Interrupts

COMP375 Computer Architecture
and Organization

Goals

- Understand what causes an interrupt.
- Understand the design options for handling an interrupt.

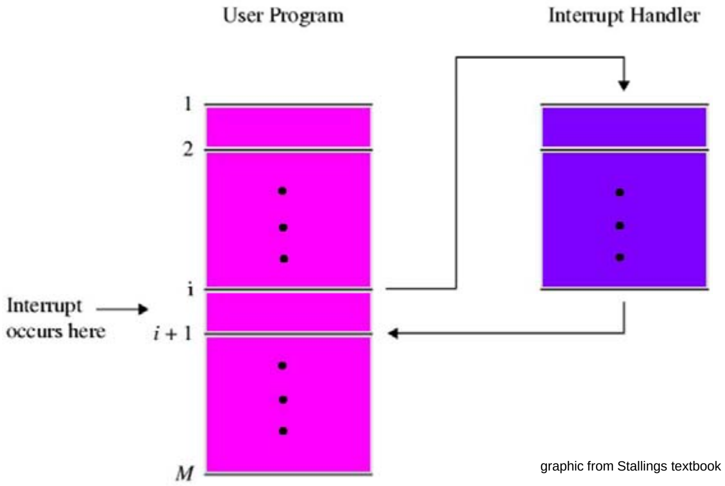
Immediate Attention

- Interrupts are a way that a running program can be stopped to allow the operating system to do something immediately.
- Some activities require the CPU to respond quickly. A very short program may be all that is necessary to handle a situation, but that program has to be run very shortly after the situation occurs.
- When a program does something wrong (divide by zero or bad pointer), the operating system needs to take over.

Interrupts and Exceptions

- An interrupt is a change in program defined flow of execution.
- When an interrupt occurs, the hardware executes the instructions at a specified address instead of following the normal program flow.
- User programs are interrupted all the time.

Transfer of Control via Interrupt



graphic from Stallings textbook

Types of Interrupts

- **External** – Generated by an I/O device
- **Internal** – Exception within a program
- **Program Generated** – Used to transfer control to the operating system

External Interrupts

- I/O devices tell the CPU that an I/O request has completed by sending an interrupt signal to the processor.
- I/O errors may also generate an interrupt.
- Most computers have a timer which interrupts the CPU every so many milliseconds.

Internal Interrupts

- When the hardware detects that the program is doing something wrong, it will usually generate an interrupt.
  - – Arithmetic error
  - – Addressing error
  - – Page fault
  - - Invalid Instruction
  - - Hardware malfunction
  - - Debugging
- A Page Fault interrupt is not the result of a program error, but it does require the operating system to get control.
- Internal interrupts are sometimes called **exceptions**.

Program Generated Interrupts

- Most computers have an instruction that generates an internal interrupt.
- Program generated interrupts are a means for user programs to call a function of the operating system
- Some systems refer to these interrupts as a **S**uper**V**isor **C**all or SVC

`int` Instruction

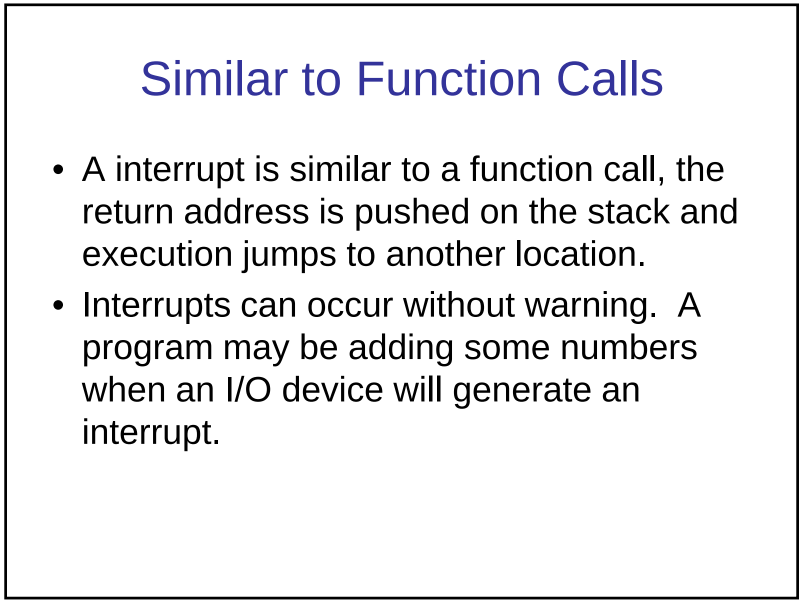
- The Intel Pentium **`int`** instruction generates a program interrupts.
- This is the mechanism for a user program to call an operating system function.
- The **`int`** instruction takes a one byte operand.
- The bottom 1K (1024 bytes) of system memory is devoted to the storage of interrupt vectors.

DOS Print Character

```
MOV AH,02  ; To select print character,
; move the appropriate number, 2, to AH.
MOV DL,"!" ; the character to output
; should be in register DL
INT 21h    ;  call the interrupt.
```

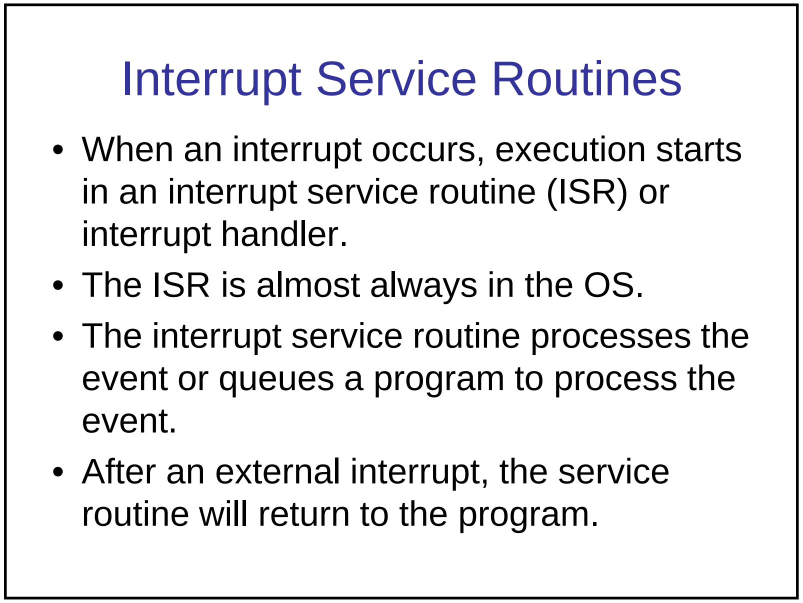
Interrupt Action

- When an interrupt occurs, the program counter and status flags are saved in a special location.
- New program counter and status flags are loaded. The location may be determined by the type of interrupt.



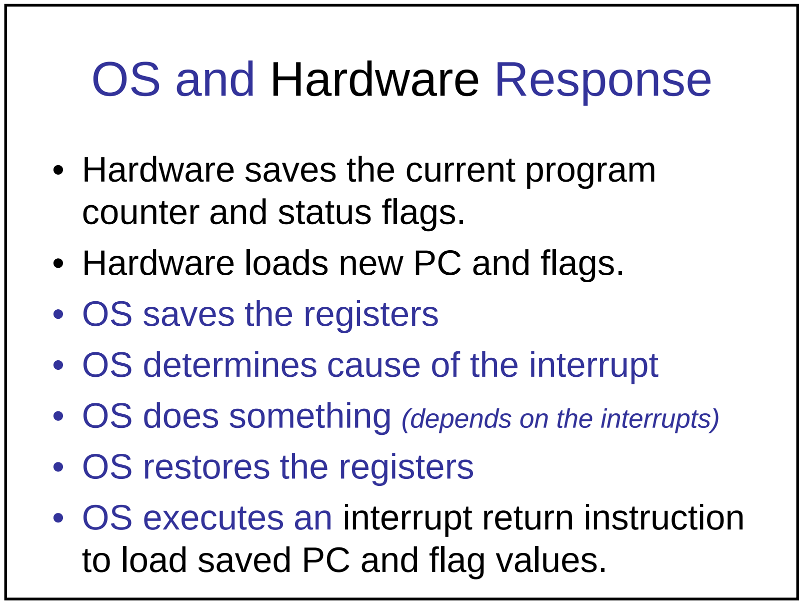
Similar to Function Calls

- A interrupt is similar to a function call, the return address is pushed on the stack and execution jumps to another location.
- Interrupts can occur without warning. A program may be adding some numbers when an I/O device will generate an interrupt.



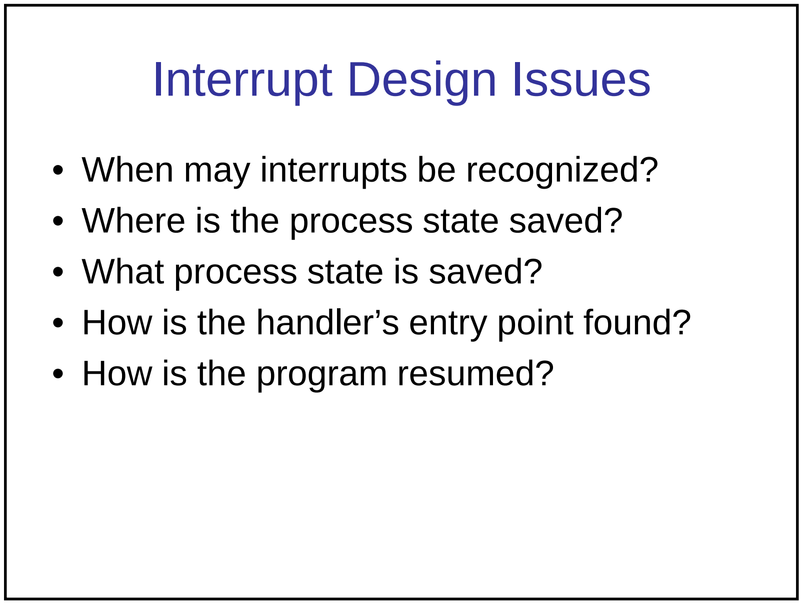
Interrupt Service Routines

- When an interrupt occurs, execution starts in an interrupt service routine (ISR) or interrupt handler.
- The ISR is almost always in the OS.
- The interrupt service routine processes the event or queues a program to process the event.
- After an external interrupt, the service routine will return to the program.



OS and Hardware Response

- Hardware saves the current program counter and status flags.
- Hardware loads new PC and flags.
- OS saves the registers
- OS determines cause of the interrupt
- OS does something *(depends on the interrupts)*
- OS restores the registers
- OS executes an interrupt return instruction to load saved PC and flag values.



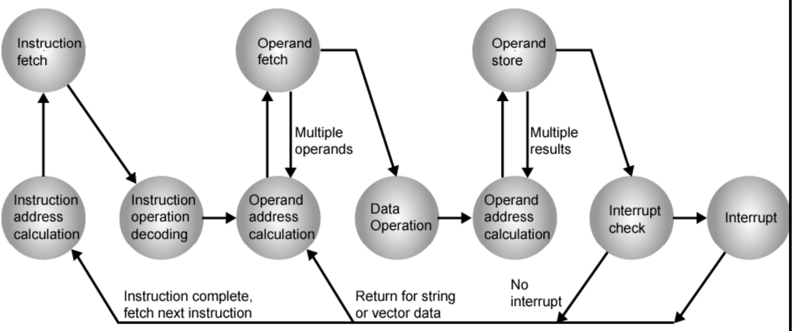
Interrupt Design Issues

- When may interrupts be recognized?
- Where is the process state saved?
- What process state is saved?
- How is the handler’s entry point found?
- How is the program resumed?

Recognizing Interrupts

- An external event can signal the CPU to interrupt at any time, even in the middle of an instruction.
- External interrupts take effect at the end of an instruction.
- Some long repeating instructions provide an opportunity to interrupt between iterations.

Instruction Cycle (with Interrupts) - State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetch next instruction
Return for string or vector data
No interrupt

graphic from Stallings textbook

Internal Interrupts

- Internal interrupts are signaled during an instruction.
- Execution of an instruction can raise an arithmetic error interrupt.
- Page faults can be created during the instruction fetch, operand fetch or operand store or all of the above.

Saving Process State

- Interrupts can be considered similar to a function call.
- The program counter and processor state register can be saved on the stack.
- It is unwise to save system information in user address space, thus the interrupt information cannot be saved on the user stack.
- A special OS stack can be used.

Special Save Areas

- Some architectures provide a special fixed location to save the executing program's state.
- Some processors, such as MIPS, save the interrupt address in a special register, the exception program counter (EPC).
- A fixed location can be overridden if you have nested interrupts.

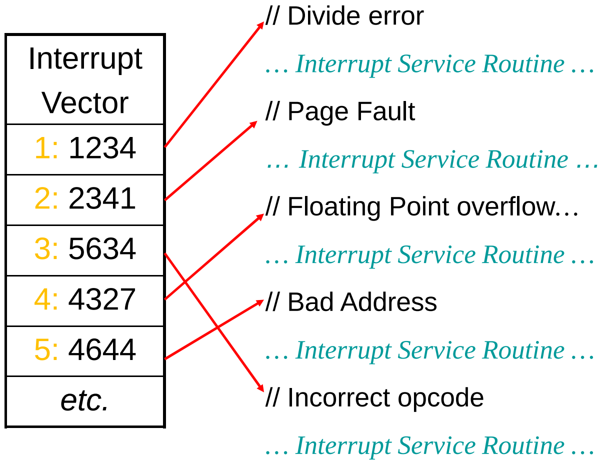
What to Save

- The processor needs to save enough information so the executing program can be resumed.
- Information usually saved:
  - Program Counter
  - Status bits
  - Registers (by OS)
  - Addressing environment (by OS)
- Current process may need to be suspended

ISR Entry Point

- It is possible for all interrupt service routines to start at the same location. The software can determine what kind of interrupt.
- The hardware can assist by using the interrupt type as an index into a table of ISR addresses.
- Each interrupt may have a different ISR entry point or classes of interrupts may have a common entry point.

Interrupt Vector Points to ISRs



Interrupt Vector
1: 1234
2: 2341
3: 5634
4: 4327
5: 4644
etc.

// Divide error

… Interrupt Service Routine …

// Page Fault

… Interrupt Service Routine …

// Floating Point overflow…

… Interrupt Service Routine …

// Bad Address

… Interrupt Service Routine …

// Incorrect opcode

… Interrupt Service Routine …

Interrupt Vector

- In the Intel Pentium each interrupt type has a number associated with it, called the interrupt request queue (IRQ) number.
- When a device interrupts, the IRQ is used as an index into a table of ISR addresses.
- The operand of the int instruction provides an index into a table of ISR addresses.

Resuming Execution

- On external interrupts, the OS generally resumes the running process. The next instruction of the process is executed.
- For some internal interrupts, it may not be possible to restart the program (i.e. addressing error).
- For some interrupts (i.e. page faults) you want to re-execute the instruction.
- For other interrupts (i.e. overflow) you may want to execute the next instruction.

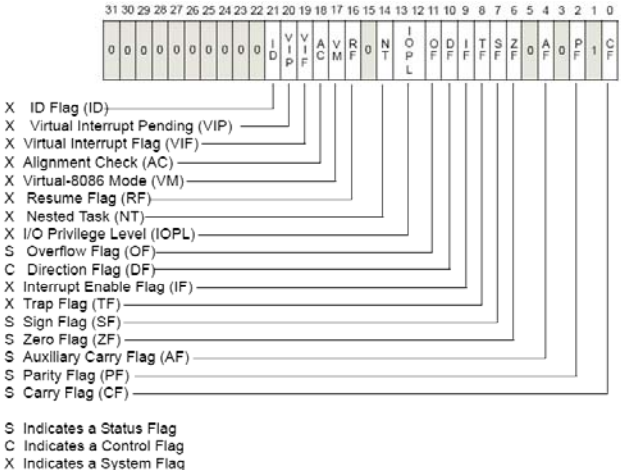
Multiple Interrupts

- An interrupt event can occur while the processor is handling a previous interrupt.
- If the return address is always stored at a fixed location, the occurrence of an interrupt while handling a previous interrupt will overwrite the previous return address.
- Most interrupt service routines start with interrupts disabled. This prevents an interrupt service routine from being interrupted.

Masking Interrupts

- Some interrupts can be temporarily disabled. Most processors can disable external interrupts.
- Most internal interrupts cannot be disabled.
- It is generally problematic to disable interrupts for a lengthy period of time.

Intel EFLAGS Register



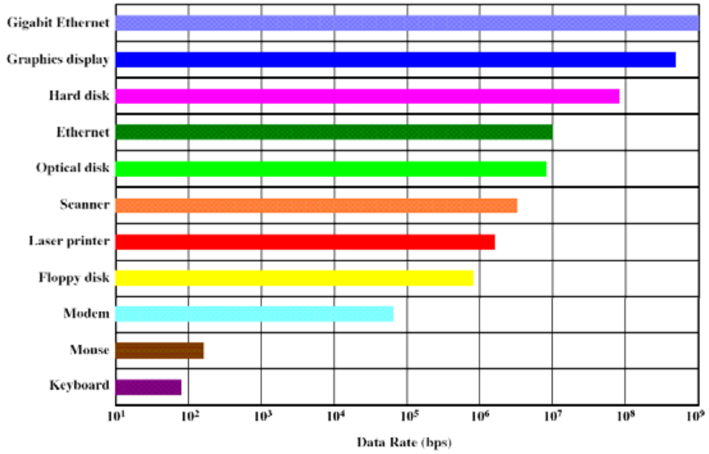
Missing Interrupts

- Many devices will interrupt once per event. If the processor fails to acknowledge the interrupt before the next event, knowledge of the first interrupt is lost.

Interrupt Priorities

- Most systems prioritize the interrupts.
- If two interrupts happen at the same time, the interrupt with the highest priority will be serviced first.

Device Speed



graphic from Stallings textbook