Instruction Descriptions

(CRAY T90™ Series)

HTM-119-A

Cray Research Proprietary

Cray Research, Inc.

Record of Revision

REVISION	DESCRIPTION
	March 1995. Original printing.
A	September 1995. Revision A incorporates various technical corrections.

Any shipment to a country outside of the United States requires a letter of assurance from Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be directed to:

CRAY RESEARCH, INC.
Logistics
6251 South Prairie View Road
Chippewa Falls, WI 54729

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
Service Publications and Training
890 Industrial Blvd.
Chippewa Falls, WI 54729

INSTRUCTION DESCRIPTIONS

Notational Conventions

This document describes all instructions in the CRAY T90 series CPU instruction set. The instruction descriptions use acronyms and abbreviations that are defined in the *Instruction Set Overview.* The following information is included with each instruction description:

- Special cases
- Hold-issue conditions
- Execution time
- Description

In some instructions, register designators are prefixed by the following letters that have special meaning to the assembler. The letters and their meanings are listed as follows:

Letter	Description
F	Floating-point operation
H	Half-precision floating-point operation
I	Reciprocal iteration
P	Population count
Q	Parity count
R	Rounded floating-point operation
Z	Leading-zero count

The following list defines some of the notations used in the instruction set:

+	Arithmetic sum
−	Arithmetic difference
*	Arithmetic product
/	Reciprocal approximation
#	Use one's complement
>	Shift value or form mask from left to right
<	Shift value or form mask from right to left
&	Logical product (AND)
!	Logical sum (inclusive OR)
\	Logical difference (exclusive OR)

An expression (*exp*) occupies the *jk*, *nm*, or *pnm* field. The *h*, *i*, *j*, and *k* designators indicate the field of the machine instruction into which the register designator constant or symbol value is placed.

Instruction 000000

Machine Instruction	CAL Syntax	Description
000000	ERR	Error exit.

Special Cases

None.

Hold-issue Conditions

Any previously issued instruction not completed.

Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction fetch operation in progress.

Execution Time

Instruction issue: 1 CP.

This instruction causes an exchange and a fetch operation. The time of completion is undetermined.

Description

Instruction 000 initiates an exchange sequence. All instructions issued before the 000 instruction are completed. The instruction buffers and data cache are voided.

When all previously issued instructions have completed, an exchange sequence occurs to the exchange package designated by the exchange address (XA) register. The program address stored in the exchange package is advanced one count from the address of the 000 instruction.

If the Enable Flag on Error Exit (FEX) bit (in the interrupt modes field of the exchange package) is set, instruction 000 sets to 1 the Error Exit (EEX) bit (in the interrupt flags field of the exchange package). If the FEX bit is not set, it clears the EEX bit to 0.

If an exchange (interrupt) request occurs at the same time as a 000 instruction, the exchange request takes precedence and causes an exchange to the exchange address (XA). The EEX flag is not set, and the program address stored in the exchange package points to the 000 instruction.

The 000 instruction is treated as an error condition; it is not generally used in program code. The instruction stops execution of an incorrectly coded program that branches to an unused area of memory (if memory was backgrounded with zeroes) or into a data area (if data is positive integers, right justified ASCII, or floating-point zeroes).

Instructions 0010 through 0012

Machine Instruction	CAL Syntax		Description
001000	PASS		Pass (no operation).
0010*jk* (*jk*≠0)M	CA,A*j*	A*k*	Set channel (A*j*) CA register (A*k*) and activate channel.
0011*jk*M	CL,A*j*	A*k*	Set channel (A*j*) CL register (A*k*).
0012*j*0M	CI,A*j*		Clear interrupt flag and error flag for channel (A*j*). Clear Device Master Clear (output channels only). Enable channel interrupt.
0012*j*1M	MC,A*j*		Clear interrupt flag and error flags for channel (A*j*). Set Device Master Clear (output channels only). Clear Ready Held (input channels only). Enable channel interrupt.
0012*j*2M	DI,A*j*		Disable channel A*j* interrupt.
0012*j*3M	EI,A*j*		Enable channel A*j* interrupt.

M Monitor mode only.
Description applies to LOSP channels. Refer to the text for descriptions of VHISP channel instructions.

Special Cases

Except for instruction 001000, instructions 0010 through 0012 are privileged to monitor mode.

If the CPU is in normal user mode, instructions 0010 through 0012 execute as no-ops.

If the CPU is in interrupt-on-monitor-instruction (IMI) mode, instructions 0010 through 0012 (except 001000) execute as no-ops, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

The operation of instructions 0010 through 0012 (except 001000) is modified by CPU configuration codes 050_8 through 055_8.

For instructions 0010*jk* and 0011*jk* in monitor mode:

- If $j = 0$, the instruction performs no operation.
- If $j \neq 0$ and $k = 0$, the CA or CL register is set to 1.

For instructions 0012*j*0 and 0012*j*1, if $j = 0$, the instruction performs no operation.

Hold-issue Conditions

Instructions 0010*jk* and 0011*jk* :

- Register A*j* or A*k* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 0012*j*0, 0012*j*1, 0012*j*2, and 0012*j*3:

- Register A*j* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Shared path ready: CPs undetermined.

Description

Except for instruction 001000, instructions 0010 through 0012 are privileged to monitor mode and provide operations useful to the operating system. Instruction 001000 (or any instruction of the form 00100*k*) performs no operation.

Instructions 0010, 0011, and 0012 control operation of the LOSP and VHISP I/O channels. Register A*j* designates the I/O channel. (When the *j* designator is 0, the instructions execute as no-op instructions.) These instructions operate somewhat differently, depending on whether a LOSP or VHISP channel is selected.

Instructions 0010*jk* and 0011*jk* initiate a channel transfer. For LOSP channels, instruction 0011*jk* first transfers the contents of register A*k* to the channel limit (CL) register of channel (A*j*). Instruction 0010*jk* then transfers the contents of register A*k* to the channel address (CA) register of channel (A*j*) and starts the transfer.

For VHISP channels, instruction 0010*jk* must be used twice, followed by instruction 0011*jk*. The first 0010*jk* instruction sets the starting block address in the solid-state storage device (SSD). The second 0010*jk* instruction sets the channel address (CA) in the mainframe. The 0011*jk* instruction sets the block length (BL) and starts the transfer.

Instructions 0012*j*0 and 0012*j*1 clear an I/O channel before or after a transfer. For LOSP channels, both instructions operate almost identically. They clear the channel interrupt, clear the Error and Done flags, and enable channel interrupts. For LOSP input channels, instruction 0012*j*1 clears the Ready Held flag. For LOSP output channels, instruction 0012*j*0 clears the Device Master Clear signal; instruction 0012*j*1 sets Device Master Clea**r**.

For VHISP channels, instruction 0012*j*0 clears the channel interrupt, clears the channel sequence, clears the Error and Done flags, and enables channel interrupts. Instruction 0012*j*1 is not used.

Instructions 0012*j*2 and 0012*j*3 disable and enable, respectively, interrupts for channel (A*j*).

Instruction 0013

Machine Instruction	CAL Syntax		Description
0013*j*0[M]	XA	A*j*	Transmit (A*j*) to exchange address.
0013*j*1[NM]	A*j*	XA	Transmit exchange address to A*j*.
001302[M]	EMI		Enable monitor interrupt mode (set EIM to 1).
001303[M]	DMI		Disable monitor interrupt mode (clear EIM to 0).

N New instruction (not available on CRAY C90 series systems).
M Monitor mode only.

Special Cases

Instruction 0013 is privileged to monitor mode.

If the CPU is in normal user mode, instruction 0013 executes as a no-op.

If the CPU is in IMI mode, instruction 0013 executes as a no-op, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

Execution of instruction 0013 is affected by the CPU set-up configuration. Refer to PRN-0957, *Triton Maintenance System,* for more information.

For instruction 0013*j*0 in monitor mode: If *j* = 0, the exchange address (XA) register is cleared.

Hold-issue Conditions

Instructions 0013*j*0 and 0013*j*1:

- Register A*j* reserved (including A0).
- Shared path reserved.

Execution Time

Instruction 0013*j*0:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 0013*j*1:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Register A*j* ready: 6 CPs.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 001302:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Interrupts enabled: CPs undetermined.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 001303:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Interrupts disabled: CPs undetermined.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Description

Instruction 0013 is privileged to monitor mode and provides operations useful to the operating system.

Instruction 0013*j*

Instruction 0013*j*0 transmits bits 5 through 20 of register A*j* to the exchange address (XA) register.

Instruction 0013*j*1

Instruction 0013*j*1 transmits the contents of the exchange address (XA) register to bits 5 through 20 of register A*j*. The remaining bits of register A*j* are cleared.

Instructions 001302 and 001303

Instructions 001302 and 001303 enable and disable, respectively, monitor interrupt mode. Monitor interrupt mode is controlled by the EIM bit.

If the CPU is in monitor mode and EIM = 0, only the following interrupts are enabled: PRE, EEX, and NEX. The following interrupts are held until EIM = 1: RPE, MEU, FPE, ORE, BPI, MEC, MCU, RTI, and AMI.

If the CPU is in monitor mode and EIM = 1, all interrupts are enabled except DL and MII. (ICP and PCI).

If the CPU is in user mode, EIM is always 1. All interrupts are enabled.

The EIM bit is set to 1 upon exchange to user mode. It cannot be changed while the CPU is in user mode.

The EIM bit is cleared to 0 upon exchange to monitor mode. Instructions 0013102 and 001303 can be used to toggle the EIM bit when the CPU is in monitor mode.

Instruction 0014

Machine Instruction	CAL Syntax		Description
0014*j*0[M]	RT	S*j*	Transmit (S*j*) to real-time clock.
0014*j*1[M]	SIPI	A*j*	Send inter-CPU interrupt to CPU (A*j*).
001401[M]	SIPI[S]		Send inter-CPU interrupt to CPU 0.
001402[M]	CIPI		Clear inter-CPU interrupt.
0014*j*3[M]	CLN	A*j*	Transmit (A*j*) to cluster number register.
0014*j*4[M]	PCI	S*j*	Transmit (S*j*) to programmable clock.
001405[M]	CCI		Clear programmable clock interrupt (clear PCI to 0).
001406[M]	ECI		Enable programmable clock interrupt (set IPC to 1).
001407[M]	DCI		Disable programmable clock interrupt (clear IPC to 0).

M Monitor mode only.
S Special CAL syntax.

Special Cases

Instruction 0014 is privileged to monitor mode.

If the CPU is in normal user mode, instruction 0014 executes as a no-op.

If the CPU is in IMI mode, instruction 0014 executes as a no-op, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

For instruction 0014*j*1 in monitor mode:

- If *j* = 0, the interrupt is sent to CPU 0.
- If a CPU attempts to interrupt itself, the instruction executes as a no-op.
- If shared access is disabled (CPU configuration code 051_8), the instruction executes as a no-op. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

For instruction 0014*j*3 in monitor mode: If *j* = 0 or if (A*j*) = 0, the cluster number is cleared to 0. Cluster 0 is a *dummy* cluster without registers. Refer to the instruction description for more information.

For instruction 0014*j*4 in monitor mode: If *j* = 0, the programmable clock is cleared.

Hold-issue Conditions

Instructions 0014*j*0 and 0014*j*4:

- Register S*j* reserved (except S0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 0014*j*1 and 0014*j*3:

- Register A*j* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 001402, 0014*j*5, 0014*j*6, and 0014*j*7:

- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction 0014*j*0:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Real-time clock valid: CPs undetermined (normal operation).
- Real-time clock valid: CPs undetermined (shared access disabled – CPU configuration code 051_8.) Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information.

Instructions 0014*j*1, 001402, 0014*j*3, and 0014*j*5:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.

Instruction 0014*j*4:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.

Instruction 0014*j*6:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Programmable clock interrupt enabled: CPs undetermined.

Instruction 0014*j*7:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.
- Programmable clock interrupt disabled: CPs undetermined.

Description

Instruction 0014 is privileged to monitor mode and provides operations useful to the operating system.

Instructions 0014*j*

Instruction 0014*j*0 transmits the contents of register S*j* to the real-time clock.

Instructions 0014*j*1

Instruction 0014*j*1 sends an inter-CPU interrupt request to the CPU designated by register A*j*. If the receiving CPU has its IIP bit set and its EIM bit set, its ICP flag sets and an exchange sequence occurs. The program that begins executing as a result of the exchange sequence must be in monitor mode and must execute instruction 001402 to clear the ICP flag. If instruction 001402 is not executed, the ICP flag initiates another exchange sequence whenever the EIM bit sets (either by instruction 001302 or by exchange to user mode).

If the receiving CPU does not have its IIP bit set or does not have its EIM bit set, it ignores inter-CPU interrupt requests.

Instruction 001402

Instruction 001402 clears an inter-CPU interrupt request. It must be used after each inter-CPU interrupt.

Instruction 0014*j*3

Instruction 0014*j*3 sets the cluster number to the contents of register A*j*. Valid cluster numbers are 0 through n, where n is the number of the clusters in the system.

The number of clusters is dependent on the system type and system configuration. CRAY T94 and CRAY T916 systems contain 18 clusters and can be configured with 9 or 18 clusters. CRAY T932 systems contain 36 clusters and can be configured with 9, 18, or 36 clusters.

Cluster 0 is a *dummy* cluster without registers. An instruction that attempts to read data from an SB, ST, or SM register of cluster 0 returns all 0's. An instruction that writes to an SB, ST, or SM register of cluster 0 or that tests a semaphore executes as a no-op.

Clusters 1 through n are valid clusters.

A cluster number larger than the number of valid clusters produces undefined results.

Instruction 0014*j*4

Instruction 0014*j*4 transmits bits 0 through 31 of register S*j* to the interrupt interval register of the programmable clock. If the IPC bit is set, an interrupt is generated once every n CPs, where n is the value loaded into the interrupt interval register.

Every n CPs, if the CPU has its IPC bit set and its EIM bit set, its PCI flag sets and an exchange sequence occurs. The program that begins executing as a result of the exchange sequence must be in monitor mode and must execute instruction 001405 to clear the PCI flag. If instruction 001405 is not executed, the PCI flag initiates another exchange sequence whenever the EIM bit sets (either by instruction 001302 or by exchange to user mode).

If the CPU does not have its IPC bit set or does not have its EIM bit set, it ignores programmable clock interrupt requests.

Instruction 001405

Instruction 001405 clears a programmable clock interrupt request. It must be used after each programmable clock interrupt.

Instructions 001406 and 001407

Instructions 001406 and 001407 enable and disable, respectively, programmable clock interrupts. Instruction 001406 sets the IPC bit to 1. Instruction 001407 clears the IPC bit to 0.

The IPC bit must be set and the EIM bit must be set to enable programmable clock interrupts.

Instruction 0015

Machine Instruction	CAL Syntax	Description
001500[M]	—[X]	Clear all performance monitor counters.
001501[NTM]	—[X]	Clear performance monitor pointer.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
M Monitor mode only.
X Not supported by CAL.

Special Cases

Instruction 0015 is privileged to monitor mode.

If the CPU is in normal user mode, instruction 0015 executes as a no-op.

If the CPU is in IMI mode, instruction 0015 executes as a no-op, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

Instruction 001501 is privileged to Triton mode. If this instruction is executed in C90 mode, the result is undefined.

Except in maintenance mode, instruction 0015 should not be issued until the performance monitor is not busy (PMBY status register 0 bit 47 clear). If this instruction is issued with PMBY set, the result is undefined.

Execution of instruction 0015 is affected by the CPU setup configuration. Refer to PRN-0957, *Triton Maintenance System*, for more information.

Hold-issue Conditions

Shared path reserved.

Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Shared path ready: CPs undetermined.

Performance monitor ready: 36 to 68 CPs.

Description

Instruction 001500 clears all performance monitor counters and the performance monitor pointer. This instruction should not be issued until the PMBY flag (status register 0 bit 47) is clear.

Instruction 001501 clears the performance monitor pointer. Except in maintenance mode, this instruction should not be issued until the PMBY flag (status register 0 bit 47) is clear. In maintenance mode, issuing instruction 073*i*25 causes the PMBY bit to set and remain set. The only way to clear the PMBY flag in this case is with instruction 001501.

Instruction 0016

Machine Instruction	CAL Syntax	Description
001600^{M}	ESI	Enable system I/O interrupts (set SIE to 1).
001640^{NM}	BCD	Broadcast cluster detach.

N New instruction (not available on CRAY C90 series systems).
M Monitor mode only.

Special Cases

Instruction 0016 is privileged to monitor mode.

If the CPU is in normal user mode, instruction 0016 executes as a no-op.

If the CPU is in IMI mode, instruction 0016 executes as a no-op, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction.

In monitor mode, if shared access is disabled (CPU configuration code 051_8), instruction 0016 executes as a no-op. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Hold-issue Conditions

Shared path reserved.

Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Shared path ready: CPs undetermined.

Description

Instruction 0016 is privileged to monitor mode and provides operations useful to the operating system.

Instructions 001600

Instruction 001600 enables I/O interrupts to occur.

When a CPU receives an I/O interrupt, subsequent I/O interrupts are disabled for the I/O group. The CPU that receives an I/O interrupt should, after servicing the interrupt, execute instruction 001600 to re-enable I/O interrupts for the I/O group.

NOTE: If the receiving CPU executes instruction 001600 before clearing the interrupt request, the request causes another interrupt.

Instructions 001640

Instruction 001640 releases all cluster assignments previously established by user-mode programs running in the particular CPU. This instruction is normally used only in Triton mode; in C90 mode, cluster assignments are automatically released on every exchange.

When a user-mode program exchanges in, the cluster number (CLN) field in its exchange package specifies the shared register cluster to which it is assigned. If the user-mode program exchanges out to a monitor-mode program and a new user is to be brought into the CPU, the monitor program will execute instruction 001640 to release the old cluster assignment.

If a user-mode program exchanges out to another user-mode program, the cluster assignment is not released. The new cluster assignment takes precedence, but the old assignment is still active. Thus, when a monitor-mode program starts, several cluster assignments can be active. Instruction 001640 releases all these assignments.

Shared module configuration functions can enable and disable the capability of having multiple cluster assignments active simultaneously. Refer to Section 4.6.1.1 of PRN-0957 for more information.

Instruction 0017

Machine Instruction	CAL Syntax	Description
0017*jk*M	BP,*k* A*j*	Transmit (A*j*) to breakpoint address *k* (*k* = 0 or 1).

M Monitor mode only.

Special Cases

Instruction 0017 is privileged to monitor mode.

If the CPU is in normal user mode, instruction 0017 executes as a no-op.

If the CPU is in IMI mode, instruction 0017 executes as a no-op, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

If $k \neq 0$ or 1, the result is undefined.

If $j = 0$, the selected breakpoint address is set to 0.

Execution of instruction 0017 is affected by the CPU setup configuration. Refer to PRN-0957, *Triton Maintenance System*, for more information.

Hold-issue Conditions

Register A*j* reserved (except A0).

Shared path reserved.

Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Shared path ready: CPs undetermined.

Description

Instruction 0017*jk* transmits the contents of register A*j* to breakpoint address *k*, where *k* is 0 or 1. If *k* = 0, the breakpoint base address is loaded. If *k* = 1, the breakpoint limit address is loaded.

A breakpoint interrupt occurs if a common memory write reference is made to a physical address within the breakpoint range (that is, to a physical address equal to or greater than the breakpoint base address and less than the breakpoint limit address). The breakpoint compare is done on bits 0 through 34 of the address; the higher-order bits are ignored. If the limit address is less than or equal to the base address, breakpoint interrupts do not occur.

Because a breakpoint interrupt occurs only when a write reference is made to a physical address less than the breakpoint limit address, it is impossible to have a breakpoint interrupt occur on the highest physical memory address.

Breakpoint interrupts do not occur on read references.

A breakpoint interrupt can occur only if the IBP bit (in the interrupt modes field of the exchange package) is set and the EIM bit is set. If the IBP bit is clear, breakpoint interrupt requests are ignored. If the IBP bit is set and the EIM bit is clear, a breakpoint interrupt request is held until the EIM bit sets.

NOTE: Breakpoint addresses are not part of the exchange package. When an exchange sequence occurs, the hardware does not automatically load or save the breakpoint addresses.

Instruction 0020

Machine Instruction	CAL Syntax		Description
00200*k*	VL	A*k*	Transmit (A*k*) to vector length register.
002000	VL	1[S]	Transmit 1 to vector length register.

S Special CAL syntax.

Special Cases

If the CPU is in IMI mode, instruction 0020 executes normally. Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

If $k = 0$, the vector length is set to 1.

If $k \neq 0$ and (A*k*) = 0, the vector length is set to 128.

Hold-issue Conditions

Register A*k* reserved (except A0).

Shared path reserved.

Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Register VL ready: 1 CP.

Shared path ready: CPs undetermined.

Description

Instruction 00200*k* transmits the contents of register A*k* to the vector length (VL) register. The vector length can be between 1 and 128 (200_8), inclusive.

Although the VL register contains 8 bits (0 through 7), only the lower 7 bits (0 through 6) of register A*k* are transmitted to the VL register. Bit 7 of the VL register is automatically set if bits 0 through 6 are 0's. Bit 7 is automatically cleared if any of bits 0 through 6 are 1's.

Instructions 0021 through 0027

Machine Instruction	CAL Syntax	Description
002100	EFI	Enable interrupt on floating-point error (set IFP to 1).
002200	DFI	Disable interrupt on floating-point error (clear IFP to 0).
002210	CBL	Clear bit matrix loaded bit (clear BML to 0).
002300	ERI	Enable interrupt on operand range error (set IOR to 1).
002301	EBP	Enable interrupt on breakpoint (set IBP to 1).
002400	DRI	Disable interrupt on operand range error (clear IOR to 0).
002401	DBP	Disable interrupt on breakpoint (clear IBP to 0).
002500	DBM	Disable bidirectional memory transfers (clear BDM to 0).
002501[N]	ESC	Enable scalar cache (set SCE to 1).
002600	EBM	Enable bidirectional memory transfers (set BDM to 1).
002601[N]	DSC	Disable and invalidate scalar cache (clear SCE to 0).
002700	CMR	Complete memory references.
002704	CPA	Complete port reads and writes (ports A, B, and C).
002705	CPR	Complete port reads (ports A and B).
002706	CPW	Complete port writes (port C).

N New instruction (not available on CRAY C90 series systems).

Special Cases

If the CPU is in IMI mode, instructions 0021 through 0026 execute as no-ops, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

The execution of instructions 0021 through 0027 is affected by the CPU setup configuration. Refer to PRN-0957, *Triton Maintenance System*, for more information.

Hold-issue Conditions

Instructions 002100, 002200, 002300 through 002500, and 002600:

- Shared path reserved.
- Status register busy.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 002210:

- Shared path reserved.
- Status register busy.

Instructions 002501 and 002601:

- Shared path reserved.
- Any memory activity in progress.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 002700:

- Shared path reserved.
- Port A, B, or C busy.
- Scalar instruction issued in previous CPs undetermined.
- Block instruction issued in previous CPs undetermined.

Instruction 002704:

- Shared path reserved.
- Port A, B, or C busy.

Instruction 002705:

- Shared path reserved.
- Port A or B busy.

Instruction 002706:

- Shared path reserved.
- Port C busy.

Execution Time

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.

Description

Instructions 002100 and 002200 set and clear the interrupt-on-floating-point (IFP) error bit in the interrupt modes register. Issuing either of these instructions also clears the floating-point error flag.

Instruction 002210 clears the bit matrix loaded (BML) bit in the status field of the exchange package. This bit is set by instructions 1740*j*4 and 1740*j*5.

Instructions 002300 and 002400 set and clear the interrupt-on-operand range (IOR) error bit in the interrupt modes register.

Instructions 002301 and 002401 set and clear the interrupt-on-breakpoint (IBP) bit in the interrupt modes register. When instruction 002401 is issued, the IBP mode becomes effective after all previously issued memory or floating-point operations can no longer cause interrupts.

Instructions 002500 and 002600 clear and set the bidirectional memory (BDM) bit in the mode register. When the BDM bit is set, block read and write operations can operate concurrently. When the BDM bit is clear, block read and write operations cannot operate concurrently; however two block read operations can operate concurrently if one uses port A and the other uses port B.

NOTE: The operating system can override the BDM bit by setting the bidirectional disable (BDD) bit, which is also in the mode register. If the BDD bit is set, bidirectional memory transfers are disabled regardless of the state of the BDM bit.

Instructions 002501 and 002601 set and clear the scalar cache enable (SCE) bit in the mode register. Instruction 002601 also invalidates cache data.

Instruction 002700 ensures completion of all memory references within the CPU issuing the instruction. Instruction 002700 does not issue until all memory references before this instruction will be able to complete in a fixed number of CPs. For example, a CPU is ensured of receiving updated data when it issues a data load instruction after a 002700 instruction. The 002700 instruction, used in conjunction with semaphore instructions, synchronizes memory references between processors.

Instructions 002704 through 002706 can be used to ensure sequential memory referencing within a CPU. These instructions do not issue until previous memory references are at a stage of execution in which their completion is guaranteed to occur before any memory references issued after these instructions. The 002704 instruction ensures that all reads and writes are at this stage. The 002705 instruction ensures that only reads occur at this stage. The 002706 ensures that only writes occur at this stage.

Instruction 0030

Machine Instruction	CAL Syntax		Description
0030*j*0	VM0	S*j*	Transmit (S*j*) to VM0.
003000	VM0	0^{S}	Clear VM0.
0030*j*1	VM1	S*j*	Transmit (S*j*) to VM1.
003001	VM1	0^{S}	Clear VM1.
$0030j2^{NT}$	VM0	A*j*	Transmit (A*j*) to VM0.
$0030j3^{NT}$	VM1	A*j*	Transmit (A*j*) to VM1.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only
S Special CAL syntax

Special Cases

Instructions 0030*j*2 and 0030*j*3 are privileged to Triton mode. If these instructions are executed in C90 mode, the results are undefined.

For instructions 0030*j*0 and 0030*j*2: If *j* = 0, register VM0 is cleared.

For instructions 0030*j*1 and 0030*j*3: If *j* = 0, register VM1 is cleared.

Hold-issue Conditions

Instructions 0030*j*0 and 0030*j*1:

- Register S*j* reserved (except S0).
- VM busy. (For instruction 175, VM busy VL/2 + 8 CPs. For instructions 070*ij*1, 146*ijk*, 147*ijk*, 005400 153*ij*0, or 005400 153*i*j1, VM busy 3 CPs.)

Instructions 0030*j*2 and 0030*j*3:

- Register A*j* reserved (except A0).
- VM busy. (For instruction 175, VM busy VL/2 + 8 CPs. For instructions 070*ij*1, 146*ijk*, 147*ijk*, 005400 153*ij*0, or 005400 153*i*j1, VM busy 3 CPs.)

Execution Time

Instructions issue: 1 CP.

Description

Instruction 0030*j*0 transmits the contents of register S*j* to the lower vector mask register (VM0). Bit 63 of register S*j* corresponds to element 0 of the vector mask; bit 0 corresponds to element 63.

Instruction 0030*j*1 transmits the contents of register S*j* to the upper vector mask register (VM1). Bit 63 of register S*j* corresponds to element 64 of the vector mask; bit 0 corresponds to element 127.

Instruction 0030*j*2 transmits the contents of register A*j* to the lower vector mask register (VM0). Bit 63 of register A*j* corresponds to element 0 of the vector mask; bit 0 corresponds to element 63.

Instruction 0030*j*3 transmits the contents of register A*j* to the upper vector mask register (VM1). Bit 63 of register A*j* corresponds to element 64 of the vector mask; bit 0 corresponds to element 127.

Instruction 0034 through 0037

Machine Instruction	CAL Syntax		Description
0034*jk* (*j*2 = 0)	SM*jk*	1,TS	Test and set semaphore *jk* (*jk* = 0 – 37_8).
0034*jk* (*j*2 = 1)	SM,A*k*	1,TS	Test and set semaphore (A*k*).
0036*jk* (*j*2 = 0)	SM*jk*	0	Clear semaphore *jk* (*jk* = 0 – 37_8).
0036*jk* (*j*2 = 1)	SM,A*k*	0	Clear semaphore (A*k*).
0037*jk* (*j*2 = 0)	SM*jk*	1	Set semaphore *jk* (*jk* = 0 – 37_8).
0037*jk* (*j*2 = 1)	SM,A*k*	1	Set semaphore (A*k*).

Special Cases

Instructions 0034 through 0037 execute as no-ops if the cluster number (CLN) is 0.

For instructions 0034 through 0037 (*j*2 = 1): If *k* = 0, semaphore 1 is addressed.

If the CPU is in IMI mode, instructions 0034 through 0037 execute normally. (Normally in IMI mode, the cluster number should be 0 so that these instructions are no-ops.) Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

If shared access is disabled (CPU configuration code 051_8), instructions 0034 through 0037 execute as no-ops. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Hold-issue Conditions

Instructions 0034, 0036, and 0037 (*j*2 = 0):

- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 0034, 0036, and 0037 (*j*2 = 1):

- Shared path reserved.
- Register A*k* reserved (except A0).
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instructions issue: 1 CP.

Shared path ready: CPs undetermined.

Description

Instructions 0034 through 0037 operate on the semaphores (SMs). If bit 2 of the *j* field is 0, bits 1 and 0 of the *j* field are combined with all 3 bits of the *k* field to select one of the lower 32 semaphores (0 through 31). If bit 2 of the *j* field is 1, the *k* field designates an A register. Bits 0 through 5 of the designated A register select the semaphore; the remaining bits of the designated A register are not used.

Instruction 0034*jk* tests and sets the semaphore designated by the *jk* field or by register A*k*. If the designated semaphore is clear, the instruction issues immediately. If it is set, the instruction holds issue until another CPU in the same cluster clears the semaphore. When the instruction issues, the semaphore is set.

If all CPUs in a cluster are holding issue on instruction 0034, the Deadlock (DL) flag is set and an exchange occurs in all CPUs in the cluster that have the interrupt on deadlock (IDL) bit set and are in user mode.

If an interrupt occurs while instruction 0034 is holding issue, the waiting on semaphore (WS) bit in the status field of the exchange package sets and an exchange is initiated with the P register pointing to the 0034 instruction.

Instruction 0036 clears the semaphore designated by *jk* or (A*k*).

Instruction 0037 sets the semaphore designated by *jk* or (A*k*).

Instruction 004

Machine Instruction	CAL Syntax	Description
00400*k*[V]	EX*k*	Exit *k*.
004000	EX[S]	Exit 0 (normal exit).

V New version of CRAY C90 series instruction.
S Special CAL syntax.

Special Cases

If *k* = 5, 6, or 7, the exit is to the exchange address instead of an exit address.

Hold-issue Conditions

Any previously issued instruction not completed.

Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction fetch operation in progress.

Execution Time

Instruction issue: 1 CP.

Following instruction issue, an additional CPs undetermined are required; this time allows for an exchange sequence (CPs undetermined) and fetch operation (CPs undetermined). Memory conflicts during the exchange sequence or fetch operation cause additional delays.

Description

Instruction 004 initiates an exit sequence. The *k* field designates one of five exit addresses (0 through 4). The *j* field is not used. All instructions issued before the 004 instruction are completed. The instruction buffers and data cache are voided.

When all previously issued instructions have completed, an exchange sequence occurs to the exchange package designated by the contents of exit address register *k* (EA*k*). The program address stored in the exchange package is advanced one count from the address of the 004 instruction.

If the FNX bit (in the interrupt modes field of the exchange package) is set, instruction 004 sets the NEX flag (in the interrupt flags field of the exchange package) to 1. Otherwise it clears the NEX flag to 0. (This is true even if k = 5, 6, or 7.)

If an exchange (interrupt) request is active when a 004 instruction issues, the exchange request takes precedence and causes an exchange to the exchange address (XA). The NEX flag is not set, and the program address stored in the exchange package points to the 004 instruction.

Instructions 023*ij*6, 023*ij*7, 027*ij*2, and 027*ij*3 read from and write to the exit address registers.

Instruction 004 issues a monitor request from a user program or transfers control from a monitor program to another program.

Instruction 005

Machine Instruction	CAL Syntax		Description
0050*jk*	J	B*jk*	Jump to B*jk*.
0051*jk*O	JINV	B*jk*	Jump to B*jk* (invalidate instruction buffers)

O Maintenance mode only.

Special Cases

Instruction 0051 is privileged to maintenance mode. If this instruction is executed in nonmaintenance mode, the result is equivalent to instruction 0050 (the instruction buffers are not invalidated).

There are several restrictions on using instruction 0051 in maintenance mode. For more information, refer to the following description.

Hold-issue Conditions

Instruction 034 or 035 in progress.

Execution Time

Instruction issue: 1 CP.

Jump completion: 12 CPs. Additional time is required if the destination address is not in the instruction buffer.

Description

Instruction 0050 sets the P register to the address contained in the B register specified by *jk*, causing the program to continue at that address. (The address specified by register B*jk* is a parcel address, not a word address.) In C90 mode, all 32 bits of the specified B register are used. In Triton mode, only the lower 32 bits are used; the upper 32 bits are ignored. This instruction is used to return from a subroutine.

In maintenance mode, instruction 0051 operates similarly to instruction 0050 except that it invalidates the instruction buffers, forcing an out-of-buffer jump to occur. In nonmaintenance mode, instruction 0051 operates identically to instruction 0050.

In maintenance mode, there are restrictions on the 32-word block of code containing a 0051 instruction that gets loaded into a single instruction buffer. If these restrictions are not met, the results are undefined.

The program flow must enter the code block at word 0, either through straight-line code or from a jump to any parcel in word 0.

Words 0 though 3 of the code block must not contain jump instructions.

The 0051 instruction must be located between words 4 and 20_8 (inclusive) of the code block.

Instruction 006

Machine Instruction	CAL Syntax		Description
006000*nm*	J	*exp*	Jump to *exp*.
006100*nm*[NT]	IJ	*exp*	Jump to address in *exp*.
0064*jknm* (*j*2=0)	JTS*jk*	*exp*	Jump to *exp* if SM*jk* = 1; else set SM*jk*.
0064*jknm* (*j*2=1)	JTS,A*k*	*exp*	Jump to *exp* if SM(A*k*) = 1; else set SM(A*k*).

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.

Special Cases

Instruction 0061 is privileged to Triton mode. If this instruction is executed in C90 mode, the result is undefined.

Instruction 0064 executes a no-op if the cluster number (CLN) is 0. (The jump is not taken.)

For instruction 0064(*j*2 = 1): If *k* = 0, semaphore 1 is addressed.

For instruction 0064: If the CPU is in IMI mode, the cluster number must be 0 so that the instruction is a no-op. (If the cluster number is not 0, the result is undefined.) The MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the second parcel of the instruction (*m* field). Refer to the *Instruction Set Overview* for more information.

For instruction 0064: If shared access is disabled (CPU configuration code 051_8), this instruction executes as a no-op. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Hold-issue Conditions

Instructions 0060 and 0061:

If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instruction 0064 (*j*2 = 0):

- Always hold issue 5 CPs.
- Shared path reserved.
- If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 0064 (*j*2 = 1):

- Always hold issue 5 CPs.
- Shared path reserved.
- Register A*k* reserved.
- If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction 0060:

- Instruction issue: 3 CPs.
- Jump completion: 7 CPs. Additional time is required if the destination address is not in the instruction buffer.

Description

Instruction 0060

Instruction 006000*nm* is an unconditional jump. It sets the P register to the 32-bit address specified by *nm*, causing program execution to continue at that address. The address specified by *nm* is a parcel address, not a word address.

Instruction 0061

Instruction 006100*nm* is an indirect unconditional jump. The 32-bit *nm* field specifies a common memory address. The lower 32 bits of the contents of that address are transferred to the P register, causing program execution to continue at that point. The instruction buffers are invalidated. The address specified by *nm* is a word address, but the lower 32 bits of that word specify a parcel address where program execution continues.

Instruction 0064

Instruction 0064*jknm* is a conditional jump instruction that tests a semaphore (SM) to determine whether or not to take the jump. If bit 2 of the *j* field is 0, bits 1 and 0 are combined with all 3 bits of the *k* field to select one of the lower 32 semaphores (0 through 31). If bit 2 of the *j* field is 1, the *k* field designates an A register. Bits 0 through 5 of the designated A register select the semaphore; the remaining bits of the designated A register are not used.

The designated semaphore is tested to determine whether or not to take the jump. If the semaphore is clear, the semaphore is set and program execution continues with the next sequential instruction. If the semaphore is set, the P register is set to the 32-bit parcel address (not word address) specified by *nm*, and program execution continues at that point.

Instruction 007

Machine Instruction	CAL Syntax		Description
007000*nm*	R	*exp*	Return jump to *exp*; set B00 to (P)+3.
007100*nm*[NT]	IR	*exp*	Return jump to address in *exp*; set B00 to (P)+3.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only

Special Cases

Instruction 0071 is privileged to Triton mode. If this instruction is executed in C90 mode, the result is undefined.

Hold-issue Conditions

Instruction 034 or 035 in progress.

If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Execution Time

Instruction 0070:

- Instruction issue: 3 CPs.
- Jump completion: 7 CPs. Additional time is required if the destination address is not in the instruction buffer.

Description

Instructions 0070 and 0071 are return jump instructions. They operate similarly to the normal jump instructions (0060 and 0061), except that the address of the next sequential instruction is stored in register B00 before the jump occurs. The intended function of these instructions is to call subroutines. When the subroutine completes, instruction 005000 can be used to return to the calling program.

Instruction 007000*nm* is a return jump. It first stores the address of the next sequential instruction in register B00. The instruction then sets the P register to the 32-bit address specified by *nm*, causing program execution to continue at that address. The address specified by *nm* is a parcel address, not a word address.

Instruction 007100*nm* is an indirect return jump. It first stores the address of the next sequential instruction in register B00. The instruction then uses the 32-bit *nm* field to specify a common memory address. The lower 32 bits of the contents of that address are transferred to the P register, causing program execution to continue at that point. The instruction buffers are invalidated. The address specified by *nm* is a word address, but the lower 32 bits of that word specify a parcel address where program execution continues.

Instructions 010 through 013

Machine Instruction	CAL Syntax		Description
010000*nm*[D]	JAZ	*exp*	Jump to *exp* if (A0) = 0.
011000*nm*[D]	JAN	*exp*	Jump to *exp* if (A0) ≠ 0.
012000*nm*[D]	JAP	*exp*	Jump to *exp* if (A0) ≥ 0.
013000*nm*[D]	JAM	*exp*	Jump to *exp* if (A0) < 0.

D Difference in operation between Triton mode and C90 mode.

Special Cases

None.

Hold-issue Conditions

If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Execution Time

Instruction issue: 3 CPs.

Jump completion: 7 CPs. Additional time is required if the destination address is not in the instruction buffer.

Description

Instructions 010 through 013 are conditional jump instructions. Each instruction tests register A0 (treated as a two's-complement integer) against a condition specified by the *h* field of the instruction. If the condition is satisfied, the P register is set to the 32-bit parcel address (not word address) specified by the *nm* field, causing program execution to continue at that address. If the condition is not satisfied, program execution continues with the next sequential instruction.

In C90 mode, register A0 is 32 bits wide and bit 31 is the sign bit. Instructions 010 and 011 test all 32 bits of register A0. Register A0 is zero only if all 32 bits are 0's; if one or more bits are 1, A0 is nonzero. Instructions 012 and 013 test only the sign bit. If the sign bit is 0, register A0 is positive or zero. If the sign bit is 1, A0 is negative.

In Triton mode, register A0 is 64 bits wide and bit 63 is the sign bit. Instructions 010 and 011 test all 64 bits of register A0. Register A0 is zero only if all 64 bits are 0's; if one or more bits are 1, A0 is nonzero. Instructions 012 and 013 test only the sign bit. If the sign bit is 0, register A0 is positive or zero. If the sign bit is 1, A0 is negative.

Instructions 014 through 017

Machine Instruction	CAL Syntax		Description
014000*nm*	JSZ	*exp*	Jump to *exp* if (S0) = 0.
015000*nm*	JSN	*exp*	Jump to *exp* if (S0) ≠ 0.
016000*nm*	JSP	*exp*	Jump to *exp* if (S0) ≥ 0.
017000*nm*	JSM	*exp*	Jump to *exp* if (S0) < 0.

Special Cases

None.

Hold-issue Conditions

If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Execution Time

Instruction issue: 3 CPs.

Jump completion: 7 CPs. Additional time is required if the destination address is not in the instruction buffer.

Description

Instructions 014 through 017 are conditional jump instructions. Each instruction tests register S0 (treated as a two's-complement integer) against a condition specified by the *h* field of the instruction. If the condition is satisfied, the P register is set to the 32-bit parcel address (not word address) specified by the *nm* field, causing program execution to continue at that address. If the condition is not satisfied, program execution continues with the next sequential instruction.

Register S0 is 64 bits wide and bit 63 is the sign bit. Instructions 014 and 015 test all 64 bits of register S0. Register S0 is zero only if all 64 bits are 0's; if one or more bits are 1, S0 is nonzero. Instructions 016 and 017 test only the sign bit. If the sign bit is 0, register S0 is positive or zero. If the sign bit is 1, S0 is negative.

Instructions 020 through 022

Machine Instruction	CAL Syntax		Description
020*i*00*nm*[D]	A*i*	*exp*	Transmit *nm* to A*i* bits 0 – 31; A*i* bits 32 – 63 = 0.
020*i*20*nm*[NT]	A*i*	A*i*:*exp*	Transmit *nm* to A*i* bits 0 – 31; A*i* bits 32 – 63 unchanged.
020*i*40*nm*[NT]	A*i*	*exp*:A*i*	Transmit *nm* to A*i* bits 32 – 63; A*i* bits 0 – 31 unchanged.
021*i*00*nm*[D]	A*i*	*exp*	Transmit not(*nm*) to A*i* bits 0 – 31; A*i* bits 32 – 63 = 1.
022*ijk*	A*i*	*exp*	Transmit *jk* to A*i* bits 0 – 5; A*i* bits 6 – 63 = 0.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
D Difference in operation between Triton mode and C90 mode.

Special Cases

Instructions 020*i*20 and 020*i*40 are privileged to Triton mode. If these instructions are executed in C90 mode, the results are undefined.

Hold-issue Conditions

Instructions 020 and 021:

- Register A*i* reserved.
- If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instruction 022: register A*i* reserved.

Execution Time

Instructions 020 and 021:

- Instruction issue: 3 CPs.
- Register A*i* ready: 1 CP.

Instruction 022:

- Instruction issue: 1 CP.
- Register A*i* ready: 1 CP.

Description

Instructions 020 through 022 transmit constants to register A*i*.

Instruction 020*i*00 transmits the *nm* field to bits 0 through 31 of register A*i*. In Triton mode, bits 32 through 63 of register A*i* are cleared to 0's.

Instruction 020*i*20 (Triton mode only) transmits the *nm* field to bits 0 through 31 of register A*i*. Bits 32 through 63 of register A*i* are not changed.

Instruction 020*i*40 (Triton mode only) transmits the *nm* field to bits 32 through 63 of register A*i*. Bits 0 through 32 of register A*i* are not changed.

Instruction 021*i*00 transmits the inverse (one's complement) of the *nm* field to bits 0 through 31 of register A*i*. In Triton mode, bits 32 through 63 of register A*i* are set to 1's.

Instruction 022*ijk* transmits the *nm* field to bits 0 through 5 of register A*i*. In C90 mode, bits 6 through 31 of register A*i* are cleared to 0's. In Triton mode, bits 6 through 63 of register A*i* are cleared to 0's.

Note that the syntax "A*i exp*" can generate several possible machine instructions, depending on the value of *exp*. In general, the assembler generates the smallest possible instruction for a given *exp*. If *exp* is explicitly –1, instruction 005400 042*i*00 is generated. If *exp* can be determined at assembly time to be between 0 and 77_8, instruction 022*ijk* is generated.

In C90 mode, all other cases of *exp* cause instruction 020*i*00*nm* or 021i00*nm* to be generated.

In Triton mode, the syntax "A*i exp*" generates instruction 005400 042*i*00 or 022*ijk* in the cases explained above. If one of these instructions cannot be generated, but *exp* can be determined at assembly time to be less than or equal to 32 bits, instruction 020*i*00*mn* or 021*i*00*mn* is generated. All other cases of *exp* cause two instructions to be generated: 020*i*20*nm* and 020*i*40*mn*.

Instruction 023

Machine Instruction	CAL Syntax		Description
023*ij*0[D]	A*i*	S*j*	Transmit (S*j*) to A*i*.
023*i*01	A*i*	VL	Transmit (VL) to A*i*.
023*ij*6[NM]	A*i*	EA,*j*	Transmit exit address *j* to A*i*.
023*ij*7[NM]	A*i*	EA,A*j*	Transmit exit address (A*j*) to A*i*.

N New instruction (not available on CRAY C90 series systems).
D Difference in operation between Triton mode and C90 mode.
M Monitor mode only.

Special Cases

Instructions 023*ij*6 and 023*ij*7 are privileged to monitor mode.

If the CPU is in normal user mode, instructions 023*ij*6 and 023*ij*7 execute as no-ops.

If the CPU is in IMI mode, instructions 023*ij*6 and 023*ij*7 execute as no-ops, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

For instruction 023*ij*0: If *j* = 0, register A*i* is cleared.

For instruction 023*ij*7 in monitor mode: If *j* = 0, the content of register EA0 is transmitted to register A*i*.

Hold-issue Conditions

Instruction 023*ij*0:

- Register S*j* reserved (except S0).
- Register A*i* reserved.

Instruction 023*i*01: register A*i* reserved.

Instruction 023*ij*6:

- Register A*i* reserved.
- Shared path busy.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 023*ij*7:

- Register A*i* or A*j* reserved.
- Shared path busy.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Instructions 023*ij*0 and 023*i*01: register A*i* ready in 1 CP.

Instructions 023*ij*6 and 023*ij*7:

- Register A*i* ready: CPs undetermined.
- Shared path ready: CPs undetermined.

Description

Instruction 023*ij*

Instruction 023*ij*0 transmits the contents of register S*j* to register A*i*. In C90 mode, only bits 0 through 31 are transmitted; bits 32 through 63 are ignored. In Triton mode, all 64 bits are transmitted.

Instruction 023*i*1

Instruction 023*i*01 transmits the contents of the vector length (VL) register to register A*i*. The value returned is between 1 and 128 (between 1 and 200_8).

Instruction 023*ij*6

Instruction 023*ij*6 transmits to register A*i* the contents of the exit address (EA) register designated by the *j* field. The *j* field must be between 0 and 4; if *j* = 5, 6, or 7, the data returned is undefined.

Instruction 023*ij*7

Instruction 023*ij*7 transmits to register A*i* the contents of the exit address (EA) register designated by bits 0 through 2 of register A*j*. If register A*j* designates register EA5, EA6, or EA7, the data returned is undefined.

Instructions 024 and 025

Machine Instruction	CAL Syntax		Description
024*ijk*[D]	A*i*	B*jk*	Transmit (B*jk*) to A*i*.
025*ijk*[D]	B*j*	A*i*	Transmit (A*i*) to B*jk*.

D Difference in operation between Triton mode and C90 mode.

Special Cases

None.

Hold-issue Conditions

Instruction 024:

- Register A*i* reserved.
- Instruction 034 or 035 in progress.
- Instruction 025 issued in the previous CP.

Instruction 025:

- Register A*i* reserved.
- Instruction 034 or 035 in progress.

Execution Time

Instruction 024:

- Instruction issue: 1 CP.
- Register A*i* ready: 3 CPs.

Instruction 025:

- Instruction issue: 1 CP.
- Register B*j* ready: 2 CPs.

Description

Instruction 024 transmits the contents of the register B*jk* to register A*i*. In C90 mode, only bits 0 through 31 are transmitted; bits 32 through 63 are undefined. In Triton mode, all 64 bits are transmitted.

Instruction 025 transmits the contents of the register A*i* to register B*jk*. In C90 mode, only bits 0 through 31 are transmitted; bits 32 through 63 are undefined. In Triton mode, all 64 bits are transmitted.

Instructions 026*ij*0 through 026*ij*3

Machine Instruction	CAL Syntax		Description
026*ij*0	A*i*	PS*j*	Transmit population count of (S*j*) to A*i*.
026*ij*1	A*i*	QS*j*	Transmit population count parity of (S*j*) to A*i*.
026*ij*2[ND]	A*i*	PA*j*	Transmit population count of (A*j*) to A*i*.
026*ij*3[ND]	A*i*	QA*j*	Transmit population count parity of (A*j*) to A*i*.

D Difference in operation between Triton mode and C90 mode.
N New instruction (not available on CRAY C90 series systems).

Special Cases

If *j* = 0, register A*i* is cleared to 0.

Hold-issue Conditions

Instructions 026*ij*0 and 026*ij*1:

- Register S*j* reserved (except S0).
- Register A*i* reserved.
- Instruction 056 or 057 issued in the previous CP.

Instructions 026*ij*2 and 026*ij*3:

- Register A*j* reserved (except A0).
- Register A*i* reserved.
- Instruction 056 or 057 issued in the previous CP.

Execution Time

Instruction issue: 1 CP.

Register A*i* ready: 7 CPs.

Description

Instructions 026*ij*0 through 026*ij*3 use the scalar population/parity/leading-zero count functional unit.

Instruction 026*ij*0 counts the number of 1 bits in register S*j* and transmits the result to bits 0 through 6 of register A*i*. In C90 mode, bits 7 through 31 of register A*i* are cleared; bits 32 through 63 are undefined. In Triton mode, bits 7 through 63 of register A*i* are cleared.

Instruction 026*ij*1 counts the number of 1 bits in register S*j*. If the number of 1 bits is even, bit 0 of register A*i* is cleared to 0. If the number is odd, bit 0 of register A*i* is set to 1. In C90 mode, bits 1 through 31 of register A*i* are cleared; bits 32 through 63 are undefined. In Triton mode, bits 1 through 63 of register A*i* are cleared.

Instruction 026*ij*2 counts the number of 1 bits in register A*j* and transmits the result to register A*i*. In C90 mode, bits 0 through 31 of register A*j* are counted. The result is transmitted to bits 0 through 5 of register A*i*. Bits 6 through 31 of register A*i* are cleared; bits 32 through 63 are undefined. In Triton mode, bits 0 through 63 of register A*j* are counted. The result is transmitted to bits 0 through 6 of register A*i*. Bits 7 through 63 of register A*i* are cleared.

Instruction 026*ij*3 counts the number of 1 bits in register A*j*. In C90 mode, bits 0 through 31 of register A*j* are counted; in Triton mode, bits 0 through 63 are counted. If the number of 1 bits is even, bit 0 of register A*i* is cleared to 0. If the number is odd, bit 0 of register A*i* is set to 1. In C90 mode, bits 1 through 31 of register A*i* are cleared; bits 32 through 63 are undefined. In Triton mode, bits 1 through 63 of register A*i* are cleared.

Instructions 026*ij*4 through 026*ij*7

Machine Instruction	CAL Syntax		Description
026*ij*4[D]	A*i*	SB,A*j*,+1	Transmit (SB(A*j*)) to A*i*; increment (SB(A*j*)) by 1.
026*ij*5[D]	A*i*	SB*j*,+1	Transmit (SB*j*) to A*i*; increment (SB*j*) by 1.
026*ij*6[D]	A*i*	SB,A*j*	Transmit (SB(A*j*)) to A*i*.
026*ij*7[D]	A*i*	SB*j*	Transmit (SB*j*) to A*i*.

D Difference in operation between Triton mode and C90 mode.

Special Cases

For instructions 026*ij*4 and 026*ij*6: If *j* = 0, register SB0 is addressed.

If the cluster number (CLN) is 0 and IMI mode is not active, instructions 026*ij*4 through 026*ij*7 clear register A*i* to 0.

If the CPU is in IMI mode, instructions 026*ij*4 through 026*ij*7 return no result to register A*i*; the previous register value is retained. (Normally, in IMI mode, the cluster number should be 0.) If the cluster number is nonzero, register A*i* is not changed, but instructions 026*ij*4 and 024*ij*5 do increment register SBA(*j*) or SB(*j*). Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information.

If shared access is disabled (CPU configuration code 051_8) and IMI mode is not active, instructions 026*ij*4 through 026*ij*7 clear register A*i* to 0. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Hold-issue Conditions

Instructions 026*ij*4 and 026*ij*6:

- Register A*i* reserved.
- Register A*j* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 026*ij*5 and 026*ij*7:

- Register A*i* reserved.
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Register A*i* ready: CPs undetermined.

Shared path ready: CPs undetermined.

Description

Instructions 026*ij*4 through 026*ij*7 transmit data from a shared B (SB) register to an A register. All CPUs assigned to the same cluster share a set of 16 SB registers; each CPU has an independent set of eight A registers. If a CPU is in C90 mode, bits 32 through 63 of the receiving A register are undefined.

Instruction 026*ij*4 transmits to register A*i* the contents of the shared B register designated by register A*j*, then increments by 1 the contents of the shared B register designated by register A*j*.

Instruction 026*ij*5 transmits to register A*i* the contents of the shared B register designated by *j*, then increments by 1 the contents of shared B register designated by *j*. This instruction can address only the lower eight SB registers (0 through 7).

Instruction 026*ij*6 transmits to register A*i* the contents of the shared B register designated by register A*j*. No incrementing is performed.

Instruction 026*ij*7 transmits to register A*i* the contents of the shared B register designated by *j*. No incrementing is performed. This instruction can address only the lower eight SB registers (0 through 7).

Instruction 027*ij*0 and 027*ij*1

Machine Instruction	CAL Syntax		Description
027*ij*0	A*i*	ZS*j*	Transmit leading zero count of (S*j*) to A*i*.
027*ij*1[NT]	A*i*	ZA*j*	Transmit leading zero count of (A*j*) to A*i*.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.

Special Cases

Instruction 027*ij*1 is privileged to Triton mode. If this instruction is executed in C90 mode, the result is undefined.

If *j* = 0, register A*i* is set to 64 (100_8).

For instruction 027*ij*0, if (S*j*) = 0, register A*i* is set to 64 (100_8).

For instruction 027*ij*1, if (A*j*) = 0, register A*i* is set to 64 (100_8).

Hold-issue Conditions

Instruction 027*ij*0:

- Register S*j* reserved (except S0).
- Register A*i* reserved.
- Instruction 056 or 057 issued in the previous CP.

Instruction 027*ij*1:

- Register A*j* reserved (except S0).
- Register A*i* reserved.
- Instruction 056 or 057 issued in the previous CP.

Execution Time

Instruction issue: 1 CP.

Register A*i* ready: 7 CPs.

Description

Instructions 027*ij*0 and 027*ij*1 use the scalar population/parity/leading-zero count functional unit.

Instruction 027*ij*0 counts the number of leading zeroes (the number of 0's to the left of the left-most 1 bit) in register S*j* and transmits the result to bits 0 through 6 of register A*i*. In C90 mode, bits 7 through 31 of register A*i* are cleared; bits 32 through 63 are undefined. In Triton mode, bits 7 through 63 of register A*i* are cleared.

Instruction 027*ij*1 (Triton mode only) counts the number of leading zeroes (the number of 0's to the left of the left-most 1 bit) in register A*j* and transmits the result to bits 0 through 6 of register A*i*. Bits 7 through 63 of register A*i* are cleared.

Instructions 027*ij*2 and 027*ij*3

Machine Instruction	CAL Syntax		Description
027*ij*2[NM]	EA*j*	A*i*	Transmit (A*i*) to exit address *j*.
027*ij*3[NM]	EA,A*j*	A*i*	Transmit (A*i*) to exit address (A*j*).

N New instruction (not available on CRAY C90 series systems).
M Monitor mode only.

Special Cases

Instructions 0027*ij*2 and 0027*ij*3 are privileged to monitor mode.

If the CPU is in normal user mode, instructions 0027*ij*2 and 0027*ij*3 execute as no-ops.

If the CPU is in IMI mode, instructions 0027*ij*2 and 0027*ij*3 execute as no-ops, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information on monitor mode and IMI mode.

For instruction 027*ij*2, if $j = 5$, 6, or 7, the instruction executes as a no-op.

For instruction 027*ij*3 in monitor mode: If $j = 0$, the contents of register A*i* are transmitted to register EA0.

For instruction 027*ij*3: If $j \neq 0$ and bits 0 through 2 of register A*j* contain a value of 5, 6, or 7, the instruction executes as a no-op.

Hold-issue Conditions

Instruction 027*ij*2:

- Register A*i* reserved.
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 027*ij*3:

- Register A*i* reserved.
- Register A*j* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Shared path ready: CPs undetermined.

Description

Instructions 027*ij*2 and 027*ij*3 are privileged to monitor mode. These instructions transmit the contents of register A*i* to an exit address (EA) register. There are five EA registers: EA0 through EA4. The exit address register is loaded from bits 5 to 20 of register A*i*. The remaining bits of register A*i* are ignored.

Instruction 027*ij*2 transmits the contents of register A*i* to the EA register designated by the *j* field of the instruction.

Instruction 027*ij*3 transmits the contents of register A*i* to the EA register designated by bits 0 through 2 of register A*j*.

Instruction 027*ij*6 and 027*ij*7

Machine Instruction	CAL Syntax		Description
027*ij*6[D]	SB,A*j*	A*i*	Transmit (A*i*) to SB(A*j*).
027*ij*7[D]	SB*j*	A*i*	Transmit (A*i*) to SB*j*.

D Difference in operation between Triton mode and C90 mode.

Special Cases

Instructions 027*ij*6 and 027*ij*7 execute as no-ops if the cluster number (CLN) is 0.

For instruction 027*ij*6: If $j = 0$, the content of register A*i* is transmitted to register SB0.

If the CPU is in IMI mode, instructions 027*ij*6 and 027*ij*7 execute normally. (Normally, in IMI mode, the cluster number should be 0 so that these instructions are no-ops.) Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

If shared access is disabled (CPU configuration code 051_8), instructions 027*ij*6 and 027*ij*7 execute as no-ops. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Hold-issue Conditions

Instruction 027*ij*6:

- Register A*i* reserved.
- Register A*j* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instruction 027*ij*7:

- Register A*i* reserved.
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Register SB*j* ready: CPs undetermined.

Shared path ready: CPs undetermined.

Description

Instructions 027*ij*6 and 027*ij*7 transmit the contents of register A*i* to a shared B (SB) register. In C90 mode, the A and SB registers are 32 bits wide. In Triton mode, they are 64 bits wide.

Instruction 027*ij*6 transmits the contents of register A*i* to the SB register designated by bits 0 through 3 of register A*j*. The remaining bits of register A*j* are ignored.

Instruction 027*ij*7 transmits the contents of register A*i* to the SB register designated by the *j* field of the instruction. This instruction can write to registers SB0 through SB7 only. Instruction 027*ij*6 must be used to write to registers SB8 through SB15.

Instructions 030 and 031

Machine Instruction	CAL Syntax		Description
030*ijk*^{D}	A*i*	A*j*+A*k*	Transmit integer sum of (A*j*) and (A*k*) to A*i*.
030*i*0*k*^{D}	A*i*	A*k*^{S}	Transmit (A*k*) to A*i*.
030*ij*0^{D}	A*i*	A*j*+1^{S}	Transmit integer sum of (A*j*) and 1 to A*i*.
031*ijk*^{D}	A*i*	A*j*-A*k*	Transmit integer difference (A*j*) and (A*k*) to A*i*.
031*i*0*k*^{D}	A*i*	-A*k*^{S}	Transmit inverse of (A*k*) to A*i*.
031*ij*0^{D}	A*i*	A*j*-1^{S}	Transmit integer difference (A*j*) and 1 to A*i*.

D Difference in operation between Triton mode and C90 mode.
S Special CAL syntax.

Special Cases

For instruction 030:

- If $j = 0$ and $k = 0$, then 1 is transmitted to register A*i*.
- If $j = 0$ and $k \neq 0$, then (A*k*) is transmitted to register A*i*.
- If $j \neq 0$ and $k = 0$, then (A*j*) + 1 is transmitted to register A*i*.

For instruction 031:

- If $j = 0$ and $k = 0$, then –1 is transmitted to register A*i*.
- If $j = 0$ and $k \neq 0$, then –(A*k*) is transmitted to register A*i*.
- If $j \neq 0$ and $k = 0$, then (A*j*) – 1 is transmitted to register A*i*.

Hold-issue Conditions

Register A*i* reserved.

Register A*j* or A*k* reserved (except A0).

Execution Time

Instruction issue: 1 CP.

Register A*i* ready: 4 CPs.

Description

Instructions 030 and 031 use the address add functional unit. This functional unit does not detect overflow.

Instruction 030 transmits to register A*i* the integer sum of the contents of registers A*j* and A*k*.

Instruction 031 transmits to register A*i* the integer difference of the contents of registers A*j* and A*k*.

The operation of both instructions depends on whether the CPU is in C90 mode or Triton mode. In C90 mode the operands and result are all 32 bits wide. Only bits 0 through 31 of registers A*j* and A*k* are used as the operands; bits 32 through 63 are ignored. The result is transmitted to bits 0 through 31 of register A*i*; bits 32 through 63 are undefined.

In Triton mode, the operands and result are 64 bits wide. Bits 0 through 63 of register A*j* and A*k* are used as the operands. The result is transmitted to bits 0 through 63 of register A*i*.

Instruction 032

Machine Instruction	CAL Syntax		Description
032*ijk*[D]	A*i*	A*j**A*k*	Address multiply.

D Difference in operation between Triton mode and C90 mode.

Special Cases

If *j* = 0, then 0 is transmitted to register A*i*.

If *j* ≠ 0 and *k* = 0, then (A*j*) is transmitted to register A*i*.

Hold-issue Conditions

Register A*i* reserved.

Register A*j* or A*k* reserved (except A0).

Execution Time

Instruction issue: 1 CP.

Register A*i* ready: 8 CPs.

Description

Instruction 032 uses the address multiply functional unit. The instruction transmits the integer product of registers A*j* and A*k* to register A*i*.

In C90 mode, the operands and result are 32 bits wide. Only bits 0 through 31 of registers A*j* and A*k* are used as the operands; bits 32 through 63 are ignored. The result is transmitted to bits 0 through 31 of register A*i*; bits 32 through 63 are undefined. Overflow is not detected.

In Triton mode, the operands and result are 48 bits wide. Only bits 0 through 47 of registers A*j* and A*k* are used as the operands; bits 48 through 63 are ignored. The result is transmitted to bits 0 through 47 of register A*i* and sign-extended to bits 48 through 63. If an overflow occurs, the address multiply interrupt (AMI) flag in the exchange package sets.

Instruction 033

Machine Instruction	CAL Syntax		Description
033*i*00[DM]	A*i*	CI	Transmit channel number of highest-priority interrupt request to A*i*.
033*ij*0 (*j*≠0)[DM]	A*i*	CA,A*j*	Transmit current address of channel (A*j*) to register A*i*.
033*ij*1 (*j*≠0)[DM]	A*i*	CE,A*j*	Transmit status/error word of channel (A*j*) to register A*i*.

D Difference in operation between Triton mode and C90 mode.
M Monitor mode only.

Special Cases

After instruction 0012*j*0 or 0012*j*1 issues (to clear a channel interrupt flag), an undetermined number of CPs must elapse before the correct-priority channel number is available for instruction 033*i*00.

If the CPU is in normal user mode, instruction 033 executes as a no-op.

If the CPU is in IMI mode, instruction 033 returns no result to register A*i*; the previous register value is retained. Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

If a CPU reads status from a non-existent channel or a channel to which it does not have access, a word of all 1's is returned.

The operation of instruction 033 is modified by CPU configuration codes 050_8 through 055_8. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information.

NOTE: Although instruction 033 can be executed in C90 mode, instructions 033*ij*0 ($j \neq 0$) and 033*ij*1 ($j \neq 0$) return data in the upper 32 bits of register A*i*. Because these bits of register A*i* are not available in C90 mode, these instructions should be used only in Triton mode.

Hold-issue Conditions

Instruction 033*i*00:

- Register A*i* reserved.
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 033*ij*0 ($j \neq 0$) and 033*ij*1 ($j \neq 0$):

- Register A*i* or A*j* reserved.
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Execution Time

Instruction issue: 1 CP.

Register A*i* ready: CPs undetermined.

Shared path ready: CPs undetermined.

Description

Instruction 033 is privileged to monitor mode and enables the operating system to monitor the operation of the I/O channels. Data is returned to register A*i*.

Instruction 033*i*0

Instruction 033*i*00 transmits to register A*i* the number of the highest-priority channel with an interrupt request. Channel priority is determined by channel number; the lowest-numbered channel has the highest priority. If no channel has an interrupt request, instruction 033*i*00 returns a value of 0.

Instruction 033*ij*

Instruction 033*ij*0 ($j \neq 0$) transmits to register A*i* the contents of the channel address (CA) register of the channel designated by register A*j*. The CA register indicates the current common memory address being used by the channel; it increments as each word is transferred to or from common memory. Data is returned to register A*i* bits 0 through 34; bits 35 through 63 are forced to 0's.

Instruction 033*ij*1

Instruction 033*ij*1 ($j \neq 0$) transmits to register A*i* a status/error word from the channel designated by register (A*j*). The status/error word is different for LOSP and VHISP channels. There are also *pseudochannels* that return interrupt status information for groups of physical channels.

For LOSP channels, instruction 033*ij*1 returns the following status/error word:

Bit(s)	Description
0 – 28	0
29	Parity error (input channels only)
30	Error
31	Not Done
32 – 63	0

For VHISP channels, instruction 033*ij*1 returns the following status/error word:

Bit(s)	Description
0 – 23	Remaining block length
24 – 41	0
42	Transfer in progress
43	Block-length error
44	Double-bit error in SSD
45	Double-bit error in mainframe
46	Fatal error
47	Not Done
48 – 62	0
63	Not Done

If bit 46 is set and bits 43 through 45 are clear, the SSD is offline.

The Not Done bit is duplicated for programming convenience.

For the pseudochannels, 70, 72, 74, and 76, instruction 033*ij*1 returns the pending interrupt flags for groups of I/O channels. This is shown in the table below. Reading the status of a pseudochannel returns the interrupt flags of 16 LOSP channels (8 input and 8 output), 4 VHISP channels, and 2 support channels to register A*i*.

		Pseudochannel			
A*i* Bit	Channel Type	70	72	74	76
0	LOSP In	100	120	140	160
1	LOSP Out	101	121	141	161
2	LOSP In	102	122	142	162
3	LOSP Out	103	123	143	163
4	LOSP In	104	124	144	164
5	LOSP Out	105	125	145	165
6	LOSP In	106	126	146	166
7	LOSP Out	107	127	147	167
8	LOSP In	110	130	150	170
9	LOSP Out	111	131	151	171
10	LOSP In	112	132	152	172
11	LOSP Out	113	133	153	173
12	LOSP In	114	134	154	174
13	LOSP Out	115	135	155	175
14	LOSP In	116	136	156	176
15	LOSP Out	117	137	157	177
16	VHISP	020	024	030	034
17	VHISP	021	025	031	035
18	VHISP	022	026	032	036
19	VHISP	023	027	033	037
20	Support	060	062	064	066
21	Support	061	063	065	067

Instructions 034 through 037

Machine Instruction	CAL Syntax		Description
034*ijk*[D]	B*jk*,A*i*	,A0	Transmit (A*i*) words from common memory starting at address (A0) to B registers starting at register *jk*.
034*ijk*[D]	B*jk*,A*i*	0,A0[S]	Transmit (A*i*) words from common memory starting at address (A0) to B registers starting at register *jk*.
035*ijk*[D]	,A0	B*jk*,A*i*	Transmit (A*i*) words from B registers starting at register *jk* to memory starting at address (A0).
035*ijk*[D]	0,A0	B*jk*,A*i*[S]	Transmit (A*i*) words from B registers starting at register *jk* to memory starting at address (A0).
036*ijk*[D]	T*jk*,A*i*	,A0	Transmit (A*i*) words from memory starting at address (A0) to T registers starting at register *jk*.
036*ijk*[D]	T*jk*,A*i*	0,A0[S]	Transmit (A*i*) words from memory starting at address (A0) to T registers starting at register *jk*.
037*ijk*[D]	,A0	T*jk*,A*i*	Transmit (A*i*) words from T registers starting at register *jk* to memory starting at address (A0).
037*ijk*[D]	0,A0	T*jk*,A*i*[S]	Transmit (A*i*) words from T registers starting at register *jk* to memory starting at address (A0).

D Difference in operation between Triton mode and C90 mode.
S Special CAL syntax.

Special Cases

The number of words transferred is determined by bits 0 through 6 of register A*i*. (The higher-order bits of A*i* are ignored.) The transfer count is therefore between 0 and 177_8 (0 and 127_{10}). Two conditions can occur, depending on the value of the transfer count, as shown in the following table.

Transfer Count (N)	Operation
$0 – 100_8$	N words are transferred. No wrap-around condition occurs.
$101_8 – 177_8$	N words are transferred. Because there are only 100_8 B or T registers, a wrap-around condition occurs. For instruction 034 or 036, N – 100_8 registers are written twice with data from two different common memory locations; the second write operation overwrites the data from the first. For instructions 035 and 037, N – 100_8 registers are read twice and written to two different common memory locations.

Hold-issue Conditions

Instruction 034:

- Register A0 or A*i* reserved.
- Instruction 035 in progress.
- Port A busy.
- Port C busy and unidirectional memory mode.

Instruction 035:

- Register A0 or A*i* reserved.
- Instruction 034 in progress.
- Port C busy.
- Port A or port B busy and unidirectional memory mode.

Instruction 036:

- Register A0 or A*i* reserved.
- Instruction 037 in progress.
- Port B busy.
- Port C busy and unidirectional memory mode.

Instruction 037:

- Register A0 or A*i* reserved.
- Instruction 036 in progress.
- Port C busy.
- Port A or port B busy and unidirectional memory mode.

Execution Time

Instruction issue: 1 CP.

Instructions 034 and 036:

- B or T registers reserved for (A*i*) + 38 CPs.
- Port A or B busy for (A*i*) + 10 CPs.

Instructions 035 and 037:

- B or T registers reserved for (A*i*) + 6 CPs.
- Port C busy for (A*i*) + 12 CPs.

NOTE: Memory conflicts can delay completion of instructions 034 through 037.

Description

Instructions 034 through 037 perform block transfers between common memory and the B or T registers. Instruction 034 transfers data from common memory to the B registers using port A. Instruction 035 transfers data from the B registers to common memory using port C. Instruction 036 transfers data from common memory to the T registers using port B. Instruction 035 transfers data from the T registers to common memory using port C.

Bits 0 through 6 of register A*i* determine the number of B or T registers used in the data transfer.

The *jk* field of the instruction specifies the first register addressed. The B or T registers are addressed sequentially as far as necessary until register B77 or T77 is reached. If the transfer is not complete at this point, register B00 or T00 is addressed next and then addressing continues sequentially.

Register A0 determines the first common memory address used in the data transfer. The memory address is incremented by 1 after each word is transferred. In C90 mode, bits 0 through 31 of register A0 determine the starting memory address; bits 32 through 63 are undefined. In Triton mode, bits 0 through 39 of register A0 determine the starting memory address; bits 40 through 63 are ignored.

In C90 mode, the B registers are 32 bits wide. For instruction 034, bits 0 through 31 of each common memory word are transmitted to the B registers; bits 32 through 63 of the B registers are undefined. For instruction 035, bits 0 through 31 of each B register are transmitted to common memory; bits 32 through 63 of the receiving memory locations are cleared to 0's.

In Triton mode the B registers are 64 bits wide. Instructions 034 and 035 transmit complete 64-bit words between common memory and the B registers.

In both C90 and Triton modes, the T registers are 64 bits wide. Instructions 036 and 037 transmit full 64-bit words between common memory and the B registers.

Instructions 040 and 041

Machine Instruction	CAL Syntax		Description
040i00*nm*	S*i*	*exp*	Transmit *nm* to S*i* bits 0 – 31; S*i* bits 32 – 63 = 0.
040i20*nm*	S*i*	S*i*:*exp*	Transmit *nm* to S*i* bits 0 – 31; S*i* bits 32 – 63 unchanged.
040i40*nm*	S*i*	*exp*:S*i*	Transmit *nm* to S*i* bits 32 – 63; S*i* bits 0 – 31 unchanged.
041i00*nm*	S*i*	*exp*	Transmit inverse (*nm*) to S*i* bits 0 – 31; S*i* bits 32 – 63 = 1.

Special Cases

None.

Hold-issue Conditions

Register S*i* reserved.

If the second or third parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Execution Time

Issue time: 3 CPs.

Register S*i* ready: 1 CP.

Description

Instructions 040 and 041 transmit constants to register S*i*.

Instruction 040*i*00 transmits the *nm* field to bits 0 through 31 of register S*i*. Bits 32 through 63 of register S*i* are cleared to 0's.

Instruction 020*i*20 transmits the *nm* field to bits 0 through 31 of register S*i*. Bits 32 through 63 of register S*i* are not changed.

Instruction 020*i*40 transmits the *nm* field to bits 32 through 63 of register S*i*. Bits 0 through 32 of register S*i* are not changed.

Instruction 041*i*00 transmits the inverse (one's complement) of the *nm* field to bits 0 through 31 of register S*i*. Bits 32 through 63 of register A*i* are set to 1's.

Note that the syntax "S*i exp*" can generate several possible machine instructions, depending on the value of *exp*. In general, the assembler generates the smallest possible instruction for a given *exp*. For example, if *exp* is explicitly 0, instruction 043*i*00 is generated.

If *exp* is explicitly 1 or –1, instruction 042*i*77 or 042*i*00 is generated. If one of these instructions cannot be generated but *exp* can be determined at assembly time to be less than or equal to 32 bits, instruction 040*i*00*mn* or 041*i*00*mn* is generated. All other cases of *exp* cause two instructions to be generated: 040*i*20*nm* and 040*i*40*mn*.

Unprefixed Instructions 042 and 043

Machine Instruction	CAL Syntax		Description
042*ijk*	S*i*	<*exp*	Form ones mask in S*i* *exp* bits from right; *jk* field gets 100_8 – *exp*.
042*ijk*	S*i*	#>*exp*[S]	Form zeroes mask in S*i* *exp* bits from left; *jk* field gets *exp*.
042*i*77	S*i*	1[S]	Transmit 1 to S*i*.
042*i*00	S*i*	-1[S]	Transmit –1 to S*i*.
043*ijk*	S*i*	>*exp*	Form ones mask in S*i* *exp* bits from left; *jk* field gets *exp*.
043*ijk*	S*i*	#<*exp*[S]	Form zeroes mask in S*i* *exp* bits from right; *jk* field gets 100_8 – *exp*.
043*i*00	S*i*	0[S]	Clear S*i*.

S Special CAL syntax

Special Cases

None.

Hold-issue Conditions

Register S*i* reserved.

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 1 CP.

Description

Instruction 042 generates a mask of 100_8 – *jk* 1's from right to left in register S*i*. For example, if *jk* = 74_8, then 100_8 – *jk* = 4. Register S*i* then contains 1's in bits 0 through 3 and 0's in bits 4 through 63. If *jk* = 77_8, only bit 0 of S*i* is set to 1; all the other bits are cleared to 0's. If *jk* = 0, all bits of S*i* are set to 1's.

Instruction 043 generates a mask of *jk* 1's from left to right in register S*i*. For example, if *jk* = 4, then register S*i* contains 1's in bits 60 through 63 and 0's in bits 0 through 59. If *jk* = 0, all bits of S*i* are cleared to 0's. If *jk* = 77_8, only bit 0 of S*i* is cleared to 0; all the other bits are set to 1's.

005400-Prefixed Instructions 042 and 043

Machine Instruction	CAL Syntax		Description
005400 042*ijk*NT	A*i*	<*exp*	Form ones mask in A*i* exp bits from right; *jk* field gets 100_8 – *exp*.
005400 042*ijk*NT	A*i*	#>*exp*S	Form zeroes mask in A*i* *exp* bits from left; *jk* field gets *exp*.
005400 042i00NT	A*i*	-1S	Transmit –1 to A*i*.
005400 043*ijk*NT	A*i*	>*exp*	Form ones mask in A*i* *exp* bits from left; *jk* field gets *exp*.
005400 043*ijk*NT	A*i*	#<*exp*S	Form zeroes mask in A*i* *exp* bits from right; *jk* field gets 100_8 – *exp*.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
S Special CAL syntax.

Special Cases

Instructions 005400 042xxx and 005400 043xxx are privileged to Triton mode. If these instructions are executed in C90 mode, the results are undefined.

Hold-issue Conditions

Register A*i* reserved.

Execution Time

Instruction issue: 2 CPs.

Register A*i* ready: 1 CP.

Description

Instruction 005400 042 generates a mask of 100_8 – *jk* 1's from right to left in register A*i*. For example, if *jk* = 74_8, then 100_8 – *jk* = 4. Register A*i* then contains 1's in bits 0 through 3 and 0's in bits 4 through 63. If *jk* = 77, only bit 0 of A*i* is set to 1; all the other bits are cleared to 0's. If *jk* = 0, all bits of A*i* are set to 1's.

Instruction 005400 043 generates a mask of *jk* 1's from left to right in register A*i*. For example, if *jk* = 4, then register A*i* contains 1's in bits 60 through 26 and 0's in bits 0 through 59. If *jk* = 0, all bits of A*i* are cleared to 0's. If *jk* = 77_8, only bit 0 of A*i* is cleared to 0; all the other bits are set to 1's.

Unprefixed Instructions 044 through 051

Machine Instruction	CAL Syntax		Description
044*ijk*	S*i*	S*j*&S*k*	Transmit logical product of (S*j*) and (S*k*) to S*i*.
044*ij*0	S*i*	S*j*&SB[S]	Transmit sign bit of (S*j*) to S*i*.
044*ij*0	S*i*	SB&S*j*[S]	Transmit sign bit of (S*j*) to S*i* (*j* ≠ 0).
045*ijk*	S*i*	#S*k*&S*j*	Transmit logical product of (S*j*) and one's complement of (S*k*) to S*i*.
045*ij*0	S*i*	#SB&S*j*[S]	Transmit (S*j*) with sign bit cleared to S*i*.
046*ijk*	S*i*	S*j*|S*k*	Transmit logical difference of (S*j*) and (S*k*) to S*i*.
046*ij*0	S*i*	S*j*|SB[S]	Transmit (S*j*) with sign bit toggled to S*i*.
046*ij*0	S*i*	SB\S*j*[S]	Transmit (S*j*) with sign bit toggled to S*i* (*j* ≠ 0).
047*ijk*	S*i*	#S*j*|S*k*	Transmit logical equivalence of (S*j*) and (S*k*) to S*i*.
047*i*0*k*	Si	#S*k*[S]	Transmit one's complement of (S*k*) to S*i*.
047*ij*0	S*i*	#S*j*|SB[S]	Transmit (S*j*) with all bits except sign bit toggled to S*i*.
047*ij*0	S*i*	#SB\S*j*[S]	Transmit (S*j*) with all bits except sign bit toggled to S*i* (*j* ≠ 0).
047*i*00	S*i*	#SB[S]	Clear sign bit and set remaining bits of S*i*.
050*ijk*	S*i*	S*j*!S*i*&S*k*	Merge (S*i*) and (S*j*) to S*i* using (S*k*) as mask.
050*ij*0	S*i*	S*j*!S*i*&SB[S]	Merge (S*i*) and sign bit of (S*j*) to S*i*.
051*ijk*	S*i*	S*j*!S*k*	Transmit logical sum of (S*j*) and (S*k*) to S*i*.
051*i*0*k*	S*i*	S*k*[S]	Transmit (S*k*) to S*i*.
051*ij*0	S*i*	S*j*!SB[S]	Transmit (S*j*) with sign bit set to S*i*.
051*ij*0	S*i*	SB!S*j*[S]	Transmit (S*j*) with sign bit set to S*i* (*j* ≠ 0).
051*i*00	S*i*	SB[S]	Set sign bit and clear remaining bits of S*i*.

S Special CAL syntax.

Special Cases

If *j* = 0, then (S*j*) = 0.

If *k* = 0, then (S*k*) sign bit (bit 63) = 1; bits 0 through 62 = 0's.

Hold-issue Conditions

Register S*i* reserved.

Register S*j* or S*k* reserved (except S0).

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 1 CP.

Description

Instructions 044 through 051 use the scalar logical functional unit.

Instruction 044

Instruction 044 transmits the logical product (AND) of register S*j* and register S*k* to register S*i*. A bit of register S*i* is set to 1 only if the corresponding bits of registers S*j* and S*k* are both 1's, as shown in the following example:

```
Sj    1 1 0 0
Sk    1 0 1 0
      -------
Si    1 0 0 0
```

Instruction 045

Instruction 045 transmits the logical product (AND) of register S*j* and the one's complement of register S*k* to register S*i*. A bit of register S*i* is set to 1 only if the corresponding bit of register S*j* is 1 and the corresponding bit of register S*k* is 0, as shown in the following example:

```
Sj    1 1 0 0
Sk    1 0 1 0
      -------
Si    0 1 0 0
```

Instruction 046

Instruction 046 transmits the logical difference (exclusive OR) of register S*j* and register S*k* to register S*i*. A bit of register S*i* is set to 1 only if the corresponding bits of registers S*j* and S*k* are different, as shown in the following example:

```
Sj    1 1 0 0
Sk    1 0 1 0
      -------
Si    0 1 1 0
```

Instruction 047

Instruction 047 transmits the logical equivalence (exclusive NOR) of register S*j* and register S*k* to register S*i*. A bit of register S*i* is set to 1 only if the corresponding bits of registers S*j* and S*k* are the same, as shown in the following example:

```
Sj    1 1 0 0
Sk    1 0 1 0
      -------
Si    1 0 0 1
```

Instruction 050

Instruction 050 merges the contents of register S*j* with the contents of register S*i* and transmits the result to register S*i*. Each bit of the result is taken from either the corresponding bit of register S*j* or the corresponding bit of register S*i*, depending on the state of the corresponding bit in register S*k*. If the S*k* bit is 1, S*j* data is used; if the S*k* bit is 0, S*i* data is used.

Instruction 050 implements the function represented by the following equation: S*i* = S*j* S*k* + S*i* S*k*'; for example:

```
Si              =   1 1 0 0 1 1 0 0
Sj              =   1 0 1 0 1 0 1 0
Sk              =   1 1 1 1 0 0 0 0

Sj Sk           =   1 0 1 0 0 0 0 0
Si Sk'          =   0 0 0 0 1 1 0 0

Sj Sk + Si Sk'  =   1 0 1 0 1 1 0 0
```

Instruction 051

Instruction 051 transmits the logical product (inclusive OR) of register S*j* and register S*k* to register S*i*. A bit of register S*i* is set to 1 if either of the corresponding bits of registers S*j* and S*k* is 1, as shown in the following example:

S*j*	1 1 0 0
S*k*	1 0 1 0
S*i*	1 1 1 0

005400-Prefixed Instructions 044 through 051

Machine Instruction	CAL Syntax		Description
005400 044*ijk*[NT]	A*i*	A*j*&A*k*	Transmit logical product of (A*j*) and (A*k*) to A*i*.
005400 045*ijk*[NT]	A*i*	#A*k*&A*j*	Transmit logical product of (A*j*) and one's complement of (A*k*) to A*i*.
005400 046*ijk*[NT]	A*i*	A*j*!A*k*	Transmit logical difference of (A*j*) and (A*k*) to A*i*.
005400 047*ijk*[NT]	A*i*	#A*j*!A*k*	Transmit logical equivalence of (A*j*) and (A*k*) to A*i*.
005400 047*i*0*k*[NT]	A*i*	#A*k*[S]	Transmit one's complement of (A*k*) to A*i*
005400 050*ijk*[NT]	A*i*	A*j*!A*i*&A*k*	Merge A*i* and A*j* to A*i* using (A*k*) as mask.
005400 051*ijk*[NT]	A*i*	A*j*!A*k*	Transmit logical sum of (A*j*) and (A*k*) to A*i*.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
S Special CAL syntax.

Special Cases

Instructions 005400 044 through 005400 051 are privileged to Triton mode. If they are executed in C90 mode, the results are undefined.

If *j* = 0, then (A*j*) = 0.

If *k* = 0, then (A*k*) = 1.

Hold-issue Conditions

Register A*i* reserved.

Register A*j* or A*k* reserved (except SA).

Execution Time

Instruction issue: 2 CPs.

Register A*i* ready: 2 CPs.

Description

Instructions 005400 044 through 005400 051 use the scalar logical functional unit.

Instruction 005400 044

Instruction 005400 044 transmits the logical product (AND) of register A*j* and register A*k* to register A*i*. A bit of register A*i* is set to 1 only if the corresponding bits of registers A*j* and A*k* are both 1, as shown in the following example:

```
Aj    1 1 0 0
Ak    1 0 1 0
      -------
Ai    1 0 0 0
```

Instruction 005400 045

Instruction 005400 045 transmits the logical product (AND) of register A*j* and the one's complement of register A*k* to register A*i*. A bit of register A*i* is set to 1 only if the corresponding bit of register A*j* is 1 and the corresponding bit of register A*k* is 0, as shown in the following example:

```
Aj    1 1 0 0
Ak    1 0 1 0
      -------
Ai    0 1 0 0
```

Instruction 005400 046

Instruction 005400 046 transmits the logical difference (exclusive OR) of register A*j* and register A*k* to register A*i*. A bit of register A*i* is set to 1 only if the corresponding bits of registers A*j* and A*k* are different, as shown in the following example:

```
Aj    1 1 0 0
Ak    1 0 1 0
      -------
Ai    0 1 1 0
```

Instruction 005400 047

Instruction 005400 047 transmits the logical equivalence (exclusive NOR) of register A*j* and register A*k* to register A*i*. A bit of register A*i* is set to 1 only if the corresponding bits of registers A*j* and A*k* are the same, as shown in the following example:

```
Aj      1 1 0 0
Ak      1 0 1 0
        -------
Ai      1 0 0 1
```

Instruction 005400 050

Instruction 005400 050 merges the contents of register A*j* with the contents of register A*i* and transmits the result to register A*i*. Each bit of the result is taken from either the corresponding bit of register A*j* or the corresponding bit of register A*i*, depending on the state of the corresponding bit in register A*k*. If the A*k* bit is 1, A*j* data is used; if the A*k* bit is 0, A*i* data is used.

Instruction 050 implements the function represented by the following equation: A*i* = A*j* A*k* + A*i* A*k*'; for example:

```
Ai                  =   1 1 0 0 1 1 0 0
Aj                  =   1 0 1 0 1 0 1 0
Ak                  =   1 1 1 1 0 0 0 0

Aj Ak               =   1 0 1 0 0 0 0 0
Ai Ak'              =   0 0 0 0 1 1 0 0

Aj Ak + Ai Ak'      =   1 0 1 0 1 1 0 0
```

Instruction 005400 051

Instruction 005400 051 transmits the logical product (inclusive OR) of register A*j* and register A*k* to register A*i*. A bit of register A*i* is set to 1 if either of the corresponding bits of registers A*j* and A*k* is 1, as shown in the following example:

```
Aj      1 1 0 0
Ak      1 0 1 0
        -------
Ai      1 1 1 0
```

Unprefixed Instructions 052 through 055

Machine Instruction	CAL Syntax		Description
052*ijk*	S0	S*i*<*exp*	Shift (S*i*) left *exp* = *jk* places to S0.
053*ijk*	S0	S*i*>*exp*	Shift (S*i*) right *exp* = 100_8 – *jk* places to S0.
054*ijk*	S*i*	S*i*<*exp*	Shift (S*i*) left *exp* = *jk* places to S*i*.
055*ijk*	S*i*	S*i*>*exp*	Shift (S*i*) right *exp* = 100_8 – *jk* places to S*i*.

Special Cases

None.

Hold-issue Conditions

Instructions 052 and 053:

- Register S*i* or S0 reserved.
- Instruction 056 or 057 issued in previous CP.

Instructions 054 and 055:

- Register S*i* reserved.
- Instruction 056 or 057 issued in previous CP.

Execution Time

Instructions 052 and 053:

- Instruction issue: 1 CP.
- Register S0 ready: 4 CPs.

Instructions 054 and 055:

- Instruction issue: 1 CP.
- Register S*i* ready: 4 CPs.

Description

Instructions 052 through 055 use the scalar shift functional unit. All shifts are end-off with zero fill.

Instruction 052 shifts the contents of register S*i* left *jk* bits and transmits the result to register S0. The shift range is 0 through 63 bits.

Instruction 053 shifts the contents of register S*i* left $100_8 - jk$ bits and transmits the result to register S0. The shift range is 1 through 64 bits.

Instruction 054 shifts the contents of register S*i* left *jk* bits and transmits the result to register S*i*. The shift range is 0 through 63 bits.

Instruction 055 shifts the contents of register S*i* left $100_8 - jk$ bits and transmits the result to register S*i*. The shift range is 1 through 64 bits.

NOTE: Certain values of *exp* cause the assembler to generate different instructions.

- For the syntax S*i* < *exp*, if *exp* = 64, the assembler generates instruction 053000, which clears register S0 to 0.
- For the syntax S*i* > *exp*, if *exp* = 0, the assembler generates instruction 052000, which leaves register S0 unchanged.
- For the syntax S0 < *exp*, if *exp* = 64, the assembler generates instruction 055000, which clears register S*i* to 0.
- For the syntax S0 > *exp*, if *exp* = 0, the assembler generates instruction 054000, which leaves register S*i* unchanged.

005400-Prefixed Instructions 052 through 055

Machine Instruction	CAL Syntax		Description
005400 052*ijk*[NT]	A0	A*i*<*exp*	Shift (A*i*) left *exp* = *jk* places to A0.
005400 053*ijk*[NT]	A0	A*i*>*exp*	Shift (A*i*) right *exp* = 100_8 – *jk* places to A0.
005400 054*ijk*[NT]	A*i*	A*i*<*exp*	Shift (A*i*) left *exp* = *jk* places to A*i*.
005400 055*ijk*[NT]	A*i*	A*i*>*exp*	Shift (A*i*) right *exp* = 100_8 – *jk* places to A*i*.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.

Special Cases

Instructions 005400 052 through 005400 055 are privileged to Triton mode. If they are executed in C90 mode, the results are undefined.

Hold-issue Conditions

Instructions 005400 052 and 053: register A*i* or A0 reserved.

Instructions 005400 054 and 055: register A*i* reserved.

Execution Time

Instructions 005400 052 and 053:

- Instruction issue: 2 CPs.
- Register A0 ready: 5 CPs.

Instructions 005400 054 and 055:

- Instruction issue: 2 CPs.
- Register A*i* ready: 5 CPs.

Description

Instructions 005400 052 through 005400 055 use the scalar shift functional unit. All shifts are end-off with zero fill.

Instruction 005400 052 shifts the contents of register A*i* left *jk* bits and transmits the result to register A0. The shift range is 0 through 63 bits.

Instruction 005400 053 shifts the contents of register A*i* left $100_8 - jk$ bits and transmits the result to register A0. The shift range is 1 through 64 bits.

Instruction 005400 054 shifts the contents of register A*i* left *jk* bits and transmits the result to register A*i*. The shift range is 0 through 63 bits.

Instruction 005400 055 shifts the contents of register A*i* left $100_8 - jk$ bits and transmits the result to register A*i*. The shift range is 1 through 64 bits.

NOTE: Certain values of *exp* cause the assembler to generate different instructions than those shown in the preceding table.

- For the syntax A*i* < *exp*, if *exp* = 64, the assembler generates instruction 005400 053000, which clears register A0 to 0.
- For the syntax A*i* > *exp*, if *exp* = 0, the assembler generates instruction 005400 052000, which leaves register A0 unchanged.
- For the syntax A0 < *exp*, if *exp* = 64, the assembler generates instruction 005400 055000, which clears register A*i* to 0.
- For the syntax A0 > *exp*, if *exp* = 0, the assembler generates instruction 005400 054000, which leaves register A*i* unchanged.

Unprefixed Instructions 056 and 057

Machine Instruction	CAL Syntax		Description
056*ijk*[D]	S*i*	S*i*,S*j*<A*k*	Shift (S*i*) and (S*j*) left (A*k*) places to S*i*.
056*ij*0[D]	S*i*	S*i*,S*j*<1[S]	Shift (S*i*) and (S*j*) left one place to S*i*.
056*i*0*k*[D]	S*i*	S*i*<A*k*[S]	Shift (S*i*) left (A*k*) places to S*i*.
057*ijk*[D]	S*i*	S*j*,S*i*>A*k*	Shift (S*j*) and (S*i*) right (A*k*) places to S*i*.
057*ij*0[D]	S*i*	S*j*,S*i*>1[S]	Shift (S*j*) and (S*i*) right one place to S*i*.
057*i*0*k*[D]	S*i*	S*i*>A*k*[S]	Shift (S*i*) right (A*k*) places to S*i*.

D Difference in operation between Triton mode and C90 mode.
S Special CAL syntax.

Special Cases

If $j = 0$, then (Sj) = 0.

If $k = 0$, then (Ak) = 1.

Hold-issue Conditions

Register S*i* reserved.

Register S*j* reserved (except S0).

Register A*k* reserved (except A0).

Instruction 056 or 057 issued in previous CP.

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 5 CPs.

Description

Instructions 056 and 057 use the scalar shift functional unit. All shifts are end-off with zero fill. Register A*k* provides the shift count. In C90 mode, only bits 0 through 31 of register A*k* are used; bits 32 through 63 are ignored. In Triton mode, all 64 bits of register A*k* are used. All shift counts are positive (or zero).

Instruction 056 concatenates registers S*i* and S*j* to form a 128-bit value. (Register S*i* supplies bits 64 through 127; register S*j* supplies bits 0 through 63.) It then left shifts the 128-bit value by the number of places specified by register A*k* and transmits bits 64 through 127 of the result to register S*i*. If the shift count is greater than 127, register S*i* is cleared.

Instruction 057 concatenates registers S*j* and S*i* to form a 128-bit value. (Register S*j* supplies bits 64 through 127; register S*i* supplies bits 0 through 63.) It then right shifts the 128-bit value by the number of places specified by register A*k* and transmits bits 0 through 63 of the result to register S*i*. If the shift count is greater than 127, register S*i* is cleared.

NOTE: A circular left shift can be effected by using instruction 056 with $i = j$ and $(Ak) \leq 64$. A circular right shift can be effected by using instruction 057 with $i = j$ and $(Ak) \leq 64$.

005400-Prefixed Instructions 056 and 057

Machine Instruction	CAL Syntax		Description
005400 056*ijk*[NT]	A*i*	A*i*,A*j*<A*k*	Shift (A*j*) and (A*k*) left (A*k*) places to A*i*.
005400 056*ij*0[NT]	A*i*	A*i*,A*j*<1[S]	Shift (A*j*) and (A*j*) left one place to A*i*.
005400 056*i*0*k*[NT]	A*i*	A*i*<A*k*[S]	Shift (A*j*) left (A*k*) places to A*i*.
005400 057*ijk*[NT]	A*i*	A*j*,A*i*>A*k*	Shift (A*j*) and (A*i*) right (A*k*) places to A*i*.
005400 057*ij*0[NT]	A*i*	A*j*,A*i*>1[S]	Shift (A*j*) and (A*i*) right one place to A*i*.
005400 057*i*0*k*[NT]	A*i*	A*i*>A*k*[S]	Shift (A*i*) right (A*k*) places to A*i*.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
S Special CAL syntax.

Special Cases

Instructions 005400 056 and 005400 057 are privileged to Triton mode. If they are executed in C90 mode, the results are undefined.

If $j = 0$, then (Sj) = 0.

If $k = 0$, then (Ak) = 1.

Hold-issue Conditions

Register A*i* reserved.

Register A*i* or Ak reserved (except A0).

Execution Time

Instruction issue: 2 CPs.

Register A*i* ready: 6 CPs.

Description

Instructions 005400 056 and 005400 057 use the scalar shift functional unit. All shifts are end-off with zero fill. Register A*k* provides the shift count. All shift counts are positive (or zero).

Instruction 005400 056 concatenates registers A*i* and A*j* to form a 128-bit value. (Register A*i* supplies bits 64 through 127; register A*j* supplies bits 0 through 63.) It then left shifts the 128-bit value by the number of places specified by register A*k* and transmits bits 64 through 127 of the result to register A*i*. If the shift count is greater than 127, register A*i* is cleared.

Instruction 005400 057 concatenates registers A*j* and A*i* to form a 128-bit value. (Register A*j* supplies bits 64 through 127; register A*i* supplies bits 0 through 63.) It then right shifts the 128-bit value by the number of places specified by register A*k* and transmits bits 0 through 63 of the result to register A*i*. If the shift count is greater than 127, register A*i* is cleared.

NOTE: A circular left shift can be effected by using instruction 005400 056 with $i = j$ and (A*k*) ≤ 64. A circular right shift can be effected by using instruction 005400 057 with $i = j$ and (A*k*) ≤ 64.

Instructions 060 and 061

Machine Instruction	CAL Syntax		Description
060*ijk*	S*i*	S*j*+S*k*	Transmit integer sum of (S*j*) and (S*k*) to S*i*.
061*ijk*	S*i*	S*j*-S*k*	Transmit integer difference of (S*j*) and (S*k*) to S*i*.
061*i*0*k*	S*i*	-S*k*[S]	Transmit negative of (S*k*) to S*i*.

S Special CAL syntax.

Special Cases

If *j* = 0, then (S*j*) = 0.

If *k* = 0, then (S*k*) sign bit (bit 63) = 1, bits 0 through 62 = 0.

Hold-issue Conditions

Register S*i* reserved.

Register S*j* or S*k* reserved (except S0).

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 4 CPs.

Description

Instructions 060 and 061 use the scalar add functional unit. This functional unit performs two's complement arithmetic. It does not detect overflow.

Instruction 060*ijk* transmits the integer sum of register S*j* and register S*k* to register S*i*.

Instruction 061*ijk* transmits the integer difference of register S*j* and register S*k* to register S*i*.

Instructions 062 and 063

Machine Instruction	CAL Syntax		Description
062*ijk*	S*i*	S*j*+FS*k*	Transmit floating-point sum of (S*j*) and (S*k*) to S*i*.
062*i*0*k*	S*i*	+FS*k*[S]	Transmit normalized (S*k*) to S*i*.
063*ijk*	S*i*	S*j*-FS*k*	Transmit floating-point difference of (S*j*) and (S*k*) to S*i*.
063*i*0*k*	S*i*	-FS*k*[S]	Transmit normalized negative of (S*k*) to S*i*.

S Special CAL syntax.

Special Cases

If $j = 0$, then (Sj) = 0. This is the floating-point representation of 0.

If $k = 0$, then (Sk) sign bit (bit 63) = 1; bits 0 through 62 = 0. This is the floating-point representation of –0, which is treated the same as 0.

Hold-issue Conditions

Register Si reserved.

Register Sj or Sk reserved (except S0).

Floating-point add functional unit busy. Instructions 170 through 173 cause this functional unit to be busy for VL/2 + 8 CPs.

Execution Time

Instruction issue: 1 CP.

Register Si ready: 8 CPs.

Description

Instructions 062 and 063 use the floating-point add functional unit. The operands of this functional unit must be in (unnormalized) floating-point format; this functional unit returns results in normalized floating-point format. It detects both underflow and overflow conditions.

If underflow occurs, the result returned is a word of all 0's (floating-point 0); no flag is set. If overflow occurs, the result returned is as follows: sign bit = 0, exponent = 60000_8, and coefficient = as calculated. An overflow also sets the floating-point error (FPE) flag in the exchange package.

Instruction 062*ijk* transmits the normalized floating-point sum of register S*j* and register S*k* to register S*i*.

Instruction 063*ijk* transmits the normalized floating-point difference of register S*j* and register S*k* to register S*i*.

Instructions 064 through 067

Machine Instruction	CAL Syntax		Description
064*ijk*	S*i*	S*j**FSk	Transmit floating-point product of (S*j*) and (S*k*) to S*i*.
065*ijk*	S*i*	S*j**HSk	Transmit half-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*.
066*ijk*	S*i*	S*j**RSk	Transmit rounded floating-point product of (S*j*) and (S*k*) to S*i*.
067*ijk*	S*i*	S*j**ISk	Transmit 2 – (S*j*) * (S*k*) to S*i* (reciprocal iteration).

Special Cases

If $j = 0$, then (Sj) = 0. This is the floating-point representation of 0.

If $k = 0$, then (Sk) sign bit (bit 63) = 1; bits 0 through 62 = 0. This is the floating-point representation of –0, which is treated the same as 0.

If the exponent fields of both operands are 0, an integer multiplication operation is performed. The result returned is as follows: sign bit = 0, exponent = 0, coefficient = most-significant 48 bits of the 96-bit product. Correct results are produced if the following conditions are satisfied:

- Both operand sign bits are 0.
- $X + Y \geq 48$, where X is the number of 0 bits to the right of the least significant 1 bit in the first operand and Y is the number of 0 bits to the right of the least significant 1 bit in the second operand.

Hold-issue Conditions

Register S*i* reserved.

Register S*j* or S*k* reserved (except S0).

Floating-point multiply functional unit busy. Instructions 160 through 167 cause this functional unit to be busy for VL/2 + 8 CPs.

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 8 CPs.

Description

Instructions 064 through 067 use the floating-point multiply functional unit. This functional unit must be in floating-point format and returns results in floating-point format. (A result is normalized only if both operands are normalized.) The functional unit detects both underflow and overflow conditions.

If underflow occurs, the result returned is a word of all 0's (floating-point 0); no flag is set. If overflow occurs, the result returned is as follows: sign bit = 0, exponent = 60000_8, and coefficient = as calculated. An overflow also sets the floating-point error (FPE) flag in the exchange package.

Instruction 064*ijk* transmits the floating-point product of register S*j* and register S*k* to register S*i*.

Instruction 065*ijk* transmits the half-precision rounded floating-point product of register S*j* and register S*k* to register Si. The least significant 19 bits of the result are cleared. This instruction can be used in the division algorithm when only 30 bits of accuracy are required.

Instruction 066*ijk* transmits the rounded floating-point product of register S*j* and register S*k* to register S*i*. This instruction can be used in the division algorithm.

Instruction 067*ijk* transmits two minus the floating-point product of register S*j* and register S*k* to register S*i*. This instruction can be used in the reciprocal and division algorithms.

Instruction 070*ij*0

Machine Instruction	CAL Syntax		Description
070*ij*0	S*i*	/HS*j*	Transmit floating-point reciprocal approximation of (S*j*) to S*i*.

Special Cases

If $j = 0$, then (Sj) = 0. This is the floating-point representation of 0. A range error occurs.

The operand is assumed to be normalized (bit 47 assumed to be 1). If the operand is not normalized, the result is invalid.

Hold-issue Conditions

Register S*i* reserved.

Register S*j* reserved (except S0).

Floating-point reciprocal approximation functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:

- Instruction 070*ij*1: (highest 1-bit position in VM)/2 + 8 CPs
- Instruction 174*ij*0: VL/2 + 12 CPs
- Instructions 174*ij1* through 174*ij*3: VL/2 + 5 CPs

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 19 CPs.

Description

Instruction 070*ij*0 uses the floating-point reciprocal approximation functional unit. The operands of this functional unit must be in normalized floating-point format; this functional unit returns results in normalized floating-point format. The functional unit handles overflow and underflow conditions identically. If the exponent of the operand is less than 20002_8 or greater than 60001_8, the result returned is as follows:

sign bit = 0, exponent = 60000_8, and coefficient = as calculated. An overflow or underflow condition also sets the floating-point error (FPE) flag in the exchange package.

Instruction 070*ij*0 transmits the floating-point reciprocal approximation of register S*j* to register S*i*. The result is accurate to 30 significant bits. The lower 18 bits are 0's. The number of significant bits can be increased to 48 by performing a reciprocal iteration (067*ijk*) and multiply (064*ijk*) sequence.

Instruction 070*ij*1

Machine Instruction	CAL Syntax		Description
070*ij*1[N]	V*i*	CI,S*j*&VM	Transmit compressed index of (S*j*) controlled by (VM) to V*i*.

N New instruction (not available on CRAY C90 series systems).

Special Cases

If *j* = 0, then (S*j*) = 0.

Hold-issue Conditions

Register V*i* reserved as operand or result.

Register S*j* reserved (except S0).

Floating-point reciprocal approximation functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:

- Instruction 070*ij*0: 8 CPs
- Instruction 174*ij*0: VL/2 + 12 CPs
- Instructions 174*ij1* through 174*ij*3: VL/2 + 5 CPs

VM register busy. The following instructions cause this register to be busy for the following intervals:

- Instructions 146, 147, and 005400 153: 3 CPs
- Instruction 175: VL/2 + 8 CPs

Execution Time

Instruction issue: 1 CP.

VM register free: 4 CPs.

Floating-point reciprocal approximation functional unit ready: (highest 1-bit position in VM)/2 + 10 CPs.

Register V*i* ready: (highest 1-bit position in VM)/2 + 13 CPs.

Description

Instruction 070*ij*0 uses the floating-point reciprocal approximation functional unit.

Instruction 070*ij*1 forms multiples of the contents of register S*j* starting with 0 (0, S*j*, 2 × S*j*, 3 × S*j*, …). It stores multiples corresponding to each 1 bit in the vector mask (VM) register in successive elements of register V*i* (beginning at element 0). The instruction terminates when all the unused bits of VM are 0's or all of VM is used.

An example is shown below:

S*j* = 2

VM = 10010110 … 0

Then V*i*[0] = 0
V*i*[1] = 6
V*i*[2] =10
V*i*[3] = 12
V*i*[4] through V*i*[127] are not changed

Instruction 070*ij*6

Machine Instruction	CAL Syntax		Description
070*ij*6[N]	S*i*	S*j**BT	Transmit bit-matrix product of (S*j*) and (B^T) to S*i*.

N New instruction (not available on CRAY C90 series systems).

Special Cases

If $j = 0$, then (Sj) = 0.

Hold-issue Conditions

Register S*i* reserved.

Register S*j* reserved (except S0).

Bit-matrix multiply functional unit busy. Instructions 1740*j*4 through 1740*j*7 cause this functional unit to be busy for VL/2 + 5 CPs.

Execution Time

Instruction issue: 1 CP.

Register S*i* ready: 9 CPs.

Description

Instruction 070*ij*6 uses the bit-matrix multiply functional unit. This functional unit contains a 64 word × 64 bit matrix called B.

Instruction 070*ij*6 transmits to register S*i* the bit-matrix product of register S*j* and matrix B^T (matrix B transposed). The matrix B must have been previously loaded with instruction 174*ij*4 or 174*ij*5.

Instruction 071

Machine Instruction	CAL Syntax		Description
071*i*0*k*[D]	S*i*	A*k*	Transmit (A*k*) with no sign extension to S*i*.
071*i*1*k*[D]	S*i*	+A*k*	Transmit (A*k*) with sign extension to S*i*.
071*i*2*k*[D]	S*i*	+FA*k*	Transmit (A*k*) as unnormalized floating-point number to S*i*.
071*i*30	S*i*	0.6	Transmit 0.75 x 2^{48} as normalized floating-point constant to S*i*.
071*i*40	S*i*	0.4	Transmit 0.5_8 as normalized floating-point constant to S*i*.
071*i*50	S*i*	1.0	Transmit 1.0 as normalized floating-point constant to S*i*.
071*i*60	S*i*	2.0	Transmit 2.0 as normalized floating-point constant to S*i*.
071*i*70	S*i*	4.0	Transmit 4.0 as normalized floating-point constant to S*i*.

D Difference in operation between Triton mode and C90 mode.

Special Cases

For instructions 071*i*0*k* and 071i1*k*: If *k* = 0, register S*i* is set to 1.

For instruction 071*i*2*k*: If *k* = 0, register S*i* is loaded as follows:

- Sign (bit 63): 0.
- Exponent (bits 48 through 62): 40060_8.
- Coefficient (bits 0 through 47): 1.

Hold-issue Conditions

Instructions 071*i*0*k* through 071*i*2*k*:

- Register S*i* reserved.
- Register A*k* reserved (except A0).

Instruction 071*i*3*k* through 071i7*k*: register S*i* reserved.

Execution Time

Instructions 071*i*0*k* and 071*i*3*k* through 071*i*7*k*:

- Instruction issue: 1 CP.
- Register S*i* ready: 1 CP.

Instruction 071*i*1*k*:

- Instruction issue: 1 CP.
- Register S*i* ready: 3 CPs.

Instruction 071*i*2*k*:

- Instruction issue: 1 CP.
- Register S*i* ready: 4 CPs.

Description

Instructions 071*ik* through 071*i*2*k*

Instructions 071*i*0*k* through 071*i*2*k* transmit the contents of register A*k* to register S*i*. These instructions operate somewhat differently, depending on whether the CPU is operating in C90 mode (32-bit A registers) or Triton mode (64-bit A registers).

Instruction 071*i*0*k* treats the contents of register A*k* as an unsigned integer and transmits it to register S*i* without sign extension. In C90 mode, the contents of register A*k* are transmitted to bits 0 through 31 of register S*i*; bits 32 through 63 of register S*i* are cleared to 0's. In Triton mode, the contents of register A*k* are transmitted to register S*i* without modification.

Instruction 071*i*1*k* treats the contents of register A*k* as a signed (two's complement) integer and transmits it to register S*i* with sign extension. In C90 mode, the contents of register A*k* are transmitted to bits 0 through 31 of register S*i*; bit 31 is copied to bits 32 through 63. In Triton mode, the contents of register A*k* are transmitted to register S*i* without modification.

Instruction 071*i*2*k* treats the contents of register A*k* as a signed (two's complement) integer and transmits it to register S*i* as an unnormalized floating-point number.

In C90 mode, instruction 071*i*2*k* loads register S*i* as follows:

- The exponent (bits 48 through 62) is set to 40060_8.
- The sign bit (bit 63) is copied from bit 31 of register A*i*.
- If the sign bit is 0, the coefficient (bits 0 through 31) is copied from register A*i*; bits 32 through 48 are cleared to 0's. If the sign bit is 1, the coefficient is the two's complement of register A*i*. Bits 32 through 48 are cleared to 0's.

In Triton mode, instruction 071*i*2*k* loads register S*i* as follows:

- The exponent (bits 48 through 62) is set to 40060_8.
- The sign bit (bit 63) is copied from bit 63 of register A*i*.
- If the sign bit is 0, the coefficient (bits 0 through 47) is copied from register A*i*. If the sign bit is 1, the coefficient is the two's complement of register A*i*. Bits 48 through 62 of register A*i* are ignored.

The following instruction sequence converts a two's complement integer in register S1 to normalized floating-point format. Register S1 must initially contain a two's complement integer whose coefficient is less than 31 (C90 mode) or 48 (Triton mode).

- 023110 A1 S1 transmit S1 to A1
- 071121 S1 +FA1 transmit A1 as unnormalized floating-point number to S1
- 062101 S1 +FS1 normalize S1

Instructions 071*i*30 through 071*i*70

Instructions 071*i*30 through 071*i*70 transmit commonly used floating-point constants to register S*i*.

Instruction 071*i*30 transmits the floating-point constant of 0.75 x bit 48 to register S*i*. (04006060000000000000_8). This constant is used to create floating-point numbers from two's complement integers whose absolute value is less than 46. The following instruction sequence converts a two's complement integer in register S1 to normalized floating-point format:

- 071230 S2 0.6
- 061121 S1 S2-S1
- 063121 S1 S2-FS1

Instruction 071*i*40 transmits the floating-point constant 0.4_8 (0.5_{10}) (04000040000000000000_8) to register S*i*.

Instruction 071*i*50 transmits the floating-point constant 1.0 (040001400000000000000_8) to register S*i*.

Instruction 071*i*60 transmits the floating-point constant 2.0 (040002400000000000000_8) to register S*i*.

Instruction 071*i*70 transmits the floating-point constant 4.0 (040003400000000000000_8) to register S*i*.

Instruction 072 and 073

Machine Instruction	CAL Syntax		Description
072*i*00	S*i*	RT	Transmit real-time clock to S*i*.
072*i*02^{V}	S*i*	SM	Transmit semaphores to S*i*.
072*ij*3	S*i*	ST*j*	Transmit (ST*j*) register to S*i*.
072*ij*6^{V}	S*i*	ST,A*j*	Transmit ST(A*j*) to S*i*.
073*i*00	S*i*	VM0	Transmit (VM0) to S*i*.
073*i*10	S*i*	VM1	Transmit (VM1) to S*i*.
073*i*20^{NT}	A*i*	VM0	Transmit (VM0) to A*i*.
073*i*30^{NT}	A*i*	VM1	Transmit (VM1) to A*i*.
073*ij*1^{VM}	S*i*	SR*j*	Transmit (SR*j*) to S*i* (monitor mode only for *j* = 2 – 7).
073*i*02^{V}	SM	S*i*	Transmit (S*i*) to semaphores.
073*ij*3	ST*j*	S*i*	Transmit (S*i*) to ST*j*.
073*i*05	SR0	S*i*	Transmit (S*i*) bits 48 – 52 to SR0.
073*i*25^{O}	SR2	S*i*	Advance performance monitor pointer.
073*i*75VO	SR7	S*i*	Transmit (S*i*) to maintenance channel.
073*ij*6^{V}	ST,A*j*	S*i*	Transmit (S*i*) to ST(A*j*).

N New instruction (not available on CRAY C90 series systems).
V New version of CRAY C90 instruction.
T Triton mode only.
M Monitor mode only for some values of *j*.
O Maintenance mode only.

Special Cases

Instructions 072*i*0

If the CPU is in IMI mode, instruction 072*i*00 executes normally. Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

Instructions 072*i*2, 072*ij*3, and 072*ij*6

For instruction 072*ij*6: If *j* = 0, register ST0 is addressed.

If the cluster number (CLN) is 0 and IMI mode is not active, instructions 072*i*02, 072*ij*3, and 072*ij*6 clear register S*i* to 0.

If the CPU is in IMI mode, instructions 072*i*02, 072*ij*3, and 072*ij*6 return no result to register S*i*; the previous register value is retained. (Normally, in IMI mode, the cluster number should be 0.) Following execution, the

MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

If shared access is disabled (CPU configuration code 051_8) and IMI mode is not active, instructions 073*i*02, 073*ij*3, and 073*ij*6 clear register S*i* to 0. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Instructions 073*i*0 and 073*i*10

None.

Instructions 073*i*20 and 073*i*30

Instructions 073*i*20 and 073*i*30 are privileged to Triton mode. If these instructions are executed in C90 mode, the results are undefined.

Instruction 073*ij*1 (*j* = 0)

If the CPU is in IMI mode, instruction 073*ij*1 (*j* = 0) executes normally. Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

Instruction 073*ij*1 (*j* = 2 or 3)

Instruction 073*ij*1 (*j* = 2 or 3) is privileged to monitor mode. The PMBY bit should be clear. If the CPU is in monitor mode and PMBY is set, the results are undefined. For more information on the PMBY bit, refer to the description of instruction 073*ij*1.

If the CPU is in normal user mode, this instruction returns 0 to register S*i*.

If the CPU is in IMI mode, this instruction executes normally except that the performance monitor pointer is prevented from advancing. Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction.

Instruction 073*ij*1 (*j* = 4 through 7)

Instruction 073*ij*1 (*j* = 4 through 7) is privileged to monitor mode. The PMBY bit should be clear. If the CPU is in monitor mode and PMBY is set, the results are undefined. For more information on the PMBY bit, refer to the description of instruction 073*ij*1.

If the CPU is in normal user mode, instruction 073*ij*1 returns 0 to register S*i*.

If the CPU is in IMI mode, this instruction returns 0 to register S*i*. Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

Instructions 073*i*2, 073*ij*3, and 073*ij*6

Instructions 073*i*02, 073*ij*3, and 073*ij*6 execute as no-ops if the cluster number (CLN) is 0.

For instruction 073*ij*6: If *j* = 0, register ST0 is loaded.

If the CPU is in IMI mode, instructions 073*i*02, 073*ij*3, and 073*ij*6 execute normally. (Normally in IMI mode, the cluster number should be 0 so that these instructions are no-ops.) Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

If shared access is disabled (CPU configuration code 051_8), instructions 073*i*02, 073*ij*3, and 073*ij*6 execute as no-ops. Refer to Section 4 of PRN-0957, *Triton Maintenance System*, for more information on configuration codes.

Instruction 073*i*5

If the CPU is in IMI mode, instruction 073*i*05 executes as a no-op, the MII flag sets, and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

Instruction 073*i*25

Instruction 073*i*25 is privileged to maintenance mode. If maintenance mode is not active, the instruction executes as a no-op.

If the CPU is in IMI mode, instruction 073*i*25 executes normally (or as a no-op if maintenance mode is not active). Following execution, the MII flag sets and an exchange sequence occurs. The P-register field of the exchange package points to the parcel following the instruction. Refer to the *Instruction Set Overview* for more information about IMI mode.

Instruction 073*i*75

Instruction 073*i*75 is privileged to maintenance mode. If maintenance mode is not active, the instruction executes as a no-op.

Before instruction 073*i*75 is executed, register SR0 bit 0 must be 1. Instruction 073*i*01 must be used to read the contents of this register.

If the CPU is in maintenance mode and IMI mode, instruction 073*i*75 executes normally. Following execution, it is trapped like other IMI-mode instructions. However, as explained above, instruction 073*i*01 must be executed first to test bit 0 of register SR0. Because instruction 073*i*01 is trapped (causing an exchange sequence) when IMI mode is active, instruction 073*i*75 should not be used when IMI mode is active.

Hold-issue Conditions

Instructions 072*i*00, 072*i*02, 072*ij*3, 073*i*02, and 073*ij*3:

- Register S*i* reserved.
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 072*ij*6 and 073*ij*6:

- Register S*i* reserved.
- Register A*j* reserved (except A0).
- Shared path reserved.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

Instructions 073*i*00 and 073*i*10:

- Register S*i* reserved.
- VM register busy.

Instructions 073*i*20 and 073*i*30:

- Register A*i* reserved.
- VM register busy.

Instructions 073*ij*1, 073*i*05, 073*i*25, and 073*i*75:

- Register S*i* reserved.
- Shared path reserved.
- Any floating-point functional unit busy.
- Any memory port busy.
- Any of the following instructions issued in the previous 5 CPs: 0013, 0014 (except 0014*j*0), 0015, 0017 through 0026, 023*ij*6, 023*ij*7, 027*ij*2, 027*ij*3, 073*ij*1, or 073*ij*5.

The following instructions cause the VM register to be busy for the following intervals:

- Instruction 175: VL/2 + 8 CPs.
- Instructions 070*ij*1, 146, 147, 005400 153*ij*0, or 005400 153*ij*1: 3 CPs.

Execution Time

Instructions 072*i*00, 072*i*02, 072*ij*3, 072*ij*6, and 073*ij*1:

- Instruction issue: 1 CP.
- Register S*i* ready: CPs undetermined.
- Shared path ready: CPs undetermined.

Instructions 073*i*00 and 073*i*10:

- Instruction issue: 1 CP.
- Register S*i* ready: 3 CPs.

Instructions 073*i*20 and 073*i*30:

- Instruction issue: 1 CP.
- Register A*i* ready: 3 CPs.

Instructions 073*i*02, 073*ij*3, 073*i*05, 073*i*25, 073*i*75, and 073*ij*6:

- Instruction issue: 1 CP.
- Shared path ready: CPs undetermined.

Description

Instruction 072

Instruction 072*i*00 transmits the contents of the real-time clock (RTC) to register S*i*.

Instruction 072*i*02 transmits the contents of all 64 semaphore (SM) bits to register S*i*.

Instruction 072*ij*3 transmits the contents of register ST*j* to register S*i*.

Instruction 072*ij*6 transmits to register S*i* the contents of the ST register designated by register A*j*.

Instructions 073*i*0, 073*i*10, 073*i*20, and 073*i*30

Instructions 073*i*00 and 073*i*10 transmit the contents of the lower and upper vector mask registers (VM0 and VM1), respectively, to register S*i*.

Instructions 073*i*20 and 073*i*30 transmit the contents of the lower and upper vector mask registers (VM0 and VM1), respectively, to register A*i*. These instructions are privileged to Triton mode.

Instruction 073*ij*1

Instruction 073*ij*1 transmits the contents of status register *j* (SR*j*) to register S*i*.

Reading the status registers returns the following values:

SR0: General status information:

- Bit 63: Cluster number not equal to 0 (CLN ≠ 0)
- Bit 57: Bit matrix loaded (BML)
- Bit 52: Interrupt on breakpoint (IBP) mode
- Bit 51: Floating-point error (FPE) flag
- Bit 50: Interrupt on floating-point error (IFP) mode
- Bit 49: Interrupt on operand range error (IOR) mode
- Bit 48: Bidirectional memory mode (BDM)
- Bit 47: Performance monitor busy (PMBY)
- Bits 40 through 46: Processor number
- Bits 32 through 39: Cluster number
- Bit 0: CPU maintenance access available (monitor mode × maintenance mode × SR7 not busy)

SR1: Not used.

SR2: Bits 0 through 47: performance monitor counter 0 through 17. (The performance monitor pointer advances automatically after reading SR2 or SR3). Reading this register is privileged to monitor mode. The PMBY bit should be clear.

SR3: Bits 0 through 47: performance monitor counter 20 through 37. (The performance monitor pointer advances automatically after reading SR2 or SR3). Reading this register is privileged to monitor mode. The PMBY bit should be clear.

SR4: Common memory error information:

- Bit 47: Uncorrectable memory error (MEU) flag
- Bit 46: Correctable memory error (MEC) flag
- Bits 32 through 45: Destination code

Reading this register is privileged to monitor mode.

SR5: Bits 32 through 43: common memory error syndrome. Reading this register is privileged to monitor mode.

SR6: Bits 32 through 44: common memory error address. Reading this register is privileged to monitor mode.

NOTE: Because the hardware writes data to registers SR4 through SR6 as a group, all three registers should be read as a group. (The order in which the registers are read is not important.) When a memory error occurs, the hardware writes the error data to registers SR4 through SR6. The data remains in these registers until all three registers are read. Then all three registers are cleared to 0 and are ready to receive another error.

If a second memory error occurs before the first memory error has been read, the data from the second error is held in a status register buffer until SR4 through SR6 (containing the first error data) are read. As soon the third register is read, the data from the second error is immediately loaded into SR4 through SR6.

If a third (fourth, etc.) memory error occurs, the data is held in a memory section buffer until space is available for it in the status register buffer. It requires about 60 CPs to transfer the data from the memory section buffer to the status register buffer. If a third read sequence occurs during this 60-CP interval, the data read is all 0's.

SR7: Miscellaneous error information:

- Bits 55 through 61: Logical Address Translation (LAT) multiple-hit error flags: ports D, C', C, B', B, A', A
- Bits 48 through 54: LAT miss error flags: ports D, C', C, B', B, A', A
- Bit 47: Register parity error (RPE) flag

- Bit 46: Shared register read error (SRRE) flag
- Bits 32 through 43: Register parity error chip number
- Bits 24 through 31: Shared register read error chip number

Reading this register is privileged to monitor mode.

NOTE: Register SR7 is automatically cleared to 0 after it is read. The following errors are captured if they occur after SR7 is cleared: the first LAT error, the first register parity error, and the first shared register read error. Errors are buffered in a manner similar to SR4 through SR6 (described previously) except that the transfer interval is much shorter (10 CPs).

Instructions 073*i*2, 073*ij*3, 073*i*5, 073*i*25, 073*i*75, and 073*ij*6

Instruction 073*i*02 transmits the entire 64 bits of register S*i* to the 64 semaphore (SM) bits.

Instruction 073*ij*3 transmits the contents of register S*i* to register ST*j*.

Instruction 073*i*05 transmits the contents of register S*i* to status register 0 (SR0). Only bits 48 through 52 of register SR0 are loaded; the remaining bits are read only.

- Bit 52: Interrupt on breakpoint (IBP) mode
- Bit 51: Floating-point error (FPE) flag
- Bit 50: Interrupt on floating-point error (IFP) mode
- Bit 49: Interrupt on operand range error (IOR) mode
- Bit 48: Bidirectional memory mode (BDM)

Instruction 073*i*25 advances the performance monitor pointer. It also causes the PMBY bit (status register 0 bit 47) to set and remain set. (The only way to clear the PMBY bit in this case is with instruction 001501.) This instruction is privileged to maintenance mode.

Instruction 073*i*75 transmits the contents of register S*i* to the maintenance channel. Before writing to this register, SR0 bit 0 must be 1. Refer to Section 4.4 of PRN-0957, *Triton Maintenance System*, for more information.

Instruction 073*ij*6 transmits the contents of register S*i* to the ST register designated by register A*j*.

Instructions 074 and 075

Machine Instruction	CAL Syntax		Description
074*ijk*	S*i*	T*jk*	Transmit (T*jk*) to S*i*.
075*ijk*	T*jk*	S*i*	Transmit (S*i*) to T*jk*.

Special Cases

None.

Hold-issue Conditions

Instruction 074:

- Register S*i* reserved.
- Instruction 036 or 037 in progress.
- Instruction 075 issued in the preceding 1 CP.

Instruction 075:

- Register S*i* reserved.
- Instruction 036 or 037 in progress.

Execution Time

Instruction 074:

- Instruction issue: 1 CP.
- Register S*i* ready: 3 CPs.

Instruction 075:

- Instruction issue: 1 CP.
- Register T*ij* ready: 2 CPs.

Description

Instruction 074 transmits the contents of register T*jk* to register S*i*.

Instruction 075 transmits the contents of register S*i* to register T*jk*.

Instructions 076 and 077

Machine Instruction	CAL Syntax		Description
076*ijk*	S*i*	V*j*,A*k*	Transmit (V*j* element (A*k*)) to S*i*.
077*ijk*	V*i*,A*k*	S*j*	Transmit (S*j*) to V*i* element (A*k*).
077*i*0*k*	V*i*,A*k*	0[S]	Clear register V*i* element (A*k*)

S Special CAL syntax.

Special Cases

For instruction 076*ijk*: If $k = 0$, the content of register V*j* element 1 (second element) is transmitted to register S*i*.

For instruction 077*ijk*:

- If $j = 0$ and $k = 0$, register V*i* element 1 (second element) is cleared to 0.
- If $j = 0$ and $k \neq 0$, register V*i* element (A*k*) is cleared to 0.
- If $j \neq 0$ and $k = 0$, the content of register S*j* is transmitted to register V*k* element 1 (second element).

Hold-issue Conditions

Instruction 076*ijk*:

- Register A*k* reserved (except A0).
- Register V*j* reserved as operand or result.
- Register S*i* reserved.

Instruction 077*ijk*:

- Register A*k* reserved (except A0).
- Register S*j* reserved.
- Register V*i* reserved as operand or result.

Execution Time

Instruction 076*ijk*:

- Instruction issue: 1 CP.
- Vector issue ready: 4 CPs.
- Register S*i* ready: 8 CPs.

Instruction 077*ijk*:

- Instruction issue: 1 CP.
- Vector issue ready: 4 CPs.
- Register V*i* ready: 4 CPs.

Description

Instructions 076*ijk* and 077*ijk* transmit data between an S register and an element of a V register. Bits 0 through 6 of register A*k* designate the element; the remaining bits of register A*k* are not used.

Instruction 076*ijk* transmits the contents of the designated element of register V*j* to register S*i*.

Instruction 077*ijk* transmits the contents of register S*j* to the designated element of register V*i*.

Instructions 10h through 13h

Machine Instruction	CAL Syntax		Description
10*hi*00*nm*[D]	A*i*	*exp*,A*h*	Load A*i* from ((A*h*) + *exp*).
10*hi*20*nm*[ND]	A*i*	*exp*,A*h*,BC	Load A*i* from ((A*h*) + *exp*) bypassing data cache and invalidating cache line.
10*hi*40*pnm*[NT]	A*i*	*exp*,A*h*	Load A*i* from ((A*h*) + *exp*).
10*hi*60*pnm*[NT]	A*i*	*exp*,A*h*,BC	Load A*i* from ((A*h*) + *exp*) bypassing data cache and invalidating cache line.
11*hi*00*nm*[D]	*exp*,A*h*	A*i*	Store (A*i*) to ((A*h*) + *exp*).
11*hi*40*pnm*[NT]	*exp*,A*h*	A*i*	Store (A*i*) to ((A*h*) + *exp*).
12*hi*00*nm*	S*i*	*exp*,A*h*	Load S*i* from ((A*h*) + *exp*).
12*hi*20*nm*[N]	S*i*	*exp*,A*h*,BC	Load S*i* from ((A*h*) + *exp*) bypassing data cache and invalidating cache line.
12*hi*40*pnm*[NT]	S*i*	*exp*,A*h*	Load S*i* from ((A*h*) + *exp*).
12*hi*60*pnm*[NT]	S*i*	*exp*,A*h*,BC	Load S*i* from ((A*h*) + *exp*) bypassing data cache and invalidating cache line.
13*hi*00*nm*	*exp*,A*h*	S*i*	Store (S*i*) to ((A*h*) + *exp*).
13*hi*40*pnm*[NT]	*exp*,A*h*	S*i*	Store (S*i*) to ((A*h*) + *exp*).

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
D Difference in operation between Triton mode and C90 mode.

Special Syntax Note

In any instruction 10*h* through 13*h*, the syntax *exp*,A*h* can be replaced by any one of three special syntax forms. These special forms are not included in the above table because of space constraints.

- The syntax *exp*,0 or *exp*, is equivalent to *exp*,A0. The common memory address is *exp*.
- The syntax ,A*h* is equivalent to 0,A*h*. The common memory address is (A*h*).

Special Cases

If *h* = 0, then the referenced common memory address is *exp*.

Instructions 10*hi*40*pnm*, 10*hi*60*pnm*, 11*hi*40*pnm*, 12*hi*40*nm*, 12*hi*60*pnm*, and 13*hi*40*pnm* are privileged to Triton mode. If these instructions are executed in C90 mode, the results are undefined.

Hold-issue Conditions

Instructions 10*hi*00*nm* and 10*hi*40*pnm*:

- Register A*h* reserved (except A0).
- Register A*i* reserved.
- If the second, third, or fourth parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instructions 10*hi*20*nm* and 10*hi*60*pnm*:

- Register A*h* reserved (except A0).
- Register A*i* reserved.
- Waiting for Memory Resume Signal.
- If the second, third, or fourth parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instructions 11*hi*00*nm* and 11*hi*40*pnm*:

- Register A*h* reserved (except A0).
- Register A*i* reserved.
- Cache miss in progress.
- If the second, third, or fourth parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instructions 12*hi*00*nm* and 12*hi*40*pnm*:

- Register A*h* reserved (except A0).
- Register S*i* reserved.
- If the second, third, or fourth parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instructions 12*hi*20*nm* and 12*hi*40*pnm*:

- Register A*h* reserved (except A0).
- Register S*i* reserved.
- Waiting for Memory Resume Signal.
- If the second, third, or fourth parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Instructions 13*hi*00*nm* and 13*hi*00*pnm*:

- Register A*h* reserved (except A0).
- Register S*i* reserved.
- Cache miss in progress.
- If the second, third, or fourth parcel is not in a buffer, wait for the instruction prefetch sequence to complete.

Execution Time

- Instruction issue: 3 CPs for 3-parcel instructions, 4 CPs for 4-parcel instructions.
- Instructions 10*hi*00*nm* and 10*hi*40*pnm*: If cache hit, register A*i* ready in CPs undetermined. If cache miss, register A*i* ready in CPs undetermined.
- Instructions 10*hi*20*nm* and 10*hi*60*pnm*: register A*i* ready in CPs undetermined.
- Instructions 12*hi*00*nm* and 12*hi*40*pnm*: If cache hit, register S*i* ready in CPs undetermined. If cache miss, register S*i* ready in CPs undetermined.
- Instructions 12*hi*20*nm* and 12*hi*60*pnm*: register S*i* ready in CPs undetermined.

Description

Instructions 10*h* through 13*h* transfer data between common memory (or the data cache) and an A or S register.

Instruction 10*h* transfers data from common memory to register A*i*. In C90 mode, bits 0 through 31 of the common memory word are transferred; bits 32 through 63 are ignored. In Triton mode, all 64 bits are transferred.

Instruction 11*h* transfers data from register A*i* to common memory. In C90 mode, data is transferred to bits 0 through 31 of the common memory; bits 32 through 63 are cleared to 0's. In Triton mode, all 64 bits are transferred.

Instruction 12*h* transfers data from common memory to register S*i*.

Instruction 13*h* transfers data from register S*i* to common memory.

The common memory address that is referenced is determined by adding the contents of register A*i* to the value of *exp*, an instruction field that begins in the second parcel of the instruction. The length of *exp* is determined by bit 2 of the *j* field. If *j* bit 2 is a 0, *exp* is a 2-parcel (32-bit) field. If it is a 1, *exp* is a 3-parcel (48-bit) field.

In C90 mode, *exp* must be a 2-parcel field. The 32-bit contents of register A*h* are added to the 32-bit *exp* to produce a 32-bit logical memory address.

In Triton mode, *exp* can be a 2- or 3-parcel field. If *exp* is 2 parcels, bits 0 through 39 of register A*h* are added to the 32-bit *exp* to produce a 40-bit logical memory address. (*exp* is right aligned and zero-filled for the addition.) Bits 40 through 63 of register A*h* are not used.

If *exp* is a 3-parcel field, bits 0 through 39 of register A*h* are added to bits 0 through 39 of *exp* to produce a 40-bit logical memory address. Bits 40 through 63 of register A*h* are not used.

For instructions 10*h* and 12*h*, *j* field bit 1 determines whether the data cache is used. If this bit is a 0, data is taken from the cache if possible. If this bit is a 1, the cache is bypassed; if the cache contains a copy of the requested data, the corresponding cache line is invalidated and no new line is requested.

Instructions 140 through 145

Machine Instruction	CAL Syntax		Description
140*ijk*	V*i*	S*j*&V*k*	Transmit logical products of (S*j*) and (V*k* elements) to V*i* elements.
141*ijk*	V*i*	V*j*&V*k*	Transmit logical products of (V*j* elements) and (V*k* elements) to V*i* elements.
142*ijk*	V*i*	S*j*!V*k*	Transmit logical sums of (S*j*) and (V*k* elements) to V*i* elements.
142*i*0*k*	V*i*	V*k*[S]	Transmit (V*k* elements) to V*i* elements.
143*ijk*	V*i*	V*j*!V*k*	Transmit logical sums of (V*j* elements) and (V*k* elements) to V*i* elements.
144*ijk*	V*i*	S*j*|V*k*	Transmit logical differences of (S*j*) and (V*k* elements) to V*i* elements.
145*ijk*	V*i*	V*j*|V*k*	Transmit logical differences of (V*j* elements) and (V*k* elements) to V*i* elements.
145*ijk*	V*i*	0[S]	Clear V*i* elements.

S Special CAL syntax.

Special Cases

If $j = 0$, then (Sj) = 0.

Hold-issue Conditions

Instructions 140, 142, and 144:

- Register S*j* reserved.
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Second vector logical functional unit busy or disabled and full vector logical functional unit busy.
  - Instructions 140 through 145 cause the second vector logical functional unit (if available) or the full vector logical functional unit to be busy for VL/2 + 5 CPs.
  - Instructions 146 and 147 cause the full vector logical functional unit to be busy for VL/2 + 5 CPs.

    - Instruction 175 causes the full vector logical functional unit to be busy for VL/2 + 6 CPs.

- Instructions 141, 143, and 145:

    - Register V*j* reserved as operand.

    - Register V*k* reserved as operand.

    - Register V*i* reserved as operand or result.

    - Instruction 076 or 077 issued in the preceding 2 CPs.

    - Second vector logical functional unit busy or disabled and full vector logical functional unit busy.

- Instructions 140 through 145 cause the second vector logical functional unit (if available) or the full vector logical functional unit to be busy for VL/2 + 5 CPs.

    - Instructions 146 and 147 cause the full vector logical functional unit to be busy for VL/2 + 5 CPs.

    - Instruction 175 causes the full vector logical functional unit to be busy for VL/2 + 6 CPs.

Execution Time

Instructions 140, 142, and 144:

- Instruction issue: 1 CP.

- Register V*k* ready: VL/2 + 4 CPs (if no delays encountered).

- Second vector logical or full vector logical functional unit ready: VL/2 + 5 CPs (if no delays encountered).

- Register V*i* ready: VL/2 + 10 CPs (if no delays encountered).

Instructions 141, 143, and 145:

- Instruction issue: 1 CP.

- Registers V*j* and V*k* ready: VL/2 + 4 CPs (if no delays encountered).

- Second vector logical or full vector logical functional unit ready: VL/2 + 5 CPs (if no delays encountered).

- Register V*i* ready: VL/2 + 10 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 140 through 145 use the second vector logical unit if it is enabled and free. If it is disabled or busy, these instructions use the full vector logical functional unit. The contents of the VL register determine the number of operations performed by these instructions. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instructions 140 through 145 use register V*k* as one of the operands. Instructions 140, 142, and 144 use register S*j* as the second operand. (The functional unit holds a copy of register S*j*. Therefore, a subsequent instruction can change register S*j* immediately without affecting the vector instruction.) Instructions 141, 143, and 145 use register V*j* as the second operand. All results are delivered to register V*i*.

Instructions 140 and 141

Instruction 140 transmits the logical products (AND) of register S*j* and register V*k* elements to register V*i* elements.

Instruction 141 transmits the logical products (AND) of register V*j* elements and register V*k* elements to register V*i* elements.

A bit of an element of register V*i* is set to 1 only if the corresponding bits of both operands are 1's, as shown in the following example:

S*j* or V*j* element	1 1 0 0
V*k* element	1 0 1 0
V*i* element	1 0 0 0

Instructions 142 and 143

Instruction 142 transmits the logical sums (OR) of register S*j* and register V*k* elements to register V*i* elements.

Instruction 143 transmits the logical sums (OR) of register V*j* elements and register V*k* elements to register V*i* elements.

A bit of an element of register V*i* is set to 1 if the corresponding bit of either operand is 1, as shown in the following example:

S*j* or V*j* element	1 1 0 0
V*k* element	1 0 1 0
V*i* element	1 1 1 0

Instructions 144 and 145

Instruction 144 transmits the logical differences (exclusive OR) of register S*j* and register V*k* elements to register V*i* elements.

Instruction 145 transmits the logical differences (exclusive OR) of register V*j* elements and register V*k* elements to register V*i* elements.

A bit of an element of register V*i* is set to 1 if the corresponding bits of the operands are different, as shown in the following example:

S*j* or V*j* element	1 1 0 0
V*k* element	1 0 1 0
V*i* element	0 1 1 0

Instructions 146 and 147

Machine Instruction	CAL Syntax		Description
146*ijk*	V*i*	S*j*!V*k*&VM	Merge (S*j*) and (V*k* elements) to V*i* elements using (VM) as mask.
146*i*0*k*	V*i*	#VM&V*k*[S]	Merge 0 and (V*k* elements) to V*i* elements using (VM) as mask.
147*ijk*	V*i*	V*j*!V*k*&VM	Merge (V*j* elements) and (V*k* elements) to V*i* elements using (VM) as mask.

S Special CAL syntax.

Special Cases

If $j = 0$, then (Sj) = 0.

Hold-issue Conditions

Instruction 146:

- Register S*j* reserved.
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- VM register busy. The following instructions cause this register to be busy for the following intervals:
  - Instructions 070*ij*1 and 005400 153: 3 CPs.
  - Instruction 175: VL/2 + 8 CPs.
- Full vector logical functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 140 through 145 (if second vector logical functional unit unavailable): VL/2 + 5 CPs.
  - Instructions 146 and 147: VL/2 + 5 CPs.
  - Instruction 175: VL/2 + 6 CPs.

Instruction 147:

- Register V*j* reserved as operand.
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- VM register busy. The following instructions cause this register to be busy for the following intervals:
  - Instructions 070*ij*1 and 005400 153: 3 CPs.
  - Instruction 175: VL/2 + 8 CPs.
- Full vector logical functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 140 through 145 (if second vector logical functional unit unavailable): VL/2 + 5 CPs.
  - Instructions 146 and 147: VL/2 + 5 CPs.
  - Instruction 175: VL/2 + 6 CPs.

Execution Time

Instruction 146:

- Instruction issue: 1 CP.
- VM register ready: 4 CPs.
- Register V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Full vector logical functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 10 CPs (if no delays encountered).

Instruction 147:

- Instruction issue: 1 CP.
- VM register ready: 4 CPs.
- Registers V*j* and V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Full vector logical functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 10 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 146 and 147 use the full vector logical functional unit. The contents of the VL register determine the number of operations performed by these instructions. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instructions 146 and 147 use register V*k* as one of the operands. Instruction 146 uses register S*j* as the second operand. (The functional unit holds a copy of register S*j*. Therefore, a subsequent instruction can change register S*j* immediately without affecting the vector instruction.) Instruction 147 uses register V*j* as the second operand. All results are delivered to register V*i*.

Instruction 146 merges register S*j* with register V*k* and transmits the result to register V*i*. Each element of the result is taken from either register S*j* or the corresponding element of register V*k*, depending on the state of the corresponding bit in the vector mask (VM) register. If the VM bit is 1, register S*j* data is used; if the VM bit is 0, register V*k* data is used.

Example:

VL	=	4
VM	=	0110 … 0_2
S2	=	7
V6 element 0	=	1
V6 element 1	=	2
V6 element 2	=	3
V6 element 3	=	4

The results of executing instruction 146726 (V7 S2!V6&VM) are:

V7 element 0	=	1
V7 element 1	=	7
V7 element 2	=	7
V7 element 3	=	4
V7 elements 4 – 127	=	unchanged

Instruction 147 merges register V*j* with register V*k* and transmits the result to register V*i*. Each element of the result is taken from the corresponding element of register V*j* or the corresponding element of register V*k*, depending on the state of the corresponding bit in the vector mask (VM) register. If the VM bit is 1, register V*j* data is used; if the VM bit is 0, register V*k* data is used.

Example:

VL	=	4
VM	=	0110 … 0_2
S2	=	7
V2 element 0	=	1
V2 element 1	=	2
V2 element 2	=	3
V2 element 3	=	4
V3 element 0	=	5
V3 element 1	=	6
V3 element 2	=	7
V3 element 3	=	8

The results of executing instruction 147123 (V1 V2!V3&VM) are:

V1 element 0	=	5
V1 element 1	=	2
V1 element 2	=	3
V1 element 3	=	7
V1 elements 4 – 127	=	unchanged

Instructions 150 and 151

Machine	Instruction	CAL Syntax		Description
	150*ijk*[D]	V*i*	V*j*<A*k*	Shift (V*j* elements) left (A*k*) places to V*i* elements.
	150*ij*0[D]	V*i*	V*j*<1[S]	Shift (V*j* elements) left one place to V*i* elements.
005400	150*ij*0	V*i*	V*j*<V0	Shift (V*j* elements) left (V0 elements) places to V*i* elements.
	151*ijk*[D]	V*i*	V*j*>A*k*	Shift (V*j* elements) right (A*k*) places to V*i* elements.
	151*ij*0[D]	V*i*	V*j*>1[S]	Shift (V*j* elements) right one place to V*i* elements.
005400	151*ij*0	V*i*	V*j*>V0	Shift (V*j* elements) right (V0 elements) places to V*i* elements.

D Difference in operation between Triton mode and C90 mode.
S Special CAL syntax.

Special Cases

If $k = 0$, then (Ak) = 1.

Hold-issue Conditions

Unprefixed instructions:

- Register V*j* reserved as operand.
- Register A*k* reserved (except A0).
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Vector shift functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 150 through 153 and 005400 150 through 005400 152: VL/2 + 5 CPs.
  - Instruction 005400 153: VL/2 + 9 CPs.

005400-prefixed instructions:

- Register V*j* or V0 reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Vector shift functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 150 through 153 and 005400 150 through 005400 152: VL/2 + 5 CPs.
  - Instruction 005400 153: VL/2 + 9 CPs.

Execution Time

Unprefixed instructions:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).
- Vector shift functional unit ready: VL/2 + 4 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

005400-prefixed instructions:

- Instruction issue: 2 CPs.
- Registers V*j* and V0 ready: VL/2 + 4 CPs (if no delays encountered).
- Vector shift functional unit ready: VL/2 + 4 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 150 and 151 (unprefixed and 005400) use the vector shift functional unit. All shifts are end-off with zero fill. A result element is cleared to zero if the shift counts exceeds 63.

The number of operations performed by instructions 150 and 151 is determined by the contents of the VL register. Operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instruction 150 shifts elements of register V*j* left by the number of places specified by register A*k* and transmits the results to elements of register V*i*. In C90 mode, if bits 6 through 31 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's. In Triton mode, if bits 6 through 63 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's.

Instruction 151 shifts elements of register V*j* right by the number of places specified by register A*k* and transmits the results to elements of register V*i*. In C90 mode, if bits 6 through 31 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's. In Triton mode, if bits 6 through 63 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's.

Instruction 005400 150 shifts elements of register V*j* left by the number of places specified by elements of register V*k* and transmits the results to elements of register V*i*. If bits 6 through 63 of an element of register V*k* are nonzero, the corresponding elements of register V*i* are cleared to 0's.

Instruction 005400 151 shifts elements of register V*j* right by the number of places specified by elements of register V*k* and transmits the results to elements of register V*i*. If bits 6 through 63 of an element of register V*k* are nonzero, the corresponding element of register V*i* is cleared to 0.

Unprefixed Instructions 152 and 153

Machine Instruction	CAL Syntax		Description
152*ijk*	V*i*	V*j*,V*j*<A*k*	Double shift (V*j* elements) left (A*k*) places to V*i* elements.
152*ij*0	V*i*	V*j*,V*j*<1[S]	Double shift (V*j* elements) left one place to V*i* elements.
153*ijk*	V*i*	V*j*,V*j*>A*k*	Double shift (V*j* elements) right (A*k*) places to V*i* elements.
153*ij*0	V*i*	V*j*,V*j*>1[S]	Double shift (V*j* elements) right one place to V*i* elements.

S Special CAL syntax

Special Cases

If *k* = 0, then (A*k*) = 1.

Hold-issue Conditions

Register V*j* reserved as operand.

Register A*k* reserved (except A0).

Register V*i* reserved as operand or result.

Instruction 076 or 077 issued in the preceding 2 CPs.

Vector shift functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:

- Instructions 150 through 153 and 005400 152: VL/2 + 5 CPs.
- Instruction 005400 153: VL/2 + 9 CPs.

Execution Time

Instruction 152:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).

- Vector shift functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 13 CPs (if no delays encountered).

Instruction 153:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).
- Vector shift functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 152 and 153 use the vector shift functional unit. All shifts are end-off with zero fill. A result element is cleared to zero if the shift count exceeds 127.

The contents of the VL register determine the number of operations performed by instructions 152 and 153. Operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instruction 152

Instruction 152 repeatedly concatenates two successive elements of register V*j*, shifts the result left by the number of places specified by register A*k*, and transmits the result to an element of register V*i*. In C90 mode, if bits 6 through 31 of register A*k* are nonzero, the first VL elements of register are cleared to 0. In Triton mode, if bits 6 through 63 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's.

The successive operations of instruction 152 work as follows. The first operation concatenates elements 0 and 1 of register V*j* (element 0 provides the most significant 64 bits, element 1 the least significant 64 bits), shifts

the result left, and transmits the most significant 64 bits to element 0 of register V*i*. The second operation concatenates elements 1 and 2 of register V*j*, shifts the result left, and transmits the upper bits to element 1 of register V*i*. This process continues for a total of VL operations. The next-to-last operation concatenates elements VL –2 and VL –1 of register V*j* , shifts the result left, and transmits the upper bits to element VL –2 of register V*i*. The last operation concatenates element VL –1 of register V*j* with a word of all 0's, shifts the result left, and transmits the upper bits to element VL –1 of register V*i*.

NOTE: If the shift count is less than or equal to 64, the function of instruction 152 can be viewed as follows. Consider each V register to be a single, large word consisting of 64 × VL bits. Instruction 152 simply shifts register V*j* left between 0 and 64 bits (with zero fill) and transmits the result to register V*i*.

Example:

VL	=	4					
A1	=	3					
V4 element 0	=	0	00000	0000	0000	0000	0007_8
V4 element 1	=	0	60000	0000	0000	0000	0005_8
V4 element 2	=	1	00000	0000	0000	0000	0006_8
V4 element 3	=	1	60000	0000	0000	0000	0007_8

The results of executing instruction 152541 (V5 V4,V4<A1) are:

V5 element 0	=	0	00000	0000	0000	0000	0073_8
V5 element 1	=	0	00000	0000	0000	0000	0054_8
V5 element 2	=	0	00000	0000	0000	0000	0067_8
V5 element 3	=	0	00000	0000	0000	0000	0070_8
V5 elements 4 – 127	=	unchanged					

Instruction 153

Instruction 153 repeatedly concatenates two successive elements of register V*j*, shifts the result right by the number of places specified by register A*k*, and transmits the result to an element of register V*i*. In C90 mode, if bits 6 through 31 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's. In Triton mode, if bits 6 through 63 of register A*k* are nonzero, the first VL elements of register V*i* are cleared to 0's.

The successive operations of instruction 153 are as follows: The first operation concatenates a word of all 0's with element 0 of register V*j* (the word of all 0's provides the most significant 64 bits, element 0 the least significant 64 bits), shifts the result right, and transmits the least significant 64 bits to element 0 of register V*i*. The second operation

concatenates elements 0 and 1 of register V*j*, shifts the result right, and transmits the lower bits to element 1 of register V*i*. This process continues for a total of VL operations. The next-to-last operation concatenates elements VL –3 and VL –2 of register V*j*, shifts the result right, and transmits the lower bits to element VL –2 of register V*i*. The last operation concatenates element VL –2 and VL –1 of register V*j*, shifts the result right, and transmits the lower bits to element VL –1 of register V*i*.

NOTE: If the shift count is less than or equal to 64, the function of instruction 153 can be viewed as follows. Consider each V register to be a single, large word consisting of 64 × VL bits. Instruction 153 simply shifts register V*j* right between 0 and 64 bits (with zero fill) and transmits the result to register V*i*.

Example:

VL	=	4					
A6	=	3					
V2 element 0	=	0	00000	0000	0000	0000	0017_8
V2 element 1	=	0	60000	0000	0000	0000	0006_8
V2 element 2	=	1	00000	0000	0000	0000	0006_8
V2 element 3	=	1	60000	0000	0000	0000	0007_8

The results of executing instruction 153026 (V0 V2,V2>A6) are:

V0 element 0	=	0	00000	0000	0000	0000	0001_8
V0 element 1	=	1	66000	0000	0000	0000	0000_8
V0 element 2	=	1	50000	0000	0000	0000	0000_8
V0 element 3	=	1	56000	0000	0000	0000	0000_8
V0 elements 4 – 127	=	unchanged					

005400-Prefixed Instructions 152 and 153

Machine Instruction	CAL Syntax		Description
005400 152*ijk*	V*i*	V*j*,A*k*	Transfer (V*j* elements) starting at element (A*k*) to V*i* elements.
005400 153*ij*0[NT]	V*i*	V*j*,[VM]	Compress V*j* by (VM) to V*i*.
005400 153*ij*1[NT]	V*i*,[VM]	V*j*	Expand V*j* by (VM) to V*i*.

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.

Special Cases

Instructions 005400 153*ij*0 and 005400 153*ij*1 are privileged to Triton mode. If these instructions are executed in C90 mode, the results are undefined.

For instruction 005400 152, if $k = 0$, then (A*k*) = 1.

For instruction 005400 153*ij*1, if $i = j$, then the instruction operates as a no-op.

Instruction 005400 153*ij*1 does not support vector chaining to subsequent instructions. That is, if the result register of a 005400 153*ij*1 instruction is the same as an operand register of a subsequent instruction, the subsequent instruction does not begin processing elements until the 005400 153*ij*1 instruction has processed all elements.

Hold-issue Conditions

Instruction 005400 152:

- Register V*j* reserved as operand.
- Register A*k* reserved (except A0).
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Vector shift functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 150 through 153 and 005400 152: VL/2 + 5 CPs.
  - Instruction 005400 153: VL/2 + 9 CPs.

Instructions 005400 153*ij*0 and 005400 153*ij*1:

- Register V*j* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- VM register busy. The following instructions cause this register to be busy for the following intervals:
  - Instructions 070*ij*1, 146, 147, and 005400 153: 3 CPs
  - Instruction 175: VL/2 + 8 CPs.
- Vector shift functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 150 through 153 and 005400 152: VL/2 + 5 CPs
  - Instruction 005400 153: VL/2 + 9 CPs.

Execution Time

NOTE: For instructions 005400 153*ij*0 and 005400 153*ij*1, X is the number of bits in VM including and to the left of the right-most 1 bit.

Instruction 005400 152:

- Instruction issue: 2 CPs.
- Register V*j* ready: VL/2 + 5 CPs (if no delays encountered).
- Vector shift functional unit ready: VL/2 + 6 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 13 CPs (if no delays encountered).

Instruction 005400 153*ij*0:

- Instruction issue: 2 CPs.
- VM register ready: 4 CPs.
- Register V*j* ready: X/2 + 4 CPs (if no delays encountered).

- Vector shift functional unit ready: X/2 + 9 CPs (if no delays encountered).
- Register V*i* ready: X/2 + 13 CPs (if no delays encountered).

Instruction 005400 153*ij*1:

- Instruction issue: 2 CPs.
- VM register ready: 3 CPs.
- Register V*j* ready: X/2 + 8 CPs (if no delays encountered).
- Vector shift functional unit ready: X/2 + 9 CPs (if no delays encountered).
- Register V*i* ready: X/2 + 16 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 005400 152 and 005400 153 use the vector shift functional unit.

Instruction 005400 152*ijk*

Instruction 005400 152*ijk* transmits the contents of elements (A*k*) through VL –1 of register V*j* to elements 0 through VL –1 –(A*k*) of register V*i*.

Instruction 005400 153*ij*

Instruction 005400 153*ij*0 compresses register V*j*, using the VM register as a mask, and transmits the result to register V*i*.

The operation of instruction 005400 153*ij*0 is as follows: First, two element counters (one each for registers V*i* and V*j*) are initialized to zero. The VM register is then scanned from right to left. For each 0 bit in VM, the V*j* element counter is incremented and the scan continues with the next bit. For each 1 bit in VM, an element is transmitted from register V*j* to register V*i* (the element counters determine the position within each register), and the element counters for *both* registers are incremented.

The element counter for register V*i* falls behind the counter for register V*j* by one position for each 0 bit in register VM. If X is the number of 1 bits in VM, only elements 0 through X –1 of register V*i* are written; elements X through 127 are not changed.

Example:

VM	=	1 0010 … 0_2
V2 element 0	=	1
V2 element 1	=	2
V2 element 2	=	3
V2 element 3	=	4

The results of executing instruction 005400 153120 (V1 V2,[VM]) are:

V1 element 0	=	1
V1 element 1	=	4
V1 elements 2 – 127	=	unchanged

Instruction 005400 153*ij*1

Instruction 005400 153*ij*1 expands register V*j*, using the VM register as a mask, and transmits the result to register V*i*.

The operation of instruction 005400 153*ij*1 is as follows: First, two element counters (one each for registers V*i* and V*j*) are initialized to zero. The VM register is then scanned from right to left. For each 0 bit in VM, the V*i* element counter is incremented and the scan continues with the next bit. For each 1 bit in VM, an element is transmitted from register V*j* to register V*i* (the element counters determine the position within each register), and the element counters for *both* registers are incremented.

The element counter for register V*j* falls behind the counter for register V*i* by one position for each 0 bit in register VM. If X is the number of 1 bits in VM, only elements 0 through X –1 of register V*j* are read; elements X through 127 are not used.

Example:

VM	=	10010 … 0_2
V1 element 0	=	1
V1 element 1	=	4

The results of executing instruction 005400 153211 (V2,[VM] V1) are:

V2 element 0	=	1
V2 element 1	=	unchanged
V2 element 2	=	unchanged
V2 element 3	=	4
V1 elements 4 – 127	=	unchanged

Instructions 154 through 157

Machine Instruction	CAL Syntax		Description
154*ijk*	V*i*	S*j*+V*k*	Transmit integer sums of (S*j*) and (V*k* elements) to V*i* elements.
155*ijk*	V*i*	V*j*+V*k*	Transmit integer sums of (V*j* elements) and (V*k* elements) to V*i* elements.
156*ijk*	V*i*	S*j*-V*k*	Transmit integer differences of (S*j*) and (V*k* elements) to V*i* elements.
156*i*0*k*	V*i*	-V*k*[S]	Transmit two's complement of (V*k* elements) to V*i* elements.
157*ijk*	V*i*	V*j*-V*k*	Transmit integer differences of (V*j* elements) and (V*k* elements) to V*i* elements.

S Special CAL syntax.

Special Cases

For instructions 154 and 156: If $j = 0$, then (Sj) = 0.

Hold-issue Conditions

Instructions 154 and 156:

- Register S*j* reserved (except S0).
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Vector add functional unit busy. Instruction 154 through 157 cause this functional unit to be busy for VL/2 + 5 CPs.

Instructions 155 and 157:

- Register V*j* or V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Vector add functional unit busy. Instructions 154 through 157 cause this functional unit to be busy for VL/2 + 5 CPs.

Execution Time

Instructions 154 and 156:

- Instruction issue: 1 CP.
- Register V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Vector add functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 9 CPs (if no delays encountered).

Instructions 155 and 157:

- Instruction issue: 1 CP.
- Registers V*j* and V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Vector add functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 9 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 154 through 157 use the vector add functional unit. This functional unit performs two's complement arithmetic. It does not detect overflow.

The contents of the VL register determine the number of operations performed by these instructions. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instructions 154 through 157 use register V*k* as one of the operands. Instructions 154 and 156 use register S*j* as the second operand. (The functional unit holds a copy of register S*j*. Therefore a subsequent

instruction can change register S*j* immediately without affecting the vector instruction.) Instructions 155 and 157 use register V*j* as the second operand. All results are delivered to register V*i*.

Instruction 154 transmits the integer sums of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 155 transmits the integer sums of register V*j* elements and register V*k* elements to register V*i* elements.

Instruction 156 transmits the integer differences of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 157 transmits the integer differences of register V*j* elements and register V*k* elements to register V*i* elements.

Instructions 160 through 167

Machine Instruction	CAL Syntax		Description
160*ijk*	V*i*	S*j**FV*k*	Transmit floating-point products of (S*j*) and (V*k* elements) to V*i* elements.
161*ijk*	V*i*	V*j**FV*k*	Transmit floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements.
162*ijk*	V*i*	S*j**HV*k*	Transmit half-precision rounded floating-point products of (S*j*) and (V*k* elements) to V*i* elements.
163*ijk*	V*i*	V*j**HV*k*	Transmit half-precision rounded floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements.
164*ijk*	V*i*	S*j**RV*k*	Transmit rounded floating-point products of (S*j*) and (V*k* elements) to V*i* elements.
165*ijk*	V*i*	V*j**RV*k*	Transmit rounded floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements.
166*ijk*[D]	V*i*	S*j**V*k*	Transmit integer products of (S*j*) and (V*k* elements) to V*i* elements.
167*ijk*	V*i*	V*j**IV*k*	Transmit 2 – (V*j* elements) * (V*k* elements) to V*i* elements (reciprocal iteration).

D Difference in operation between Triton mode and C90 mode.

Special Cases

For instructions 160, 162, 164, and 166: If $j = 0$, then (Sj) = 0.

Hold-issue Conditions

Instructions 160, 162, 164, and 166:

- Register S*j* reserved (except S0).
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Floating-point multiply functional unit busy. Instruction 160 through 167 cause this functional unit to be busy for VL/2 + 5 CPs.

Instructions 161, 163, 165, and 167:

- Register V*j* or V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Floating-point multiply functional unit busy. Instructions 160 through 167 cause this functional unit to be busy for VL/2 + 5 CPs.

Execution Time

Instructions 160, 162, 164, and 166:

- Instruction issue: 1 CP.
- Register V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Floating-point multiply functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

Instructions 161, 163, 165, and 167:

- Instruction issue: 1 CP.
- Registers V*j* and V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Floating-point multiply functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 160 through 167 use the floating-point multiply functional unit. Except for instruction 166, this functional unit uses operands that are in floating-point format and returns results in floating-point format. (A result is normalized only if both operands are normalized.) Except for instruction 166, the functional unit detects both underflow and overflow conditions. If underflow occurs, the result returned is a word of all 0's (floating-point 0); no flag is set. If overflow occurs, the result returned is as follows: sign bit = 0, exponent = 60000_8, and coefficient = as calculated. An overflow also sets the floating-point error (FPE) flag in the exchange package.

The contents of the VL register determine the number of operations performed by these instructions. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instructions 160 through 167 use register V*k* as one of the operands. Instructions 160, 162, 164, and 166 use register S*j* as the second operand. (The functional unit holds a copy of register S*j*. Therefore, a subsequent instruction can change register S*j* immediately without affecting the vector instruction.) Instructions 161, 163, 165, and 167 use register V*j* as the second operand. All results are delivered to register V*i*.

Instructions 160 and 161

Instructions 160 and 161 perform floating-point multiplication without rounding.

Instruction 160 transmits the floating-point products of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 161 transmits the integer sums of register V*j* elements and register V*k* elements to register V*i* elements.

Instructions 162 and 163

Instructions 162 and 163 perform half-precision rounded floating-point multiplication. The least significant 19 bits of each result are cleared. These instructions can be used in the division algorithm when only 30 bits of accuracy are required.

Instruction 162 transmits the half-precision rounded floating-point products of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 163 transmits the half-precision rounded floating-point products of register V*j* elements and register V*k* elements to register V*i* elements.

Instructions 164 and 165

Instructions 164 and 165 perform rounded floating-point multiplication. These instructions can be used in the division algorithm.

Instruction 164 transmits the rounded floating-point products of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 165 transmits the rounded floating-point products of register V*j* elements and register V*k* elements to register V*i* elements.

Instruction 166

Instruction166 performs integer multiplication. No overflow conditions are reported.

In C90 mode, instruction 166 transmits 32-bit integer products of register S*j* and elements of register V*k* to elements of register V*i*. The S*j* operand must be left shifted 31 places so that it is in bit positions 31 through 62; bits 0 through 30 must be 0's. The V*k* operands must be left shifted 16 places so that they are in bit positions 16 through 47; bits 0 through 15 must be 0's. The results are transmitted to bits 0 through 31 of the register V*i* elements; bits 32 through 63 are cleared to 0's.

In Triton mode, instruction 166 transmits the 40-bit integer products of register S*j* and elements of register V*k* to elements of register V*i*. Only bits 0 through 39 of the operands are used; bits 40 through 63 are ignored. The result is stored in bit positions 0 through 39 and sign-extended to bits 40 through 63.

Instruction 167

Instruction 167 transmits two minus the floating-point product of register V*j* elements and register V*k* elements to register V*i* elements. This instruction can be used in the reciprocal and division algorithms.

Instructions 170 through 173

Machine Instruction	CAL Syntax		Description
170*ijk*	V*i*	S*j*+FV*k*	Transmit floating-point sums of (S*j*) and (V*k* elements) to V*i* elements.
170*i*0*k*	V*i*	+FV*k*[S]	Transmit normalized (V*k* elements) to V*i* elements.
171*ijk*	V*i*	V*j*+FV*k*	Transmit floating-point sums of (V*j* elements) and (V*k* elements) to V*i* elements.
172*ijk*	V*i*	S*j*-FV*k*	Transmit floating-point differences of (S*j*) and (V*k* elements) to V*i* elements.
172*i*0*k*	V*i*	-FV*k*[S]	Transmit normalized negatives of (V*k* elements) to V*i* elements.
173*ijk*	V*i*	V*j*-FV*k*	Transmit floating-point differences of (V*j* elements) and (V*k* elements) to V*i* elements.

S Special CAL syntax.

Special Cases

For instructions 170 and 172: If $j = 0$, then (Sj) = 0.

Hold-issue Conditions

Instructions 170 and 172:

- Register S*j* reserved (except S0).
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Floating-point add functional unit busy. Instructions 170 through 173 cause this functional unit to be busy for VL/2 + 5 CPs.

Instructions 171 and 173:

- Register V*j* reserved as operand.
- Register V*k* reserved as operand.

- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Floating-point add functional unit busy. Instructions 170 through 173 cause this functional unit to be busy for VL/2 + 5 CPs.

Execution Time

Instructions 154 and 156:

- Instruction issue: 1 CP.
- Register V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Vector add functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

Instructions 155 and 157:

- Instruction issue: 1 CP.
- Registers V*j* and V*k* ready: VL/2 + 4 CPs (if no delays encountered).
- Vector add functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 170 through 173 use the floating-point add functional unit. For this functional unit, operands must be in floating-point format; this functional unit returns results in normalized floating point-format. The functional unit detects both underflow and overflow conditions. If underflow occurs, the result returned is a word of all 0's (floating-point 0); no flag is set. If overflow occurs, the result returned is as follows: sign bit = 0, exponent = 60000_8, and coefficient = as calculated. An overflow also sets the floating-point error (FPE) flag in the exchange package.

The contents of the VL register determine the number of operations performed by instructions 170 through 173. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instructions 170 through 173 use register V*k* as one of the operands. Instructions 170 and 172 use register S*j* as the second operand. (The functional unit holds a copy of register S*j*. Therefore, a subsequent instruction can change register S*j* immediately without affecting the vector instruction.) Instructions 171 and 173 use register V*j* as the second operand. All results are delivered to register V*i*.

Instruction 170 transmits the floating-point sums of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 171 transmits the floating-point sums of register V*j* elements and register V*k* elements to register V*i* elements.

Instruction 172 transmits the floating-point differences of register S*j* and elements of register V*k* to elements of register V*i*.

Instruction 173 transmits the floating-point differences of register V*j* elements and register V*k* elements to register V*i* elements.

Instruction 174*ij*0

Machine Instruction	CAL Syntax		Description
174*ij*0	V*i*	/HV*j*	Transmit floating-point reciprocal approximation of (V*j* elements) to V*i* elements.

Special Cases

The operands must be normalized (bit 47 should be 1). If an operand is not normalized, the result is invalid.

Hold-issue Conditions

Register V*j* reserved as operand.

Register V*i* reserved as operand or result.

Instruction 070*ij*0 issued in the preceding 19 CPs.

Instruction 076 or 077 issued in the preceding 2 CPs.

Floating-point reciprocal approximation functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:

- Instruction 070*ij*1: VL/2 + 10 CPs.
- Instruction 174*ij*0: VL/2 + 12 CPs.
- Instructions 174*ij*1 through 174*ij*3: VL/2 + 5 CPs.

Execution Time

Instruction issue: 1 CP.

Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).

Floating-point reciprocal approximation functional unit ready: VL/2 + 12 CPs (if no delays encountered).

Register V*i* ready: VL/2 + 19 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instruction 174*ij*0 uses the floating-point reciprocal approximation functional unit. This functional unit assumes that operands are in normalized floating-point format and returns results in normalized floating-point format. The functional unit handles overflow and underflow conditions identically. If the exponent of the operand is less than 20002_8 or greater than 60001_8, the result returned is as follows: sign bit = 0, exponent = 60000_8, and coefficient = as calculated. An overflow or underflow also sets the floating-point error (FPE) flag in the exchange package.

The number of operations performed by instruction 174*ij*0 is determined by the contents of the VL register. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instruction 174*ij*0 transmits the floating-point reciprocal approximation of register V*j* elements to register V*i* elements. The result is accurate to 30 significant bits. The lower 18 bits are 0's. The number of significant bits can be increased to 48 by performing a reciprocal iteration (167*ijk*) and multiply (160*ijk*) sequence.

Instructions 174*ij*1 through 174*ij*3

Machine Instruction	CAL Syntax		Description
174*ij*1	V*i*	PV*j*	Transmit population count of (V*j* elements) to V*i* elements.
174*ij*2	V*i*	QV*j*	Transmit population count parity of (V*j* elements) to V*i* elements.
174*ij*3	V*i*	ZV*j*	Transmit leading zero count of (V*j* elements) to V*i* elements.

Special Cases

For instruction 174*ij*3: If (V*j* element) = 0, then (V*i* element) = 64.

Hold-issue Conditions

Register V*j* reserved as operand.

Register V*i* reserved as operand or result.

Instruction 070*ij*0 issued in the preceding 19 CPs.

Instruction 076 or 077 issued in the preceding 2 CPs.

Vector population/parity/leading-zero count functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:

- Instruction 070*ij*1: VL/2 + 10 CPs.
- Instruction 174*ij*0: VL/2 + 12 CPs.
- Instructions 174*ij*1 through 174*ij*3: VL/2 + 5 CPs.

Execution Time

Instruction issue: 1 CP.

Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).

Vector population/parity/leading-zero count functional unit ready: VL/2 + 12 CPs (if no delays encountered).

Register V*i* ready: VL/2 + 9 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 174*ij*1 through 174*ij*3 use the vector population/parity/leading-zero count functional unit. This functional unit shares some logic with the floating-point reciprocal approximation functional unit.

The contents of the VL register determine the number of operations performed by instructions 174*ij*1 through 174*ij*3. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instruction 174*ij*1 counts the number of 1 bits in elements of register V*j* and transmits the results to bits 0 through 6 of the corresponding elements of register V*i*. Bits 7 through 63 of the V*i* elements are cleared to 0's.

Instruction 174*ij*2 counts the number of 1 bits in elements of register V*j*. Bit 0 of each count is transmitted to bit 0 of the corresponding elements of register V*i*. Bits 1 through 63 of the V*i* elements are cleared to 0's.

Instruction 174*ij*3 counts the number of zeroes to the left of the most-significant 1 bit in elements of register V*j* and transmits the results to bits 0 through 6 of the corresponding elements of register V*i*. Bits 7 through 63 of the V*i* elements are cleared to 0's.

Instructions 1740*j*4, 1740*j*5, and 174*ij*6

Machine Instruction	CAL Syntax		Description
1740*j*4	BMM	LV*j*	Transmit V*j* elements 0 – 63 to B matrix.
1740*j*5[N]	BMM	UV*j*	Transmit V*j* elements 64 – 127 to B matrix.
174*ij*6	V*i*	V*j**BT	Transmit bit-matrix product of (V*j*) and (B^T) to V*i*.

N New instruction (not available on CRAY C90 series systems).

Special Cases

None.

Hold-issue Conditions

Instructions 1740*j*4 and 1740*j*5:

- Register V*j* reserved as operand.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Bit-matrix multiply functional unit busy. Instructions 1740*j*4, 1740*j*5, and 174*ij*6 cause this functional unit to be busy for VL/2 + 5 CPs.

Instruction 174*ij*6:

- Register V*j* reserved as operand.
- Register V*i* reserved as an operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- Bit-matrix multiply functional unit busy. Instructions 1740*j*4, 1740*j*5, and 174*ij*6 cause this functional unit to be busy for VL/2 + 5 CPs.

Execution Time

Instructions 1740*j*4 and 1740*j*5:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).
- Bit-matrix multiply functional unit ready: VL/2 + 5 CPs (if no delays encountered).

Instruction 174*ij*6:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).
- Bit-matrix multiply functional unit ready: VL/2 + 5 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 8 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

Instructions 1740*j*4, 1740*j*5, and 174*ij*6 use the bit-matrix multiply functional unit. This functional unit contains a 64 word × 64 bit matrix called B.

Instruction 1740*j*4 transmits elements 0 through 63 of register V*j* to the B matrix. It also sets the Bit Matrix Loaded (BML) bit in register SR0. This bit is saved in the exchange package (status field bit 0).

Instruction 1740*j*5 transmits elements 64 through 127 of register V*j* to the B matrix. It also sets the Bit Matrix Loaded (BML) bit in the status field of the exchange package.

Instruction 174*ij*6 transmits to register V*i* the bit-matrix product of the contents of register V*j* and the contents of B^T (matrix B transposed). B must have been previously loaded with instruction 1740*j*4 or 1740*j*5.

Instruction 175

Machine Instruction	CAL Syntax		Description
1750*j*0	VM	V*j*,Z	Set VM bit if (V*j* element) = 0.
1750*j*1	VM	V*j*,N	Set VM bit if (V*j* element) ≠ 0.
1750*j*2	VM	V*j*,P	Set VM bit if (Vj element) ≥ 0.
1750*j*3	VM	V*j*,M	Set VM bit if (V*j* element) < 0.
175*ij*4	V*i*,VM	V*j*,Z	Set VM bit if (V*j* element) = 0 and store compressed indices of V*j* elements = 0 in V*i*.
175*ij*5	V*i*,VM	V*j*,N	Set VM bit if (V*j* element) ≠ 0 and store compressed indices of V*j* elements ≠ 0 in V*i*.
175*ij*6	V*i*,VM	V*j*,P	Set VM bit if (V*j* element) ≥ 0 and store compressed indices of V*j* elements ≥ 0 in V*i*.
175*ij*7	V*i*,VM	V*j*,M	Set VM bit if (V*j* element) < 0 and store compressed indices of V*j* elements < 0 in V*i*.

Special Cases

None.

Hold-issue Conditions

Instructions 175*ij*0 through 175*ij*3:

- Register V*j* reserved as operand.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- VM register busy. The following instructions cause this register to be busy for the following intervals:
  - Instructions 070*ij*1, 146, 147, and 005400 153: 3 CPs.
  - Instruction 175: VL/2 + 8 CPs.
- Full vector logical functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 140 through 145 (if second vector logical functional unit unavailable): VL/2 + 5 CPs.
  - Instructions 146 and 147: VL/2 + 5 CPs.
  - Instruction 175: VL/2 + 6 CPs.

Instructions 175*ij*4 through 175*ij*7:

- Register V*j* reserved as operand.
- Register V*i* reserved as operand or result.
- Instruction 076 or 077 issued in the preceding 2 CPs.
- VM register busy. The following instructions cause this register to be busy for the following intervals:
  - Instructions 070*ij*1, 146, 147, and 005400 153: 3 CPs.
  - Instruction 175: VL/2 + 8 CPs.
- Full vector logical functional unit busy. The following instructions cause this functional unit to be busy for the following intervals:
  - Instructions 140 through 145 (if second vector logical functional unit unavailable): VL/2 + 5 CPs.
  - Instructions 146 and 147: VL/2 + 5 CPs.
  - Instruction 175: VL/2 + 6 CPs.

Execution Time

Instructions 175*ij*0 through 175*ij*3:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).
- VM register ready: VL/2 + 8 CPs (if no delays encountered).
- Full vector logical functional unit ready: VL/2 + 6 CPs (if no delays encountered).

Instructions 175*ij*4 through 175*ij*7:

- Instruction issue: 1 CP.
- Register V*j* ready: VL/2 + 4 CPs (if no delays encountered).
- VM register ready: VL/2 + 8 CPs (if no delays encountered).

- Full vector logical functional unit ready: VL/2 + 6 CPs (if no delays encountered).
- Register V*i* ready: VL/2 + 12 CPs (if no delays encountered).

NOTE: Vector instructions may or may not start execution immediately and may or may not execute continuously; they execute as data becomes available. In particular, a memory conflict that slows execution of some elements of a vector load can cause delays in all instructions in the operation chain, starting with that load.

Description

The 175 instructions use the full vector logical function unit. The contents of the VL register determine the number of operations performed by these instructions. All operations start with element 0 of the V registers and increment the element number by 1 for each operation performed.

Instructions 175*ij* through 175*ij*3

Instructions 175*ij*0 through 175*ij*3 test the elements of register V*j* for a specified condition. If the condition is true, the corresponding bit of the VM register is set to 1. If the condition is false, the corresponding bit of the VM register is cleared to 0. The number of elements tested is determined by the VL register; bits (VL) through 127 of VM are cleared to 0's.

Instruction 175*ij*0 tests each V*j* element (up to element VL–1) for a zero condition and sets VM accordingly. An element is zero if all bits are 0's.

Instruction 175*ij*1 tests each V*j* element (up to element VL–1) for a nonzero condition and sets VM accordingly. An element is nonzero if one or more bits are 1.

Instruction 175*ij*2 tests each V*j* element (up to element VL–1) for a positive or zero condition and sets VM accordingly. An element is positive or zero if its sign bit (bit 63) is 0.

Instruction 175*ij*3 tests each V*j* element (up to element VL–1) for a negative condition and sets VM accordingly. An element is negative if its sign bit (bit 63) is 1.

Instructions 175*ij*4 through 175*ij*7

Instructions 175*ij*4 through 175*ij*7 test the elements of register V*j* for a specified condition in exactly the same way as instructions 175*ij*0 through 175*iij*3. In addition to loading the VM register, instructions 175*ij*4 through 175*ij*7 generate a compressed index in register V*i* of the element that meets the test condition.

The compressed index is generated as follows. The number of the first element of register V*j* that meets the test condition is stored in element 0 of register V*i*. The number of the second element that meets the test condition is stored in element 1 of register V*i*. The number of the final element that meets the test condition is stored in element X – 1 of register V*i*, where X is the number of elements in register V*j* that meet the test condition. Elements X through 127 of register V*i* are not changed.

Instruction 175*ij*4 tests each V*j* element (up to element VL–1) for a zero condition. It sets VM accordingly and generates a compressed index in register V*i*. An element is zero if all bits are 0's.

Instruction 175*ij*5 tests each V*j* element (up to element VL–1) for a nonzero condition. It sets VM accordingly and generates a compressed index in register V*i*. An element is nonzero if one or more bits are 1.

Instruction 175*ij*6 tests each V*j* element (up to element VL–1) for a positive or zero condition. It sets VM accordingly and generates a compressed index in register V*i*. An element is positive or zero if its sign bit (bit 63) is 0.

Instruction 175*ij*7 tests each V*j* element (up to element VL–1) for a negative condition. It sets VM accordingly and generates a compressed index in register V*i*. An element is negative if its sign bit (bit 63) is 1.

Example:

```
VM              =  0101100111010 … 0₂
V2 element 0    =  1
V2 element 1    =  0
V2 element 2    =  5
V2 element 3    =  0
V2 element 4    =  0
V2 element 5    =  4
V2 element 6    =  6
V2 element 7    =  0
V2 element 8    =  0
V2 element 9    =  0
```

The results of executing instruction 174124 (V1,VM V2,Z) are:

V1 element 0	=	1
V1 element 1	=	3
V1 element 2	=	4
V1 element 3	=	7
V1 element 4	=	8
V1 element 5	=	9
V1 elements 6 – 127	=	unchanged

Instructions 176 and 177

Machine Instruction		CAL Syntax		Description
	176*i*0*k*	V*i*	,A0,A*k*	Load V*i* from memory starting at address (A0) and incrementing by (A*k*).
	176*i*00	V*i*	,A0,1[S]	Load V*i* from consecutive memory addresses starting at (A0).
	176*i*1*k*	V*i*	,A0,V*k*	Load V*i* from memory using addresses (A0) + (V*k*).
005400	176*ijk*[NT]	V*i*:V*j*	,A0:A*k*,V*k*	Load V*i* from memory using addresses (A0) + (V*k*) and load V*j* from memory using addresses (A*k*) + (V*k*).
	1770*jk*	,A0,A*k*	V*j*	Store (V*j*) to memory starting at address (A0) and incrementing by (A*k*).
	1770*j*0	,A0,1	V*j*[S]	Store (V*j*) to consecutive memory addresses starting at (A0).
	1771*jk*	,A0,V*k*	V*j*	Store (V*j*) to memory using addresses (A0) + (V*k*).

N New instruction (not available on CRAY C90 series systems).
T Triton mode only.
S Special CAL syntax.

Special Cases

For instructions 176*i*0*k* and 1770*jk*: If *k* = 0, (A*k*) = 0 and the address increment is 1.

Hold-issue Conditions

Instruction 176*i*0*k*:

- Register A0 or A*k* reserved.
- Register V*i* reserved as operand or result.
- Scalar load issued in the preceding 4 CPs.
- Cache miss sequence in progress.
- Ports A and B busy.
- Bidirectional memory (BDM) bit clear and port C busy.

Instruction 176*i*1*k*:

- Register A0 reserved.
- Register V*k* reserved as operand.
- Register V*i* reserved as operand or result.

- Scalar load issued in the preceding 4 CPs.
- Ports A and B busy.
- Bidirectional memory (BDM) bit clear and port C busy.

Instruction 005400 176*ijk*:

- Register A0 or A*k* reserved.
- Register V*k* reserved as operand.
- Register V*i* or V*j* reserved as operand or result.
- Scalar load issued in the preceding 4 CPs.
- Cache miss sequence in progress.
- Port A or B busy.
- Bidirectional memory (BDM) bit clear and port C busy.

Instruction 1770*jk*:

- Register A0 or A*k* reserved.
- Register V*j* reserved as operand.
- Port C busy.
- Bidirectional memory (BDM) bit clear and port A or B busy.

Instruction 1771*jk*:

- Register A0 reserved.
- Register V*j* or V*k* reserved as operand.
- Port C busy.
- Bidirectional memory (BDM) bit clear and port A or B busy.

Execution Time

Instruction issue:

- Unprefixed instructions: 1 CP.
- 005400-prefixed instruction: 2 CPs.

Instruction 176*i*0*k*:

- Port A or B busy: VL/2 + 5 CPs.
- At a minimum, V*i* ready in VL/2 + 40 CPs (4-processor systems and smaller), VL/2+ 57 CPs (8-processor systems and larger).

Instruction 176*i*1*k* and 005400 176*ijk*:

- Port A or B busy: VL/2 + 10 CPs.
- At a minimum, V*i* ready in VL/2 + 43 CPs (CRAY T94 and smaller systems), VL/2+ 60 CPs (CRAY T98 and larger systems).

Instruction 1770*jk*:

- Register V*j* busy: VL/2 + 4 CPs.
- Port C busy: VL/2 + 5 CPs.

Instruction 1771*jk*:

- Registers V*j* and V*k* busy: VL/2 + 4 CPs.
- Port C busy: VL/2 + 10 CPs.

For instruction 176, additional time can be added to V*i* access because of memory conflicts. There can also be about 3 CPs of additional delay in some cases because of system path-length differences and memory-timing phases.

Description

Instructions 176 and 177 transfer blocks of data between common memory and the V registers. The contents of the vector length (VL) register determine the number of words transferred.

Instruction 176*ik*

Instruction 176*i*0*k* transfers data from common memory to register V*i*. Register A0 contains the initial (base) address. Register A*k* contains the address increment (stride), which can be either positive or negative.

The first data word is transferred from memory address (A0) to element 0 of register V*i*. Then the memory address is incremented by (A*k*) and the memory word at the new address is transferred to the next element of register V*i*. This process continues until (VL) words have been transferred.

Instruction 176*i*1*k*

Instruction 176*i*1*k* transfers data from common memory to register V*i*. Register A0 contains the initial (base) address. Register V*k* contains address indexes. This is the gather instruction.

For each element (word) transferred to register V*i*, the memory address is the sum of (A0) and the corresponding element of register V*k*. Thus V*i*[0] is loaded from address (A0) + (V*k*[0]), V*i*[1] is loaded from address (A0) + (V*k*[1]), etc.

Instruction 005400 176*ijk*

Instruction 005400 176*ijk* transfers data from common memory to registers V*i* and V*j* in two separate data transfers that occur simultaneously. Register A0 contains the base address for the transfer to register V*i*. Register A*k* contains the base address for the transfer to register V*j*. Register V*k* contains address indexes for both transfers. This is the double gather instruction.

For each element (word) transferred to register V*i*, the memory address is the sum of (A0) and the corresponding element of register V*k*. Thus V*i*[0] is loaded from address (A0) + (V*k*[0]), V*i*[1] is loaded from address (A0) + (V*k*[1]), etc.

For each element (word) transferred to register V*j*, the memory address is the sum of (A*k*) and the corresponding element of register V*k*. Thus V*j*[0] is loaded from address (A*k*) + (V*k*[0]), V*j*[1] is loaded from address (A*k*) + (V*k*[1]), etc.

Instruction 1770*jk*

Instruction 1770*jk* transfers data from register V*j* to common memory. Register A0 contains the initial (base) address. Register A*k* contains the address increment (stride), which can be either positive or negative.

The first data word is transferred from element 0 of register V*j* to memory address (A0). Then the memory address is incremented by (A*k*) and the next element of register V*i* is transferred to the new address. This process continues until (VL) words have been transferred.

Instruction 1771*jk*

Instruction 1771*jk* transfers data from register V*j* to common memory. Register A0 contains the initial (base) address. Register V*k* contains address indexes. This is the scatter instruction.

For each element (word) transferred from register V*i*, the memory address is the sum of (A0) and the corresponding element of register V*k*. Thus V*i*[0] is stored to address (A0) + (V*k*[0]), V*i*[1] is stored to address (A0) + (V*k*[1]), etc.

Common memory addressing

In C90 mode, for instructions 176*i*0*k* and 1770*jk*, the 32-bit contents of register A0 are added to the 32-bit contents of register A*k* to produce 32-bit logical memory addresses. For instructions 176*i*1*k* , 1771*jk*, and 005400 176*ijk*, the 32-bit contents of register A0 (also register A*k* for 005400 176*ijk*) are added to bits 0 through 39 of register V*k* to produce 40-bit logical addresses.

In Triton mode, the A and V registers are all 64 bits wide. Only bits 0 through 39 are used for common memory addressing, producing 40-bit logical addresses.