IA-32 Instructions II

- Memory mapped Input/Output
  - The code below is an example illustrating the use of mapped I/O
  - Polls keyboard to echo characters until CR (carriage return) is hit

```
        LEA   EBX, LOC          Initialize register EBX to point to the
                                  address of the first location in main memory
                                  where the characters are to be stored.
READ:   BT    KBD_STATUS, 1     Wait for a character to be entered
        JNC   READ                in the keyboard buffer KBD_DATA.
        MOV   AL, KBD_DATA      Transfer character into AL (this clears KIN to 0).
        MOV   [EBX], AL         Store the character in memory
        INC   EBX                 and increment pointer.
ECHO:   BT    DISP_STATUS, 2    Wait for the display to become ready.
        JNC   ECHO
        MOV   DISP_DATA, AL     Move the character just read to the display
                                  buffer register (this clears DOUT to 0).
        CMP   AL, CR            If it is not CR, then
        JNE   READ                branch back and read another character.
```

IA-32 Instructions II

- Port Mapped I/O (isolated I/O) is common with IA-32
  - Requires special Input/Output instructions
  - Examples:

```
IN  AL,DX   ; reads one byte from the devices whose port number is in DX
IN  AX,DX   ; reads two bytes of data from the port
IN  EAX,DX  ; reads 4 bytes

OUT  61h,AX  ; sends two bytes to output port number 61h
OUT  DX,AL   ; sends one byte to the port whose number is in DX
```

Port mapped scheme uses I/O address space separate from the program address space.

IA-32 Instructions II

- IA-32 Interrupts and Exceptions
  - Two interrupt lines
    - NMI non-maskable is always accepted by processor
    - INTR user interrupt is maskable
      - Enabled/disabled using IF flag within status register
      - Accepted if priority > privilege level of currently running program
  - Events other than external interrupts cause exceptions
    - Vector number is assigned to each interrupt or exception
    - This index identifies an IDT (interrupt descriptor table) entry
    - Table entry contains starting address of corresponding handler
    - I/O devices are connected through an APIC
      - Advanced Programmable Interrupt Controller
      - Controller implements priority scheme and sends vector number to CPU

IA-32 Instructions II

- Actions taken for Interrupts or Exceptions

1. Push status register, code segment register and EIP onto processor stack
2. For exceptions due to abnormal condition, push cause of exception
3. Disable further interrupts of the same type
4. Use vector number to load appropriate handler address from IDT into EIP

When finished, the handler uses the IRET instruction to resume the program.

IRET pops EIP, code segment register and status register from stack
Handler must undo any changes it made to stack pointer before executing IRET