Interrupts
I. Instruction Execution

Figure 1.1 Computer Components: Top-Level View
Basic Instruction Cycle

Figure 1.2 Basic Instruction Cycle
Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)
1. FETCH: The PC contains 300, the address of the first instruction. This instruction is loaded into the IR.
Comment: The FETCH: The PC contains 300, the address first four bits in the IR ("1") give the opcode for "LOAD." The remaining twelve bits ("940") specify the memory address where the data is to be loaded from.

2. EXECUTE: The data is loaded into the accumulator. AC now has the value "003"

3. FETCH: The PC is incremented to 301 and the next instruction is fetched.
Comment: The first four bits ("5") give the opcode for "ADD". The remaining twelve bits ("941") give the address of the data to be
4. EXECUTE: The contents of the AC and the contents of location 941 are added and stored back into the AC. AC now has the value "005"

5. FETCH: The PC is incremented to 302 and the next instruction is fetched. Comment: The first four bits ("2") give the opcode for "STORE." The remaining twelve bits ("941") give the address of where to store the result.

6. EXECUTE: The contents of the AC are stored at address 941.
II. Interrupts

Definition: Interrupt is an event that causes the normal fetch-execute cycle to be interrupted so that the event can be taken care of.

Types of interrupts:
- I/O
- Timer
- Program
  - divide by zero
  - overflow or underflow
  - memory protection violation
  - illegal instruction
  - Parity error
- Hardware failure
Why Interrupts?

- Why it's desirable to have interrupts
  - Without interrupts process Process 4 has to keep checking to see if its I/O is done (wasted CPU cycles).
  - With interrupts, Process 2 can run while the I/O is being done. I/O interrupt will let the CPU know when 4's I/O is completed.
  - Process 2 gets interrupted and the I/O interrupt handler takes over. It then passes control to the scheduler who chooses Process 5 to run next.
Program Timing

(a) Without interrupts

(b) With interrupts

Figure 1.8 Program Timing: Short I/O Wait
Interrupt Handler

- **Interrupt Handler** - OS code to take care of each type of interrupt.

**Figure 1.6 Transfer of Control via Interrupts**
Instructions w/Interrupts

Figure 1.7 Instruction Cycle with Interrupts
III. Clocks and Timer Interrupts

- PURPOSE OF CLOCKS
  _ To generate a timer interrupt to control the amount of time a processes has the CPU.
  _ Keep time of day
  _ Accounting, monitoring
  _ Alarm clock processes, watch dog processes
Clocks (continued)

- TWO TYPES OF HARDWARE CLOCKS
  _A. Continuous interrupt clock_
  - Causes an interrupt on every pulse (every 50-60 HZ)
  - Too frequent for the OS, so the interrupt handler software ignores "N-1" interrupts and responds to every Nth interrupt.

  _B. Countdown interrupt clock_
  - Capable of being programmed to cause an interrupt at desired intervals.
  - OS sets clock for a certain number of ticks. Clock interrupts the CPU after that many ticks.
Clocks (continued)

Crystal oscillator generates clock pulses (ticks).

Counter is decremented on each clock pulse.

OS clock routine loads counter with desired countdown interval (in ticks).

Clock Diagram