001747 327
00002172 001750 001751 001752 001753 327
00002173 001754 001755 001756 001757 175
00002174 001760 001761 001762 001763 327
00002175 001764 001765 001766 001767 175
```

```
00002176 001770 001771 001772 001773 175
00002177 001774 001775 001776 001777 327
00002200 000000 000000 001001 001000 000
00002201 000000 000000 001002 001001 000
00002202 000000 000001 001003 001002 000
00002203 000000 000002 001004 001003 000
00002204 000000 000003 001005 001004 000
00002205 000000 000004 001006 001005 000
00002206 000000 000005 001007 001006 000
00002207 000000 000006 001010 001007 000
00002210 000000 000007 001011 001010 000
00002211 000000 000010 001012 001011 000
00002212 000000 000011 001013 001012 000
00002213 000000 000012 001014 001013 000
00002214 000000 000013 001015 001014 000
00002215 000000 000014 001016 001015 000
00002216 000000 000015 001017 001016 000
00002217 000000 000016 001020 001017 000
00002220 000000 000017 001021 001020 000
00002221 000000 000020 001022 001021 000
00002222 000000 000021 001023 001022 000
00002223 000000 000022 001024 001023 000
00002224 000000 000023 001025 001024 000
00002225 000000 000024 001026 001025 000
00002226 000000 000025 001027 001026 000
00002227 000000 000026 001030 001027 000
00002230 000000 000027 001031 001030 000
00002231 000000 000030 001032 001031 000
00002232 000000 000031 001033 001032 000
00002233 000000 000032 001034 001033 000
00002234 000000 000033 001035 001034 000
00002235 000000 000034 001036 001035 000
00002236 000000 000035 001037 001036 000
00002237 000000 000036 001040 001037 000
00002240 000000 000037 001041 001040 000
00002241 000000 000040 001042 001041 000
00002242 000000 000041 001043 001042 000
00002243 000000 000042 001044 001043 000
00002244 000000 000043 001045 001044 000
00002245 000000 000044 001046 001045 000
00002246 000000 000045 001047 001046 000
00002247 000000 000046 001050 001047 000
00002250 000000 000047 001051 001050 000
00002251 000000 000050 001052 001051 000
00002252 000000 000051 001053 001052 000
00002253 000000 000052 001054 001053 000
00002254 000000 000053 001055 001054 000
00002255 000000 000054 001056 001055 000
00002256 000000 000055 001057 001056 000
00002257 000000 000056 001060 001057 000
00002260 000000 000057 001061 001060 000
00002261 000000 000060 001062 001061 000
00002262 000000 000061 001063 001062 000
00002263 000000 000062 001064 001063 000
00002264 000000 000063 001065 001064 000
00002265 000000 000064 001066 001065 000
00002266 000000 000065 001067 001066 000
00002267 000000 000066 001070 001067 000
00002270 000000 000067 001071 001070 000
00002271 000000 000070 001072 001071 000
00002272 000000 000071 001073 001072 000
00002273 000000 000072 001074 001073 000
00002274 000000 000073 001075 001074 000
```

```
00002275 000000 000074 001076 001075 000
00002276 000000 000075 001077 001076 000
00002277 000000 000076 001100 001077 000
00002300 000000 000077 001101 001100 000
00002301 000000 000100 001102 001101 000
00002302 000000 000101 001103 001102 000
00002303 000000 000102 001104 001103 000
00002304 000000 000103 001105 001104 000
00002305 000000 000104 001106 001105 000
00002306 000000 000105 001107 001106 000
00002307 000000 000106 001110 001107 000
00002310 000000 000107 001111 001110 000
00002311 000000 000110 001112 001111 000
00002312 000000 000111 001113 001112 000
00002313 000000 000112 001114 001113 000
00002314 000000 000113 001115 001114 000
00002315 000000 000114 001116 001115 000
00002316 000000 000115 001117 001116 000
00002317 000000 000116 001120 001117 000
00002320 000000 000117 001121 001120 000
00002321 000000 000120 001122 001121 000
00002322 000000 000121 001123 001122 000
00002323 000000 000122 001124 001123 000
00002324 000000 000123 001125 001124 000
00002325 000000 000124 001126 001125 000
00002326 000000 000125 001127 001126 000
00002327 000000 000126 001130 001127 000
00002330 000000 000127 001131 001130 000
00002331 000000 000130 001132 001131 000
00002332 000000 000131 001133 001132 000
00002333 000000 000132 001134 001133 000
00002334 000000 000133 001135 001134 000
00002335 000000 000134 001136 001135 000
00002336 000000 000135 001137 001136 000
00002337 000000 000136 001140 001137 000
00002340 000000 000137 001141 001140 000
00002341 000000 000140 001142 001141 000
00002342 000000 000141 001143 001142 000
00002343 000000 000142 001144 001143 000
00002344 000000 000143 001145 001144 000
00002345 000000 000144 001146 001145 000
00002346 000000 000145 001147 001146 000
00002347 000000 000146 001150 001147 000
00002350 000000 000147 001151 001150 000
00002351 000000 000150 001152 001151 000
00002352 000000 000151 001153 001152 000
00002353 000000 000152 001154 001153 000
00002354 000000 000153 001155 001154 000
00002355 000000 000154 001156 001155 000
00002356 000000 000155 001157 001156 000
00002357 000000 000156 001160 001157 000
00002360 000000 000157 001161 001160 000
00002361 000000 000160 001162 001161 000
00002362 000000 000161 001163 001162 000
00002363 000000 000162 001164 001163 000
00002364 000000 000163 001165 001164 000
00002365 000000 000164 001166 001165 000
00002366 000000 000165 001167 001166 000
00002367 000000 000166 001170 001167 000
00002370 000000 000167 001171 001170 000
00002371 000000 000170 001172 001171 000
00002372 000000 000171 001173 001172 000
00002373 000000 000172 001174 001173 000
```

```
00002374 000000 000173 001175 001174 000
00002375 000000 000174 001176 001175 000
00002376 000000 000175 001177 001176 000
00002377 000000 000176 001000 001177 000
00002400 000000 000177 001001 001000 000
00002401 000000 000200 001001 001000 000
00002402 000000 000200 001001 001000 000
00002403 000000 000200 001201 001200 000
00002404 000000 000200 001202 001201 000
00002405 000000 000201 001203 001202 000
00002406 000000 000202 001204 001203 000
00002407 000000 000203 001205 001204 000
00002410 000000 000204 001206 001205 000
00002411 000000 000205 001207 001206 000
00002412 000000 000206 001210 001207 000
00002413 000000 000207 001211 001210 000
00002414 000000 000210 001212 001211 000
00002415 000000 000211 001213 001212 000
00002416 000000 000212 001214 001213 000
00002417 000000 000213 001215 001214 000
00002420 000000 000214 001216 001215 000
00002421 000000 000215 001217 001216 000
00002422 000000 000216 001220 001217 000
00002423 000000 000217 001221 001220 000
00002424 000000 000220 001222 001221 000
00002425 000000 000221 001223 001222 000
00002426 000000 000222 001224 001223 000
00002427 000000 000223 001225 001224 000
00002430 000000 000224 001226 001225 000
00002431 000000 000225 001227 001226 000
00002432 000000 000226 001230 001227 000
00002433 000000 000227 001231 001230 000
00002434 000000 000230 001232 001231 000
00002435 000000 000231 001233 001232 000
00002436 000000 000232 001234 001233 000
00002437 000000 000233 001235 001234 000
00002440 000000 000234 001236 001235 000
00002441 000000 000235 001237 001236 000
00002442 000000 000236 001240 001237 000
00002443 000000 000237 001241 001240 000
00002444 000000 000240 001242 001241 000
00002445 000000 000241 001243 001242 000
00002446 000000 000242 001244 001243 000
00002447 000000 000243 001245 001244 000
00002450 000000 000244 001246 001245 000
00002451 000000 000245 001247 001246 000
00002452 000000 000246 001250 001247 000
00002453 000000 000247 001251 001250 000
00002454 000000 000250 001252 001251 000
00002455 000000 000251 001253 001252 000
00002456 000000 000252 001254 001253 000
00002457 000000 000253 001255 001254 000
00002460 000000 000254 001256 001255 000
00002461 000000 000255 001257 001256 000
00002462 000000 000256 001260 001257 000
00002463 000000 000257 001261 001260 000
00002464 000000 000260 001262 001261 000
00002465 000000 000261 001263 001262 000
00002466 000000 000262 001264 001263 000
00002467 000000 000263 001265 001264 000
00002470 000000 000264 001266 001265 000
00002471 000000 000265 001267 001266 000
00002472 000000 000266 001270 001267 000
```

00002473 000000 000267 001271 001270 000
00002474 000000 000270 001272 001271 000
00002475 000000 000271 001273 001272 000
00002476 000000 000272 001274 001273 000
00002477 000000 000273 001275 001274 000
00002500 000000 000274 001276 001275 000
00002501 000000 000275 001277 001276 000
00002502 000000 000276 001300 001277 000
00002503 000000 000277 001301 001300 000
00002504 000000 000300 001302 001301 000
00002505 000000 000301 001303 001302 000
00002506 000000 000302 001304 001303 000
00002507 000000 000303 001305 001304 000
00002510 000000 000304 001306 001305 000
00002511 000000 000305 001307 001306 000
00002512 000000 000306 001310 001307 000
00002513 000000 000307 001311 001310 000
00002514 000000 000310 001312 001311 000
00002515 000000 000311 001313 001312 000
00002516 000000 000312 001314 001313 000
00002517 000000 000313 001315 001314 000
00002520 000000 000314 001316 001315 000
00002521 000000 000315 001317 001316 000
00002522 000000 000316 001320 001317 000
00002523 000000 000317 001321 001320 000
00002524 000000 000320 001322 001321 000
00002525 000000 000321 001323 001322 000
00002526 000000 000322 001324 001323 000
00002527 000000 000323 001325 001324 000
00002530 000000 000324 001326 001325 000
00002531 000000 000325 001327 001326 000
00002532 000000 000326 001330 001327 000
00002533 000000 000327 001331 001330 000
00002534 000000 000330 001332 001331 000
00002535 000000 000331 001333 001332 000
00002536 000000 000332 001334 001333 000
00002537 000000 000333 001335 001334 000
00002540 000000 000334 001336 001335 000
00002541 000000 000335 001337 001336 000
00002542 000000 000336 001340 001337 000
00002543 000000 000337 001341 001340 000
00002544 000000 000340 001342 001341 000
00002545 000000 000341 001343 001342 000
00002546 000000 000342 001344 001343 000
00002547 000000 000343 001345 001344 000
00002550 000000 000344 001346 001345 000
00002551 000000 000345 001347 001346 000
00002552 000000 000346 001350 001347 000
00002553 000000 000347 001351 001350 000
00002554 000000 000350 001352 001351 000
00002555 000000 000351 001353 001352 000
00002556 000000 000352 001354 001353 000
00002557 000000 000353 001355 001354 000
00002560 000000 000354 001356 001355 000
00002561 000000 000355 001357 001356 000
00002562 000000 000356 001360 001357 000
00002563 000000 000357 001361 001360 000
00002564 000000 000360 001362 001361 000
00002565 000000 000361 001363 001362 000
00002566 000000 000362 001364 001363 000
00002567 000000 000363 001365 001364 000
00002570 000000 000364 001366 001365 000
00002571 000000 000365 001367 001366 000

```
00002572 000000 000366 001370 001367 000
00002573 000000 000367 001371 001370 000
00002574 000000 000370 001372 001371 000
00002575 000000 000371 001373 001372 000
00002576 000000 000372 001374 001373 000
00002577 000000 000373 001375 001374 000
00002600 000000 000374 001376 001375 000
00002601 000000 000375 001377 001376 000
00002602 000000 000376 001200 001377 000
00002603 000000 000377 001201 001200 000
00002604 000000 000400 001201 001200 000
00002605 000000 000400 001201 001200 000
00002606 000000 000400 001401 001400 000
00002607 000000 000400 001402 001401 000
00002610 000000 000401 001403 001402 000
00002611 000000 000402 001404 001403 000
00002612 000000 000403 001405 001404 000
00002613 000000 000404 001406 001405 000
00002614 000000 000405 001407 001406 000
00002615 000000 000406 001410 001407 000
00002616 000000 000407 001411 001410 000
00002617 000000 000410 001412 001411 000
00002620 000000 000411 001413 001412 000
00002621 000000 000412 001414 001413 000
00002622 000000 000413 001415 001414 000
00002623 000000 000414 001416 001415 000
00002624 000000 000415 001417 001416 000
00002625 000000 000416 001420 001417 000
00002626 000000 000417 001421 001420 000
00002627 000000 000420 001422 001421 000
00002630 000000 000421 001423 001422 000
00002631 000000 000422 001424 001423 000
00002632 000000 000423 001425 001424 000
00002633 000000 000424 001426 001425 000
00002634 000000 000425 001427 001426 000
00002635 000000 000426 001430 001427 000
00002636 000000 000427 001431 001430 000
00002637 000000 000430 001432 001431 000
00002640 000000 000431 001433 001432 000
00002641 000000 000432 001434 001433 000
00002642 000000 000433 001435 001434 000
00002643 000000 000434 001436 001435 000
00002644 000000 000435 001437 001436 000
00002645 000000 000436 001440 001437 000
00002646 000000 000437 001441 001440 000
00002647 000000 000440 001442 001441 000
00002650 000000 000441 001443 001442 000
00002651 000000 000442 001444 001443 000
00002652 000000 000443 001445 001444 000
00002653 000000 000444 001446 001445 000
00002654 000000 000445 001447 001446 000
00002655 000000 000446 001450 001447 000
00002656 000000 000447 001451 001450 000
00002657 000000 000450 001452 001451 000
00002660 000000 000451 001453 001452 000
00002661 000000 000452 001454 001453 000
00002662 000000 000453 001455 001454 000
00002663 000000 000454 001456 001455 000
00002664 000000 000455 001457 001456 000
00002665 000000 000456 001460 001457 000
00002666 000000 000457 001461 001460 000
00002667 000000 000460 001462 001461 000
00002670 000000 000461 001463 001462 000
```

```
00002671 000000 000462 001464 001463 000
00002672 000000 000463 001465 001464 000
00002673 000000 000464 001466 001465 000
00002674 000000 000465 001467 001466 000
00002675 000000 000466 001470 001467 000
00002676 000000 000467 001471 001470 000
00002677 000000 000470 001472 001471 000
00002700 000000 000471 001473 001472 000
00002701 000000 000472 001474 001473 000
00002702 000000 000473 001475 001474 000
00002703 000000 000474 001476 001475 000
00002704 000000 000475 001477 001476 000
00002705 000000 000476 001500 001477 000
00002706 000000 000477 001501 001500 000
00002707 000000 000500 001502 001501 000
00002710 000000 000501 001503 001502 000
00002711 000000 000502 001504 001503 000
00002712 000000 000503 001505 001504 000
00002713 000000 000504 001506 001505 000
00002714 000000 000505 001507 001506 000
00002715 000000 000506 001510 001507 000
00002716 000000 000507 001511 001510 000
00002717 000000 000510 001512 001511 000
00002720 000000 000511 001513 001512 000
00002721 000000 000512 001514 001513 000
00002722 000000 000513 001515 001514 000
00002723 000000 000514 001516 001515 000
00002724 000000 000515 001517 001516 000
00002725 000000 000516 001520 001517 000
00002726 000000 000517 001521 001520 000
00002727 000000 000520 001522 001521 000
00002730 000000 000521 001523 001522 000
00002731 000000 000522 001524 001523 000
00002732 000000 000523 001525 001524 000
00002733 000000 000524 001526 001525 000
00002734 000000 000525 001527 001526 000
00002735 000000 000526 001530 001527 000
00002736 000000 000527 001531 001530 000
00002737 000000 000530 001532 001531 000
00002740 000000 000531 001533 001532 000
00002741 000000 000532 001534 001533 000
00002742 000000 000533 001535 001534 000
00002743 000000 000534 001536 001535 000
00002744 000000 000535 001537 001536 000
00002745 000000 000536 001540 001537 000
00002746 000000 000537 001541 001540 000
00002747 000000 000540 001542 001541 000
00002750 000000 000541 001543 001542 000
00002751 000000 000542 001544 001543 000
00002752 000000 000543 001545 001544 000
00002753 000000 000544 001546 001545 000
00002754 000000 000545 001547 001546 000
00002755 000000 000546 001550 001547 000
00002756 000000 000547 001551 001550 000
00002757 000000 000550 001552 001551 000
00002760 000000 000551 001553 001552 000
00002761 000000 000552 001554 001553 000
00002762 000000 000553 001555 001554 000
00002763 000000 000554 001556 001555 000
00002764 000000 000555 001557 001556 000
00002765 000000 000556 001560 001557 000
00002766 000000 000557 001561 001560 000
00002767 000000 000560 001562 001561 000
```

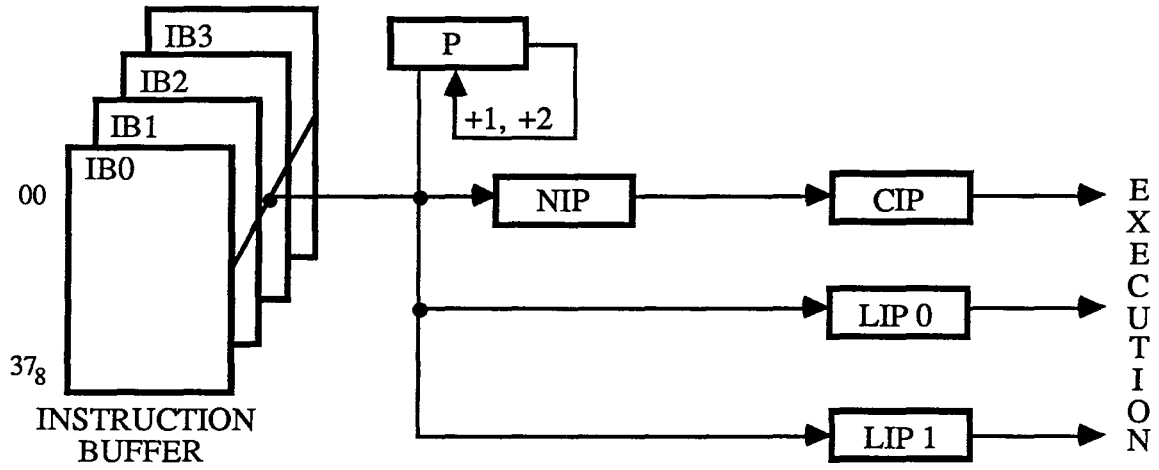
```
00002770 000000 000561 001563 001562 000
00002771 000000 000562 001564 001563 000
00002772 000000 000563 001565 001564 000
00002773 000000 000564 001566 001565 000
00002774 000000 000565 001567 001566 000
00002775 000000 000566 001570 001567 000
00002776 000000 000567 001571 001570 000
00002777 000000 000570 001572 001571 000
00003000 000000 000571 001573 001572 000
00003001 000000 000572 001574 001573 000
00003002 000000 000573 001575 001574 000
00003003 000000 000574 001576 001575 000
00003004 000000 000575 001577 001576 000
00003005 000000 000576 001400 001577 000
00003006 000000 000577 001401 001400 000
00003007 000000 000600 001401 001400 000
00003010 000000 000600 001401 001400 000
00003011 000000 000600 001601 001600 000
00003012 000000 000600 001602 001601 000
00003013 000000 000601 001603 001602 000
00003014 000000 000602 001604 001603 000
00003015 000000 000603 001605 001604 000
00003016 000000 000604 001606 001605 000
00003017 000000 000605 001607 001606 000
00003020 000000 000606 001610 001607 000
00003021 000000 000607 001611 001610 000
00003022 000000 000610 001612 001611 000
00003023 000000 000611 001613 001612 000
00003024 000000 000612 001614 001613 000
00003025 000000 000613 001615 001614 000
00003026 000000 000614 001616 001615 000
00003027 000000 000615 001617 001616 000
00003030 000000 000616 001620 001617 000
00003031 000000 000617 001621 001620 000
00003032 000000 000620 001622 001621 000
00003033 000000 000621 001623 001622 000
00003034 000000 000622 001624 001623 000
00003035 000000 000623 001625 001624 000
00003036 000000 000624 001626 001625 000
00003037 000000 000625 001627 001626 000
00003040 000000 000626 001630 001627 000
00003041 000000 000627 001631 001630 000
00003042 000000 000630 001632 001631 000
00003043 000000 000631 001633 001632 000
00003044 000000 000632 001634 001633 000
00003045 000000 000633 001635 001634 000
00003046 000000 000634 001636 001635 000
00003047 000000 000635 001637 001636 000
00003050 000000 000636 001640 001637 000
00003051 000000 000637 001641 001640 000
00003052 000000 000640 001642 001641 000
00003053 000000 000641 001643 001642 000
00003054 000000 000642 001644 001643 000
00003055 000000 000643 001645 001644 000
00003056 000000 000644 001646 001645 000
00003057 000000 000645 001647 001646 000
00003060 000000 000646 001650 001647 000
00003061 000000 000647 001651 001650 000
00003062 000000 000650 001652 001651 000
00003063 000000 000651 001653 001652 000
00003064 000000 000652 001654 001653 000
00003065 000000 000653 001655 001654 000
00003066 000000 000654 001656 001655 000
```

```
00003067 000000 000655 001657 001656 000
00003070 000000 000656 001660 001657 000
00003071 000000 000657 001661 001660 000
00003072 000000 000660 001662 001661 000
00003073 000000 000661 001663 001662 000
00003074 000000 000662 001664 001663 000
00003075 000000 000663 001665 001664 000
00003076 000000 000664 001666 001665 000
00003077 000000 000665 001667 001666 000
00003100 000000 000666 001670 001667 000
00003101 000000 000667 001671 001670 000
00003102 000000 000670 001672 001671 000
00003103 000000 000671 001673 001672 000
00003104 000000 000672 001674 001673 000
00003105 000000 000673 001675 001674 000
00003106 000000 000674 001676 001675 000
00003107 000000 000675 001677 001676 000
00003110 000000 000676 001700 001677 000
00003111 000000 000677 001701 001700 000
00003112 000000 000700 001702 001701 000
00003113 000000 000701 001703 001702 000
00003114 000000 000702 001704 001703 000
00003115 000000 000703 001705 001704 000
00003116 000000 000704 001706 001705 000
00003117 000000 000705 001707 001706 000
00003120 000000 000706 001710 001707 000
00003121 000000 000707 001711 001710 000
00003122 000000 000710 001712 001711 000
00003123 000000 000711 001713 001712 000
00003124 000000 000712 001714 001713 000
00003125 000000 000713 001715 001714 000
00003126 000000 000714 001716 001715 000
00003127 000000 000715 001717 001716 000
00003130 000000 000716 001720 001717 000
00003131 000000 000717 001721 001720 000
00003132 000000 000720 001722 001721 000
00003133 000000 000721 001723 001722 000
00003134 000000 000722 001724 001723 000
00003135 000000 000723 001725 001724 000
00003136 000000 000724 001726 001725 000
00003137 000000 000725 001727 001726 000
00003140 000000 000726 001730 001727 000
00003141 000000 000727 001731 001730 000
00003142 000000 000730 001732 001731 000
00003143 000000 000731 001733 001732 000
00003144 000000 000732 001734 001733 000
00003145 000000 000733 001735 001734 000
00003146 000000 000734 001736 001735 000
00003147 000000 000735 001737 001736 000
00003150 000000 000736 001740 001737 000
00003151 000000 000737 001741 001740 000
00003152 000000 000740 001742 001741 000
00003153 000000 000741 001743 001742 000
00003154 000000 000742 001744 001743 000
00003155 000000 000743 001745 001744 000
00003156 000000 000744 001746 001745 000
00003157 000000 000745 001747 001746 000
00003160 000000 000746 001750 001747 000
00003161 000000 000747 001751 001750 000
00003162 000000 000750 001752 001751 000
00003163 000000 000751 001753 001752 000
00003164 000000 000752 001754 001753 000
00003165 000000 000753 001755 001754 000
```

```
00003166 000000 000754 001756 001755 000
00003167 000000 000755 001757 001756 000
00003170 000000 000756 001760 001757 000
00003171 000000 000757 001761 001760 000
00003172 000000 000760 001762 001761 000
00003173 000000 000761 001763 001762 000
00003174 000000 000762 001764 001763 000
00003175 000000 000763 001765 001764 000
00003176 000000 000764 001766 001765 000
00003177 000000 000765 001767 001766 000
00003200 000000 000766 001770 001767 000
00003201 000000 000767 001771 001770 000
00003202 000000 000770 001772 001771 000
00003203 000000 000771 001773 001772 000
00003204 000000 000772 001774 001773 000
00003205 000000 000773 001775 001774 000
00003206 000000 000774 001776 001775 000
00003207 000000 000775 001777 001776 000
00003210 000000 000776 001600 001777 000
00003211 000000 000777 001601 001600 000
00003212 000000 001000 001601 001600 000
00003213 000000 001000 001601 001600 000
00003214 000000 001000 001601 001600 000
00003215 000000 001000 001601 001600 000
00003216 000000 001000 001601 001600 000
00003217 000000 001000 001601 001600 000
00003220 000000 001000 001601 001600 000
00003221 000000 001000 001601 001600 000
00003222 000000 001000 001601 001600 000
00003223 000000 001000 001601 001600 000
00003224 000000 001000 001601 001600 000
00003225 000000 001000 001601 001600 000
00003226 000000 001000 001601 001600 000
00003227 000000 001000 001601 001600 000
00003230 000000 001000 001601 001600 000
00003231 000000 001000 001601 001600 000
00003232 000000 001000 001601 001600 000
00003233 000000 001000 001601 001600 000
00003234 000000 001000 001601 001600 000
00003235 000000 001000 001601 001600 000
00003236 000000 001000 001601 001600 000
00003237 000000 001000 001601 001600 000
00003240 000000 001000 001601 001600 000
00003241 000000 001000 001601 001600 000
00003242 000000 001000 001601 001600 000
00003243 000000 001000 001601 001600 000
00003244 000000 001000 001601 001600 000
00003245 000000 001000 001601 001600 000
00003246 000000 001000 001601 001600 000
00003247 000000 001000 001601 001600 000
00003250 000000 001000 001601 001600 000
00003251 000000 001000 001601 001600 000
00003252 000000 001000 001601 001600 000
00003253 000000 001000 001601 001600 000
00003254 000000 001000 001601 001600 000
00003255 000000 001000 001601 001600 000
00003256 000000 001000 001601 001600 000
00003257 000000 001000 001601 001600 000
00003260 000000 001000 001601 001600 000
00003261 000000 001000 001601 001600 000
00003262 000000 001000 001601 001600 000
00003263 000000 001000 001601 001600 000
00003264 000000 001000 001601 001600 000
```

INSTRUCTION ISSUE

Instruction issue is the process by which the CRAY Y-MP introduces instructions into the Central Processor Unit (CPU) for execution.



During the instruction issue operation, the P register selects an instruction parcel from one of four instruction buffers. The P register increments by (+1, +2), depending on a 2 or 3 parcel instruction. A plus two is used for 3 parcel instructions when in 32-bit mode, while a plus one is used for 1 and 2 parcel instructions when not in 32-bit mode. The CRAY Y-MP loads the selected instruction parcels into the NIP register. While the instruction parcel is in the NIP register, a pre-decode is done to determine if the instruction is a 1, 2, or 3 parcel instruction.

If the instruction in NIP is a 1 parcel instruction, the instruction parcel is advanced to CIP, and the P register is incremented by a plus one. If the instruction in NIP is decoded as a 2 parcel instruction, the parcel will move into CIP upon getting the Go Issue. The next parcel of the 2 parcel instruction will be loaded into NIP/LIP. When issuing the 2 parcel instruction, both CIP and LIP are unloaded. The next clock period that LIP would normally go into CIP, the instruction is prevented from being decoded by a Blank NIP. All during this operation the P register has been incrementing by one as long as there are no Hold Issue conditions, where the P register would be held. The 2 parcel instructions include the (005 - 007) instruction and the branch instructions, which are the (01X) instructions. The (01H) is not used in Y-mode.

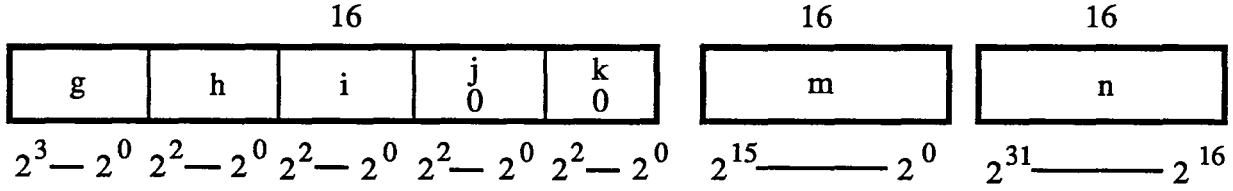
During a 3 parcel instruction, the 3 parcel instruction is pre-decoded in the NIP register. When the first parcel of the 3 parcel instruction advances to CIP, the second and third parcels are loaded into the LIP and LIP 1 registers at the same time. When the 3 parcel instruction issues all three registers, CIP, LIP, and LIP 1 will unload their instruction parcels into execution. This unique operation allows a 3 parcel instruction to issue in 2 clock periods. The 3 parcel instruction can

only be used in 32-bit mode, where the (020, 021, 040, 041, 10X - 13X) instructions are converted into 3 parcel instructions.

When the second parcel of a 2 or 3 parcel instruction moves from the LIP 0 register to execution, Blank NIP is loaded into CIP. The CPU decodes the Blank NIP as a no operation instruction when it issues from the CIP register. The P register is then incremented by plus two for a 3 parcel instruction to address the next instruction within the IB. When Blank NIP is loaded into CIP, a new parcel is being loaded into NIP and being decoded for a 1, 2, or 3 parcel instruction.

THREE PARCEL INSTRUCTION FORMAT

The format of the 3 parcel instruction while in machine code is the first parcel containing the instruction decode, which are the (ghijk) fields of the instruction. However, the (jk) field is not used. The second parcel of a 3 parcel instruction is the (m) field, which contains 16 bits. The 16 bits are numbered (2^{15} - 2^{0}). The third parcel of a 3 parcel instruction contains the (n) field, which also contains 16 bits numbered (2^{31} - 2^{16}). The example below shows the format.



The reason the (m) field is before the (n) field is so that when not in 32-bit mode, the 2 parcel instruction can issue using the (jk) as part of the expression, and the third parcel would be another instruction. However, after the instruction issues, the (mn) field is reversed into its proper bit position (nm). The 3 parcel instruction includes the (020 - 021) instruction, the (040 - 041) instruction, and the (10X - 13X) instructions.

Hardware Trng.
YM10/02A J.E.S.

BRANCH CONTROL

The branch control portion of the CRAY Y-MP decodes and controls the issue of all branch instructions. All branch instructions, whether in Y-mode or X-mode, are decoded as a 2 parcel instruction. This is because the P register can only increment up to four million words, and a 2 parcel branch instruction would reach four million words. A branch instruction, if executed, loads the P register with a new address, depending on the branch instruction. As a result, the program branches to the instruction at the new P register address.

The branch control logic decodes all branch instructions at NIP time. The branch is decoded at NIP time so that the path into the NIP register can be disabled for the next instruction parcel. It is important that no more instructions enter the NIP register from the instruction buffer until it is determined if the branch will be taken. This is because the instructions cannot be backed up once they enter NIP, except for the Text and Set instructions.

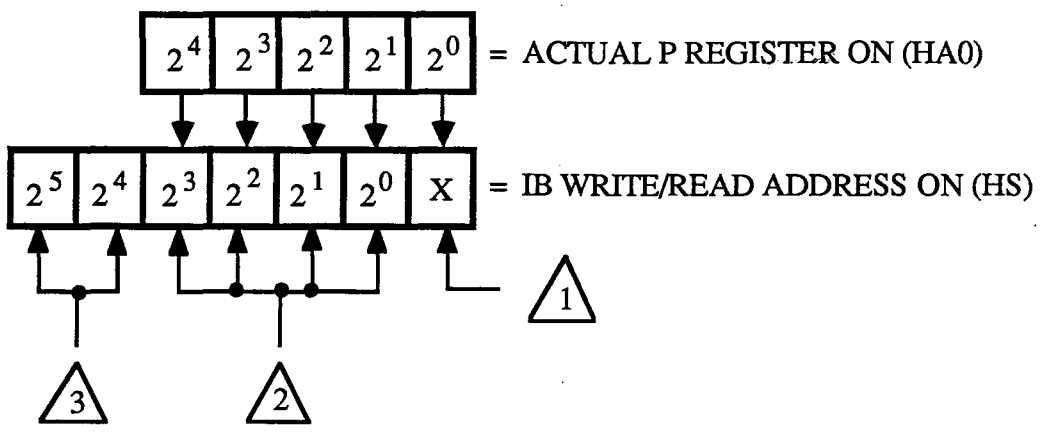
The instruction issue logic treats all branch instructions as a 2 parcel instruction. Therefore, it effectively takes 2 clock periods to issue a branch instruction.

ISSUE CONTROL BLOCK DIAGRAM

Before an instruction can issue, it first must be read out of the Instruction buffer. The Instruction buffer is loaded by a Fetch operation. The Fetch operation uses Port D and requests one word at a time from memory, competing with the other CPU for its memory words. Within the CPU, Port D has the highest priority when used for a fetch operation. As the words arrive from memory to the Instruction buffers, the (HA0) option controls the writing and reading of the Instruction buffers. The Instruction buffers are divided into even and odd words of the four Instruction buffers 0 - 3. Each Instruction buffer will hold 32_{10} CRAY words, or 128 instruction parcels. The Instruction buffers are contained on the (HS) options. The Go Bypass is not used on the (HS) option.

The (HA0) option controls the reading and writing of the Instruction buffers. The (HA) contains an Instruction buffer (IB) Write address, and a IB Read address. The IB Write address and Read address contain 6 bits (2^5 - 2^0). The IB address breakdown is as shown on the next page.

Hardware Trng.
YM10/03 J.E.S.



1 (X) - The address bit (2^0) will select the even or odd address words within the IB. (HS0 - HS3) contain all the even words, and (HS4 - HS7) contain all the odd words. Address bit (2^0) = 0 selects IB0, while (2^0) = 1 selects IB1.

2 (2^3 - 2^0) - Leaves the (HA) option as (R18 - R21) for IB0 or (R24 - R27) for IB1 as address bits (2^0 - 2^3), and selects one of sixteen addresses within an Instruction buffer.

3 (2^5 - 2^4) - Leaves the (HA) option as (R22, R23) for IB0, and (R28, R29) for IB1 as address bits (2^4, 2^5). Address bits (2^4 - 2^5) are the Instruction buffer pointers. The (HS) option (2^4) is used as part of the internal storage address, and (2^5) is used to select one of thirty two address words.

(2^4, 2^5) = 0	IB0
(2^4, 2^5) = 1	IB1
(2^4, 2^5) = 2	IB2
(2^4, 2^5) = 3	IB3

The IB Read address is active as long as the Write address is greater than the Read address, and the Read address is greater than or equal to the Fetch Starting address, with no end around. Or, the Read address can be active as long as it is greater than or equal to the Fetch Starting address with an end around occuring, or the Read address is active when it is less than the Write address with an end around occuring.

Hardware Trng.
YM10/04 J.E.S.

Besides the (HA) option controlling the reading and writing of the IB, it also contains P register bits (2^4 - 2^0, 2^{-1}, 2^{-2}). P register bits (2^0, 2^{-1}, 2^{-2}) are sent to the (HC, HD) option to select one of 8 parcels, which were read out of the (HS) option to be gated into the NIP/LIP register. P register bits (2^4 - 2^0) are loaded into the IB Read address and incremented as each word is being read out of the IB. P register bits (2^4 - 2^0) are also loaded into the IB Write address during a Fetch operation. The IB Write address is then incremented each time a word is read from memory, which is caused by the Go Write Fetch from the (YZ0) option. The P register is (2^{21} - 2^{-2}) incremented by a plus one or a plus two each time a Go Issue is present. A P + 2 is done on a 3 parcel instruction, while a P + 1 is done on a 1 or 2 parcel instruction. The P register can be loaded or changed in four different ways:

- Exchange process while muxing in a new P register value
- Branch instruction, an unconditional branch instruction, or a conditional branch instruction which the conditions were made
- Return jump instruction - on a 005ijk instruction when (Bjk) is loaded in the P register
- P + 1, P + 2 - Go Issue causing an increment of the P register, and changing the value of the P register

The (HA) option also generates Blank NIP when the IB data is invalid, or during a Branch instruction when the P register has to be reloaded. The (HA) will also check for a 2 or 3 parcel split caused by the second or third parcel being in another IB, or not in an IB and having to be fetched.

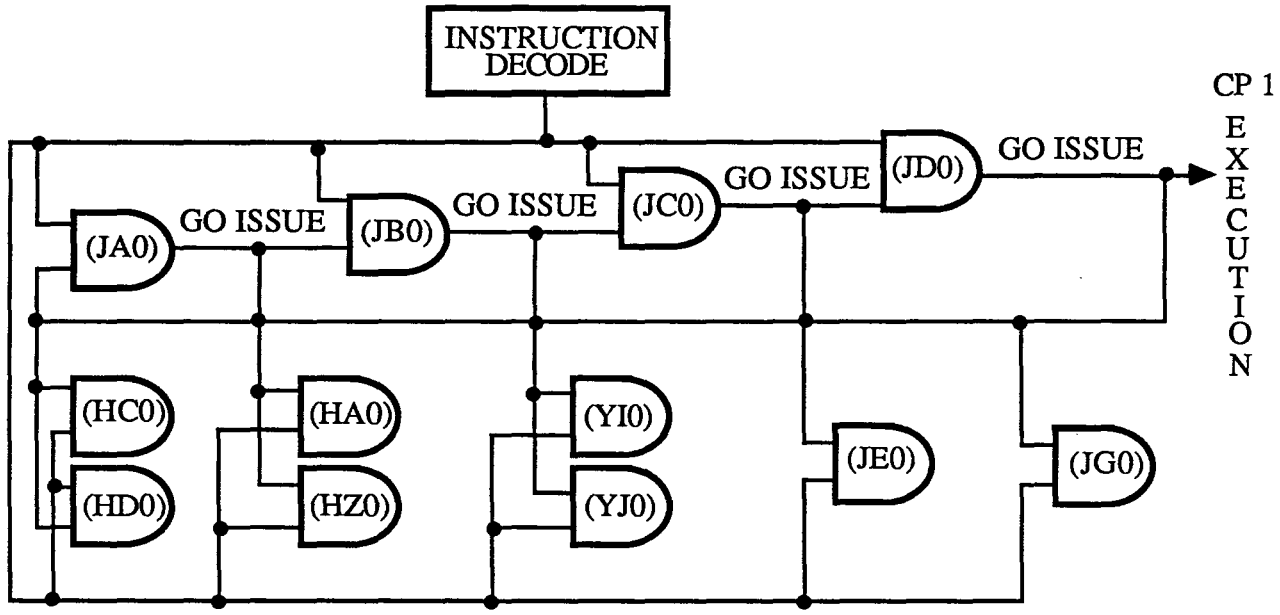
The Control Dump (CD) arriving from the (YZ0) and sent to the (HA) option will be passed to the (HZ0), causing the (HZ0) to invalidate the IB. This forces a Fetch operation on the (YZ0) option. The (HA0), upon getting the CD, will fill four buffers from memory. The (HA) will then send a Go Read IB even or a Go Read IB odd to the (YZ0) to force a read of all Instruction buffers (IB0 - IB3) through the issue circuits back to memory, along with the P register on the (YZ, YH) options.

The (HZ0) option contains the upper P register bit (2^{21} - 2^5), along with the IBAR 0 - 3 and the buffer valid bits. The (HZ0) will compare the IBAR against the P register to determine if the P register has coincidence with the Instruction buffers. If not, a Fetch operation is forced on the (YZ0) option. The (HZ0) also contains the buffer counter, which is forced to zero on an Exchange sequence.

The (HC, HD) options contain the NIP/LIP or LIP 1 registers. The NIP registers are loaded from the IB or (HS) option. If available, the (HC, HD) option will have 8 parcels of data to select into the NIP/LIP registers. Which parcels are chosen is determined by the decode of P register bits (2^0, 2^{-1}, 2^{-2}). The (HC) will also perform the pre-decode for a 2 or 3 parcel instruction. The (HC, HD) option also contains the LIP 1 register, which is loaded for a 3 parcel instruction when in 32-bit mode.

ISSUE CONTROL

The Issue control involves the moving of instruction parcels from NIP/LIP to CIP, and then into execution. The Go Issue control term controls the flow of instruction from NIP/LIP to CIP into execution. The Go Issue control term also involves chaining together the various (J-, H-) options such that as they decode the various instructions, any one of of the (J-) options needing to Hold Issue on that particular instruction will hold up the Go Issue, and all the (J-, H-) options will be prevented from moving further. Basically what we are looking at is a series of And gates, such that if any one of the gates are not made, the Go Issue would not be made.

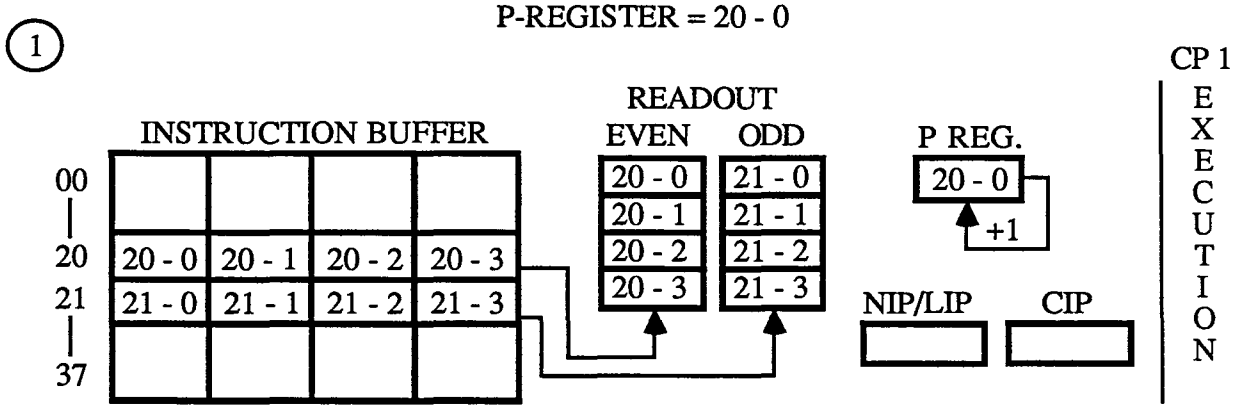


The (JA, JB, JC, JD) Go Issues are on the sharing end of the Go Issue circuit, while the (HA, HB, YI, YJ, JE, JG) Go Issues are on the receiving end. These options only need to be told whether to advance or to continue with what they were doing.

The diagrams on the following pages outline the execution of the 2 or 3 parcel instruction, which summarizes Issue control.

Hardware Tmg.
YM10/06A J.E.S.

TWO PARCEL INSTRUCTION ISSUE

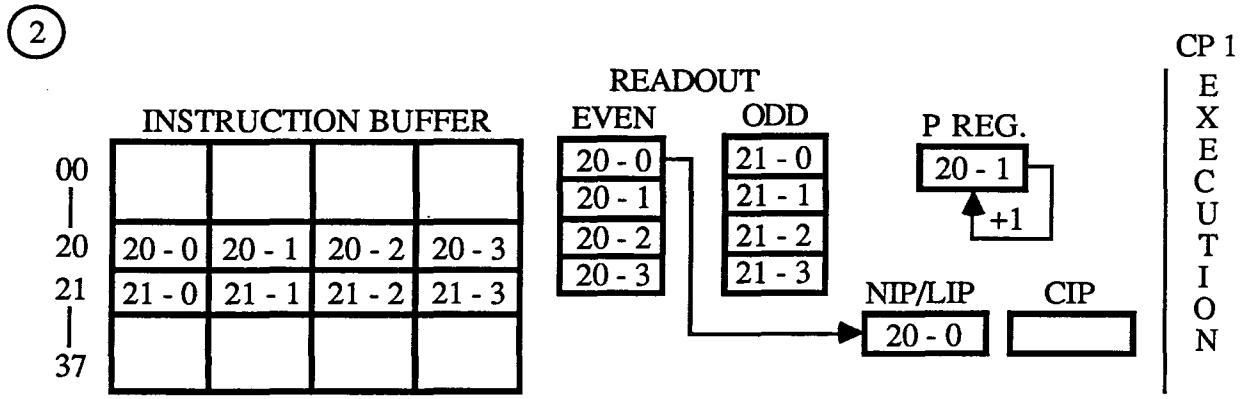


A. Fetch operation loads the instruction buffers and Readout registers as words from memory become available on the (HS) option.

B. The P register is pointing to the next parcel to be read out of the Readout register.

C. The Readout register is interweaved into the Even and Odd addresses.

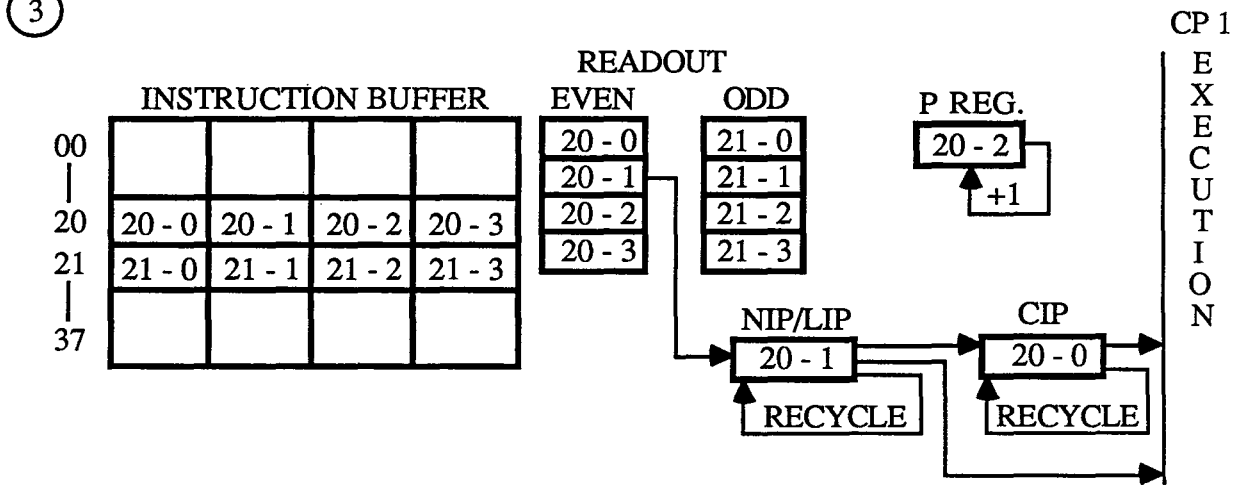
D. Because 8 parcels are present on the Readout registers, the passed pointer is eliminated. P register bits (2^0, 2^{-1}, 2^{-2}) are used to select the correct parcel while in NIP/LIP.



Hardware Trng.
YM10/07 J.E.S.

A. Increment P register when the instruction 20 - 0 enters NIP.

B. Pre-decode at NIP for a 2 parcel instruction while 20 - 0 is in NIP.



A. Pre-decode loads 20 - 1 into LIP register, and 20 - 0 into CIP when 20 - 0 was decoded as a 2 parcel instruction in NIP/LIP.

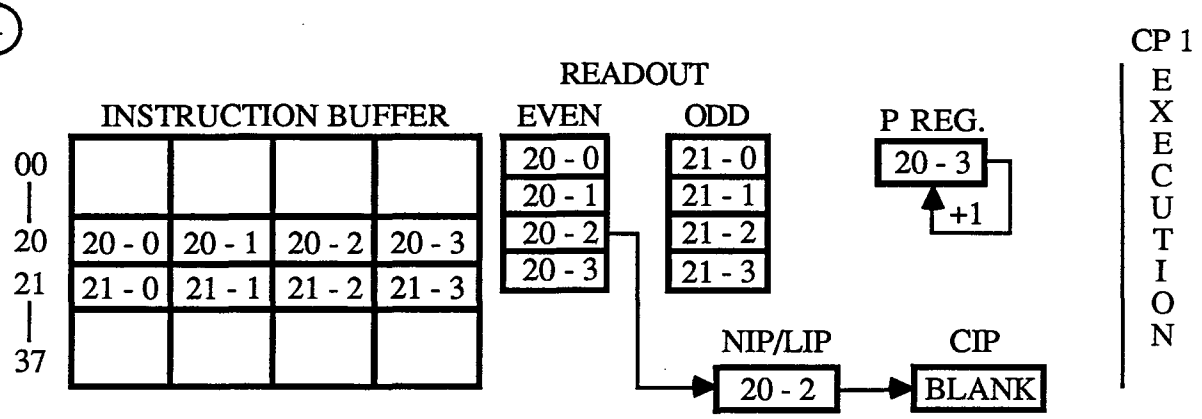
B. Increment P register by plus one to 20 - 2.

C. Check for conflicts or reservation. If there are conflicts, recycle CIP and LIP.

D. If there are no conflicts, issue in the next clock period.

Hardware Trng.
YM10/08 J.E.S.

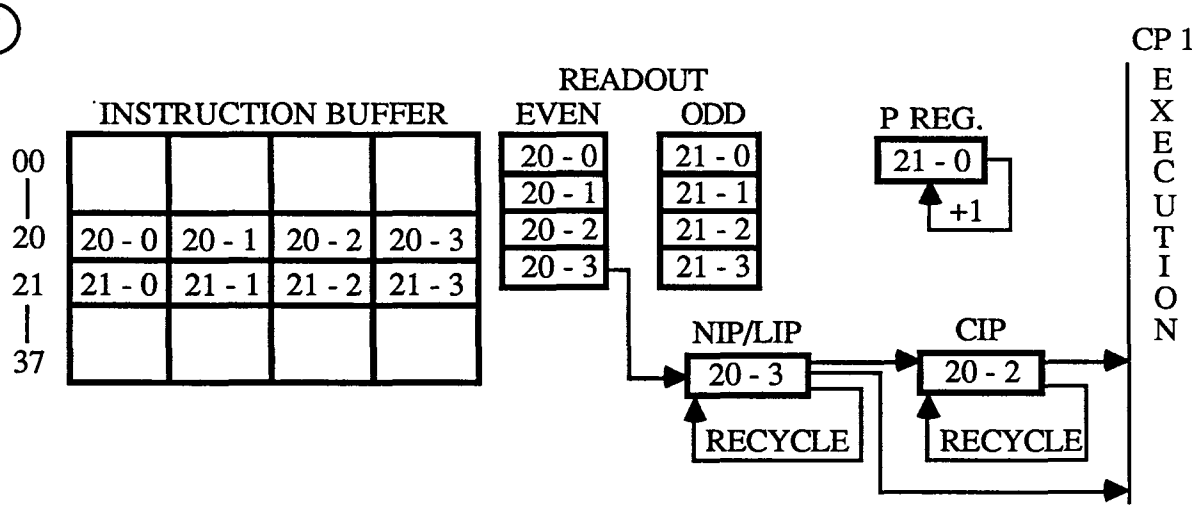
(4)



A. When 20 - 2 enters NIP/LIP, Blank is generated in CIP to prevent 20 - 1 from being decoded as an instruction.

B. Load 20 - 2 into NIP and increment the P register to 20 - 3. Do a pre-decode on 20 - 2 for a 2 parcel instruction.

(5)



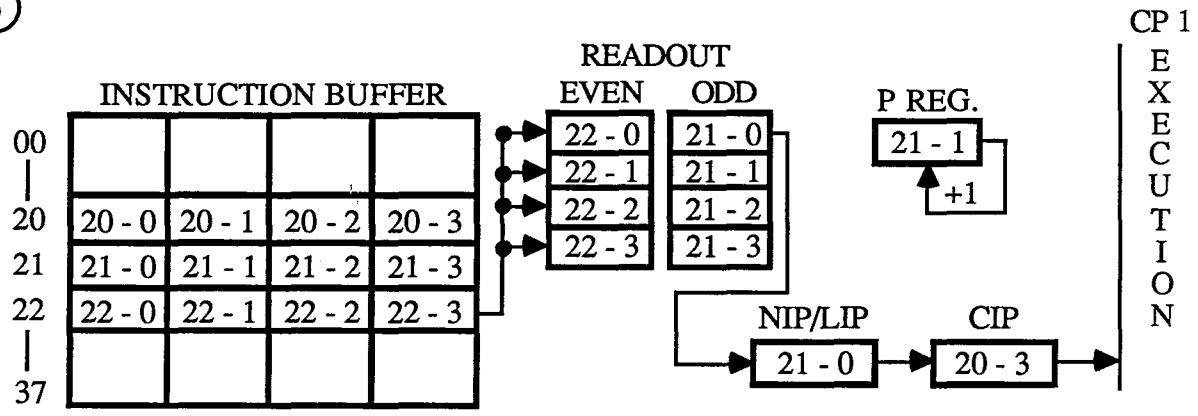
A. If 20 - 2 was a 2 parcel instruction, 20 - 2 and 20 - 3 would issue together.

Hardware Trng.
YM10/09 J.E.S.

B. If 20 - 2 was at 1 parcel instruction, only 20 - 2 would issue and 20 - 3 would move into CIP.

C. Check for conflicts on 20 - 2 while in CIP. If there are conflicts, recycle CIP and NIP/LIP.

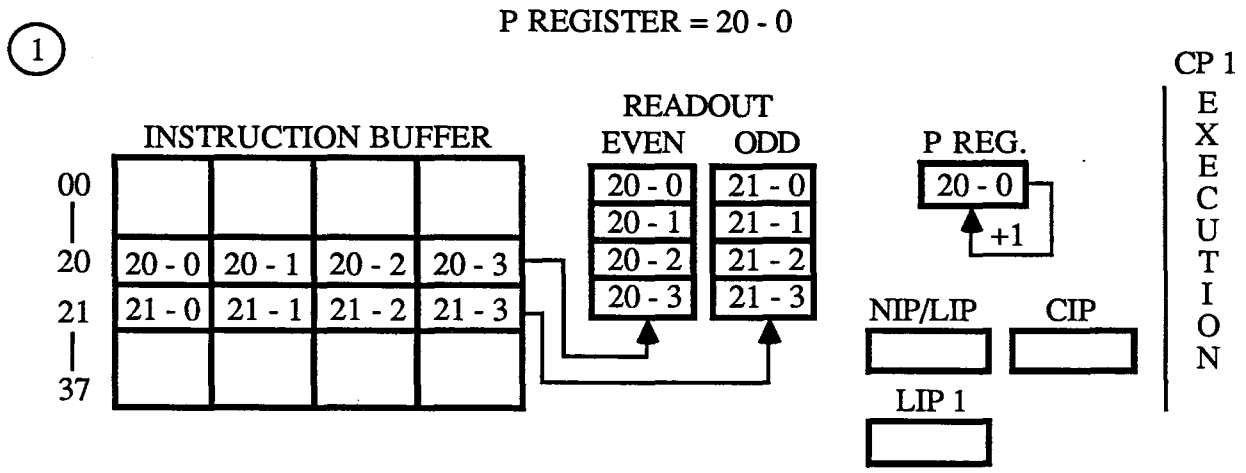
(6)



A. Readout even is loaded with the next word (22 - 0 through 22 - 3) caused by an Enable Read of IB1 on the (HA) option. This will advance the IB Read address as long as there is no Go Branch at the Buffer Boundary, and there is coincidence on IB1 and IB0.

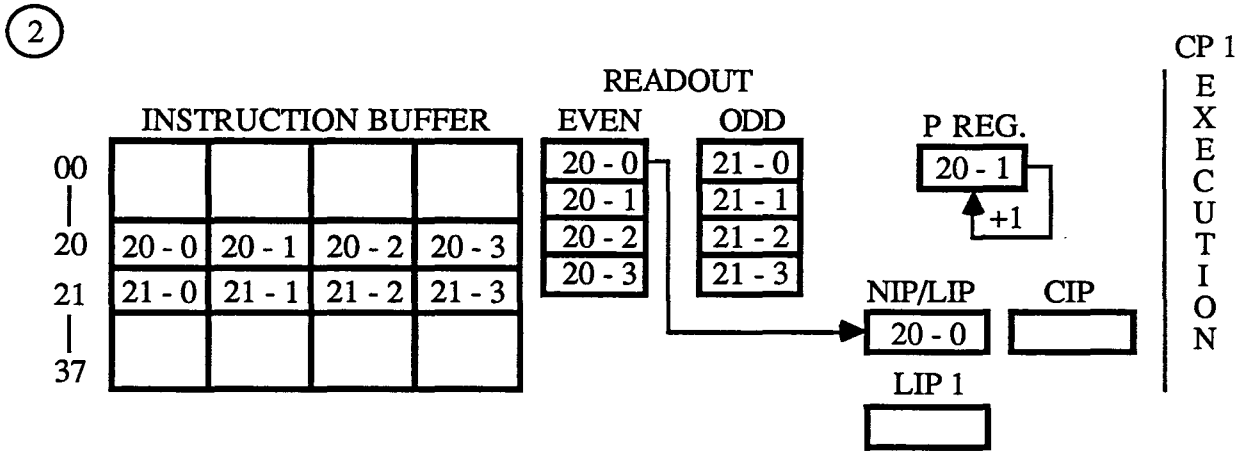
B. Return to step 2.

THREE PARCEL INSTRUCTION ISSUE



A. The Fetch operation loads the instruction buffers and Readout registers as words from memory become available.

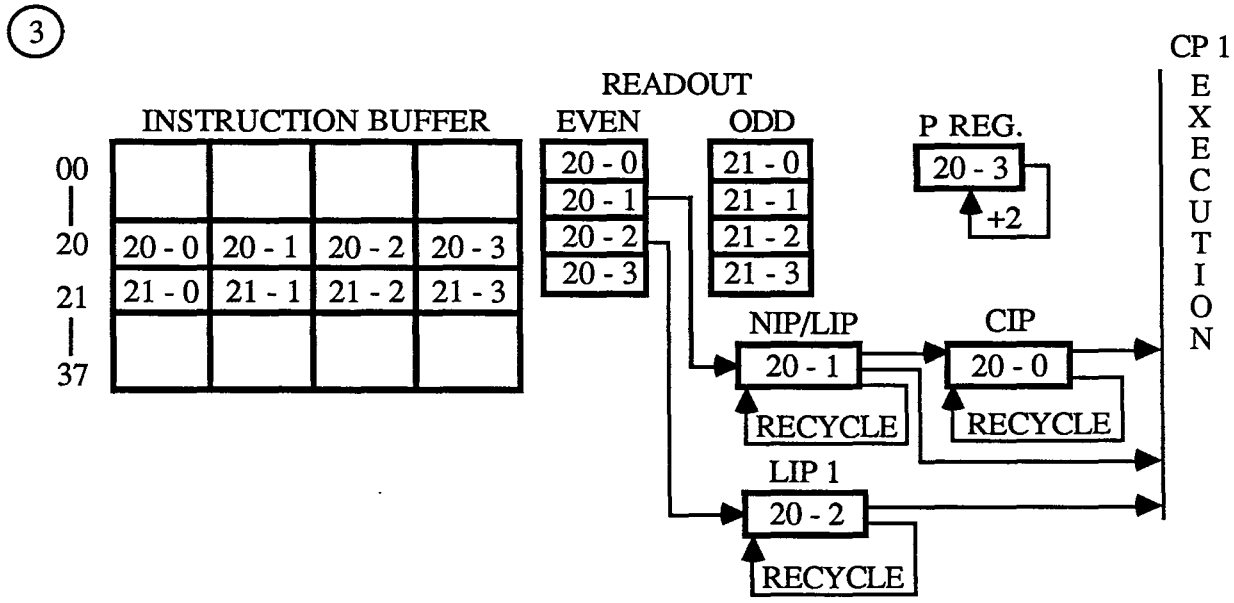
B. The P register is pointing to the next parcel to be read out of the Readout register.



Hardware Trng.
YM10/11 J.E.S.

A. Increment P register when instruction 20 - 0 enters NIP/LIP.

B. Pre-decode 20 - 0 for a 2 or 3 parcel instruction when in 32-bit mode.



A. Go Issue advances 20 - 0 into CIP.

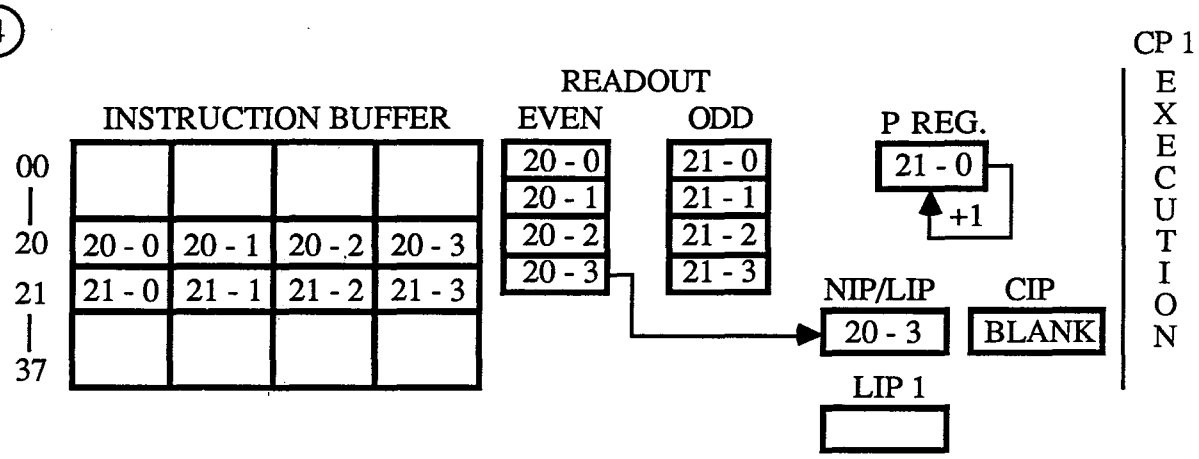
B. Pre-decode loads 20 - 1 into LIP and 20 - 2 into LIP 1 when a 20 - 0 was decoded as a 3 parcel instruction in NIP/LIP.

C. P register is incremented by plus two to point to the new instruction at 20 - 3.

D. Check for conflicts or reservations. If there are conflicts, recycle CIP, LIP, and LIP 1.

E. If there are no conflicts, issue the 3 parcel instruction from CIP, LIP, and LIP 1.

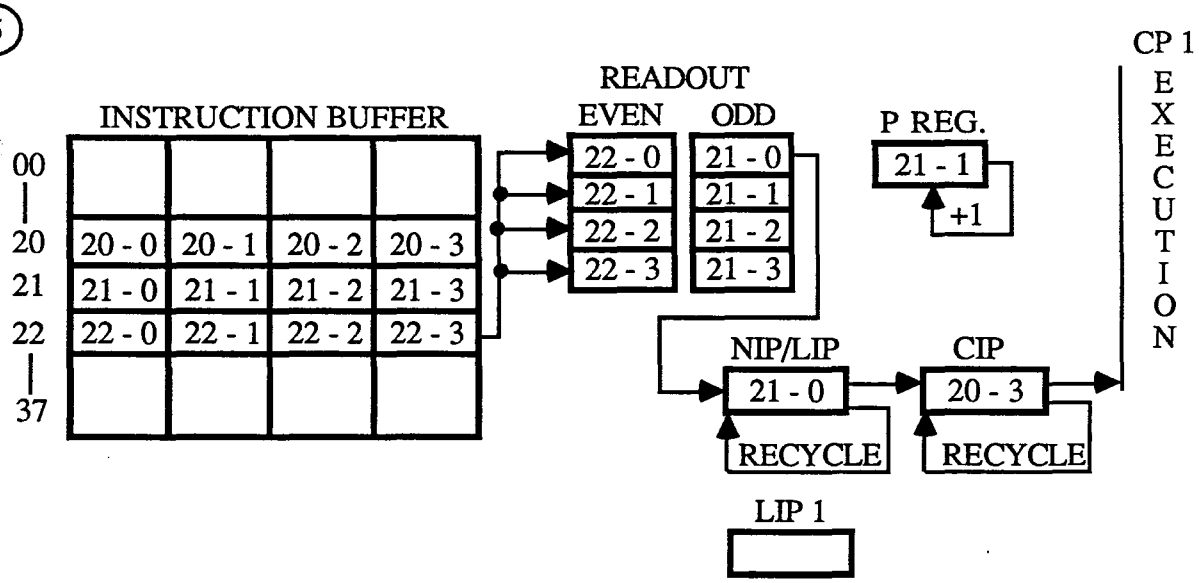
(4)



A. When 20 - 3 enters NIP/LIP, Blank is generated in CIP to prevent 20 - 1 from being decoded as an instruction.

B. Load 20 - 3 into NIP and increment the P register to 21 - 0. Do a pre-decode on 20 - 3 for a 2 or 3 parcel instruction.

(5)



Hardware Trng.
YM10/13 J.E.S.

A. If 20 - 3 was a 2 parcel instruction, 20 - 3 and 21 - 0 would issue together if there were no conflicts.

B. If 20 - 3 was a 1 parcel instruction, 20 - 3 would issue and 21 - 0 would move into CIP if there were no conflicts.

C. Advance the Instruction Read address and load the Even Readout register with (22 - 0 thru 22 - 3).

D. Return to step 2.

CRAY X-MP
INSTRUCTION ISSUE X-MODE

ONE PARCEL

030ijk	Integer sum of (Aj) and (Ak) to Ai
022ijk	Transmit the constant formed by jk to Ai

TWO PARCEL

0050jk	Jump to (Bjk)
006ijkm	Jump to ijkm
007ijkm	Return jump to ijkm; and load B00 with P register
01hijkm	Transmit the constant formed by the ijkm 24 bits to Ah (Special case i 2^2 must equal to a one)
010ijkm	Branch to ijkm if (A0) equals zero (Special case i 2^2 must equal to a zero)
011ijkm	Branch to ijkm if (A0) not equals zero (Special case i 2^2 must equal to a zero)
012ijkm	Branch to ijkm if (A0) is positive (Special case i 2^2 must equal to a zero)
013ijkm	Branch to ijkm if (A0) is negative (Special case i 2^2 must equal to a zero)
014ijkm	Branch to ijkm if (S0) equals zero (Special case i 2^2 must equal to a zero)
015ijkm	Branch to ijkm if (S0) not equals zero (Special case i 2^2 must equal to a zero)
016ijkm	Branch to ijkm if (S0) is positive (Special case i 2^2 must equal to a zero)
017ijkm	Branch to ijkmif (S0) is negative (Special case i 2^2 must equal to a zero)
020ijkm	Transmit jkm to Ai
021ijkm	Transmit ones complement of jkm to Ai
040ijkm	Transmit jkm to Si
041ijkm	Transmit ones complement of jkm to Si
10hijkm	Read from Memory address [jkm + (Ah) + DBA] to Ai
11hijkm	Write (Ai) to Memory address [jkm + (Ah) + DBA]
12hijkm	Read from Memory address [jkm + (Ah) + DBA] to Si
13hijkm	Write (Si) to Memory address [jkm + (Ah) + DBA]
056ijk	Shift (Si and Sj) left (Ak) places to Si
057ijk	Shift (Sj and Si) right (Ak) places to Si
176i0k	Read VL words to Vi from Memory address [(Ao) + (DBA)] incremented by (Ak)
1770j0	Write VL words from Vj to Memory address to [(Ao) + (DBA)] incremented by (Ak)

Hardware Trng.
YM10/18A J.E.S.

CRAY Y-MP
INSTRUCTION ISSUE Y-MODE

ONE PARCEL

030ijk	Integer sum of (Aj) and (Ak) to Ai
022ijk	Transmit the constant formed by jk to Ai

TWO PARCEL

0050jk	Jump to (Bjk)
006ijkm	Jump to ijkm
007ijkm	Return jump to ijkm; and load B00 with P register
010ijkm	Branch to ijkm if (A0) equals zero
011ijkm	Branch to ijkm if (A0) not equals zero
012ijkm	Branch to ijkm if (A0) is positive
013ijkm	Branch to ijkm if (A0) is negative
014ijkm	Branch to ijkm if (S0) equals zero
015ijkm	Branch to ijkm if (S0) not equals zero
016ijkm	Branch to ijkm if (S0) is positive
017ijkm	Branch to ijkm if (S0) is negative
056ijk	Shift (Si and Sj) left (Ak) places to Si
057ijk	Shift (Sj and Si) right (Ak) places to Si
176i0k	Read VL words to Vi from Memory address [(Ao) + (DBA)] incremented by Ak
1770j0	Write VL words from Vj to Memory address [(Ao) + (DBA)] incremented by Ak

THREE PARCEL

020i00mn	Transmit nm to Ai
021i00mn	Transmit ones complement of nm to Ai
040i00mn	Transmit nm to Si
041i00mn	Transmit ones complement of nm to Si
10hi00mn	Read from Memory address [nm + (Ah) + DBA] to Ai
11hi00mn	Write (Ai) to Memory address [nm + (Ah) + DBA]
12hi00mn	Read from Memory address [nm + (Ah) + DBA] to Si
13hi00mn	Write (Si) to Memory address [nm + (Ah) + DBA]

Hardware Trng.
YM10/19B J.E.S.

CRAY Y-MP ISSUE CONTROL OPTIONS PER MODULE

OPTION TYPE	NUMBER USED	DESCRIPTION
(HA)	1	Lower Program Counter
(HZ)	1	Upper Program Counter
(HC)	1	Upper NIP/LIP gh
(HD)	1	Lower NIP/LIP ijk
(HE)	1	Exchange Parameters XA, VL, CLN, FLAGS, MODES
(HF)	1	Exchange Parameters P, IBA, ILA, DBA, DLA
(HG)	1	Exchange Parameters Errors
(HH)	1	Interrupting Channel Dead lock, Error Channel
(HI)	1	Performance Monitor Selection
(HJ)	1	Performance Monitor Counters
(HS)	15	IB/B/T/PM No parity
(JA)	1	Address Register Issue
(JB)	1	Scalar Register Issue
(JC)	1	Vector Register Issue
(JD)	1	Functional Unit Issue
(JE)	1	Address Register Timing
(JF)	1	Vector Write Timing
(JG)	1	Scalar Register Timing Designator MUX
(JQ)	1	SR Control Byte Local Semaphores
(JR)	4	Shared Register Data Select
(JS)	4	CPU Shared Register Data RTC/PC
	41 Total	

Hardware Trng.
YM10/15 J.E.S.

CRAY Y-MP ISSUE CONTROL (OPTIONS INVOLVED)

(HA0) Option

The (HA0) option contains the P register bits (2^4 - 2^0, 2^{-1}, 2^{-2}) which is used for part of the initial Fetch address, and is also used to address the Instruction buffers. The (HA0) contains an IB Write address which is incremented as the Fetch word arrives from memory, and an IB Read address which is incremented by plus one or plus two as the Go Issue arrives. The (HA0) also decodes the Branch instructions for both conditional and unconditional branches to determine if the P register needs to be reloaded or left alone. Also, the (HA) option insures that on a Control Dump (CD) the write of the four buffers is completed.

(HZ0) Option

The (HZ0) option contains the P register bits (2^{21} - 2^5), and advances the P register by plus one or plus two, depending on the instruction that is issued. The P register can be loaded one of three ways; on an Exchange sequence, on a Branch instruction, or on a 005 instruction. The Go Issue allows the P register to advance. The P register is also compared against the IBAR 0 - 3 to check for coincidence. The (HZ0) also contains the Buffer Counter, which is cleared on an Exchange and a Control Dump, along with the Buffer valid bits.

(HC0) Option

The (HC0) option contains the NIP/LIP and LIP 1 registers for the (gh) field of instruction, or the (m9 - 15) of a 2 parcel instruction, or the (n25 - 31) of a 3 parcel instruction. The (HC) will perform a pre-decode on the (gh) field of the instruction to determine if a 2 or 3 parcel instruction has issued. The (HC) generates and fanout the Wait Exchange during the Exchange process.

(HD0) Option

The (HD0) option contains the lower half of the instruction, either (ijk) field of the instruction, or (m0 - m8) on a 2 parcel instruction, or (n16 - 24) on a 3 parcel instruction. The lower half of the instruction basically contains the operand and destination field for the instructions, so no instruction decode is needed.

(HS0 - HS7) Options

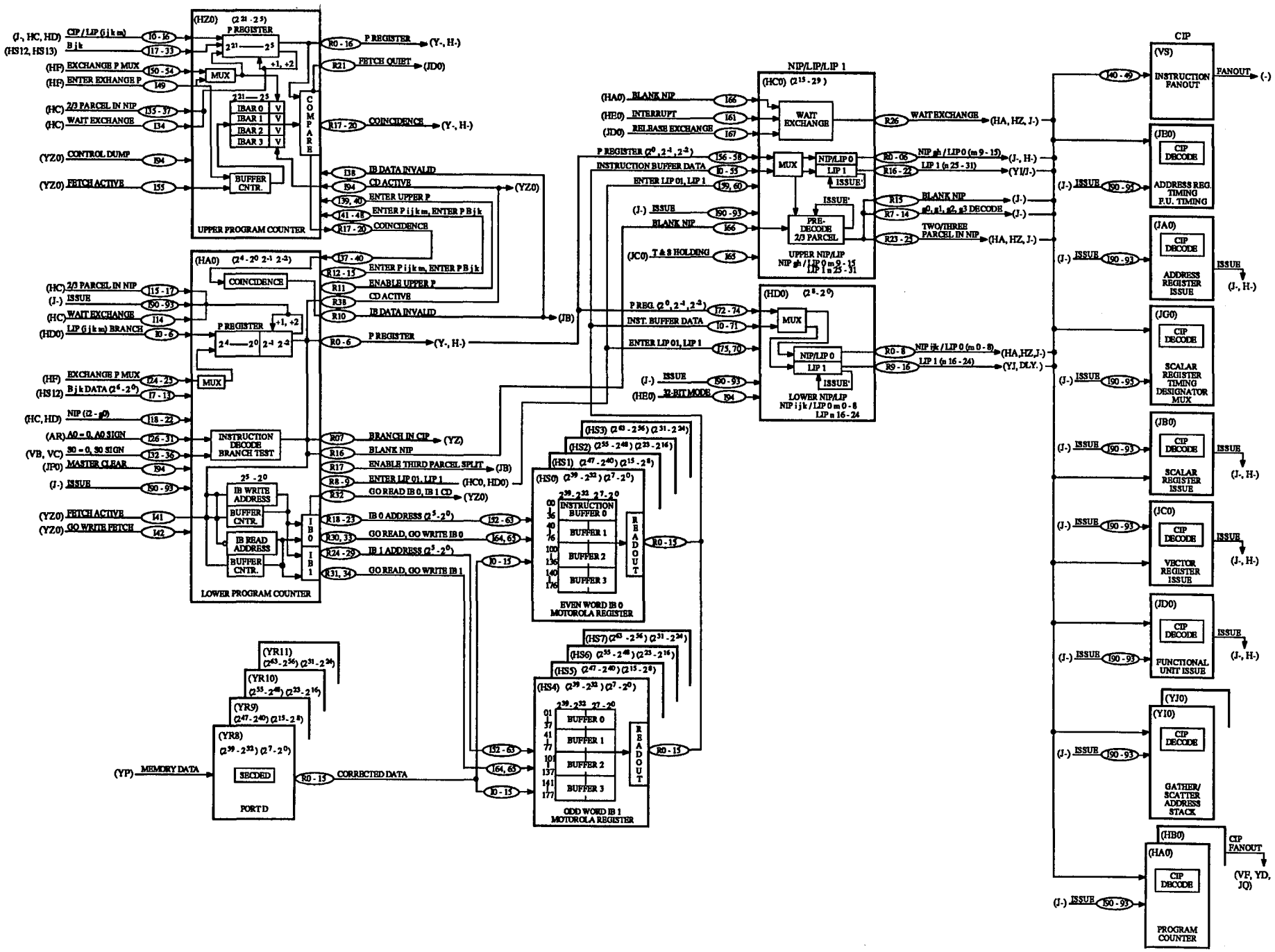
The (HS) options contain the Instruction buffers which are divided into the even/odd words. (HS0 - HS3) contain the even address words, and (HS4 - HS7) contain the odd address words. Each (HS) option contains 16 bits of the 64-bit word. The 16 bits are separated by 32 bits. (HS0) contains bits (2^0 - 2^7) and bits (2^{32} - 2^{39}). The (HS) option has 64 locations for 16 bits. The storage locations are divided into four 16 address locations. One address location would be IB0. Instruction buffer zero for data bits (2^{39} - 2^{32}) (2^7 - 2^0) are found on (HS0) for even addresses and (HS4) for odd addresses. The (HS) option also contains a bypass path which is not used.

(J-) Option

The various (J-) options contain the CIP decode for the various instructions. Before an instruction can issue, reservation must be checked on the operand, the destination fields of the instruction, and on the functional units. If a conflict occurs, the Go Issue is held until the conflict is resolved. Holding the Go Issue stops the instruction flow.

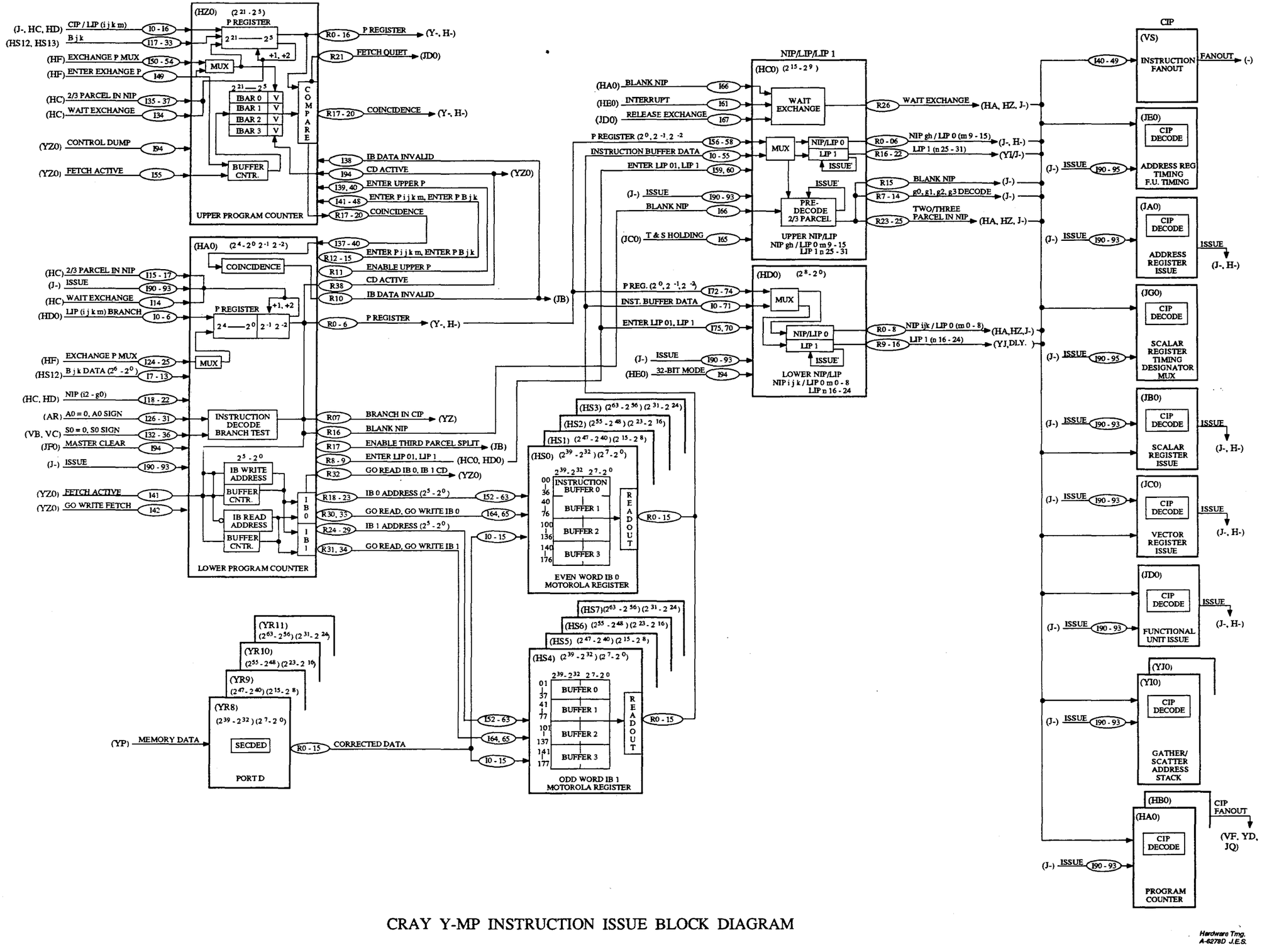
(YR8 - YR11) Options

The (YR) options receive the words from memory, and perform SECDED prior to the word being sent to the Instruction buffers or (HS) options.



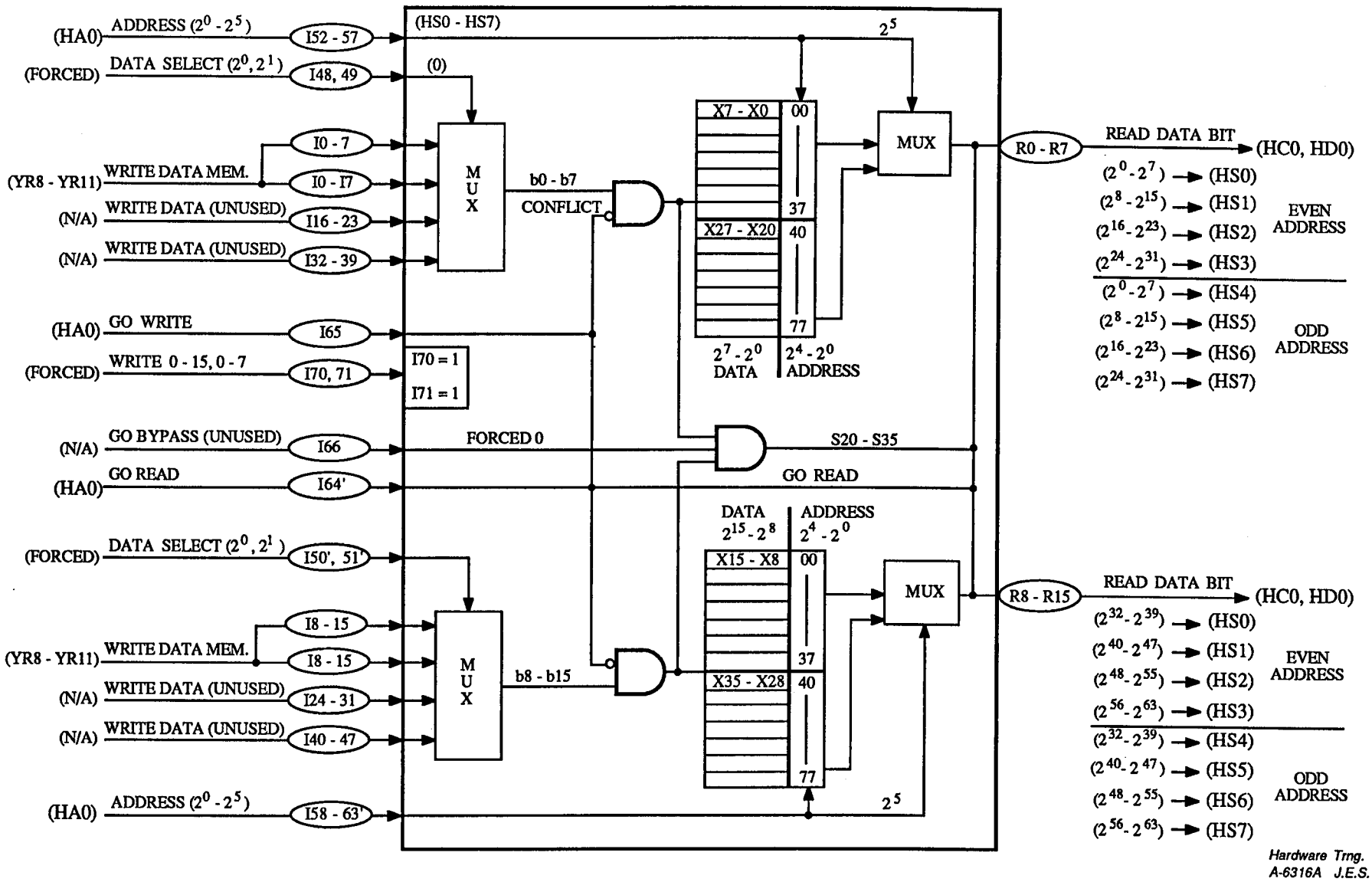
CRAY Y-MP INSTRUCTION ISSUE BLOCK DIAGRAM

CRAY PROPRIETARY



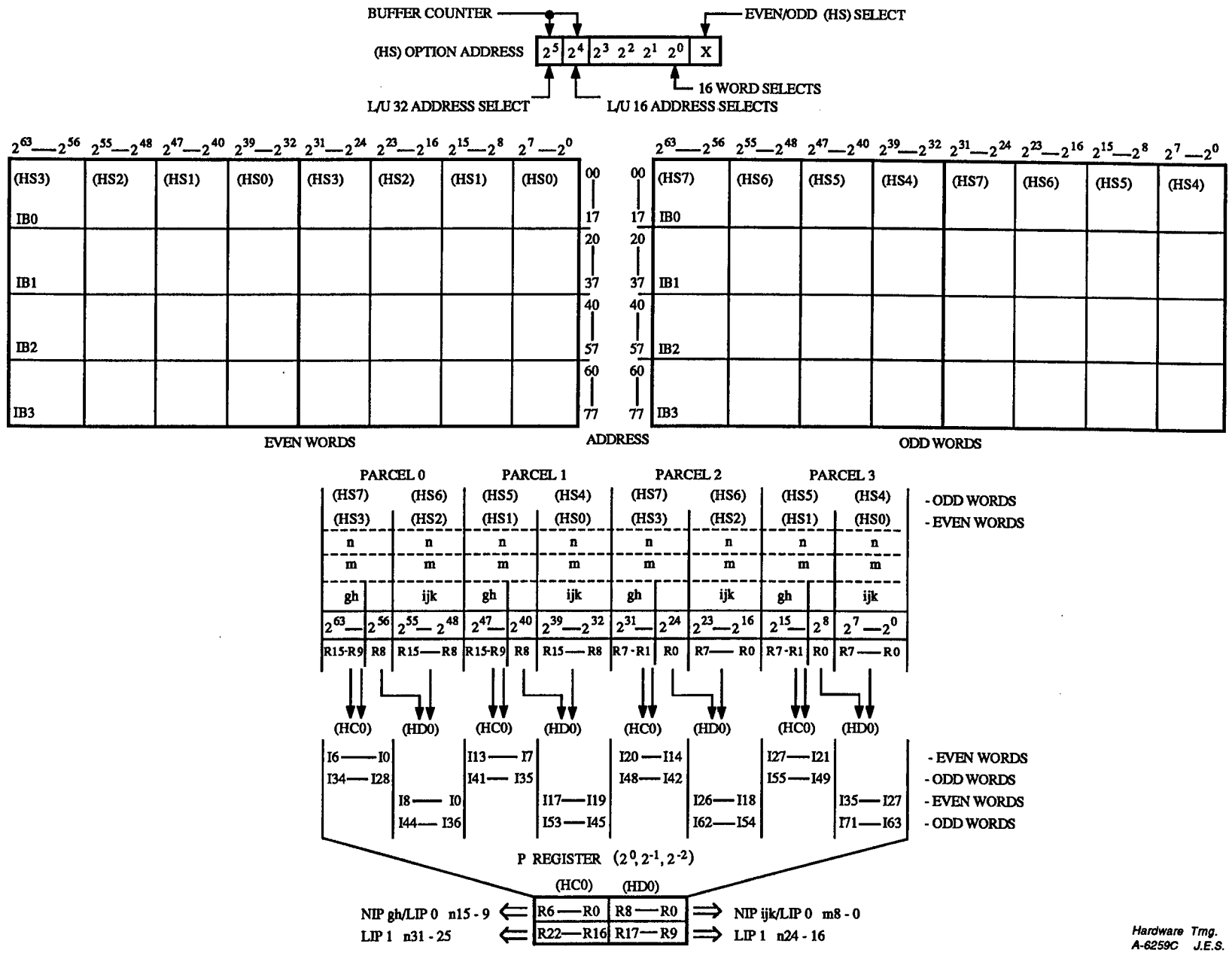
CRAY Y-MP INSTRUCTION ISSUE BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP (HS) BLOCK DIAGRAM
INSTRUCTION BUFFERS

CRAY PROPRIETARY



Hardware Tmg.
A-6259C J.E.S.

CRAY Y-MP NIP/LIP READOUT

CRAY PROPRIETARY

CRAY Y-MP A BOARD FANOUTS

(VS10) I41 - Enter S0 - 3 Upper (VC)
I42 - Vjk VFU0 (CP 1) (VB/VC)
I43 - Vjk VFU1 (CP 1) (VB/VC)
I44 - K0 (CP 1) (VD)
I45 - J/I Bit 0' (CP 1) (VD/YE)
I46 - Enter S4 - 7 Upper (VC)
I47 - J/I Bit 1' (CP 1) (VD/YE)
I48 - K1 (CP 1) (VD)
I49 - K2 (CP 1) (VD)

(VS0) I41 - K/J/I Bit 0' (CP 1) (VB/VC)
I42 - SFU Bit 0 (VB/VC)
I43 - SFU Bit 1 (VB/VC)
I44 - SFU Bit 2 (VB/VC)
I45 - I/J Bit 0 (CP 1) (VA)
I46 - K/J/I Bit 1' (CP 1) (VB/VC)
I47 - I/O Ch. # Bit 0' (HH)
I48 - I/J Bit 1 (CP 1) (VA)
I49 - I/J Bit 2 (CP 1) (VA)

(VS30) I41 - Enter S i0 (VB/VC)
I42 - H/K/J/I Bit 0 (CP 1) (YI/YJ)
I43 - H/K/J/I Bit 1 (CP 1) (He/Hh)
I44 - H/K/J/I Bit 2 (CP 1) (HE/HH)
I45 - nm4' (VF/YD)
I46 - Enter S i1 (VB/VC)
I47 - J/I Bit 2' (CP 1) (VD/YE)
I48 - Go VA (CP 1) (VA)
I49 - I/O Master Clear (VS)

(VS20) I41 - K/J/I Bit 2' (CP 1) (VB/VC)
I42 - SFU Bit 4 (CP 5) (VB/VC)
I43 - J/I Bit 1 (CP 1) (Vb/Vc)
I44 - SFU Bit 3 (VB/VC)
I45 - Vi VFU0 (CP 1) (VA)
I46 - J/I Bit 0 (CP 1) (VB/VC)
I47 - Vi VFU1 (CP 1) (VA)
I48 - Vi VFU2 (CP 1) (VA)
I49 - Vi VFU3 (VA)

(VS50) I41 - Vjk VFU2 (CP 1) (VB/VC)
I42 - Gather/Scatter (CP 1) (VB)
I43 - nm0' (VF/YD)
I44 - nm1' (VF/YD)
I45 - nm2' (VF/Yd)
I46 - nm3' (VF/YD)
I47 - nm7' (VF/YD)
I48 - nm6' (VF/YD)
I49 - nm5' (VF/YD)

(VS40) I41 - VSHIFT1 (CP 4) (VA)
I42 - H0' (CP 2) (VB/VC)
I43 - H/K/J/I Bit 2 (CP 2) (VB/VC)
I44 - J/I Select (CP 1) (VB/VC)
I45 - Bjk0 (CP 5) (YC/YD)
I46 - SSHIFT1 (CP 4) (VB/VD)
I47 - H/K/J/I Bit 1 (CP 2) (VB/VD)
I48 - Ajki0' (CP 3) (YC/YD)
I49 - Ajki1' (CP 3) (YC/YD)

(VS70) I41 - Enter S0 - 3 Lower (VB)
I42 - Bjk1 (CP 5) (YC/YD)
I43 - Bjk2 (CP 5) (YC/YD)
I44 - Bjk3 (CP 5) (YC/YD)
I45 - Bjk4 (CP 5) (YC/YD)
I46 - Enter S4 - 7 Lower (VB)
I47 - Bjk5 (CP 5) (YC/YD)
I48 - Bjk6 (CP 5) (YC/YD)
I49 - Bjk7 (CP 5) (YC/YD)

(VS60) I41 - J/I Bit 2 (CP 1) (VB/VC)
I42 - Sk Cntrl. 0 - 61 (CP 1)(VB/VC)
I43 - Ajki2' (CP 3) (YC/YD/JS)
I44 - Ajki3' (CP 3) (YC/YD/JS)
I45 - Ajki4' (CP 3) (YC/YD/JS)
I46 - Ajki5' (CP 3) (YC/YD/JS)
I47 - Ajki6' (CP 3) (YC/YD/JS)
I48 - Ajki7' (CP 3) (YC/YD/JS)
I49 - I/O Master Clear (YK/YL/YM)

(YC00) (4-2) - Port C Conflict (YC/YD)
(YD00) (1-2) - Port C Section 0 Enable (YC/YD)
(YD04) (1-2) - Port C Section 1 Enable (YC/YD)

Hardware Trng.
YM10/20 J.E.S.

CRAY Y-MP B BOARD FANOUTS

(VS11) I41 - Enter S4 - 7 Upper (VC)
I42 - I/O Syn. Bit 66 (HH)
I43 - I/O Syn. Bit 70 (HH)
I44 - I/O Syn. Bit 67 (HH)
I45 - I/O Syn. Bit 71 (HH)
I46 - Enter S0 - 3 Upper (VC)
I47 - Exch. Seq. (CP 6) (HG)
I48 - Monitor Mode' (HI/HJ)
I49 - I/O Master Clear (D-/HH)

(VS1) I41 - K/J/I Bit 1' (CP 1) (VB/VC)
I42 - I/O Syn. Bit 64 (HH)
I43 - I/O Syn. Bit 68 (HH)
I44 - I/O Syn. Bit 65 (HH)
I45 - I/O Syn. Bit 69 (HH)
I46 - K/J/I Bit 0' (CP 1) (VB/VC)
I47 - K0 (CP 1) (VF/VG/VH)
I48 - K1 (CP 1) (VF/VG)
I49 - K2 (CP 1) (VF/VG)

(VS31) I41 - Enter S i1 (VB/VC)
I42 - Vjk VFU 0 (CP 1) (VB/VC)
I43 - Vjk VFU 1 (CP 1) (VB/VC)
I44 - I/O Master Clear (YN/HD)
I45 - Ajki16' (CP 3) (YC/YD)
I46 - Enter S i0 (VB/VC)
I47 - Ajki17' (CP 3) (YC/YD)
I48 - Ajki18'(CP 3) (YC/YD)
I49 - Ajki19' (YC/YD)

(VS21) I41 - J/I Bit 0 (CP 1) (VB/VC)
I42 - SFU Bit 0 (VB/VC)
I43 - SFU Bit 1 (VB/VC)
I44 - SFU Bit 2 (VB/VC)
I45 - Ajki20' (CP 3) (YC/YD/JS)
I46 - K/J/I Bit 2' (CP 1) (VB/VC)
I47 - Ajki21' (CP 3) (YC/YD/JS)
I48 - Ajki22' (CP 3) (YC/YD/JS)
I49 - Ajki23' (CP 3) (YC/YD/JS)

(VS51) I41 - nm12' (VF/YD/AR)
I42 - nm13' (VF/YD/AR)
I43 - nm14' (VF/YD/AR)
I44 - nm15' (VF/YD/AR)
I45 - Bjk8 (CP 5) (YC/YD)
I46 - Bjk9 (CP 5) (YC/YD)
I47 - Bjk10 (CP 5) (YC/YD)
I48 - Vjk VFU2 (CP 1) (VB/VC)
I49 - Gather/Scatter (CP 1) (VB)

(VS41) I41 - SSHIFT1 (CP 4) (VB/VC)
I42 - SFU Bit 4 (CP 5) (VB/VC)
I43 - J/I Bit 1 (CP 1) (VB/VC)
I44 - SFU Bit 3 (VB/VC)
I45 - H/K/J/I Bit 1 (CP 2) (VB/VC)
I46 - VSHIFT1 (CP 4) (VA)
I47 - H/K/J/I Bit 2 (CP 2) (VB/VC)
I48 - H0' (CP 2) (VB/VC)
I49 - 32 Bit Mode (EAM') (YJ/YF/AR)

(VS71) I41 - Enter S4 - 7 Lower (VB)
I42 - Bjk11 (CP 5) (YC/YD)
I43 - Bjk12 (CP 5) (YC/YD)
I44 - Bjk13 (CP 5) (YC/YD)
I45 - Bjk14 (CP 5) (YC/YD)
I46 - Enter S0 - 3 Lower (VB)
I47 - Bjk15 (CP 5) (YC/YD)
I48 - Ajki8' (CP 5) (YC/YD)
I49 - Ajki9' (CP 5) (YC/YD)

(VS61) I41 - Ajki10' (CP 3) (YC/YD/JS)
I42 - Ajki11' (CP 3) (YC/YD/JS)
I43 - Ajki12' (CP 3) (YC/YD/JS)
I44 - J/I Select (CP 1) (VB/VC)
I45 - Ajki13' (CP 3) (YC/YD/JS)
I46 - Ajki14' (CP 3) (YC/YD/JS)
I47 - Ajki15' (CP 3) (YC/YD/JS)
I48 - J/I Bit 2 (CP 1) (VB/VC)
I49 - Sk Cntrl. 0 - 61 (CP1) (VB/VC)

(YN0) I29 - Exch. Pckge. Bit 42' (YC/YD)
I30 - Exch. Pckge. Bit 43' (YC/YD)
I31 - Exch. Pckge. Bit 44' (YC/YD)
I32 - Exch. Pckge. Bit 45' (YC/YD)
I33 - Exch. Pckge. Bit 46' (YC/YD)

(YN1) I29 - Exch. Pckge. Bit 47' (YC/YD)
I30 - Exch. Pckge. Bit 48' (YC/YD)
I31 - Exch. Pckge. Bit 49' (YC/YD)
I32 - Exch. Pckge. Bit 50' (YC/YD)
I33 - Exch. Pckge. Bit 51' (YC/YD)

Hardware Trng.
YM10/21 J.E.S.

CRAY Y-MP B BOARD FANOUTS

(VF1) I8 -
I9 -
I10 -
I11 -

(YB0) I86 - Port A Range Error (YB)
I87 - Port B Range Error (YB)
I88 - Port A+B Range Error (HE)
I89 - Port C Range Error (HE)

(YD5) (1-2) - Port C Sec. 3 Enable (YC/YD)

(JR1) I31 - Correctable Memory Error (HE)
I46 - Uncorrectable Memory Error (HE)
I47 - Register Parity Error (HE)
I48 - Uncorrectable I/O Memory Error (D-)

(VH0) I66 -

(VH0) I61 - Not Exch. Seq. (YA)
I62 - Not Exch. Seq. (CP 5) (VS)

(YC1) (4-2) - Port D Conflict (YC/YD/YQ)

(YD1) (1-2) - Port C Sec. 2 Enable (YC/YD)

Hardware Trng.
YM10/22 J.E.S.

CRAY Y-MP C BOARD FANOUTS

(VS12) I41 - Enter S0 - 3 Upper (VC)
I42 - Vjk VFU0 (CP 1) (VB/VC)
I43 - Vjk VFU1 (CP 1) (VB/VC)
I44 - Resync. I/O M.C. (EXT.)
I45 - I/J Bit 0 (CP 1) (JF)
I46 - Enter S4 - 7 Upper (VC)
I47 - Resync. I/O M.C. (EXT.)
I48 - I/J Bit 1 (CP 1) (JF)
I49 - I/J Bit 2 (CP 1) (JF)

(VS32) I41 - Enter S i0 (VB/VC)
I42 - MCU Interrupt (EXT.)
I43 - (P-1 + P-2)' (WS') (HG)
I44 - nm24' (VF/AR)
I45 - Not Go Write Seq. (VNU) (HG)
I46 - Enter S i1 (VB/VC)
I47 - Go VA (CP 1) (VA)
I48 - nm16' (VF/YD/AR)
I49 - nm17' (VF/YD/AR)

(VS52) I41 - Vjk VFU2 (CP 1) (VB/VC)
I42 - Gather/Scatter (VB)
I43 - nm18' (VF/YD/AR)
I44 - nm19' (VF/YD/AR)
I45 - nm20' (VF/YD/AR)
I46 - nm21' (VF/YD/AR)
I47 - nm22' (VR/YD)
I48 - nm23' (VF/YD)
I49 - Bjk16 (CP 5) (YC/YD)

(VS72) I41 - Enter S0 - 3 Lower (VB)
I42 - Bjk17 (CP 5) (YC/YD)
I43 - Bjk18 (CP 5) (YC/YD)
I44 - Bjk19 (CP 5) (YC/YD)
I45 - Bjk20 (CP 5) (YC/YD)
I46 - Enter S4 - 7 Lower (VB)
I47 - Bjk21 (CP 5) (YC/YD)
I48 - Bjk22 (CP 5) (YC/YD)
I49 - Bjk23 (CP 5) (YC/YD)

(VF2) I8 - Enter Data Stack (A, S)
I9 - Enter Data Stack (A, S)
I10 - Enable EC Port B (YQ)
I11 - Enable Vk to CB (YQ)

(YL0) I48 - Gated B Data Bit 24 (YC/YD)
I49 - Gated B Data Bit 25 (YC/YD)

(YC2) (4-2) - Floating-point Range Error (VS42)

(YD6) I49 -

(VS2) I41 - K/J/I Bit 0' (CP 1) (VB/VC)
I42 - SFU Bit 0 (VB/VC)
I43 - SFU Bit 1 (VB/VC)
I44 - SFU Bit 2 (VB/VC)
I45 - I/J Bit 0 (CP 1) (VA)
I46 - K/J/I Bit 1' (CP 1) (VB/VC)
I47 - Resync. CPU M.C. (EXT.)
I48 - I/J Bit 1 (CP 1) (VA)
I49 - I/J Bit 2 (CP 1) (VA)

(VS22) I41 - K/J/I Bit 2' (CP 1) (VB/VC)
I42 - SFU Bit 4 (CP 5) (VB/VC)
I43 - J/I Bit 1 (CP 1) (VB/VC)
I44 - SFU Bit 3 (VB/VC)
I45 - Vi VFU0 (CP 1) (VA)
I46 - J/I Bit 0 (CP 1) (VB/VC)
I47 - Vi VFU1 (CP 1) (VA)
I48 - Vi VFU2 (CP 1) (VA)
I49 - Vi VFU3 (VA)

(VS42) I41 - VSHIFT1 (CP 4) (VA)
I42 - H0' (CP 2) (VB/VC)
I43 - H/K/J/I Bit 2 (CP 2) (VB/VC)
I44 - J/I Select (CP 1) (VB/VC)
I45 - FPE' Interrupt (HE)
I46 - SSHIFT1 (CP 4) (VB/VC)
I47 - H/K/J/I Bit 1 (CP 2) (VB/VC)
I48 - Enable Data to CB (YQ)
I49 - Enable Data to CB (YQ)

(VS62) I41 - J/I Bit 2 (CP 1) (VB/VC)
I42 - Sk Cntrl. 0 - 61 (CP 1) (VB/VC)
I43 -
I44 -
I45 -
I46 -
I47 -
I48 -
I49 -

(VH1) I66 - Disable Data to CB (YC)
I61 - Go S/V Flting. Recip. (RO)
I62 - Go S/V Flting. Add (FA)
I63 - Go S/V Flting. Mult. (MI)
I64 - Disable Data to CB (YQ)
I67 - Disable Vk to CB (YC)

(YL2) I48 - Gated B Data Bit 28 (YC/YD)
I49 - Gated B Data Bit 29 (YC/YD)

(YD2) I49 -

YM10/23

CRAY Y-MP D BOARD FANOUTS

(VS13) I41 - Enter S4 - 7 Upper (VC)
I42 - SR Cntrl. Byte Bit 2' (HS)
I43 - SR Cntrl. Byte Bit 3' (HS)
I44 - SR Cntrl. Byte Bit 4' (HS)
I45 - SR Cntrl. Byte Bit 5' (HS)
I46 - Enter S0 - 3 Upper (VC)
I47 - Cluster # Bit 0 (HS)
I48 - Cluster # Bit 1' (HS)
I49 - H0 (CP 1) (VS)

(VS3) I41 - K/J/I Bit 1' (CP 1) (VB/VC)
I42 - Sk Cntrl. 62-63 (CP 1) (VC)
I43 - LOSP-IN I/O Inter. (EXT.)
I44 - LOSP-OUT I/O Inter. (EXT.)
I45 - Cluster # Bit 2' (HS)
I46 - K/J/I Bit 0' (CP 1) (VB/VC)
I47 - K0 (CP 1) (VF/VG/VH)
I48 - K1 (CP 1) (VF/VG)
I49 - K2 (CP 1) (VF/VG)

(VS33) I41 - Enter S i1 (VB/VC)
I42 - Vjk VFU0 (CP 1) (VB/VC)
I43 - Vjk VFU1 (CP 1) (VB/VC)
I44 - Port D Ch, # Bit 0 (HG)
I45 - SSD Phase Sync. to CPU
I46 - Enter S i0 (VB/VC)
I47 - H/K/J/I Bit 0 (CP 1) (HE/HH)
I48 - H/K/J/I Bit 1 (CP 1) (VG/VH/VS)
I49 - H/K/J/I Bit 2 (CP 1) (VG/VS)

(VS23) I41 - J/I Bit 0 (CP 1) (VB/VC)
I42 - SFU Bit 0 (VB/VC)
I43 - SFU Bit 1 (VB/VC)
I44 - SFU Bit 2 (VB/VC)
I45 - 32 Bit Mode' (EAM') (VG/JA)
I46 - K/J/I Bit 2' (CP 1) (VB/VC)
I47 - Prog. Clock Int. (HE)
I48 - Not Go I/O Write (HH)
I49 - Port D Ch. # Bit 1 (HG)

(VS53) I41 - nm25' (VF/YD/AR)
I42 - nm26'M (VF/YD/AR)
I43 - nm27' (VF/YD/AR)
I44 - nm28' (VF)/YD/AR)
I45 - nm29' (VF/YD/AR)
I46 - nm30' (VF/YD/AR)
I47 - nm31' (VF/YD/AR)
I48 - Vjk VFU2 (CP 1) (VB/VC)
I49 - Gather/Scatter (CP 1) (VB)

(VS43) I41 - SSHIFT1 (CP 4) (VB/VC)
I42 - SFU Bit 4 (CP 5) (VB/VC)
I43 - J/I Bit 1 (CP 1) (VB/VC)
I44 - SFU Bit 3 (VB/VC)
I45 - H/K/J/I Bit 1 (CP 2) (VB/VC)
I46 - VSHIFT1 (CP 4) (VA)
I47 - H/K/J/I Bit 1 (CP 2) (VB/VC)
I48 - H0' (CP 2) (VB/VC)
I49 - 32 Bit Mode' (EAM') (VF/YL)

(VS73) I41 - Enter S4 - 7 Lower (VB)
I42 - SR Ctrl. Byte Bit 3' (DE/AF)
I43 -
I44 -
I45 - G3 (CP 1) (VG)
I46 - Enter S0 - 3 Lower (VB)
I47 -
I48 - Ajki24' (CP 3) (YC/YD)
I49 - Ajki25' (CP 3) (YC/YD)

(VS63) I41 - Ajki26' (CP 3) (YC/YD/JS)
I42 - Ajki27' (CP 3) (YC/YD/JS)
I43 - Ajki28' (CP 3) (YC/YD/JS)
I44 - J/I Select (CP 1) (VB/VC)
I45 - Ajki29' (CP 3) (YC/YD/JS)
I46 - Ajki30' (CP 3) (YC/YD/JS)
I47 - Ajki31' (CP 3) (YC/YD/JS)
I48 - J/I Bit 2 (CP 1) (VB/VC)
I49 - Sk Cntrl. 0 - 61 (CP 1) (VB/VC)

(VF3) I8 - H/K/J/I Bit 0' (CP 1) (VF/VG/VH)
I9 - J/K/H Bit 0' (CP 1) (VF/VG)
I10 - J/K/H/ Bit 1' (CP 1) (VF/VG)
I11 - J/K/H/ Bit 2' (CP 1) (VF/VG)

(YO0) I24 - Exch. Pckge. Bit 52 (YC/YD)
I25 - Exch. Pckge. Bit 53 (YC/YD)
I26 - Exch. Pckge. Bit 54 (YC/YD)
I27 - Exch. Pckge. Bit 55 (YC/YD)

(YC3) (4-2) - Enter Data Stack Port C (YC/YD)

(YD3) (1-2) - Select A Data (YD/YC)

(YD7) (1-2) - Select B Data (YD/YC)

(YL1) I48 - Gated B Data Bit 26 (YD/YC)
I49 - Gated B Data Bit 27 (YD/YC)

(YL3) I48 - Gated B Data Bit 30 (YC/YD)
I49 - Gated B Data Bit 31 (YC/YD)

(JR3) I48 - VHISP I/O Inter. (EXT.)

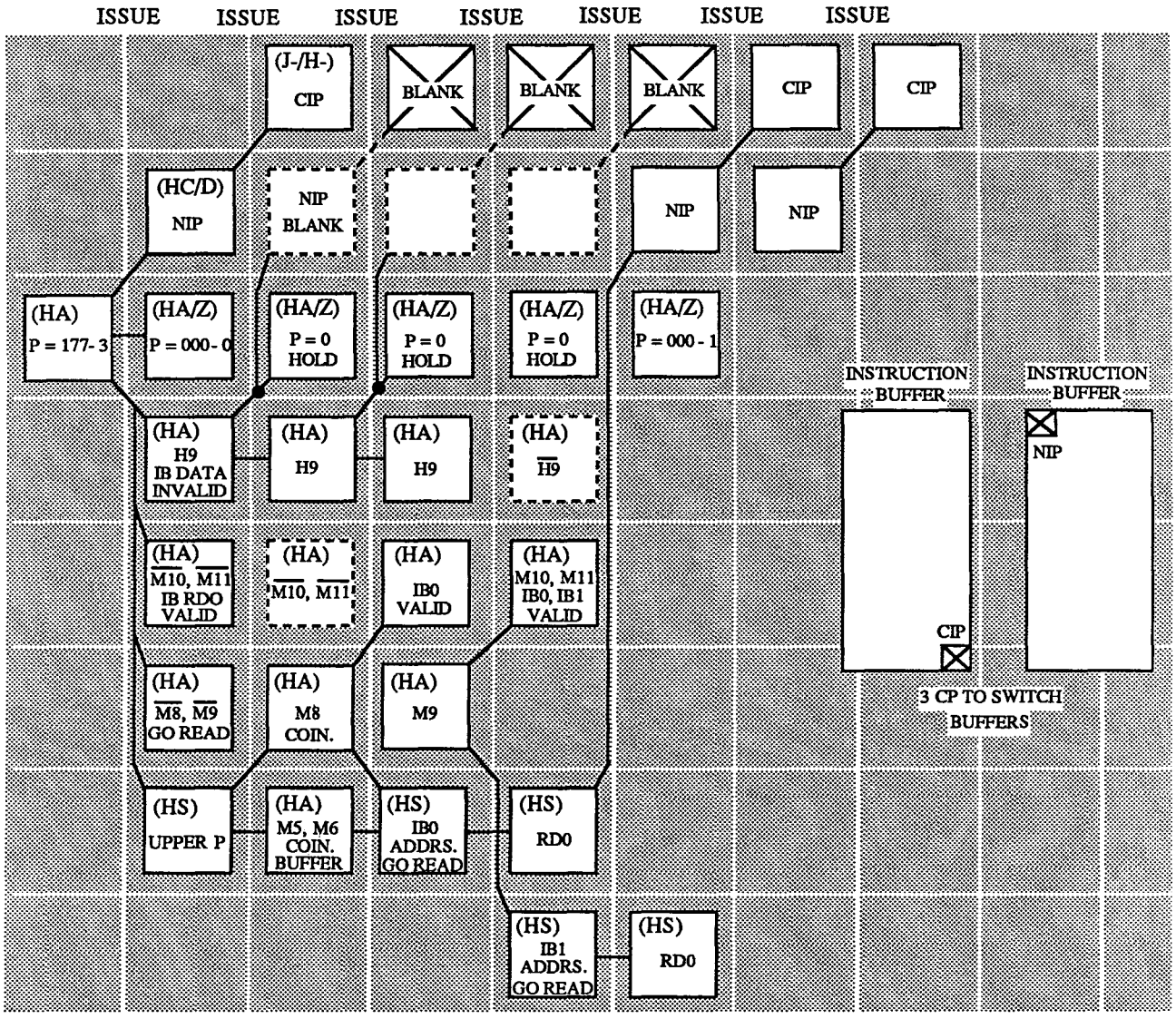
YM10/24

(JG0) Test Point Select Chart

Decode				T.P.	Selects
I97	I96	I95	I94	Term	Description
0	0	0	0	P0	CIP k 2^0
0	0	0	1	P1	CIP k 2^1
0	0	1	0	P2	CIP k 2^2
0	0	1	1	P103	CIP j 2^0
0	1	0	0	P104	CIP j 2^1
0	1	0	1	P105	CIP j 2^2
0	1	1	0	P6	CIP i 2^0
0	1	1	1	P7	CIP i 2^1
1	0	0	0	P8	CIP i 2^2
1	0	0	1	P9	CIP h 2^0
1	0	1	0	P10	CIP h 2^1
1	0	1	1	P11	CIP h 2^2
1	1	0	0	P12	CIP g 2^0
1	1	0	1	P13	CIP g 2^1
1	1	1	0	P14	CIP g 2^2
1	1	1	1	P15	CIP g 2^3

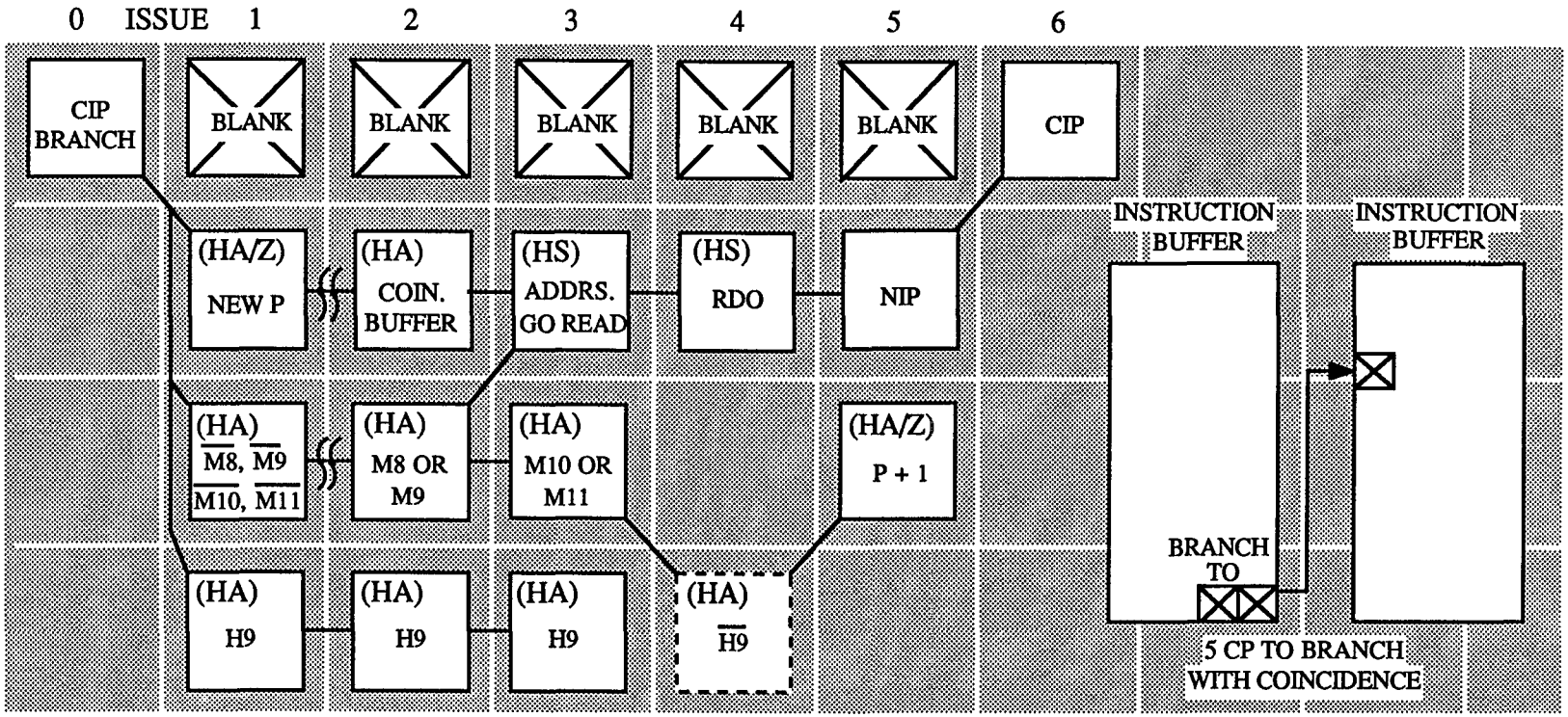
Hardware Trng.
A-6319 J.E.S.

CRAY Y-MP
INSTRUCTION ISSUE CIP TEST POINTS



Hardware Trng.
A-6495A J.E.S.

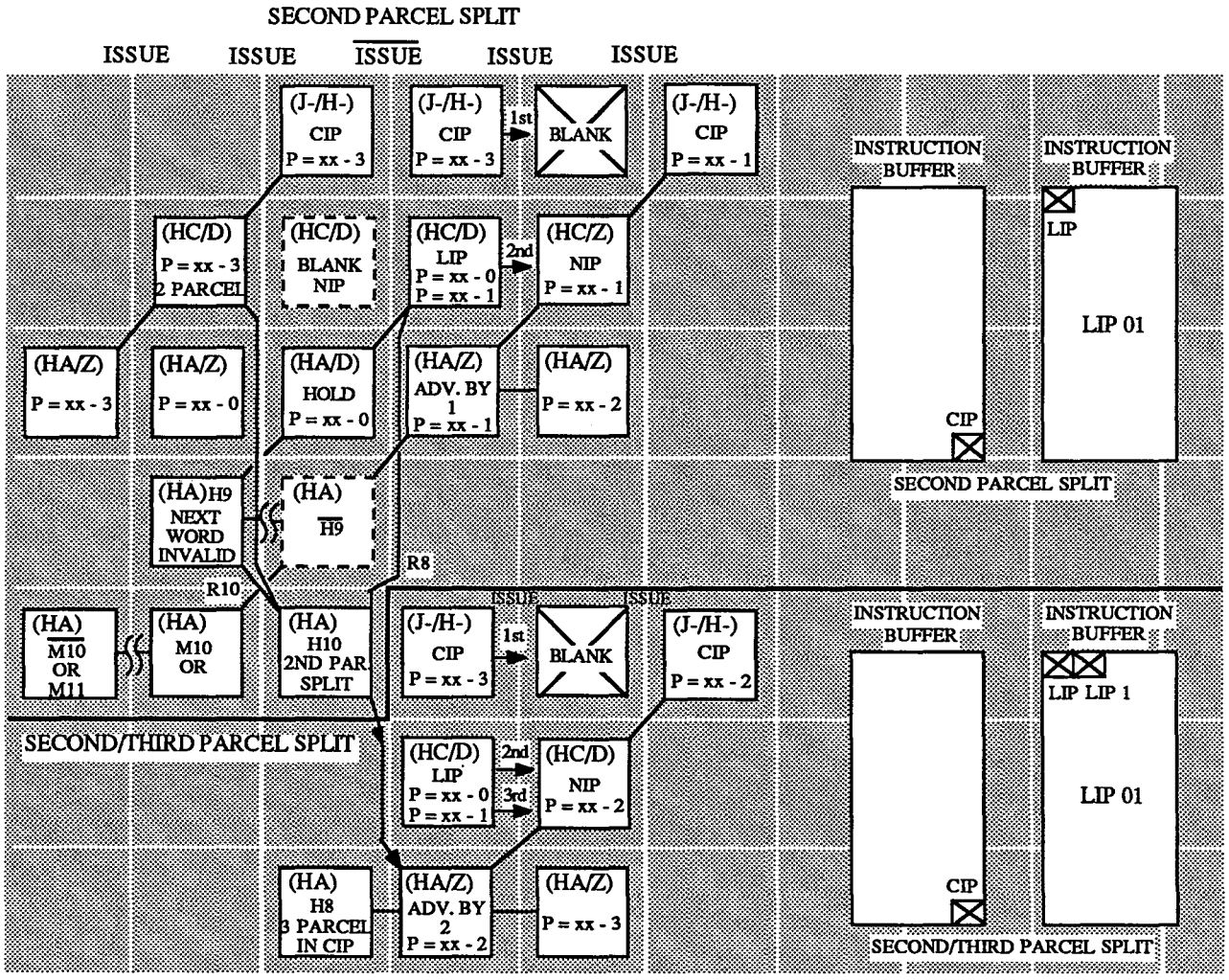
CRAY Y-MP (HA) OPTION BUFFER BOUNDARY
WITH COINCIDENCE



Hardware Trng.
A-6494A J.E.S.

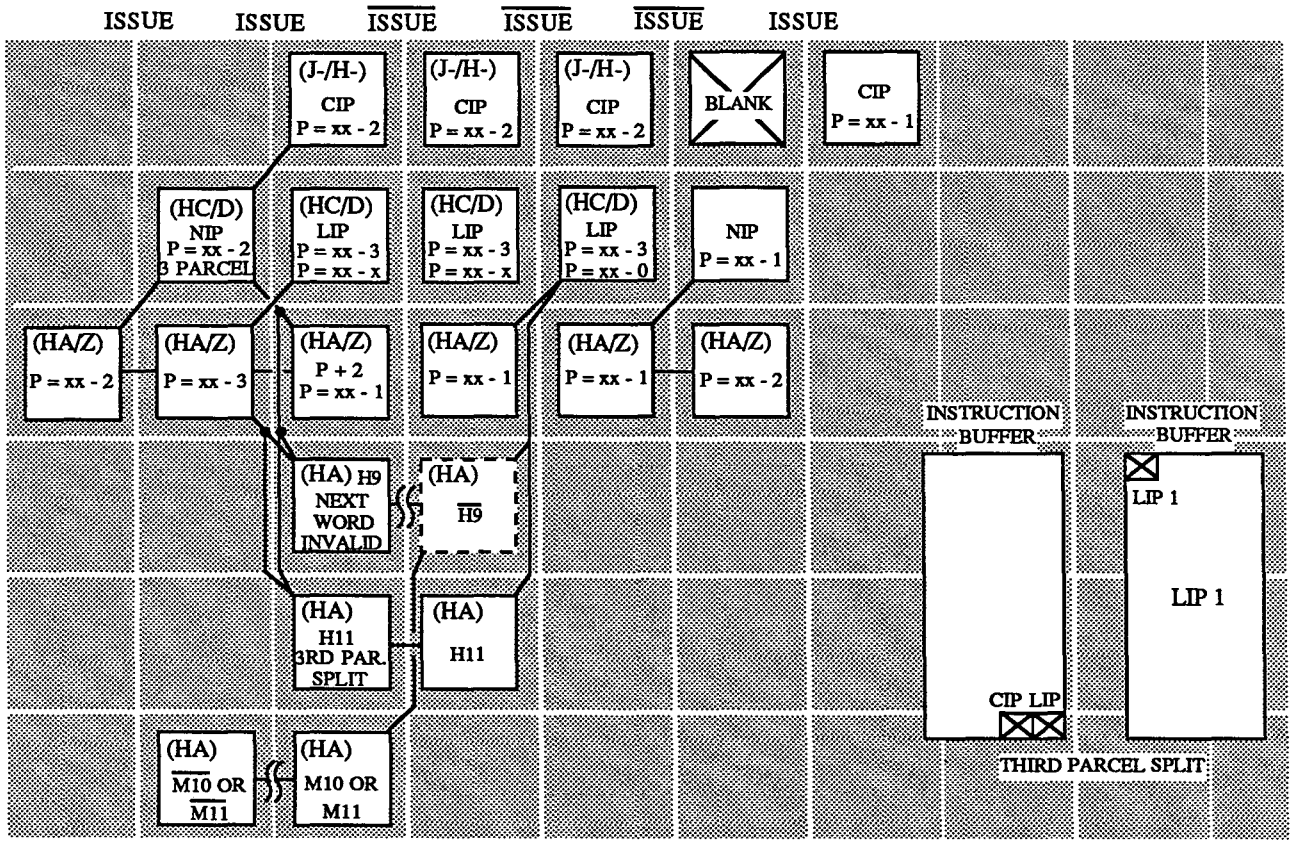
CRAY Y-MP BRANCH WITH COINCIDENCE TIMING

CRAY PROPRIETARY



Hardware Trng.
A-6493B J.E.S.

CRAY Y-MP SECOND/THIRD PARCEL SPLIT



Hardware Trng.
A-6492B J.E.S.

CRAY Y-MP THIRD PARCEL SPLIT

CRAY Y-MP SHARED REGISTERS

The Shared registers in the CRAY Y-MP/8 consist of nine clusters of Shared B registers (SB), Shared T registers (ST), and the Semaphore register (SM), while the Cray Y-MP/1/2/4 contains seven clusters. The clusters are numbered one through nine, or one through seven, with the cluster zero not being used. Any processor using cluster zero will cause the Shared register instruction to execute as a no-op. There are eight SB registers numbered SB0 - SB7, and eight ST registers. There are 32 SM registers which are 1-bit in length. The SB registers are 32 bits in length, while the ST registers are 64 bits in length.

The Shared registers (SB, ST) are used to pass A register values and S register values from one processor to the next. The Shared registers are the intermediate storage between the processors.

The Semaphore registers are used to control the flow of a program between two or more processors. This causes a particular CPU to Hold Issue until it is appropriate to begin execution. In order for a processor to use the Semaphore, there must be another processor in execution with the same cluster number value. If not, the processor that is waiting on Semaphore would be in a Deadlock condition. This will set a flag in the Exchange Package, causing an exchange. Another condition which would cause a Deadlock would be when all processors are in the same cluster, and all processors are Holding Issue. This would cause all processors to do an exchange.

The Shared registers (SB, ST, SM) take up a lot of space, containing approximately 96 options, not counting the fanouts. Each CRAY Y-MP board layer contains approximately 78 options, which means that the Shared registers take up well over one board layer. The 96 Shared register options are distributed among the eight CPU modules. Each CPU module contains approximately 12 options of the Shared registers.

The different types of instructions allow the SB register to be loaded from the A register, and the A register to be loaded from the SB register. The ST registers are allowed to be loaded from the S registers, or the S register can be loaded from the ST register. The SM registers can be loaded from the S register bits (2^{63} - 2^{32}), and vice versa. Any bit in the Semaphore can be set or cleared. A 0034jk instruction or Test and Set will examine the contents of SMjk. If SMjk = 0, the instruction will set SMjk equal to a one. However, if upon examining SMjk it is found that SMjk = 1, the processor will Hold Issue until SMjk has been cleared.

One of the easiest ways to understand how the Shared registers are used is to follow the flow of the Program Exercise on the following page. Notice that part of the program is executing in CPU 0 cluster 1, and the other half is executing in CPU 1 cluster 1. The idea of the program is to demonstrate shared processing among two CPUs. In order for the sequence of events to work properly, CPU 0 must be in execution before CPU 1. CPU 0 will set SM00 in step 1, such that when CPU 1 does a Test and Set on SM00, it will Hold Issue until CPU 0 clears SM00. Likewise, when CPU 1 sets SM01 in step 1, it will cause CPU 0 to Hold Issue when CPU 0 gets to step 11. CPU 0 will Hold Issue until CPU 1 clears SM01 in step 11. When CPU 0 processes its job in steps 2 - 6, it will pass the location of where in memory its results are kept by using the SB register. CPU 1 will read up the value of the SB register and continue to process the program in steps 3 - 8. In summary, you can get a feel for how the Test and Set instruction can be used with this program, and the importance of the SB/ST registers.

The difficulty in the Shared registers is knowing what clusters the various CPUs are using, and being able to multiplex the A/S register to any SB/ST register in any one of nine clusters. To

Hardware Trng.
YM09/11A J.E.S.

CRAY Y-MP SHARED REGISTERS PROGRAM EXERCISE

Supply the machine code for the following operations

CPU 0 VL = 100 CLUSTER 7

STEP

1. Set SM00 _____ 3700
2. Load A3 with 4 _____ 20300 01000
3. Set interprocessor interrupt for CPU 4 _____ SIPI A3 1431
4. Load A0 with 1000_8 _____ 200 00 01000
5. Read Memory to V0 _____ V0, A0 A 176000
6. Put all ones into S1 _____ 42100
7. Do logical sum of V0 S1 → V0 _____ 14201 0
8. Increment A0 by 7000_8 _____
9. Load A1 with 20000_8 _____ 20100 020000
10. Subtract A0 from A1 → A0 _____ 31010
11. Jump to exit if A0 = 0 _____ JAZ 10000 Exit
12. SB7 ← A0 _____
13. Write V0 to memory _____ where
14. Decrement A0 by 6000_8 _____
15. Clear SM00 _____
16. T & S SM01 _____
17. Jump to Step 3 _____
18. Exit _____

CPU 4 VL = 100 CLUSTER 7

STEP

1. Clear interprocessor interrupt _____
2. Set SM01 _____
3. T & S SM00 _____
4. Read SB7 → A0 _____
5. Read Memory to V0 _____
6. Put all ones into S1 _____
7. Logical product of S1 V0 → V0 _____
8. Increment A0 by 10000_8 _____
9. Load A1 with 30000_8 _____
10. Subtract A0 from A1 → A0 _____
11. Jump to exit if A0 = 0 _____
12. Write V0 to memory _____
13. Clear SM01 _____
14. Jump to Step 2 _____
15. Exit _____

Hardware Trng.
A-6390C J.E.S.

allow only one Shared register instruction to issue between the CPUs, the (JC) options will check between the other (JC) options with a CPU A - H SR request against a SR in progress. When a conflict between CPUs exists, physical CPU 0 has the highest priority, followed by CPU 1 - 7. CPU 0 cannot make another request until all other conflicting CPUs have gone.

Shared Register SB/ST Write Operation

The Shared registers block diagram shows the SB/ST being stored in the (HS14 - HS15) options. The SB/ST for clusters 1 - 7 are stored in the (HS14) option, while (HS15) contains the SB/ST registers for clusters 8 and 9. The (HS) options are controlled by the (JR0 - 3) options. The (JR) options each contain 2 bits of the Shared registers to be sent to the (HS) options, while the 32 (JR) options are spread across all eight CPU modules.

The (JR) SR byte control and data bits arrive from the various (JS0 - 3) options. If CPU 0 wants to load S2 into ST3 for a 073233 instruction in cluster 3, the (JS0 - 3) in CPU 0 would pass the byte controls, and then the S2 data bits to all (JR0 - 3) options in all CPUs. The CPU number (0, 0, 0) in progress would have been supplied by the (JQ0) in CPU 0 to all (JR) options. Each (JR) option receives data bits (N, N+32) along with a portion of the controls, which the (JR0 - 3) share among themselves per each CPU. The SR byte controls are decoded to determine what type of operation to perform R/W of the SB/ST registers. The cluster number is also decoded, and becomes part of the address which is sent to the (HS14 - 15) SB/ST registers. Upon assembling the write address, the S2 data is then stored into the ST3 register in cluster 3 on the (HS14) option. The S2 data was originally read from the S registers on the (VB, VC) options, and passed to the (JS0 - 3) options in CPU 0. The (JS0 - 3) then distributed the S2 data bits to the 32 (JR) options. The (JR) options then loaded the S2 data bits into the SB/ST registers located on the (HS) option.

Shared Register SB/ST Read Operation

To read the contents of SB4 and send the results to A5 for cluster 4 for CPU 0, a 026547 instruction would be used. The instruction decode at CIP time would occur on the (JC) option in CPU 0. The (JC) would first determine if a Shared register instruction was in progress before proceeding. If not, the (JC) will generate a request to the other (JC) option to determine if a conflict among any other CPU exists. If no other CPUs are requesting a Shared register operation, CPU 0 (JC) option would procede by sending its SR controls and SR in progress to the (JQ0) option in CPU 0. The (JQ0) will pass the SR controls to the (JS0 - 3) options to allow the SB4 data to be sent to the A5 registers. The (JQ0) also sends out a SR in progress, which is encoded on the (YR11) option to represent CPU 0. The (YR11) supplies this information to the (JR0 - 3) options. The (JS0 - 3), upon receiving the SR controls from the (JQ0) option, will pass the SR controls to the (JR0 - 3) options. The (JR) option will decode the SR controls and determine that SB4 is to be read from cluster 4. Among all CPUs, the (JR0 - 3) will supply the (HS14) with the proper address to read SB4 in cluster 4. Once the (HS14) has been addressed, the (HS14) will output the contents of SB4 back to the (JR0 - 3) options. The (JR) options will then pass the SB data to the (JS0 - 3) option in CPU 0, and then the (JS0 - 3) will send the SB data to the A register. The (JQ) will send to the (JA, JB) the four or five clock period access times which are needed to check for Access Conflicts, and load the results into (Ai) or (Si).

Hardware Trng.
YM09/12 J.E.S.

Semaphore Operation

The Semaphores (SM) are located in two places, with the Master SM kept on the (JR) option, and the Local SM kept on the (JQ) option. The Semaphores are 1-bit registers, with 32 Semaphore registers per cluster. There are eight (JQ) options, with one per CPU module. Each CPU module also contains a Local Semaphore register. The Local Semaphores on the (JQ) are controlled by the (JR0 - JR3) common controls. When the (JR) option decodes a Semaphore instruction to Write Clear/Set the SM, the (JR) will pass the control terms to the (JQ) and (JS) options. The Local SM and Master SM are updated at the same time.

If the Local Semaphore is to be loaded, as in a 073i02 instruction, the (JS) will select the SM data from the Scalar register and pass it to the (JQ) option. The (JQ) will accept the SM data if the local CPU has the same cluster number as the global cluster number. During a Clear/Set of the Semaphore, the (JR) will pass the byte controls to the (JQ) option. Here also, if the cluster number of the local CPU matches the global cluster number, the Local Semaphore will be changed according to the byte controls. The Master SM for the appropriate cluster number will always be updated.

When a 0034jk instruction is decoded on the (JC) option, it is on a CPU level. After getting the go ahead to make a SR request, the (JC) option will send a Test and Set signal, along with the byte controls to the (JQ) within the same CPU to perform a Test and Set instruction on the cluster that the CPU is working within. If the Semaphore flag is set to a one, the (JQ) will pass a Go Test and Set signal back to the (JC) option, allowing the CPU to Hold Issue until the Semaphore flag gets cleared. The (JQ) also sends a signal called waiting on Semaphore to the (YR) option, which the (YR) passes to the (HH) option to check for a Deadlock condition. When the 0034jk instruction is cleared prior to Holding Issue or the 0034jk instruction performs a Test and Set on Semaphore (jk) and finds that the Semaphore flag is equal to a zero, the (JQ) option must then set the Semaphore flag in all the (JQ) options with the same cluster value. To do this, the (JQ) will pass its SR control bytes to the (JS) option, which will distribute the SR control bytes to all 32 (JR) options. The (JR) options will decode the SR control bytes, along with the cluster number, and pass the set Semaphore (jk) control byte to all the (JQ) options in all CPU modules, along with the cluster number. The (JQ) will compare the global cluster number against their Local cluster number. If they match, the (JQ) will allow the Semaphore (jk) to be set. The Master SM will also be updated, depending on the SMjk bit and the cluster value.

Shared Register Instructions

The shared registers are made up of the semaphores, shared B and shared T registers on the Cray X-MP. The CRAY Y-MP shared registers are made up of the same registers as the CRAY X-MP, but access to the shared register path is also required by the I/O instructions, interprocessor interrupt, and real-time clock. The master copy of all shared registers exists in only one place - global. The global set of registers can only be modified one register every 2 CPs. This is done with requests from any active CPU present in the chassis (8 for CRAY Y-MP8). The control associated with any one CPU is called local, standing for "local to the CPU." For a CRAY Y-MP8 there are 8 sets of local control. Present in the local control is a single semaphore register associated with the cluster that the CPU is executing in. For all instructions, the global set of registers is the source or destination for all read, write, or modifications. For any instruction changing the semaphores, the global set will be changed, and any local set will be modified if it is in the same cluster. By maintaining a local set of the semaphores in the CPU, less time is necessary to determine if a test and set instruction may issue. The time it takes to determine if an instruction can issue, (including conflict resolution), and inform any other CPU of the intention to modify a shared register takes 2 CPs. This is why the maximum rate of issue for shared register instructions is 2 CPs per instruction. Any shared register instruction from a CPU has to go through several steps to actually access the shared registers. Upon receiving the instruction into CIP (Current Instruction Parcel), issue is always dropped for 1 CP unconditionally. This allows the control time to determine if any other CPU is trying to access the shared registers at the same time (multiple CPU conflict). After conflict arbitration is complete, the reference now goes to the global registers. On a read shared register, the data is retrieved from the register and sent to the CPU. On a write shared register, the global register is entered or modified. If the write is to a semaphore register, then any CPU within the same cluster will have its local copy of the semaphore register updated.

To help determine what happens in a particular CPU, the timing for issue and CIP have been included. Issue down means that the current instruction is stopped in CIP and will stay there until issue goes back up. Since issue always goes down for at least 1 CP, all shared register instructions will be in CIP for at least 2 CPs.

Timing between CPUs is much more difficult to determine. Instructions can never by closer together than every 2 CPs. This implies that CPU 0 can be in CP 4 of a 72xx3, CPU 3 can be in CP 2 of a 0036xx, and CPU 2 can be ready to issue a 0034xx in CP 0.

Conflict Arbitration on CRAY Y-MP8

The CRAY Y-MP always arbitrates the request at the local control. To actually send a reference to the shared registers requires the instruction to make it past two steps in the local control. The first step, in CP 0 (CIP), is to determine whether this instruction has highest priority. The second step, in the case of the 0034xx instruction in CP 6 (still CIP), is to determine if this instruction can actually go or not due to the Semaphore bit being set or if test and sets are disabled. During CP 0 (CIP), the priority is determined. There are two levels of priority. The first level determines if a CPU request can be set. Once a CPU has made a request and is set, no other CPU with a lower number can have its request set until the first CPU has been serviced. While the first CPU is being serviced (is in progress) any other CPU with a higher number can have its request set.

Hardware Trng.
YM29/01 J.E.S.

There is still a lockout possibility that happens when the entire loop time, (including the time within the lock, 0034xx to 0036xx), becomes shorter than the time it takes for all the CPUs to go through the lock sequentially. To prevent starvation, the entire loop time has to be greater than [(# of CPUs - 1) * (amount of time spent inside lock)]. The mode of starvation is that the higher CPU numbers will starve first. This is due to the second level of request arbitration.

For example, if CPU 5 makes a request and gets Request set, CPU 0 - 4 can make a request but it will not get Request set. However, CPU 7 can make a request and be set. Now CPU 6 can make a request but can not be set because CPU 7 has been set. If no requests are set and there are simultaneous requests from two or more CPUs, all requests will be set.

Once the CPUs have their request set, the second level determines which CPU will go in progress first. The progress priority goes from CPU 0 - 7, with CPU 0 having highest priority and CPU 7 having lowest priority.

The request is also used as a shared register Busy signal. Normally it is left set for 1 CP after the reference is sent to global. This will guarantee that references will be no closer together than every 2 CPs. In the case of the 0014xx and I/O instructions, it may stay up for many clock periods, signifying that the shared registers are busy that many clock periods. The timing that follows for the CRAY Y-MP is signified by shared registers reserved. This reservation will block any subsequent references to shared registers until it clears. A reference, once set, will execute unless a couple of conditions happen. If the Semaphore bit is set, or if test and sets are disabled, the 0034xx is aborted and no attempt will be made to make another request until the bit clears and test and sets are enabled.

CPU 2 and 5 Shared Request

CPU 0							
CPU 1							
CPU 2		Makes request	Holds in CIP Request cannot be set	Request set	In progress Request still set	In progress	Done
CPU 3							
CPU 4							
CPU 5	Makes request	Request set	Goes in progress Request set still up	In progress	Done		
CPU 6							
CPU 7							

NOTE: This is not a timing diagram. This is a sequence diagram.

Hardware Trng.
A-8305 J.E.S.

CRAY PROPRIETARY

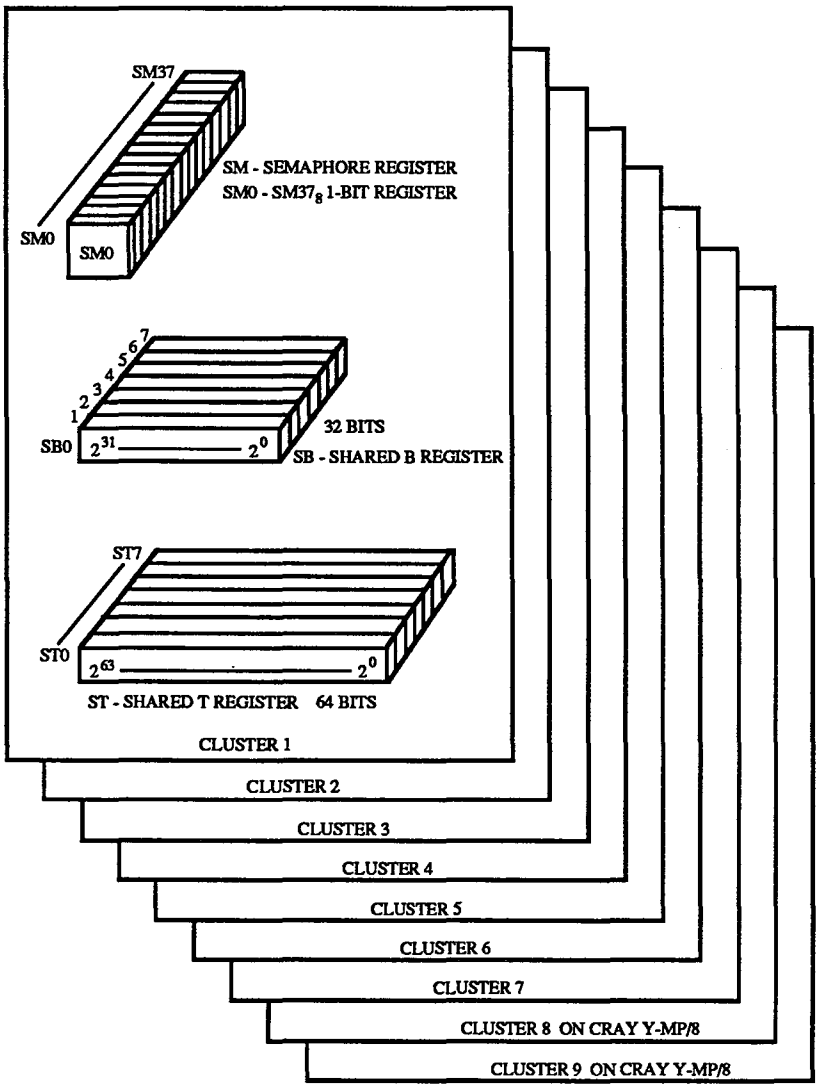
CPU 1, 4 and 6 Shared Request

CPU 0									
CPU 1		Makes request	Holds request Not set	Holds request Not set	Holds request Not set	Hold request Gets set	Goes in progress Request set still up	In progress Request se goes down	Done
CPU 2									
CPU 3									
CPU 4	Makes request	Request set	In progress Request set still up	In progress Request set goes down	Done				
CPU 5									
CPU 6		Makes request	Request set	Holds request set	Goes in progress Request set goes down	Done			
CPU 7									

NOTE: This is not a timing diagram. This is a sequence diagram.

Hardware Trng.
A-8304 J.E.S.

CRAY PROPRIETARY



NOTE: SM0 = 2^{63} IN SCALAR REGISTER
SM31 = 2^{32} IN SCALAR REGISTER

SM =	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Si =	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}

Hardware Trng.
A-6395A J.E.S.

CRAY Y-MP SHARED REGISTERS

CRAY Y-MP SHARED REGISTER INSTRUCTIONS

READ/WRITE SB/ST/SM REGISTER

026ij7 - Transmit (SBj) to Ai
027ij7 - Transmit (Ai) to SBj
072ij3 - Transmit (STj) to Si
073ij3 - Transmit (Si) to STj
072i02 - Transmit (SM) to Si loaded into Si bits (2^{63} - 2^{32}), clears (2^{31} - 2^{0}), 2^{63} of Si is SM bit 2^{0}
073i02 - Transmit (Si) to SM. Write SM from (Si) bits (2^{63} - 2^{32})

OPERATING SEMAPHORES

0034jk - Test and set SMjk $0 \leq jk \leq 37_8$
Test SMjk if value is zero, instruction sets SMjk. If value is one, instruction hold issue until SMjk is cleared.
0036jk - Clear SMjk $0 \leq jk \leq 37_8$
0037jk - Set SMjk $0 \leq jk \leq 37_8$
** 0014j3 - Set cluster number to expression = (Aj), where $0 \leq (Aj) \leq 11$
CRAY Y-MP mode; (0 = no - op)

**Priviledged to monitor mode

Hardware Trng.
YM09/17A J.E.S.

SHARED REGISTERS (OPTIONS INVOLVED)

(AR0 - AR3) Option

The (AR) options contain the A registers, which are loaded from the (JS0 - JS3) options when loading the SB register into an A register.

(HH0) Option

The (HH0) option, in CPUs 2 and 3 only, check to see if a Deadlock condition exists among all CPUs. A Deadlock condition can be caused by all CPUs Waiting On Semaphore (WS) within the same cluster, or a CPU WS with no other CPU having the same cluster value. When a Deadlock condition occurs, the (HH) option will output a Deadlock to the (JR) option, causing the (JR) to send a Deadlock flag to the (HE) option.

(HS14 - HS15) Options

The (HS14 - HS15) options contain the ST and SB registers for all nine clusters. Cluster 0 is not used. (HS14) contains SB/ST register values for clusters 1 - 7, while (HS15) contains SB/ST register values for clusters 8 and 9. The Shared registers utilize the Bypass path on the (HS) option, which no other (HS) type option uses. The Bypass path allows the data being written into the registers to also become present on the output. This is used as a delay path or a distribution path back to the (JR) options.

(JC0) Option

The (JC) option is the issue control for the Shared register instructions. Prior to allowing a Shared register instruction to issue, the (JC) must make a SR request. All the (JC) options between the CPUs intercommunicate to any other CPU which is requesting the Shared registers within the same clock period. If an SR request exists, the (JC) will allow CPU 0 to progress first, followed by CPU 1 - 7. Once the Shared register instruction is allowed to progress, the (JC) will pass the SR control and SR in progress signals to the (JQ) option. The (JC) also decodes the Real-time clock (RTC) and Interrupt Interval instructions, and passes these instruction decodes to the (JS) options, where the Real-time clock is located.

(JQ0) Option

The (JQ) option contains the Semaphore register for whatever cluster value a particular CPU is using. There is one (JQ) per CPU module. If any CPU changes a Semaphore bit, all (JQs) within the same cluster are updated. This is done by comparing the local cluster to the global cluster number. If the two do not match, the (JQ) will not allow its Semaphore registers to be changed. However, if there is a match, the particular (JQ) will change its Semaphore registers according to the contents of the SR control. If a Test and Set instruction is issued and the particular Semaphore flag is set, the (JQ) will send a signal called Waiting On Semaphore, which is compared on the (HH) option for a Deadlock condition.

(JR0 - JR3) Options

There are four (JR) options per CPU module. Each (JR) option handles two bits of the ST register or one bit of the SB register, and one bit of the SM register for nine clusters. The ST register bits are spread apart by 32 bits. The (JR) options supply the control to load the SB/ST register values into the (HS14 - HS15) options. The (JR) options receive from CPUs 0 - 7 the (JS) options, from which the SB/ST data is to be loaded into the (HS) options, and also the SR control bytes from the various CPUs. The SR control bytes are divided among the four (JR) options as they come in from the CPUs. Which (JR) handles what control bits is determined by the forced conditions on the (JR) option. The SR control bytes are then shared across the four (JR) options. Part of these SR control bytes contain the cluster number, which becomes part of the (HS) option address and is sent to the (JQ) options for a Test and Set instruction. The (JR) option will decode the SR control bytes to determine what type of instruction has issued. If the instruction is a Test and Set, the (JR) will pass the Test and Set controls to the (JQ) option along with the cluster number, allowing the (JQ) option to Set/Clear or load the SM if the clusters match, along with updating the Master SM. If the instruction decoded is a Read/Write of the SB/ST registers, the (JR) will carry out the instruction function and pass the read data of the ST/SB to the (JS) options.

(JS0 - JS3) Options

There are four (JS) options, all which operate independently from the other (JS) options. The (JS) options are used to multiplex the ST/SB or SM register data to the proper A, S, or SM registers. Each (JS) option contains two sets of 8 bits for a total of 16 bits. (JS0) handles data bits (2^0 - 2^7) and (2^{32} - 2^{39}). The (JS) also has 16 outputs which are used to fanout the SR control bytes, and the data to the 32 (JR) options which is shared across all eight CPU modules. Contained across the four (JS) options per CPU is the Real-time clock (RTC) and the programmable clock (PC).

(JT10 - JT71) Options

The (JT) options are contained on the Shared/Clock module. One (JT) option replaces two (JR) options in a CRAY Y-MP 1/2/4 processor. Each (JT) options 4 bits of the ST register, 2 bits of the SB register, and 2 bits of the global semaphores for seven clusters. This is possible because Shared logic from CPU 4 - 7 has been deleted on the (JT) option. The CRAY Y-MP/28 has two CPU modules, which causes (JT10, JT11) not to be used. (JT10 - JT11) would replace the missing (JR0 - JR3) options in CPU 1 of a CRAY Y-MP/1.

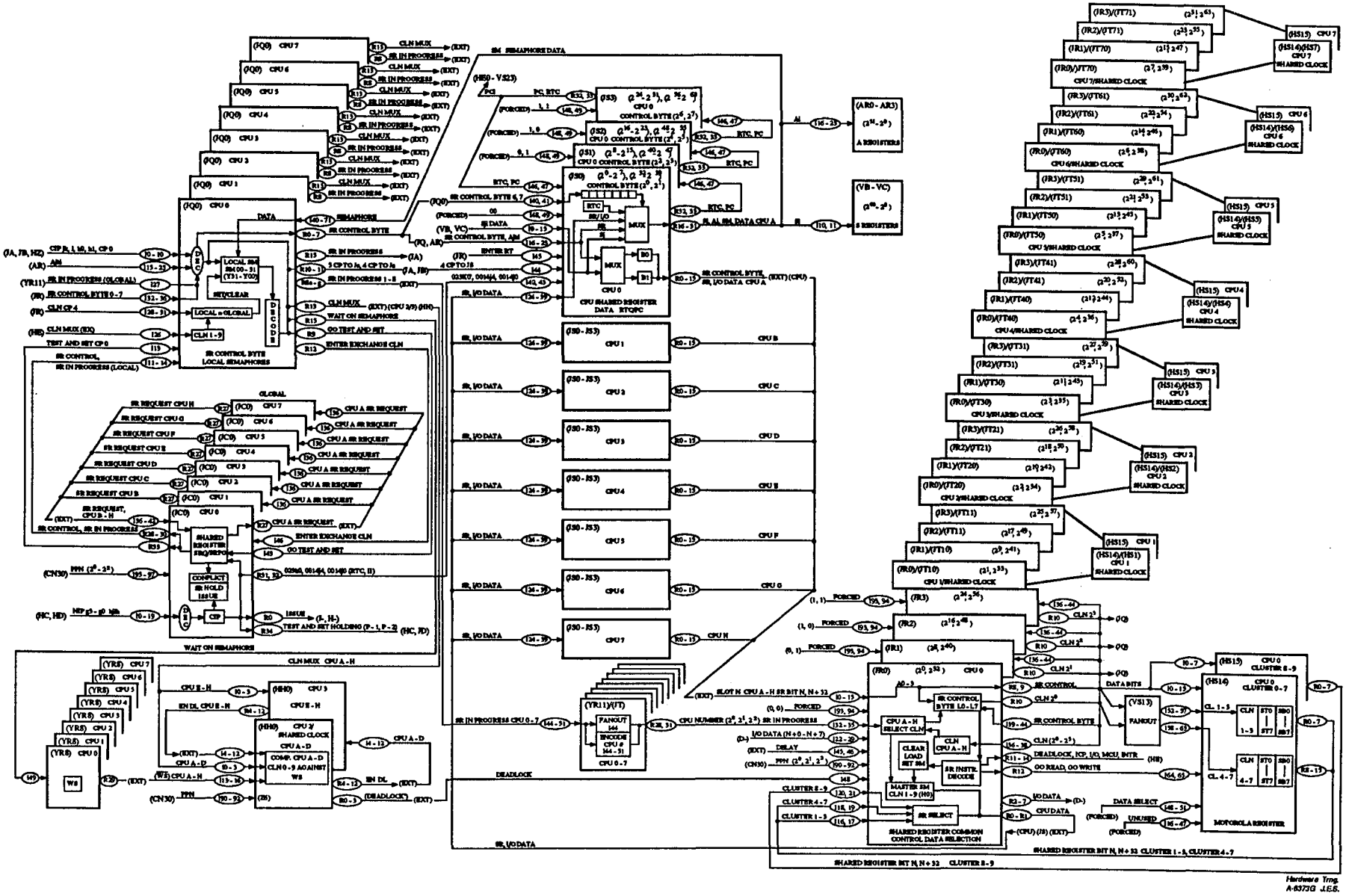
(VB - VC) Option

The (VB - VC) option contains the Scalar registers, which are loaded from the (JS0 - JS3) options when loading the SB registers into an S register.

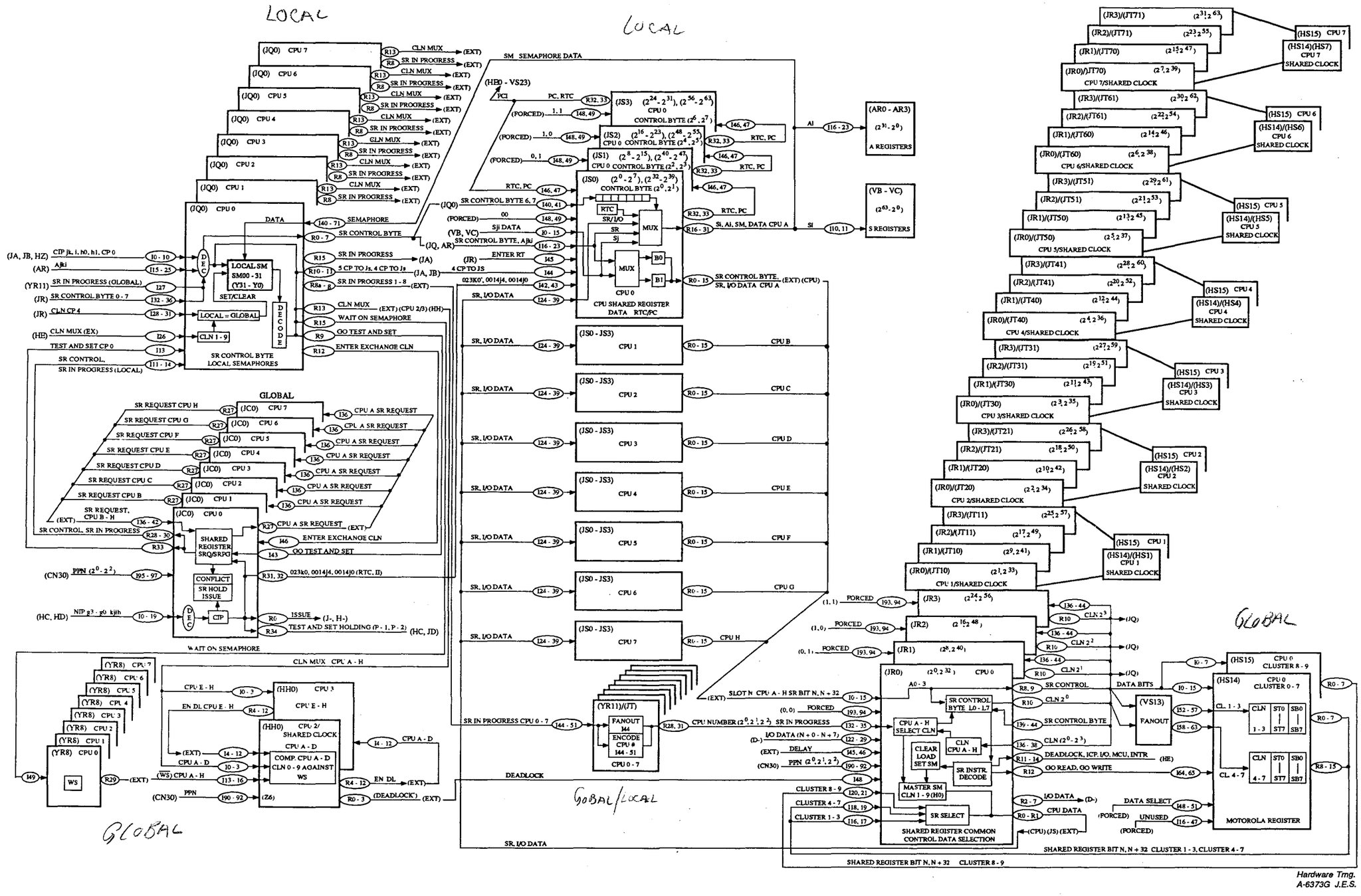
(YR8, YR11) Options

The (YR11) option will receive one signal from the various CPUs 0 - 7 and encode a value equivalent to where the signal originated from, and then fanout this value to the four (JR) options. This will happen on each CPU module. For example, CPU 3 will send a signal called SR In Progress to all CPU modules (YR11) option, which arrives as one signal. The (YR11) will then encode a value where that signal came from, in this case CPU 3, so the (YR11) will output a binary value of three (011) to the four (JR) options which arrive on the (JR) option as CPU number (2^0, 2^1, 2^2) with SR in progress.

The (YR8) will send the Wait On Semaphore (WS) flag from the (JQ0) option to the (HH0) option in CPU 2 and CPU 3.



CRAY Y-MP SHARED REGISTERS BLOCK DIAGRAM



CRAY Y-MP SHARED REGISTERS BLOCK DIAGRAM

INSTRUCTION	CAL	CONTROL BYTE (JS CONTROL) 7	6	5	4	3	2	1	0
0036jk	SMjk 0	0	0	0	j1	j0	k2	k1	k0
0034jk 0037jk	SMjk 1, TS SMjk 1	0	0	1	j1	j0	k2	k1	k0
0010jk	CA, Aj Ak	0	1	-	-	1	0	0	0
033ij0	Ai CA, Aj	0	1	-	-	0	0	0	0
0011jk	CL, Aj Ak	0	1	-	-	1	0	0	1
033ij1	Ai CE, Aj	0	1	-	-	0	0	0	1
0012j0	CI, Aj	0	1	-	-	1	0	1	0
033i00	Ai CI	0	1	-	-	0	0	1	0
0012j1	MC, Aj	0	1	-	-	1	0	1	1
0014j0	RT, Sj	0	1	-	-	-	1	0	0
0014j1	SIPI Aj	0	1	-	-	-	1	0	1
0014j3	CLN Aj	0	1	-	-	-	1	1	1
026ij7	Ai SBj	1	0	j2	j1	j0	k2	0	1
027ij7	SBj Ai	1	0	j2	j1	j0	k2	0	0
072i02	Si SM	1	1	-	-	-	-	1	0
073i02	SM Si	1	1	-	-	-	-	1	1
072ij3	Si STj	1	1	j2	j1	j0	k2	0	0
073ij3	STj Si	1	1	j2	j1	j0	k2	0	1

Shown below are the bit positions from the A registers which use the Shared register control byte path from the (JQ0) to the (JS0 - 3). The channel number for CA, CE, CI, CL uses 5 bits and the Partial address for CA, CL sends eight of its bits on the path.

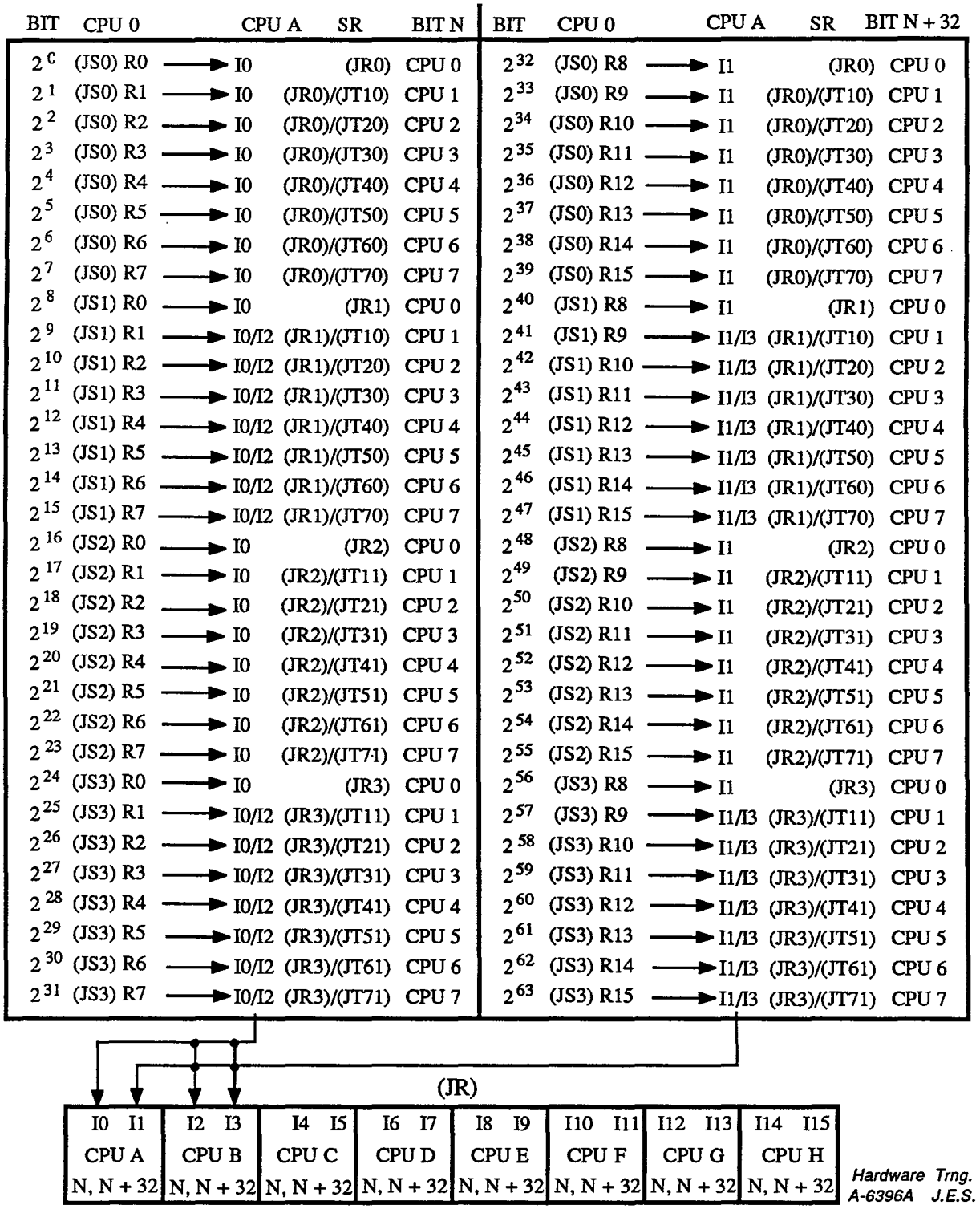
CHANNEL NUMBER (Aj)	-	-	-	4	3	2	1	0
ADDRESS (Aki)	25	24	17	16	9	8	1	0

(JS) OPTION CONTROL

I41	I40	DESCRIPTION
0	0	SELECT CONTROL BYTE
0	1	I/O DATA
1	0	Ai TO SR
1	1	SELECT S TO SR

Hardware Trng.
A-8427A D.E.

CRAY Y-MP SHARED REGISTER CONTROL BYTE



CRAY Y-MP (JR0 - 3) SHARED REGISTER BIT LAYOUT

MASTER SEMAPHORES SM CLUSTERS 1 - 9			LOCAL SEMAPHORES CLUSTERS 1 - 9			
BIT POSITION	LOCATION		BIT POSITION	LOCATION		BOOLEAN TERM
SM00 - 2^{63}	(JR3)/(JT71)	CPU 7	SM00 - 2^{63}	(JQ0)	CPU 0 - 7	Y31
SM01 - 2^{62}	(JR3)/(JT61)	CPU 6	SM01 - 2^{62}	(JQ0)	CPU 0 - 7	Y30
SM02 - 2^{61}	(JR3)/(JT51)	CPU 5	SM02 - 2^{61}	(JQ0)	CPU 0 - 7	Y29
SM03 - 2^{60}	(JR3)/(JT41)	CPU 4	SM03 - 2^{60}	(JQ0)	CPU 0 - 7	Y28
SM04 - 2^{59}	(JR3)/(JT31)	CPU 3	SM04 - 2^{59}	(JQ0)	CPU 0 - 7	Y27
SM05 - 2^{58}	(JR3)/(JT21)	CPU 2	SM05 - 2^{58}	(JQ0)	CPU 0 - 7	Y26
SM06 - 2^{57}	(JR3)/(JT11)	CPU 1	SM06 - 2^{57}	(JQ0)	CPU 0 - 7	Y25
SM07 - 2^{56}	(JR3)	CPU 0	SM07 - 2^{56}	(JQ0)	CPU 0 - 7	Y24
SM08 - 2^{55}	(JR2)/(JT71)	CPU 7	SM08 - 2^{55}	(JQ0)	CPU 0 - 7	Y23
SM09 - 2^{54}	(JR2)/(JT61)	CPU 6	SM09 - 2^{54}	(JQ0)	CPU 0 - 7	Y22
SM10 - 2^{53}	(JR2)/(JT51)	CPU 5	SM10 - 2^{53}	(JQ0)	CPU 0 - 7	Y21
SM11 - 2^{52}	(JR2)/(JT41)	CPU 4	SM11 - 2^{52}	(JQ0)	CPU 0 - 7	Y20
SM12 - 2^{51}	(JR2)/(JT31)	CPU 3	SM12 - 2^{51}	(JQ0)	CPU 0 - 7	Y19
SM13 - 2^{50}	(JR2)/(JT21)	CPU 2	SM13 - 2^{50}	(JQ0)	CPU 0 - 7	Y18
SM14 - 2^{49}	(JR2)/(JT11)	CPU 1	SM14 - 2^{49}	(JQ0)	CPU 0 - 7	Y17
SM15 - 2^{48}	(JR2)	CPU 0	SM15 - 2^{48}	(JQ0)	CPU 0 - 7	Y16
SM16 - 2^{47}	(JR1)/(JT70)	CPU 7	SM16 - 2^{47}	(JQ0)	CPU 0 - 7	Y15
SM17 - 2^{46}	(JR1)/(JT60)	CPU 6	SM17 - 2^{46}	(JQ0)	CPU 0 - 7	Y14
SM18 - 2^{45}	(JR1)/(JT50)	CPU 5	SM18 - 2^{45}	(JQ0)	CPU 0 - 7	Y13
SM19 - 2^{44}	(JR1)/(JT40)	CPU 4	SM19 - 2^{44}	(JQ0)	CPU 0 - 7	Y12
SM20 - 2^{43}	(JR1)/(JT30)	CPU 3	SM20 - 2^{43}	(JQ0)	CPU 0 - 7	Y11
SM21 - 2^{42}	(JR1)/(JT20)	CPU 2	SM21 - 2^{42}	(JQ0)	CPU 0 - 7	Y10
SM22 - 2^{41}	(JR1)/(JT10)	CPU 1	SM22 - 2^{41}	(JQ0)	CPU 0 - 7	Y09
SM23 - 2^{40}	(JR1)	CPU 0	SM23 - 2^{40}	(JQ0)	CPU 0 - 7	Y08
SM24 - 2^{39}	(JR0)/(JT70)	CPU 7	SM24 - 2^{39}	(JQ0)	CPU 0 - 7	Y07
SM25 - 2^{38}	(JR0)/(JT60)	CPU 6	SM25 - 2^{38}	(JQ0)	CPU 0 - 7	Y06
SM26 - 2^{37}	(JR0)/(JT50)	CPU 5	SM26 - 2^{37}	(JQ0)	CPU 0 - 7	Y05
SM27 - 2^{36}	(JR0)/(JT40)	CPU 4	SM27 - 2^{36}	(JQ0)	CPU 0 - 7	Y04
SM28 - 2^{35}	(JR0)/(JT30)	CPU 3	SM28 - 2^{35}	(JQ0)	CPU 0 - 7	Y03
SM29 - 2^{34}	(JR0)/(JT20)	CPU 2	SM29 - 2^{34}	(JQ0)	CPU 0 - 7	Y02
SM30 - 2^{33}	(JR0)/(JT10)	CPU 1	SM30 - 2^{33}	(JQ0)	CPU 0 - 7	Y01
SM31 - 2^{32}	(JR0)	CPU 0	SM31 - 2^{32}	(JQ0)	CPU 0 - 7	Y00

Hardware Trng.
A-6394B J.E.S.

CRAY Y-MP MASTER/LOCAL SEMAPHORE SM REGISTER LOCATIONS

	CLN								
	0, 1 8, 9	X07	X06	X05	X04	X03	X02	X01	X00
	2, 3	X27	X26	X25	X24	X23	X22	X21	X20
R TERMS		7	6	5	4	3	2	1	0
	4, 5	X15	X14	X13	X12	X11	X10	X09	X08
	6, 7	X35	X34	X33	X32	X31	X30	X29	X28
R TERMS		15	14	13	12	11	10	09	08
DATA BITS [CPU 0 - 7 (+N)]		2^{56}	2^{48}	2^{40}	2^{32}	2^{24}	2^{16}	2^{8}	2^{0}

(HS14, HS15) ADDRESS = 2^5 2^4 2^3 2^2 2^1 2^0

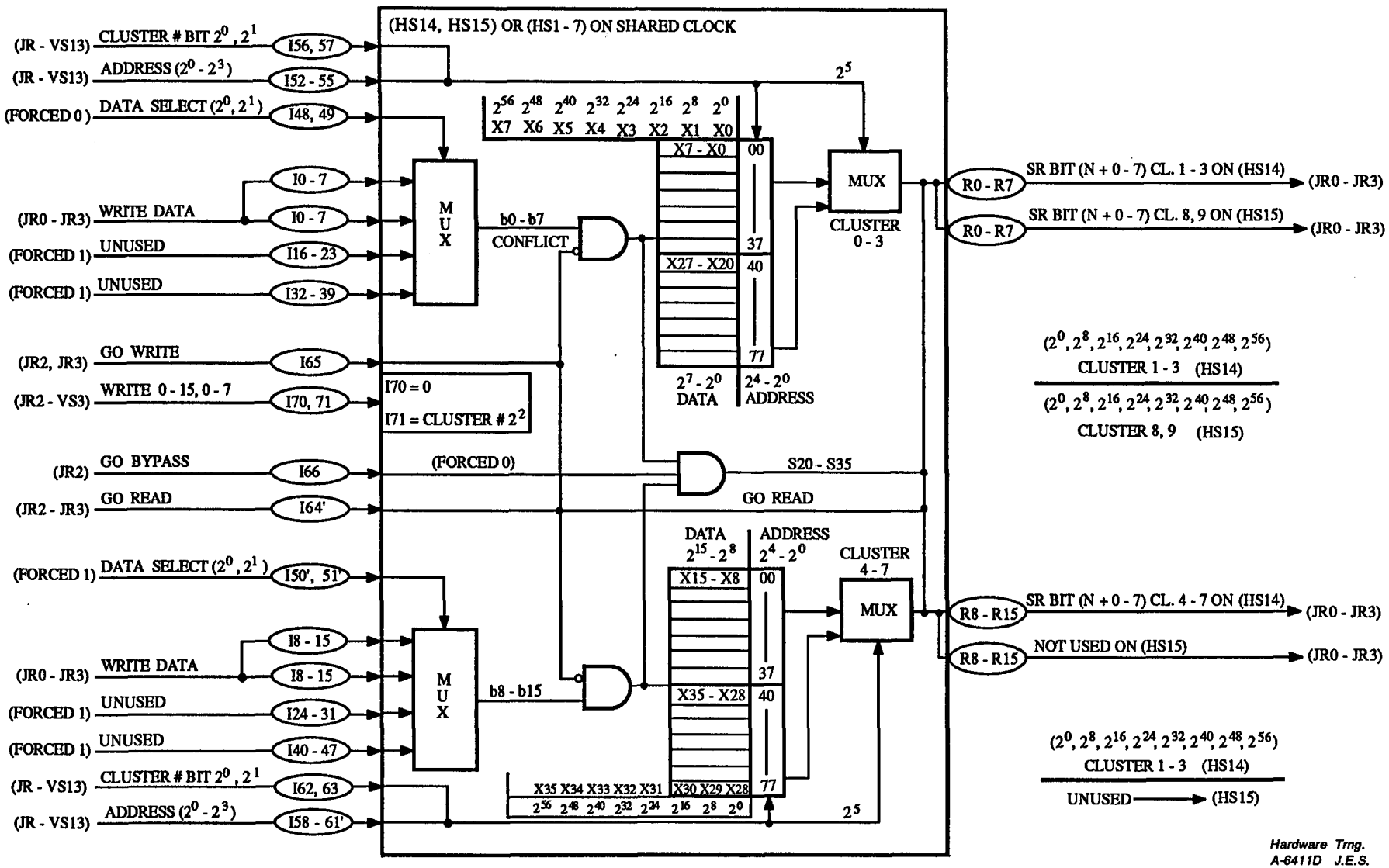
- 2^0 — SELECT SB = 1, ST = 0
- 2^1 2^2 2^3 — SB/ST SELECT, SB0 - 7, ST0 - 7
- 2^4 2^5 — CLUSTER SELECT (0 - 3)

REGISTER	ADDRESS I57	I56	I55	I54	I53	I52	DATA
SB/ST	2^5	2^4	2^3	2^2	2^1	2^0	(N + 0 - 7)
ST0	0	0	0	0	0	0	X0 - X7
SB0	0	0	0	0	0	1	CLUSTER 0
ST1	0	0	0	0	1	0	ON (HS14)
SB1	0	0	0	0	1	1	NOT USED
ST2	0	0	0	1	0	0	OR
SB2	0	0	0	1	0	1	CLUSTER 8
ST3	0	0	0	1	1	0	ON (HS15)
SB3	0	0	0	1	1	1	
ST4	0	0	1	0	0	0	X0 - X7
SB4	0	0	1	0	0	1	CLUSTER 0
ST5	0	0	1	0	1	0	ON (HS14)
SB5	0	0	1	0	1	1	NOT USED
ST6	0	0	1	1	0	0	OR
SB6	0	0	1	1	0	1	CLUSTER 8
ST7	0	0	1	1	1	0	ON (HS15)
SB7	0	0	1	1	1	1	
ST0	0	1	0	0	0	0	X0 - X7
							CLUSTER 1 ON (HS14) OR CLUSTER 9 ON (HS15)
SB3	0	1	0	1	1	1	
ST4	0	1	1	0	0	0	X0 - X7
							CLUSTER 1 ON (HS14) OR CLUSTER 9 ON (HS15)
SB7	0	1	1	1	1	1	
ST0	1	0	0	0	0	0	X20 - X27
							CLUSTER 2
SB3	1	0	0	1	1	1	
ST4	1	0	1	0	0	0	X20 - X27
							CLUSTER 2
SB7	1	0	1	1	1	1	
ST0	1	1	0	0	0	0	X20 - X27
							CLUSTER 3
SB3	1	1	0	1	1	1	
ST4	1	1	1	0	0	0	X20 - X27
							CLUSTER 3
SB7	1	1	1	1	1	1	

REGISTER	ADDRESS I63	I62	I61	I60	I59	I58	DATA
SB/ST	2^5	2^4	2^3	2^2	2^1	2^0	(N + 0 - 7)
ST0	0	0	0	0	0	0	X8 - 15
							CLUSTER 4
SB3	0	0	0	1	1	1	
ST4	0	0	1	0	0	0	X8 - 15
							CLUSTER 4
SB7	0	0	1	1	1	1	
ST0	0	1	0	0	0	0	X8 - 15
							CLUSTER 5
SB3	0	1	0	1	1	1	
ST4	0	1	1	0	0	0	X8 - 15
							CLUSTER 5
SB7	0	1	1	1	1	1	
ST0	1	0	0	0	0	0	X28 - 35
							CLUSTER 6
SB3	1	0	0	1	1	1	
ST4	1	0	1	0	0	0	X28 - 35
							CLUSTER 6
SB7	1	0	1	1	1	1	
ST0	1	1	0	0	0	0	X28 - 35
							CLUSTER 7
SB3	1	1	0	1	1	1	
ST4	1	1	1	0	0	0	X28 - 35
							CLUSTER 7
SB7	1	1	1	1	1	1	

Hardware Trng.
A-6397C J.E.S.

CRAY Y-MP (HS14, HS15) ST/SB REGISTER POSITIONS



CRAY Y-MP (HS) BLOCK DIAGRAM
SHARED REGISTERS CLUSTERS 1 - 9

CRAY PROPRIETARY

(JR0) FORCED 0 (0, 0) = (I93, I94) = (Z3, Z4) = Z10
(JR1) FORCED 0 (0, 1) = (I93, I94) = (Z3, Z4) = Z11
(JR2) FORCED 0 (1, 0) = (I93, I94) = (Z3, Z4) = Z12
(JR3) FORCED 0 (1, 1) = (I93, I94) = (Z3, Z4) = Z13

(JR0)	(JR1)	(JR2)	(JR3)
CLN BIT 2^0 CPU A - H	CLN BIT 2^1 CPU A - H	CLN BIT 2^2 CPU A - H	CLN BIT 2^3 CPU A - H

E20, I36, I38 CLUSTER NUMBERS (JR)

(JR0) = CLUSTER NUMBER CLN 0 - 3 K0 - 3

(JR0) ⟶ E20 = k0 CLN 2^0
(JR1) R10 ⟶ I36 = k1 CLN 2^1
(JR2) R10 ⟶ I37 = k2 CLN 2^2
(JR3) R10 ⟶ I38 = k3 CLN 2^3

(JR1) = CLUSTER NUMBER CLN 0 - 3 K0 - 3

(JR0) R10 ⟶ I36 = k0 CLN 2^0
(JR1) ⟶ E20 = k1 CLN 2^1
(JR2) R10 ⟶ I37 = k2 CLN 2^2
(JR3) R10 ⟶ I38 = k3 CLN 2^3

(JR2) = CLUSTER NUMBER CLN 0 - 3 K0 - 3

(JR0) R10 ⟶ I36 = k0 CLN 2^0
(JR1) R10 ⟶ I37 = k1 CLN 2^1
(JR2) ⟶ E20 = k2 CLN 2^2
(JR3) R10 ⟶ I38 = k3 CLN 2^3

(JR3) = CLUSTER NUMBER CLN 0 - 3 K0 - 3

(JR0) R10 ⟶ I36 = k0 CLN 2^0
(JR1) R10 ⟶ I37 = k1 CLN 2^1
(JR2) R10 ⟶ I38 = k2 CLN 2^2
(JR3) ⟶ E20 = k3 CLN 2^3

Hardware Trng.
A-6393A J.E.S.

CRAY Y-MP (JR) CLUSTER NUMBERS

(JR0) FORCED (0, 0) = (I93, I94) = Z3, Z4 = Z10
(JR1) FORCED (0, 1) = (I93, I94) = Z3, Z4 = Z11
(JR2) FORCED (1, 0) = (I93, I94) = Z3, Z4 = Z12
(JR3) FORCED (1, 1) = (I93, I94) = Z3, Z4 = Z13

(JR0)	(JR1)	(JR2)	(JR3)
CONTROL BYTE $2^0, 2^1$ CPU A - H	CONTROL BYTE $2^2, 2^3$ CPU A - H	CONTROL BYTE $2^4, 2^5$ CPU A - H	CONTROL BYTE $2^6, 2^7$ CPU A - H

(D0, 1), I39 - I44 = SR CONTROL BYTE

(JR0) = SR CONTROL BYTE L0 - 7

(JR0) ⟶ D0, D1 = L0 - 1 SR CONTROL BYTE 0, 1
(JR1) R8, 9 ⟶ I39, 40 = L2 - 3 SR CONTROL BYTE 2, 3
(JR2) R8, 9 ⟶ I41, 42 = L4 - 5 SR CONTROL BYTE 4, 5
(JR3) R8, 9 ⟶ I43, 44 = L6 - 7 SR CONTROL BYTE 6, 7

(JR1) = SR CONTROL BYTE L0 - 7

(JR0) R8, 9 ⟶ I39, 40 = L0 - 1 SR CONTROL BYTE 0, 1
(JR1) ⟶ D0, D1 = L2 - 3 SR CONTROL BYTE 2, 3
(JR2) R8, 9 ⟶ I41, 42 = L4 - 5 SR CONTROL BYTE 4, 5
(JR3) R8, 9 ⟶ I43, 44 = L6 - 7 SR CONTROL BYTE 6, 7

(JR2) = SR CONTROL BYTE L0 - 7

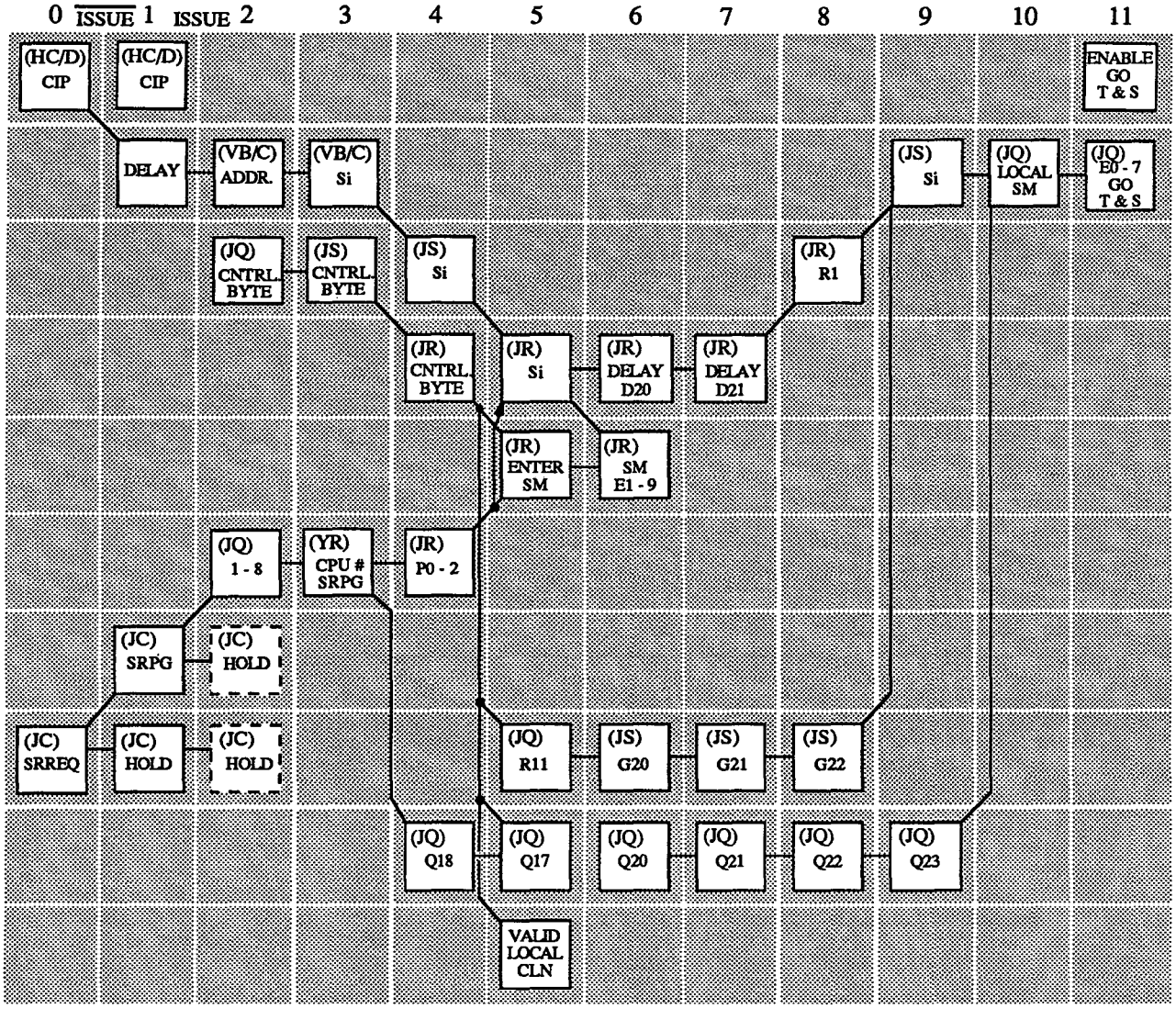
(JR0) R8, 9 ⟶ I39, 40 = L0 - 1 SR CONTROL BYTE 0, 1
(JR1) R8, 9 ⟶ I41, 42 = L2 - 3 SR CONTROL BYTE 2, 3
(JR2) ⟶ D0, D1 = L4 - 5 SR CONTROL BYTE 4, 5
(JR3) R8, 9 ⟶ I43, 44 = L6 - 7 SR CONTROL BYTE 6, 7

(JR3) = SR CONTROL BYTE L0 - 7

(JR0) R8, 9 ⟶ I39, 40 = L0 - 1 SR CONTROL BYTE 0, 1
(JR1) R8, 9 ⟶ I41, 42 = L2 - 3 SR CONTROL BYTE 2, 3
(JR2) R8, 9 ⟶ I43, 44 = L4 - 5 SR CONTROL BYTE 4, 5
(JR3) ⟶ D0, D1 = L6 - 7 SR CONTROL BYTE 6, 7

Hardware Trng.
A-6392A J.E.S.

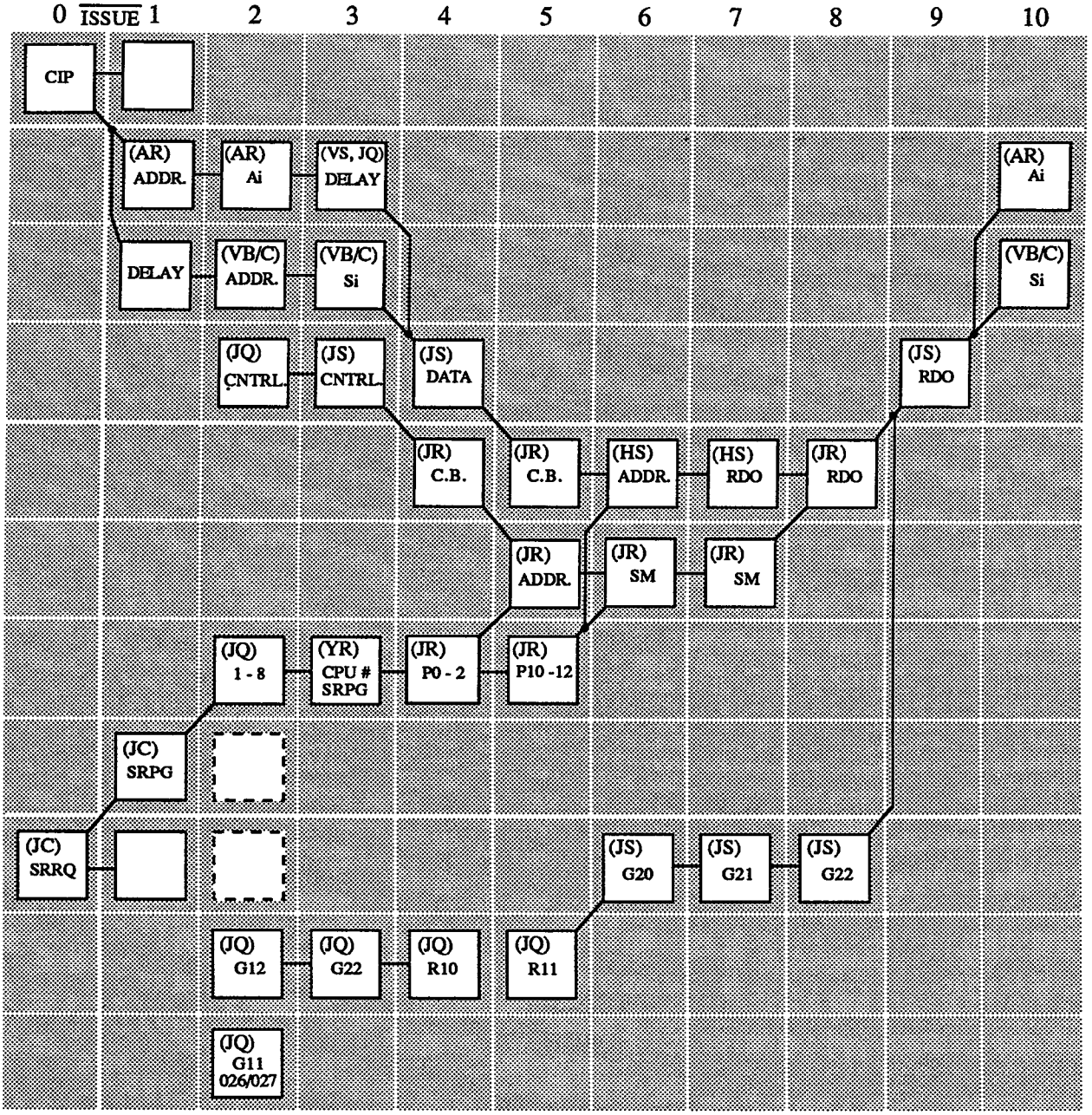
CRAY Y-MP (JR), SR CONTROL BYTES



073i02 TRANSMIT (Si) TO SM

Hardware Trng.
A-6505D J.E.S.

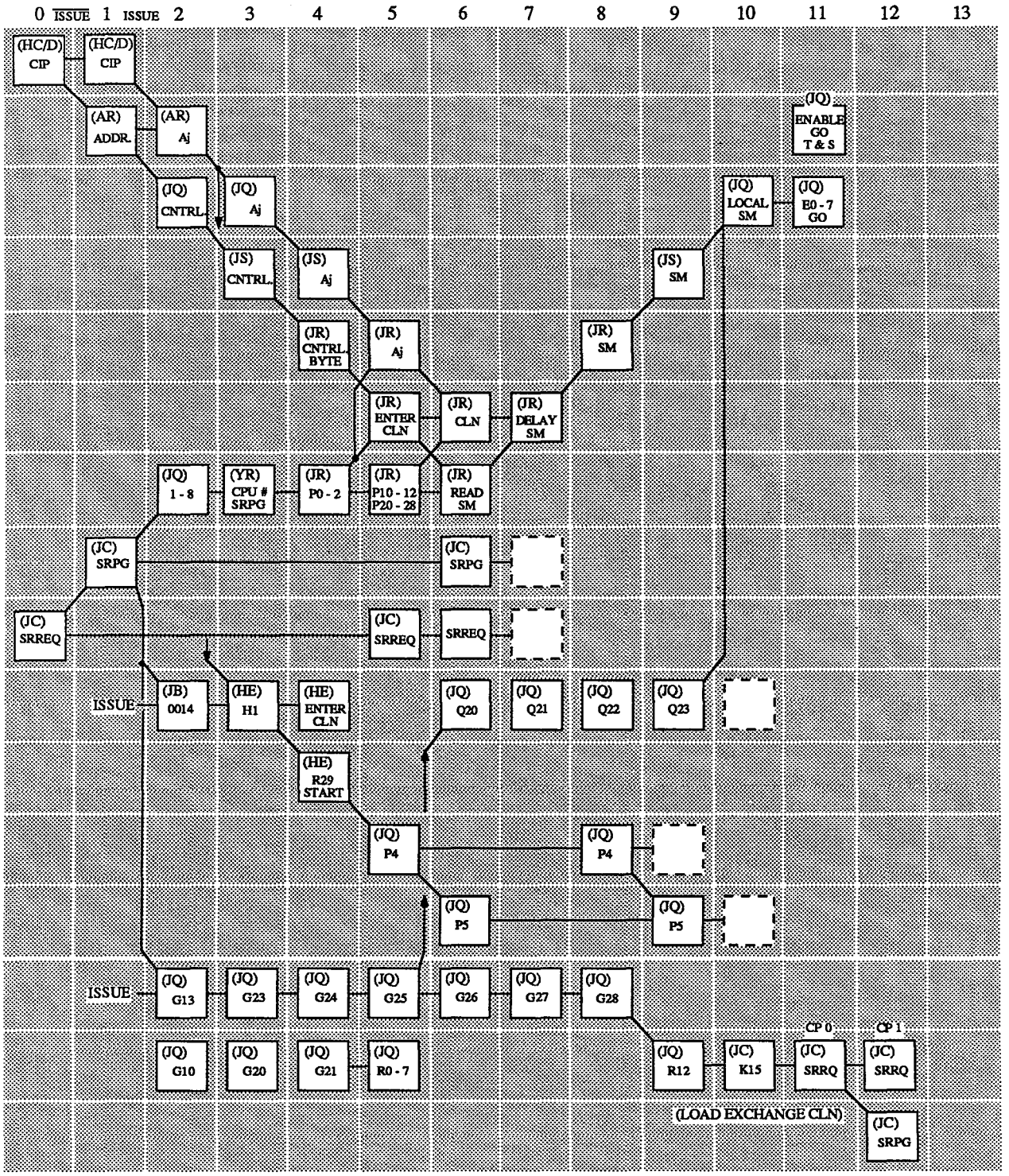
CRAY Y-MP SHARED REGISTER



026, 027/072, 073 INSTRUCTIONS

Hardware Trng.
A-6506B J.E.S.

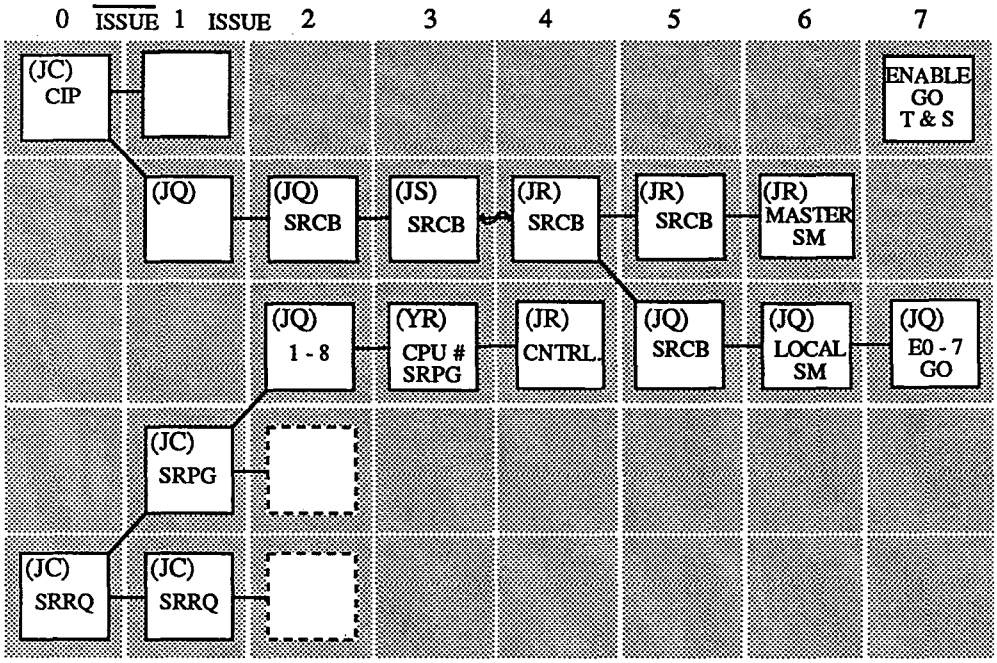
CRAY Y-MP SHARED REGISTER



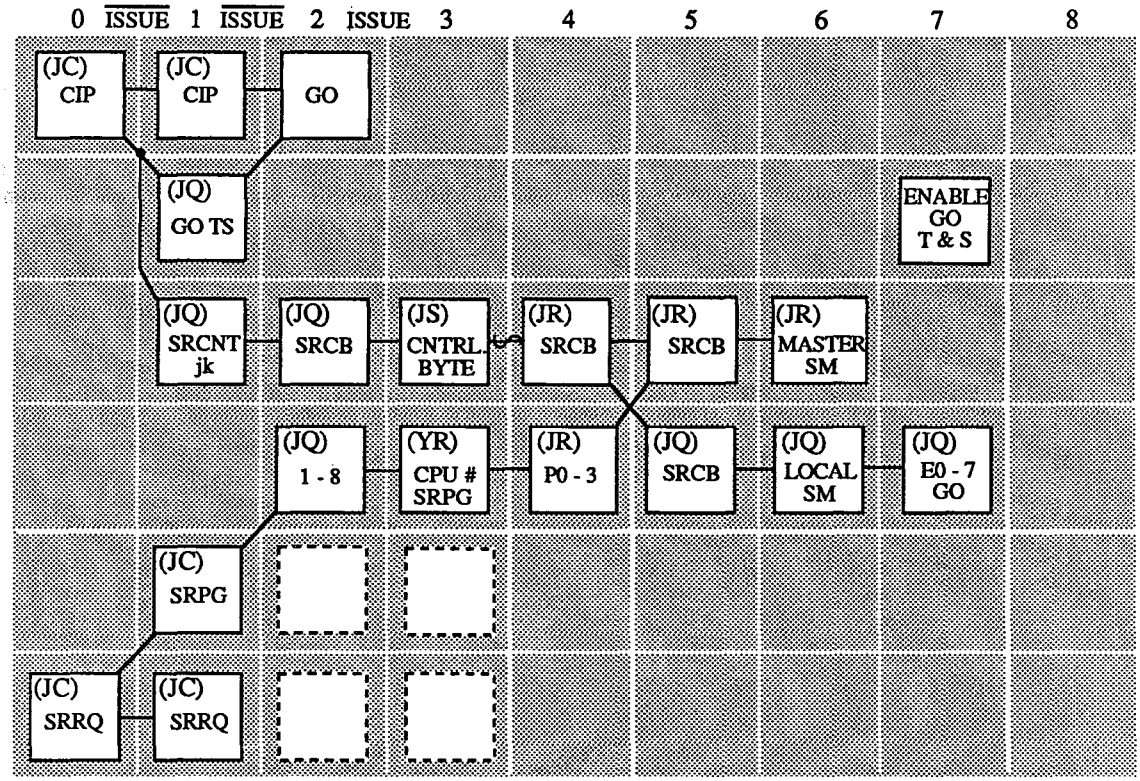
0014j3 LOAD CLN REGISTER WITH (Aj) WHERE $0 \leq (Aj) \leq 9$

Hardware Tmg. A-6508B J.E.S.

CRAY Y-MP SHARED REGISTER



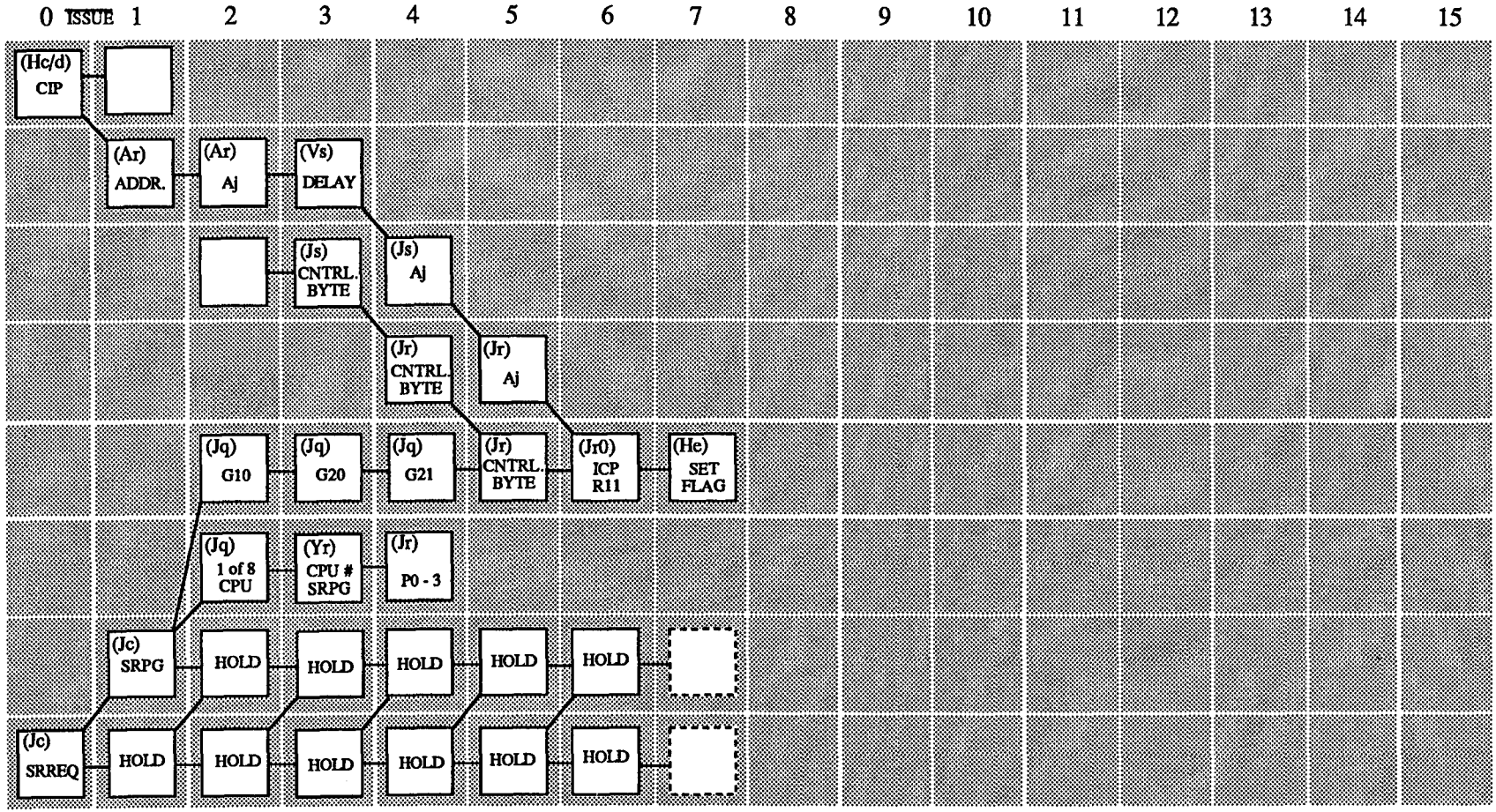
0036jk/0037jk SET OR CLEAR SEMAPHORE (jk) $0 = jk = 37_8$



0034jk TEST AND SET SEMAPHORE (jk) $0 \leq jk \leq 37_8$

Hardware Trng.
A-6507A J.E.S.

CRAY Y-MP SHARED REGISTERS



0014j1 - SET INTERPROCESSOR INTERRUPT REQUEST OF CPU (Aj)

CRAY Y-MP SHARED REGISTER ICP REQUEST

CRAY PROPRIETARY

CRAY Y-MP REAL-TIME/PROGRAMMABLE CLOCK INSTRUCTIONS

CODE	CAL		DESCRIPTION
++0014j0	RT	Sj	Enter RTC register with (Sj) (2^{63} - 2^{4})
072i00	Si	RT	Transmit (RTC) to Si
++0014j4	PCI	Sj	Enter Interrupt Interval (II) register with (Sj)
++001405	CCI		Clear PCI request
++001406	ECI		Enable PCI request
++001407	DCI		Disable PCI request

++ Priviledged to monitor mode.

Hardware Trng.
YM09/05A D.E.

CRAY Y-MP REAL-TIME CLOCK

The CRAY Y-MP contains one 64-bit real-time clock (RTC) per CPU. The RTC is located on four (JS) options, each handling 16 bits. (JS0) handles bits (2^0 - 2^7, 2^{32} - 2^{39}), (JS1) handles bits (2^8 - 2^{15}, 2^{40} - 2^{47}), (JS2) handles bits (2^{16} - 2^{23}, 2^{48} - 2^{55}), and (JS3) handles bits (2^{24} - 2^{31}, 2^{56} - 2^{63}).

The RTC increments once every clock period except when being written into or read from. This synchronization allows for clock period timing of program execution.

Writing to the RTC with the 0014j0 instruction will result in a copy of the Sj register from the CPU issuing the instruction being sent to all eight RTC registers. This is accomplished by using the Shared register data select (JR) option and Shared register (HS) option paths. Reading the RTC with a 072i00 instruction copies the RTC register (JS) options to the S register (VB, VC) options, within the CPU.

RTC instruction decode and control signals are generated and fanned out on the (JC) and (JQ) options.

CRAY Y-MP REAL-TIME CLOCK (OPTIONS INVOLVED)

(HS14 - HS15) Options

The (HS14 - 15) options found in each CPU contain the Shared SB/ST registers. The RTC utilizes the Shared register data paths to fanout the data from the (JS) option of the CPU issuing the 0014j0 instruction to all eight RTCs.

(JC0) Option

The (JC) option, one per CPU, handles Instruction Decode, Issue, and Hold Issue conditions associated with the real-time clock. The Boolean terms R27, I36 are the interprocessor communications on the (JC) option necessary to determine Shared register Hold Issue conditions. If the 0014j0 instruction issues, the (JC) option transfers instruction decodes to the RTC (JS) option as well as Shared register in progress (SRPG), and SR control signals to the (JQ) option.

(JQ0) Option

The (JQ0) option, one per CPU, generates a Shared register control byte (SRCB) bits 6, 7 specifically which are sent to the RTC (JS) option to send Sj to the Shared registers. The (JQ) option also does a 1 to 8 fanout of the Shared register in progress signal going to the (YR11) option in each CPU.

(JR0 - JR3) Options

The four (JR) options are located in each CPU, for a total of 32 options each handling 2 bits (n, n + 32). The Sj data passes through the Shared register data select (JR) options on its way to the eight CPUs via the (HS) options. The (JR) options also decode the appropriate SR control bytes to generate a write RT control signal to eight CPUs (JS) options, and a bypass signal on the Shared registers depending on the forced position constant on the (JR) option.

(JS0 - JS3) Options

Each (JS) option, located one per board on the CPU module, contains 16 bits of the 64-bit RTC register. The 16 bits are called RT lower and RT upper respectively. The value being loaded into the RTC may be from Sj for the 0014j0 instruction, Shared register (SR) data as a result of fanout from the 0014j0 instruction, or the RTC itself after incrementing its current value. If the value is from Sj and is being written into (JS0), the lower 4 bits are forced zero to establish a known starting point for all 0014j0 instructions.

When RT lower of (JS0) equals 377_8, a RT carry R32 term is generated and sent to RT lower of (JS1). When RT lower of (JS0) and (JS1) equals 177777_8 a RT carry is sent to (JS2), and so forth.

(VB, VC) Options

The (VB, VC) options, (16 each per CPU), contain the S registers. Two bits per option, (VB) handles bits 0 - 31 and the (VC) handles bits 32 - 64. The 0014j0 and 072i00 instructions use the S registers as operand and result respectively for the RTC.

(YR11) Option

The (YR11) option, (one per CPU), is used as a fanin of the SRPG control signal. The signal goes to the Shared register data select options (JR).

CRAY Y-MP PROGRAMMABLE CLOCK

The CRAY Y-MP programmable clock (PC) is a 32-bit counter, (one per CPU), which decrements once every clock period. This clock is located on four (JS) options each handling 8 bits.

The programmable clock may be loaded from Sj with the 0014j4 instruction, minimum count of 8_{10}. When the programmable clock equals zero, an interrupt request (PCI) is generated and will cause an interrupt if the enable PCI request (0014j6 instruction) has been previously executed. The interrupt request will remain set until cleared by executing a 0014j5 instruction. If the interrupt request is not needed, a 0014j7 instruction will disable the PCI request from causing an interrupt.

The PCI request is enabled and disabled on the (HE) option and the interrupt signal if generated is sent to the (HC) option which handles instruction pre-decode and Wait Exchange.

Control signals for the loading of the programmable clock come from the (JC) option.

PROGRAMMABLE CLOCK (OPTIONS INVOLVED)

(HE0) Option

The (HE) option handles the modes and flags for the CRAY Y-MP. If the PCI request generates an interrupt, a flag will be set in the Exchange Package of the current program. The (HE0) option also decodes the k field of the PCI instructions to disable, enable, or clear the PCI.

(JC0) Option

The (JC0) option handles decode and issue of the 0014j4 instruction. The control signals are then sent to the (JS) options to load the programmable clock with contents of Sj.

(JS0 - JS3) Options

The four (JS) options, one on each board of the CPU module, each handle 8 bits of the 32-bit programmable clock. When the lower most 8 bits of the clock equals zero, a programmable clock borrow is propagated to the next higher 8 bits of the clock. When the 32-bit clock equals zero, a programmable clock borrow is propagated from the upper most 8 bits of the clock. This borrow is the PCI request, as well as the reload of the programmable clock with a copy of the Interrupt Interval.

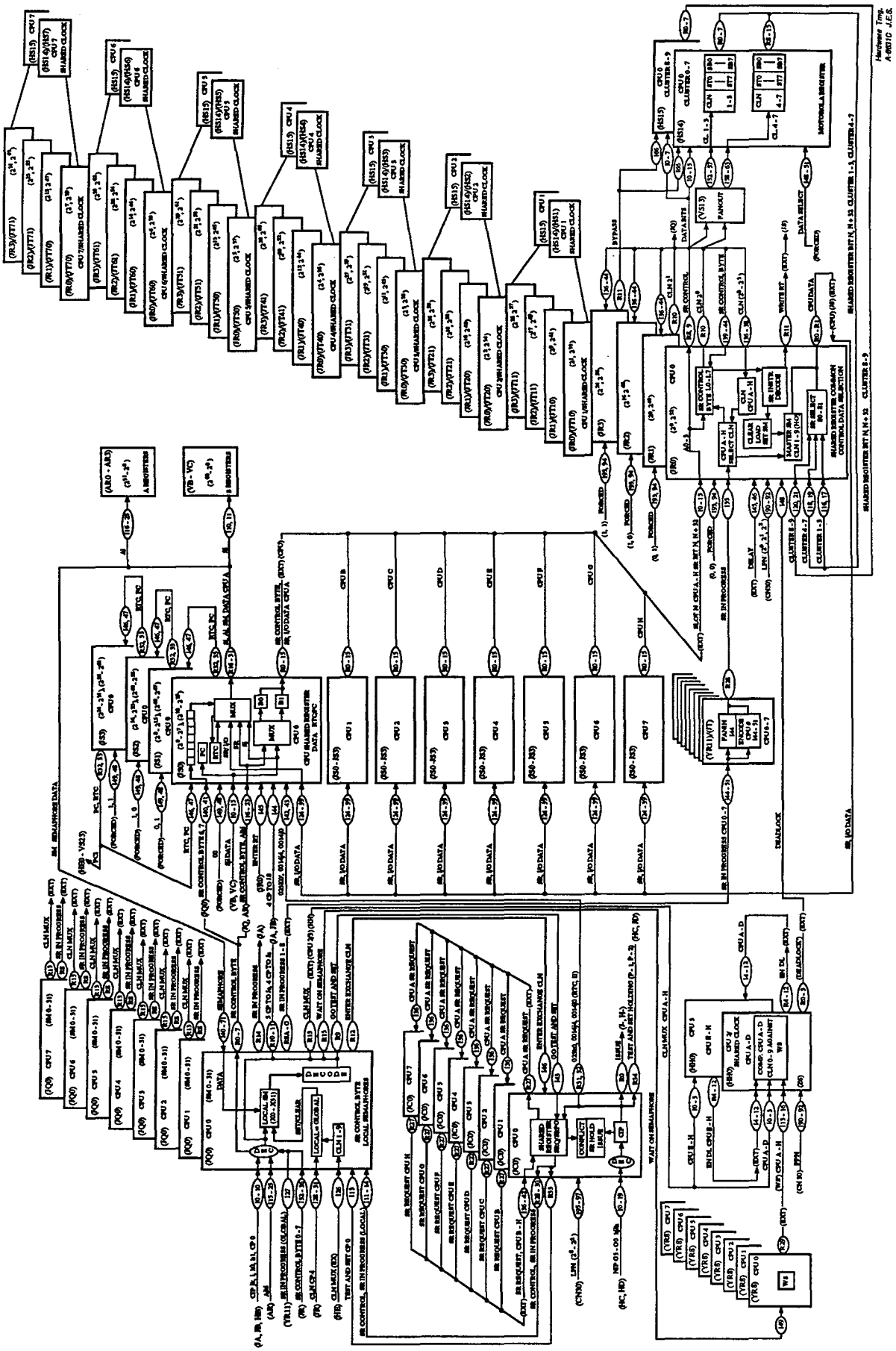
The Interrupt Interval is loaded from Sj with a 0014j4 instruction, into the programmable clock on the (JS) options. The interval may not be less than 8_{10}.

(VB0 - VB15) Options

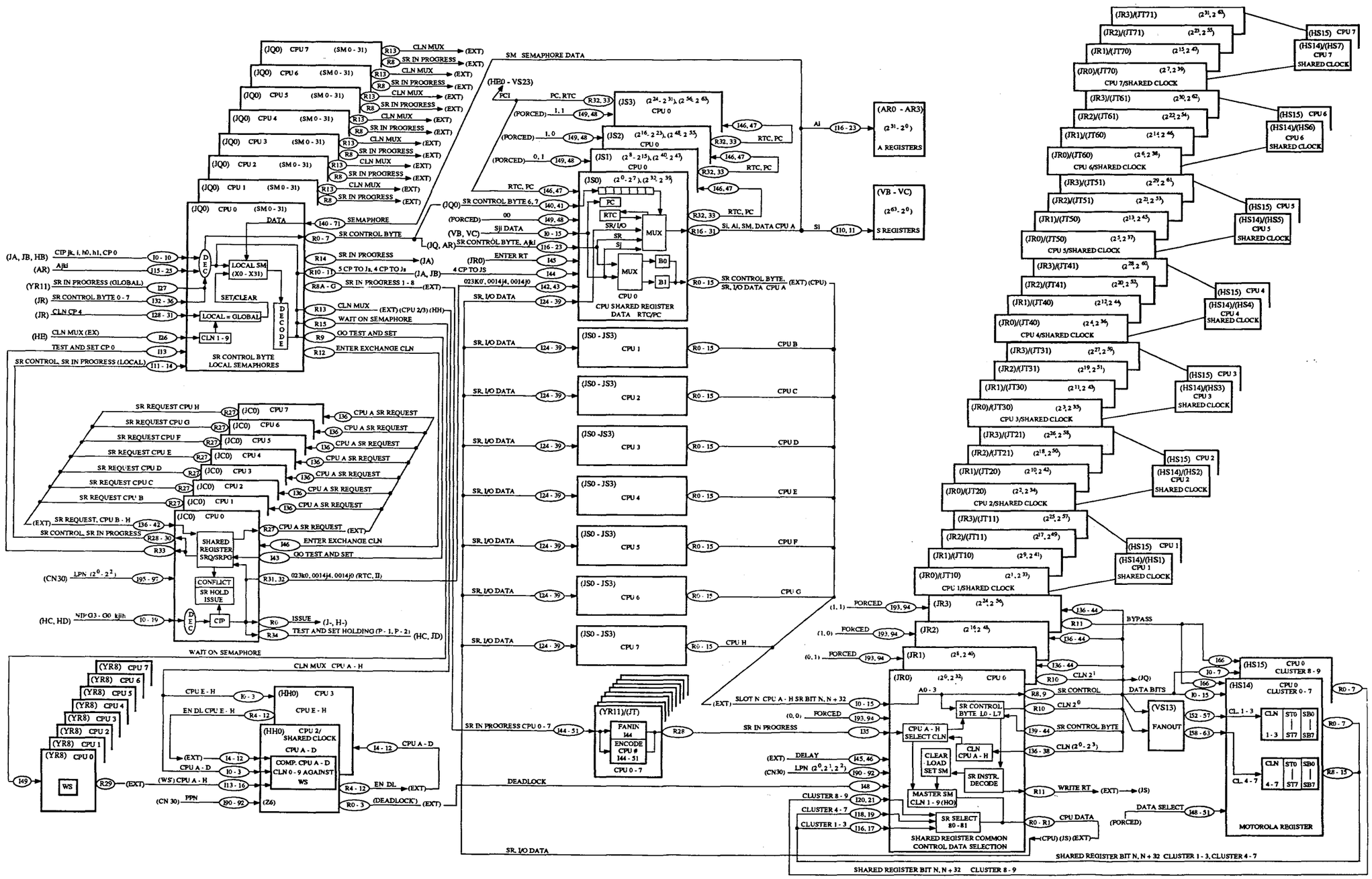
Each (VB) option handles 2 bits (n, n + 1) of the lower 32 bits of the S registers. The (VB) options provide the user specified interrupt interval for the 0014j4 instruction.

(VS23) Option

The (VS23) option is used as a delay of the PCI request.

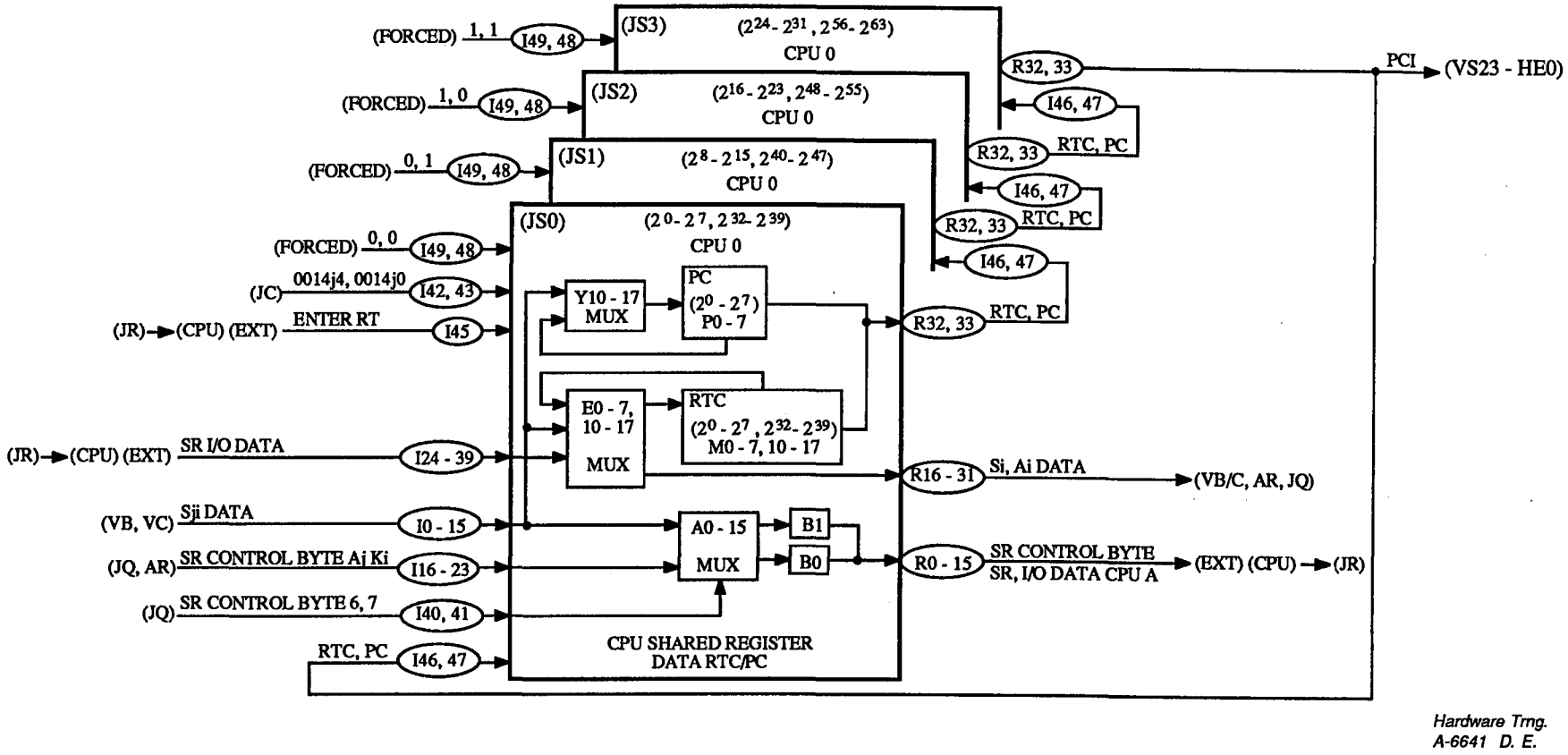


CRAY Y-MP RTC/PC BLOCK DIAGRAM



CRAY Y-MP RTC/PC BLOCK DIAGRAM

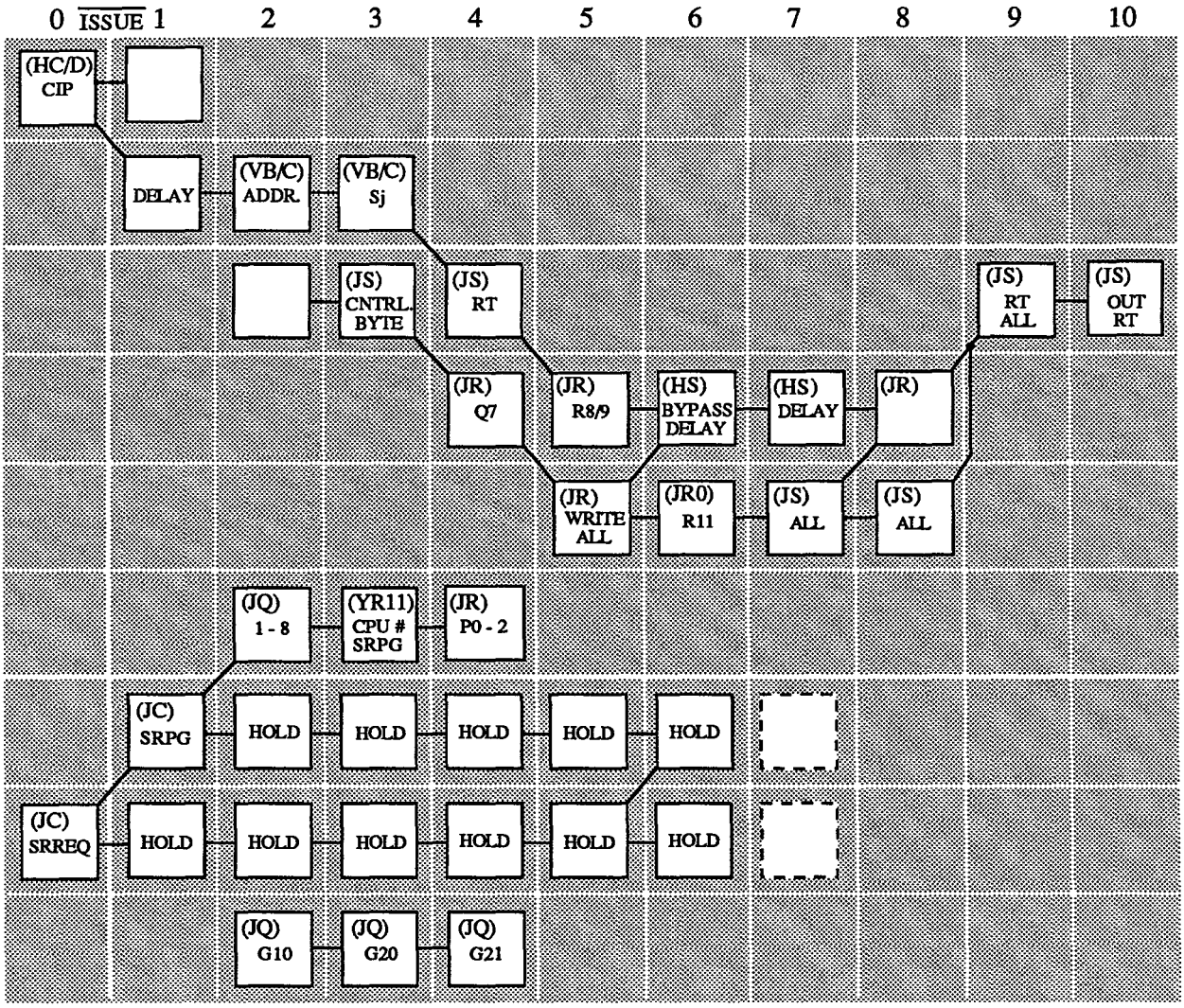
CRAY PROPRIETARY



Hardware Trng.
A-6641 D. E.

CRAY Y-MP (JS) BLOCK DIAGRAM RTC/PC

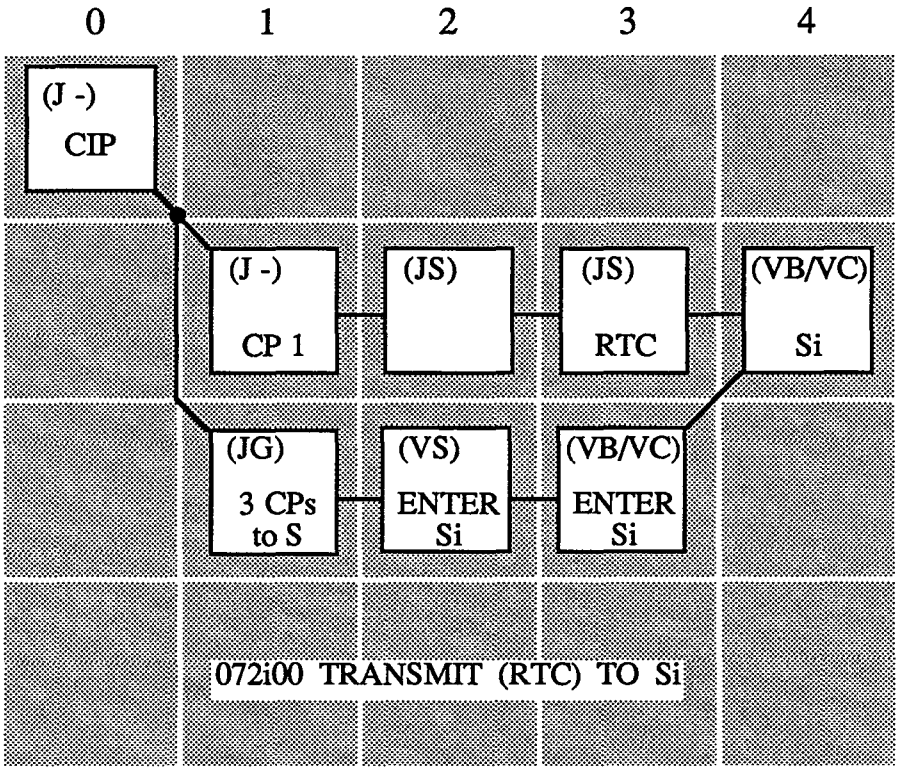
CRAY PROPRIETARY



0014j0 Enter RTC register with (Sj)

Hardware Trng.
A-6637B D.E.

CRAY Y-MP SHARED REGISTER RTC CONTROL



CRAY Y-MP REAL-TIME CLOCK

CRAY Y-MP PERFORMANCE MONITOR

The CRAY Y-MP Performance Monitor is contained on three options. The (HI0) handles the control portion of the logic, the (HJ0) contains the counters, and the (HS16) is used to store the accumulated counts.

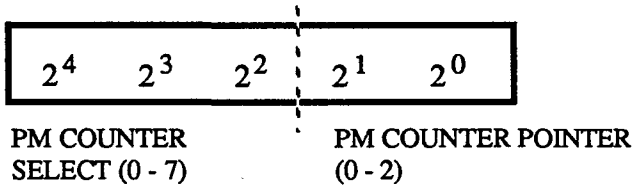
The (HI0) option controls the address and go write to the (HS16) option. The (HJ0) option contains the eight counters and will increment the counters whenever an Advance Counter (0 - 7) is sent from the (HI0) option. The count on the (HJ0) is the current count while an accumulated count is held on the (HS16). As the current count is logging events, the accumulated counts on the (HS16) are constantly being updated by reading the accumulated counts from the (HS16) to the (HJ0) option and updating the current count to it, then writing the updated accumulated count back to storage on the (HS16). The Performance Monitor counters are Enabled only when in user mode.

Read/Write Sequence

To read the accumulated counts from (HS16) to (HJ0) and write the updated value back to (HS16) requires 6 CPs. The Performance Monitor (PM) counts are 48 bits wide and are stored in 3 parcels of 16 bits each. A PM counter pointer will select the proper parcel. To update the accumulated count, a three reads followed by three writes sequence is performed. This sequence occurs through all 8 counters repeatedly, one at a time.

Addressing

The addressing which is generated on the (HI0) option, selects the proper counter on the (HJ0) and proper counter and parcel on the (HS16). The 5 bits of address are made up of the PM counter pointer, bits (2^0, 2^1) for the parcel on (HS16), and the PM counter select, bits (2^2, 2^3, 2^4) on (HS16) and (HJ0).



Hardware Trng.
YM13/22B D.E.

CRAY Y-MP PERFORMANCE MONITOR (OPTIONS INVOLVED)

(HI0) Option

The (HI0) option is the control for the operation of the Performance Monitor. As the various hardware events occur, the (HI0) classifies them, and the group selected by the 0015j0 instruction will generate an Advance Counter signal which goes to the (HJ0) option to increment the counter. The (HI0) also generates the Performance Monitor addressing, which selects the proper counter on the (HJ0) option and provides addressing for the storage option (HS16). The addressing sequence also generates a Go Write to the (HS16) which writes the updated accumulated counts back into storage.

(HJ0) Option

The (HJ0) option contains the eight Performance Counters which are held on the G terms. Only the lower 7 bits of the Performance Counters 0 - 5 and the lower 9 bits of counters 6 and 7 are held on the (HJ) for incrementing. The total 48 bits are stored on the (HS16) option. An occurance of a selected hardware event will increment the (HJ) counters, by the Advance Counter sent from (HI0).

On a read of the Performance Monitor 073i11, the data or count is brought from the (HS16) to the (HJ0) for final summing before being sent to an S register.

(HS16) Option

The (HS16) option is used for storing the Performance Counter's accumulated counts. The Performance Counter data comes from the (HJ) and is stored at the 5-bit address sent from the (HI) R10 - 14. The upper 3 bits of the address (also sent to the (HJ)) specify the counter number, and the lower 2 bits specify which 16-bit parcel, 0, 1, or 2. A read/write sequence controlled by (HI0) constantly updates the accumulated counts on the (HS16) by reading each parcel to the (HJ0), adding the current count to it, and writing it back to the (HS16).

CRAY Y-MP PERFORMANCE MONITOR AND MAINTENANCE MODE INSTRUCTIONS

	Code	Description
++	0015j0	Select Performance Monitor
++	0015j1	Set Maintenance Mode
	073i01	Transmit Status Register Bits to Si
++	073i11	Read Performance Counter to Si
++	073021	Increment Performance Counter (upper)
++	073031	Clear All Maintenance Modes
++	073061	Increment Performance Counter (lower)

++ Priviledged to monitor mode

Hardware Trng.
YM13/24A D.E.

Instructions

The Performance Monitor has two types of instructions associated with its use. They are the user and maintenance instructions. In this case user does not refer to non-monitor but rather to the user of the Performance Monitor. The user instructions here are run in monitor mode. The user instructions select the Performance Monitor (0015j0) and read the Performance Monitor (073i11).

The 0015j0 instruction will select one of four groups of hardware events to be monitored while in user (non-monitor) mode. The event listing table can be found on the group selects chart later in this section. If selected, the Performance Counters will increment events as shown on the table.

On a read of the Performance Monitor, one 073i11 instruction will read 16 bits (2^{15} - 2^{0}) of Performance Counter 0 into bit locations 2^{47} - 2^{32} of Si. Following a 4 CP delay, another 073i11 may be issued, which would read 16 bits (2^{31} - 2^{16}) of Performance Counter 0 into 2^{47} - 2^{32} of Si. The Performance Counters are 48 bits wide, and therefore would require 3 reads to be completely read to the Scalar registers (Si). The sequence for reading the Performance Counters would be:

(Monitor Mode)

073i11 Performance Counter 0 bits 2^{15} - 2^{0} ---------- Si 2^{47} - 2^{32}
(4 CP delay)
073i11 Performance Counter 0 bits 2^{31} - 2^{16} ---------- Si 2^{47} - 2^{32}
(4 CP delay)
073i11 Performance Counter 0 bits 2^{47} - 2^{32} ---------- Si 2^{47} - 2^{32}
(4 CP delay)
073i11 Performance Counter 1 bits 2^{15} - 2^{0} ---------- Si 2^{47} - 2^{32}

.
.
.

The maintenance instructions are used to test the validity of the Performance Monitor. To set up for testing of the Performance Monitor, the user instructions 0015j0 and 073i11 are issued in monitor mode to clear all Performance Monitor counters, and address the counter to be tested. Three 073i11 instructions per counter are required. A 48 CP delay is required between a 0015j0 instruction and any maintenance instructions. The maintenance instruction 073i61 will advance bit 2^{0} of the counter under test, and within 9 CPs the 073i21 instruction must be issued, which will advance bits 2^{22} and 2^{38} of the counter under test. A 073i31 will clear the address, and the counter can be read out to an S register with a 073i11 instruction.

GROUP	PERFORMANCE COUNTER	MONITORED EVENT	INCREMENT PER CLOCK PERIOD
0		NUMBER OF:	
	0	INSTRUCTIONS ISSUED	+1
	1	CLOCK PERIODS HOLDING ISSUE	+1
	2	FETCHES	+1
	3	FLOATING ADD OPERATIONS	+1
	4	FLOATING MULTIPLY OPERATIONS	+1
	5	FLOATING RECIPROCAL OPERATIONS	+1
	6	CPU MEMORY REFERENCES	+3
	7	I/O MEMORY REFERENCES	+1
1		HOLD ISSUE ON:	
	0	SEMAPHORES	+1
	1	SHARED REGISTERS	+1
	2	A REGISTERS	+1
	3	S REGISTERS	+1
	4	V REGISTERS	+1
	5	VECTOR FUNCTIONAL UNIT	+1
	6	SCALAR MEMORY REFERENCES	+1
	7	BLOCK MEMORY REFERENCES	+1
2		(MEMORY) NUMBER OF:	
	0	FETCHES	+1
	1	FETCH CONFLICTS	+1
	2	I/O REFERENCES	+1
	3	I/O CONFLICTS	+1
	4	SCALAR REFERENCES	+1
	5	PORT C REFERENCES (WRITE)	+1
	6	CPU REFERENCES	+3
	7	CPU CONFLICTS	+3
3		NUMBER OF (XXX - XXX) INSTR.:	
	0	000 - 017	+1
	1	020 - 077	+1
	2	100 - 137	+1
	3	140 - 157, 175	+1
	4	160 - 174	+1
	5	176 - 177	+1
	6	VECTOR INTEGER OPERATIONS	+3
	7	VECTOR FLOATING-POINT OPERATIONS	+3

Hardware Trng.
A-6831 D.E.

CRAY Y-MP PERFORMANCE MONITOR GROUP SELECTS

The 0015j1 instruction (where j may equal 0 - 5) is used to enter different maintenance modes. The maintenance mode switch located on the CRAY Y-MP mainframe maintenance panel must be switched on during execution of the 0015j1 instruction. Failure to do so results in a no-op. The maintenance mode, selected by executing the 0015 instruction, will remain set until a 073031 issues or the maintenance mode switch is switched off. Allow 10 CPs for maintenance modes to become effective. Also note that switching on the maintenance mode disables normal error correction. Instruction 001501 disables Port A error correction. Instruction 001511 disables Port B error correction. Instruction 001521 disables Port D error correction. The 001531 instruction will route (036 Memory to Treg) read data through Port D error correction instead of Port B. Instruction 001541 replaces specific data bits with check byte bits and vice versa for memory reads and writes. The bit pattern shown below will replace data bits with corresponding check byte bits on reads using Ports A, B, and D. The same pattern applies and replaces check byte bits with data bits on writes using Ports C and D.

Data bit	Check byte bit
0	64
8	65
16	66
24	67
32	68
40	69
48	70
56	71

Instruction 001551 replaces specific vector data bits with check byte bits during a write of memory (Port C) for the 1771jk instruction. The specific bit pattern is shown below.

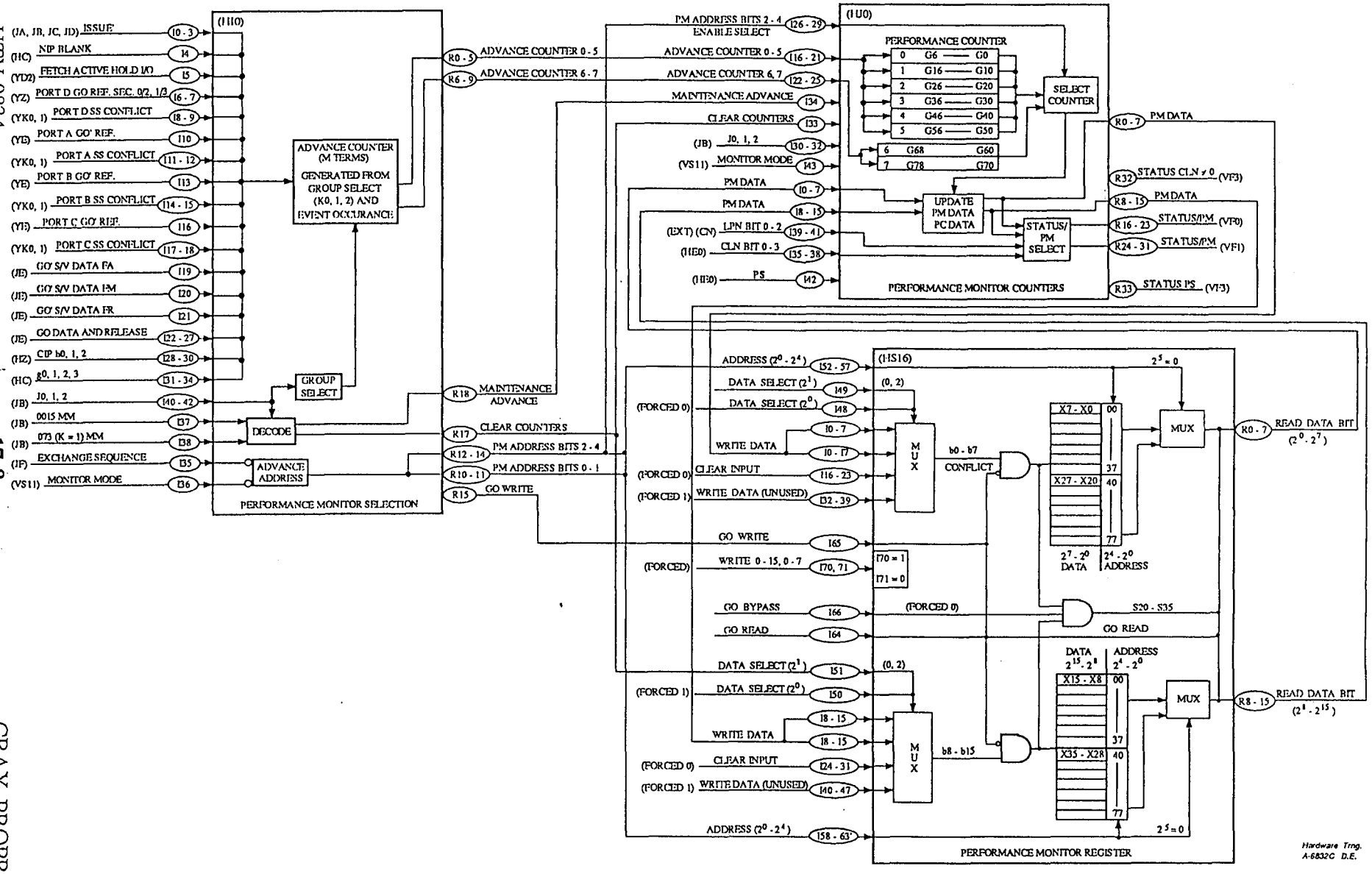
Vk Data bit	Check byte bit
0	64
1	65
2	66
3	67
4	68
5	69
6	70
7	71

The 073i01 instruction transmits the status of the following flags, modes, or fields from the Exchange Package to the upper 32 bits of Si, while setting the lower 32 bits to all ones.

Si bit

2^{63}	CLN $\neq 0$
2^{57}	Program State (PS)
2^{54}	Register Parity Error (RPE)
2^{53}	Uncorrectable Memory Error (UME)
2^{52}	Correctable Memory Error (CME)
2^{51}	Floating-point Error (FPS)
2^{50}	Floating-point Interrupt Enabled (IFP)
2^{49}	Operand Range Interrupt Enabled (IOR)
2^{48}	Bidirectional Memory Enable (BDM)
2^{42}	Processor Number bit 2^2 (PN)
2^{41}	Processor Number bit 2^1 (PN)
2^{40}	Processor Number bit 2^0 (PN)
2^{35}	Cluster Number bit 2^3 (CLN)
2^{34}	Cluster Number bit 2^2 (CLN)
2^{33}	Cluster Number bit 2^1 (CLN)
2^{32}	Cluster Number bit 2^0 (CLN)

A more detailed description of each of these flags, modes, or fields is located in the Exchange section of the CRAY Y-MP Hardware Training Volume (HTV-0834).



CRAY Y-MP PERFORMANCE MONITOR BLOCK DIAGRAM

CRAY PROPRIETARY

CRAY Y-MP ADDRESS REGISTERS

The CRAY Y-MP contains eight Address registers, A0-A7, each 32 bits in length. These registers are used primarily to contain addresses for memory references and as index registers for block transfers and loop control. They also are used to contain shift counts and information about I/O channels, as well as to receive data from memory, S registers, B registers, Shared B registers, Vector Length and Exchange Address registers, from four functional units (Scalar Pop, LZ, Address Add and Multiply), and the Constant data from the jk or nm fields of the instruction. There are two functional units serving the Address registers exclusively: the Address Add and the Address Multiply. When the computer is running in Y-mode, all 32 bits of these registers are utilized, but when it is running in X-mode, only the lower 24 bits are used and the upper eight are zeroed out.

Data enters these registers after passing through one or two mux's, which are controlled by the Address functional unit (AFU) source code. The 3 bits making up this source code are used to select data from one of seven incoming data paths. Then the 3 bits of the destination i field are used to generate the appropriate enter signal: Enter A0, Enter A1, etc. to gate the data into the proper A register.

The Bypass data path directly into the Readout registers will be utilized in addition to the normal path of data entering the A registers in the case where, for example, in a two instruction sequence, the j or k field of the second instruction is the same as the i field of the first instruction. In this case, the incoming data also flows through an additonal mux,which again uses the AFU source code to select the proper path along which to send the data directly to one of the two Readout registers. Each Readout register has a separate mux and Bypass path to steer data into it. Which of the two Readout registers is used is instruction dependent and is determined at the time of issue.

To read data out of an A register, a jk/i or jkh/i designator is sent from the issue control option to gate data out of the appropriate A register and into one of the Readout registers. Which of the two Readout registers is used is instruction dependent and determined at the time of issue. If A0 is specified as an operand in the j, k, or h field of an instruction, the actual A0 data is replaced by either zero or one as it enters the Readout register. If A0 is in the k field, the data is replaced by one, and if it is in the j or h field, the data is zeroed out. A0 data is not replaced by a zero or one when A0 is used on a B, T, or V Memory reference, or on a Branch instruction where the contents of A0 are tested for zero, non-zero, positive, or negative.

Bypass is disabled in the three special cases just mentioned. However, if A0 is reserved as a destination register and an instruction requiring A0 in the j, k, or h field occurs, it will not hold issue.

CRAY Y-MP ADDRESS REGISTERS (OPTIONS INVOLVED)

(AR0 - AR3) Options

The (AR) options each contain 8 bits of the Address registers A0-A7, with (AR0) containing bits (2^0 - 2^7), etc. Most of the Address Add functional unit is also contained on these options, except for the final section carry generation, which is done on the (YJ0) option. After each (AR) has computed the section carry and enable for its 8-bit group, it sends them to the (YJ) for the computation of the final section carries, which are then sent back to each (AR).

(JA0) Option

The (JA) option is the Address Register Issue control for the Address registers. The (JA) option decodes the instruction that was sent from the NIP register from the (HC, HD) options. If the (JA) recognizes the instruction as one of its own, the (JA) will check for conflicts on the instruction before allowing the instruction to issue at CIP time. If the (JA) does not recognize the instruction, or there are no conflicts, the (JA) will send out a Go Issue to the other (J-, H-) options and advance the instruction into execution. Types of conflicts that the (JA) checks for are:

- Access conflict - When two or more results will be arriving at the A register within the same clock period. Access conflict is determined by figuring the time it takes to issue an instructiuon and get results, compared to the previously issued instruction time it takes to get results.
- Ah, Ai, Aj, Ak - Register conflict occurs when a reserved register is requested by another instruction, causing a Hold Issue until the reservation is cleared.
- 005 Hold 2 CP - 005 instruction in conflict with 007/024 instruction
- 0010-0011 - Will unconditionally Hold Issue after a 0010-0011 instruction.
- A0, B conflict - Occurs when A0/B is busy and a A0/B is requested.

The (JA) also controls the Read/Write of the B registers by supplying the address bit (2^5 - 2^0) and the Go Read or Go Write to the (HS12, HS13) options. The operand select (ji/k), (k/ji), (jkh/0), and (k/jh0) are fanout to the four (AR) options on the (JA) option.

(JE0) Option

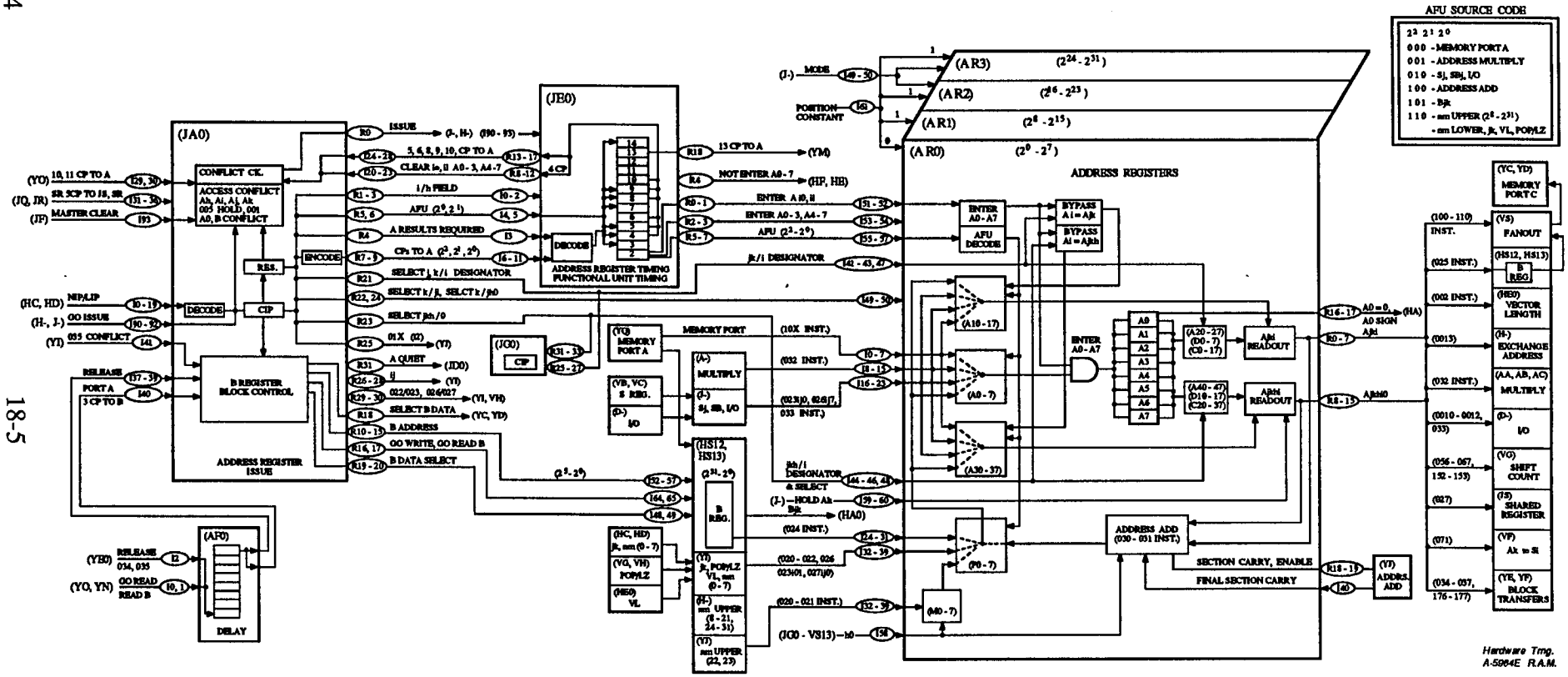
The (JE) responsibility is to supply the (JA) with the access times that the A registers are being written to, in order to determine if any previously issued instruction will conflict with a current instruction in CIP. The reason for this is because of the variable time it takes to read memory. Once the (JE) has decremented the (Ai) result and AFU bits to 2 CPs, the (Ai) selects and AFU are passed to the (AR) option to select the correct AFU path into the A register, and select the correct A register A0-A7. The AFU and (Ai) select codes are fanout to the four (AR) options by the (JE) option. The (JE) option is used for Vector functional unit timing, which is discussed in the Vector register unit.

CRAY Y-MP ADDRESS OPTIONS PER MODULE

OPTION TYPE	NUMBER USED	DESCRIPTION
(AA)	1	Address Multiply First level - Lower bits
(AB)	3	Address Multiply First Level - Middle bits
(AC)	1	Address Multiply First Level - Upper bits
(AD)	1	Address Multiply Final - Lower bits
(AF)	1	Address Multiply Final - Upper bits
(AR)	4	Address Register/ Address Add

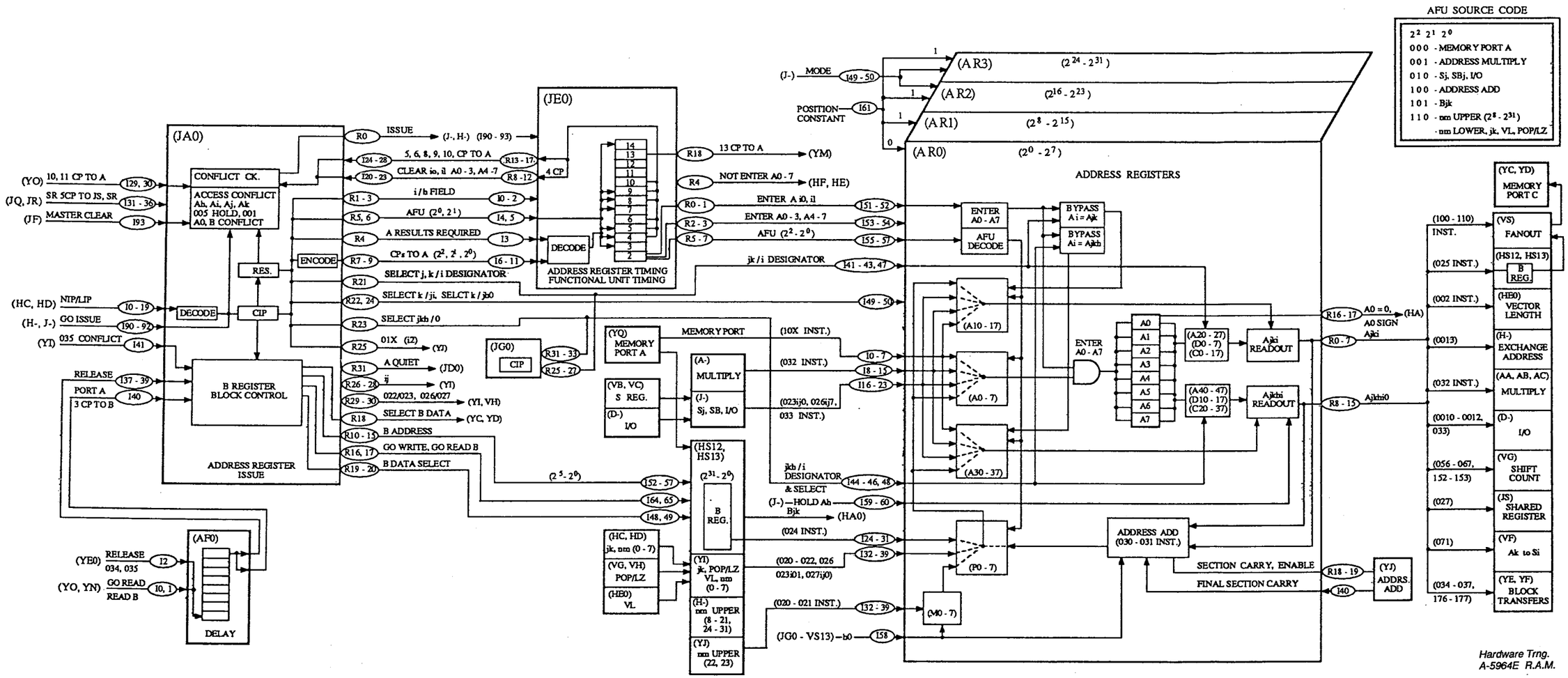
11 Total Options

Hardware Trng.
YM08/12 J.E.S.



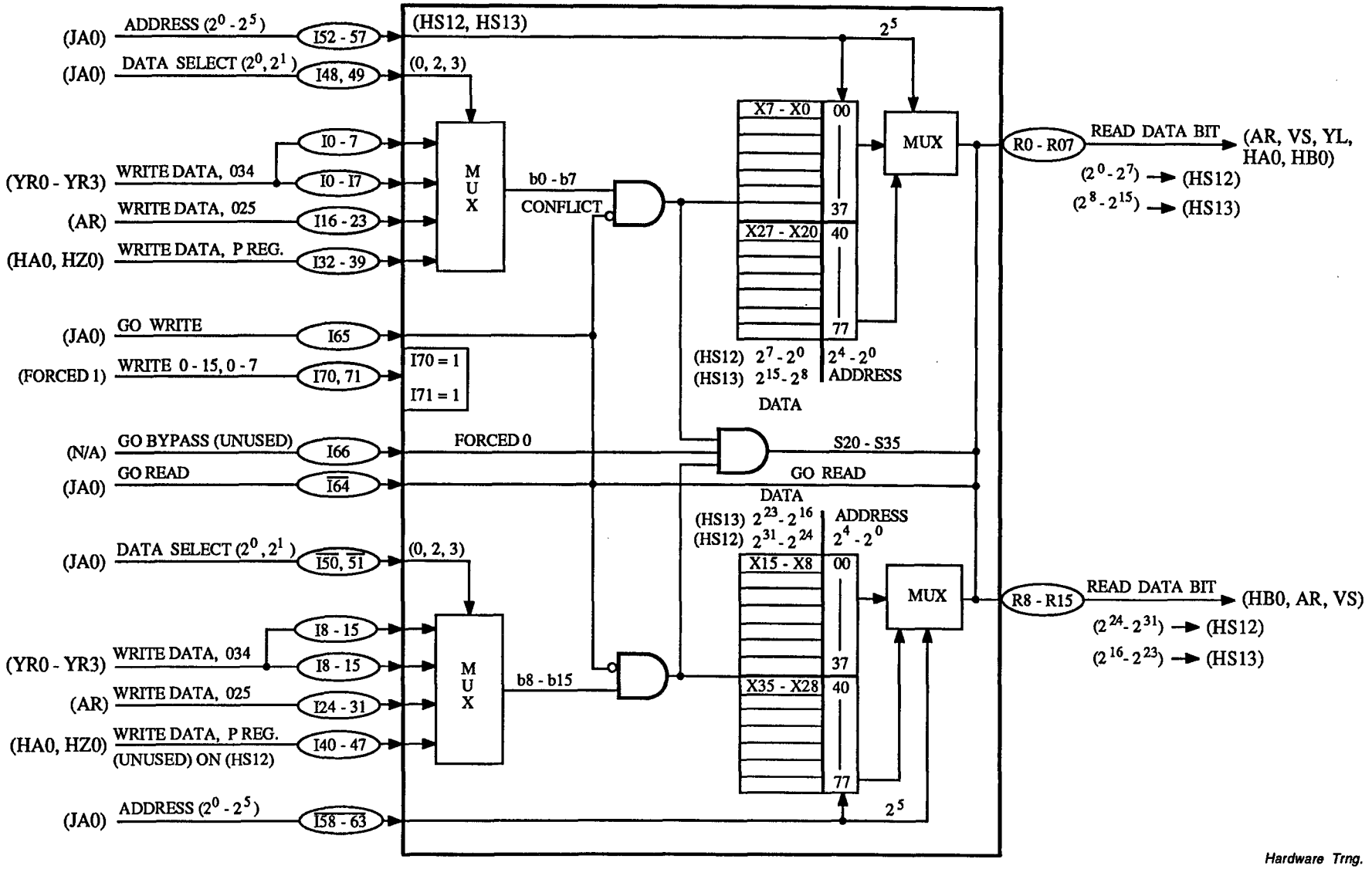
CRAY Y-MP ADDRESS REGISTERS BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP ADDRESS REGISTERS BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP (HS) BLOCK DIAGRAM
B REGISTERS

CRAY PROPRIETARY

Bjk	(Hs12) 2^{31} - 2^{24}	(Hs13) 2^{23} - 2^{16}	(Hs13) 2^{15} - 2^{8}	(Hs12) 2^{7} - 2^{0}
00	X15 - X8	X15 - X8	X7 - X0	X7 - X0
01				
02				
03				
04				
05				
06				
07				
10				
11				
12				
13				
14				
15				
16				
17				
20				
21				
22				
23				
24				
25				
26				
27				
30				
31				
32				
33				
34				
35				
36				
37	X15 - X8	X15 - X8	X7 - X0	X7 - X0
40	X35 - X28	X35 - X28	X27 - X20	X27 - X20
41				
42				
43				
44				
45				
46				
47				
50				
51				
52				
53				
54				
55				
56				
57				
60				
61				
62				
63				
64				
65				
66				
67				
70				
71				
72				
73				
74				
75				
76				
77	X35 - X28	X35 - X28	X27 - X20	X27 - X20

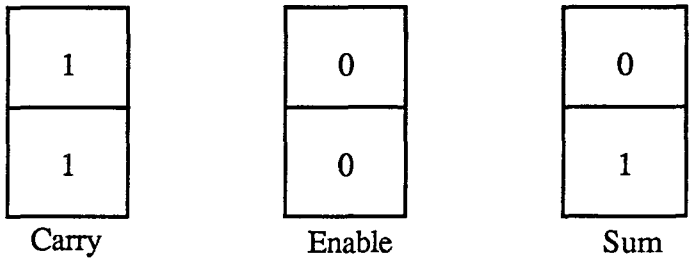
Hardware Trng.
A-6685 J.E.S.

CRAY Y-MP B REGISTERS BIT ASSOCIATION
(034ijk, 035ijk)

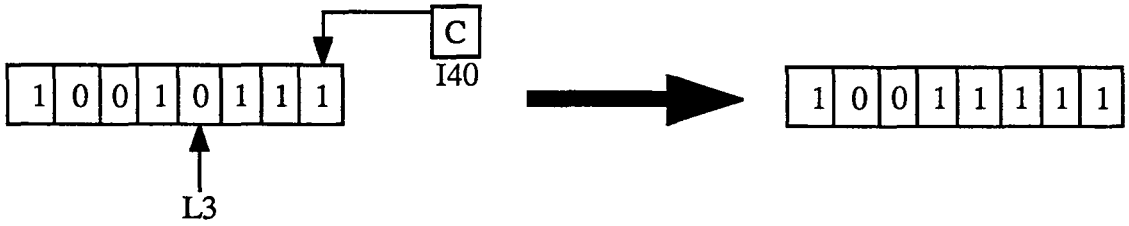
ADDRESS ADD FUNCTIONAL UNIT

The Address Add functional unit is used to combine together two 24-bit operands (X Mode) or two 32-bit operands (Y Mode) through either addition or subtraction. The two instructions utilizing this functional unit are the 030ijk for addition and the 031ijk for subtraction, the distinguishing characteristic being h bit 2^0 , or h0 as it is referred to in the Boolean. For either operation, it takes 3 CPs from the time the j and k operands leave the A registers until the result enters the appropriate Ai register.

The add operation starts by comparing the j and k operands bit by bit to determine whether each bit position contains a carry (both bits are 1), or a sum (the bits are different).



Then a set of enable and carry terms are generated to determine how far the carries will propagate, first within each 4-bit group and then within each 8-bit section. Next, partial sums, accurate for each separate 8-bit section, are formed from the bit sums and carry terms. After determining whether there will be any intersection carries (carries passed from one 8-bit section to another), the final sum is generated from the partial sums, intersection carries, and internal group enables. The internal group enables determine exactly to which bit position an entering carry will be propagated within each 8-bit section. For example, if a carry entered the 8-bit section shown below, it would be propagated to the bit position indicated by the internal group enable term L3, giving the result shown on the right:



Subtraction is performed by adding the twos complement of the k operand. This is done by using the ones complement of the k operand in determining the initial bit sums and carries, and then adding the extra one by forcing a carry into bit position 2^0 of the final summation.

Hardware Trng.
YM09/01A R.A.M.

CRAY Y-MP ADDRESS ADD FUNCTIONAL UNIT INSTRUCTIONS

Address Add/Subtract

030ijk - Integer sum of Aj and Ak to Ai

030i0k - Transmit Ak to Ai

030i00 - Transmit one to Ai

030ij0 - Integer sum of Aj and one to Ai

031ijk - Integer difference Aj minus Ak to Ai

031i00 - Transmit minus one to Ai

031i0k - Transmit the negative of Ak to Ai

031ij0 - Integer difference of Aj minus one to Ai

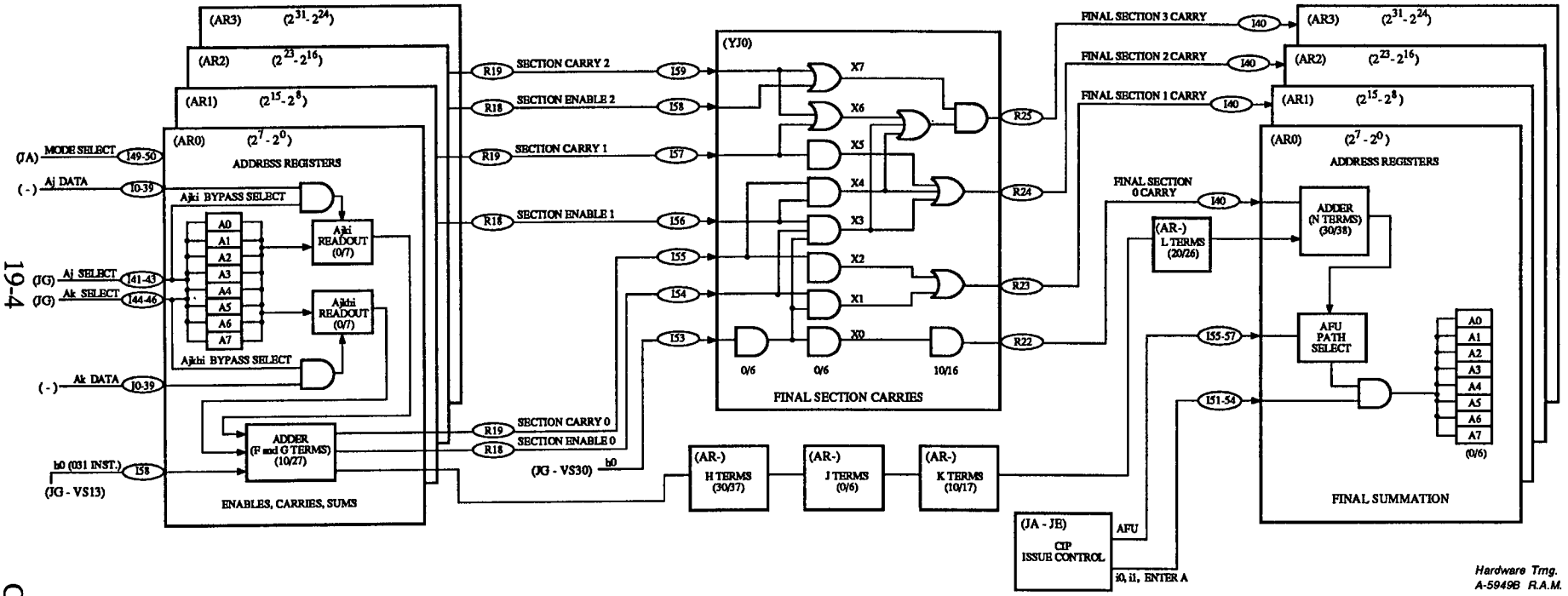
ADDRESS ADD FUNCTIONAL UNIT (OPTIONS INVOLVED)

(AR0 - AR3) Options

The (AR) options each handle 8 bits of the j and k operands, with (AR0) handling bits 2^0 - 2^7, (AR1) handling bits 2^8 - 2^{15}, and so on. Each (AR) establishes the bit positions within its 8-bit section as carries, or sums, and uses them to generate the section enables and carries, which are then sent to the (YJ0) option for processing. Partial sums of each 8-bit section and the internal group enables are also generated on the (ARs), and these are combined with the final section carries received from the (YJ0) to obtain the final summation. The result is then entered into the appropriate A register, also located on the (AR) options.

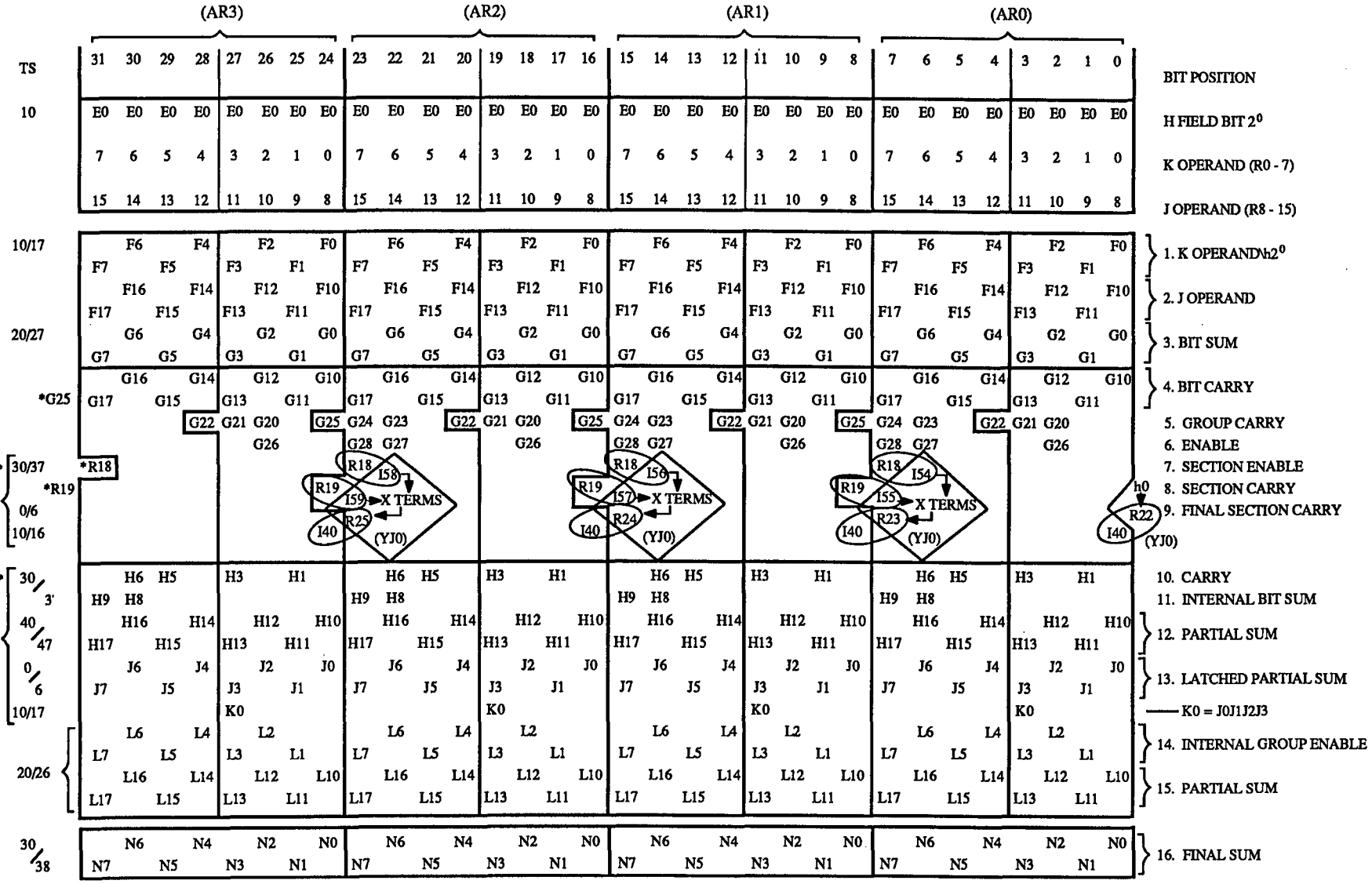
(YJ0) Option

The (YJ0) option combines the separate section enables and carries received from (AR0 - 3) to generate the final section carries. These indicate whether or not there will be a carry out of a particular 8-bit section as part of the final 32-bit result. These final section carries are then sent back to the appropriate (AR) option. In the case of a subtraction, the (YJ0) sends a final section carry to (AR0), causing a one to be added into bit position 2^0 to complete the twos complement arithmetic.



CRAY Y-MP ADDRESS ADD BLOCK DIAGRAM

CRAY PROPRIETARY



Hardware Trng.
A-6833A D.E.

CRAY Y-MP ADDRESS ADD FLOW DIAGRAM (Sheet 1 of 2)

CRAY PROPRIETARY

CRAY Y-MP ADDRESS ADD FLOW CHART - Explanation of terms (Sheet 2 of 2)

1. K operand\h$2^0$ - K operand is exclusively ORed with bit 2^0 of the h field of the instruction.

2. J operand - R8 - 15 make F10 - 17.

3. Bit Sum - The J and K operands are different.

4. Bit Carry - Both J and K operands are one.

5. Group Carry - G20 and G21 make G22 carry to bit position 4.
   G23 and G24 make G25 carry to bit position 8.

6. Enable - There are no bit enables from the previous lower enable G term to this one, therefore, a carry would propagate out of this bit position.

7. Section Enable - There are no bit enables for this section, an entering carry would propagate to the next section.

8. Section Carry - Carry out of this section.

9. Final Section Carry - There will be a carry out of the 8-bit section for the final sum.

10. Carry - There will be a carry out of the bit position indicated.

11. Internal Bit Sums - There are bit sums below the bit position indicated for this 4-bit group, therefore, an entering carry will propagate out to the bit position indicated.

12, 13, 15. Partial Sum - Sum of 8-bit section.

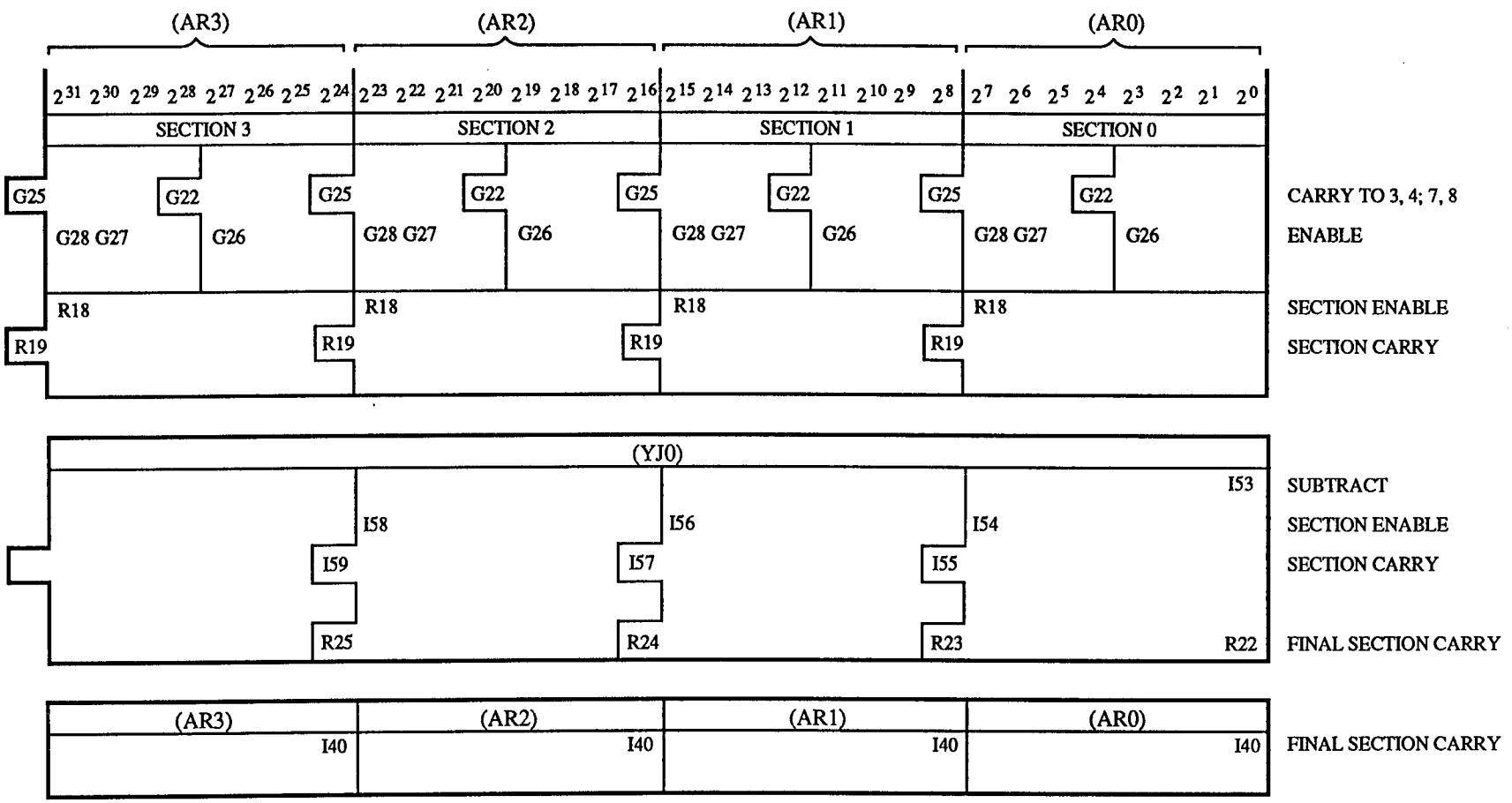
14. Group Enables - All sum bits below this bit position are set, therefore, an entering carry will propagate to this bit position.

16. Final Sum - Partial Sum and I40 (carry from lower section).

* These terms are made on (AR3) but are unused.

** These terms in bracket A are made concurrent with terms in bracket B.

Hardware Trng.
YM13/25 D.E.



(AR0) I40 term is made from a Subtract.

(AR1) I40 term is made from a Subtract and a Section 0 Enable or a Section 0 Carry.

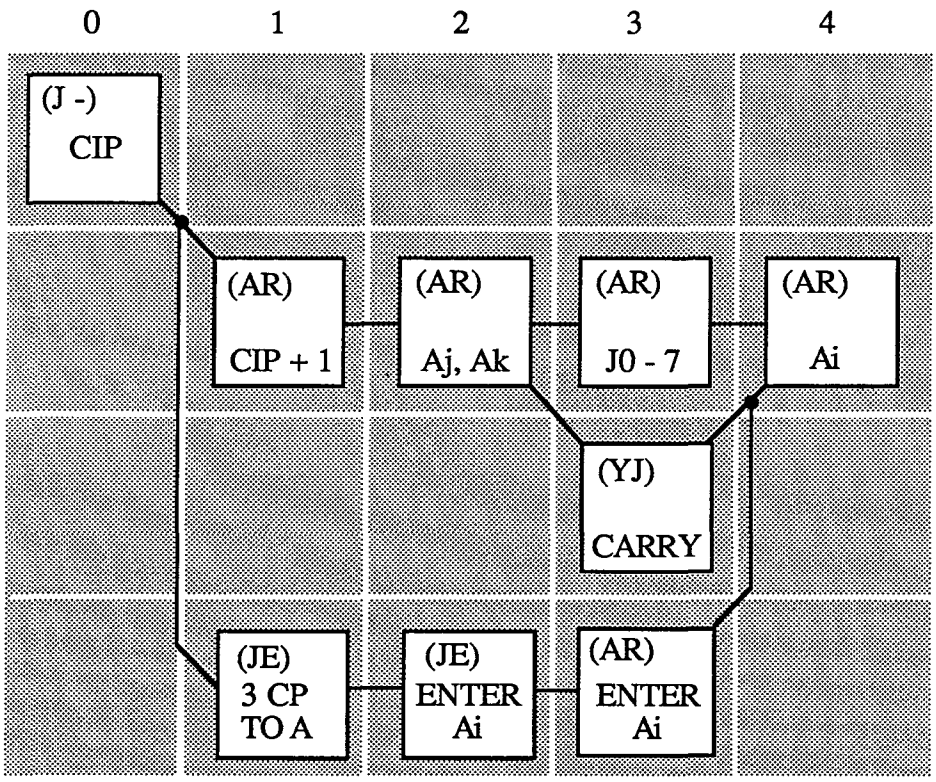
(AR2) I40 term is made from a Subtract and Section 0, 1 Enabled or a Section 0 Carry with a Section 1 Enable or a Section 1 Carry.

(AR3) I40 term is made from a Subtract and Section 0, 1, 2 Enable, or a Section 0 Carry with the Section 1, 2 Enabled, or Section 1 Carry with a Section 2 Enable or a Section 2 Carry.

Hardware Trng.
A-6229A J.E.S.

CRAY Y-MP PROPAGATED ENABLES AND SECTIONS ADDRESS ADD

CRAY PROPRIETARY



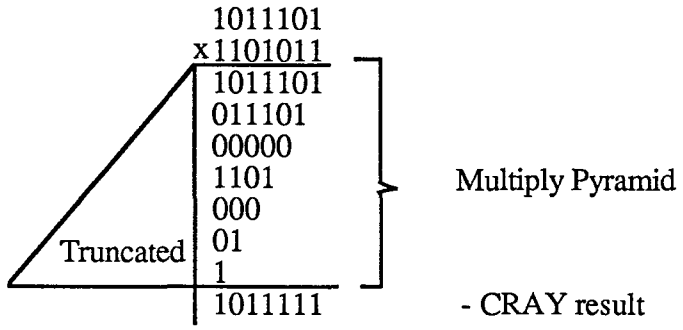
030ijk - INTEGER SUM OF (Aj) AND (Ak) TO Ai
031ijk - INTEGER DIFFERENCE OF (Aj) LESS (Ak) TO Ai *A-7733A*

CRAY Y-MP ADDRESS ADD

ADDRESS MULTIPLY FUNCTIONAL UNIT

The Address Multiply functional unit is used to multiply two 24-bit operands (X-mode) or two 32-bit operands (Y-mode) to produce a 24 or 32-bit result. If the machine is running in X-mode, only the lower 24 bits of the final result are accessible, while in Y-mode, it is the lower 32 bits that are accessible. In either case, the remaining upper bits are simply truncated, as shown in the simplified 7-bit binary multiply example below:

```
    1011101
  x 1101011
    1011101
   1011101
  0000000
 1011101
0000000
1011101
1011101
10011011011111 - actual result
```



The actual addition of terms in the multiply pyramid is accomplished through a series of 3 and 4-bit adds and carries, and in some cases even a 5-bit add is used. The instruction utilizing this functional unit is the 032ijk, where j and k designate the operand registers and i is the result register. A total of 6 CPs elapses from the time the j and k operands leave the A registers until the result enters the appropriate Ai register.

Hardware Trng.
YM08/06 R.A.M.

CRAY Y-MP ADDRESS MULTIPLY FUNCTIONAL UNIT INSTRUCTIONS

ADDRESS MULTIPLY

032ijk	Ai	Aj * Ak	Integer product of (Aj) times (Ak) to Ai
032ij0	Ai	Aj * 1	Transmit (Aj) to Ai
032i0k	Ai	Ak * 0	Zero fill Ai

ADDRESS MULTIPLY FUNCTIONAL UNIT (OPTIONS INVOLVED)

(AA0) Option

The (AA0) option receives bits (2^0 - 2^{13}) of both the j and k operands, forms their logical product, and begins the summation of the resulting terms in these bit positions. The partial summation of bit positions (2^5 - 2^{16}) is sent to the (AD0) option for the final summation; while the summation for bit positions (2^0 - 2^4) is finalized on board, latched for two more clock periods, and then sent directly to the appropriate A register via the AFU path select on the (AR) options.

(AB) Options

The three (AB) options form the logical products and begin the summation of the middle bits in the pyramid. To do this, all three options receive bits (2^0 - 2^{19}) of the j operand, while (AB0) also receives bits (2^0 - 2^{19}) of the k operand, (AB1) receives bits (2^6 - 2^{25}) of the k operand, and (AB2) receives bits (2^{12} - 2^{31}) of the k operand. Different terms of the partial summations obtained on (AB0) and (AB1) are sent to (AD0) and (AF0) for the final summation, while all the terms of the partial summation obtained on (AB2) are sent only to (AF0).

(AC0) Option

The (AC0) option performs the same functions as the (AA) and (AB) options: multiplication and partial summation. It receives bits (2^0 - 2^{11}) of the k operand and bits (2^{14} - 2^{31}) of the j operand, and outputs its partial summation bits to the (AD0) and (AF0) options for the final summation.

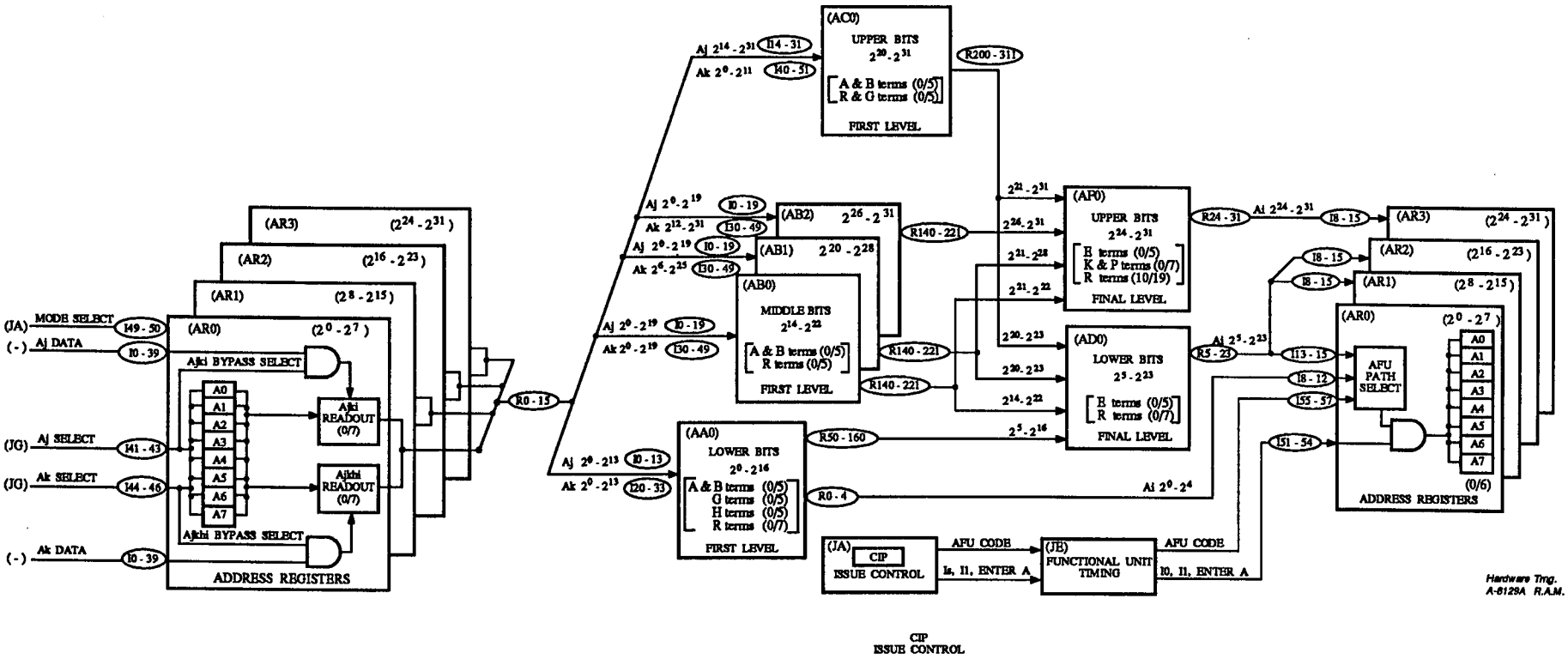
(AD0) Option

The (AD0) option receives partial summations from the (AA), (AB), and (AC) options, and works out the final summation for bit positions (2^5 - 2^{23}). It sends its results to the appropriate A register via the AFU path select mux on the (AR) options.

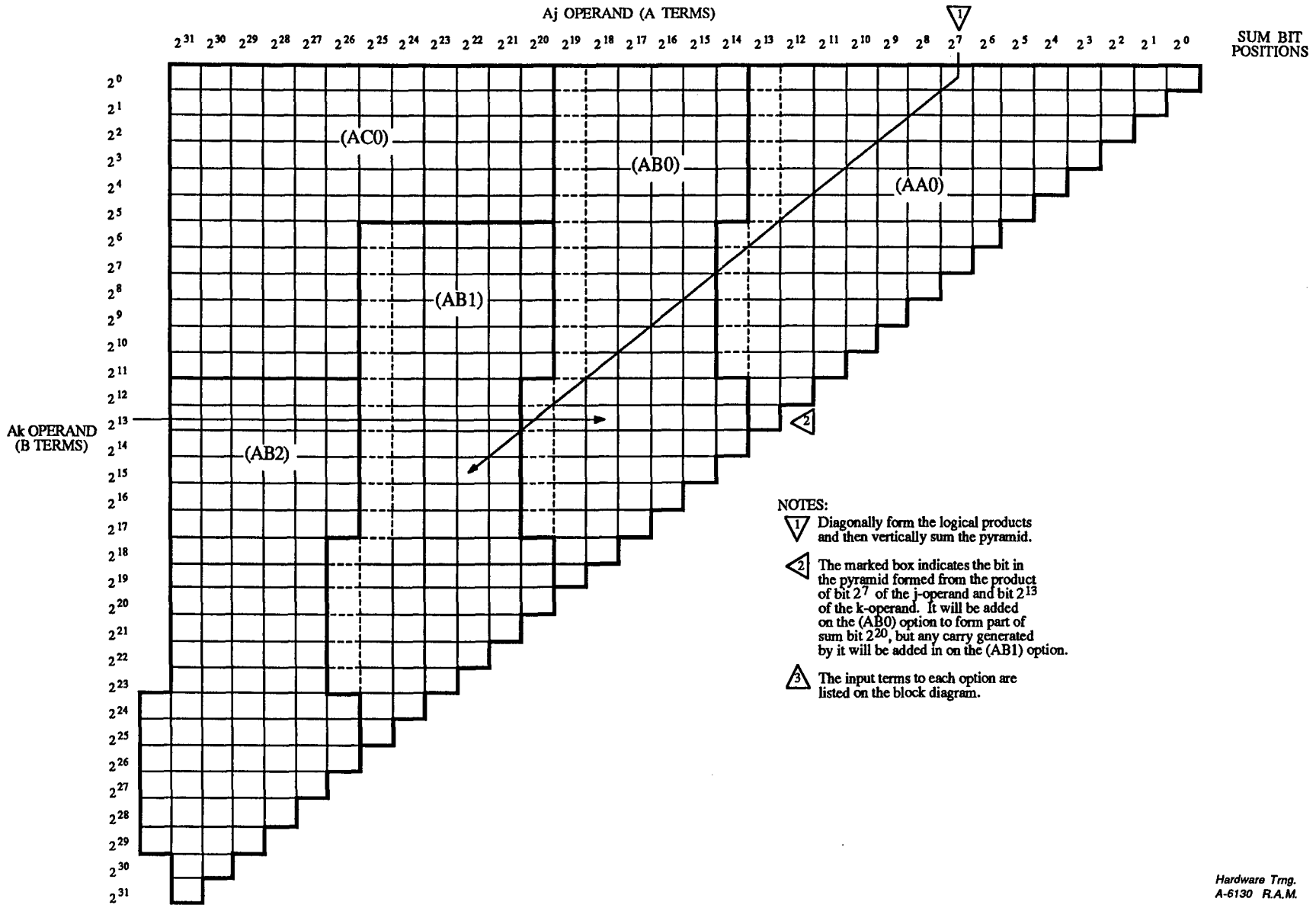
(AF0) Option

The (AF0) option is similar in function to the (AD0) option, but it receives input terms only from the (AB) and (AC) options, and obtains the final summation for bit positions (2^{24} - 2^{31}).

Hardware Trng.
YM08/07 R.A.M.



CRAY Y-MP ADDRESS MULTIPLY BLOCK DIAGRAM



Hardware Trng.
A-6130 R.A.M.

CRAY Y-MP ADDRESS MULTIPLY
FIRST LEVEL LOGICAL PRODUCTS PYRAMID

CRAY PROPRIETARY

(AA)

a13 b00	a12 b00	a11 b00	a10 b00	a09 b00	a08 b00	a07 b00	a06 b00	a05 b00	a04 b00	a03 b00	a02 b00	a01 b00	a00 b00
a12 b01	a11 b01	a10 b01	a09 b01	a08 b01	a07 b01	a06 b01	a05 b01	a04 b01	a03 b01	a02 b01	a01 b01	a00 b01	
a11 b02	a10 b02	a09 b02	a08 b02	a07 b02	a06 b02	a05 b02	a04 b02	a03 b02	a02 b02	a01 b02	a00 b02		
a10 b03	a09 b03	a08 b03	a07 b03	a06 b03	a05 b03	a04 b03	a03 b03	a02 b03	a01 b03	a00 b03			
a09 b04	a08 b04	a07 b04	a06 b04	a05 b04	a04 b04	a03 b04	a02 b04	a01 b04	a00 b04				
a08 b05	a07 b05	a06 b05	a05 b05	a04 b05	a03 b05	a02 b05	a01 b05	a00 b05					
a08 b06	a07 b06	a06 b06	a05 b06	a04 b06	a03 b06	a02 b06	a01 b06	a00 b06					
a07 b07	a06 b07	a05 b07	a04 b07	a03 b07	a02 b07	a01 b07	a00 b07						
a06 b08	a05 b08	a04 b08	a03 b08	a02 b08	a01 b08	a00 b08							
a05 b09	a04 b09	a03 b09	a02 b09	a01 b09	a00 b09								
a04 b10	a03 b10	a02 b10	a01 b10	a00 b10									
a03 b11	a02 b11	a01 b11	a00 b11										
a01 b12	a00 b12												
a00 b13													

(AC)

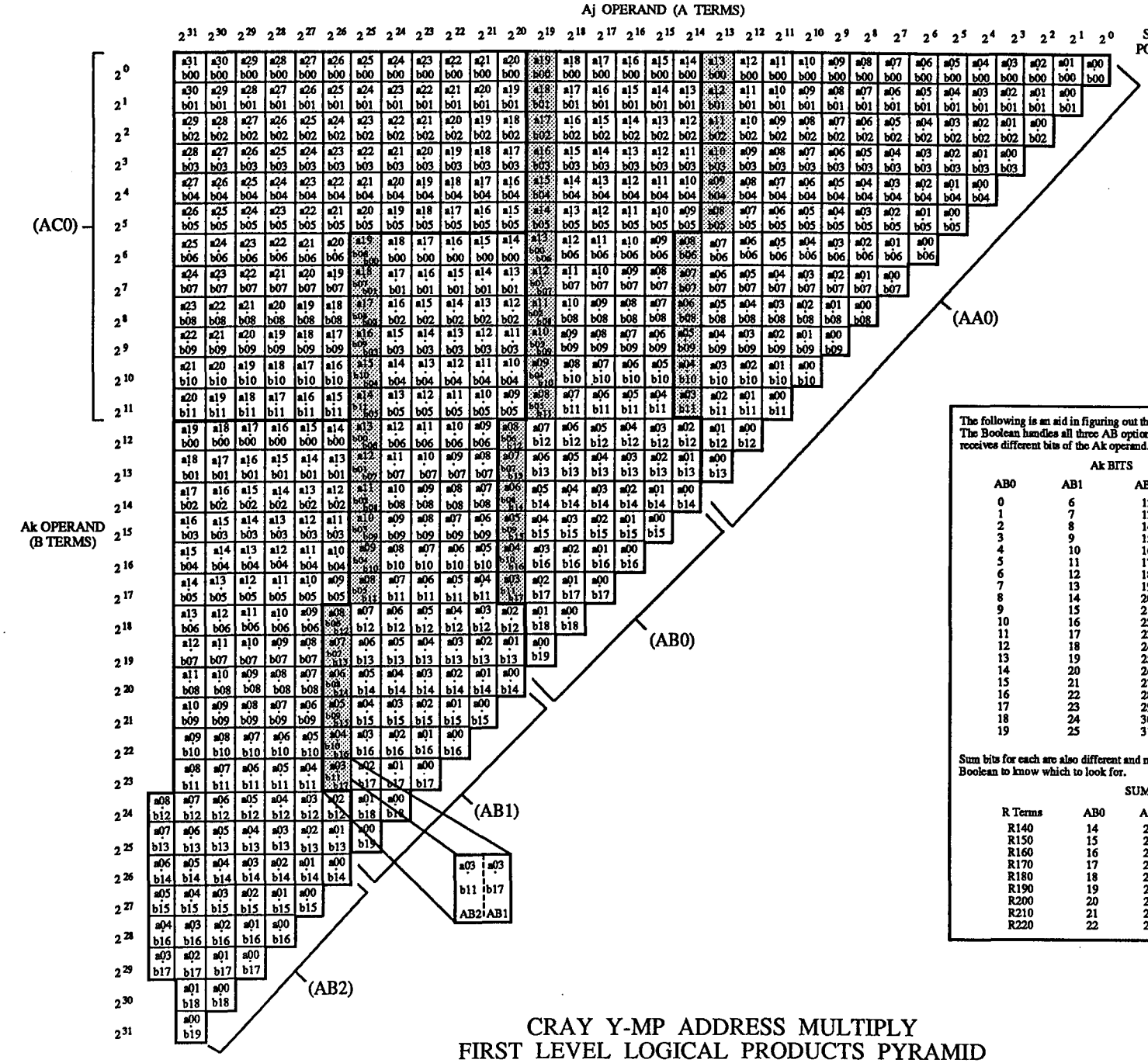
a31 b00	a30 b00	a29 b00	a28 b00	a27 b00	a26 b00	a25 b00	a24 b00	a23 b00	a22 b00	a21 b00	a20 b00	a19 b00
a30 b01	a29 b01	a28 b01	a27 b01	a26 b01	a25 b01	a24 b01	a23 b01	a22 b01	a21 b01	a20 b01	a19 b01	a18 b01
a29 b02	a28 b02	a27 b02	a26 b02	a25 b02	a24 b02	a23 b02	a22 b02	a21 b02	a20 b02	a19 b02	a18 b02	a17 b02
a28 b03	a27 b03	a26 b03	a25 b03	a24 b03	a23 b03	a22 b03	a21 b03	a20 b03	a19 b03	a18 b03	a17 b03	a16 b03
a27 b04	a26 b04	a25 b04	a24 b04	a23 b04	a22 b04	a21 b04	a20 b04	a19 b04	a18 b04	a17 b04	a16 b04	a15 b04
a26 b05	a25 b05	a24 b05	a23 b05	a22 b05	a21 b05	a20 b05	a19 b05	a18 b05	a17 b05	a16 b05	a15 b05	a14 b05
a25 b06	a24 b06	a23 b06	a22 b06	a21 b06	a20 b06	a19 b06						
a24 b07	a23 b07	a22 b07	a21 b07	a20 b07	a19 b07	a18 b07						
a23 b08	a22 b08	a21 b08	a20 b08	a19 b08	a18 b08	a17 b08						
a22 b09	a21 b09	a20 b09	a19 b09	a18 b09	a17 b09	a16 b09						
a21 b10	a20 b10	a19 b10	a18 b10	a17 b10	a16 b10	a15 b10						
a20 b11	a19 b11	a18 b11	a17 b11	a16 b11	a15 b11	a14 b11						

(AB)

a19 b00	a18 b00	a17 b00	a16 b00	a15 b00	a14 b00	a13 b00
a18 b01	a17 b01	a16 b01	a15 b01	a14 b01	a13 b01	a12 b01
a17 b02	a16 b02	a15 b02	a14 b02	a13 b02	a12 b02	a11 b02
a16 b03	a15 b03	a14 b03	a13 b03	a12 b03	a11 b03	a10 b03
a15 b04	a14 b04	a13 b04	a12 b04	a11 b04	a10 b04	a09 b04
a14 b05	a13 b05	a12 b05	a11 b05	a10 b05	a09 b05	a08 b05
a13 b06	a12 b06	a11 b06	a10 b06	a09 b06	a08 b06	
a12 b07	a11 b07	a10 b07	a09 b07	a08 b07	a07 b07	
a11 b08	a10 b08	a09 b08	a08 b08	a07 b08	a06 b08	
a10 b09	a09 b09	a08 b09	a07 b09	a06 b09	a05 b09	
a09 b10	a08 b10	a07 b10	a06 b10	a05 b10	a04 b10	
a08 b11	a07 b11	a06 b11	a05 b11	a04 b11	a03 b11	
a08 b12	a07 b12	a06 b12	a05 b12	a04 b12	a03 b12	a02 b12
a07 b13	a06 b13	a05 b13	a04 b13	a03 b13	a02 b13	a01 b13
a06 b14	a05 b14	a04 b14	a03 b14	a02 b14	a01 b14	a00 b14
a05 b15	a04 b15	a03 b15	a02 b15	a01 b15	a00 b15	
a04 b16	a03 b16	a02 b16	a01 b16	a00 b16		
a03 b17	a02 b17	a01 b17	a00 b17			
a01 b18	a00 b18					
a00 b19						

Hardware Trng.
A-6228 J.E.S.

CRAY Y-MP ADDRESS MULTIPLY
LOGICAL PRODUCT TEMPLATE



The following is an aid in figuring out the AB option Boolean. The Boolean handles all three AB options (0, 1, 2), but each option receives different bits of the Ak operand.

Ak BITS

AB0	AB1	AB2	I Terms
0	6	12	0
1	7	13	1
2	8	14	2
3	9	15	3
4	10	16	4
5	11	17	5
6	12	18	6
7	13	19	7
8	14	20	8
9	15	21	9
10	16	22	10
11	17	23	11
12	18	24	12
13	19	25	13
14	20	26	14
15	21	27	15
16	22	28	16
17	23	29	17
18	24	30	18
19	25	31	19

Sum bits for each are also different and must be shifted in the AB Boolean to know which to look for.

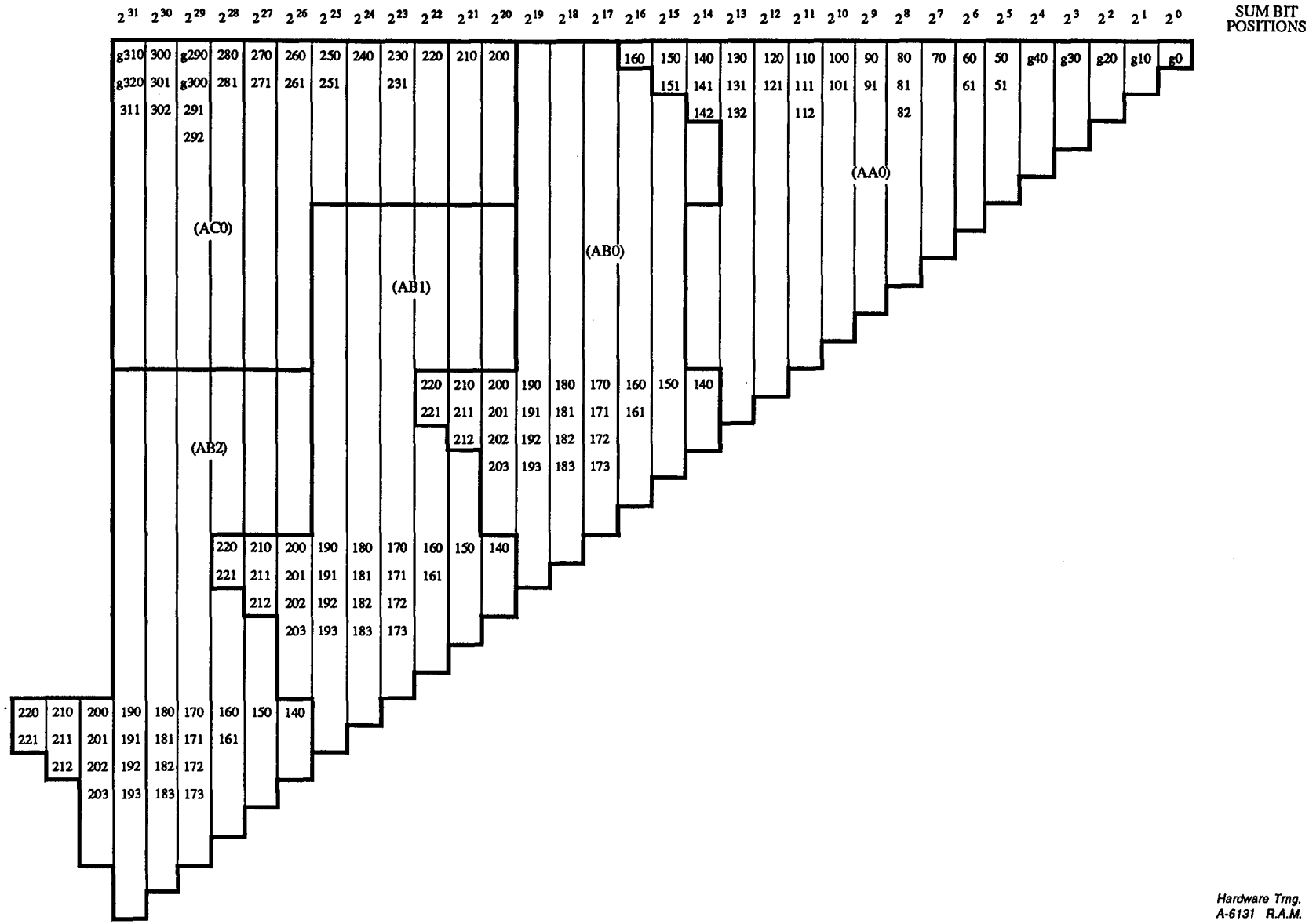
SUM BITS

R Terms	AB0	AB1	AB2
R140	14	20	26
R150	15	21	27
R160	16	22	28
R170	17	23	29
R180	18	24	30
R190	19	25	31
R200	20	26	
R210	21	27	
R220	22	28	

A-7718A

CRAY Y-MP ADDRESS MULTIPLY
FIRST LEVEL LOGICAL PRODUCTS PYRAMID

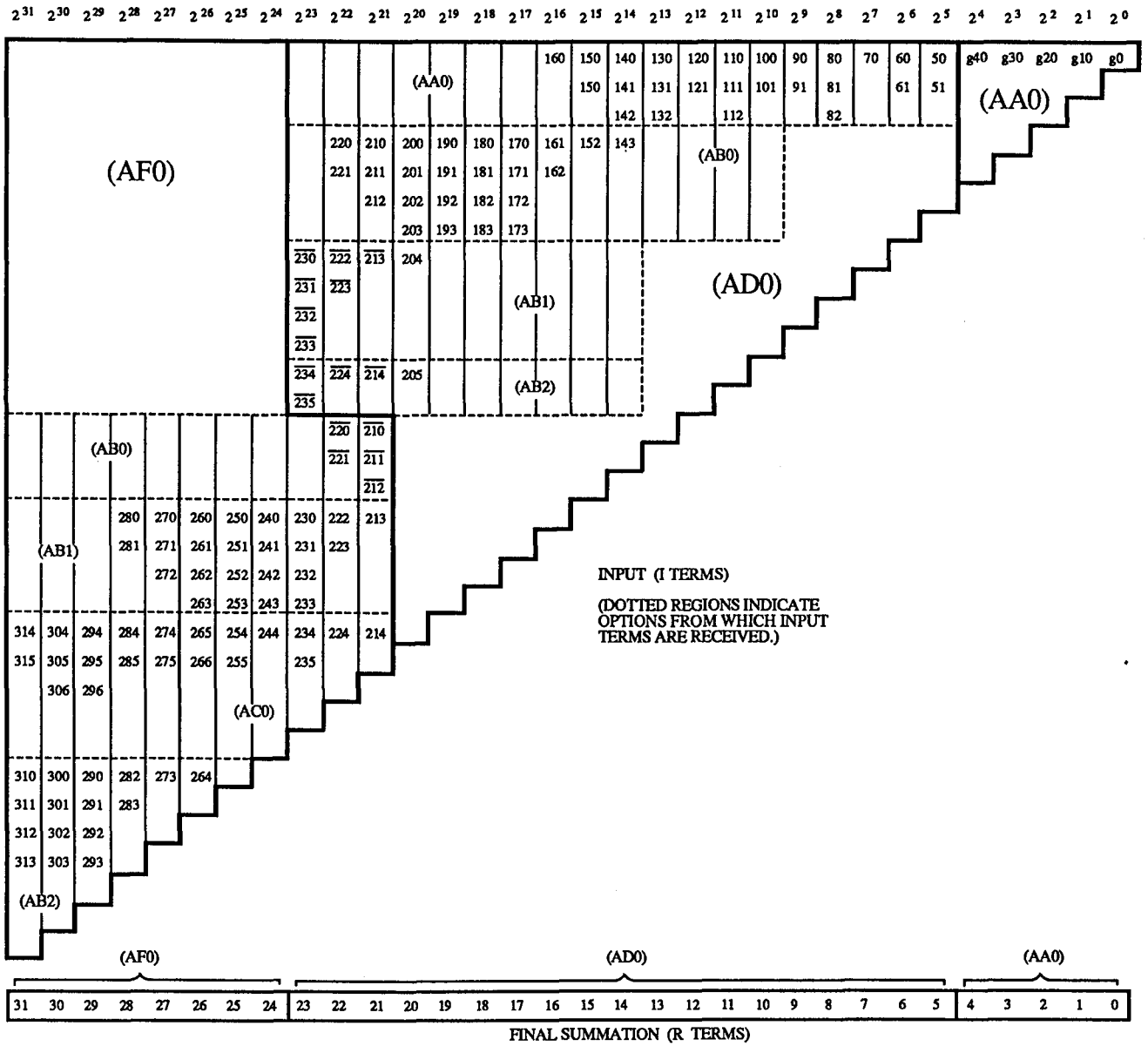
CRAY PROPRIETARY



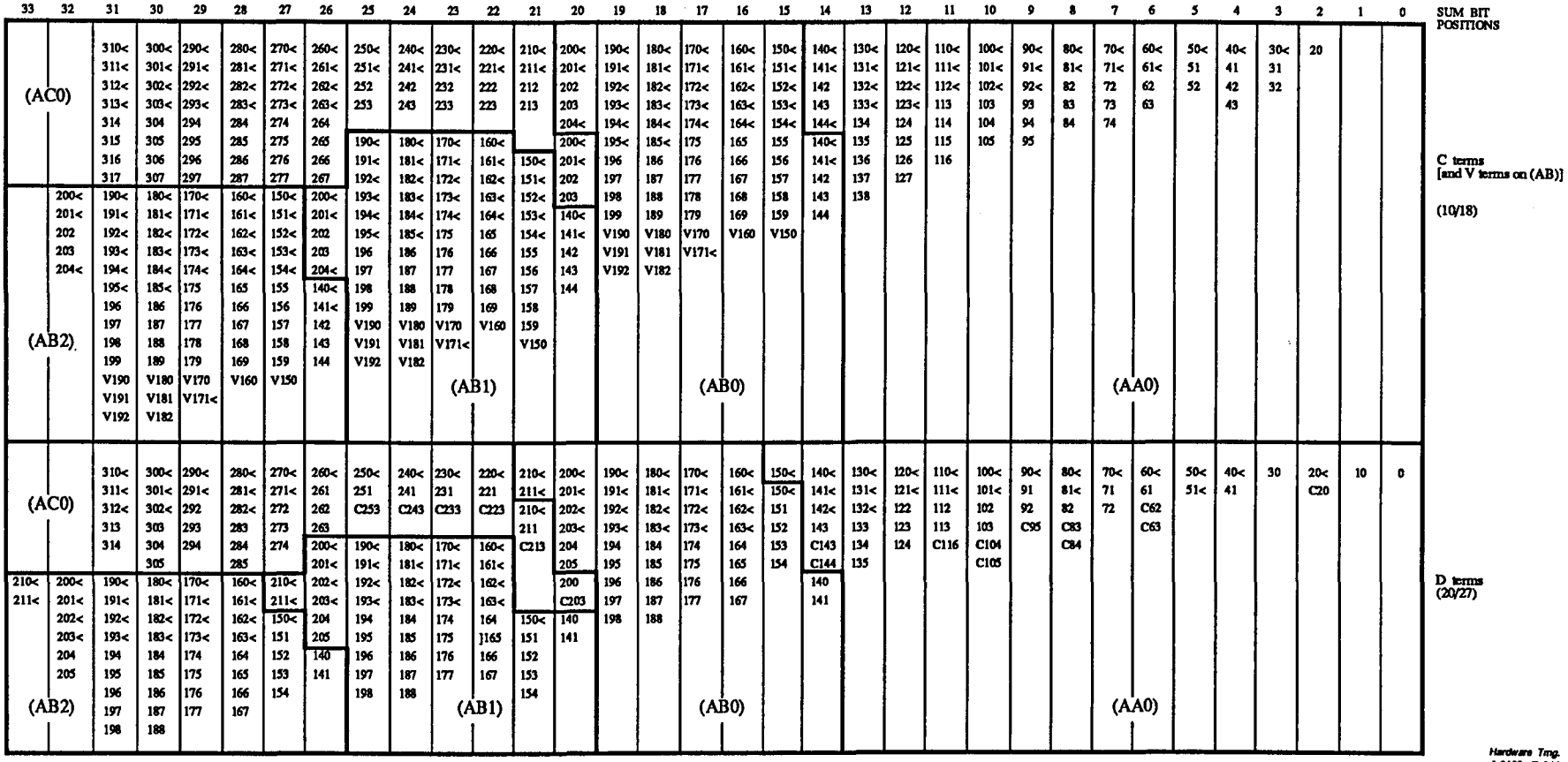
Hardware Trng.
A-6131 R.A.M.

CRAY Y-MP ADDRESS MULTIPLY
FIRST LEVEL PARTIAL SUMMATION
(G AND R TERMS)

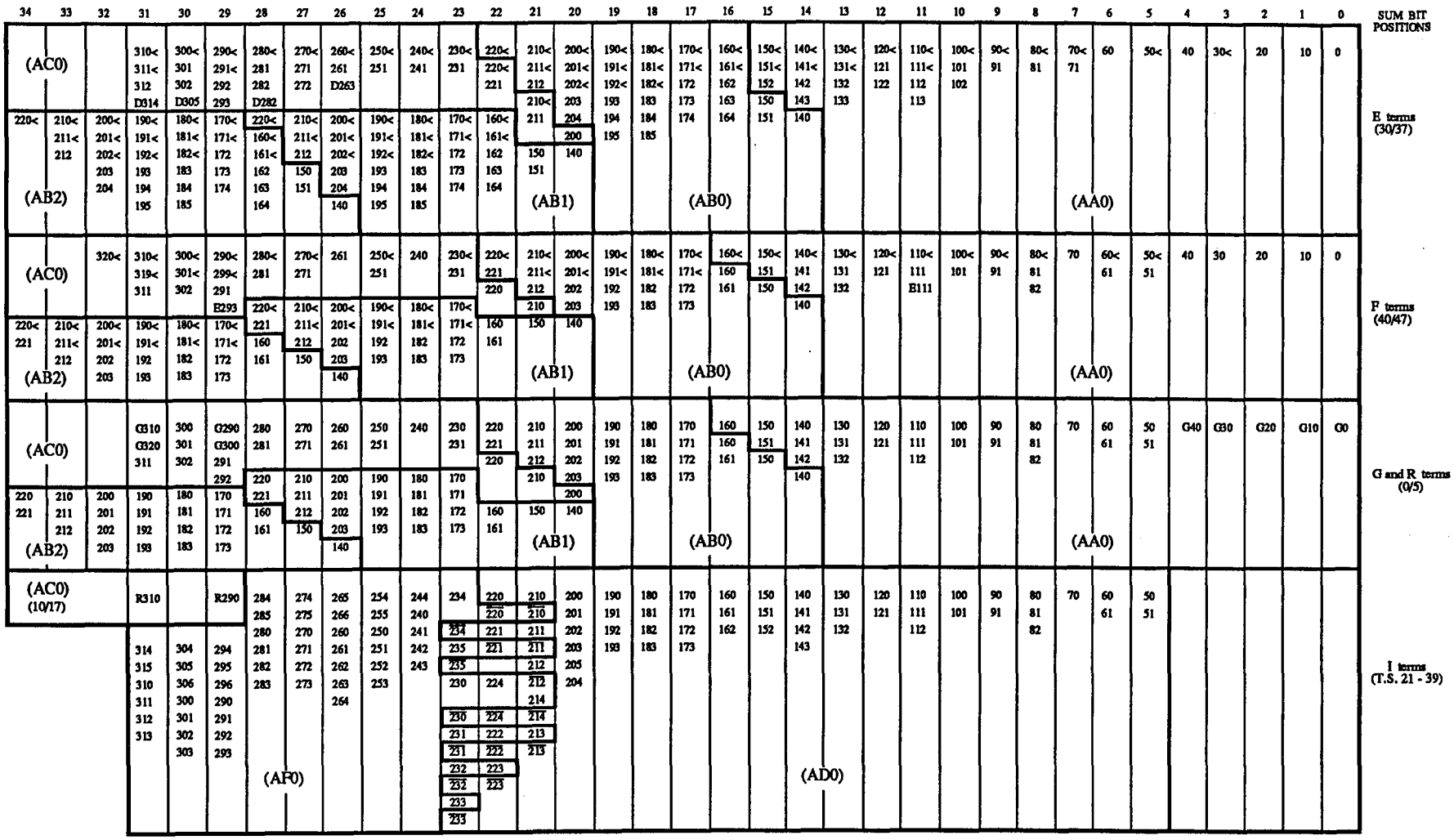
CRAY PROPRIETARY



CRAY Y-MP ADDRESS MULTIPLY FINAL LEVEL INPUT & SUMMATION (I AND R TERMS)



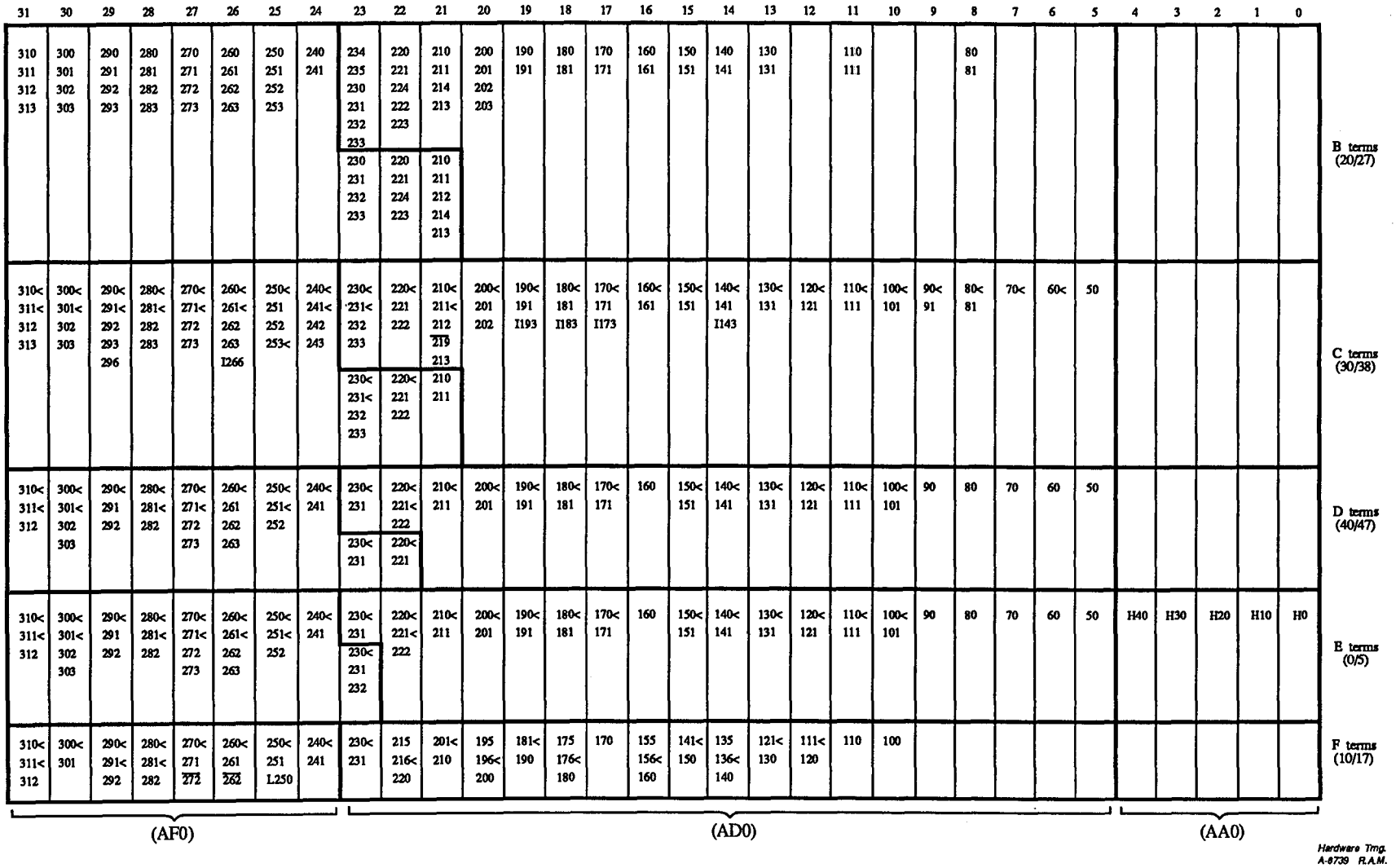
CRAY Y-MP ADDRESS MULTIPLY PYRAMID
(sheet 1 of 4)



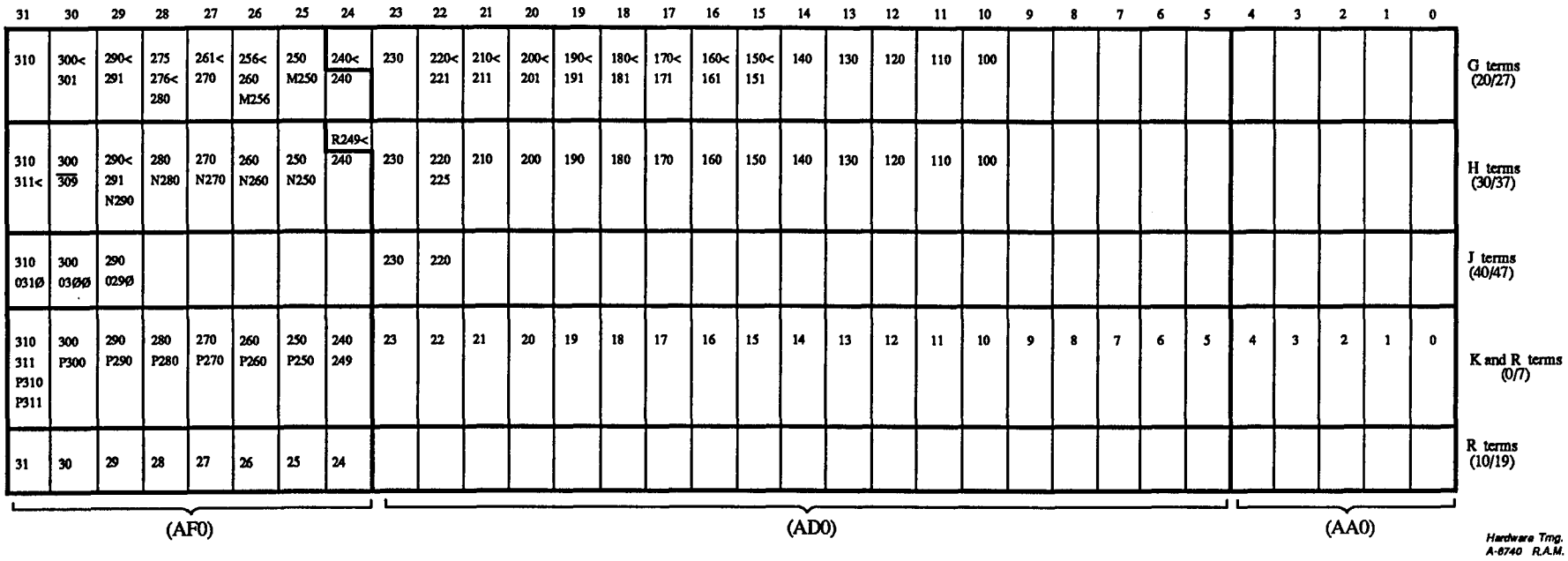
CRAY Y-MP ADDRESS MULTIPLY PYRAMID
(sheet 2 of 4)

CRAY PROPRIETARY

CRAY PROPRIETARY

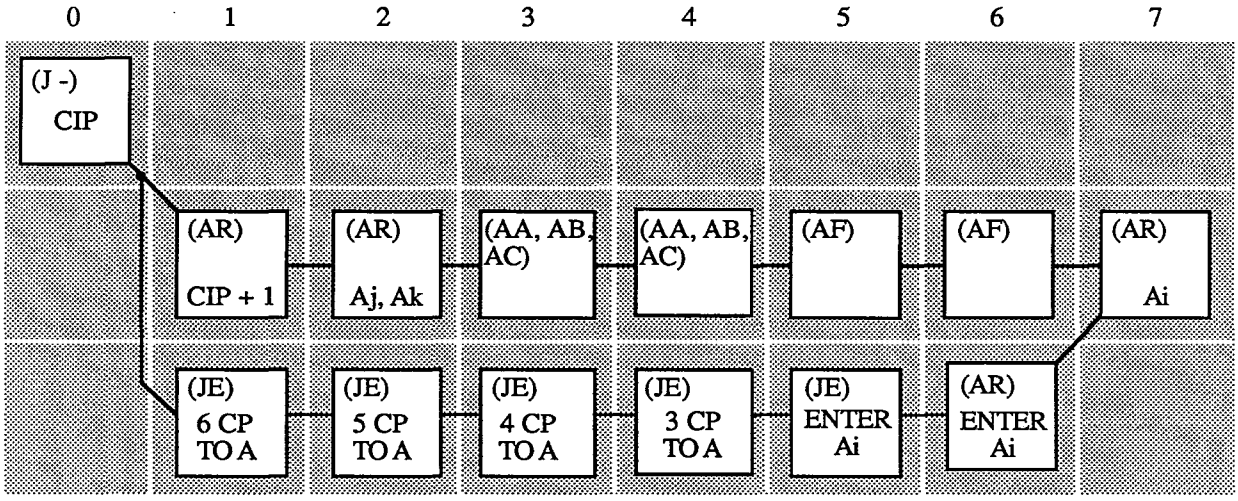


CRAY Y-MP ADDRESS MULTIPLY PYRAMID
(sheet 3 of 4)



CRAY Y-MP ADDRESS MULTIPLY PYRAMID
(sheet 4 of 4)

CRAY PROPRIETARY



032ijk - INTEGER PRODUCT OF (Aj) AND (Ak) TO

A-7732A

CRAY Y-MP ADDRESS MULTIPLY

CRAY Y-MP SCALAR REGISTERS

There are eight Scalar registers, S0 - S7, each is 64 bits in length. The Scalar registers have their own functional units, which are the Scalar Add, Shift, Logical, Population Parity, and Leading Zero. The Scalar registers also share the Floating-point functional units with the Vector register; these include the Floating Add, Floating Reciprocal and Floating Multiply. When a Scalar instruction that uses any of the Floating-point functional units is in CIP, the scalar issue control must check to see if the Vector registers have the functional unit reserved; if so, the Scalar instruction will wait in CIP until the Floating-point functional unit reservation has cleared, and then start execution.

The Scalar registers can receive results from memory by using Memory Port A during Scalar Reference (13X) instructions. Scalar registers can also be loaded from the T registers, which are an intermediate storage for the Scalar registers. There are 64_{10} T registers (T00 - T77), each of which is 64 bits in length. The T registers are block loaded to/from memory through the (036, 037) instructions. Because of the number of clock periods it takes to get one word from memory in a scalar reference, it makes sense to block load the T registers into the Scalar registers when needed or vice-versa. The Scalar registers can also be loaded with an element from any Vector register (V0 - V7); and any Scalar register, except S0, can be loaded into any Vector register (V0 - V7). Finally, the Scalar register can be loaded from the Shared registers (ST, SM), Vector Mask register (VM), Status register (SR), Performance Monitor (PM), Real-time clock (RTC), or with constant values.

While data for the Scalar registers can come from many sources, it must be sent to a single Scalar register. Which Scalar register receives these results is dependent on the (i) field of the Scalar instruction that issued. Before functional unit data can be loaded into the (Si) register, the functional unit path must be selected. To select a functional unit path, a SFU source code is generated from the instruction that is being decoded in CIP. The SFU source code along with the enter S0 - S7 directs the data into the correct Scalar register.

In order to get results from a functional unit you must first send operands out to the functional unit. The operands can be either (Si), (Sj), or (Sk) depending on the instruction that is decoded in CIP. The correct operand is selected from a decode of the (Sk), (Sj), (Si) operand bits, or Bypassed if Si = Sk, or Si = Sji. The operand (Si), (Sj), or (Sk) are steered to the correct Scalar functional unit by decoding the SFU Select Code for that particular functional unit. When steering operands to the Floating-point functional units, the fact that there is no VFU select will automatically specify a scalar operand to the Floating-point functional unit.

The Scalar registers have two features each of which saves a clock period. One is the Bypass feature and the other is the location of the Scalar Logical functional unit which is on the same (VB, VC) options as the Scalar registers, which allows Scalar Logical instructions to be completed in less than 1 CP. The Bypass feature is automatically utilized when the result register is the same as an operand register for the next instruction. For example, if an 060123 instruction was issued followed by an 064421, the 064421 would hold issue until the 060123 instruction brought up the Enter S1 signal, at which time the 064421 instruction would issue, obtaining S1 data from the Bypass path as Bypass (Sk); S2 data would be read directly out of the Scalar register. The two operands S2 and S1 would then be sent to the Floating Multiply functional unit.

All the Scalar registers can be either used as an operand or destination register, except for S0, which is special when used as an operand. When an instruction uses S0 as the (j) field, the result read from S0 would be all zeroes, all 64 bits. When S0 is used for the k field of the instruction S0 is read with only bit (2^{63}) set. When S0 is used for the (i) field of the instruction, the contents of (S0) is read as expected, as for example in an 056ijk instruction.

The Scalar registers use a SFU source code which assigns a functional unit path to a Scalar register on a write, and a path for the operands on a read of S0 - S7. The Enter S0 - S7 signals select one of eight Scalar registers into which to load the result data. Enter S0 - S7 comes from the decode of the (i) field of the instruction. The read paths are selected from the decode of the (i), (j), or (k) field of the instructions, depending on the instruction that issued.

SCALAR REGISTERS (OPTIONS INVOLVED)

(JB0) Option

The (JB0) contains the Scalar register issue control. The Go Issue from a previously issued instruction advances the next instruction into CIP. During CIP time, the (J-, H-) options will determine if the instruction needs to be controlled by the (JB) option, if not the (JB) will generate a Go Issue to the (J-, H-) issue control options. All (J-, H-) options must agree on a Go Issue before the next instruction can enter CIP. If the instruction in CIP is recognized by the (JB) option as one of its instructions to control, the (JB) will check for the following conflicts and if they exist, Hold Issue until the conflict is resolved.

- Access conflict - Two instructions would be trying to write the Scalar registers within the same clock period, hold until Access conflicts go away.
- Conflict (Sk), (Sj), (Si) - (Sk), (Sj), or (Si) are being reserved as destinations and wanting to be used as operands or destination, hold until reservation is cleared.
- Parcel Split - Caused by 2/3 parcel in CIP and the data is invalid, hold until the Enter LIP 1 or LIP 01 arrives.
- S0, T conflict - Reservations or Access with S0 or the T registers, hold until the continue hold S0, T drops.
- Scalar Floating Reciprocal - Scalar Floating Reciprocal (SFR) followed by a SFR causes a Hold Issue until the first SFR has had a chance to progress.
- Hold After 056 - 057 - Any (00x - 03x, 04x - 07x, 10x - 17x) instruction will hold after an 056 - 057 instruction has issued, due to taking 2 CPs to readout (Si) and (Sj) from the Scalar registers.
- RT Access - Reading the Real-time clock can conflict with the reading of the Shared T (ST) because they both use the same path on the Scalar registers.

Once the Scalar instructions issues the number of clock periods it takes is encoded and sent to the (JG) option.

(JG0) Option

The (JG) option receives the encoded instruction times and decodes the encodes for the instruction times and places the instruction times into the proper delay chain which runs from (18 - 3) times at times (12, 11, 8, 7, 6 and 5) the signals are passed to the (JB) option to determine if an Access conflict exists with an instruction waiting in CIP. The (JG) option can also be test pointed to look at the ghijk fields of the instructions. The (JG) second major function is to send out the CIP (hijk) designators to the various parts of the mainframe, these will then be a fanout on the (VS, VF) options.

(VB0 - VB15) Options

The (VB) options contain half of the Scalar registers. There are 16 (VB) options, each of which contain 2 bits of the 64-bit word. (VB0) contains bits (2^1, 2^0), while (VB15) contains bits (2^{31}, 2^{30}). Each (VB) option contains 2 bits of every Scalar register S0 - S7. The (VB) options also contain the Scalar Logical functional unit which execute the (044 - 051) instructions. The Scalar Logical instructions are performed in less than 1 CP, since the functional unit does not need to be concerned with propagated carries, but only corresponding bit positions in order to do the logical functions. The (VB) options contain a Bypass path which allows either (Sk), (Si), or (Sj) data to bypass the S register load path and load the result of a previous instruction directly into the Readout register. The Bypass path is used when (Si), (Sj), or (Sk) equals (Si) of the previous instruction.

(VC0 - VC15) Options

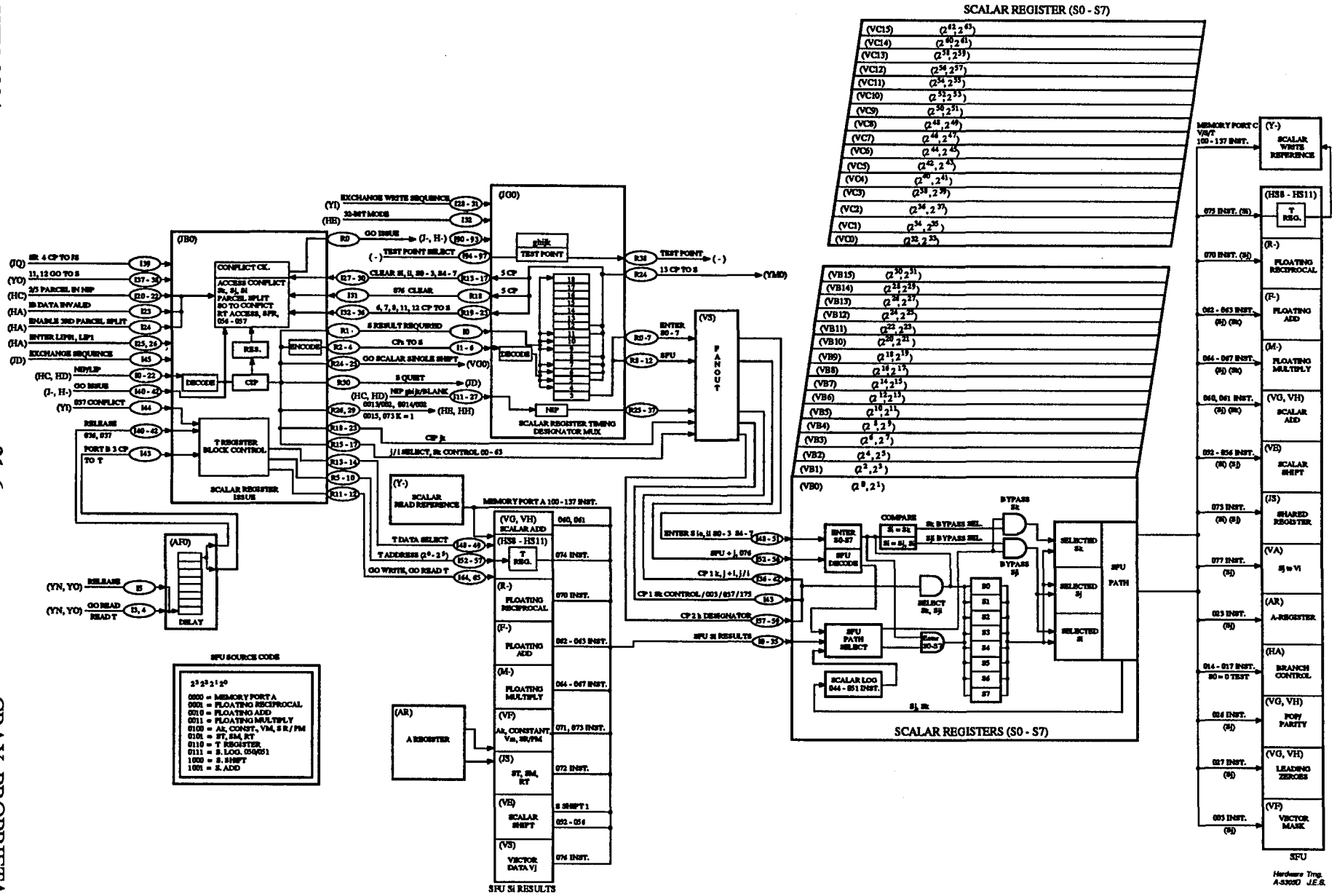
The (VC) options are identical to the (VB) options, except that the (VC) options handle the upper 32 bits of the 64-bit Scalar registers. The (VC) options, like the (VB) options, can receive results from: Memory, T registers, Floating Reciprocal, Floating Add, Floating Multiply, Ak, Constants, Vector Mask (VM), Status register (SR), Performance Monitor (PM), Shared T (ST), Semaphore (SM), Real Time Clock (RTC), Scalar Shift, Scalar Add, and Vector Element. The (VC), like the (VB) can send operands (Sj), (Sk), or (Si) to: Memory, T registers, Floating Reciprocal, Floating Add, Floating Multiply, Scalar Add, Scalar Shift, Shared register (SR), Vector Element, A register, Branch Control, Population, Parity, and Leading Zero Count.

CRAY Y-MP VECTOR/SCALAR OPTIONS

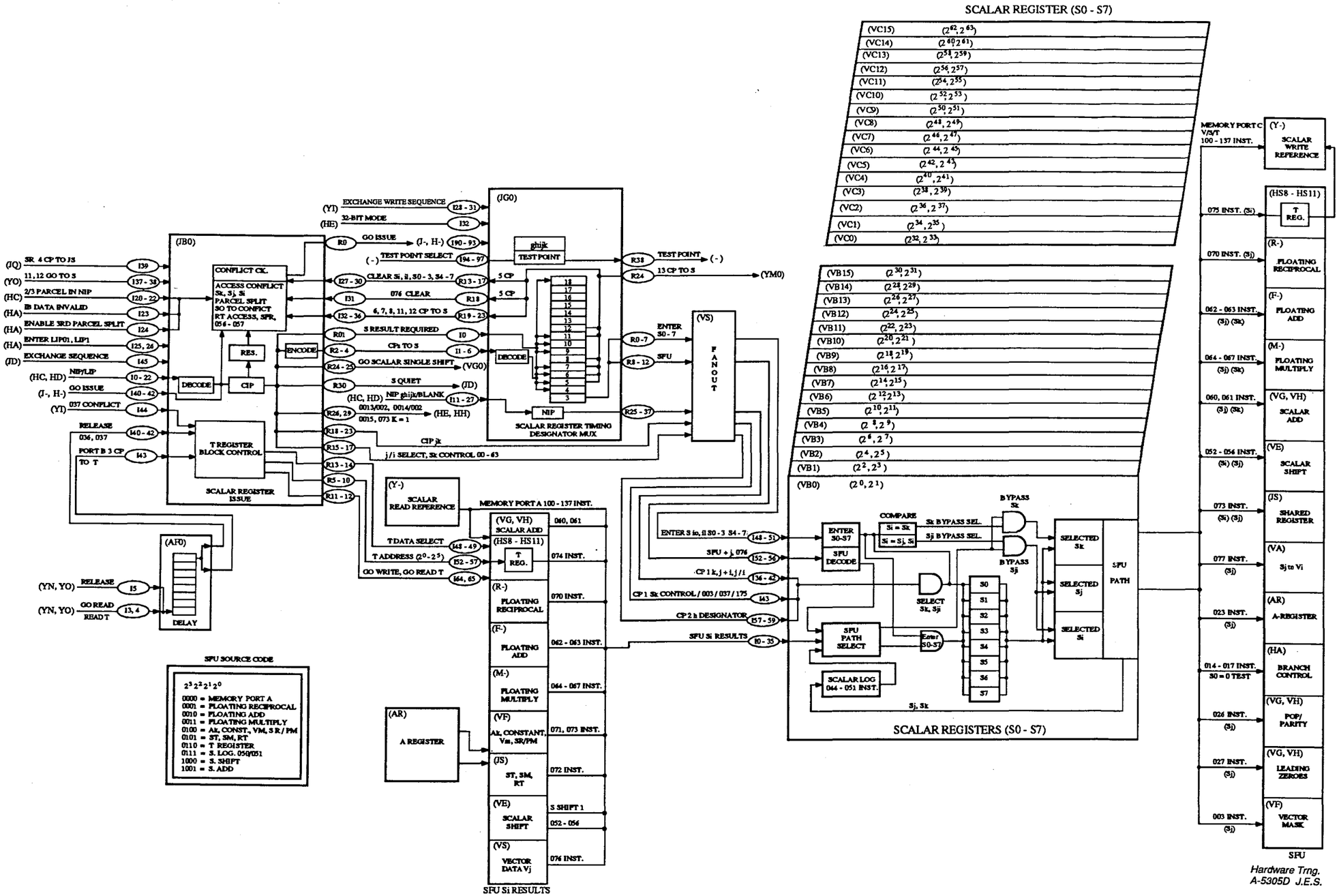
OPTION TYPE	NUMBER USED	DESCRIPTION
VA	16	Steering Vector Results
VB	16	Vector Operands/Scalar Register lower
VC	16	Vector Operands/Scalar Registers upper
VD	4	Vector Register Control
VE	4	Vector/Scalar Shift
VF	4	Vector Mask, Scalar Constant
VG	2	Vector/Scalar Add, Pop. Shift Control
VH	2	Vector/Scalar Add, Pop/Leading Zero
VS	32	Vector Registers
	96	TOTAL V Series Options

YM08/05

CRAY PROPRIETARY



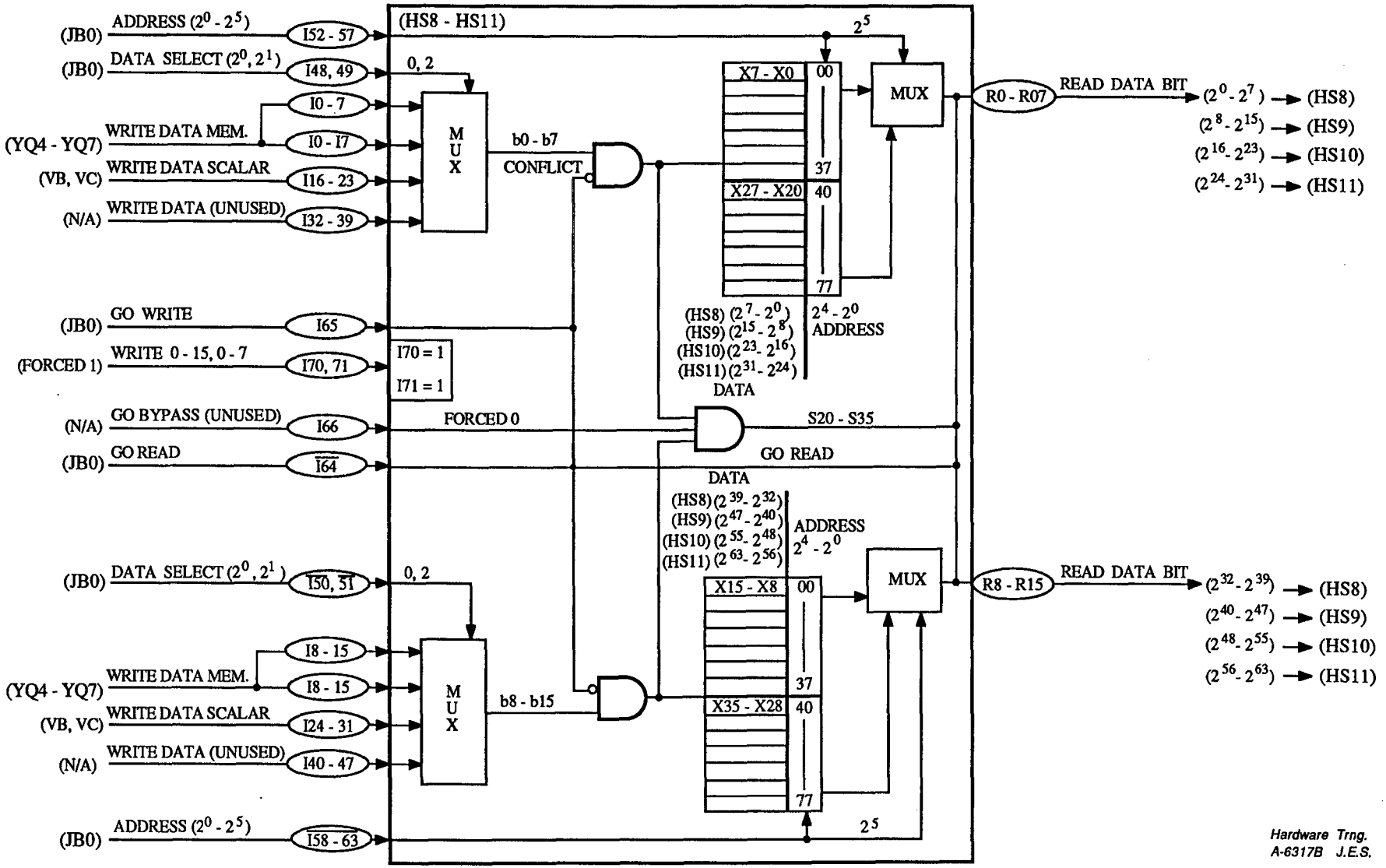
CRAY Y-MP SCALAR REGISTERS BLOCK DIAGRAM



CRAY Y-MP SCALAR REGISTERS BLOCK DIAGRAM

Hardware Trng.
A-5305D J.E.S.

CRAY PROPRIETARY



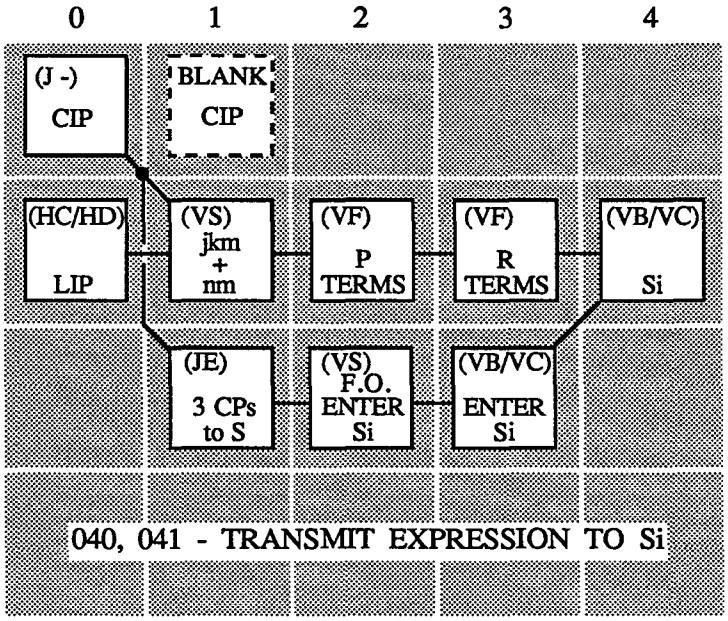
CRAY Y-MP (HS) BLOCK DIAGRAM
T REGISTERS

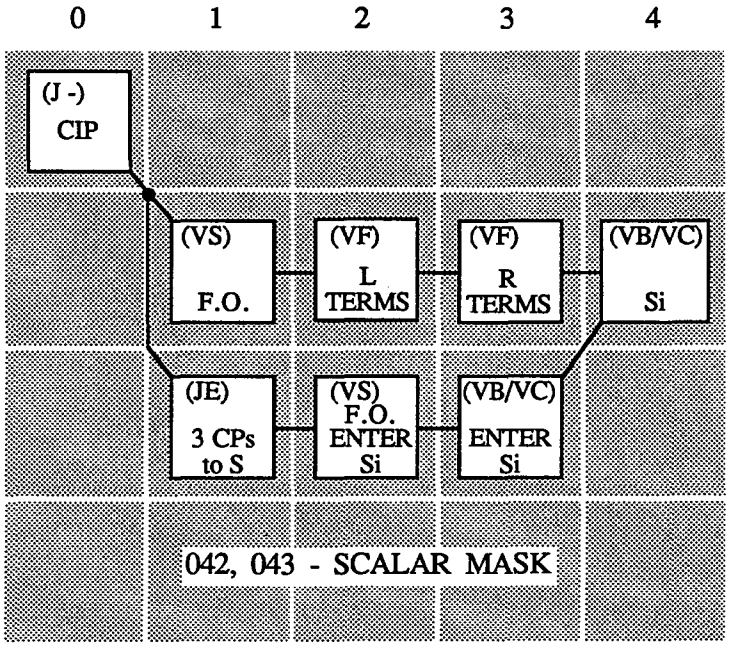
CRAY PROPRIETARY

Tjk	(Hs11) 2^{63} - 2^{56}	(Hs10) 2^{55} - 2^{48}	(Hs9) 2^{47} - 2^{40}	(Hs8) 2^{39} - 2^{32}	(Hs11) 2^{31} - 2^{24}	(Hs10) 2^{23} - 2^{16}	(Hs9) 2^{15} - 2^{8}	(Hs8) 2^{7} - 2^{0}
00	X15 - X8	X15 - X8	X15 - X8	X15 - X8	X7 - X0	X7 - X0	X7 - X0	X7 - X0
01								
02								
03								
04								
05								
06								
07								
10								
11								
12								
13								
14								
15								
16								
17								
20								
21								
22								
23								
24								
25								
26								
27								
30								
31								
32								
33								
34								
35								
36								
37	X15 - X8	X15 - X8	X15 - X8	X15 - X8	X7 - X0	X7 - X0	X7 - X0	X7 - X0
40	X35 - X28	X35 - X28	X35 - X28	X35 - X28	X27 - X20	X27 - X20	X27 - X20	X27 - X20
41								
42								
43								
44								
45								
46								
47								
50								
51								
52								
53								
54								
55								
56								
57								
60								
61								
62								
63								
64								
65								
66								
67								
70								
71								
72								
73								
74								
75								
76								
77	X35 - X28	X35 - X28	X35 - X28	X35 - X28	X27 - X20	X27 - X20	X27 - X20	X27 - X20

Hardware Trng.
A-6684 J.E.S.

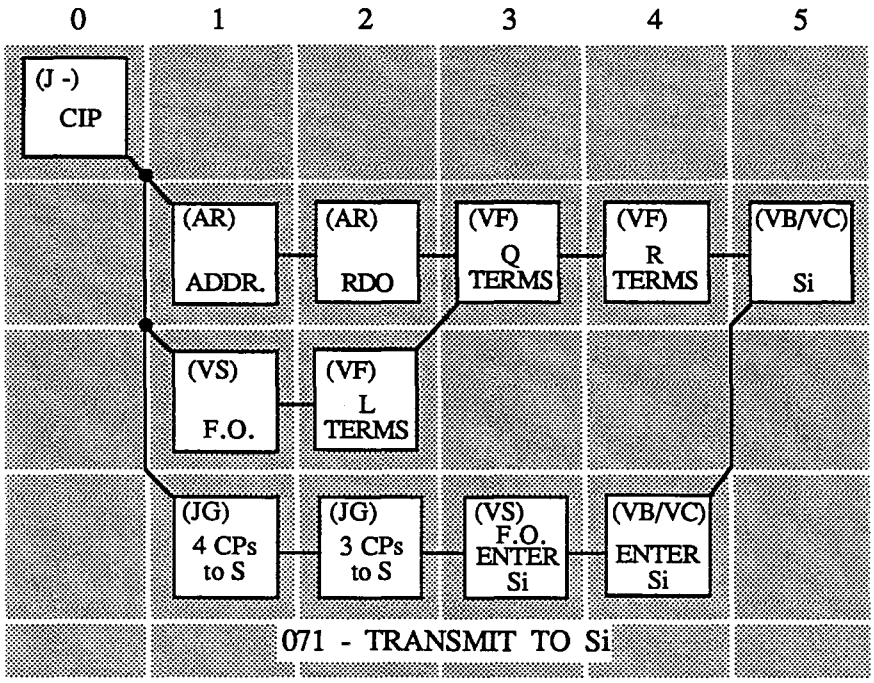
CRAY Y-MP T REGISTERS BIT ASSOCIATION
(036ijk, 037ijk)

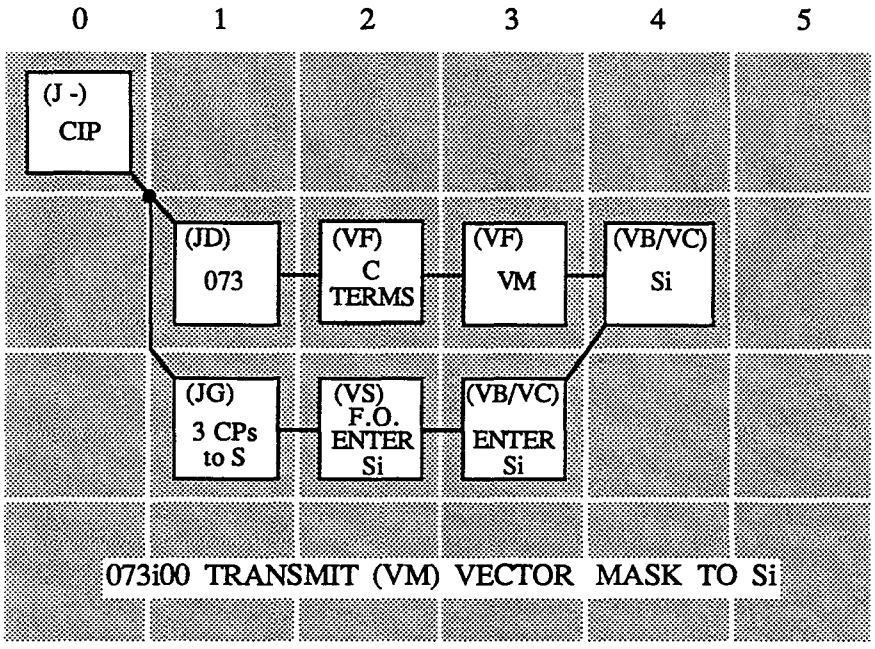




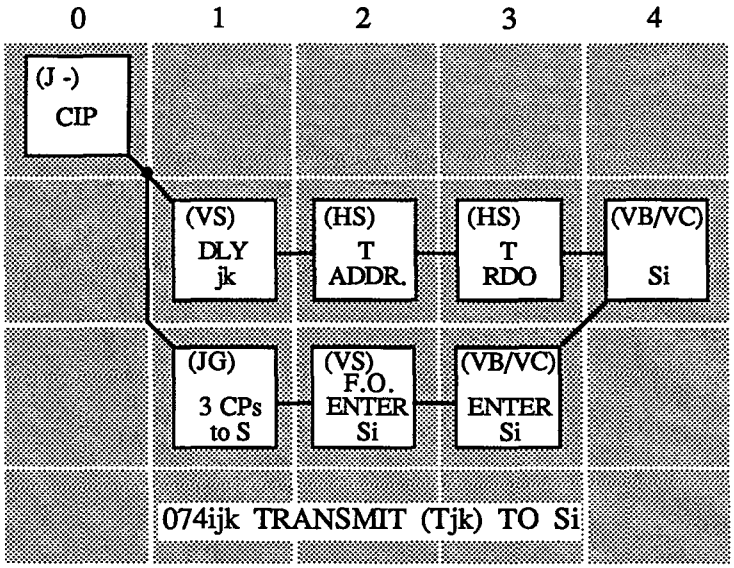
A-8262

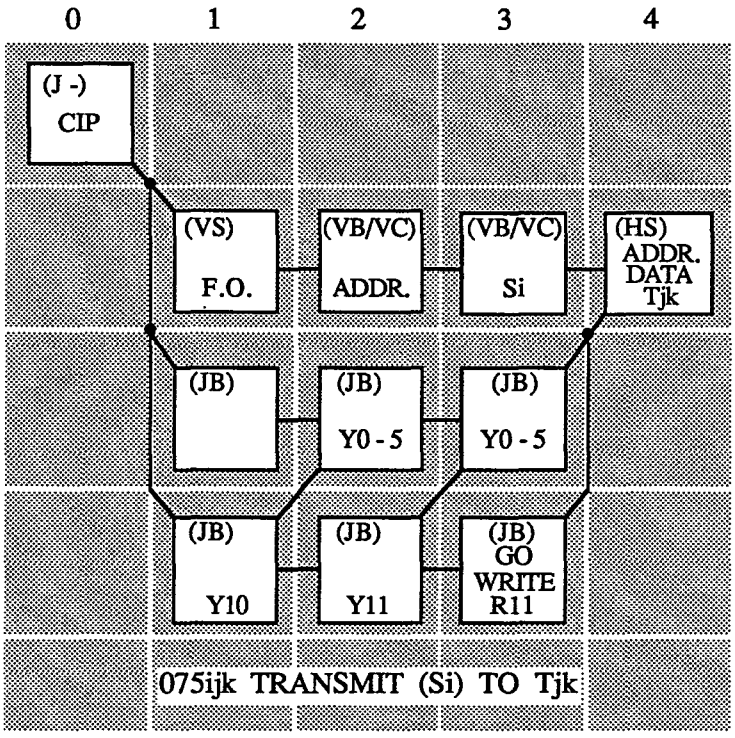
CRAY Y-MP SCALAR INSTRUCTIONS


0
1
2
3
4
5
(J -)
CIP
(AR)
ADDR.
(AR)
RDO
(VF)
Q
TERMS
(VF)
R
TERMS
(VB/VC)
Si
(VS)
F.O.
(VF)
L
TERMS
(JG)
4 CPs
to S
(JG)
3 CPs
to S
(VS)
F.O.
ENTER
Si
(VB/VC)
ENTER
Si
071 - TRANSMIT TO Si


0
1
2
3
4
5
(J -)
CIP
(JD)
073
(VF)
C
TERMS
(VF)
VM
(VB/VC)
Si
(JG)
3 CPs
to S
(VS)
F.O.
ENTER
Si
(VB/VC)
ENTER
Si
073i00 TRANSMIT (VM) VECTOR MASK TO Si

A-8261





A-8264

CRAY Y-MP SCALAR INSTRUCTIONS

SCALAR LOGICAL FUNCTIONAL UNIT INSTRUCTIONS

044ijk - Logical Product of (Sj) and (Sk) to Si

044ij0 - Sign Bit of (Sj) to Si ($Sj \neq 0$)

045ijk - Logical Product of (Sj) and Ones Complement of (Sk) to Si

045ij0 - (Sj) With Sign Bit Cleared to Si

046ijk - Logical Difference of (Sj) and (Sk) to Si ($Sj \neq 0$)

046ij0 - Toggle Sign Bit of Sj, Then Enter Into Si

047ijk - Logical Equivalence of (Sk) and (Sj) to Si

047i0k - Transmit Ones Complement of (Sk) to Si

047ij0 - Logical Equivalence of (Sj) and Sign Bit to Si ($Sj \neq 0$)

047i00 - Enter Ones Complement of Sign Bit Into Si

050ijk - Logical Product of (Si) and (Sk) Complement OR'd With Logical Product of (Sj) and (Sk)

050ij0 - Scalar Merge of (Si) and Sign Bit of (Sj) to Si

051ijk - Logical Sum of (Sj) and (Sk) to Si

051i0k - Transmit (Sk) to Si

051ij0 - Logical Sum of (Sj) and Sign Bit to Si ($Sj \neq 0$)

051i00 - Enter Sign Bit Into Si

Hardware Trng.
YM05/24A J.E.S.

SCALAR LOGICAL (044 - 051) INSTRUCTIONS ON (VB, VC) OPTIONS

Scalar Registers

TS = 0/6 ----------

-

```
M278    B0-1' = A0-1' A2-3' M84-85 ;T2 W0        .S0 REGISTER
        B10-11' = A0-1' A2-3' M84-85 ;T2 W1      .S1
        B20-21' = A0-1' A2-3' M84-85 ;T2 W2      .S2
```

(B0 - B1) - Scalar register bits (N, N+1) of S0

(B70 - B71) - Scalar register bits (N, N+1) of S7

Scalar Si, Sj, Sk Read Select

TS = 10/17 ----------

-

```
M259    C0 = MUX (Z999,B10,B20,B30):DCD(Q101,Q100)          .Sk SELECT
        C1 = MUX (Q107,B11,B21,B31):DCD(Q101,Q100)
        C2-3 = MUX (B40-41,B60-61,B50-51,B70-71):DCD(Q100,Q101)

        C10-11 = MUX (B0-1,B10-11,B20-21,B30-31):DCD(Q104,Q103)   .Sji SELECT
        C12-13 = MUX (B40-41,B60-61,B50-51,B70-71):DCD(Q103,Q104)

        C20-21 = MUX (B0-1,B10-11,B20-21,B30-31):DCD(S101,S100)   Si FOR LOG
        C22-23 = MUX (B40-41,B60-61,B50-51,B70-71):DCD(S100,S101)
```

TS = 20/27 ----------

-

```
X251    #,D0-1 = MUX (C0-1,C2-3):DCD (Q2)          .Sk
X254    #,D10-11 = MUX (C10-11 U3',C12-13):DCD (Q5)    .Sji
```

Hardware Trng.
YM05/25C J.E.S.

Scalar Select Read Path or Bypass

TS = 39/46 ----------

```
M259    A0 = I0 V0 S4'+I2 V0 S4+I4 V1 S4'+I6 V1 S4          .Si DATA
        A1 = I1 V0 S4'+I3 V0 S4+I5 V1 S4'+I7 V1 S4
M202    A2-3' = H10-11 H12-13

M259    A10 = I2 V10 S4+I0 V10 S4'+I6 V11 S4+I4 V11 S4'     .Sk BYPASS
        A11 = I3 V10 S4+I1 V10 S4'+I7 V11 S4+I5 V11 S4'
M254    A12-13' = MUX(D0-1',H10-11 H12-13):DCD(V12)

M259    A20 = I0 V20 S4'+I2 V20 S4+I4 V21 S4'+I6 V21 S4     .Sji BYPASS
        A21 = I1 V20 S4'+I3 V20 S4+15 V21 S4'+I7 V21 S4
M254    A22-23'=MUX(D10-11',H10-11 H12-13):DCD(V22)
```

(D0 - 1) - Read path of Sk
(D10 - 11) - Read path of Sji

(H10 - 11) and (H12 - 13) are Bypass from SFU or Floating functional unit

Operand Selection Sk, Sji

TS = 0/6 ----------

```
M293    E0-1' = A10-11'A12-13';T2                           .Sk OPERAND CP 3
        E2-3' = A20-21'A22-23';T2                           .Sji OPERAND CP 3

        E10-12 = I57-59;T2                                  .h DESIGNATOR CP 3
```

Hardware Trng.
YM05/26A J.E.S.

TS = 10/17 ----------

-

```
M221    F0-1 = E2-3'E0-1' \E2-3' E10        .BIT ENABLE
        F2-3 = E2-3 E0-1 \E2-3 E10          .BIT CARRY
        F4-5 = E12' E0-1' \E12'E10          .Sk FOR 050,051
M202    F6' = K13'K14'                      .SELECT T OR Si TO MEM
        F7 = E32
```

TS = 20/27 ----------

-

```
M254    G0-1 = MUX(F2-3,F0-1'F2-3'):DCD (E11)          .S LOGICAL
        G2-3' = MUX(C20-21' E10',C22-23' E10'):DCD(S3) .Si FOR 050
X204    #,G4 = # # F0' F2'                             .SUM 0 CP 3
X221    #,G5 = F2 \F1' F3'                             .    1
X201    #,G6-7 = E30-31
```

Select the Results for Scalar Register

TS = 32/38 ----------

-

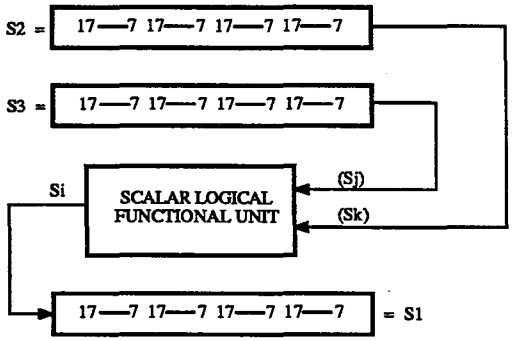
```
M254    H0-1 = MUX(I10-11 V2,I8-9 V2):DCD(S4')         .Ak,CON,VM + ST,SM,RT
M314    H2-3 = G2-3 F4-5 V3 S4+G0-1 V3 S4+I16-17 V3 S4' .T + LOG
M254    H4-5 = MUX(I14-15 V4,I13-14 V4):DCD(F7)        .SHIFT
M225    H6 = (I18+V5') \ (G6+V5') '                    .S ADD CP 5
        H7 = (I18 G6+V5') \ (G7+V5') '
M254    H8-9 = MUX(I16-17 K13,E2-3):DCD(K14)           .T OR Si TO MEM

At      H10-11 = H0-1'H2-3'
        H12-13 = H4-5'H6-7
```

The H terms are then sent to the A terms where the results are gated into the B term for S0 - S7.

Hardware Trng.
YM05/27A J.E.S.

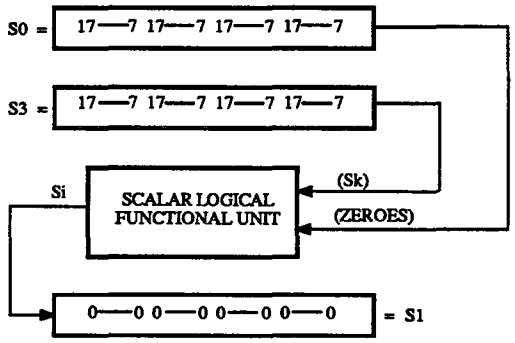
TYPICAL FUNCTIONAL UNIT OPERATION

FOR AN 044123 INSTRUCTION:



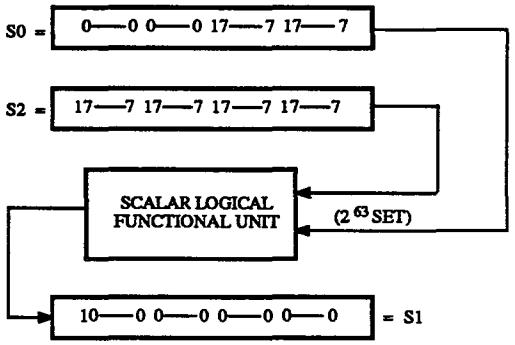
SPECIAL SYNTAX FORM Sj = 0

FOR AN 044103 INSTRUCTION:



SPECIAL SYNTAX FORM Sk = 0

FOR AN 044120 INSTRUCTION:

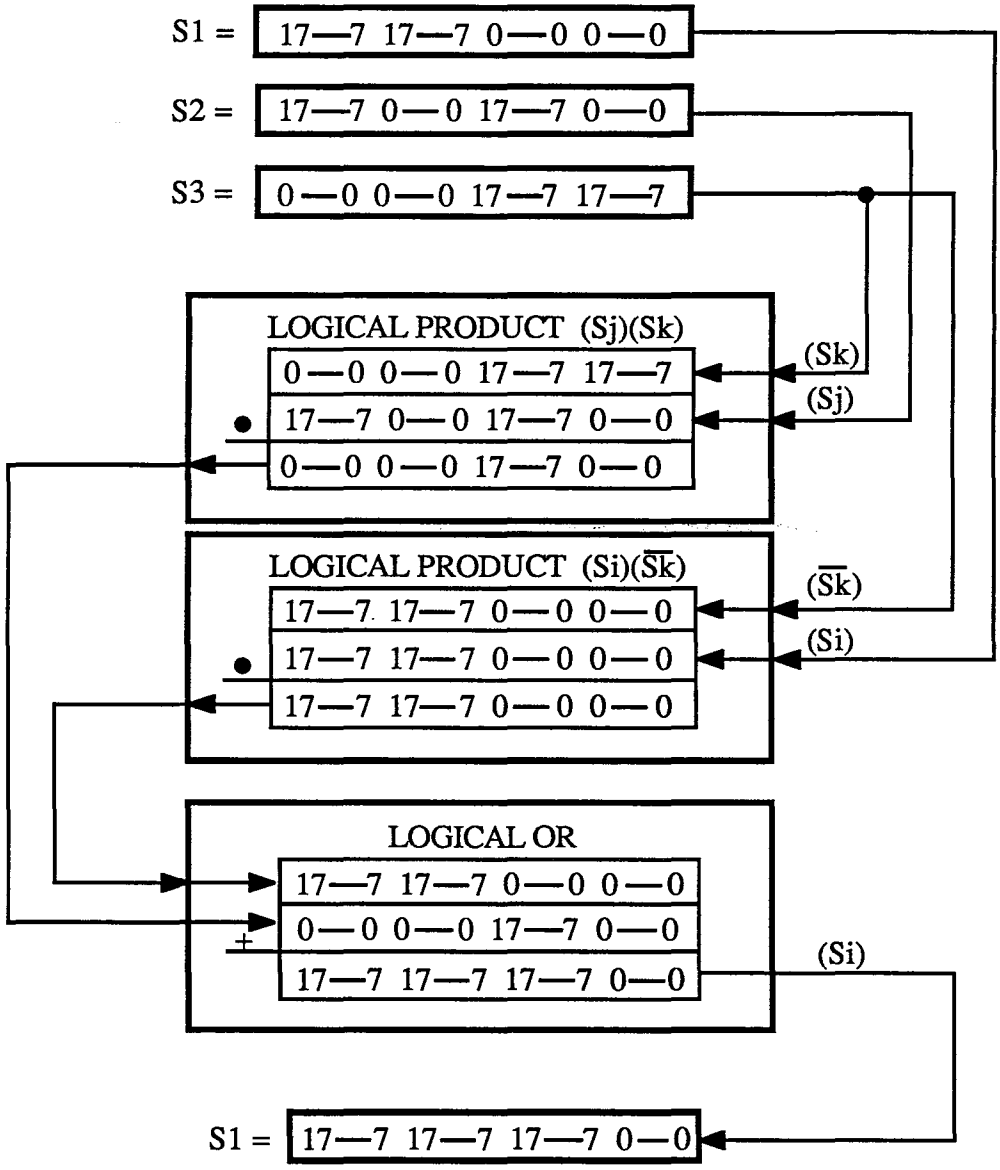


REGISTER	VALUE
Si i = 0	(S0)
Sj j = 0	0_{63}
Sk k = 0	2^{63}

Hardware Tmg. A-5304A J.E.S.

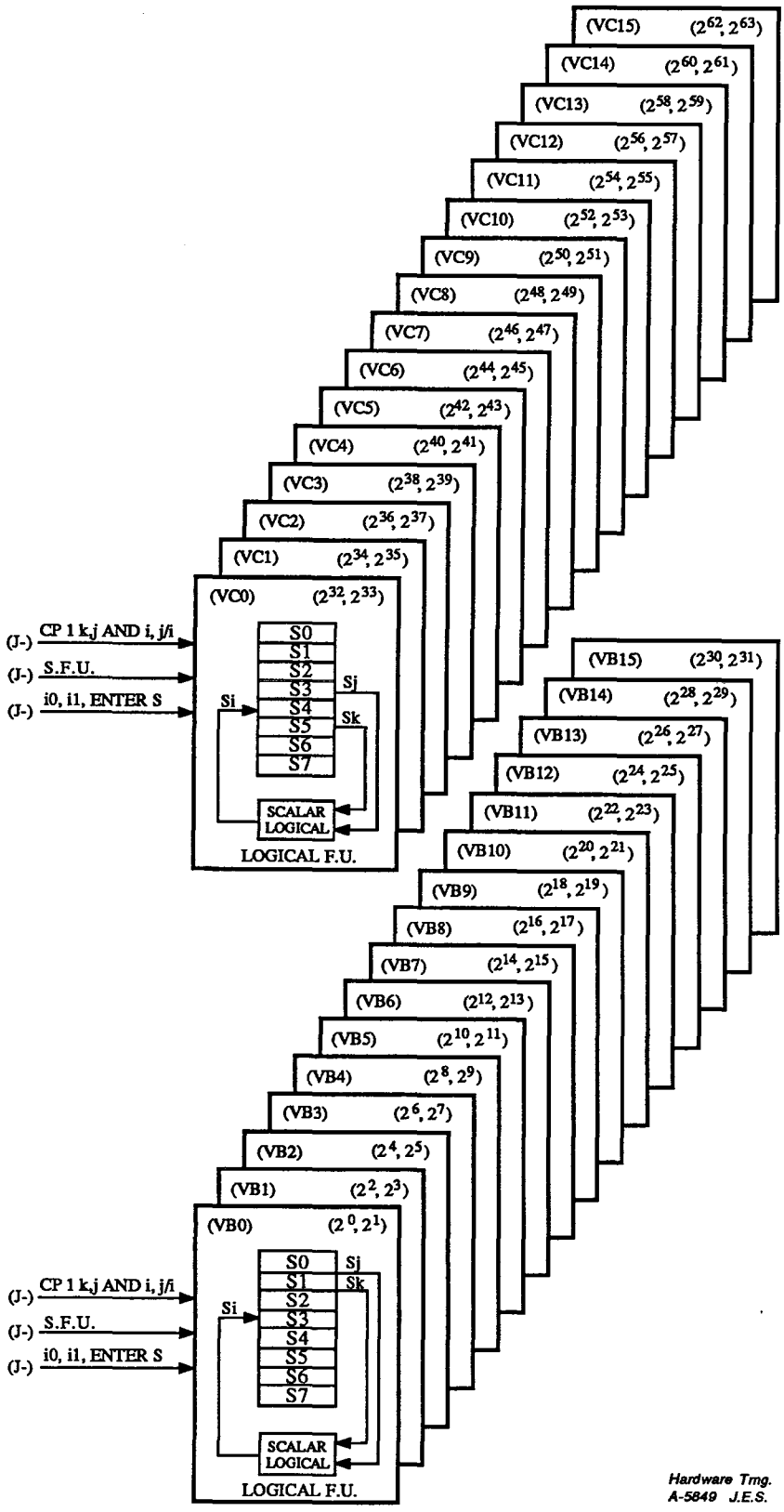
CRAY Y-MP SCALAR LOGICAL FUNCTIONAL UNIT

FOR 050123 INSTRUCTION: Si = (Sj)(Sk) + (Si)($\overline{Sk}$)

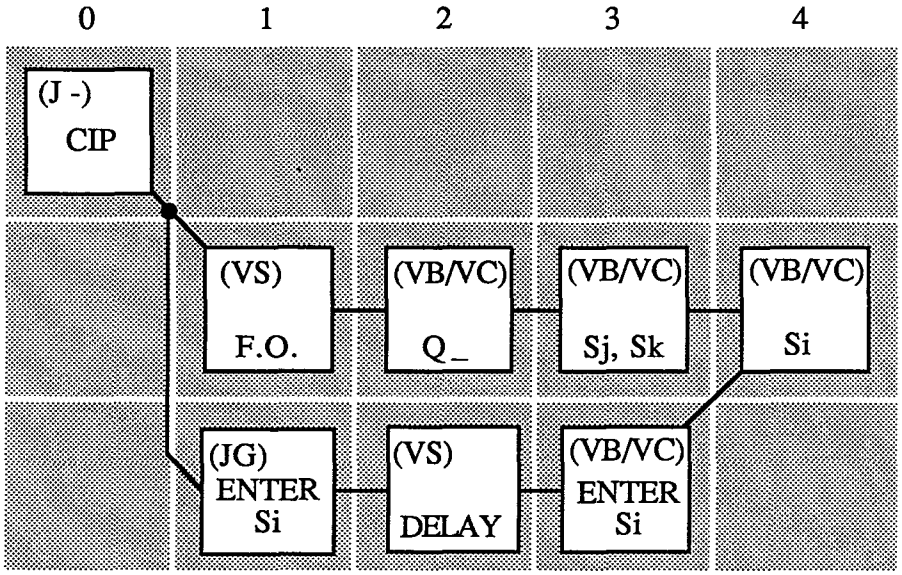


Hardware Trng.
A-5303A J.E.S.

CRAY Y-MP (050ijk) INSTRUCTION OPERATION



CRAY Y-MP SCALAR LOGICAL BLOCK DIAGRAM



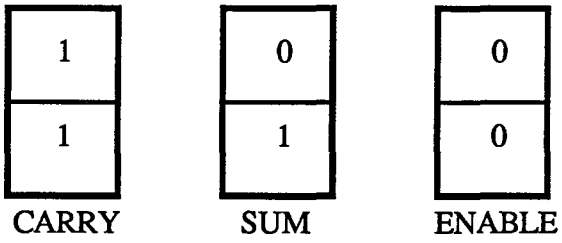
044 - 051 - SCALAR LOGICAL A-7730A

CRAY Y-MP SCALAR LOGICAL

SCALAR ADD FUNCTIONAL UNIT

The Scalar Add functional unit adds together two 64-bit operands, using a 2-bit lookahead add. To utilize the Scalar Add functional unit an 060ijk instruction or an 061ijk instruction is used. The 060ijk instruction is the Add instruction, determined by h bit (2^0) equaling a (0), while the 061ijk instruction is the Subtract instruction, determined by h bit (2^0) equaling a (1). The Scalar Add functional unit takes 3 CPs from the time the operands begin summing to the time the results are complete.

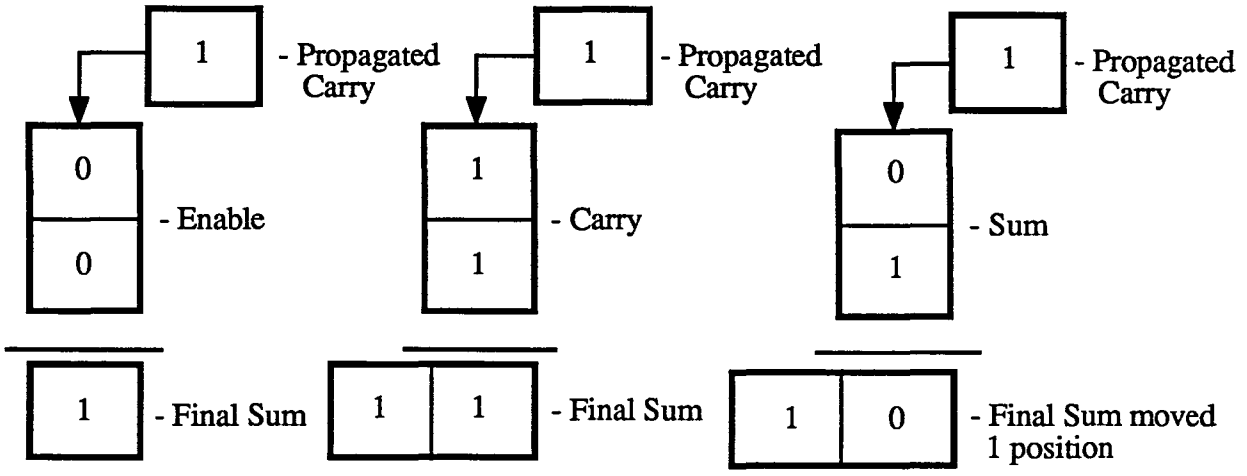
To perform the Scalar Add, the (Sj) and (Sk) operands are compared to determine whether each bit position generates a Carry, Enable, or a Sum. The Carry is when the 2 bits of (Sj) and (Sk) for a particular bit position are equal to ones. The Enable is when the 2 bits of (Sj) and (Sk) are equal to zeros; and the Sum is when the 2 bits of (Sj) and (Sk) are different.



Once the two operands bit positions are defined, the Carry terms are looked at to see how far the Carry will propagate, first on the group level and then on the section level. An Enable term at a particular bit position will stop the Carry from propagating beyond that position.

After all the Carries have been propagated, the propagated carries are ANDed with the Enable, Sum, and Carry bits for each position.

A propagated Carry added to an Enable will set the corresponding bit position. A propagated Carry added to a Sum condition will set the next corresponding bit position.



Hardware Trng.
YM06/13A J.E.S.

To perform a subtract operation, or 061ijk instruction, two's complement arithmetic is used, wherein the (Sk) operand is complemented and one is added to the summation.

Example of 061 123 instruction: 7 - 1 = 6

S2 =	0——0	0——0	0——0	0——07	
S3 =	17——7	17——7	17——7	17——76	1's complement of Sk
+					
S1 =	0——0	0——0	0——0	0——05	
				+ 1	Plus 1 to complete 2's complement
	0——0	0——0	0——0	0——06	

Example of 061 123 instruction: 5 - 7 = -2

S2 =	0——0	0——0	0——0	0——05	
S3 =	17——7	17——7	17——7	17——70	1's complement of Sk
+					
	17——7	17——7	17——7	17——75	
				+ 1	Plus 1
	17——7	17——7	17——7	17——76	= -2 in 2's complement arithmetic

Hardware Trng.
YM06/14 J.E.S.

CRAY Y-MP SCALAR REGISTER FUNCTIONAL UNIT INSTRUCTIONS

SCALAR ADD/SUBTRACT

060ijk - Integer sum of (Sj) and (Sk) to Si

060i0k - Transmit (Sk) to Si

060ij0 - Integer sum of (2^{63}) and (Sj) to Si

061ijk - Integer difference of (Sj) minus (Sk) to Si

061i0k - Transmit the negative of (Sk) to Si

061ij0 - Integer difference of (2^{63}) and (Sj) to Si

CAL FORMAT

Si	Sj + Sk	Integer sum (Sj) and (Sk) to Si
Si	Sk	Transmit (Sk) to Si
Si	Sj + S0	Integer sum (2^{63}) and (Sj) to Si
Si	Sj - Sk	Integer difference of (Sj) and (Sk) to Si
Si	- Sk	Transmit negative of (Sk) to Si
Si	Sj - S0	Integer difference of (Sj) and (2^{63}) to Si

SCALAR ADD FUNCTIONAL UNITS (OPTIONS INVOLVED)

(VB0 - VB15) Options

The (VB) options handle the lower 32 bits of the (Sj) and (Sk) operands. Each (VB) handles 2 bits each of (Sj) and (Sk), per the bit positions, and identifies the bits as Enables, Carries, or Sums. The Enable and Carry are partially summed on a group level: a group being 2 bits in a row per each (VB) option. The partial group summations are sent to the (VG) options for a Section Summation. When the Section Summation is complete, the (VG) sends the partial summation back to the (VB) options to complete the summation and store the results into the Scalar register, which is also on the (VB) option.

(VC0 - VC15) Options

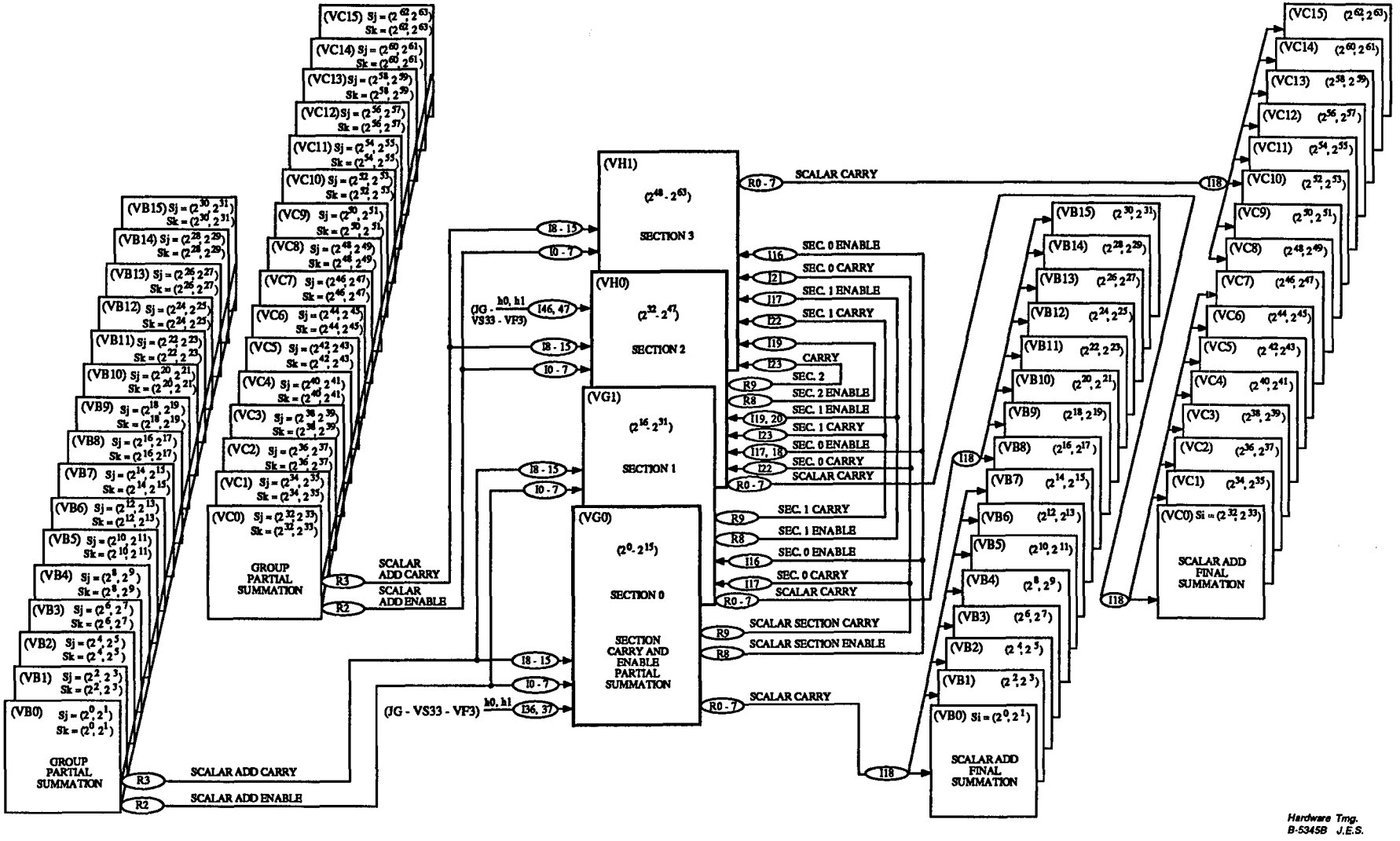
The (VC) options are identical to the (VB) options except they contain the upper 32 bits of the (Sk), (Sj) operands. The (VC) options send their partial group summations to the (VH) options.

(VG0, VG1) Options

The (VG0) option handles the lower 16 bits of the summation - bits (2^0 - 2^{15}), or Section 0. The (VG1) option handles the next 16 bits of the summation - bits (2^{16} - 2^{31}), or Section 1. The (VG) options are sent the Group Carry and Group Enables from the (VB) options. They then check to see how far the Carries will propagate on a section level. The (VG0) will output a Section Carry or Section Enable to the (VG1) and (VH0 - 1) options in the higher bit positions. Finally, only the propagated Carries, called Scalar Carries, are sent back to the (VB) options for the final summation.

(VH0, VH1) Options

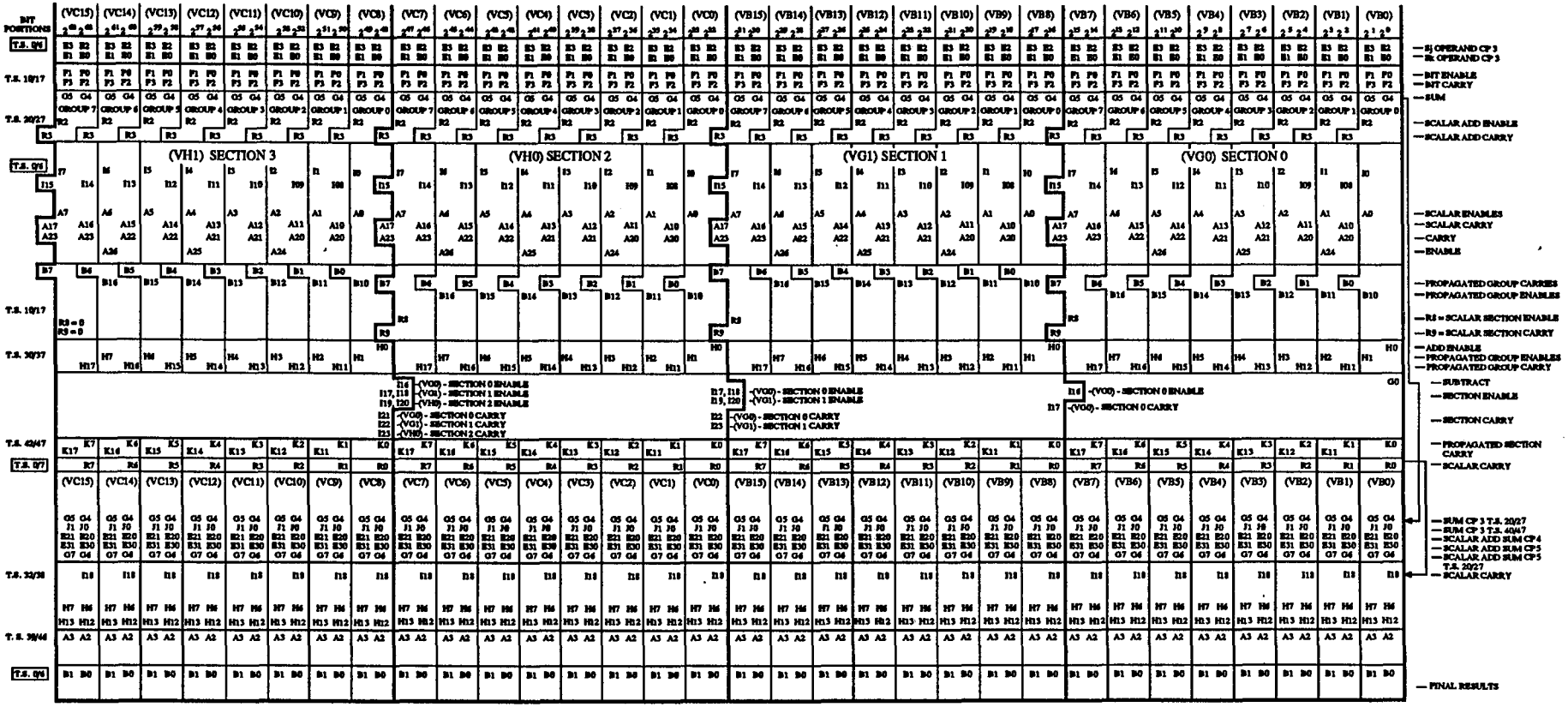
The (VH) options are identical to the (VG) options except the (VH) options have more inputs for Section Enables and Section Carries from the lower bit sections. The (VH) options receive the Group Carries and Enables from the (VC) options, and send the propagated Carries, called Scalar Carries, back to the (VC) options for the final summation.



CRAY Y-MP SCALAR REGISTER ADD BLOCK DIAGRAM

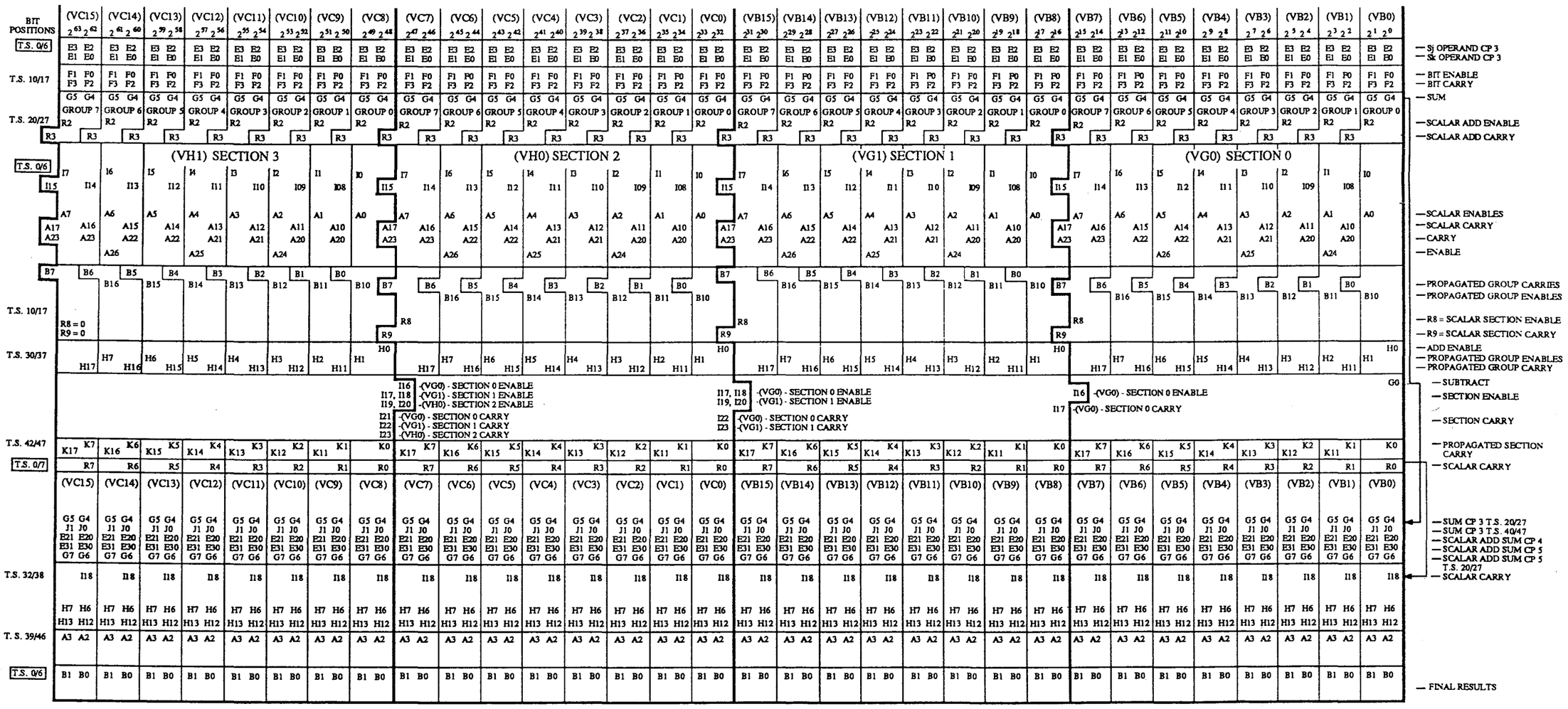
CRAY PROPRIETARY

CRAY PROPRIETARY



Hardware Trng.
A-5344D J.E.S.

CRAY Y-MP SCALAR ADD FUNCTIONAL UNIT



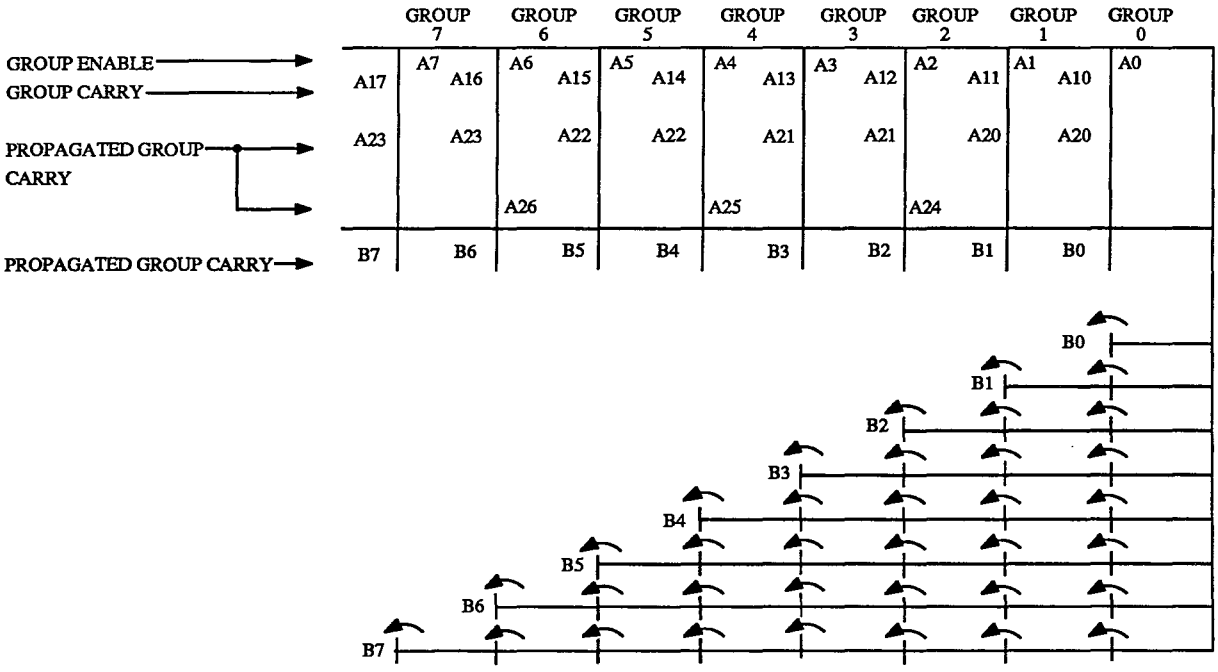
Hardware Tmg.
A-5344D J.E.S.

CRAY Y-MP SCALAR ADD FUNCTIONAL UNIT

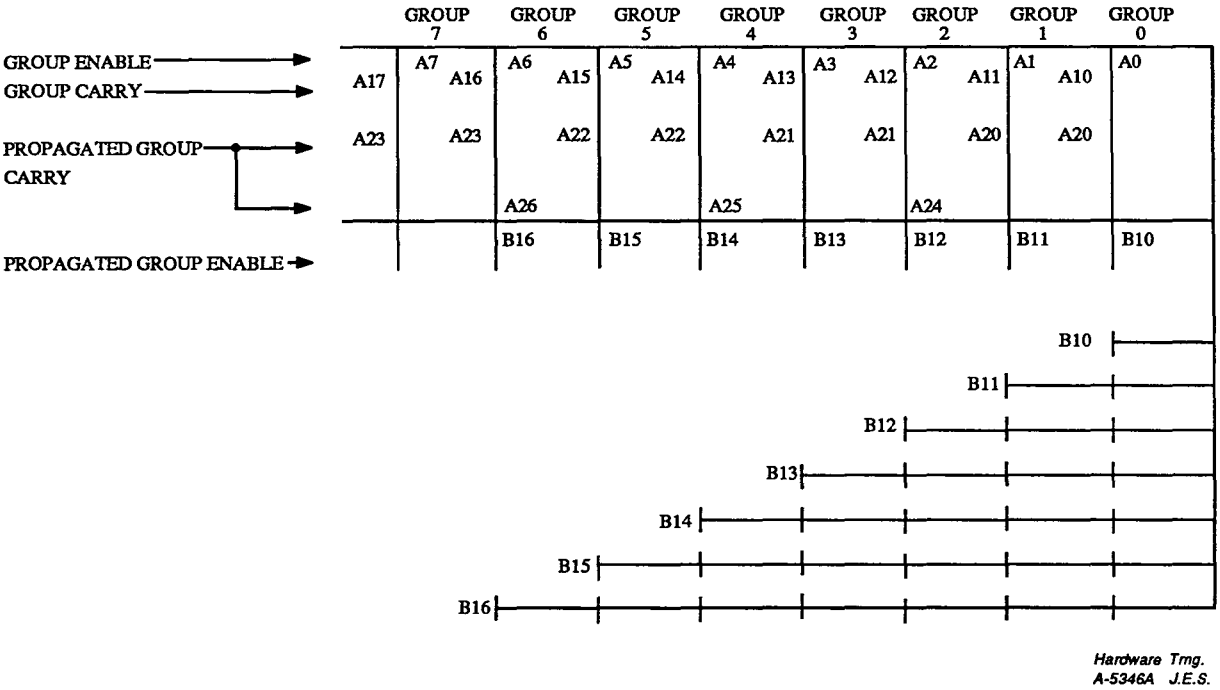
CRAY PROPRIETARY

0	0	1
0	1	1
ENABLE	SUM	CARRY

PROPAGATED GROUP CARRY



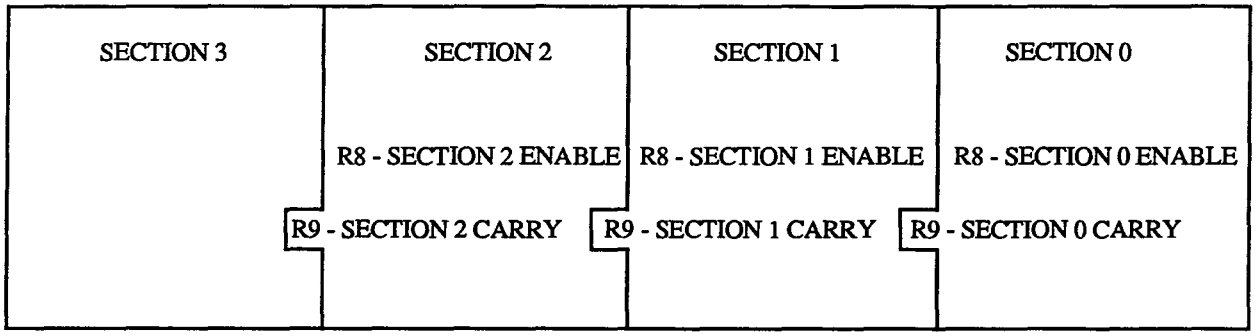
PROPAGATED GROUP ENABLE



Hardware Tmg.
A-5346A J.E.S.

CRAY Y-MP SCALAR ADD
2 BIT LOOK-AHEAD ADD

PROPAGATED SECTION ENABLE OR CARRIES



SECTION 2 CARRY = SUBTRACT • SECTION 0 ENABLE • SECTION 1 ENABLE • SECTION 2 ENABLE

OR

SECTION 0 CARRY • SECTION 1 ENABLE • SECTION 2 ENABLE

OR

SECTION 1 CARRY • SECTION 2 ENABLE

OR

SECTION 2 CARRY

SECTION 1 CARRY = SUBTRACT • SECTION 0 ENABLE • SECTION 1 ENABLE

OR

SECTION 0 CARRY • SECTION 1 ENABLE

OR

SECTION 1 CARRY

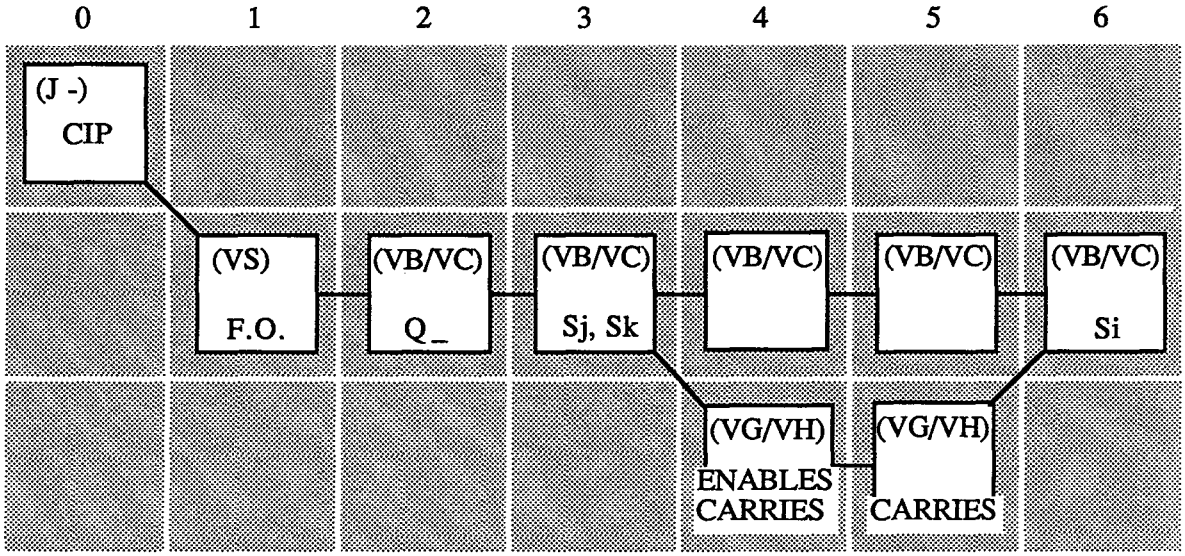
SECTION 0 CARRY = SUBTRACT • SECTION 0 ENABLE

OR

SECTION 0 CARRY

Hardware Trng.
A-5348A J.E.S.

CRAY Y-MP SCALAR ADD



060ijk - INTEGER SUM OF (Sj) AND (Sk) TO Si
061ijk - INTEGER DIFFERENCE OF (Sj) LESS (Sk) TO Si

A-7729

CRAY Y-MP SCALAR INTEGER ADD

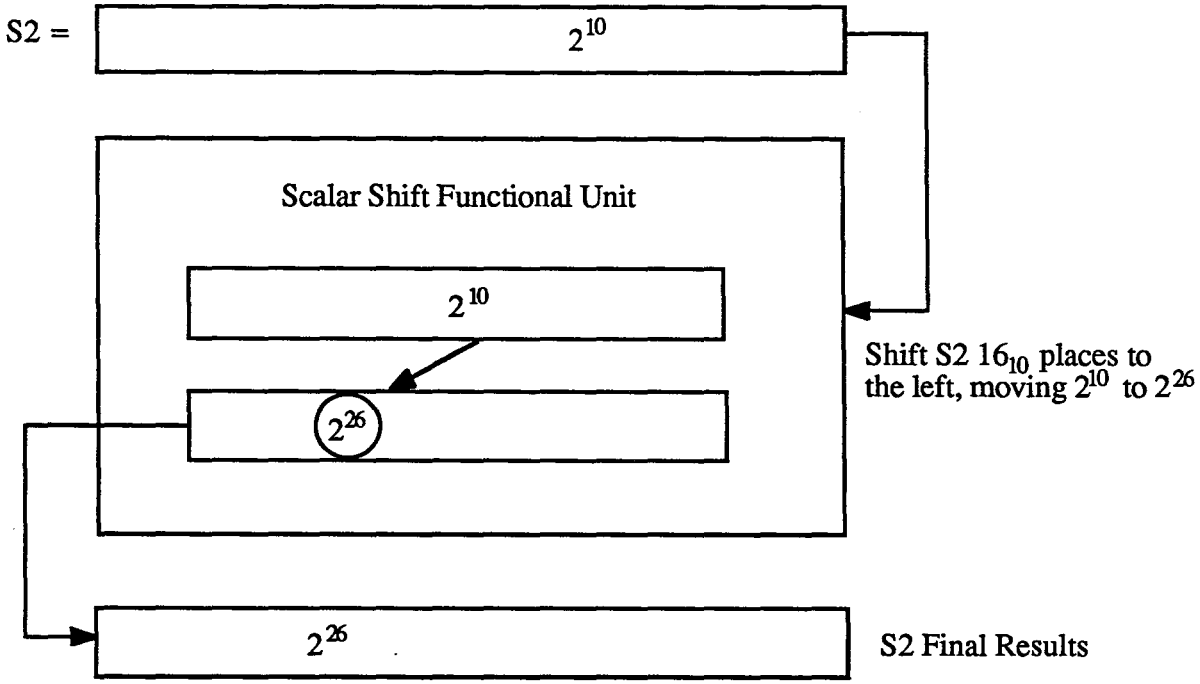
SCALAR SHIFT FUNCTIONAL UNIT

The Scalar Shift functional unit will perform Scalar Single or Scalar Double shifts in the left or right direction. To perform a single shift, the (Si) operand is moved in the left direction for a left or a right shift, but with the right shift the count is in twos complement form. For a single shift, the (Si) operand is shifted by the number of places specified by the (jk) field of the instruction.

For a Double Shift the constant of the Ak register of the instruction contains the shift count. Only the lower 7 bits of (Ak) are used to specify a valid shift; any bits set in (Ak) bit position (2^{31}-2^{7}) would cause the result register Si to be zeroed. For the Double Shift the (Sj) and (Si) operands are concatenated to form a 128-bit register, and so setting all 7 bits in (Ak) would shift the 128-bit register 127 places.

Scalar Single Left Shift

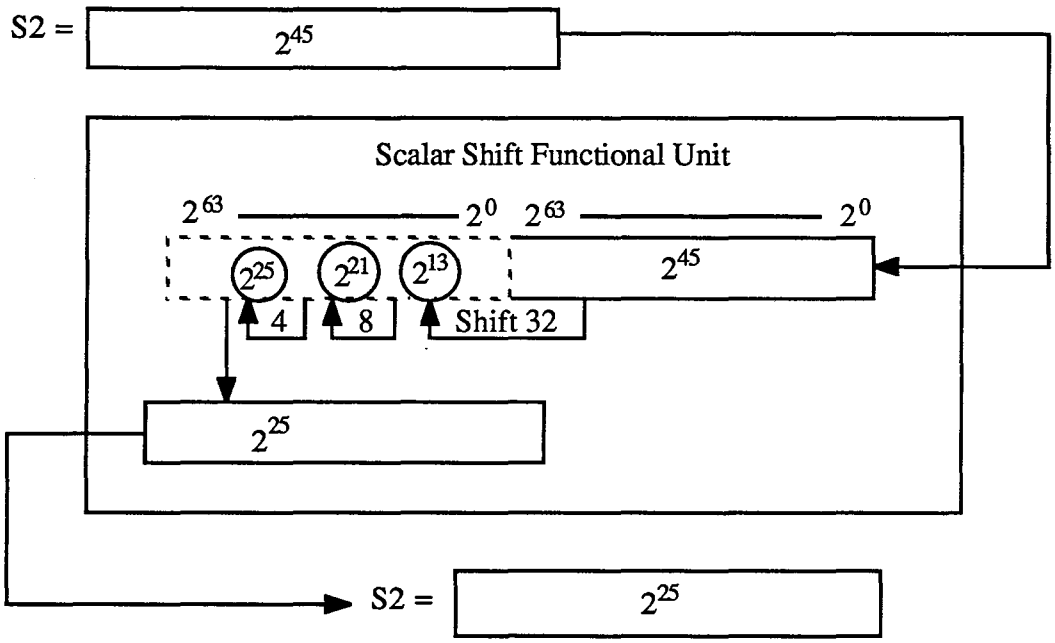
The following figure demonstrates how a single left shift is performed for an 054220 instruction with data bit 2^{10} set.



Hardware Trng.
YM07/13A J.E.S.

Scalar Single Right Shift

The following figure demonstrates how a Single Right shift is performed, while using left shifts and a twos complement shift count. Let's use a 055254 instruction to shift data bit 2^{45} to the right $20_{10} = 24_8$ places. Notice that the 055254 (jk) field contains 54_8, which is the twos complement of 24_8, causing S2 to be shifted to the left 54_8 places to set 2^{25} of the result.

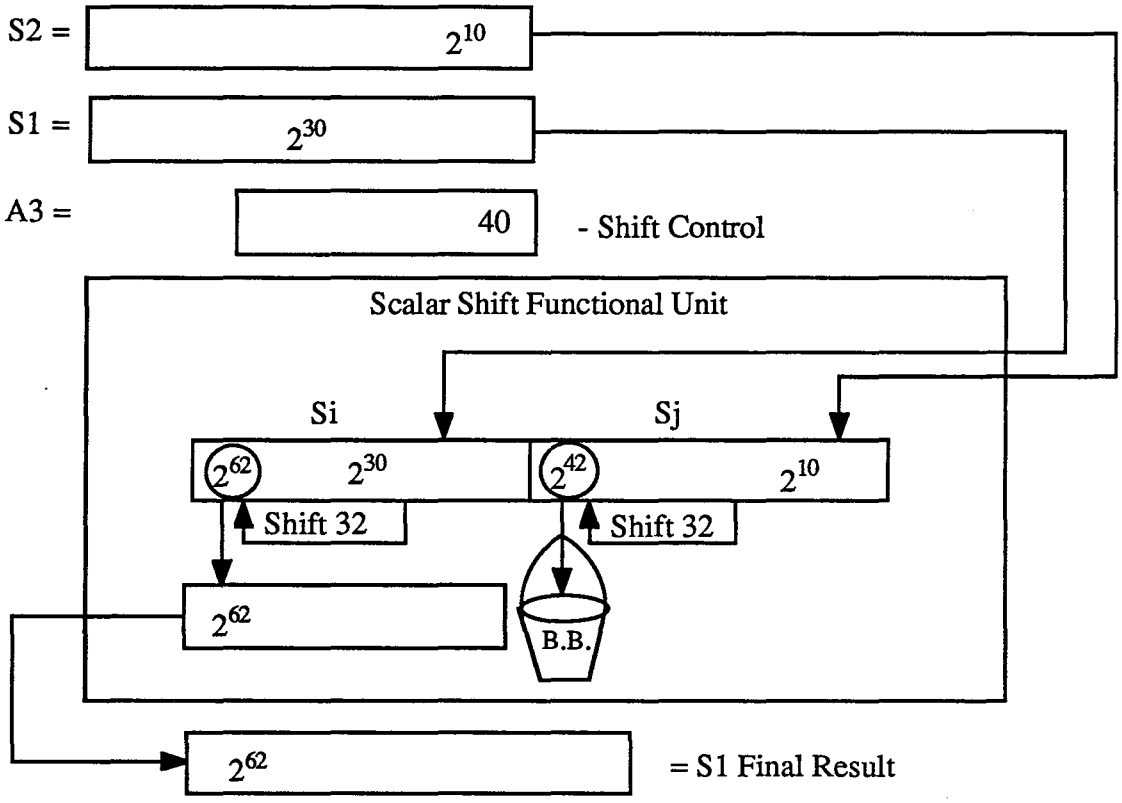


To shift S2, 54_8 places to the left, note that 54_8, or 101100_2, can be broken down into shifts of 32, 8, and 4. Shifting bit 2^{45} to the left 32 places would put it at bit (2^{13}) in the dotted box, shifting (2^{13}) eight more places to the left would put it at (2^{21}) in the dotted box, and finally, shifting (2^{21}) four more places to the left would put (2^{21}) at bit position (2^{25}), which is the final result. It is easy to see that Right shifting bit 2^{45}, 20_{10} places will move the bit to position 2^{25}.

Scalar Double Left Shift

Double shifts are the same as Single shifts except that they use 128-bit registers instead of 64-bit registers.

Shown is a Double Left shift using a 056123 instruction with A3 = 40_8 , or 32_{10} , S1 having bit 2^{30} set, and S2 having bit 2^{10} set. When performing a Double Left shift, since the contents of (Sj) are moved into (Si), the two registers are positioned as shown, with (Si) ahead of (Sj).

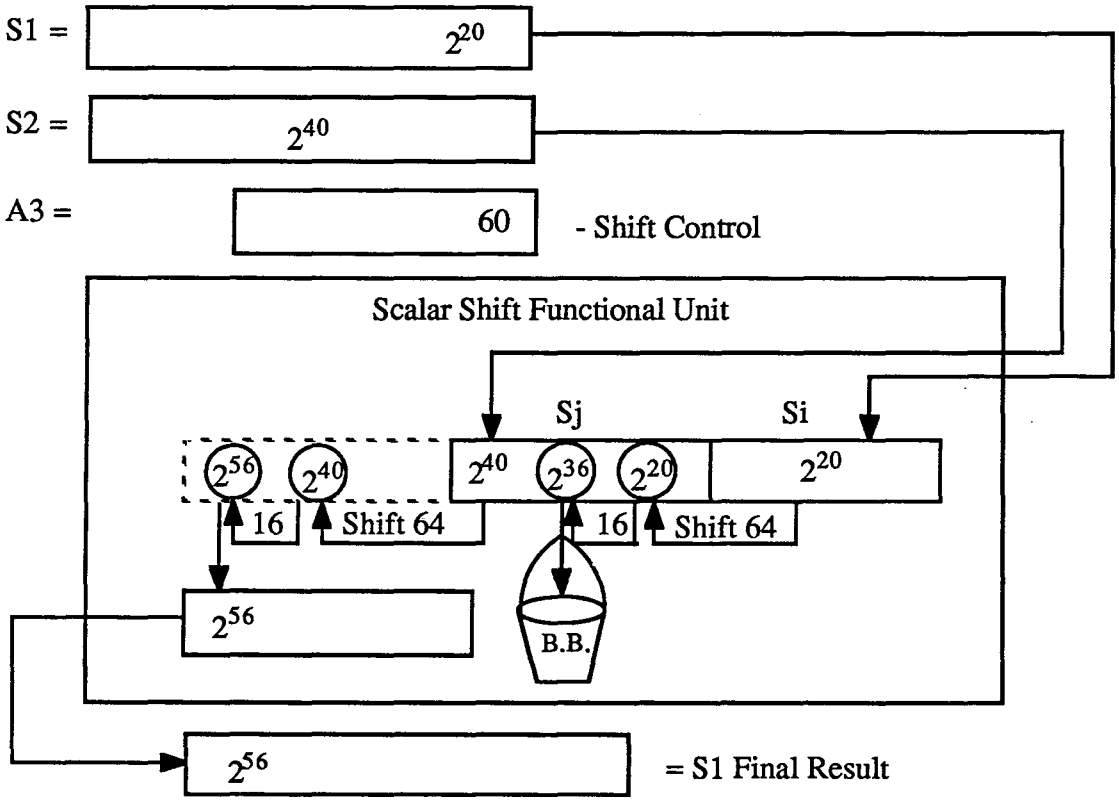


Shifting Si and Sj to the left 32 places puts 2^{30} of S1 at (2^{62}), and 2^{10} of S2 at (2^{42}). Since (2^{42}) of S2 did not make it to the result register S1, it is lost to the "Bit Bucket". The result bit 2^{62} is then sent to the Si or S1 register. The Sj or S2 register is left unaltered.

Hardware Trng.
YM07/15 J.E.S.

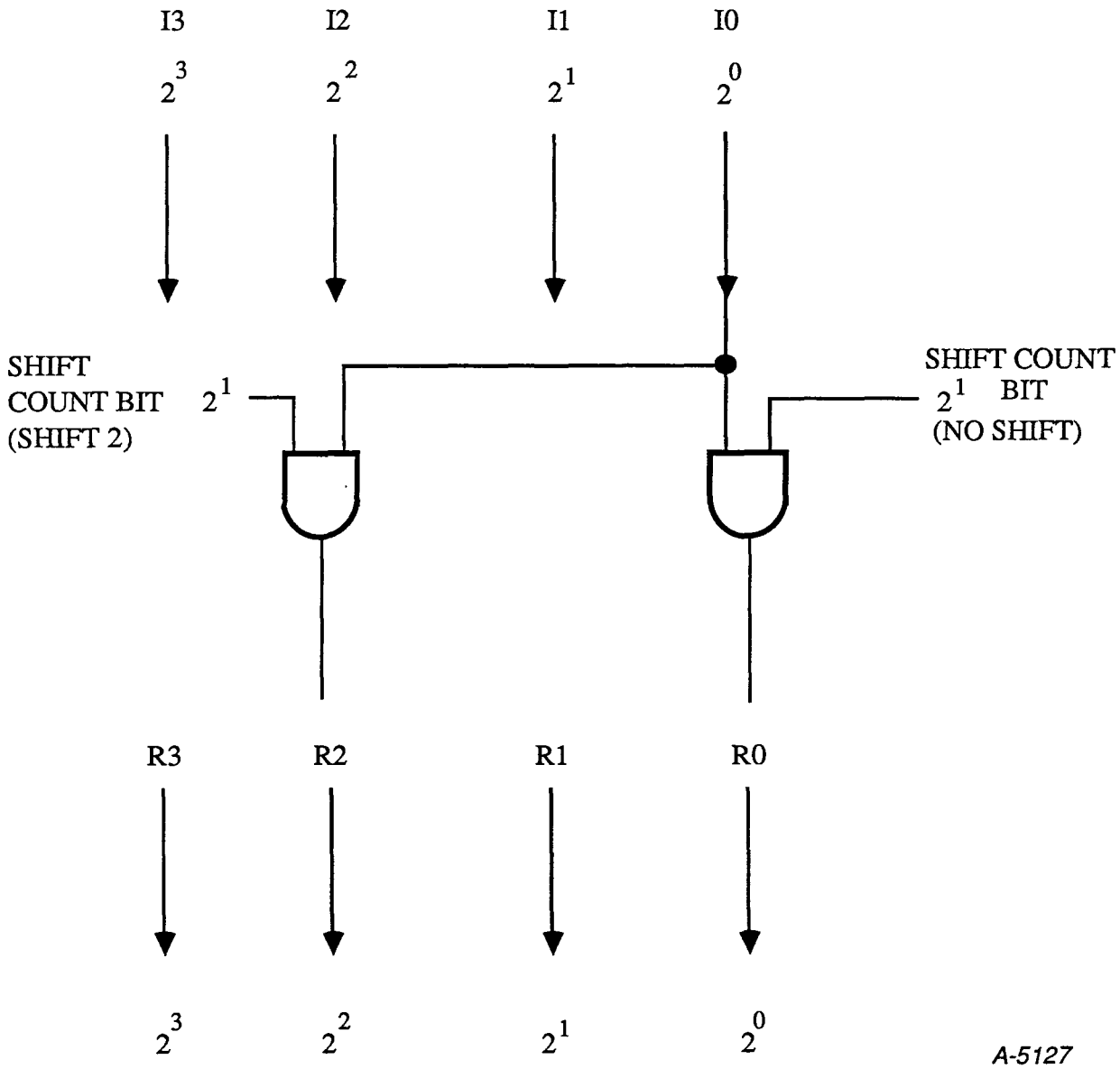
Scalar Double Right Shift

To perform a Scalar Double Right shift, an 057 instruction is used. Demonstrated below is an 057123 instruction with the indicated parameters.



To Right shift (Sj) and (Si) using left shifts, the twos complement is first performed on A3, which currently equals 60_8 or 48_{10}. Since the twos complement is 120_8 or 1010000_2 or 80_{10}, the required shift can be accomplished through successive shifts of 64_{10} and 16_{10}. A left shift of 64_{10} would move 2^{40} of S2 to (2^{40}) inside the dotted box and 2^{20} of S1 to (2^{20}) of S2. A further move of 16_{10} places to the left would put (2^{40}) inside the dotted box at (2^{56}) and (2^{20}) of S2 at (2^{36}). Because bit (2^{36}) did not make it into the dotted box or result register, it is lost to the "Bit Bucket", while bit (2^{56}) is sent to the Final Result.

Hardware Trng.
YM07/16A J.E.S.



CRAY Y-MP BASIC SHIFT OPERATION

CRAY Y-MP SCALAR SHIFT FUNCTIONAL UNIT INSTRUCTIONS

Instructions Single Shift Right/Left

052ijk	Shift (Si) left exp = jk places to S0
053ijk	Shift (Si) right exp = 100_8 - jk places to S0
054ijk	Shift (Si) left exp = jk places to Si
055ijk	Shift (Si) right exp = 100_8 - jk places to Si

Instructions Double Shift Right/Left

056ijk	Shift (Si) and (Sj) left (Ak) places to Si
056ij0	Shift (Si) and (Sj) left one place to Si
056i0k	Shift (Si) left (Ak) places to Si
057ijk	Shift (Sj) and (Si) right (Ak) places to Si
057ij0	Shift (Sj) and (Si) right one place to Si
057i0k	Shift (Si) right (Ak) places to Si

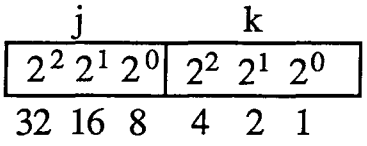
CAL Formats

052ijk	S0	Si < exp
053ijk	S0	Si > exp
054ijk	Si	Si < exp
055ijk	Si	Si > exp
056ijk	Si	Si, Sj < Ak
056ij0	Si	Si, Sj < 1
056i0k	Si	Si < Ak
057ijk	Si	Sj, Si > Ak
057ij0	Si	Sj, Si > 1
057i0k	Si	Si > Ak

Hardware Trng.
YM07/22 J.E.S.

Single Left Shift Instruction

The Shift Count is the (jk) field of the instruction.



052ijk Results to S0

054ijk Results to Si

Exercise #1: Write the instruction to Shift S2 Left 20_{10} places, putting the results into S0.

Answer:

1. 052ijk - Left shift instruction result goes to S0
2. jk field - Shift count

   $20_{10} = 24_8$ = jk field
3. 052224 - Final instruction

Exercise #2 Write the instruction to Shift S4 Left 35_{10} places, putting the results into S4.

Answer:

1. 054ijk - Left shift instruction results goes to Si
2. jk field - Shift count

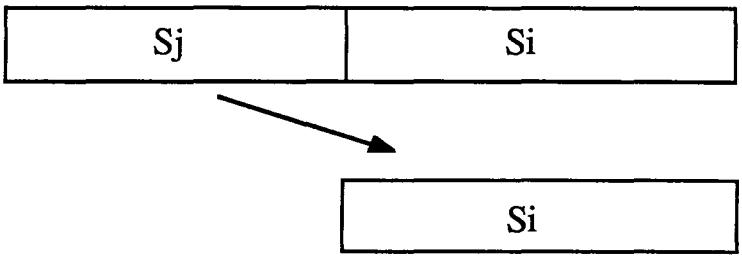
   $35_{10} = 43_8$
3. 054443 - Final instruction

CRAY Y-MP SINGLE LEFT SHIFT

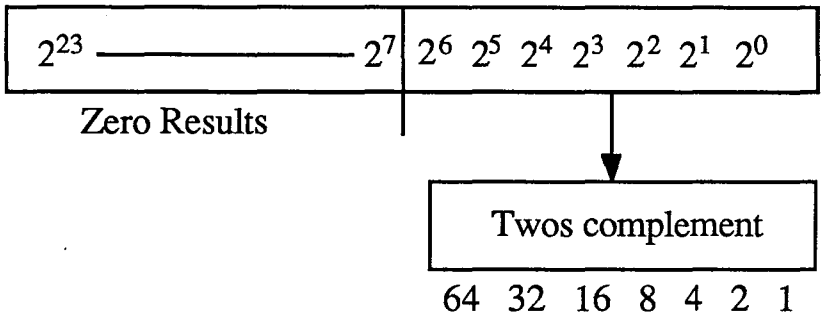
Hardware Trng.
YM07/18 J.E.S.

Double Right Shift Instruction

057ijk Shift (Sj) and (Si) Right by Ak places to Si



Ak - Contains the shift count, A Register bits (2^6 - 2^0) contains the valid shifts counts. Any bits (2^{23} - 2^7) set will zero the results of Si. Also, the hardware is provided to generate the twos complement of the shift count on (Ak) register bit(2^6 - 2^0)



On a Right Double shift, you always shift the contents of (Sj) into Si. This is done inside the Shift functional unit, also the twos complement of the shift count is done inside the functional unit.

Example: Write the instruction to Right Double shift S4 and S5 32_{10} places, results into S4.

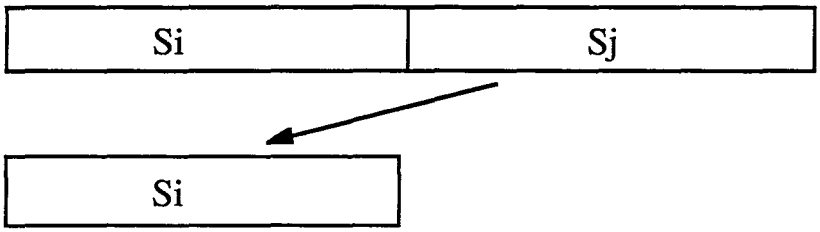
Answer: 057454 - Final Instruction where A4 = 40_8 hardware will generate a shift count of 140_8 inside the Functional Unit.

CRAY Y-MP DOUBLE SHIFT RIGHT

Hardware Trng.
YM07/19 J.E.S.

Double Left Shift Instruction

056ijk - Shift (Si) and (Sj) Left by (Ak) places to Si



Ak - Contains the shift count, A Register bits (2^6 - 2^0) contains the valid shift counts. Any bits (2^{23} - 2^7) set will zero the result of Si.

2^{23} ——— 2^7	2^6 2^5 2^4 2^3 2^2 2^1 2^0	= Ak
Zeroes Results	64 32 16 8 4 2 1	= Valid decimal shifts

On a Left Double shift you always shift the contents of (Sj) into Si. This is done inside the Shift functional unit.

Example: Write the instruction to Left Double shift S2 and S3 64_{10} places, result into S2.

Answer: 056234 - Final Instruction, where A4 = 100_8

CRAY Y-MP DOUBLE SHIFT LEFT

Single Right Shift Instruction

The Single Right shift count is the jk field of the instruction which must be in the twos complement form or a 100_8 minus the number of places to Right shift.

053ijk	Results to S0
055ijk	Results to Si

Exercise #1: Write the instruction to Shift S5 Right 10_{10} places, putting the results into S0.

Answer:

1. 053ijk - Right shift instruction results to S0

2. jk field - Shift count in twos complement equals 66_8.
   $10_{10} = 12_8 = 001010$
   twos compl. 110101
   $\underline{\quad +1}$
   $110110 = 66_8$
   or $100_8 - 12_8 = 66_8$

3. 053566 - Final instruction

Exercise #2: Write the instruction to Shift S7 Right 28_{10} places.

Answer:

1. 055ijk Right shift instruction results to Si.

2. jk field - Shift count in twos complement equals

   $28_{10} = 34_8 = 011100$
   twos complement = 100011
   $\underline{\quad +1}$
   $100100 = 44_8$

   or $100_8 - 34_8 = 44_8$

3. 055744 - Final Instruction

CRAY Y-MP SINGLE RIGHT SHIFT

SCALAR SHIFT (OPTIONS INVOLVED)

(VB0 - VB15, VC0 - VC15)

The (VB, VC) options supply the (VE) options with the (Sj) or (Sk) operands. These operands are readout of the Scalar registers which are also located on the (VB, VC) options. When performing a Scalar Double shift, the (Sj) and (Si) operands are readout of the (VB, VC) options at different clock periods. For an 056ijk instruction (Scalar Double Left shift), (Si) is readout at CP 3, while (Sj) is readout at CP 4. For an 057ijk instruction, (Scalar Double Right shift), (Sj) is readout at CP 3, and (Si) is readout at CP 4.

The (VB, VC) options receive the results of the shifts from the (VE) options. All the shifts of 64, 32, 16, 8, 4 or 2 are performed on the (VE) options while the shift of one is done on the inputs to the (VB, VC) options. To perform a shift of one the (VB, VC) options each receive 3 bits of the results, and shift those bits into the final bit positions.

(VE2, VE3) Options

The (VE2) option contains the even bits of the operands while the (VE3) option contains the odd bits; therefore, a shift of two on the (VE) option will cause a shift of four in the result register. The two (VE) options perform all the Scalar shifts, which include the Single Left, Single Right, Double Left, and Double Right.

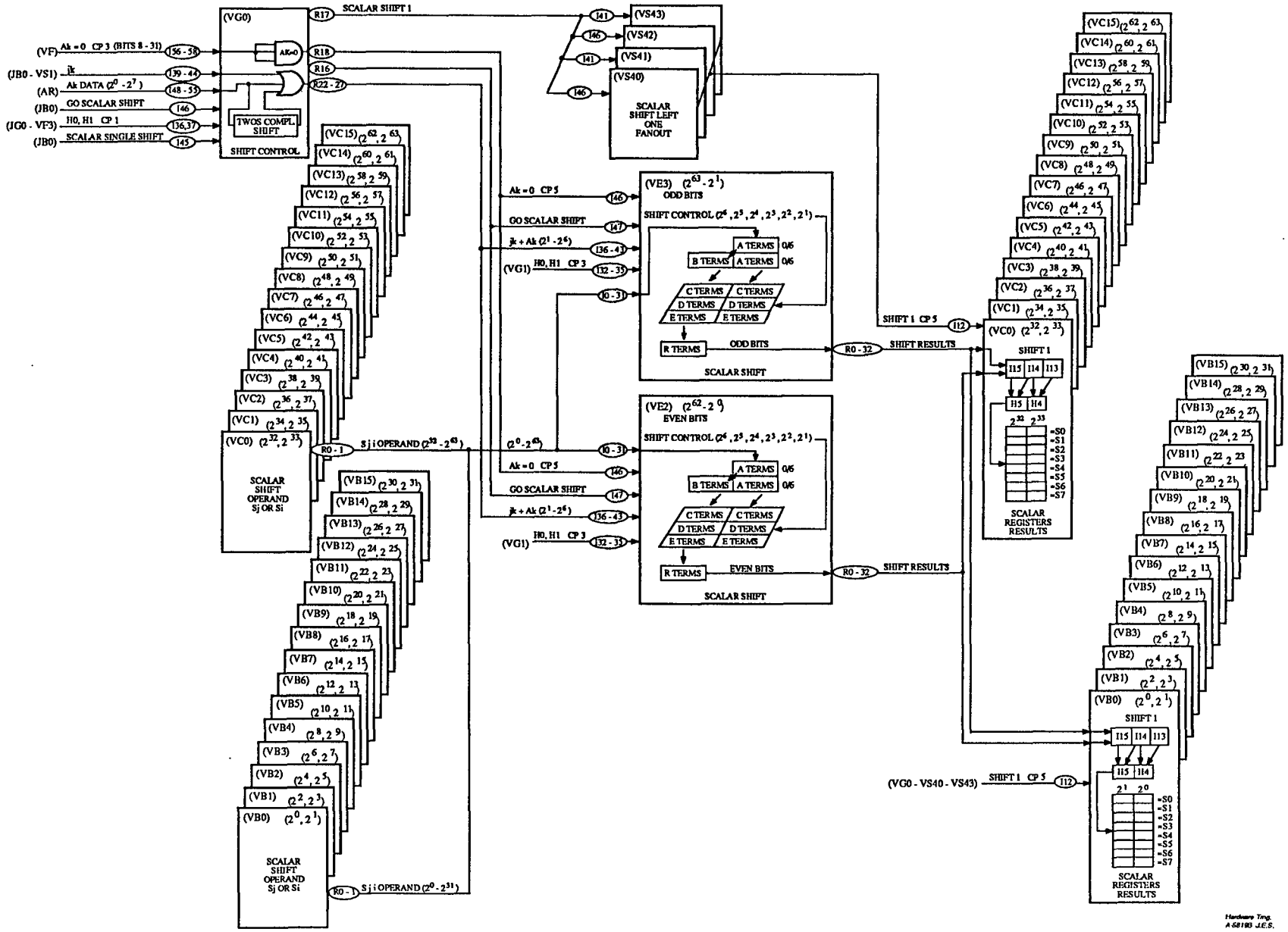
The Single Left and Single Right shift counts are obtained from the (jk) field of the instruction. When a Single Right shift is to be performed, the (jk) field must contain the twos complement of the Right shift count.

The Double Left shift obtains its shift count from the (Ak) field of the instruction, where Ak bits (2^6 - 2^0) contain the valid shift count. The (VE) receives a signal (Ak upper = 0) at CP 4 which means that bits (2^{31} - 2^7) of the Ak Register are all equal to zero. If any of the bits (2^{31} - 2^7) are set, the resultant shift count will be set to zero. The Double Right shift also obtains its shift count from the (Ak) field of the instruction; however, bits 2^6 - 2^0 of the Ak field will be in the twos complement form before arriving at the (VE) options. Both the (jk) and (Ak) field will be converted to the twos complement form on right shifts on the (VG0) option. The (VE) options can determine the type of instruction by decoding h bits 2^1 and 2^0, which they received from the issue fanout option (J-). After zeroing out h bit 2^1 on an 052 - 053 instruction, the (VEs) then use h bit 2^0 to determine a Right or Left shift, and h bit 2^1 to determine a Single or Double shift.

(VG0) Option

The (VG0) option supplies the (VE) option with the shift count. During a Right Double shift the (VG) option will supply the (VE) option with the twos complement shift count. During a Double Shift the (VG) will supply the (VE) option with a signal (Ak = 0) in the event that Ak bits (2^{31} - 2^7) are not equal to zeroes the results on the (VEs) shifts are forced to zero. The (Ak = 0) is orginated from the (VF0) option which polls all the (VF0 - VF3) options for this information.

Hardware Trng.
YM07/17A J.E.S.



CRAY Y-MP SCALAR SHIFT SINGLE/DOUBLE BLOCK DIAGRAM

CRAY PROPRIETARY

T.S. 0/7

2^{62} 2^{60} 2^{58} 2^{56} 2^{54} 2^{52} 2^{50} 2^{48} 2^{46} 2^{44} 2^{42} 2^{40} 2^{38} 2^{36} 2^{34} 2^{32} | 2^{30} 2^{28} 2^{26} 2^{24} 2^{22} 2^{20} 2^{18} 2^{16} 2^{14} 2^{12} 2^{10} 2^{08} 2^{06} 2^{04} 2^{02} 2^{00} (VE2)

2^{63} 2^{61} 2^{59} 2^{57} 2^{55} 2^{53} 2^{51} 2^{49} 2^{47} 2^{45} 2^{43} 2^{41} 2^{39} 2^{37} 2^{35} 2^{33} | 2^{31} 2^{29} 2^{27} 2^{25} 2^{23} 2^{21} 2^{19} 2^{17} 2^{15} 2^{13} 2^{11} 2^{09} 2^{07} 2^{05} 2^{03} 2^{01} (VE3)

I31 I30 I29 I28 I27 I26 I25 I24 I23 I22 I21 I20 I19 I18 I17 I16 | I15 I14 I13 I12 I11 I10 I09 I08 I07 I06 I05 I04 I03 I02 I01 I00

A31 A30 A29 A28 A27 A26 A25 A24 A23 A22 A21 A20 A19 A18 A17 A16 | A15 A14 A13 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00

LEFT SINGLE SHIFT:

A0 - A31 ----► C16 - C47 SHIFT $\overline{32}$, H2

A0 - A15 ----► C32 - C47 SHIFT 32, H3

RIGHT SINGLE SHIFT:

A16 - A31 ----► C00 - C15 SHIFT $\overline{32}$, H0 (AFTER TWOS COMPLEMENT)

A00 - A31 ----► C00 - C31 SHIFT 32, H1 (AFTER TWOS COMPLEMENT)

SHIFT DERIVED FROM (jk) FIELD OF THE INSTRUCTION.

C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32 | C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16 | C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00 SHIFT 32 = 2^5

		2^4, 2^3
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 24	1 1
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 16	1 0
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 8	0 1
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 0	0 0
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12 ALIGNMENT		2^2, 2^1
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 6	1 1
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 4	1 0
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 2	0 1
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 0	0 0

E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15 ALIGNMENT

T.S. 0/7

R31 R30 R29 R28 R27 R26 R25 R24 R23 R22 R21 R20 R19 R18 R17 R16 R15 R14 R13 R12 R11 R10 R09 R08 R07 R06 R05 R04 R03 R02 R01 R00 R32 ← OVERLAP DATA

2^{62} 2^{60} 2^{58} 2^{56} 2^{54} 2^{52} 2^{50} 2^{48} 2^{46} 2^{44} 2^{42} 2^{40} 2^{38} 2^{36} 2^{34} 2^{32} 2^{30} 2^{28} 2^{26} 2^{24} 2^{22} 2^{20} 2^{18} 2^{16} 2^{14} 2^{12} 2^{10} 2^{08} 2^{06} 2^{04} 2^{02} 2^{00} (VE2)

2^{63} 2^{61} 2^{59} 2^{57} 2^{55} 2^{53} 2^{51} 2^{49} 2^{47} 2^{45} 2^{43} 2^{41} 2^{39} 2^{37} 2^{35} 2^{33} 2^{31} 2^{29} 2^{27} 2^{25} 2^{23} 2^{21} 2^{19} 2^{17} 2^{15} 2^{13} 2^{11} 2^{09} 2^{07} 2^{05} 2^{03} 2^{01} (VE3)

NOTE: Shift 1 is done on the inputs to the (VB, VC) options.

RIGHT/LEFT SINGLE SHIFT CONTROL

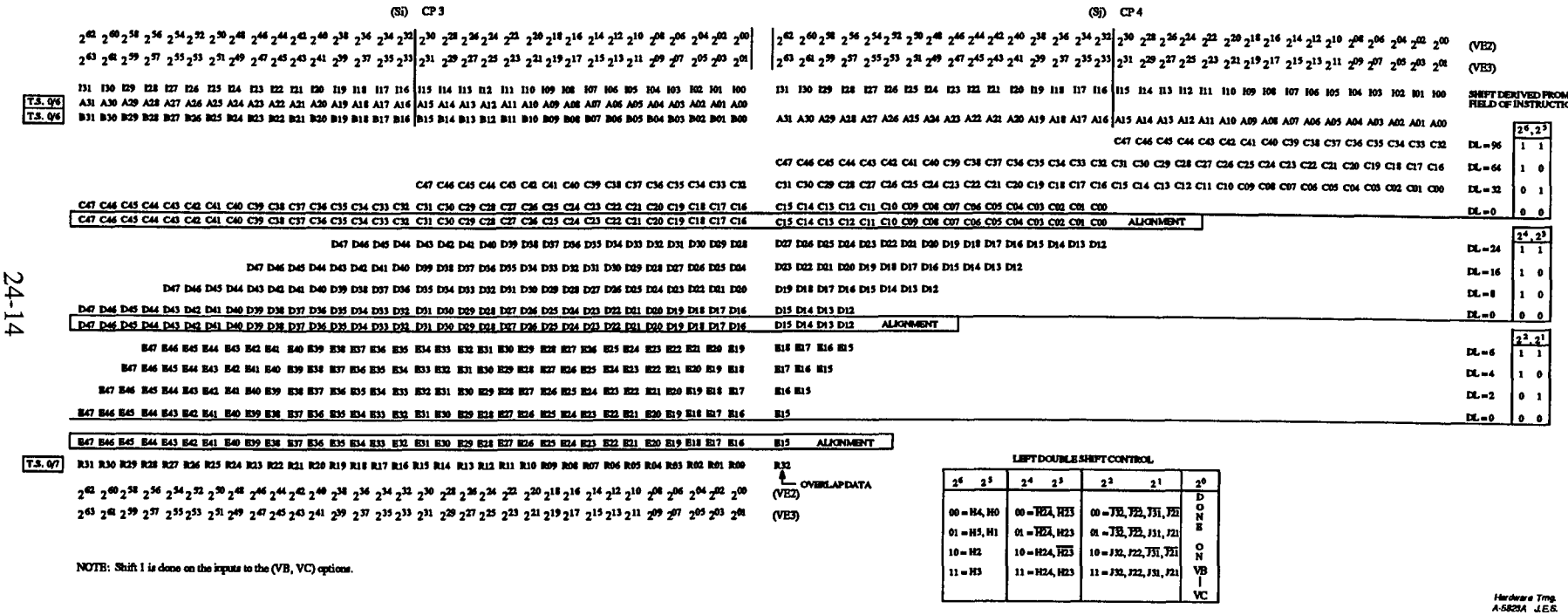
2^5	2^4 2^3	2^2 2^1	2^0
LEFT SINGLE SHIFT			
0 = H2	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	DONE ON VB | VC
1 = H3	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
	11 = H24, H23	11 = J32, J22, J31, J21	
RIGHT SINGLE SHIFT			
0 = H0	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	
1 = H1	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
	11 = H24, H23	11 = J32, J22, J31, J21	

Hardware Trng.
A-5826A J.E.S.

CRAY Y-MP SCALAR LEFT/RIGHT SINGLE SHIFT - 052ijk - 055ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY

CRAY PROPRIETARY



NOTE: Shift I is done on the inputs to the (VB, VC) options.

Hardware Trng.
A-5828A J.E.S.

CRAY Y-MP SCALAR LEFT DOUBLE SHIFT - 056ijk INSTRUCTION (VE) OPTION

(Si) CP 3

(Sj) CP 4

$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}$ | $2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$ (VE2)

$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}$ | $2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$ (VE3)

T.S. 0/6 I31 I30 I29 I28 I27 I26 I25 I24 I23 I22 I21 I20 I19 I18 I17 I16 | I15 I14 I13 I12 I11 I10 I09 I08 I07 I06 I05 I04 I03 I02 I01 I00

T.S. 0/6 A31 A30 A29 A28 A27 A26 A25 A24 A23 A22 A21 A20 A19 A18 A17 A16 | A15 A14 A13 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00

T.S. 0/6 B31 B30 B29 B28 B27 B26 B25 B24 B23 B22 B21 B20 B19 B18 B17 B16 | B15 B14 B13 B12 B11 B10 B09 B08 B07 B06 B05 B04 B03 B02 B01 B00

SHIFT DERIVED FROM (Ak) FIELD OF INSTRUCTION

	$2^6, 2^5$
DL = 96	1 1
DL = 64	1 0
DL = 32	0 1
DL = 0	0 0

	$2^4, 2^3$
DL = 24	1 1
DL = 16	1 0
DL = 8	1 0
DL = 0	0 0

	$2^2, 2^1$
DL = 6	1 1
DL = 4	1 0
DL = 2	0 1
DL = 0	0 0

(Si)	(Sj)
C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32	C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32
C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16	C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16
C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00	C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00
C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32 C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16	C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00 ALIGNMENT
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28	D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24	D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32	D19 D18 D17 D16 D15 D14 D13 D12
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16	D15 D14 D13 D12
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16	D15 D14 D13 D12 ALIGNMENT
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19	E18 E17 E16 E15
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18	E17 E16 E15
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17	E16 E15
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16	E15
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16	E15 ALIGNMENT

T.S. 0/7 R31 R30 R29 R28 R27 R26 R25 R24 R23 R22 R21 R20 R19 R18 R17 R16 R15 R14 R13 R12 R11 R10 R09 R08 R07 R06 R05 R04 R03 R02 R01 R00 R32

OVERLAP DATA

$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}\ 2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$ (VE2)

$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}\ 2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$ (VE3)

NOTE: Shift 1 is done on the inputs to the (VB, VC) options.

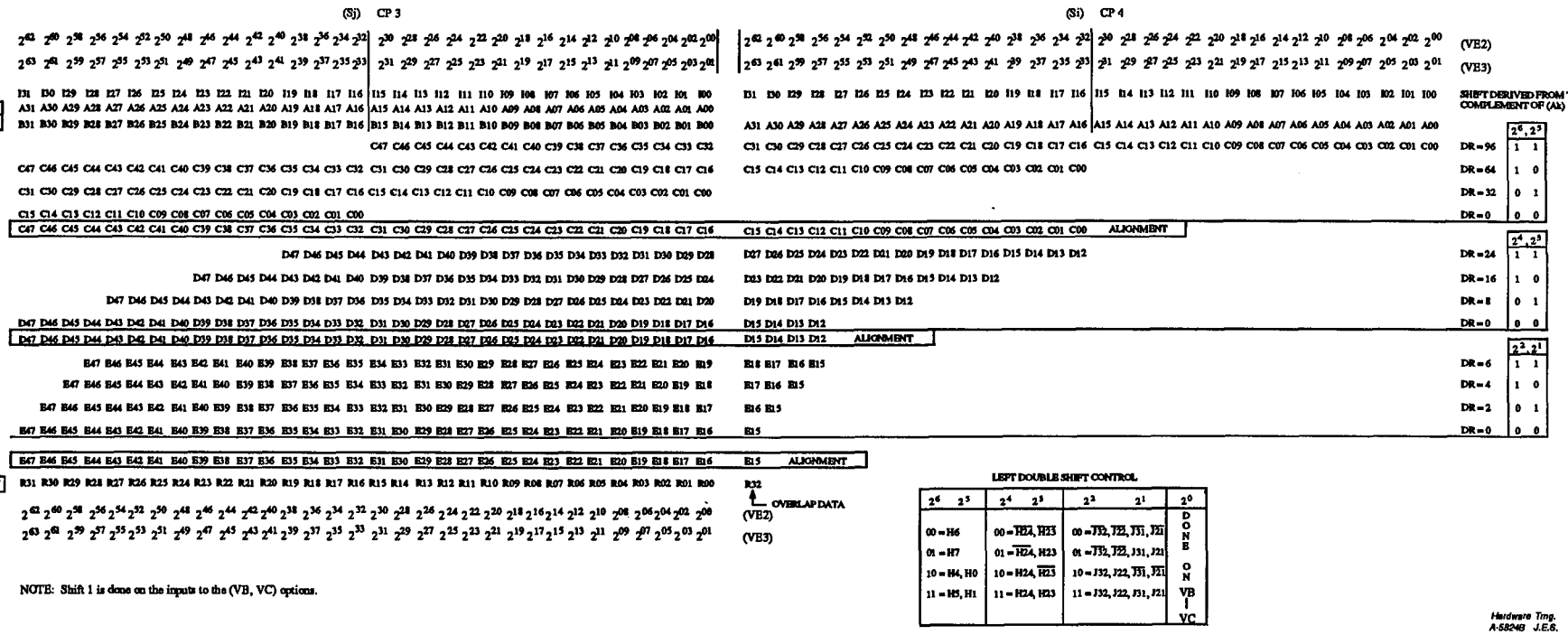
LEFT DOUBLE SHIFT CONTROL

$2^6\ 2^5$	$2^4\ 2^3$	$2^2\ 2^1$	2^0
00 = H4, H0	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	DONE ON VB | VC
01 = H5, H1	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
10 = H2	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
11 = H3	11 = H24, H23	11 = J32, J22, J31, J21	

Hardware Trng.
A-5823A J.E.S.

CRAY Y-MP SCALAR LEFT DOUBLE SHIFT - 056ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY



NOTE: Shift 1 is done on the inputs to the (VB, VC) options.

Hardware Tmg.
A-58249 J.E.S.

CRAY Y-MP SCALAR RIGHT DOUBLE SHIFT - 057ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY

(Sj) CP 3 (Si) CP 4

	$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}$ | $2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$	$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}$ | $2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$	(VE2)
	$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}$ | $2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$	$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}$ | $2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$	(VE3)
	I31 I30 I29 I28 I27 I26 I25 I24 I23 I22 I21 I20 I19 I18 I17 I16 | I15 I14 I13 I12 I11 I10 I09 I08 I07 I06 I05 I04 I03 I02 I01 I00	I31 I30 I29 I28 I27 I26 I25 I24 I23 I22 I21 I20 I19 I18 I17 I16 | I15 I14 I13 I12 I11 I10 I09 I08 I07 I06 I05 I04 I03 I02 I01 I00	SHIFT DERIVED FROM TWOS COMPLEMENT OF (Ak)
T.S. 0/6	A31 A30 A29 A28 A27 A26 A25 A24 A23 A22 A21 A20 A19 A18 A17 A16 | A15 A14 A13 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00		
T.S. 0/6	B31 B30 B29 B28 B27 B26 B25 B24 B23 B22 B21 B20 B19 B18 B17 B16 | B15 B14 B13 B12 B11 B10 B09 B08 B07 B06 B05 B04 B03 B02 B01 B00	A31 A30 A29 A28 A27 A26 A25 A24 A23 A22 A21 A20 A19 A18 A17 A16 | A15 A14 A13 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00	

	$2^6, 2^5$
DR = 96	1 1
DR = 64	1 0
DR = 32	0 1
DR = 0	0 0

		C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32	C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16 C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00
	C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32 | C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16	C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00	
	C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16 C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00		
	C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00		
	C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32 | C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16	C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00 ALIGNMENT	

	$2^4, 2^3$
DR = 24	1 1
DR = 16	1 0
DR = 8	0 1
DR = 0	0 0

		D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28	D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12
		D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24	D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12
		D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20	D19 D18 D17 D16 D15 D14 D13 D12
	D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16	D15 D14 D13 D12	
	D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16	D15 D14 D13 D12 ALIGNMENT	

	$2^2, 2^1$
DR = 6	1 1
DR = 4	1 0
DR = 2	0 1
DR = 0	0 0

		E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19	E18 E17 E16 E15
		E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18	E17 E16 E15
		E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17	E16 E15
	E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16	E15	
	E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16	E15 ALIGNMENT	

T.S. 0/7	R31 R30 R29 R28 R27 R26 R25 R24 R23 R22 R21 R20 R19 R18 R17 R16 R15 R14 R13 R12 R11 R10 R09 R08 R07 R06 R05 R04 R03 R02 R01 R00	R32 ← OVERLAP DATA	
	$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}\ 2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$	(VE2)	
	$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}\ 2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$	(VE3)	

NOTE: Shift 1 is done on the inputs to the (VB, VC) options.

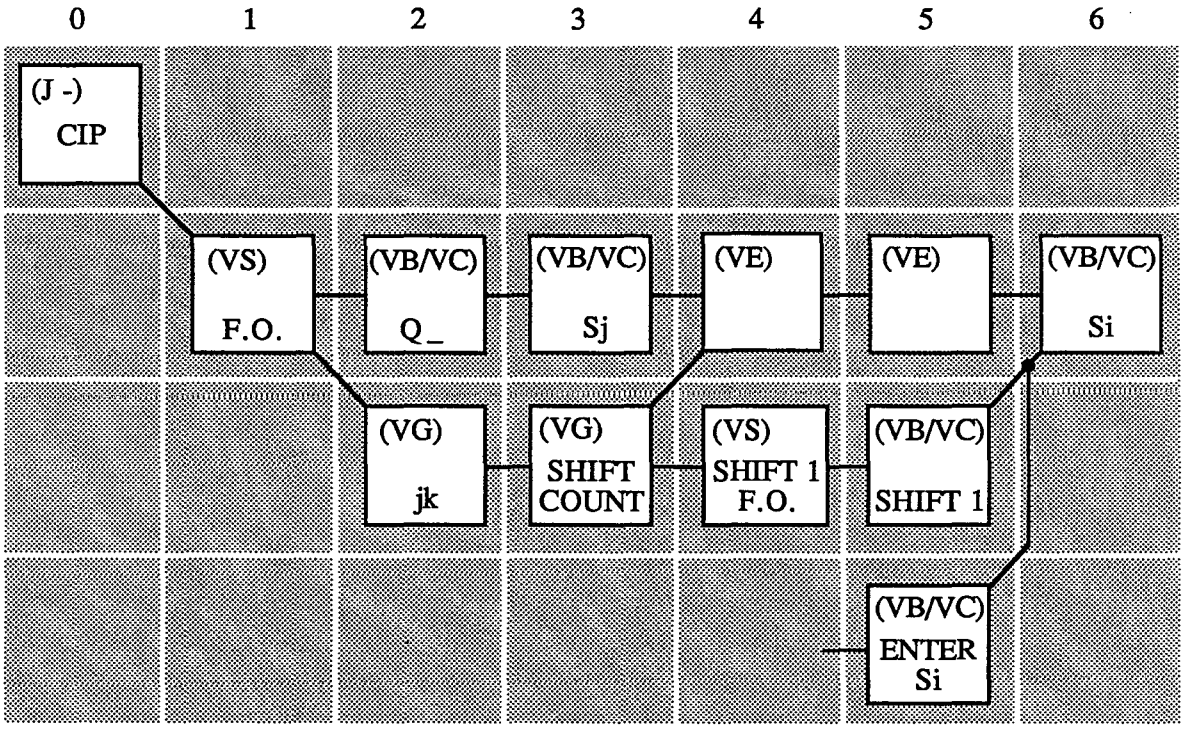
LEFT DOUBLE SHIFT CONTROL

$2^6\ 2^5$	$2^4\ 2^3$	$2^2\ 2^1$	2^0
00 = H6	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	DONE ON VB | VC
01 = H7	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
10 = H4, H0	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
11 = H5, H1	11 = H24, H23	11 = J32, J22, J31, J21	

Hardware Tmg.
A-5824B J.E.S.

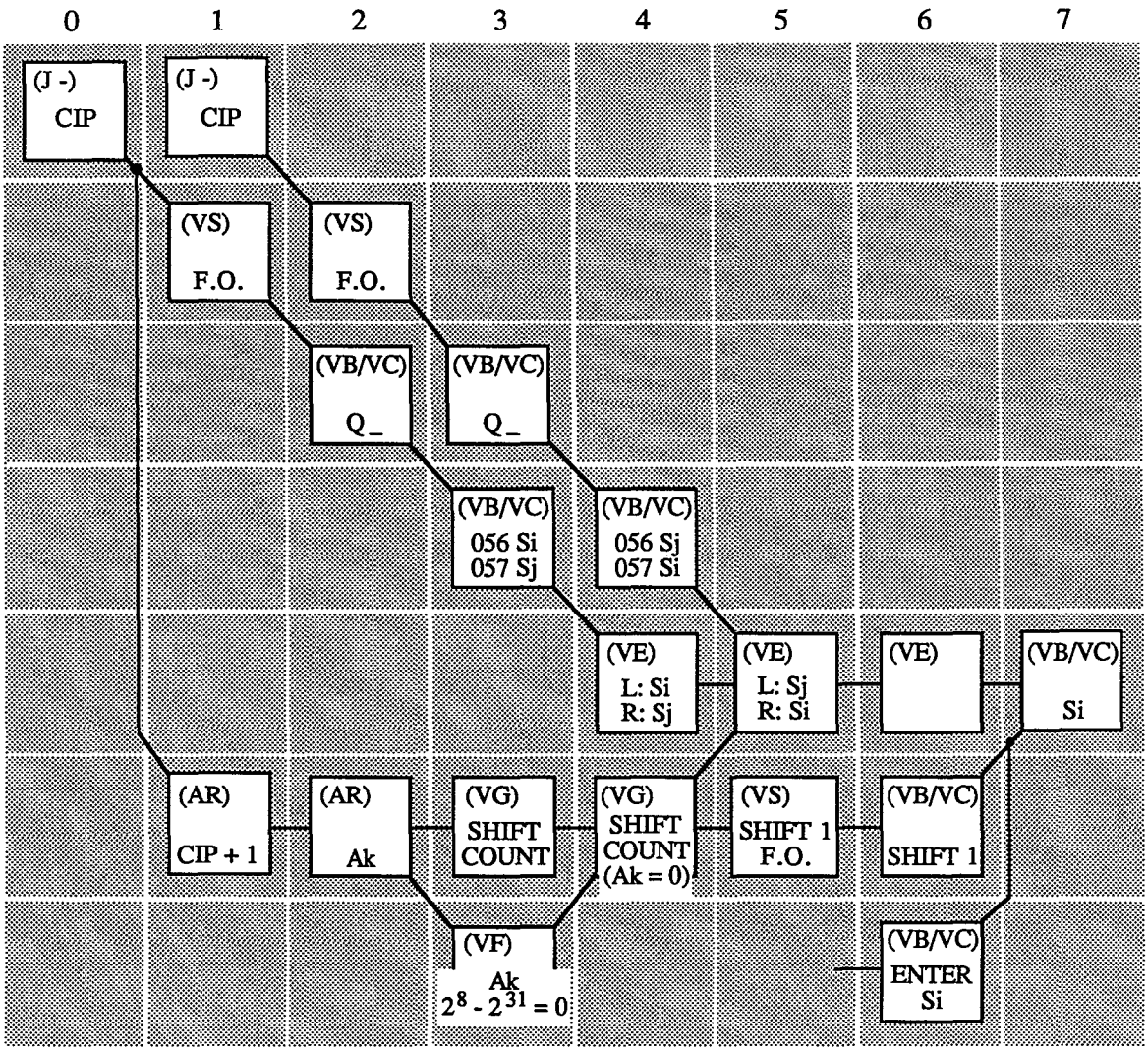
CRAY Y-MP SCALAR RIGHT DOUBLE SHIFT - 057ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY



052 - 055 - SCALAR SINGLE SHIFT (LEFT OR RIGHT) *A-7731*

CRAY Y-MP SCALAR SINGLE SHIFT



056 - 057 - SCALAR DOUBLE SHIFT (LEFT OR RIGHT)

A-7728

CRAY Y-MP SCALAR DOUBLE SHIFT

SCALAR POPULATION COUNT

The Scalar Population Count functional unit counts the number of one bits in the Scalar register specified by the (j) field of instruction 026ijk (k = 0 or 1). The maximum count would be 100_8 or 64_{10} for the corresponding number of one bits set in the Scalar register, and the smallest would be zero for no bits set in the Scalar register.

The k field of the instruction determines whether or not the entire population count will be recorded in Ai. If it is an 026ij0 instruction, all 7 bits of the final population count are sent to the A register. When an 026ij1 instruction is issued, the entire S register is counted for the number of one bits set, but then only bit 2^0 of the count is sent to the A register. If bit 2^0 of the count is equal to a zero then the count has even parity, indicating an even number of bits set; and if bit 2^0 of the count is a one, then the count has odd parity. The Scalar Population Count takes 4 CPs to execute inside the functional unit.

Scalar Pop Count (Options Involved)

(VG0, VG1) Option

The (VG0, VG1) option each receive 16 bits of the 64-bit word, and partially sum the number of one bits to five terms. The 16 bits spend 1 CP on the (VG) options. To sum the 16 bits, the (VG) graphically stands the bit on end and performs 3-bit adds on 3-bit groups.

The carries generated propagate to the next higher bit position, while the sums remain in the same position. There are four levels of additions performed on the (VG) options, which originally receive their input from the (VB) options, or the lower 32 bits of the Scalar registers. The partial summations (R0 - R4) terms, are sent to the (VH0) option for the final summation.

(VH0, VH1) Options

The (VH) options each receive 16 bits from the (VC) options, or upper 32 bits of the S registers. The (VH) options are identical to the (VG) options in performing four levels of additions. The (VH0) option however, is different from the (VH1) and the (VG0, VG1) options when doing a population count, in that the (VH0) receives the partial count from the (VG0, VG1) and (VH1) and continues the population count to its completion. The (VH0) will continue to add the partial sums along with its own partial sums for an additional 2 CPs. The final sum is sent to the (AR) options from the (VH0) option as R terms (R16 - R22). If an 026ij1 instruction was issued, then only R16 is sent to the (AR) option, as the parity of the population count.

SCALAR POPULATION COUNT FUNCTIONAL UNIT INSTRUCTIONS

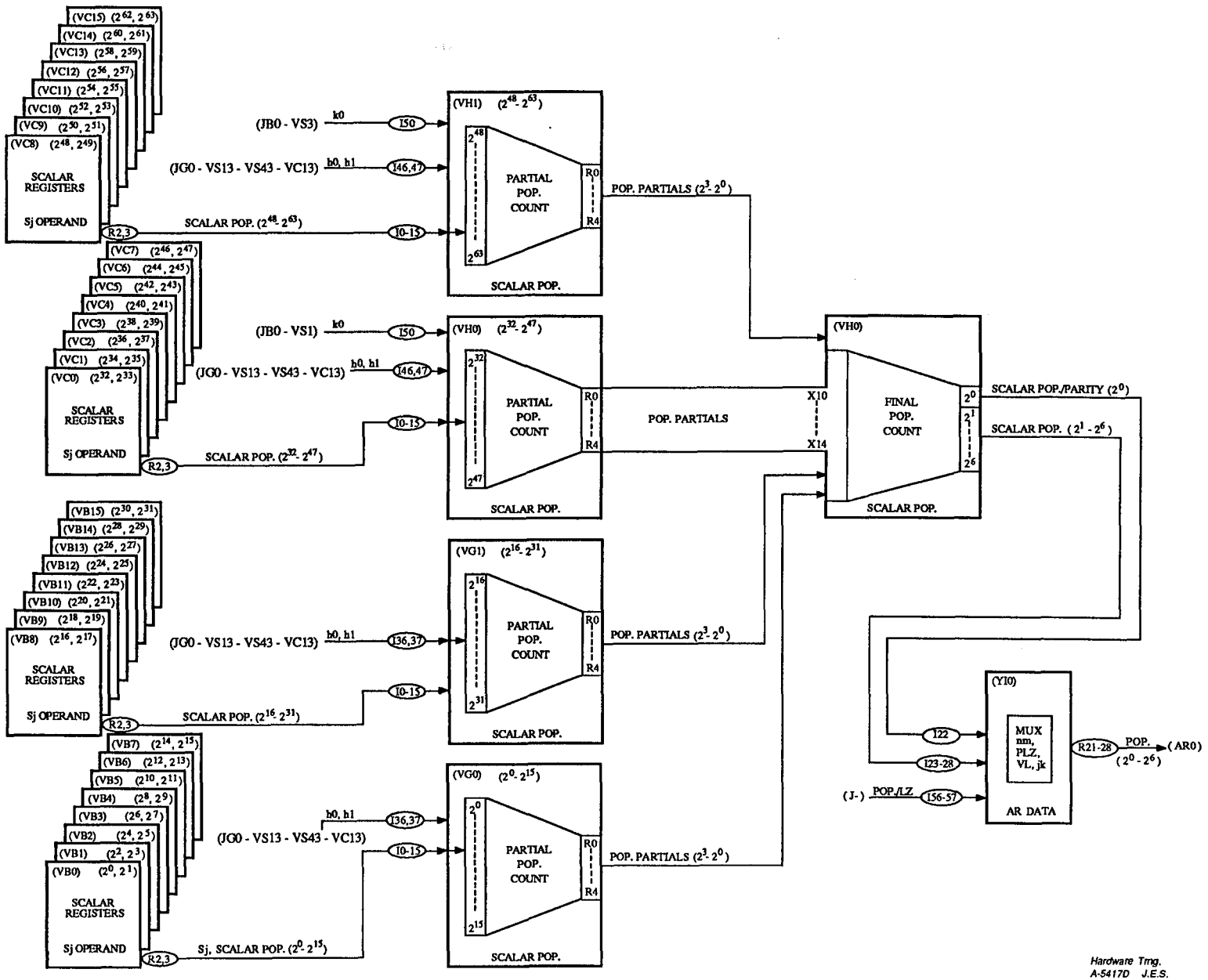
INSTRUCTIONS

026ij0	Ai PSj	Population count of (Sj) to Ai
026ij1	Ai QSj	Population count parity of (Sj) to Ai

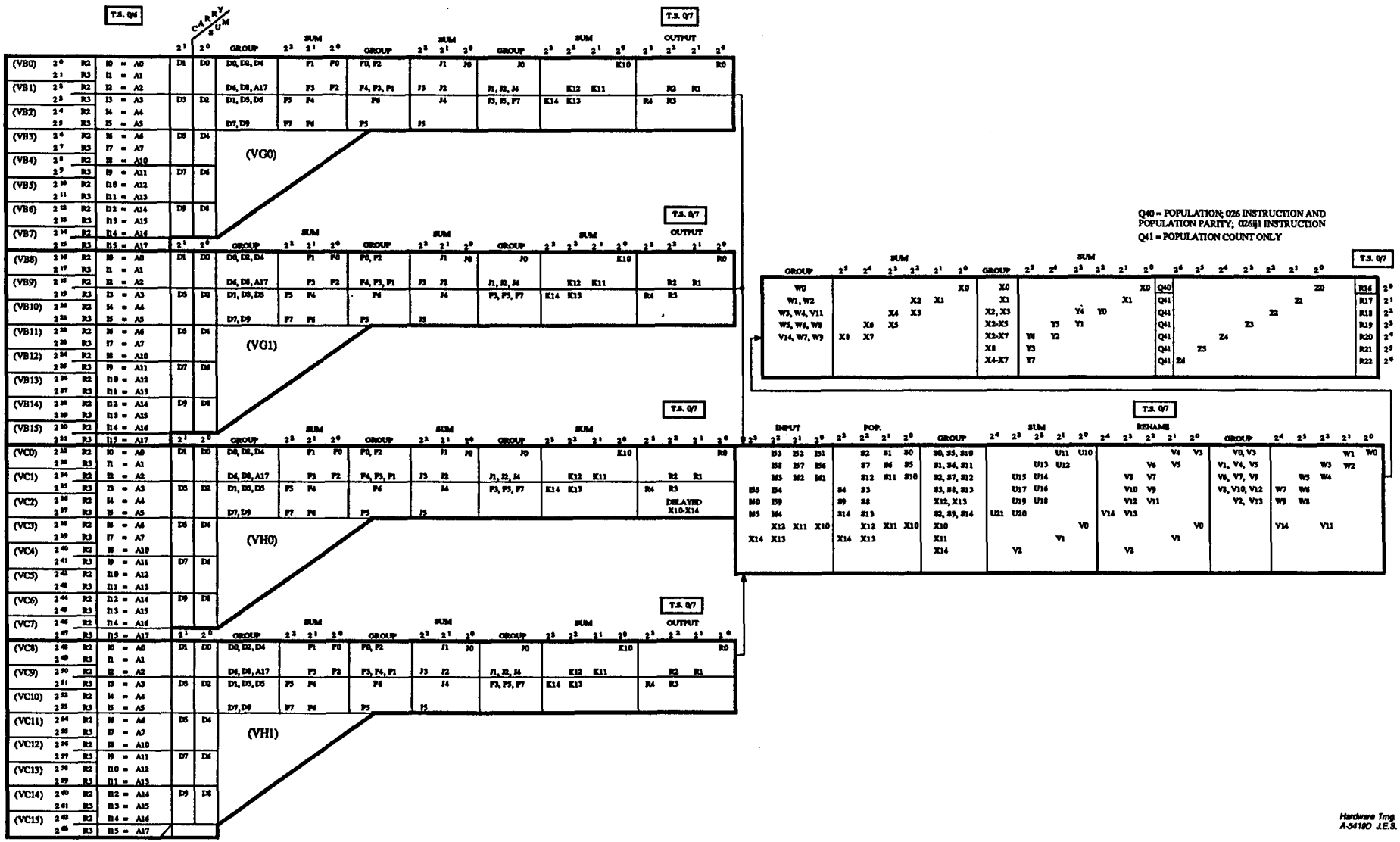
The 026ij0 instruction counts the number of one bits in (Sj) and transfers the count to Ai.

The 026ij1 instruction counts the number of one bits in (Sj) and transfers only bit 2^0 of the count to Ai. If $2^0=0$ there is even parity, and if $2^0=1$ there is odd parity.

Hardware Trng.
YM05/23 J.E.S.

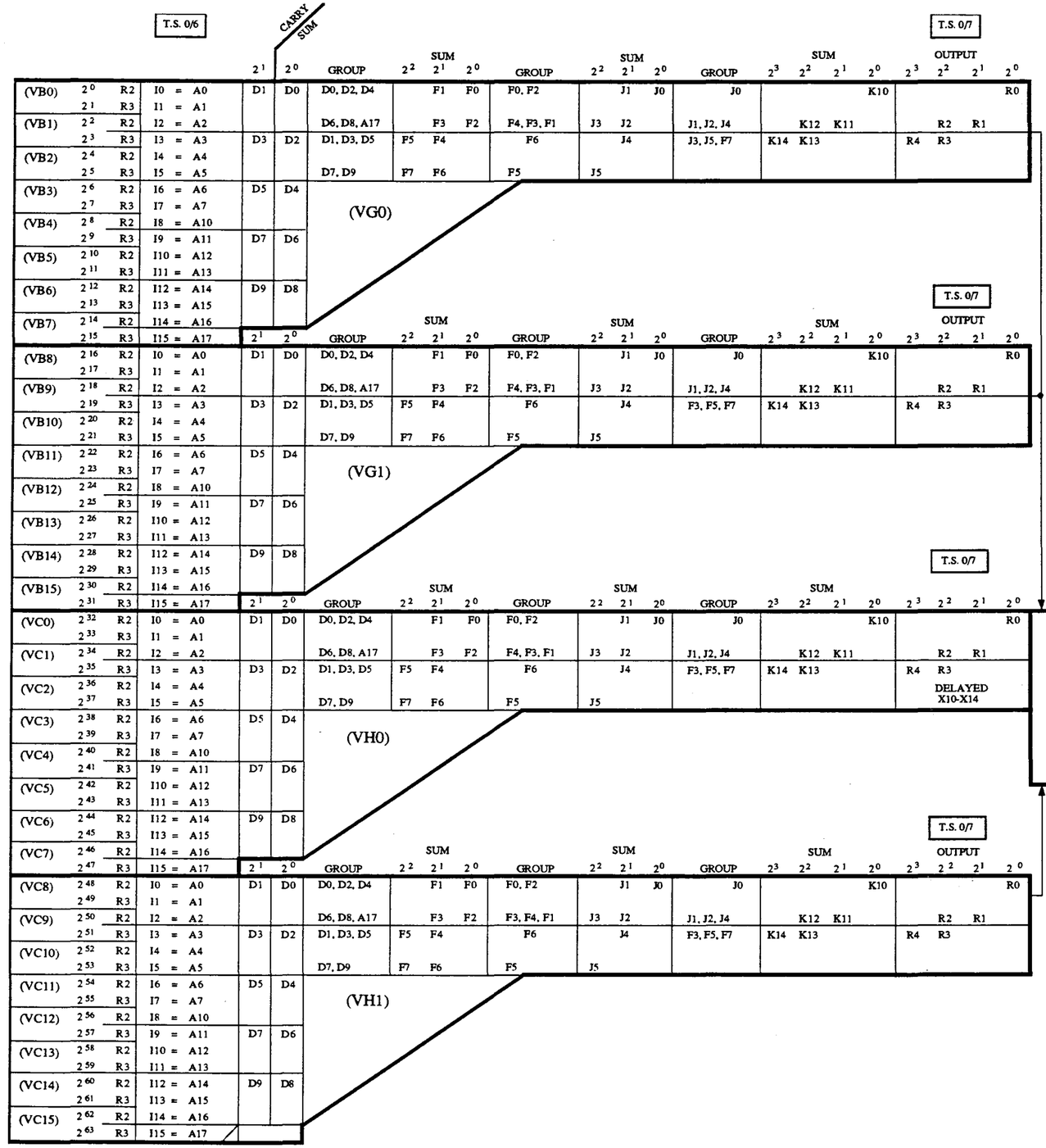


CRAY Y-MP SCALAR POPULATION/PARITY COUNT BLOCK DIAGRAM



Hardware Trng.
A-5419D J.E.S.

CRAY Y-MP SCALAR POPULATION COUNT FLOW CHART

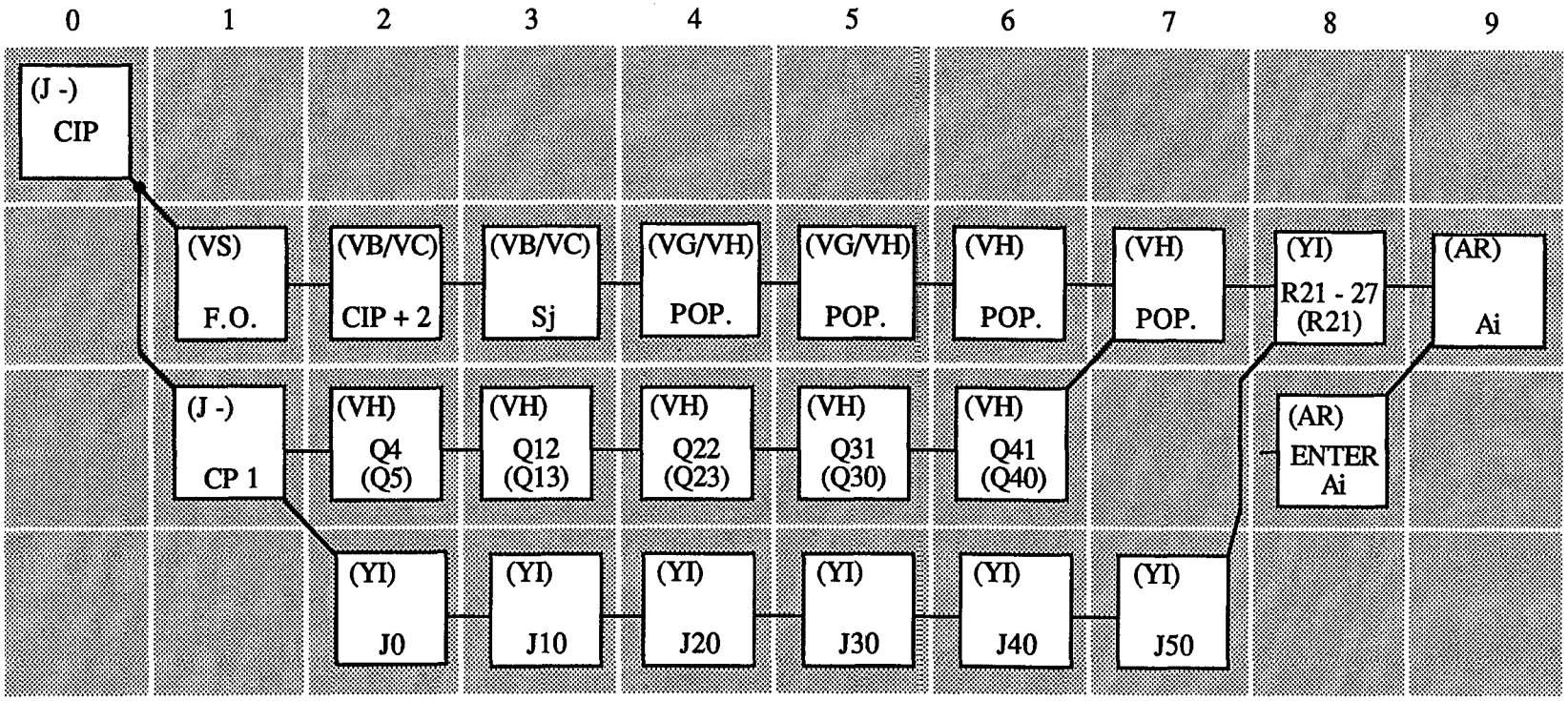


Q40 = POPULATION; 026 INSTRUCTION AND POPULATION PARITY; 026ij1 INSTRUCTION

Q41 = POPULATION COUNT ONLY

GROUP	SUM	GROUP	SUM		T.S. 0/7
W0	X0	X0	X0	Q40	R16 2^0
W1, W2	X2 X1	X1	X1	Q41	R17 2^1
W3, W4, V11	X4 X3	X2, X3	Y4 Y0	Q41	R18 2^2
W5, W6, W8	X6 X5	X2-X5	Y5 Y1	Q41	R19 2^3
V14, W7, W9	X8 X7	X2-X7	Y6 Y2	Q41	R20 2^4
		X8	Y3	Q41	R21 2^5
		X4-X7	Y7	Q41	R22 2^6

| | | |



026ij0 - TRANSMIT POPULATION COUNT OF (Sj) TO Ai
(026ij1 - TRANSMIT POPULATION COUNT PARITY OF (Sj) TO Ai)

A-7734

CRAY Y-MP SCALAR POPULATION COUNT

CRAY PROPRIETARY

SCALAR LEADING ZEROES COUNT

The Scalar Leading Zeroes Count functional unit counts the number of zeroes preceding the first 1-bit in a specified Scalar register. The result is then transferred to the lower 7 bits of an Ai register. To utilize the Leading Zeroes functional unit an 027ij0 instruction is used, where (Sj) is the operand and Ai the result register. The (Sj) operand is divided into four groups of 16 bits each and sent to four options, two (VGs) and two (VHs). Each option counts the number of leading zeroes in its 16-bit group, making the individual zeroes count anywhere from (0 - 20_8). The partial counts are then sent to one (VH) option which compiles the results from each of the 4 parcels into a single count. The final count, which can be anywhere from (0 - 100_8), is then sent to the specified A register.

The Scalar Leading Zeroes Count functional unit shares the same paths into the (VG, VH) as the Scalar Population Count and Scalar Add functional unit. Also the R terms (R0 - R4) are shared by the Scalar Add and Scalar Pop as well as some internal Boolean terms. The Scalar Leading Zeroes count takes 3 CPs inside the functional unit. If a Scalar Population Count instruction (026ij0) or 026ij1) is issued before a Scalar Leading Zeroes Count instruction (027ij0), the 027 will hold issue for 1 CP, to prevent a conflict on the (AR) option.

Scalar Leading Zeros Count (Option Involved)

(VG0, VG1) Options

Each (VG) option contains 16 of the 64 bits of the (Sj) Scalar register sent from the (VB) option. The (VG0) contains bits (2^{15} - 2^0), while (VG1) contains bits (2^{31} - 2^{16}). The (VG) options count the number of leading zeroes in each of their 16-bit groups. The maximum numbers would be 20_8, or all 16 bits equal to zeroes. The 5-bit result is sent to the (VB0) options, to determine the final count of all 64 bits. To determine the Leading Zero Count on 16 bits, the 16 bits are divided into four 4-bit groups. The count within each of the 4-bit groups generates the C terms, which are then combined to form the E terms, which give the final count (0 - 20_8) of zeroes within the group.

(VH0, VH1) Options

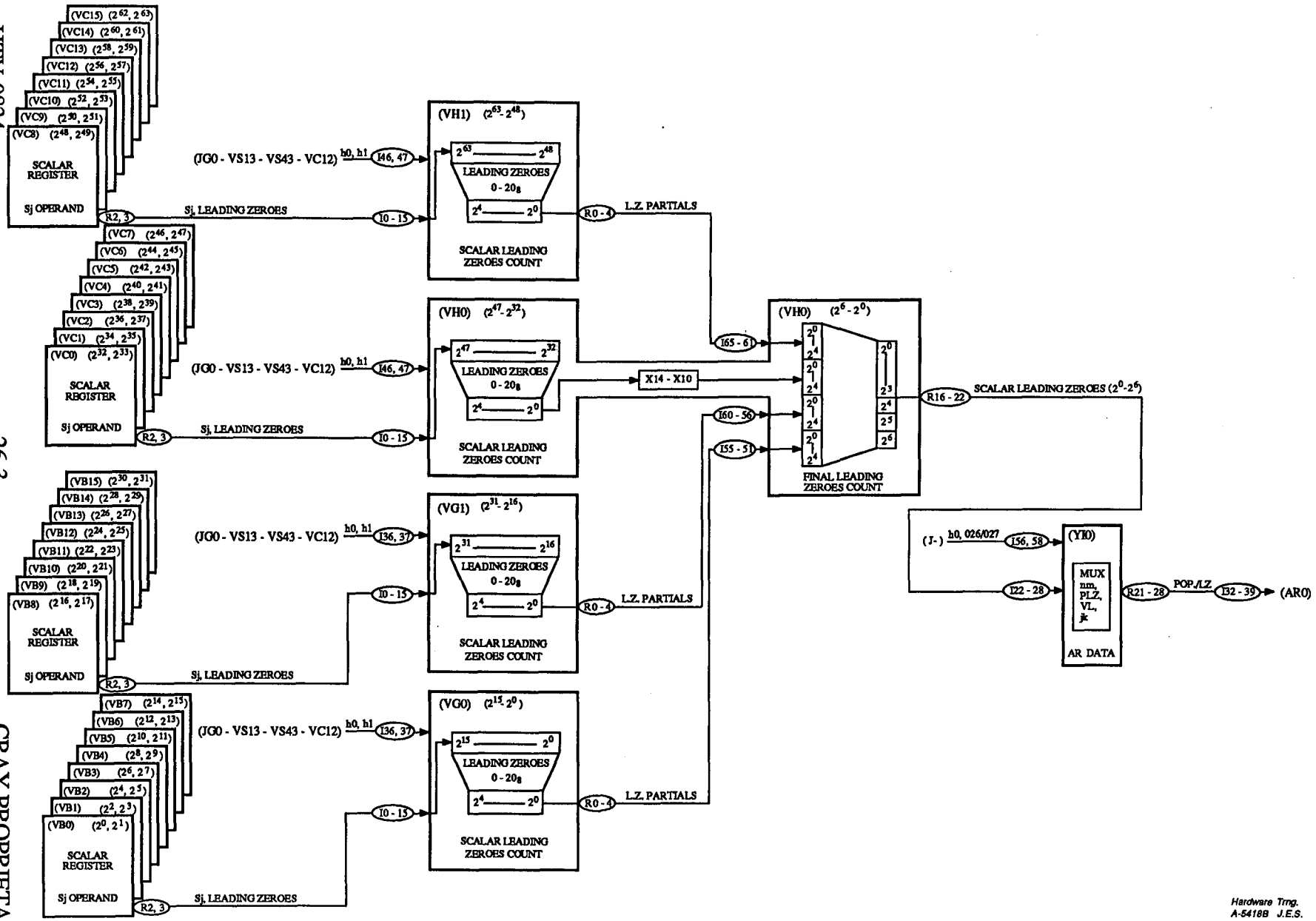
The (VH) options are identical to the (VG) options except the (VH0) option will finalize the Leading Zero Count. By checking the counts from the other three partial sums plus its own, the (VH0) can determine what the final sum should be. If all 4 parcels have a count of (20_8), the (VH0) option will only send bit (2^6) to the (AR) option, for a count of 100_8, or else 6 bits (2^5 - 2^0) are sent to the (AR) options. To set (2^5) of the result, the count could be somewhere between (40_8 - 77_8). To set (2^4) of the result, the count should be somewhere between (20_8 - 37_8) or (60_8 - 77_8). Bits (2^3 - 2^0) of the result are selected from one of the following: from (VH1) when the count is not equal to (20_8), from (VH0) when the count is greater then (20_8) but less than (40_8), from (VG1) when the count is greater then (40_8) but less then (60_8), or from (VG0) when the count is greater than (60_8) but less than (100_8).

Hardware Trng.
YM06/11B J.E.S.

CRAY Y-MP SCALAR LEADING ZEROES COUNT FUNCTIONAL UNIT INSTRUCTION

INSTRUCTION

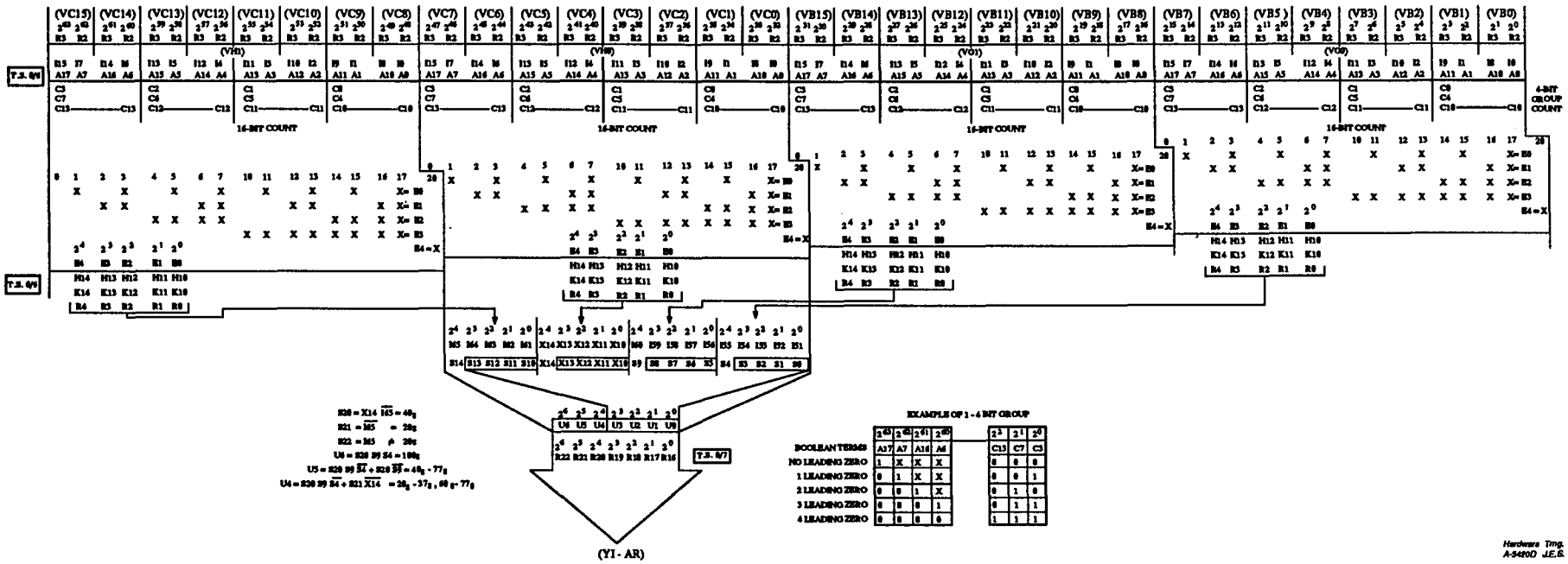
027ij0	Ai	ZSj	Leading zero count of (Sj) to Ai



Hardware Trng.
A-5418B J.E.S.

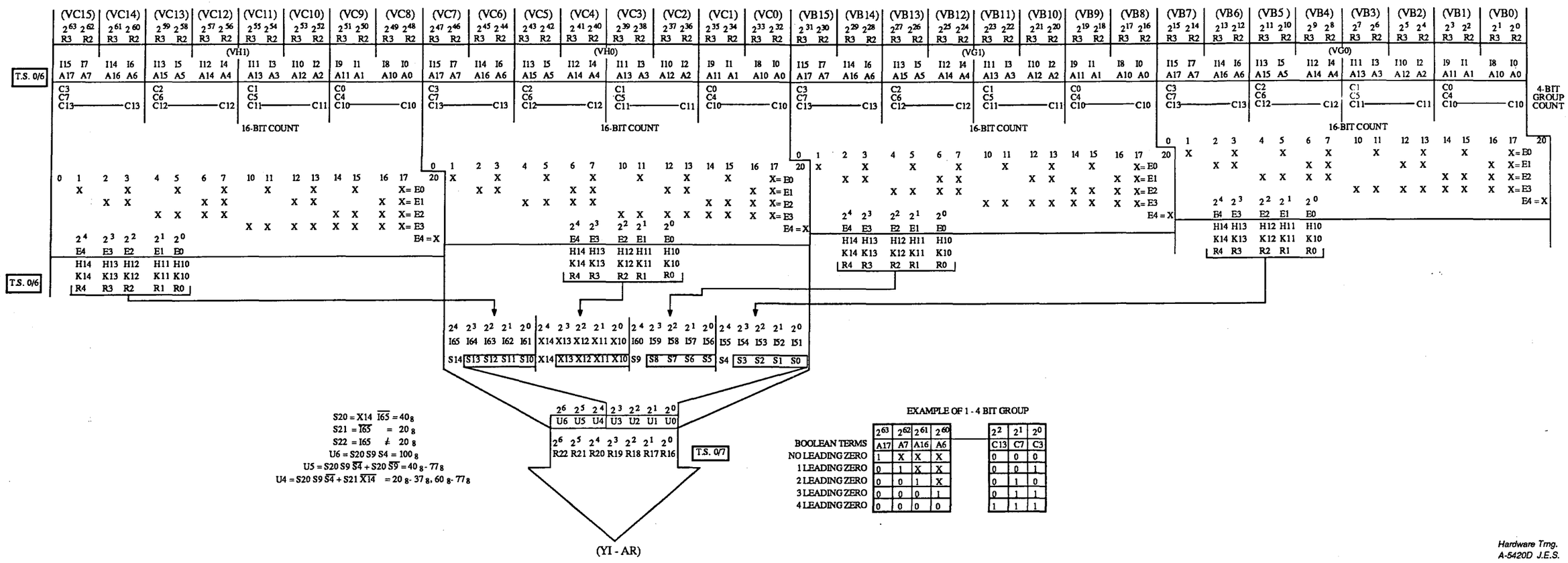
CRAY Y-MP SCALAR LEADING ZEROES COUNT
BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP SCALAR LEADING ZEROES FLOW CHART

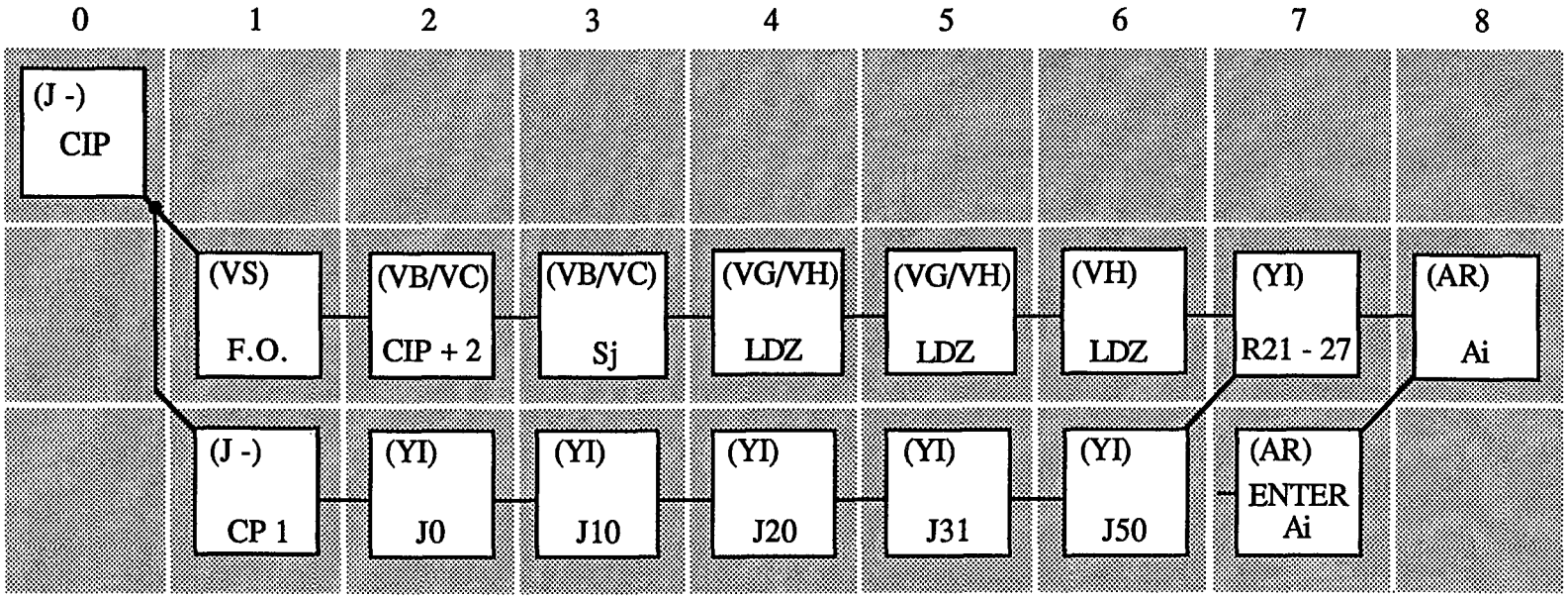
CRAY PROPRIETARY



CRAY Y-MP SCALAR LEADING ZEROES FLOW CHART

Hardware Trng.
A-5420D J.E.S.

CRAY PROPRIETARY



027ij0 - TRANSMIT LEADING ZERO COUNT OF (Sj) TO Ai

A-7735

CRAY PROPRIETARY

CRAY Y-MP SCALAR LEADING ZERO

CRAY Y-MP VECTOR REGISTERS

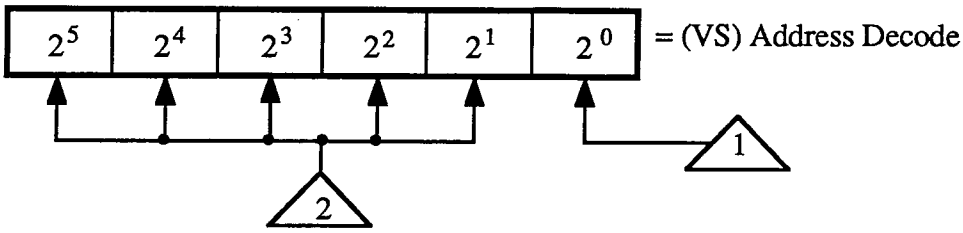
The CRAY Y-MP contains eight Vector registers (V0 - V7), each of which in turn contain 64_{10} elements, with each element 64_{10} bits in length. The Vector registers have their own Integer functional units including the Vector Add, Vector Logical, Vector Logical 2, Vector Shift, and Vector Pop. They also share with the Scalar registers, the Floating-point functional units, which include the Floating Add, Floating Reciprocal, and Floating Multiply. Vector register data can also be sent to memory or read from memory in amounts ranging from 1 to 64 elements per instruction. Any element of a Vector register can be loaded into a Scalar register, and any Scalar register can be loaded into any element of a Vector register.

The Vector instructions are all one parcel instructions in which the (ijk) fields contain the operands and the (gh) fields contain the instruction decodes. The (gh) fields tell the machine what functional units will be needed, and the (ijk) fields tell which Vector registers will be used. Generally, the (k) field of a Vector instruction contains some Vector register (V0 - V7) as an operand. The (j) field can be either a Scalar or Vector register depending on the instruction, and is used for an operand. The (i) field is generally a Vector register and used as the destination or resultant register.

Because the Vector operands can be sent to so many different functional units, the (VB, VC) options are used to multiplex the various operands (Vj/Sj, Vk) to the correct functional units. To receive results back from the various functional units, the (VA) options are used to multiplex the various functional units to one of eight Vector registers.

To control the Vector registers, a (VD) option is used. The (VD) contains a Read Address register, a Read Vector Length register, and a Write Address register for each Vector register. Having both a Read and a Write Address register, allows a simultaneous Read and Write of the same Vector register to occur, as long as the Write stays ahead of the Read during a chain operation. The Read Address register points to a particular element of a Vector register. On starting up a Vector instruction, the Read/Write addresses are set to zero to allow a Read/Write of element zero. As long as the Read Address register does not equal the Vector Length register, the address is allowed to increment to the next element of the Vector register. When the Read Address register finally equals the Vector Length register, the (VD) option will stop sending addresses to the Vector register, and will generate a Release Vjk to the (JE) option. The Release Vjk will drop the reservation on (Vj, Vk), and also release the reservation on the functional unit since no more operands are being sent to the functional unit. The (Vi) reservation is held on the (JC) option and is released when the Write sequence is complete. The Go Write Release stops the Write address from incrementing, while the Go Write allows the Write address to increment on the (VD) option.

The (VS) options contain the Vector registers with four options needed to handle each register. Each option contains 16 bits of each of the 64 elements making up a particular Vector register. The Read or Write address sent from the (VD) option is broken down into bits (2^5 - 2^0) as shown on the following page.



1 Read Even/Odd or Write Even/Odd - Bit 2^0 of the original address sent from the (VD) option becomes Read Even or Read Odd for a Read of the Vector register; or Write Even or Write Odd for a Write to the Vector register. This allows the 64 elements to be interleaved on even or odd boundaries so that Read and Write operations can occur simultaneously.

2 Original address bits (2^1 - 2^5) are renumbered coming into the (VS) option as (2^0 - 2^4) and become a 1 of 32 element selects for a Read or Write operation.

The (VS) options also contain a bypass path which is used when the Read address = Write address. When there is a conflict between a Read and a Write both wanting even or odd elements, the Write will be held for 1 CP while the Read will go ahead.

Vector Register Issue Control

The Vector register issue control is contained on a series of (J-) options which are the (JC, JD, JE, JF) options. The (ghijk) instruction decodes arrive from the (HC, HB) options NIP register. The instruction is decoded on the (JD, JC) options to determine if the instruction is a Vector instruction. If not, the (JD, JC) options will generate a Go Issue for that instruction; however, if the instruction is a Vector instruction to be controlled by the (JC, JD) options, the (JC, JD) will check for conflicts before issuing the Vector instruction. Upon resolving the conflict at CIP time the Vector instruction will issue, reserving the Vector functional unit (VFU) and the operands and destination registers along with the Port A, B, C if they are being used.

In order to read the operands out of the Vector register, the (JC) option will send a Go Read sequence (V0 - V7) to the (VD) option along with Pair Required if two operands are to be read-out of the Vector register at the same time. Pair Required ensures that during a chaining operation both operands have the same element ready to be read. The (JC0) option sends the Go Write sequence to the (VD) option to set up the (VDs) to control the writing of the Vector register elements. The (VD), upon receiving the Go Read/Write sequence, will clear the Read/Write Address registers and load the Read Vector Length register during a Read sequence. The (VD) then takes over reading the Vector register elements if the Read operation is required.

To steer the Vector data, to the correct Vector functional unit (VFU), the VFU Select Code is used. The VFU Select Code is generated on the (JD) option and sent to the (VA, VB, VC) options. The (VB, VC) uses the VFU to steer the operands to the correct FU. The (VA) latches and holds the VFU upon getting the Go (VA) option signal. This is because of the time spent before it can be used by the FU sending results back to the Vector register. The operand selection on the (VB, VC) options is received from the (JG) option after a fanout on the (VS) option. The (VB, VC) decodes the operands (ijk) and sets up the paths for the data from the Vector register to be routed to the correct FU using the VFU Select Code. The data (Vj, Sj), (Vk), or (Vi) is then sent to the FU.

In order for the results to be loaded back into the Vector register, a number of timing conditions need to be set up. The timing starts on a Read operation when the (VD) has passed the address to the correct Vector registers to read the vector elements. The (VD) will also generate a "Go Data Vjk" each time a vector element is being read. The Go Data Vjk is sent to the (JE0) option where a compare for the FU against the operand used is performed. Upon identifying the Go Data Vjk the (JE) will generate a Go Data FU X where "X" is the FU being used by the operands, "X" can be either (FR, FA, FM, VA, VS, or VL). The Go Data FU X is passed to the (JF0) option where another compare takes place. This time however, the compare is against the FU and (Vi) to determine which Vector register will receive the results from the FU. The Go Data FU X will match a correct (Vi) and, depending on the FU, will start a delay chain equivalent to the time it takes the FU to send results to the Vector registers. At FU delay time CP 4 the (JF) will generate a Go Write V0 - V7 from the Vector instructions (Vi) field. The Go Write V0 - V7 is passed to the (VD) option to prepare for a write into the (Vi) Vector register. The (VD) upon getting the Go Write will send its Write address to the Vector register and then increment its Write address by plus 1.

During a Vector Read of memory on a 176 instruction, the Port A, B and Go Read are passed directly to the (JF) option so the (JF) can start a delay chain for a read memory reference, and then generate the Go Write V0 - V7 to the (VD) at CP 4 times.

On a Vector register Read operation, when the Read address is equal to the Vector Length on the (VD) option, the (VD) will generate a Release Vjk to the (JC, JE) options. On the (JC) the Release Vjk will release the reservations on the operand registers. On the (JE) the Release Vjk will match a V0 - V7 and send a Release VFU to the (JF) option, along with a Release to the (JD) to release reservations on the functional unit.

On a Vector register Write operation from memory the release does not originate from the (VD) option but from the memory control (YE) option. The (YE) will generate the Port A/B/C Release when the (Ak/VL) words have been decremented to zero. The (YE) then sends the release to the (JD0) option to release the Port reservation A/B/C and also send the Release to the (JF) to release the (Vi 0 - 7) reservation on the (JC, VD) options. The (JC) upon receiving the Release Vi 0 - 7 will release the reservation on (Vi). The (VD), upon receiving the Release V0 - V7, will prevent the Write address from incrementing.

VECTOR REGISTERS
GENERAL

I. 8 Vector registers = V0 - V7

II. Each Vector register = 64_{10} or 100_8 Elements

III. Each Element = 64_{10} bits in length

IV. Elements are interleaved Even/Odd

V. Each Vector register has a Read Address register and a Write Address register.

VI. Each Vector register has a Read Vector Length register.

VII. Vector Register Instructions contain 1 or 2 operands; only one destination.

VIII. Per the instruction the operand Vj/Vk are steered to 1 of 10 Functional Units.

IX. Per the instruction the destinations are steered to 1 of 8 Vector registers from 1 of 10 Functional Units.

X. Vector Registers use a Bypass path when the Read Address = Write Address.

XI. On a Read Even/Odd or Write Even/Odd Address Conflict the Write Address is held for 1 CP.

XII. Automatic chaining as long as Write Address is greater than or equal to Read Address. Destination can be used as another instruction operand.

XIII. Vector instruction can hold after CIP by monitoring the Read and Write Address and Data Readies.

XIV. Upon issuing a Vector Instruction Reservation are held on: Operand, Destination, and Functional Units.

XV. Operand reservation is held until the Read Address is equal to the Vector Length register.

Hardware Trng.
YM07/11 J.E.S.

CRAY Y-MP VECTOR REGISTER INSTRUCTIONS

00200k	-	Transmit (Ak) to VL register
002000	-	Transmit 1 to VL register
023i01	-	Transmit (VL) to Ai
076ijk	-	Transmit (Vj, element (Ak)) to Si
077ijk	-	Transmit (Sj) to Vi element (Ak)
077i0k	-	Clear Vi element (Ak)
176i0k	-	Read (VL) words to Vi from Memory address ((A0) + (DBA)) incremented by (Ak)
176i00	-	Read (VL) words to Vi from Memory address ((A0) + (DBA)) incremented by 1
176i1k	-	Read (VL) words to Vi from Memory address ((A0) + (Vk) + (DBA))
1770jk	-	Write (VL) words from Vj to Memory address ((A0) + (DBA)) Increment by (Ak)
1770j0	-	Write (VL) words from Vj to Memory address ((A0) + (DBA)) incremented by 1
1771jk	-	Write (VL) words from Vj to Memory address ((A0) + (Vk) + (DBA))
155ijk	-	Integer sums of (Vj) and (Vk) to Vi

Hardware Trng.
YM07/12 J.E.S.

VECTOR REGISTERS (OPTIONS INVOLVED)

(JC0) Option

The (JC) option decodes the instruction at CIP time, and checks for the Vector instruction reservation on (Vi, Vj, Vk) before allowing the Vector instruction to advance into execution. The types of conditions the (JC) would Hold Issue on are:

(Vi, Vj, Vk), reservation - Check for (Vi, Vj or Vk) being reserved for operands; if (Vijk) are reserved as a destination and the next instructions would want to use the same Vector register as an operand it would be allowed to issue. This is called chaining.

076/077, 002 - Scalar reference will hold out any vector references until the Hold Issue drops on the Scalar reference.

SR Hold - Stop the issue if a Test & Set and holding or a Shared register is in progress.

The (Vi, Vj, Vk) reservation are cleared upon receiving the Release Vjk from the (VD) or Release Vi 0 - 7 from the (JF) option.

(JD0) Option

The (JD) option decodes the instruction at CIP time, and checks for the Vector functional units (VFU) reservation and the Ports A, B, C reservations, before allowing the Vector instructions to advance into execution. Types of conditions that the (JD) would Hold Issue on are:

VFU conflict - Conflict with Vector functional unit being reserved.

176 - 177, Scalar hold - Hold for 1 CP on a 176 - 177 instruction because of multiplexing (Ao) and (Ak) out of the A registers. Scalar hold occurs because a scalar cannot intermix with a Vector operation and vice versa.

Hold 003, 073 - Conflicts that occur with the Vector Logical functional unit and the Vector Mask register.

002 (i1 - 7) - Hold Issue on the 002X conflicts.

The VFU reservation is cleared upon receiving the Release FR, FA, FM, VA, VS, VL from the (JE0) option, which was generated on the (JE0) option from the (VD) options Release Vjk. The port reservation which is also held on the (JD0) option is cleared by the (YE0) decremented block length being equal to zero. The Release Memory port, however, is held on the (YO) option until the Bank conflict has been resolved.

(JE0) Option

The (JE0) option directs the Release Vjk to the correct functional unit "X", and directs the Go Data Vjk to the correct functional unit "X". To do these two operations the (JE) is given the (jk) field of the instruction and the Go FR, FA, FM, VA, VS, VL or Go VFU. What the (JE) will do with these two pieces of information is match the corresponding Release Vjk to the original load (Vjk), or the Go Data Vjk to the originally loaded (Vjk) and send off the Go Data VFU or the Release VFU.

(JF0) Option

The (JF) option is responsible for generating the Release Vi 0 - 7, and supplying the (VD) option with the Go Write V0 - 7 to load the FU result at the appropriate times. The (JF) contains a FU delay chain which the Go Write V0 - 7 for FU X are loaded into and delayed to clock period 4 time, when Go Write V0 - 7 is generated. To determine which (Vi) belongs to which FU, the (JF) is given the (Vi) and Go VFU. Upon getting the Go Data VFU from the (JE) option the (JF) will enter into the delay chain the correct (Vi0 - Vi7) from the match up.

(HC0, HD0) Option

The (HC0, HD0) options supplies the (JD, JE) option with the instruction (ghijk) decodes upon receiving the Go Issue signal.

(VA0 - VA15) Options

The 16 (VA) options steer the Write Data to 1 of 8 Vector registers. The Write Data can originate from 1 of 10 functional units within the CRAY Y-MP. Each (VA) option contains 4 consecutive bits of the 64-bit element for a total of 16 (VA) options. To select 1 of 8 Vector registers, the (VA) receives the (i) field of the instruction decode and generates (X0 -X7). (X0 - X7) will latch and hold a 4-bit VFU Select Code for 1 of 8 Vector registers. When the data arrives from the functional unit, a decode on the VFU Select will Enable the data to be steered to the appropriate Vector register.

(VB0 - VB15) Options

The (VB0 - VB15) options contain the lower 32 bits of the 64-bit elements, with each (VB) containing 2 bits of each 64-bit element. The 16 (VB) options steer the Read Vector operand (Vk/Vj) or (Sj) to 1 of 11 functional units. The (VB) receives a VFU Select Code which latches and holds the (Vj/Vk) instruction decode. The (Vj/Vk) is decoded and used to Enable the correct Vector register data (V0 - V7) to be sent out to 1 of 11 functional units. The (VBs) also contain the lower 32 bits of the Scalar registers. When the Vector instruction uses (Sj) instead of (Vj), (Sj) is simply read out of the (VBs) by the decode (j) field of the instruction, which could represent (S0 - S7).

Hardware Trng.
YM07/09 J.E.S.

(VC0 - VC15) Options

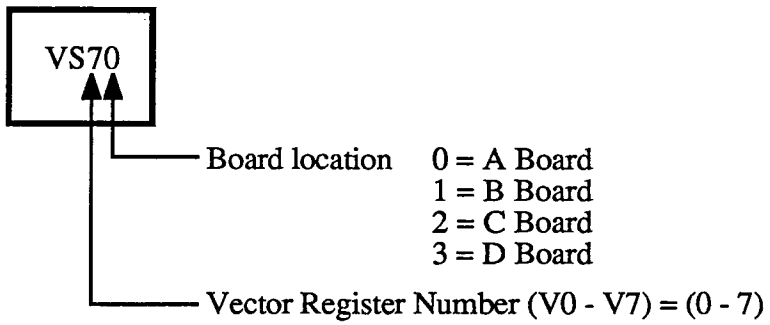
The (VC) options are identical to the (VB) options except that the (VCs) contain the upper 32 bits of each 64-bit element, and they can only output data to 10 functional units instead of 11. For a Memory Reference instruction, Gather/Scatter uses (Vk) as a relative address; in this case, (Vk) is only 32 bits in length and would be found on the (VB) options, but not in the (VC) options.

(VD0 - VD3) Options

The (VD) options control the reading and writing of the Vector registers which are on the (VS) options. Each (VD) option controls two Vector registers. (VD0) controls Vector register V0 and V1. The (VD) contains a number of registers in order to perform its function. The (VD) contains a Read Address register, a Write Address register, and a Read VL register. The Read/Write Address register is set to zero upon getting the Go Read/Write Sequence signal from the (JC) option, and incremented by plus 1. The Write address is increment by one when the Go Write is present, or terminated by the Release sent by the (JF) option. The Read Address register is incremented by plus 1 each clock period as long as the elements are available to be read and monitored by the data ready (V0 - V7) when an operand pair is required. The Read address will be stopped from incrementing if it is equal to the Read Vector Length register, or if the Read address is equal to the Write address. If the Read address is equal to the Write address the bypass path would be used. However, the Read address cannot be greater than the Write address on a chaining condition.

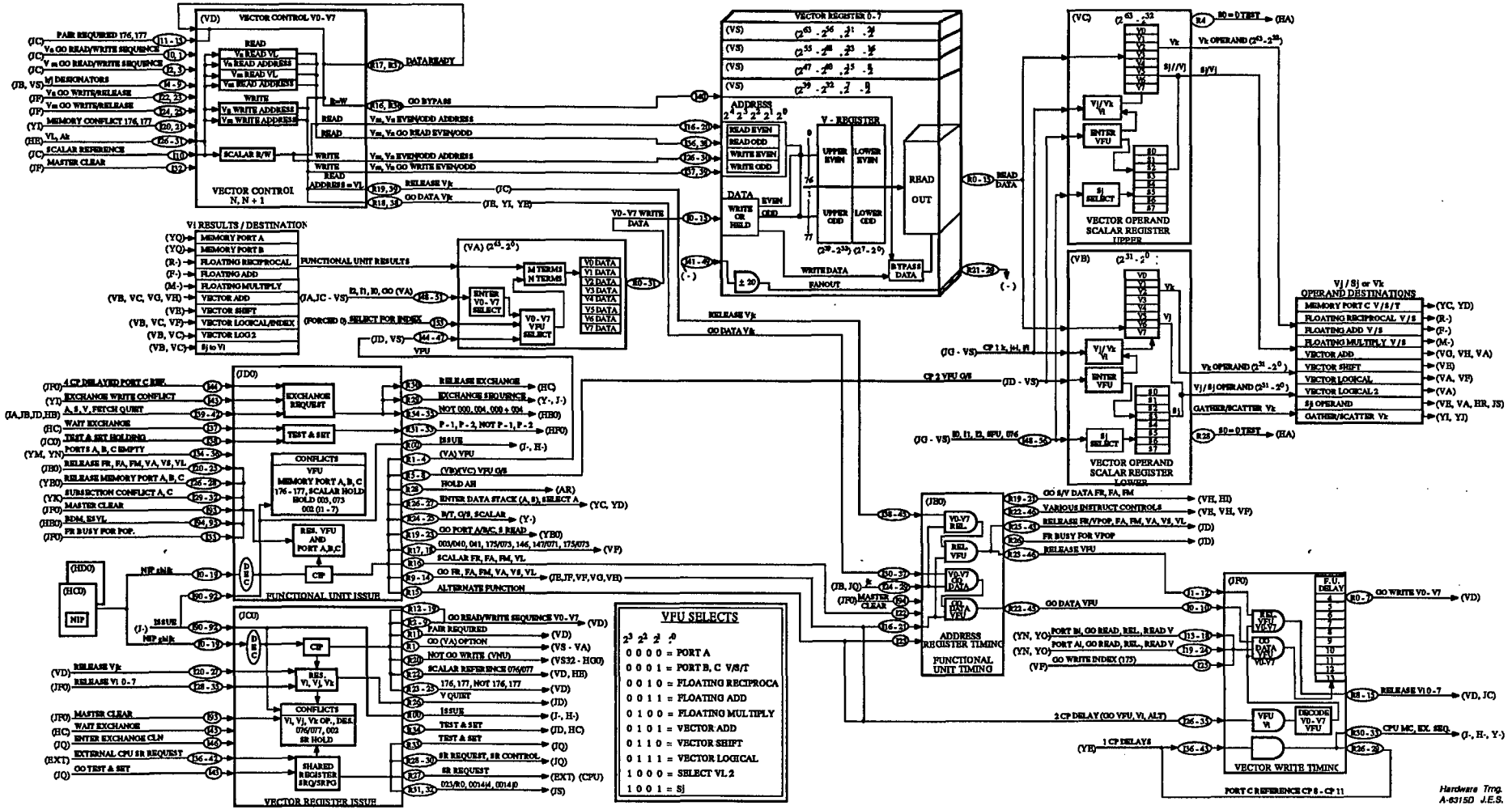
(VS0 - VS73) Options

The (VS) options contain the Vector registers. Four (VS) make up one Vector register, giving a total of 32 (VS) options for all eight registers (V0 - V7). Vector register 0 is found on (VS0), (VS1), (VS2), and (VS3), Vector register 7 is found on (VS70), (VS71), (VS72), and (VS73). When looking at Vector register 7, it can be noted that (VS70) is found on the A Board, (VS71) is on the B Board, etc. as shown on the following page.



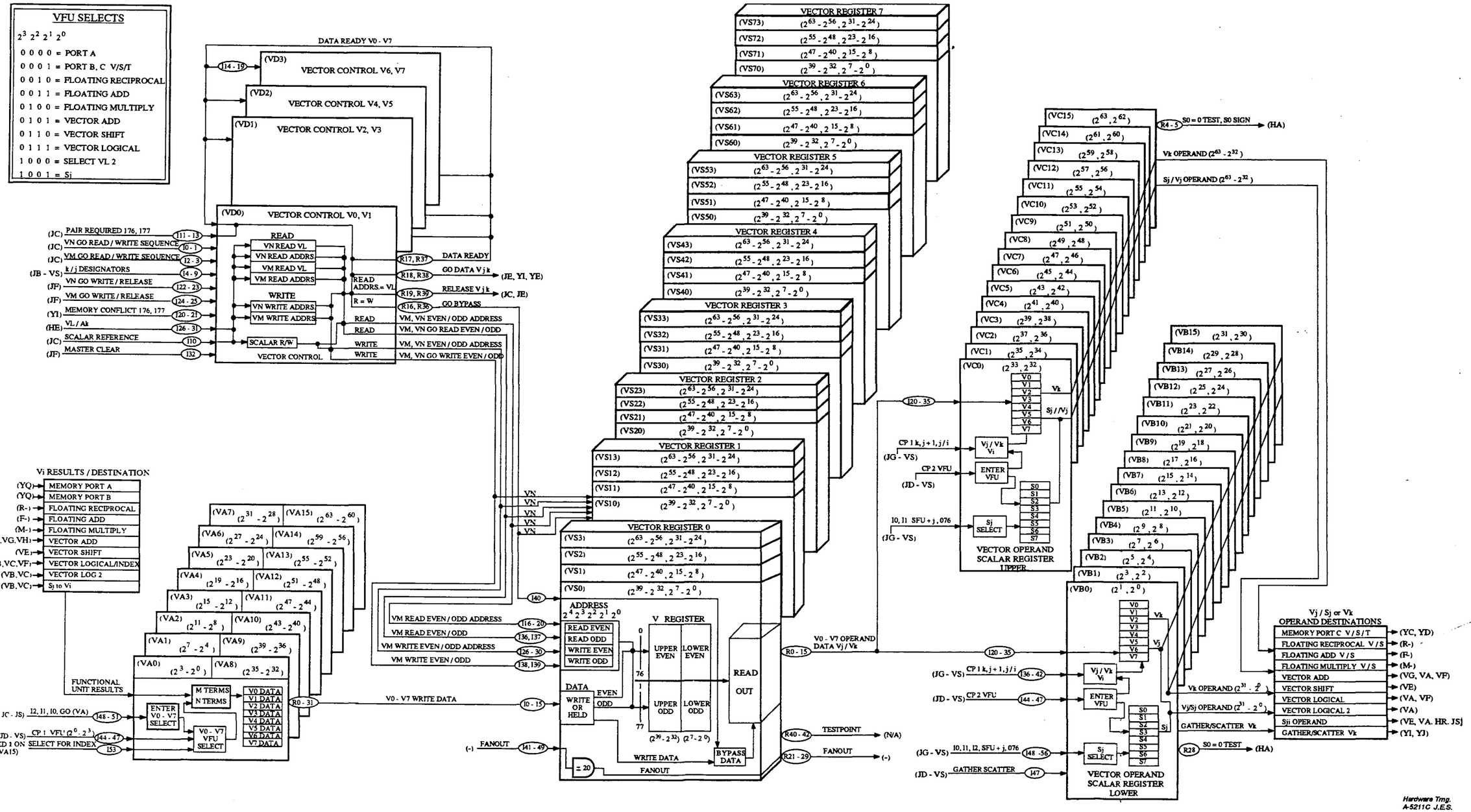
Each (VS) option contains 16 bits of the 64-bit element. However, the 16 bits are divided into two 8-bit groups, with the groups being 24 bits apart. For example, (VS0) contains bits (2^{32} - 2^{39}) and bits (2^{0} - 2^{7}). The Vector register elements are divided into 32 even and 32 odd elements. The even or odd elements are further divided into 32 lower registers or 32 upper registers. The even lower register on the (VS0) option would represent data bits (2^{0} - 2^{7}), while the even upper register would represent data bits (2^{32} - 2^{39}) on the (VS0) option.

Hardware Trng.
YM07/10 J.E.S.



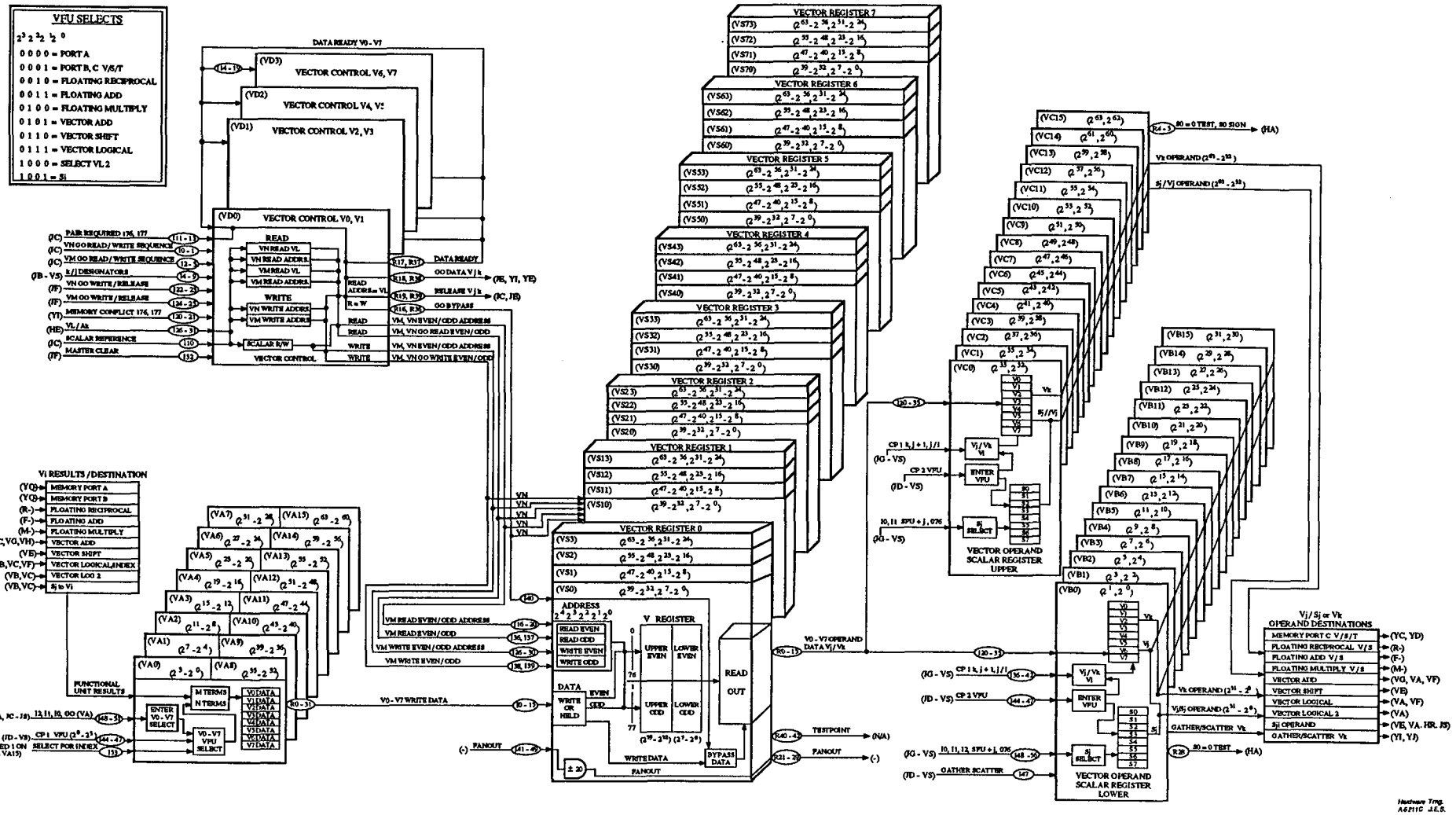
CRAY Y-MP VECTOR REGISTER BLOCK DIAGRAM

PART I

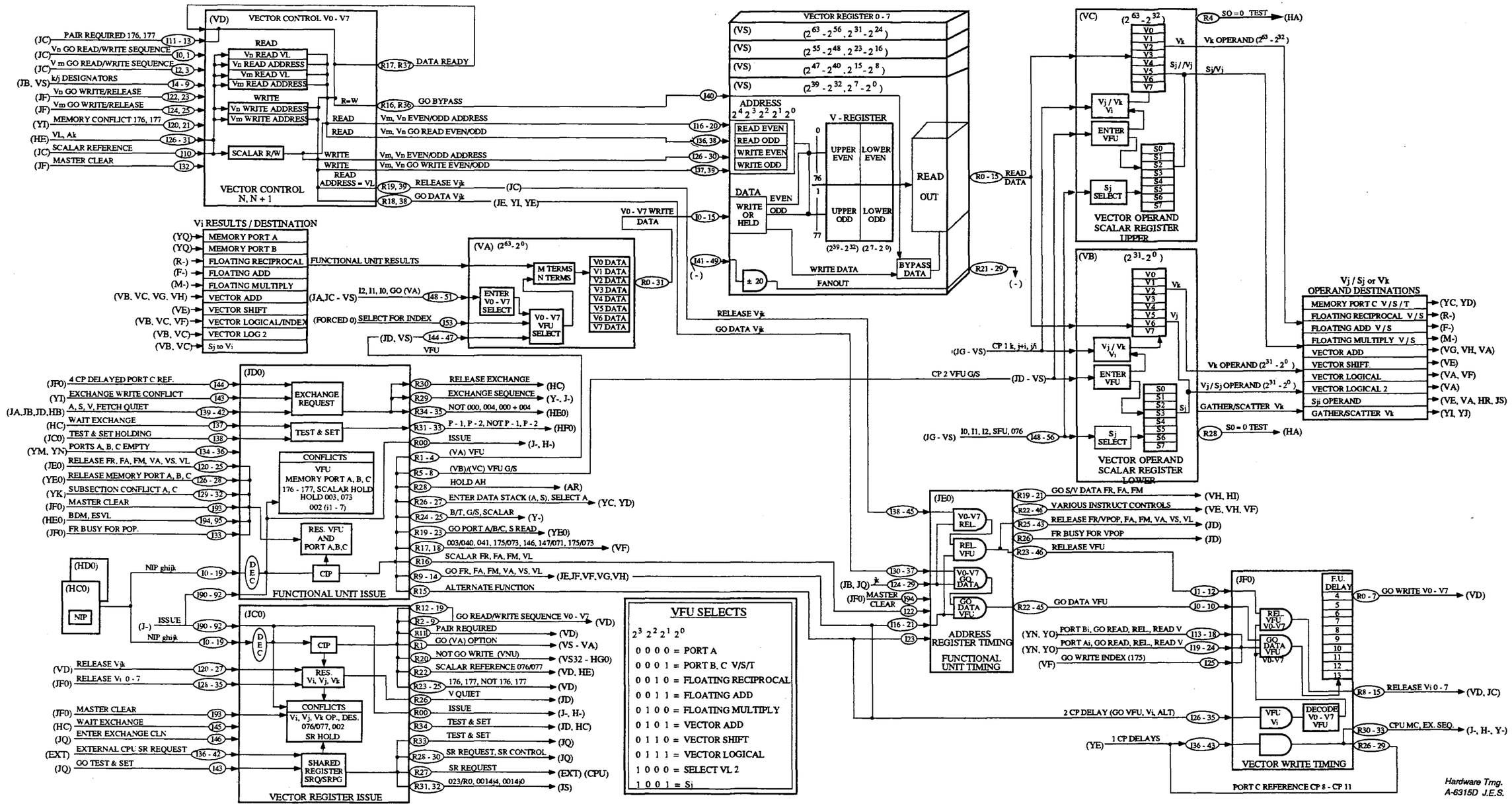


CRAY Y-MP VECTOR REGISTER BLOCK DIAGRAM
PART II

CRAY PROPRIETARY

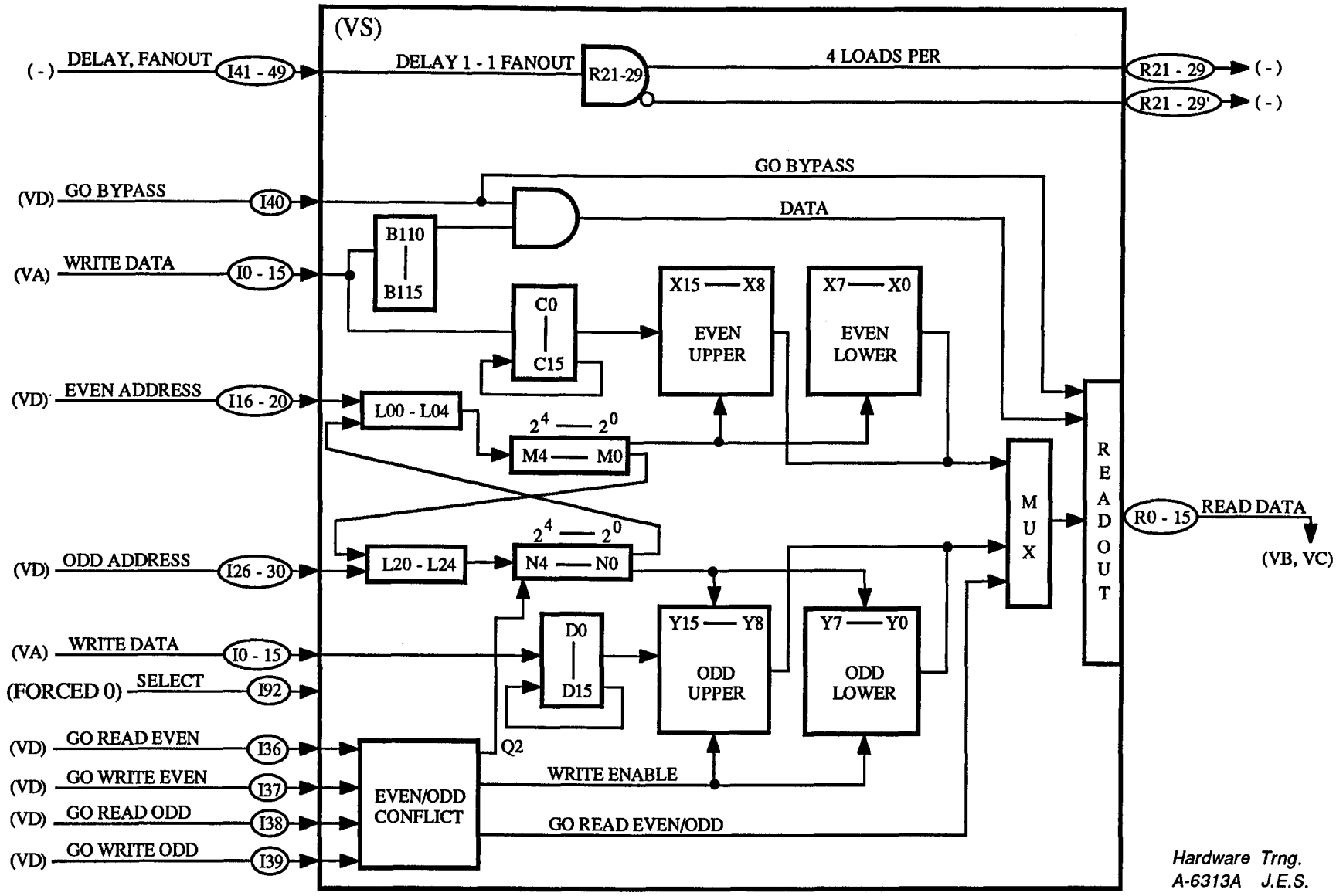


CRAY Y-MP VECTOR REGISTER BLOCK DIAGRAM
PART II



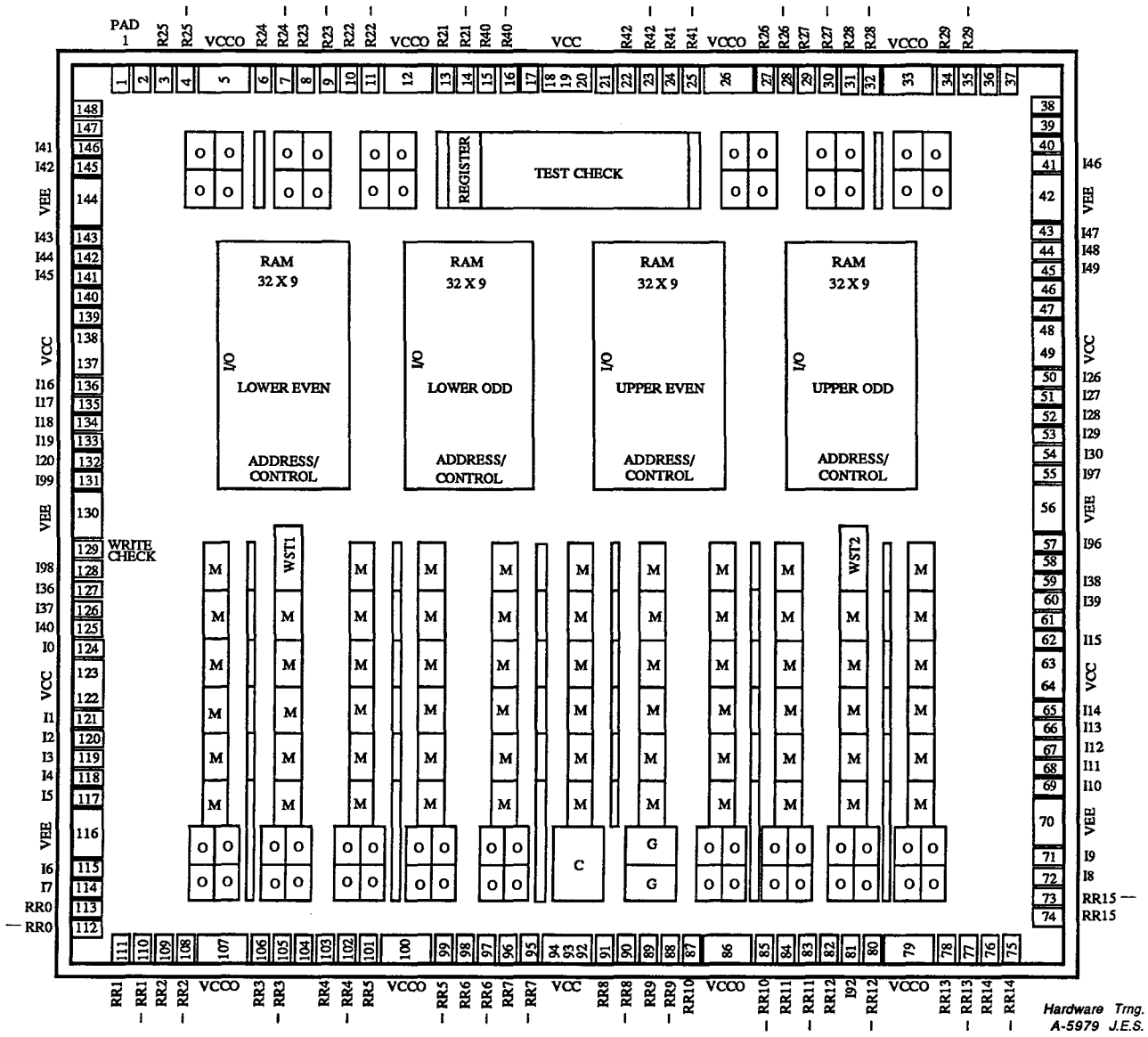
CRAY Y-MP VECTOR REGISTER BLOCK DIAGRAM
PART I

CRAY PROPRIETARY

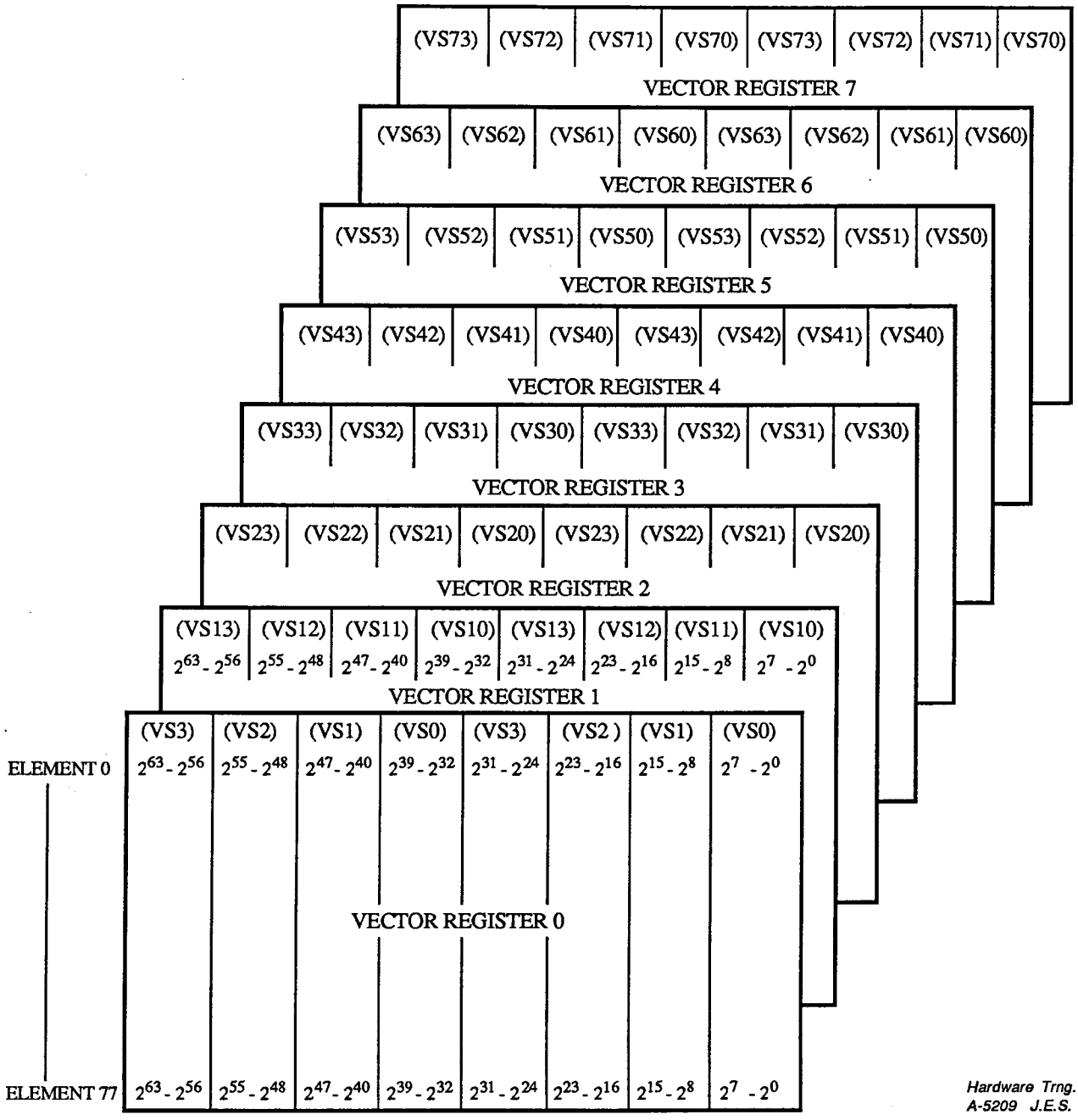


CRAY Y-MP VECTOR REGISTER (VS) OPTION
BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP MCA1500M CHIP PLAN



CRAY Y-MP VECTOR REGISTER BIT MAPPING

	(VS3) $(2^{63}-2^{56})$	(VS2) $(2^{55}-2^{48})$	(VS1) $(2^{47}-2^{40})$	(VS0) $(2^{39}-2^{32})$	(VS3) $(2^{31}-2^{24})$	(VS2) $(2^{23}-2^{16})$	(VS1) $(2^{15}-2^{8})$	(VS0) $(2^{7}-2^{0})$	
ELEMENT 0				X15-X8	← EVEN UPPER REG.			X07-X00	← EVEN LOWER REG.
1				Y15-Y8	← ODD UPPER REG.			Y07-Y00	← ODD LOWER REG.
2				X15-X8				X07-X00	
3				Y15-Y8				Y07-Y00	
4				X15-X8				X07-X00	
5				Y15-Y8				Y07-Y00	
6				X15-X8				X07-X00	
7				Y15-Y8				Y07-Y00	
10				X15-X8				X07-X00	
11				Y15-Y8				Y07-Y00	
12				X15-X8				X07-X00	
13				Y15-Y8				Y07-Y00	
14				X15-X8				X07-X00	
15				Y15-Y8				Y07-Y00	
16				X15-X8				X07-X00	
17				Y15-Y8				Y07-Y00	
20				X15-X8				X07-X00	
21				Y15-Y8				Y07-Y00	
22				X15-X8				X07-X00	
23				Y15-Y8				Y07-Y00	
24				X15-X8				X07-X00	
25				Y15-Y8				Y07-Y00	
26				X15-X8				X07-X00	
27				Y15-Y8				Y07-Y00	
30				X15-X8				X07-X00	
31				Y15-Y8				Y07-Y00	
32				X15-X8				X07-X00	
33				Y15-Y8				Y07-Y00	
34				X15-X8				X07-X00	
35				Y15-Y8				Y07-Y00	
36				X15-X8				X07-X00	
37				Y15-Y8				Y07-Y00	
40				X15-X8				X07-X00	
41				Y15-Y8				Y07-Y00	
42				X15-X8				X07-X00	
43				Y15-Y8				Y07-Y00	
44				X15-X8				X07-X00	
45				Y15-Y8				Y07-Y00	
46				X15-X8				X07-X00	
47				Y15-Y8				Y07-Y00	
50				X15-X8				X07-X00	
51				Y15-Y8				Y07-Y00	
52				X15-X8				X07-X00	
53				Y15-Y8				Y07-Y00	
54				X15-X8				X07-X00	
55				Y15-Y8				Y07-Y00	
56				X15-X8				X07-X00	
57				Y15-Y8				Y07-Y00	
60				X15-X8				X07-X00	
61				Y15-Y8				Y07-Y00	
62				X15-X8				X07-X00	
63				Y15-Y8				Y07-Y00	
64				X15-X8				X07-X00	
65				Y15-Y8				Y07-Y00	
66				X15-X8				X07-X00	
67				Y15-Y8				Y07-Y00	
70				X15-X8				X07-X00	
71				Y15-Y8				Y07-Y00	
72				X15-X8				X07-X00	
73				Y15-Y8				Y07-Y00	
74				X15-X8				X07-X00	
75				Y15-Y8				Y07-Y00	
76				X15-X8				X07-X00	
77				Y15-Y8				Y07-Y00	

Hardware Trng.
A-5210 J.E.S.

CRAY Y-MP VECTOR REGISTER 0
ARRAY LOCATIONS

VECTOR/SCALAR OPERANDS STEERING

(VC15)	(VC14)	(VC13)	(VC12)	(VC11)	(VC10)	(VC9)	(VC8)	(VC7)	(VC6)	(VC5)	(VC4)	(VC3)	(VC2)	(VC1)	(VC0)
$2^{63},2^{62}$	$2^{61},2^{60}$	$2^{59},2^{58}$	$2^{57},2^{56}$	$2^{55},2^{54}$	$2^{53},2^{52}$	$2^{51},2^{50}$	$2^{49},2^{48}$	$2^{47},2^{46}$	$2^{45},2^{44}$	$2^{43},2^{42}$	$2^{41},2^{40}$	$2^{39},2^{38}$	$2^{37},2^{36}$	$2^{35},2^{34}$	$2^{33},2^{32}$

Vector/Scalar J or K Operand Selection to:

Sji operand; Scalar Add enable, carry; S0 = 0 test, sign; S0 - sign; Memory Port C; Floating Reciprocal, Floating Add; Floating Multiply; Vector Add (sum, enable); Vector Shift, Vector Logical, Vector Logical 2.

(VB15)	(VB14)	(VB13)	(VB12)	(VB11)	(VB10)	(VB9)	(VB8)	(VB7)	(VB6)	(VB5)	(VB4)	(VB3)	(VB2)	(VB1)	(VB0)
$2^{31},2^{30}$	$2^{29},2^{28}$	$2^{27},2^{26}$	$2^{25},2^{24}$	$2^{23},2^{22}$	$2^{21},2^{20}$	$2^{19},2^{18}$	$2^{17},2^{16}$	$2^{15},2^{14}$	$2^{13},2^{12}$	$2^{11},2^{10}$	$2^{9},2^{8}$	$2^{7},2^{6}$	$2^{5},2^{4}$	$2^{3},2^{2}$	$2^{1},2^{0}$

Vector/Scalar J or K Operand Selection to:

Sji operand; Scalar Add enable, carry; Gather/Scatter Memory Port C; Floating Reciprocal; Floating Add; Floating Multiply; Vector Add (sum, enable, carry); Vector Shift; Vector Logical; Vector Logical 2; S0 = 0 test, 0, 1.

VECTOR RESULTS STEERING

(VA15)	(VA14)	(VA13)	(VA12)	(VA11)	(VA10)	(VA9)	(VA8)	(VA7)	(VA6)	(VA5)	(VA4)	(VA3)	(VA2)	(VA1)	(VA0)
$2^{63}-2^{60}$	$2^{59}-2^{56}$	$2^{55}-2^{52}$	$2^{51}-2^{48}$	$2^{47}-2^{44}$	$2^{43}-2^{40}$	$2^{39}-2^{36}$	$2^{35}-2^{32}$	$2^{31}-2^{28}$	$2^{27}-2^{24}$	$2^{23}-2^{20}$	$2^{19}-2^{16}$	$2^{15}-2^{12}$	$2^{11}-2^{8}$	$2^{7}-2^{4}$	$2^{3}-2^{0}$

Vector Results (Vi) Selected From:

Memory Port A, B; Floating Reciprocal; Floating Add; Floating Multiply; Vector Add (sum, carry); Vector Shift (shift 1, overlap), Vector Index, Vector Logical; Vector Logical 2 Results; Sji.

Hardware Tmg.
A-5208A J.E.S.

CRAY Y-MP VECTOR STEERING (VA, VB, VC) OPTIONS

VECTOR ADD FUNCTIONAL UNIT

The Vector Add functional unit is very similar to the Scalar Add functional unit. The Vector Add functional unit starts the add process on the (VB, VC) options, after the Vj/Vk operands or Sj/Vk operands have been selected from the Vector/Scalar registers. During a Scalar/Vector Add, the Sj copy is held on the (VB, VC) options for the duration of the Vector Add process, which is dependent on the Vector Length register. When a Subtract instruction is issued, the Vk operand is complemented on the (VB, VC) options. A partial Add is performed on the (VB, VC) options, where the bits are identified as Sums, Enables, or Carries. The Enables and Carries are sent to the (VG, VH) options to determine how far the Carries will propagate. The Sums and a few of the Carries are sent to the (VA) options to determine how far on a group level the Carry bit will propagate. When the (VG, VH) have determined how far the Carry will propagate, this information is sent to the (VA) options to determine the final summation. The Final Vector Add result is then routed into the correct (Vs) Vector Result register through a multiplexer, which previously used the VFU select code to choose results from the Vector Add.

The VFU select code for Vector Add is a (0101). The Vector Add takes 2 CPs inside the functional unit to Add two 64-bit operands.

CRAY Y-MP VECTOR ADD FUNCTIONAL UNIT INSTRUCTIONS

INSTRUCTIONS

154ijk	Integer Sum of (Sj) and (Vk) to Vi
154i0k	Transmit (Vk) to Vi
155ijk	Integer Sum of (Vj) and (Vk) to Vi
156ijk	Integer Differences of (Sj) and (Vk) to Vi
156i0k	Transmit Negative of (Vk) to Vi
157ijk	Integer Difference of (Vj) and (Vk) to Vi

CAL FORMAT

Vi	Sj + Vk	Sum of (Sj) and (Vk)
Vi	Vk	Transmit (Vk) to Vi
Vi	Vj + Vk	Sum of (Vj) and (Vk)
Vi	Sj - Vk	Difference of (Sj) - (Vk)
Vi	-Vk	Transmit Negative (Vk) to Vi
Vi	Vj - Vk	Difference of (Vj) - (Vk)

Hardware Trng.
YM07/03 J.E.S.

VECTOR ADD FUNCTIONAL UNIT OPTIONS INVOLVED

(VA0-VA15) Options

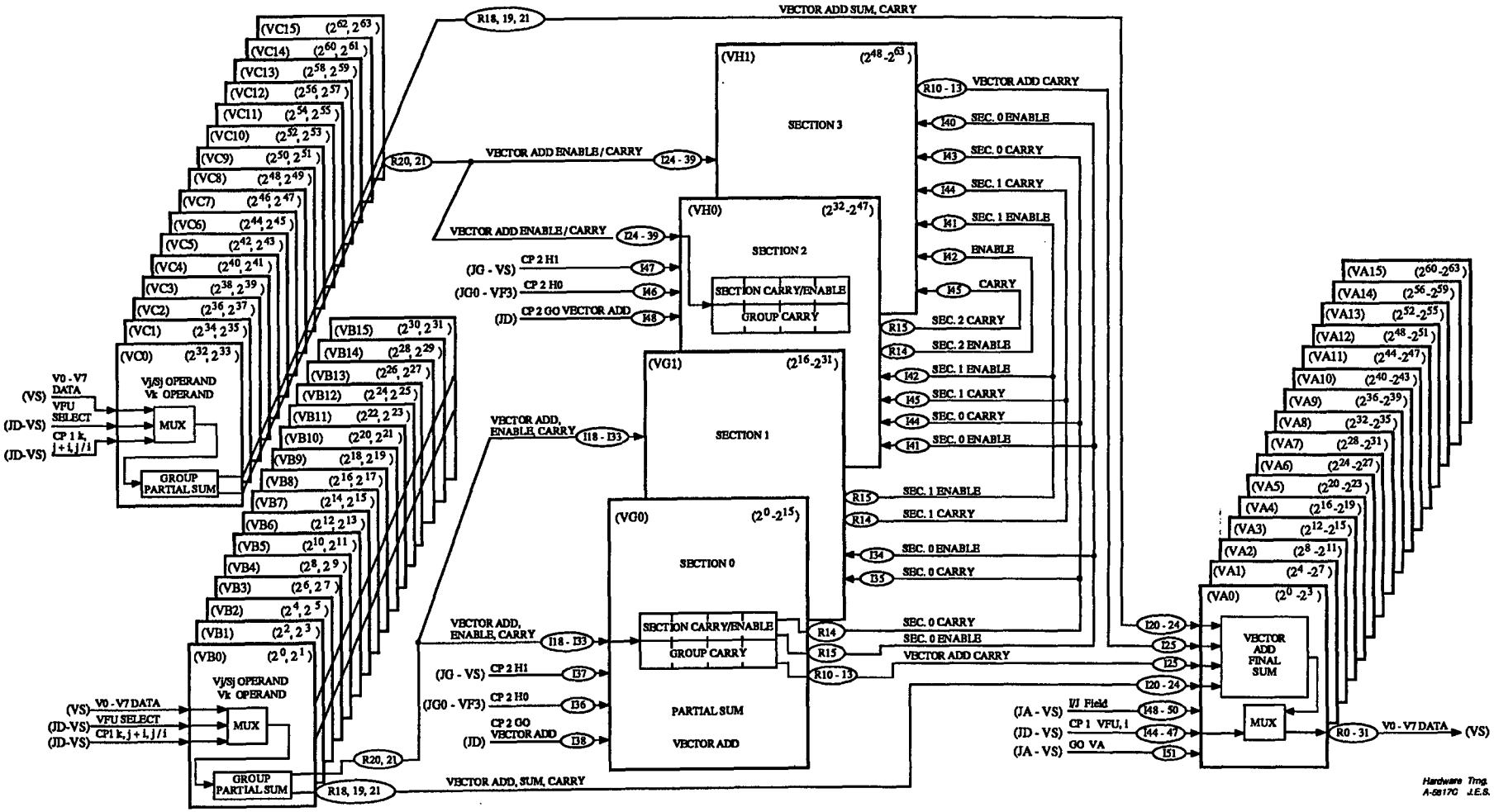
The (VA0-VA15) options each contain 4 bits of the final result of the Vector Add process. Early in the summation the (VA) options receive the partial Sum and part of the Carry information from the (VB, VC) options. The (VA) options partially add the sum and carry conditions and wait for the final carry to come in from the (VG, VH) options. Then the (VA) will perform the final summation and send the results to the correct Vector register.

(VB-VC) Options

The (VB-VC) options are identical and contain 2 bits of the Vj/Vk operands or the Sj/Vk operands. From the instruction decode, if h bit $2^0 = 0$ the (VB-VC) options will select a Sj operand and a Vk operand. The Sj operand is latched and held by the L58 term, which is made from h bit 2^0 being equal to a zero. When h bit 2^0 equals a one then a Vj and Vk operand are selected at the appropriate times, after the (VD) option has addressed the (VS) option and the (VS) option has passed the correct operands to the (VB-VC) options.

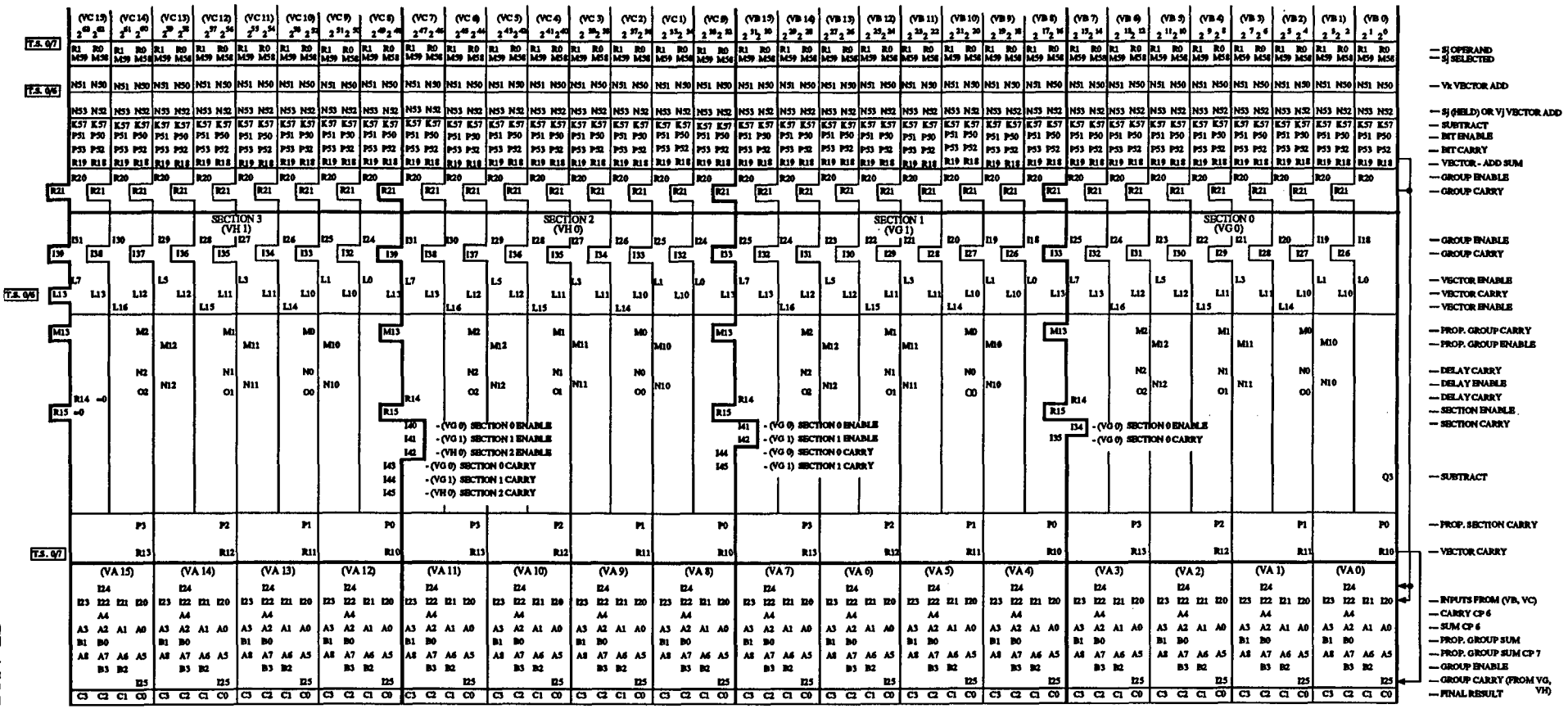
(VG-VH) Options

Each (VG-VH) option handles 16 bits of the 64-bit Addition process the (VG0) handles bits (2^0-2^{15}), while the (VG1) handles bit (2^{16}-2^{31}). The (VH0) handles bits (2^{32}-2^{47}), while the (VH1) handles bits (2^{48}-2^{63}). Because the addition is right justified, the (VG0) will pass to (VG1, VH0, VH1) a Section 0 Enable and a Section 0 Carry. The Section 0 Enable will produce a Carry if a subtract instruction was issued, and a plus one will be added into the lowest position. The plus one is used to complete the twos complement arithmetic for a subtract. The plus one originates from h bit 2^1 of the instruction decode. The (VG1) will pass a Section 1 Enable and a Section 1 Carry to the (VH0, VH1) options, when the appropriate conditions exist. The (VH0) option will pass a Section 2 Enable and a Section 2 Carry to the (VH1) option. The (VH1) does not pass any Section Enable or Carry information because the Vector Add does not perform an End Around Carry: any bits above 2^{63} are lost.



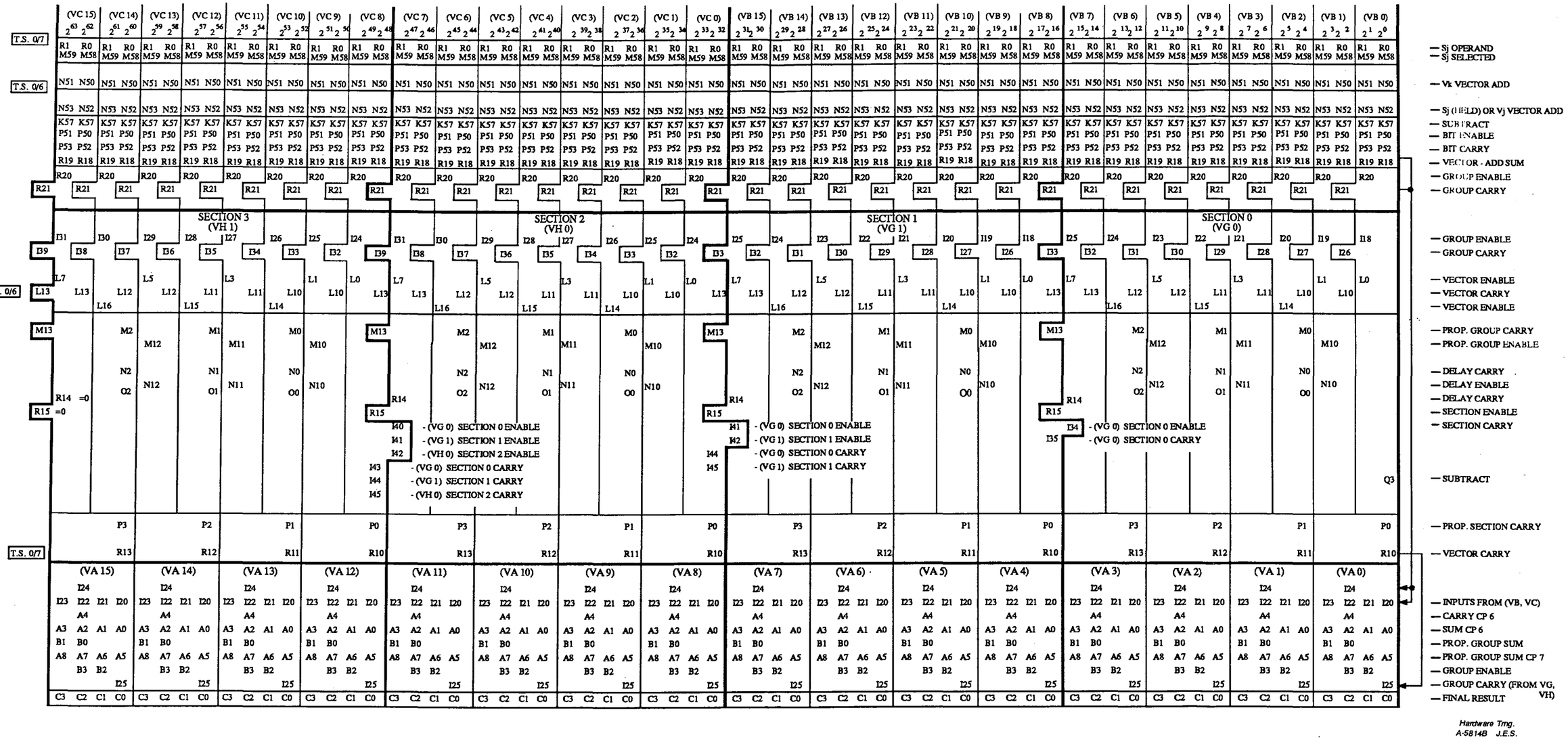
CRAY Y-MP VECTOR ADD FUNCTIONAL UNIT BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP VECTOR ADD FLOW CHART

CRAY PROPRIETARY



CRAY Y-MP VECTOR ADD FLOW CHART

CRAY PROPRIETARY

VECTOR SHIFT FUNCTIONAL UNIT

The Vector Shift functional unit uses the same option types as the Scalar Shift. However, the Vector Shift uses different (VE) options, so as not to conflict with the Scalar Shift.

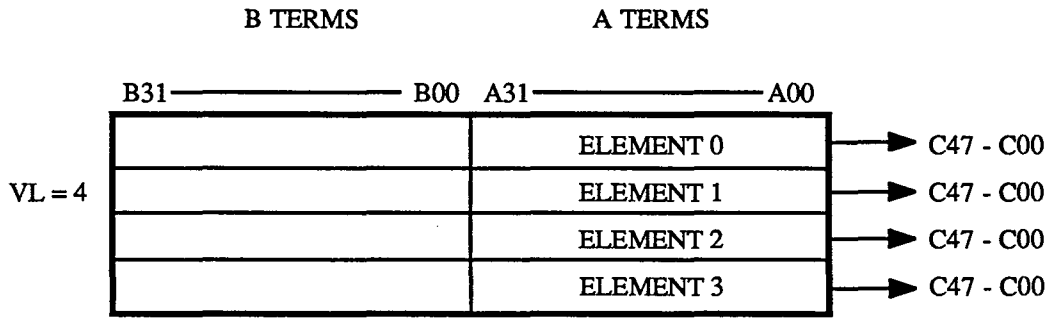
When performing a Vector Shift, the elements of the Vj operand are each shifted the number of places specified by the Ak register. For right shifts, the Ak shift count is put into the twos complement form; this is done on the (VG) option. All shifts are performed on the (VE) options, except a shift of one which is done on the (VA) options.

The Vector Shift has two extra signals that the Scalar Shift does not need, and those are Vector Begin and Vector End used during a Vector Double Right and Vector Double Left shift. The Vector Begin is used on a Right Double Shift to add leading zeroes to element zero before it is shifted. This causes the first valid shift to have zeroes added to it equal to the number of spaces that was right shifted . The Vector End signal is generated on a Left Double shift, and is used to add a field of zeroes to the end of the last element before shifting it. This causes the last element to have a number of zeroes appended to it equal to the number of spaces it was left shifted. As with the Scalar Double shifts, if any bits from 2^7 - 2^{31} of Ak are set, the results of each shift going through the (VE) options will be zeroed, this effect being caused by enabling both Vector Begin and Vector End.

Hardware Trng.
YM05/17 J.E.S.

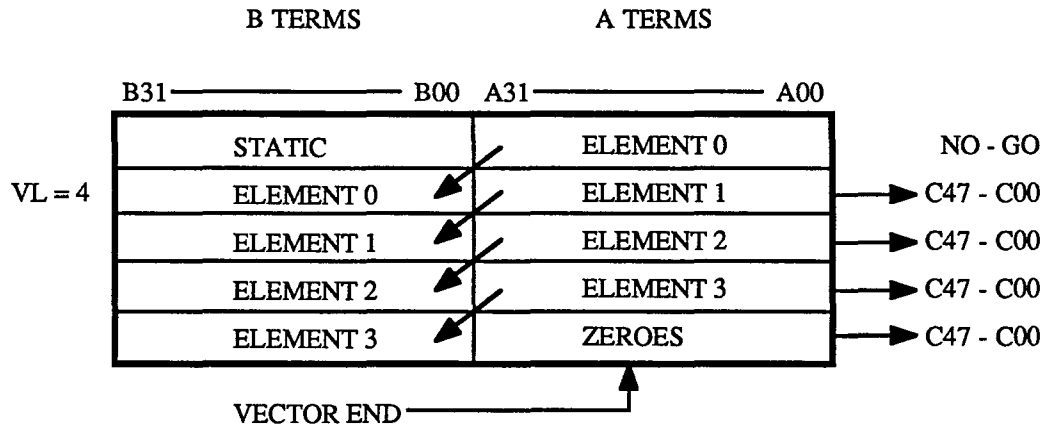
CRAY Y-MP VECTOR SHIFT OPERATION

Single Vector Shift 150, 151 Instruction



For a Vector Single Shift, Element 0 of Vj comes into (VE) options and goes to C47 - C00 terms dependent on the shift count followed by Element 1, then Element 2, and finally Element 3 when the Vector Length = 4.

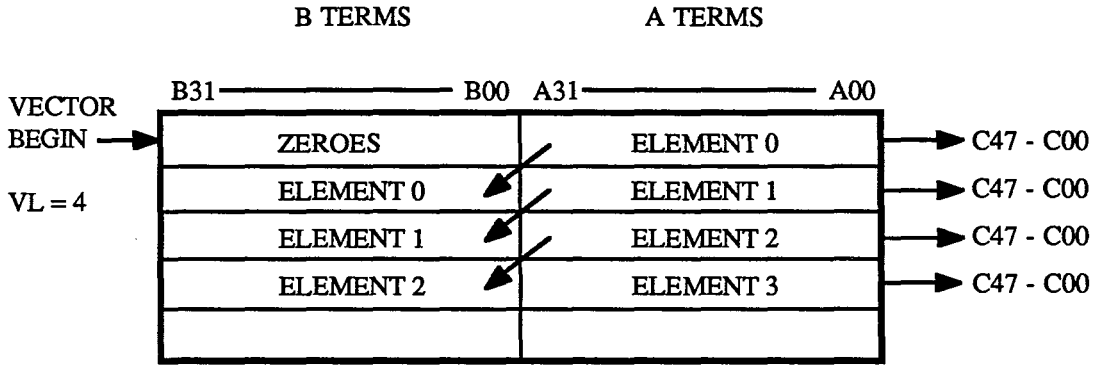
Left Double Vector Shift 152 Instruction



For a Left Double Vector Shift, Element 0 of Vj comes into the (VE) options. The output of this shift is lost to No Go Write on the (VS) option; however, a copy of Element 0 is saved in the B terms. When Element 1 arrives 1 clock period (CP) behind Element 0, the two Elements 0 and 1 are sent to the C47 - C00 terms dependent on the shift count as one Double Left Shift operation. The following elements are paired in a similar manner. On the last element, Element 3, is paired with zeroes generated by Vector End. Vector End causes a zero fills on the end of a Left Double Shift.

Hardware Trng.
YM05/18A J.E.S.

Right Double Vector Shift 153 Instruction



For a Right Double Vector Shift, Element 0 is paired with zeroes, which are generated by Vector Begin. This causes a Right Double Shift to be preceded with zeroes. The next clock period, Element 1, is paired with a copy of Element 0 and sent through the functional unit on C47 - C00 terms. This continues until the last element arrives, which is Element 3, paired with a copy of Element 2. A Right Double Shift takes 1 less CP than a Left Double Shift because a Right Double Shift does not have to wait for Element 1 to begin the shift.

CRAY Y-MP VECTOR SHIFT FUNCTIONAL UNIT INSTRUCTIONS

INSTRUCTIONS

150ijk	Shift (Vj) Left (Ak) Places to Vi
150ij0	Shift (Vj) Left One Place to Vi
151ijk	Shift (Vj) Right (Ak) Places to Vi
151ij0	Shift (Vj) Right One Place to Vi
152ijk	Double Shift (Vj) Left (Ak) Places to Vi
152ij0	Double Shift (Vj) Left One Place to Vi
153ijk	Double Shift (Vj) Right (Ak) Places to Vi
153ij0	Double Shift (Vj) Right One Place to Vi

CAL FORMAT

Vi	Vj<Ak	Vj Shifted Left by Ak Places to Vi
Vi	Vj<1	Vj Shifted Left One Place to Vi
Vi	Vj>Ak	Vj Shifted Right Ak Places to Vi
Vi	Vj>1	Vj Shifted Right One Place to Vi
Vi	Vj, Vj<Ak	Double Left Shift Ak Places to Vi
Vi	Vj, Vj<1	Double Left Shift One Place to Vi
Vi	Vj, Vj>Ak	Double Right Shift Ak Places to Vi
Vi	Vj, Vj>1	Double Right Shift One Place to Vi

Hardware Trng.
YM05/21 J.E.S.

VECTOR SHIFT FUNCTIONAL UNIT - OPTIONS INVOLVED

(VA0 - VA15) Options

The (VA0 - VA15) options perform the shift of 1 when the Vector Shift 1 control signal is present. Each (VA) option contains 4 bits of the final result of the shift, which are then steered to the correct Vector register specified by the Enter V0-V7 select signals. Since the (VA) option also steers other Vector functional unit data to the Vector registers, a Vector functional unit select code is assigned to enable the correct functional unit data into the Vector register V0-V7. For the Vector Shift functional unit, the Vector functional unit select code is equal to (0110).

(VB - VC) Options

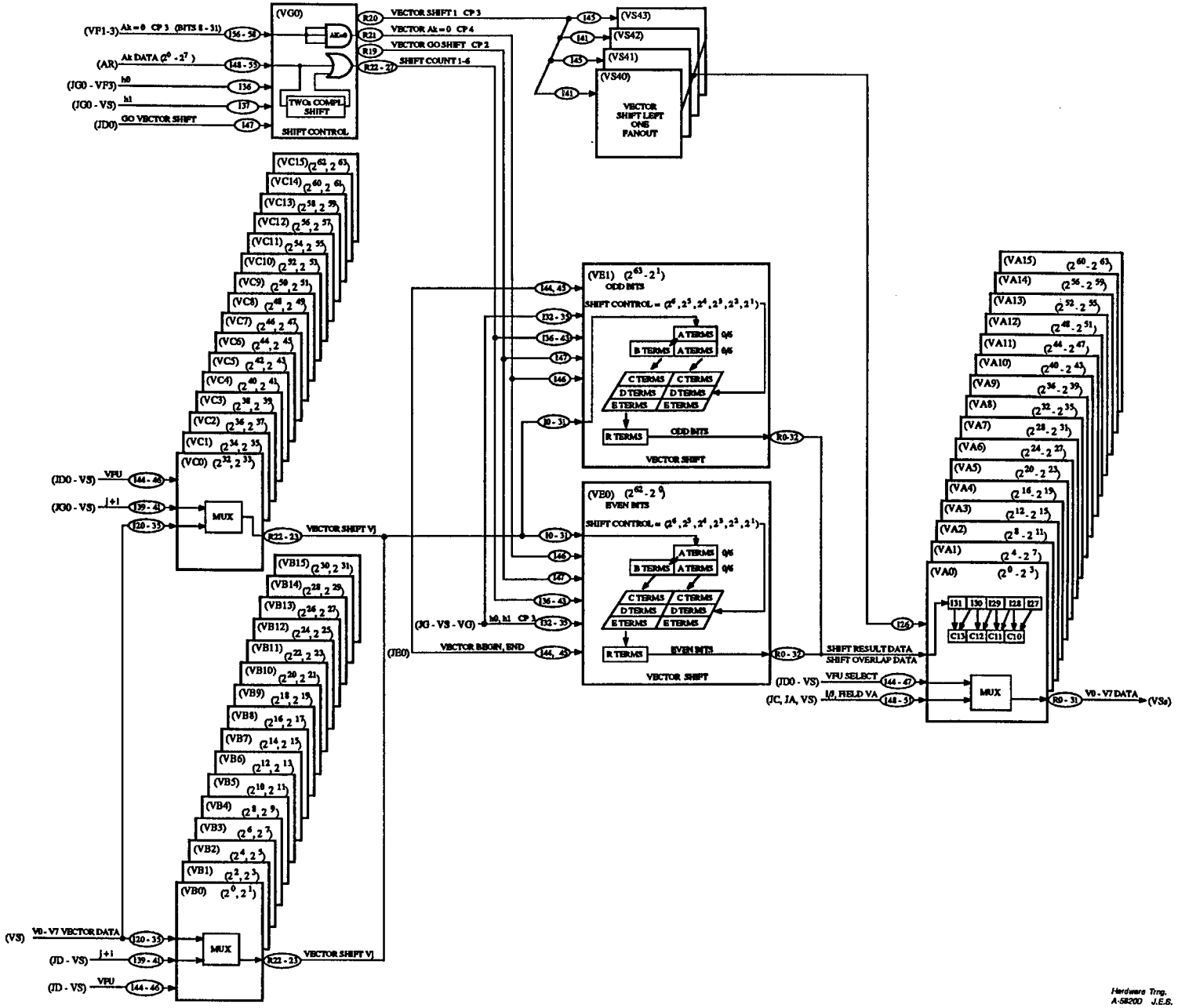
The (VB - VC) options steer the elements of Vj to the Vector Shift functional unit. To do this, the (VB - VC) need the (j) decode of the instruction to select the correct V0-V7 register, as well as the Vector functional unit select code to steer the operand to the Vector Shift functional unit.

(VE0 - VE1) Options

The (VE) option handles either the even numbered bits or the odd numbered bits of the Vj operand. The (VE) performs shifts of (64, 32, 16, 8, 4, 2). During a right shift the twos complement of the shift count is sent to the (VE) option by the (VG) option. The (VE) receives the h0, h1 instruction decode to determine if a right or left shift is to be performed, and then activates the appropriate control terms. The (VE) options also receive the Vector Begin and Vector End signals. Vector Begin is used on a right double shift and causes leading zeroes to be concatenated to element 0 before it has been shifted. Vector End is used on a left Double Shift, and causes trailing zeroes to be concatenated to the last element.

(VG0) Option

The (VG) supplies the (VE) and (VA) options with the needed shift control terms. During a right shift, the (VE) will perform the twos complement on the shift count. The (VEs) will receive shift control bits (2^6, 2^5, 2^4, 2^3, 2^2, 2^1) while the (VAs) receive (2^0), or shift 1 control terms. The (VG) also checks Ak bits (2^7-2^{31}) to be sure they are all zero, and if so the shift is allowed to be performed on the (VE) options.



CRAY Y-MP VECTOR SHIFT SINGLE/DOUBLE BLOCK DIAGRAM

$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}\ 2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$ (VB0)

$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}\ 2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$ (VE1)

I31 I30 I29 I28 I27 I26 I25 I24 I23 I22 I21 I20 I19 I18 I17 I16 | I15 I14 I13 I12 I11 I10 I09 I08 I07 I06 I05 I04 I03 I02 I01 I00

T.S. 0/7 A31 A30 A29 A28 A27 A26 A25 A24 A23 A22 A21 A20 A19 A18 A17 A16 | A15 A14 A13 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00

LEFT SINGLE SHIFT:

A0 - A31 ----► C16 - C47 SHIFT $\overline{32}$, H2

A0 - A15 ----► C32 - C47 SHIFT 32, H3

RIGHT SINGLE SHIFT:

A16 - A31 ----► C00 - C15 SHIFT $\overline{32}$, H0 (AFTER TWOs COMPLEMENT)

A00 - A31 ----► C00 - C31 SHIFT 32, H1 (AFTER TWOs COMPLEMENT)

SHIFT DERIVED FROM (Ak) FIELD OF THE INSTRUCTION.

		$2^4, 2^3$
C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32 | C31 C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16 | C15 C14 C13 C12 C11 C10 C09 C08 C07 C06 C05 C04 C03 C02 C01 C00	SHIFT 32 = 2^5	
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 24	1 1
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 16	1 0
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 8	0 1
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12	SHIFT 0	0 0
		$2^2, 2^1$
D47 D46 D45 D44 D43 D42 D41 D40 D39 D38 D37 D36 D35 D34 D33 D32 D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12 ALIGNMENT		
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 6	1 1
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 4	1 0
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 2	0 1
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15	SHIFT 0	0 0
E47 E46 E45 E44 E43 E42 E41 E40 E39 E38 E37 E36 E35 E34 E33 E32 E31 E30 E29 E28 E27 E26 E25 E24 E23 E22 E21 E20 E19 E18 E17 E16 E15 ALIGNMENT		

T.S. 0/7 R31 R30 R29 R28 R27 R26 R25 R24 R23 R22 R21 R20 R19 R18 R17 R16 R15 R14 R13 R12 R11 R10 R09 R08 R07 R06 R05 R04 R03 R02 R01 R00 R32 ← OVERLAP DATA

$2^{62}\ 2^{60}\ 2^{58}\ 2^{56}\ 2^{54}\ 2^{52}\ 2^{50}\ 2^{48}\ 2^{46}\ 2^{44}\ 2^{42}\ 2^{40}\ 2^{38}\ 2^{36}\ 2^{34}\ 2^{32}\ 2^{30}\ 2^{28}\ 2^{26}\ 2^{24}\ 2^{22}\ 2^{20}\ 2^{18}\ 2^{16}\ 2^{14}\ 2^{12}\ 2^{10}\ 2^{08}\ 2^{06}\ 2^{04}\ 2^{02}\ 2^{00}$ (VE0)

$2^{63}\ 2^{61}\ 2^{59}\ 2^{57}\ 2^{55}\ 2^{53}\ 2^{51}\ 2^{49}\ 2^{47}\ 2^{45}\ 2^{43}\ 2^{41}\ 2^{39}\ 2^{37}\ 2^{35}\ 2^{33}\ 2^{31}\ 2^{29}\ 2^{27}\ 2^{25}\ 2^{23}\ 2^{21}\ 2^{19}\ 2^{17}\ 2^{15}\ 2^{13}\ 2^{11}\ 2^{09}\ 2^{07}\ 2^{05}\ 2^{03}\ 2^{01}$ (VE1)

NOTE: Shift 1 is done on the inputs to the (VA) option.

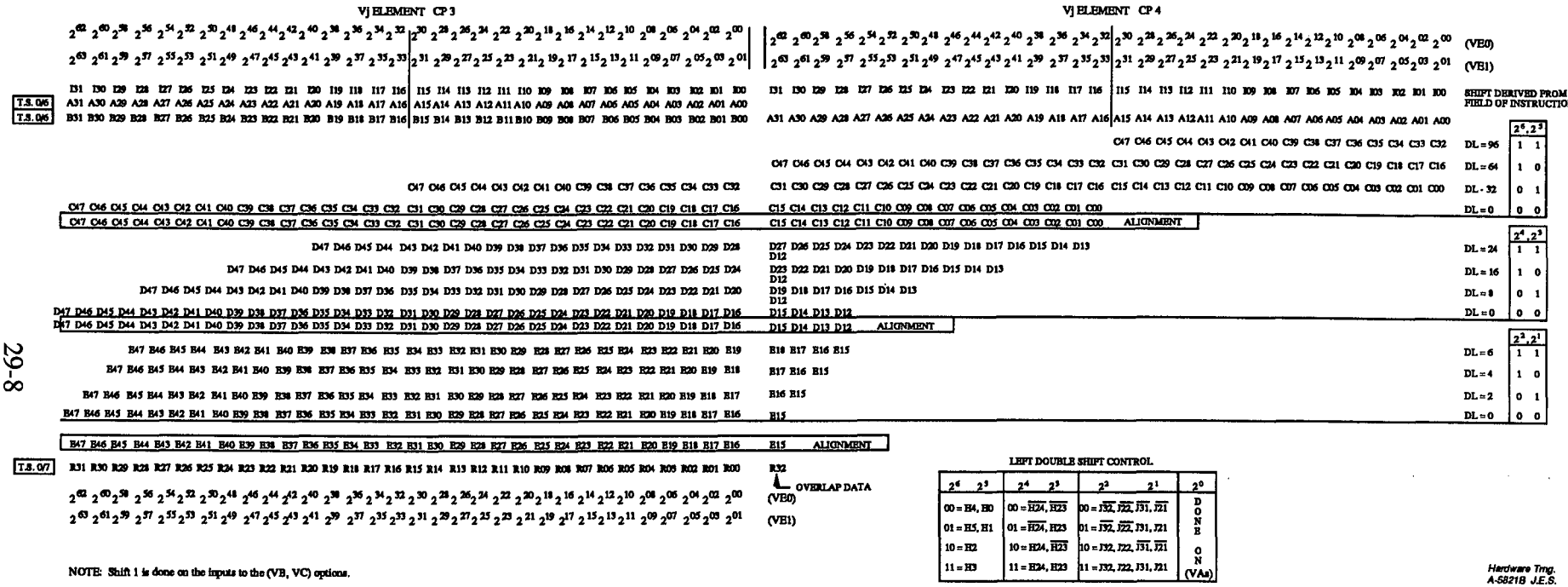
RIGHT/LEFT SINGLE SHIFT CONTROL

2^5	2^4 2^3	2^2 2^1	2^0
LEFT SINGLE SHIFT			
0 = H2	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	DONE ON (VAs)
1 = H3	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
	11 = H24, H23	11 = J32, J22, J31, J21	
RIGHT SINGLE SHIFT			
0 = H0	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	
1 = H1	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
	11 = H24, H23	11 = J32, J22, J31, J21	

Hardware Trng. A-5822B J.E.S.

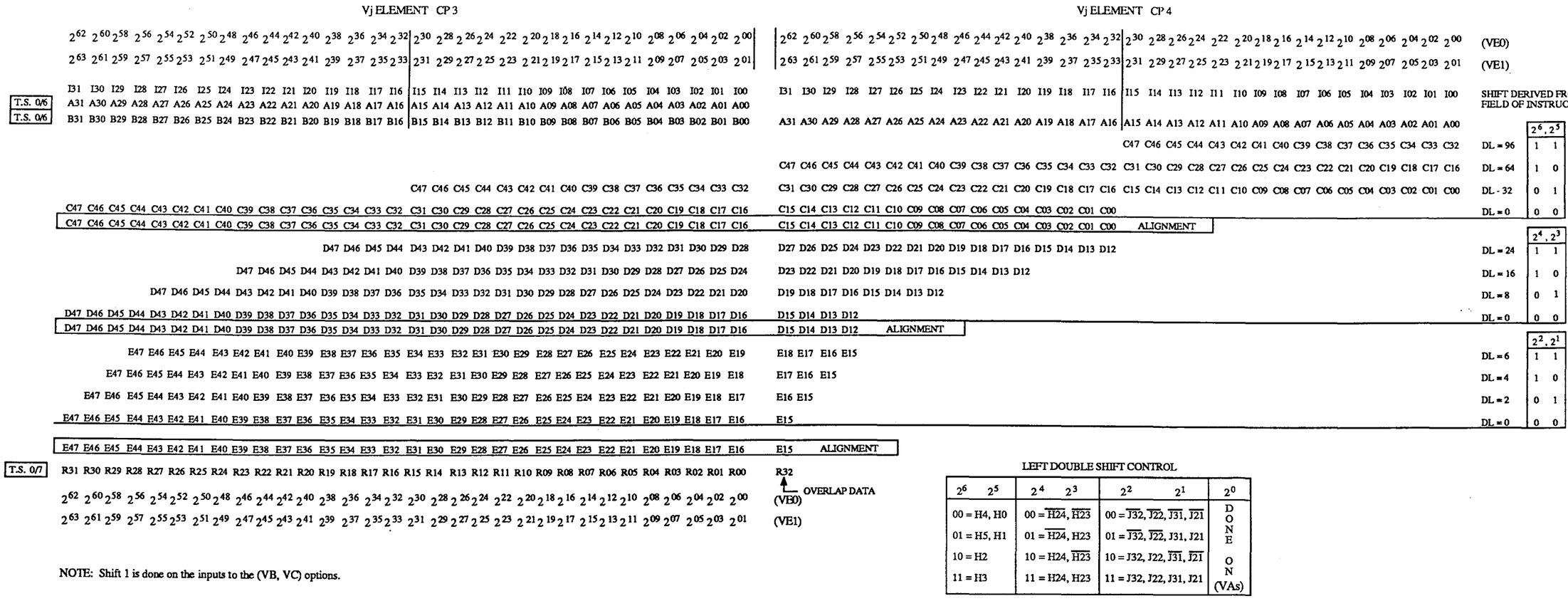
CRAY Y-MP VECTOR LEFT/RIGHT SINGLE SHIFT - 150ijk - 151ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY



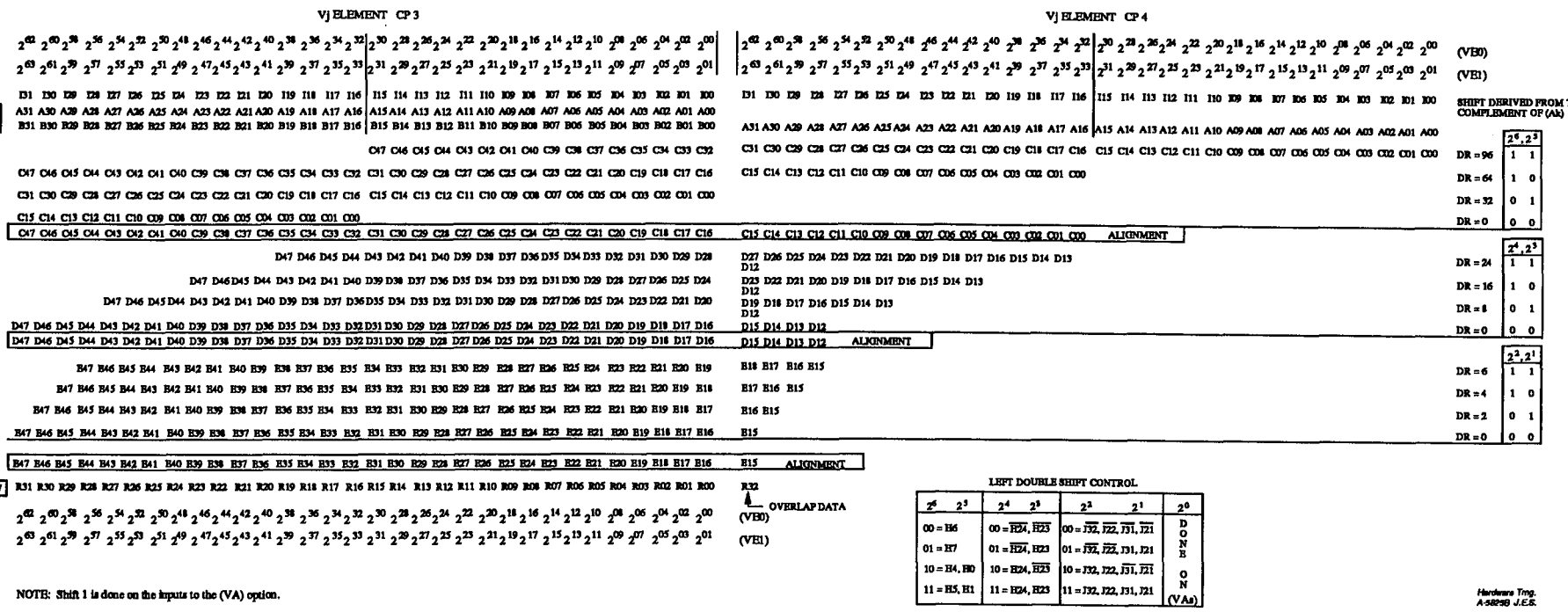
CRAY Y-MP VECTOR LEFT DOUBLE SHIFT - 152ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY



CRAY Y-MP VECTOR LEFT DOUBLE SHIFT - 152ijk INSTRUCTION (VE) OPTION

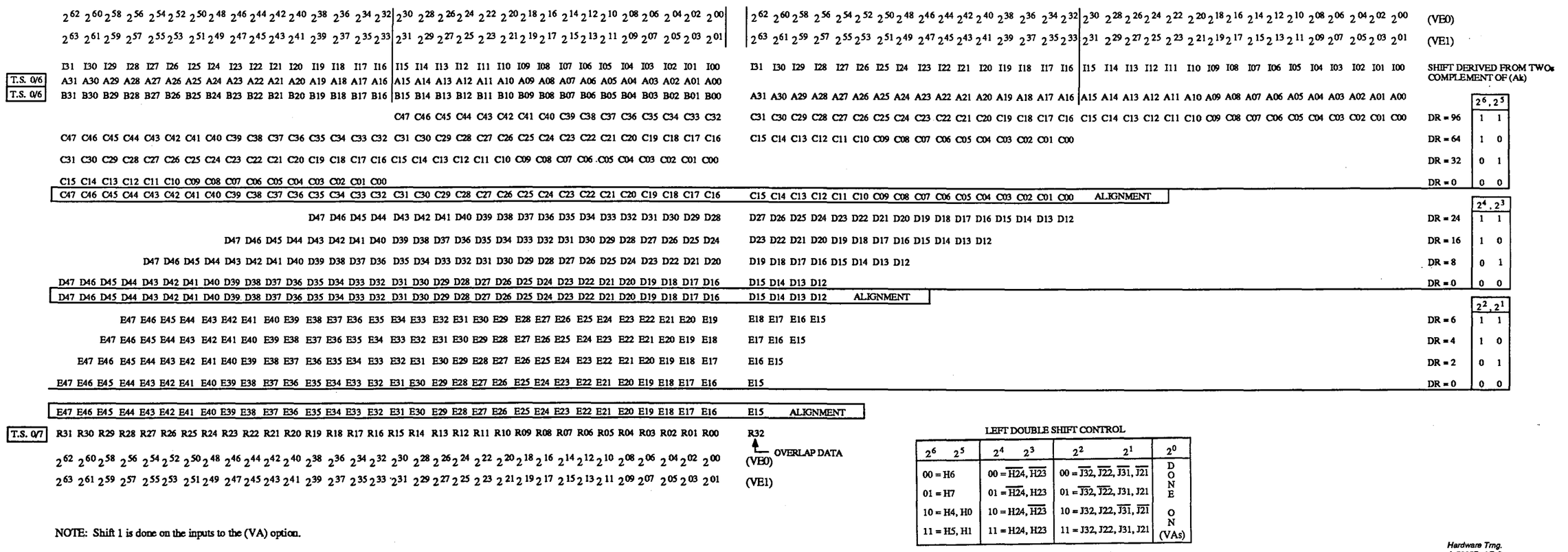
CRAY PROPRIETARY



NOTE: Shift 1 is done on the inputs to the (VA) option.

CRAY Y-MP VECTOR RIGHT DOUBLE SHIFT - 153ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY



LEFT DOUBLE SHIFT CONTROL

2^6 2^5	2^4 2^3	2^2 2^1	2^0
00 = H6	00 = $\overline{H24}$, $\overline{H23}$	00 = $\overline{J32}$, $\overline{J22}$, $\overline{J31}$, $\overline{J21}$	DONE ON (VAs)
01 = H7	01 = $\overline{H24}$, H23	01 = $\overline{J32}$, $\overline{J22}$, J31, J21	
10 = H4, H0	10 = H24, $\overline{H23}$	10 = J32, J22, $\overline{J31}$, $\overline{J21}$	
11 = H5, H1	11 = H24, H23	11 = J32, J22, J31, J21	

Hardware Trng.
A-5825B J.E.S.

CRAY Y-MP VECTOR RIGHT DOUBLE SHIFT - 153ijk INSTRUCTION (VE) OPTION

CRAY PROPRIETARY

VECTOR POP COUNT - OPTION INVOLVED

(RO1) Option

The (RO1)option performs the Vector Pop Count operation. The Vector Pop Count uses a 174ijk instruction, an instruction which is also used for a Vector Reciprocal operation. A decode of the k field tells the Reciprocal unit whether to perform a Reciprocal Approximation or to perform a population count on the elements of Vj.

Because the Vector Pop Count uses the Reciprocal units data path into and back out of the Vector registers, the Vector Pop would reserve the Reciprocal functional unit for Vector Length plus 4 CPs. A Vector Reciprocal operation would then reserve the Vector Pop operation for Vector Length plus 4 CPs as well.

A 174ij1 instruction counts the numbers of one bits in each element of Vj for Vector Length elements and enters the results into the corresponding Vi elements. For example, if all bits were set in Vj elements 0, then 100_8 would be loaded into Vi element zero.

A 174ij2 instruction counts the number of one bits in each element of Vj for Vector length elements and enters the parity of the results into the corresponding Vi element.

To determine the parity of the number of one bits set in a Vector element, the (RO1) option will only send bit 2^0 of the count to the corresponding Vi element. A one indicates odd parity and a zero indicates even parity. For example, if the number of one bits totaled 100_8, the parity, would be zero indicating even parity, as 2^0 of 100_8 equals a zero. If the count were, say 35_8, bit 2^0 would be equal to a one indicating an odd number of bits set.

Hardware Trng.
YM05/15 J.E.S.

VECTOR REGISTER POP COUNT FUNCTIONAL UNIT INSTRUCTIONS

174ij1 Population Counted of (Vj) to Vi

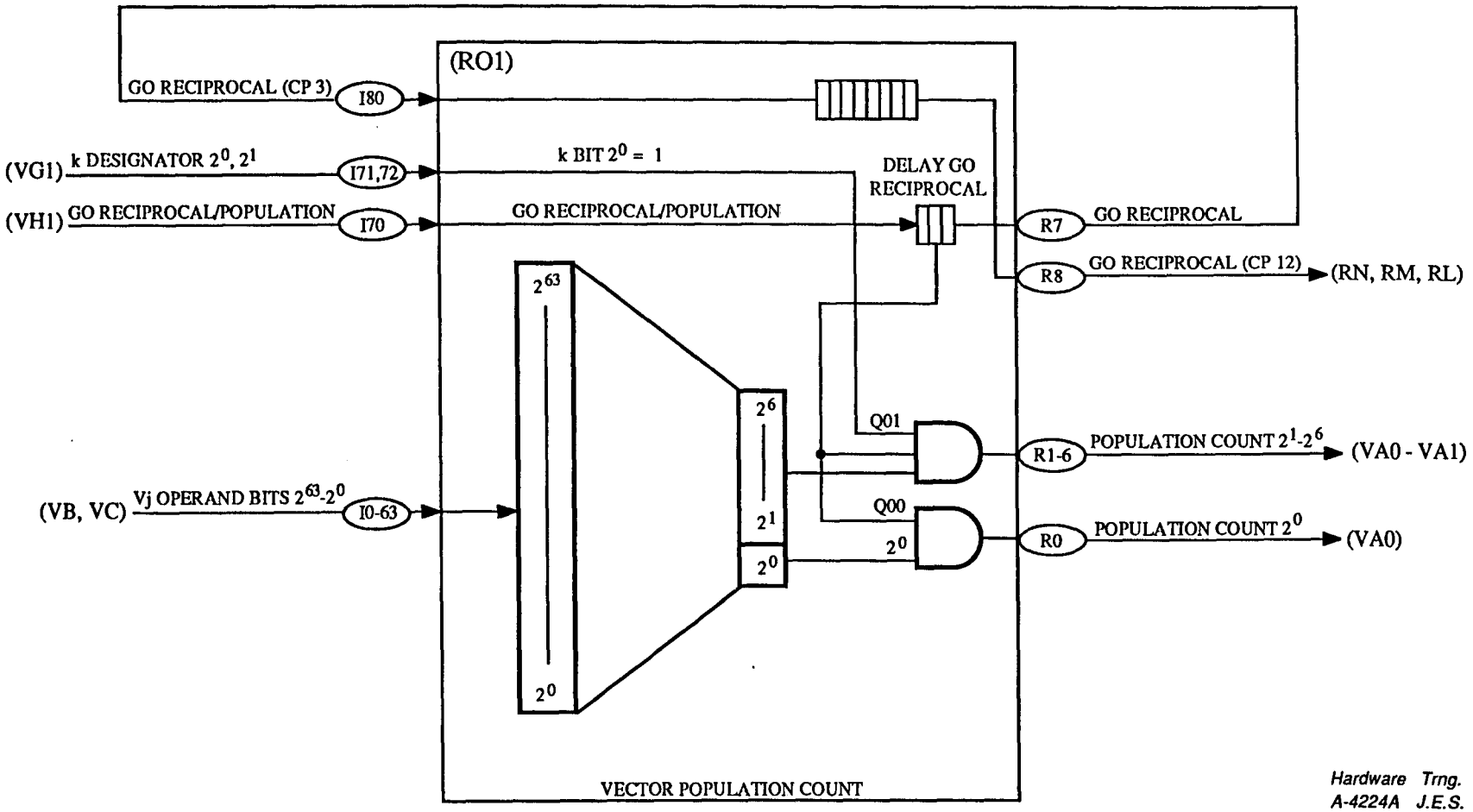
174ij2 Population Count Parities of (Vj) to Vi

CAL FORMAT

Vi PVj = 174ij1

Vi QVj = 174ij2

Hardware Trng.
YM05/16 J.E.S.



CRAY Y-MP VECTOR POPULATION COUNT
FUNCTIONAL UNIT BLOCK DIAGRAM

CRAY PROPRIETARY

VECTOR LOGICAL FUNCTIONAL UNIT

The Vector Logical functional unit is a very large functional unit because of the number of instructions that it handles. Because of this there is a second Vector Logical unit which doubles up on the 140-145 instructions; this unit is called the Vector Logical 2 in the CRAY Y-MP. To make the CRAY Y-MP compatible with the CRAY X-MP, whenever the Vector Logical 2 functional unit is busy, a floating multiply instruction will hold issue in CIP and vice versa. The Vector Logical unit executes the 140-147 instructions: the 146-147 instructions are referred to as the Vector Merge. The 175ij0-175ij7 instructions, which are called the Test of Vj element, are also performed in the Vector Logical unit. The Vector Logical unit also executes the Scalar Mask instructions, and the Vector Mask instructions which are the 073i00 and the 0030j0 instructions. The Vector Logical unit is also used as a path for the Status register and Performance Monitor data to Si, using the 073i01 and the 073i11 instructions.

Vector Logical 140-145 Instruction

The Vector Logical unit, as the unit is called, performs the Logical operations AND, OR and XOR, using either Sj and Vk for a Scalar/Vector instruction or Vj and Vk for a Vector/Vector instruction. The Vector Logical unit is located on the (VB, VC) options. The (VB, VC) options send their results to the (VA) options where the Vector Logical results are steered to the correct Vi register. The (VB0 - VB3) options send their results to the (VF0) option and then to the (VAs), this path being shared also by the Vector Mask Index using a 175ij4-175ij5 instruction.

Vector Logical 2 (140-145 Instruction)

The Vector Logical 2 unit executes the 140-145 instructions, but does not use the same Boolean equation as the Vector Logical unit. To be compatible with the CRAY X-MP, the Vector Logical 2 has the same path select code as the Floating Multiply functional unit. However, the Logical operations are performed on the (VB, VC) options, while the CRAY X-MP performs the Logical operations inside the Floating Multiply unit. The results of the Vector Logical 2 unit are then sent directly to the (VA) options, where the Vector Logical 2 has its own path to the Result registers V0-V7.

Vector Merge (146-147 Instructions)

The 146 instruction performs the same operation as the 147 instruction except the 146 instruction uses Sj and Vk as operands, while the 147 instruction uses Vj and Vk operands. Associated with the 146/147 instruction are the Vector Mask register VM. The VM register bit 2^{63} corresponds to Vector element 0, while VM bit 2^{0} corresponds to Vector element 77. When the VM bit is equal to a 1 the corresponding element of the j operand is loaded into Vi, and when the VM bit is equal to a 0 the corresponding element of the k operand is loaded into Vi, thus allowing the merging of the j operand and k operand into Vi, dependent upon the VM register. Associated with the VM register is an Index counter which is used to address the VM register bits.

Hardware Trng.
YM05/06A J.E.S.

The Index counter is cleared on the startup of the 146/147 instruction by the Go Vector Logical signal, and it is advanced by one by the Go VL Data signal. The Go VL Data is sent for each element which is read from the Vector registers. The VM register is distributed across the four (VF) options, in the same bit patterns as the Vector Logical data bits. Before a Vector element arrives on the (VB, VC) options, the associated VM bit is addressed from the index value and then distributed to all four (VF) options to be fanned out to all (VB, VC) options. The (VB, VC) options will then select either the j operand or k operand, depending on the contents of the VM register. The j/k operand is then sent to the (VA) options to be loaded into the Vi register.

Vector Test Vj (1750j0-175ij7 Instructions)

The 175 instruction performs a test on the Vj operand determined by the decode of the k field of the instruction. If the k field is equal to a 0 or 4, the Vj operand is tested for Vj = 0. If the k field is equal to a 1 or 5, the Vj operand is tested for Vj ≠ 0. If the k field of the instruction is equal to a 2 or 6, the Vj operand is tested to see if it is positive, i.e. bit 2^{63} equal to a zero; and if the k field is equal to a 3 or 7, the Vj operand is tested to see if it is negative, i.e. bit 2^{63} equal to a one. When the test is satisfied the associated VM bit gets set to a one. On the 175ij4-175ij7 instruction an additional operation takes place, in which the associated index counter is sent to the Vi register when the test is written into consecutive locations of Vi; this is referred to as the Compressed Index.

Scalar/Vector Mask/Ak/Scalar Constants (003/073, 040-043, 071 Instructions

Besides assisting in the previously mentioned instructions, the Vector Logical unit also performs the Scalar Mask, Scalar Constants, and Vector Mask instructions. The Vector Mask is loaded through the 0030j0 instruction and read by the 073i00 instruction. When loading the VM register from a Scalar register the data bits are sent in 2 CP; first the even data bits, and then the next clock period, the odd data bits. The Scalar instructions 040-041 will load into Si either the jkm field of a 2 parcel instruction or the nm field of a 3 parcel instruction.

During an 042-043 instruction the jk field is decoded for the number of ones or zeroes to be placed into Si from the left of the Scalar register. In an 042 instruction zeroes are placed in Si, with the jk field determining how many, and in an 043 instruction ones are placed in Si, with the jk field again determining how many. The Vector Logical unit also routes data from the Status register (SR) bits to Si and the Performance Monitor (PM) to Si, by using the 073i01 and the 073i11 instructions respectively. The Scalar Constants are also generated in the Vector Logical unit. This entails sending the A register data to Si, creating a floating-point number from the integer number received, and then generating the commonly used floating-point numbers .75 x 2^{48}, .5, 1.0, 2.0, and 4.0, depending on the j field of the 071ij0 instruction.

VECTOR REGISTERS VECTOR LOGICAL FUNCTIONAL UNIT INSTRUCTIONS

VECTOR LOGICAL AND VECTOR LOGICAL 2

140ijk	Vi	Sj & Vk	Logical Products of (Sj) and (Vk) to Vi
140i0k	Vi	0	Clear Vi
141ijk	Vi	Vj & Vk	Logical Products of (Vj) and (Vk) to Vi
142ijk	Vi	Sj ! Vk	Logical Sums of (Sj) and (Vk) to Vi
142i0k	Vi	Vk	Transmit (Vk) to Vi
143ijk	Vi	Vj ! Vk	Logical Sums of (Vj) and (Vk) to Vi
144ijk	Vi	Sj / Vk	Logical Difference of (Sj) and (Vk) to Vi
144i0k	Vi	Vk	Transmit (Vk) to Vi
145ijk	Vi	Vj / Vk	Logical Difference of (Vj) and (Vk) to Vi
145iii	Vi	0	Clear Vi

VECTOR MERGE

146ijk	Vi	Sj ! Vk & Vm	Transmit (Sj) if VM bit = 1; (Vk) if VM bit = 0 to Vi
146i0k	Vi	#Vm & Vk	Vector Merge of (Vk) and 0 to Vi
147ijk	Vi	Vj ! Vk & Vm	Transmit (Vj) if VM bit = 1; (Vk) if VM bit = 0 to Vi

CRAY Y-MP VECTOR REGISTERS VECTOR LOGICAL FUNCTIONAL UNIT INSTRUCTION

VECTOR TEST Vj

1750j0	VM	Vj, Z	VM = 1, Where (Vj) = 0
1750j1	VM	Vj, N	VM = 1, Where (Vj) ≠ 0
1750j2	VM	Vj, P	VM = 1, if (Vj) Positive; 0 is Positive
1750j3	VM	Vj, M	VM = 1; if (Vj) Negative; 1 is Negative
175ij4	Vi	VM, Vj, Z	VM = 1; Where Vj = 0 and Element Index is Loaded Into (compressed Vi)
175ij5	Vi	VM Vj, N	VM = 1; Where Vj ≠ 0 and Element Index is Loaded Into (compressed Vi)
175ij6	Vi	VM Vj, P	VM = 1 if (Vj) Positive and Element Index is Loaded Into (compressed Vi)
175ij7	Vi	VM Vj, M	VM = 1 if (Vj) Negative and Element Index is Loaded Into (compressed Vi)

Hardware Trng.
YM05/10 J.E.S.

LOGICAL OPERATIONS

140ijk Logical product Sj & Vk Elements to Vi
141ijk Logical product Vj Elements & Vk Elements to Vi

Sj/Vj	177777	000000	177777	177777	1 1 0 0
Vk	000000	000000	177777	177777	1 0 1 0
Vi	000000	000000	177777	177777	1 0 0 0

142ijk Logical sum Sj ! Vk Elements to Vi
143ijk Logical sum Vj Elements ! Vk Elements to Vi

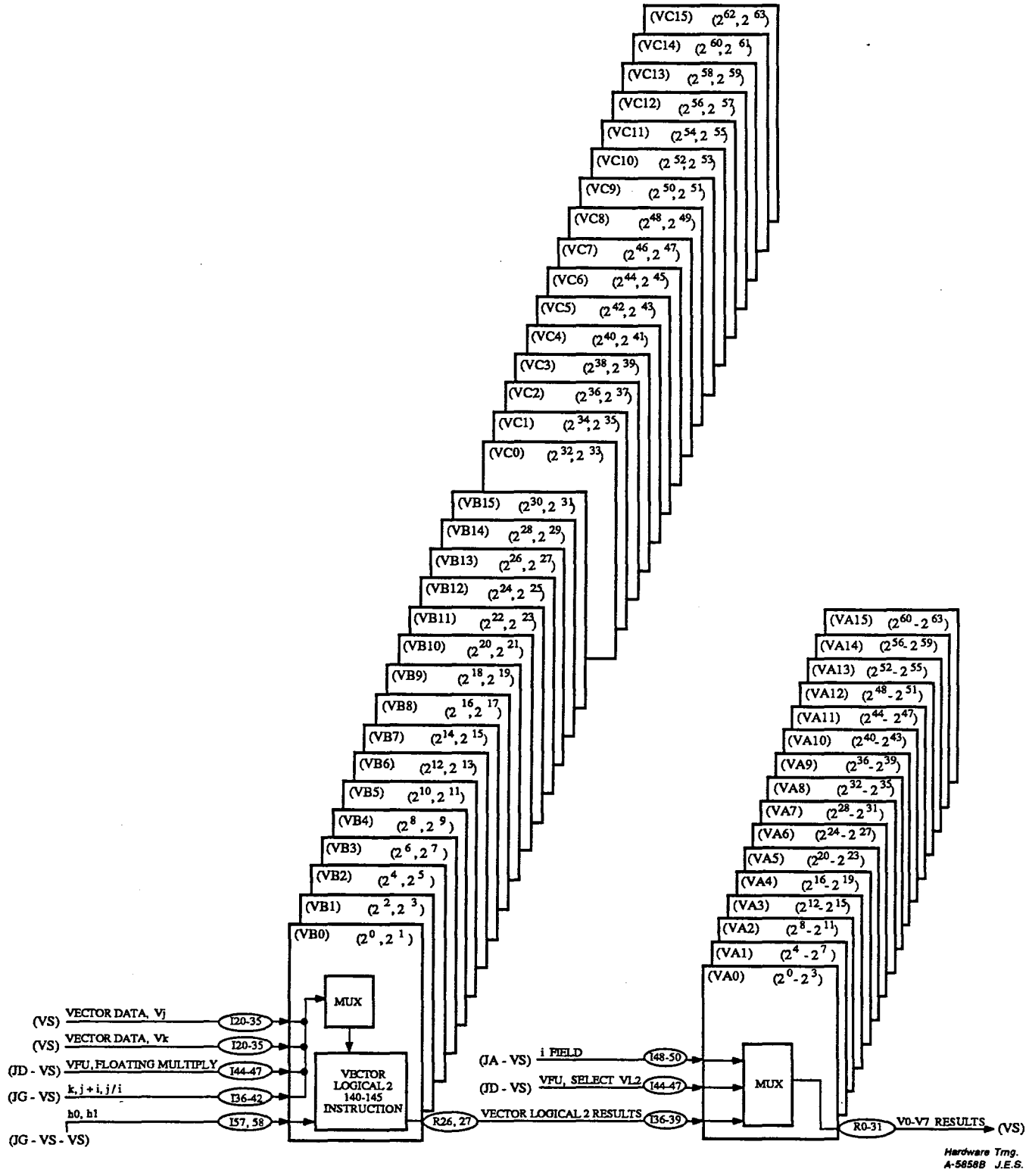
Sj/Vj	177777	000000	177777	177777	1 1 0 0
Vk	000000	000000	177777	177777	1 0 1 0
Vi	177777	000000	177777	177777	1 1 1 0

144ijk Logical difference Sj \ Vj Elements to Vi
145ijk Logical difference Vj Elements \ Vk Elements to Vi

Sj/Vj	177777	000000	177777	177777	1 1 0 0
Vk	000000	000000	177777	177777	1 0 1 0
Vi	177777	000000	000000	000000	0 1 1 0

Hardware Trng.
A-5864A J.E.S.

CRAY Y-MP VECTOR LOGICAL INSTRUCTIONS FLOW



CRAY Y-MP VECTOR LOGICAL 2 140-145 INSTRUCTIONS

VECTOR LOGICAL 2

Select Vj/Sj or Vk operand from FL Multiply

TS = 38/45 ----------

```
M259    M40-41=I20-21 L40 K40'+I22-23 L40 K40+I24-25 L41 K40'+I26-27 L41 K40
        M42-43=I28-29 L42 K40'+I30-31 L42 K40+I32-33 L43 K40'+I34-35 L43 K40
        M44-45=I20-21 L44 K43'+I22-23 L44 K43+I24-25 L45 K43'+I26-27 L45 K43
        M46-47=I28-29 L46 K43'+I30-31 L46 K43+I32-33 L47 K43'+I34-35 L47 K43
M393    M48-49=DLY (R0-1 K46')
```

Select Vk=(V0-V7), Vj = (V0-V7) or Sj (S0-S7)

TS = 0/6 ----------

```
M293    N40-41'=M40-41'M42-43':T2                    .FM k
M278    N42-43'=M44-45'M46-47'M48-49';T2 L48         .  j
```

N40-41 = Vk = (V0-V7)
N42-43 = Vj = (V0-V7) or Sj (S0-S7)
L48 = Holds Sj

Vector Logical 2 (AND, OR, XOR)

TS = 10/17 ----------

```
X212    #,P40-41'=N40-41 N42-43  K47'+N40-41'N42-43'    .V LOGICAL 2
X201    #,P42-43=N40-41 N42-43
```

K47 = h bit 2^1 or h1
P40 = (OR) Function + (XOR) Function for VL2
P42 = (AND) Function for VL2

Hardware Trng.
YM05/13 J.E.S.

Output Results of VL2 to (VAs)

TS = 20/27 ----------

```
R26-27=MUX (P40-41, P42-43):DCD (K48'K47')     .V LOGICAL 2 RESULT
```

K48, K47 = h1, h2 instruction
R26 = VL2 bit ($N + 2^0$)
R27 = VL2 bit ($N + 2^1$)

VECTOR LOGICAL UNIT - OPTIONS INVOLVED

(VB-VC) Options

The (VB, VC) options assist the (VF) option in performing the various Vector Logical instructions. The (VB, VC) will perform all of the Vector Logical 2 operating (the 140-145 instructions), as well as the full Vector Logical operations. The (VB, VC) also sends Scalar register data to the VM registers.

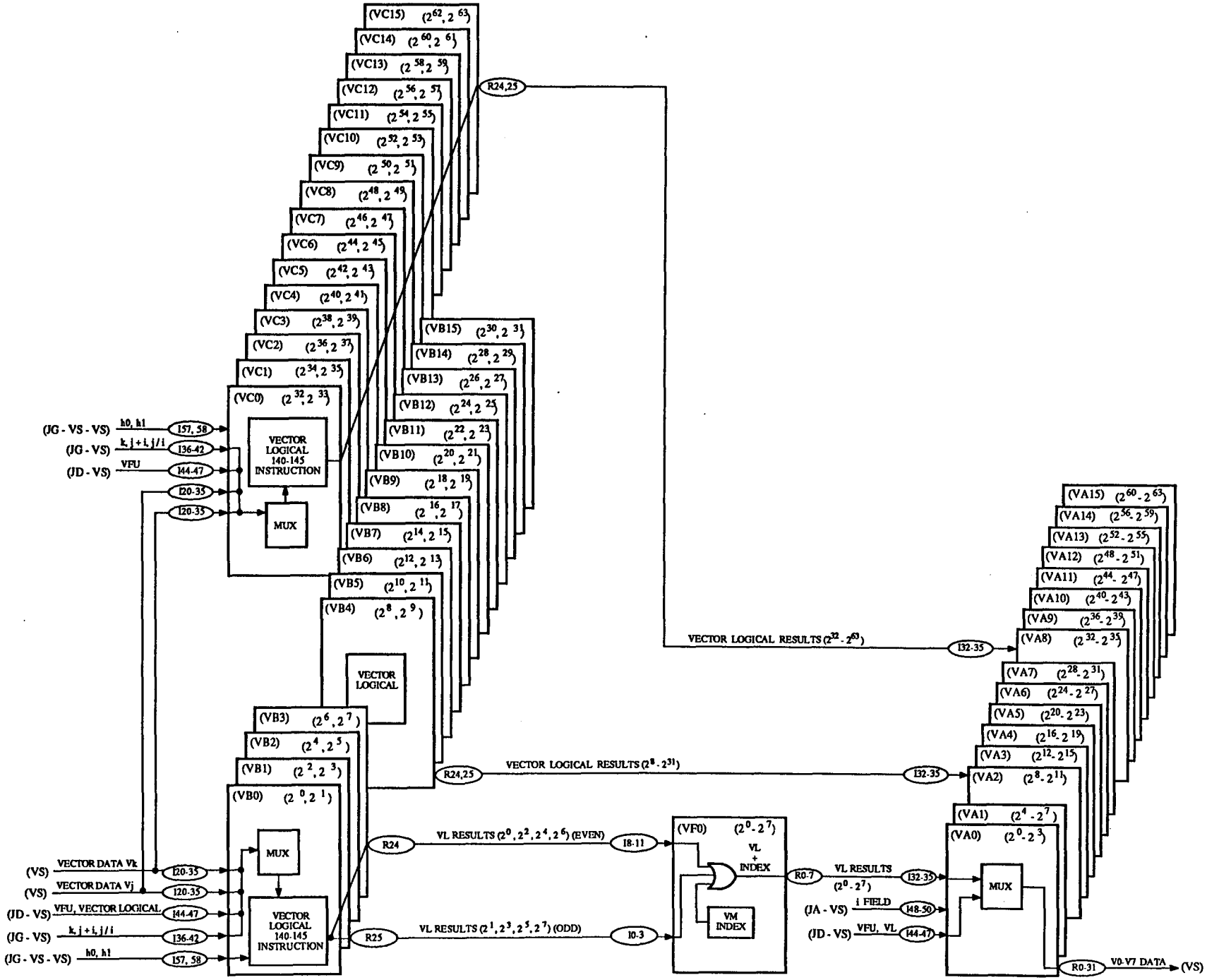
During a 146/147 instruction the (VB, VC) options are responsible for selecting either j operand or k operand data, depending on the contents of the VM register. When executing a 175 instruction, the (VB, VC) options perform a partial test on the data bit and send the results to the (VF) options.

(VF0-VF3) Options

The (VF) options contain the VM register, and are responsible for performing the Scalar and Vector Mask instructions, as well as the Scalar Constant instructions. The (VF) options finalize the test of the Vj elements in a 175 instruction, and set the appropriate VM register bit. The (VF) supplies the (VB, VC) options with the VM registers bit for the 146/147 instruction. The (VF) also supplies the (VG0) option with the control term Ak = 0 which is used on a double shift.

(VA) Options

The (VA) options select the data from the Vector Logical unit by using the VFU select code, and send the data to 1 of 8 Vector registers by decoding the i field of the instruction. The (VA) options are used by the 140-145 instructions, the 146/147 instructions and the 175ij4-175ij7 instructions.



CRAY Y-MP VECTOR LOGIC[illegible]0-145 INSTRUCTION

Hardware Tr...
A-5860B ...

VECTOR LOGICAL 140-145 INSTRUCTION

Select Vj/Sj or Vk operands

TS = 38/45 ----------

```
M259    M60-61=I20-21 L60 K60'+I22-23 L60 K60+I24-25 L61 K60'+I26-27 L61 K60
        M62-63=I28-29 L62 K60'+I30-31 L62 K60+I32-33 L63 K60'+I34-35 L63 K60
        M70-71=I20-21 L70 K70'+I22-23 L70 K70+I24-25 L71 K70'+I26-27 L71 K70
        M72-73=I28-29 L72 K70'+I30-31 L72 K70+I32-33 L73 K70'+I34-35 L73 K70
        M74-75=I20-21 L74 K73'+I22-23 L74 K73+I24-25 L75 K73'+I26-27 L75 K73
        M76-77=I28-29 L76 K73'+I30-31 L76 K73+I32-33 L77 K73'+I34-35 L77 K73
M393    M78-79+DLY (R0-1 K76')
```

M70-71 = Vector Logical operand Vk = V0-V3
M72-73 = Vector Logical operand Vk = V4-V7
M74-75 = Vector Logical operand Vj = V0-V3
M76-77 = Vector Logical operand Vj = V4-V7
M78-79 = Vector Logical operand Sj = S0-S7

Select Vk=(V0-V7), Vj = (V0-V7) or Sj (S0-S7)

TS = 0/6 ----------

```
M293    N70-71'=M70-71'M72-73';T2                    .V LOG k
M278    N72-73'=M74-75'M76-77'M78-79';T2 L78         .      j
```

L78 - Holds (N72-73) when Sj operand is used

Hardware Trng.
YM05/11 J.E.S.

Vector Logical (AND, OR, XOR)

TS = 10/17 ----------

```
M314    P70-71'=N70-71'K78'K77'+N70-71 N72-73 K78+&      .V LOGICAL
            N72-73'K78'K77'+N70-71'N72-73'
```

$\overline{P70}=\overline{N70}+\overline{N72}$ → AND Function (140-141) instruction

$\overline{P70}=\overline{N70}\ \overline{N72}$ → OR Function (142-143) instruction

$\overline{P70}=N70\ N72 + \overline{N70}\ \overline{N72}$ → XOR Function (144-145) instruction

K77=h1, K78 = h2 - instruction decode

Output Vector Logical Results

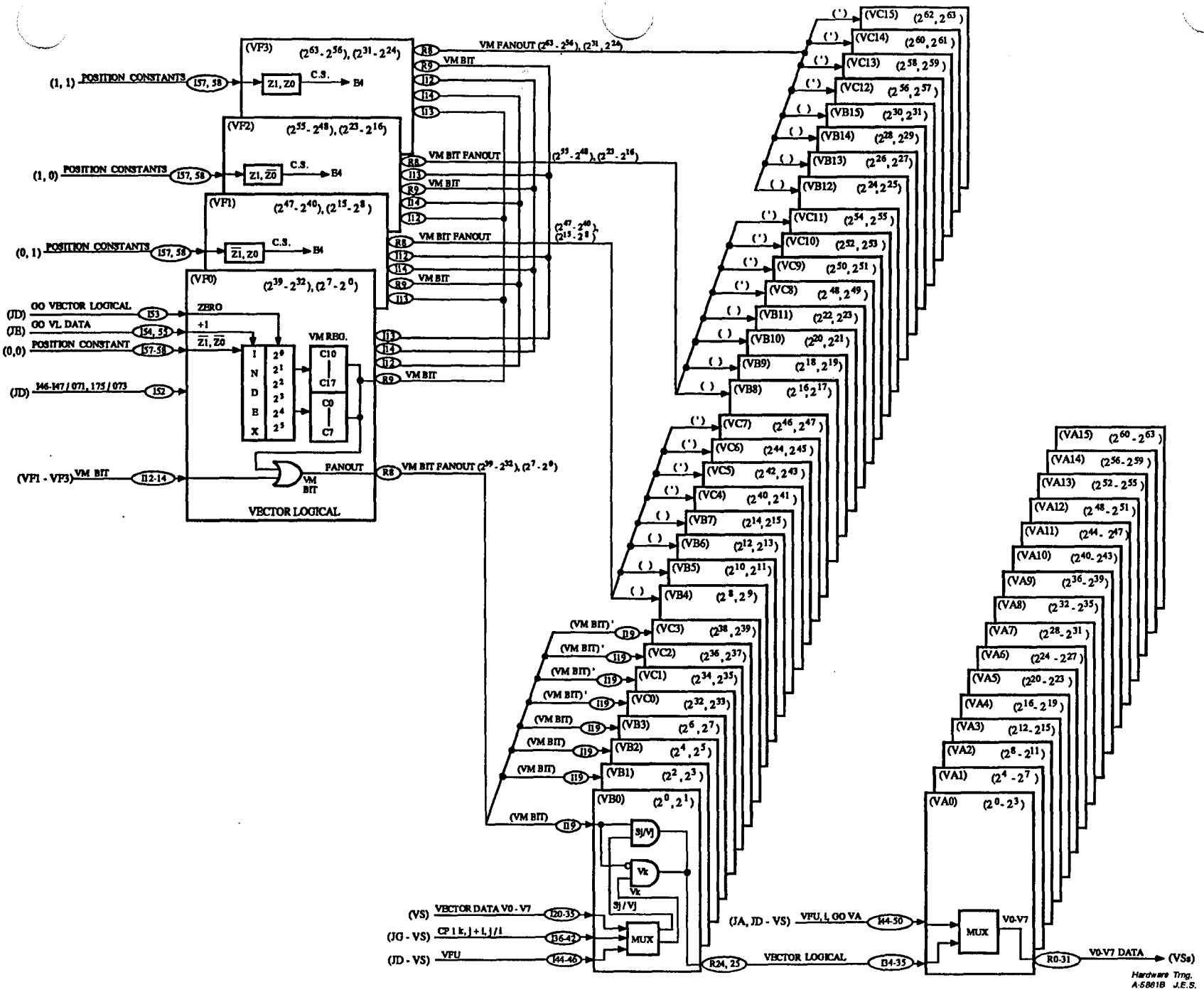
TS =20/27 ----------

```
X254    R24'=MUX(P70', P72' ):DCD(P75)      .V LOGICAL RESULT
        R25'=MUX(P71', P73'P74 '):DCD(P75)  .V LOGICAL RESULT+Vj TEST + Sj
```

R24 = VL + Merge bit ($n + 2^0$)
R25 = VL + Vj test + Sj to Vm bit ($n + 2^1$)

Hardware Trng.
YM05/12 J.E.S.

CRAY PROPRIETARY

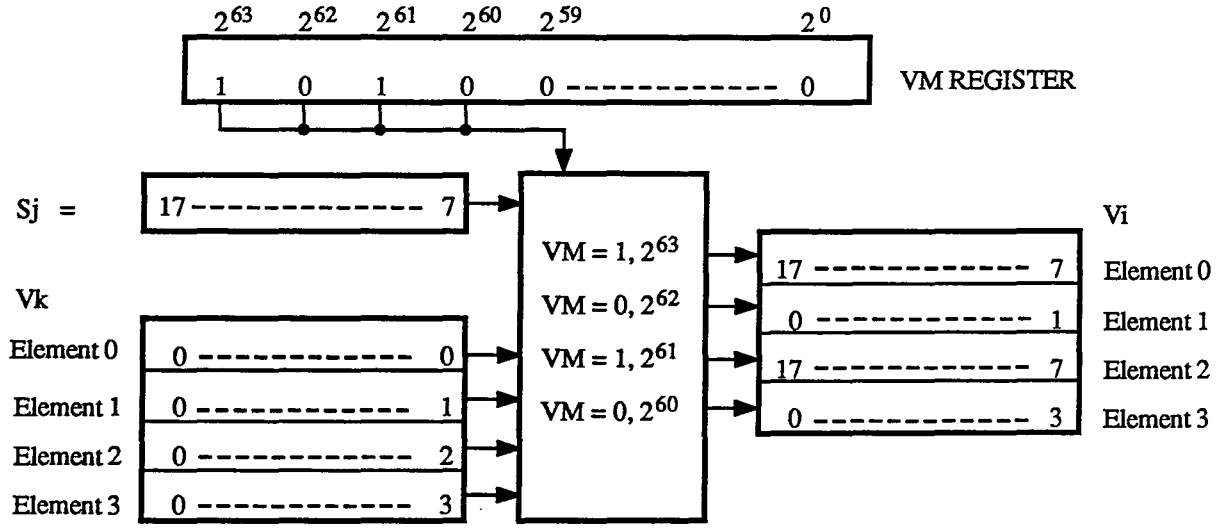


CRAY Y-MP VECTOR MERGE 146 / 147 INSTRUCTION BLOCK DIAGRAM

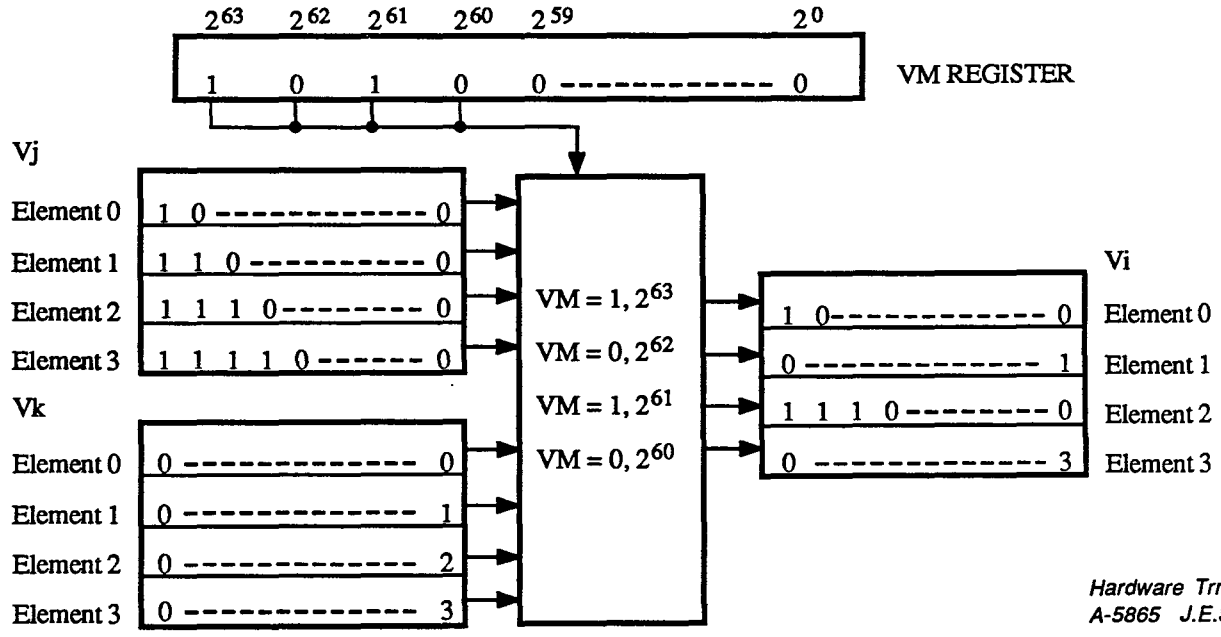
146ijk Transmit Sj to Vi Elements if VM Bit = 1
Transmit Vk Elements to Vi Elements if VM Bit = 0

147ijk Transmit Vj Elements to Vi Elements if VM Bit = 1
Transmit Vk Elements to Vi Elements if VM Bit = 0

146ijk INSTRUCTION



147ijk INSTRUCTION



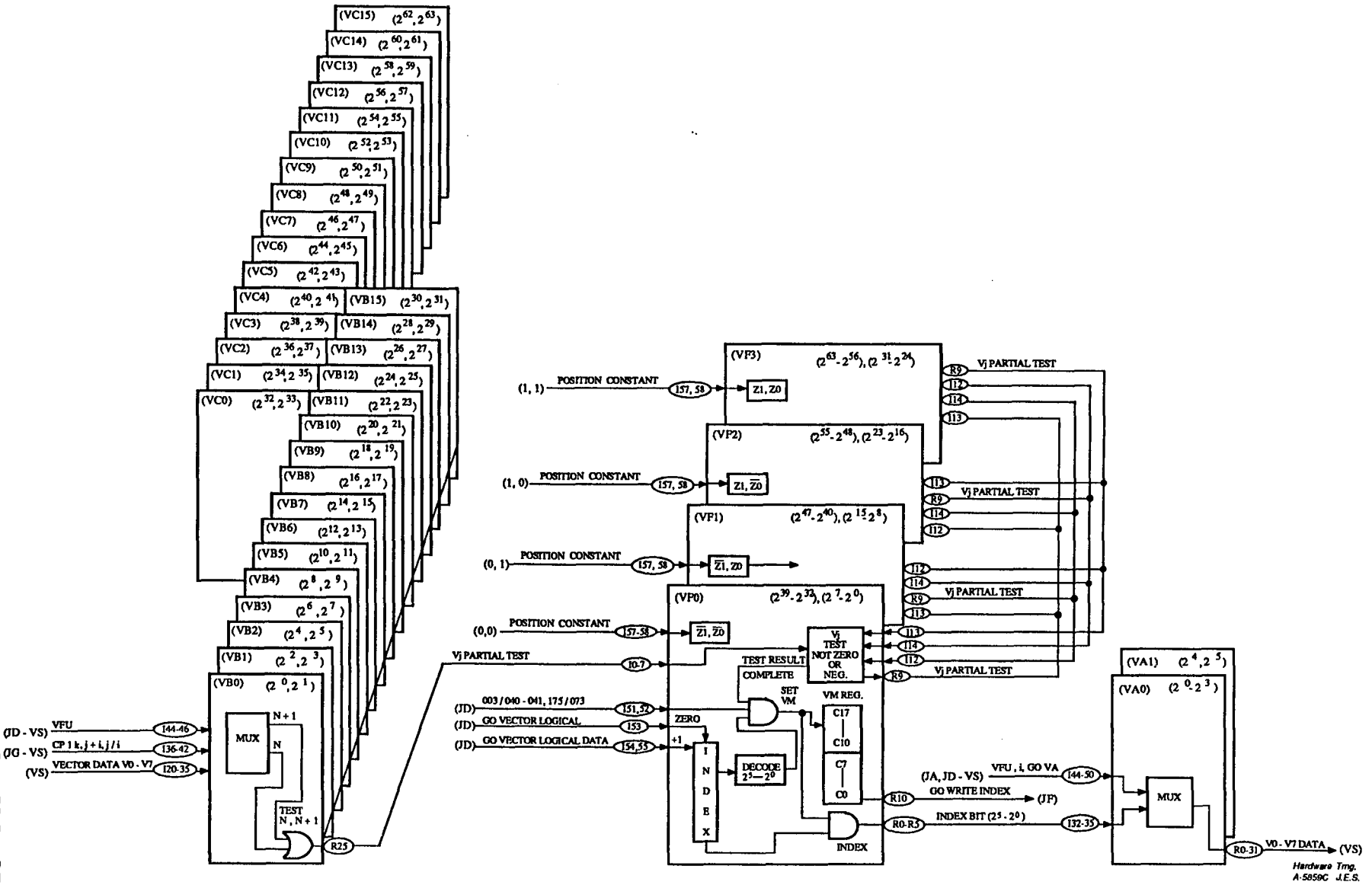
Hardware Trng.
A-5865 J.E.S.

CRAY Y-MP VECTOR LOGICAL INSTRUCTIONS FLOW

VM BIT	INDEX D5 D4 D3 D2 D1 D0 2^5 2^4 2^3 2^2 2^1 2^0	Boolean	Option	VM Bit	INDEX D5 D4 D3 D2 D1 D0 2^5 2^4 2^3 2^2 2^1 2^0	Boolean	Option
2^{63}	0 0 0 0 0 0	C17	(VF3)	2^{31}	1 0 0 0 0 0	C07	(VF3)
2^{62}	0 0 0 0 0 1	C16	(VF3)	2^{30}	1 0 0 0 0 1	C06	(VF3)
2^{61}	0 0 0 0 1 0	C15	(VF3)	2^{29}	1 0 0 0 1 0	C05	(VF3)
2^{60}	0 0 0 0 1 1	C14	(VF3)	2^{28}	1 0 0 0 1 1	C04	(VF3)
2^{59}	0 0 0 1 0 0	C13	(VF3)	2^{27}	1 0 0 1 0 0	C03	(VF3)
2^{58}	0 0 0 1 0 1	C12	(VF3)	2^{26}	1 0 0 1 0 1	C02	(VF3)
2^{57}	0 0 0 1 1 0	C11	(VF3)	2^{25}	1 0 0 1 1 0	C01	(VF3)
2^{56}	0 0 0 1 1 1	C10	(VF3)	2^{24}	1 0 0 1 1 1	C00	(VF3)
2^{55}	0 0 1 0 0 0	C17	(VF2)	2^{23}	1 0 1 0 0 0	C07	(VF2)
2^{54}	0 0 1 0 0 1	C16	(VF2)	2^{22}	1 0 1 0 0 1	C06	(VF2)
2^{53}	0 0 1 0 1 0	C15	(VF2)	2^{21}	1 0 1 0 1 0	C05	(VF2)
2^{52}	0 0 1 0 1 1	C14	(VF2)	2^{20}	1 0 1 0 1 1	C04	(VF2)
2^{51}	0 0 1 1 0 0	C13	(VF2)	2^{19}	1 0 1 1 0 0	C03	(VF2)
2^{50}	0 0 1 1 0 1	C12	(VF2)	2^{18}	1 0 1 1 0 1	C02	(VF2)
2^{49}	0 0 1 1 1 0	C11	(VF2)	2^{17}	1 0 1 1 1 0	C01	(VF2)
2^{48}	0 0 1 1 1 1	C10	(VF2)	2^{16}	1 0 1 1 1 1	C00	(VF2)
2^{47}	0 1 0 0 0 0	C17	(VF1)	2^{15}	1 1 0 0 0 0	C07	(VF1)
2^{46}	0 1 0 0 0 1	C16	(VF1)	2^{14}	1 1 0 0 0 1	C06	(VF1)
2^{45}	0 1 0 0 1 0	C15	(VF1)	2^{13}	1 1 0 0 1 0	C05	(VF1)
2^{44}	0 1 0 0 1 1	C14	(VF1)	2^{12}	1 1 0 0 1 1	C04	(VF1)
2^{43}	0 1 0 1 0 0	C13	(VF1)	2^{11}	1 1 0 1 0 0	C03	(VF1)
2^{42}	0 1 0 1 0 1	C12	(VF1)	2^{10}	1 1 0 1 0 1	C02	(VF1)
2^{41}	0 1 0 1 1 0	C11	(VF1)	2^{09}	1 1 0 1 1 0	C01	(VF1)
2^{40}	0 1 0 1 1 1	C10	(VF1)	2^{08}	1 1 0 1 1 1	C00	(VF1)
2^{39}	0 1 1 0 0 0	C17	(VF0)	2^{07}	1 1 1 0 0 0	C07	(VF0)
2^{38}	0 1 1 0 0 1	C16	(VF0)	2^{06}	1 1 1 0 0 1	C06	(VF0)
2^{37}	0 1 1 0 1 0	C15	(VF0)	2^{05}	1 1 1 0 1 0	C05	(VF0)
2^{36}	0 1 1 0 1 1	C14	(VF0)	2^{04}	1 1 1 0 1 1	C04	(VF0)
2^{35}	0 1 1 1 0 0	C13	(VF0)	2^{03}	1 1 1 1 0 0	C03	(VF0)
2^{34}	0 1 1 1 0 1	C12	(VF0)	2^{02}	1 1 1 1 0 1	C02	(VF0)
2^{33}	0 1 1 1 1 0	C11	(VF0)	2^{01}	1 1 1 1 1 0	C01	(VF0)
2^{32}	0 1 1 1 1 1	C10	(VF0)	2^{00}	1 1 1 1 1 1	C00	(VF0)

Hardware Trng.
A-5863 J.E.S.

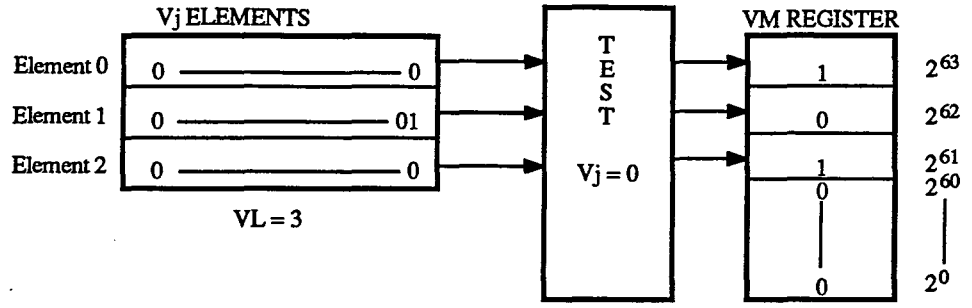
CRAY Y-MP VECTOR MASK REGISTER



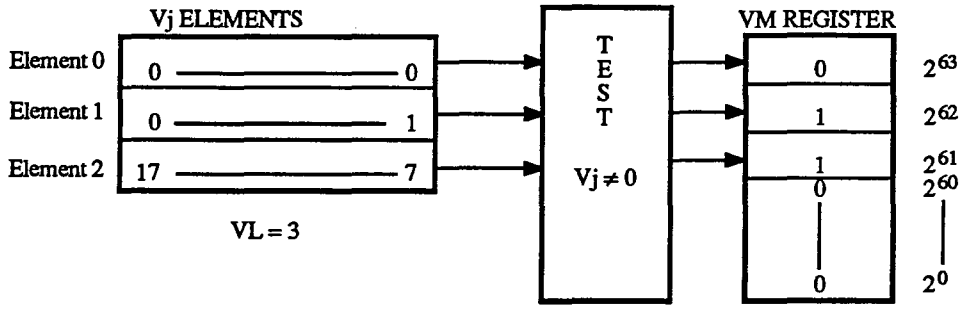
CRAY Y-MP VECTOR TEST Vj 175ij0 - 175ij7 INSTRUCTION BLOCK DIAGRAM

1750jk INSTRUCTION

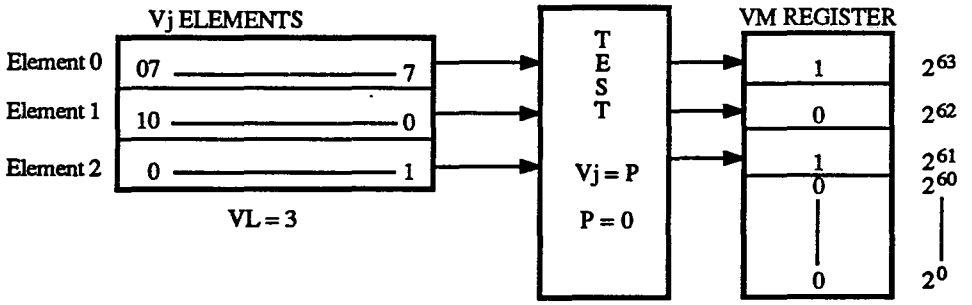
1750j0 - VM = 1 WHERE (Vj) = 0



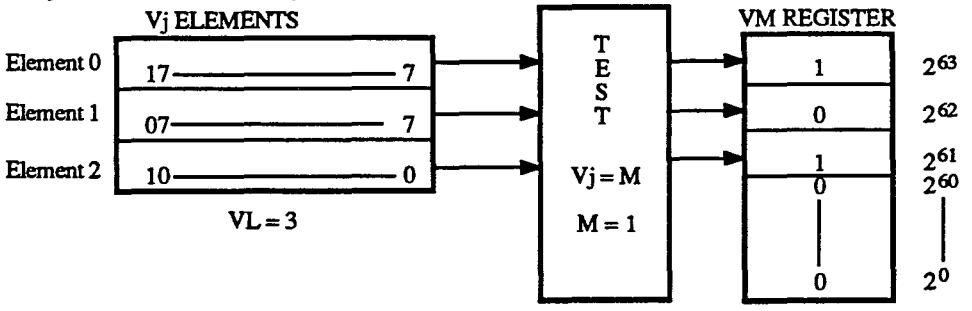
1750j1 - VM = 1 WHERE (Vj) ≠ 0



1750j2 - VM = 1 WHERE (Vj) POSITIVE



1750j3 - VM = 1 WHERE (Vj) NEGATIVE

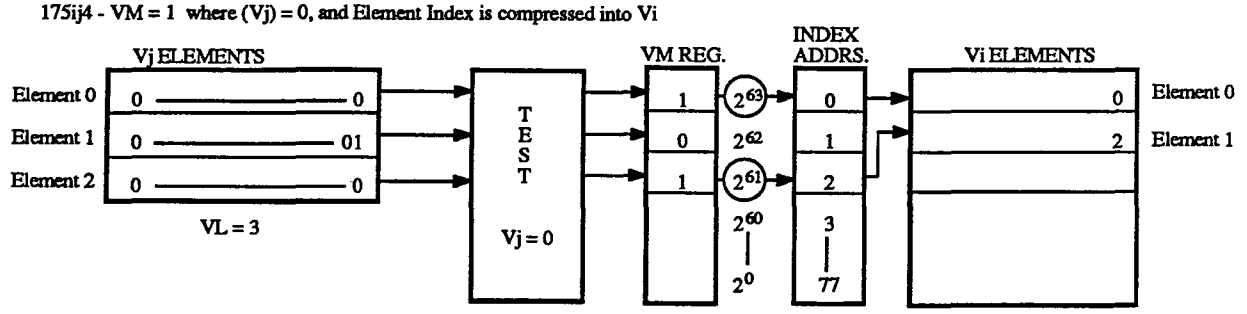


Hardware Trng.
A-5866 J.E.S.

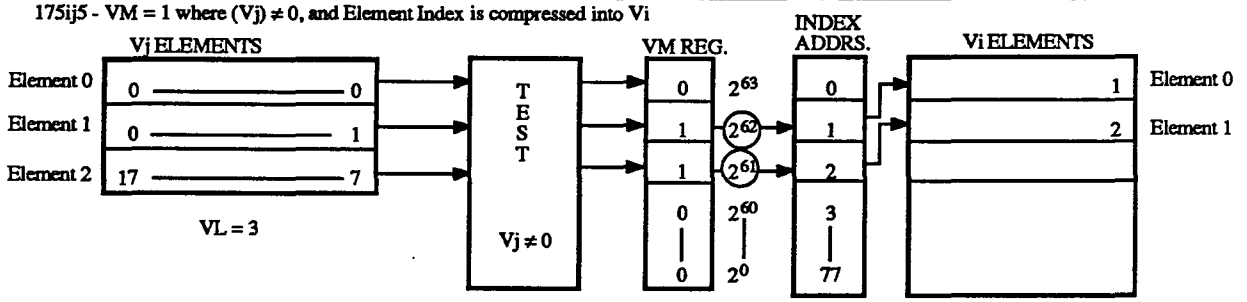
CRAY Y-MP VECTOR LOGICAL INSTRUCTIONS

175ijk INSTRUCTION
Compressed/Index

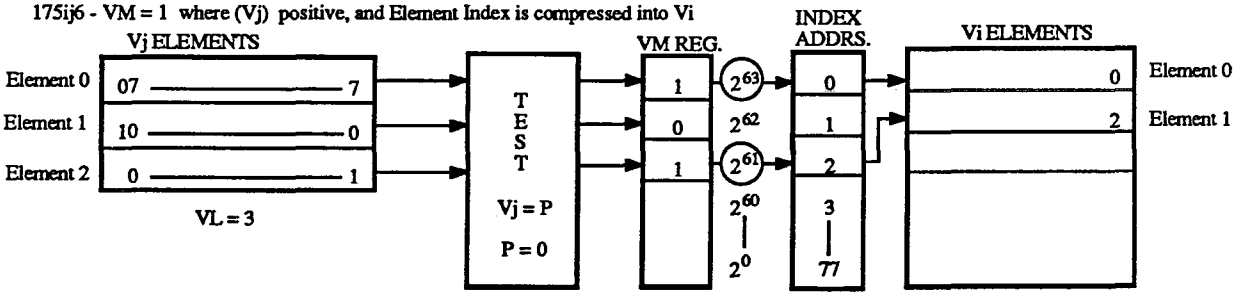
175ij4 - VM = 1 where (Vj) = 0, and Element Index is compressed into Vi



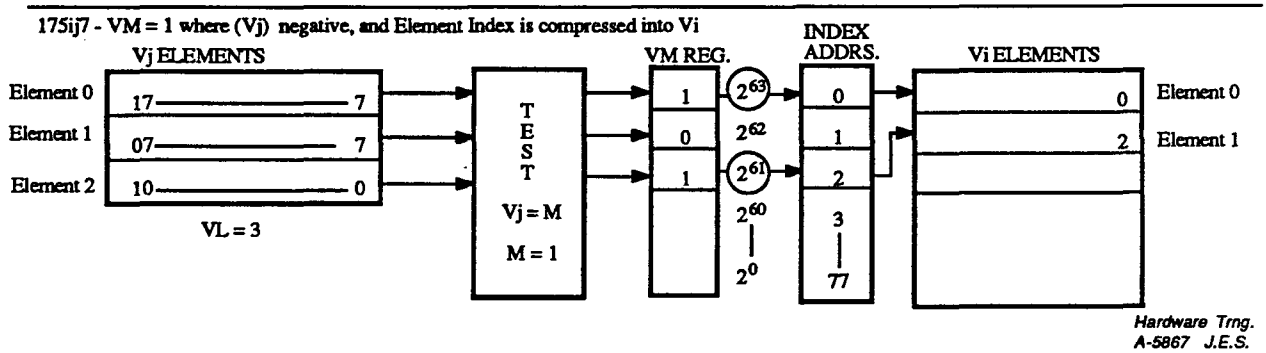
175ij5 - VM = 1 where (Vj) ≠ 0, and Element Index is compressed into Vi



175ij6 - VM = 1 where (Vj) positive, and Element Index is compressed into Vi



175ij7 - VM = 1 where (Vj) negative, and Element Index is compressed into Vi



Hardware Trng.
A-5867 J.E.S.

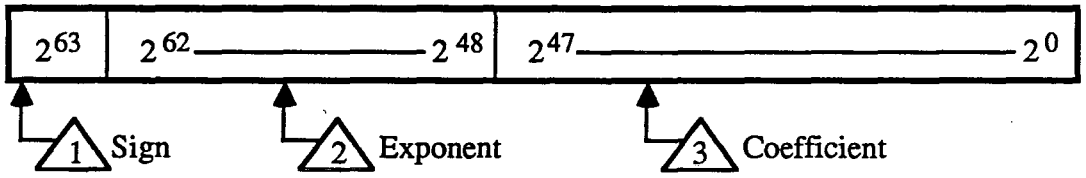
CRAY Y-MP VECTOR LOGICAL INSTRUCTIONS FLOW

CRAY Y-MP FLOATING-POINT FORMAT

Floating-point Arithmetic Introduction

Floating-point numbers are represented in a standard format throughout the CPU. Floating-point numbers are used by the Floating-point functional units which are Floating Add, Floating Multiply, and Floating Reciprocal. This format is a compact representation of a binary exponent and coefficient within a 64-bit word. The coefficient is in the hardware bit positions (2^{47} - 2^{0}) of the 64-bit word. The sign of the coefficient is in bit position (2^{63}). The exponent is in hardware bit positions (2^{62} - 2^{48}).

FLOATING-POINT FORMAT



1. Sign - Sign of the coefficient is in bit position (2^{63}, 2^{63}) = 0 and is a positive sign of the coefficient. 2^{63} = 1 is a negative sign of the coefficient.

2. Exponent - The exponent of the floating-point format is represented as a biased integer number in bit positions (2^{62} - 2^{48}). Bit position (2^{61}) represents the sign of this exponent. Bit position (2^{62}) represents the bias.

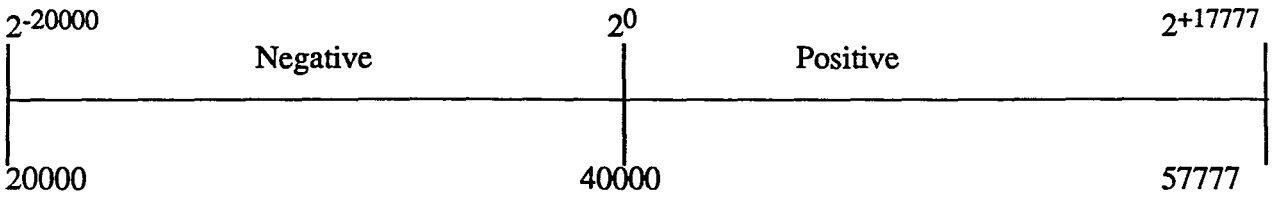
3. Coefficient - The coefficient is a signed magnitude number. For a negative coefficient the sign bit (2^{63}) is set to a "1". The coefficient is a positive fraction indicating how far from zero the number is, but in the negative direction indicated by the sign bit.

Exponent Ranges

The exponent of the floating-point format is represented as a biased integer in bit positions (2^{62} - 2^{48}). The bias added to the exponent is 40000_8, which represents an exponent of 2^0. The positive range of the exponent is 40000_8 - 57777_8 and the negative range is 37777_8 - 20000_8. Thus the biased range of the exponent is from 2^{-20000} through 2^{+17777}.

Hardware Trng.
YM06/01 J.E.S.

EXPONENT NUMBER LINE



The floating-point format of the CRAY Y-MP allows the expression of numbers to be accurate to about 15 decimal digits in the approximate decimal range of 10^{-8191} - 10^{+8191}. This gives the 48-bit coefficient the ability to become an extremely large or small number.

Exponent Range Checks, Underflow, Overflow

Even the CRAY Y-MP has its limitations; hardware range check is made on the exponent to ensure that numbers being represented in a Floating-point functional unit are within the limits of the CRAY Y-MP. Double precision numbers can be used to increase the limitation.

Overflow

An Overflow condition occurs when the exponent has exceeded 57777. To check this condition the hardware checks bit positions (2^{62}) and (2^{61}) of the exponent. If (2^{62}) and (2^{61}) both equal a "1", an Overflow condition exits or a number equal to or greater than 60000 has occurred in the functional unit.

When an Overflow condition occurs, a flag is set in the Exchange Package indicating a Floating-point Range Error. This occurs for exponents equal to or greater than 60000 during a floating-point add or floating-point multiply operation. During a floating-point reciprocal operation the exponent is complemented and has a value of two added to it, however the range check occurs before this takes place.

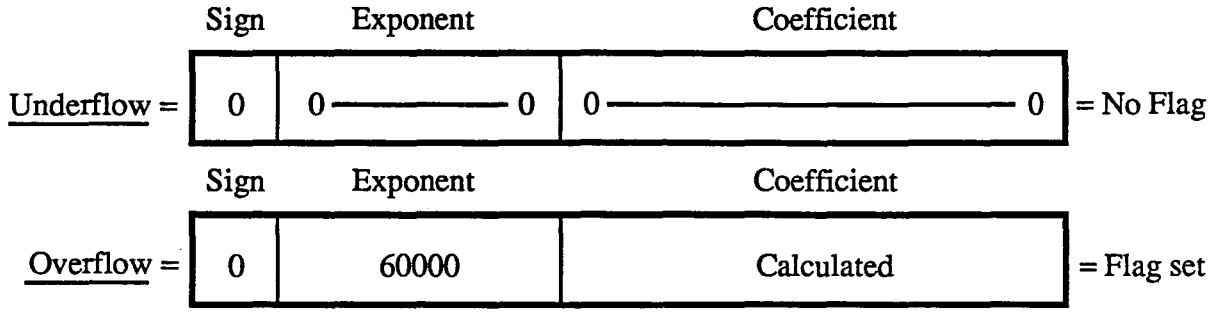
Underflow

An Underflow condition occurs when an exponent is less than or equal to 17777_8, or the exponent bits (2^{62}) and (2^{61}) are both equal to zero. For an Underflow condition no Error flag is set, however, the exponent and coefficient are both forced to zeroes. This occurs for a floating-point add and floating-point multiply operation. Again, the floating-point reciprocal operation complements the exponent and adds a value of two to the exponent. An exponent of less than or equal to 17777_8 during a reciprocal causes an Underflow condition which in turn causes an Overflow condition on the original exponent, forcing the exponent to 60000_8.

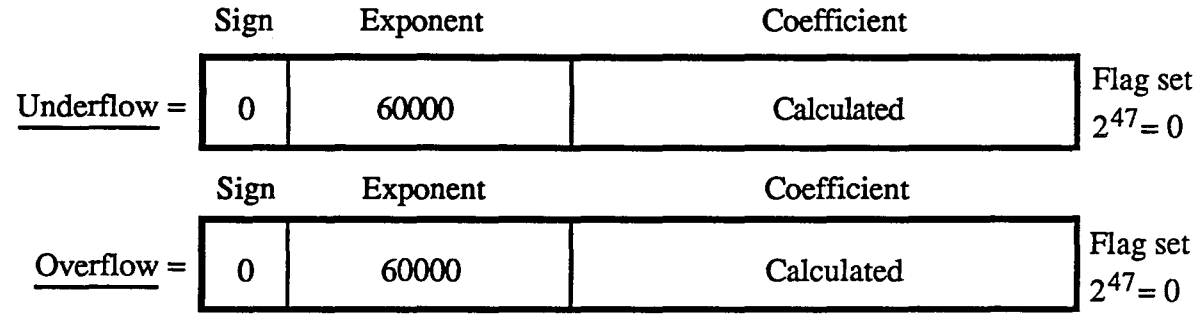
The floating-point multiply follows the underflow rules with the exception of when both of the exponents are equal to zero. If this occurs the floating-point multiply allows the multiply to complete as an integer multiply, leaving the exponent and sign bit equal to zero.

Hardware Trng.
YM06/02 J.E.S.

Floating-Point Add or Floating-Point Multiply



Floating Reciprocal



Normalized Floating-Point

A floating-point number is normalized if the most significant bit, (2^{47}), of the coefficient is a non-zero number. This condition of having a normalized coefficient implies that the coefficient has been shifted to the left as far as possible and therefore, the floating-point number has no leading zeroes in the coefficient. The Floating-point functional unit treats a normalized floating-point number differently.

The Reciprocal functional unit must have a normalized floating-point number in order to receive the correct results. The Floating-point Multiply functional unit does not require normalized floating-point numbers to get correct results. However, because only the upper half of the multiply pyramid is kept and the lower half is truncated, more accurate results occur when the two operands are normalized.

If the two operands are normalized, normalized results are received from the Multiply functional unit.

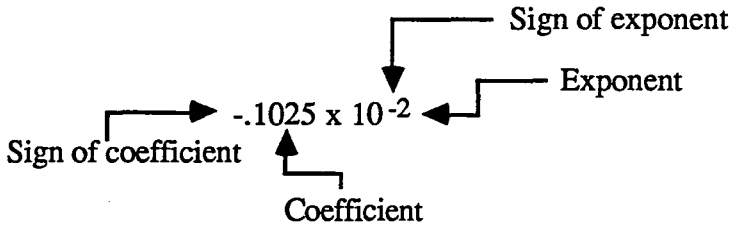
The Floating-point Add functional unit does not require normalized operands. However, the last thing the Floating-point Add functional unit does is normalize its results. Floating-point numbers can be explicitly routed through the Floating-point Add functional unit to take advantage of this normalizing process.

Hardware Trng.
YM06/03 J.E.S.

Conversion From Integer to Floating-Point Format

When converting integers to floating-point it is necessary to understand the construction of the floating-point format. The components of the floating-point format can be expressed in the form of scientific notation.

Example: The components of the floating-point format can be compared to the standard scientific notation in power of tens as follows.



Note the effect on the exponent with the shifting of the coefficients decimal point to the left. The following numbers all have the same values:

$1025. \times 10^{-2}$
102.5×10^{-1}
10.25×10^{0}
1.025×10^{1}
$.1025 \times 10^{2}$

When converting an integer number to a floating-point number on the CRAY Y-MP, shift the binary point to the left. This converts the integer number into a fraction which is represented as the coefficient. The normal mechanical method of conversion is to left shift the binary point 48 places, which equals 60 octal. This method assumes that the integer is equal to or less than 48 bits and is a whole number.

To normalize this floating-point number, the number of leading zeroes is counted until the first 1-bit, and then the number of leading zeroes is subtracted from the exponent.

Example #1 - Issue a 071424 instruction to change (A4) into a floating-point format.

#1 A4 = | 00000001777 | A4 = A integer, whole number

S4 = | 040060 000000 000000 001777 |

S4 = Floating-point numbers in CRAY Y-MP format

Hardware Trng.
YM06/04 J.E.S.

#2 Issue a 062404 to normalize S4

S4 = 040012 1777000 000000 000000 The number of leading zeroes is subtracted from the exponent.

Lets take a closer look at Example #1 A4 = 00000001777 which is an integer or whole number. A4 can also be represented as:

A4 = $00000001777.0 \times 2^{0}$ Which is the same value that is in A4.

Because the coefficient has 48_{10} places and the binary point is moved to the left 48_{10} places or 60_8, as in S4. The exponent must have 60_8 added to the exponent to represent the number of places the binary point was moved to keep the same number value. The exponent now equals 40060 with the bias bit (2^{62}) set, and 1777 in the lowest position of coefficient.

S4 = 040060 . 000000 000000 001777 .

To normalize S4 the Floating-point Add functional unit is used to take advantage of the normalizing capability that the Floating-point Add functional unit has. The 062404 instruction is used to normalize S4.

S4 = 040012 . 1777000 000000 000000

When the Floating-point Add functional unit normalizes S4, it counts the number of leading zeroes and subtracts this value from the exponent. It then adds in the bias. The coefficient is then left shifted 46_8 places, as represented in the normalized S4.

The sign of the integer number originally in A4 was positive or zero which is bit (2^{23}) of an A register A4 in X-mode or (2^{31}) in Y-mode. To represent the sign of the coefficient in S4, which is bit (2^{31}) or (2^{23}) of A4, it must also be set to a zero.

Example #2 - Issue a 071424 to represent the negative integer numbers in A4 to floating-point format.

#1 A4 = 37777777776 Integer number in A4 equals a negative 2, in twos complement form.

S4 = 140060 . 000000 000000 000002 S4 is the result of the completion of the 071424 instruction.

S4 has the sign bit set which is bit (2^{63}) of S4 indicating that the coefficient has a negative fraction that is two positions from zero, but on the negative side of the number line. The coefficient is taken out of the twos complement form and is now shown as a Sign Magnitude number.

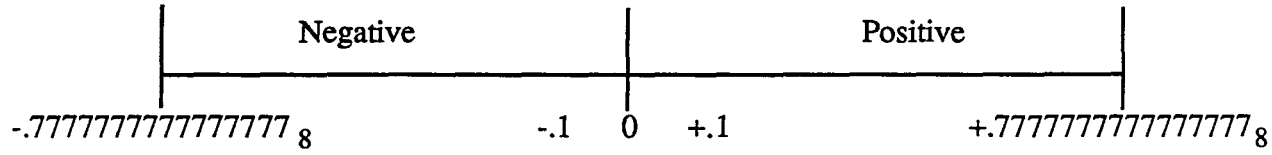
Hardware Trng.
YM06/05A J.E.S.

#2 Issue a 062404 to normalize S4.

S4 = | 140002 . 100000 000000 000000 |

Again, the sign bit 2^{63} is set indicating a negative number in the coefficient. The number of leading zeroes is subtracted from the exponent to make the exponent equal to 40002. The coefficient is then left shifted 56_8 positions in the Floating Add functional unit.

COEFFICIENT NUMBER LINE



Hardware Trng.
YM06/06 J.E.S.

CRAY Y-MP CONSTANT INSTRUCTIONS

INSTRUCTIONS

071i0k	Transmit (Ak) to Si with no sign extension
071i1k	Transmit (Ak) to Si with sign extension
071i2k	Transmit (Ak) to Si as unnormalized floating-point number. Exponent equal 40060.
071i30	Transmit constant 0.75 x 2^{48} to Si Si = 040060 140000 000000 000000
071i40	Transmit constant 0.5 to Si Si = 040000 100000 000000 000000
071i50	Transmit constant 1.0 to Si Si = 040001 100000 000000 000000
071i60	Transmit constant 2.0 to Si Si = 040002 100000 000000 000000
071i70	Transmit constant 4.0 to Si Si = 040003 100000 000000 000000

CAL FORMAT

071i0k	Si	Ak
071i1k	Si	+Ak
071i2k	Si	+FAk
071i30	Si	0.6
071i40	Si	0.4
071i50	Si	1.
071i60	Si	2.
071i70	Si	4.

Hardware Trng.
YM06/07 J.E.S.

SCALAR MASK/VECTOR MASK VECTOR LOGICAL FUNCTIONAL UNIT INSTRUCTIONS

INSTRUCTION SCALAR MASK

040ijkm	Si	Exp.	Transmit Exp. = jkm to Si
040i00mn	Si	Exp.	Transmit Exp. = nm to Si
041ijkm	Si	Exp.	Transmit Exp. = Ones complement of jkm to Si
041i00mn	Si	Exp.	Transmit Exp. = Ones complement of nm to Si
042ijk	Si<	Exp.	Form zeroes mask exp. bits in Si from the left ; jk field get exp.
042i77	Si	1	Enter 1 into Si
042i00	Si	-1	Enter -1 into Si; (Si = 177777 177777 177777 177777)
043ijk	S >	Exp.	Form ones mask exp. bits in Si from the left: jk field get exp.
043i00	Si	0	Clear Si

INSTRUCTION VECTOR MASK

0030j0	VM	Sj	Transmit (Sj) to VM register
073i00	Si	VM	Transmit (VM) to Si

INSTRUCTION SR/PM

073i01	Si	SRj	Transmit (SRj) to Si
073i11			Read performance counter to Si

Hardware Trng.
YM06/08 J.E.S.

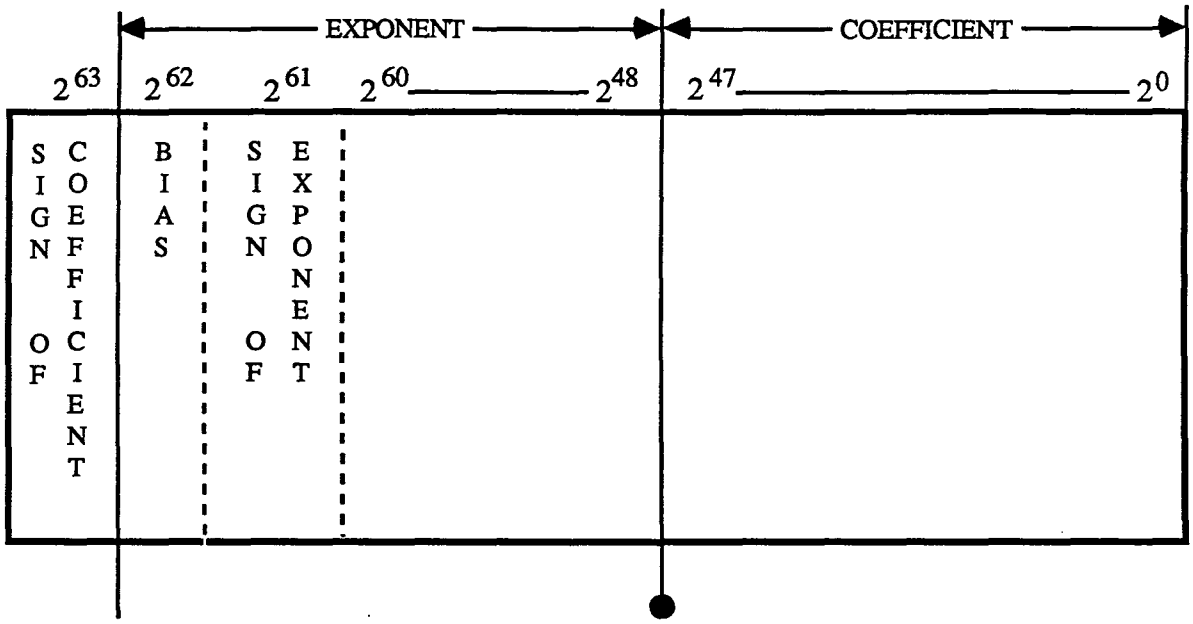
FLOATING-POINT RANGE ERRORS

	EXPONENT 2^{63} -------- 2^{48}	COEFFICIENT 2^{47} ---------- 2^{0}
FLOATING-POINT ADD:		
Underflow	000000	000000 000000 000000
Overflow	060000	Calculated plus error
FLOATING-POINT MULTIPLY:		
Underflow	000000	000000 000000 000000
Overflow	060000	Calculated plus error
FLOATING-POINT RECIPROCAL:		
Underflow	060000	Calculated plus error*
Overflow	060000	Calculated plus error*

*Bit 2^{47} will be clear

*Both set error flag

Hardware Trng.
YM06/09 J.E.S.

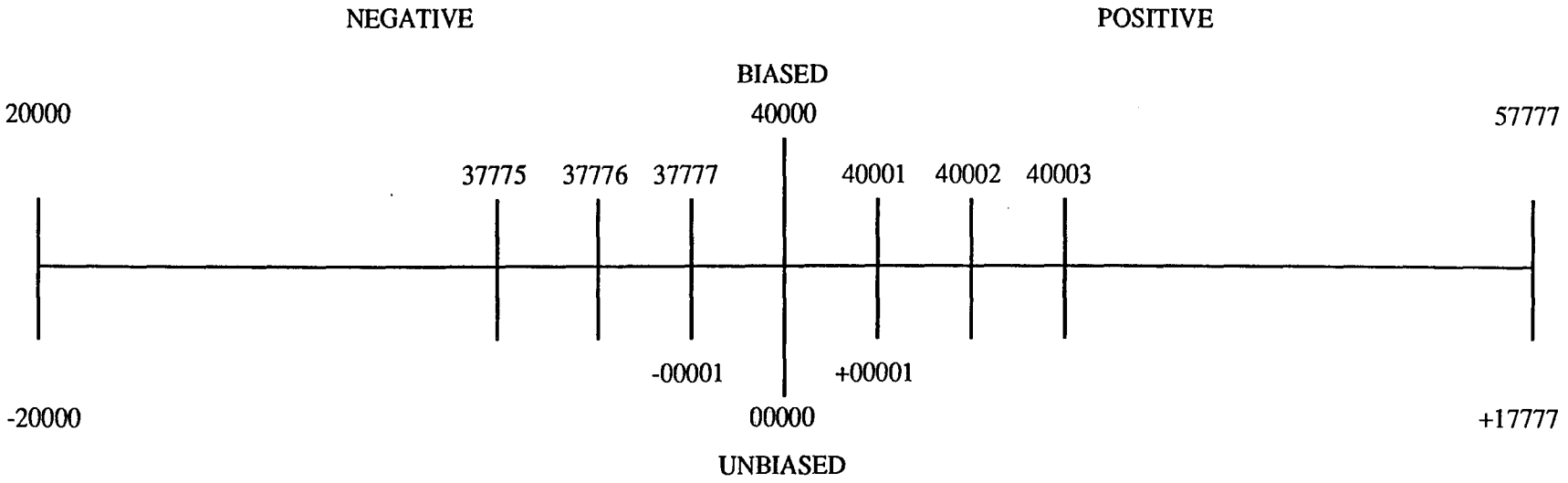


$2^{61} = 0$
$2^{62} = 0$
} UNDERFLOW

$2^{61} = 1$
$2^{62} = 1$
} OVERFLOW

Hardware Trng.
A-5848A J.E.S.

CRAY Y-MP FLOATING-POINT REPRESENTATION



In terms of decimal values, the approximate decimal range is from 10^{-2466} through 10^{+2466}

Hardware Trng.
A-5847 J.E.S.

CRAY Y-MP EXPONENT REAL NUMBER LINE

CRAY PROPRIETARY

INTEGER VALUE IN S1 TO FLOATING-POINT FORMAT IN S3

Where S1 = 000000 000000 173743 026721

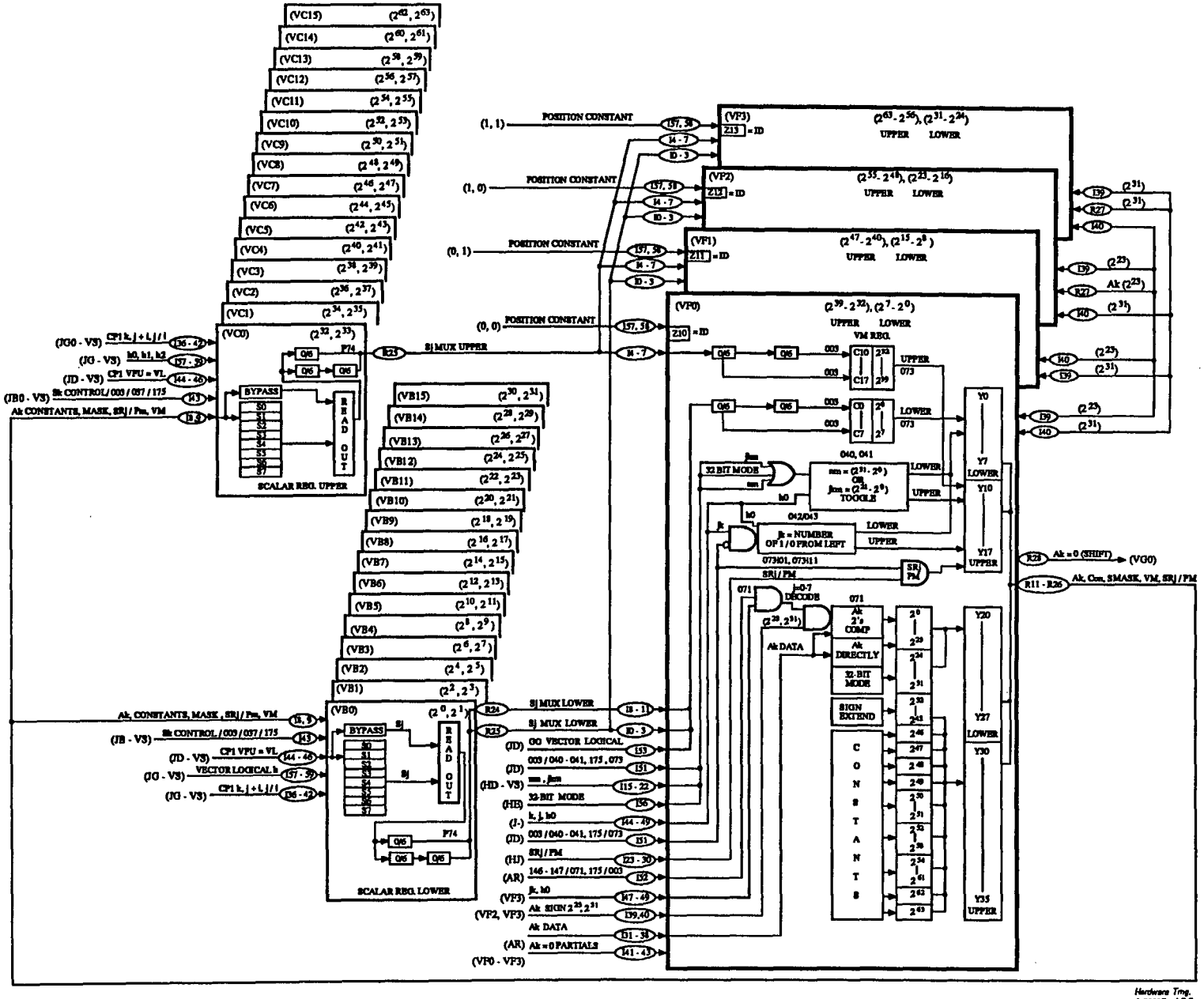
ALGORITHM = #1 071230 S2 = $.75 \times 2^{48}$

#2 061121 S1 = S2 - S1

#3 063321 S3 = S2 - FS1

#1	S2 =	040060	140000	000000	000000	
#2	S2 =	040060	140000	000000	000000	
	-S1 =	000000	000000	173743	026721	
	S2 =	040060	140000	000000	000000	
	S1 =	177777	177777	004034	151056	1's Compliment
		040060	137777	004034	151056	
					+1	2's Compliment
	S1 =	040060	137777	004034	151057	
#3	S2 =	040060	140000	000000	000000	
	-S1 =	040060	137777	004034	151057	
	S2 =	040060	140000	000000	000000	
	S1 =	040060	040000	173743	026720	1's Compliment
		040060	.000000	173743	026720	
					+1	Carry Across Binary Point
	S3	040060	000000	173743	026721	Unnormalized
	S3	040040	173743	026721	000000	Normalized

Hardware Trng.
YM06/10 J.E.S.



CRAY Y-MP SCALAR/VECTOR MASK/Ak/SCALAR CONSTANTS
003/073, 040 - 041, 042 - 043, 071 INSTRUCTIONS

FLOATING-POINT ADD FUNCTIONAL UNIT

The Floating-Point Add functional unit performs three operations. They are to equalize exponents, add coefficients, and normalize the results.

To equalize the exponents the larger of the two exponents is retained. The coefficient of the smaller exponent is right shifted by the difference of the two exponents, or until both exponents are equal. Because the coefficient is only 48 bits, any shift beyond 48 bits will cause the smaller operand coefficient to become zeroes.

Addition of the two coefficients takes place after the exponents are equalized and the coefficient is adjusted accordingly. The sign bits of the two operands and the type of instruction issued, add/subtract, will determine if an Add Operation or Subtract Operation takes place. If the sign bits are equal, and a subtract instruction or the sign bits are unequal and an add instruction is issued, then a subtraction operation will occur. This operation requires performing the ones complement arithmetic on the larger J/K operand and allowing an end around carry or "carry in lower" which causes the results of the add to toggle. If the sign bits are equal, and an add instruction or the sign bits are unequal and a subtract instruction is issued, then an add operation will occur.

Add/Subtract Operation		
J = K sign bit	•	Add instruction = Add operation
J ≠ K sign bit	•	Add instruction = Subtract operation
J = K sign bit	•	Subtract instruction = Subtract operation
J ≠ K sign bit	•	Subtract instruction = Add operation

The last operation to be performed by the Floating Add functional unit is to normalize the results. To normalize the results the coefficient is left shifted by the number of leading zeroes. The coefficient is normalized when a one bit is in bit position (2^{47}) of the coefficient. The exponent must also be decremented accordingly. If a carry across the binary point occurs during an add operation the coefficient is right shifted by one and the exponent increases by one. During a subtract operation if a carry across the binary point occurs, an end around carry will take place.

A range check is performed on the exponent to determine if its greater than or equal to 60000_8 or less than or equal to 17777_8. If the exponent is greater than or equal to 60000_8 an overflow condition exists, in which case the exponent is forced to 60000_8 and a flag is sent to the exchange package indicating a floating-point error. The calculated coefficient is sent regardless of the overflow condition. If the exponent is less than or equal to 17777_8 an underflow condition exists in which case both the exponent and coefficient are set to zeroes, no flag is set for this condition. A coefficient equal to zero also will cause an underflow condition.

CRAY Y-MP FLOATING ADD FUNCTIONAL UNIT INSTRUCTIONS

Scalar

062ijk Scalar floating sum of (Sj) and (Sk) to Si
062i0k Normalize (Sk) to Si
063ijk Scalar floating difference of (Sj) minus (Sk) to Si
063i0k Transmit normalized negative of (Sk) to Si will normalize the coefficient and toggle the sign bit

Vector

170ijk Vector floating sum of (Sj) and (Vk elements) to Vi
170i0k Normalize (Vk elements) to Vi
171ijk Vector floating sum of (Vi elements) and (Vk elements) to Vi
172ijk Vector floating difference of (Sj) minus (Vk elements) to Vi
172i0k Transmit normalized negatives of (Vk elements) to Vi will normalize the coefficient and toggle the sign bit
173ijk Vector floating difference of (Vj elements) minus (Vk elements) to Vi

CAL

Scalar

Si	Sj+FSk	Scalar floating add
Si	Sj+FSk	Scalar normalized Sk
Si	SJ-FSk	Scalar floating difference
Si	SJ-FSk	Scalar normalized negative Sk

Vector

Vi	SJ+FVk	Vector, scalar sum
Vi	SJ+FVk	Vector, normalized Vk
Vi	Vj+FVk	Vector, vector sum
Vi	SJ+FVk	Vector, scalar difference
Vi	SJ-FVk	Vector, normalized negative Vk
Vi	Vj-FVk	Vector, vector difference

YM05/05

FLOATING ADD EXAMPLES

Add Instruction

J =	040002	140000	000000	000000	$+3_8$
K =	140003	140000	000000	000000	$+ \ -6_8$
					-3_8

Subtract Operation

040003	060000	000000	000000	Shift J
140003	140000	000000	000000	Retain K
040003	060000	000000	000000	
140003	037777	177777	177777	Toggle K
140003	117777	177777	177777	Add coefficients CBP Retain exponent and sign of larger
140003	060000	000000	000000	Toggle result
140002	140000	000000	000000	Normalize

Subtract Instruction

J =	040003	140000	000000	000000	6_8
K =	140002	140000	000000	000000	$- \ -3_8$
					11_8

Add Operation

040003	140000	000000	000000	
040002	140000	000000	000000	Complement K Sign bit
040003	140000	000000	000000	Retain J
040003	060000	000000	000000	Shift K
040003	1.020000	000000	000000	Add coefficients CBP
040004	110000	000000	000000	Shift right to normalize; adjust exponents

Hardware Trng.
YM03/06 J.E.S.

FLOATING ADD EXAMPLES

Add Instruction

J =	040004	004000	000000	000000	$.4_8$
K =	140003	140000	000000	000000	$+ -6.0_8$
					-5.4_8

Subtract Operation

040004	004000	000000	000000	Retain J
140004	060000	000000	000000	Shift K
040004	173777	177777	177777	Toggle J
140004	060000	000000	000000	
040004	1.053777	177777	177777	Add coefficients CBP Retain exponent and sign of larger
040004	053777	177777	177777	
			+1	End around carry
140004	054000	000000	000000	Toggle sign bit
140003	130000	000000	000000	Normalize

Add Instruction

J =	040003	140000	000000	000000	6_8
K =	040002	140000	000000	000000	$+ 3_8$
					11_8

Add Operation

040003	140000	000000	000000	Retain J
040003	060000	000000	000000	Shift K
040003	1.020000	000000	000000	Add coefficients CBP
040004	110000	000000	000000	Normalize result

Hardware Trng.
YM03/07 J.E.S.

CRAY Y-MP FLOATING ADD OPTIONS USED

OPTION TYPE	NUMBER USED	DESCRIPTION
FA	1	Coefficient align and exponent control
FB	4	Coefficient alignment bit N + 0 - N + 46
FC	1	Final sum bits 2^{23} - 2^{0}
FD	1	Final sum bits 2^{47} - 2^{24}
FE	2	Coefficient normalize bit N+ 0 - N + 23
FF	1	Final exponent adjust
	10 Total Options	

FLOATING-POINT ADD (OPTIONS INVOLVED)

(FA) Option

The (FA) receives the J and K exponents from the Scalar/Vector registers. The (FA) determines which exponent is larger, J or K, in order to equalize the exponents and right shift the smaller operand's coefficient by the difference.

The following control term tells the (FB) option how much to right shift.

(J>K) - J is larger than K by 48 shifts, so zero out the K coefficient.

(J+1=K) - J is only different from K by one on the exponent bits 2^{62}-2^{54}.

(J=K) - Bits 2^{62}-2^{54} of the exponents are the same for J and K.

(K>J) - K is larger than J by 48 shifts, so zero out the J coefficient.

(K+1=J) - K is only different from J by one on exponent bits 2^{62}-2^{54}.

(K=J) - Bits 2^{62}-2^{54} of the exponents are the same for K and J.

Underflow is detected on the (FA) option by checking bits (2^{62}, 2^{61}) of the larger exponent for both being zero. If this occurs, the (FA) insures that the final coefficient equals zeroes on the (FEs), while the (FF) will set the final exponent to zeroes if the coefficient equal zeroes or if (2^{62}, 2^{61}) of the exponents are both equal to zero.

Unequal signs determine if an add or subtract operation is taking place on the (FE, FD) options and thereby allows an end around carry or a shift (-1).

(FB) Options

There are four (FB) options. Two (FB) options input the J coefficient while the other two input the K coefficient. On the two (FB) options the J and K coefficient are divided into odd and even bits. Except for a right shift of one, all bits are present on the (FB) options.

To determine how many places to right shift the smaller coefficient, the exponent bits (2^{53}-2^{48}) are subtracted for a difference of less than 48 places. A difference of greater than 48 places causes the smaller exponent's coefficient to be zeroed. The difference is only valid if the (FB) receives control term (J=K, K=J) from the (FA). If the (FBs) receives a control term (J+1=K, K+1=J) from the (FA), the (FB) utilizes a shift count which is the twos complement of the difference. If the control term (J>K, K>J) is sent to the (FB), the smaller coefficient, either J or K, gets zeroed, because the right shifts are greater than 48_{10} places. The unshifted or larger coefficient is toggled during a subtract operation, but not during an add operation.

Hardware Trng.
YM05/01B J.E.S.

After the right shift is completed on the smaller coefficient, the J and K coefficients are sent to the (FC, FD) options to be added together.

(FC, FD) Options

The (FC, FD) options add the two 48-bit coefficients together. The (FC) options handle the lower half, or bits (2^{23}-2^{0}), while the (FD) handles the upper half. The two intercommunicate with section enables and section carries, looking for a carry across the binary point. Depending on the control term "sign unequal" the (FC, FD) will allow an end around carry or a shift (-1). If a shift (-1) occurs, which happens when the signs are equal, the (FD) passes sum bit 2^{24} to the (FC) option, and 2^{47} on the (FD) is forced set. The results of the add are toggled during a subtract operation, and there is no carry across the binary point. The (FC, FD) options perform the right shifts of one or shift (-1) when the signs are equal and there is a carry across the binary point.

When the (FC, FD) have finished adding the J and K coefficients, the (FC, FD) will count the number of leading zeroes and send the results to the (FE) and (FF) options. The 48-bit coefficient is divided first into six groups of 8 bits each, with the upper five groups reporting if their group contains all zeroes; and then into two groups of 24 bits with similar reporting to (FE) and (FF) options.

(FE) Options

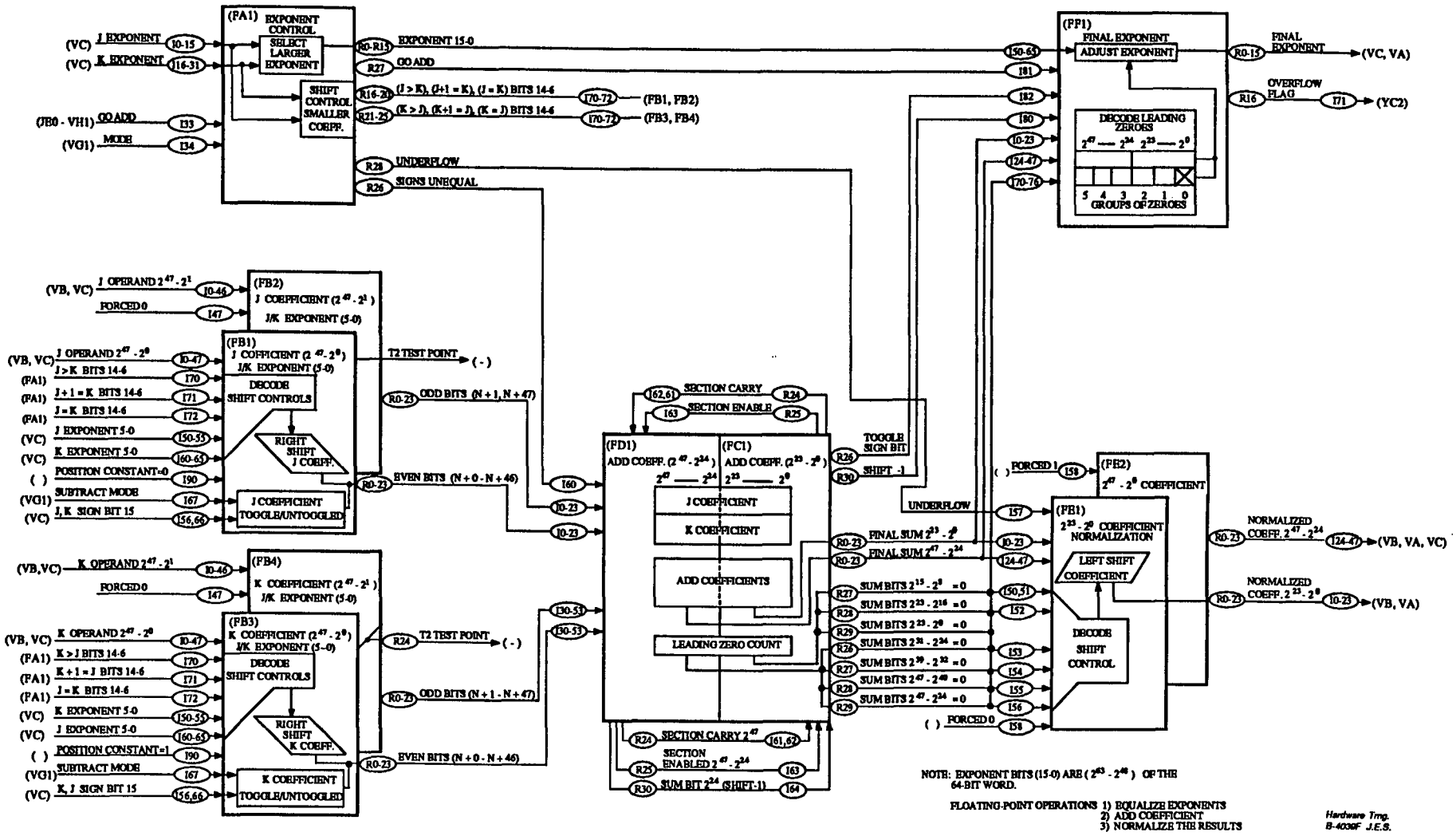
There are two (FE) options. Option (FE1) handles the lower 24 bits of the final coefficient, while (FE2) handles the upper 24 bits. Both (FE) options decode the leading zeroes count sent from the (FC, FD) options, to determine how many places to left shift the coefficient for normalization.

If an underflow condition is sent to the (FEs) the results or outputs of the (FEs) are zeroed out. The final normalized 48-bit coefficient is sent to either the Scalar register or Vector register.

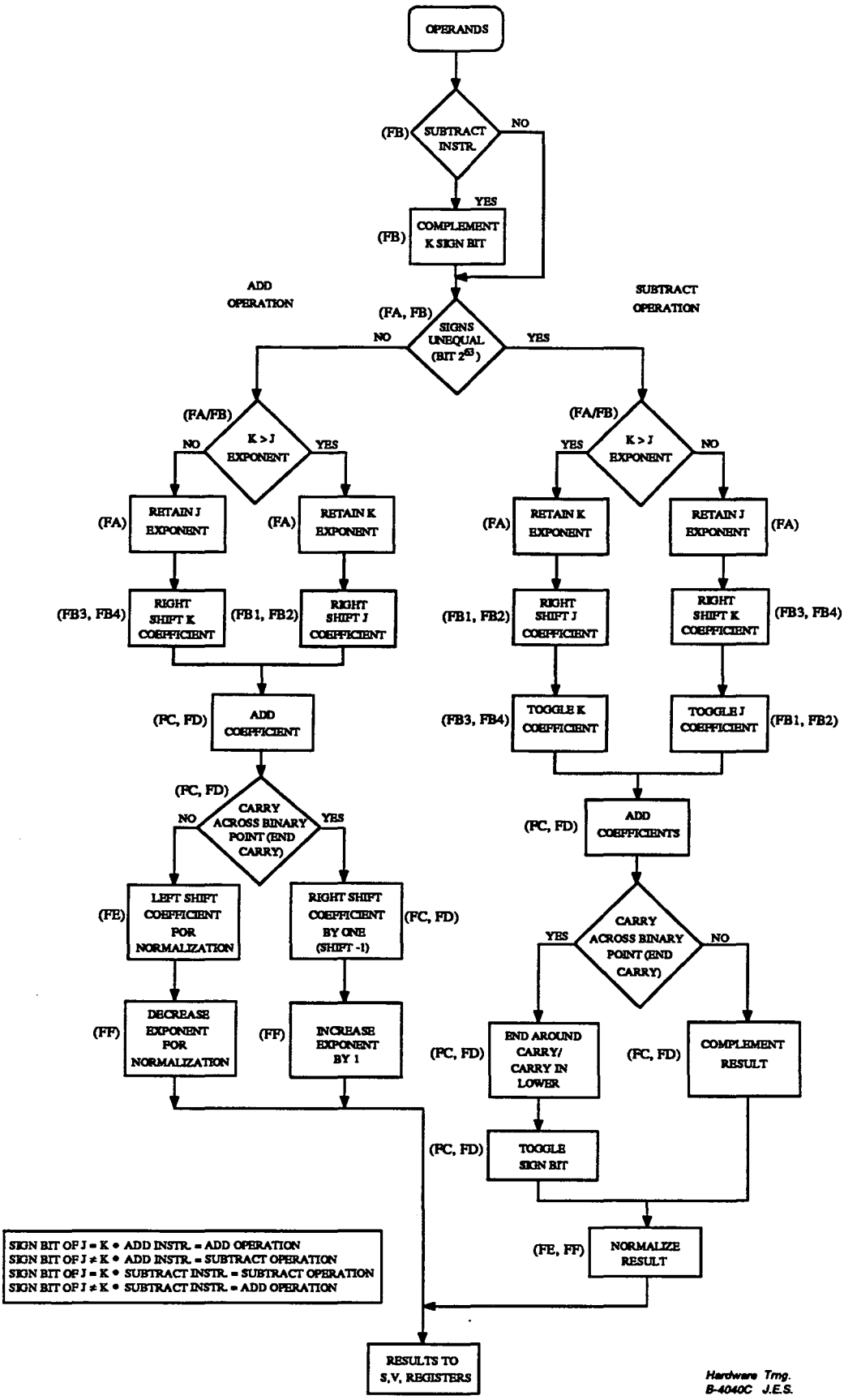
(FF) Option

The (FF) option handles the exponent. The (FF) receives from the (FC, FD) options a copy of the coefficient along with the leading zero count to determine how many places from (47_{10}-0) to subtract from the exponent. The exponent would get cleared if the coefficient equals zero, or if bits (2^{62}, 2^{61}) of the original exponent are equal to zero. No flag is sent for an underflow condition. If exponent bits (2^{62}, 2^{61}) are both equal to a one, an overflow condition exists, in which case the exponent is forced to 60000_{8} and an overflow flag gets set. During an overflow condition the calculated coefficient is sent from the (FE) option regardless of the overflow condition, to the Scalar/Vector registers. The final adjusted exponent along with the sign of the coefficient is sent to the Scalar register or Vector register from the (FF) option.

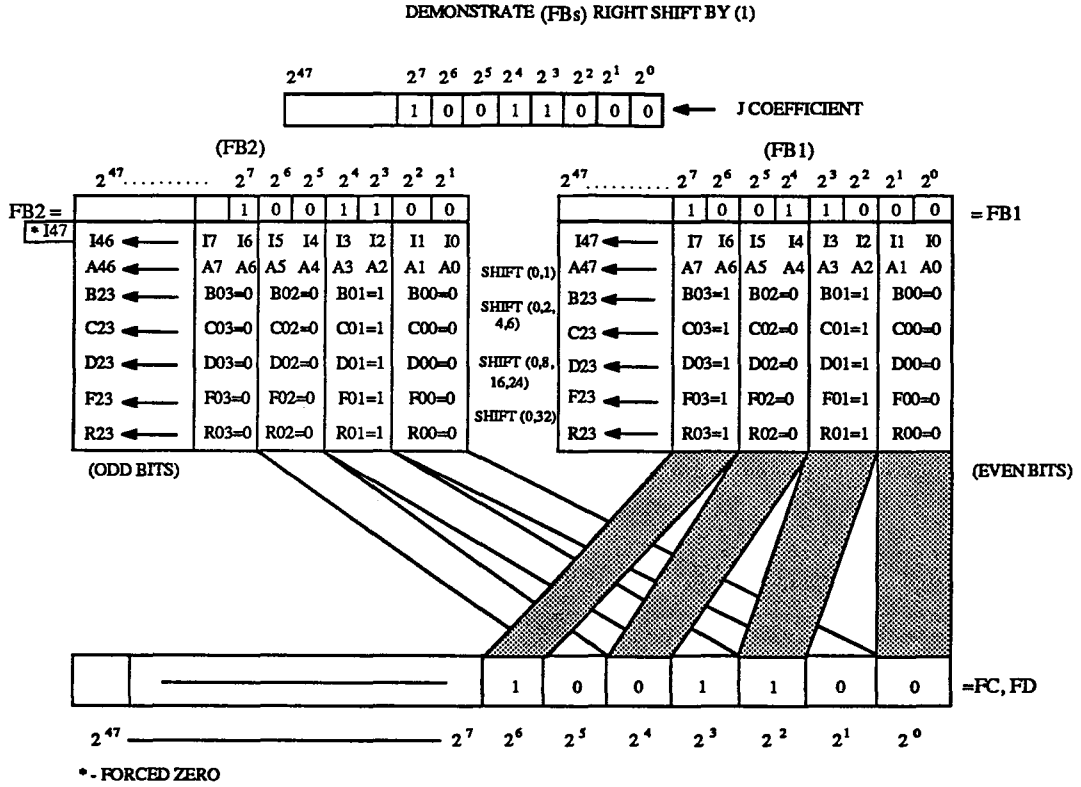
CRAY PROPRIETARY



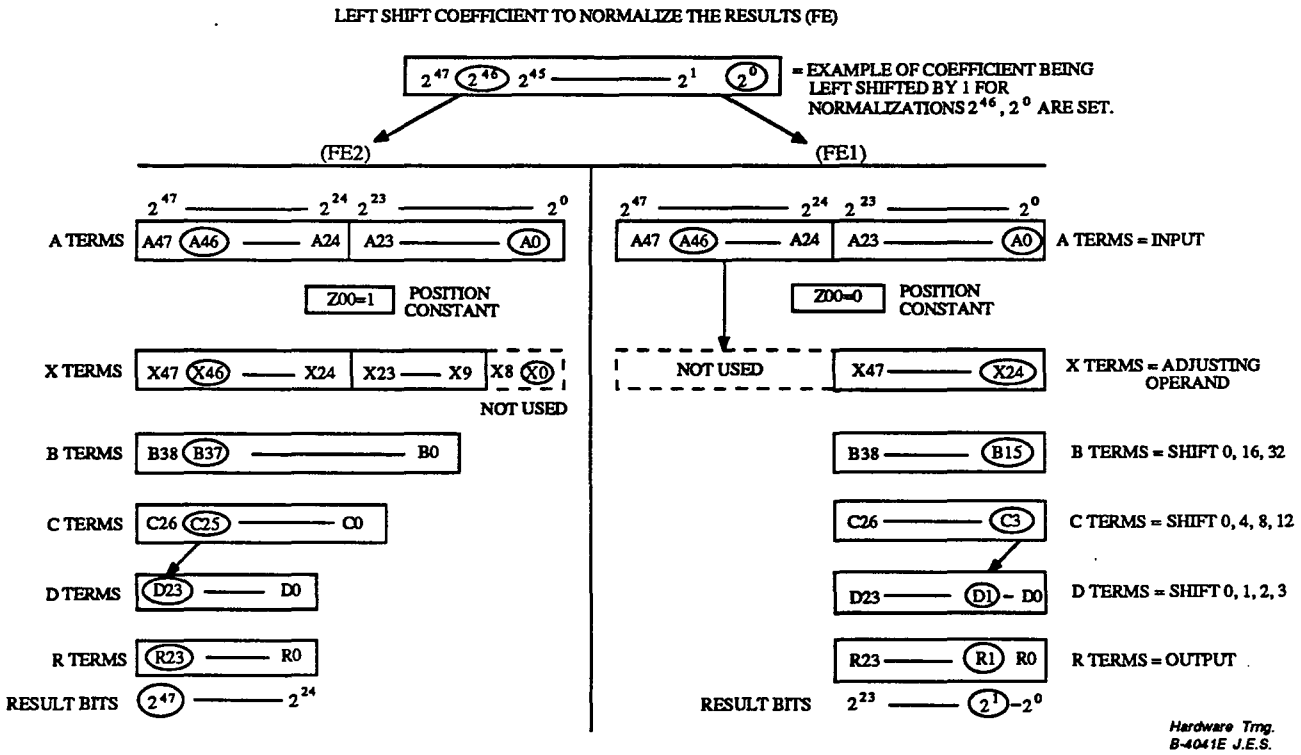
CRAY Y-MP FLOATING-POINT ADD BLOCK DIAGRAM



CRAY Y-MP FLOATING-POINT ADD FLOW CHART



CRAY Y-MP FLOATING ADD (FB) RIGHT SHIFT



CRAY Y-MP FLOATING-POINT ADD (FE) LEFT SHIFT

CRAY Y-MP FLOATING RECIPROCAL APPROXIMATION OVERVIEW

The Floating Reciprocal unit is used in conjunction with the Floating Multiply unit to complete a divide sequence, for example:

$$\frac{10}{2} = 5 \qquad \text{as} \qquad \frac{10}{1} \times \frac{1}{2} = 5.0$$

The Floating Reciprocal unit is used to find a value which is equivalent to 1/2 or .5. The Floating Reciprocal unit would not come up with a value .5 but, an approximation of .5 which would be .4999999999. The reciprocal value of 1/2 or .4999999999 may then be multiplied by 10 using the Floating Multiply unit to get the value of 5.0 or 4.999999999.

When using the results of the Floating Reciprocal functional unit not all of the bits in the coefficient are used. In fact, the unit will output 33 bits of the coefficient of which only 30 bits are accurate. Thirty bits of accuracy is obtained from doing two iterations of Newton's formula inside the reciprocal unit. If 48 bits of accuracy is required, or full precision, a third iteration of Newton's formula would need to be performed. The third iteration is performed using the Floating Multiply functional unit.

The hardware will run the coefficient through two iterations of Newton's formula while the exponent will get complemented and a value of + 2 added to the complement. The exponent and coefficient are then sent to the Scalar/Vector registers. The Floating Reciprocal unit performs two iterations of Newton's formula which are as seen in the hardware:

$-A_1 = -2A_0 + A_0^2B$ First iteration

$-A_2 = -2A_1 + A_1^2B$ Second iteration

Looking closely at the two formulas you will notice three multiplies which are:

a) A_0^2B = A_1 pyramid

b) $A_1 * A_1$ = A_1^2 pyramid

c) A_1^2B = A_2 pyramid

The A_1 pyramid is formed by $A_0^2 * B_1$. A_0^2 is gotten from a look-up table along with $-2A_0$ on the (RA) option, and B_1 is 24 bits or bits ($2^{47} - 2^{24}$) of the original coefficient. These two operands are multiplied together. Once the multiply is complete the pyramid is summed using 3-bit adds and finalized by using 2-bit adds. While the adds are taking place the $-2A_0$, which is also looked-up, is added into the summation in two places along with some forced ones and zeroes. When the summation has completed, the first iteration of Newton's formula is complete, which is the following:

$$-A_1 = -2A_0 + A_0^2B$$

Hardware Trng.
YM03/03A J.E.S.

The second and third multiply pyramids are to form the second iteration which is:

$$-A_2 = -2A_1 + A_1^2B$$

Looking at the second iteration, there are two multiplies that take place. The first multiply is A_1 * A_1 for A_1^2, and the second multiply is taking A_1^2 and multiplying it by B2. The multiply of A1 * A_1 takes 18 bits of A_1 and multiplies it to itself. When multiplying a number by itself, two things happen: for one, the pyramid is duplicated in two places; and second, a portion of the pyramid does not need to be multiplied. Because the pyramid is duplicated, only half of the pyramid needs to be multiplied, and when adding the pyramid the summation is multiplied by two to pick up the other half that was not produced. To perform a multiply of two the summation would need to be shifted to the left by one place, or instead, shift the portion which does not need to be multiplied to the right one place. This requires fewer bits to be shifted and gives the same results.

Once A_1^2 pyramid summation has completed, the next step would be to perfom the multiply of A_1^2 * B_2 and while adding the A_2 pyramid put $-2A_1$ into the summation to form $-A2 = -2A_1 + A_1^2B$.

The third and final multiply will take 36 bits of the A_1^2 pyramid and multiply it by 36 bits of B_2. The bits that makeup B_2 are coefficient bits (2^{47} - 2^{11}). While the summation of the A_2 pyramid is taking place, $-2A_1$ is put into the summation into sum bit positions (2^{-2} - 2^{-17}). When the pyramid has been completely summed, bits (2^{47} - 2^{15}) are sent to the result register along with the exponent. Of the 33 coefficient bits only 30 are considered accurate.

Underflow/Overflow

When the Floating Reciprocal unit finds that the exponent is greater than or equal to 60000_8, the Floating Reciprocal unit will flag an overflow condition, and force the exponent to 60000 and 2^{47} equal to zero. The exponent range check is performed prior to complementing and adding of a plus two, to the exponent on the (RN) options. When the exponent is less than 20000_8 the Floating Reciprocal unit will flag an underflow condition and force the exponent again to 60000_8 and 2^{47} equal to zero. Both underflow and overflow conditions would send a flag to the exchange package.

Reciprocal Approximation Iteration Formula

PROBLEM: Given a number B such that $.5 \leq B < 1$, find a number A such that $\frac{1}{A} = B$.

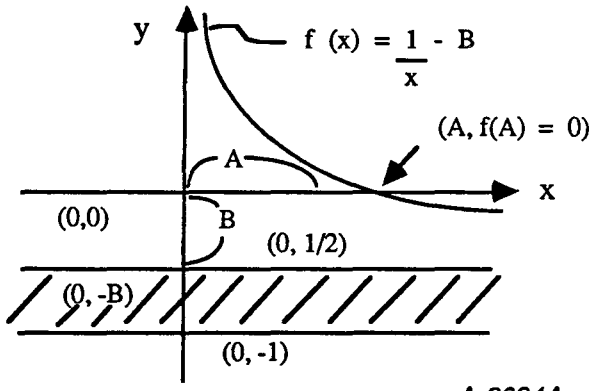
(Note that the restriction on the domain of B is only natural, since any normalized coefficient will always lie between .5 and 1.)

To find a good approximation value to A, first consider the function $f(x) = \frac{1}{x} - B$.

Now find a number A such that $f(A) = \frac{1}{x} - B = 0$. In other words, the number A we are seeking is the root of the equation $\frac{1}{x} - B = 0$.

From the graph, the value of A is the x-coordinate of the point located at the intersection of the curve of f (x) and the x-axis. To find the approximate value of A, use Newton's method for approximating the real roots of an arbitary equation f (x) = 0:

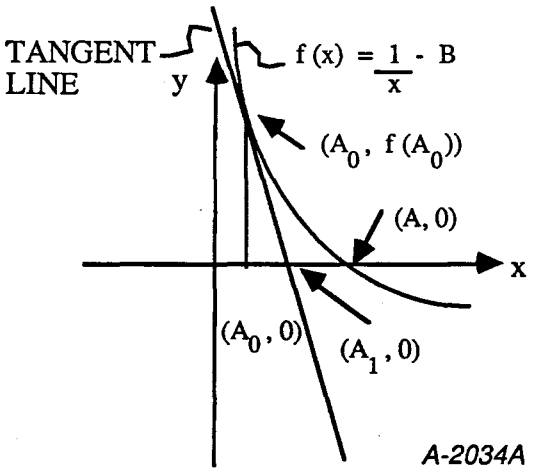
Step 1: Select an initial value A_0 (by using the look-up tables in CRAY Hardware) which is close to the value of A.



A-2034A

YM04/01

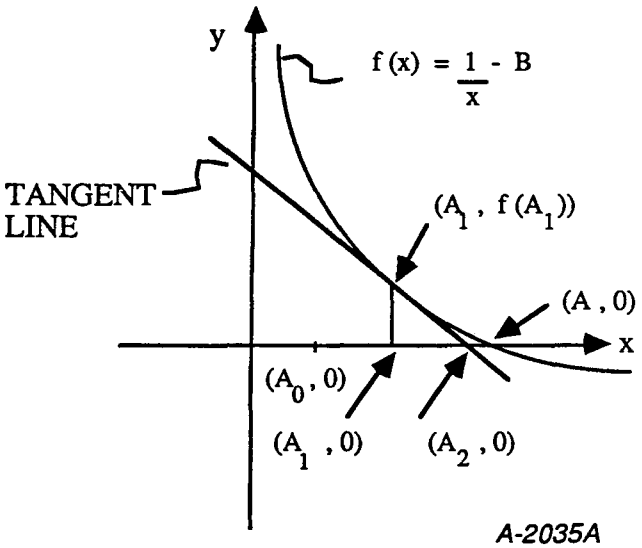
Step 2: Draw a tangent line at $(A_0, f(A_0))$ and find A_1, where $(A_1, 0)$ is the intersection of the tangent line and the x-axis.



A-2034A

From the equation of the tangent line $y = \frac{-x}{A_0^2} - B + \frac{2}{A_0}$, substituting $x = A_1$, $y = 0$, and factoring out A_0^2 we find that $A_1 = 2A_0 - A_0^2B$.

Step 3: Draw a tangent line at $(A_1, f(A_1))$ and find A_2, where $(A_2, 0)$ is the intersection of the tangent line and the x-axis.



A-2035A

From the equation of the tangent line $y = \frac{-x}{A_1^2} - B + \frac{2}{A_1}$, substituting $x = A_2$, $y = 0$, and factoring out A_1^2, we find that $A_2 = 2A_1 - A_1^2B$.

As you see, A_2 is much closer to A than either A_0 or A_1. By repeating this process, a good approximation to the value of A can be found.

YM04/02

NEWTON'S FORMULA DERIVATION

$$A_{(i+1)} = 2A_i - A^2_iB,$$

where:

$A_{(i+1)}$ = Next iteration

A_i = Current iteration

B = Divisor

A_0 and A_0^2 in look-up table

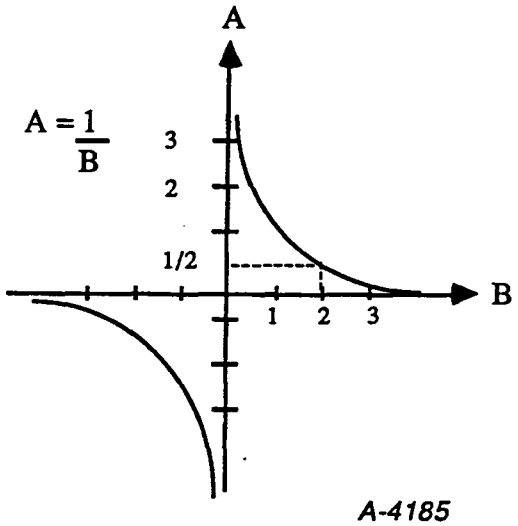
An initial guess starts the process with A_0, i.e.

A_1	$= 2A_0 - A_0^2 B$	1st approximation
A_2	$= 2A_1 - A_1^2 B$	2nd approximation result of reciprocal unit
A_3	$= 2A_2 - A_2^2 B$	3rd approximation
A_n	$= 2A_{(n-1)} - A^2_{(n-1)} B$	nth approximation

The more iterations, the closer the approximation approaches true reciprocal value; but the closer the initial guess A_0 is to the final result, the fewer the number of iterations needed to obtain a good approximation.

Further Explanation and Examples

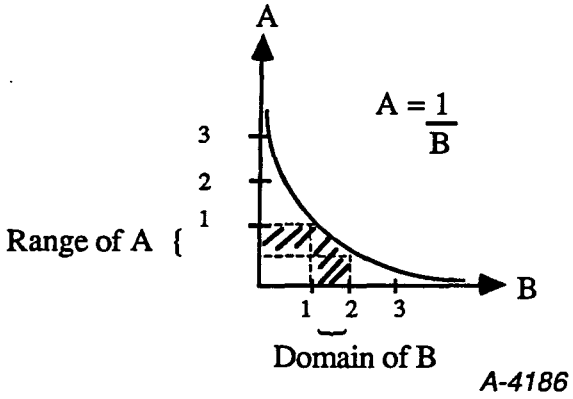
As can be seen from the foregoing discussion, the CRAY computes the reciprocal A of a number B by approximating the solution to the equation $\frac{1}{x} - B = 0$, using Newton's approximation method. This is the same as trying to read the value of A from the graph of the hyperbola $A = \frac{1}{B}$:



A-4185

YM04/03A

Since the CRAY uses only the absolute of B, we can restrict ourselves to the right-hand branch of the hyperbola above. Also, since the natural domain of the normalized coefficient of B is such that $.5 \leq B < 1$ (or $.4_8 \leq B < 1$), this will only produce A values such that $1 < A \leq 2$. We can further restrict the range of possible A values to $.4_8 < A \leq 1$ by expanding the domain of B to $1 \leq B < 2$. This can easily be done by shifting the normalized coefficient of B one place to the left, resulting in the change of Bs domain from $.4 \leq B < 1$ to $1 \leq B < 2$. We now have the whole story: to find the reciprocal A of any number B by reading it from the graph of the hyperbola $A = \frac{1}{B}$, first normalize the coefficient of B, then shift it one place to the left. This will force B to lie between 1 and 2 ($1 \leq B < 2$), and A to lie between .4 and 1 ($.4 < A \leq 1$), as indicated below:



A-4186

Now find the value of B on the horizontal axis, follow it up to where it intersects the curve, and then to the left to find the reciprocal on the A axis. For example, to find the reciprocal of 24.4375_{10}, do the following steps:

24.4375_{10}

1) 30.3400_8 Convert to octal number

2) $.6070 \times 2^5$ Convert to normalized floating point number

3) 1.4160×2^4 Left shift coefficient by 1 place

4) $.5172 \times 2^{-4}$ Read reciprocal from graph

5) $.0247_8$ Adjust binary point as specified by exponent (In this case, right shift coefficient by 4 places)

6) $.0408_{10}$ Convert back to decimal

In practice, however, the CRAY does not store the graph of a hyperbola internally, and so step 4 of the above example is replaced by the following steps of Newton's approximation method as discussed previously:

4a) Find largest B value in hardware lookup table that is ≤ 1.416 (In this case, $B = 1.414$).

YM04/04

4b) Compute

1st iteration: $A_1 = 2\,A_0 - A_0^2B$ using table values

$= -(-2A_0 + A_0^2B)$ for $-2A_0$ and A_0^2

$= -(.534 + .334400 \times 1.414)$

$= .524052$

4c) 2nd iteration: $A_2 = -(-2A_1 + A_1^2B)$

$= -(.527654 + .3417076 \times 1.416)$

$= .517116$

Note that the reason the CRAY obtains such a high degree of accuracy after only two iterations is because of the closeness of the initial guess A_0 to the final result.

HANDLING OF B's EXPONENT

B = 40000 + E 1XXXXX XXXXXX XXXXXX

Exponent Coefficient

Value Of B = 2^E * .1XXX--------X Normalized Floating-Point Number

B = 2^{E-1} * 1.XXX-------X Left Shift Coefficient By 1

Let b = 1.XXX-------X Rename Coefficient

Then B = 2^{E-1} * b

$$\frac{1}{B} = \frac{1}{2^{E-1} * b} = \frac{1}{2^{E-1}} * \frac{1}{b}$$

Let n = (E-1)

Simplify Negative Exponent

$$\frac{1}{2^n} = \frac{2^{-n}}{1} \quad \text{OR} \quad \frac{1}{2^{(E-1)}} = \frac{2^{-(E-1)}}{1} = \frac{2^{-E+1}}{1}$$

$$\frac{1}{B} = \frac{2^{-E+1}}{1} * \frac{1}{b}$$

Another method which is used in the mainframe: perform ones complement plus 2.

Example: 51132 Exponent

26645 Peform ones Complement
\+ 2 Add One For Normalization
Add One For twos Complement
26647

YM04/06B

CRAY Y-MP FLOATING RECIPROCAL FUNCTIONAL UNIT INSTRUCTIONS

Scalar

070ij0 Floating Reciprocal Approximation of (Sj) TO Si

174ij0 Floating Reciprocal Approximation of (Vj Elements) to Vi

CAL Format

Si /HSj Scalar

Vi /HVj Vector Reciprocal

CRAY Y-MP FLOATING RECIPROCAL OPTIONS USED

OPTION TYPE	NUMBER USED	DESCRIPTION
RA	3	A_0, A_0^2
RB	3	$A_0^2 * B_1$
RC	1	$A_0^2 * B_1$
RD	2	A_1 or A_1^2 Final Summation
RE	1	A_1^2 Summation
RF	1	A_1^2 Summation
RG	2	$A_1^2 * B_2$ First Level Sum
RH	3	$A_1^2 * B_2$ First Level Sum
RI	1	$A_1^2 * B_2$ First Level Sum
RJ	2	$A_1^2 * B_2$ First Level Sum
RK	1	$A_1^2 * B_2$ First Level Sum
RL	3	Final Coefficient Sum lower
RM	1	Final Coefficient Sum upper
RN	1	Result Exponent
RO	1	Vector Pop
	26	Total options

YM04/08A

CRAY Y-MP FLOATING-POINT RECIPROCAL OPTIONS INVOLVED

(RA1, RA2, RA3) Options

The three (RA) options work in parallel and output the same results at the same time; however, the outputs go to different locations. The (RA1) sends its outputs to the (RB1) option, the (RA2) sends its results to the (RB2) options, and (RA3) sends its outputs to the (RB3) and (RC1) options. The (RA) chips receive the upper 7 bits of the coefficient, numbered (2^{46}-2^{40}), and from these bits the (RAs) will generate values for A_0^2 and A_0.

(RB1, RB2, RB3) Options

The (RB) options first fanout the B coefficient to form the A_1 pyramid and A_2 pyramid. The (RB1) option forms part of the A_1 pyramid which is generated from the logical products of A_0^2 and B_1. After the logical products are completed, the summation of the pyramid begins, using 3-bit adds. During the summation, $-2A_0$ is put into (RB1 and RB2) for rounding purposes, and the (RB3) adds forced ones into the summation. Because of the way the pyramid is formed, a portion of (RB3)'s output is not needed, and these terms are disabled by forcing zeroes into the appropriate inputs.

(RC1) Option

The (RC1) option handles the upper 9 bits of the A_1 pyramid. During the summation $-2A_0$ is added in to complete the first iteration $A_1 = -2A_0 + A_0^2B$. The (RC1) does not complete the summation on the upper end of the pyramid but sends its partial sums to the (RD1) option.

(RD1) Option

The (RD1) option completes the summation of the A_1 pyramid. The (RD1) receives its inputs from the (RB1, RB2, RB3) and (RC1) options. When the (RD1) has completed the summation for A_1, the (RD1) will fanout A_1 to the (RF1) and (RE1) to form the A_1^2 pyramid; also the (RD1) sends A_1 to the (RC1) option for a 5 CP delay, after which it is added into the A_2 pyramid. The (RD1) option also delays for 8 CPs, 6 bits of B_2, which are then sent to the A_2 pyramid.

(RD2) Option

The (RD2) option is identical to the (RD1) option except that the (RD2) option finalizes the upper half of the A_1^2 pyramid. There are forced zero and one conditions present to make the (RD2) option respond differently from the (RD1) option. After the (RD2) option has completed the summation of the A_1^2 pyramid, the (RD2) will fanout its results of A_1^2 to the (RG1), (RH1, RH2, RH3), (RI1), (RJ), (RK1) options to help form the A_2 pyramid. The A_2 pyramid is formed from the product of A_1^2 and B_2. The (RD2) option also delays for 8 CPs 6 bits of B_2, which are then sent to the A_2 pyramid.

(RE1) Option

The (RE1) option forms the lower half of the A_1^2 pyramid. The (RE1) finalizes the summation on bits (2^{-19}-2^{-36}) of A_1^2, which is then fanned out to the A_2 pyramid on the (RG2), (RH2), (RJ1, RJ2) and (RK1) options. The (RE1) also forces (2^{-34} and 2^{-35}) equal to zero, because whether or not the bits are set in the pyramid, the summation will always come up with zeroes in these positions.

(RF1) Option

The (RF1) option forms the upper half of the A_1^2 pyramid. The (RF1), however, does not finalize the summation of A_1 times A_1, or A_1^2, but sends its partial summation to the (RD2) for completion. The (RF1 and RE1) options form the A_1^2 pyramid. However, not all the bits within the pyramid are actually produced, since this would result in a considerable amount of duplication for two reasons. The commutativity of multiplication assures us that when both operands are the same, bit 2^1 * bit 2^2 will yield the same result as bit 2^2 * bit 2^1, and therefore the result of such a multiplication does not need to be computed twice. Also, the logical product of a bit with itself is the same bit (i.e., 1 * 1 = 1 and 0 * 0 = 0), and so all products of the form bit 2^N * bit 2^N do not need to be computed at all, since the answer is always bit 2^N.

(RG1, RG2) Options

The (RG) options form part of the A_2 pyramid, which is made from A_1^2 * B_2. Once the pyramid is formed, the (RG) option begins summing the pyramid. The (RG) options sum bit positions (2^{-34}-2^{-27}) of the A_2 pyramid. The (RG) options do not complete the summation but send their partial summation to the (RL1, RL2) options for completion.

(RH1, RH2, RH3) Options

The (RH) options are identical to the (RG) options except that the input terms that make-up (A0 - A12) and (B0 - B12) are complemented. Since many of the terms that would be produced by the (RH2) option are not used because of their location in the pyramid, these outputs are disabled by forcing ones into the appropriate input. The partial summation of A_2 is sent to the (RL1, RL2, RL3, RM1) options for completion.

(RI1) Option

The (RI1) option handles the upper 13 bits of the A_2 pyramid. The (RI1) computes the product of $A_1^2 * B_2$, and partially sums the pyramid. The partial sum is then sent to the (RM1 and RL3) options for completion.

(RJ1, RJ2) Option

The (RJ) options handle the least significant bits of the pyramid, which take care of rounding up sum bits ($2^{-38} - 2^{-32}$) of the A_2 pyramid, so that the final pyramid contains only bit positions ($2^{-1} - 2^{-33}$), though only bits ($2^{-1} - 2^{-30}$) are considered accurate. The (RJ) options do not complete the summation, but sends their partial sum to the (RL1) option for completion.

(RK1) Option

The (RK1) option handles three different areas of the A_2 pyramid. One of the positions is sum bits ($2^{-20} - 2^{-13}$), another is sum bit positions ($2^{-33} - 2^{-27}$), and finally sum bit positions ($2^{-38} - 2^{-34}$). The (RK1) option does not finalize these positions, but passes its partial summations to the (RL1, RL2, RL3) options.

(RL1, RL2, RL3) Options

The (RL) options finalize the A_2 pyramid summation of $A_1^2 * B_2$. The (RL1) will output result bits ($2^{20} - 2^{15}$) to the Scalar or Vector registers, while (RL2) will output result bits ($2^{28} - 2^{21}$). (RL3) will add into its summation bits ($2^{-17} - 2^{-12}$) of $-2A_1$, and when the summation is completed it will output result bits ($2^{36} - 2^{29}$) to the Scalar or Vector registers. The outputs from the (RL) options are forced to zeros unless the control term Go Reciprocal is present.

(RM1) Option

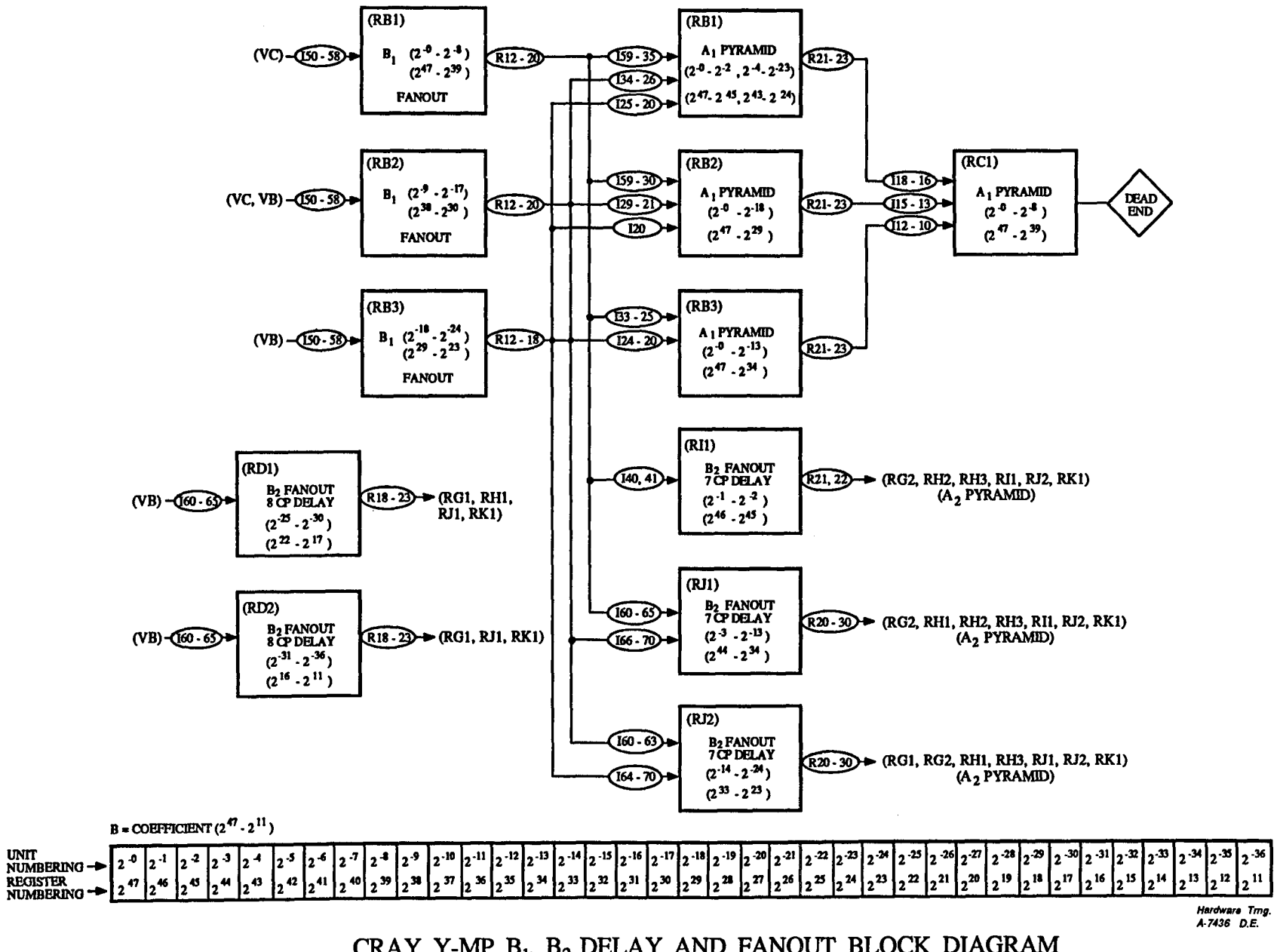
The (RM1) option handles the upper summation of the A_2 pyramid, bit positions (2^{-2}-2^{-14}). While the summation is taking place, the (RM1) option adds in bits (2^{-2}-2^{-14}) of $-2A_1$, which it received from the (RC1) option after a 5 CP delay. When the (RM1) option has completed its summation, it will output bits (2^{46}-2^{37}) to the Scalar or Vector registers. The (RM1) option also delays exponent bits (2^{63}-2^{60}) for 12 CPs, and then sends them to the (RN1) option.

(RN1) Option

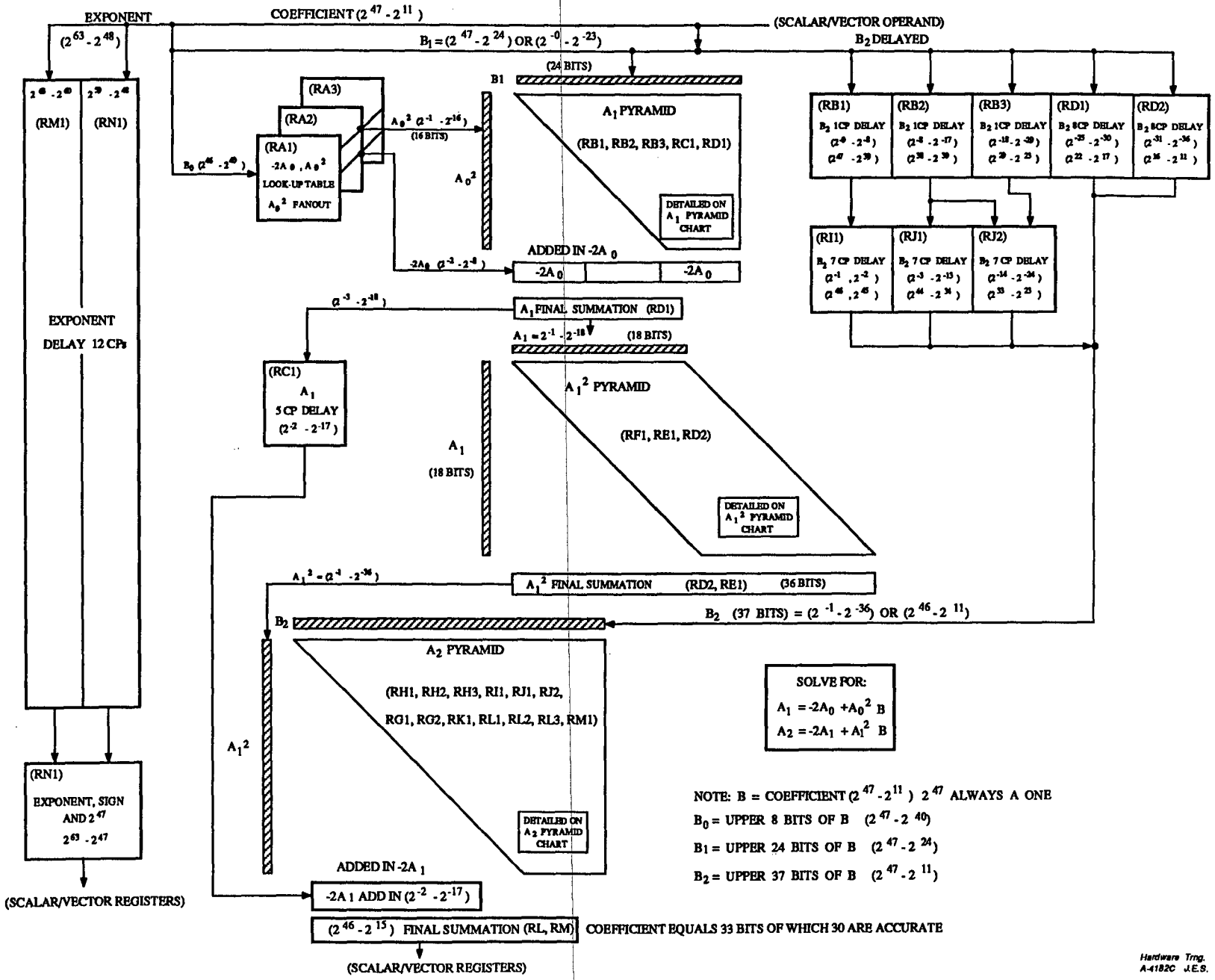
The (RN1) option delays the exponent bits (2^{59}-2^{48}) for 12 CPs, while the A_1, A_1^2 and A_2 pyramids are being formed and summed. Exponent bits (2^{63}-2^{60}) are delayed on the (RM1) option. After the 12 CP delay, the (RN1) option will receive all the bits of the Exponent (2^{63}-2^{48}), of which the sign bit (2^{63}) is left unchanged, and bits (2^{62}-2^{48}) are first complemented and then a value of (+2) is added into this complemented exponent. At CP 14, the exponent (bits 2^{62}-2^{48}), the Sign (bit 2^{63}), and bit (2^{47}), which is forced to a "1", are sent to the Scalar or Vector registers, provided the Go Reciprocal signal is present. Otherwise, the results are forced to zeroes. The (RN1) also performs a range check on the final exponent by checking bits (2^{62}, 2^{61}). If both of these bits are the same, a Range Error has occurred, in which event the resultant exponent is forced to 60000 and a Flag is sent to the Exchange package indicating a Floating-point Range Error.

(RO1) Option

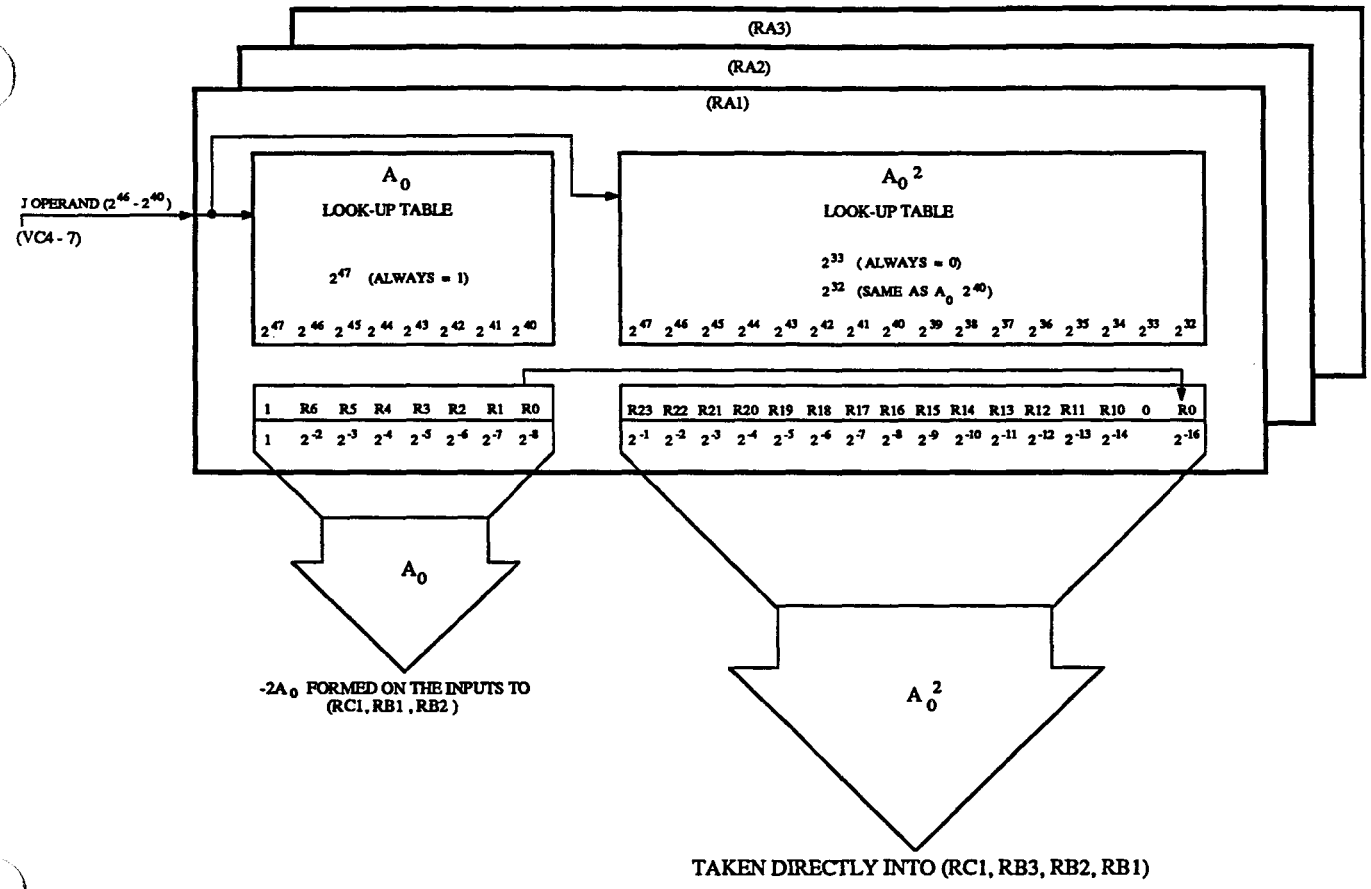
The (RO1) option is the control for the Reciprocal Functional unit. The (RO1) fans out the Go Reciprocal control term, which allows the Reciprocal results to be sent to the Scalar or Vector registers. The (RO1) also decodes the k field of the instruction to determine whether to perform a Vector Pop Count or Floating Reciprocal operation. The (RO1) option contains the entire Vector Pop/Parity unit.



CRAY Y-MP B_1, B_2 DELAY AND FANOUT BLOCK DIAGRAM



CRAY Y-MP FLOATING RECIPROCAL UNIT FLOW CHART



Example of generating A_0^2 and $-2A_0$: Sj or Vj = | 040000 | 152036 | 132461 | 000000 | (parcel format)

A) Take upper 8 bits of the coefficient = .1520 (in parcel format)

B) Shift 8 bits to the left by 1 = 1.520 (no longer in parcel format)

C) Look-up A_0 and A_0^2 for 1.520 = A_0 = .464 (A_0 only 8 bits)

A_0^2 = .271220 (A_0^2 only 16 bits)

D) A_0 becomes $-2A_0$ by complementing A_0 = ($-A_0$) and shifting A_0 to the left by 1 for a multiple of 2. = ($-2A_0$)

Example of .464:

1) .464 = .10011010

2) Complement = .01100101

3) Shifted by 1= 0.1100101

4) 0.1100101 = .624 which is the value for $-2A_0$

5) .624 is numbered 0. 1 1 0 0 1 0 1 in the Boolean

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7} 2^{-8}

E) A_0^2 is taken directly from (RA) 2^{33} (always = 0) and 2^{32} is the same as 2^{40} of A_0 .

Hardware Tmg. A-41628 J.E.S.

CRAY Y-MP A_0 AND A_0^2 LOOK-UP TABLES

(RA) $A_0{}^2$ FANOUT

$A_0{}^2$	(RA) R TERMS	SENT (RA1)	(RA2)	(RA3)	(RA3)			
		RECEIVED						
		(RB1)	(RB2)	(RB3)	(RC1)			
2^{-16}	R00	I00, I40	I00		I20			
2^{-14}	R10	I01	I01	I01				
2^{-13}	R11	I02	I02	I02				
2^{-12}	R12	I03	I03	I03				
2^{-11}	R13	I04	I04	I04				
2^{-10}	R14	I05	I05	I05				
2^{-09}	R15	I06	I06	I06	I00			
2^{-08}	R16	I07	I07	I07	I01			
2^{-07}	R17	I08	I08	I08	I02			
2^{-06}	R18	I09	I09	I09	I03			
2^{-05}	R19	I10	I10	I10	I04			
2^{-04}	R20	I11	I11	I11	I05			
2^{-03}	R21	I12	I12	I12	I06			
2^{-02}	R22	I13	I13	I13	I07			
2^{-01}	R23	I14	I14	I14	I08			

(RA) $-2A_0$ FANOUT

$-2A_0$ BITS	(RA) R TERMS	SENT (RA1)	(RA2)	(RA3)				
		RECEIVED						
		(RB1)	(RB2)	(RC1)				
-8	R00	I00, I40	I00	I20				
-7	R01	I41		I21				
-6	R02	I42		I22				
-5	R03	I43		I23				
-4	R04	I44		I24				
-3	R05		I40	I25				
-2	R06		I41	I26				

Hardware Trng.
A-4166B J.E.S.

CRAY Y-MP FLOATING RECIPROCAL $A_0{}^2$ AND $-2A_0$ FANOUT

RECIPROCAL TABLE LOOK-UP VALUES

B	A_0	A_0^2	$-2A_0$
1.000	.776	.774004	.000
1.004	.772	.764044	.010
1.010	.766	.754144	.020
1.014	.762	.744304	.030
1.020	.756	.734504	.040
1.024	.752	.724744	.050
1.030	.750	.721100	.054
1.034	.744	.711420	.064
1.040	.740	.702000	.074
1.044	.734	.672420	.104
1.050	.732	.666644	.110
1.054	.726	.657344	.120
1.060	.722	.650104	.130
1.064	.720	.644400	.134
1.070	.714	.635220	.144
1.074	.710	.626100	.154
1.100	.706	.622444	.160
1.104	.702	.613404	.170
1.110	.700	.610000	.174
1.114	.674	.601020	.204
1.120	.672	.575444	.210
1.124	.666	.566544	.220
1.130	.664	.563220	.224
1.134	.660	.554400	.234
1.140	.656	.551104	.240
1.144	.652	.542344	.250
1.150	.650	.537100	.254
1.154	.646	.533644	.260
1.160	.642	.525204	.270
1.164	.640	.522000	.274
1.170	.636	.516604	.300
1.174	.632	.510244	.310
1.200	.630	.505100	.314
1.204	.626	.501744	.320
1.210	.624	.476620	.324
1.214	.620	.470400	.334
1.220	.616	.465304	.340
1.224	.614	.462220	.344
1.230	.612	.457144	.350
1.234	.610	.454100	.354
1.240	.604	.446020	.364
1.244	.602	.443004	.370
1.250	.600	.440000	.374
1.254	.576	.435004	.400

Reciprocal Table Look-Up Values (Continued)

B	A_0	A_0^2	$-2A_0$
1.260	.574	.432020	.404
1.264	.572	.427044	.410
1.270	.570	.424100	.414
1.274	.566	.421144	.420
1.300	.564	.416220	.424
1.304	.562	.413304	.430
1.310	.560	.410400	.434
1.314	.556	.405504	.440
1.320	.554	.402620	.444
1.324	.552	.377744	.450
1.330	.550	.375100	.454
1.334	.546	.372244	.460
1.340	.544	.367420	.464
1.344	.542	.364604	.470
1.350	.540	.362000	.474
1.354	.536	.357204	.500
1.360	.534	.354420	.504
1.364	.532	.351644	.510
1.370	.530	.347100	.514
1.374	.526	.344344	.520
1.400	.524	.341620	.524
1.404	.522	.337104	.530
1.410	.520	.334400	.534
1.414	.520	.334400	.534
1.420	.516	.331704	.540
1.424	.514	.327220	.544
1.430	.512	.324544	.550
1.434	.510	.322100	.554
1.440	.506	.317444	.560
1.444	.506	.317444	.560
1.450	.504	.315020	.564
1.454	.502	.312404	.570
1.460	.500	.310000	.574
1.464	.476	.305404	.600
1.470	.476	.305404	.600
1.474	.474	.303020	.604
1.500	.472	.300444	.610
1.504	.470	.276100	.614
1.510	.470	.276100	.614
1.514	.466	.273544	.620
1.520	.464	.271220	.624

Reciprocal Table Look-Up Values (Continued)

B	A_0	A_0^2	$-2A_0$
1.524	.462	.266704	.630
1.530	.462	.266704	.630
1.534	.460	.264400	.634
1.540	.456	.262104	.640
1.544	.456	.262104	.640
1.550	.454	.257620	.644
1.554	.452	.255344	.650
1.560	.452	.255344	.650
1.564	.450	.253100	.654
1.570	.446	.250644	.660
1.574	.446	.250644	.660
1.600	.444	.246420	.664
1.604	.442	.244204	.670
1.610	.442	.244204	.670
1.614	.440	.242000	.674
1.620	.436	.237604	.700
1.624	.436	.237604	.700
1.630	.434	.235420	.704
1.634	.434	.235420	.704
1.640	.432	.233244	.710
1.644	.430	.231100	.714
1.650	.430	.231100	.714
1.654	.426	.226744	.720
1.660	.426	.226744	.720
1.664	.424	.224620	.724
1.670	.422	.222504	.730
1.674	.422	.222504	.730
1.700	.420	.220400	.734
1.704	.420	.220400	.734
1.710	.416	.216304	.740
1.714	.416	.216304	.740
1.720	.414	.214220	.744
1.724	.412	.212144	.750
1.730	.412	.212144	.750
1.734	.410	.210100	.754
1.740	.410	.210100	.754
1.744	.406	.206044	.760
1.750	.406	.206044	.760
1.754	.404	.204020	.764
1.760	.404	.204020	.764
1.764	.402	.202004	.770
1.770	.402	.202004	.770
1.774	.400	.200000	.774

(RB) B_1 FANOUT SENT CP 2

		SENT						
		(RB1)	(RB2)	(RB3)				
		RECEIVED						
B_1 BITS	R TERMS	(RC1)	(RC1)	(RC1)				
2^{-0}	R21	I18						
2^{-1}	R22	I17						
2^{-2}	R23	I16						
2^{-3}	R21		I15					
2^{-4}	R22		I14					
2^{-5}	R23		I13					
2^{-6}	R21			I12				
2^{-7}	R22			I11				
2^{-8}	R23			I10				

(RB1) B_1 FANOUT SENT CP 1

		RECEIVED						
B_1, B_1 T5 BITS	R TERMS	(RB1)	(RB2)	(RB3)	(RJ1)	(RJ2)	(RI1)	
2^{-0}	R12	I59	I38	I33				
2^{-1}	R13	I60	I37	I32			I40	
2^{-2}	R14	I61	I36	I31			I41	
2^{-3}	R15		I35 I59	I30	I60			
2^{-4}	R16	I39	I34 I60	I29	I61			
2^{-5}	R17	I38	I33 I61	I28	I62			
2^{-6}	R18	I37	I32	I27 I59	I63			
2^{-7}	R19	I36	I31	I26 I60	I64			
2^{-8}	R20	I35	I30	I25 I61	I65			

(RB2) B_1 FANOUT SENT CP 1

		RECEIVED						
B_1 BITS	R TERMS	(RB1)	(RB2)	(RB3)	(RJ1)	(RJ2)		
2^{-9}	R12	I34	I29	I24	I66			
2^{-10}	R13	I33	I28	I23	I67			
2^{-11}	R14	I32	I27	I22	I68			
2^{-12}	R15	I31	I26	I21	I69			
2^{-13}	R16	I30	I25	I20	I70			
2^{-14}	R17	I29	I24			I60		
2^{-15}	R18	I28	I23			I61		
2^{-16}	R19	I27	I22			I62		
2^{-17}	R20	I26	I21			I63		

Hardware Trng.
A-4164B J.E.S.

CRAY Y-MP FLOATING RECIPROCAL B_1, B_2 FANOUT
(sheet 1 of 3)

(RB3) B_1 FANOUT SENT CP 1

B_1 BITS	R TERMS	RECEIVED						
		(RB1)	(RB2)	(RJ2)				
2^{-18}	R12	I25	I20	I64				
2^{-19}	R13	I24		I65				
2^{-20}	R14	I23		I66				
2^{-21}	R15	I22		I67				
2^{-22}	R16	I21		I68				
2^{-23}	R17	I20		I69				
2^{-24}	R18			I70				

(RD1) B_2 FANOUT SENT AFTER 8 CP DELAY

B_2 BITS	R TERMS	RECEIVED						
		(RG1)	(RH1)	(RJ1)	(RK1)			
2^{-25}	R18	I27	I20	I36				
2^{-26}	R19	I26		I35				
2^{-27}	R20	I25		I34	I39			
2^{-28}	R21	I24		I33	I38			
2^{-29}	R22	I23		I32	I37			
2^{-30}	R23	I22		I31	I36			

(RD2) B_2 FANOUT SENT AFTER 8 CP DELAY

B_2 BITS	R TERMS	RECEIVED						
		(RG1)	(RJ1)	(RK1)				
2^{-31}	R18	I21	I30	I35				
2^{-32}	R19	I20		I34				
2^{-33}	R20			I33				
2^{-34}	R21			I32				
2^{-35}	R22			I31				
2^{-36}	R23			I30				

Hardware Trng.
A-4164B J.E.S.

CRAY Y-MP FLOATING RECIPROCAL B_1, B_2 FANOUT
(sheet 2 of 3)

(RI1) B_2 FANOUT SENT AFTER 7 CP DELAY

B_2 BITS	(RI1) R TERMS	RECEIVED						
		(RH3)	(RH2)	(RI1)	(RK1)	(RG2)	(RJ2)	
2^{-1}	R21	I37	I31	I30	I56	I38	I44	
2^{-2}	R22	I36	I30	I29	I55	I37	I43	

(RJ1) B_2 FANOUT SENT AFTER 7 CP DELAY

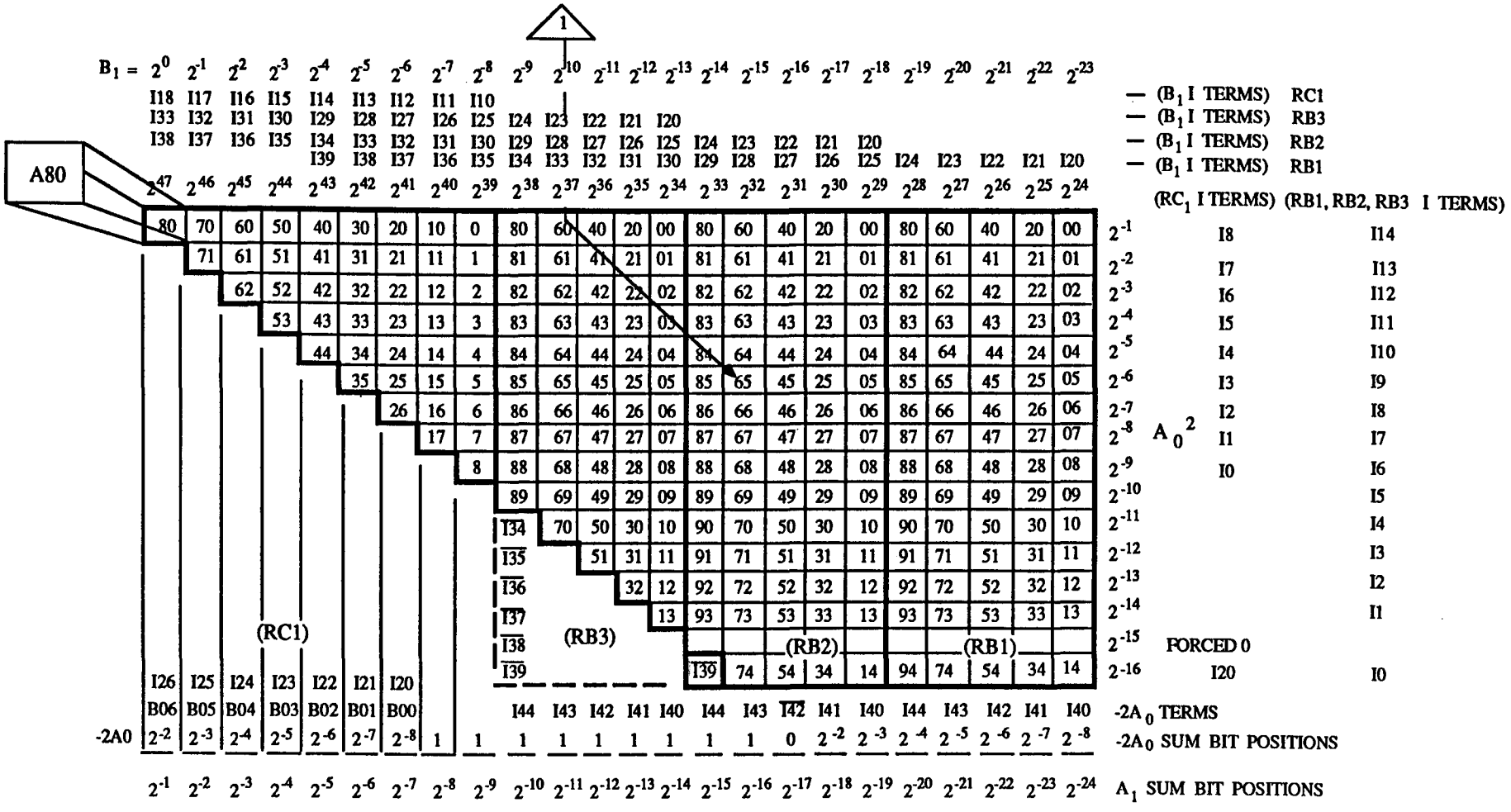
B_2 BITS	(RJ1) R TERMS	RECEIVED						
		(RH3)	(RH2)	(RI1)	(RK1)	(RG2)	(RJ2)	(RH1)
2^{-3}	R20	I35	I29	I28	I54	I36	I42	
2^{-4}	R21	I34	I28	I27	I53	I35	I41	
2^{-5}	R22	I33	I27	I26	I52	I34	I40	
2^{-6}	R23	I32	I26	I25	I51	I33	I39	
2^{-7}	R24	I31	I25	I24	I50	I32	I38	I38
2^{-8}	R25	I30	I24	I23		I31	I37	I37
2^{-9}	R26	I29	I23	I22		I30	I36	I36
2^{-10}	R27	I28	I22	I21		I29	I35	I35
2^{-11}	R28	I27	I21	I20		I28	I34 I56	I34
2^{-12}	R29	I26	I20			I27	I33 I55	I33
2^{-13}	R30	I25			I42	I26	I32 I54	I32

(RJ2) B_2 FANOUT SENT AFTER 7 CP DELAY

B_2 BITS	(RJ2) R TERMS	RECEIVED						
		(RH3)	(RH1)	(RJ2)	(RJ1)	(RG2)	(RG1)	(RK1)
2^{-14}	R20	I24	I31	I31 I53	I56	I25	I38	I41
2^{-15}	R21	I23	I30	I30 I52	I55	I24	I37	I40
2^{-16}	R22	I22	I29		I54 I45	I23	I36	
2^{-17}	R23	I21	I28		I53 I44	I22	I35	
2^{-18}	R23	I20	I27		I52 I43	I21	I34	
2^{-19}	R25		I26		I51 I42	I20	I33	
2^{-20}	R26		I25		I50 I41		I32	
2^{-21}	R27		I24		I40		I31	
2^{-22}	R28		I23		I39		I30	
2^{-23}	R29		I22		I38		I29	
2^{-24}	R30		I21		I37		I28	

Hardware Trng.
A-4164B J.E.S.

CRAY Y-MP FLOATING RECIPROCAL B_1, B_2 FANOUT
(sheet 3 of 3)

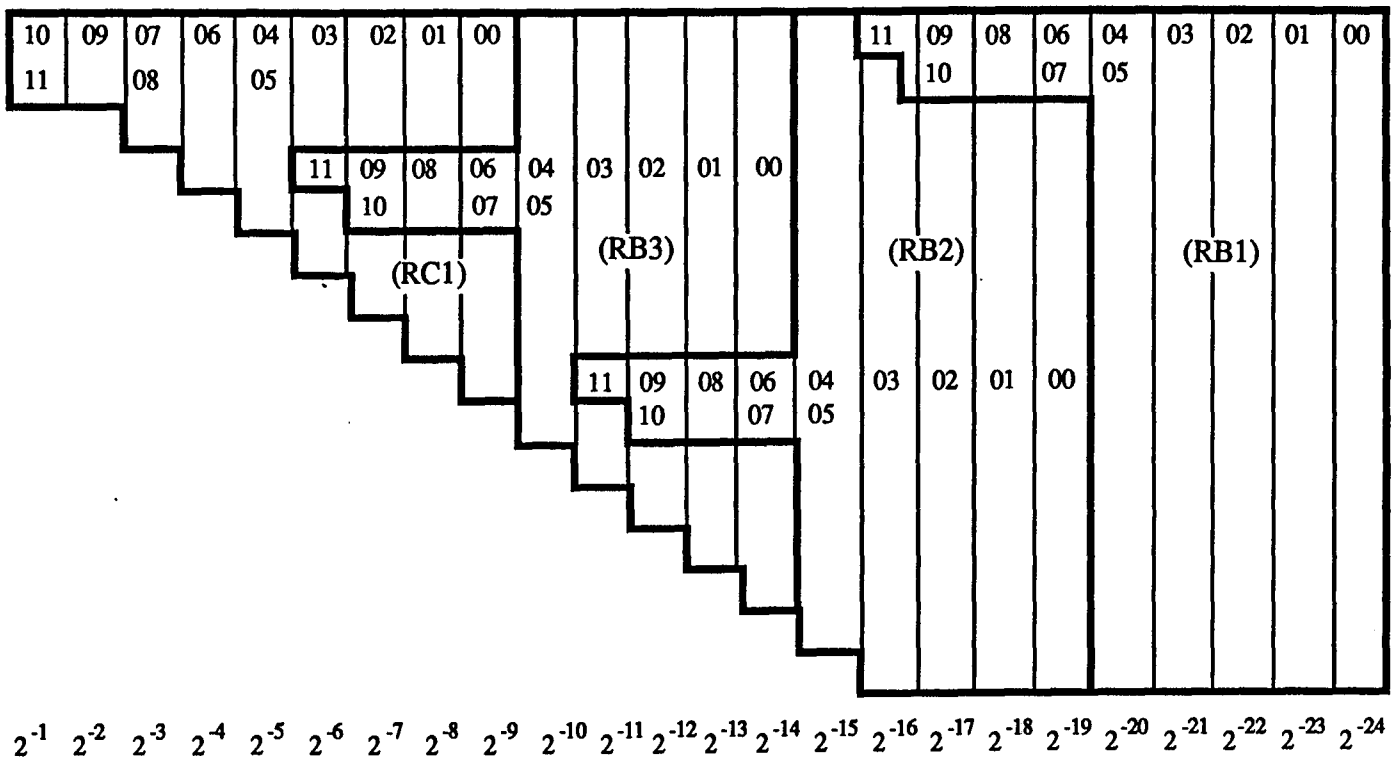


1 Diagonally form the Logical Products, vertically sum the pyramid.

Hardware Trng.
A-4173D J.E.S.

CRAY Y-MP A_1 PYRAMID LOGICAL PRODUCTS (RB1, RB2, RB3) AND (RC1) CP 1, CP 2

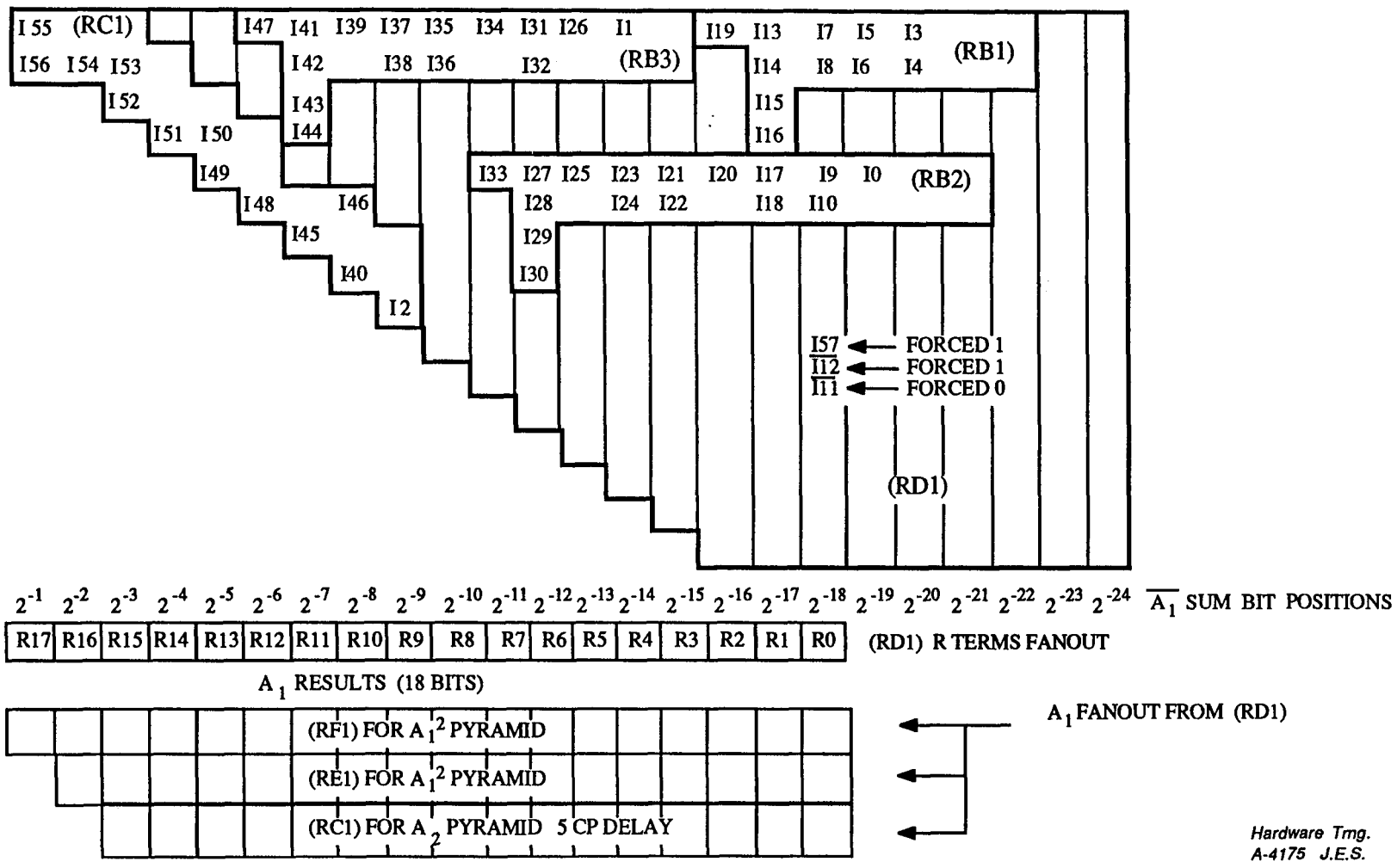
CRAY PROPRIETARY



Hardware Trng.
A-4174A J.E.S.

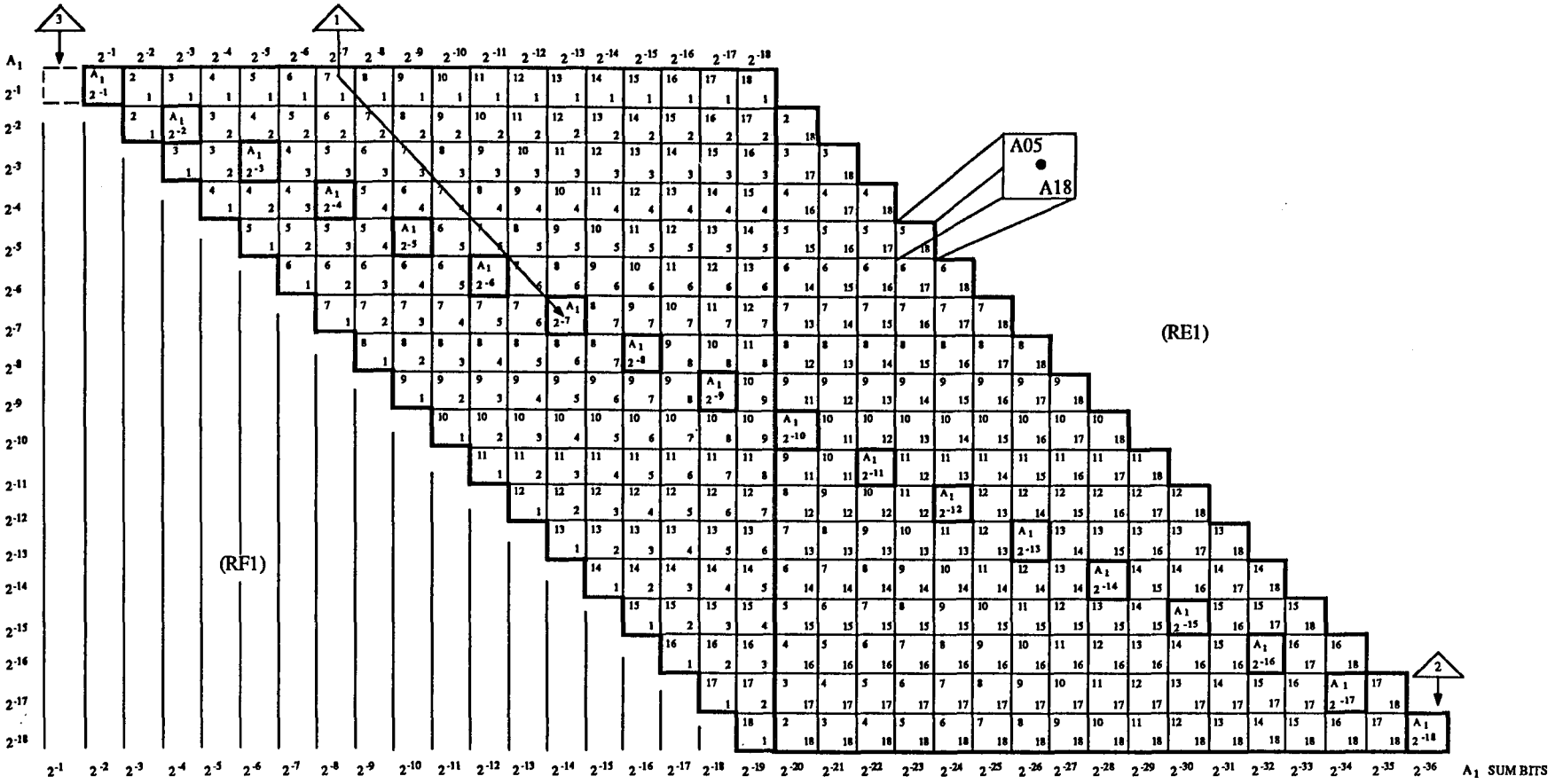
CRAY Y-MP A_1 PYRAMID PARTIAL SUMMATION R TERMS CP 4
(RB1, RB2, RB3) AND (RC1)

CRAY PROPRIETARY



A_1 FINAL SUMMATION (RD1) (R0 - R17) CP 5

CRAY PROPRIETARY



1 Diagonally form the Logical Products, vertically sum the pyramid.

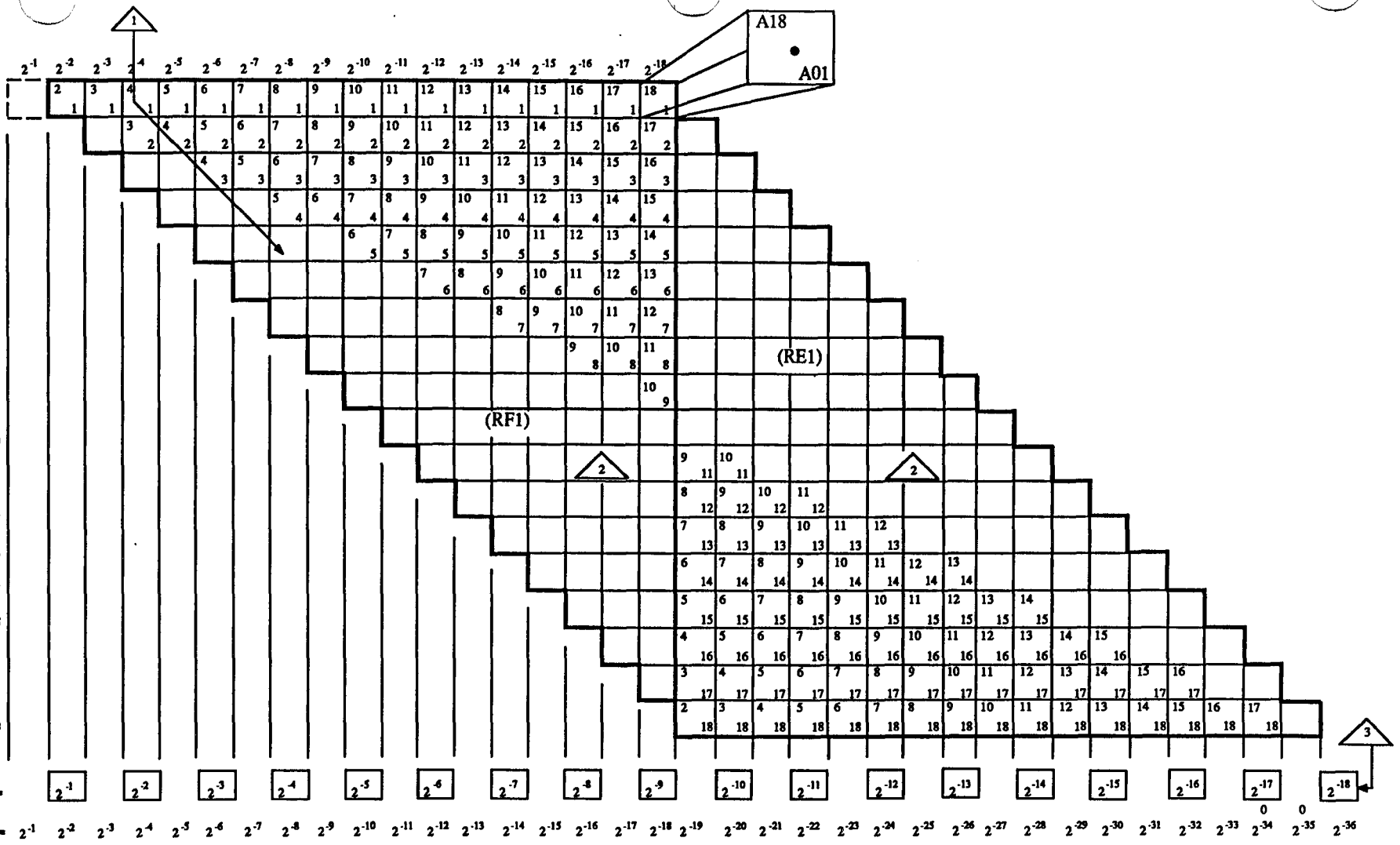
2 The square of A_1 causes the same operand to be multiplied by itself. Example: Bit $2^{-18} * 2^{-18} = 2^{-18}$

3 Use all 36 units of the results including the carry into the highest bit position.

Hardware Trng.
A-4169B J.E.S.

CRAY Y-MP A_1^2 FULL SUMMATION PYRAMID CP 6
(RF1) AND (RE1)

CRAY PROPRIETARY



1 Diagonally form the Logical Products, vertically sum the pyramid.

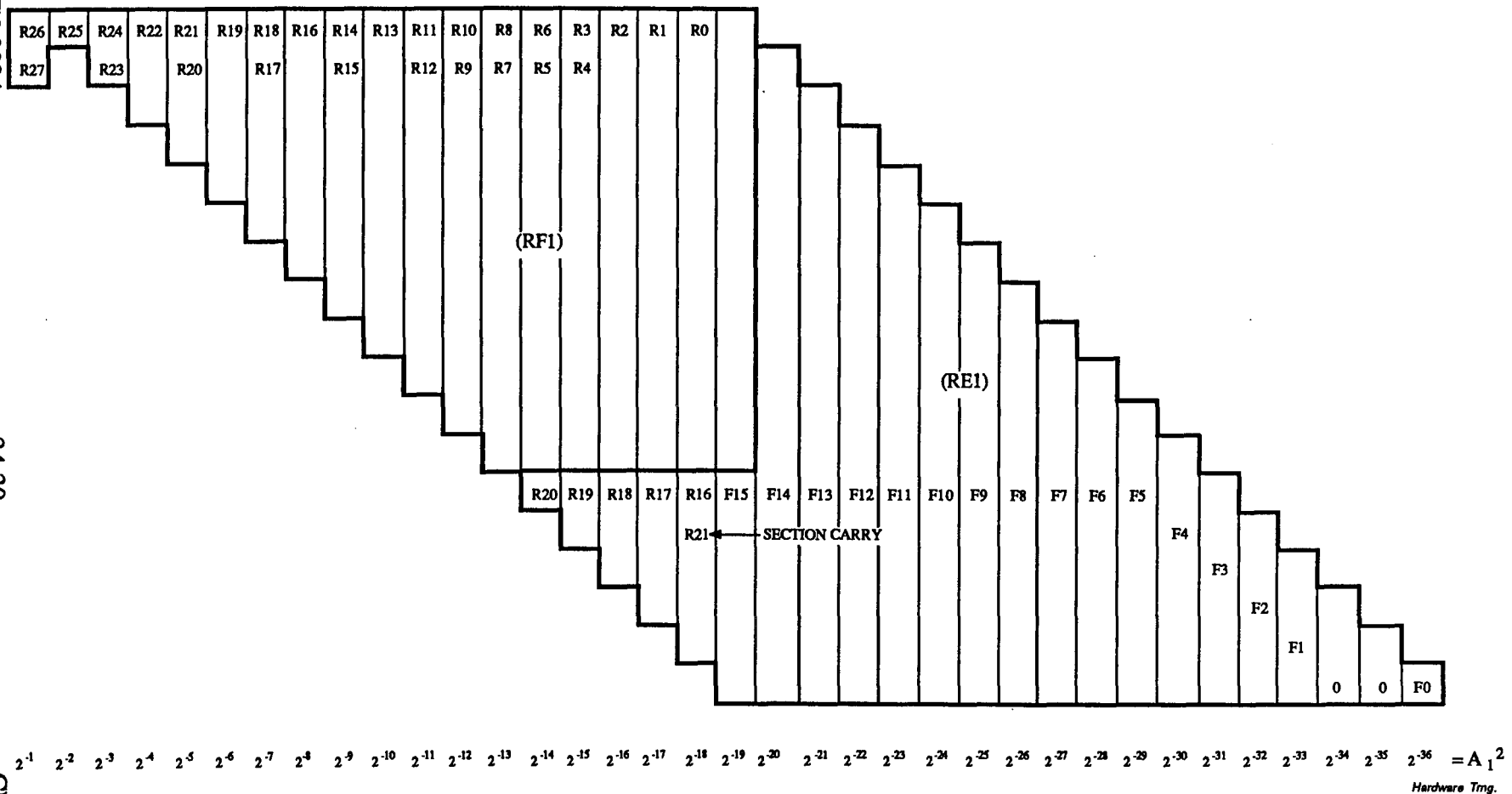
2 This part of the pyramid is not produced, however, when any one of the shown Logical Products are produced, they have a sum value of two.

3 A_1 is added into the summation right shifted one place, the summation adjusted to give the same results as a Full Summation.

Hardware Trng.
A-4170A J.E.S.

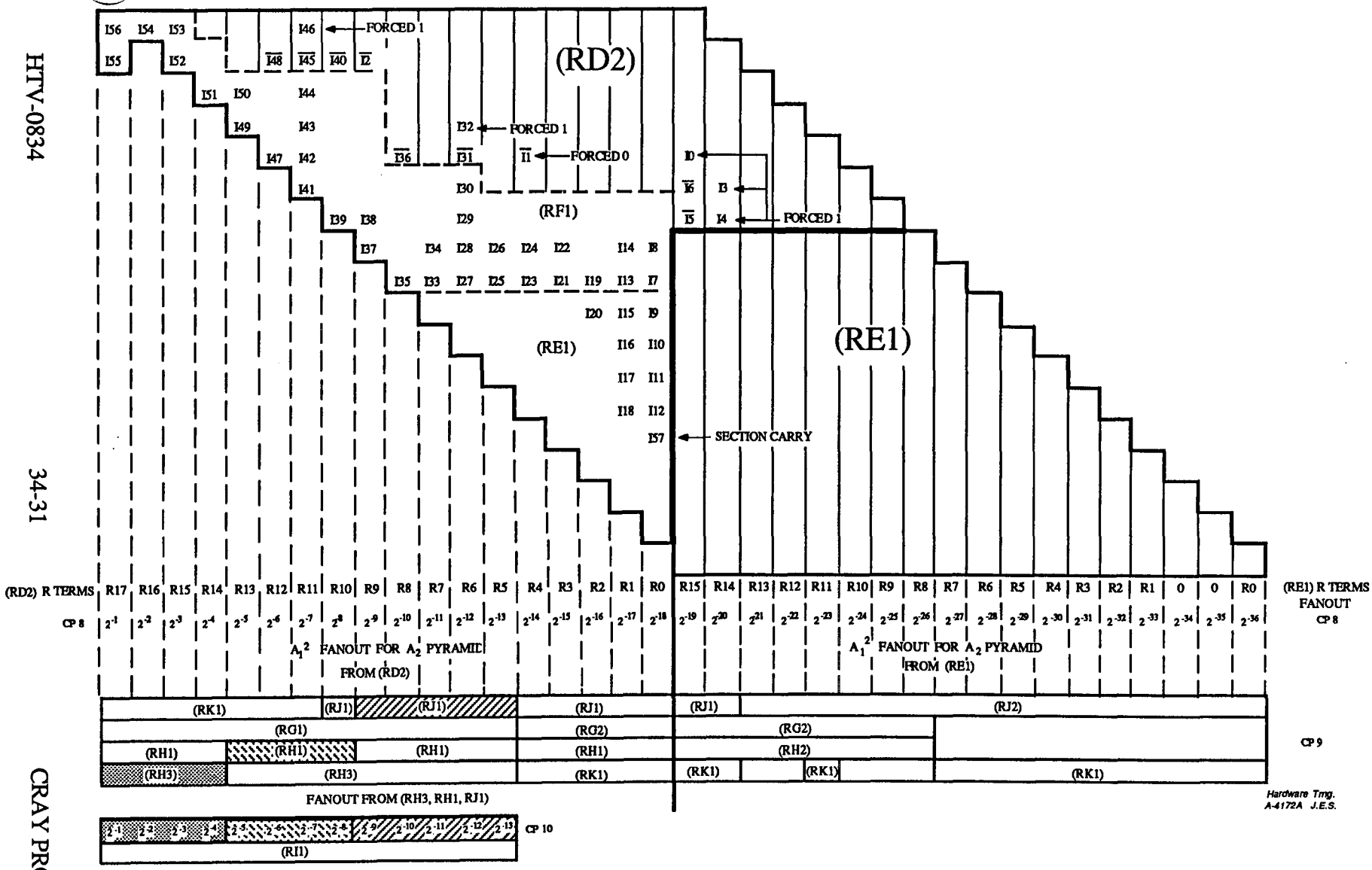
CRAY Y-MP A_1^2 ESSENTIAL SUMMATION PYRAMID CP 6 (RF1) AND (RE1)

CRAY PROPRIETARY



A_1^2 PARTIAL SUMMATION CP 7
(RF1) AND (RE1)

CRAY PROPRIETARY



A_1^2 FINAL SUMMATION CP 8 AND
A_1^2 FANOUT TO FORM A_2 PYRAMID
CP 9 (RD2) AND (RE1)

CRAY PROPRIETARY

(RD2) SENT A_1^2 FANOUT CP 8

A_1^2 BITS	(RD2) R TERMS	RECEIVED						
		(RJ1)	(RG2)	(RH2)	(RK1)	(RH3)	(RG1)	(RH1)
2^{-01}	R17				I26	I12	I12	I12
2^{-02}	R16				I25	I11	I11	I11
2^{-03}	R15				I24	I10	I10	I10
2^{-04}	R14				I23	I09	I09	I09
2^{-05}	R13				I22	I08	I08	I08
2^{-06}	R12				I21	I07	I07	I07
2^{-07}	R11				I20	I06	I06	I06
2^{-08}	R10	I12				I05	I05	I05
2^{-09}	R09	I11				I04	I04	I04
2^{-10}	R08	I10				I03	I03	I03
2^{-11}	R07	I09				I02	I02	I02
2^{-12}	R06	I08				I01	I01	I01
2^{-13}	R05	I07				I00	I00	I00
2^{-14}	R04	I06	I12	I12	I17			
2^{-15}	R03	I05	I11	I11	I16			
2^{-16}	R02	I04	I10	I10	I15			
2^{-17}	R01	I03	I09	I09	I14			
2^{-18}	R00	I02	I08	I08	I13			

(RE1) SENT A_1^2 FANOUT CP 8

A_1^2 BITS	(RE1) R TERMS	RECEIVED						
		(RJ1)	(RH2)	(RG2)	(RJ2)	(RK1)		
2^{-19}	R15	I01	I07	I07		I12		
2^{-20}	R14	I00 I20	I06	I06		I11		
2^{-21}	R13		I05	I05	I02			
2^{-22}	R12		I04	I04	I01			
2^{-23}	R11		I03	I03	I00	I10		
2^{-24}	R10		I02	I02	I12			
2^{-25}	R09		I01	I01	I11			
2^{-26}	R08		I00	I00	I10			
2^{-27}	R07				I09	I06		
2^{-28}	R06				I08	I05		
2^{-29}	R05				I07	I04		
2^{-30}	R04				I06	I03		
2^{-31}	R03				I05	I02		
2^{-32}	R02				I04	I01		
2^{-33}	R01				I03	I00		
2^{-36}	R00				I20			

Hardware Trng.
A-4165A J.E.S.

CRAY Y-MP FLOATING RECIPROCAL A_1^2 FANOUT
(sheet 1 of 2)

(RH3) SENT A_1^2 FANOUT CP 9

A_1^2 BITS	(RH3) R TERMS	RECEIVED						
		(RI1)						
2^{-1}	R24	I12						
2^{-2}	R23	I11						
2^{-3}	R22	I10						
2^{-4}	R21	I9						

(RH1) SENT A_1^2 FANOUT CP 9

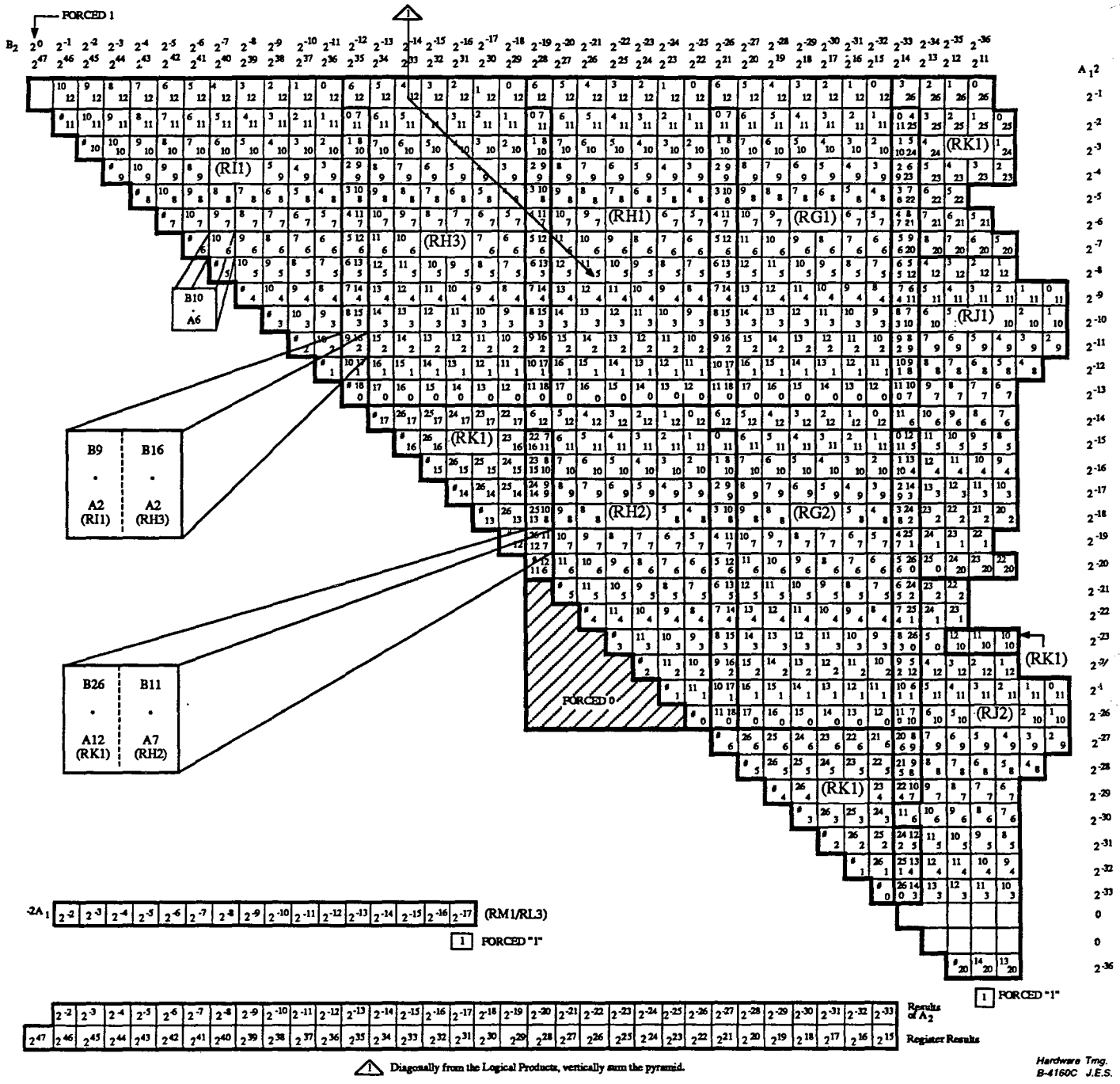
A_1^2 BITS	(RH1) R TERMS	RECEIVED						
		(RI1)						
2^{-5}	R24	I8						
2^{-6}	R23	I7						
2^{-7}	R22	I6						
2^{-8}	R21	I5						

(RJ1) A_1^2 FANOUT CP 9

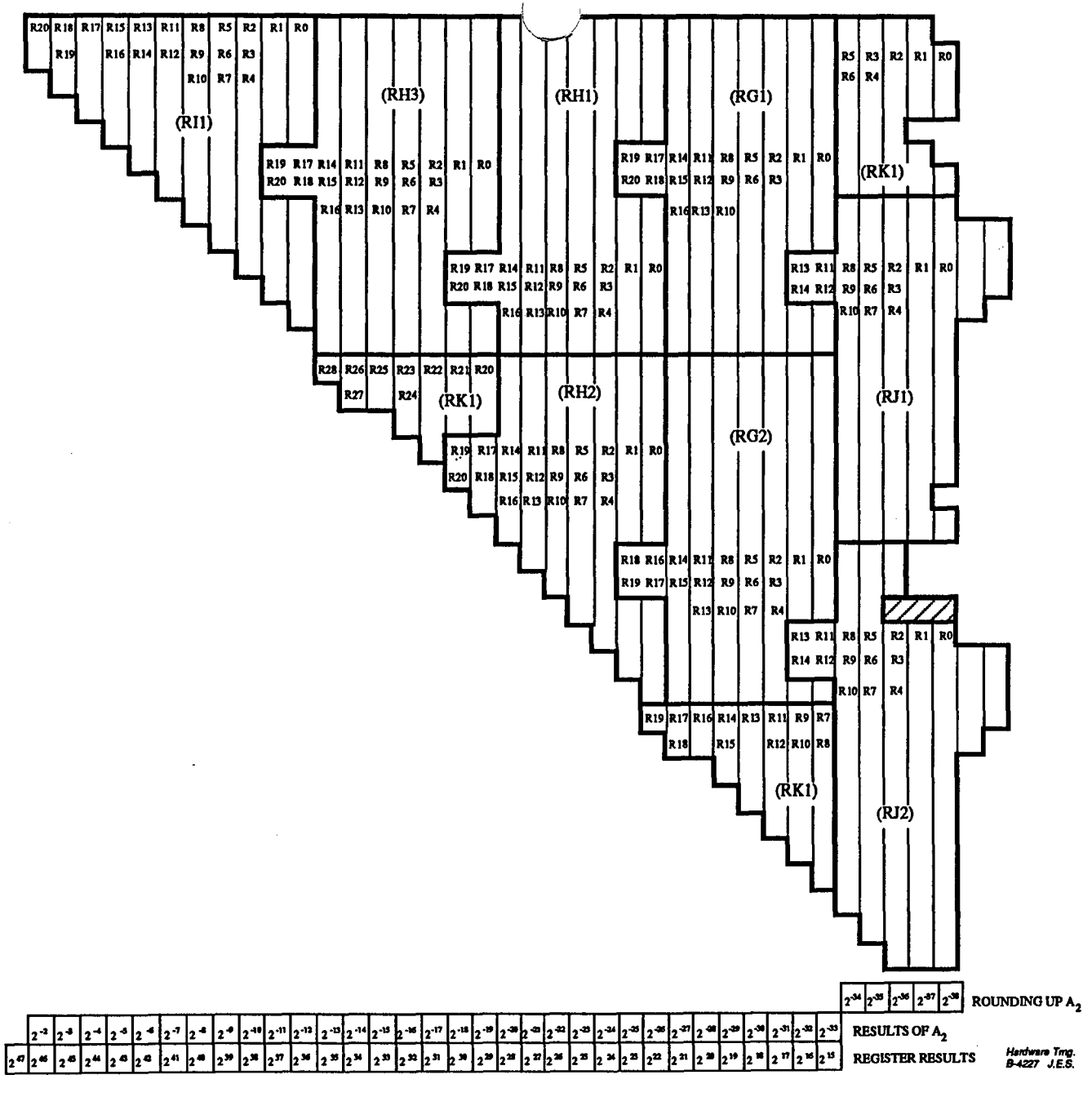
A_1^2 BITS	(RJ1) R TERMS	RECEIVED						
		(RI1)						
2^{-9}	R19	I4						
2^{-10}	R18	I3						
2^{-11}	R17	I2						
2^{-12}	R16	I1						
2^{-13}	R15	I0						

Hardware Trng.
A-4165 J.E.S.

CRAY Y-MP FLOATING RECIPROCAL A_1^2 FANOUT (SHEET 2 OF 2)

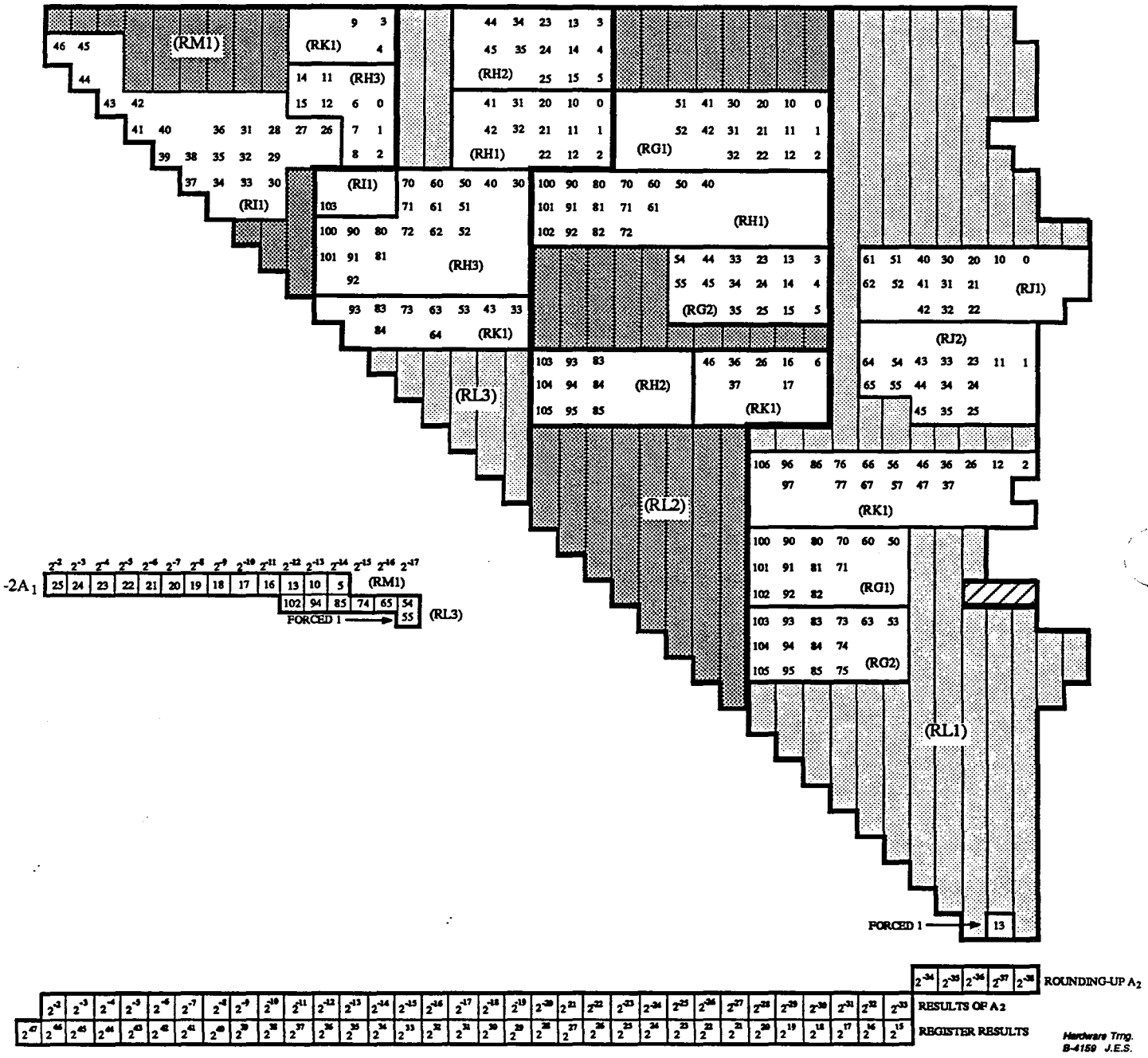


CRAY Y-MP FLOATING RECIPROCAL A_2 PYRAMID LOGICAL PRODUCTS CP 9

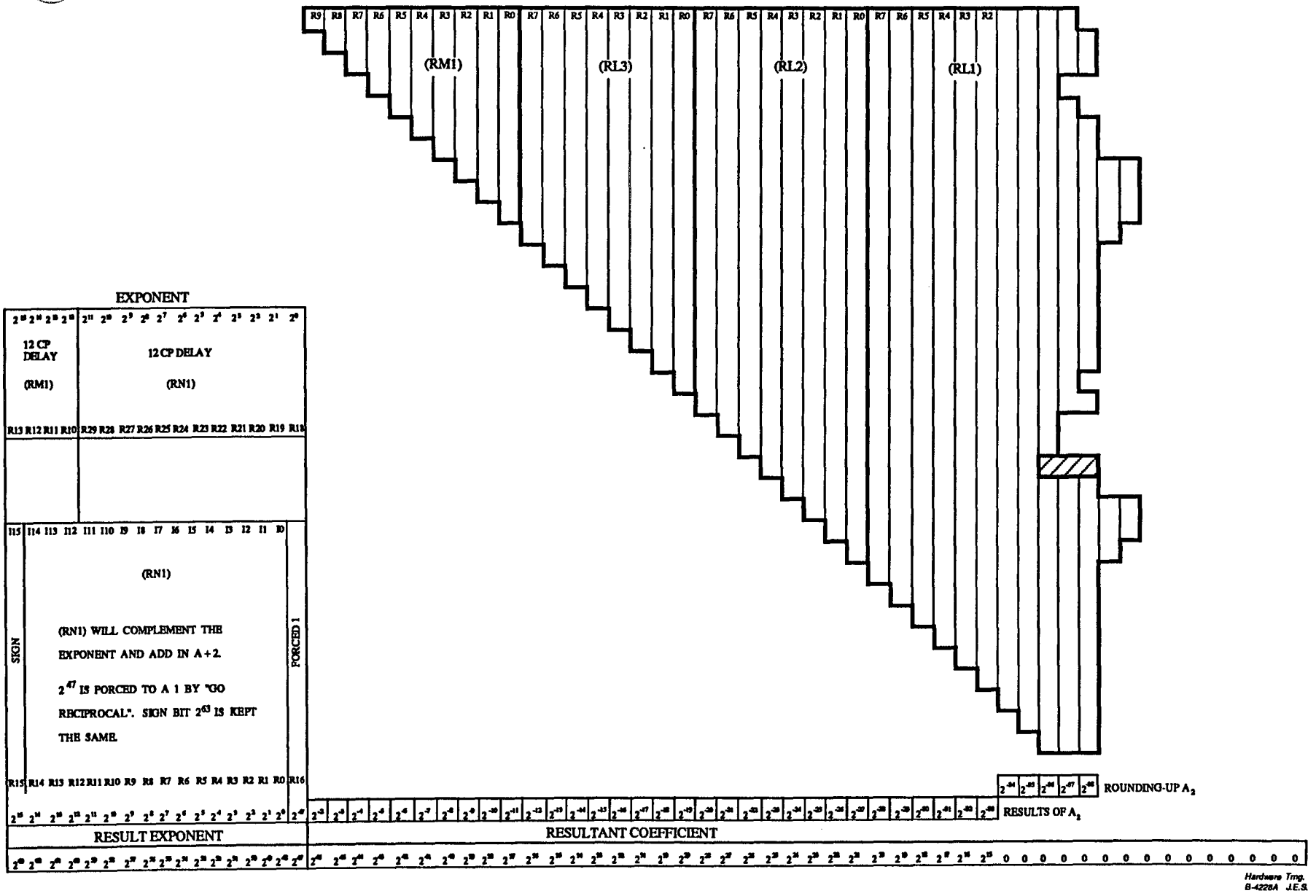


A2 PYRAMID PARTIAL SUMMATION
R TERMS CP 10

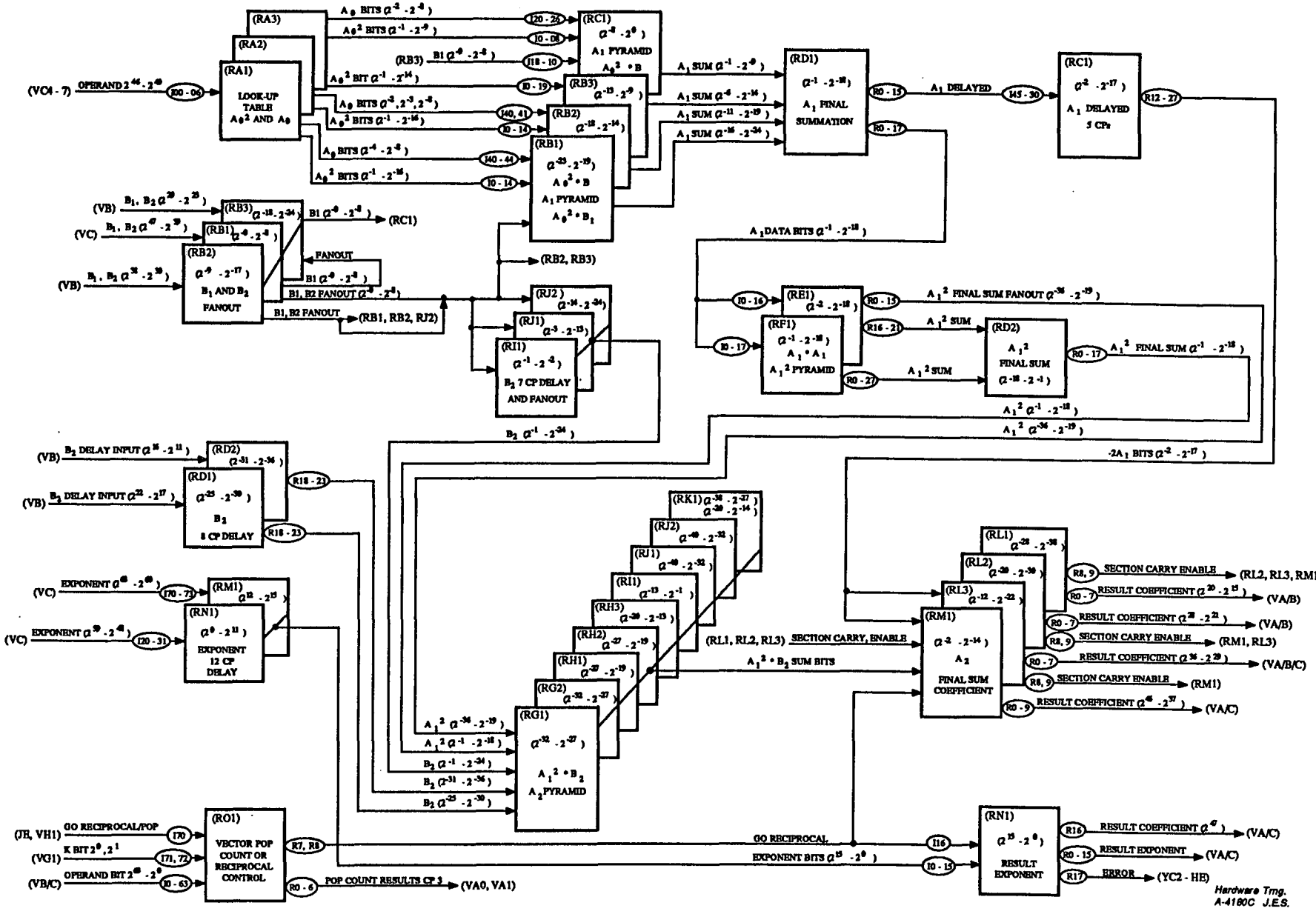
CRAY PROPRIETARY



A_2 PYRAMID FINAL SUMMATION
I TERMS CP 11



CRAY Y-MP FLOATING-POINT RECIPROCAL FINAL RESULT CP 14



CRAY Y-MP FLOATING RECIPROCAL BLOCK DIAGRAM

FLOATING-POINT MULTIPLY FUNCTIONAL UNIT

The Floating-point Multiply functional unit is shared by the Scalar and Vector registers. Two floating-point operands are sent to the Multiply functional unit by either the Scalar or Vector registers. The signs of the two operands are exclusive ORed, the exponents are added together and the two 48-bit coefficients are multiplied together. Multiplying two 48-bit numbers together will produce a 96-bit result. Since our result register, either a Scalar or Vector register, can only hold 48 bits in the coefficient, only the upper 48 bits of the 96-bit result are kept. The lower 48 bits are lost; in fact, they are never generated.

The Floating-point Multiply functional unit can also be used to multiply two integer numbers together. To multiply two 24-bit integer numbers together simply set each operand's exponent equal to zero, and place each integer value left-justified in the operand's coefficient area. A complete multiply will be performed with no truncate as long as the total number of bits in the two operands does not exceed 48-bit positions.

The Floating-point Multiply functional unit can also be utilized to generate a third iteration in conjunction with the Reciprocal functional unit. Generating the third iteration will create a full precision coefficient, utilizing all 48 bits of the coefficient. The full precision reciprocal number can then be multiplied by the multiplier to complete a divide. If full precision is not needed, then there is no need to generate a third iteration; instead, the two iterations that are generated in the Reciprocal functional unit are multiplied by the multiplier using a special multiply instruction which will add in 2 round bits and truncate the lower 19 bits of the coefficient.

The Floating-point Multiply functional unit has the same Range Error conditions as the Floating-point add. If an overflow condition exists, the floating-point number has gone beyond the limits of the machine. When an overflow condition occurs the result register will receive the calculated coefficient with an exponent forced to 60000_8. An overflow condition also causes a flag to be sent to the Exchange Package. An underflow condition exists when the resultant exponent is equal to or less than 17777_8. When an underflow condition exists, both the final exponent and the coefficient will be forced to zero, but no flag will be sent to the Exchange Package.

The Floating-point Multiply functional unit performs the 064ijk - 067ijk instructions for the Scalar registers, and the 160ijk - 167ijk instructions for the Vector registers. Because the multiply unit is shared by both the Scalar and Vector registers, a functional unit reservation must be checked before allowing one of the above instructions to issue.

The 064ijk instruction, which is the Scalar equivalent of the 160ijk - 161ijk instruction for the Vector registers, performs a Floating-point Multiply of two Scalar registers. An integer multiply can also be performed, as mentioned earlier.

The 065ijk instruction, which is the equivalent of the 162ijk - 163ijk instruction for Vector registers, is used in conjunction with the Reciprocal functional unit to complete a divide sequence. In other words, an 065ijk instruction would be issued after an 070 instruction. The 065ijk instruction will add 2 bits into the final summation, in bit positions 2^{17} and 2^{16}. These 2 bits, which are generated from Constant C, are called strong round bits, because they have a strong affect on the answer. When the final summation is completed the 065ijk instruction also causes the lower 19 bits to be truncated; the control term to do this is called Strong Round Truncate.

Hardware Trng.
YM01/06A J.E.S.

The 066ijk instruction, which is the equivalent of the 164ijk - 165ijk instruction for the Vector register, is used only after the third iteration has been completed within the Floating-point Multiply unit. The 066ijk instruction generates two weak round bits from Constant A. These 2 bits are called weak round bits because they are added into the lower portion of the summation, and hence have only a weak effect on the final summation.

The 067ijk instruction, which is the equivalent of the 167ijk instruction for the Vector registers, is used to partially form the third iteration as follows. The third iteration is equal to $A_3 = 2A_2 - A_2^2B$. The 067ijk instruction solves for $(-2 + A_2 * B)$, by first multiplying A_2 times B, and then adding (-2) to the product. The (-2) addition is accomplished by adding "1" to each bit position $(2^{46} - 2^0)$ during the summation of A_2*B. These "1" bits are comprised of effective 48 "1" bits and are called "-1 constants". They are generated from the control terms constant B and C. The 067ijk instruction also generates a control term called Complement which toggles its final result so as to convert $-A_3 = -2 + A_2*B$ to $A_3 = 2 - A_2*B$. At this point the 064ijk instruction is used to complete the third iteration by multiplying A_2 times the result of the 067ijk instruction. In other words, $A_2 * (2-A_2 * B) = 2A_2 - A_2^2B$. In conclusion, the 067ijk instruction, along with the 064ijk instruction generate the third iteration equation $A_3 = 2A_2 - A_2^2B$.

In Y-mode the 166ijk instruction uses the Floating-point Multiply unit to create a 32-bit address multiply of (Sj) and (Vk) to (Vi). The products can be used in Scatter-Gather memory operations, where (Sj) is the constant. The operands must be left-shifted before the 166ijk instruction is issued. The (Sj) operand must be shifted left 31_{10} places, leaving the operand in bit position $(2^{62} - 2^{31})$ and bit positions $(2^{30} - 2^0)$ must be equal to zero to get valid results. The (Vk) operand must also be left shifted 16_{10} places leaving the operands in bit position $(2^{47} - 2^{16})$, again bit positions $(2^{15} - 2^0)$ must be zero filled before the multiply can take place. Only the lower 32 bits of the integer multiply are gated to (Vi), the upper 32 bits are zeroed prior to leaving the Floating-point Multiply unit.

In X-mode the 166ijk instruction becomes 2 minus the product of (Sj) times (Vk).

CRAY Y-MP FLOATING-POINT MULTIPLY OPTIONS USED

OPTION TYPE	NUMBER USED	DESCRIPTION
(MA)	3	First Level Multiply
(MB)	5	First Level Multiply
(MC)	4	First Level Multiply
(MD)	1	First Level Multiply
(MF)	1	First Level Multiply
(MG)	8	Second Level Multiply
(MH)	1	Final Sum Bits 2^{95} - 2^{83}
(MI)	3	Final Sum ....Bits 2^{82} - 2^{71}, 2^{70} - 2^{59}, 2^{58} - 2^{47}
(MJ)	1	Multiply Control and Exponent Sum
(MP)	2	First Level Multiply
(MQ)	1	First Level Multiply
(MR)	4	Operand Fanout

34 Total Options

Hardware Trng.
YM01/08 J.E.S.

CRAY Y-MP FLOATING-POINT MULTIPLY FUNCTIONAL UNIT INSTRUCTIONS

SCALAR INSTRUCTIONS

064ijk	Scalar floating product of (Sj) times (Sk) to (Si)
065ijk	Scalar floating product, half precision, (Sj) times (Sk) to (Si)
066ijk	Scalar floating product, full precision, (Sj) times (Sk) to (Si)
067ijk	Scalar floating product, 2 minus the product of (Sj) times (Sk) to Si

CAL FORMAT

Si	Sj*FSk	Scalar floating product
Si	Sj*HSk	Half precision
Si	Sj*RSk	Full precision
Si	Sj*ISk	Iteration, 2 - Sj*Sk

YM01/13

CRAY Y-MP FLOATING-POINT MULTIPLY FUNCTIONAL UNIT INSTRUCTIONS

VECTOR INSTRUCTIONS

160ijk	Vector floating product (Sj) times (Vk elements) to Vi
161ijk	Vector floating product (Vj elements) times (Vk elements) to Vi
162ijk	Half precision, (Sj) times (Vk elements) to Vi
163ijk	Half precision, (Vj elements) times (Vk elements) to Vi
164ijk	Full precision, (Sj) times (Vk elements) to Vi
165ijk	Full precision, (Vj elements) times (Vk elements) to Vi
166ijk	Two minus the floating-point products of (Sj) and (Vk) to Vi (X-mode)
166ijk	32-bit integer products of (Sj) and (Vk) to Vi (Y-mode)
167ijk	Iteration, 2 - (Vj elements) times (Vk elements) toVi

CAL FORMAT

Vi	Sj*FVk	Vector, scalar product
Vi	Vj *FVk	Vector, vector product
Vi	Sj*HVk	Half precision, vector scalar
Vi	Vj*HVk	Half precision, vector vector
Vi	Sj*RVk	Full precision, vector scalar
Vi	Vj*RVk	Full precision, vector vector
Vi	Sj*Vk	32-bit address multiply, scalar vector
Vi	Vj*IVk	Iteration, 2 minus vector vector

YM01/14

DIVIDE SEQUENCE

S6 = S1/S2

Accurate To 29 Bits:

#1	070320	S3 = 1/S2
#2	065613	S6 = S1 * FS3

Accurate To 48 Bits:

		S6 = S1/S2
#1	070320	S3 = 1/S2
#2	067432	S4 = (2 - [S3*S2])
#3	064543	S5 = S4*S3
#4	066651	S6 = S5*S1

#1	$A_1 = 2A_0 - A_0^2B$	First Iteration
	$A_2 = 2A_1 - A_1^2B$	Second Iteration
#2	$S4 = (2 - (A_2*B))$	Third Iteration
#3	$A_3 = A_2(2 - (A_2*B))$	
	or	
	$A3 = 2A_2 - A_2^2B$	
#4	$S6 = A_3*S1$	Third Iteration * S1

YM01/15

FLOATING-POINT MULTIPLY (OPTIONS INVOLVED)

(MA01, MA08, MA13) Options

The three (MA) options perform two functions: one is to multiply a j operand times a k operand, and the second is to start the summation of bit positions (2^{47} - 2^{40}) of the pyramid; the partial summation is sent to the (MG) option for further summation. The (MA) option also assists in the fanout of the j operand. The summation is performed by using a 3-bit adder sum cell m285 and a 3-bit adder carry cell m286; some macro equations use the 3-bit adder sum/carry cell m281. The (MA01) assists in the fanout of the j operand to the (MF07) option.

(MB02, MB03, MB09, MB10, MB14) Options

The five (MB) options perform the same two functions as the (MA) options: multiplication and partial summation. The (MB) options perform a partial summation on bits (2^{61} - 2^{48}) of the pyramid. The (MB10, MB14) options assist in the fanout of the k operand to the (MF07) option.

(MC04, MC05, MC06, MC11) Options

The four (MC) options perform the same two functions as the (MB) options: multiplication and partial summation. The (MC) option is the same as the (MB) option except for the j and k inputs and the (a, b) Boolean terms. The (MC) options perform a partial summation on bits (2^{82} - 2^{62}) of the pyramid. The (MC06) also assists in the fanout of the j operand to the (MF07) option.

(MD17) Option

The (MD) option performs the same two functions as the (MA) options: multiplication and summation. The (MD) option performs a partial summation on bits (2^{55} - 2^{40}) of the pyramid.

(MF07) Option

The (MF) option performs the same two functions as the (MA) options: multiplication and partial summation. The (MF) also decodes the mode bits and assists the (MJ) in generating Constant B to the (MG) options.

Hardware Trng.
YM01/09 J.E.S.

(MG20 - MG27) Options

The (MG) options perform the second level summation, or continues to perform 3-bit adds on the pyramid from where the (MA, MB, MC, MD, MF, MP and MQ) options left off. The (MG) options add in the extra bits that get generated from either Constant A, Constant B, or Constant C. The (MG) options continue to sum the pyramid from bit positions (2^{82} - 2^{40}). The (MG) options pass the partial summation to the (MI) option for the final summation.

(MH28) Option

The (MH) option performs the same functions as the (MG) options, however, its summation proceeds from a series of 3-bit adds through to a series of 2-bit look-ahead adds on its way to the final summation. The (MH) option sums bit positions (2^{94} - 2^{80}) of the multiply pyramid, which are output to the Scalar or Vector registers as bits (2^{47} - 2^{36}). The (MH28) option also checks to see if bit (2^{95}) of the summation is set. If (2^{95}) is not set by a propagation from carries and enables, then the final result on the (MIs) is left shifted by one place. The left shift of one place ensures that the resultant coefficient will be Normalized, which is required if Normalized operands were originally multiplied together. During an integer multiply the left shift is prevented from happening. When performing a 32-bit Address Multiply the (MH28) option zeroes out coefficient bit (2^{47} - 2^{36}) to the result register.

(MI30, MI31, MI32) Options

The (MI) options perform the final summation of the multiply pyramid. The summation up to this point has been reduced to 2 bits in a row, so a 2-bit look-ahead add is performed. The three (MI) options each handle 12 bits of the coefficient and 5 bits of the exponent. The (MI) and (MH) options intercommunicate with sum bit enables and carries to see if each one's sum bit (2^{95}) is set or not. If (2^{95}) is not set, the (exponent -1) is generated on the (MI) options, and the coefficient is shifted to the left one place. During an integer multiply the left shift is prevented from happening. The (MI) also receives the original Go Multiply signal and the mode bits, which are fanned out to the (MJ) and (MF) options. When performing a 32-bit Address Multiply the (MI32) option zeroes out coefficient bits (2^{47} - 2^{32}) to the result register, results bit position (2^{63} - 2^{48}) are zeroed out by all three (MI) options seeing each (MI) handle 5 bits of the exponent.

(MJ18) Option

The (MJ) option is the control option for the multiplication and exponent summation. The Go Multiply signal, together with the mode bits, allows the (MJ) to calculate what type of multiply instruction was issued, and then to generate the appropriate control terms for that instruction.

Following is a list of some of the control terms:

Constant A - Associated with the 066ijk or 164ijk - 165ijk instructions, Constant A will add into the summation two weak round bits.

Constant B - Associated with the 067ijk or 167ijk instruction, Constant B causes 48 "1" bits to be added into the summation.

Constant C - Associated with the 065ijk - 067ijk or 164ijk - 167ijk instructions, Constant C generates two strong round bits which are added into the summation.

Complement - Generated from the 067ijk or 167ijk instruction, Complement causes the final result to be toggled.

Underflow - Underflow causes the final exponent and coefficient to be zeroed.

Integer - Integer multiply occurs when both exponents are equal to zero and prevents a left shift of one from occurring on the (MI) and (MJ) options.

Strong Round Truncate - Associated with the 065ijk or 162ijk - 163ijk instructions, this signal causes the lower 19 bits of the coefficient to be truncated.

Borrow Enable - This signal is generated whenever one of the 5-bit groups ($2^{52} - 2^{48}$), ($2^{57} - 2^{53}$), or ($2^{62} - 2^{58}$) of the exponent contains all zeroes. This is used by the (MI) options to generate the exponent -1, in the case when the resultant coefficient is left shifted by one.

Range Error - Range Error is set to a one when an overflow condition has occurred. The (MJ) checks for an overflow condition and sends an exponent of 60000_8 to the (MI) options. The (MIs) also output the calculated coefficient.

Go Address Multiply - Generated from the 166ijk instruction which allows result bits ($2^{63} - 2^{32}$) to be zeroed on the output. Go Address Multiply allows the Floating-point Multiply unit to perform a 32-bit Address Multiply on J operand bit ($2^{62} - 2^{31}$) and K operand bit ($2^{16} - 2^{47}$).

Underflow to (MH28) - Causes the (MH) option to zero out coefficient bit ($2^{47} - 2^{36}$) during a 32-bit Address Multiply.

The (MJ) does an exclusive-or of the sign bits of the two operands to obtain the final sign bit (2^{63}), which is then sent to either the Scalar or Vector registers.

The summed exponent is passed to the (MI) options, where either the exponent or exponent -1 is sent on to the Scalar or Vector registers, depending on whether sum bit (2^{95}) is set or not.

(MP40, MP41) Options

The two (MP) options perform the same two functions as the (MA) option: multiplication and partial summation. The (MP40) option performs a partial summation on bits (2^{68} - 2^{55}) of the pyramid, while the (MP41) performs a partial summation on bits (2^{81} - 2^{69}) of the pyramid.

(MQ42) Option

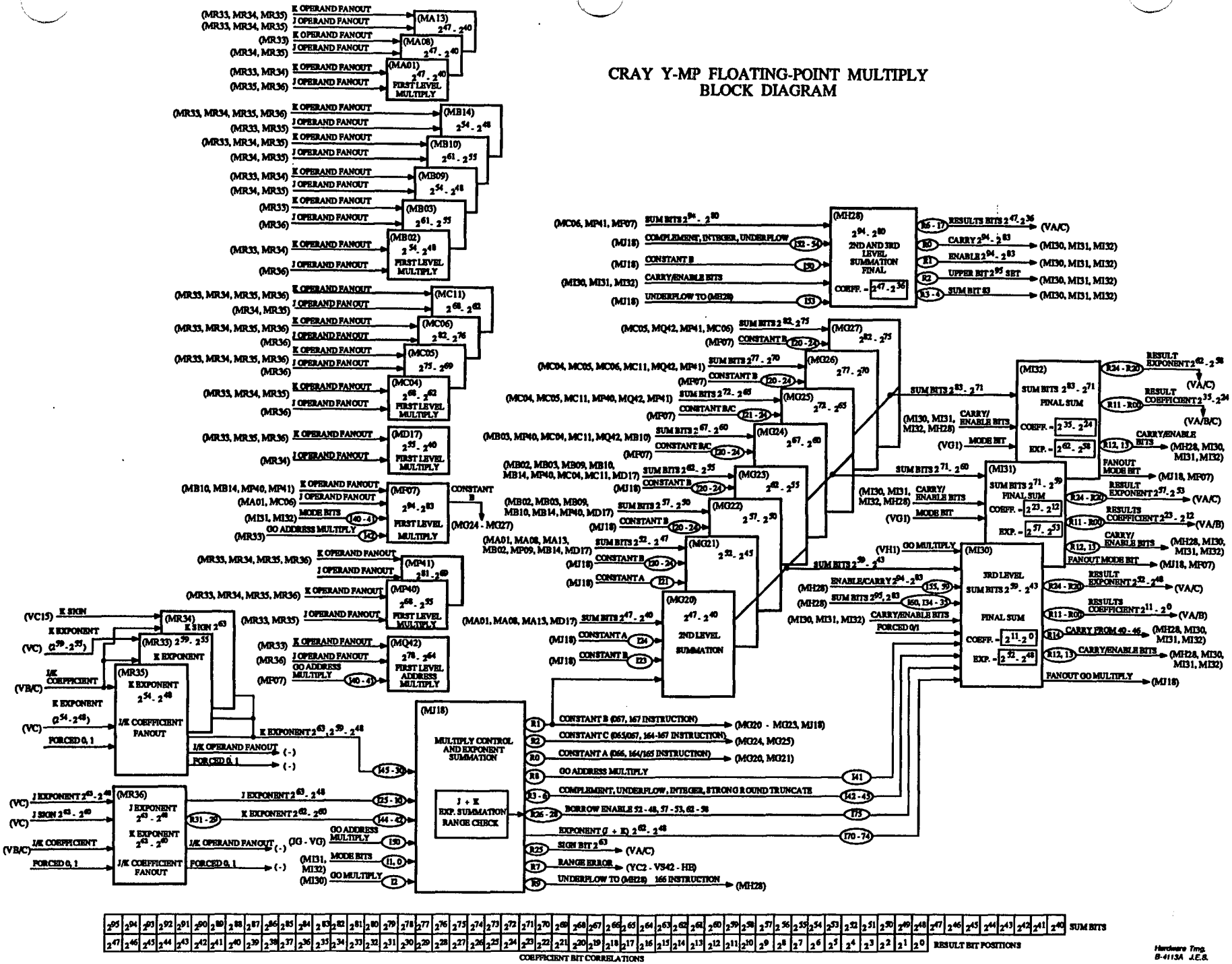
The (MQ42) option was added to the Floating-point Multiply unit to allow a 32-bit Address Multiply. The products can be used in Scatter-Gather memory operations, where (Sj) is the constant. The J operand must be left shifted 31_{10} places and the K operand left shifted 16_{10} places in order to utilize this option.

(MR33 - MR36) Options

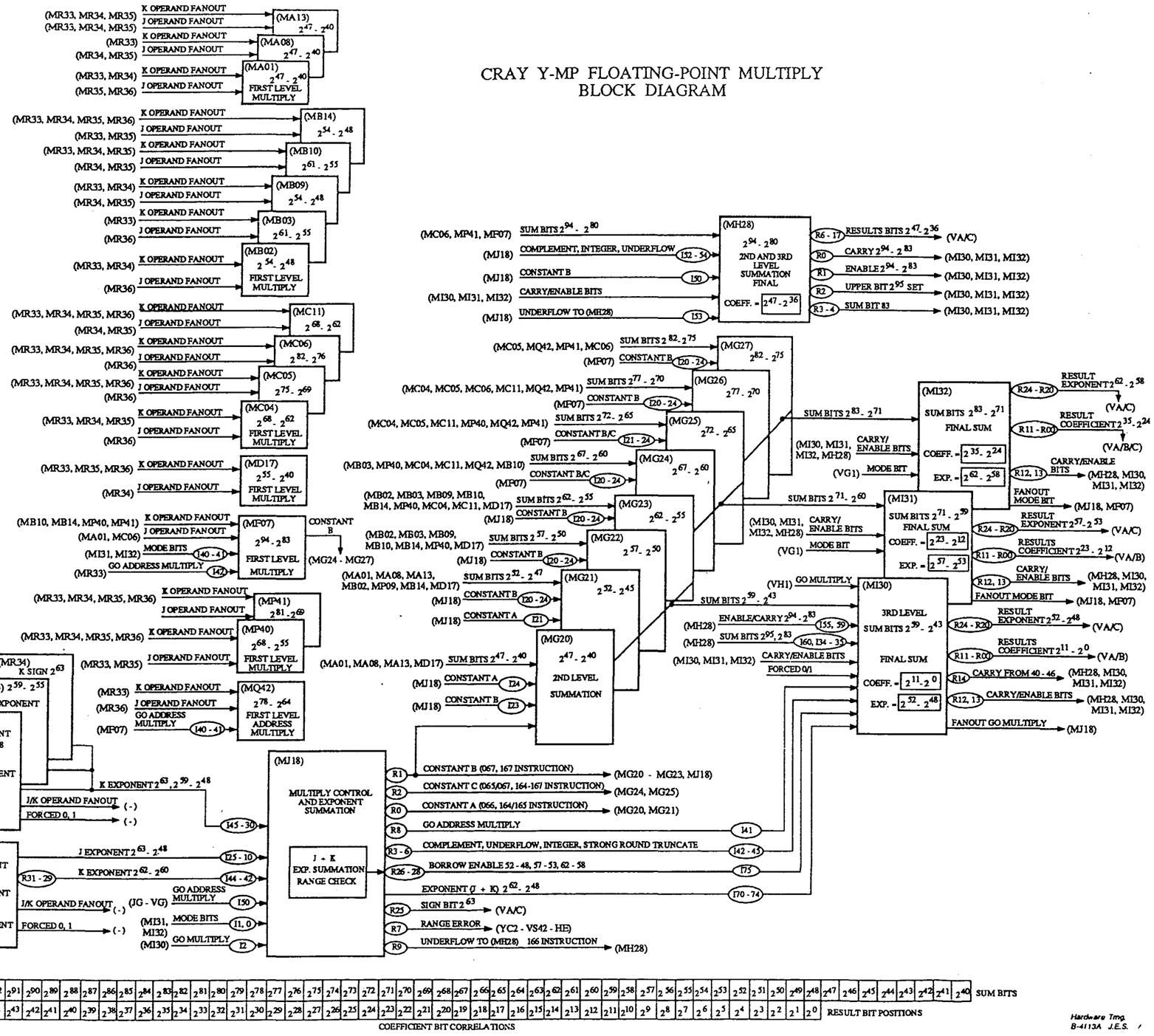
Four (MR) options are responsible for receiving the original two floating-point or integer operands. The exponents of the j and k operands are sent to the (MJ) for summation. The coefficients are fanned out to the (MA, MB, MC, MD, MF, MP, and MQ) options to form the multiply pyramid. Many copies of the original coefficient are needed for the various options that make up the pyramid, some of which are also fanned out by the (MA01), (MB10), (MB14), (MC06), (MP40) and (MP41) options.

Hardware Trng.
YM01/12A J.E.S.

CRAY Y-MP FLOATING-POINT MULTIPLY BLOCK DIAGRAM



CRAY Y-MP FLOATING-POINT MULTIPLY BLOCK DIAGRAM



CRAY PROPRIETARY

(MA01) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^0	(MR34)	R0	I20	2^{28}	(MR35)	R8	I0
2^1	(MR34)	R1	I21	2^{29}	(MR35)	R9	I1
2^2	(MR34)	R2	I22	2^{30}	(MR35)	R10	I2
2^3	(MR34)	R3	I23	2^{31}	(MR35)	R11	I3
2^4	(MR34)	R4	I24	2^{32}	(MR35)	R12	I4
2^5	(MR34)	R5	I25	2^{33}	(MR35)	R13	I5
2^6	(MR34)	R6	I26	2^{34}	(MR35)	R14	I6
2^7	(MR34)	R7	I27	2^{35}	(MR36)	R4	I7
2^8	(MR33)	R23	I28	2^{36}	(MR36)	R5	I8
2^9	(MR33)	R24	I29	2^{37}	(MR36)	R6	I9
2^{10}	(MR33)	R25	I30	2^{38}	(MR36)	R7	I10
2^{11}	(MR33)	R26	I31	2^{39}	(MR36)	R8	I11
2^{12}	(MR33)	R27	I32	2^{40}	(MR36)	R9	I12
				2^{41}	(MR36)	R10	I13
				2^{42}	(MR36)	R11	I14
				2^{43}	(MR36)	R12	I15
				2^{44}	(MR36)	R13	I16
				2^{45}	(MR36)	R14	I17
				2^{46}	(MR36)	R15	I18
				2^{47}	(MR36)	R16	I19

(MA08) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{13}	(MR33)	R28	I20	2^{15}	(MR35)	R17	I0
2^{14}	(MR33)	R0	I21	2^{16}	(MR35)	R18	I1
2^{15}	(MR33)	R1	I22	2^{17}	(MR35)	R19	I2
2^{16}	(MR33)	R2	I23	2^{18}	(MR35)	R20	I3
2^{17}	(MR33)	R3	I24	2^{19}	(MR35)	R21	I4
2^{18}	(MR33)	R4	I25	2^{20}	(MR35)	R22	I5
2^{19}	(MR33)	R5	I26	2^{21}	(MR35)	R23	I6
2^{20}	(MR33)	R6	I27	2^{22}	(MR35)	R15	I7
2^{21}	(MR33)	R7	I28	2^{23}	(MR35)	R16	I8
2^{22}	(MR33)	R8	I29	2^{24}	(MR34)	R17	I9
2^{23}	(MR33)	R9	I30	2^{25}	(MR34)	R18	I10
2^{24}	(MR33)	R10	I31	2^{26}	(MR34)	R19	I11
2^{25}	(MR33)	R11	I32	2^{27}	(MR34)	R20	I12
				2^{28}	(MR35)	R08	I13
				2^{29}	(MR35)	R09	I14
				2^{30}	(MR35)	R10	I15
				2^{31}	(MR35)	R11	I16
				2^{32}	(MR35)	R12	I17
				2^{33}	(MR35)	R13	I18
				2^{34}	(MR35)	R14	I19

Hardware Trng.
A-4480A J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 1 of 9)

(MA13) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{26}	(MR33)	R31	I20	2^{02}	(MR34)	R10	I0
2^{27}	(MR33)	R30	I21	2^{03}	(MR34)	R11	I1
2^{28}	(MR34)	R31	I22	2^{04}	(MR34)	R12	I2
2^{29}	(MR34)	R30	I23	2^{05}	(MR34)	R13	I3
2^{30}	(MR34)	R29	I24	2^{06}	(MR34)	R14	I4
2^{31}	(MR34)	R28	I25	2^{07}	(MR34)	R15	I5
2^{32}	(MR34)	R27	I26	2^{08}	(MR34)	R16	I6
2^{33}	(MR35)	R24	I27	2^{09}	(MR33)	R12	I7
2^{34}	(MR34)	R26	I28	2^{10}	(MR33)	R13	I8
2^{35}	(MR34)	R25	I29	2^{11}	(MR33)	R14	I9
2^{36}	(MR34)	R24	I30	2^{12}	(MR33)	R15	I10
2^{37}	(MR34)	R23	I31	2^{13}	(MR33)	R16	I11
2^{38}	(MR34)	R22	I32	2^{14}	(MR33)	R22	I12
				2^{15}	(MR35)	R17	I13
				2^{16}	(MR35)	R18	I14
				2^{17}	(MR35)	R19	I15
				2^{18}	(MR35)	R20	I16
				2^{19}	(MR35)	R21	I17
				2^{20}	(MR35)	R22	I18
				2^{21}	(MR35)	R23	I19

(MB02) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{01}	(MR34)	R1	I13	2^{35}	(MR36)	R4	I0
2^{02}	(MR34)	R2	I14	2^{36}	(MR36)	R5	I1
2^{03}	(MR34)	R3	I15	2^{37}	(MR36)	R6	I2
2^{04}	(MR34)	R4	I16	2^{38}	(MR36)	R7	I3
2^{05}	(MR34)	R5	I17	2^{39}	(MR36)	R8	I4
2^{06}	(MR34)	R6	I18	2^{40}	(MR36)	R9	I5
2^{07}	(MR34)	R7	I19	2^{41}	(MR36)	R10	I6
2^{08}	(MR33)	R23	I20	2^{42}	(MR36)	R11	I7
2^{09}	(MR33)	R24	I21	2^{43}	(MR36)	R12	I8
2^{10}	(MR33)	R25	I22	2^{44}	(MR36)	R13	I9
2^{11}	(MR33)	R26	I23	2^{45}	(MR36)	R14	I10
2^{12}	(MR33)	R27	I24	2^{46}	(MR36)	R15	I11
2^{13}	(MR33)	R28	I25	2^{47}	(MR36)	R16	I12
2^{14}	(MR33)	R0	I26				
2^{15}	(MR33)	R1	I27				
2^{16}	(MR33)	R2	I28				
2^{17}	(MR33)	R3	I29		FORCED 1 FANOUT J, K		I32
2^{18}	(MR33)	R4	I30				
2^{19}	(MR33)	R5	I31				

Hardware Trng.
A-4481A J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 2 of 9)

(MB03) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{8}	(MR33)	R23	I13	2^{35}	(MR36)	R4	I0
2^{9}	(MR33)	R24	I14	2^{36}	(MR36)	R5	I1
2^{10}	(MR33)	R25	I15	2^{37}	(MR36)	R6	I2
2^{11}	(MR33)	R26	I16	2^{38}	(MR36)	R7	I3
2^{12}	(MR33)	R27	I17	2^{39}	(MR36)	R8	I4
2^{13}	(MR33)	R28	I18	2^{40}	(MR36)	R9	I5
2^{14}	(MR33)	R0	I19	2^{41}	(MR36)	R10	I6
2^{15}	(MR33)	R1	I20	2^{42}	(MR36)	R11	I7
2^{16}	(MR33)	R2	I21	2^{43}	(MR36)	R12	I8
2^{17}	(MR33)	R3	I22	2^{44}	(MR36)	R13	I9
2^{18}	(MR33)	R4	I23	2^{45}	(MR36)	R14	I10
2^{19}	(MR33)	R5	I24	2^{46}	(MR36)	R15	I11
2^{20}	(MR33)	R6	I25	2^{47}	(MR36)	R16	I12
2^{21}	(MR33)	R7	I26				
2^{22}	(MR33)	R8	I27				
2^{23}	(MR33)	R9	I28				
2^{24}	(MR33)	R10	I29				
2^{25}	(MR33)	R11	I30		FORCED 1 J/K FANOUT		I32
2^{26}	(MR33)	R31	I31				

(MB09) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{14}	(MR33)	R0	I13	2^{22}	(MR35)	R15	I0
2^{15}	(MR33)	R1	I14	2^{23}	(MR35)	R16	I1
2^{16}	(MR33)	R2	I15	2^{24}	(MR34)	R17	I2
2^{17}	(MR33)	R3	I16	2^{25}	(MR34)	R18	I3
2^{18}	(MR33)	R4	I17	2^{26}	(MR34)	R19	I4
2^{19}	(MR33)	R5	I18	2^{27}	(MR34)	R20	I5
2^{20}	(MR33)	R6	I19	2^{28}	(MR35)	R8	I6
2^{21}	(MR33)	R7	I20	2^{29}	(MR35)	R9	I7
2^{22}	(MR33)	R8	I21	2^{30}	(MR35)	R10	I8
2^{23}	(MR33)	R9	I22	2^{31}	(MR35)	R11	I9
2^{24}	(MR33)	R10	I23	2^{32}	(MR35)	R12	I10
2^{25}	(MR33)	R11	I24	2^{33}	(MR35)	R13	I11
2^{26}	(MR33)	R31	I25	2^{34}	(MR35)	R14	I12
2^{27}	(MR33)	R30	I26				
2^{28}	(MR34)	R31	I27				
2^{29}	(MR34)	R30	I28				
2^{30}	(MR34)	R29	I29		FORCED 1 FANOUT J, K		I32
2^{31}	(MR34)	R28	I30				
2^{32}	(MR34)	R27	I31				

Hardware Trng.
A-4482 J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 3 of 9)

(MB10) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{21}	(MR33)	R7	I13	2^{22}	(MR35)	R15	I0
2^{22}	(MR33)	R8	I14	2^{23}	(MR35)	R16	I1
2^{23}	(MR33)	R9	I15	2^{24}	(MR34)	R17	I2
2^{24}	(MR33)	R10	I16	2^{25}	(MR34)	R18	I3
2^{25}	(MR33)	R11	I17	2^{26}	(MR34)	R19	I4
2^{26}	(MR33)	R31	I18	2^{27}	(MR34)	R20	I5
2^{27}	(MR33)	R30	I19	2^{28}	(MR35)	R8	I6
2^{28}	(MR34)	R31	I20	2^{29}	(MR35)	R9	I7
2^{29}	(MR34)	R30	I21	2^{30}	(MR35)	R10	I8
2^{30}	(MR34)	R29	I22	2^{31}	(MR35)	R11	I9
2^{31}	(MR34)	R28	I23	2^{32}	(MR35)	R12	I10
2^{32}	(MR34)	R27	I24	2^{33}	(MR35)	R13	I11
2^{33}	(MR35)	R24	I25	2^{34}	(MR35)	R14	I12
2^{34}	(MR34)	R26	I26				
2^{35}	(MR34)	R25	I27				
2^{36}	(MR34)	R24	I28		FORCED 0 J/K FANOUT		I32
2^{37}	(MR34)	R23	I29				
2^{38}	(MR34)	R22	I30				
2^{39}	(MR33)	R29	I31				

(MB14) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{27}	(MR33)	R30	I13	2^{09}	(MR33)	R12	I0
2^{28}	(MR34)	R31	I14	2^{10}	(MR33)	R13	I1
2^{29}	(MR34)	R30	I15	2^{11}	(MR33)	R14	I2
2^{30}	(MR34)	R29	I16	2^{12}	(MR33)	R15	I3
2^{31}	(MR34)	R28	I17	2^{13}	(MR33)	R16	I4
2^{32}	(MR34)	R27	I18	2^{14}	(MR33)	R22	I5
2^{33}	(MR35)	R24	I19	2^{15}	(MR35)	R17	I6
2^{34}	(MR34)	R26	I20	2^{16}	(MR35)	R18	I7
2^{35}	(MR34)	R25	I21	2^{17}	(MR35)	R19	I8
2^{36}	(MR34)	R24	I22	2^{18}	(MR35)	R20	I9
2^{37}	(MR34)	R23	I23	2^{19}	(MR35)	R21	I10
2^{38}	(MR34)	R22	I24	2^{20}	(MR35)	R22	I11
2^{39}	(MR33)	R29	I25	2^{21}	(MR35)	R23	I12
2^{40}	(MR36)	R28	I26				
2^{41}	(MR35)	R1	I27				
2^{42}	(MR36)	R29	I28		FORCED 0 FANOUT J, K		I32
2^{43}	(MR35)	R3	I29				
2^{44}	(MR36)	R30	I30				
2^{45}	(MR35)	R5	I31				

Hardware Trng.
A-4483 J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 4 of 9)

(MC04) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{15}	(MR33)	R1	I13	2^{35}	(MR36)	R4	I0
2^{16}	(MR33)	R2	I14	2^{36}	(MR36)	R5	I1
2^{17}	(MR33)	R3	I15	2^{37}	(MR36)	R6	I2
2^{18}	(MR33)	R4	I16	2^{38}	(MR36)	R7	I3
2^{19}	(MR33)	R5	I17	2^{39}	(MR36)	R8	I4
2^{20}	(MR33)	R6	I18	2^{40}	(MR36)	R9	I5
2^{21}	(MR33)	R7	I19	2^{41}	(MR36)	R10	I6
2^{22}	(MR33)	R8	I20	2^{42}	(MR36)	R11	I7
2^{23}	(MR33)	R9	I21	2^{43}	(MR36)	R12	I8
2^{24}	(MR33)	R10	I22	2^{44}	(MR36)	R13	I9
2^{25}	(MR33)	R11	I23	2^{45}	(MR36)	R14	I10
2^{26}	(MR33)	R31	I24	2^{46}	(MR36)	R15	I11
2^{27}	(MR33)	R30	I25	2^{47}	(MR36)	R16	I12
2^{28}	(MR34)	R31	I26				
2^{29}	(MR34)	R30	I27				
2^{30}	(MR34)	R29	I28		FORCED 1 J/K FANOUT		I32
2^{31}	(MR34)	R28	I29				
2^{32}	(MR34)	R27	I30				
2^{33}	(MR35)	R24	I31				

(MC05) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{22}	(MR33)	R8	I13	2^{35}	(MR36)	R4	I0
2^{23}	(MR33)	R9	I14	2^{36}	(MR36)	R5	I1
2^{24}	(MR33)	R10	I15	2^{37}	(MR36)	R6	I2
2^{25}	(MR33)	R11	I16	2^{38}	(MR36)	R7	I3
2^{26}	(MR33)	R31	I17	2^{39}	(MR36)	R8	I4
2^{27}	(MR33)	R30	I18	2^{40}	(MR36)	R9	I5
2^{28}	(MR34)	R31	I19	2^{41}	(MR36)	R10	I6
2^{29}	(MR34)	R30	I20	2^{42}	(MR36)	R11	I7
2^{30}	(MR34)	R29	I21	2^{43}	(MR36)	R12	I8
2^{31}	(MR34)	R28	I22	2^{44}	(MR36)	R13	I9
2^{32}	(MR34)	R27	I23	2^{45}	(MR36)	R14	I10
2^{33}	(MR35)	R24	I24	2^{46}	(MR36)	R15	I11
2^{34}	(MR34)	R26	I25	2^{47}	(MR36)	R16	I12
2^{35}	(MR34)	R25	I26				
2^{36}	(MR34)	R24	I27				
2^{37}	(MR34)	R23	I28		FORCED 1 J/K FANOUT		I32
2^{38}	(MR34)	R22	I29				
2^{39}	(MR33)	R29	I30				
2^{40}	(MR36)	R28	I31				

Hardware Trng.
A-4484 J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 5 of 9)

(MC06) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{29}	(MR34)	R30	I13	2^{35}	(MR36)	R4	I0
2^{30}	(MR34)	R29	I14	2^{36}	(MR36)	R5	I1
2^{31}	(MR34)	R28	I15	2^{37}	(MR36)	R6	I2
2^{32}	(MR34)	R27	I16	2^{38}	(MR36)	R7	I3
2^{33}	(MR35)	R24	I17	2^{39}	(MR36)	R8	I4
2^{34}	(MR34)	R26	I18	2^{40}	(MR36)	R9	I5
2^{35}	(MR34)	R25	I19	2^{41}	(MR36)	R10	I6
2^{36}	(MR34)	R24	I20	2^{42}	(MR36)	R11	I7
2^{37}	(MR34)	R23	I21	2^{43}	(MR36)	R12	I8
2^{38}	(MR34)	R22	I22	2^{44}	(MR36)	R13	I9
2^{39}	(MR33)	R29	I23	2^{45}	(MR36)	R14	I10
2^{40}	(MR36)	R28	I24	2^{46}	(MR36)	R15	I11
2^{41}	(MR35)	R1	I25	2^{47}	(MR36)	R16	I12
2^{42}	(MR36)	R29	I26				
2^{43}	(MR35)	R3	I27				
2^{44}	(MR36)	R30	I28				
2^{45}	(MR35)	R5	I29		FORCED 1 J/K FANOUT		I32
2^{46}	(MR36)	R31	I30				
2^{47}	(MR35)	R7	I31				

(MC11) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{28}	(MR34)	R31	I13	2^{22}	(MR35)	R15	I0
2^{29}	(MR34)	R30	I14	2^{23}	(MR35)	R16	I1
2^{30}	(MR34)	R29	I15	2^{24}	(MR34)	R17	I2
2^{31}	(MR34)	R28	I16	2^{25}	(MR34)	R18	I3
2^{32}	(MR34)	R27	I17	2^{26}	(MR34)	R19	I4
2^{33}	(MR35)	R24	I18	2^{27}	(MR34)	R20	I5
2^{34}	(MR34)	R26	I19	2^{28}	(MR35)	R8	I6
2^{35}	(MR34)	R25	I20	2^{29}	(MR35)	R9	I7
2^{36}	(MR34)	R24	I21	2^{30}	(MR35)	R10	I8
2^{37}	(MR34)	R23	I22	2^{31}	(MR35)	R11	I9
2^{38}	(MR34)	R22	I23	2^{32}	(MR35)	R12	I10
2^{39}	(MR33)	R29	I24	2^{33}	(MR35)	R13	I11
2^{40}	(MR36)	R28	I25	2^{34}	(MR35)	R14	I12
2^{41}	(MR35)	R1	I26				
2^{42}	(MR36)	R29	I27				
2^{43}	(MR35)	R3	I28				
2^{44}	(MR36)	R30	I29		FORCED 0 J/K FANOUT		I32
2^{45}	(MR35)	R5	I30				
2^{46}	(MR36)	R31	I31				

Hardware Trng.
A-4485 J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 6 of 9)

(MD17) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{39}	(MR33)	R29	I10	2^{00}	(MR34)	R8	I0
2^{40}	(MR36)	R28	I11	2^{01}	(MR34)	R9	I1
2^{41}	(MR35)	R1	I12	2^{02}	(MR34)	R10	I2
2^{42}	(MR36)	R29	I13	2^{03}	(MR34)	R11	I3
2^{43}	(MR35)	R3	I14	2^{04}	(MR34)	R12	I4
2^{44}	(MR36)	R30	I15	2^{05}	(MR34)	R13	I5
2^{45}	(MR35)	R5	I16	2^{06}	(MR34)	R14	I6
2^{46}	(MR36)	R31	I17	2^{07}	(MR34)	R15	I7
2^{47}	(MR35)	R7	I18	2^{08}	(MR34)	R16	I8

(MF07) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{36}	(MB10)	R24	I20	2^{35}	(MC06)	R21	I0
2^{37}	(MB10)	R25	I21	2^{36}	(MC06)	R22	I1
2^{38}	(MB10)	R26	I22	2^{37}	(MC06)	R23	I2
2^{39}	(MB14)	R21	I23	2^{38}	(MC06)	R24	I3
2^{40}	(MB14)	R22	I24	2^{39}	(MC06)	R25	I4
2^{41}	(MB14)	R23	I25	2^{40}	(MC06)	R26	I5
2^{42}	(MB14)	R24	I26	2^{41}	(MC06)	R27	I6
2^{43}	(MB14)	R25	I27	2^{42}	(MA01)	R26	I7
2^{44}	(MB14)	R26	I28	2^{43}	(MA01)	R27	I8
2^{45}	(MB14)	R27	I29	2^{44}	(MA01)	R28	I9
2^{46}	(MP40)	R40	I30	2^{45}	(MA01)	R29	I10
2^{47}	(MP41)	R40	I31	2^{46}	(MA01)	R30	I11
				2^{47}	(MA01)	R31	I12

Hardware Trng.
A-4486 J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 7 of 9)

(MP40) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{34}	(MR34)	R26	I20	2^{09}	(MR33)	R12	I0
2^{35}	(MR34)	R25	I21	2^{10}	(MR33)	R13	I1
2^{36}	(MR34)	R24	I22	2^{11}	(MR33)	R14	I2
2^{37}	(MR34)	R23	I23	2^{12}	(MR33)	R15	I3
2^{38}	(MR34)	R22	I24	2^{13}	(MR33)	R16	I4
2^{39}	(MR33)	R29	I25	2^{14}	(MR33)	R22	I5
2^{40}	(MR36)	R28	I26	2^{15}	(MR35)	R17	I6
2^{41}	(MR35)	R1	I27	2^{16}	(MR35)	R18	I7
2^{42}	(MR36)	R29	I28	2^{17}	(MR35)	R19	I8
2^{43}	(MR35)	R3	I29	2^{18}	(MR35)	R20	I9
2^{44}	(MR36)	R30	I30	2^{19}	(MR35)	R21	I10
2^{45}	(MR35)	R5	I31	2^{20}	(MR35)	R22	I11
2^{46}	(MR36)	R31	I32	2^{21}	(MR35)	R23	I12
2^{47}	(MR35)	R7	I33				

(MP41) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{35}	(MR34)	R25	I20	2^{22}	(MR35)	R15	I0
2^{36}	(MR34)	R24	I21	2^{23}	(MR35)	R16	I1
2^{37}	(MR34)	R23	I22	2^{24}	(MR34)	R17	I2
2^{38}	(MR34)	R22	I23	2^{25}	(MR34)	R18	I3
2^{39}	(MR33)	R29	I24	2^{26}	(MR34)	R19	I4
2^{40}	(MR36)	R28	I25	2^{27}	(MR34)	R20	I5
2^{41}	(MR35)	R1	I26	2^{28}	(MR35)	R8	I6
2^{42}	(MR36)	R29	I27	2^{29}	(MR35)	R9	I7
2^{43}	(MR35)	R3	I28	2^{30}	(MR35)	R10	I8
2^{44}	(MR36)	R30	I29	2^{31}	(MR35)	R11	I9
2^{45}	(MR35)	R5	I30	2^{32}	(MR35)	R12	I10
2^{46}	(MR36)	R31	I31	2^{33}	(MR35)	R13	I11
2^{47}	(MR35)	R7	I32	2^{34}	(MR35)	R14	I12
	FORCED 1		I33				

Hardware Trng.
A-4487A J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 8 of 9)

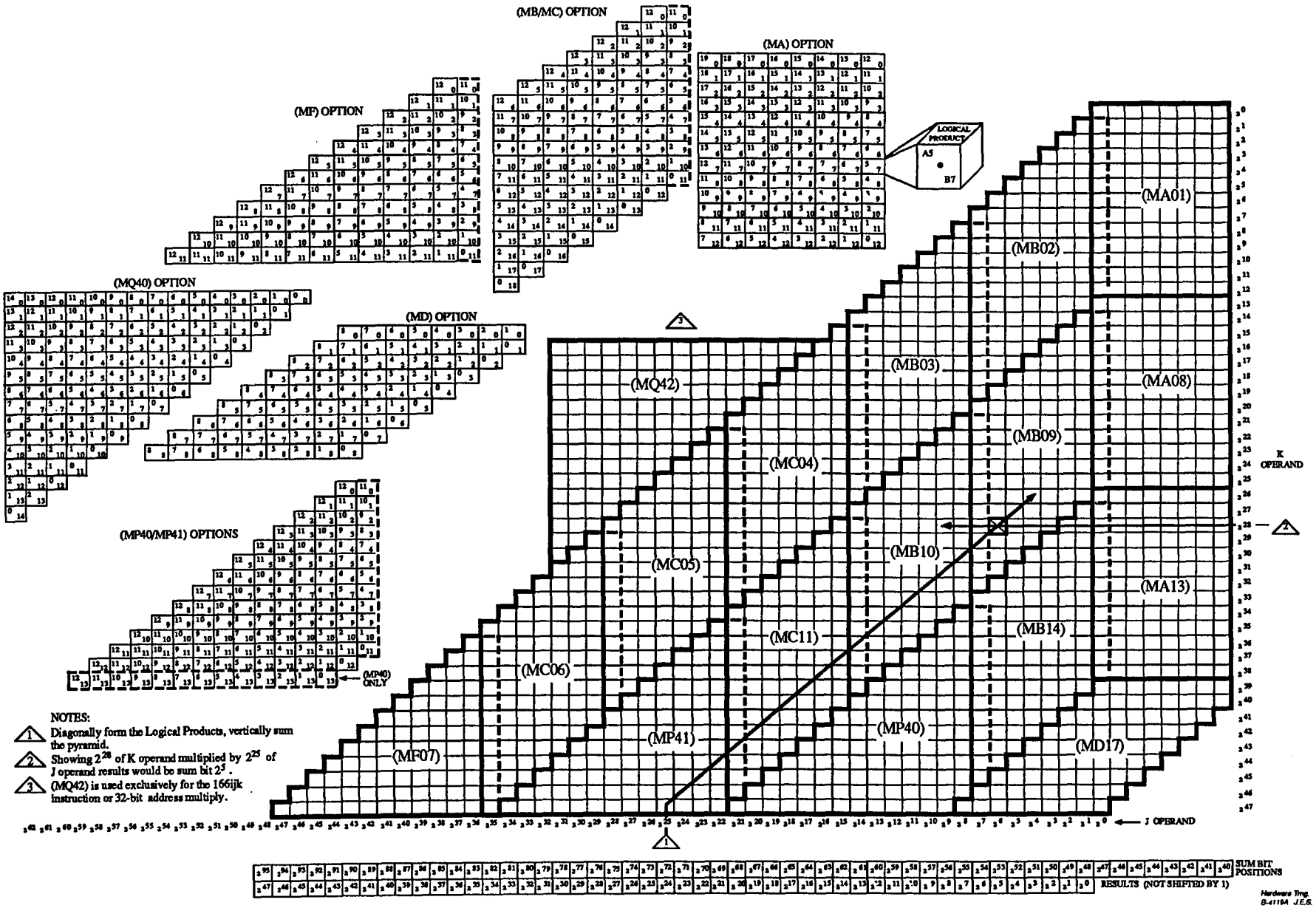
(MQ42) RECEIVING (I TERMS)

K OPERAND FANOUT				J OPERAND FANOUT			
K BIT	FANOUT	R TERM	I TERM	J BIT	FANOUT	R TERM	I TERM
2^{16}	(MR33)	R2	I20	2^{48}	(MR36)	R0	I0
2^{17}	(MR33)	R3	I21	2^{49}	(MR36)	R1	I1
2^{18}	(MR33)	R4	I22	2^{50}	(MR36)	R2	I2
2^{19}	(MR33)	R5	I23	2^{51}	(MR36)	R3	I3
2^{20}	(MR33)	R6	I24	2^{52}	(MR36)	R17	I4
2^{21}	(MR33)	R7	I25	2^{53}	(MR36)	R18	I5
2^{22}	(MR33)	R8	I26	2^{54}	(MR36)	R19	I6
2^{23}	(MR33)	R9	I27	2^{55}	(MR36)	R20	I7
2^{24}	(MR33)	R10	I28	2^{56}	(MR36)	R21	I8
2^{25}	(MR33)	R11	I29	2^{57}	(MR36)	R22	I9
2^{26}	(MR33)	R31	I30	2^{58}	(MR36)	R23	I10
2^{27}	(MR33)	R30	I31	2^{59}	(MR36)	R24	I11
2^{28}	(MR34)	R32	I32	2^{60}	(MR36)	R25	I12
2^{29}	(MR34)	R33	I33	2^{61}	(MR36)	R26	I13
2^{30}	(MR33)	R32	I34	2^{62}	(MR36)	R27	I14

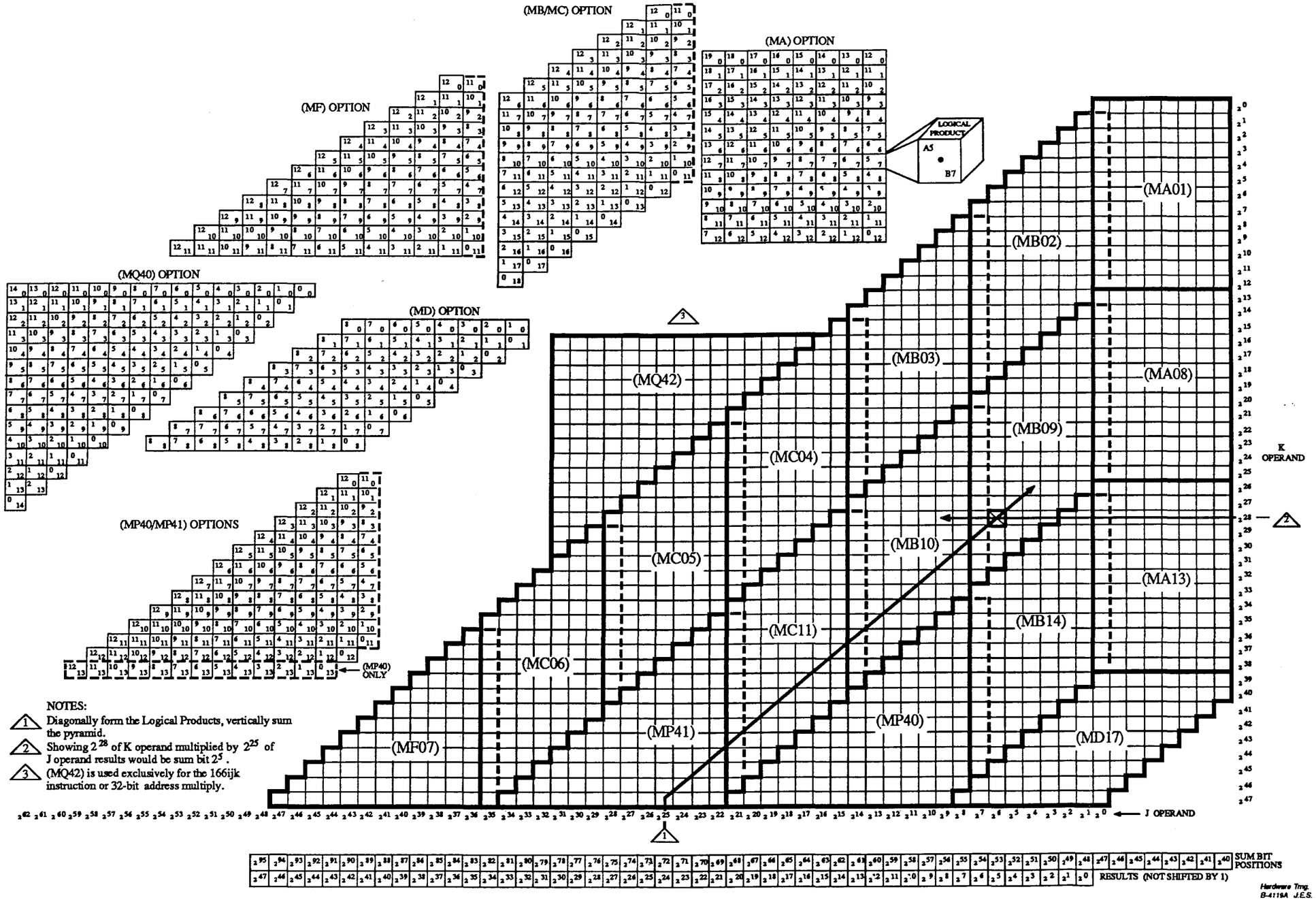
Hardware Trng.
A-4488A J.E.S.

CRAY Y-MP (MR) OPTIONS
FANOUT J AND K OPERANDS
(sheet 9 of 9)

CRAY PROPRIETARY

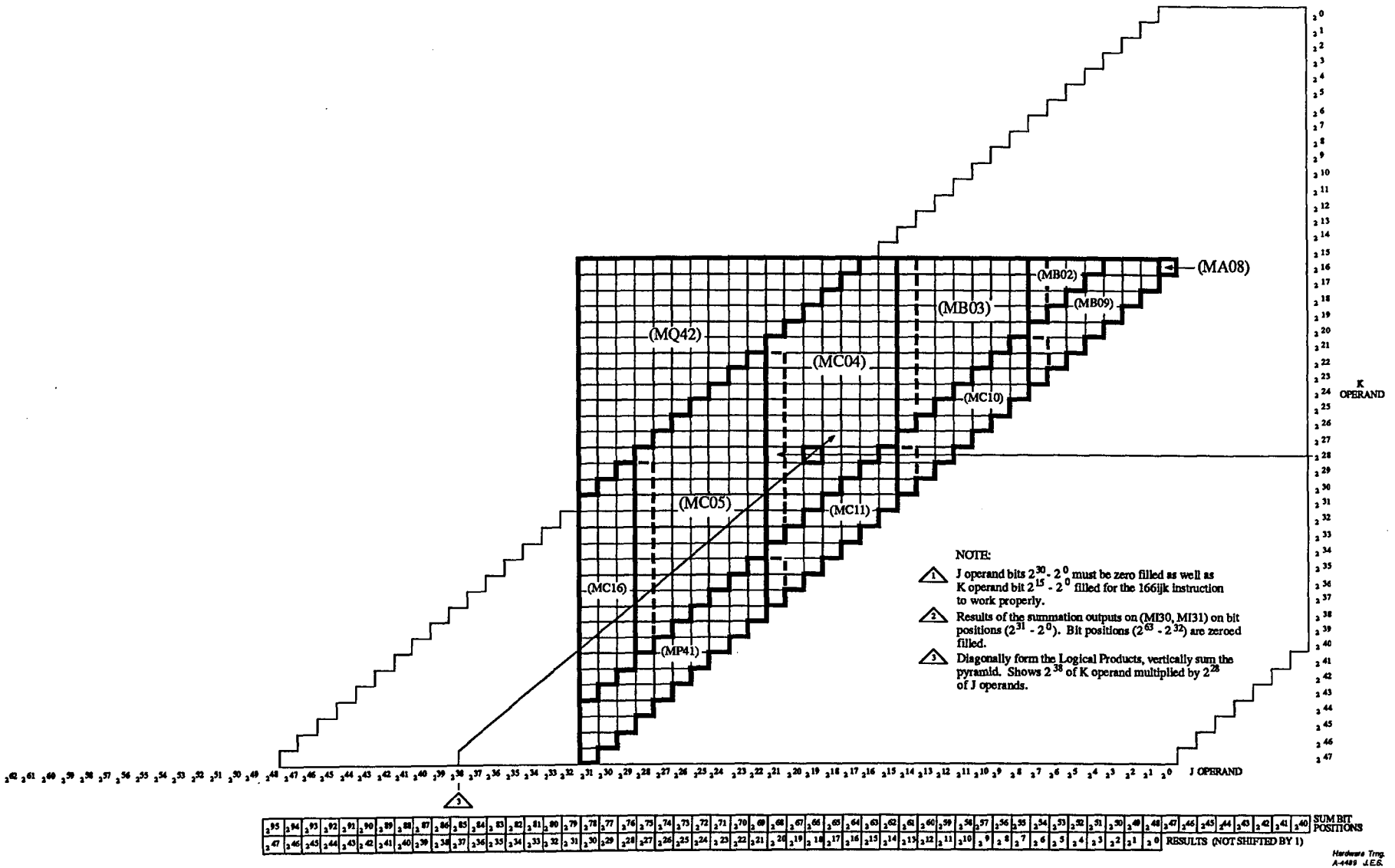


CRAY Y-MP FLOATING-POINT MULTIPLY
LOGICAL PRODUCT PYRAMID FIRST LEVEL



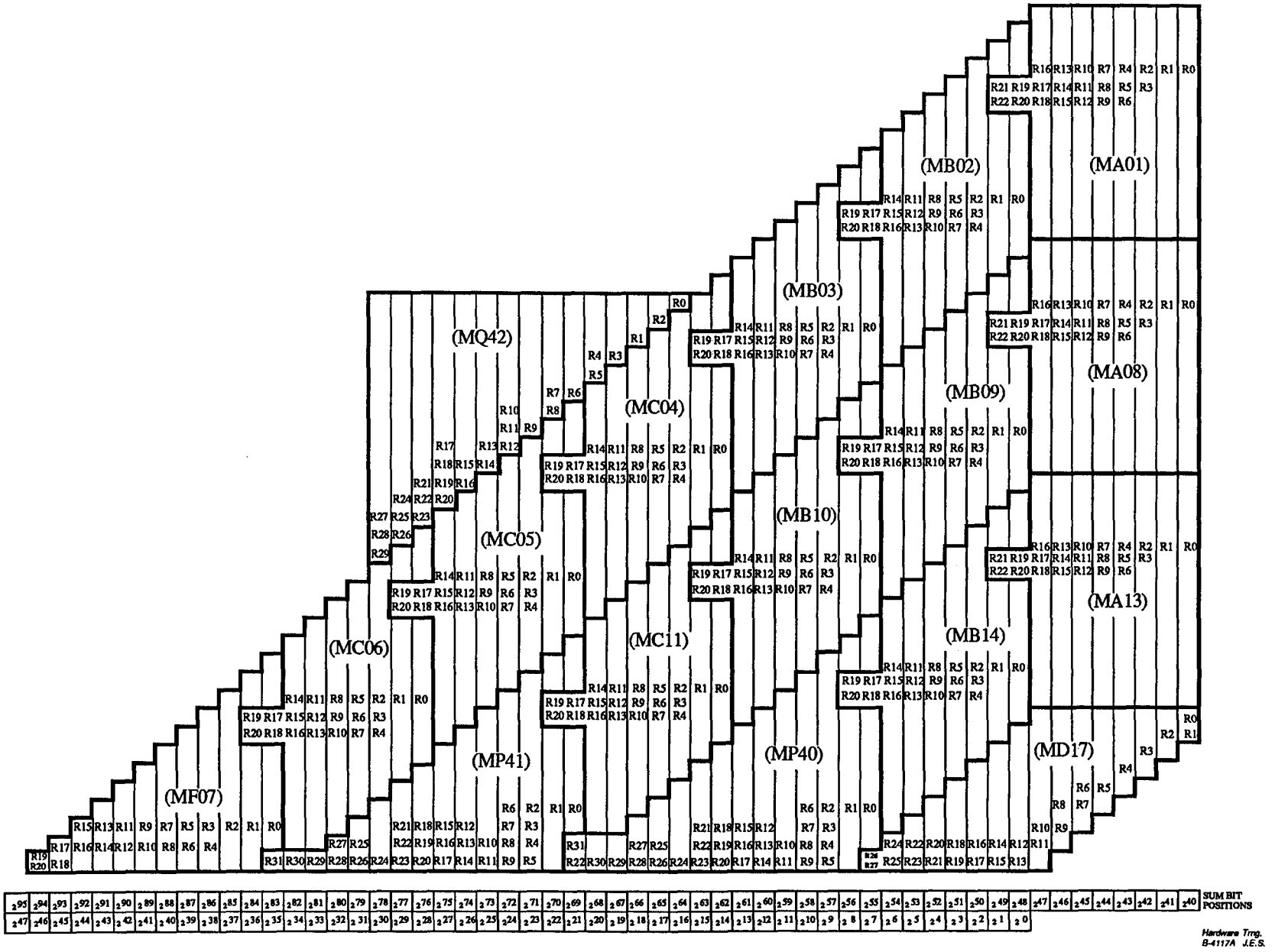
CRAY Y-MP FLOATING-POINT MULTIPLY
LOGICAL PRODUCT PYRAMID FIRST LEVEL

CRAY PROPRIETARY

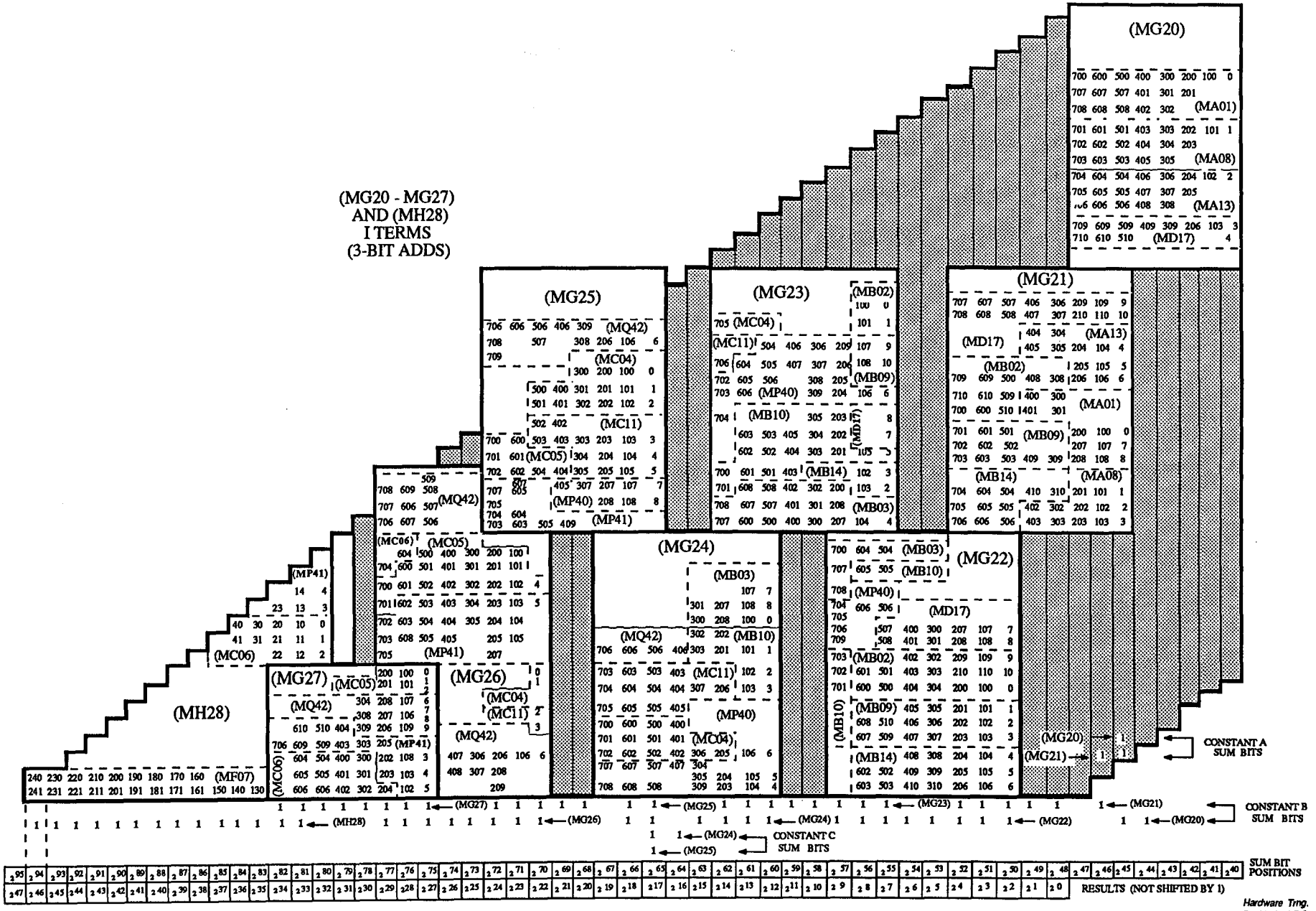


CRAY Y-MP 32-BIT ADDRESS MULTIPLY
LOGICAL PRODUCTS AND PARTIAL SUMMATION

CRAY PROPRIETARY

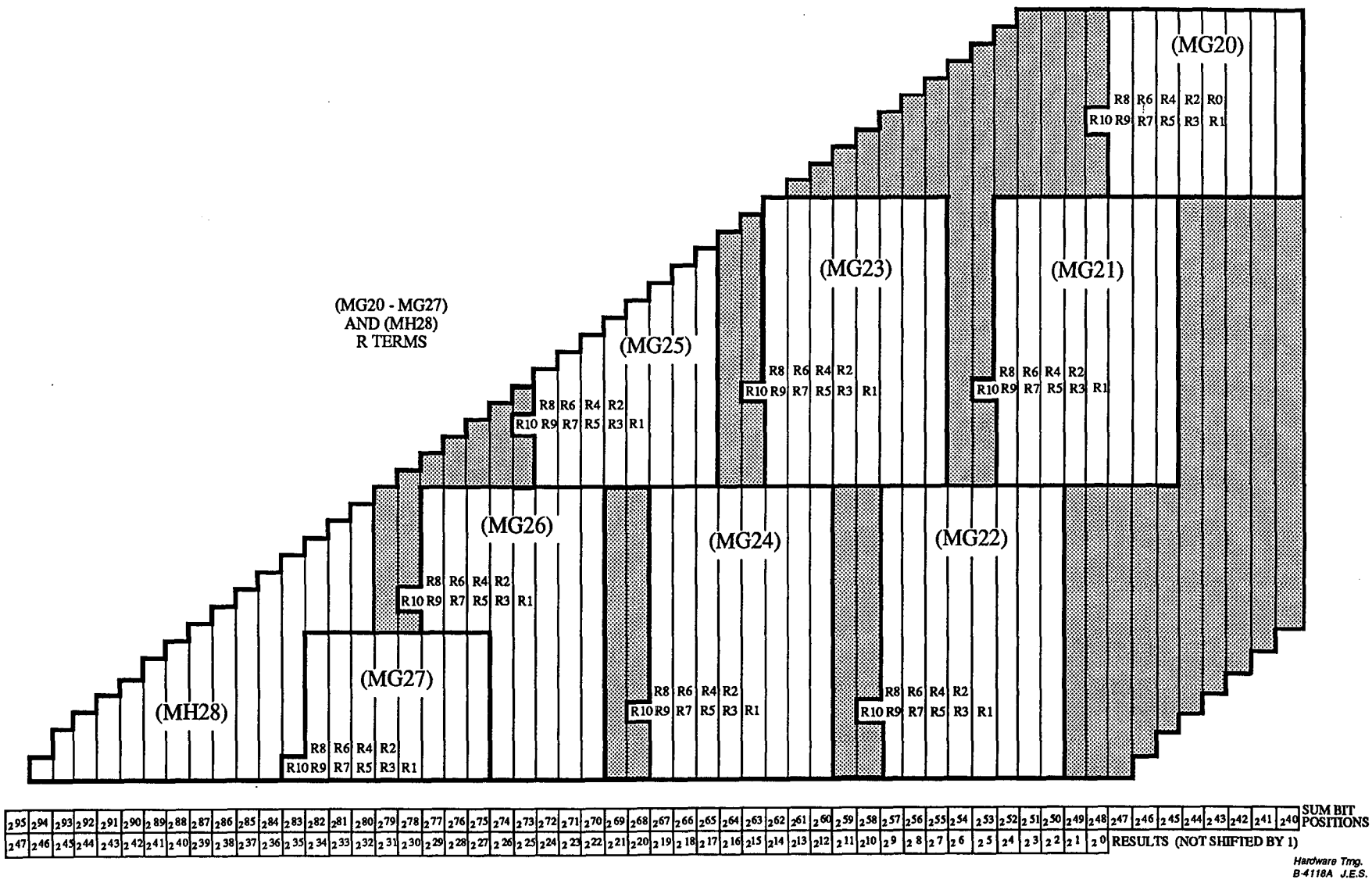


CRAY Y-MP FLOATING-POINT MULTIPLY
FIRST LEVEL SUMMATION (MA-MQ) R TERMS

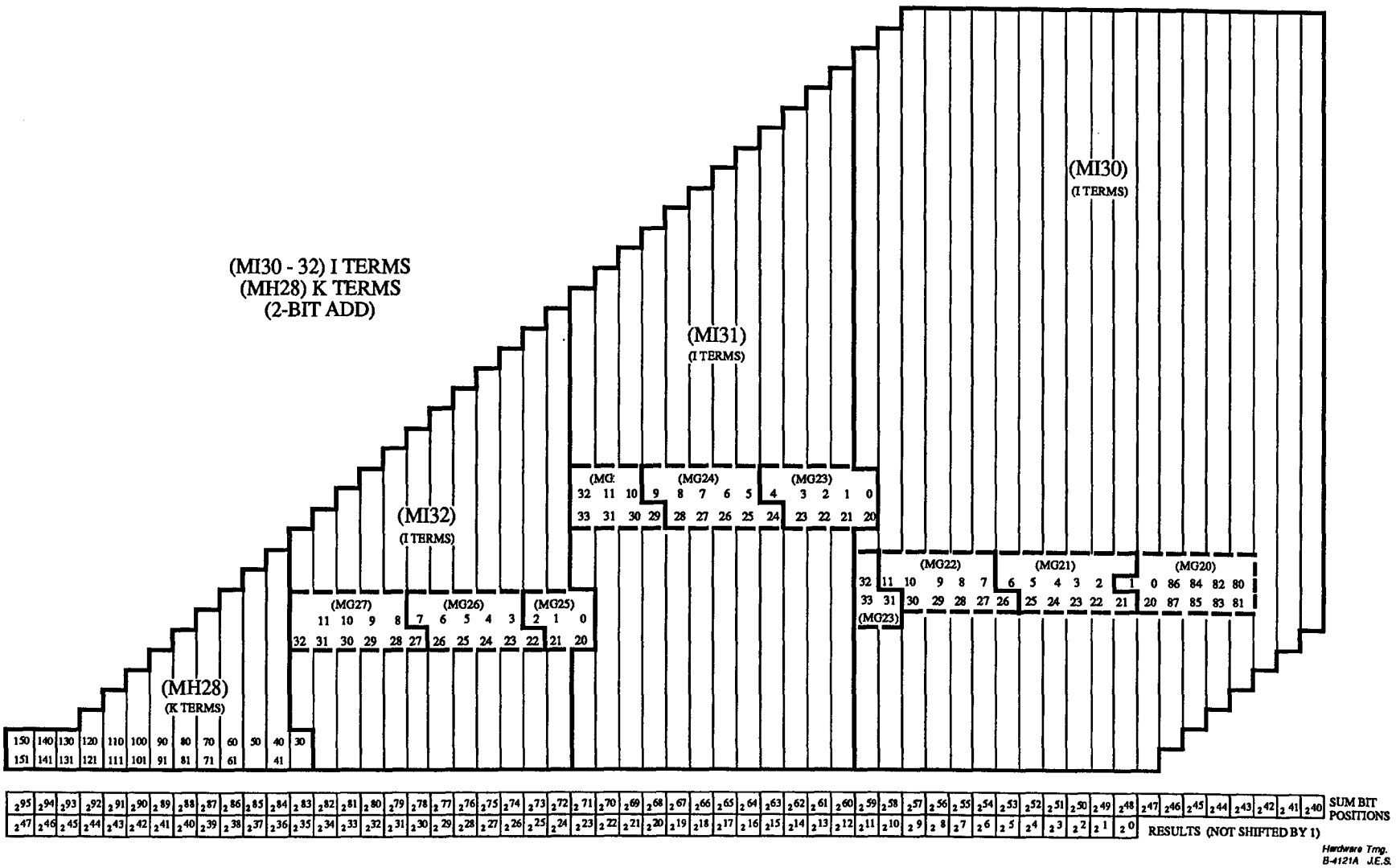


CRAY Y-MP FLOATING MULTIPLY
SECOND LEVEL PARTIAL SUMMATION

CRAY PROPRIETARY

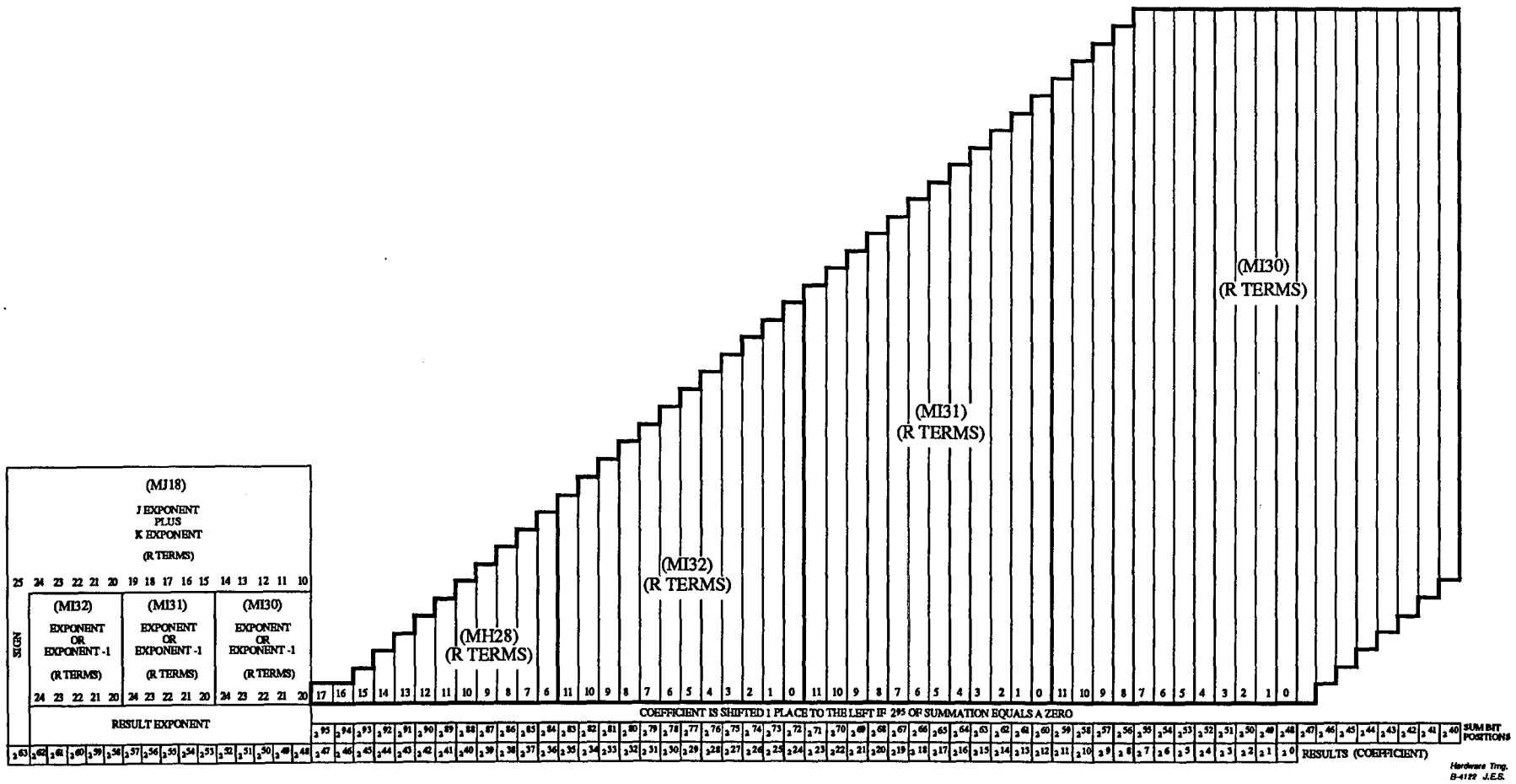


CRAY Y-MP FLOATING-POINT MULTIPLY
PARTIAL SUMMATION SECOND LEVEL MULTIPLY



CRAY Y-MP FLOATING MULTIPLY
3RD LEVEL PARTIAL SUMMATION

CRAY PROPRIETARY

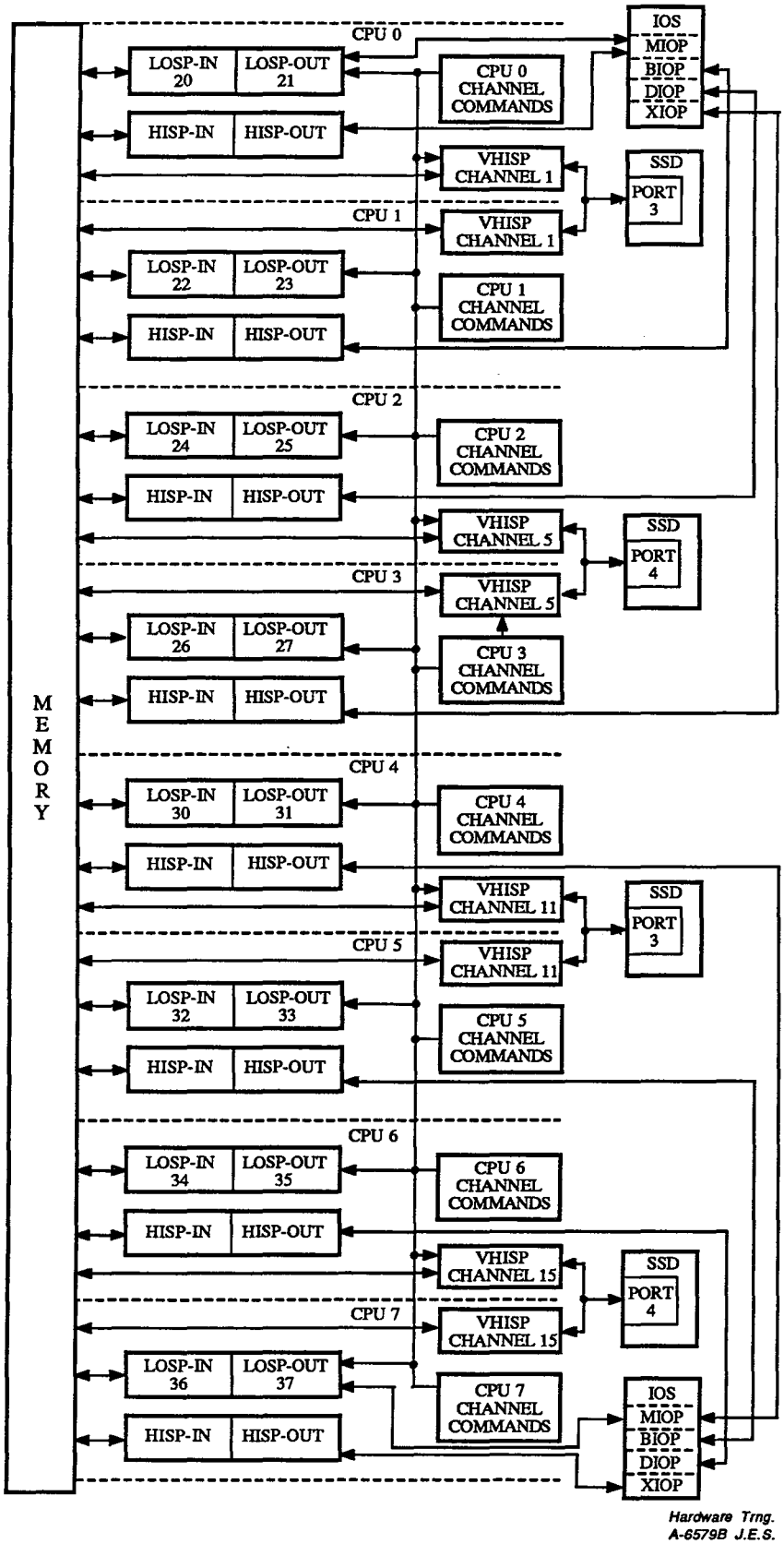


CRAY Y-MP FLOATING MULTIPLY
EXPONENT/COEFFICIENT FINAL RESULTS

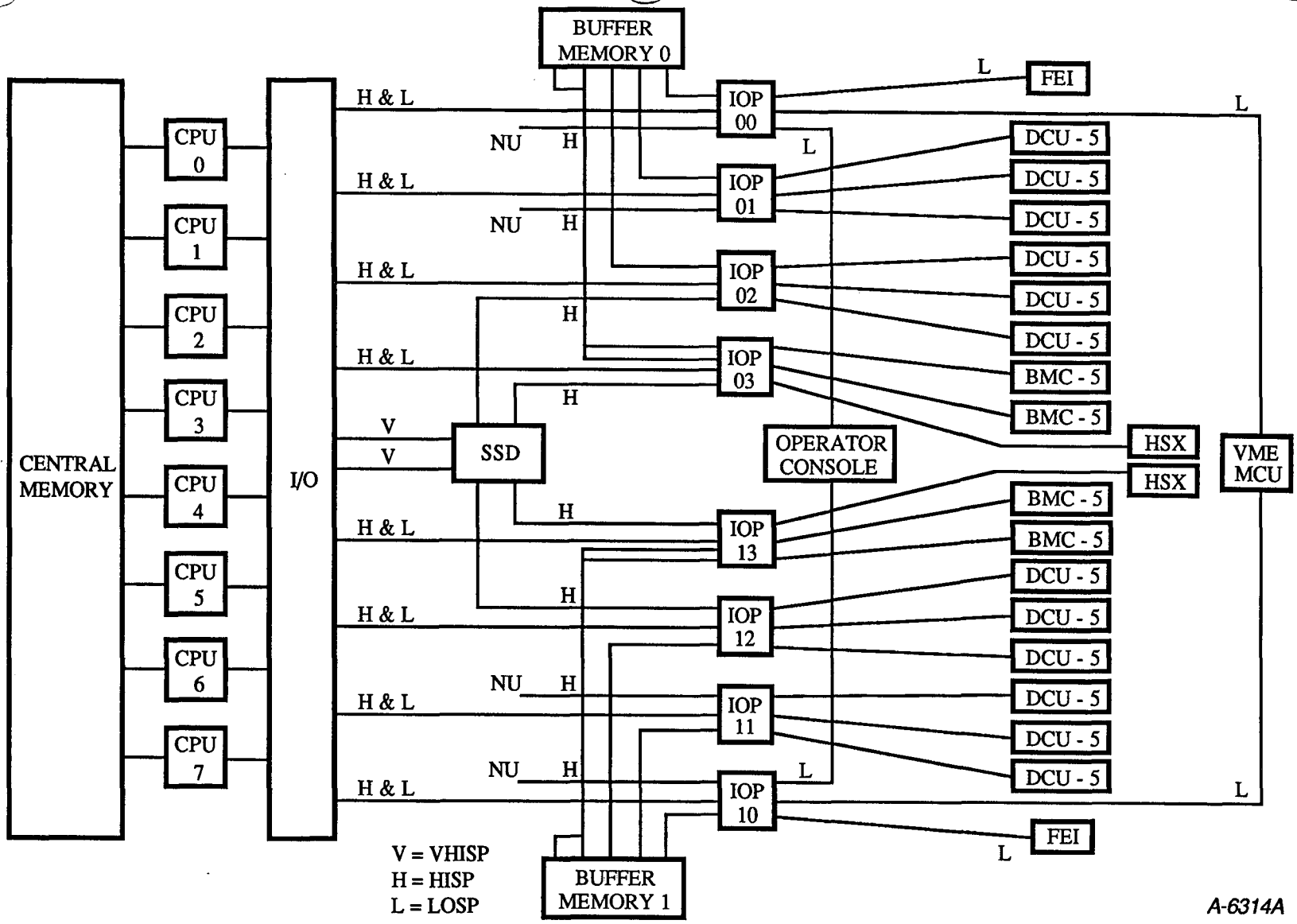
OBJECTIVES - Channels

Upon completion of this section, the student will be able to:

- Program the channel in machine language
- Define the control signal between the channel and external device
- Describe the channel memory priority scheme and how data is assembled and disassembled
- Trace a data bit through the channel
- Describe the sequence of events within the channel control modules for both an input and output operation



CRAY Y-MP I/O LAYOUT



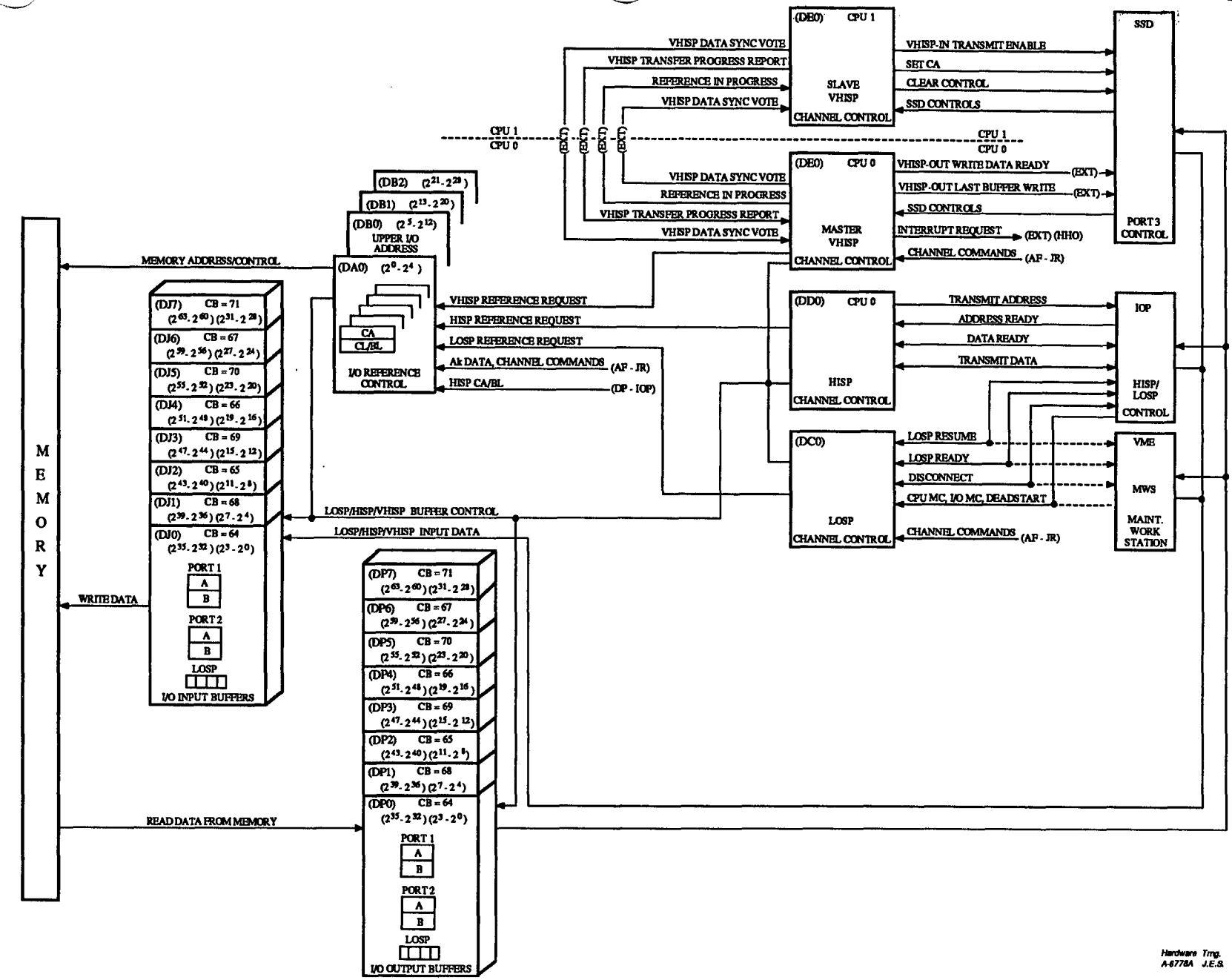
CRAY Y-MP 8/32 SYSTEM CONFIGURATION WITH IOS-D

CRAY Y-MP I/O OPTIONS PER MODULE

OPTION TYPE	NUMBER USED	DESCRIPTION
(DA)	1	I/O Lower address, reference control
(DB)	3	I/O Upper address
(DC)	1	I/O control, LOSP controller
(DD)	1	I/O control, HISP channel controller
(DE)	1	I/O control, VHISP channel controllers
(DJ)	8	Register input buffer
(DP)	8	Register output buffer

23 Total options

Hardware Trng.
YM01/03 J.E.S.



CRAY Y-MP I/O CHANNELS BLOCK DIAGRAM

CRAY PROPRIETARY

DEADSTART SEQUENCE

1. Set I/O MC and CPU MC

   a. Setting I/O MC will set:

      CA = 0
      CL = 0

2. Input channel goes active when I/O MC drops

3. Send CA to memory. Reference valid time increment CA register by 1

4. Stop if:

   a. CA = CL
   b. Error
   c. Disconnect

5. Write 20000_8 words to memory for IOS or 20000_8 for MWS

6. Drop CPU MC:

   a. Force an exchange with memory location 0 - 17_8

7. Execute the program

8. Monitor in the CPU, monitor has control of the machine

9. Dependent on the monitor, monitor control I/O interrupt or never do another exchange

10. With interrupt driven monitor now exchange from 1400 - 1577

11. With monitor program send 20000_8 words to the IOP or MWS and cause an exchange to 1400 - 1577

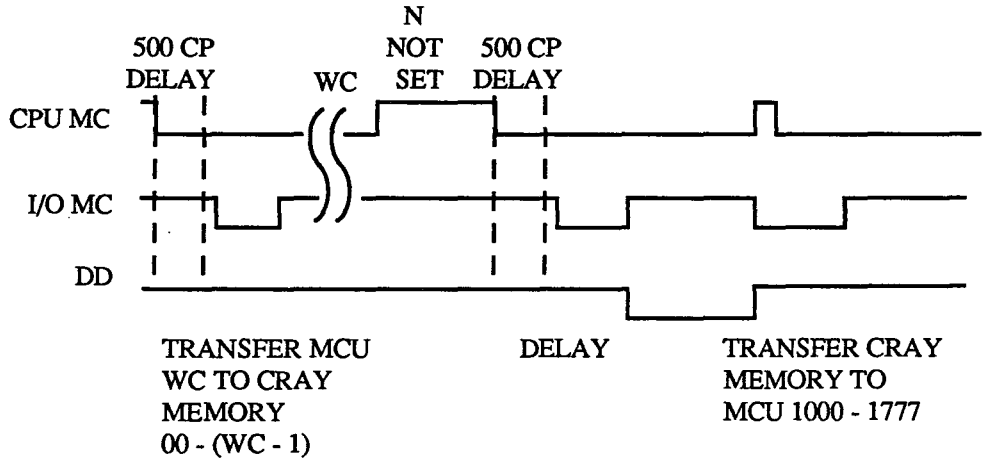
YM01/04B

DEADSTART/DEAD DUMP MODE

DD DELAY WC N

Boot 1 is Similar
DELAY = 1 to 1000
WC = 1 to 10000
N = 1 Prevent an exchange

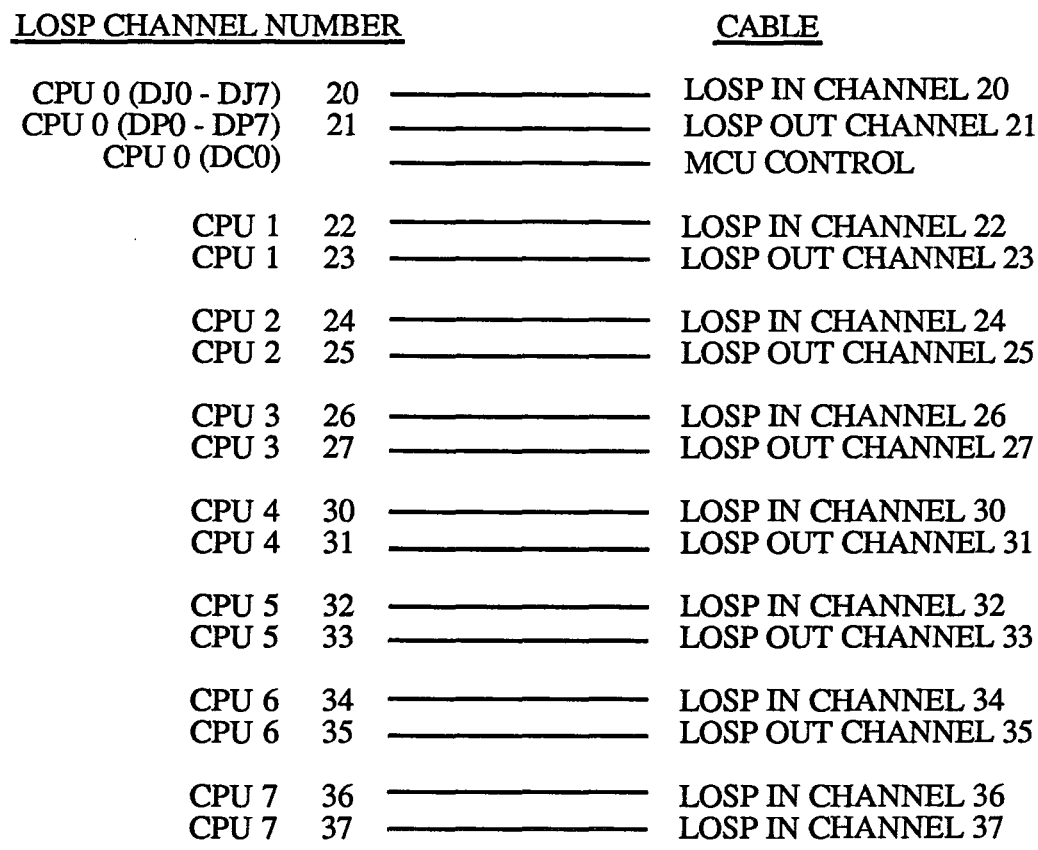
1. Set CPU MC and I/O MC
2. Drop I/O MC
3. Dropping I/O MC set input channel active (CA = 0, CL = 0)
4. Transfer up to 10000_8 initial words to the lower 10000 words of Cray memory
5. Drop CPU MC when transfer is complete; if N = 1 CPU MC will not drop, therefore, preventing an exchange or fetch
6. Wait for time delay to expire
7. Set CPU MC and I/O MC
8. Drop I/O MC and set DD; dead dump prevents channel 20_8 from going active
9. Setting DD causes Cray to output 10000 words over channel 21_8
   a. DD activates the output channel
   b. CA = 0
   c. CL = 0
10. Go to line 1

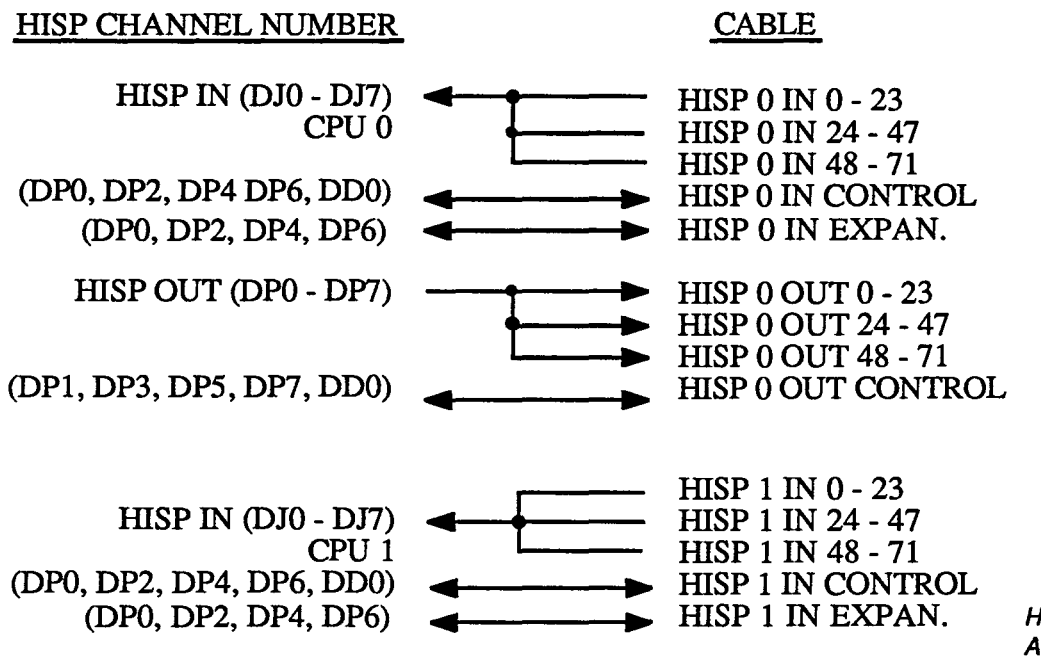


I/O MC - Resync on (DC)
Active - Channel 20 on (DC)
CPU MC - Resync on (DC)
DD - LOSP-Out on (DC)

YM01/05A

CRAY Y-MP CABLE IDENTIFICATION





Hardware Trng.
A-7719B J.E.S.

LOSP CHANNEL

General Overview

The CRAY Y-MP contains a total of eight LOSP channel pairs, eight HISP channel pairs and four VHISP channel pairs. The channels are distributed among the eight CPU modules. Each CPU module contains one LOSP channel pair, one HISP channel pair and half of a VHISP channel pair; the other half of the VHISP channel pair is located on the adjacent CPU module.

All I/O references to memory use Port D. Port D has the lowest priority among Port A/B/C, but it has the highest priority when used by fetch. If fetch won over I/O when competing for Port D, the next reference to use Port D would be I/O. This would prevent fetch from continuously using Port D and not allowing any I/O references. Within a CPU there could be a LOSP, HISP or VHISP in different stages of operation. When a conflict occurs among the channels to use Port D, LOSP would have the highest priority, then HISP followed by VHISP. The fetch or I/O priority is determined on the (YZ) option while the channel priority is determined on the (DA) option.

LOSP channel terminology is associated with the direction of the data flow in respect to the CRAY Y-MP memory. LOSP-In would pertain to the data flow going into the memory, while LOSP-Out would pertain to the data flow leaving the memory. Similar terminology is used on the HISP-In, and HISP-Out.

The function of the LOSP channel is to connect any 16-bit asynchronous device to the CRAY Y-MP. Associated with the 16 bits are 4 parity bits in which case odd parity is performed over the channel. On a LOSP input sequence, a parity error would cause all transfers to memory to be equal to zero from the time the error occurred. A LOSP-In parity error would set the Error flag, but not cause an interrupt. The I/O Interrupt flag would be set on a LOSP-In when CA = CL or when a disconnect was received from the external device. On a LOSP-Out, the parity would be generated prior to sending the 16 bits of data to the external device. An error on a LOSP-Out would be caused by a resume and the channel not being active. The I/O Interrupt flag on LOSP-Out would be caused from CA = CL or the Error flag being set.

Hardware Trng.
YM02/01 J.E.S.

LOSP-INPUT SEQUENCE

Clearing The Channel

The first and very important instruction to use prior to using a LOSP channel or VHISP channel is to clear the channel with an (0012) instruction. Clearing the channel will clear out the Error flag and any latches which were left in an undesirable condition from the last user of the channel. Also, clearing the channel causes the (DE) option to remember the CPU number that will be using that particular channel. The CPU number is important when determining where to send the I/O interrupt when it occurs. The I/O interrupt is directed toward the CPU that last issued an (0012) on the channel with the interrupt. However, if there is a preferred CPU to receive the I/O interrupts, the I/O interrupt would then be directed toward that CPU. This is accomplishd by setting the Select for External Interrupt (SEI) bit in the Exchange Package. If the SEI bit is not set, the I/O interrupt is directed toward the CPU which is Waiting on Semaphore (WS). Finally, if SEI or WS is not set, then the I/O interrupt is directed toward the CPU that last cleared the channel with the I/O interrupt.

Loading The CL Register

Loading the CL register is accomplished by issuing an (0011) instruction. On a deadstart or dead dump sequence, the CL register is loaded with zeroes when I/O MC rises and then drops setting CL = 00, seeing that no instruction can issue from CIP at this point. When issuing an (0011) instruction, the instruction is decoded along with the Shared registers, seeing that I/O is shared among the CPUs. The (0011) instruction, before it is allowed to issue, must first check to see if the Shared registers are in Progress or in Request, (SRREQ) or (SRPG). If the Shared registers are quiet, the (0011) will then issue and keep the Shared registers busy for 6 clock periods. The (JC) option will have decoded the (0011) instruction and then pass the control bytes and SRPG to the (JQ) option. The (JQ) will also receive Aj/Ak bits (2^4 - 2^0) which are passed along to the (JS) options. The (JQ) will also send (YR11) option the SRPG. The (YR11) will encode a CPU number from where the SRPG originated from. This is used on the (JR) option and (DE) option to identify which CPU is using the channel. The (Ak) value which is to be loaded into the appropriate CL register is sent from the (AR) option to the (JS) options.

The (JS0 - JS3) options will multiplex 2 bits of (Ak) out to all the (JR0 - JR3) options within each CPU 0 - 7. The (Aj) field of the (0011) instruction identifies which channel is to be used and is sent to all (JR) options ahead of (Ak) from the (JS). Each (JR) option will decode the (Aj) field to determine if the channel number is appropriate for their CPU, if so, the CL register is then loaded into that CPU. For example, if Aj = 36_8, then CPU 7 would respond to loading the CL register. The (JR0 - JR3), with the matching channel number, will then send (Ak) to the (DA/DB) option along with the channel commands. The channel commands are decoded on the (DA/DB) to determine what to do with (Ak), and in this case, for the (0011) instruction load (Ak) into LOSP-In CL.

Hardware Trng.
YM02/02 J.E.S.

Loading The CA Register

Loading the CA register would be accomplished by the (0010) instruction, except on a deadstart or dead dump sequence in which case CA would be forced when raising and lowering I/O MC to zero, CA = 00, seeing that no instruction could execute at this time. The loading of CA register is similar to loading the CL register, except after the CA register has been loaded the channel is set active.

LOSP-Out Reference Request

Once the channel has been set active, after loading the CA/CL registers, the (DC) option will generate a reference request to the (DA) option to begin the output sequence. The (DA), upon getting the reference request, will check for any conflicts with the HISP or VHISP channels located on the CPU module. If there are no conflicts, the (DA) will generate some I/O channel designators which are used to track the reference through memory. The (DA/DB) will then pass the CA register contents to the (YH) option and increment the CA by plus 1. The Memory address is held on the (YH) option for Subsection conflicts and Bank conflicts on Port D.

The (DA) will then pass the I/O designator and I/O reference request to the (YZ) option. The (YZ) option will set Port D active as long as fetch is not requesting Port D or already using Port D. Once fetch is done using Port D, I/O is allowed before another fetch is allowed to use Port D. The Port D reference request, then being used by I/O, would be sent to the (YK) option where an SS conflict is checked for within the same section. Along with the Port priority, Port D gets to be highest priority when I/O was held by a fetch, otherwise Port D and I/O are the lowest priority.

The Port D reference is then checked for Bank conflicts among the other CPUs after which time the controls and addresses are passed to the proper section in memory. The address which was being held on the (YH) option is passed to the (YA) option for a port and section select.

LOSP-Out Read Data Disassembly

The (YZ) continues to track the I/O read reference by delaying the I/O channel designators through the SS conflicts and Bank conflicts. When the conflicts and delays have expired, the (YZ) will supply the (DP) options with an I/O capture LOSP data being sent to the (DP) from memory. There are eight (DP) options and each receives 8 bits of data from memory. When sending the 16 bits plus 4 bits of parity to the external device, the initial 8 bits from memory must be serially swapped such that each (DP) will output 2 bits while four of the (DPs) will also output the parity bits.

The serial swap requires 14 CPs to complete during which time the data per each (DP) is sent out one bit at a time to two different (DP) options. Once the serial swap is completed, the data is correctly aligned to be sent to the external device.

The first transfer will be with parcel 0 data bit (2^{63} - 2^{48}) and 4 parity bits. Along with the data, the (DC) option will send a ready so that the external device knows that there is valid data on the wires. Once the external device has accepted the 16 bits of data, the external device sends the (DC) option a Resume signal. The Resume will increment the Parcel Counter to one and continue this sequence until the Parcel Counter is equal to three. When the parcel count equals three, the entire 64-bit data word has been sent to the external device. As long as CA $\neq$ CL, another reference request could be generated and another word addressed from memory. When CA = CL, an I/O Interrupt flag is set.

LOSP-In Write Data Assembly

When inputting data from an external device, 2 bits are brought into the eight (DJ) options along with four of the (DJs) receiving the parity bits. The 2 bits are serially swapped by being sent out one bit at a time to two other (DJs); such that in 6 CPs, the 2 bits are in their proper positions at which time odd parity is checked on each of the 4-bit groups. Parity error for any one of the four (DJs) is passed to the (DC) option. A Resume is sent to the external device once the swap is completed and the (DC) will then wait for the next ready or disconnect from the external device. Upon receiving the ready, the (DC) will increment the parcel count, wait for the swap to complete, and send another Resume to the external device. This sequence continues until the (DC) parcel count equals zero, which means the entire 64 bits of data has been assembled on the (DJ) option. When the parcel count equals zero, the (DC) can generate a reference request to the (DA) and write the word into memory. This sequence continues until CA = CL or a disconnect is received on the (DC) at which time the (DC) will generate an I/O interrupt.

CRAY Y-MP
I/O INSTRUCTION SET

**0012j1		Clear the Interrupt flag and Error flag for the channel indicated by (Aj); set device master clear (output channel); clear device ready held (input channel)
**0012j0		Clear the Interrupt flag and Error flag for the channel indicated by (Aj); clear device master clear (output channel)
**0011jk	CL, Aj Ak	Set the Limit address (CL) register for the channel indicated by (Aj) to (Ak)
**0010jk	CA, Aj Ak	Set the Current address (CA) register for channel indicated by (Aj) to (Ak) and activate the channel
033i00	Ai CI	Transmit interrupting channel number to Ai
033ij0	Ai CA, Aj	Transmit address of channel (Aj) to Ai
033ij1	Ai CE, Aj	Transmit Error flag of channel (Aj) to Ai

**Privileged instruction - Monitor mode only

CRAY Y-MP MSEE MEMORY DUMP

```
MODE  TB                                  16B CMOS/I - X   3.0      00/00/00              :MSEE
AUTO                                                               01:12:58
C  X 0000                                                  C
C                                                          C
   P   00000020-0  XA   0000  A0  00000000     1000  000000  000000  000000  000000
   BI  00000000    VL    000  A1  00000000     1001  000000  000000  000000  000000
   LI  77777700    VU      0  A2  00000000     1002  000000  000000  000000  000000
   BD  00000000    PN      0  A3  00000000     1003  000000  000000  000000  000000
   LD  77777700    PS      0  A4  00000000     1004  000000  000000  000000  000000
   FL     0 000    CN      0  A5  00000000     1005  000000  000000  000000  000000
   M0    00  21    EM      0  A6  00000000     1006  000000  000000  000000  000000
   2V     0        0EMA    1  A7  00000000     1007  000000  000000  000000  000000

   S0  000000  000000  000000  000000           20  100500  000026  100600  000025
   S1  000000  000000  000000  000000           21  030356  022421  001143  001046
   S2  000000  000000  000000  000000           22  022720  001270  033700  031074
   S3  000000  000000  000000  000000           23  011000  000111  001240  006000
   S4  000000  000000  000000  000000           24  000105  000000  000000  000000
   S5  000000  000000  000000  000000           25  000000  000000  000000  000000
   S6  000000  000000  000000  000000           26  000000  000000  000000  004000
   S7  000000  000000  000000  000000           27  000000  000000  000000  000000

ENTER      :SN?
```

1. Transfer memory locations 4000_8 - 10000_8 to the MCU image area.

2. Can MSEE be used as a simple monitor?

YM03/05A

LOSP Channel (Options Involved)

(DA0) Option

The (DA) option is the I/O reference control between the channel and memory. The (DA) will check for priority conflicts between the LOSP, HISP, and VHISP reference requests. When a conflict exists, the LOSP has highest priority followed by the HISP and then the VHISP. The (DA) also contains the lower 5 bits of the CA, CL registers for LOSP-In, LOSP-Out, HISP-In, HISP-Out and VHISP CA registers. The (DA) option contains the lower 5 bits (2^4 - 2^0) of the I/O address which are the section and subsection selects for the memory controls. The (DA) receives channel commands from the (JR) via the (AF) option, which will be decoded to determine what channel is active; and therefore, what CA register to send to memory.

(DB0 - DB2) Options

Each (DB) option contains 8 bits of the I/O address, for LOSP-In, LOSP-Out, HISP-In, HISP-Out and VHISP CA registers similar to the (DA) option. (DB0) handles I/O address bit (2^{12} - 2^5), (DB1) handles I/O address bits (2^{20} - 2^{13}), and (DB2) handles address bits (2^{28} - 2^{21}). (DB2) however, sends only (2^{27} - 2^{21}) to the (YH) option. After each reference, the CA register is incremented by plus 1 for the next reference. The (DA/DB) also checks to see if CA = CL for the bits that the (DA/DB) handles and if so, the CA = CL signal is sent to the (DC) option to see if all the bits of CA are equal to CL.

(DC0) Option

The (DC) option is the LOSP controller, different from the (DA) option in that the (DC) communicates with the external device, where as the (DA) communicates with the memory controls. The (DC) resync, the control signals for the LOSP channel which are Ready, Resume, Disconnect, Console Interrupt, I/O MC, CPU MC, and Deadstart. When an I/O interrupt exists, the (DC) will send an interrupt to the (DE) option. I/O interrupts can be caused by CA = CL, Error flag set, or a Disconnect. The (DC) controls the loading and unloading of the parcel data from the (DJ/DP) options. When assembling a 64-bit word, the (DC) will supply the (DJs) with a Sample Data signal each time a parcel of data is valid, or a Data Out Advance to the (DP) options when disassembling a 64-bit word. On an LOSP-In operation, the (DC) will track the number of reference requests with the number of reference acknowledges, such that the reference request will increment a counter by plus 1 and the reference acknowledges will decrement the counter by minus 1. The counter must be equal to zero before the LOSP-In can cause an I/O interrupt. The (DC) also contains a sample Hold-off Counter which will hold off on the Sample LOSP-In Data to the (DJ) until the LOSP data from the external device has been swapped into its proper position. When the parcel count equals zero and sample hold-off equals a count of five, the (DC) will

generate a reference request to the (DA) option. On an LOSP-Out, sample hold-off is not used. However, a swap cycle timer is used which starts counting when the data is loaded into the (DPs) from memory, and when the swap cycle equals 14_8, the data is in the proper position to be sent to the external device.

(DE0) Option

The (DE) options main function is the VHISP controller, however, an added function is to handle the LOSP and VHISP controls after the LOSP or VHISP channel has completed the input/output operation. One of the functions the (DE) handles is sending the interrupt request to the (HH) option along with the CPU number that was using the interrupted channel. The I/O interrupt is serially gated along with the CPU number to the (HH) option. The (DE) also initiates or generates the controls for the (033) instruction which are to read the interrupting channel number, CA or status word, and the Error flag.

(DJ0 - DJ7) Options

The purpose of the (DJ) options are to assemble a 64-bit word from the 16 bits being sent by the external device. The 16-bit initially would come in 2 bits per (DJ) while (DJ4 - DJ7) would receive 3 bits, the extra bit being parity. The 2 data bits are serially swapped across two other (DJs) in 6 CPs until the data bits are aligned into their proper position to be sent to memory.

The (DJ) receives two major control signals which are LOSP-In Sample Data and Go LOSP Data. Sample Data tells the (DJ), which was sent by the (DC) option, that the data is valid on the (DJs); while Go LOSP Data tells the (DJs) to send their assembled 64-bit word to the (YR) option for check bit generation.

(DP0 - DP7) Options

The (DP) options are similar to the (DJs) except the (DPs) would take a 64-bit word from memory and divide the bits up into 2 bits per (DP) for 4 parcels. Once the data arrives from memory, the (DP) will latch the data upon receiving the I/O capture LOSP data from the (YZ) option. The (DP) will then serially swap the data over 14 CPs, at which time the data would be in the proper position to send parcel 0 and the 4 parity bits to the external device upon receiving the LOSP-Out Data Out Advance from the (DC) option.

(HH0) Option

The (HH) option is responsible for determining which CPU to direct the I/O interrupt to, which is dependent upon if there is an SEI bit set on a particular CPU or if a CPU is waiting for a semaphore WS. If there is no preference, then the CPU that last cleared the interrupting channel would get the interrupt. Once the CPU has been interrupted, the CPU would want to know which channel number has interrupted so a (033i00) instruction is issued. When the (HH) sent the I/O interrupt, it also sent out the lowest channel number to the (JR) option, such that when the CPU read the interrupting channel number, this value could be gated out of the (JR) options to the (JS) options and finally loaded into the A register.

(JR0 - JR3) Options

The (JR) options are responsible for multiplexing the CA/status word or Error flag from the (DA/DB) out to the (JS) options. When loading the CA register on a (0010/0011) instruction, the (Ak) data is sent to all (JR) options across all the CPUs. Prior to sending (Ak), (Aj) was sent which contains the channel number. If the channel number does not match the physical CPU location that handles that channel number, the (Ak) value is not used.

(JS0 - JS1) Options

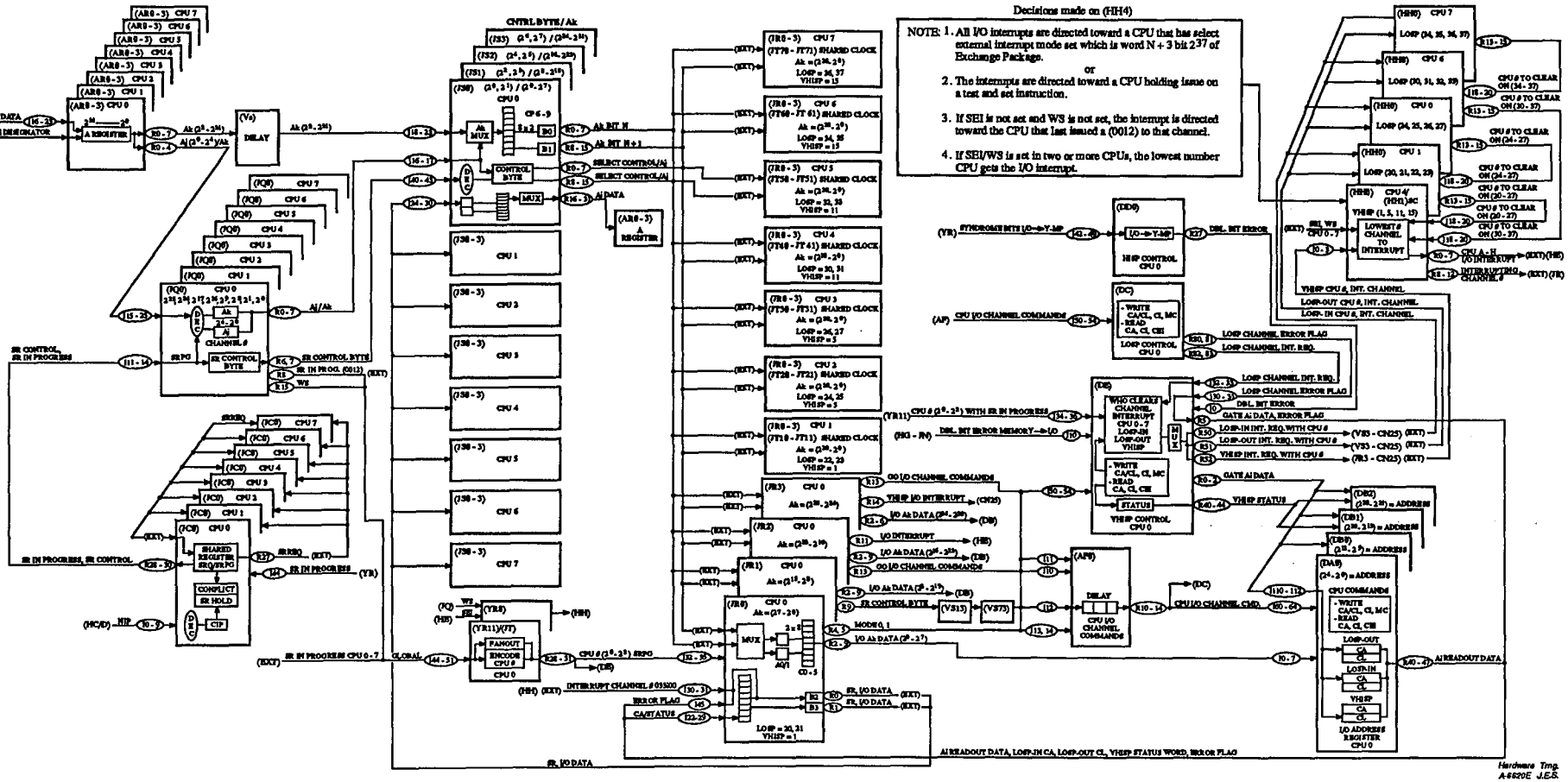
The (JS) options are responsible for getting the channel commands (Aj) and (Ak) to all the CPUs and to route the CA/status and Error flag to the (Ai) registers.

(JC0) Option

The (JC) option decodes the channel instruction at CIP time and checks to see if the Shared registers are in progress or any other CPU requesting the Shared registers. If not, the (JC) will set the Shared registers in progress for 6 CPs.

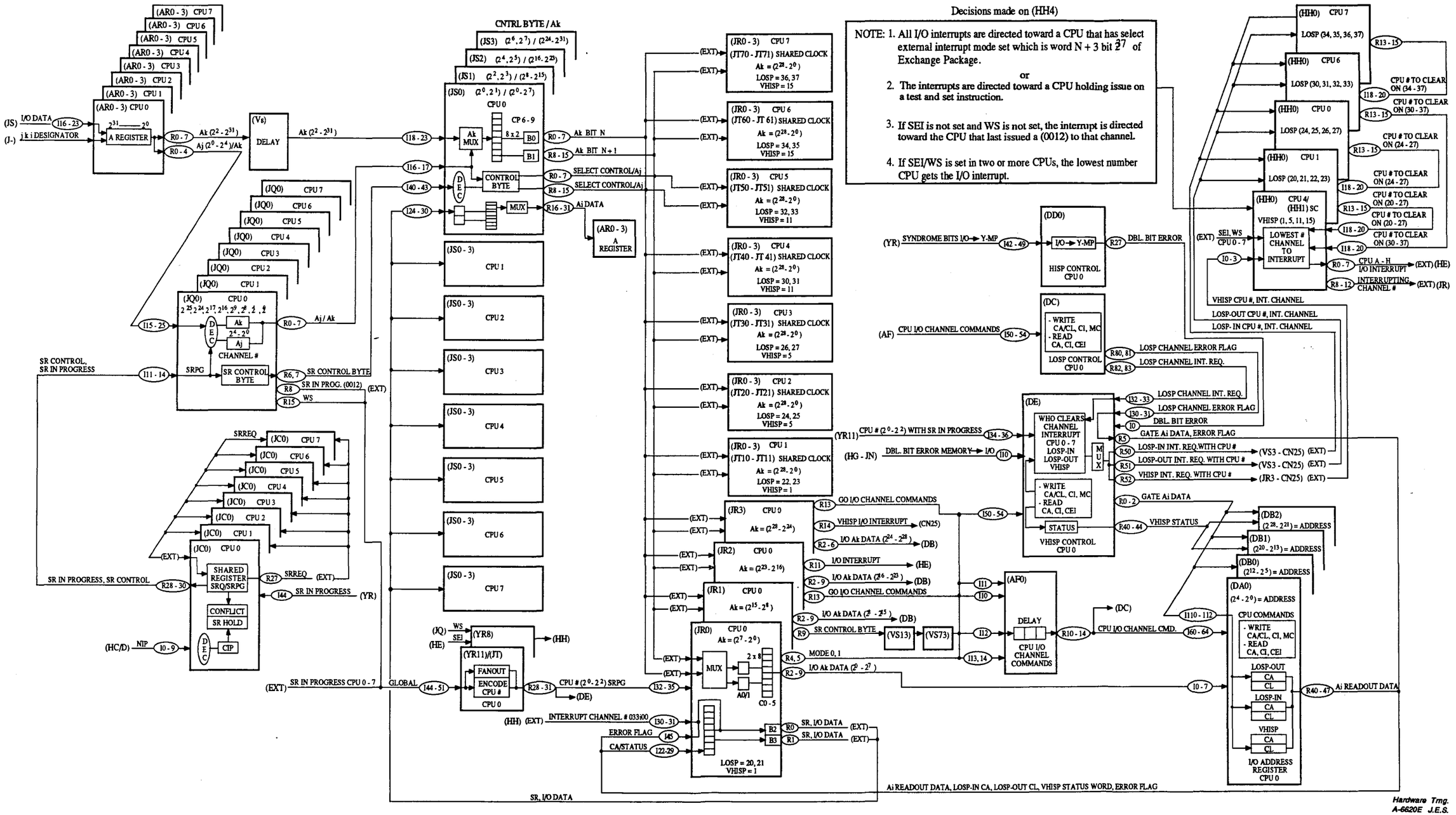
(JQ0) Option

The (JQ) option will pass the lower 5 bits of Aj/Ak to the (JS) options along with the Shared register byte controls.



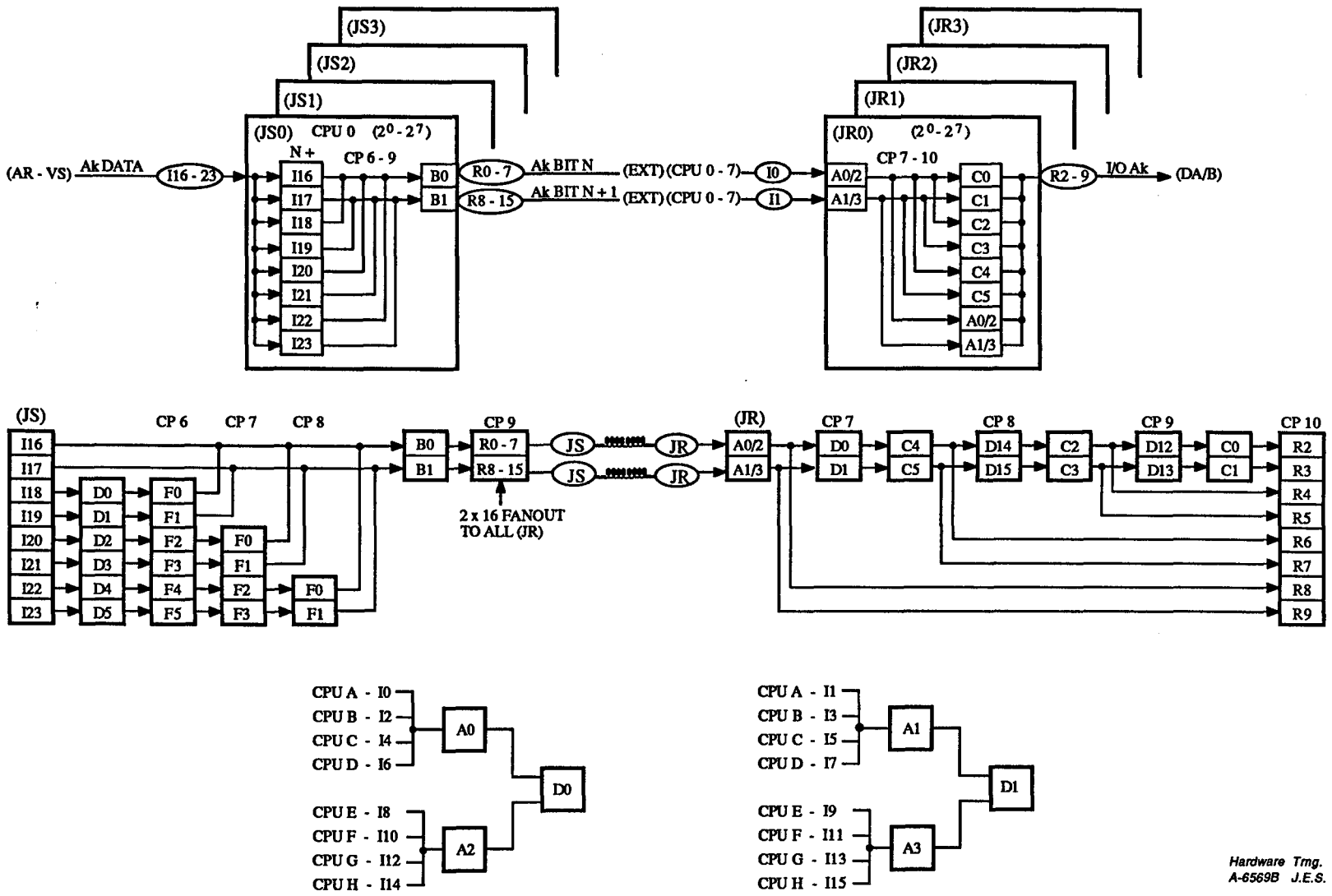
CRAY Y-MP LOADING CA/CL REGISTERS DS/DD, CLEAR CONTROL (0010 - 0011 - 0012) INSTRUCTION
READ INTERRUPTING CHANNEL NUMBER, CA, ERROR FLAG (033i00, 033ij0, 033ij1)

CRAY PROPRIETARY

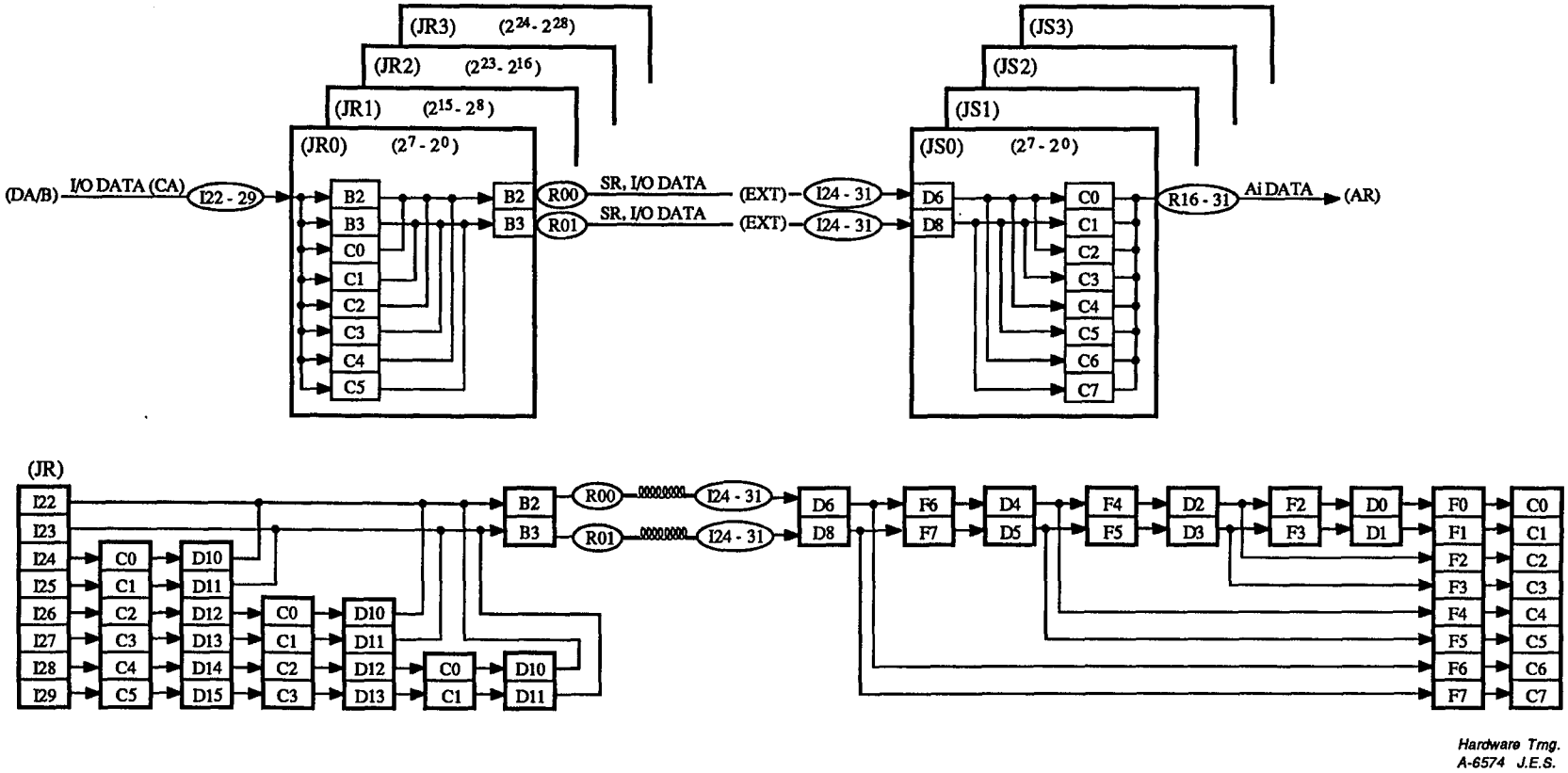


CRAY Y-MP LOADING CA/CL REGISTERS DS/DD, CLEAR CONTROL (0010 - 0011 - 0012) INSTRUCTION READ INTERRUPTING CHANNEL NUMBER, CA, ERROR FLAG (033i00, 033ij0, 033ij1)

CRAY PROPRIETARY

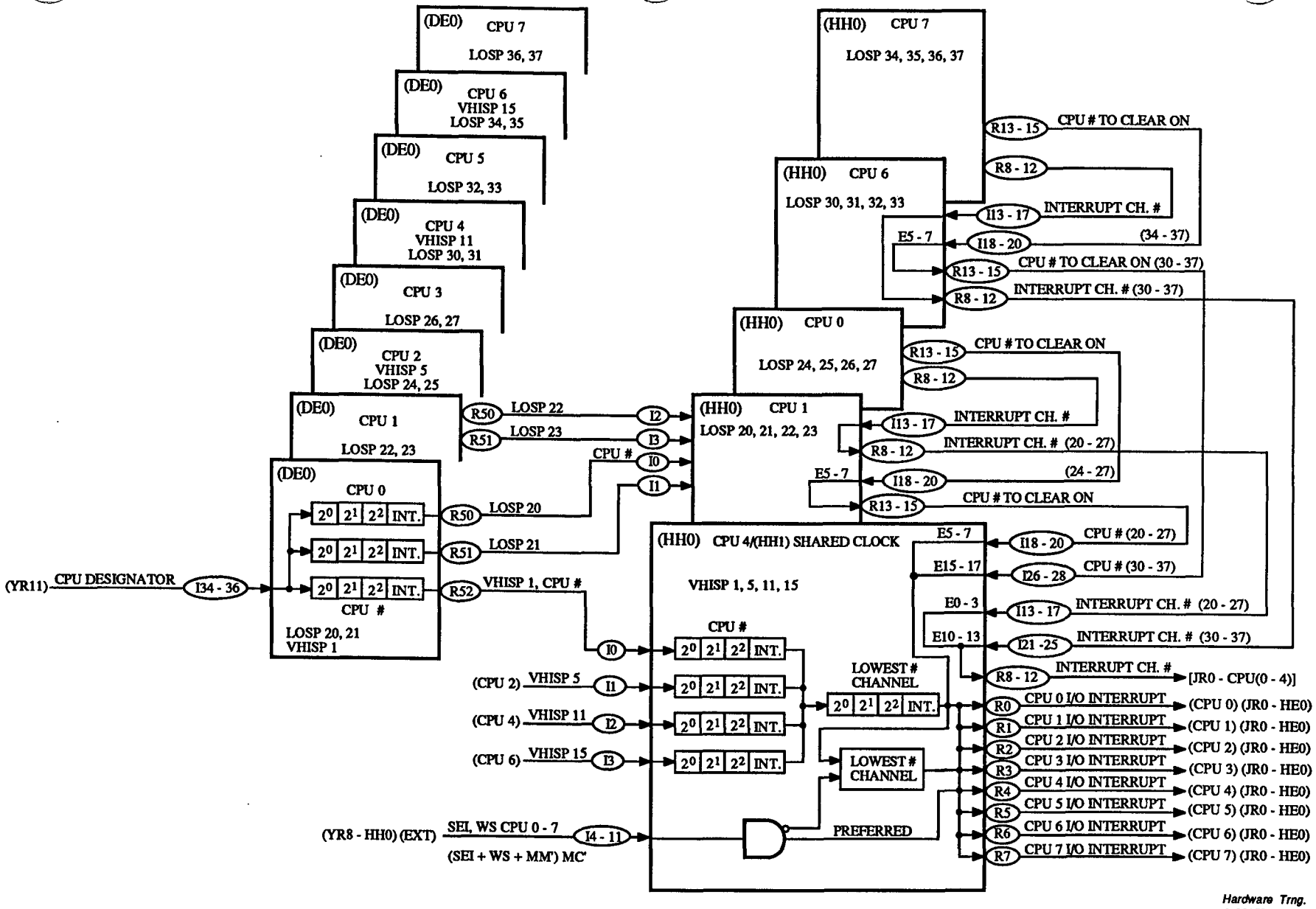


CRAY Y-MP (JS) - (JR) Ak DATA ASSEMBLY/DISASSEMBLY
FOR (0010, 0011) (0012) INSTRUCTIONS



Hardware Trng.
A-6574 J.E.S.

CRAY Y-MP (JR) - (JS) Ai DATA ASSEMBLY/DISASSEMBLY
FOR (033) INSTRUCTION



CRAY Y-MP (HH4) LOWEST INTERRUPTING CHANNEL

LOSP-IN I/O INTERRUPT

(DE0)	CPU 0	CN25	R4	CPU 1	CN25	I10	LOSP 20	I/O INTERRUPT	(HH0)	I0	CPU 1
	CPU 1	CN25	R4	CPU 1	CN25	I12	LOSP 22	I/O INTERRUPT	(HH0)	I2	CPU 1
	CPU 2	CN25	R4	CPU 0	CN25	I10	LOSP 24	I/O INTERRUPT	(HH0)	I0	CPU 0
	CPU 3	CN25	R4	CPU 0	CN25	I12	LOSP 26	I/O INTERRUPT	(HH0)	I2	CPU 0
	CPU 4	CN25	R4	CPU 6	CN25	I10	LOSP 30	I/O INTERRUPT	(HH0)	I0	CPU 6
	CPU 5	CN25	R4	CPU 6	CN25	I12	LOSP 32	I/O INTERRUPT	(HH0)	I2	CPU 6
	CPU 6	CN25	R4	CPU 7	CN25	I10	LOSP 34	I/O INTERRUPT	(HH0)	I0	CPU 7
	CPU 7	CN25	R4	CPU 7	CN25	I12	LOSP 36	I/O INTERRUPT	(HH0)	I2	CPU 7

LOSP-OUT I/O INTERRUPT

(DE0)	CPU 0	CN25	R3	CPU 1	CN25	I11	LOSP 21	I/O INTERRUPT	(HH0)	I1	CPU 1
	CPU 1	CN25	R3	CPU 1	CN25	I13	LOSP 23	I/O INTERRUPT	(HH0)	I3	CPU 1
	CPU 2	CN25	R3	CPU 0	CN25	I11	LOSP 25	I/O INTERRUPT	(HH0)	I1	CPU 0
	CPU 3	CN25	R3	CPU 0	CN25	I13	LOSP 27	I/O INTERRUPT	(HH0)	I3	CPU 0
	CPU 4	CN25	R3	CPU 6	CN25	I11	LOSP 31	I/O INTERRUPT	(HH0)	I1	CPU 6
	CPU 5	CN25	R3	CPU 6	CN25	I13	LOSP 33	I/O INTERRUPT	(HH0)	I3	CPU 6
	CPU 6	CN25	R3	CPU 7	CN25	I11	LOSP 35	I/O INTERRUPT	(HH0)	I1	CPU 7
	CPU 7	CN25	R3	CPU 7	CN25	I13	LOSP 37	I/O INTERRUPT	(HH0)	I3	CPU 7

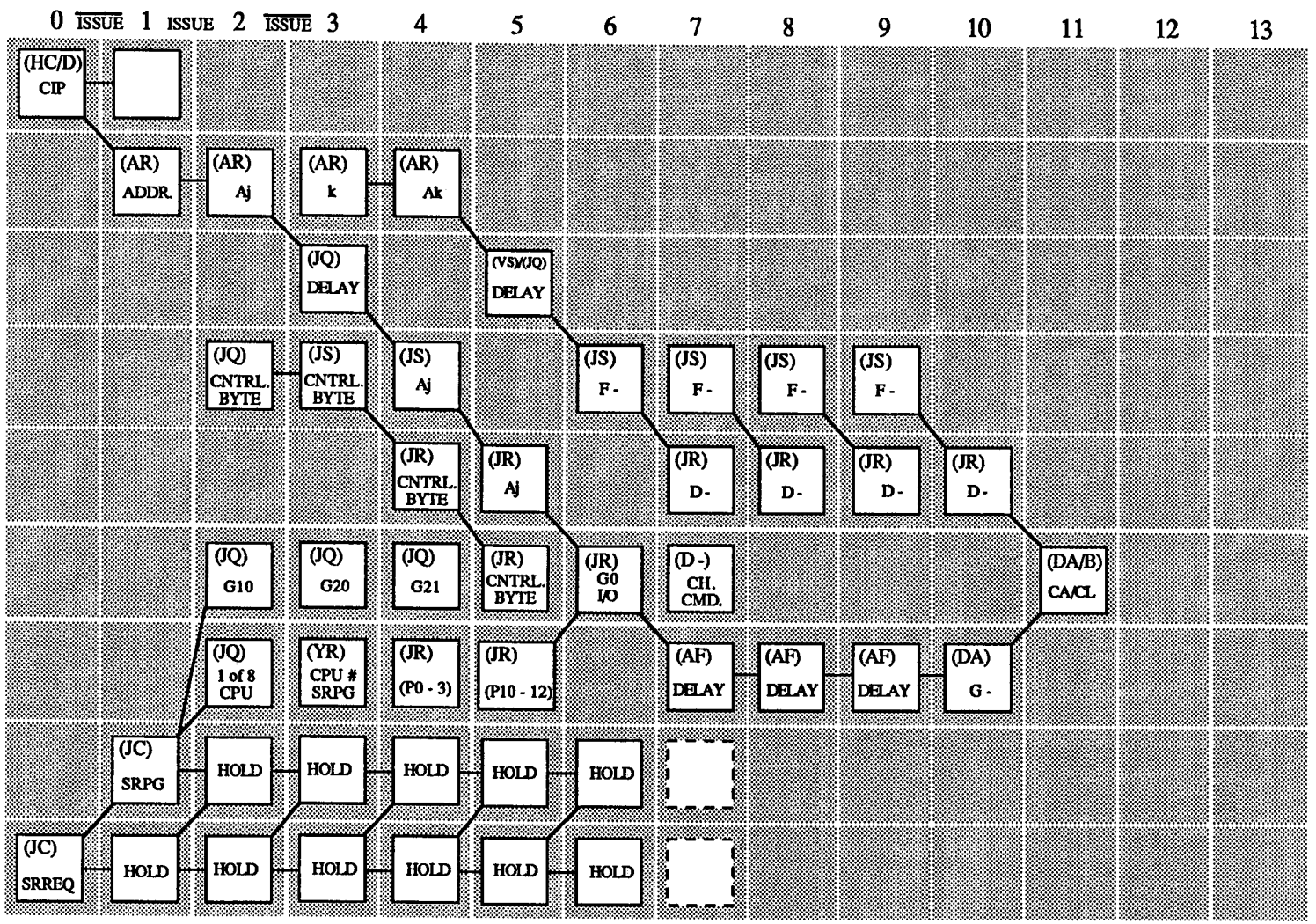
VHISP I/O INTERRUPT

(DE0)	CPU 0	CN25	R2	CPU 4	CN25	I10	VHISP 1	I/O INTERRUPT	(HH0)	I0	CPU 4
	CPU 2	CN25	R2	CPU 4	CN25	I11	VHISP 5	I/O INTERRUPT	(HH0)	I1	CPU 4
	CPU 4	CN25	R2	CPU 4	CN25	I12	VHISP 11	I/O INTERRUPT	(HH0)	I2	CPU 4
	CPU 6	CN25	R2	CPU 4	CN25	I13	VHISP 15	I/O INTERRUPT	(HH0)	I3	CPU 4

Hardware Trng.
A-6610A J.E.S.

CRAY PROPRIETARY

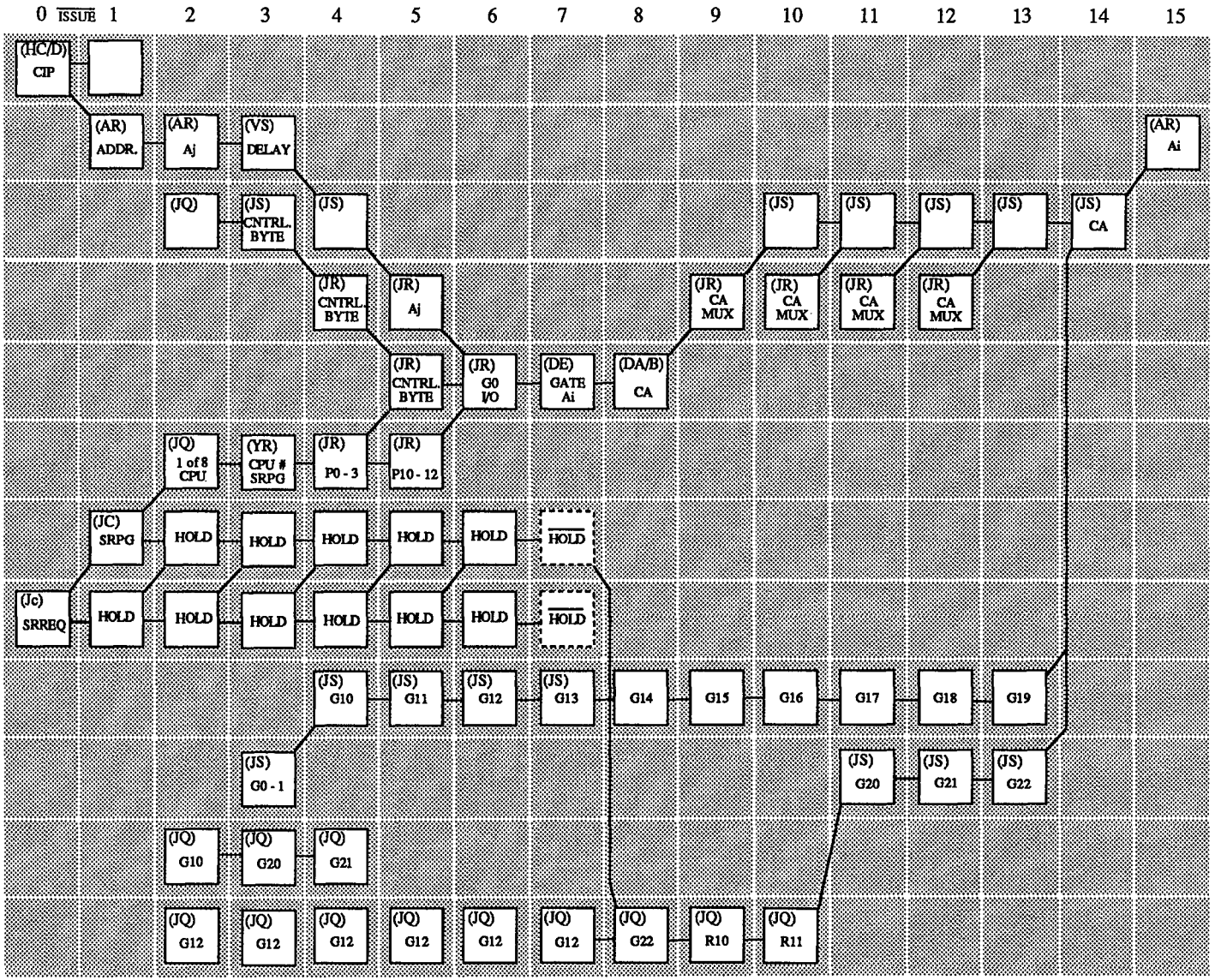
CRAY Y-MP SERIAL INTERRUPT



0010jk - SET CHANNEL (Aj) CA REGISTER TO (Ak) AND BEGIN I/O SEQUENCE
0011jk - SET CHANNEL (Aj) CL REGISTER TO (Ak)
0012j0 - CLEAR CHANNEL (Aj) INTERRUPT AND ERROR FLAGS

Hardware Trng.
A-6584B J.E.S.

CRAY Y-MP SHARED REGISTER I/O CONTROLS



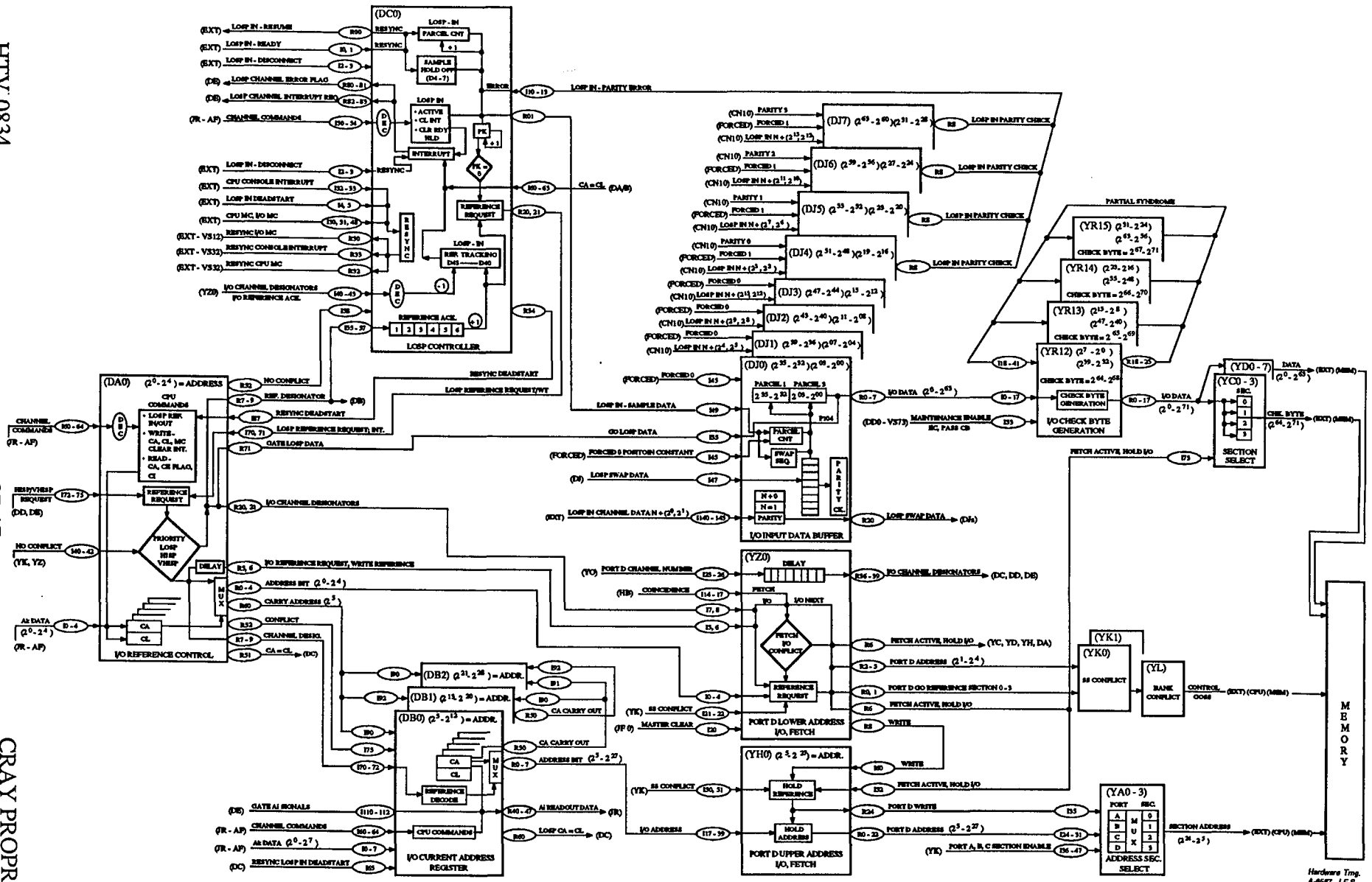
033ij0 - ADDRESS OF CHANNEL (Aj) TO Ai
033ij1 - ERROR FLAG OF CHANNEL (Aj) to Ai

Hardware Trng. A-6585 J.E.S.

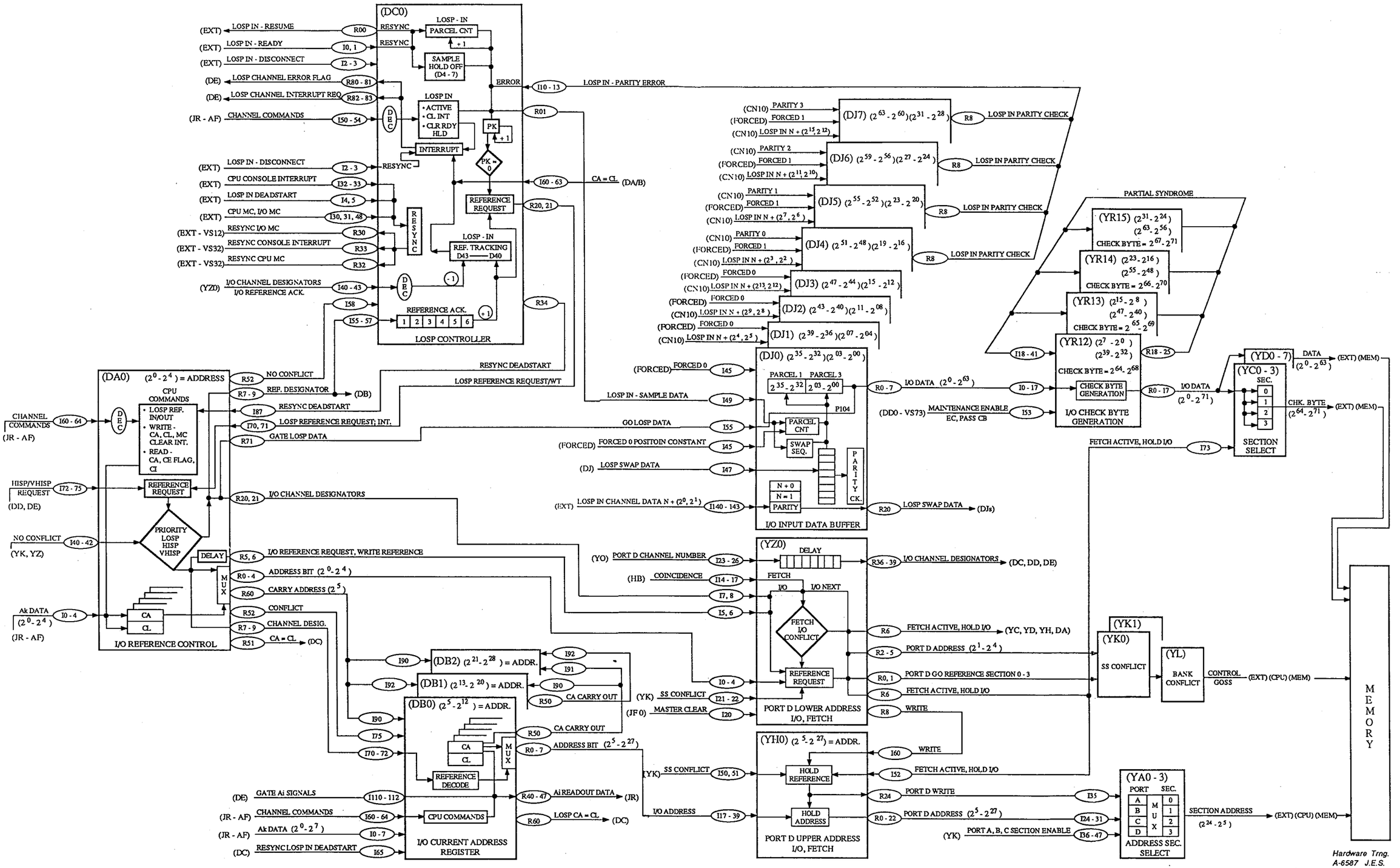
CRAY Y-MP SHARED REGISTER I/O CONTROL

CRAY PROPRIETARY

CRAY PROPRIETARY

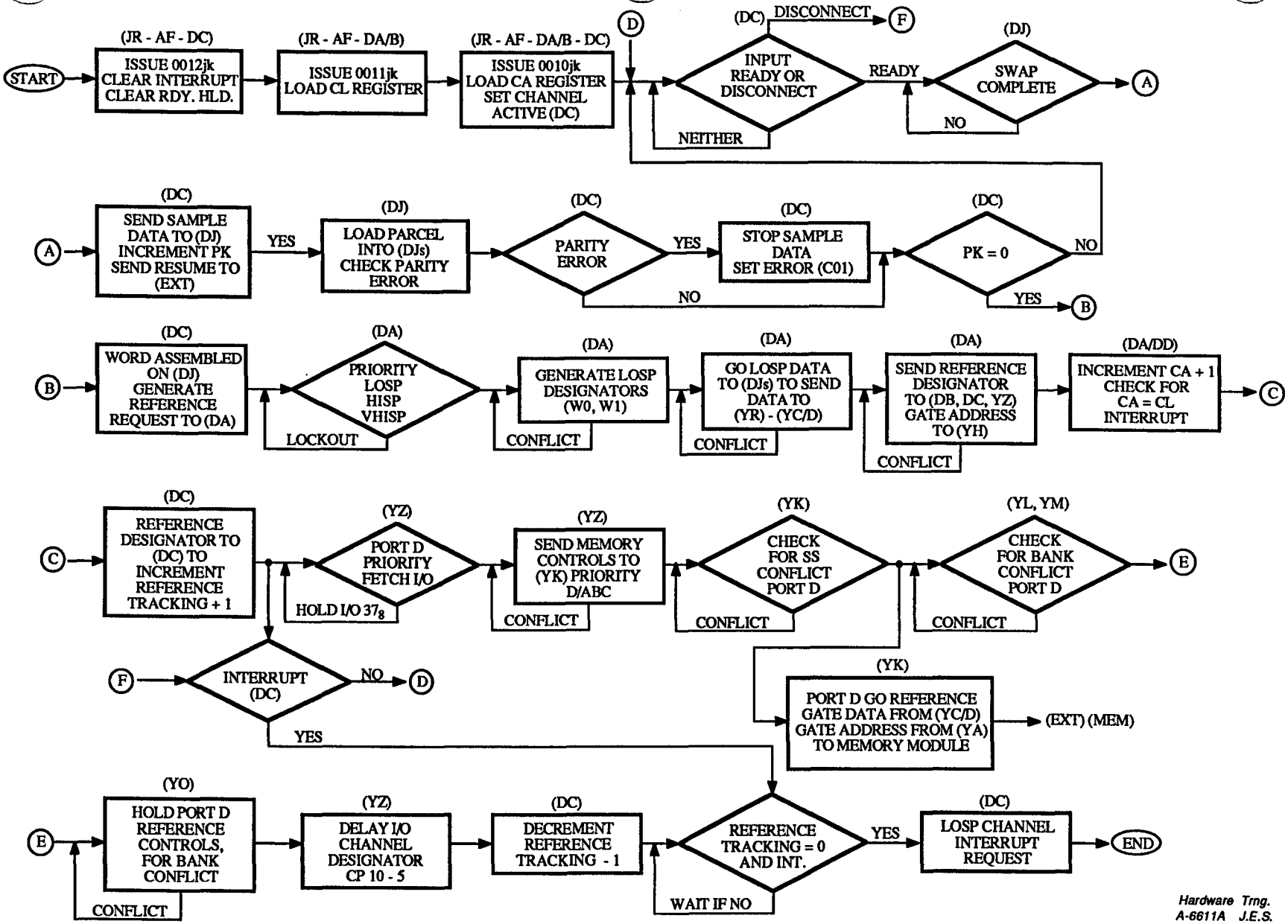


CRAY Y-MP LOSP - IN BLOCK DIAGRAM



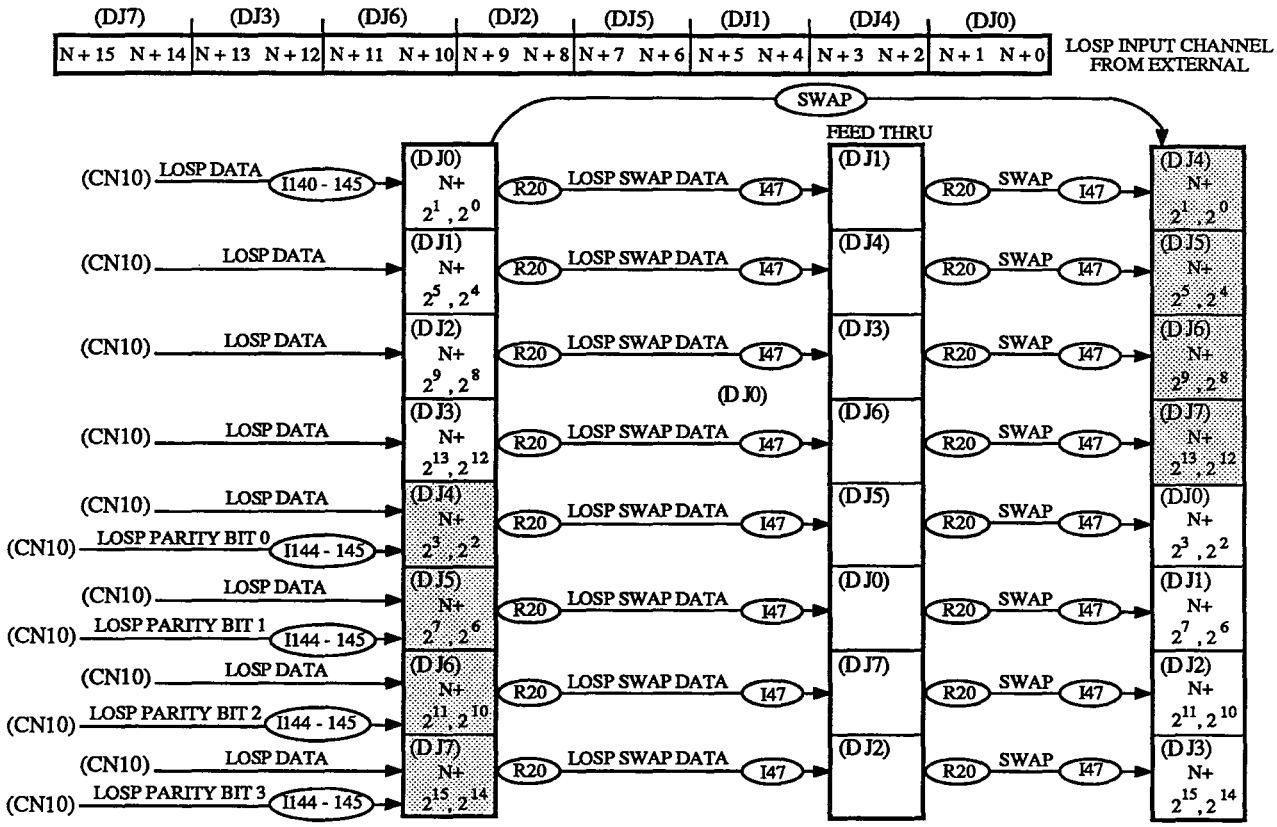
CRAY Y-MP LOSP-IN BLOCK DIAGRAM

CRAY PROPRIETARY



CRAY Y-MP LOSP-IN FLOW CHART

CRAY PROPRIETARY

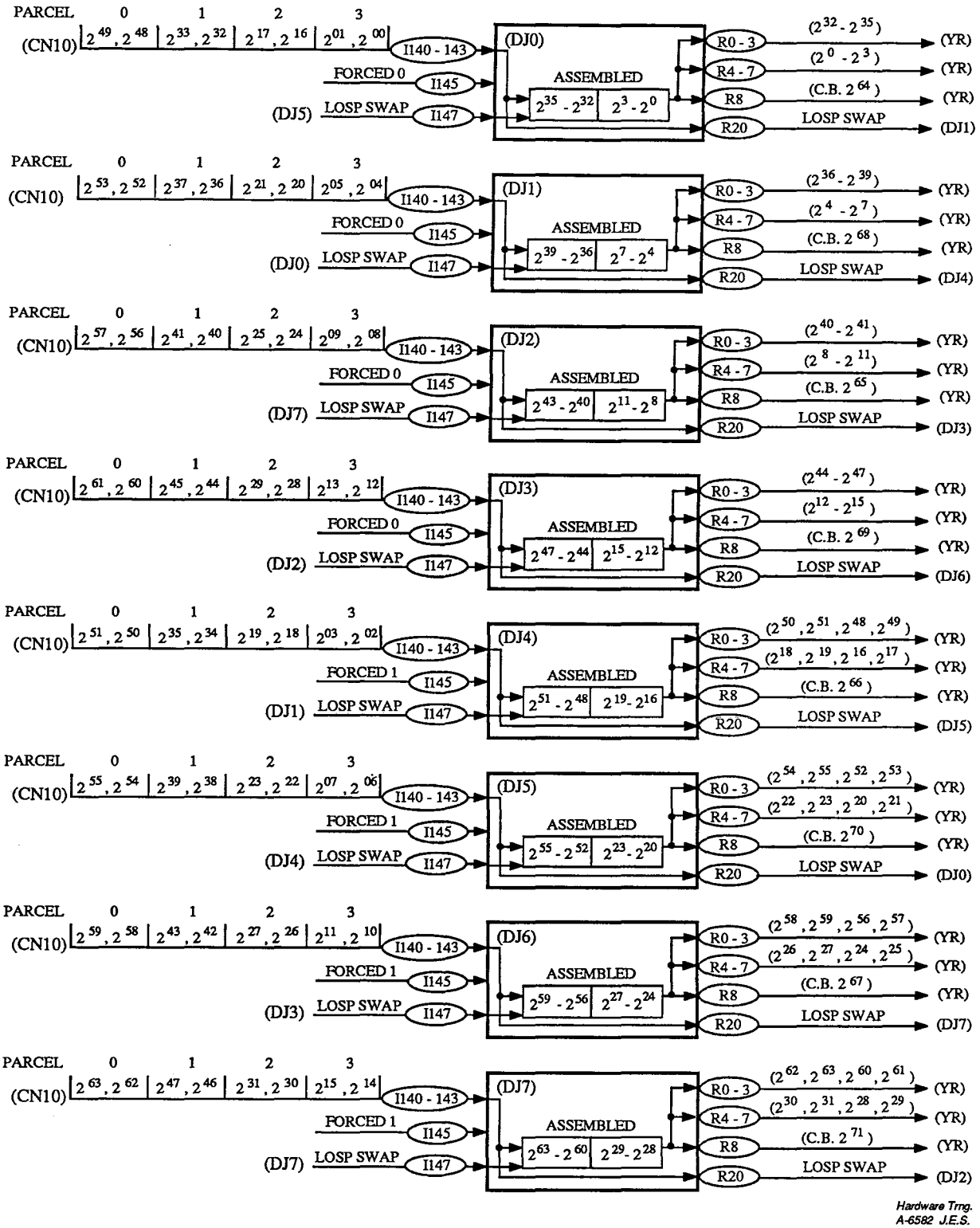


RESULTS ASSEMBLED

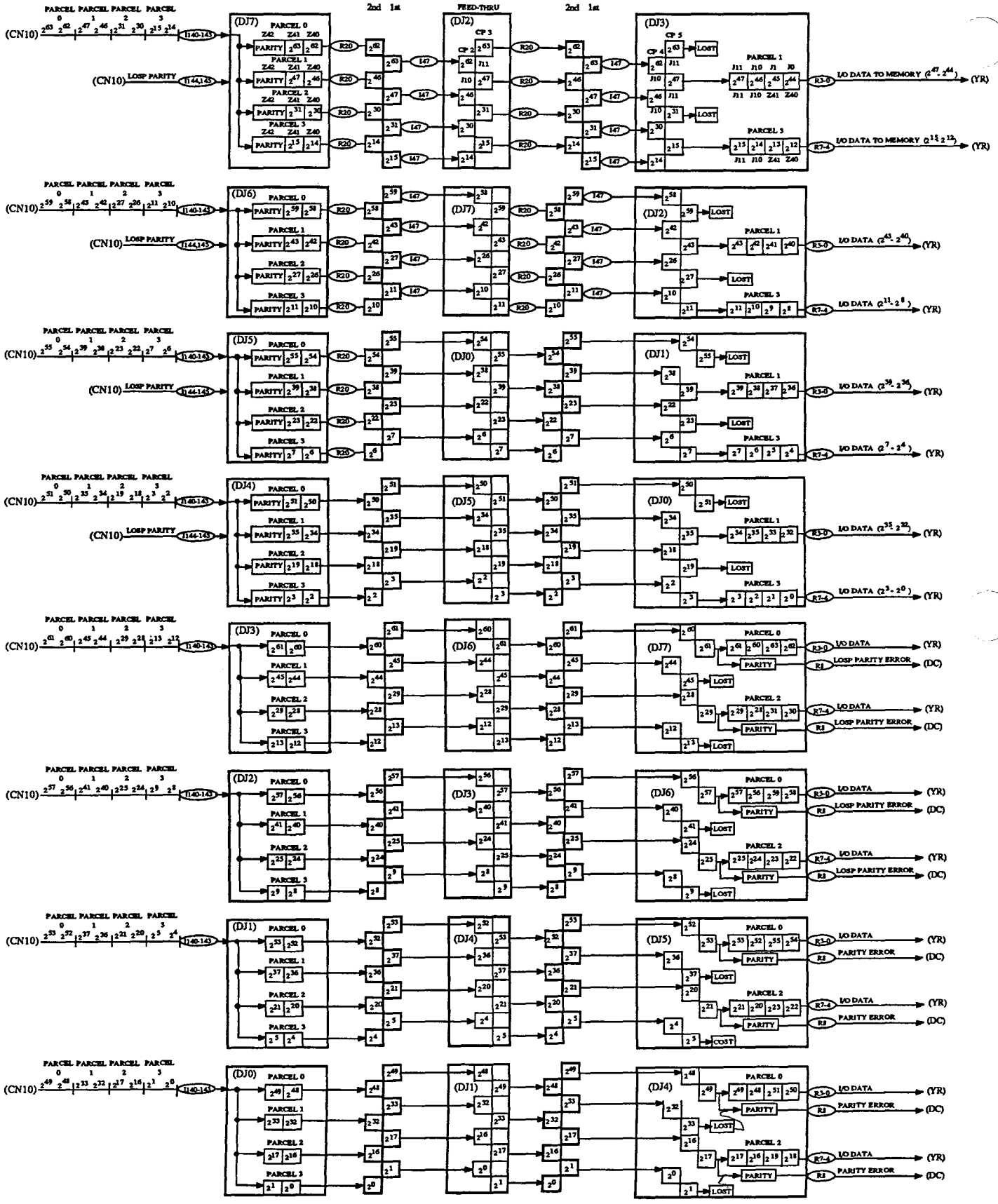
(DJ7)	(DJ6)	(DJ5)	(DJ4)	
2^{63} 2^{62} 2^{61} 2^{60}	2^{59} 2^{58} 2^{57} 2^{56}	2^{55} 2^{54} 2^{53} 2^{52}	2^{51} 2^{50} 2^{49} 2^{48}	PARCEL 0
(DJ3)	(DJ2)	(DJ1)	(DJ0)	
2^{47} 2^{46} 2^{45} 2^{44}	2^{43} 2^{42} 2^{41} 2^{40}	2^{39} 2^{38} 2^{37} 2^{36}	2^{35} 2^{34} 2^{33} 2^{32}	PARCEL 1
(DJ7)	(DJ6)	(DJ5)	(DJ4)	
2^{31} 2^{30} 2^{29} 2^{28}	2^{27} 2^{26} 2^{25} 2^{24}	2^{23} 2^{22} 2^{21} 2^{20}	2^{19} 2^{18} 2^{17} 2^{16}	PARCEL 2
(DJ3)	(DJ2)	(DJ1)	(DJ0)	
2^{15} 2^{14} 2^{13} 2^{12}	2^{11} 2^{10} 2^{09} 2^{08}	2^{07} 2^{06} 2^{05} 2^{04}	2^{03} 2^{02} 2^{01} 2^{00}	PARCEL 3

Hardware Trng. A-6580 J.E.S.

CRAY Y-MP LOSP INPUT DATA ASSEMBLY

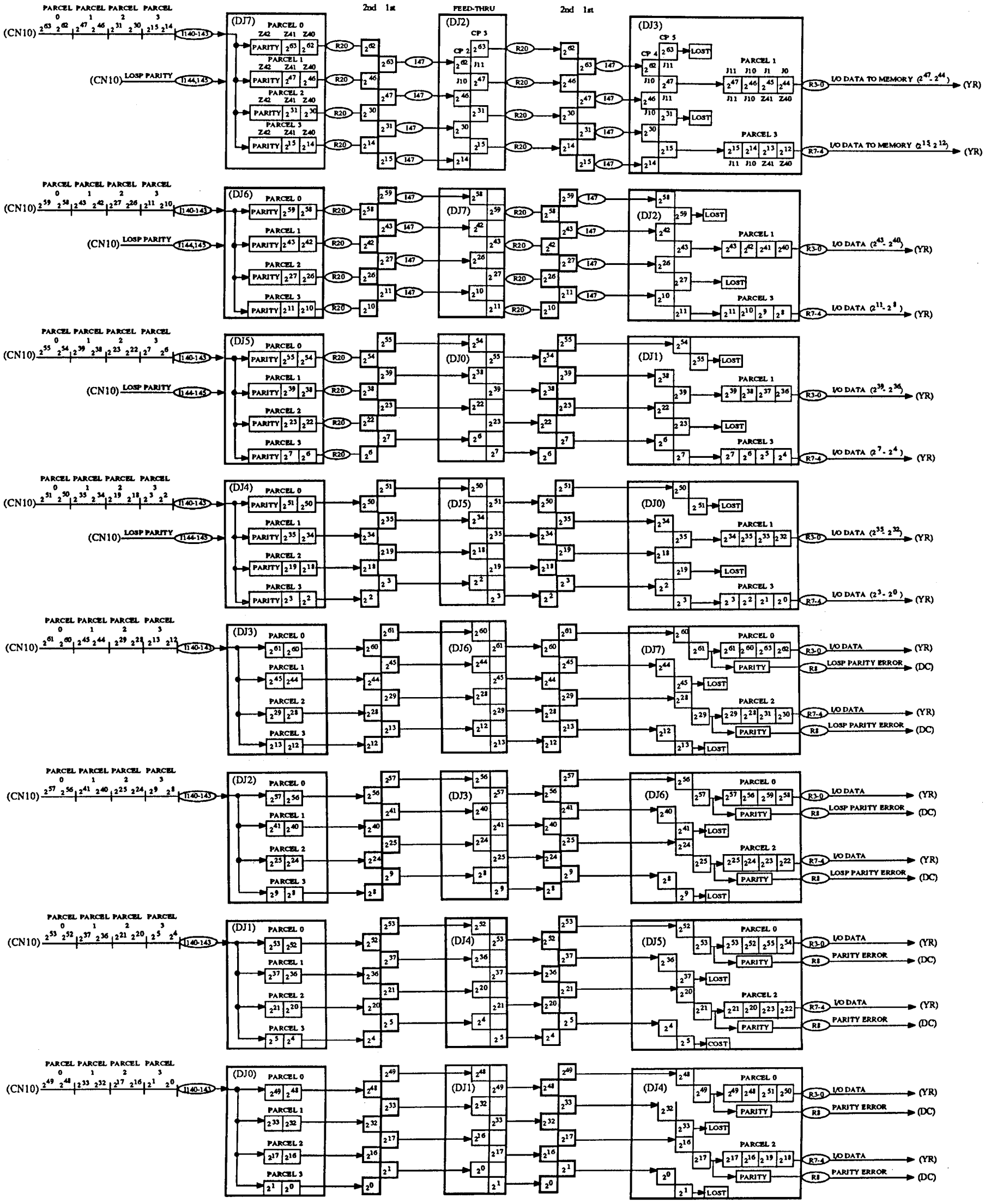


CRAY Y-MP LOSP-IN DATA ASSEMBLY



Hardware Trng.
A-6619 J.E.S.

CRAY Y-MP LOSP-IN SWAPPED DATA

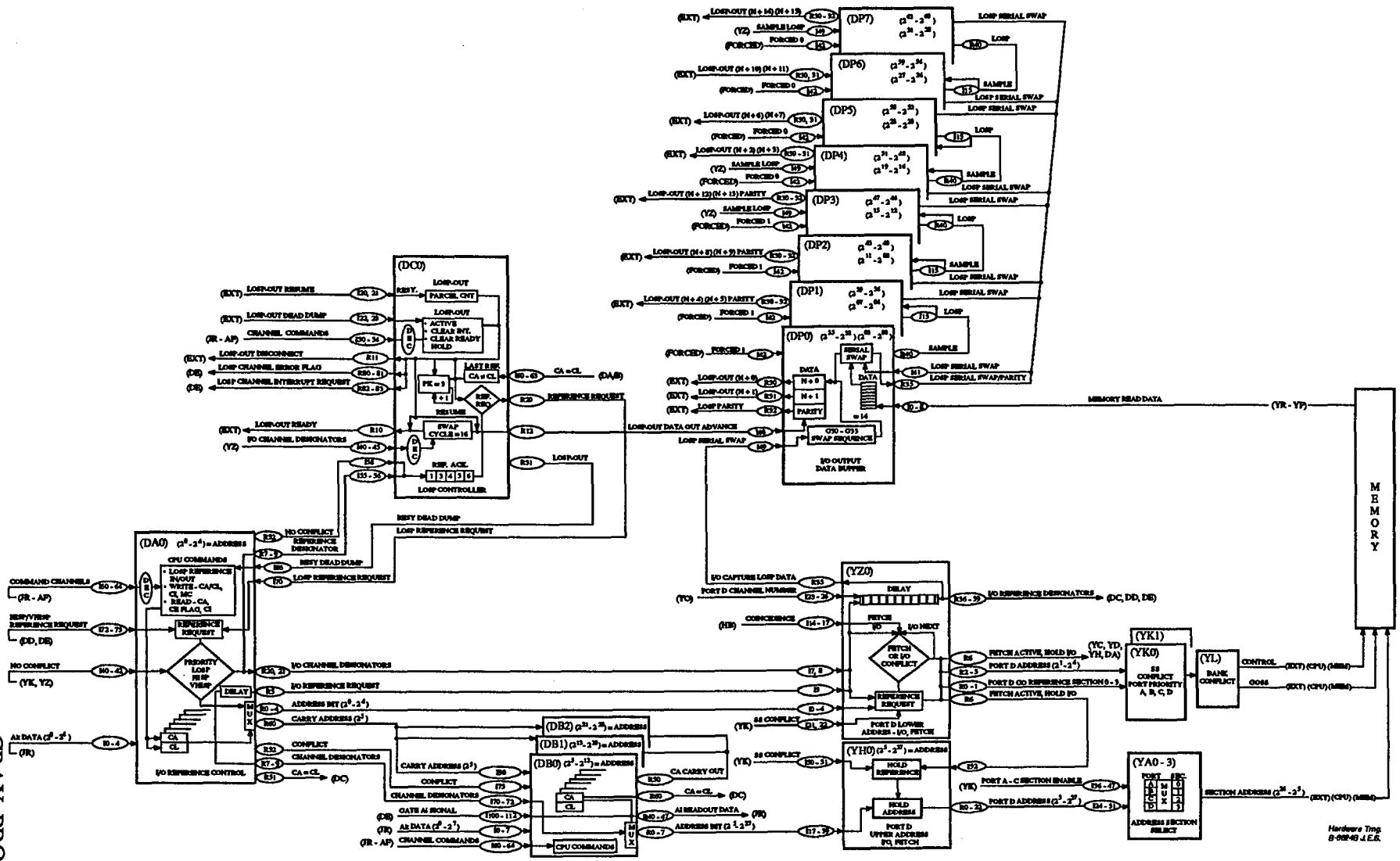


CRAY Y-MP LOSP-IN SWAPPED DATA

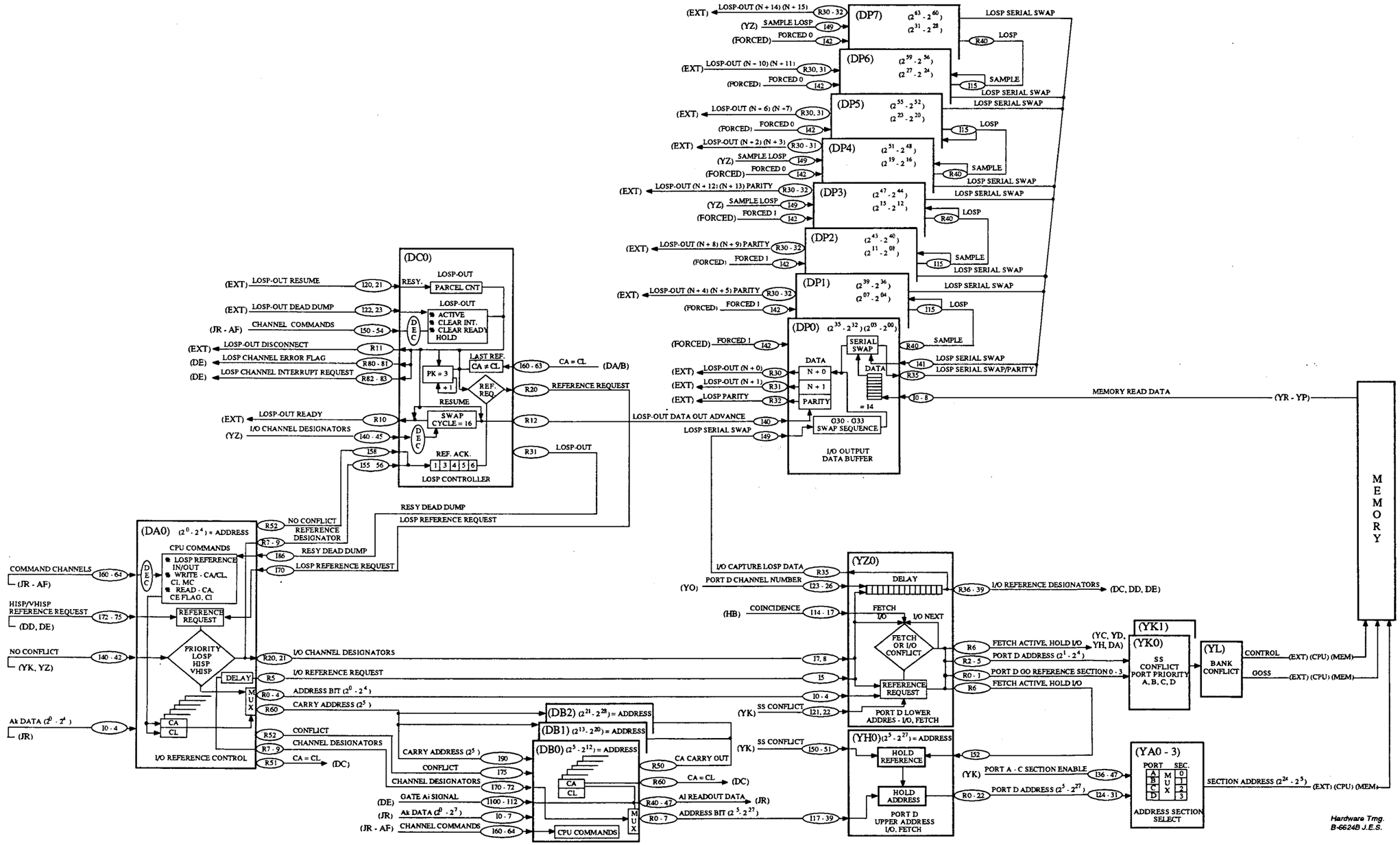
Hardware Trng.
A-6619 J.E.S.

CRAY PROPRIETARY

CRAY PROPRIETARY

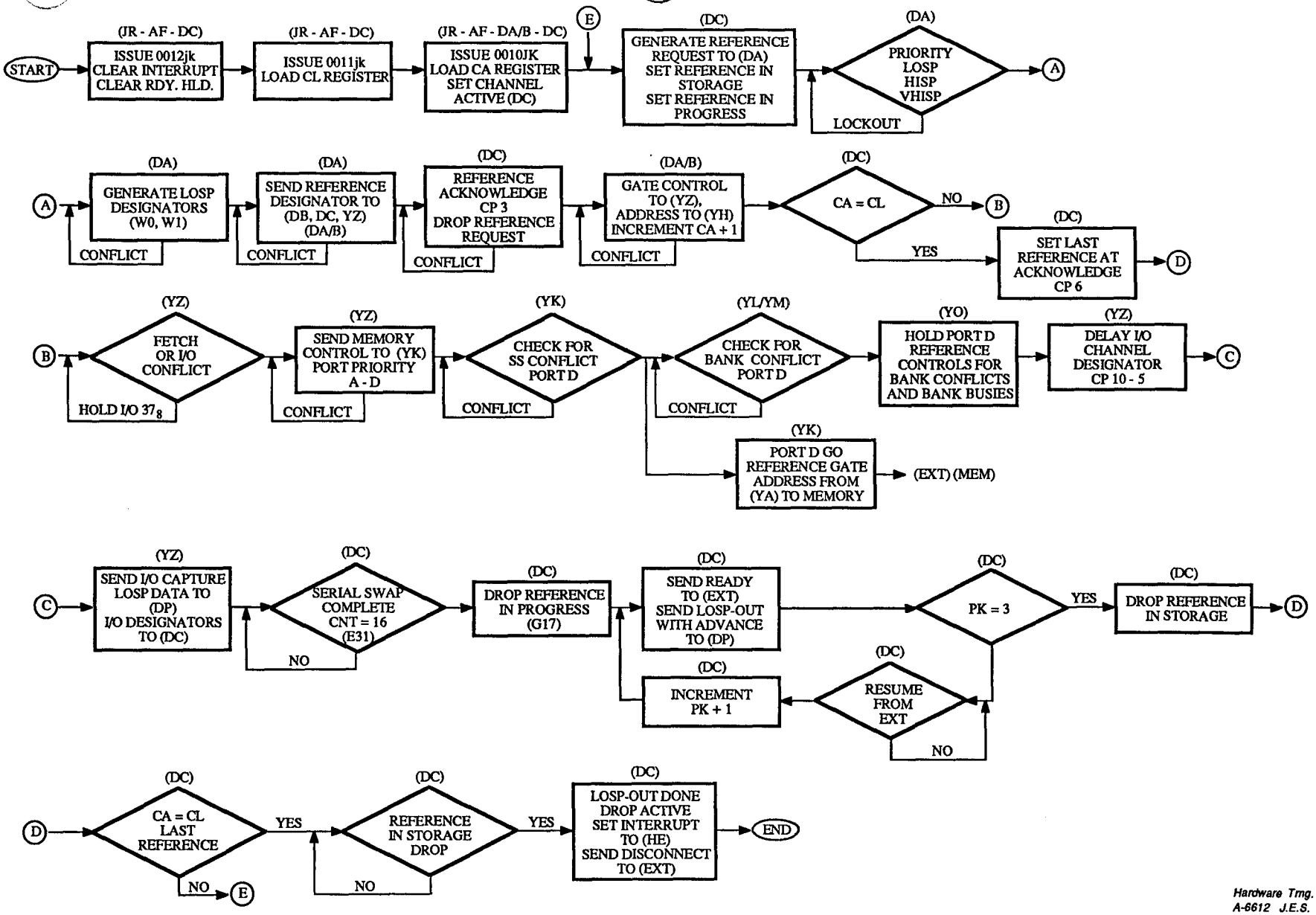


CRAY Y-MP LOSP-OUT BLOCK DIAGRAM



CRAY Y-MP LOSP-OUT BLOCK DIAGRAM

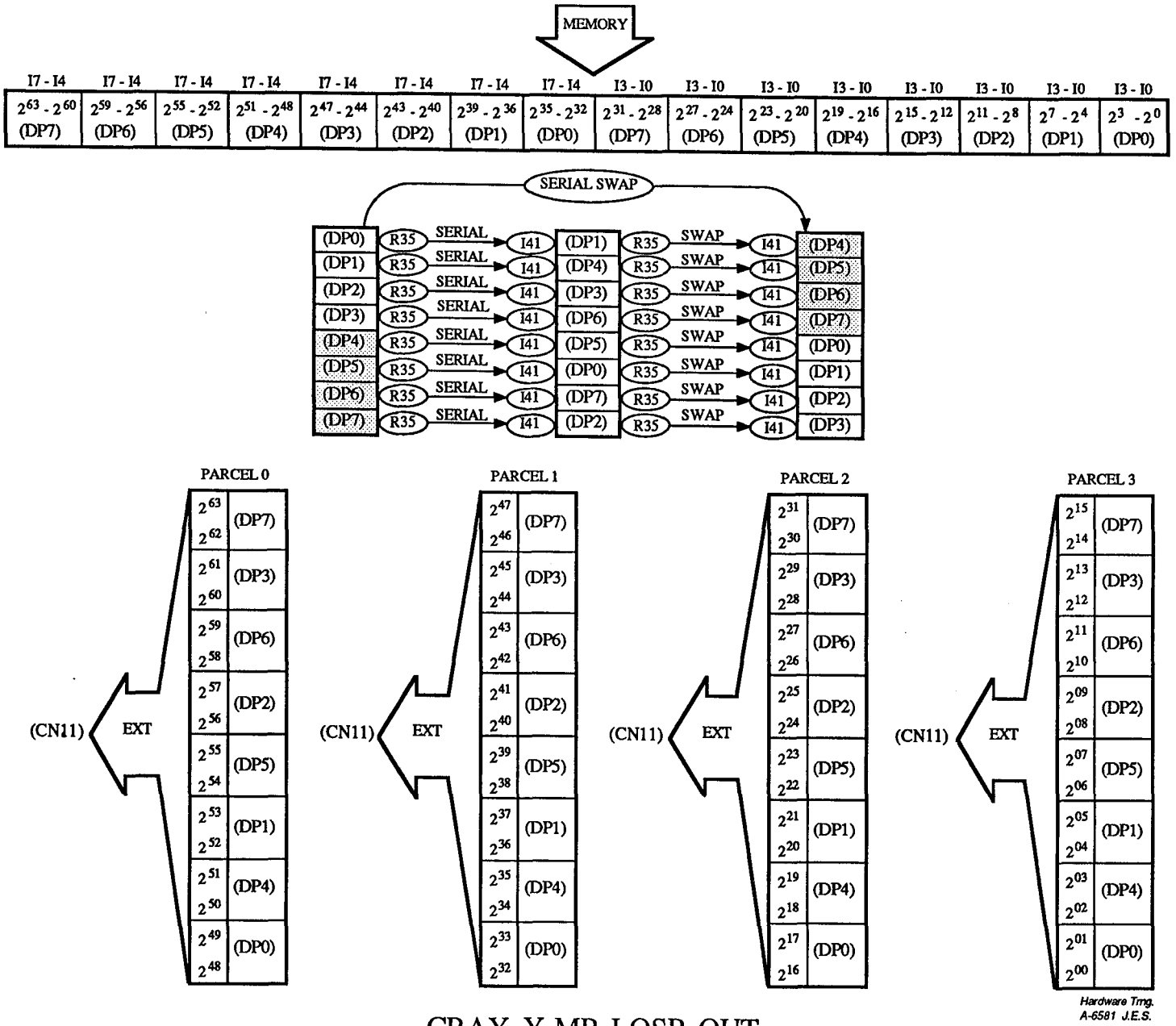
CRAY PROPRIETARY



Hardware Trng.
A-6612 J.E.S.

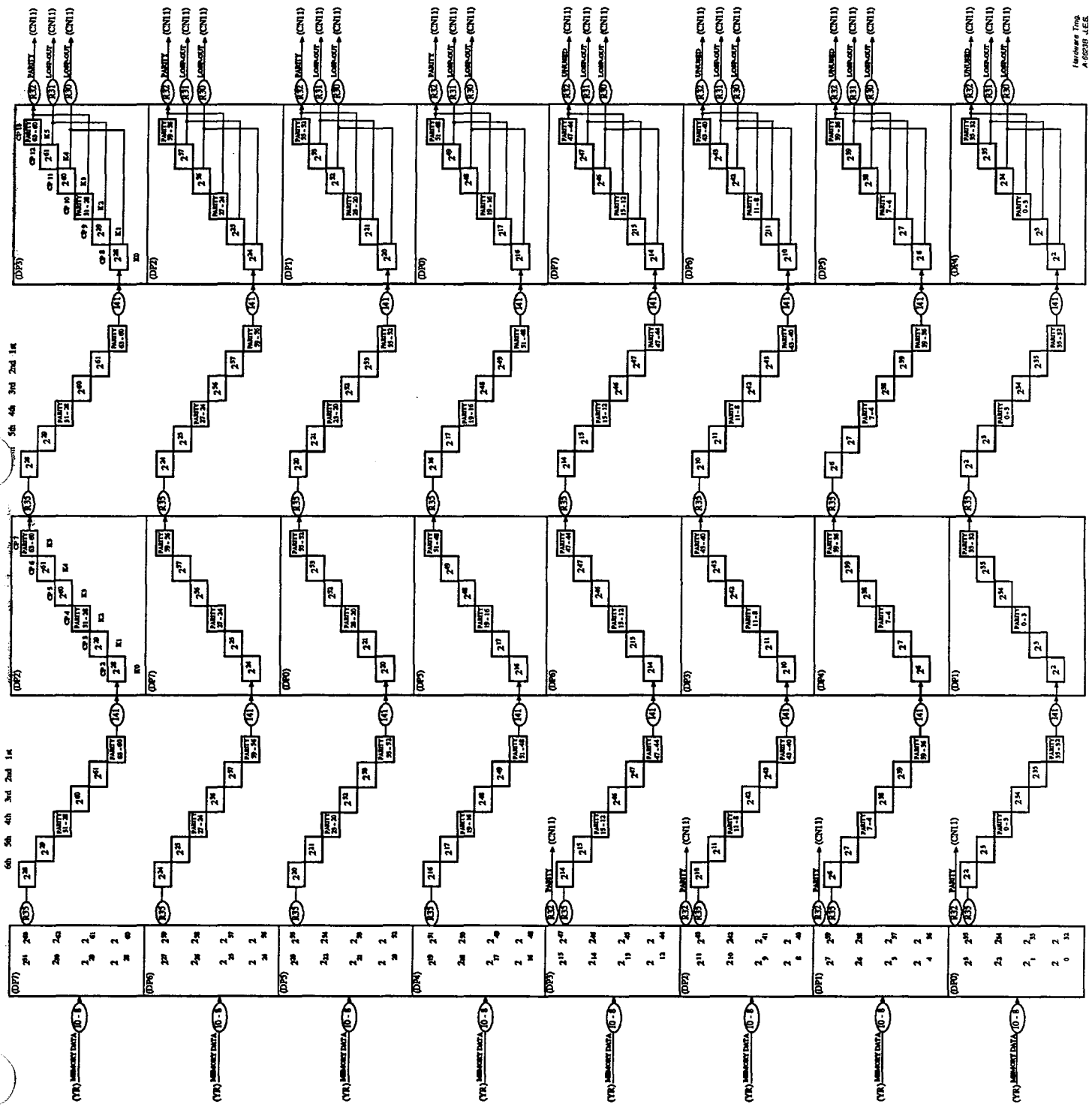
CRAY Y-MP LOSP-OUT FLOW CHART

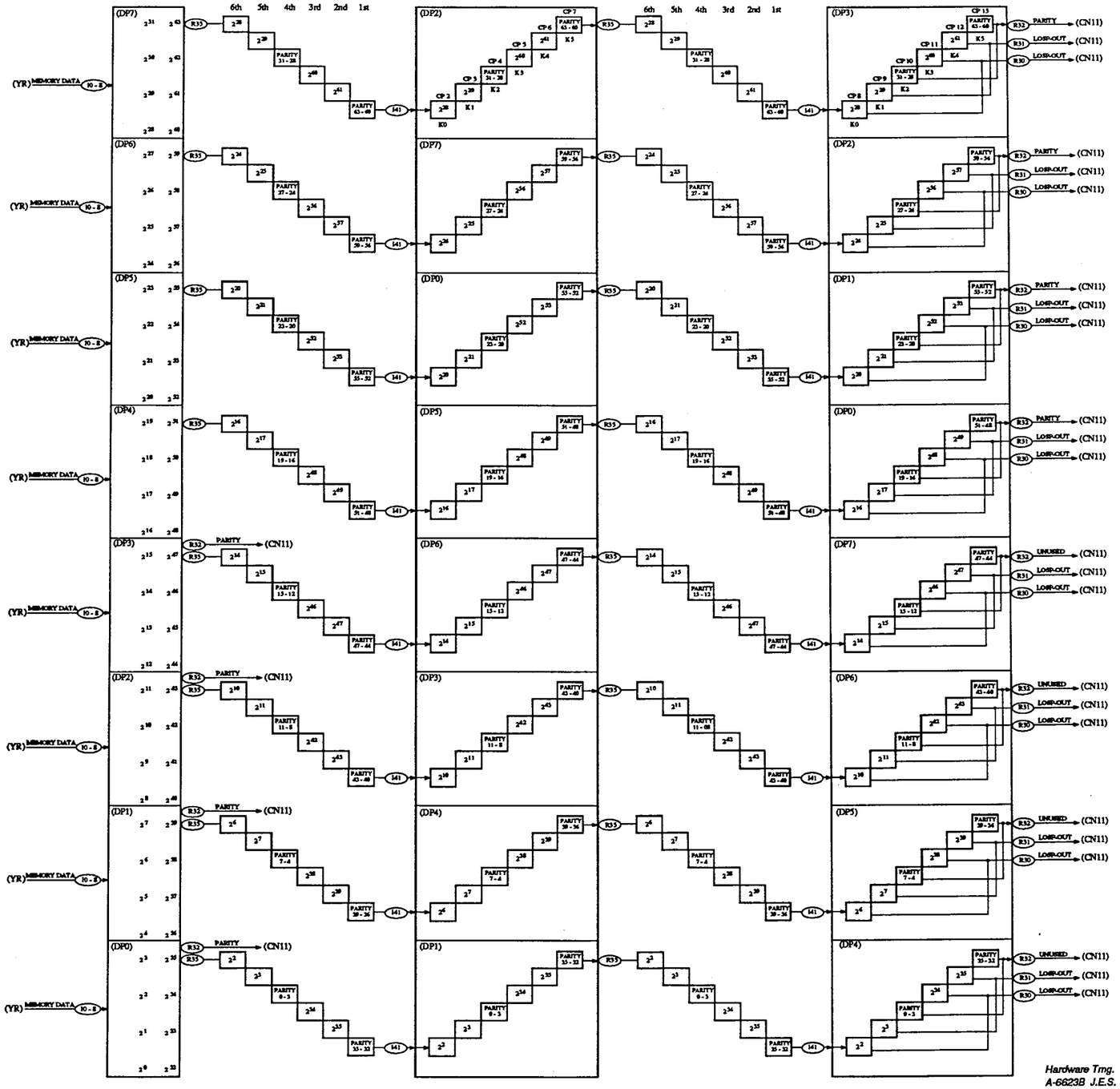
CRAY PROPRIETARY



CRAY Y-MP LOSP OUT

CRAY PROPRIETARY

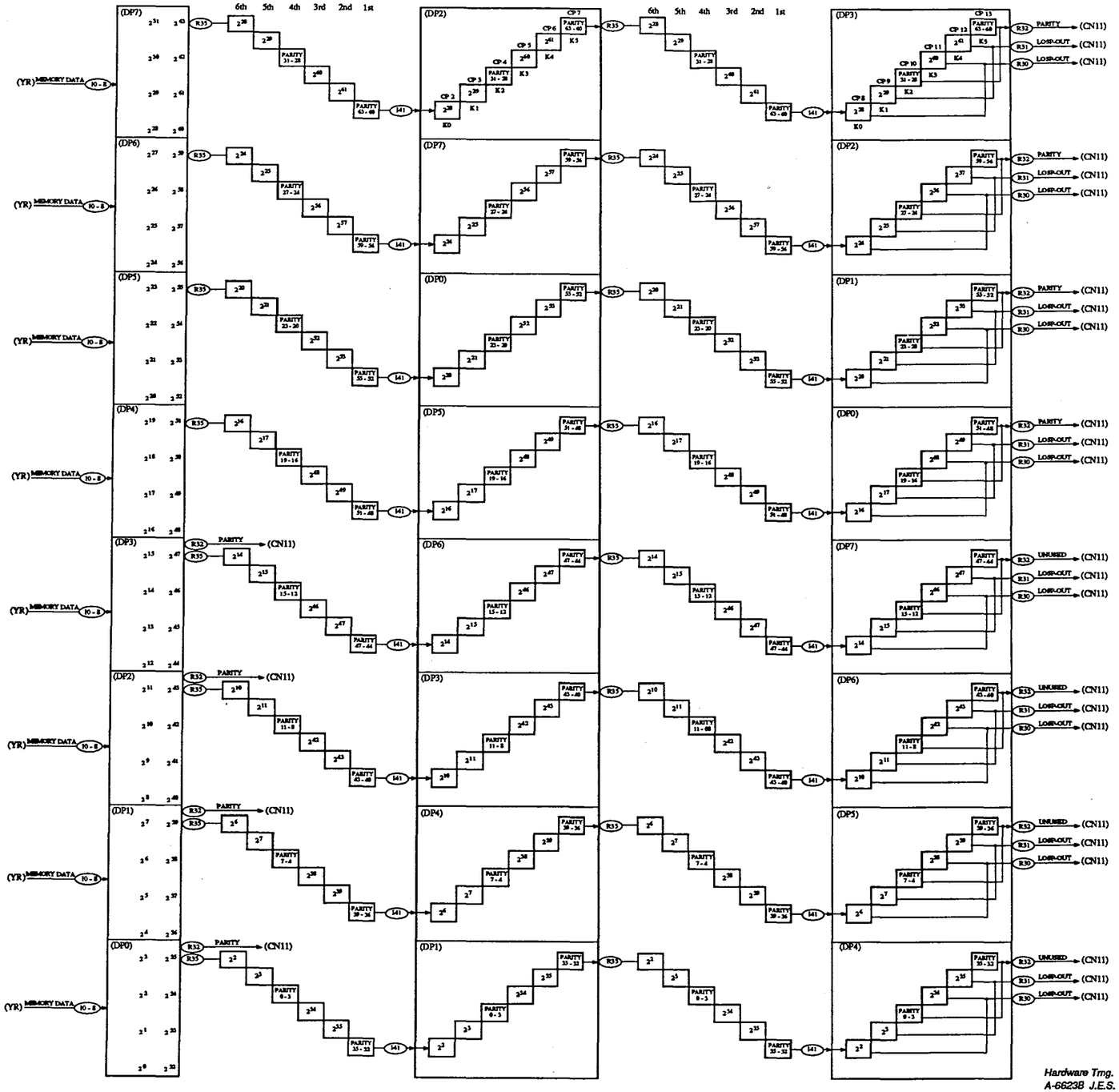




CRAY Y-MP
LOSP-OUT SERIAL SWAP

Hardware Trng.
A-6623B J.E.S.

CRAY PROPRIETARY



CRAY Y-MP
LOSP-OUT SERIAL SWAP

Hardware Tmg.
A-6623B J.E.S.

CRAY PROPRIETARY

CRAY Y-MP I/O HISP OVERVIEW

The HISP channels are designed to be compatible with the CRAY X-MP and perform a transfer of 100 megabytes per second or one word transfer over the cable every 12 clock periods or 75ns. Each data transfer is proceeded by one data ready. The data ready proceeds the data by 4 clock periods ≈ 25 ns. The data Ready is on for 3 clock periods ≈ 19.5 ns.

Each CPU module contains one HISP channel pair identified as HISP-In and HISP-Out, contained in the CRAY Y-MP 8/32 there are eight HISP channel pairs.

The HISP channel is treated as a slave device in the CRAY Y-MP, with all the controls being originated over in the IOP. The only control originated on the CRAY Y-MP is the Transmit address on the HISP-In or HISP-Out. Transmit address is used to tell the IOP that the particular HISP In/Out is not in use or in error.

The HISP channel takes a backdoor approach to getting into and out of memory; for getting into and out of memory; essentially the CRAY Y-MP has no active control of the channel usage except for Port D access. The IOP supplies the block length or number of words to transfer and the starting address of where to read/write the block of data from memory. To determine whether a read/write operation will be taking place in the CRAY Y-MP memory depends on which Transmit address HISP-In or HISP-Out is used by the IOP.

HISP-Out Operation

The HISP-Out will be reading data from memory and sending the data over the IOP. Between the CRAY Y-MP memory and the IOP there are two buffers, A/B which are referred to as the foreground and background buffer. Each buffer is capable of storing 17_8 words and once the buffer has been filled, the buffers will handoff such that the foreground will become the background. The foreground on a HISP-Out operation will always be loaded from the CRAY Y-MP memory while the background will always be reading out data to the IOP.

The foreground buffer is filled by the CRAY Y-MP making 17_8 reference requests to memory one word at a time, and using Port D on a read reference request. Port D is also used by the Fetch operation where fetch and I/O would share Port D. A fetch would have priority over I/O, however, the next reference to use Port D would be I/O if the I/O reference request was held back.

Once a buffer is filled the (DD) option will start reading the background buffer to the IOP by suppling the (DP) options with the read address. Each word sent to the IOP is proceeded by one data ready which is 4 CPs ahead of the data ≈ 25 ns. When the (DD) makes a reference request the (DA/B) will decrement the block length by -1. The (DD) will keep track of the references coming out of the (YZ).

The (DD) will keep track of the number of references coming out of memory and when the count is equal to 17_8 the (DD) knows that a buffer is full and can start sending the data to the IOP, and perform a handoff to start filling the next buffer. As the reference requests come into the (DA/B) options and the block length continues to get decremented, eventually the block length on the (DA/B) will equal zero (BL = 0). When the BL = 0 the (DD) will know that the next I/O

reference channel number coming from the (YZ) will be the last reference. Because the (DD) was keeping track of the number of references it will use this value and compare it against the Readout buffer address. When the two are equal the (DD) will set the last word flag (LWF) to the IOP, and as long as no errors occurred the (DD) will allow Transmit address to be enabled on HISP-Out.

While sending data to the IOP the (DD) knows that the IOP has room for 16_{10} words by first checking to see if transmit data has been sent from the IOP. Transmit data will go away if the IOP does not have room for 16_{10} words. The LWF would tell the IOP that the CRAY Y-MP has finished sending data.

HISP-In Operation

When the IOP wants to use the HISP-In channel on the CRAY Y-MP the IOP will first check to see if the CRAY Y-MP has set the Transmit address for HISP-In. The IOP will then send over the address and block length to the CRAY Y-MP in three transfers, each transfer being 16 bits in length. The address is held on the (DA/B) options while the block length is held on the (DD0) option along with 4 bits (2^0 -2^3) of the block length which are also being held on the (DA0) option for the last transfer. Once the address and block length has been loaded, and the parity checked on each transfer, the IOP will start passing data to the CRAY Y-MP as long as the HISP-In Transmit data is enabled from the CRAY Y-MP. Each word sent from the IOP is accompanied by a data ready. The data ready is sent to the (DD) where the block length gets decremented by -1, and the Write address gets incremented by +1. The Write address along with the buffer select is sent to the (DJ) option to load the data into Buffer A/B. Once a buffer is full the (DD0) will handoff the buffers and a make reference request to the (DA) option where the (DA) will determine the priority of the HISP reference, allowing a LOSP to go ahead. The (DA) upon allowing a HISP-In to make a reference will then start reading the background buffer on the (DJ) options to memory by suppling the address and buffer selects to the (DJ's).

When the IOP has finished sending the CRAY Y-MP data and its BL = 0 the LWF is sent to the CRAY Y-MP. The LWF is sent to the (DD) option which is passed to the (DA) option where the lower 4 bits of the original block length were held. Those block length bits are then used to determine how many words were put into the last buffer and then how many need to be read out on the last word being read out of the buffer, and the block length now equal to zero. The (DA) will send the designator along with the last word which is tracked through memory. When the reference tracking comes back to the (DD0) option the (DD) knows that the last word and everything prior has been sent to memory, so the (DD) will then drop active on HISP-In and set Transmit address if no memory errors occurred.

HISP CHANNELS OPTION INVOLVED

(DA0) Option

The (DA) option contains the Memory address bits (2^0 - 2^4), which is loaded from the IOP along with the lower Block Length bits (2^0 - 2^3) on a HISP-In which is used to determine the last reference. The purpose of the (DA) is to generate the reference request for the HISP channel, but first checking to see if the LOSP is also making a request reference. If so, the LOSP channel is allowed access first. Once the (DA) has been given the reference request on a HISP-In, the (DA) will supply the (DJ) options with the proper address to start sending the data to memory. On a HISP-Out the (DA) contains Block Length bits (2^0 - 2^4). The (DA) will gate the designators always on a HISP-Out, but only on a HISP-In last write.

(DB0 - DB2) Options

The (DB) options each contain 8 bits of Memory address. On the HISP-Out operation each (DB) option contains 3 bits of the Block Length. Both the address and Block Length are loaded from the IOP. As each memory reference is requested, the address is incremented by +1, while the Block Length is decremented by -1. As each (DB)'s Block Length equals zero the (DB) will generate a Block Length borrow-out condition such that when all (DB)'s have a borrow-out the Block Length register would equal zero. When the Block Length equals zero, BL = 0, the HISP-Out reference request is killed.

(DD0) Option

The (DD0) option is the HISP controller. The (DD) checks for errors while the IOP is sending over the address and Block Length, as well as passing double bit memory errors over to the IOP. On a HISP-In operation, the (DD) contains the Block Length register which is decremented by -1 as each data ready is received from the IOP. On a HISP-In the (DD) will supply the (DJ) options with the Buffer Write address and on a HISP-Out supply the (DP) options with the Buffer Read address. The (DD) supplies the control signals over to the IOP such as Transmit address, address error or uncorrectable error, transmit data on a HISP-In, HISP-Out data ready and HISP-Out Last Word flag. On a HISP-Out, Last Word flag is set when the BL = 0 and the reference count on the (DD) is equal to the Readout address for the (DP) options. The (DD) receives a Clear channel from the IOP after a previous error condition or after the master clear was set. Under normal conditions the Clear channel is not sent.

(DJ0 - DJ7) Options

The (DJ) options contain the buffers for the HISP-In operation. Data is written into the buffers from the IOP. The Write address is supplied by the (DD) option, while the Buffer Read address is supplied by the (DA) option. Each (DJ) option is capable of handling 8 data bits plus 1 check bit, which can be stored into either Port 1 or Port 2. Port 1 and Port 2 contain two 16-word buffers called Buffer A and Buffer B. Buffers A and B are fully duplexed such that Buffer A can be read while Buffer B can be written to. The (DJ) options are symmetrical such that (DJ0) receives the true address while (DJ5) would receive an inverted address. Port 1 is used on (DJ0, DJ3, DJ4 and DJ7), while Port 2 is used on (DJ1, DJ2, DJ5 and DJ6) for the HISP transfers.

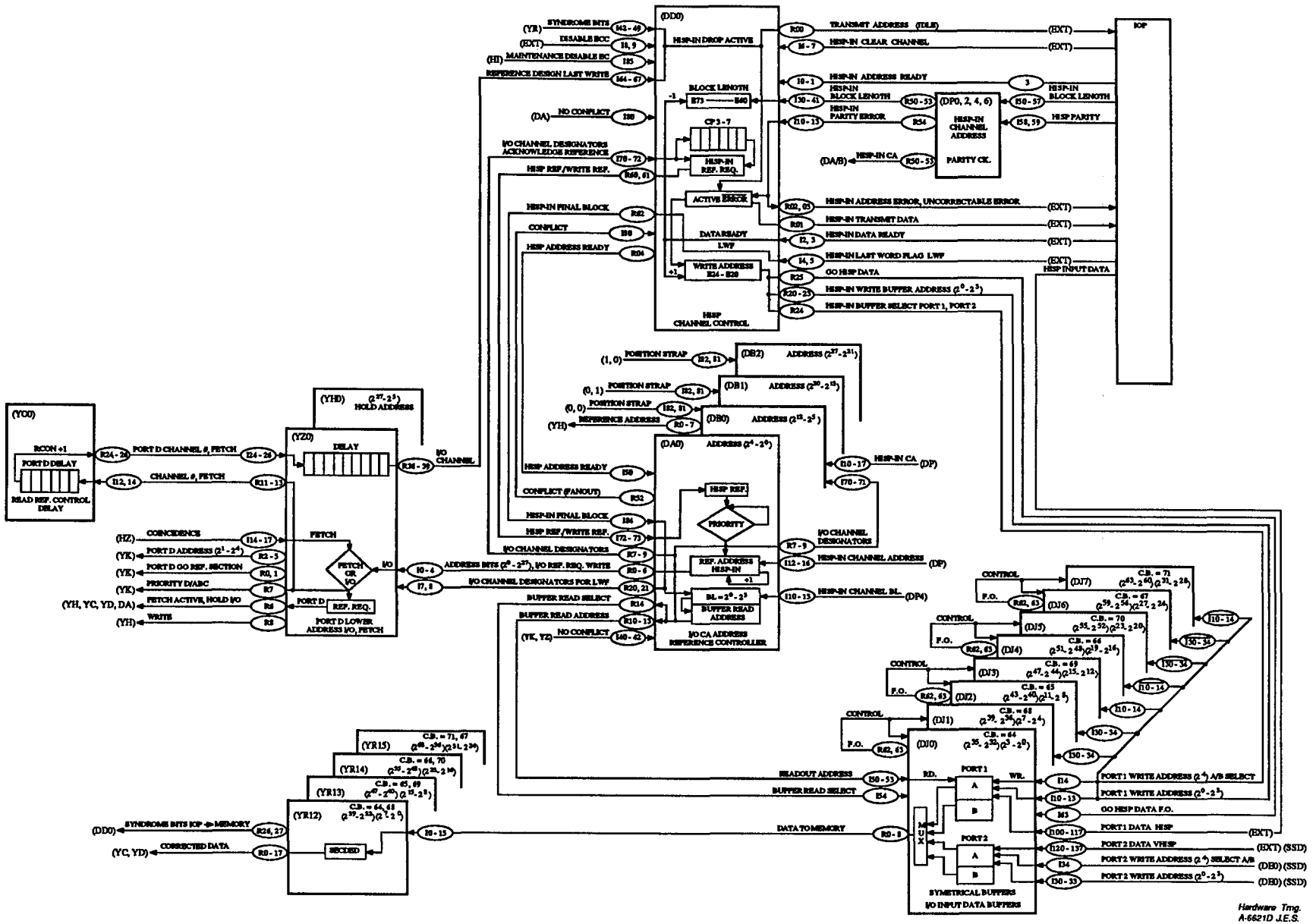
(DP0 - DP7) Options

The (DP) option is similar to the (DJ) option. The (DP) receives the Buffer Write address from the (YZ) option while the Buffer Read address is sent from the (DD) option. The (DP) is a symmetrical buffer similar to the (DJ). The (DP) is also responsible for receiving the initial starting address and Block Length from the IOP. The (DP0, DP2, DP4, and DP6) are responsible for receiving and checking for parity errors on the Starting Address and Block Length for a HISP-In operation, while (DP1, DP3, DP5 and DP7) perform the same operation for the HISP-Out operation.

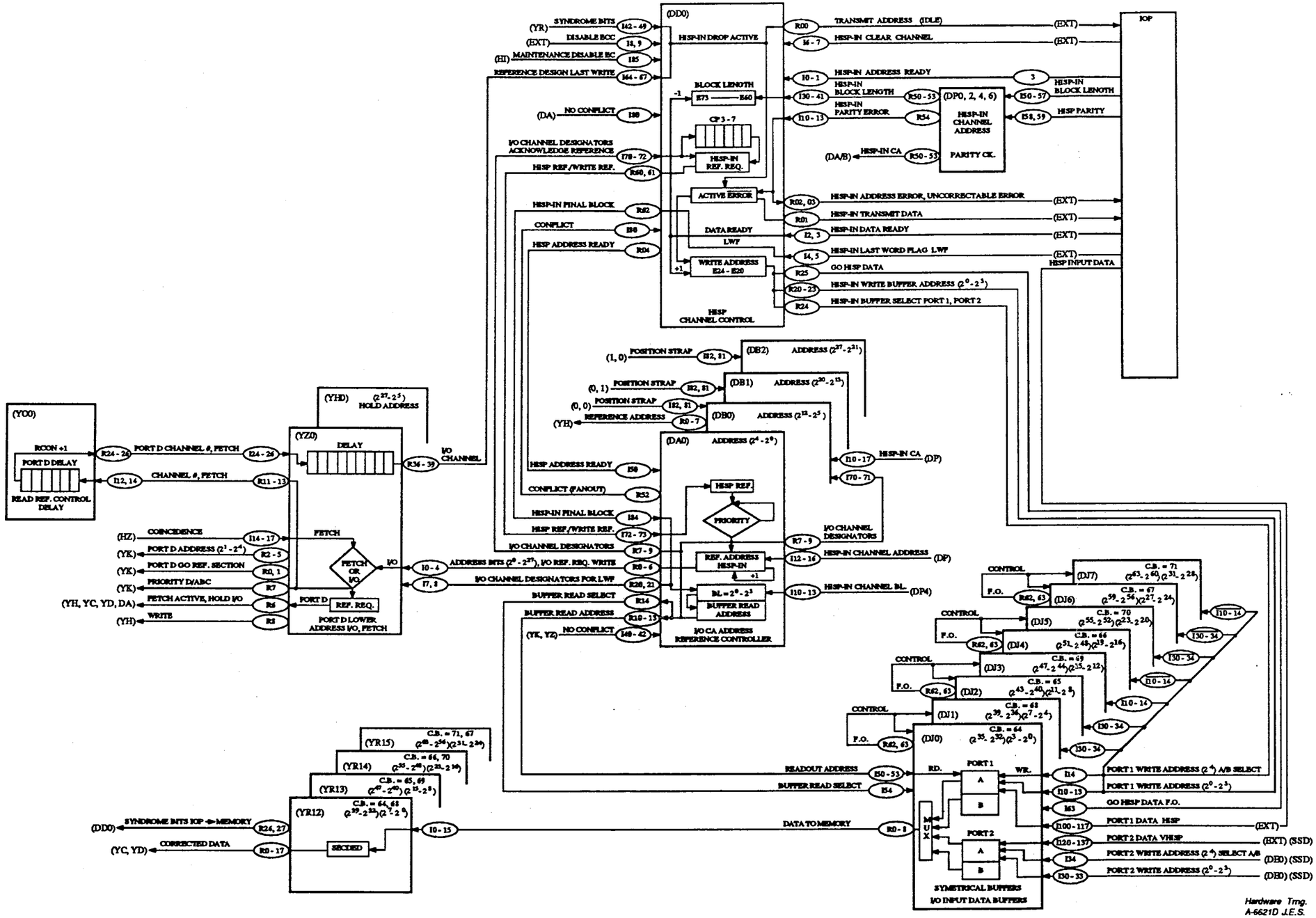
(YZ0 - YH0) Options

The (YZ) option will assign the HISP reference request to Port D. Before a HISP reference can use Port D, Port D must be checked against a fetch request. The (YZ) handles Memory address bits (2^0 - 2^4) which become Port D reference controls. The (YH) option handles the Upper address bits (2^5 - 2^{27}). During a HISP-In operation, the (YZ) will delay the channel designator only on the last write, which is sent to the (DD) option to set the Last Word flag. On a HISP-Out, each reference is sent with channel designators, bits (2^0 - 2^2), which is used to track the word coming out of memory. The (YZ) will also supply the (DP) option with the Buffer Write address on a HISP-Out operation.

Hardware Trng.
YM03/02 J.E.S.

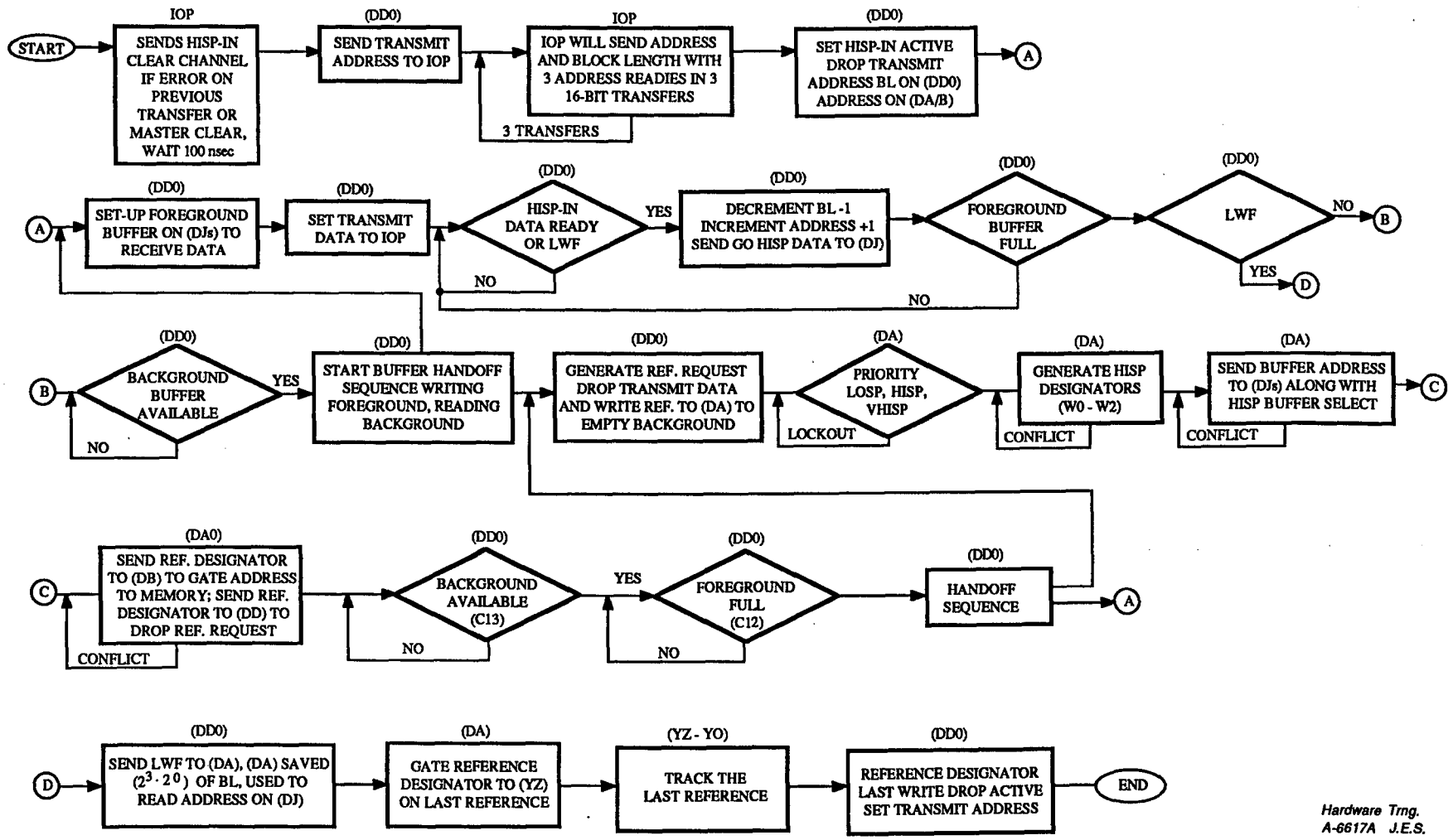


CRAY Y-MP HISP-IN BLOCK DIAGRAM



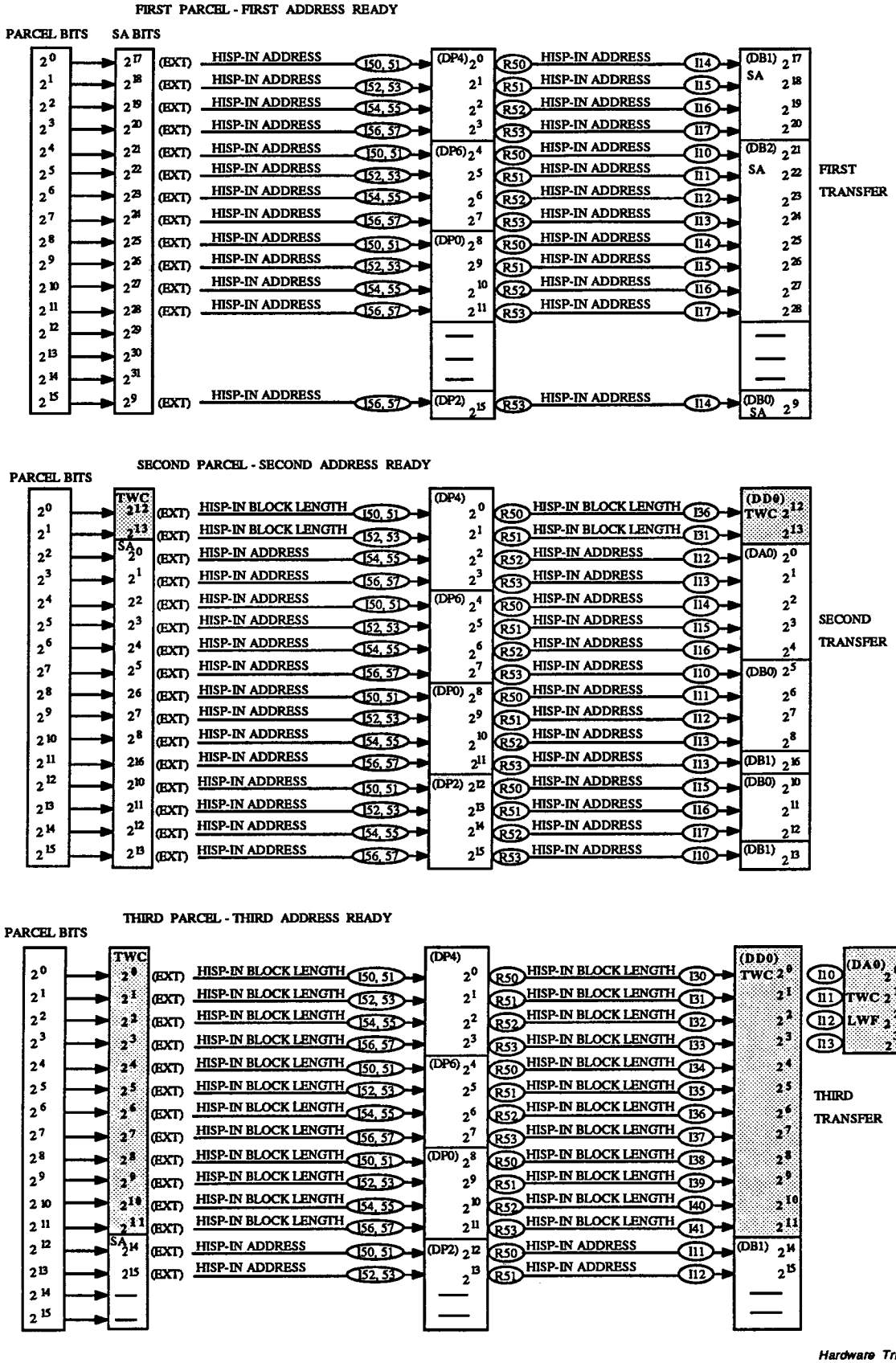
CRAY Y-MP HISP-IN BLOCK DIAGRAM

CRAY PROPRIETARY



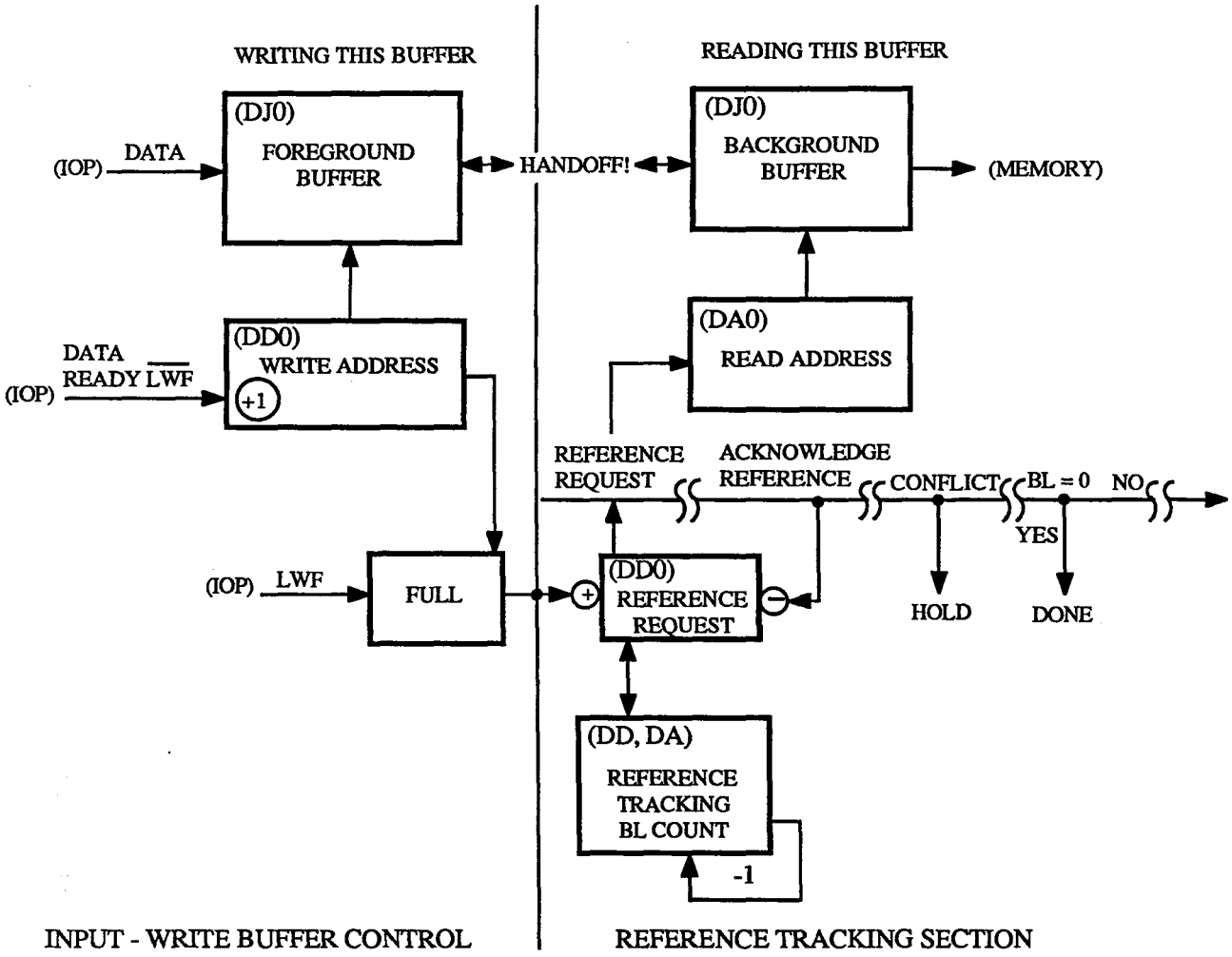
Hardware Trng.
A-6617A J.E.S.

CRAY Y-MP HISP-IN CONTROL SEQUENCE



Hardware Trng.
A-6618B J.E.S.

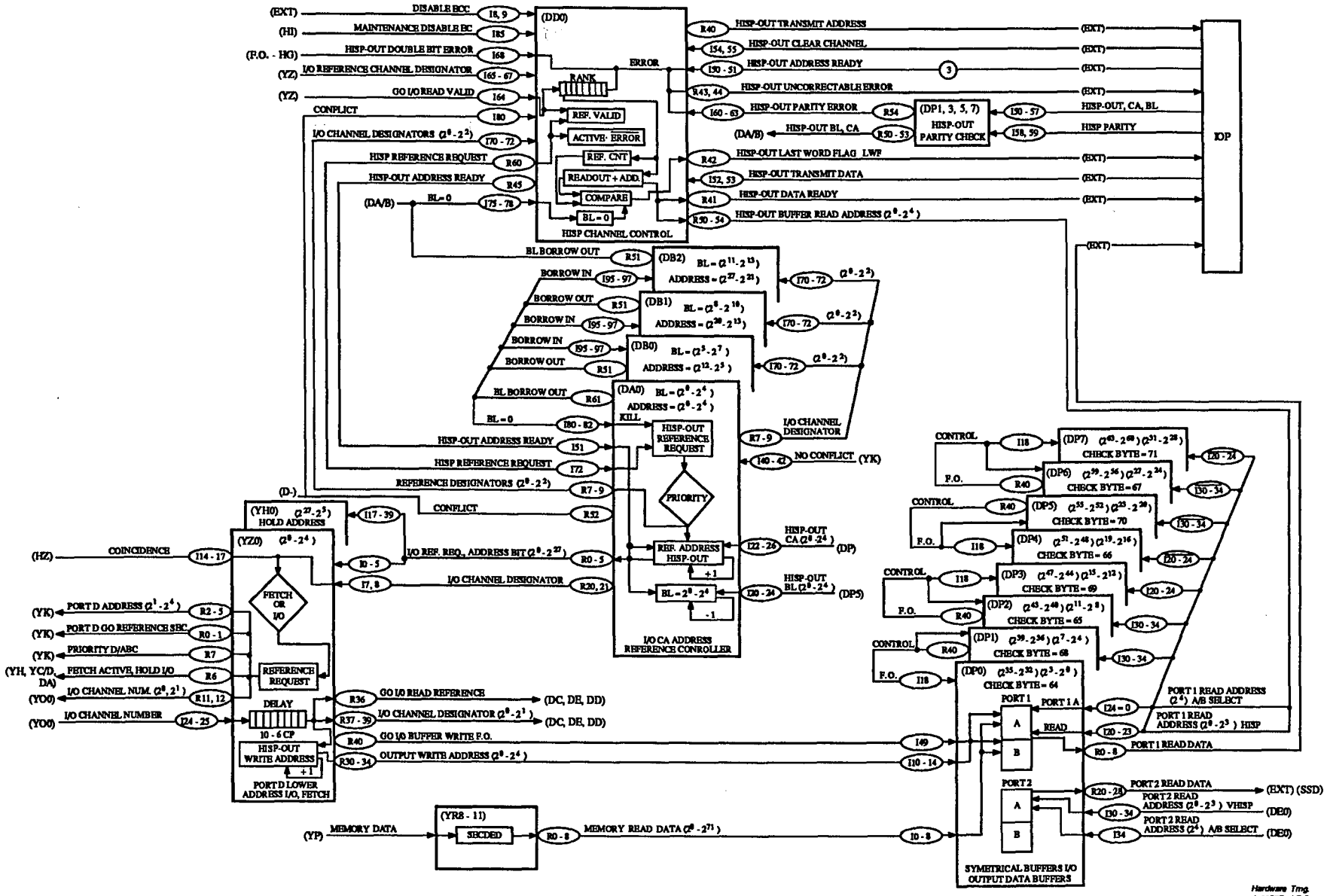
CRAY Y-MP HISP-IN TRANSFER ADDRESS/BLOCK LENGTH



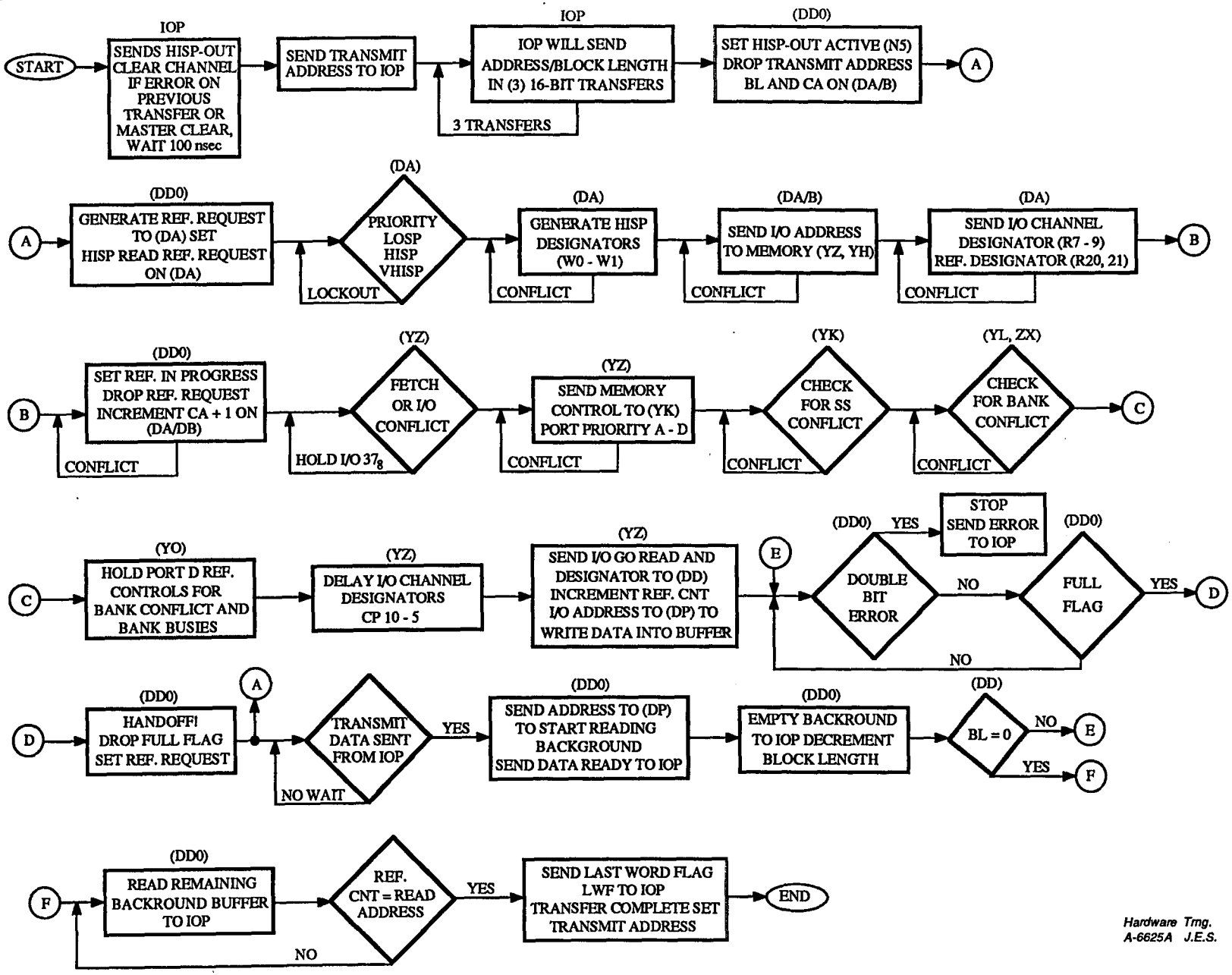
Hardware Trng.
A-6616B J.E.S.

CRAY Y-MP HISP-IN FOREGROUND/BACKGROUND OPERATION

CRAY PROPRIETARY



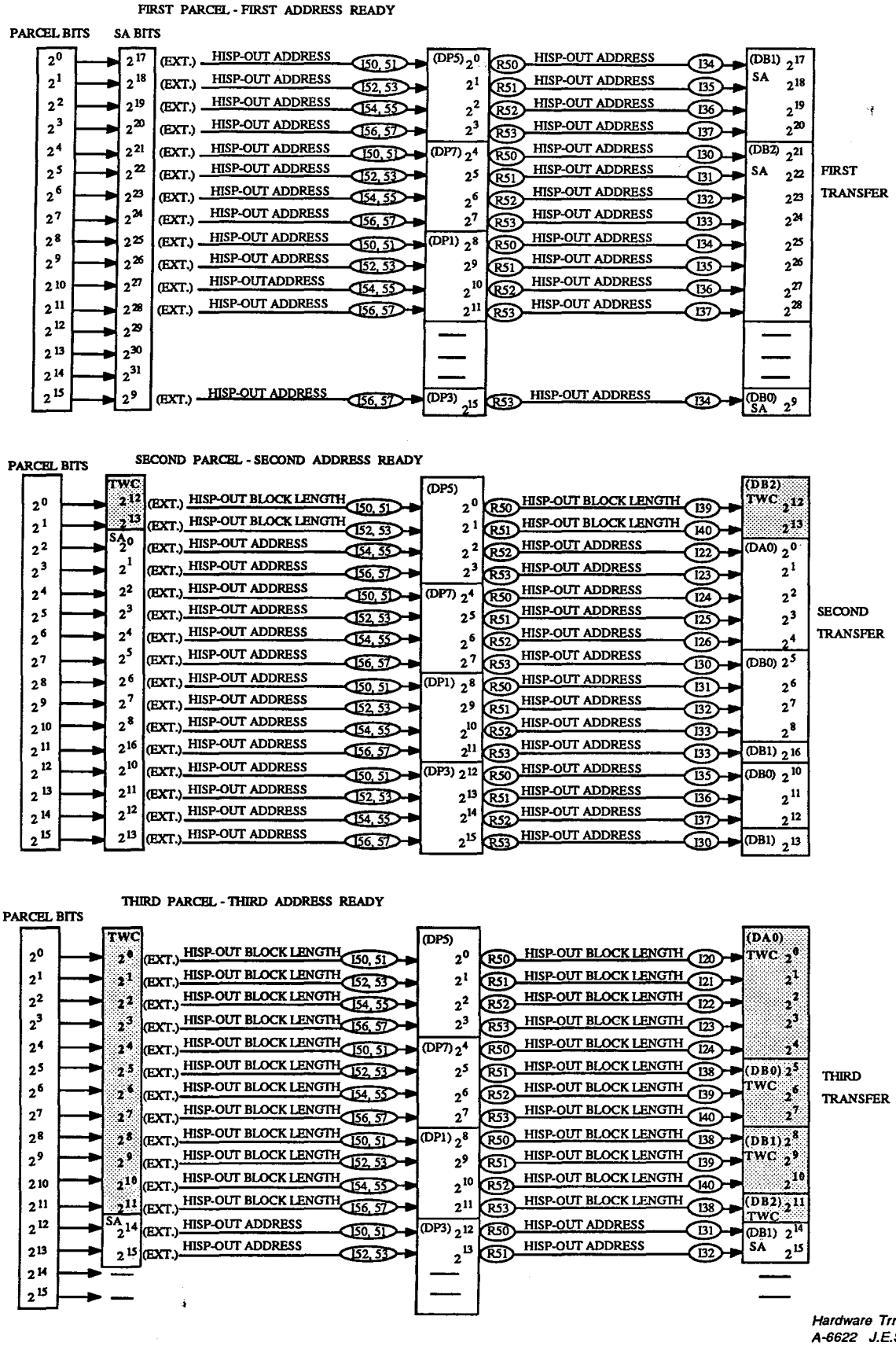
CRAY Y-MP HISP-OUT BLOCK DIAGRAM



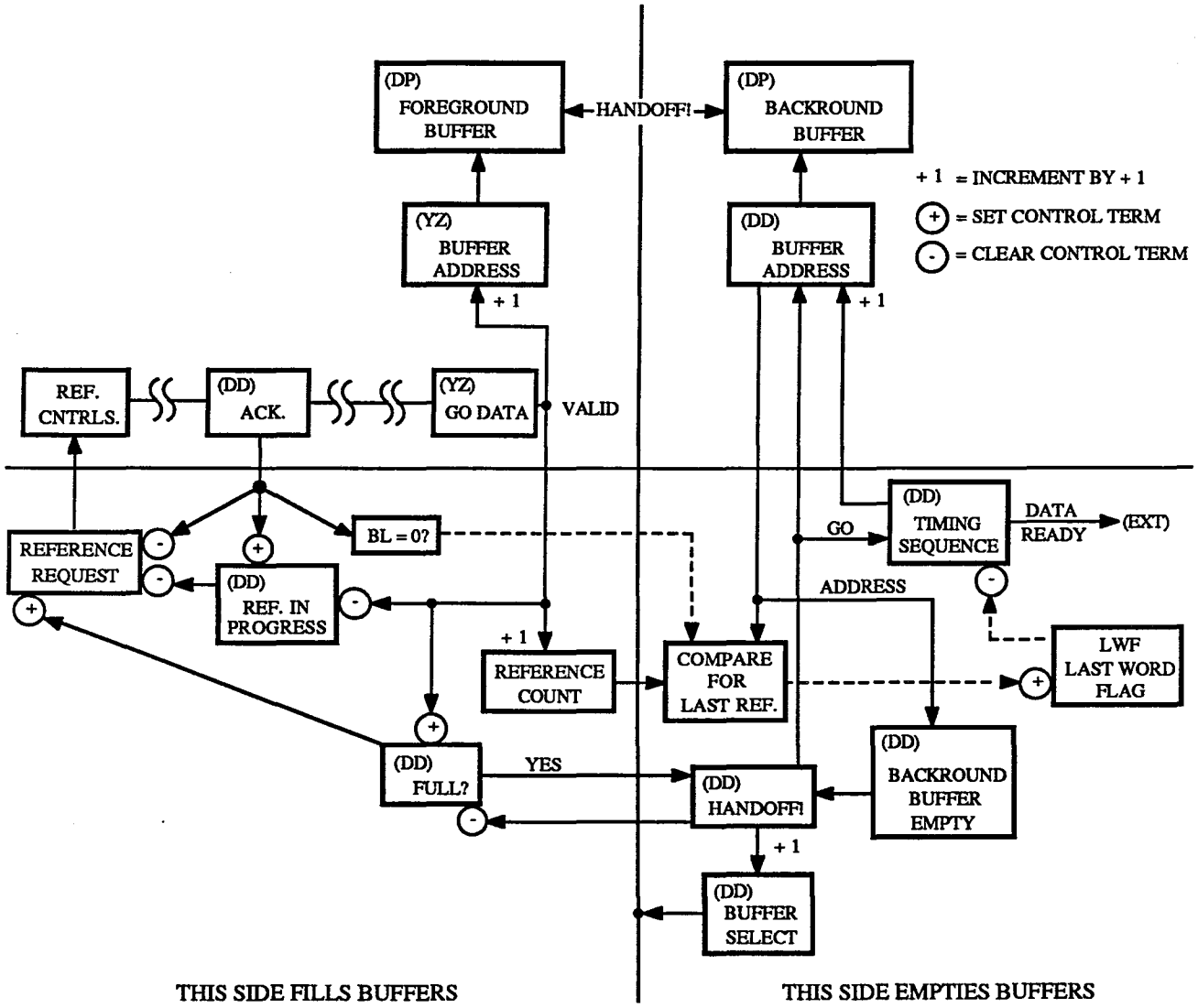
Hardware Trng.
A-6625A J.E.S.

CRAY Y-MP HISP-OUT FLOWCHART

CRAY PROPRIETARY



CRAY Y-MP HISP-OUT TRANSFER ADDRESS/BLOCK LENGTH



HISP-OUT SEQUENCE OF EVENTS:

0. Set reference request; reference controls.
1. Acknowledge will drop reference request. Set reference in progress.
2. Check for BL = 0 or last reference. BL = 0 go to step 11.
3. Wait for Go Data from (YZ).
4. Drop reference in progress when 16 references.
5. Increment reference count.
6. Check for Buffer Full? If not, generate reference request; go to step 1.
7. Handoff! Foreground and background if buffer is full.
8. Start reading backround by using the timing sequence. Send Data Ready to IOP.
9. Send Read address from (DD) to (DP); decrement BL.
10. Backround buffer empty enable Handoff!, clear Full, set reference request, go to step 1.
11. Last reference.
12. Compare reference count against Buffer address. When equal, set LWF; transfer complete.

A-6626B

CRAY Y-MP HISP-OUT BUFFER CONTROLS

CRAY Y-MP TO SSD OVERVIEW

The CRAY Y-MP is capable of handling four 1250 megabyte channels numbered in the system as channels (1, 5, 11, 15). One of the VHISP channels require two CPUs to work in unison. VHISP channel 1 uses CPU 0 and CPU 1 Port D access to memory. VHISP channel 5 is made up of CPU 2 and 3, VHISP channel 11 is made up of CPU 4 and 5, and finally VHISP channel 15 is made up of CPU 6 and 7.

VHISP 1 = CPU 0, 1
VHISP 5 = CPU 2, 3
VHISP 11 = CPU 4, 5
VHISP 15 = CPU 6, 7

To transfer data across the cables between the CRAY Y-MP and SSD at 1250 megabytes requires 32 word transfers per Data Ready. The thirty-two 72-bit words are made up of the two buffers located in the CRAY Y-MP. Each buffer is capable of storing sixteen 72-bit words.

The Block Length (BL) register contains the number of blocks of data transferred. One bit in the Block Length would represent sixty-four 72-bit Cray words to be transferred.

To initiate a CRAY Y-MP transfer, the following instructions are issued with the (Aj) channel numbers equal to (1, 5, 11, 15). Any CPU could use any of the VHISP channel numbers which are available.

0012j0 - Clear interrupt on channel (Aj)
0010jk - Load SSD starting Block address
0010jk - Load CRAY Y-MP starting address
0011jk - Transfer Block Length, Read/Write, transfer initiated
033i00 - Read channel number to Ai
033ij1 - Read Error flag/status word

In the CRAY Y-MP, the two CPUs that make up a VHISP channel create a problem in that both CPU modules are identical. Depending on the slot that the CPU module is in determines if the CPU module is a master VHISP controller or a slave. All the even numbered CPUs become master VHISP controllers. The two CPU modules divide up protocol between the SSD and the CRAY Y-MP. In general, the odd numbered CPUs communicate with the SSD while the even numbered CPUs monitor the controls with the CRAY Y-MP memory.

The (DE) option is responsible for controlling the flow of data between the SSD and the CRAY Y-MP. There are two (DE) options that make up the VHISP control, one (DE) option per CPU module. The (DE) on CPU 0 for VHISP channel 1 is the master controller while the (DE) on CPU 1 is the slave controller. In order for the (DE) to both communicate between the SSD and CRAY Y-MP, the (DEs) logic latches on two different clock speeds which are 12 ns and 6 ns, or system clock. The two (DE) options need to communicate between themselves so that they will work in unison. To make this happen, there are basically two control lines that are used, they are VHISP Data Ready Vote and VHISP Transfer Progress.

VHISP Transfer Progress

The VHISP transfer progress is a 3-bit code which is passed between the CPUs (DE) option on the 12 ns cycle rate. The 3-bit code contains the progress report which is passed from CPU 1 to CPU 0. The progress report from CPU 1s (DE) option sent to CPU 0s (DE) option would contain three signals. The signals are:

- Off-line - Meaning the SSD is off-line, Select Code = (111)
- VHISP Double Bit error - On a VHISP-In or VHISP-Out operation, Select Code = (011)
- Acknowledge Clear - Coming back from the SSD on a (0012) instruction, Select Code = (010)

The transfer progress is passed from one (DE) option to the next using the VHISP clock timing signal which runs at 12 ns. The transfer progress status is sent over every other clock as a serial communication stream. Between the 12 ns clock is sent reference in progress which is used to determine on a CRAY Y-MP through SSD, the last buffer write, and also sent as the status word. Reference in progress ensures that if the VHISP is deactivated and reference in progress is active, the condition is flagged as part of the status word.

VHISP Data Ready Vote

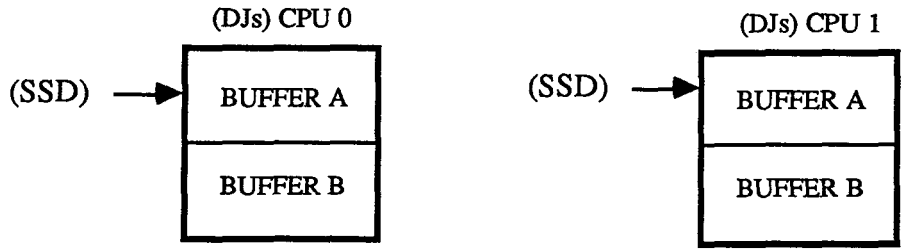
The VHISP Data Ready Vote is used to handshake between the two (DE) options. On a VHISP-In operation, the (DE) options need to determine when to allow a transmit enable to be sent to the SSD. On VHISP channel 1, CPU 1s (DE) option would send the transmit enable to the SSD. Before a transmit enable can be sent to the SSD, both CPU 0 and 1 must have a buffer empty and ready to be filled from the SSD. If CPU 0 has its buffer empty, CPU 0 will send a Data Ready Vote to CPU 1 indicating that CPU 0 is ready to receive another buffers worth of data from the SSD. CPU 1 will wait until it also has a buffer empty and will then send a Data Ready Vote to CPU 0. Because CPU 0 and CPU 1 have both voted, the decision is unanimous to allow a transmit enable to be sent to the SSD.

On a HISP-Out, the Data Ready Vote takes place to decide if both CPUs have a buffer full, and that it is time to send a Data Ready to empty both buffers. For VHISP channel 1, the Data Ready is sent from CPU 0, only when the SSD has previously sent a transmit enable.

VHISP-In Buffer Control

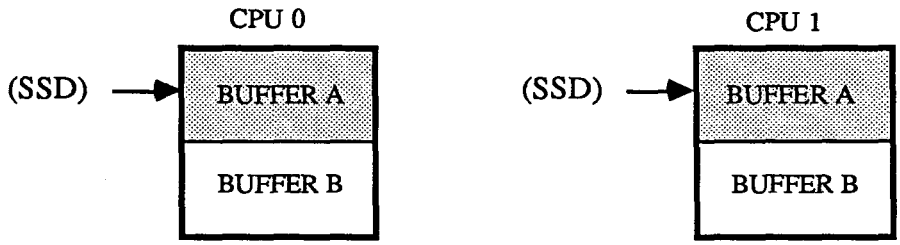
To put into perspective how the two CPUs work in unison, and how data ready vote and transfer progress report works, the following figures demonstrate the flow of data for VHISP-In, channel 1.

A)



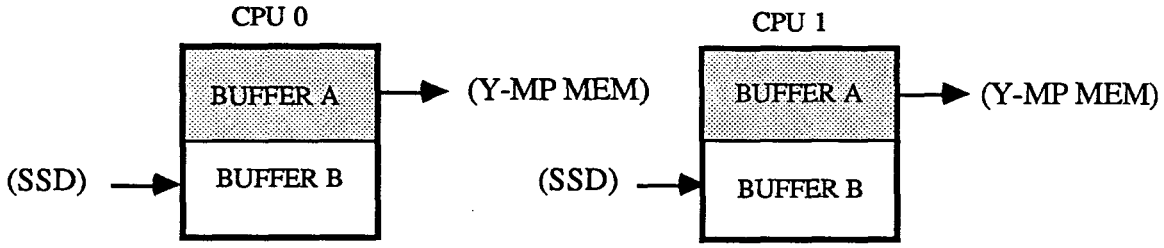
1. Transmit Enable was sent to the SSD to fill an empty buffer on the CRAY Y-MP.
2. A Data Ready is received from the SSD indicating that a buffer worth of data is coming.

B)



1. Both buffers are filled from the SSD.
2. Transmit Enable will stay active because both CPUs still have an empty buffer.

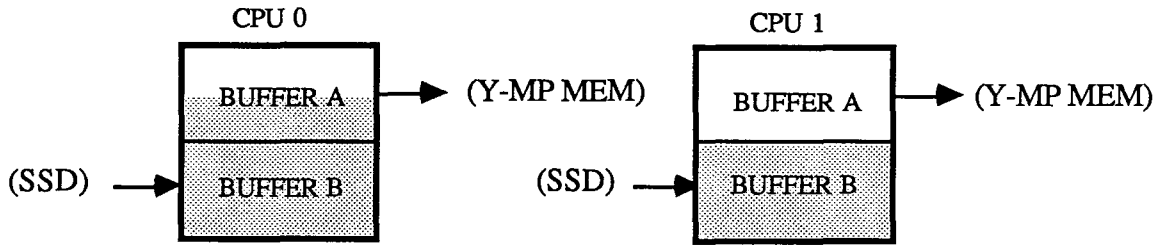
C)



1. CPU 0 and CPU 1 will now generate a reference request to empty their buffers to memory.
2. The SSD can also start filling the other buffer.

Hardware Trng.
YM13/14 J.E.S.

D)

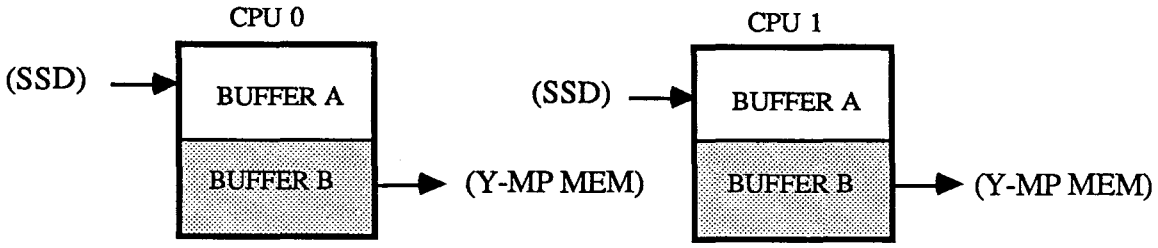


1. Buffer A of CPU 0 has not completed emptying Buffer A.

2. Buffer A of CPU 1 has emptied its buffer to memory, so CPU 1s (DE) will pass a Data Sync Vote to CPU 0. Because CPU 0 is not finished emptying its buffer, CPU 0 has not sent out a Data Sync Vote to CPU 1.

3. Transmit enable is now dropped to the SSD to prevent the SSD from sending another buffer worth of data which the VHISP channel does not have room for.

E)



1. Once CPU 0 has emptied its buffer, a Data Sync Vote is sent to CPU 1. At this time, the vote is unanimous, so a Transmit Enable is sent to the SSD indicating that another two buffers worth of data can be received.

2. This sequence continues until the Block Length would equal zero, BL = 0, or an error occurs.

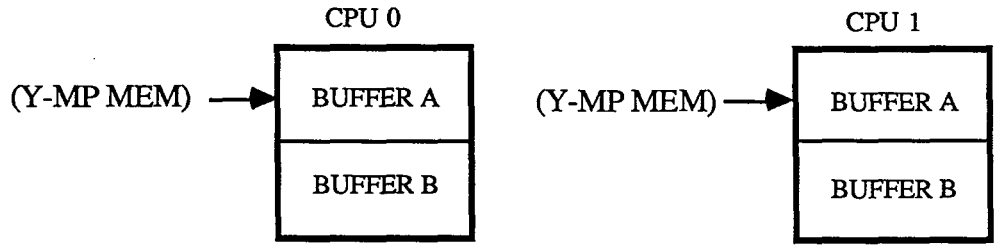
3. If CPU 0 receives an error, it will shut down and ignore CPU 1 because CPU 0 is master. If CPU 1 receives an error, the error information is passed to CPU 0s (DE) option through a 3-bit code which is sent as the Transfer Progress Report.

4. In the VHISP-In operation, the flow of data is throttled back by the Transmit Enable.

VHISP-Out Buffer Control

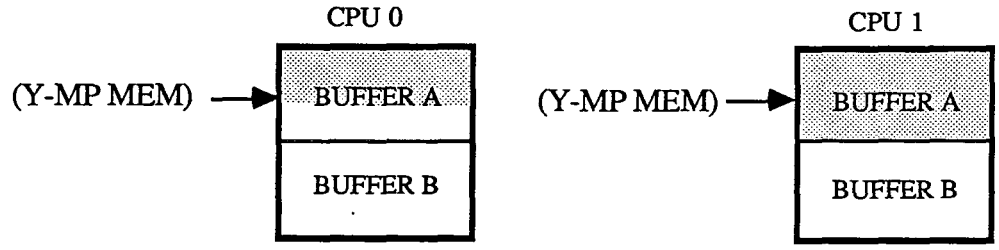
The VHISP-Out works just the opposite of the VHISP-In data flow. The flow of data is contingent upon there being an empty buffer instead of a full buffer as with the VHISP-In.

A)



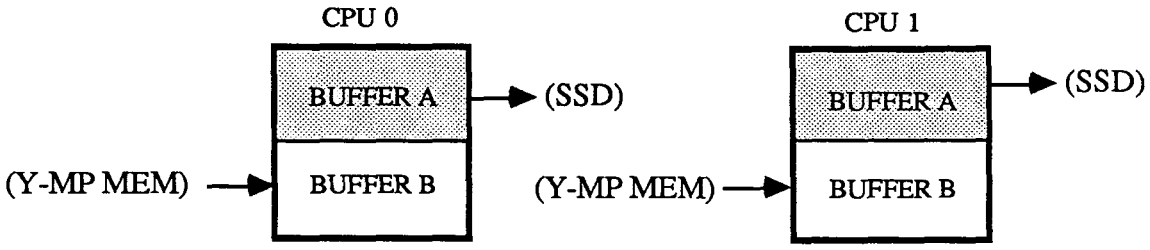
1. For a VHISP-Out a reference request is generated to fill the buffers from memory.
2. Because this is a start-up, a second reference request is generated to also fill the other buffer.

B)



1. Buffer A of CPU 1 has been filled from memory and CPU 1 would have sent a Data Sync Vote to CPU 0.
2. CPU 0 has not completed filling its buffer, so CPU 1 has to wait for CPU 0 Data Sync Vote.

C)

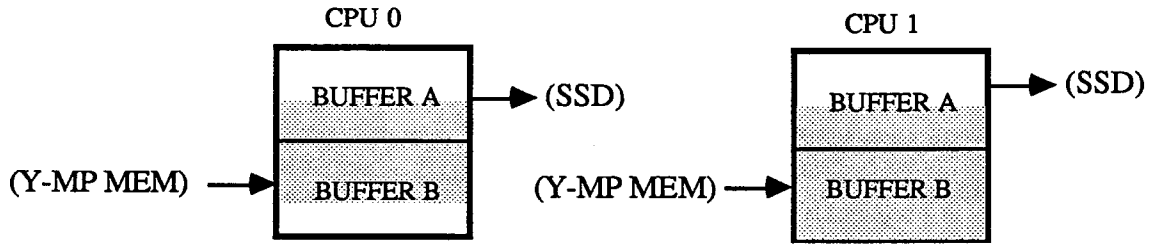


1. CPU 0 has finished filling its buffer and the vote is unanimous. As long as the SSD has sent a Transmit Enable to CPU 0. CPU 0 will send one Data Ready and the flow of 32 words are sent to the SSD.

Hardware Trng.
YM13/16 J.E.S.

2. The second reference request takes place right after the first buffer has been filled, to fill the next buffer.

D)

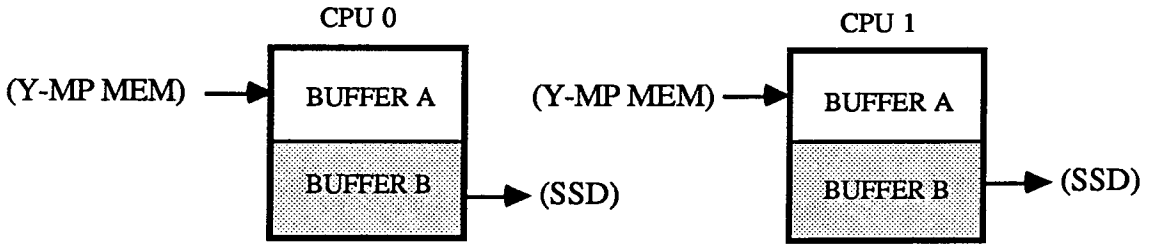


1. When the Readout address of the data being sent to the SSD equals five, another reference request can be generated to start filling Buffer A.

2. CPU 1s Buffer B is filled and CPU 0 has sent a Data Sync Vote to CPU 1.

3. CPU 0s Buffer B is not filled so no Data Sync Vote is sent to CPU 0, which keeps CPU 0 from sending data to the SSD.

E)



1. Once CPU 0 has finished filling its buffer, CPU 0 will send a Data Sync Vote to CPU 1. Because the vote is unanimous and the SSD again has Transmit Enable active, CPU 0 will send a Data Ready which will allow a transfer of 32 words to the SSD.

2. It has been determined that as the last word leaves Buffer A to the SSD, the second reference request would bring in the next word from memory, assuming no memory conflicts.

3. This sequence continues until the BL = 0. What keeps the VHISP-Out throttled back is how fast the SSD can keep up the Transmit Enable, and how fast the CRAY Y-MP memory can keep the buffers full.

4. If BL = 0 or an error occurs, CPU 0 will terminate the transfer and send an interrupt to the Exchange Package.

Block Length/Address Control

The VHISP channel starting address is loaded on the second (0010) instruction. Once the address has been loaded, the (DE) in CPU 1 for VHISP Channel 1 would generate a signal called Bump CA + 1, which would increment CPU 1s starting Memory address by plus 1. If the starting address was even, then CPU 0 would retrieve all the even addresses from memory while CPU 1 would handle all the odd words. Once CPU 0 or CPU 1 has made a memory request and received an acknowledge, the VHISP Memory address is Bumped by + 2.

The Block Length register is loaded upon executing the (0011) instruction. The Block Length is sent to the SSD and remains in the CRAY Y-MP on the (DA/DB) options. The Block Length register on a VHISP-In operation is decremented by -2 when the address is sent to memory. On the VHISP-Out operation, the Block Length is decremented by -2 when the reference acknowledge comes back from the (DA) option and the (DA) option sends a Bump CA + 1 to the (DA/DB), which causes the CA to increment by +2 and the Block Length to decrement by -2.

VHISP OPTIONS INVOLVED

(DA0, DB0, DB1, DB2) Options

The (DA0) option handles the lower 5 bits of the I/O Memory address bits (2^0 - 2^4) and the lower Block Length bits (2^0 - 2^4). On a VHISP start-up sequence, when using the first (0010) instruction to load the starting SSD address, the address is sent from CPU 0s (DA/DB)s for VHISP channel 1. The (DB0 - DB2) handle the Upper Block Length and Memory address bits (2^5 - 2^{23}). Currently, the Block Length is used up to (2^{17}), and the I/O address is used up to (2^{24}).

The (DA0) option also becomes the I/O reference control. All of the I/O Reference Requests are passed to the (DA) from the (DB) option. The (DA) will determine the type of reference Read/Write and the priority between LOSP, HISP, and VHISP. VHISP would have the lowest priority against HISP and LOSP. On a VHISP-Out operation, I/O channel designators are sent with each reference to track the reference through memory and used to identify who's I/O data has come out of memory, LOSP, HISP, or VHISP. On a VHISP-In operation, only the last word written to memory has a I/O channel designator sent with the write reference. When the I/O channel designators are received back from memory to the (DE) option, the (DE) option knows that the last word has successfully been written into memory, and can then drop active.

(DE0) Option

The (DE) is the VHISP control. It requires two (DE) options per VHISP channel to handle all of the VHISP control between the SSD and the CRAY Y-MP. For VHISP channels (1, 5, 11, 15), the even numbered CPU slots in the mainframe are the master VHISP controls, and the odd numbered slots are the slave VHISP controls. For VHISP channel 1, the (DE) option in CPU 0 is the master, while the (DE) option in CPU 1 is the slave. Upon starting up the SSD, the VHISP is responsible for loading the SSD starting address, Block Length, and X-MP starting address. In general, the (DE) in CPU 1 handles the SSD protocol, while the (DE) in CPU 0 controls the overall flow of the data to and from memory.

The (DE) option is one of the most complicated options, along with the (HA) option. A book could be written trying to cover all of the functions that the (DE) option handles. Following is a brief list of some of its functions per CPU.

VHISP Channel 1 CPU 0 (DE) Option

- Receive VHISP Block Lengh Error from SSD - occurs on a VHISP-Out operation when the Y-MP BL = 0, but the SSD Block Length does not equal zero, BL $\neq$ 0.
- VHISP Double Bit Errors - double bit memory error occured while reading the SSD memory or while receiving data on the input to the SSD.

- VHISP-In Done and Empty - SSD Block Length equals zero.
- VHISP-In Data Ready - one data ready sent with each buffer worth of data passed to the Y-MP.
- VHISP-Out Transmit Enable - occurs on a VHISP-Out operation to let the Y-MP know that the SSD has room for 32 Cray words.
- VHISP-Out Last Buffer Write - Sent from the Y-MP to the SSD when the Y-MP Block Length register equals zero, BL = 0.
- VHISP-Out Write Data - one data ready is passed to the SSD on a VHISP-Out operation.
- Capture VHISP Control - Sent to the (DA/DB) options to load the CA and BL registers, CPU 0. (DA/DB) will pass the VHISP Starting Block address to SSD.
- Interrupt Request - when the VHISP channel has finished or an error occurs, the I/O interrupt is directed to the (HH0) option in CPU 5.
- VHISP Status Word - upon getting an interrupt, a status word is created and sent to the (DA/DB) options and is read out on a (033) instruction to Ai through the Shared registers.
- VHISP Data Sync Vote - to synchronize CPU 0 and CPU 1 to determine if it's time to send data to the SSD, or to receive data from the SSD.
- VHISP Transfer Progress Report - CPU 1 uses the transfer progress report to inform the master (DE) of the status of the slave (DE); such things as off-line, double bit errors, Acknowledge Clear, and reference in progress.

VHISP Channel 1 CPU 1 (DE) Option

- Capture VHISP Controls - similar to CPU 0, except CPU 1 sends the VHISP Block Length to the SSD.
- VHISP Bump CA + 1 - causes CA to be incremented by plus 1 on start-up, and during a VHISP-Out operation to increment CA + 2 after reviewing an Acknowledge Reference from (DA).
- Set CA - The VHISP Starting Block address is active on the wires to the SSD, along with the Block Length register. When Set CA is sent to the SSD, the SSD loads the starting address and Block Length simultaneously.
- VHISP-In Read Transmit Enable - sent to the SSD when the Y-MP has two buffers ready to be filled from the SSD.
- VHISP-Out Transmit Enable - received by the (DE) from the SSD, indicating that the SSD has room for 32 Cray words.
- VHISP-In Data Ready - one data ready passed to (DE) at CPU 1 per 32 Cray words.

- SSD Off-line - indicating to the Y-MP that the SSD is off-line, static signal.
- VHISP Clear Controls - sent to the SSD upon issuing an (0012) instruction, and to clear out the SSD port controls.
- Acknowledge Clear - received back from the SSD after the port has cleared out successfully.
- Reference in Progress - sent from CPU 0 to time out exactly when both buffers are full or empty, and also to prevent an error condition setting while a reference is in progress.
- Data Sync Vote - similar to the (DE) in CPU 0, it is used to handshake between the (DE) options.

The instruction sequence to start an operation is defined with (Aj) = 1/5/11/15.

0012j0	CI, Aj	Clear interrupt
0010jk	CA, Aj Ak	(Ak) = SSD starting Block address ISSUED 1ST
0010jk	CA, Aj Ak	(Ak) = CRAY Y-MP Memory address ISSUED 2nd
0011jk	CL, Aj Ak	(Ak) = Transfer block length ISSUED 3RD
		Read/write mode:
		Transfer initiated
		Wait for interrupt on channel number 1/5/11/15
033i00		Read channel number to Ai
033ij1	Ai CE, Aj	Read Error flags/status word
0012j0	CI, Aj	Clear interrupt/clear interface

Notes

SSD Starting Block Address

All transfers to or from the SSD must be in blocks of 64 words.

CRAY X-MP Memory Address

Address used to transfer information into CRAY Y-MP memory.

Block Length

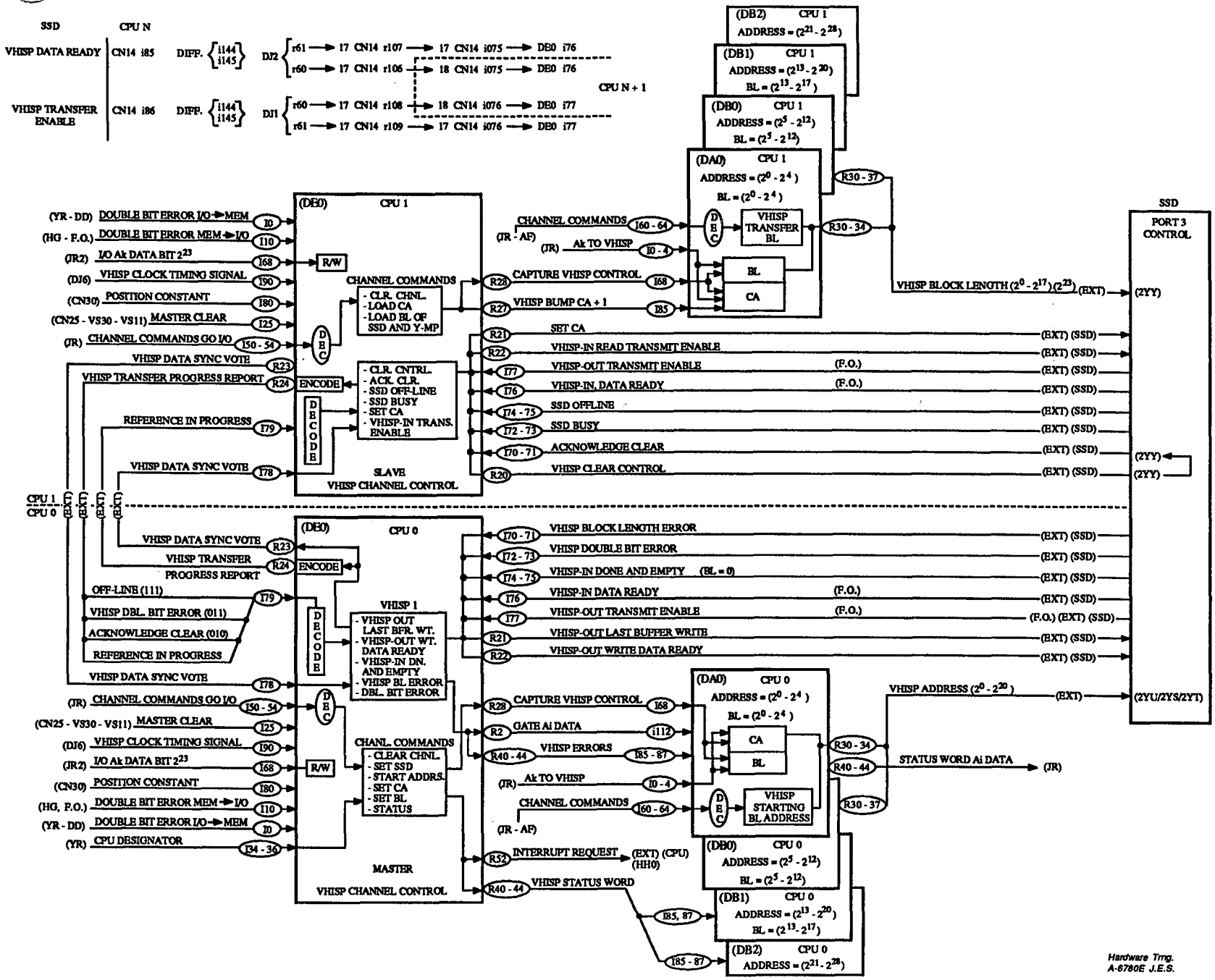
Block length pertains to the number of 64_{10} word blocks to be transferred. The maximum block is 777777_8. If the Ak register bit 2^{23} is cleared Read operation (SSD to CRAY Y-MP memory). If 2^{23} is set, it is a write operation (CRAY Y-MP memory to SSD).

BIT	ORIG	DESCRIPTION	DESTINATION
2^0	(DA0)	VHISP BL = 2^0	(JR0) ~~~> (JS0)
2^1	(DA0)	VHISP BL = 2^1	(JR0) ~~~> (JS0)
2^2	(DA0)	VHISP BL = 2^2	(JR0) ~~~> (JS0)
2^3	(DA0)	VHISP BL = 2^3	(JR0) ~~~> (JS0)
2^4	(DA0)	VHISP BL = 2^4	(JR0) ~~~> (JS0)
2^5	(DB0)	VHISP BL = 2^5	(JR0) ~~~> (JS0)
2^6	(DB0)	VHISP BL = 2^6	(JR0) ~~~> (JS0)
2^7	(DB0)	VHISP BL = 2^7	(JR0) ~~~> (JS0)
2^8	(DB0)	VHISP BL = 2^8	(JR1) ~~~> (JS1)
2^9	(DB0)	VHISP BL = 2^9	(JR1) ~~~> (JS1)
2^{10}	(DB0)	VHISP BL = 2^{10}	(JR1) ~~~> (JS1)
2^{11}	(DB0)	VHISP BL = 2^{11}	(JR1) ~~~> (JS1)
2^{12}	(DB0)	VHISP BL = 2^{12}	(JR1) ~~~> (JS1)
2^{13}	(DB1)	VHISP BL = 2^{13}	(JR1) ~~~> (JS1)
2^{14}	(DB1)	VHISP BL = 2^{14}	(JR1) ~~~> (JS1)
2^{15}	(DB1)	VHISP BL = 2^{15}	(JR1) ~~~> (JS1)
2^{16}	(DB1)	VHISP BL = 2^{16}	(JR2) ~~~> (JS2)
2^{17}	(DB1)	VHISP BL = 2^{17}	(JR2) ~~~> (JS2)
2^{18}	(DE)	VHISP - REQUEST IN PROGRESS	(DB1) → (JR2) ~~~> (JS2)
2^{19}	(DE)	VHISP - BLOCK LENGTH ERROR	(DB1) → (JR2) ~~~> (JS2)
2^{20}	(DE)	VHISP - SSD DOUBLE BIT ERROR	(DB2) → (JR2) ~~~> (JS2)
2^{21}	(DE)	VHISP - CRAY Y-MP DOUBLE BIT ERROR	(DB2) → (JR2) ~~~> (JS2)
2^{22}	(DE)	VHISP - FATAL ERROR	(DB2) → (JR2) ~~~> (JS2)

Hardware Trng.
A-6583 J.E.S.

033ijl where Aj = (1, 5, 11, 15)

CRAY Y-MP VHISP STATUS WORD



CRAY Y-MP VHISP INSTRUCTION (0010 - 0012) BLOCK DIAGRAM

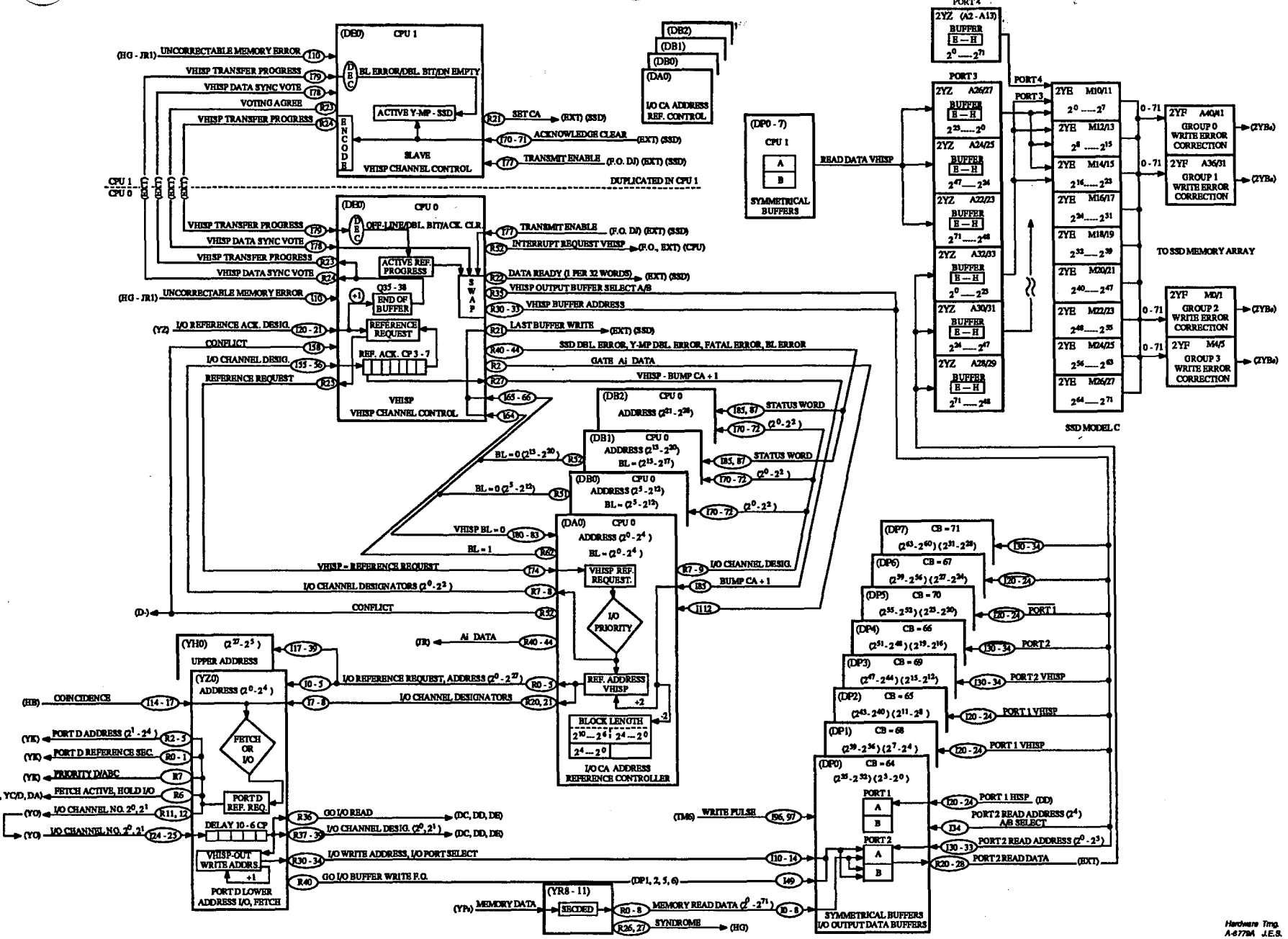
CRAY PROPRIETARY

FIRST (0010jk)			SECOND (0010jk)			(0011jk)					
SSD START BLOCK ADDRESS CPU 0			Y-MP MEMORY ADDRESS CPU 0/1			Y-MP BLOCK LENGTH CPU 0/1			SSD BLOCK LENGTH CPU 1		
BIT	TERM	OPTION	BIT	TERM	OPTION	BIT	TERM	OPTION	BIT	TERM	OPTION
2^0	R30	(DA0)	2^0	A40	(DA0)	2^0	A100	(DA0)	2^0	R30	(DA0)
2^1	R31	(DA0)	2^1	A41	(DA0)	2^1	A101	(DA0)	2^1	R31	(DA0)
2^2	R32	(DA0)	2^2	A42	(DA0)	2^2	A102	(DA0)	2^2	R32	(DA0)
2^3	R33	(DA0)	2^3	A43	(DA0)	2^3	A103	(DA0)	2^3	R33	(DA0)
2^4	R34	(DA0)	2^4	A44	(DA0)	2^4	J44	(DA0)	2^4	R34	(DA0)
2^5	R30	(DB0)	2^5	A40	(DB0)	2^5			2^5	R30	(DB0)
2^6	R31	(DB0)	2^6	A41	(DB0)	$2^6/2^0$	J50	(DA0)	2^6	R31	(DB0)
2^7	R32	(DB0)	2^7	A42	(DB0)	$2^7/2^1$	J51	(DA0)	2^7	R32	(DB0)
2^8	R33	(DB0)	2^8	A43	(DB0)	$2^8/2^2$	J52	(DA0)	2^8	R33	(DB0)
2^9	R34	(DB0)	2^9	A44	(DB0)	$2^9/2^3$	J53	(DA0)	2^9	R34	(DB0)
2^{10}	R35	(DB0)	2^{10}	A45	(DB0)	$2^{10}/2^4$	J54	(DA0)	2^{10}	R35	(DB0)
2^{11}	R36	(DB0)	2^{11}	A46	(DB0)	$2^{11}/2^5$	J40	(DB0)	2^{11}	R36	(DB0)
2^{12}	R37	(DB0)	2^{12}	A47	(DB0)	$2^{12}/2^6$	J41	(DB0)	2^{12}	R37	(DB0)
2^{13}	R30	(DB1)	2^{13}	A40	(DB1)	$2^{13}/2^7$	J42	(DB0)	2^{13}	R30	(DB1)
2^{14}	R31	(DB1)	2^{14}	A41	(DB1)	$2^{14}/2^8$	J43	(DB0)	2^{14}	R31	(DB1)
2^{15}	R32	(DB1)	2^{15}	A42	(DB1)	$2^{15}/2^9$	J44	(DB0)	2^{15}	R32	(DB1)
2^{16}	R33	(DB1)	2^{16}	A43	(DB1)	$2^{16}/2^{10}$	J45	(DB0)	2^{16}	R33	(DB1)
2^{17}	R34	(DB1)	2^{17}	A44	(DB1)	$2^{17}/2^{11}$	J46	(DB0)	2^{17}	R34	(DB1)
2^{18}	R35	(DB1)	2^{18}	A45	(DB1)	$2^{18}/2^{12}$	J47	(DB0)			
2^{19}	R36	(DB1)	2^{19}	A46	(DB1)	$2^{19}/2^{13}$	J40	(DB0)			
2^{20}	R37	(DB1)	2^{20}	A47	(DB1)	$2^{20}/2^{14}$	J41	(DB1)			
2^{21}	R30	(DB2)	2^{21}	A40	(DB2)	$2^{21}/2^{15}$	J42	(DB1)			
2^{22}	R31	(DB2)	2^{22}	A41	(DB2)	$2^{22}/2^{16}$	J43	(DB1)			
2^{23}	R32	(DB2)	2^{23}	A42	(DB2)	$2^{23}/2^{17}$	J44	(DB1)	2^{23}	R32	(DB2)
2^{24}	R33	(DB2)	2^{24}	A43	(DB2)	$2^{24}/$				R/W	
2^{25}	R34	(DB2)	2^{25}	A44	(DB2)	$2^{25}/$			$2^{23}=1$	WRITE TO SSD	
2^{26}	R35	(DB2)	2^{26}	A45	(DB2)	$2^{26}/$			$2^{23}=0$	READ OF SSD	
2^{27}	R36	(DB2)	2^{27}	A46	(DB2)	$2^{27}/$					
2^{28}	R37	(DB2)	2^{28}	A47	(DB2)	$2^{28}/$					
						$/2^{23}$	Q04	(DE0)			

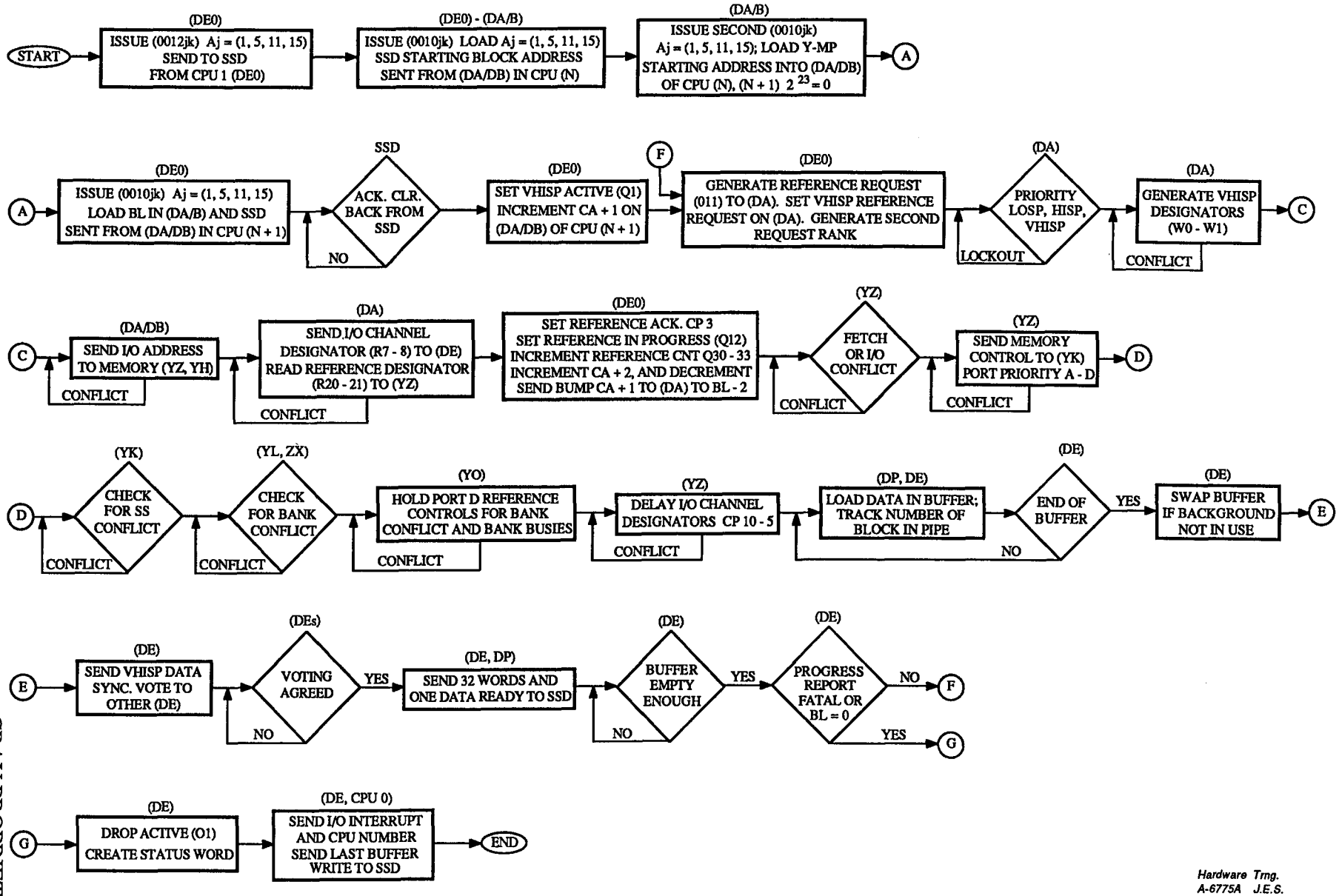
Hardware Trng.
A-6773A J.E.S.

CRAY Y-MP SSD STARTING BLOCK ADDRESS/BLOCK LENGTH

CRAY PROPRIETARY



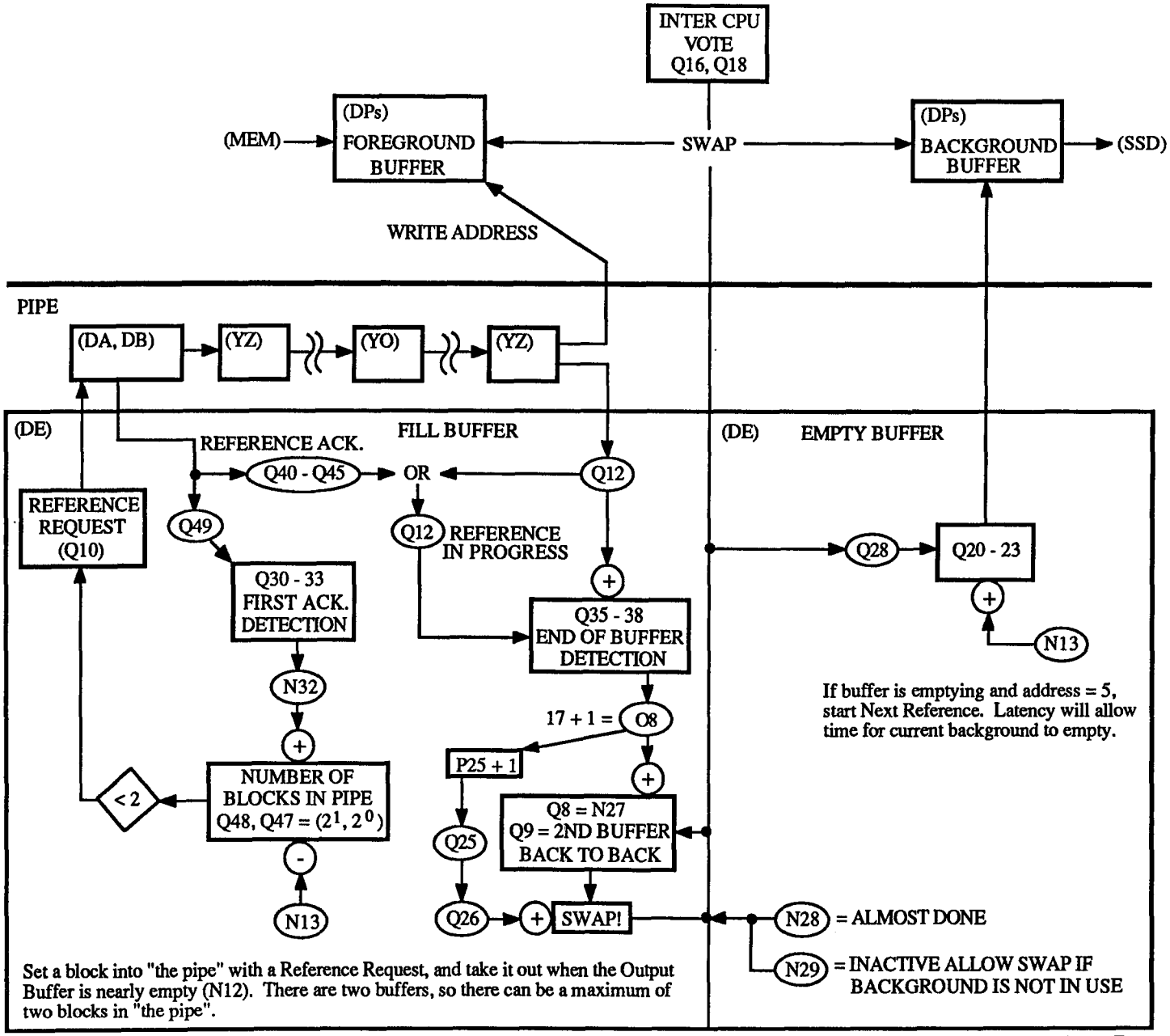
CRAY Y-MP VHISP-OUT Y-MP→SSD BLOCK DIAGRAM



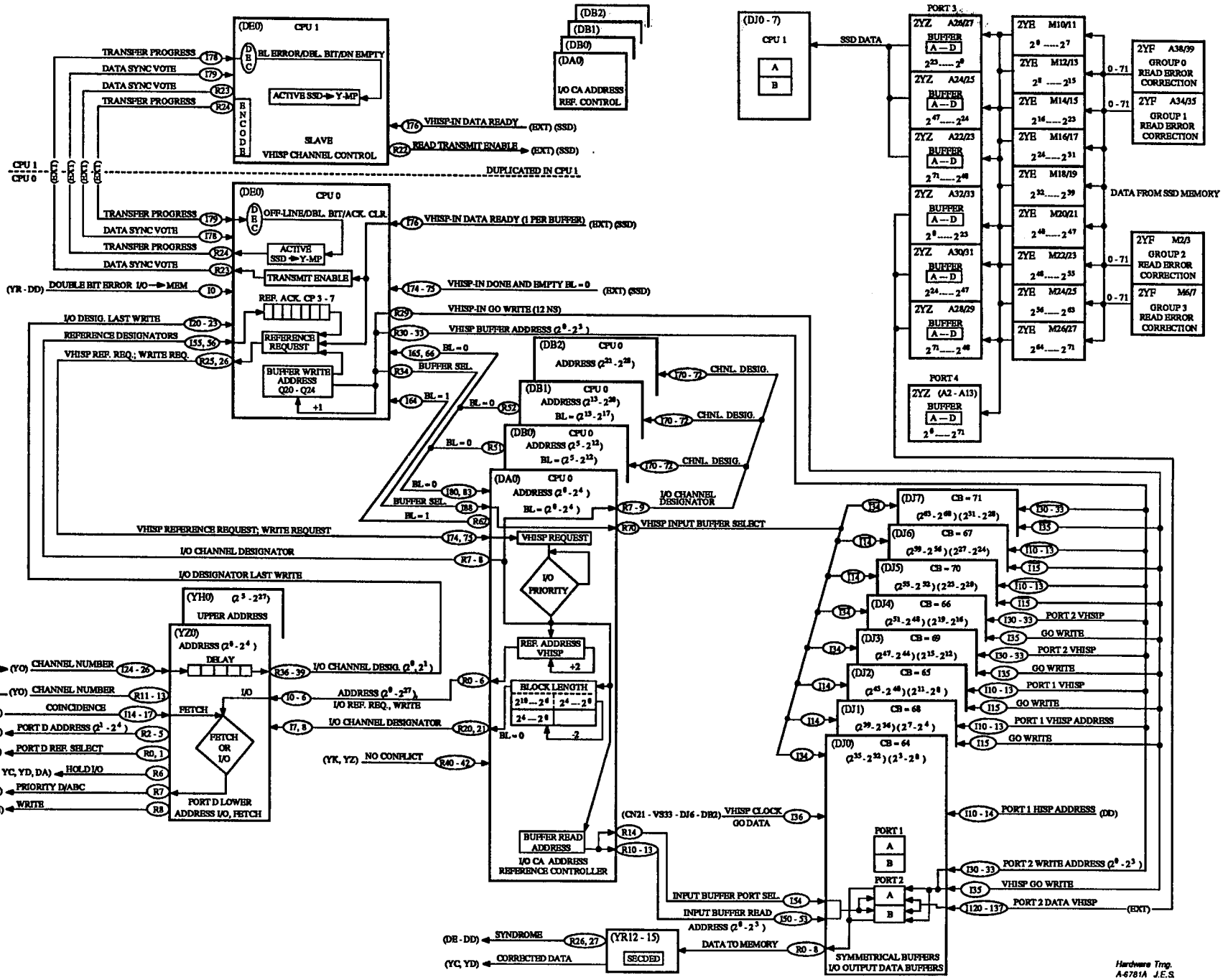
Hardware Trng.
A-6775A J.E.S.

CRAY PROPRIETARY

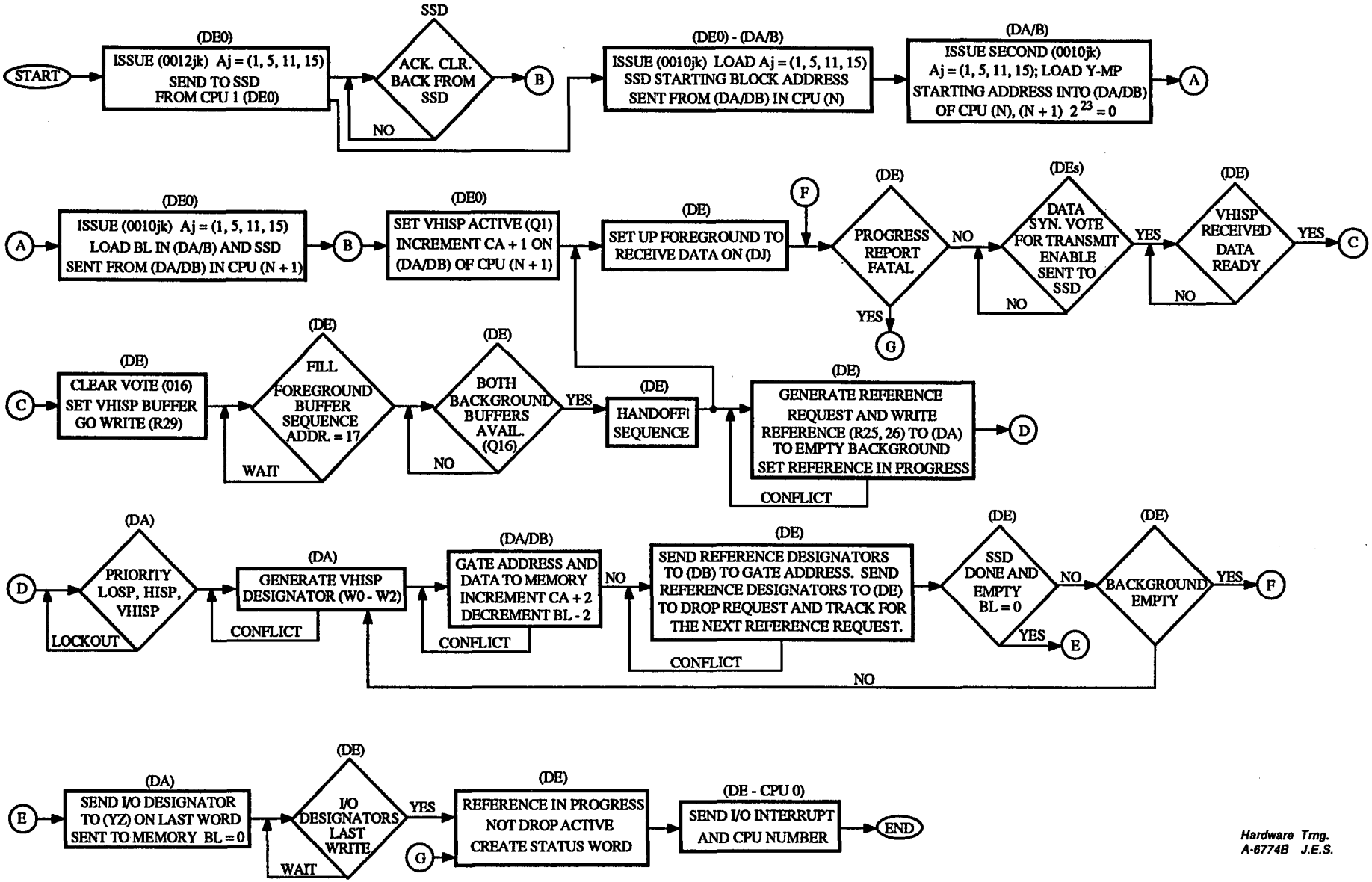
CRAY Y-MP VHISP-OUT READ Y-MP → SSD FLOW CHART



CRAY Y-MP VHISP-OUT BUFFER CONTROL



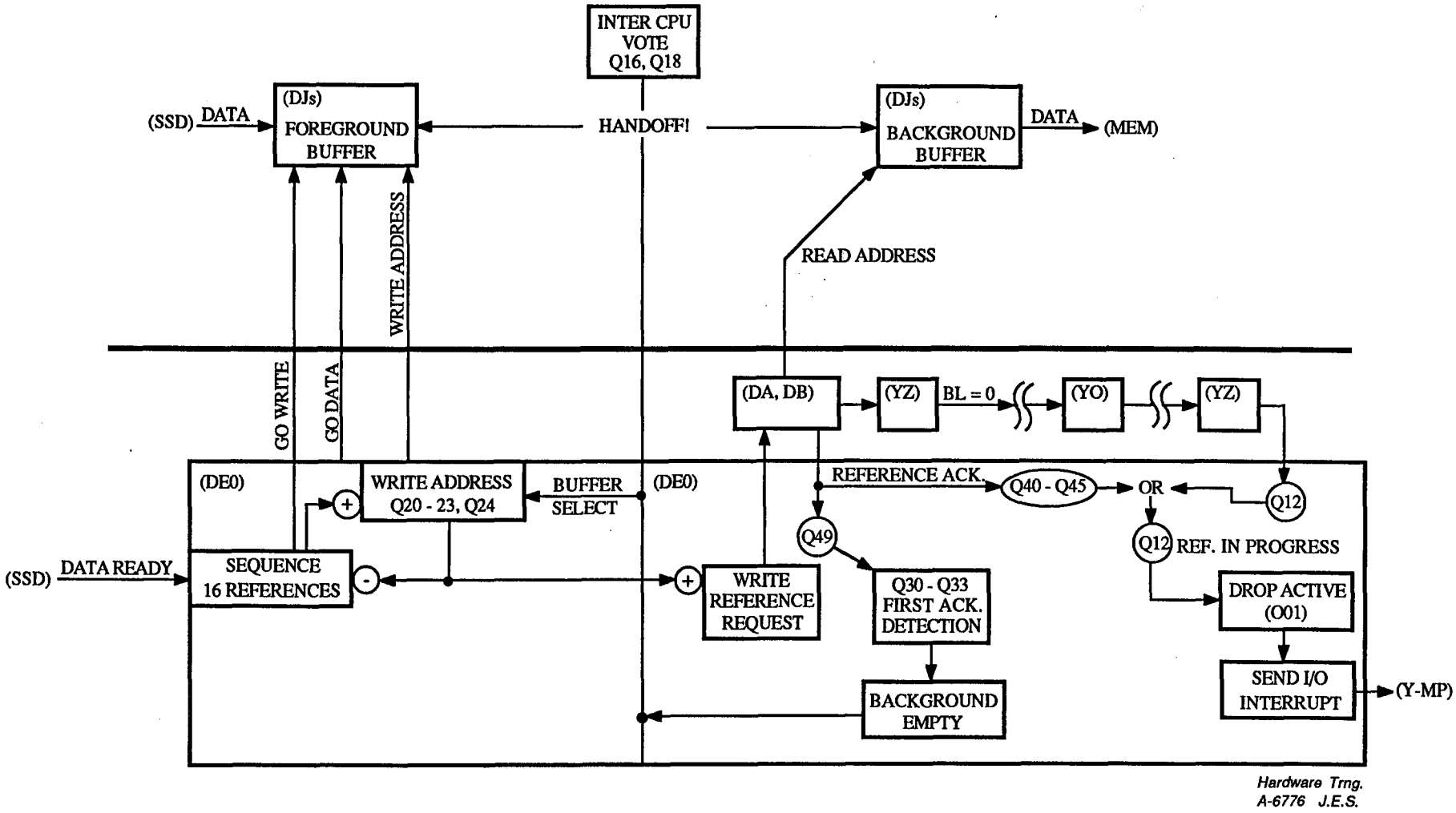
CRAY Y-MP VHISP-IN S[illegible]→Y-MP BLOCK DIAGRAM



Hardware Trng.
A-6774B J.E.S.

CRAY Y-MP VHISP-IN READ SSD →Y-MP FLOW CHART

CRAY PROPRIETARY



CRAY Y-MP VHISP-IN BUFFER CONTROL SSD → Y-MP

CRAY PROPRIETARY

SSD "MODEL C"

SECTION 0	SECTION 1	SECTION 2	
RAS	RAS	RAS	
0	0	0	
1	1	1	
2	2	2	
3	3	3	GROUP 0
4	4	4	
5	5	5	
6	6	6	
7	7	7	
SECTION 3 RAS 01234567			
SECTION 0	SECTION 1	SECTION 2	
RAS	RAS	RAS	
0	0	0	
1	1	1	
2	2	2	
3	3	3	GROUP 1
4	4	4	
5	5	5	
6	6	6	
7	7	7	
SECTION 3 RAS 01234567			
SECTION 0	SECTION 1	SECTION 2	
RAS	RAS	RAS	
0	0	0	
1	1	1	
2	2	2	
3	3	3	GROUP 2
4	4	4	
5	5	5	
6	6	6	
7	7	7	
SECTION 3 RAS 01234567			
SECTION 0	SECTION 1	SECTION 2	
RAS	RAS	RAS	
0	0	0	
1	1	1	
2	2	2	
3	3	3	GROUP 3
4	4	4	
5	5	5	
6	6	6	
7	7	7	
SECTION 3 RAS 01234567			

ADDRESS $2^8, 2^7, 2^6, 2^5$

2^5 - STARTS WITH ZERO

$2^7, 2^6$ - ROW ADDRESS STROBE

2^8 - GROUP BIT

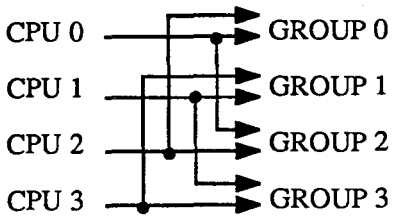
Four words written/read per RAS per section.

GROUP BIT = 0

CPU 0 → GROUP 0
CPU 1 → GROUP 1
CPU 2 → GROUP 2
CPU 3 → GROUP 3

GROUP BIT = 1

CPU 0 → GROUP 2
CPU 1 → GROUP 3
CPU 2 → GROUP 0
CPU 3 → GROUP 1



A-6701

ERROR LOGGING

Each CPU module records errors unique to that CPU and sends a copy of the error information to the Error Logger. Three types of errors are recorded by the CPU; they are: Memory Errors, Register Parity Errors and Input HISP/VHISP Parity Errors.

Memory Errors and Register Parity Errors are logged in the (HG0) option as they occur in words one through five of the exchange package area. When an exchange occurs, a copy of the current (HG0) error information is sent to the (HH0) option for submittal to the Error Logger. This information is transferred as five 8-bit bytes. Based on the error type (Memory, RPE), only certain bytes of these five will be held on the (HH0) option. Bytes that are not needed may either be overwritten or simply not latched upon receipt at the (HH0) option. Memory Error and Register Parity Error information is not sent to the (HH0) option as it occurs, but is instead copied to the (HH0) option only during an exchange. Since Uncorrectable Memory Errors are probably Register Parity Errors, they will usually cause an interrupt with a subsequent exchange, these errors will be logged as detected. Correctable Memory Errors, however, will usually only be logged due to an exchange for some reason such as a PCI, EXIT, ORE, etc.; unless the Interrupt on the Correctable Memory Error Mode bit is set. HISP and VHISP Input Parity Errors are sent directly to the (HH0) option upon detection from the (YR12 - YR15) options. These I/O Errors are logged as they are received.

All CPU modules share a single error channel path to the system Error Logger. The sharing of this single path requires fanin and control circuits as follows: Each CPUs (HH0) option upon receiving error information from its own (HG0) option or the I/O channels mounted on that module will send an error channel request to the CPU 5 (HH0) option. Error channel allocation is on a first come, first serve basis. In the event of a tie, priority is given based on the physical CPU slot with CPU 0 getting the highest priority and CPU 7 getting the lowest.

The CPU 5 (HH0) option controls the usage of the error channel and will issue a Go Error Channel to the CPU requesting the channel and having the highest priority. This requesting CPU will then set an active flag and send the first of eight 4-bit data transfers to the Error Logger. Data bit 0 will go via the CPU 0 (HH0) option, data bit 1 via the CPU 1 (HH0) option, data bit 2 via the CPU 6 (HH0) option, and data bit 3 via the CPU 7 (HH0) option. These four (HH0) options provide an 8 to 1 fanin for any of the eight CPUs to transfer data to the Error Logger.

The first 4-bit data transfer from any CPU will always have a single bit set, as it has one bit identifying a Correctable Memory Error, one bit identifying an Uncorrectable Memory Error, one bit identifying a Register Parity Error, and one bit identifying an I/O Error. This first 4-bit data transfer starts an Error Logger input cycle. The Error Logger will answer the CRAY Y-MP by sending eight Error Logger Resume signals, one per 4-bit data transfer. Each resume signal is routed via the CPU 0 (HH0) option and fanned out in the CPU 1 (HH0) option. The resume is then broadcast to all eight CPU (HH0) options. Only the currently active CPU (HH0) option will use this signal by sending 4-bit data transfers to the Error Logger in response to each of the Error Logger Resume signals. The first two 4-bit data transfers will always contain the error type (Memory, RPE, I/O) and the physical processor number (0 - 7) reporting the error. The remaining six 4-bit data transfers vary in content based upon the error type.

Upon receipt of the eighth Error Logger Resume signal, the active CPU (HH0) option resets his active flag in order to log another set of exchange package error information from the (HG0) option. This (HH0) option also sends an Error Channel Complete signal to the CPU 5 (HH0) option to free up the error channel for another CPU to use.

YM03/08A

ERROR LOGGER OPTIONS INVOLVED

(HG0) Option

The (HG) option collects the information which is sent to the Error Logger from the exchange parameters, which include the Syndrome, Bank, Chip Select, Ports and Error type. The exchange parameters are sent to the (HH) only during an Exchange Sequence. When a UCME, CME or RPE is set, the (HG) will pass a Go Error to the (HH) option along with the exchange parameters. Upon receiving the Go Error, the (HH) will know the exchange parameters need to be sent to the Error Logger

(HH0) CPU A - H Option

The (HH) option receives and collects information from the exchange parameter as well as from the HISP/VHISP Input Error syndrome. The (HH) with the error information from CPU A - H will pass an error channel request to the (HH0) in CPU 5 location. The (HH) at CPU 5 will make sure that the (HH) from CPU A - H is the lowest numbered CPU and will pass a Go Error Channel back to the CPU that is the lowest number. The (HH) that receives the Go Error Channel from the (HH) at CPU 5 will then start passing its collected error information to the Error Logger.

(HH0) CPU 0, 1, 6, 7 Option

The (HH) options in CPU location 0, 1, 6, 7 are responsible for passing the error information to the Error Logger. (HH0) in CPU 0 sends bit 2^0, while (HH0) in CPU 1 sends bit 2^1, along with (HH0) in CPU 6 sending bit 2^2, and finally (HH0) in location CPU 7 will send bit 2^3. The (HH0) will do a 8 to 1 fanin from all the other CPUs (HH0) options.

(HH0) CPU 5 Option

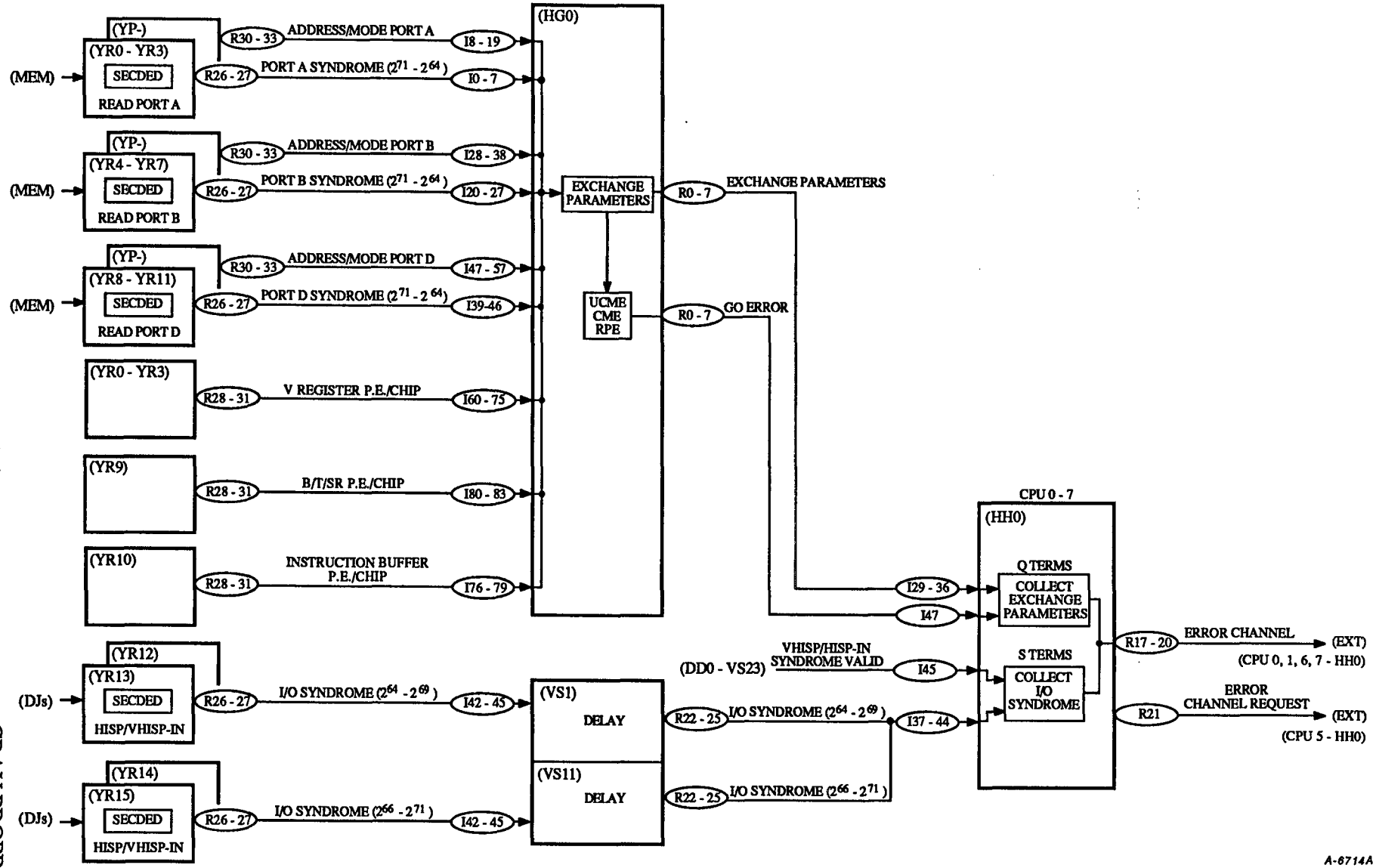
The (HH0) in CPU 5 will become the control as to which CPU will be sending its information to the Error Logger. The CPU with the lowest letter would have the highest priority. This is the same priority used to send the lowest numbered interrupting channel number.

(HH0) CPU 0 Option

The (HH0) in location CPU 0 will receive the hand shakes resume and reset from the Error Logger. Resume is to allow another 4 bits to be sent, while a reset indicates that the transfer is complete.

(HH0) CPU 1, CPU 6 Option

(HH0) CPU 1 performs a 1 to 8 fanout of resume to the eight CPUs (HH0) options, while (HH0) at CPU 6 performs a 1 to 8 fanout of the reset to the eight CPUs (HH0) options.



CRAY Y-MP COLLECT ERROR LOGGER INFORMATION
(HH0) CPU 0 - 7

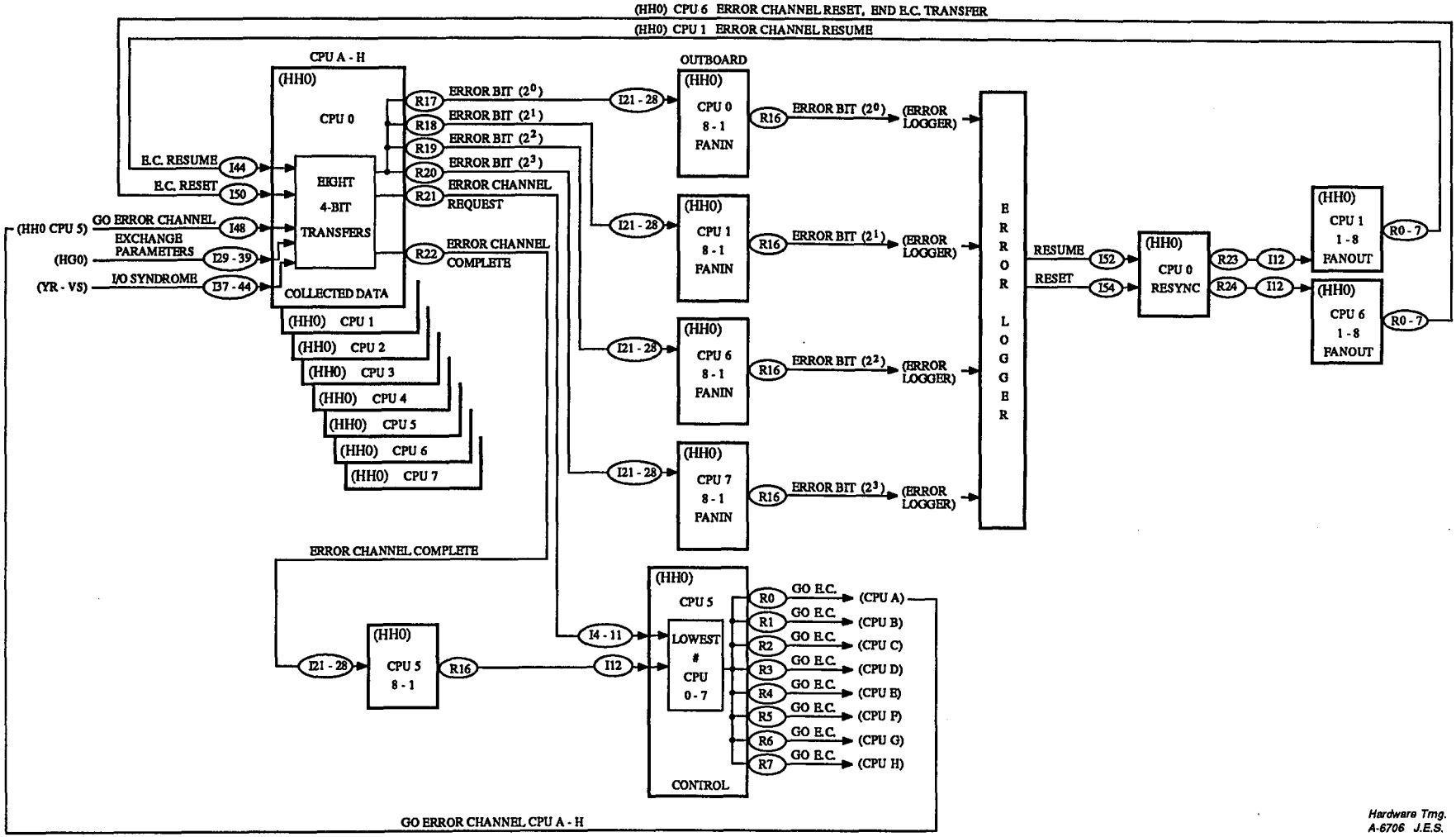
CRAY PROPRIETARY

CRAY PROPRIETARY

MEMORY ERROR SENT TO (HH) FROM (HG)	Q TERMS 7	6	5	4	Q TERMS 3	2	1	0	Q TERMS 17	16	15	14	Q TERMS 13	12	11	10	Q TERMS 18	27	26	25	24	23	22	21	PROC. # PPN 2^2	2^1	2^0	Z999	I/O ERROR	RPE	# TENANCY OF (0:15)
SYNDROME (X0)	7	6	5	4	3	2	1	0																	Z2	Z1	Z0		S12	V22	
													MEMORY ERROR SENT TO (HH) FROM (HG)																			
BANK (X1)									7	6	5	4	3	2	1	0															CDTVA EMBOSNUS N-(SP-1H 1 S10
CHIP SELECT (X2)																	C.S.														
TYPE/PORT MODE (X3)																		UC ME	CME	PORT D	B	A	RM 2^1	RM 2^0						
													WRITE TO ERROR LOGGER 4-BIT TRANSFERS (2^3-2^0)																		
1ST TRANSFER = U0																		2^1	2^0										2^3	2^2	
2ND TRANSFER = U1																									2^2	2^1	2^0		2^3		
3RD TRANSFER = U2					2^3	2^2	2^1	2^0																							
4TH TRANSFER = U3	2^3	2^2	2^1	2^0																											
5TH TRANSFER = U4													2^3	2^2	2^1	2^0															
6TH TRANSFER = U5									2^3	2^2	2^1	2^0																			
7TH TRANSFER = U6																	2^0						2^3	2^2				2^1			
8TH TRANSFER = U7																				2^2	2^1	2^0									2^3
REGISTER	Z999	Z999	FUN.	FUN.	FUN.	CHIP	CHIP	CHIP			REGISTER PARITY ERROR 5TH BYTE TRANSFER																				
PARITY CHIP #																															
X0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0																							
U100					2^3	2^2	2^1	2^0																							
U101	2^7	2^6	2^5	2^4																											
																		I/O SYNDROME ERROR VHISP/HISP													
																		V21	V20												
1ST TRANSFER = U0																		2^1	2^0										2^3	2^2	
2ND TRANSFER = U1																									2^2	2^1	2^0		2^3		
3RD TRANSFER = U2					2^3	2^2	2^1	2^0																							
4TH TRANSFER = U3	2^3	2^2	2^1	2^0																											
5TH TRANSFER = U4													2^3	2^2	2^1	2^0															
6TH TRANSFER = U5									2^3	2^2	2^1	2^0																			
7TH TRANSFER = U6																	2^0						2^3	2^2				2^1			
8TH TRANSFER = U7																				2^2	2^1	2^0									2^3

Hardware Trng.
A-6707A J.E.S.

CRAY Y-MP ERROR LOGGER ASSEMBLY AND DISASSEMBLY
MEMORY ERROR SENT TO (HH)



CRAY Y-MP ERROR LOGGER BLOCK DIAGRAM

SAFETY PRACTICES

All Field Engineers are expected to follow reasonable and appropriate precautions with respect to electrical, mechanical, and personal safety hazards while working on computer system equipment. You should pay careful attention to entries in the maintenance documentation labeled "DANGER" or "WARNING", which identify hazardous areas or procedures encountered in maintaining the system equipment. The following additional procedures should be followed when working on equipment:

Personal

1. You are responsible for insuring that no action on your part causes unsafe conditions that may expose personnel to hazards in any device.

2. You should never work alone on equipment having exposed operating mechanical parts or exposed hazardous power components. If you MUST do so, notify your EIC or manager. In any case, the following precautions must be observed:

   a. Someone familiar with the power-off controls must be in the immediate area.

   b. Personal jewelry (rings, wristwatches, bracelets, necklaces, etc.) shall be removed. A small box in the FE tool kit makes a good storage place for these items.

   c. If using one hand, keep the other one in your pocket.

   d. Avoid wearing loose articles of clothing that can be snagged and drawn into moving machinery. Wear short-sleeve shirts or pull sleeves above the elbow. Neckties, where required, should be tucked in between the second and third shirt button or fastened about three inches from the end with a tietack or tieclasp, preferably non-conductive. Don't use tie chains. Clip-on type neckties are preferable to the regular ones; if caught they will pull free without causing injury.

   f. While working in equipment put red tape strips across any power controls, or use "DO NOT OPERATE" tags where available.

3. Keep FE tool kits out of walkways; put them on or under a desk or table.

4. Put doors and covers removed from a machine in a safe, out-of-the-way location where nobody will trip over them or cause them to fall on top of someone. All machine covers MUST be restored in place before the machine is returned to the customer.

5. All safety covers, guards, shields, groundstraps, panels, etc., shall be properly reinstalled after maintenance is finished.

6. Maintain good housekeeping practices during and following each maintenance activity. Do not permit tools, manuals, wipers, paper, trash and the like to accumulate in the work area, and CLEAN UP AFTER YOURSELF.

Electrical

1. Remove ALL AC and DC power when removing or installing major assemblies, working inside power supplies or power control enclosures, performing detailed mechanical maintenance procedures, or doing wiring and/or module changes in the machine. If possible, turn off and lock or tag the circuit breaker in the service panel on the wall; unplug the main power supply cord.

2. Only use well-insulated pliers, screwdrivers, test leads, etc., when working on or near live circuits.

3. Do not disconnect or otherwise disable safety grounding systems even if the equipment is powered off. These are installed for YOUR protection.

4. Avoid coming in contact with grounds, such as equipment frames, metal floor tile edgings, electrical conduits, and the like. If possible, locally purchase rubber or vinyl mats.

Mechanical

1. Do not use chemicals, greases, oils or solvents that have not been specifically approved by the equipment manufacturer for that device. His recommendations are usually based on extensive experience with this equipment in service.

2. Use the proper tools for the job. Improper use of tools can result in personal injury or equipment damage.

3. Replace worn or broken tools or test equipment as quickly as possible.

4. If the machine is running, DO NOT reach in to the works; remember, they are YOUR fingers and you only get one set per lifetime.

5. If using a strobelight on mechanical devices, DON'T TOUCH ANYTHING; it may be moving.

6. Safety glasses or goggles must be used if you are:

   a. Driving pins, riveting, swedging, and similar activities.

   b. Using an electric drill, grinder, reamer, etc.

   c. Installing or removing springs under tension or compression.

   d. Using any type of solvent, spray, or chemical for cleaning or touch-up painting.

   e. Any other activity which may endanger the eyes. They are YOUR eyes, and you need them for this type of work.

7. When lifting, use a method (bent knees, straight back) that will not injure the spine or strain back muscles. Be realistic as to what your capacity for lifting really is.

ABOVE ALL, USE GOOD JUDGEMENT AND COMMON SENSE - A MOMENT OF THOUGHT BEFORE YOU ACT CAN SAVE HOURS OF AGONIZING AFTERTHOUGHT.

SAFETY PRECAUTIONS

Power Tool Precautions

Power tools can be hazardous when improperly used. There are several types of power tools, based on the power source they use: electric, pneumatic, liquid fuel, hydraulic, and powder-actuated.

Employees should be trained in the use and limitations of their power tools. They should understand potential hazards and safety precautions to prevent those hazards from occurring.

The following general precautions should be observed by power tool users:

- Never carry a tool by the cord or hose.
- Never yank the cord or the hose to disconnect from the receptacle.
- Keep cords and hoses away from heat, oil, and sharp edges.
- Disconnect tools when not in use, before servicing, and when changing accessories such a blades, bits, and cutters.
- All observers should be kept at a safe distance from the work area.
- Secure work with clamps or a vise, freeing both hands to operate the tool.
- Avoid accidental starting. The worker should not hold a finger on the switch button while carrying a plugged-in tool.
- Tools should be maintained with care. They should be kept sharp and clean for the best performance. Follow instructions in the user's manual for lubricating and changing accessories.
- Be sure to keep good footing and maintain good balance.
- The proper apparel should be worn. Loose clothing, ties, or jewelry can become caught in moving parts.
- All portable electric tools that are damaged must be removed, or tagged "DO NOT USE."

Guards

Hazardous moving parts of a power tool need to be safeguarded. For example, belts, gears, shafts, pulley, sprockets, spindles, drums, fly wheels, chains, or other reciprocating, rotating or moving parts of equipment must be guarded if such parts are exposed to contact by employees.

Guards, as necessary, should be provided to protect the operator and others from:

- Point of operation
- In-running nip points, rotating parts
- Flying chips and sparks

Safety guards must never be removed when the tool is being used. For example, portable circular saws must be equipped with guards. An upper guard must cover the entire blade of the saw. A retractable lower guard must cover the teeth of the saw, except when it makes contact with the work material. The lower guard must automatically return to the covering position when the tool is withdrawn from the work.

Safety Switches

The following hand-held powered tools must be equipped with a momentary contact "on/off" control switch: drills, tappers, fastener drivers, horizontal, vertical and angle grinders with wheels larger than two inches in diameter, disc sanders, belt sanders, reciprocating saws, saber saws, and other similar operations. These tools may also be equipped with a lock-on control provided that turn off can be accomplished by a single motion of the same finger or fingers that turn it on.

The following hand-held powered tools may be equipped wi'h only a positive "On-off" control switch: platen sanders, grinders with wheels two inches or less in diameter, routers, planers, laminate trimmers, nibblers, shears, scroll saws, and jigsaws with blade shanks one-fourth inch wide or less.

Other hand-held powered tools such as circular saws, chain saws, and percussion tools without positive accessory holding means must be equipped with a constant pressure switch that will shut off the power when the pressure is released.

Electric Tools

Employees using electric tools must be aware of several dangers; the most serious is the possibility of electrocution.

Among the chief hazards of electric-powered tools are burns and slight shocks which can lead to injuries or even heart failure. Under certain conditions, even a slight shock can result in fibrillation of the heart and eventual death. A shock also can cause the user to fall off a ladder or other elevated work surface.

To protect the user from shock, tools must either have a three-wire cord with ground or be double insulated. Three-wire cords contain two current-carrying conductors and a grounding conductor. One end of the grounding conductor connects to the tool's metal housing. The other end is grounded through a prong on the plug. Anytime an adapter is used to accommodate a two-hole receptacle, the adapter wire must be attached to a known ground. The third prong should never be removed from the plug.

Double insulation is more convenient. The user and the tools are protected in two ways: by normal insulation on the wires inside, and by a housing that cannot conduct electricity to the operator in the event of a malfunction.

The following general practices should be followed when using electric tools:

- Electric tools should be operated within their design limitations.
- Gloves and safety footwear are recommended during use of electric tools.
- When not in use tools should be stored in a dry place.
- Electric tools should not be used in damp or wet locations.
- Work areas should be well lighted.

ESD (ELECTROSTATIC DISCHARGE) PROTECTION

The purpose of this guideline is to identify those steps taken by personnel to try to prevent damage to electronic devices due to electrostatic discharge. This guideline applies to all persons working on any electronic machines (Cray chassis, disk drives, expanders, microcomputers, etc.), modules and boards.

All persons working the checkout bays should be familiar with and follow these guidelines.

Wrist Straps

A wrist strap connected to a suitable ground will be worn whenever a module or board is being handled outside of the machine and not in a protective container. This applies to Cray chassis, disk drives, microcomputers, etc. Specifically:

- whenever a module/board is being inserted, removed, or being scoped while extended
- whenever a module is being handled at a static free workstation or teststation
- whenever an unprotected module is being removed or put into the spares cabinet

The wrist straps used are expandable with stainless steel fibers interwoven into the material. After a while, the steel fibers break and the wrist strap becomes ineffective. The straps should be checked bi-weekly and those that are intermittent or open should not be used.

Static Free Workstation

A static free workstation is a work area whose surface has been covered with a static dissipative material connected to ground.

All checkout bays have at least one static free workstation. Also, the module teststation is a static free workstation.

The static free workstations are for working on modules and should be kept clean of books and paper. The only things permitted on the static free workstations are modules, tools, and those documents necessary for the work being done at that time.

The static free workstation is the only place where any work will be done on a module (ohming, tear down, etc.). Laying the unprotective module/board on a power supply bay cushion or laying a static protective bag between the module and cushion while ohming is prohibited.

Static Dissipative Footwear

Static dissipative footwear are shoes, shoe covers, or heel straps made of conductive material used to prevent charge build-up when employing conductive flooring.

Whenever working on equipment in the checkout bays or rooms with conductive flooring, static dissipative footwear will be worn.

Personnel who have been provided the HYTEST static dissipative shoes, should wear the shoes in the checkout bays. Personnel temporarily testing equipment in the checkout bays (Field Engineers, Training, etc.), if they do not have the approved HYTEST shoes, they will need to wear heel straps. Field Engineers can obtain an authorization slip for HYTEST shoes from Kathy Eisold, Hardware Training, as required.

Although leather is conductive, leather soled shoes are not necessarily static dissipative and are not acceptable substitutes for approved static dissipative footwear.

Static Shielding Containers

When modules are carried between buildings, the covered aluminum cans or the module boxes will be used.

When modules are carried between bays in the same building, the static shielding bags will be used. To be effective, the bags should not be used like a "potholder". The module should be placed inside the bag and the bag opening folded over. Bags that are torn or have a lot of punctures should be discarded and not used.

Pink polyurethane material/bags are anti-static (they cannot generate a charge on their surface), but they are not static shielding. The pink polyurethane material can be wrapped around the module and then the module put into the shielding bag. However, using the pink polyurethane packing material instead of the static shielding bag is not satisfactory.

Smocks

All personnel working in a computer bay or a training lab will be expected to wear an anti-static smock. Each individual should have one of their own. For Field Engineers who do not have smocks, see Kathy Eisold in Hardware Training.

SHOCKS

How Shocks Occur

Electricity travels in closed circuits, and its normal route is through a conductor. Shock occurs when the body becomes a part of the electrical circuit. The current must enter the body at one point and leave at another. Shock normally occurs in one of three ways. The person must come in contact with the following: both wires of the electrical circuit; one wire of an energized circuit and the ground; or a metallic part that has become "hot" by being in contact with an energized wire, while the person is also in contact with the ground.

The metal parts of electrical tools and machines may become "hot" if there is a break in the insulation of the tool or machine wiring. The worker using these tools and machines is made less vulnerable to electrical shock when a low-resistance path from the metallic case of the tool or machine to the ground is established. This is done through the use of an equipment grounding conductor - a low-resistance wire that causes the unwanted current to pass directly to the ground rather than through the body of the person in contact with the tool or machine. If the equipment grounding conductor has been properly installed, it has a low resistance to ground, and the worker is being protected.

Severity of the Shock

The severity of the shock received when a person becomes a part of an electrical circuit is affected by three primary factors: the amount of current flowing through the body (measured in amperes); the path of the current through the body; and the length of time the body is in the circuit. Other factors which may affect the severity of shock are the frequency of the current, the phase of the heart cycle when shock occurs, and the general health of the person prior to shock.

The effects from electric shock depend upon the type of circuit, its voltage, resistance, amperage, pathway through the body, and duration of the contact. Effects can range from a barely perceptible tingle to immediate cardiac arrest. Although there are not absolute limits or even known values which show the exact injury from any given amperage, the following table shows the general relationship between the degree of injury and amount of amperage for a 60-cycle hand-to-foot path of one second's duration of shock.

Effects of Electrical Current in the Human Body

CURRENT	REACTION
1 Milliampere	Preception level. Just a faint tingle.
5 Milliamperes	Slight shock felt; not painful but disturbing. Average individual can let go. However, strong involuntary reactions to shocks in this range can lead to injuries.
6-25 Milliamperes (women)	Painful shock, muscular control is lost.
9-30 Milliamperes (men)	This is called the freezing current or "let-go" range.
50-150 Milliamperes	Extreme pain, respiratory arrest, severe muscular contractions*. Individual cannot let go. Death is possible.
1,000-4,300 Milliamperes	Ventricular fibrillation. (The rhythmic pumping action of the heart ceases.) Muscular contraction and nerve damage occur. Death is most likely.
10,000 Milliamperes	Cardiac arrest, severe burns and probable death.

* If the extensor muscles are excited by the shock, the person may be thrown away from the circuit.

PERSONAL PROTECTION

Arm and Hand Protection

Examples of injuries to arms and hands are burns, cuts, electrical shock, amputation, and absorption of chemicals.

There is a wide assortment of gloves, hand pads, sleeves, and wristlets for protection from various hazardous situations.

The protective device should be selected to fit the job. For example, some gloves are designed to protect against specific chemical hazards. Employees may need to use gloves which have been tested and provide insulation from burns and cuts such as wire mesh, leather, and canvas. The employee should become acquainted with the limitations of the clothing used.

Ear Protection

Exposure to high noise levels can cause hearing loss or impairment. It can create physical and psychological stress. There is no cure for noise-induced hearing loss, so the prevention of excessive noise exposure is the only way to avoid hearing damage. Specifically designed protection is required, depending on the type of noise encountered.

Performed or molded ear plugs should be individually fitted by a professional. Waxed cotton, foam, or fiberglass wool earplugs are self-forming. When properly inserted, they work as well as most molded earplugs.

Some earplugs are disposable, to be used one time and then thrown away. The non-disposable type should be cleaned after each use for proper protection. Plain cotton is ineffective as protection against hazardous noise.

Earmuffs need to make a perfect seal around the ear to be effective. Glasses, long sideburns, long hair and facial movements, such as chewing, can reduce protection. Special equipment is available for use with glasses or beards.

For extremely noisy situations, earplugs should be worn in addition to earmuffs. When used together, earplugs and earmuffs change the nature of sounds; all sounds are reduced including one's own voice, but other voices or warning signals are easier to hear.

Eye and Face Protection

Eye and face protective equipment is required by OSHA where there is a reasonable probability of preventable injury when such equipment is used. Employers must provide a type of protector suitable for work to be performed and employees must use the protectors. These stipulations apply also to supervisors, management personnel, and visitors while they are in hazardous areas.

The BLS study found that about 60 percent of workers who suffered eye injuries were not wearing eye protective equipment. When asked why they were not wearing face protection at the time of the accident, workers indicated that face protection was not normally used or practiced in

their type of work, or it was not required for the type of work performed at the time of the accident.

Suitable eye protectors must be provided where machines or operations present the hazard of flying objects, glare, liquids, injurious radiation, or a combination of these hazards. Protectors must meet the following minimum requirements:

- Provide adequate protection against particular hazards for which they are designed.
- Be reasonably comfortable when worn under the designated conditions.
- Fit snugly without interfering with the movements or vision of the wearer.
- Be durable.
- Be capable of being disinfected.
- Be easily cleanable.
- Be kept clean and in good repair.

Respiratory Protection

Hazards to the lungs are not always easy to detect. Some of the most common hazards are lack of oxygen and presence of harmful dust, fogs, smokes, mists, fumes, gases, vapors, or sprays including substances which may cause cancer, lung impairment, other diseases, or death. Respirators prevent entry of harmful substances into the body during breathing and make sure that the user has an adequate supply of clean air. Some respirators also provide a separate supply of air so work can be done where there is inadequate oxygen.

Prevention of atmospheric contamination at the worksite generally should be accomplished as far as feasible by engineering control measures (for example, enclosure or confinement of the contaminant-producing operation, general and local ventilation) or by work-practice controls such as substitution of less toxic materials. Historically, the industrial hygiene profession has sought to control hazardous air contamination through engineering or work-practice means. However, when effective engineering controls are not feasible, or while those controls are being installed, appropriate respirators must be used. The user should be aware that respirators have their limitations and are not a substitute for effective engineering controls. Where respirators are necessary for health protection, specific procedures are necessary to overcome potential deficiencies and to assure effectiveness.

CRAY Y-MP FLUORINERT SAFETY

Although the CRAY Y-MP Fluorinert tank is vented directly into the computer bay, there should be no danger of having perfluoroisobutylenes (PFIB) formed, because the fluorinert has no contact with electricity. Much less fluorinert escapes into the room for the CRAY Y-MP than from removing boards from the CRAY-2, which are wet with fluorinert.

Fluorinert in the CRAY Y-MP (called FX-74) is contained inside the cold plates. Calculations have been performed to estimate the quantity of fluorinert lost to the atmosphere due to the venting of the CRAY Y-MP, and it was found to be minimal. The 3M Company has set a maximum standard of 1000 ppm of fluorinert in the air. Therefore, monitoring of FX-74 is not needed, because it would be extremely difficult to get greater than 1000 ppm of fluorinert in the air.

FX-74 is a refined existing product of the Fluorinert family. It has a low toxicity, and the inhalation toxicity of FX-74 is considered nonexistent. It has a high affinity for oxygen, and is used as a blood substitute. Sutdies have found that it would take extremely high heating to get enough evaporation into the air for oxygen depletion.