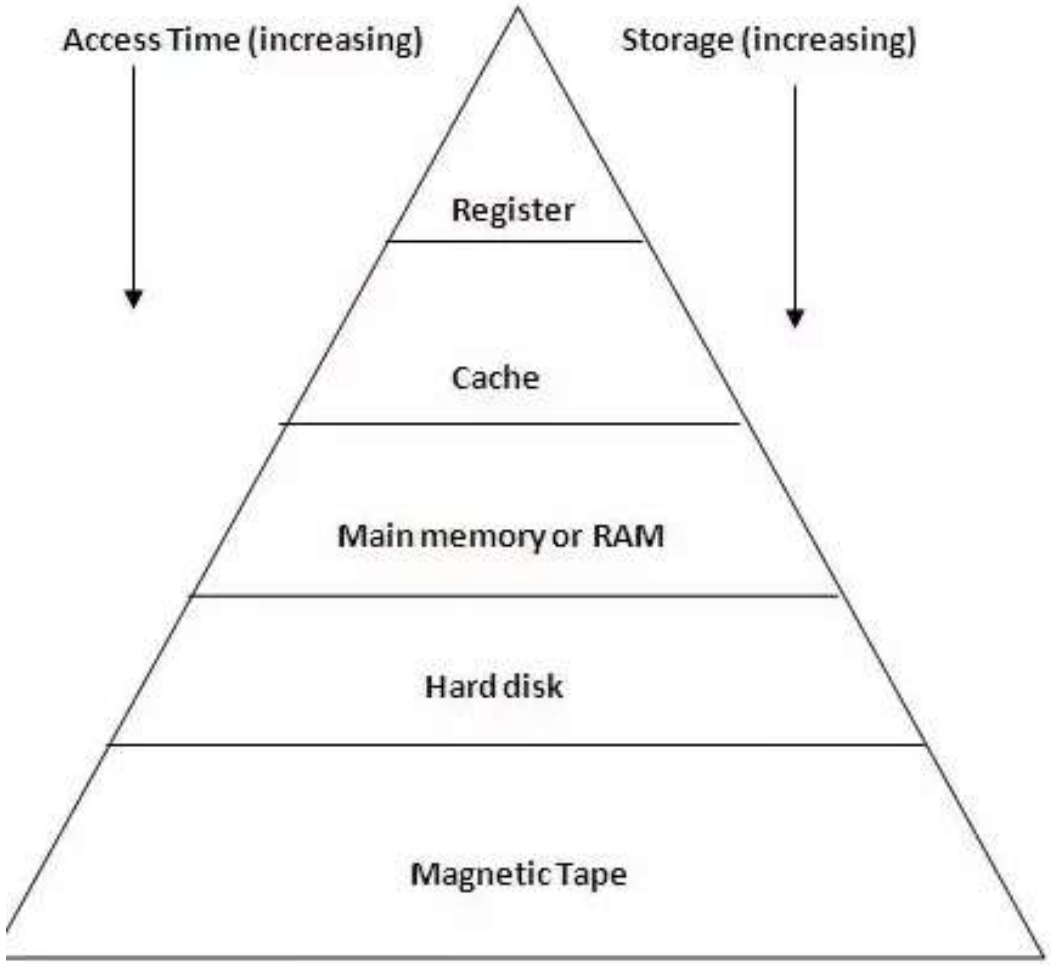
Memory Hierarchy

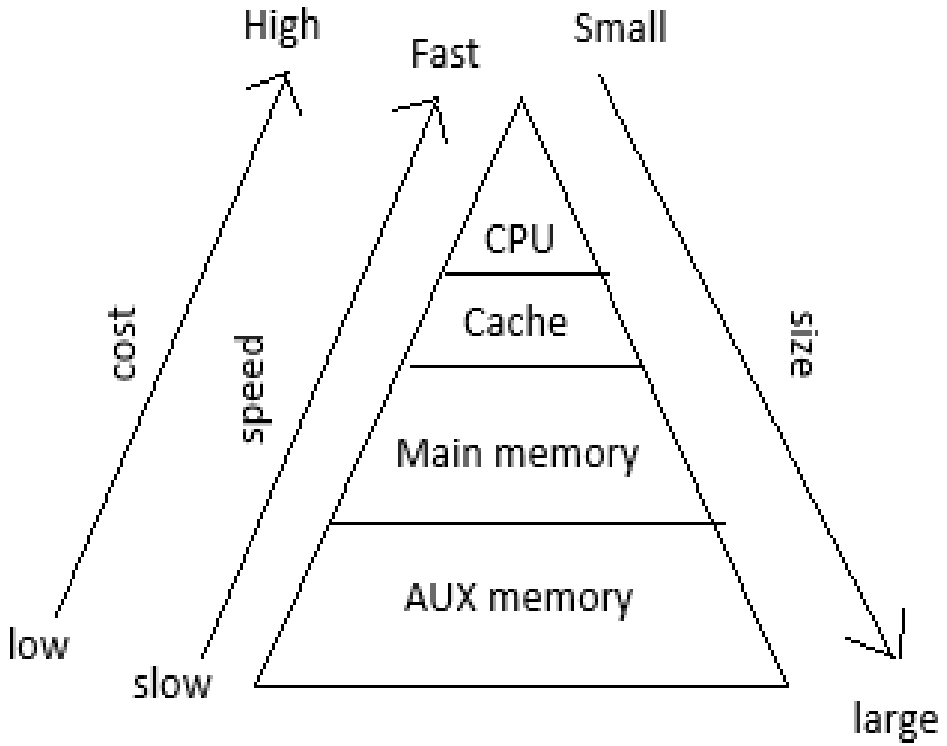
Dr.Pritee Parwekar

Memory Hierarchy

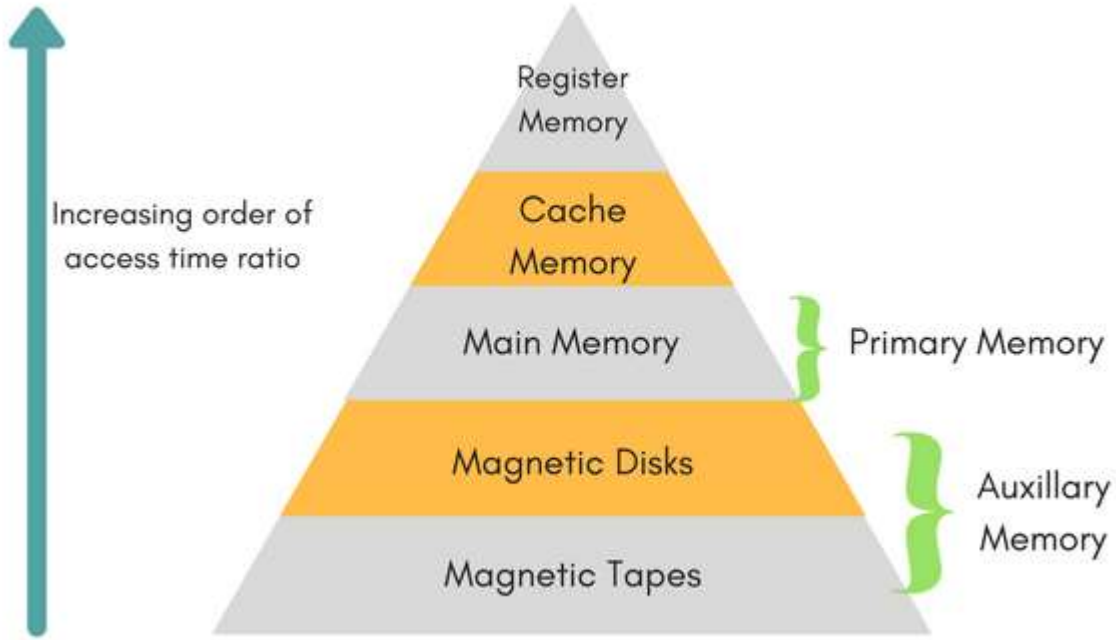
- Major constraints in memory
  - Amount
  - Speed
  - Expense
- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed

Memory Hierarchy



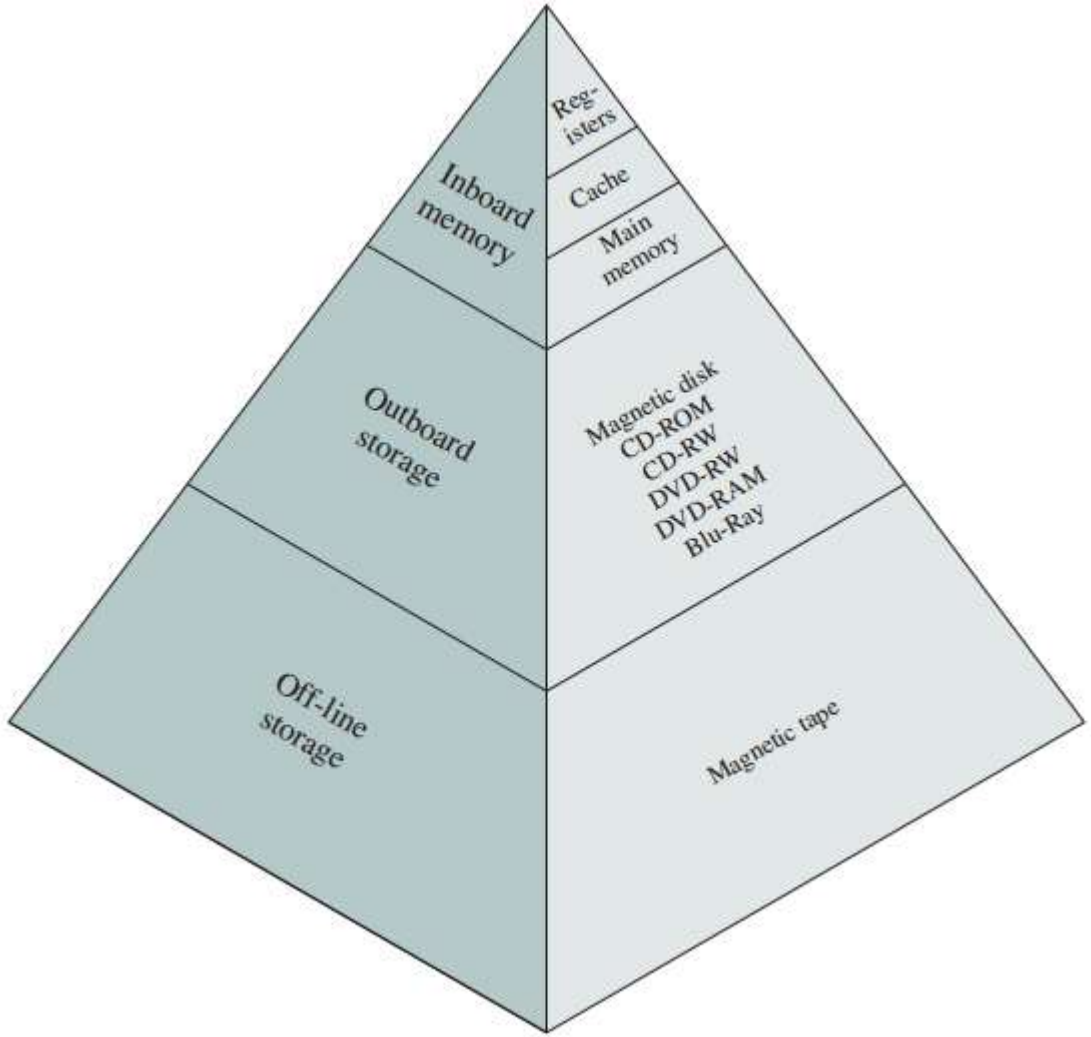


Memory chart



The Memory Hierarchy

- Going down the hierarchy
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time
  - Decreasing frequency of access to the memory by the processor



Secondary Memory

- Auxiliary memory
- External
- Nonvolatile
- Used to store program and data files

Instruction Execution

- A program consists of a set of instructions stored in memory
- Two steps
  - Processor reads (fetches) instructions from memory
  - Processor executes each instruction

Computer Components Top-Level View

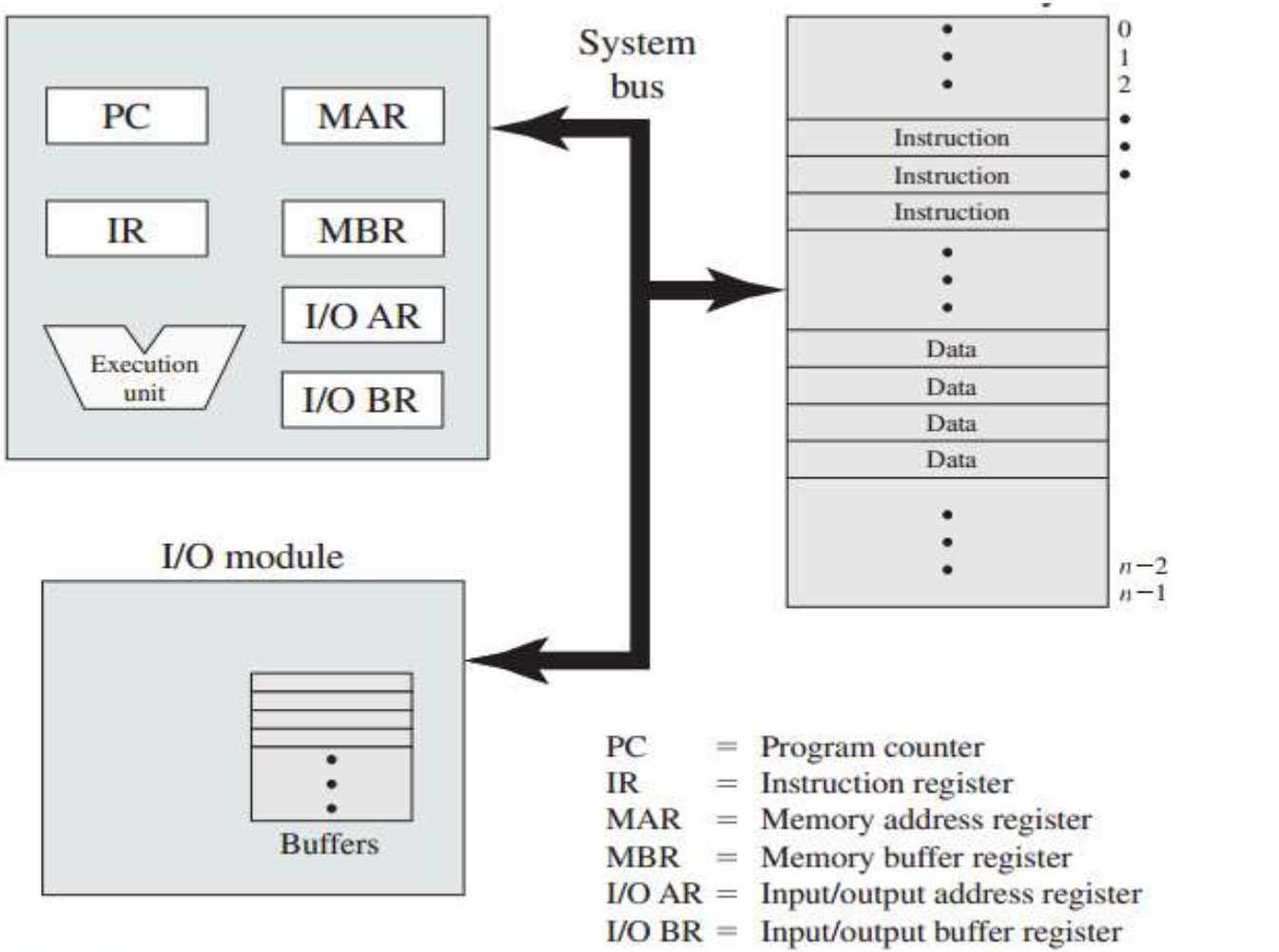


Figure 1.1 Computer Components: Top-Level View

Basic Instruction Cycle

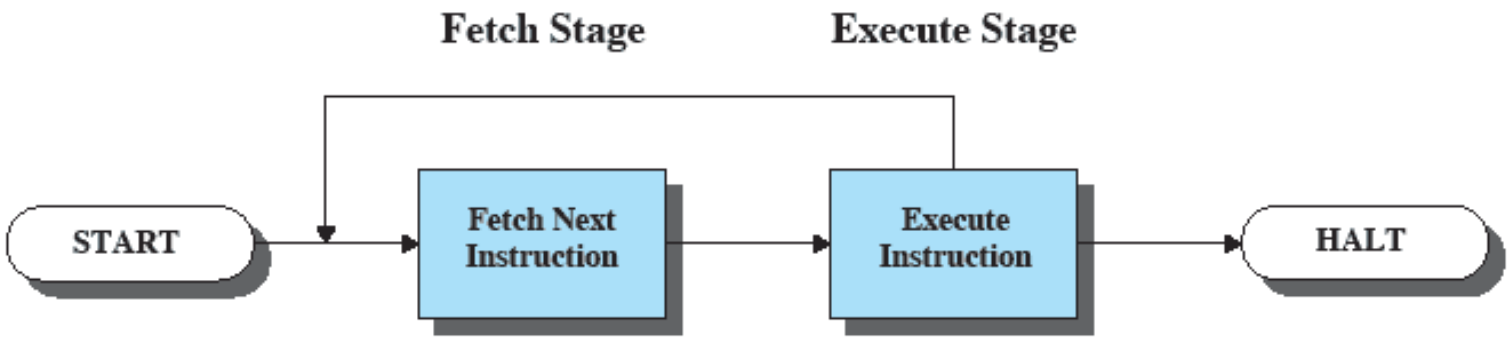


Figure 1.2 Basic Instruction Cycle

Fetch cycle

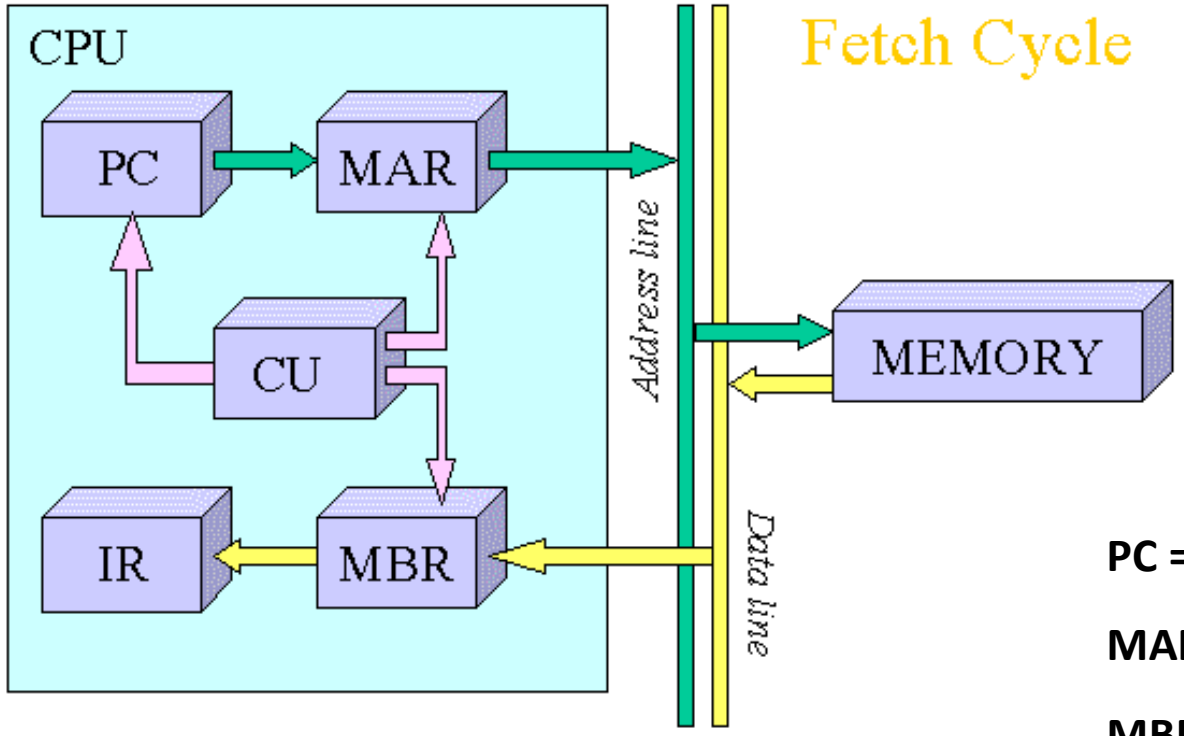


Figure 3: An animated fetch cycle

PC => MAR

MAR => memory => MBR

MBR => IR

PC incremented

Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
  - PC is incremented after each fetch

Fetch Cycle

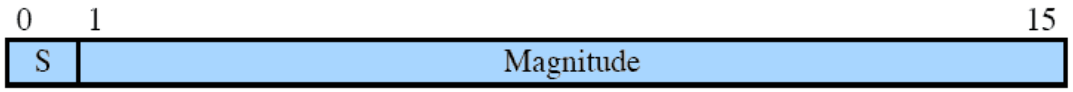
- To start off the fetch cycle, the address which is stored in the program counter (PC) is transferred to the memory address register (MAR).
- The CPU then transfers the instruction located at the address stored in the MAR to the memory buffer register (MBR) via the data lines connecting the CPU to memory.
- This transfer from memory to CPU is coordinated by the control unit (CU).
- To finish the cycle, the newly fetched instruction is transferred to the instruction register (IR) and unless told otherwise, the CU increments the PC to point to the next address location in memory

Instruction Register

- Fetched instruction loaded into instruction register
- Categories
  - Processor-memory,
  - processor-I/O,
  - Data processing,
  - Control

Characteristics of a Hypothetical Machine

(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

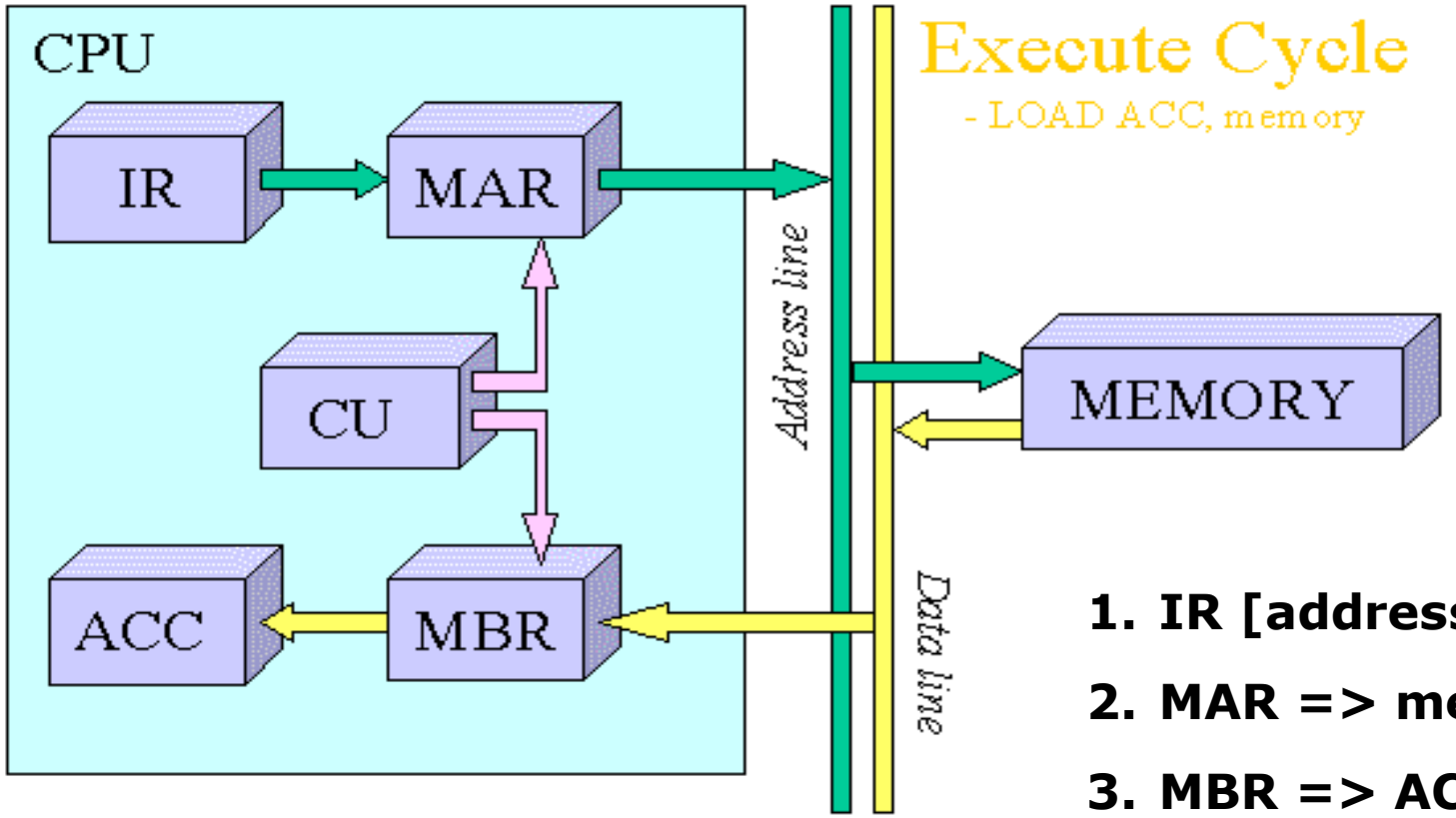
Execution cycle - Actions

- CPU - Memory: Data may be transferred from memory to the CPU or from the CPU to memory.
- CPU - I/O: Data may be transferred from an I/O module to the CPU or from the CPU to an I/O module.
- Data Processing: The CPU may perform some arithmetic or logic operation on data via the arithmetic-logic unit (ALU).
- Control: An instruction may specify that the sequence of operation may be altered. For example, the program counter (PC) may be updated with a new memory address to reflect that the next instruction fetched, should be read from this new location.

Example of execute cycle ,for instruction LOAD ACC, memory

- This operation loads the accumulator (ACC) with data that is stored in the memory location specified in the instruction.
- The operation starts off by transferring the address portion of the instruction from the IR to the memory address register (MAR).
- The CPU then transfers the instruction located at the address stored in the MAR to the memory buffer register (MBR) via the data lines connecting the CPU to memory.
- This transfer from memory to CPU is coordinated by the CU. To finish the cycle, the newly fetched data is transferred to the ACC.

Example of execution cycle for LOAD ACC, memory



1. **IR [address portion] => MAR**
2. **MAR => memory => MBR**
3. **MBR => ACC**

Figure 4: Animated Execute Cycle [LOAD ACC, memory] operation

Example of Program Execution

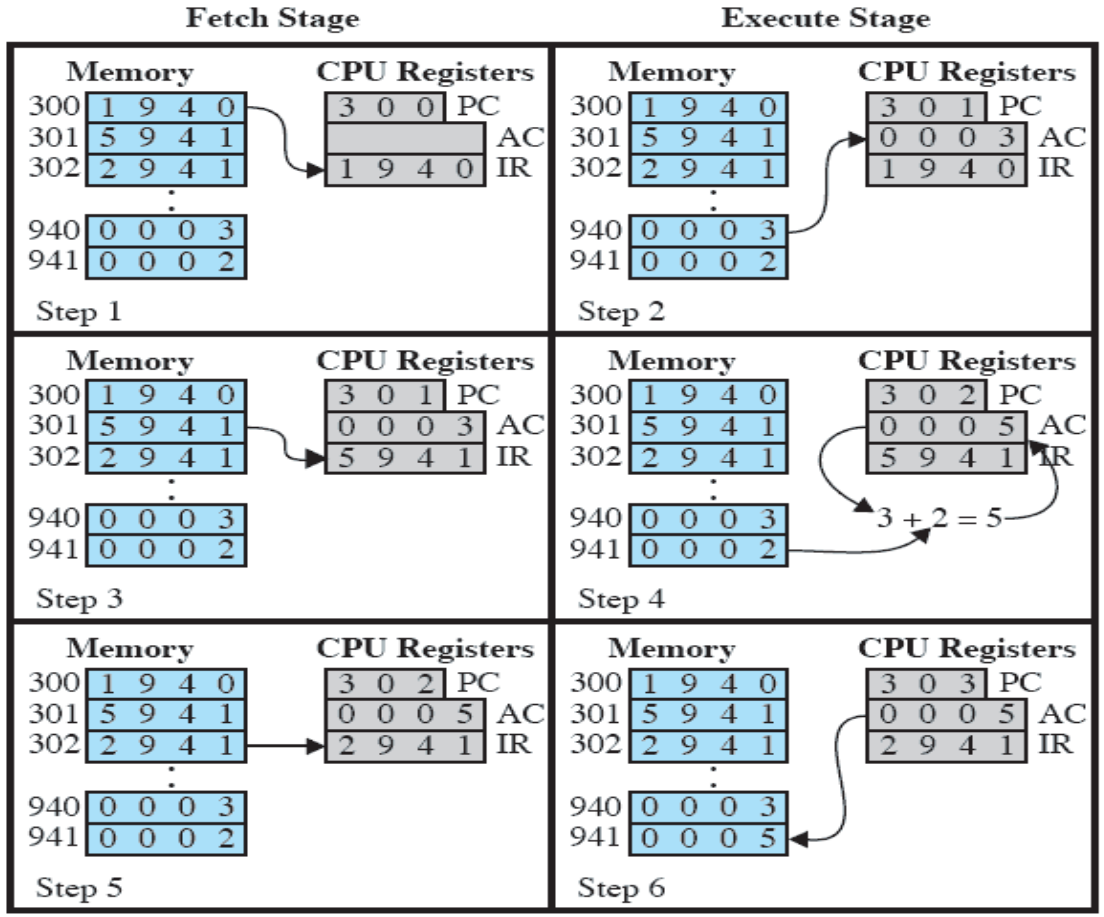


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

Interrupts

- It is a signal that gets the attention of the CPU and is usually generated when I/O is required. (or any class of interrupts)
- Hardware interrupts are generated when a key is pressed or when the mouse is moved.
- Software interrupts are generated by a program requiring disk input or output.

Interrupts

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, and 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.

Transfer of control via Interrupts

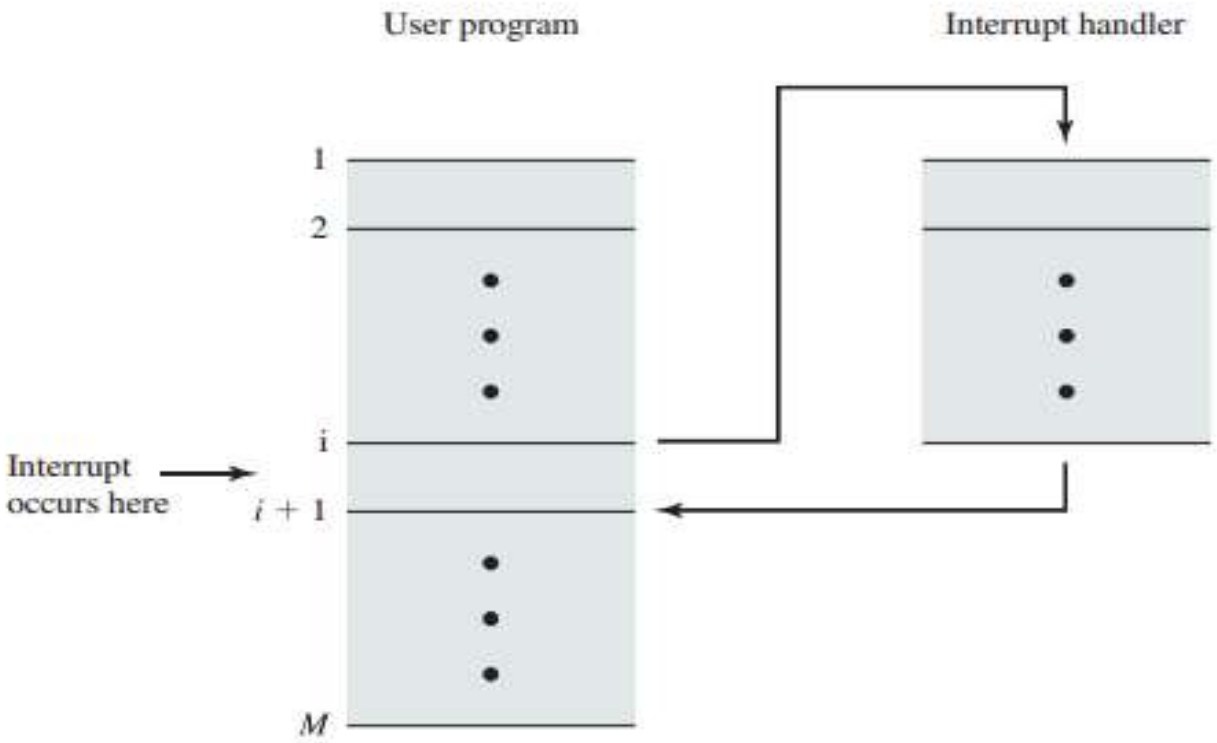


Figure 1.6 Transfer of Control via Interrupts

Program flow with and without interrupts

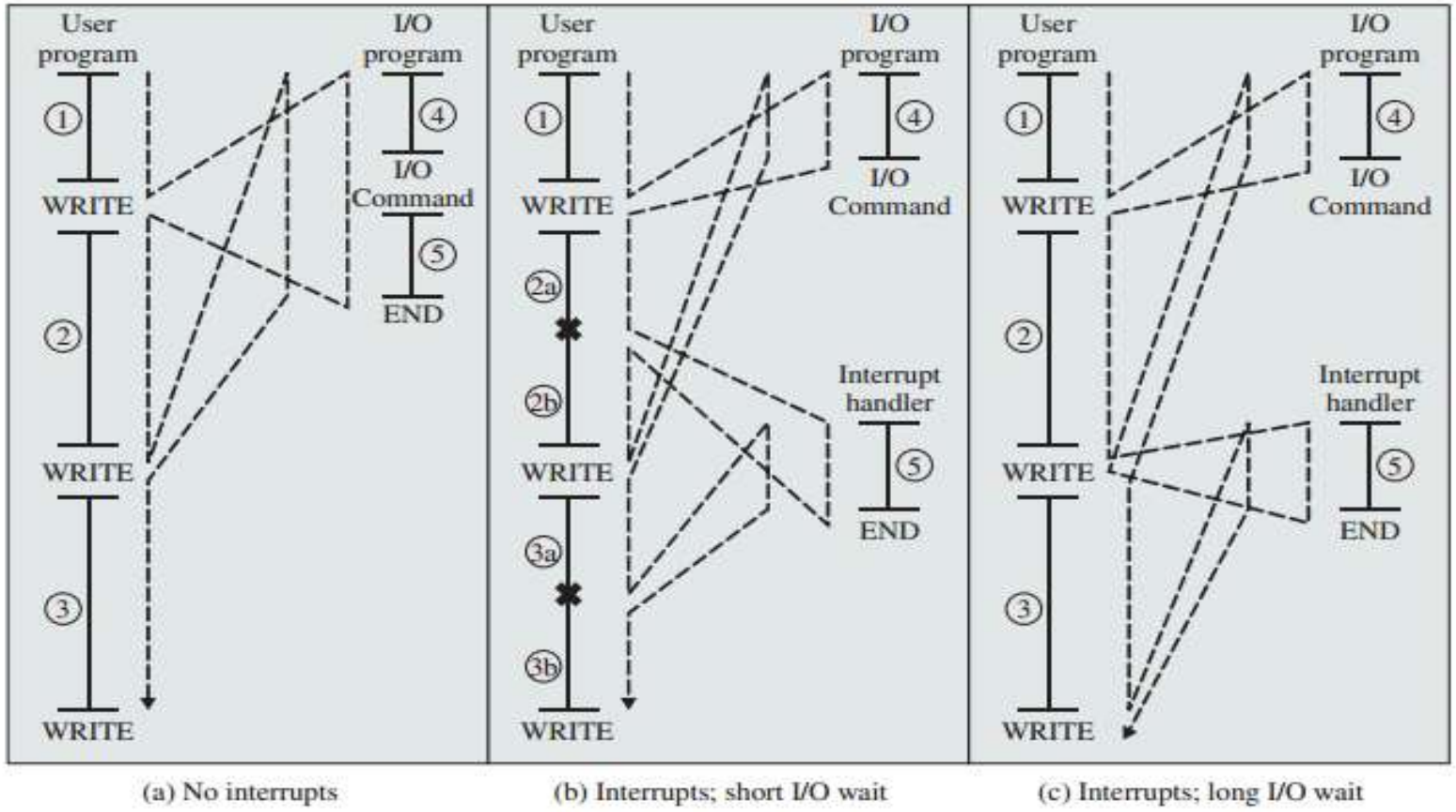


Figure 1.5 Program Flow of Control without and with Interrupts

Interrupt Processing

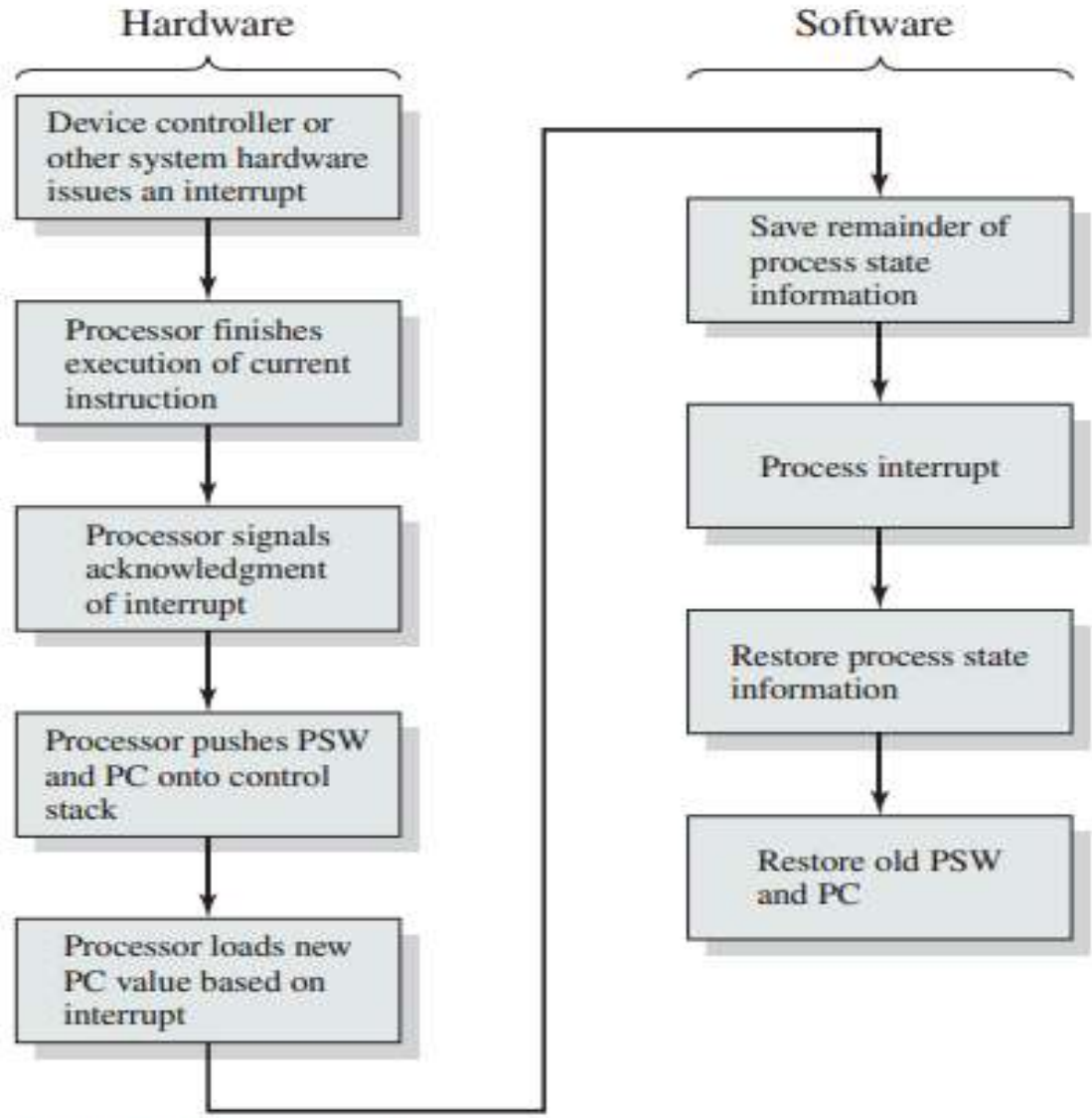


Figure 1.10 Simple Interrupt Processing

Instruction Cycle with Interrupt

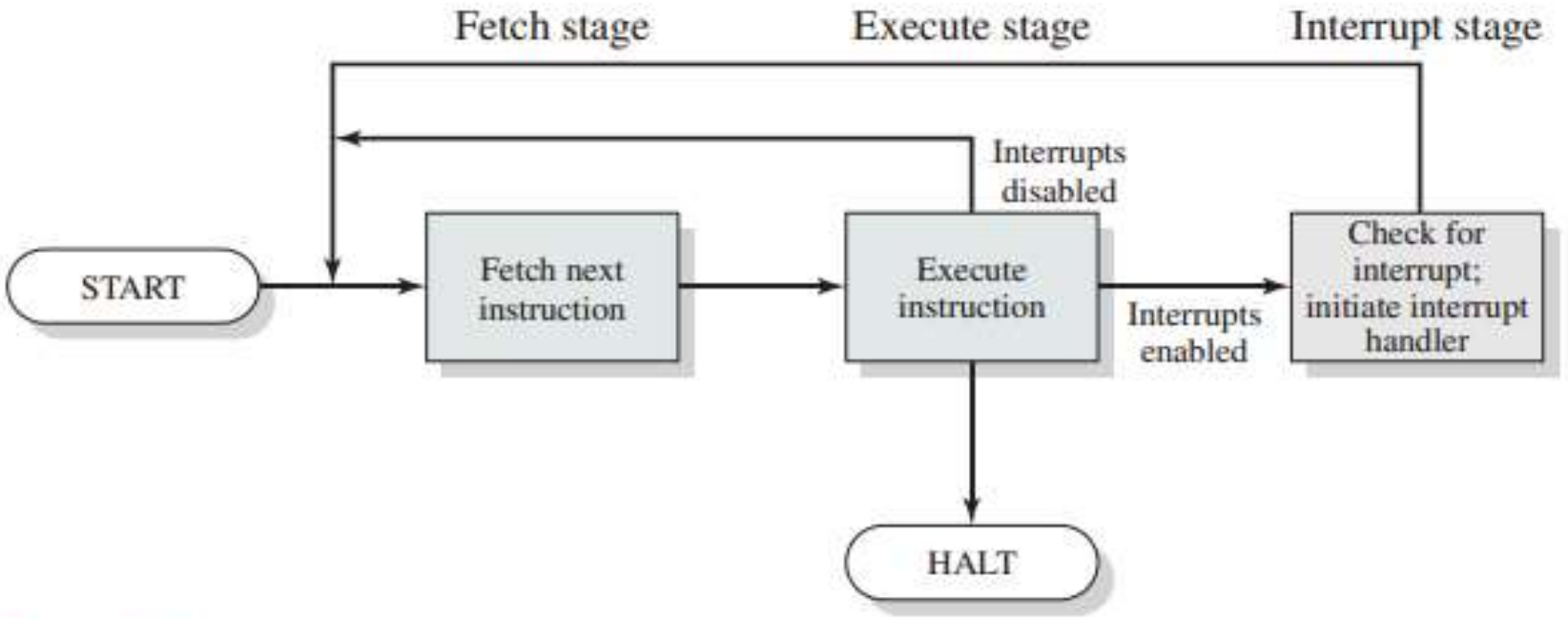


Figure 1.7 Instruction Cycle with Interrupts

References

- William Stallings, “Operating Systems – internals and design principles”, Prentice Hall, 7thEdition, 2011.
- William Stallings “Operating Systems – Internals and design principles”, Pearson Education, 5th Edition.