DEVICE DRIVERS AND INTERRUPTS SERVICE MECHANISM

Lesson-9: Multiple Interrupt Sources and Priority Mechanism

1. Multiple interrupt-calls

Interrupt-service calls

- There can be interrupt-service calls in case a number of higher priority interrupt sources activates in succession.
- A return from any of the ISR is to the lower priority pending ISR

Processor interrupt service mechanisms

- Certain processors permit in-between routine diversion to higher priority interrupts unless all interrupts or interrupts of priority greater than the presently running routine are masked or ISR executed DI instruction
- These processors provide in order to prevent diversion in-between the running ISR completely by provisioning for masking all interrupts by primary level bit and or DI instruction
- These processors also provide in order to prevent diversion in-between the running ISR selectively by provisioning for masking selectively the interrupt service by secondary level bits for the ISR interrupt source groups

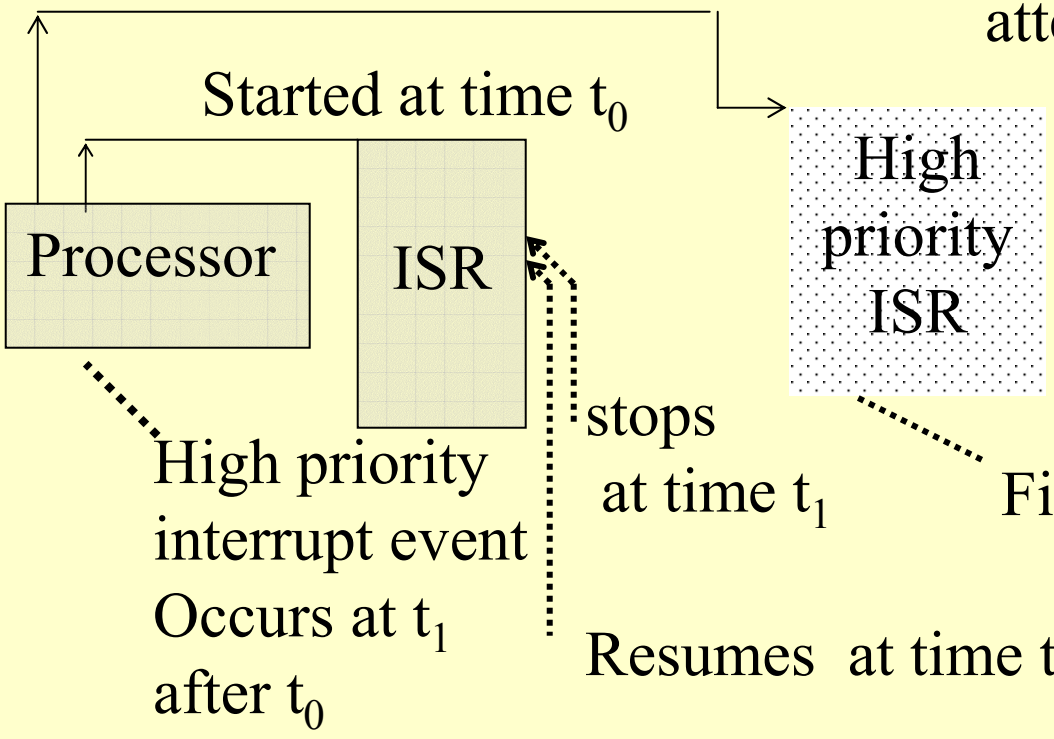
Context saving

- These processors may also provide for auto saving the CPU registers (context) when ISR execution starts and auto return of saved values into the CPU registers on return from ISR. This help in fast transfer to higher priority interrupt

In-between Diversion to higher priority interrupts from the present interrupt service routine

Starts after time t_1 + t' where t' is context switch time

ISR should be coded short to enable low priority ISRs to get attention fast



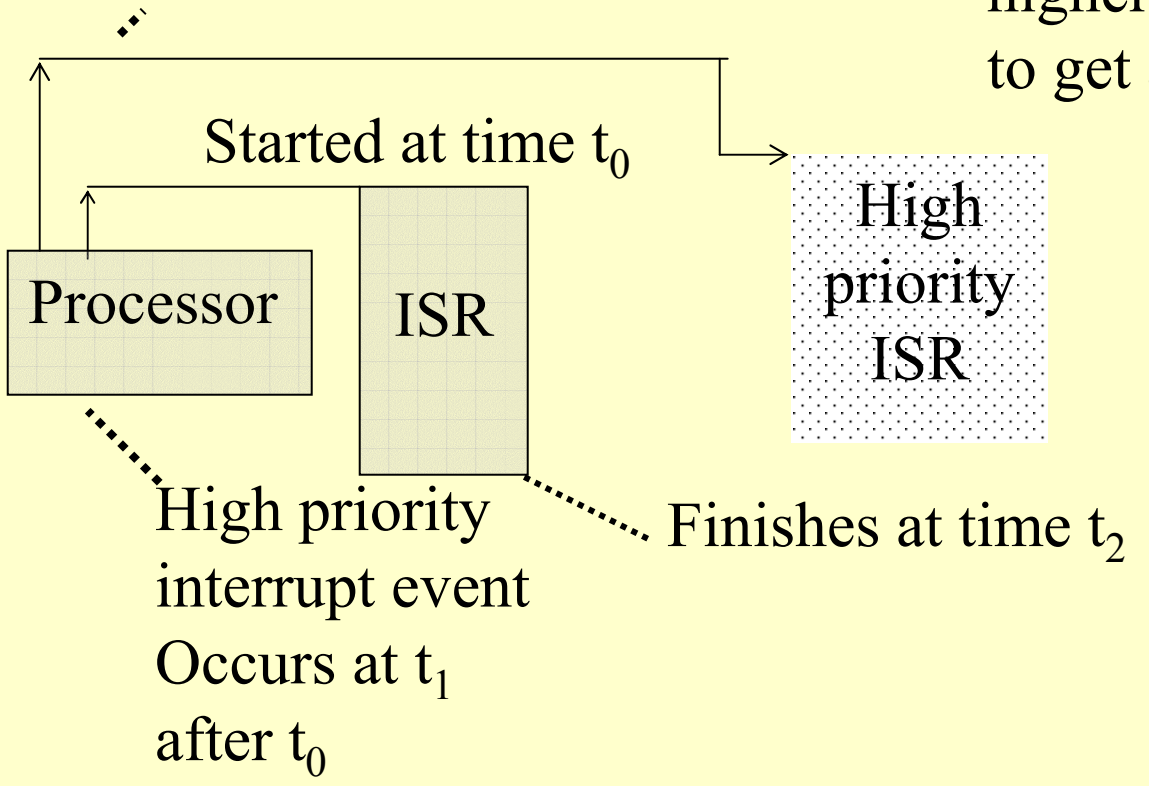
Processor interrupt service mechanisms

- Certain processors don't permit in-between routine diversion to higher priority interrupts
- These processors provide auto disabling of all maskable interrupt when ISR execution starts and auto re-enabling of all maskable interrupt on return from ISR
- These processors may also provide for auto saving the CPU registers (context) when ISR execution starts and auto return of saved values into the CPU registers on return from ISR. This help in fast transfer to pending higher priority interrupt

Diversion to higher priority interrupts at the end of the present interrupt service routine only

Starts after time t_2 + t' only where t' is context switch time

ISR should be coded short to enable higher priority ISRs to get attention fast.



2. Hardware Assignment of priorities

Need for the assigning a priority order by hardware

- Certain interrupts need fast attention
- For example, clock interrupt on a system timer overflow, detection of illegal opcode by the processor, division by 0,….
- When there are multiple device drivers, traps, exceptions, signals due to hardware and software interrupts the assignment of the priorities for each source or source group is required so that the ISRs of smaller deadline execute earlier by assigning them higher priorities Hardware-defined priorities can be used as such.

Why does the hardware assign the presumed priority?

- Several interrupts occur at the same time during the execution of a set of instructions, and either all or a few are enabled for service.
- The service using the source corresponding ISRs can only be done in a certain order of priority.
- Hardware-defined priorities can be used as such

Hardware assignment of priorities

- ARM7 provides two types of the interrupt sources (requests) – IRQs (interrupt requests) and FIQs (fast interrupt requests).
- Interrupts in 80x86 assigned interrupt-types and interrupt of type 0 has highest priority and 255 as lowest priority

Multiple sources of interrupts

- Multiple devices
- Processor hardware assigns a priority p_{hw} to each source (including traps or exceptions) or source-group a pre-assumed priority (or level or type).
- p_{hw} represents the hardware presumed priority for the source (or group)
- Assume number be among 0, 1, 2, ..., k, ..., ***m***-1.
- Let p_{hw} = 0 mean the highest; p_{hw} =1 next to lowest;.....; p_{hw} = ***m***-1 assigned the lowest.

Example of seven devices or source groups

- The processor's hardware assign $p_{hw} = 0, 1, 2, \ldots, 6$.
- The hardware service priorities will be in order $p_{hw} = 0, 1, 2, \ldots, 6$

Example of the 80x86 family processor six interrupt sources

- division by zero,
- single step,
- NMI (non maskable interrupt from RAM parity error, etc.),
- break point,
- overflow and
- print screen.
- These interrupts can be assumed to be of p_{hw} = 0, 1, 2, 3, 4 and 5, respectively.

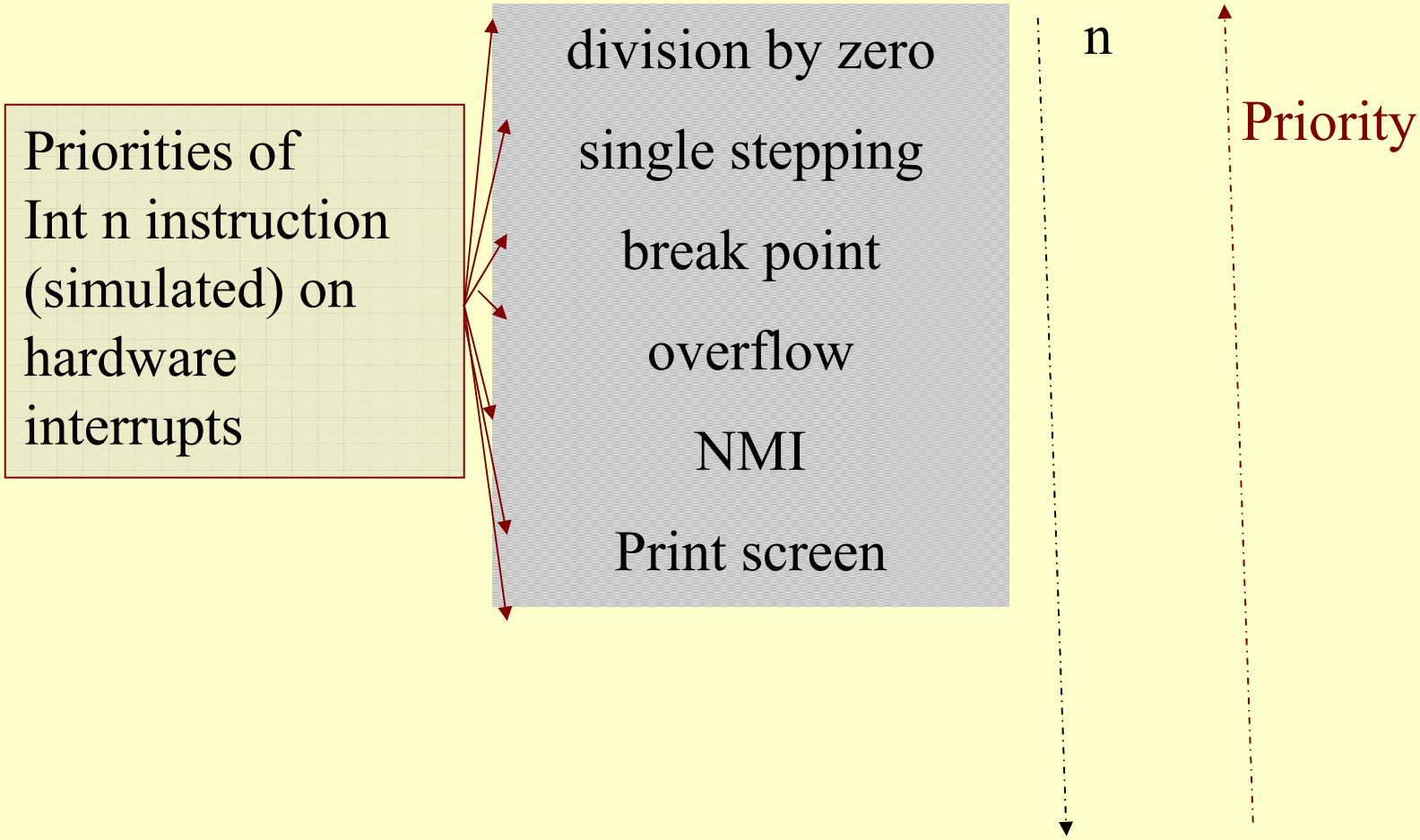
Example of the 80x86 family processor

- Assigns highest priority for a *division by zero*. This is so because it is an exceptional condition found in user software itself.
- Next priority *single stepping* as the as the user enables this source of interrupt because of the need to have break point at the end of each instruction whenever a debugging of the software is to be done.
- Next priority NMI– because external memory *read* error needs urgent attention.
- Print screen has lowest priority

Vectored priority polling method

- **A** processor interrupt mechanism may internally provide for the number of vectors, ISR_VECTADDRs
- Assigns the ISR_VECTADDR as well as p_{hw}
- There is a call at the end of each instruction cycle (or at the return from an ISR) for a highest priority source among those enabled and pending.
- Vectored priorities in 80x86 are as per the n_{type}
- n_{type} = 0 highest priority and n_{type} = 0xFF (=255) lowest priority

80x86 family processor six interrupt sources



3. Software defined priorities

Software defined priority Setting

- In certain processors, software can re-define the priorities
- Software defined priorities override the hardware ones
- 8051 has priority register to define an interrupt priority = 1 (high) or low (=0)

Summary

We learnt

- multiple device drivers, traps, exceptions, signals due to hardware and software interrupts
- Assignment of the priorities for each source or source group required so that the ISRs of smaller deadline execute earlier by assigning them higher priorities
- Hardware-defined priorities can be used as such
- Software defined priorities override the hardware ones

End of Lesson 9 of Chapter 4