Lecture 1

Chapter 1

Lecture outline

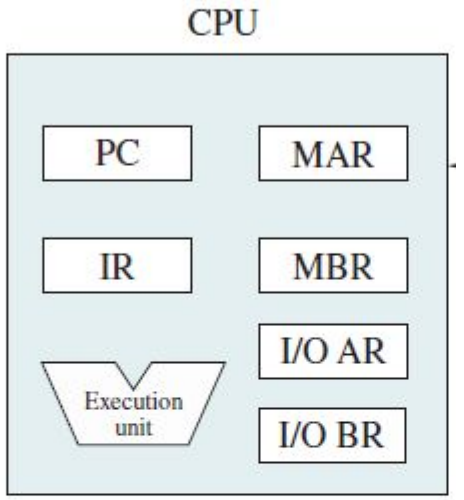
- **BASIC ELEMENTS**
- **EVOLUTION OF THE MICROPROCESSOR**
- **INSTRUCTION EXECUTION**
- **INTERRUPTS**
- **THE MEMORY HIERARCHY**

BASIC ELEMENTS

- A computer consists of 3 main components
    - processor,
    - memory,
    - and I/O components, with
    - one or more modules of each type.

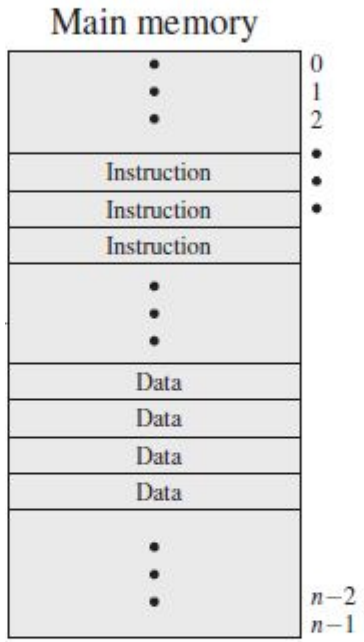
Processor

- Controls the operation of the computer
- performs its data processing functions.
- it is often referred to as the **central processing unit** (CPU).



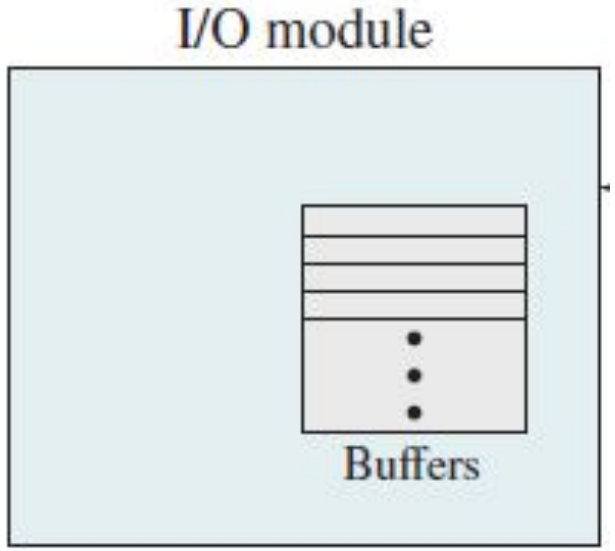
Main memory:

- Stores data and programs.
- memory is typically volatile
- Main memory is also referred to as *real memory* or *primary memory.*



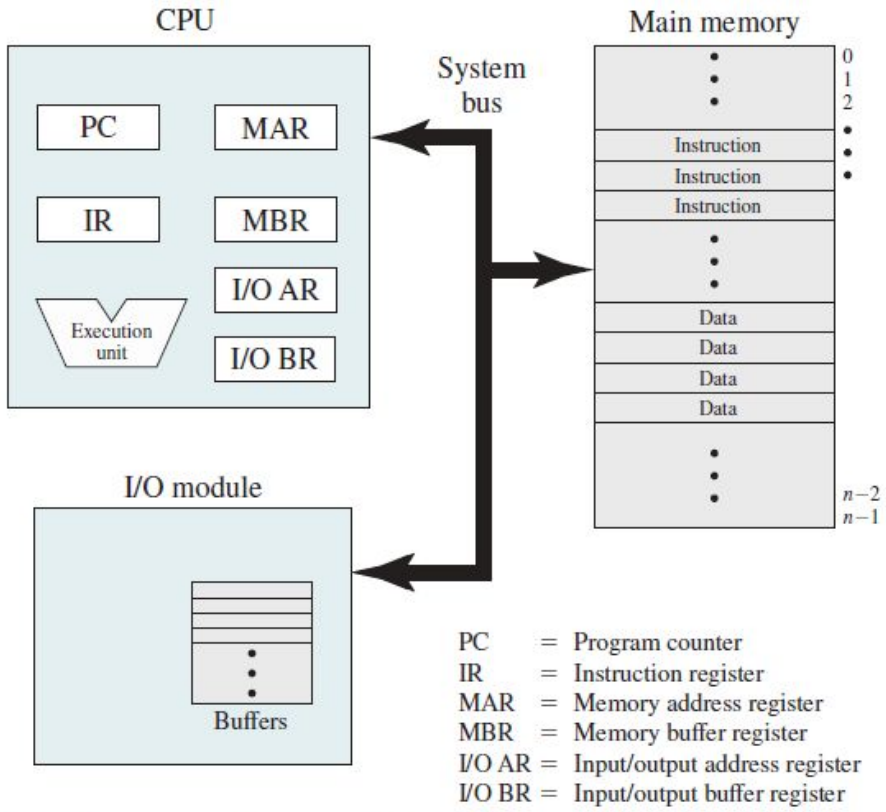
I/O modules:

- Move data between the computer and its external environment.
- The external environment consists of a variety of devices
  - Secondary memory devices (e.g., disks)
  - communications equipment
  - terminals.



System bus:

- Provides for communication among processors, main memory, and I/O modules.

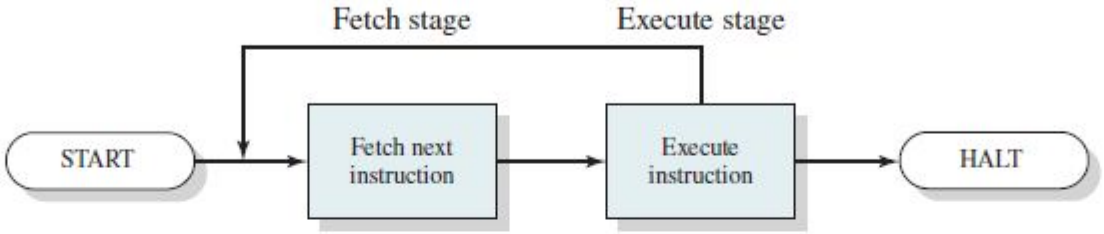


EVOLUTION OF THE MICROPROCESSOR

- The hardware revolution that brought about desktop and handheld computing was the invention of the microprocessor, which contained a processor on a single chip.
- Not only have microprocessors become the fastest general-purpose processors available, they are now multiprocessors; each chip (called a socket) contains multiple processors (called cores), each with multiple levels of large memory caches, and multiple logical processors sharing the execution units of each core.
- Graphical Processing Units (GPUs) provide efficient computation on arrays of data

INSTRUCTION EXECUTION

- A program to be executed by a processor consists of a set of instructions stored in memory.
- Instruction processing consists of two steps:
  - The processor reads (*fetches*) instructions from memory one at a time
  - and executes each instruction.
- The processing required for a single instruction is called an *instruction cycle.*

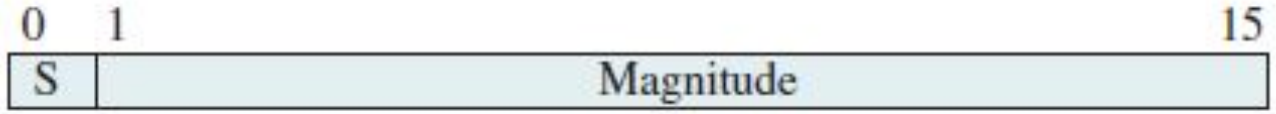


PC and IR

- PC: program counter
  - The program counter (PC) holds the address of the next instruction to be fetched.
  - The processor always increments the PC after each instruction fetch

- IR: Instruction Register
  - The fetched instruction is loaded into the instruction register

(a) Instruction format



(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine

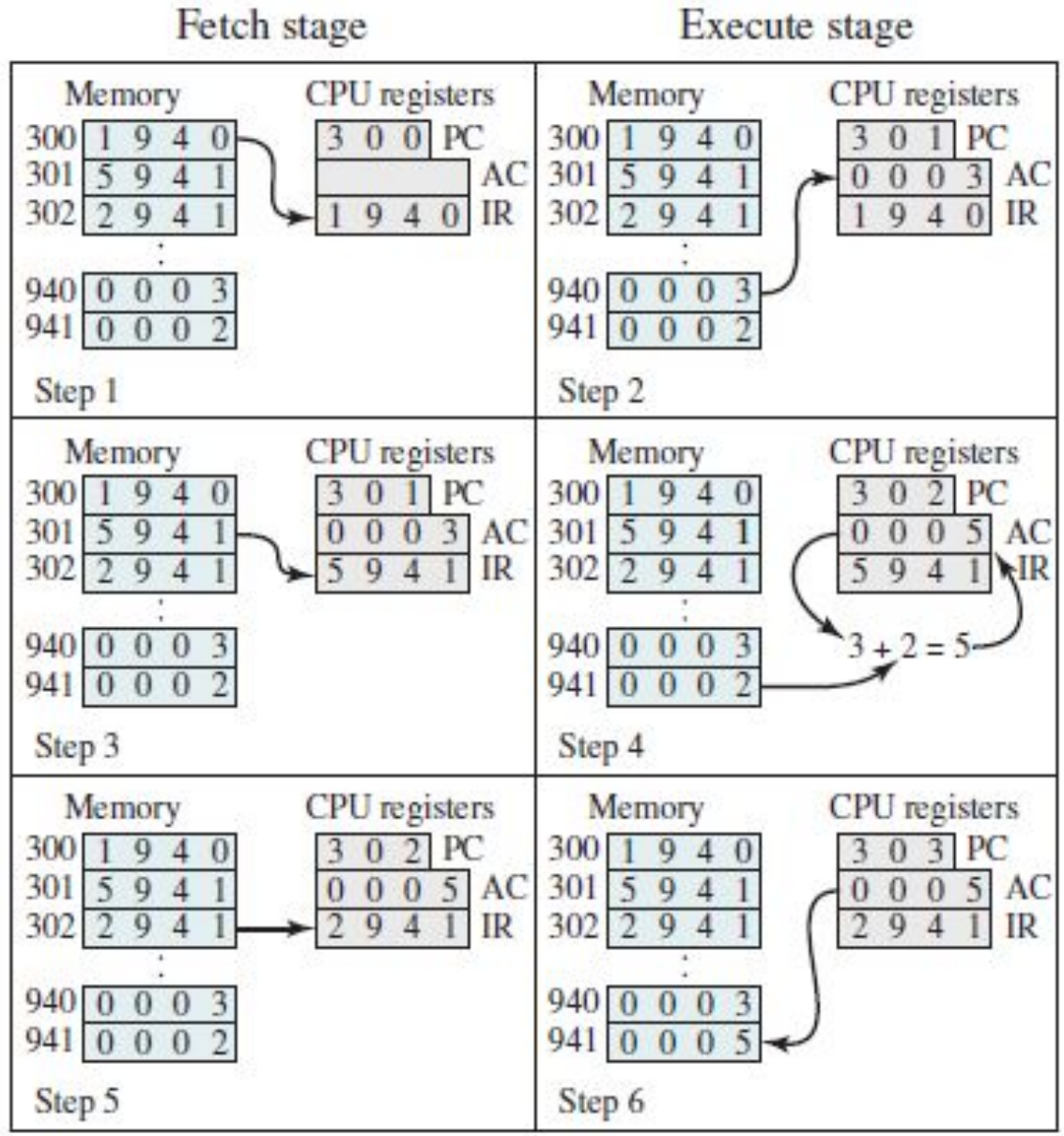


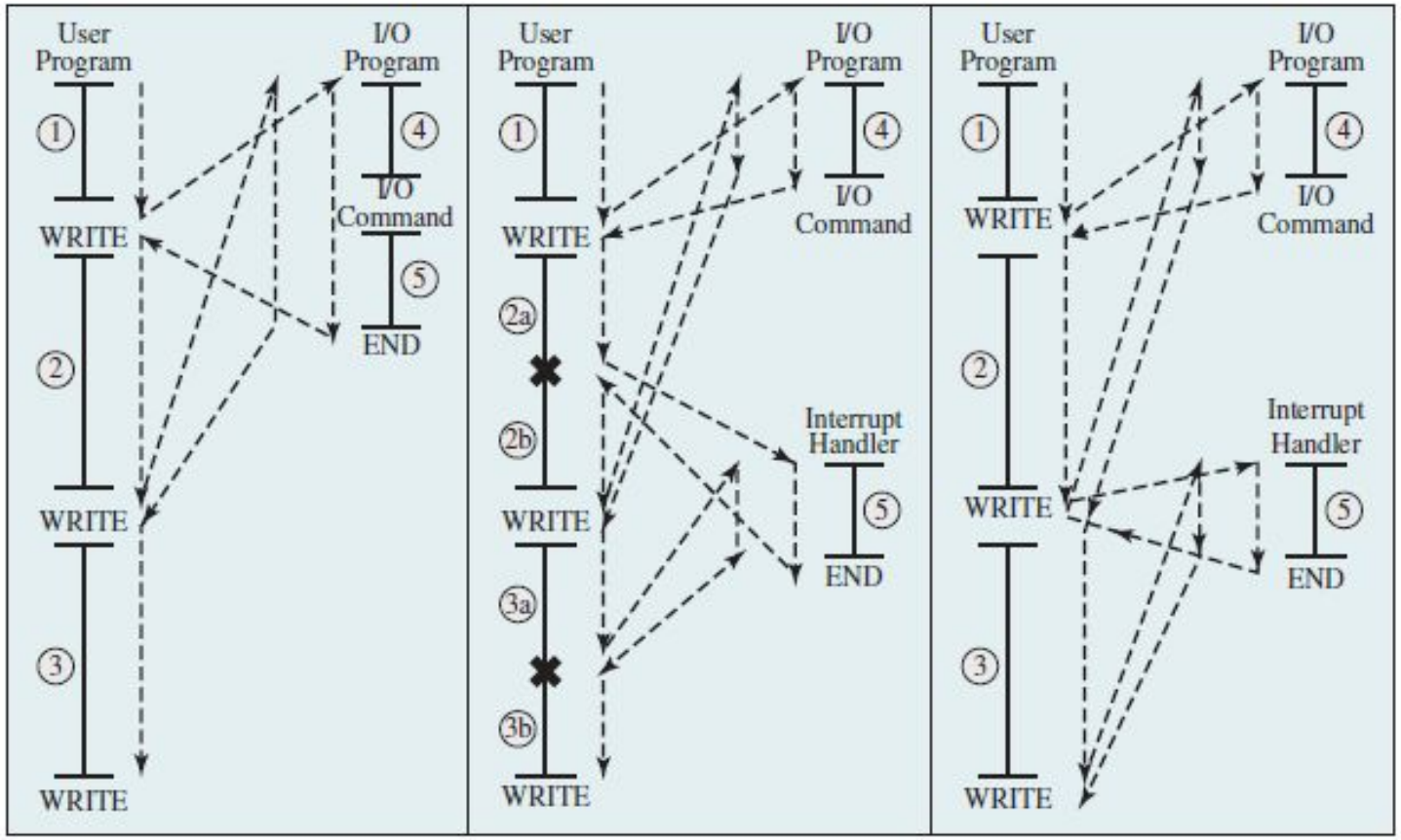
Figure 1.4 **Example of Program Execution (contents of memory and registers in hexadecimal)**

INTERRUPTS

- An interrupt is a signal sent from a device or from a software to the operating system Interrupts are provided primarily as a way to improve processor utilization.

Table 1.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.



(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

✖ = interrupt occurs during course of execution of user program

Figure 1.5 Program Flow of Control Without and With Interrupts

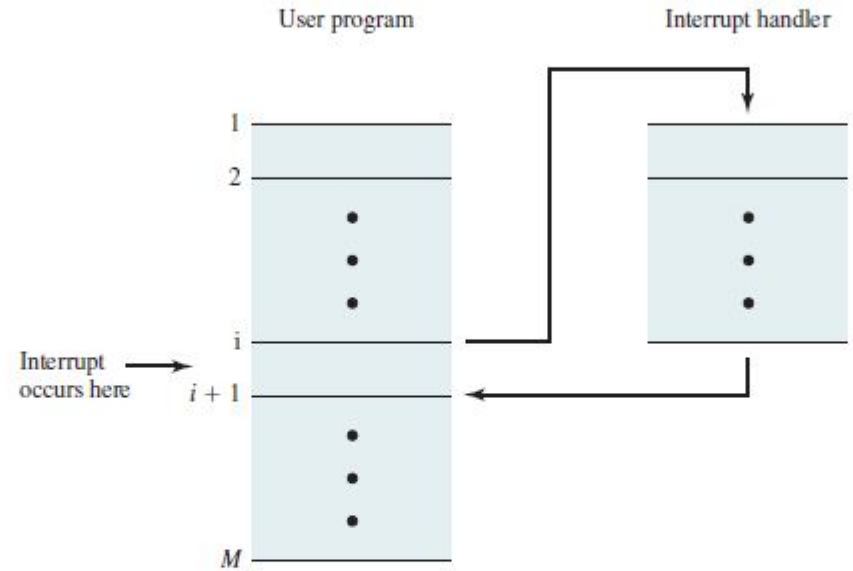


Figure 1.6 Transfer of Control via Interrupts

Instruction Cycle with Interrupts

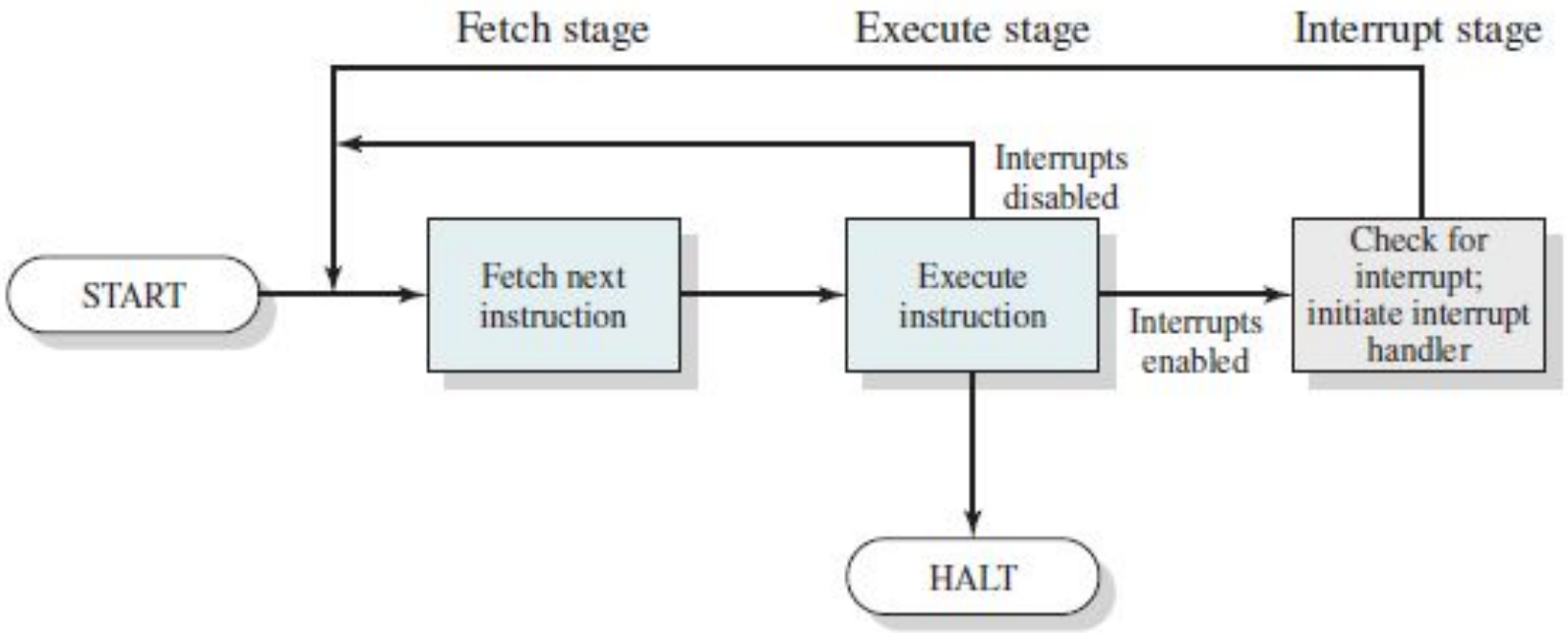


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Processing

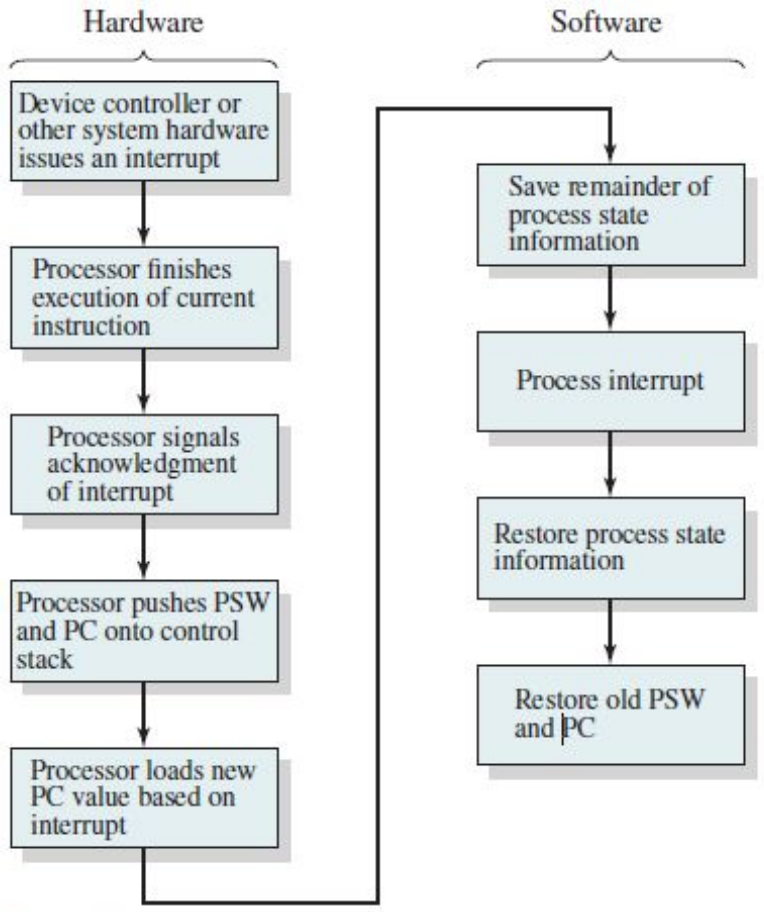


Figure 1.10 Simple Interrupt Processing

Interrupt Processing steps 1

1. The device issues an interrupt signal to the processor.
2. The processor finishes execution of the current instruction before responding to the interrupt
3. The processor tests for a pending interrupt request, determines there is one and sends an acknowledgment signal to the device that issued the interrupt.
4. The acknowledgment allows the device to remove its interrupt signal.
5. The processor next needs to prepare to transfer control to the interrupt routine.
6. To begin, it saves information needed to resume the current program at the point of interrupt.
7. The minimum information required is the program status word3 (PSW) and the location of the next instruction to be executed, which is contained in the program counter (PC).

Interrupt Processing steps 2

8. The processor then loads the program counter with the entry location of the interrupt-handling routine that will respond to this interrupt.

9. The processor then loads the program counter with the entry location of the interrupt-handling routine that will respond to this interrupt.

10. The interrupt handler may now proceed to process the interrupt.

11. When interrupt processing is complete, the saved register values are retrieved from the stack and restored to the registers

Multiple Interrupts 1

- one or more interrupts can occur while an interrupt is being processed.
- For example, a program may be receiving data from a communications line, and printing results at the same time.
- Two approaches can be taken to dealing with multiple interrupts.
  - The first is to disable interrupts while an interrupt is being processed. A *disabled interrupt*
  - simply means the processor ignores any new interrupt request signal.
  - If an interrupt occurs during this time, it generally remains pending and will be checked by the processor after the processor has re-enabled interrupts.
- The drawback to the preceding approach is that it does not take into account relative priority or time-critical needs.

Multiple Interrupts 2

- A second approach is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be interrupted

THE MEMORY HIERARCHY

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed

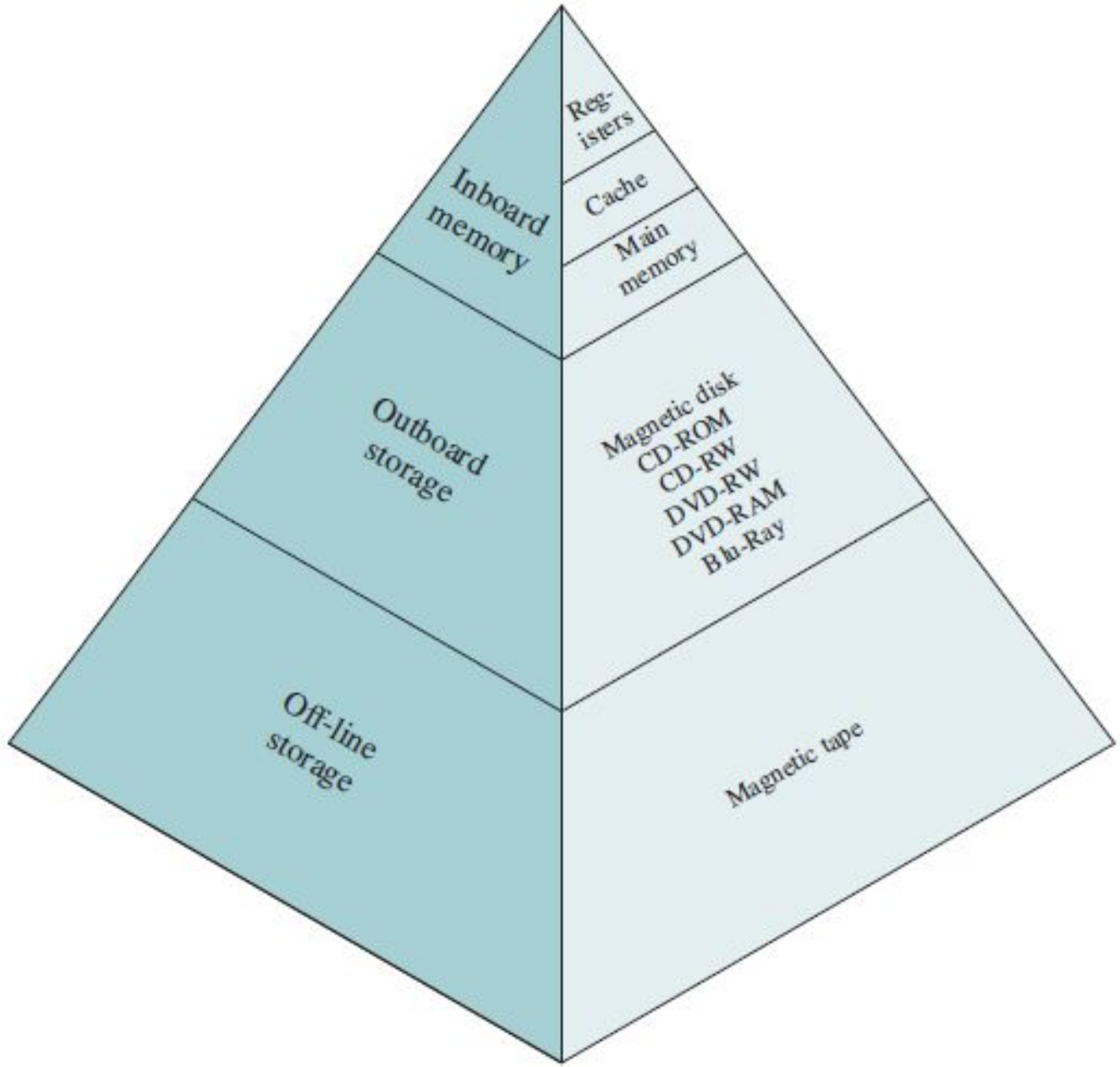


Figure 1.14 The Memory Hierarchy