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 5: Interruptions
Interruptions are anomalies that occur during instruction processing, causing transfer of the flow control to an interruption handling routine. In the process, the hardware automatically saves certain processor state. Upon completion of interruption processing, a RETURN FROM INTERRUPTION instruction is executed, which restores the saved processor state, and the execution proceeds with the interrupted instruction.

From the viewpoint of response to interruptions, the processor behaves as if it were not pipelined. That is, it behaves as if a single instruction is fetched and executed, and any interruption conditions raised by that instruction are handled at that time. If there are none, the next instruction is fetched, and so on.

**Interrupt Classes**

Faults, traps, interrupts, and checks are the different classes of interruptions that may happen during instruction processing. Definitions of the four classes of interruptions are as follows:

- **Fault**: The current instruction requests a legitimate action which cannot be carried out due to a system problem, such as the absence of a page from main memory. After the system problem has been corrected, the faulting instruction will execute normally. Faults are synchronous with respect to the instruction stream.

- **Trap**: Traps include two sorts of possibilities: either the function requested by the current instruction cannot or should not be carried out, or system intervention is desired by the user before or after the instruction is executed. Examples of the first type include arithmetic operations that result in signed overflow and instructions executed with insufficient privilege for their intended function. Such instructions are normally not re-executed. Examples of the second type include the debugging support traps. Traps are synchronous with respect to the instruction stream.

- **Interrupt**: An external entity (for example, an I/O device or the power supply) requires attention. Interrupts are asynchronous with respect to the instruction stream.

- **Check**: The processor has detected an internal malfunction. Checks can be either synchronous or asynchronous with respect to the instruction stream.

All four classes of interruptions are handled in the same way. The interruptions are categorized into four groups based on their priorities:

<table>
<thead>
<tr>
<th>Group 1:</th>
<th>1 High-priority machine check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 2:</td>
<td>2 Power failure interrupt</td>
</tr>
<tr>
<td></td>
<td>3 Recovery counter trap</td>
</tr>
<tr>
<td></td>
<td>4 External interrupt</td>
</tr>
<tr>
<td></td>
<td>5 Low-priority machine check</td>
</tr>
<tr>
<td></td>
<td>29 Performance monitor interrupt</td>
</tr>
</tbody>
</table>
The interruption numbers in the above list are the individual vector numbers that determine which interruption handler is invoked for each interruption. The group numbers determine when the particular interruption will be processed during the course of instruction execution. The order the interruptions are listed within each group (not the interruption numbers) determines the priority of simultaneous interruptions (from highest to lowest).

**Interruption Handling**

Interruption handling is implemented as a fast context switch (which is much simpler than a complete process swap). When an interruption occurs, the hardware takes the following actions:

1. The PSW in effect at the time of the interruption is saved in the IPSW. For group 2 and 3 interruptions, the saved PSW is the value at the beginning of execution. For group 4 interruptions, the saved PSW is the value after the execution of the instruction.

2. The defined bits in the PSW are set as follows:
   - W: Set to the value of the default width bit.
   - E: Set to the value of the default endian bit.
M  Set to 1 if the interruption is a high-priority machine check; otherwise, set to 0.
all other bits  Set to 0 (interrupts are masked, absolute accesses are enabled, etc.).

3. IA information in the IIA queues is frozen (as a result of setting the PSW Q-bit to 0 in step 2 above).

In order to enable restarting of instructions in the presence of delayed branching, at least two addresses must be saved, pointing to the next two instructions to be executed after returning from the interruption. The hardware, therefore, maintains IIA Space and IIA Offset queues, which have two elements and contain the addresses and privilege levels of these instructions. The IIA queues are kept up-to-date whenever the Q-bit in the PSW is 1. When an interruption is taken, the addresses of the pending instructions are preserved in the queues. The elements of the queues may be obtained by reading the IIASQ and IIAOQ registers (CRs 17 and 18, respectively).

4. The current privilege level is set to the highest privilege level (zero).

5. Information about the interrupting instruction is saved in the Interruption Parameter Registers (IPRs) if the PSW Q-bit was 1 at the time of the interruption. If the PSW Q-bit was 0, the IPRs are unchanged. If the details of an instruction associated with the interruption are potentially useful in processing it, the instruction is loaded into the Interruption Instruction Register (IIR or CR 19). If there is an address associated with the interruption, it is loaded into the Interruption Space and Interruption Offset registers (ISR or CR 20, and IOR or CR 21). See “Interruption Parameter Registers (IPRs)” on page 2-15 for a description of the format of these registers.

6. General registers 1, 8, 9, 16, 17, 24, and 25 are copied to the shadow registers if the PSW Q-bit was 1 at the time of the interruption. If the PSW Q-bit was 0, the shadow registers are unchanged.

7. Execution begins at the address given by:

   \[ \text{Interruption Vector Address} + (32 \times \text{interruption_number}) \]

   \text{Interruption_number} is the unique integer value assigned to that particular interruption. Vectoring is accomplished by performing an indexed branch into the Interruption Vector Table indexed by this integer. The Interruption Vector Table contains the first eight instructions of each of the interruption handling routines. The value in the Interruption Vector Address register (CR 14) must be aligned on a 2 Kbyte boundary.

---

**Programming Note**

It is the responsibility of interruption handlers to unmask external interrupts (by setting the PSW I-bit to 1) as soon as possible, so as to minimize the worst-case latency of external interrupts.

---

**Instruction Recoverability**

When execution of instructions is interrupted, the minimal processor state that is required to be saved and restored is that necessary to correctly continue execution of the instruction stream after processing of the interruption. Processor state is defined to include any register contents, PSW bits, or other information that may affect the operation performed by an instruction. For example, if an interruption is...
taken immediately before an ADD instruction, its source registers must be restored, but its target register need not be (unless it is also one of the source registers).

**Masking and Nesting of Interruptions**

Disabling an interruption prevents it from occurring. The interruption does not wait until re-enabled. It is not kept pending. Masking an interruption does not prevent the recognition of a pending interruption condition, but delays the occurrence of the interruption until it is “unmasked”.

The IA state is collected in the IIA queues only while the PSW Q-bit is 1; it is usually not possible to resume execution after an interruption which is taken while the PSW Q-bit is 0.

The machine state is saved in registers rather than memory when an interruption occurs, and interruption handlers must leave interruptions disabled until they have saved the machine state in memory. Once the machine state is saved, nested interrupts can be allowed.

Since it is desirable to catch hardware faults as soon as possible, interruption handlers should generally not mask high-priority machine checks. If a machine check occurs before the machine state has been saved, the interrupted process may need to be aborted. The occurrence of traps and faults within interruption handlers can be avoided by careful writing of the handlers.

**Interruption Priorities**

High-priority machine checks (which belong to Group 1) may occur and be processed at any time. They may be synchronous or asynchronous with instruction processing, may be associated with more than one instruction, and their precise meaning and processing is implementation dependent.

All interruptions other than high-priority machine checks are taken between instructions. Multiple simultaneous interruptions may occur because a number of instructions are capable of raising several synchronous interruptions simultaneously, and because certain interruptions are asynchronous with respect to the instruction stream.

Group 2 interruptions occur asynchronously with respect to the instruction stream.

Group 3 interruptions are synchronous with respect to the instruction stream and are signalled before completion of the instruction that produces them.

Group 4 interruptions are synchronous with respect to the instruction stream and are signalled either after completion of the instruction that causes them, or when a change in privilege level is about to happen.

Relative priorities are not assigned to the 64 external interrupts by the hardware. When multiple external interrupts occur simultaneously, software may select their order of service, based on the contents of EIR.

**Return from Interruption**

The RETURN FROM INTERRUPTION instruction restores the PSW and the instruction address queues. If the old PSW stored in IPSW (CR 22) has interruptions enabled (or unmasked), interruptions are re-enabled before execution of the first of the continuation instructions. The PSW Q-bit may reliably be set
to 1 only by a RETURN FROM INTERRUPTION instruction. An attempt to set the PSW Q-bit to 1 with a
SET SYSTEM MASK or MOVE TO SYSTEM MASK instruction is an undefined operation.

Adding the “, R” (restore) completer to the RETURN FROM INTERRUPTION instruction does everything
that a normal RETURN FROM INTERRUPTION instruction does, and in addition causes the values in the
shadow registers to be copied to GRs 1, 8, 9, 16, 17, 24, and 25. Execution of a RETURN FROM
INTERRUPTION with the “, R” completer leaves the contents of the shadow registers undefined.

Executing a RETURN FROM INTERRUPTION instruction with the PSW Q-bit 0 and the IPSW Q-bit 0
leaves the IPRs unchanged.

<table>
<thead>
<tr>
<th>Using RFI</th>
<th>Using RFI,R</th>
</tr>
</thead>
<tbody>
<tr>
<td>interrupt</td>
<td>interrupt</td>
</tr>
<tr>
<td>save GRs</td>
<td>&lt;no save&gt;</td>
</tr>
<tr>
<td>[process interrupt]</td>
<td>[process interrupt]</td>
</tr>
<tr>
<td>restore GRs</td>
<td>&lt;no restore&gt;</td>
</tr>
<tr>
<td>RFI</td>
<td>RFI,R</td>
</tr>
</tbody>
</table>

Programming Note

Only those interruptions which are themselves uninterruptible (they leave the PSW Q-bit 0)
may return from the interruption using the RFI,R instruction. Interruption handling code which
is interruptible (they set the PSW Q-bit to 1) must return from the interruption using the RFI
instruction.

Fast interruption handling is achieved using shadow registers, since GRs 1, 8, 9, 16, 17, 24, and
25 are copied to the shadow registers on interruptions. In this example, it is assumed that at
most seven general registers need to be used in the interruption handling routine.

Interruption Descriptions

The sections that follow provide descriptions of each of the interrupts defined in the PA-RISC
architecture.

Group 1 Interruptions

High-priority Machine Check (1)

Cause: A hardware error has been detected that must be handled before processing can continue

Parameters: Implementation dependent

IIA Queue: Front – Implementation dependent
            Back – Implementation dependent

Notes: The actions taken when a hardware error is detected depend on the seriousness of the
error. Damage extensive enough to prevent proper execution of instructions will halt the
machine and generate an external indication of the occurrence of the check. Damage which allows a subset of the instructions to execute (e.g., inoperative TLB) generates a high-priority machine check interruption. This is maskable by setting the PSW M-bit to 1, so that machine checks within the machine check handler can be prevented. The causes of high-priority machine checks are implementation dependent, as is the means of controlling their reporting.

Group 2 Interruptions

Power Failure Interrupt (2)

<table>
<thead>
<tr>
<th>Cause:</th>
<th>The machine is about to lose power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>none</td>
</tr>
<tr>
<td>IIA Queue:</td>
<td>Front – Address of the instruction to be executed at the time of the interruption Back – Address of the following instruction</td>
</tr>
<tr>
<td>Notes:</td>
<td>This interruption is masked and kept pending when the PSW I-bit is 0.</td>
</tr>
</tbody>
</table>

Recovery Counter Trap (3)

<table>
<thead>
<tr>
<th>Cause:</th>
<th>Bit 0 of the recovery counter is 1 and the PSW R-bit is 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>none</td>
</tr>
<tr>
<td>IIA Queue:</td>
<td>Front – Address of the instruction to be executed at the time of the interruption Back – Address of the following instruction</td>
</tr>
<tr>
<td>Notes:</td>
<td>The recovery counter can be used to log interruptions during normal operation and to simulate interruptions during recovery from a fault.</td>
</tr>
</tbody>
</table>

External Interrupt (4)

<table>
<thead>
<tr>
<th>Cause:</th>
<th>A module writes to the processor’s IO_EIR register, or the interval timer compares equal to its associated comparison register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>none</td>
</tr>
<tr>
<td>IIA Queue:</td>
<td>Front – Address of the instruction to be executed at the time of the interruption Back – Address of the following instruction</td>
</tr>
<tr>
<td>Notes:</td>
<td>Each external interrupt level has associated with it one bit in the External Interrupt Enable Mask Register (CR 15) and one bit in the External Interrupt Request Register (CR 23). When a module writes into the EIR register, the bit position corresponding to the value written is set to 1. If the default width bit is 1, the bit to set directly corresponds to the value; if 0, the bit to set is the value + 32. For example if the value 5 is written, then bit 5 of the EIR register is set to 1 if the default width bit is 1, and bit 37 of the EIR is set if the default width bit is 0. If the corresponding bit in CR 15 is 1 and the PSW I-bit is 1, an external interrupt is taken; otherwise, the interrupt is masked, and is kept pending. Interrupt handling software sets bits in the EIR to 0 by executing a MOVE TO CONTROL</td>
</tr>
</tbody>
</table>
REGISTER instruction with the appropriate mask.
If multiple sources can set the same interrupt, it is the responsibility of software to correctly respond to all of the interrupting sources.

### Low-priority Machine Check (5)

**Cause:** A hardware error has been detected which is recoverable and does not require immediate handling

**Parameters:** Implementation dependent

**IIA Queue:**
- Front – Address of the instruction to be executed at the time of the interruption
- Back – Address of the following instruction

**Notes:** Errors which have been detected and recovered from by hardware to the point that operation can continue in a degraded fashion are reported via the low-priority machine check interruption. This interruption is masked and kept pending when the PSW I-bit is 0. The causes of low-priority machine checks are implementation dependent, as is the means of controlling their reporting.

### Performance Monitor Interrupt (29)

**Cause:** An implementation-dependent event related to the performance monitor coprocessor requires software intervention

**Parameters:** Implementation dependent

**IIA Queue:**
- Front – Address of the instruction to be executed at the time of the interruption
- Back – Address of the following instruction

**Notes:** This interruption is masked and kept pending when the PSW F-bit is 0.

### Group 3 Interruptions

#### Instruction Tlb Miss Fault/instruction Page Fault (6)

**Cause:** The instruction TLB entry needed by instruction fetch is absent, and if instruction TLB misses are handled by hardware, the hardware miss handler could not find the translation in the Page Table

**Parameters:** none

**IIA Queue:**
- Front – Address of the instruction causing the fault
- Back – Address of the following instruction

**Notes:** Only if an instruction is to be executed can an instruction TLB miss fault occur.

#### Instruction Memory Protection Trap (7)

**Cause:** Instruction address translation is enabled and the access rights check fails for an instruction fetch or instruction address translation is enabled, the PSW P-bit is 1, and the protection identifier checks fails for an instruction fetch
Parameters: none

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Notes: This interruption does not occur for absolute accesses.

Illegal Instruction Trap (8)

Cause: An attempt is being made to execute an illegal instruction or to execute a BRANCH with GATE instruction with the PSW B-bit equal to 1

Parameters: IIR – The illegal instruction causing the trap

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Notes: Illegal instructions are the unassigned major opcodes. Unassigned sub-opcodes are undefined operations (undefined sub-opcodes may cause the illegal instruction trap). On some implementations, DIAGNOSE may be an illegal instruction.

Break Instruction Trap (9)

Cause: An attempt is made to execute a BREAK instruction

Parameters: IIR – The BREAK instruction causing the trap

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Privileged Operation Trap (10)

Cause: An attempt is being made to execute a privileged instruction without being at the most privileged level (priv= 0)

Parameters: IIR – The privileged instruction causing the trap

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Notes: The list of privileged instructions is: DIAG, IDTLBT, HITLBT, LCI, LDDA, LDWA, LPA, MTSM, PDTLB, PDTLBE, PITLB, PITLBE, RFI, RSM, SSM, STDA, STWA.

Privileged Register Trap (11)

Cause: An attempt is being made to write to a privileged space register or access a privileged control register without being at the most privileged level (priv= 0)

Parameters: IIR – The instruction causing the trap

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Notes: This interruption may be caused by the MOVE TO SPACE REGISTER, MOVE TO CONTROL REGISTER, or MOVE FROM CONTROL REGISTER instructions.
Overflow Trap (12)

Cause: A signed overflow is detected in an instruction which traps on overflow

Parameters: IIR – The instruction causing the trap

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Conditional Trap (13)

Cause: The condition succeeds in an instruction which traps on condition

Parameters: IIR – The instruction causing the trap

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Assist Exception Trap (14)

Cause: A coprocessor or special function unit has detected an exceptional condition or operation.
An exceptional operation may include unimplemented operations or operands.

Parameters: IIR – For immediate traps, the SFU or coprocessor instruction that was executing when an exception is reported with a trap. It may or may not be related to the condition causing the exception. For delayed traps, any instruction corresponding to the SFU or coprocessor.
See “Interruptions and Exceptions” on page 10-4.

IIA Queue: Front – Address of the instruction causing the trap
Back – Address of the following instruction

Data Tlb Miss Fault/data Page Fault (15)

Cause: The data TLB entry needed by operand access of a load, store, or semaphore instruction is absent, and if data TLB misses are handled by hardware, the hardware miss handler could not find the translation in the Page Table

Parameters: ISR – space identifier of data address
IOR – offset of data address
IIR – The instruction causing the fault

IIA Queue: Front – Address of the instruction causing the fault
Back – Address of the following instruction

Notes: This interruption does not occur for absolute accesses.

Non-access Instruction Tlb Miss Fault (16)

Cause: The instruction TLB entry needed for the target of a FLUSH INSTRUCTION CACHE instruction is absent, and if TLB misses are handled by hardware, the hardware miss handler could not find the translation in the Page Table

Parameters: ISR – space identifier of virtual address to be flushed
IOR – offset of virtual address to be flushed
IIR – The instruction causing the fault

IIA Queue:  
Front – Address of the instruction causing the fault
Back – Address of the following instruction

Notes:  
This interruption source is distinguished from other TLB misses because a page fault should not result in reading the faulting page from disk. This interruption does not occur for absolute accesses.

Non-access Data Tlb Miss Fault/non-access Data Page Fault (17)

Cause:  
The data TLB entry needed by a LOAD PHYSICAL ADDRESS, PROBE ACCESS, PROBE ACCESS IMMEDIATE, FLUSH INSTRUCTION CACHE, PURGE DATA CACHE, or a FLUSH DATA CACHE instruction is not present, and if TLB misses are handled by hardware, the hardware miss handler could not find the translation in the Page Table

Parameters:  
ISR – space identifier of virtual address
IOR – offset of virtual address
IIR – The instruction causing the fault

IIA Queue:  
Front – Address of the instruction causing the fault
Back – Address of the following instruction

Notes:  
These interruption sources are distinguished from other TLB misses because a page fault should not result in reading the faulting page from disk. This interruption does not occur for absolute accesses.

Data Memory Access Rights Trap (26)

Cause:  
Data address translation is enabled, and an access rights check fails on an operand reference for a load, store, or semaphore instruction, or a cache purge operation

Parameters:  
ISR – space identifier of the virtual address
IOR – offset of the virtual address
IIR – The instruction causing the trap

IIA Queue:  
Front – Address of the instruction causing the trap
Back – Address of the following instruction

Notes:  
This interruption does not occur for absolute accesses.

Data Memory Protection Id Trap (27)

Cause:  
Data address translation is enabled, the PSW P-bit is 1, and a protection identifier check fails on an operand reference for a load, store, or semaphore instruction, or a cache purge operation

Parameters:  
ISR – space identifier of the virtual address
IOR – offset of the virtual address
IIR – The instruction causing the trap

IIA Queue:  
Front – Address of the instruction causing the trap
Back – Address of the following instruction
Notes: This interruption does not occur for absolute accesses.

**Unaligned Data Reference Trap (28)**

**Cause:** Data address translation is enabled, and a load or store instruction is attempted to an unaligned address

**Parameters:**
- ISR – space identifier of the virtual address
- IOR – offset of the virtual address
- IIR – The instruction causing the trap

**IIA Queue:**
- Front – Address of the instruction causing the trap
- Back – Address of the following instruction

**Notes:** Unaligned data reference traps are not detected for absolute accesses or semaphore instructions – they are undefined operations. Only unaligned virtual memory loads and stores (including coprocessor loads and stores) are defined to terminate with the unaligned data reference trap.

**Data Memory Protection Trap/unaligned Data Reference Trap (18)**

**Cause:** Data address translation is enabled, and an access rights check or a protection identifier check fails on an operand reference for a load, store, or semaphore instruction, or a cache purge operation; a load or store instruction is attempted to an unaligned address with virtual address translation enabled (unaligned absolute references and semaphore instructions are undefined operations)

**Parameters:**
- ISR – space identifier of the virtual address
- IOR – offset of the virtual address
- IIR – The instruction causing the trap

**IIA Queue:**
- Front – Address of the instruction causing the trap
- Back – Address of the following instruction

**Notes:** This interruption does not occur for absolute accesses. Only unaligned virtual memory loads and stores (including coprocessor loads and stores) are defined to terminate with the data memory protection trap. Execution of a semaphore instruction with unaligned (16 byte boundaries) addresses is an undefined operation.

This trap is retained for compatibility with the earlier revisions of the architecture. In PA-RISC 1.1 (Second Edition) and later revisions, processors must use traps 26, 27, and 28 which provide equivalent functionality.

**Data Memory Break Trap (19)**

**Cause:** Store and semaphore instructions or cache purge operations to a page with the B-bit 1 in the data TLB entry

**Parameters:**
- ISR – space identifier of the virtual address
- IOR – offset of the virtual address
- IIR – The instruction causing the trap

PA-RISC 2.0 Architecture

Interruptions 5-11
IIA Queue:  Front – Address of the instruction causing the trap
            Back – Address of the following instruction
Notes:  This trap is disabled if the PSW X-bit is 1. This interruption does not occur for absolute accesses.

**Tlb Dirty Bit Trap (20)**

**Cause:**  Store and semaphore instructions to a page with the D-bit 0 in the data TLB entry

**Parameters:**
- ISR – space identifier of the data address
- IOR – offset of the data address
- IIR – The instruction causing the trap

**IIA Queue:**  Front – Address of the instruction causing the trap
            Back – Address of the following instruction

**Notes:**  Software is invoked to update the dirty bit in the data TLB entry and the Page Table. This interruption does not occur for absolute accesses.

**Page Reference Trap (21)**

**Cause:**  Load, store, and semaphore instructions to a page with the T-bit 1 in its data TLB entry

**Parameters:**
- ISR – space identifier of the virtual address
- IOR – offset of the virtual address
- IIR – The instruction causing the trap

**IIA Queue:**  Front – Address of the instruction causing the trap
            Back – Address of the following instruction

**Notes:**  This interruption does not occur for absolute accesses.

**Assist Emulation Trap (22)**

**Cause:**  An attempt is being made to execute an SFU instruction for an SFU whose corresponding bit in the SFU Configuration Register (SCR) is 0 or to execute a coprocessor instruction for a coprocessor whose corresponding bit in the Coprocessor Configuration Register (CCR) is 0

**Parameters:**
- ISR – space identifier of the data address
- IOR – offset of the data address
- IIR – The instruction causing the trap

**IIA Queue:**  Front – Address of the instruction causing the trap
            Back – Address of the following instruction

**Notes:**  ISR and IOR contain valid data only if the instruction is a coprocessor load or store.
Group 4 Interruptions

Higher-privilege Transfer Trap (23)

Cause: An instruction is about to be executed at a higher privilege level than the instruction just completed and the PSW H-bit is 1

Parameters: none

IIA Queue: Front – Address of the instruction with the higher privilege level
           Back – Address of the following instruction

Lower-privilege Transfer Trap (24)

Cause: An instruction is about to be executed at a lower privilege level than the instruction just completed and the PSW L-bit is 1

Parameters: none

IIA Queue: Front – Address of the instruction with the lower privilege level
           Back – Address of the following instruction

Taken Branch Trap (25)

Cause: A taken branch was executed, and the PSW T-bit is 1

Parameters: none

IIA Queue: Front – Address of the instruction to be executed after the branch
           Back – Address of the branch target

Notes: This interruption occurs after the execution of the branch instruction, and the address of the branch instruction itself is not available. The address at the front of the IIA queue is the address of the instruction to be executed next. If the branch has nullification specified, this is the address of the nullified instruction (the PSW N-bit is 1 in this case).