# A walk through interrupts on the PPC 823

# Review – software viewpoint

- Interrupt occurs
  - Processor saves limited state
    - SRR0 – Next instruction to be executed when return from interrupt.
    - SRR1 – Copy of MSR
    - MSR is modified. Most important is EE bit=0
  - Processor jumps to the Interrupt Service Routine.
  - ISR executes.
    - Needs to save state so that it can put it all back before return from interrupt.
    - *May enable nested interrupts*
    - Does its thing ← **<u>This is a big step…</u>**
    - Restores state (*disables nested interrupts if needed!)*
    - Return from interrupt (rfi instruction)

# Details of software viewpoint

- How does the ISR know where to branch to?
  - Table 7.1 indicates the low 5-hex digits of the address.
  - The IP bit of the MSR (page 6-21) determines the high order bits.
- But even then, we need to c different things for different interrupt lines.
  - SIVEC…

Table 7-1. Offset of First Instruction by Interrupt Type

| OFFSET (HEX) | INTERRUPT TYPE |
|---|---|
| 00000 | Reserved |
| 00100 | System Reset |
| 00200 | Machine Check |
| 00300 | Data Storage |
| 00400 | Instruction Storage |
| 00500 | External |
| 00600 | Alignment |

## Table 7-1. Offset of First Instruction by Interrupt Type

| OFFSET (HEX) | INTERRUPT TYPE |
|---|---|
| 00000 | Reserved |
| 00100 | System Reset |
| 00200 | Machine Check |
| 00300 | Data Storage |
| 00400 | Instruction Storage |
| 00500 | External |
| 00600 | Alignment |

# So how do we run the right code?

- SIVEC holds an 8 bit code which indicates which interrupt was the highest level external interrupt.
  - Page 12-6.
  - This can be used to figure out which “line” is the highest priority interrupt.
- While a switch statement would work, it is fairly ugly.
  - Rather we use an indirect branch.
  - Sample code is on page 12-11

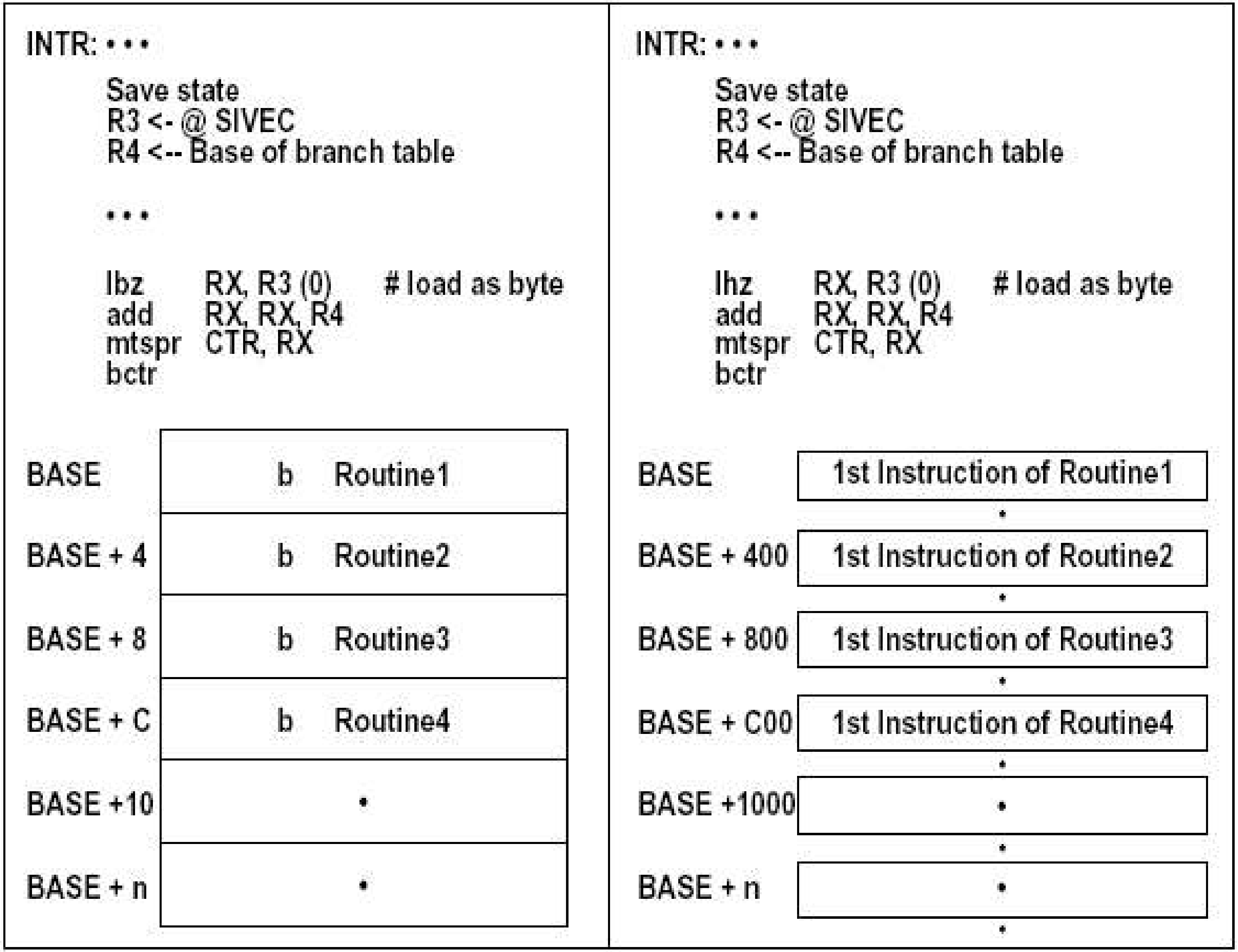



**Figure 12-3. Interrupt Table Handling Example**

# Ok, so we've got the software mostly down…

- Some issues
  - Changing the MSR could be important.
  - We need to be sure we save *everything* we change.
    - Don't forget the condition register.
  - Not sure what to do to **stop** the interrupt
    - We need to clear something somewhere…
- So onto hardware.

# MPC 823 -- hardware

- So how about from the hardware side?
  - We’ve already seen one special-purpose register (SIVEC)
  - It turns out there are a few more
    - SIPEND is a vector of pending interrupts
    - SIMASK is a vector that allows one to mask out certain interrupts.
    - SIEL toggles the *IRQ* interrupts from edge to level
  - We also have to understand what the different souces of interrupts might do.
    - Let’s start there.

# “External” Interrupts

- There are two basic types of external interrupts
  - Those generated off-chip and those generated on-chip.
    - Hardware devices you generate on the FPGA will all be off-chip. These use the IRQ interrupts
    - Timers and other on-chip devices use the LVL interrupts.
  - Each has somewhat different functionality.

# Clearing interrupts

- Every interrupt does/should have a single point of reset.
  - Think of it this way.  Somewhere out there is something that is saying “this interrupt is occurring—the device is asking for service”
  - The ISR needs *some way* to clear this and indicate that the device is being dealt with.
  - Doing so remove the interrupt from SIPEND.

# Clearing LVL interrupts.

- Always level-sensitive.
  - Probably why named the way they are.
- Interrupt source is either CIPR (16-497) or an event bit associated with the interrupt.
- *You do NOT clear them by writing to SIPEND!*

# Clearing IRQ interrupts

- If the IRQ is edge-triggered the SIPEND register stores the fact that an interrupt has occurred.
  - The interrupt is cleared at SIPEND in this case
- If the IRQ is level-sensitive the interrupt is cleared by direct communication to the I/O device.
  - When designing I/O devices that generate interrupts keep these things in mind.

# Interrupt related registers

- We've seen SIVEC but there are more…

# SIEL

- System Interrupt Edge/Level
  - Determines if IRQ interrupts are level or edge sensitive.
  - Also controls if IRQ can wake the processor from low-power mode.

SIEL

| BIT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIELD | ED0 | WM0 | ED1 | WM1 | ED2 | WM2 | ED3 | WM3 | ED4 | WM4 | ED5 | WM5 | ED6 | WM6 | ED7 | WM7 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | (IMMR & 0xFFFF0000) + 0x018 | | | | | | | | | | | | | | | |

# SIPEND

- Keeps track of the external interrupts that are PENDing.
  - Used to clear edge-triggered IRQ interrupts.
  - Can be used rather than SIVEC to figure out what interrupt to service if you really really want to.

SIPEND

| BIT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIELD | IRQ0 | LVL0 | IRQ1 | LVL1 | IRQ2 | LVL2 | IRQ3 | LVL3 | IRQ4 | LVL4 | IRQ5 | LVL5 | IRQ6 | LVL6 | IRQ7 | LV7 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | (IMMR & 0xFFFF0000) + 0x010 | | | | | | | | | | | | | | | |

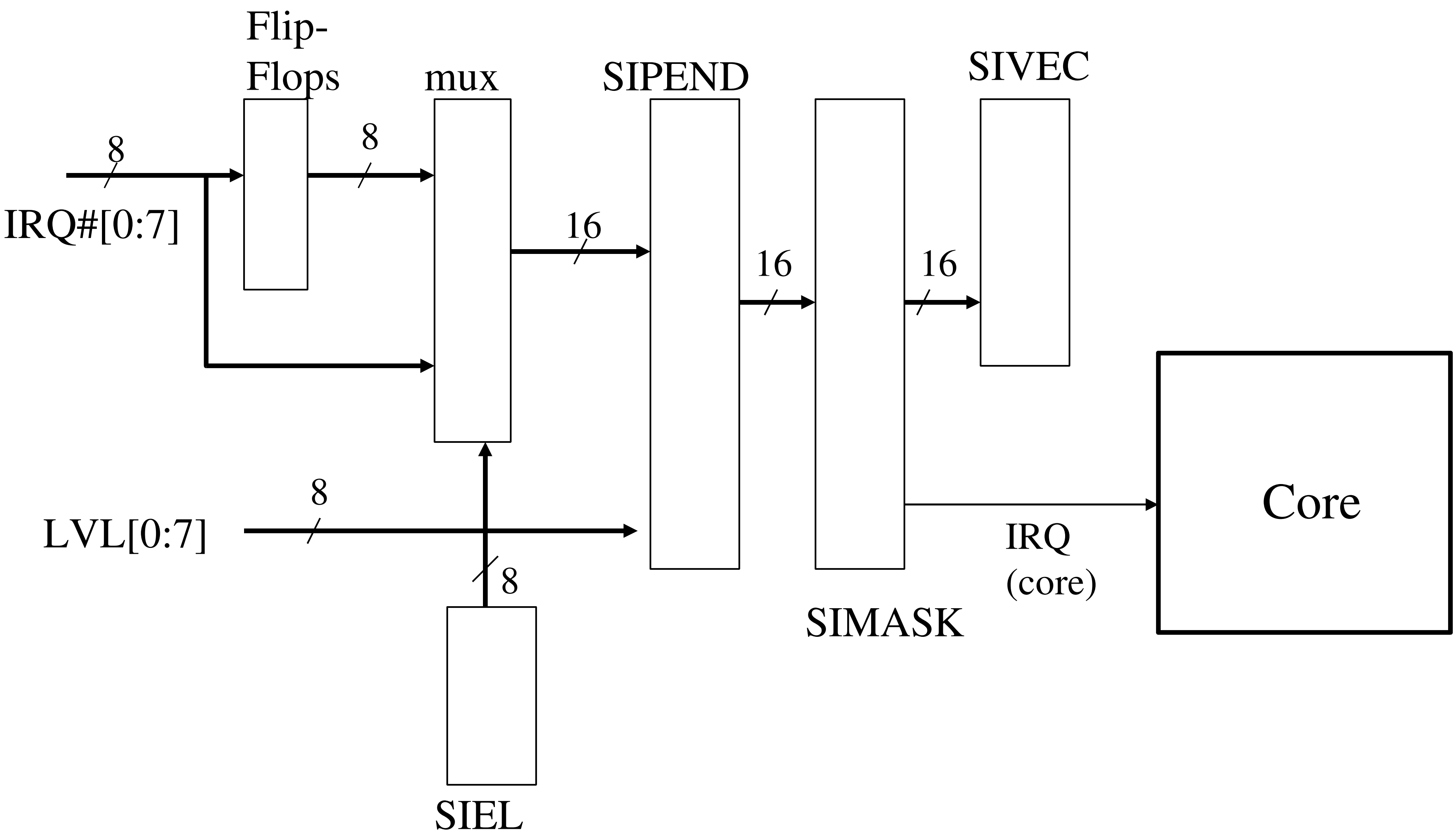
# SIMASK

- Allows certain interrupts to be ignored or masked out.
  - Is ANDed with SIPEND to determine which interrupts are allowed to interrupt the core and set their value in SIVEC

SIMASK

| BIT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIELD | IRM0 | LVM0 | IRM1 | LVM1 | IRM2 | LVM2 | IRM3 | LVM3 | IRM4 | LVM4 | IRM5 | LVM5 | IRM6 | LVM6 | IRM7 | LVM7 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | (IMMR & 0xFFFF0000) + 0x014 | | | | | | | | | | | | | | | |

# PPC: Graphical view



## Table 12-1. Priority of System Interface Unit Interrupt Sources

| NUMBER | PRIORITY LEVEL | INTERRUPT SOURCE DESCRIPTION | INTERRUPT CODE |
|---|---|---|---|
| 0 | Highest | $\overline{\text{IRQ0}}$ | 00000000 |
| 1 | | Level 0 | 00000100 |
| 2 | | $\overline{\text{IRQ1}}$ | 00001000 |
| 3 | | Level 1 | 00001100 |
| 4 | | $\overline{\text{IRQ2}}$ | 00010000 |
| 5 | | Level 2 | 00010100 |
| 6 | | $\overline{\text{IRQ3}}$ | 00011000 |
| 7 | | Level 3 | 00011100 |
| 8 | | $\overline{\text{IRQ4}}$ | 00100000 |
| 9 | | Level 4 | 00100100 |
| 10 | | $\overline{\text{IRQ5}}$ | 00101000 |
| 11 | | Level 5 | 00101100 |
| 12 | | $\overline{\text{IRQ6}}$ | 00110000 |
| 13 | | Level 6 | 00110100 |
| 14 | | $\overline{\text{IRQ7}}$ | 00111000 |
| 15 | Lowest | Level 7 | 00111100 |
| 16-31 | | Reserved | — |

SIPEND

| BIT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIELD | IRQ0 | LVL0 | IRQ1 | LVL1 | IRQ2 | LVL2 | IRQ3 | LVL3 | IRQ4 | LVL4 | IRQ5 | LVL5 | IRQ6 | LVL6 | IRQ7 | LV7 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | (IMMR & 0xFFFF0000) + 0x010 | | | | | | | | | | | | | | | |

SIMASK

| BIT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIELD | IRM0 | LVM0 | IRM1 | LVM1 | IRM2 | LVM2 | IRM3 | LVM3 | IRM4 | LVM4 | IRM5 | LVM5 | IRM6 | LVM6 | IRM7 | LVM7 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | (IMMR & 0xFFFF0000) + 0x014 | | | | | | | | | | | | | | | |

SIEL

| BIT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIELD | ED0 | WM0 | ED1 | WM1 | ED2 | WM2 | ED3 | WM3 | ED4 | WM4 | ED5 | WM5 | ED6 | WM6 | ED7 | WM7 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| ADDR | (IMMR & 0xFFFF0000) + 0x018 | | | | | | | | | | | | | | | |

Table 7-1. Offset of First Instruction by Interrupt Type

| OFFSET (HEX) | INTERRUPT TYPE |
|---|---|
| 00000 | Reserved |
| 00100 | System Reset |
| 00200 | Machine Check |
| 00300 | Data Storage |
| 00400 | Instruction Storage |
| 00500 | External |
| 00600 | Alignment |
| 00700 | Program |
| 00800 | Floating Point Unavailable |
| 00900 | Decrementer |
| 00A00 | Reserved |
| 00B00 | Reserved |
| 00C00 | System Call |
| 00D00 | Trace |
| 00E00 | Floating Point Assist |
| 01000 | Implementation-Dependent Software Emulation |
| 01100 | Implementation-Dependent Instruction TLB Miss |
| 01200 | Implementation-Dependent Data TLB Miss |
| 01300 | Implementation-Dependent Instruction TLB Error |
| 01400 | Implementation-Dependent Data TLB Error |
| 01500 - 01BFF | Reserved |
| 01C00 | Implementation-Dependent Data Breakpoint |
| 01D00 | Implementation-Dependent Instruction Breakpoint |
| 01E00 | Implementation-Dependent Peripheral Breakpoint |
| 01F00 | Implementation-Dependent Nonmaskable Development Port |