Announcement

TA Office Hours (1102 DCL):

Monday	**2:00 – 3:00pm**	**Xiaolei**
Tuesday	**2:30 – 3:30pm**	**Tim**
Thursday	**2:30 – 3:00pm**	**Tim**

Last time: Basic concepts of Interrupt.

Today: learn how to set up an ISR for hardware and software interrupts and nested priority interrupts.

Inline AT&T Style Assembly Example

```
void add_sub(int x, int y)
{
        printf(“Start: x=%i, y=%i\n”, x, y);

        __asm__(“addl %1, %0”
                “subl %0, %1”
               :”=a”(y)
               :”b”(x), “0”(y) );

        printf(“End:   x=%i, y=%i\n”, x, y);
}
```

What will be printed when **`add_sub(2, 23)`** is called?

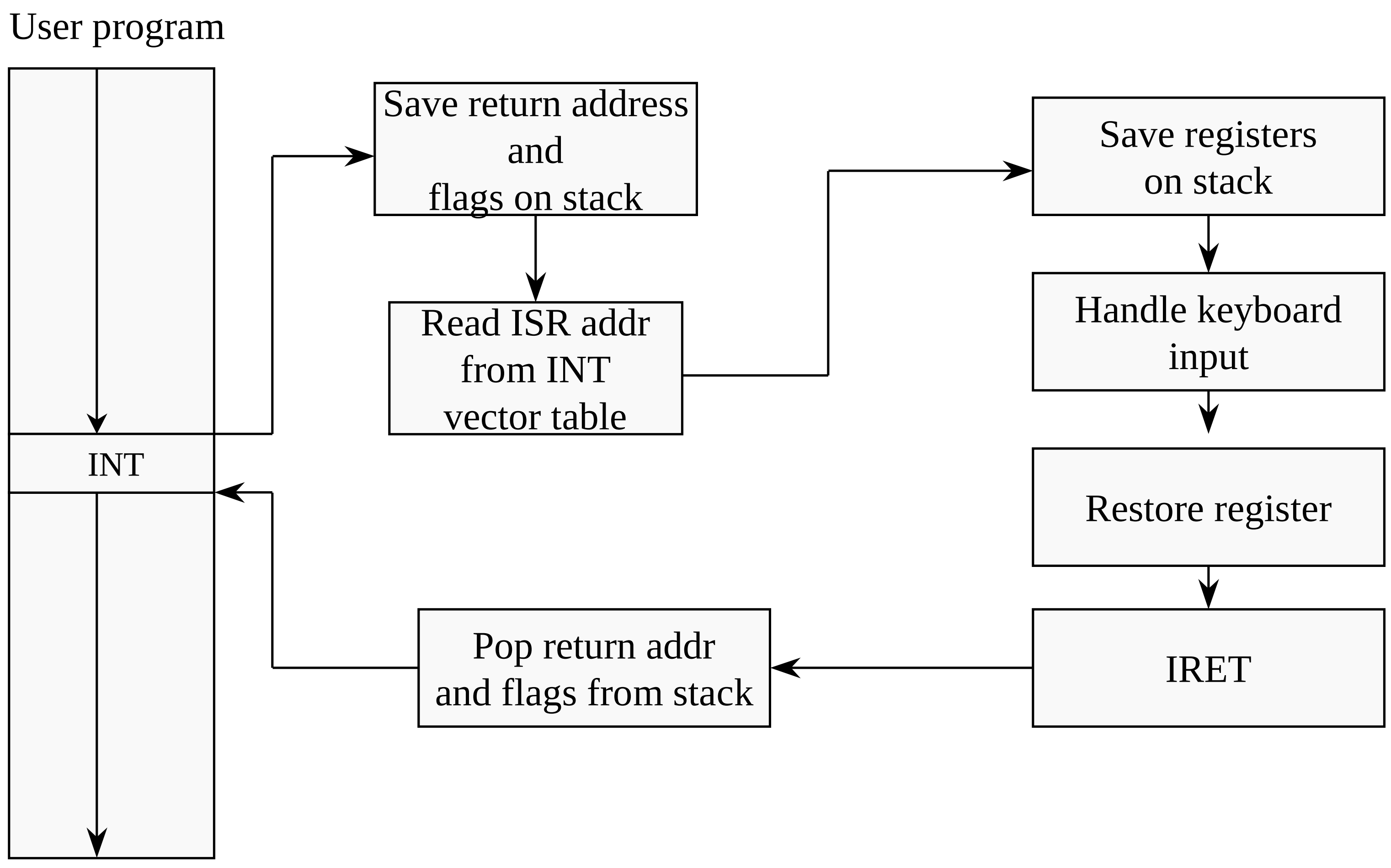
Review: Generating an Interrupt

- Interrupts can be generated by microprocessor, for example, detecting an overflow. The interrupt type number of processor generated interrupts are defined by the processor designers.

- Interrupts can be generated by software. For example, the interrupt type number 3H was, by convention, reserved as a software interrupt for break points used by debugger. To invoke it
  - INT 3H

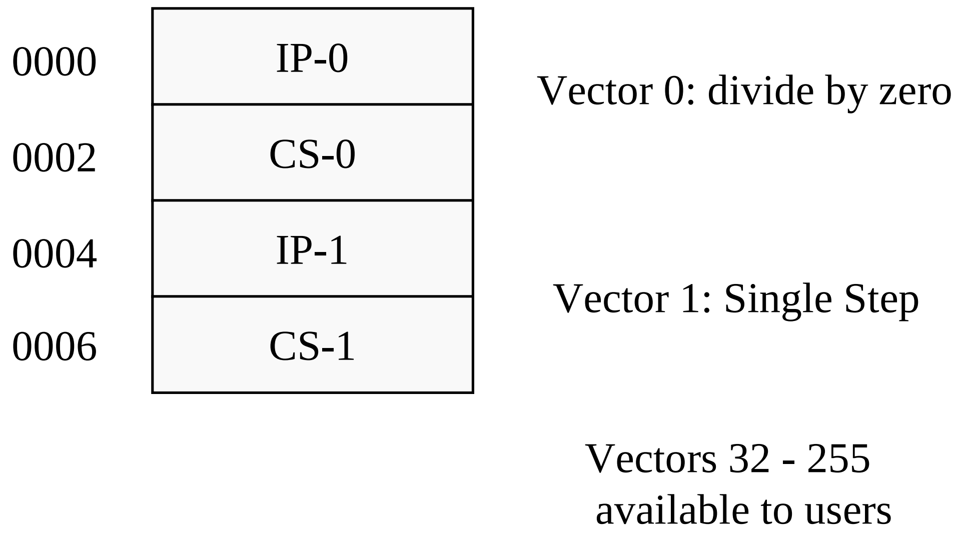
- Interrupt can be generated by external hardware and the hardware is responsible to put the 8 bit interrupt type number on the address bus <AD0…AD7> after it receives the acknowledgement of interrupt from the microprocessor.

Review: Interrupt Model



User program
INT
Save return address and flags on stack
Read ISR addr from INT vector table
Save registers on stack
Handle keyboard input
Restore register
IRET
Pop return addr and flags from stack

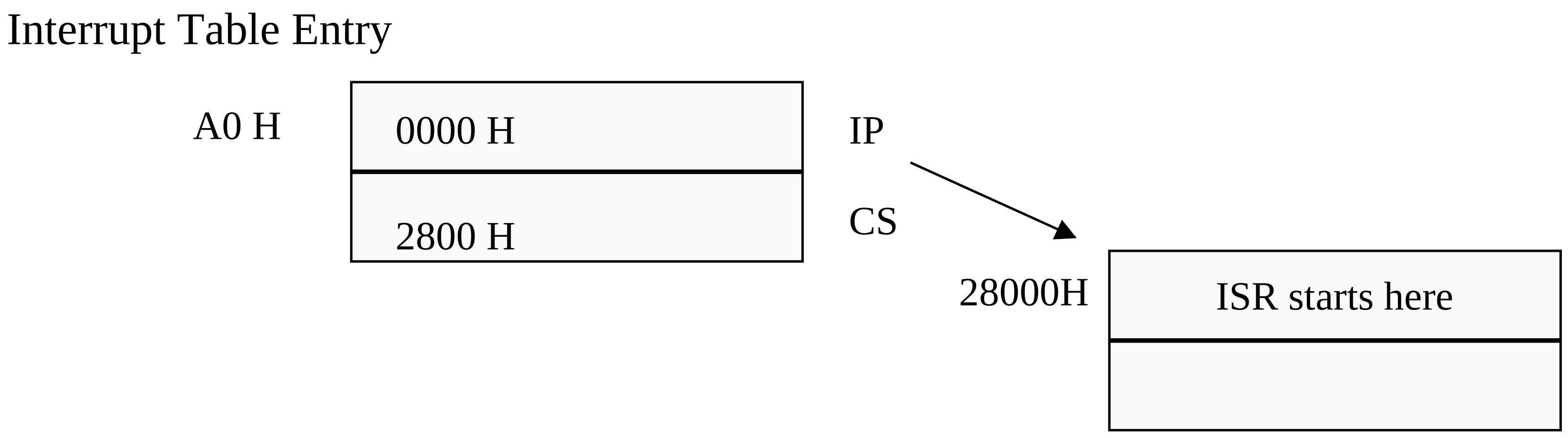
Review: Interrupt Vector Table



Example: INT 01H, where the ISR address is computed by multiplying the interrupt type number, 01H, by 4.

Big Picture

•Suppose that the ISR starts right at 28000H, the starting address of the code segment.



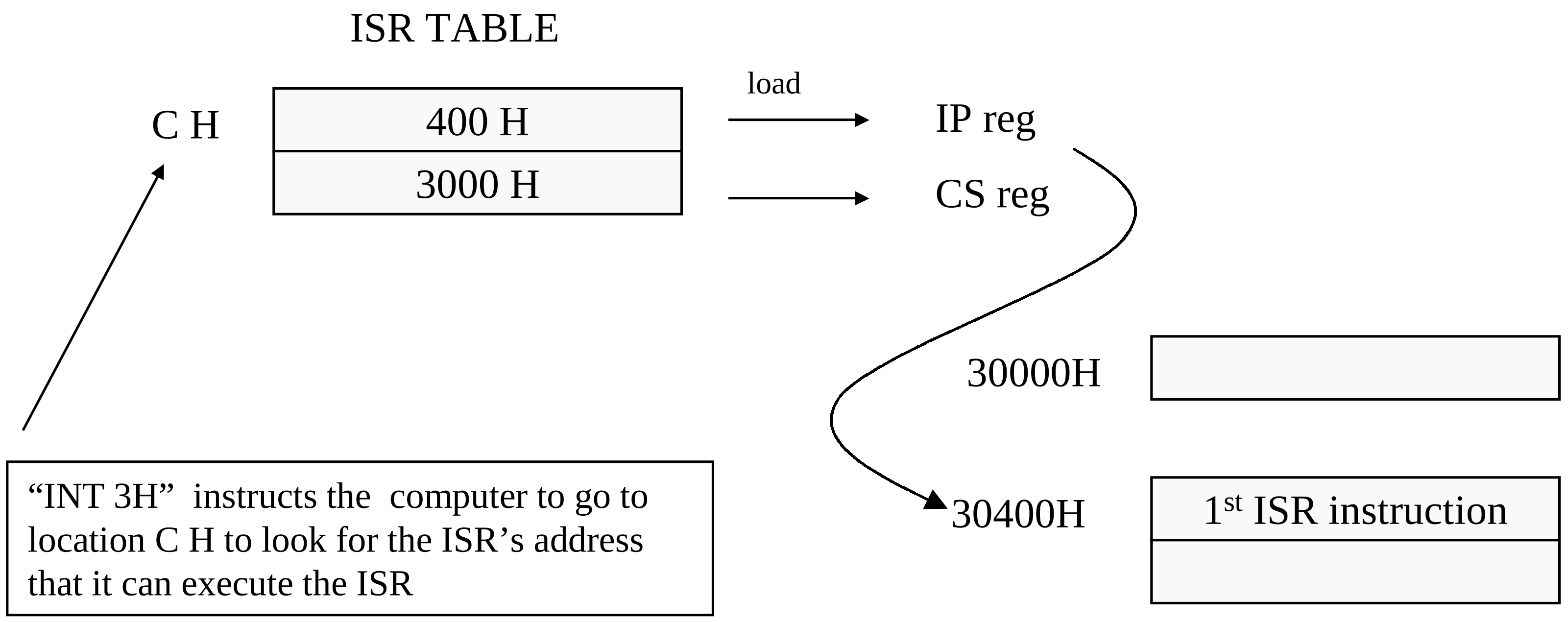
•(If the ISR starts after 28000H, location __________ stores the offset from 28000H)

Why Leave all the data on the table?

- The hardware saves the CS (code segment), IP (instruction pointer), and status of the program being interrupt and then jumps to the ISR.
- Why is the data used by the program being interrupted not automatically saved?
- Answer: 1) not all registers may need to be saved. 2) when you set up a breakpoint at line 17 in your program for debugging, the compiler inserts a software interrupt call, INT 3H, right before line 17.
- When the breakpoint ISR is executed, it can see all the register values left behind by your program right before*** line 17 is executed.
- *** or after, depending on the debugger convention.

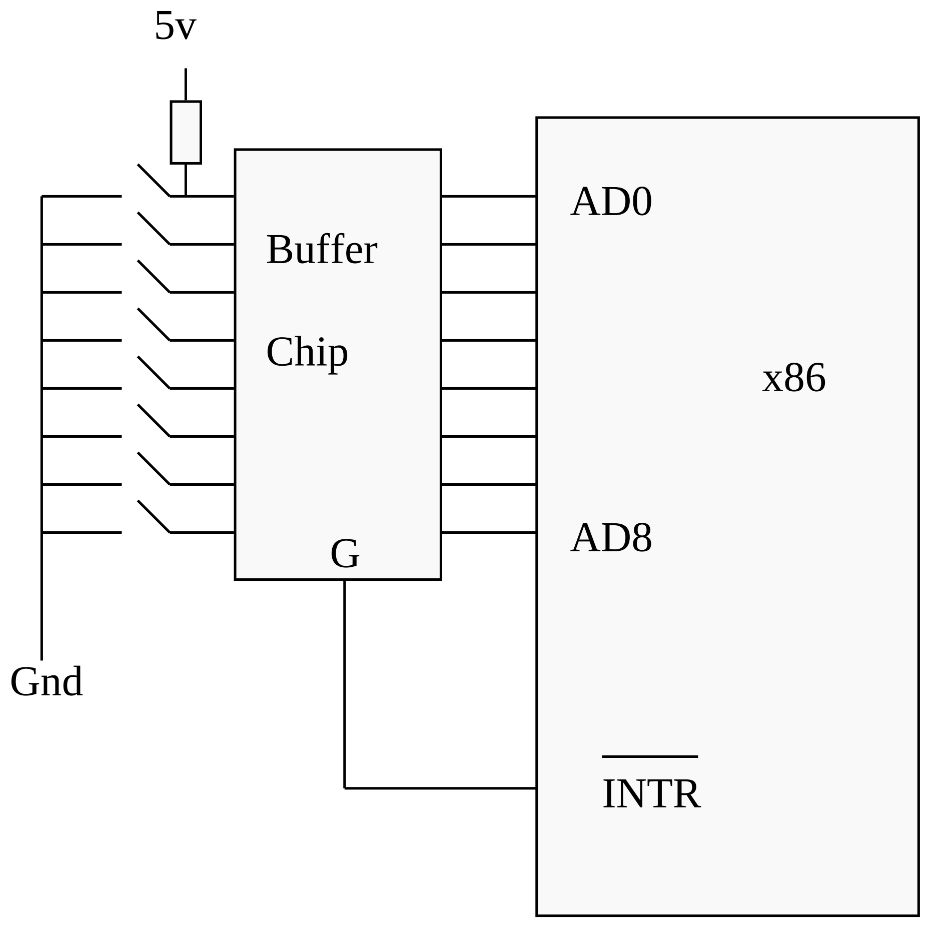
Example: Breakpoint

- Suppose that the breakpoint ISR code begins at 30400H
- when a user sets a break point at line 17, the compiler inserts “INT 3H” call into the object code of your program, right before the object code generated for line 17
- Before line 17 is executed, the ISR is invoked so that you can see all the values of the variables used by your program right before line 17 is executed. And you can single step line 17 and see what difference does line 17 makes. …



“INT 3H” instructs the computer to go to location C H to look for the ISR’s address that it can execute the ISR

Hardware Interrupt Type Number



Prioritized Interrupt Basic

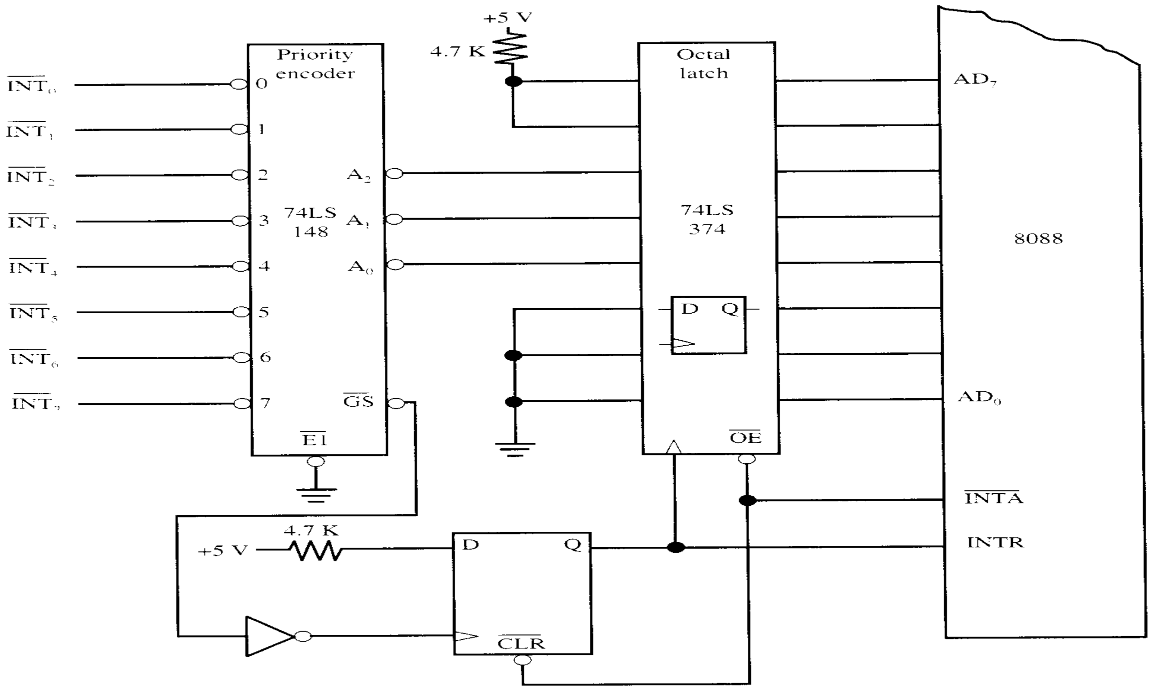


FIGURE 5.9 Prioritized interrupt circuitry

-when several interrupts are pending, the highest one is selected (largest) by the encoder.

- The interrupt type number is 11xxx000.

Nested Priority Interrupt

- when a processor is serving a maskable interrupt, it will be interrupted only when there is a newly arrived higher priority interrupt request.

Priority Interrupt
Controller Chips

•provide a priority mask register to disable interrupt up to any priority level.

•provide a interrupt mask register to disable any particular interrupt request line.

• status registers

Nested Interrupts - 1

- STI ;set (enable) interrupt flag
- CLI ;clear (disable) interrupt flag

- Recall from last lecture that when the hardware passes to the control to ISR
  – Save the content of the status (flag) register by pushing it to the stack.
  – <u>Clear the Interrupt (IF) and trap (TF) to disable the INTR pin and the trap for, e.g., single step execution.</u>
  – 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).

Nested Interrupt - 2

- What if we want to enable a higher priority interrupt while serving a lower priority interrupt?
- Should we just enable interrupt after we enter the ISR? Why or why not?
- Answer: if we execute STI after saving the context in the ISR, the INTR line will be enabled. At this point, any new interrupt, with higher or lower priority can cause a nested interrupt. (This is not exactly what we had in mind.)

Example: Maskable Nested Priority Interrupt

Supposed priority 3 interrupt arrives first, followed immediately priority level 5 (higher interrupt)

<u>interrupt level 3 ISR</u>

- save context AND the old priority mask
- set priority mask level 3 (level 3 or lower disabled)
- enable interrupt
- serving interrupt

$\longrightarrow$

<u>interrupt level 5 ISR</u>

•save context AND the old priority mask
•set priority mask level 5 (level 5 or lower disabled)
•enable interrupt
•serving interrupt
•restore context AND the old priority mask
•IRET

$\longleftarrow$

•serving interrupt
•restore context AND the old priority mask
•IRET

Summary

Today, we have learned how to

- set up the interrupt vector table
- how maskable nested priority interrupts work. (note: only the truly highest priority urgent events should use the NMI).

Next, we will study periodic tasks and timers in POSIX