DEVICE DRIVERS AND INTERRUPTS SERVICE MECHANISM

Lesson-6: Interrupt Sources

1. Hardware Sources of interrupts

Hardware device sources of interrupts

- Hardware sources can be internal or external for interrupt of ongoing routine and thereby diversion to corresponding ISR.
- The internal sources from devices differ in different processor or microcontroller or device and their versions and families
- External sources and ports also differ in different processors or microcontrollers

Interrupt sources (or groups of interrupt sources)

- Each of the interrupt sources (or groups of interrupt sources) demands a temporary transfer of control from the presently executed routine to the ISR corresponding to the source (when a source not masked)

Internal Hardware Device Sources

1 Parallel Port

2. UART Serial Receiver Port - [Noise, Overrun, Frame-Error, IDLE, RDRF in 68HC11]

3. Synchronous Receiver byte Completion

4. UART Serial Transmit Port-Transmission Complete, [For example, TDRE (transmitter data register Empty]

5. Synchronous Transmission of byte completed

6. ADC Start of Conversion

Internal Hardware Device Sources

7. ADC End of Conversion
8. Pulse-Accumulator overflow
9. Real Time Clock time-outs
10. Watchdog Timer Reset
11. Timer Overflow on time-out
12. Timer comparison with Output compare Registers
13. Timer capture on inputs

External Hardware Device interrupt with also sending vector address

- INTR in 8086 and 80x86 – The device provides the ISR Address or Vector Address or Type externally on data bus after interrupt at INTR pin

External Hardware Device interrupt with internal generation of sending vector address

- Maskable Pins (interrupt request pin)
  —INT0 and INT 1 in 8051, IRQ in 68HC11

External hardware related interrupt at INTR Pin in 80x86 processor

1. When INTR pin activates on an interrupt from the external device, the processor issues two cycles of acknowledgements in two clock cycles through INTA (interrupt acknowledgement) pin.
2. During the second cycle of acknowledgement, the external device sends the type of interrupt information on data bus.
3. Information is for one byte for *n*.
4. 80x86 action is execution as per action on software instruction INT *n*,

External Hardware Device Nonmaskable Interrupts with Internal Vector Address Generation

1. Non-Maskable Pin– NMI in 8086 and 80x86
2. Within first few clock cycles unmaskable declarable Pin (interrupt request pin) but otherwise maskable XIRQ in 68HC11]

Sources of interrupts due to Processor Hardware detecting Software error

- Software sources for interrupt are related to processor detecting computational error during execution such as division by 0, illegal opcode or overflow (for example, multiplication of two numbers exceeding the limit)

Software error Related Sources (exceptions or SW -traps)

1. Division by zero detection (or *trap*) by hardware
2. Over-flow detection by hardware
3. Under-flow detection by hardware
4. Illegal opcode detection by hardware

Examples of Software error *exception* or *trap* related sources

- Interrupt of ongoing program computations in certain processors.
- Division by zero (also known as type 0 interrupt as it is also generated by and software interrupt instruction *INT 0* instruction in 80x86)
- Overflow (also known as type 2 interrupt as it is also generated by INT 2 instruction) in 80x86. These two interrupts, types 0 and 2 generate by the hardware with the ALU part of the processor.

2. Software Interrupts

Sources of interrupts due to software code detecting Software error or exceptional condition and executing software interrupt instruction

- Software sources for interrupt are related to software detecting computational error or exceptional condition during execution and there up on executing a SWI (software interrupt) instruction, which causes processor interrupt of ongoing routine.

Software Interrupt by a software instruction

- Software interrupt is interrupt-generated, for example, by a software instruction Int *n* in 80x86 processor or SWI *m* in ARM7, where n is interrupt type and *m* is 24 bits related to ISR address pointer and ISR input parameters pointer

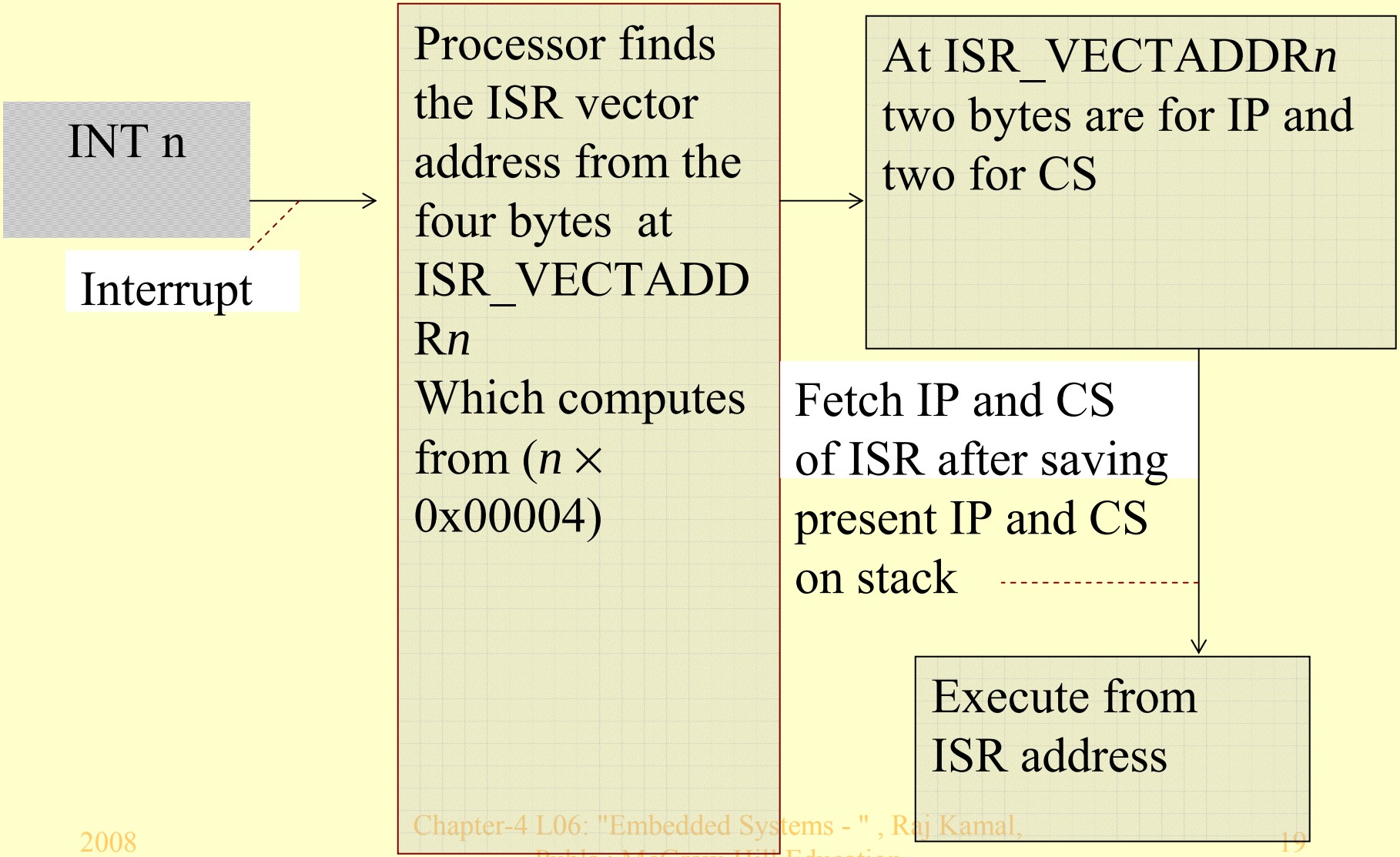
Steps on interrupt of type *n* or on Software Instruction INT *n* in 80x86

1. INT *n* means executes interrupt of type *n*
2. *n* can be between 0 and 255.
3. INT *n* causes the processor vectoring to address 0x00004 $\times$ *n* for finding IP and CS registers values for diversion to the ISR.

8086 and 80x86 two byte instructions INT *n*

- *n* represents type and is the second byte. This means ‘generate type *n* interrupt’ and processor hardware gets the ISR address using the vector address 0x00004 × *n*. When *n* =1, it represents single step trap in 8086 and 80x86

80x86 Steps on INT *n* Innstruction or on interrupt of type *n*



Examples of Software Instruction Related Interrupts Sources

- Handling of –ve number square root throws an exception, that means executes an SWI, which is handled by SWI instruction SWI *n* (Similar but not analogous to INT *n* in 80x86) in the instruction set of a processor

Examples of Software Instruction Related Interrupts Source from Signal

- Certain software instruction for interrupting for diversion to interrupt service routine or another task or thread, also called signal handler.
- These are used for signaling (or switching) to another routine from an ongoing routine or task or thread

Software interrupt instructions

- Software instructions are also used for trapping some run-time error conditions (called throwing exceptions) and executing exceptional handlers on catching the exceptions

Examples

- Instruction, SWI in 68HC11.
- Single byte instruction INT0 in 80x86. It generates type 0 interrupt
- Type 0 interrupt means generation of interrupt with corresponding vector address 0x00000.
- Instead of the type 0 interrupt by instruction, 8086 and 80x86 hardware may also generate interrupt on a division by zero

Examples

- Single byte 8086 and 80x86 instruction TYPE3 (corresponding vector address 0x00C0H). This generates an interrupt of type 3, called break point interrupt.
- Break point interrupt instruction is like a PAUSE instruction.
- PAUSE─ a temporary stoppage of a running program, enables a program to do some housekeeping, and then return back to instruction after the break point by pressing any key.

Summary

We learnt

- Hardware sources of interrupts from ports, external pins, ADC, timer, counter, and other devices in the system
- Software instruction related or software-defined condition related software interrupts
- Interrupts of various types on trap

We learnt

- Interrupts are essential
- ISRs implement SWIs for error handling, software timer, signaling another routine to run and device driver functions

End of Lesson 6 of Chapter 4