Announcement

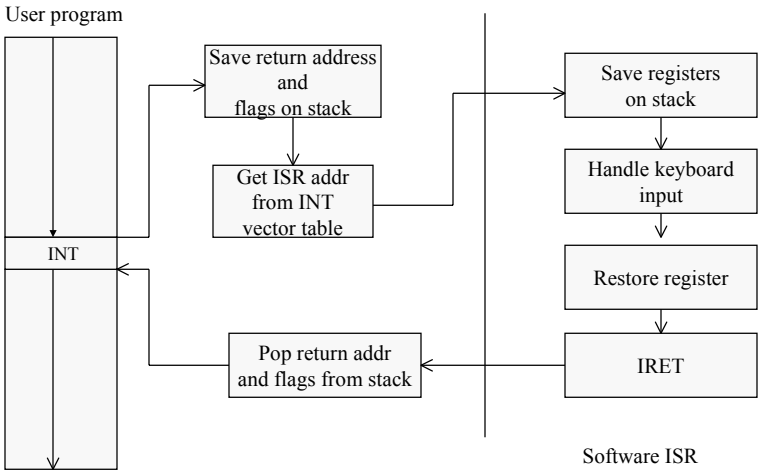
- We have learned the addressing, data transfer instructions and practiced simple device I/O in the lab.
- We now review interrupt handling
  - lecture 5: basics of interrupt
  - lecture 6: priority interrupts

Why Interrupts are Needed

A detour from normal program execution to handle

- slow I/O devices, the CPU is doing something else, and responds to
  - the keyboard as soon as a key has been pressed.
  - Printer which it becomes ready to receive new data.
- various error conditions
  - divide by zero
  - invalid op code
  - stack segment overflow
  - protection faults: privilege rules violated, write to protected code segment…
- What is the difference from interrupt and polling? When to use interrupt and when to use polling?

The Interrupt Model



The Addresses of ISRs: Interrupt Vector Table

- Interrupt vector table stores the addresses of the ISRs. The interrupt service routine (ISR) are identified by the starting address of the vector.

0000	IP-0
0002	CS-0
0004	IP-1
0006	CS-1

INT 0H: divide by zero

INT 1H: Single Step

IP: instruction pointer

CS: code segment register

Vectors 32 - 255 available to users

INT #H is stored at address: 4 * #H. Thus INT 1H is stored at 0004H

How to Pass An Interrupt Type Number to Computer

- Self Service: for CPU logic generated interrupts, e.g. bus error etc.
- Software interrupt: you supply the number by using the instruction: INT #H.Could you give me a couple of examples that software interrupts are needed?

- Quite often, an external I/O devices generates an interrupt. Again, your are responsible for providing the INT #H.
  - The I/O device pulls the interrupt request line INTR
  - The CPU responds by acknowledging $\overline{\text{INTA}}$
  - The I/O device clears INTR and puts a 8 bit interrupt number of data bus AD0 .. AD7

- The interrupt number on the device can be set by an 8 bit dip-switch. It must
  - not _____ be a used one
  - have _____ corresponding Interrupt vector table entry & ISR
- OR _____ will happen. crash

Help Your Computer to Find the Right ISR

- Supposed that interrupt number 34H has not been used and we would like to store the ISR at memory location 30400H. Code segment starts at 30000H. That is, we would like the ISR starts with an offset 400H to the segment base adr. 30000H.
- IP = _____H (1)
- CS = _____H (2)
- INT Vector Address = _____ H (3)
- 1) 400 H (2) 3000 H 3) D0H (34H*4H)

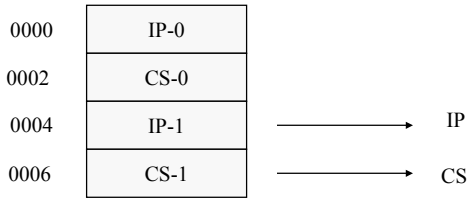
- 0 PUSHA (4)
- 1 MOV DS, 0 (5)
- 2 MOV AX, 400H (6)
- 3 MOV [D0H], AX //store IP (7)
- 4 MOV AX, 3000H (8)
- 5 MOV [D1H], AX //store CS (9)
- 6 POPA (10)
- There are 2 bugs, can you spot and correct them?

- (5) not allowed (9) D2H (2 byte)

Interrupt Handling Sequence- 1

Since the interrupt is a detour to the normal execution, we need to

- Save the context and jump to interrupt service routine
  - Save the content of the status (flag) register by pushing it to the stack.
  - Clear the Interrupt (IF) and trap (TF) to disable the INTR pin and the trap for, e.g., single step execution.
  - Save the content of the code segment register (CS).
  - Save the content of instruction pointer (IP).
  - Place interrupt vector's content to IP and CS (jump to ISR).



Interrupt Handling Sequence - 2

- Interrupt service routines are responsible to handle the event that causes the interrupt.
- They need save the values of the registers before using them
  - PUSH AX ; save register AX
  - PUSH BX
  - Body of Code
  - POP BX ; restore register BX
  - POP AX
  - IRET ; return

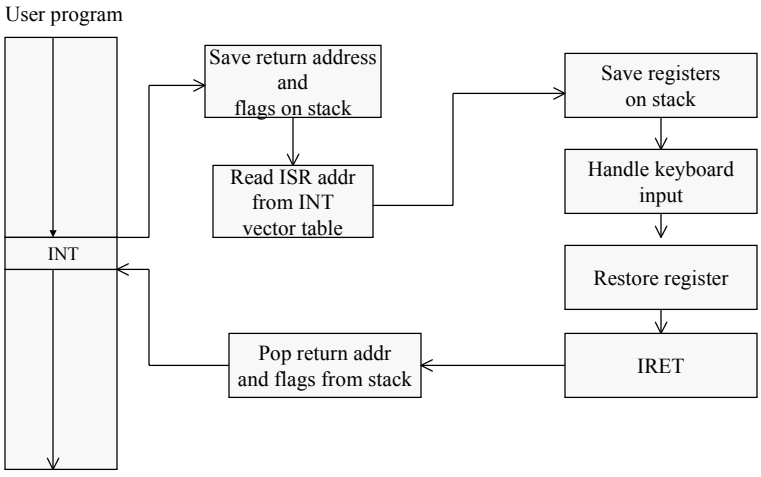
Interrupt Handling Sequence - 3

The execution of return from interrupt IRET causes

- the saved flag register is pop and restored
- the saved code segment and instruction pointers are restored.

The program resumes the computation at the point prior to the interrupt.

The Interrupt Model



Summary

- We have examined the basics of interrupts.
- Next, we will look at prioritized interrupt.