Interrupts

Lecture 6

Interrupts

- All computers provide the mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing
- Most common classes of interrupts are:
  - Program: generated by some condition e.g. overflow, division by zero
  - Timer: Generated by internal processor timer, allow Operating system to perform certain functions at regular intervals
  - I/O: from I/O controller
  - Hardware failure: generated by a failure such as power

Interrupts

- The mechanism by which other system modules may interrupt the normal processing of the CPU
  - These devices are slower than the CPU
  - CPU can waste vast amounts of processing cycles waiting for these slow devices to perform their tasks
- Interrupts let the CPU execute its normal instruction sequence and pause to service the external devices only when they signal (the interrupts) that they are ready for the CPU's attention

Interrupts

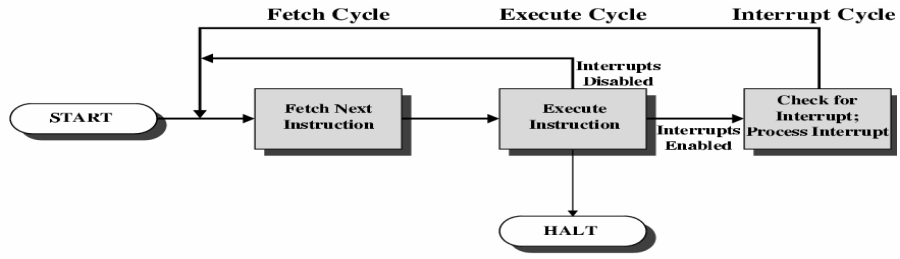
- The processor and the O/S are responsible for
  - recognizing an interrupt, suspending the user program, servicing the interrupt, and then resuming the user program
- Each interrupt is associated with a procedure that directs the actions of the CPU when an interrupt occurs.
  - Nonmaskable interrupts are high-priority interrupts that cannot be ignored.

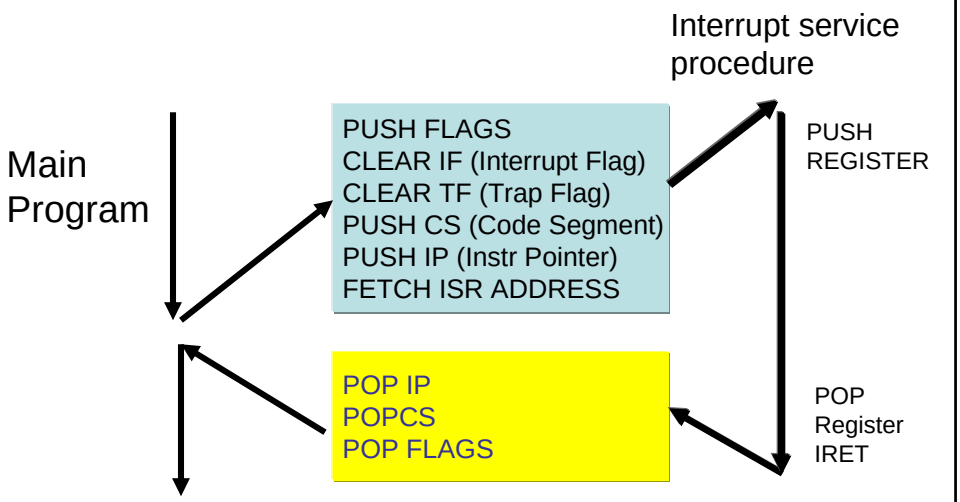
Interrupts

- Interrupts are processed in an interrupt cycle within the overall instruction cycle
  - At the end of an instruction cycle (operand storage step), check to see if any interrupts are pending
- If there aren't any,
  - proceed with the next instruction
- If there are
  - Suspend execution of the program and save its "state"
  - Jump to the interrupt service routine and resume the "normal" instruction cycle
  - When the ISR is completed, restore the state of the program and resume its operation

Interrupt and Instruction Cycle

- To accommodate interrupts, in interrupt cycle is added to the instruction cycle
  - Processor check for the occurrence of interrupt
  - If no interrupt the processor proceeds to fetch cycle and fetch next instruction of the current program
  - In case of interrupt
    - It suspends execution of current program and saves the address of next instruction
    - Set the PC to starting address of interrupt handler routine



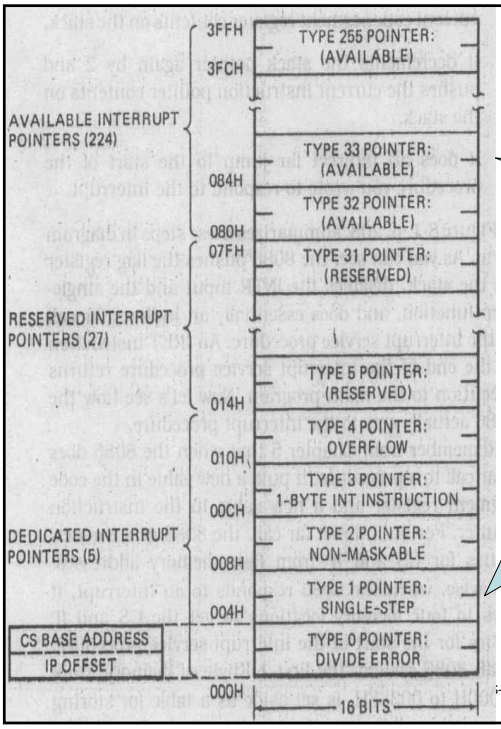


Vector table

- In 8086 the first 1 Kbyte of memory (00000H to 003FFH) is set aside as a table for storing the starting addresses of interrupt service procedures
- Table can hold starting addresses of 256 interrupt procedures
- Starting address of an interrupt service procedure is called “interrupt vector”

Interrupt Vectors

- A method to discriminate interrupt sources
- Methods
  - Interrupt number used for address calculation
  - Actual instruction used for branch
- Interrupt priority
  - Assign different priority to interrupt sources
  - Intel 8259

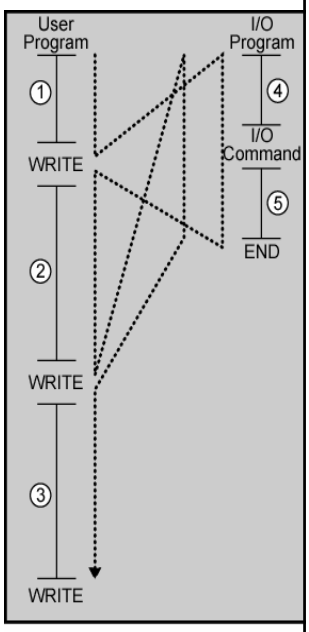


224 out of 256 interrupt types are available for hardware and software interrupts

Vector for each interrupt type requires four memory location.
To respond an interrupt 8086 multiplies the type by 4 to produce the desired address in the vector table

Interrupts

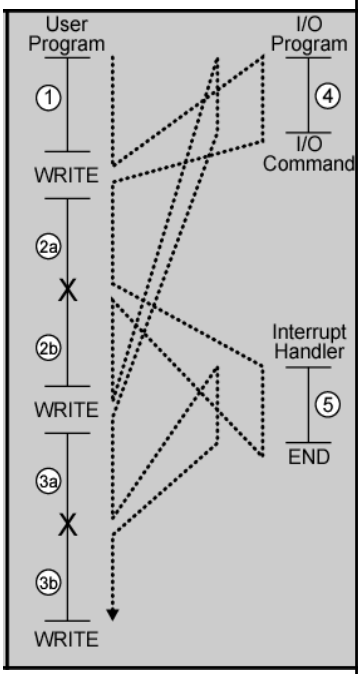
- Provided primarily as a way to improve processing efficiency?
- User program performs a series of WRITE calls
  - Code segment 1,2 and 3 are the sequence of instructions not involving I/O
- WRITE call are to an I/O program, to perform actual I/O operation
  - Code segment 4 is to prepare for actual I/O operation
  - Code 5 to complete the operation
- Since no interrupt so the user program will stop for some time



(a) No interrupts

interrupt

- With interrupts the processor can be executing other instructions while I/O operation in progress
  - When user program reaches to the system call from WRITE
  - The I/O program invoked and all preparation work will be done by the I/O
  - After this short time the control returns to the user program
  - I/O will keep on doing its job
- I/O module will generate an interrupt request
  - Processor will respond by suspending it current execution and branch the current execution



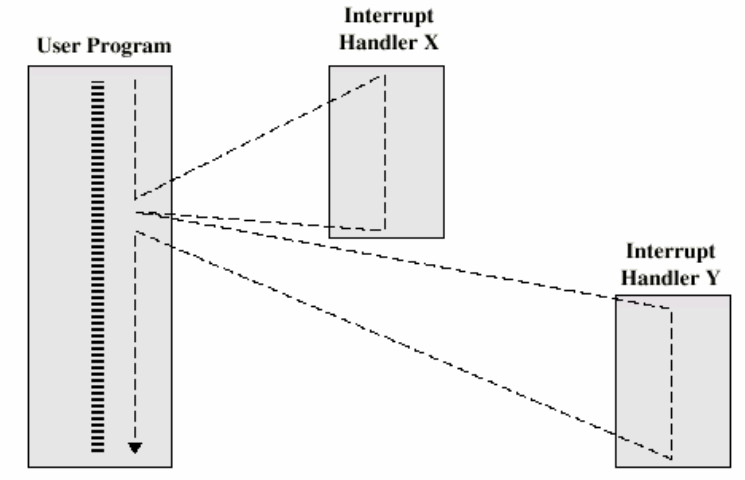
Multiple interrupts

- A typical system can support several to several dozen interrupts
  - How should the system respond if more than 1 interrupt occurs at the same time?
    - Systems prioritize the various interrupts
    - At the start of the interrupt cycle, the highest priority pending interrupt will be serviced
    - Remaining interrupt requests will be serviced in turn
- What if an interrupt occurs while an ISR is being executed (a result of a previous interrupt)
  - Ignore the second interrupt (by disabling interrupts) until the ISR completes
  - Recognize and service the interrupt only if it has a higher priority than the one

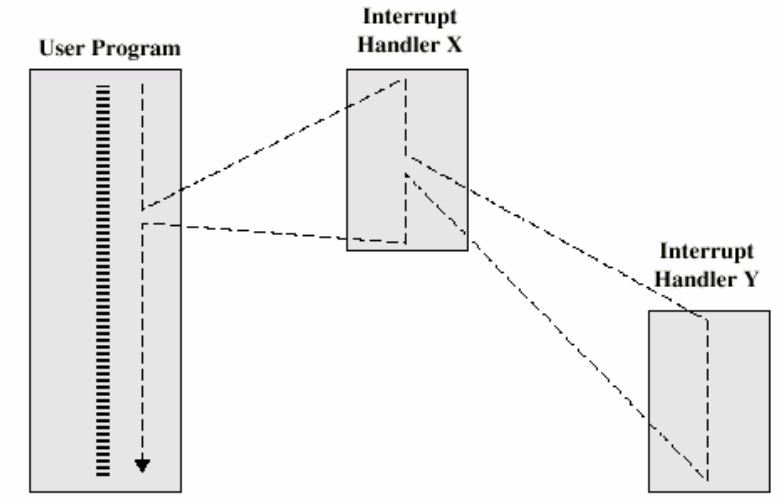
Multiple Interrupts

- Disable interrupts: while an interrupt is being processed
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
    - Drawback: don't give relative pri
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts - Sequential



Multiple Interrupts – Nested



Time Sequence of Multiple Interrupts