PA-RISC 2.0 Architecture

This document is an excerpt from PA-RISC 2.0 ARCHITECTURE, by Gerry Kane, published by Prentice Hall PTR, isbn 0-13-182734-0. Print copies of the book can be ordered at www.hp.com/hpbooks.

This document contains:
Chapter 2: Processing Resources
2 Processing Resources

The PA-RISC instruction set is only one aspect of the processor architecture; the following components are also specified:

• Processing Resources — what registers and register sets are available to the user and to system software
• Data Types — how data is organized and what data types are available to the user
• Memory and I/O Addressing — how system memory and the input/output facilities are organized and accessed.

This chapter describes the processing resources and data types in a PA-RISC system. The memory and I/O addressing aspects are described in Chapter 3, “Addressing and Access Control”.

The software-accessible registers (that is, the processing resources) are the storage elements within a processor that are manipulated by the instructions. These resources participate in instruction control flow, computations, interruption processing, protection mechanisms, and virtual memory management. The software-accessible registers can be divided into two groups: non-privileged registers and privileged registers. Privileged registers are those that generally can be accessed using instructions that can be executed only when at the most privileged level. Figure 2-1 illustrates the registers provided in the PA-RISC architecture.

![Figure 2-1. Software Accessible Registers](image-url)
Non-Privileged Software-Accessible Registers

These registers can be accessed by any program at any time, regardless of the current privilege level and include those typically needed by application software (as opposed to system software.)

- General Registers (GR 0..GR 31)
- Space Registers (SR 0..SR 7 - SR5-SR7 are privileged.)
- Instruction Address Queues
- Coprocessor Registers
- Special Function Unit Registers
- subset of Control Registers (Timer, SAR, CR26,27))

General Registers

Thirty-two 64-bit general registers provide the central resource for all computation (Figure 2-2). They are numbered GR 0 through GR 31, and are available to all programs at all privilege levels.

GR 0, GR 1, GR2, and GR 31 have special functions.

- GR 0, when referenced as a source operand, delivers zeros. When GR 0 is used as a destination, the result is discarded.
- GR 1 is the implicit target of the ADD IMMEDIATE LEFT instruction.
- GR 2 is the instruction address offset link register for the long displacement form of the normal call instruction (BRANCH AND LINK).
- GR 31 is the instruction address offset link register for the base-relative interspace procedure call instruction [BRANCH EXTERNAL instruction with the (optional) L (for link) completer].

GR 1, GR2, and GR 31 can also be used as general registers; however, software conventions may at times restrict their use.
Space Registers

A PA-RISC system provides eight space registers, numbered SR 0 through SR 7, which contain space IDs for virtual addressing. Instructions specify space registers either directly in the instruction or indirectly through general register contents.

Instruction addresses, computed by branch instructions, may use any of the space registers. SR 0 is the instruction address space link register for the base-relative interspace procedure call instruction [BRANCH EXTERNAL instruction with the (optional) L (for link) completer]. Data operands can specify SR 1 through SR 3 explicitly, and SR 4 through SR 7 indirectly, via general registers.

SR 1 through SR 7 have no special functions; however, their use will normally be constrained by software conventions. For example, the following convention supports non-overlapping process groups. SR 1 through SR 3 provide general-use virtual pointers. SR 4 tracks the instruction address (IA) space and provides access to literal data contained in the current code segment. SR 5 points to a space containing process private data, SR 6 to a space containing data shared by a group of processes, and SR 7 to a space containing the operating system’s public code, literals, and data. Figure 2-3 illustrates this convention.

SRs 5 through 7 can be modified only by code executing at the most privileged level.
Space registers, as well as IASQ, IIASQ, and ISR which are described later, may be any size between 32 bits and 64 bits to support a virtual address size between 64 and 96 bits.

**Instruction Address Queues**

The Instruction Address Queues hold the address of the currently executing instruction and the address of the instruction that will be executed after the current instruction, termed the following instruction. Note that the following instruction is not necessarily the next instruction in the linear code space. These two queues are each two elements deep. The Instruction Address Offset Queue (IAOQ) elements are each 64 bits wide. The high-order 62 bits contain the word offset of the instruction while the 2 low-order bits maintain the privilege level of the corresponding instruction. There are four privilege levels: 0, 1, 2, and 3 with 0 being the most privileged level.

The Instruction Address Space Queue (IASQ) contains the space ID of the current and following instructions. The IASQ may be from 32 to 64 bits in size. The space ID of the current instruction, when executing without instruction address translation enabled, is not specified and may contain any value.

The front elements of the two queues (IASQ_Front and IAOQ_Front) form the virtual address of the current instruction while the back elements of the two queues (IASQ_Back and IAOQ_Back) contain the address of the following instruction. Figure 2-4 shows this structure. Two addresses are maintained to support the delayed branching capability (See “Concept of Delayed Branching” on page 4-1).
Control Registers (non-privileged)

Although most of the Control Registers can be accessed only by privileged instructions, the Shift Amount Register (SAR), Interval Timer, and temporary registers CR26,27 are accessible at any time and are described in the paragraphs that follow.

Shift Amount Register
The Shift Amount Register or SAR (CR 11), is a 6-bit register used by the variable shift, extract, deposit, and branch on bit instructions. It specifies the number of bits a quantity is to be shifted. The remaining 58 bits are ignored bits.

Interval Timer
The Interval Timer (CR 16) consists of two internal registers. One of the internal registers is a 64-bit counter which continually counts up by 1 at a rate which is implementation-dependent and between twice the “peak instruction rate” and half the “peak instruction rate”. Reading the Interval Timer returns the value of this internal 64-bit register. The other internal register contains a 32-bit comparison value and is set by writing to the Interval Timer. When the least significant 32 bits of the counter register and the comparison register contain identical values, a bit in the External Interrupt Request Register is set to 1. This causes an external interrupt, if not masked. The W bit (Wide enable) in the Processor Status Word (PSW - see Table 2-1) determines which bit of the EIRR is set. If the W bit is 0, the timer comparison causes bit 32 to be set to 1. If the W bit is 1, the timer comparison causes bit 0 of the EIER to be set to 1.

The Interval Timer can only be written by code executing at the most privileged level. If the PSW S-bit is 1, the Interval Timer can only be read by code executing at the most privileged level; otherwise, it can be read by code executing at any privilege level.

In a multiprocessor system, each processor must have its own Interval Timer. Each Interval Timer need not be synchronized with the other Interval Timers in the system, nor do they need to be clocked at the same frequency.

If, as part of a power-saving mode, the processor clock is reduced below the “peak instruction rate”, the Interval Timer continues to count at its peak rate. If the processor clock is stopped, the Interval Timer may also stop.

Temporary Registers
Two of the eight 64-bit temporary registers (CRs 26 and 27) are readable by code executing at any privilege level and writable only by code executing at the most privileged level.

Coprocessor Registers
Each coprocessor may have its own register set. The coprocessor mechanism is described in “Assist Instructions” on page 6-19. The floating-point coprocessor registers are described in Chapter 8, “Floating-point Coprocessor”. The performance monitor coprocessor registers are described in Chapter 11, “Performance Monitor Coprocessor”.

PA-RISC 2.0 Architecture  Processing Resources  2-5
SFU Registers

Each special function unit may have its own register set. The SFU mechanism is described in “Assist Instructions” on page 6-19.

Branch Target Stack

The Branch Target Stack (or BTS) is an optional processing resource which is used to accelerate indirect branches, such as subroutine returns. The BTS is managed by software, and in processors which implement it, can provide the branch target address in place of the general register specified in the branch instruction.

Conceptually, the BTS is a stack of 63-bit registers. The number of registers is implementation dependent, and can be 0. Each register holds an instruction address plus a valid bit. Although the BTS is not directly readable, it can be thought of as being laid out as in Figure 2-5.

![Figure 2-5. Branch Target Stack](image)

Certain instructions push an address onto the top of the stack, forcing all other entries down one register, with the old value of the last register (bottom of stack) being discarded. When a value is pushed onto the stack, the valid bit is set to 1 for that entry.

Other operations pop an address from the top of the stack. If the valid bit associated with the address is 1, the address may be used as a branch target, to decrease the latency of the branch. If the entry is invalid, it is ignored, and the branch target is calculated the normal way (using the specified general register). When the stack is popped, each entry moves up one register, and the register at the bottom of the stack is marked invalid.

The Branch Nomination Register (or BNR) is a register which holds one instruction address. It also has a valid bit associated with it. The BNR allows software to make use of the BTS in a called function, even though the caller function does not attempt to use the stack (perhaps because it is older code).

![Figure 2-6. Branch Nomination Register](image)

Implementation of the BTS is optional. Hardware may invalidate entries in the stack at any time, so software may not rely on entries remaining valid.
Privileged Software-Accessible Registers

These registers can be accessed only when the processor is in the most privileged mode and are intended for use by system software.

- Processor Status Word (PSW)
- Shadow Registers (SHR 0..SHR 6)
- Control Registers (CR 0..CR 31)

**Processor Status Word (PSW)**

Processor state is encoded in a 64-bit register called the Processor Status Word (PSW). When an interruption occurs, the current value of the PSW is saved in the Interruption Processor Status Word (IPSW) and usually all defined PSW bits are set to 0. The format of the PSW is shown in Figure 2-7.

![Figure 2-7. Processor Status Word](image)

The PSW is set to the contents of the IPSW by the RETURN FROM INTERRUPTION instruction. The interruption handler may restore the original PSW, modify selected bits, or may change the PSW to an entirely new value.

The E, O, W, F, R, Q, P, D, and I bits of the PSW are known as the system mask. Each of these bits, with the exception of the Q-bit, may be set to 1, set to 0, written, and read by the system control instructions that manipulate the system mask. The Q-bit is specially defined. It can be set to 0 by system control instructions that manipulate the system mask, but setting it to 1 when the current value is 0 is an undefined operation. The only instruction that can set the Q-bit to 1 is the RETURN FROM INTERRUPTION instruction.

Some of the PSW bits are termed mask/unmask bits whereas others are termed disable/enable bits. Interruptions that are masked remain pending whereas those that are disabled are ignored.
The PSW fields are described in Table 2-1.

Table 2-1. Processor Status Word

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rv</td>
<td>Reserved bits.</td>
</tr>
<tr>
<td>W</td>
<td>Wide 64-bit address formation enable. When 1, full 64-bit-offset addressing is enabled. When 0, addresses are truncated to 32-bit offsets, for compatibility with existing PA-RISC 1.0 and 1.1 applications.</td>
</tr>
<tr>
<td>E</td>
<td>Little endian memory access enable. When 0, all memory references are big endian. When 1, all memory references are little endian. Implementation of this bit is optional. If it is not implemented, all memory references are big endian and this bit is a reserved bit.</td>
</tr>
<tr>
<td>S</td>
<td>Secure Interval Timer. When 1, the Interval Timer is readable only by code executing at the most privileged level. When 0, the Interval Timer is readable by code executing at any privilege level.</td>
</tr>
<tr>
<td>T</td>
<td>Taken branch trap enable. When 1, any taken branch is terminated with a taken branch trap.</td>
</tr>
<tr>
<td>H</td>
<td>Higher-privilege transfer trap enable. When 1, a higher-privilege transfer trap occurs whenever the following instruction is of a higher privilege.</td>
</tr>
<tr>
<td>L</td>
<td>Lower-privilege transfer trap enable. When 1, a lower-privilege transfer trap occurs whenever the following instruction is of a lower privilege.</td>
</tr>
<tr>
<td>N</td>
<td>Nullify. The current instruction is nullified when this bit is 1. This bit is set to 1 by an instruction that nullifies the following instruction.</td>
</tr>
<tr>
<td>X</td>
<td>Data memory break disable. The X-bit is set to 0 after the execution of each instruction, except for the RETURN FROM INTERRUPTION instruction which may set it to 1. When 1, data memory break traps are disabled. This bit allows a simple mechanism to trap on a data store and then proceed past the trapping instruction.</td>
</tr>
<tr>
<td>B</td>
<td>Taken branch. The B-bit is set to 1 by any taken branch instruction and set to 0 otherwise. This is used to ensure that the BRANCH instruction with the ,GATE completer (the privilege increasing instruction) cannot be used to compromise system security.</td>
</tr>
<tr>
<td>C</td>
<td>Code (instruction) address translation enable. When 1, instruction addresses are translated and access rights checked.</td>
</tr>
<tr>
<td>V</td>
<td>Divide step correction. The DIVIDE STEP (integer division primitive) instruction records intermediate status in this bit to provide a non-restoring divide primitive.</td>
</tr>
<tr>
<td>M</td>
<td>High-priority machine check mask. When 1, high-priority machine checks (HPMCs) are masked. Normally 0, this bit is set to 1 after an HPMC and set to 0 after all other interruptions.</td>
</tr>
</tbody>
</table>
The instructions marked with an asterisk set the carry/borrow bits only if the \( L \) (logical) completer is not specified.

After an add which sets them, each bit is set to 1 if a carry occurred out of its corresponding digit, and set to 0 otherwise. After a subtract which sets them, each bit is set to 0 if a borrow occurred into its corresponding digit, and set to 1 otherwise. Bits \( \{24..31\} \) hold the digit carries from the upper half of the ALU, and bits \( \{48..55\} \) hold the digit carries from the lower half.

<table>
<thead>
<tr>
<th>C/B</th>
<th>Carry/borrow bits. The following instructions update the PSW carry/borrow bits from the corresponding carry/borrow outputs of the 4-bit digits of the ALU:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{ADD}^* ) \text{ADDI} \text{DS} \text{SHLADD}^* \text{SUB} \text{SUBL}</td>
</tr>
</tbody>
</table>

The instructions marked with an asterisk set the carry/borrow bits only if the \( L \) (logical) completer is not specified.

After an add which sets them, each bit is set to 1 if a carry occurred out of its corresponding digit, and set to 0 otherwise. After a subtract which sets them, each bit is set to 0 if a borrow occurred into its corresponding digit, and set to 1 otherwise. Bits \( \{24..31\} \) hold the digit carries from the upper half of the ALU, and bits \( \{48..55\} \) hold the digit carries from the lower half.

| O  | Ordered references. When 1, virtual memory references to pages with the corresponding TLB O-bit 1, and all absolute memory references, are ordered. When 0, memory references (except those explicitly marked as ordered or strongly ordered) may be weakly ordered. Note that references to I/O address space, references to pages with the TLB U-bit 1, semaphore instructions, and TLB purge instructions are always strongly ordered. |

| F  | Performance monitor interrupt unmask. When 1, the performance monitor interrupt is unmasked and can cause an interruption. When 0, the interruption is held pending. Implementation of this bit is required only if the performance monitor is implemented and the performance monitor has the ability to interrupt. If it is not implemented, this bit is a reserved bit. |

| R  | Recovery Counter enable. When 1, recovery counter traps occur if bit 0 of the recovery counter is a 1. This bit also enables decrementing of the Recovery Counter. |

| Q  | Interruption state collection enable. When 1, interruption state is collected. Used in processing the interruption and returning to the interrupted code, this state is recorded in the Interruption Instruction Address Queue (IIAQ), the Interruption Instruction Register (IIR), the Interruption Space Register (ISR), and the Interruption Offset Register (IOR). |

| P  | Protection identifier validation enable. When this bit and the C-bit are both equal to 1, instruction references check for valid protection identifiers (PIDs). When this bit and the D-bit are both equal to 1, data references check for valid PIDs. When this bit is 1, probe instructions check for valid PIDs. |

| D  | Data address translation enable. When 1, data addresses are translated and access rights checked. |

| I  | External interrupt, power failure interrupt, and low-priority machine check interruption unmask. When 1, these interruptions are unmasked and can cause an interruption. When 0, the interruptions are held pending. |
Shadow Registers

There are seven registers shadow registers. Upon interruption, if the PSW Q-bit was 1, the contents of GRs 1, 8, 9, 16, 17, 24, and 25 are copied into shadow registers SHR 0, 1, 2, 3, 4, 5, and 6, respectively. If an interruption is taken with the PSW Q-bit equal to 0, the shadow registers are unchanged. The contents of these general registers are restored from their shadow registers when a RETURN FROM INTERRUPTION instruction with the (optional) R (for restore) completer is executed.

Control Registers

There are twenty-five defined control registers, numbered CR 0, and CR 8 through CR 31, which contain system state information.

The control registers are shown in Figure 2-8 and described in the following sections. (The control registers that can be accessed in the non-privileged state are described earlier in the section “Control Registers (non-privileged)” on page 2-5.) Moving the contents of a control register to a general register copies the register contents right aligned into the general register. Moving the contents of a general register to a control register copies the entire general register into the control register.

Control registers 1 through 7 are reserved registers.
The Recovery Counter (CR 0) is a 32-bit counter that can be used to provide software recovery of hardware faults in fault-tolerant systems, and can also be used for debugging purposes. CR 0 counts down by 1 during the execution of each non-nullified instruction for which the PSW R-bit is 1. The Recovery Counter is restored if the instruction terminates with a group 1, 2, or 3 interruption (see Chapter 4, “Control Flow”). When the leftmost bit of the Recovery Counter is 1, a recovery counter trap occurs. The trap and the decrement operation can be disabled by setting the PSW R-bit to 0. The value

---

**Figure 2-8. Control Registers**

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR 0</td>
<td>non-existent</td>
</tr>
<tr>
<td>CR 8</td>
<td>Protection ID 1 (WD) Protection ID 2 (WD)</td>
</tr>
<tr>
<td>CR 9</td>
<td>Protection ID 3 (WD) Protection ID 4 (WD)</td>
</tr>
<tr>
<td>CR 10</td>
<td>reserved SCR CCR</td>
</tr>
<tr>
<td>CR 11</td>
<td>ignored SAR</td>
</tr>
<tr>
<td>CR 12</td>
<td>Protection ID 5 (WD) Protection ID 6 (WD)</td>
</tr>
<tr>
<td>CR 13</td>
<td>Protection ID 7 (WD) Protection ID 8 (WD)</td>
</tr>
<tr>
<td>CR 14</td>
<td>Interruption Vector Address reserved</td>
</tr>
<tr>
<td>CR 15</td>
<td>External Interrupt Enable Mask</td>
</tr>
<tr>
<td>CR 16</td>
<td>Interval Timer</td>
</tr>
<tr>
<td>CR 17</td>
<td>Interruption Instruction Address Space Queue</td>
</tr>
<tr>
<td>CR 18</td>
<td>Interruption Instruction Address Offset Queue</td>
</tr>
<tr>
<td>CR 19</td>
<td>reserved Interruption Instruction Register</td>
</tr>
<tr>
<td>CR 20</td>
<td>Interruption Space Register</td>
</tr>
<tr>
<td>CR 21</td>
<td>Interruption Offset Register</td>
</tr>
<tr>
<td>CR 22</td>
<td>Interruption Processor Status Word</td>
</tr>
<tr>
<td>CR 23</td>
<td>External Interrupt Request Register</td>
</tr>
<tr>
<td>CR 24</td>
<td>Temporary Register</td>
</tr>
<tr>
<td>CR 31</td>
<td>Temporary Register</td>
</tr>
</tbody>
</table>

---
of the Recovery Counter may be read reliably only when the PSW R-bit is 0. (Reading the Recovery Counter when the PSW R-bit is 1 returns an undefined result.) The Recovery Counter may be written reliably only when the PSW R-bit is 0. (Writing the Recovery Counter when the PSW R-bit is 1 is an undefined operation.) If the PSW R-bit is set to 0 by either the RESET SYSTEM MASK or the MOVE TO SYSTEM MASK instruction, the Recovery Counter may not be read or written reliably prior to the execution of the eighth instruction after the RESET SYSTEM MASK or the MOVE TO SYSTEM MASK instruction. An interruption, or a RETURN FROM INTERRUPTION instruction which sets the PSW R-bit to 0, does not have this restriction.

Protection Identifiers
The protection identifiers (CRs 8, 9, 12, 13) designate up to eight groups of pages which are accessible to the currently executing process. When translation is enabled, the eight protection identifiers (PIDs) are compared with a page access identifier in the TLB entry to validate an access. (See “Access Control” on page 3-11.) The rightmost bit of each of the eight PIDs is the write disable (WD) bit. When the WD-bit is 1, that PID cannot be used to grant write access. This allows each process sharing memory to have different access rights to the memory without the overhead of changing the access identifier and access rights in the TLB. When the PSW P-bit is 0, the PIDs, including the WD-bits, are ignored.

Each of the 8 PID registers can be from 16 to 32 bits wide (including the WD bit), with the remaining bits being reserved bits. The length of the PIDs is implementation dependent.

Coprocessor Configuration Register (CCR)
The Coprocessor Configuration Register or CCR (bits 56..63 of CR 10) is an 8-bit register which records the presence and usability of coprocessors. The bit positions are numbered 0 through 7, and correspond to a coprocessor with the same unit identifier. Bits 0 and 1 correspond to the floating-point coprocessor, and bit 2 corresponds to the performance monitor coprocessor. Bit 7 is the rightmost bit of the CCR. It receives bit 63 from a general register when a general register is written to CR 10. The upper 48 bits of CR 10, and bits within the CCR corresponding to coprocessors which are not present, are reserved bits.

The behavior of the floating-point coprocessor with respect to the state of CCR bits 0 and 1 and the behavior of the performance monitor coprocessor with respect to the state of CCR bit 2, are specified in “Coprocessor Instructions” on page 6-22. For other coprocessors, setting a bit in the CCR to 1 enables the use of the corresponding coprocessor, if present and operational. If a CCR bit is 0, the corresponding coprocessor, if present, is logically decoupled. This decoupling must ensure that the state of a coprocessor does not change as long as its corresponding CCR bit is 0. When a CCR bit is set to 0 and an attempt is made to execute an instruction which references the corresponding coprocessor, it causes an assist emulation trap. It is an undefined operation to set to 1 any CCR bit corresponding to a coprocessor which is not present.

SFU Configuration Register (SCR)
The SFU Configuration Register or SCR (bits 48..55 of CR 10), is an 8-bit register which records the presence and usability of special function units. The bit positions are numbered 0 through 7, and correspond to an SFU with the same unit identifier. Bit 7 is the rightmost bit of the SCR. It receives bit 55 from a general register when a general register is written to CR 10. The upper 48 bits of CR 10, and bits within the SCR corresponding to SFUs which are not present, are reserved bits.
For all SFUs, setting a bit in the SCR to 1 enables the use of the corresponding SFU, if present and operational. If an SCR bit is 0, the corresponding SFU, if present, is logically decoupled. This decoupling must ensure that the state of an SFU does not change as long as its corresponding SCR bit is 0. When an SCR bit is set to 0 and an attempt is made to execute an instruction which references the corresponding SFU, it causes an assist emulation trap. The operation of an SFU when its corresponding SCR bit is 0 is explained in more detail in “Special Function Unit (SFU) Instructions” on page 6-20. It is an undefined operation to set to 1 any SCR bit corresponding to an SFU which is not present.

**Interruption Vector Address (IVA)**
The Interruption Vector Address or IVA (CR 14) contains the absolute address of the base of an array of service procedures assigned to the interruption classes. The lower 11 bits of the IVA are reserved Therefore, the address written to it must be a multiple of 2048. For implementations with fewer than 64 bits of physical address, the upper bits of the IVA corresponding to unimplemented physical address bits are reserved. The array of interruption service procedures is indexed by the interruption numbers given in Chapter 4, “Control Flow”.

**External Interrupt Enable Mask (EIEM)**
The External Interrupt Enable Mask or EIEM (CR 15), is a 64-bit register containing a bit for each of the 64 external interrupts. Each 0 bit in the EIEM masks external interrupts corresponding to that bit position.

**Interruption Instruction Address Queues**
The Interruption Instruction Address Space Queue or IIASQ (CR 17) and the Interruption Instruction Address Offset Queue or IIAOQ (CR 18) are collectively termed the interruption instruction address or IIA queues. They are used to save the Instruction Address and privilege level information for use in processing interruptions. The registers are arranged as two two-element deep queues. The queues generally contain the addresses (including the privilege level field in the rightmost two bits of the offset part) of the two instructions in the IA queues at the time of the interruption. The IIASQ may be from 32 to 64 bits wide.

The IIA queues are continually updated whenever the PSW Q-bit is 1 and are frozen by an interruption (PSW Q-bit becomes 0). After such an interruption, the IIA queues contain copies of the information from the IA queues. The IIAOQ contains the address offsets of the interruption point in the same format as the IAOQ. The IIASQ has a different format from that of the IASQ. The IIASQ contains the upper portion of the GVA (global virtual address) of the interruption point, if code address translation was enabled. (Note that if the PSW W-bit was 0, the upper portion of the GVA is simply the space ID.)
If code address translation was disabled at the time of the interruption, then the IIAOQ contains the absolute offsets of the interruption point, and the IIASQ contains zeros. (Note that if the PSW W-bit was 0, the absolute offsets in the IIAOQ may be truncated to only those bits of the physical address space that are implemented, and the upper bits forced to zeros.)

On a return from interruption, the values in the IIA queues are used to reform the IA queues for the return point. The values in the IIAOQ are copied to the IAOQ. The new values for the IASQ are formed as follows (see also Figure 2-12): the lower 30 bits of the IASQ are formed by taking the bitwise AND of the lower 30 bits of the values in the IIASQ with the complement of bits \(\{2..31\}\) of the values in the
IIAQ. Bits \{32..33\} of the IIASQ are copied to the same bits in the IASQ. The upper 32 bits of the IIASQ (or as many as are implemented) are copied to the corresponding bits of the IASQ. This reforms the original space identifiers.

![Diagram](image)

Figure 2-12. Reforming Space Identifiers

Reading the IIAOQ (CR 18) while the PSW Q-bit is 0 retrieves the offset and privilege level portions of the front element in the IIAOQ. Writing into IIAOQ while the PSW Q-bit is 0 advances the IIAOQ and then sets the offset and privilege level portions of the back element of the IIAOQ. Reading the IIASQ (CR 17) while the PSW Q-bit is 0 retrieves the GVA portion of the front element of the IIASQ. Writing into IIASQ while the PSW Q-bit is 0 advances the IIASQ and then writes into the back element of the IIASQ. The effect of reading or writing either queue register while the PSW Q-bit is 1 is an undefined operation.

The state contained in the IIA queues is undefined when a RETURN FROM INTERRUPTION instruction sets the PSW Q-bit to 0, or when system control instructions are used to set the PSW Q-bit to 0. If an interruption is taken with the PSW Q-bit equal to 0, the IIA queues are unchanged.

** Interruption Parameter Registers (IPRs) **

The Interruption Parameter Registers (IPRs) are used to pass an instruction and a virtual address to an interruption handler. Three registers comprise the IPRs: the Interruption Instruction Register or IIR (CR 19), Interruption Space Register or ISR (CR 20), and Interruption Offset Register or IOR (CR 21). They are used to pass an instruction and a virtual address to an interruption handler. The values in these registers for each interruption class are specified in Chapter 4, “Control Flow”. These values are set (or frozen) at the time of the interruption whenever the PSW Q-bit is 1. The ISR may be from 32 to 64 bits wide.

The value loaded into the IOR is the lower 32 bits of the virtual address offset without truncating the rightmost bits or setting them to 0, plus the 2 bits of the base register which was used to form the address. If the PSW W-bit was 1, the upper 2 bits of the IOR (called the b field) are equal to bits \{0..1\} from the base register. If the PSW W-bit was 0, the b field is equal to bits \{32..33\} from the base register. The other bits of the IOR are forced to 0.
The value loaded into the ISR is the upper portion of the GVA, if data translation was enabled. (Note that if the PSW W-bit was 0, the upper portion of the GVA is simply the space ID.)

If data translation was disabled at the time of the interruption, the IOR contains the lower 32 bits of the absolute offset. The upper 2 bits of the IOR are undefined, and may be set to any value. The other bits in the IOR are forced to 0. The ISR contains the upper portion of the absolute offset, zero-extended. If the PSW W-bit was 0 as well, the ISR contains 0.

The interruption parameter registers can be read or written reliably only when the PSW Q-bit is 0. (Reading an interruption parameter register when the PSW Q-bit is 1 returns an undefined result.) The state contained in the IPRs is undefined when a RETURN FROM INTERRUPTION instruction sets the PSW Q-bit to 0, or when system control instructions are used to set the PSW Q-bit to 0. If an interruption is taken with the PSW Q-bit equal to 0, the IPRs are unchanged.

**Interruption Processor Status Word (IPSW)**

The Interruption Processor Status Word or IPSW (CR 22) receives the value of the PSW when an interruption occurs. The format of the IPSW is identical to that of the PSW. The IPSW always reflects the state of the machine at the point of interruption, regardless of the state of the PSW Q-bit. As in the PSW, the unnamed bits are reserved bits.

The IPSW can be read or written reliably only when the PSW Q-bit is 0. (Reading the IPSW when the PSW Q-bit is 1 returns an undefined result.) The state contained in the IPSW is undefined when a RETURN FROM INTERRUPTION instruction sets the PSW Q-bit to 0, or when system control instructions are used to set the PSW Q-bit to 0.

**External Interrupt Request Register (EIRR)**

The External Interrupt Request register or EIRR (CR 23) is a 64-bit register containing a bit for each external interrupt. When 1, a bit designates that an interruption is pending for the corresponding
external interrupt. Both the PSW I-bit (external interrupt, power failure interrupt, and low-priority machine check unmask) and the corresponding bit position in the External Interrupt Enable Mask (CR 15) must be 1 for an interruption to occur.

A MOVE TO CONTROL REGISTER instruction with CR 23 as its target bitwise ANDs the complement of the contents of the source register with the previous contents of CR 23, and places this result in CR 23. Thus the processor can only set the EIR register bits to 0.

A processor’s EIR register is also memory mapped into the physical address space as the IIO_EIR register to enable other processors and I/O modules to interrupt the processor. When a module writes to it, the bit specified by the value written is set to 1. The W bit (Wide enable) in the Processor Status Word (PSW - see Table 2-1) determines whether the EIRR operates as a 32-bit register or a 64-bit register. When the W bit is 0, the EIRR operates effectively as a 32-bit register. Values written to the IO_EIR are interpreted as 5-bit numbers, which cause one of the bits in the range {32..63} to be set to 1. When the W bit is 1, the EIRR operates as a 64-bit register. Values written to the IO_EIR are interpreted as 6-bit numbers, which cause one of the bits in the range {0..63} to be set to 1.

Temporary Registers
Six of the eight 64-bit temporary registers (CRs 24, 25, 28..31) are accessible only by code executing at the most privileged level. They provide space to save the contents of the general registers for interruption handlers in the operating system kernel.

The other two temporary registers (CRs 26 and 27) are readable by code executing at any privilege level and writable only by code executing at the most privileged level.

Unused Registers and Bits
Currently, there are several registers and bit-fields within registers that do not have any function assigned to them. All such processing resources are classified into four categories:

1. Reserved bits — Currently unused bits, but reserved for possible future use. A READ operation is legal, and the value read back is all zeros. A WRITE operation is legal but the value written must be all zeros. Writing ones is an undefined operation. (For example, writing ones may cause these bits to no longer read as zeros.)

2. Nonexistent bits — Architecturally these bits do not exist. A READ operation is legal and may return zeros or what was last written. A WRITE operation is also legal but does not have any effect on system functionality.

3. Undefined bits — Architecturally these bits are undefined. A READ operation is legal and the value read is undefined. A WRITE operation is also legal but does not have any effect on system functionality.

4. Ignored bits — Architecturally these bits are ignored. A READ operation is legal and the value read is all zeroes. A WRITE operation is also legal but does not have any effect on system functionality.

5. Reserved registers — A register that is numbered but currently unused. Both READ and WRITE operations are undefined operations.
Data Types

The fundamental data types that are recognized are bits, bytes, integers, floating-point numbers, and decimal numbers. Their formats are described briefly in this section. Each item of data is addressed by its lowest-numbered byte.

Bits
Memory is not addressed to the resolution of bits; however, efficient support is provided to manipulate and test individual bits in the general registers.

Bytes
Bytes are signed or unsigned 8-bit quantities:

- Signed Byte
  - s
  - value
  - 1
  - 7

- Unsigned Byte
  - value
  - 8

Bytes are packed four to a word and may represent a two’s complement signed value in the range -128 through +127, an unsigned value in the range 0 through 255, an arbitrary collection of eight bits, or an ASCII character.

Integers
Integers may be 16, 32, or 64 bits wide, signed or unsigned:

- Signed Halfword
  - s
  - value
  - 1
  - 15

- Unsigned Halfword
  - value
  - 16

- Signed Word
  - s
  - value
  - 1
  - 31

- Unsigned Word
  - value
  - 32

- Signed Doubleword
  - s
  - value
  - 1
  - 63
Signed integers are in two’s complement form. Halfword integers can be stored in memory only at even byte addresses, word integers only at addresses evenly divisible by four, and doubleword integers only at addresses evenly divisible by eight.

Floating-Point Numbers

The binary floating-point number representation conforms to the ANSI/IEEE 754-1985 standards. Single-word (32-bit), double-word (64-bit), and quadruple-word (128-bit) binary formats are supported.

Single-precision floating-point numbers must be aligned on word boundaries. Double-precision and quad-precision numbers must be aligned on doubleword boundaries. See Chapter 8, “Floating-point Coprocessor”, for detailed information on the floating-point formats.

Packed Decimal Numbers

Packed decimal data is always aligned on a word boundary. It consists of 7, 15, 23, or 31 BCD digits, each four bits wide and having a value in the range of 0x0 to 0x9, followed by a 4-bit sign as shown in the following figure:

<table>
<thead>
<tr>
<th>MSD</th>
<th>• • •</th>
<th>LSD</th>
<th>sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The standard sign for a positive number is 0xC, but any value except 0xD will be interpreted as positive. 0xD indicates a minus sign for a negative number. 0xB is not supported as an alternative minus sign.

Byte Ordering (Big Endian/Little Endian)

The optional E-bit in the PSW controls whether loads and stores use big endian or little endian byte ordering. When the E-bit is 0, all larger-than-byte loads and stores are big endian — the lower-addressed bytes in memory correspond to the higher-order bytes in the register. When the E-bit is 1, all larger-than-byte loads and stores are little endian — the lower-addressed bytes in memory correspond to the lower-order bytes in the register. Load byte and store byte instructions are not affected by the E-bit. The E-bit also affects instruction fetch.

Processors which implement the PSW E-bit must also provide an implementation-dependent, software writable default endian bit. The default endian bit controls whether the PSW E-bit is set to 0 or 1 on interruptions and also controls whether data in the page table is interpreted in big endian or little endian format by the hardware TLB miss handler, if implemented (See “Hardware TLB Miss Handling” on page F-3).

Figure 2-15 shows various loads in big endian format. Figure 2-16 shows various loads in little endian format.
format. Stores are not shown but behave similarly.

The E-bit also affects instruction fetch. When the E-bit is 0, instruction fetch is big endian — the lower-addressed bytes in memory correspond to the higher-order bytes in the instruction. When the E-bit is 1, instruction fetch is little endian — the lower-addressed bytes in memory correspond to the lower-order bytes in the instruction.

Figure 2-15. Big Endian Loads

Figure 2-16. Little Endian Loads
Architecturally, the instruction byte swapping can occur either when a cache line is moved into the
instruction cache (I-cache) or as instructions are fetched from the I-cache into the pipeline.
Processors must support running code with either endian form from the same cache line. This relieves
software of the responsibility of keeping track of what might have been brought in under different
forms.

---

**Engineering Note**

For processors which swap instructions on I-cache move-in, one way to meet this requirement is to implement endian tag bits and force a miss if the tag does not match the current value of PSW[E].

Processors which swap instructions as they are fetched from the I-cache do not need to do anything extra to meet this requirement.

---

Since the PSW E-bit is an instruction fetch resource (see “Instruction Pipelining” on page 4-9), SET SYSTEM MASK, RESET SYSTEM MASK or MOVE TO SYSTEM MASK instructions which change the PSW E-bit must be followed by seven palindromic NOP instructions — that is, instructions which are NOPs when interpreted in either big or little endian order

---

**Programming Note**

One example of a palindromic NOP instruction is LDI 26,0 (opcode 0x34000034)