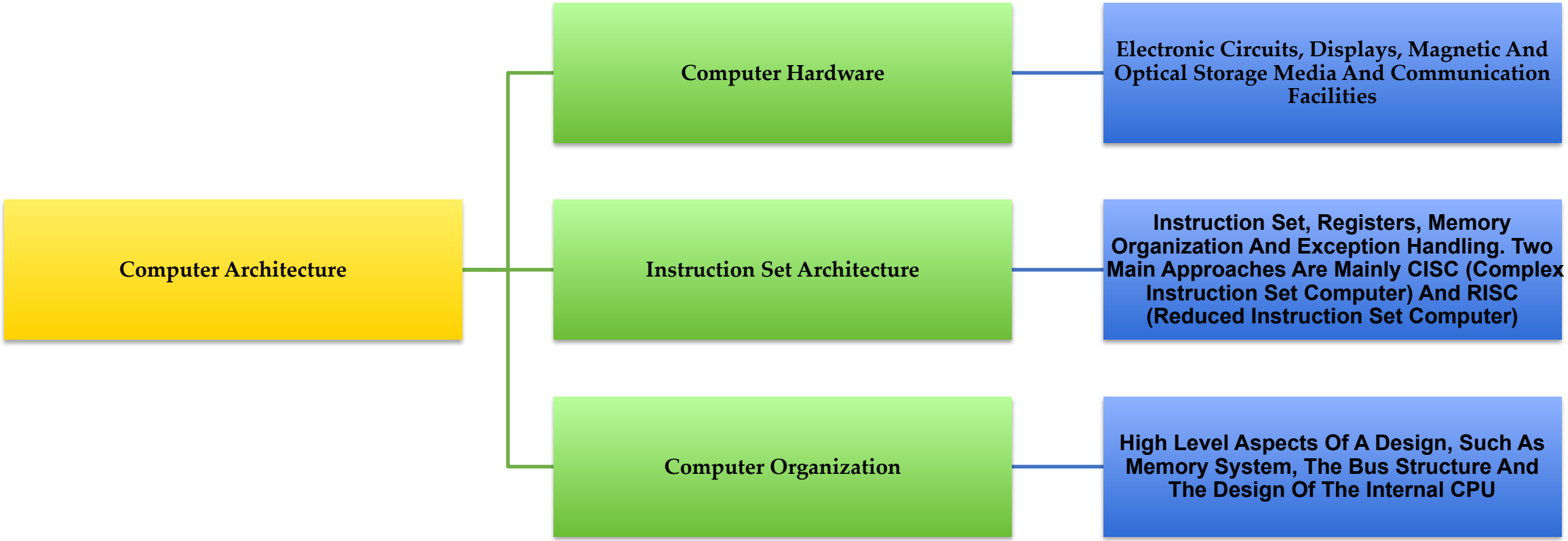
21CSS201T
COMPUTER ORGANIZATION AND ARCHITECTURE

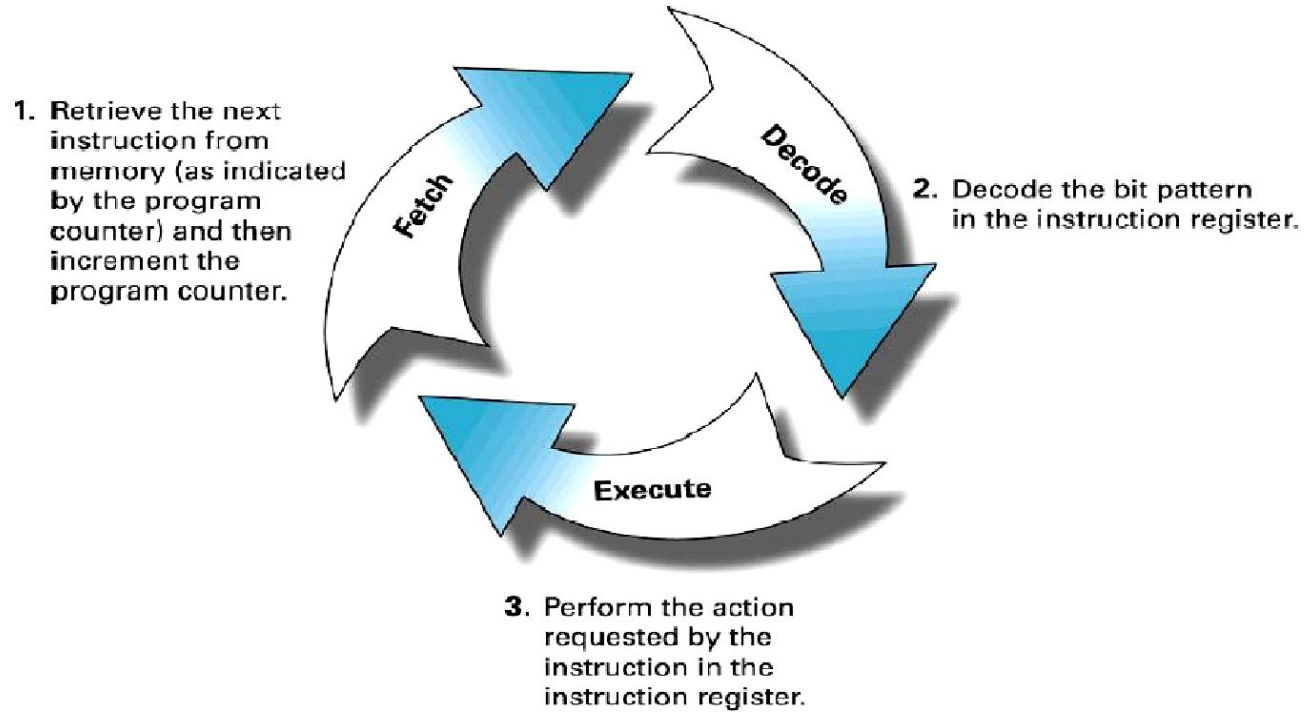
UNIT-2
Topic : Basic Structure of a Computer

Basic Structure of Computers



Computer Types

- Computer is a fast electronic calculating machine which accepts digital input, processes it according to the internally stored instructions (Programs) and produces the result on the output device. The internal operation of the computer can be as depicted in the figure below:



Computer Types – contd.

Micro Computer

- A personal computer; designed to meet the computer needs of an individual. Provides access to a wide variety of computing applications, such as word processing, photo editing, e-mail, and internet.

Laptop Computer

- A portable, compact computer that can run on power supply or a battery unit. All components are integrated as one compact unit. It is generally more expensive than a comparable desktop. It is also called a Notebook.

Work Station

- Powerful desktop computer designed for specialized tasks. Generally used for tasks that requires a lot of processing speed. Can also be an ordinary personal computer attached to a LAN (local area network).

Super Computer

- A computer that is considered to be fastest in the world. Used to execute tasks that would take lot of time for other computers. For Ex: Modelling weather systems, genome sequence, etc (Refer site: http://www.top500.org/)

Main Frame

- Large expensive computer capable of simultaneously processing data for hundreds or thousands of users. Used to store, manage, and process large amounts of data that need to be reliable, secure, and centralized.

Hand Held

- It is also called a PDA (Personal Digital Assistant). A computer that fits into a pocket, runs on batteries, and is used while holding the unit in your hand. Typically used as an appointment book, address book, calculator and notepad.

Multi core

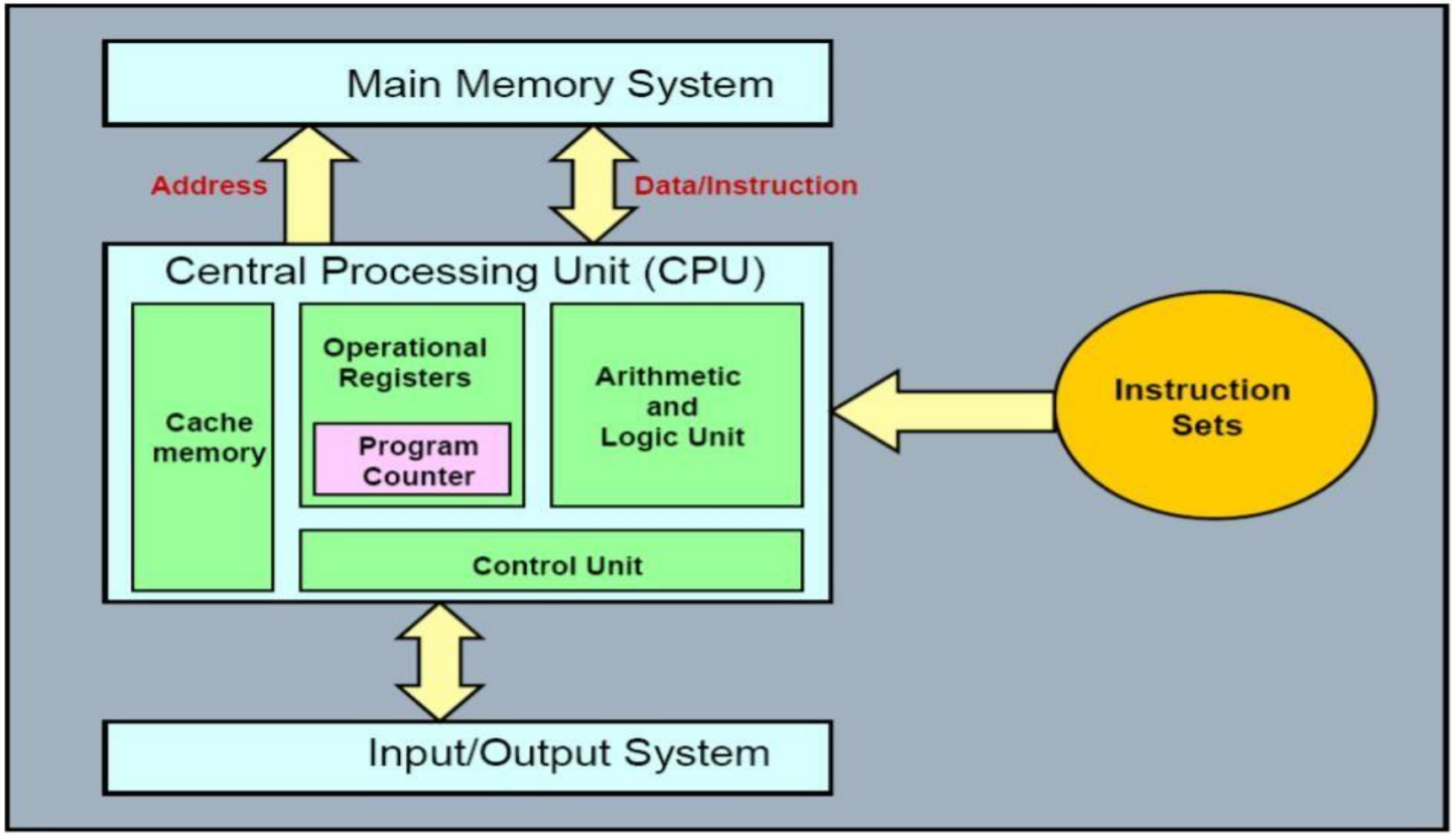
- Have Multiple Cores – parallel computing platforms. Many Cores or computing elements in a single chip. Typical Examples: Sony Play station, Core 2 Duo, i3, i7 etc.

Functional Units of a Computer

FUNCTIONAL UNITS OF COMPUTER

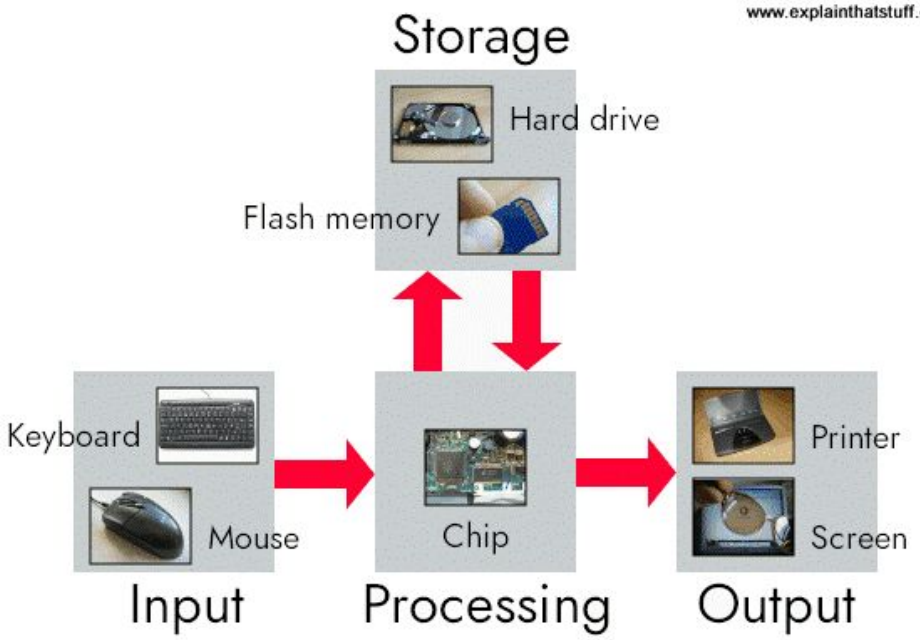
- **Input Unit**
- **Output Unit**
- **Central processing Unit (ALU and Control Units)**
- **Memory**
- **Bus Structure**

Basic Functional Unit of a Computer



What is a computer?

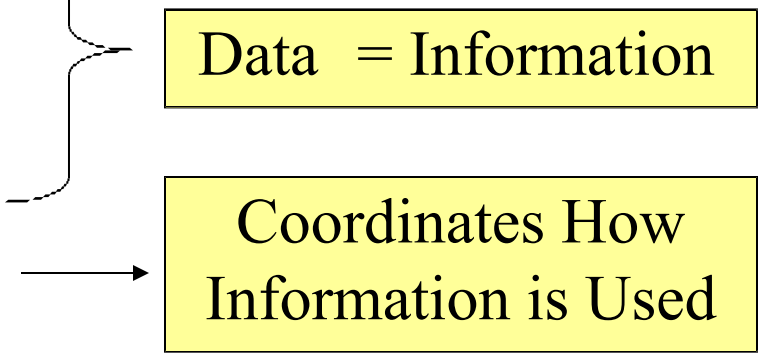
- a computer is a sophisticated electronic calculating machine that:
  - Accepts input information,
  - Processes the information according to a list of internally stored instructions and
  - Produces the resulting output information.
- Functions performed by a computer are:
  - Accepting information to be processed as input.
  - Storing a list of instructions to process the information.
  - Processing the information according to the list of instructions.
  - Providing the results of the processing as output.



Functions

- **ALL computer functions are:**

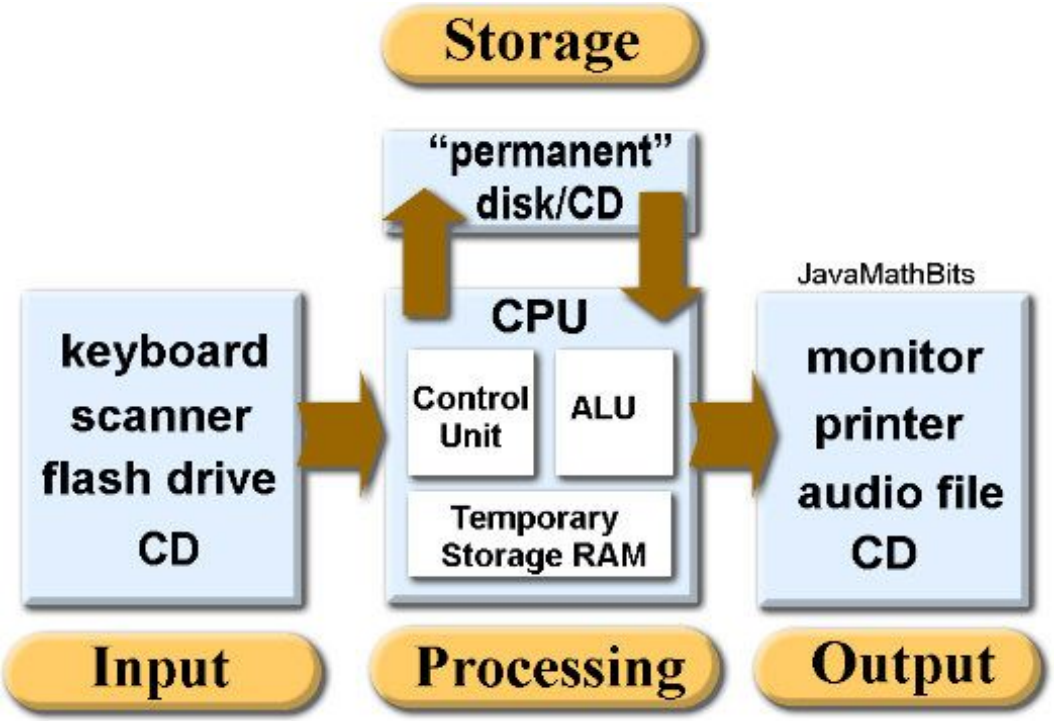
  - **Data PROCESSING**
  - **Data STORAGE**
  - **Data MOVEMENT**
  - **CONTROL**



Functions of a computer

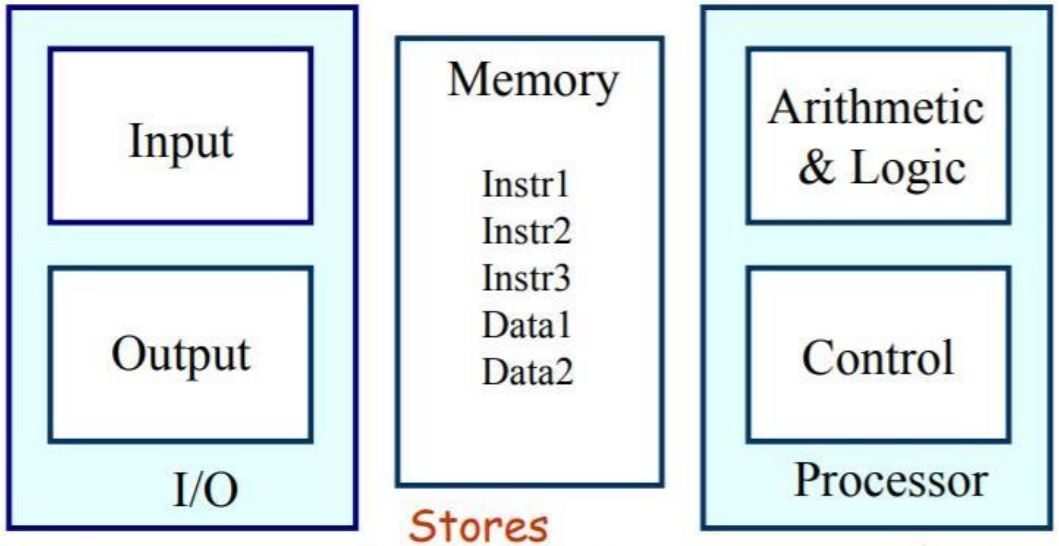
The operations performed by a computer using the functional units can be summarized as follows:

- It accepts information (program and data) through input unit and transfers it to the memory.
- Information stored in the memory is fetched, under program control, into an arithmetic and logic unit for processing.
- Processed information leaves the computer through an output unit.
- The control unit controls all activities taking place inside a computer.



Input unit accepts information:
- Human operators,
- Electromechanical devices (keyboard)
- Other computers

Arithmetic and logic unit(ALU):
- Performs the desired operations on the input information as determined by instructions in the memory



Output unit sends results of processing:
- To a monitor display,
- To a printer

Stores information:
- Instructions,
- Data

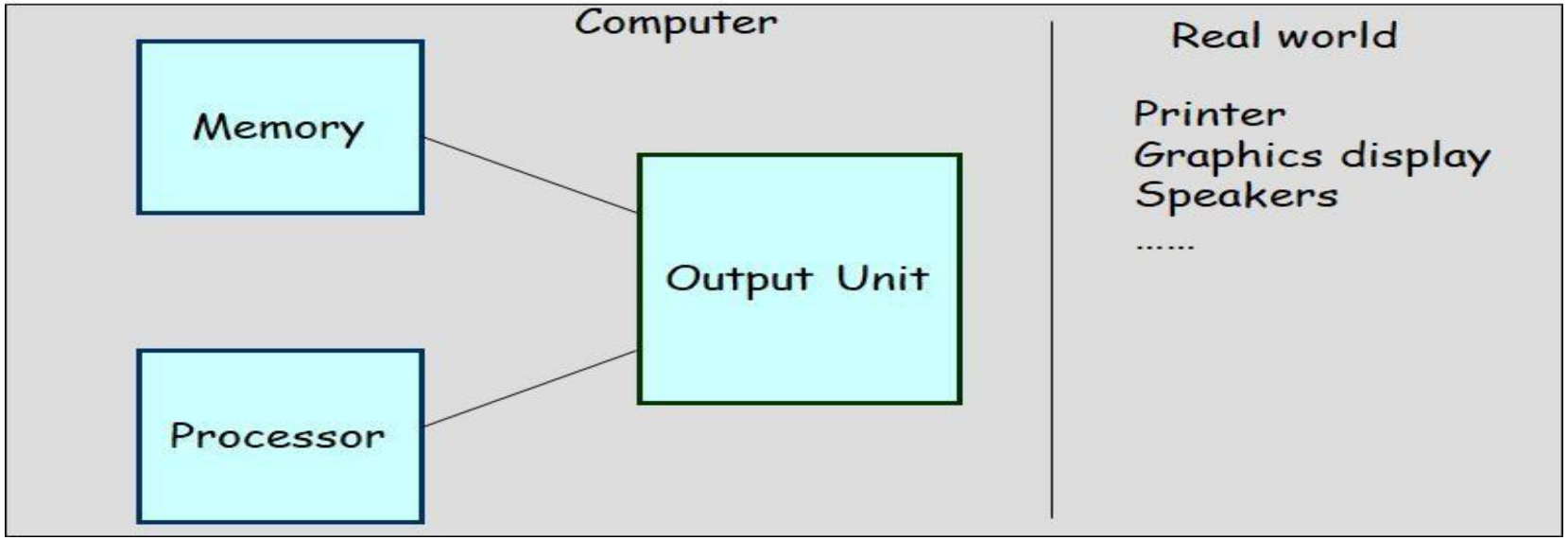
Control unit coordinates various actions
- Input,
- Output
- Processing

Arithmetic and Logic Unit (ALU)

- Operations are executed in the Arithmetic and Logic Unit (ALU).
  - Arithmetic operations such as addition, subtraction.
  - Logic operations such as comparison of numbers.
- In order to execute an instruction, operands need to be brought into the ALU from the memory.
  - Operands are stored in general purpose registers available in the ALU.
  - Access times of general purpose registers are faster than the cache.
- Results of the operations are stored back in the memory or retained in the processor for immediate use.

Output unit

•Computers represent information in a specific binary form. Output units:
- Interface with output devices.
- Accept processed results provided by the computer in specific binary form.
- Convert the information in binary form to a form understood by an output device.



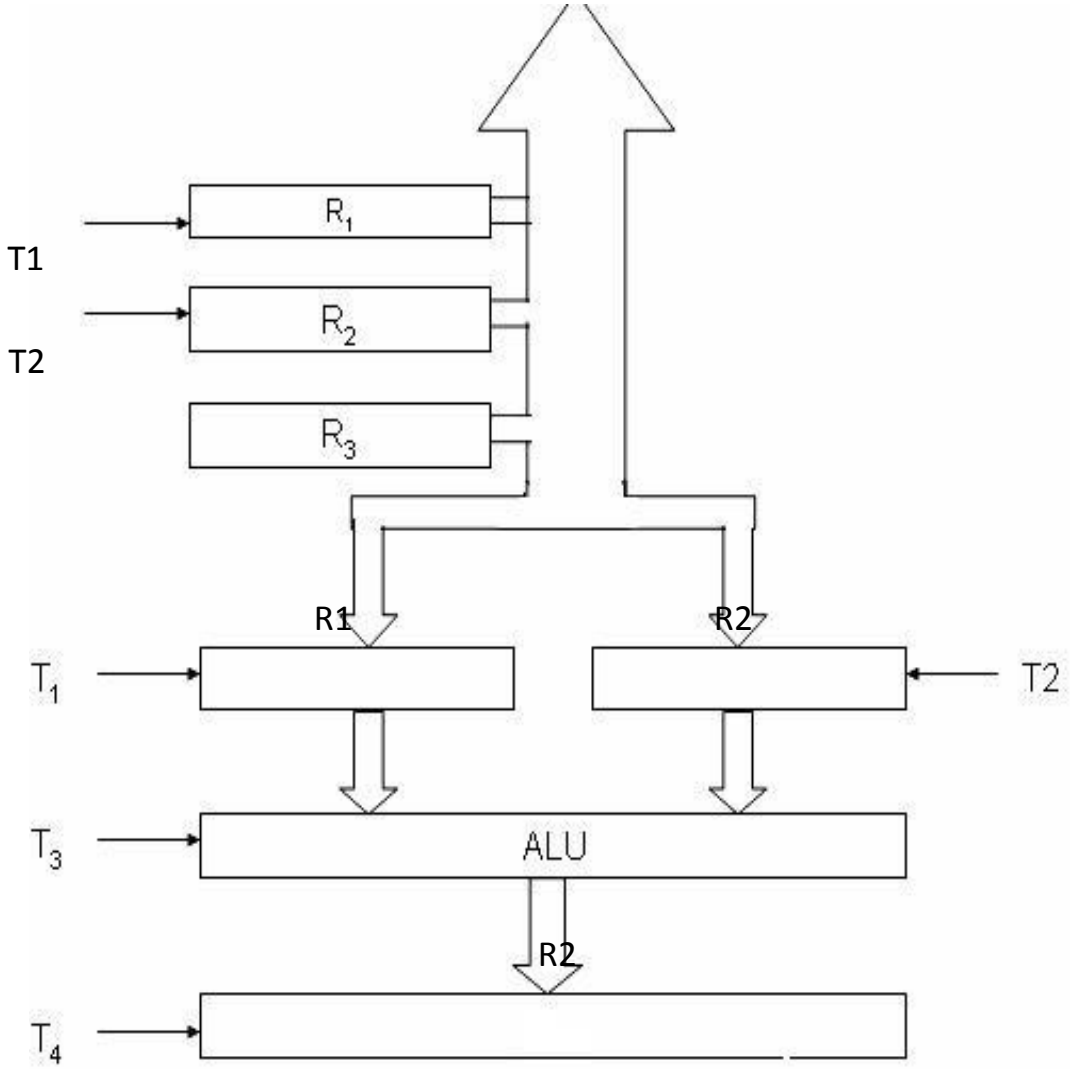
Control unit

- Operation of a computer can be summarized as:
  - Accepts information from the input units (Input unit).
  - Stores the information (Memory).
  - Processes the information (ALU).
  - Provides processed results through the output units (Output unit).
- Operations of Input unit, Memory, ALU and Output unit are coordinated by Control unit.
- Instructions control “what” operations take place (e.g. data transfer, processing).
- Control unit generates timing signals which determines “when” a particular operation takes place.

CPU (Central processing Unit)

- The “brain” of the machine
- Responsible for carrying out computational task
- Contains ALU, CU, Registers
- ALU Performs Arithmetic and logical operations
- CU Provides control signals in accordance with some timings which in turn controls the execution process
- Register Stores data and result and speeds up the operation

CONTROL UNIT



- Control unit works with a reference signal called Processor clock
- Processor divides the operations into basic steps
- Each basic step is executed in one clock cycle

Example

Add R1, R2

T1 → **Enable R1**

T2 → **Enable R2**

T3 → **Enable ALU for addition operation**

T4 → **Enable out put of ALU to store result of the operation**

Operational Concepts

Basic Function of Computer

- To Execute a given task as per the appropriate program
- Program consists of list of instructions stored in memory

A Typical Instruction

Add R0, LOCA

- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- Place the sum into register R0.
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction is fetched from the memory into the processor – the operand at LOCA is fetched and added to the contents of R0 – the resulting sum is stored in register R0.

Load R2, LOC

This instruction reads the contents of a memory location by the label LOC and loads them into processor register R2.

The original contents of location LOC are preserved, whereas those of register R2 are overwritten.

Execution of this instruction requires several steps.

First, the instruction is fetched from the memory into the processor.

Next, the operation to be performed is determined by the control unit.

The operand at LOC is then fetched from the memory into the processor.

Finally, the operand is stored in register R2

Add R4, R2, R3

adds the contents of registers R2 and R3, then places their sum into register R4.

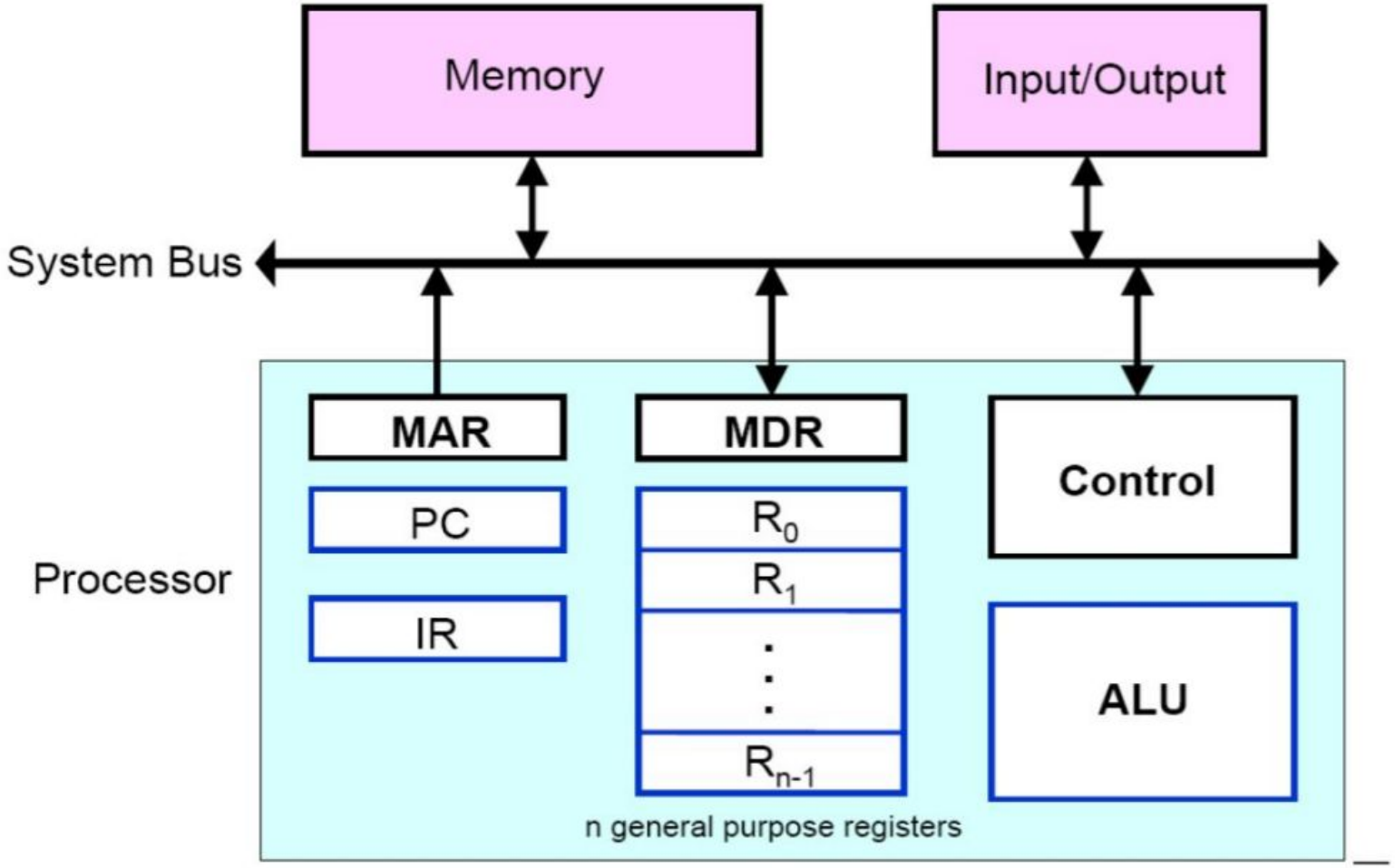
The operands in R2 and R3 are not altered, but the previous value in R4 is overwritten by the sum.

Store R4, LOC

This instruction copies the operand in register R4 to memory location LOC.

The original contents of location LOC are overwritten, but those of R4 are preserved.

For Load and Store instructions, transfers between the memory and the processor are initiated by sending the address of the desired memory location to the memory unit and asserting the appropriate control signals.



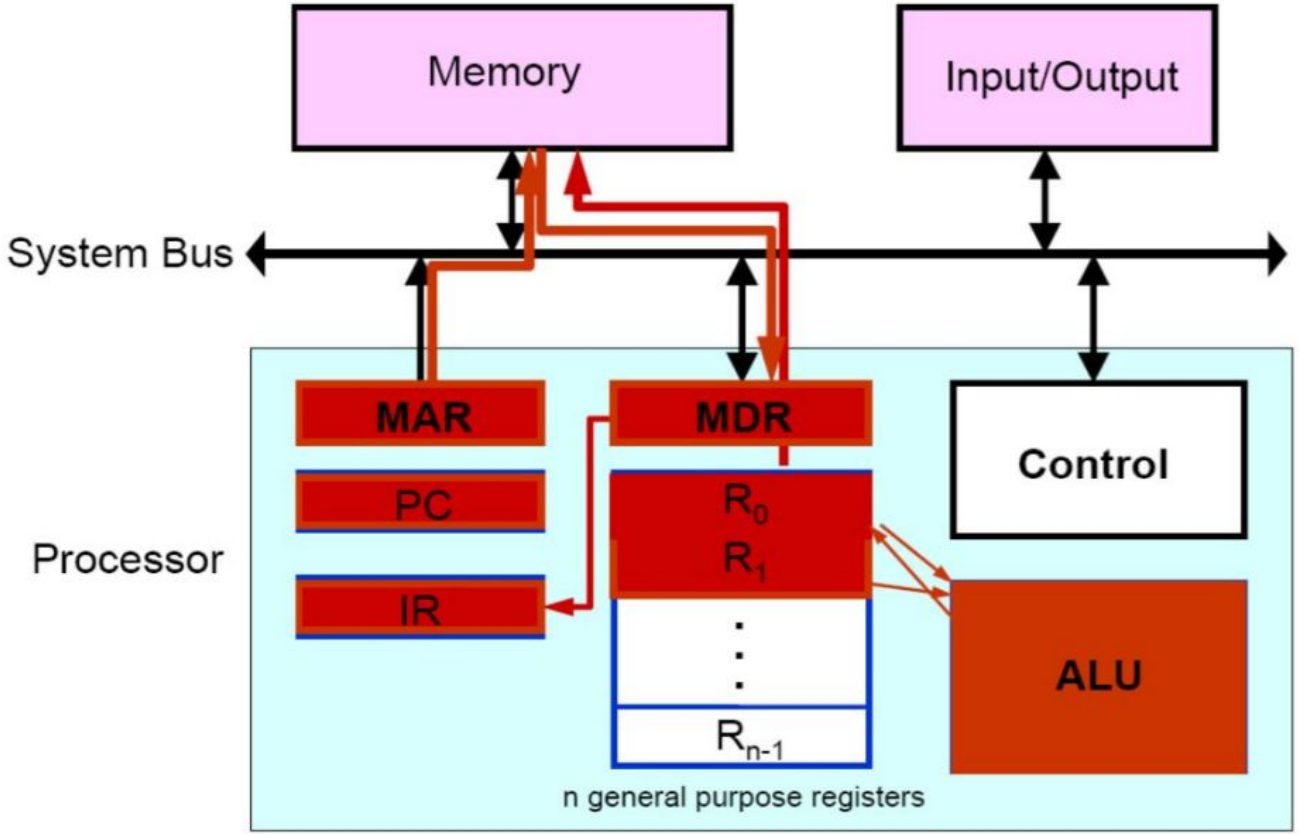
Connections between Processor and memory

Registers

Registers are fast stand-alone storage locations that hold data temporarily. Multiple registers are needed to facilitate the operation of the CPU. Some of these registers are

- **Two registers-MAR (Memory Address Register) and MDR (Memory Data Register) : To handle the data transfer between main memory and processor. MAR-Holds addresses, MDR-Holds data**
- **Instruction register (IR) : Hold the Instructions that is currently being executed**
- **Program counter (PC) : Points to the next instructions that is to be fetched from memory**
- **General-purpose Registers: are used for holding data, intermediate results of operations. They are also known as scratch-pad registers.**

Basic Operational Concepts



Instruction Fetch – Steps Involved

- Program gets into the memory through an input device.
- Execution of a program starts by setting the PC to point to the first instruction of the program.
- The contents of PC are transferred to the MAR and a Read control signal is sent to the memory.
- The addressed word (here it is the first instruction of the program) is read out of memory and loaded into the MDR.
- The contents of MDR are transferred to the IR for instruction decoding

Instruction Execution – Steps Involved

- The operation field of the instruction in IR is examined to determine the type of operation to be performed by the ALU.
- The specified operation is performed by obtaining the operand(s) from the memory locations or from GP registers.

1) Fetching the operands from the memory requires sending the memory location address to the MAR and initiating a Read cycle.

2) The operand is read from the memory into the MDR and then from MDR to the ALU.

Instruction Execution – Steps Involved (Contd..)

3) The ALU performs the desired operation on one or more operands 13 fetched in this manner and sends the result either to memory location or to a GP register.

4) The result is sent to MDR and the address of the location where the result is to be stored is sent to MAR and Write cycle is initiated.

Thus, the execute cycle ends for the current instruction and the PC is incremented to point to the next instruction for a new fetch cycle.

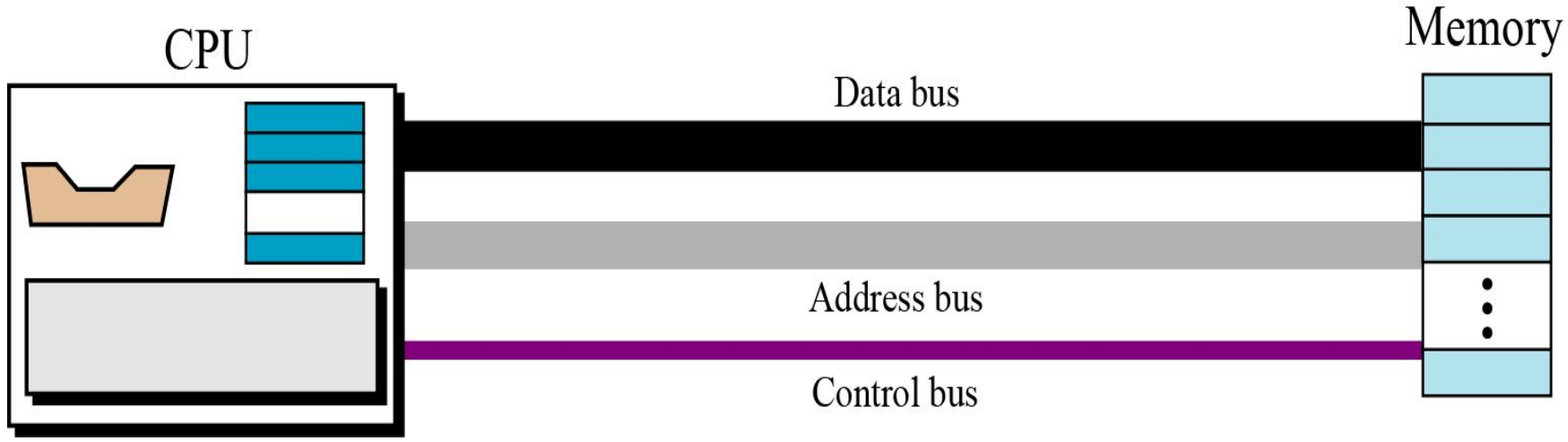
Interrupt

- An interrupt is a request from I/O device for service by processor
- Processor provides requested service by executing interrupt service routine (ISR)
- Contents of PC, general registers, and some control information are stored in memory .
- When ISR completed, processor restored, so that interrupted program may continue

Bus Structures

Connecting CPU and memory

The CPU and memory are normally connected by three groups of connections, each called a **bus**: *data bus*, *address bus* and *control bus*

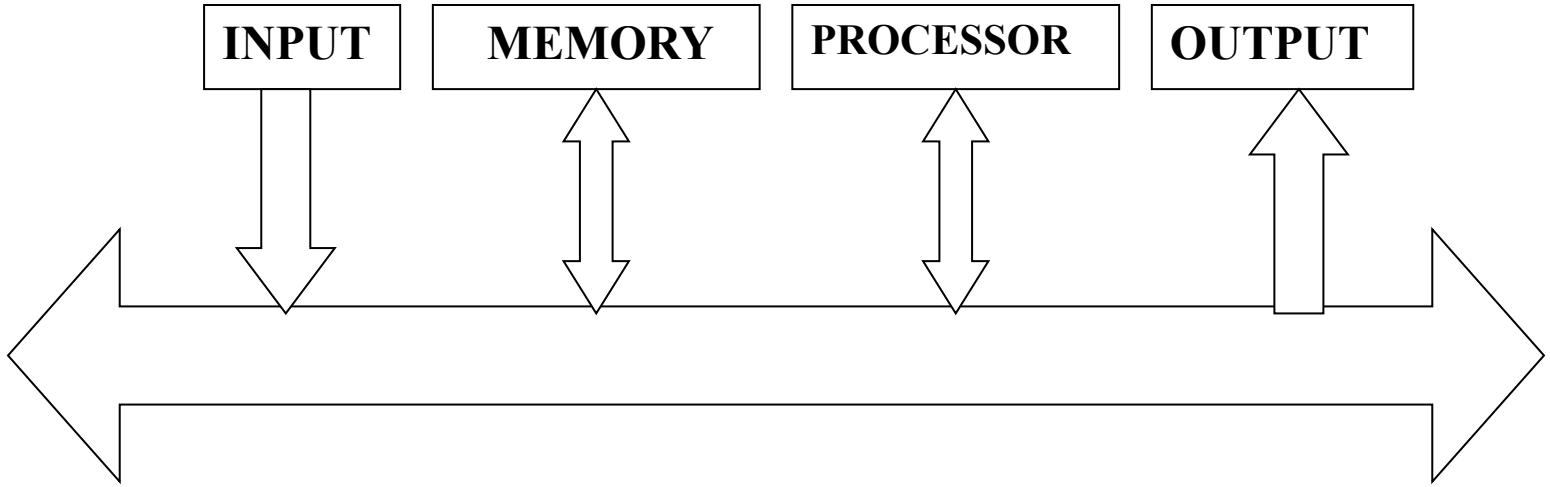


Connecting CPU and memory using three buses

BUS STRUCTURE

•Group of wires which carries information form CPU to peripherals or vice – versa

•**Single bus structure**: Common bus used to communicate between peripherals and microprocessor



Single Bus Structure

Drawbacks of the Single Bus Structure

- The devices connected to a bus vary widely in their speed of operation.
  - Some devices are relatively slow, such as printer and keyboard.
  - Some devices are considerably fast, such as optical disks.
  - Memory and processor units operate are the fastest parts of a computer.
- Efficient transfer mechanism thus is needed to cope with this problem.
  - A common approach is to include buffer registers with the devices to hold the information during transfers .
  - An another approach is to use two-bus structure and an additional transfer mechanism

TWO BUS STRUCTURE:

• One bus can be used to fetch instruction other can be used to fetch data, required for execution.

•The bus is said to perform two distinct functions

•The main advantage of this structure is good operating speed but on account of more cost.

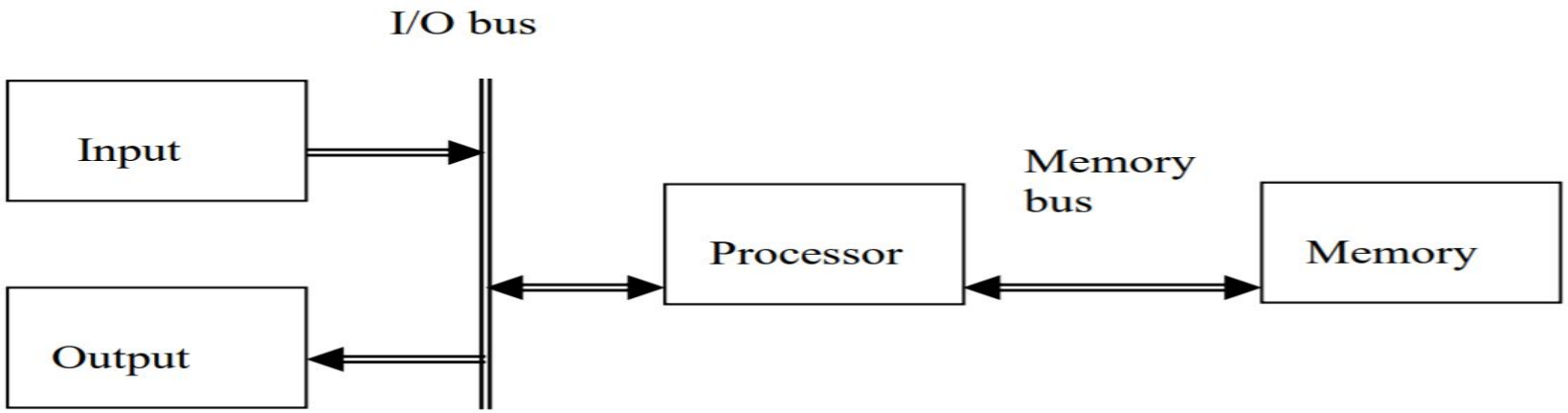
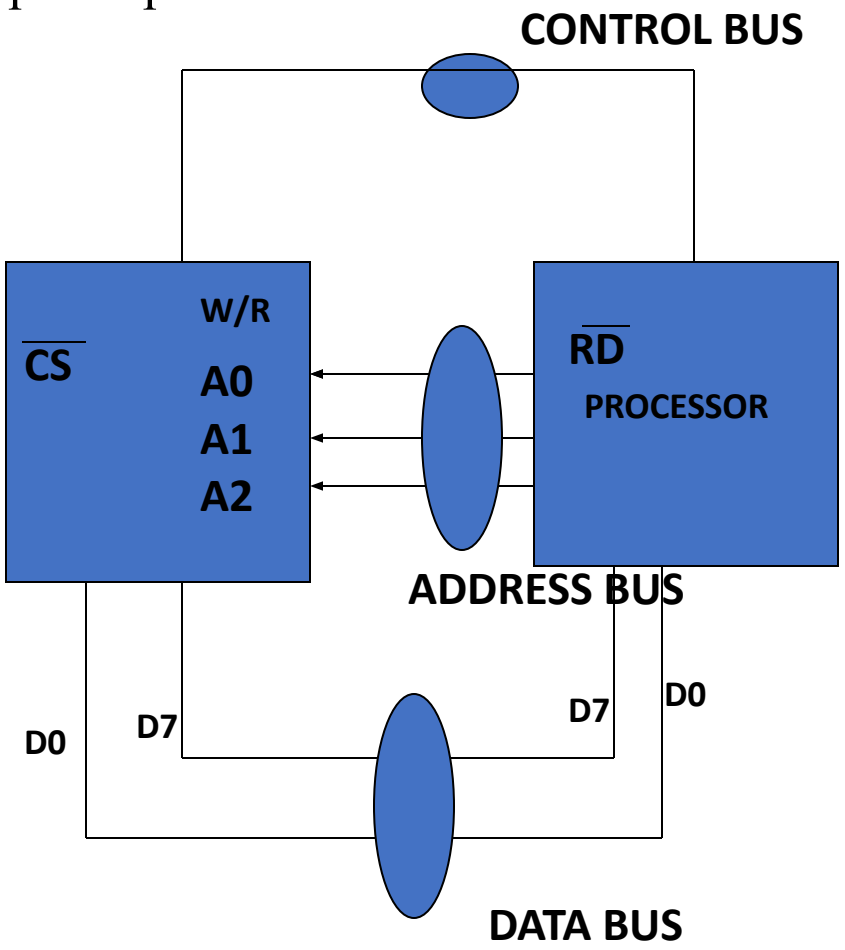


Figure 2.2 Two-bus Structure

MULTI BUS STRUCTURE

To improve performance multi bus structure can be used.



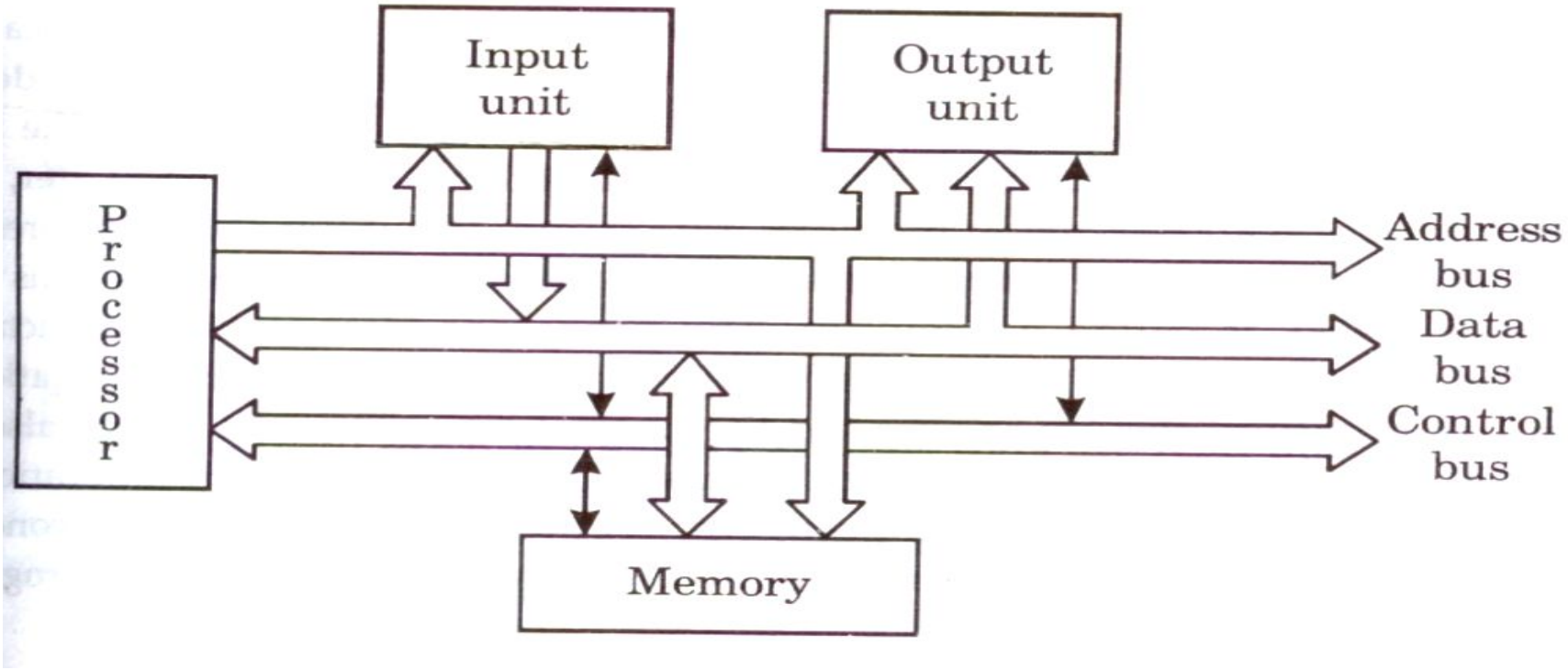
A_2	A_1	A_0	Selected location
0	0	0	0^{th} Location
0	0	1	1^{st} Location
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

- $2^3 = 8$ i.e. 3 address line is required to select 8 location

- In general $2^x = n$ where x number of address lines (address bit) and n is number of location

- **Address bus** : unidirectional : group of wires which carries address information bits form processor to peripherals (16,20,24 or more parallel signal lines)
- **Data bus**: bidirectional : group of wires which carries data information bit form processor to peripherals and vice – versa
- **Control bus**: bidirectional: group of wires which carries control signals form processor to peripherals and vice – versa

Figure below shows address, data and control bus and their connection with peripheral and microprocessor

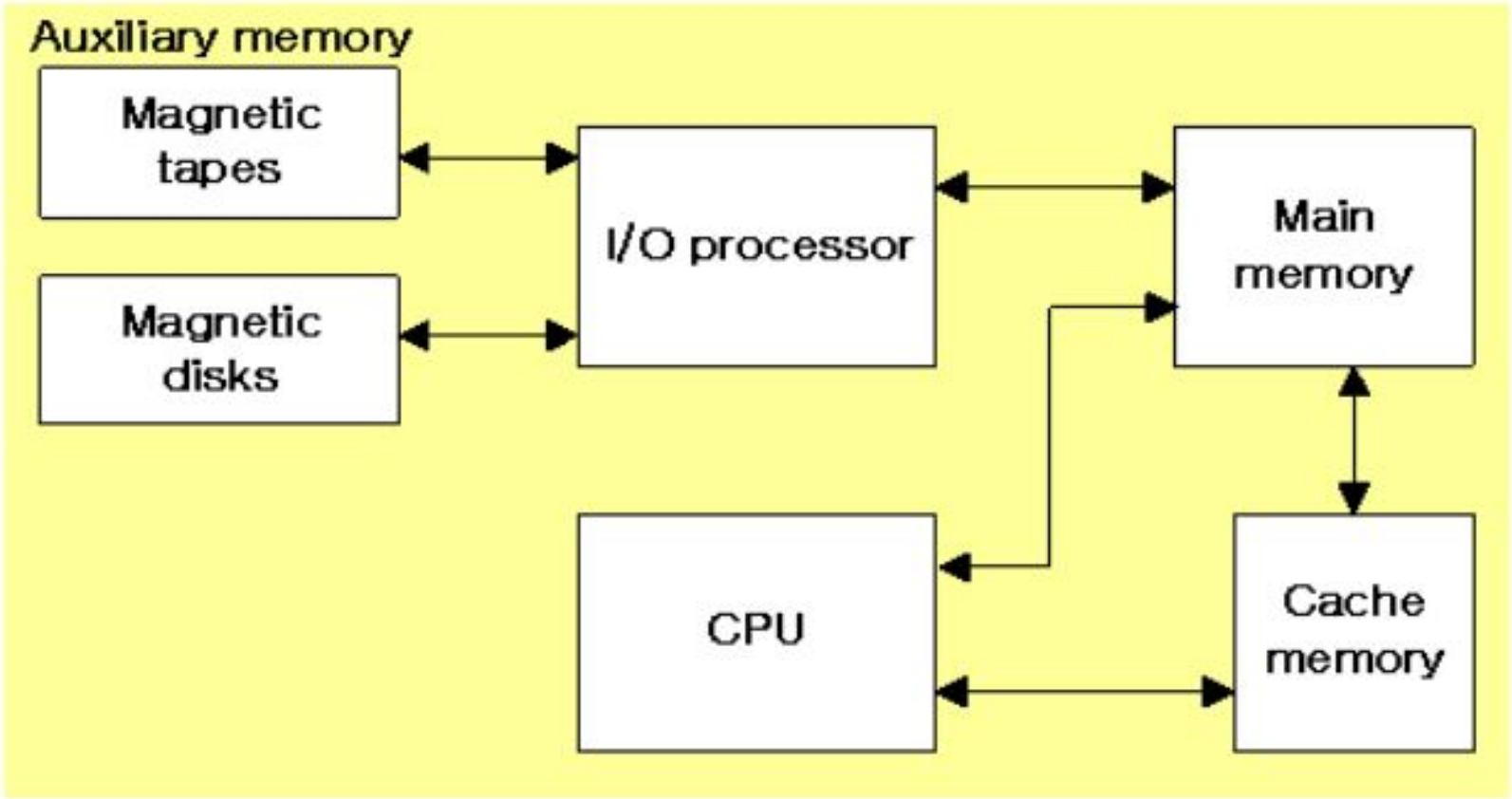


Single bus structure showing the details of connection

Memory Locations and Addresses

MEMORY HIERARCHY

Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system



Memory Locations and Addresses

Memory consists of many millions of storage cells, each of which can store 1 bit.

Data is usually accessed in *n*-bit groups. *n* is called word length.

The memory of a computer can be schematically represented as a collection of words as shown in Figure 1.

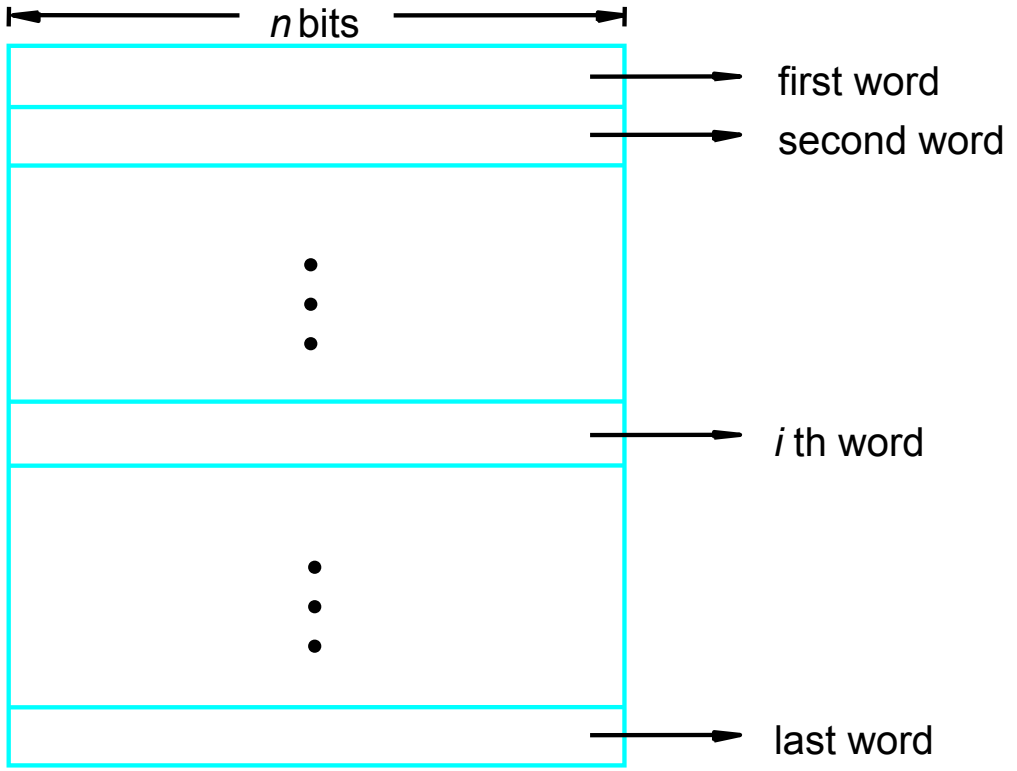


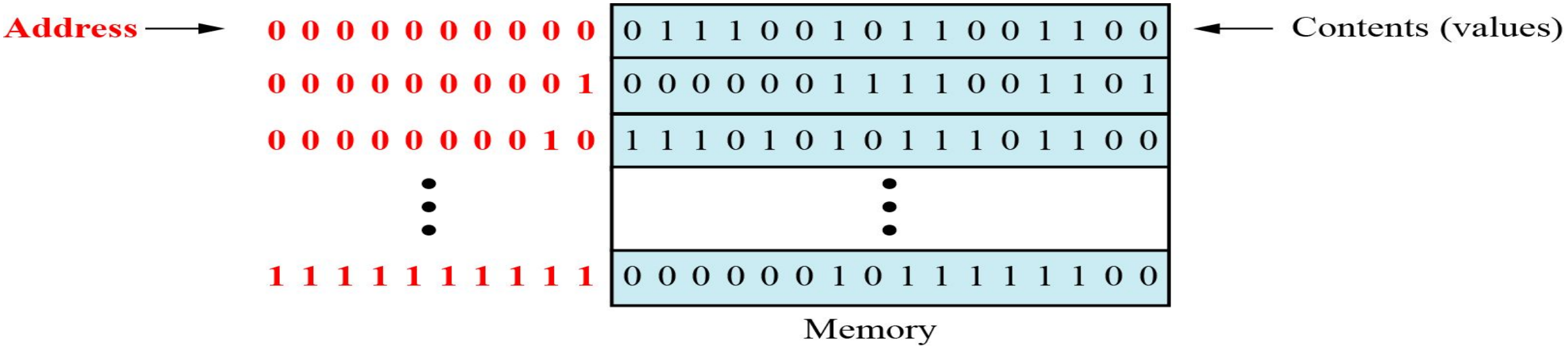
Figure 1 Main Memory words.

MEMORY LOCATIONS AND ADDRESSES

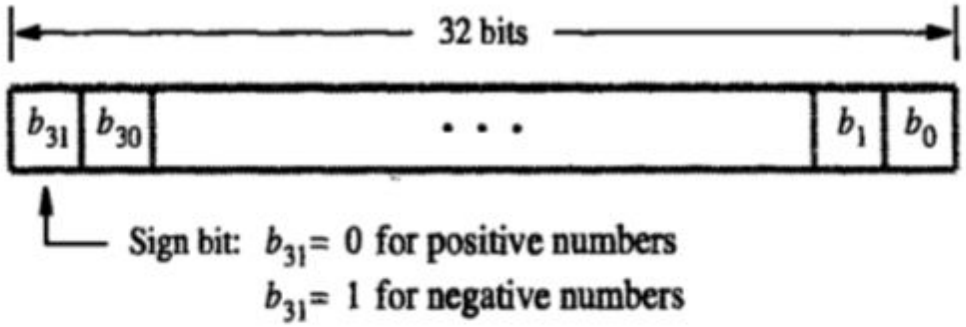
- **Main memory** is the second major subsystem in a computer. It consists of a collection of storage locations, each with a unique identifier, called an **address**.

- Data is transferred to and from memory in groups of bits called **words**. A word can be a group of 8 bits, 16 bits, 32 bits or 64 bits (and growing).

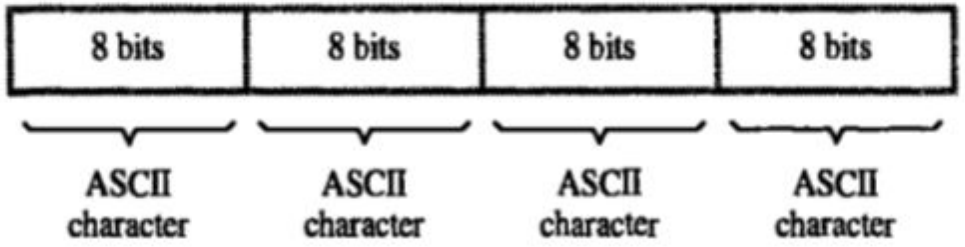
- If the word is 8 bits, it is referred to as a **byte**. The term "byte" is so common in computer science that sometimes a 16-bit word is referred to as a 2-byte word, or a 32-bit word is referred to as a 4-byte word.


Address
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0
1 1 1 1 1 1 1 1 1 1
0 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0
0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 1
1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 0
0 0 0 0 0 0 1 0 1 1 1 1 1 1 0 0
Contents (values)
Memory

Main memory



(a) A signed integer



(b) Four characters

Figure 2.6 Examples of encoded information in a 32-bit word.

Address space

- To access a word in memory requires an identifier. Although programmers use a name to identify a word (or a collection of words), at the hardware level each word is identified by an address.

- The total number of uniquely identifiable locations in memory is called the **address space**.

- For example, a memory with 64 kilobytes (16 address line required) and a word size of 1 byte has an address space that ranges from 0 to 65,535.

Table 5.1 Memory units

Unit	*Exact Number of Bytes*	*Approximation*
kilobyte	2^{10} (1024) bytes	10^3 bytes
megabyte	2^{20} (1,048,576) bytes	10^6 bytes
gigabyte	2^{30} (1,073,741,824) bytes	10^9 bytes
terabyte	2^{40} bytes	10^{12} bytes

Memory addresses are defined using unsigned binary integers.

Example 1

A computer has 32 MB (megabytes) of memory. How many bits are needed to address any single byte in memory?

Solution

The memory address space is 32 MB, or 2^{25} ($2^5 \times 2^{20}$). This means that we need $\log_2 2^{25}$, or **25 bits**, to address each byte.

Example 2

A computer has 128 MB of memory. Each word in this computer is eight bytes. How many bits are needed to address any single word in memory?

Solution

The memory address space is 128 MB, which means 2^{27}. However, each word is eight (2^3) bytes, which means that we have 2^{24} words. This means that we need $\log_2 2^{24}$, or **24 bits**, to address each word.

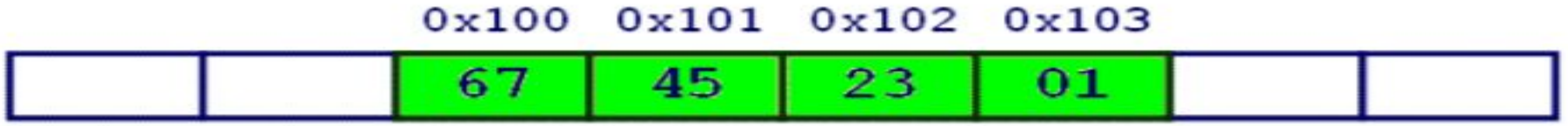
MEMORY OPERATIONS

- Today, **general-purpose computers** use a set of instructions called a **program** to process data.
- A computer executes the program to create output data from input data
- Both program instructions and data operands are stored in memory
- Two basic operations requires in memory access
  - **Load operation (Read or Fetch)**-Contents of specified memory location are read by processor
  - **Store operation (Write)-** Data from the processor is stored in specified memory location

Assignment of Byte Address

- **Big-endian** and **little-endian** are terms that describe the order in which a sequence of bytes are stored in computer **memory**.
- **Big-endian** is an order in which the "**big**end" (most significant value in the sequence) is stored first (at the lowest storage address).
- **Little-endian** is an order in which the "**Little** end" (least significant value in the sequence) is stored first (at the lowest storage address).

Big Endian

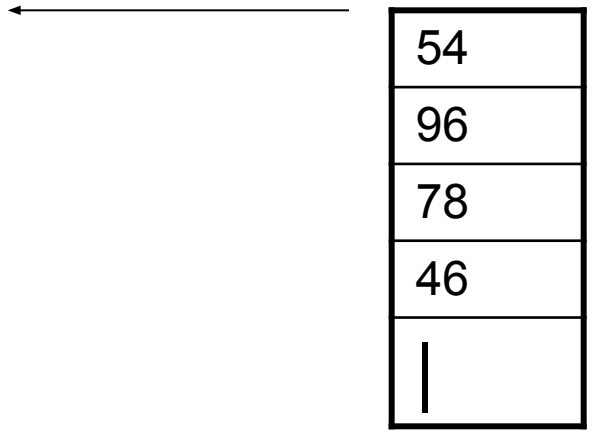


Little Endian

Assignment of byte addresses

- Little Endian (e.g., in DEC, Intel)
  - » low order byte stored at lowest address
  - » byte0 byte1 byte2 byte3

- Eg: 46,78,96,54 (32 bit data)
- H BYTE L BYTE



- 8000
- 8001
- 8002
- 8003
- 8004

Big Endian

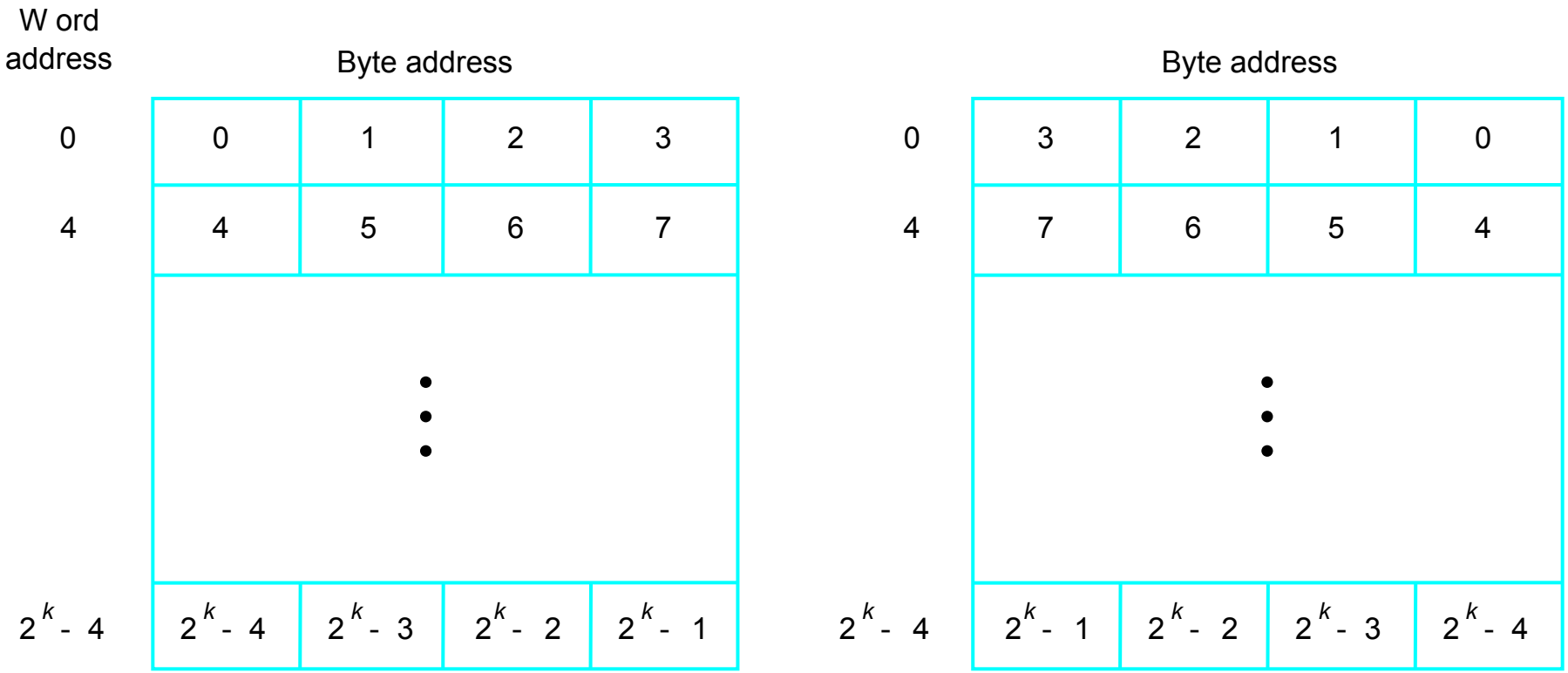
- Big Endian (e.g., in IBM, Motorolla, Sun, HP)
  - » high order byte stored at lowest address
  - » byte3 byte2 byte1 byte0

- Programmers/protocols should be careful when transferring binary data between Big Endian and Little Endian machines

Big-Endian and Little-Endian Assignments

Big-Endian: lower byte addresses are used for the most significant bytes of the word

Little-Endian: opposite ordering. lower byte addresses are used for the less significant bytes of the word



(a) Big-endian assignment (b) Little-endian assignment

Byte and word addressing.

- In case of 16 bit data, aligned words begin at byte addresses of 0,2,4,……
- In case of 32 bit data, aligned words begin at byte address of 0,4,8,……
- In case of 64 bit data, aligned words begin at byte addresses of 0,8,16,……..
- In some cases words can start at an arbitrary byte address also then, we say that word locations are unaligned

Instruction and instruction sequencing

- **Instruction**: is command to the microprocessor to perform a given task on specified data.
- **Instruction Set**: The entire group of these instructions are called instruction set.
- **instruction sequencing :** The order in which the instructions in a program are carried out.

4 TYPES OF OPERATION:-

A computer must have instructions capable of performing 4 types of operation

- Data transfer between memory and processor register
- Arithmetic and logic operation
- Program sequencing and control
- I/O transfer

Register transfer notation (RTN)

Transfer between processor registers & memory, between processor register & I/O devices

Memory locations, registers and I/O register names are identified by a symbolic name in uppercase alphabets

- LOC,PLACE,MEM are the address of memory location
- R1 , R2,… are processor registers
- DATA_IN, DATA_OUT are I/O registers
- Contents of location is indicated by using square brackets [ ]
- RHS of RTN always denotes a values, and is called Source
- LHS of RTN always denotes a symbolic name where value is to be stored and is called destination
- Source contents are not modified
- Destination contents are overwritten

Examples of RTN statements

- R2 ← [LOCN]
- R4 ← [R3] +[R2]

Data transfer Instructions

They are also called copy instructions.

Some instructions in 8086:

MOV -Copy from the source to the destination

LDA - Load the accumulator

STA - Store the accumulator

PUSH - Push the register pair onto the stack

POP - Pop off stack to the register pair

Data Manipulation Instructions

- To perform the operations by the ALU
- Three categories:
  - Arithmetic Instructions
  - Logical and bit manipulation instructions
  - Shift instructions

Arithmetic Instructions

Used to perform arithmetic operations

Some instruction in 8086

INC □ Increment the data by 1

DEC □ Decreases data by 1

ADD □ perform sum of data

ADC □ Add with carry bit.

MUL □ perform multiplication

Logical and bit manipulation instructions

Used to perform logical operations

Some instructions are:

AND bitwise AND operation

OR bitwise OR operation

NOT invert each bit of a byte or word

XOR Exclusive-OR operation over each bit

Shift instructions

used for shifting or rotating the contents of the register

Some instructions are:

SHR shift bits towards the right and put zero(S) in MSBs

ROL rotate bits towards the left, i.e. MSB to LSB and to Carry Flag [CF]

RCL rotate bits towards the left, i.e. MSB to CF and CF to LSB.

Instruction Formats
(Types of instruction based on the address field)

- A instruction in computer comprises of groups called fields.
- These field contains different information
- The most common fields are:

Operation field : specifies the operation to be performed like addition.

Address field : contain the location of operand

Mode field : specifies how to find the operand

- A instruction is of various length depending upon the number of addresses it contain.
- On the basis of number of address, instruction are classified as:
  - **Zero Address Instructions**
  - **One Address Instructions**
  - **Two Address Instructions**
  - **Three Address Instructions**

Zero Address Instructions

Used in stack based computers which do not use address field in instruction

- Location of all operands are defined implicitly
- Operands are stored in a structure called pushdown stack

Operation

- If processor supports ALU operations one data in memory and other in register then the instruction sequence is
  - MOVE D, Ri
  - ADD E, Ri
  - MOVE Ri, F
- If processor supports ALU operations only with registers then one has to follow the instruction sequence given below
  - LOAD D, Ri
  - LOAD E, Rj
  - ADD Ri, Rj
  - MOVE Rj, F

Example: Evaluate (A+B) * (C+D)

- Using Zero-Address instruction

1. PUSH A ; TOS ← A
2. PUSH B ; TOS ← B
3. ADD ; TOS ← (A + B)
4. PUSH C ; TOS ← C
5. PUSH D ; TOS ← D
6. ADD ; TOS ← (C + D)
7. MUL ; TOS ← (C+D)*(A+B)
8. POP X ; M[X] ← TOS

One address Instruction

- Syntax- Operation source/destination
- In this type either a source or destination operand is mentioned in the instruction
- Other operand is implied to be a processor register called Accumulator
- Eg: ADD B (general) ←
- Load D; ACC ← [memlocation _D]
- ADD E; ACC (ACC) ← (ACC) + (E)
- STORE F; memlocation_ F (ACC)

One address Instruction

- This use a implied ACCUMULATOR register for data manipulation.

Operation Destination

- One operand is in accumulator and other is in register or memory location.

Example: Evaluate (A+B) * (C+D)

- Using One-Address Instruction

```
1.  LOAD  A   ; AC ← M[A]
2.  ADD   B   ; AC ← AC + M[B]
3.  STORE T   ; M[T] ← AC
4.  LOAD  C   ; AC ← M[C]
5.  ADD   D   ; AC ← AC + M[D]
6.  MUL   T   ; AC ← AC * M[T]
7.  STORE X   ; M[X] ← AC
```

Two Address Instruction

- Syntax : Operation source, destination

- Eg: MOVE E,F MOVE D,F

  ADD D,F OR ADD E,F

Perform ADD A,B,C using 2 instructions

```
MOVE  B,C
ADD   A,C
```

❖Disadvantage: Single instruction is not sufficient to perform the entire operation.

Two Address Instruction

This is common in commercial computers.

Operation Source, Destination

Here two address can be specified in the instruction.

Example: Evaluate (A+B) * (C+D)

Using Two address Instruction:

1. MOV R1,A ; R1=M[A]
2. ADD R1,B ;R1=R1+M[B]
3. MOV R2,C ;R2=M[C]
4. ADD R2,D ;R2=R2+M[D]
5. MUL R1,R2 ; R1=R1*R2
6. MOV X,R1 ; M[X]=R1

Three Address Instruction

- Syntax: Operation source 1, source 2, destination
- Eg: ADD D,E,F

where D,E,F are memory location

- Advantage: Single instruction can perform the complete operation
- Disadvantage : Instruction code will be too large to fit in one word location in memory

Three Address Instruction

This has three address field to specify a register or a memory location.

Program created are much short in size

creation of program much easier

does not mean that program will run much faster

Example: Evaluate (A+B) * (C+D)

Using Three address Instruction

1. ADD R1,A,B ;R1=M[A]+M[B]
2. ADD R2,C,D ;R2=M[C]+M[D]
3. MUL X,R1,R2 ;M[X]=R1*R2

Instruction Cycle

- the basic operational process of a computer.
- also known as **fetch-decode-execute cycle**
- This process is repeated continuously by CPU from boot up to shut down of computer.

steps that occur during an instruction cycle:

1. **Fetch the Instruction**
2. **Decode the Instruction**
3. **Read the Effective Address**
4. **Execute the Instruction**

1. **Fetch the Instruction**

The instruction is fetched from memory address that is stored in PC(Program Counter) and stored in the (instruction register) IR.

At the end of the fetch operation, PC is incremented by 1 and it then points to the next instruction to be executed.

2. **Decode the Instruction**

The instruction in the IR is decoded(Interpreted).

3. Read the Effective Address

If the instruction has an indirect address, the effective address is read from the memory.

Otherwise operands are directly read in case of immediate operand instruction.

4. Execute the Instruction

The Control Unit passes the information in the form of control signals to the functional unit of CPU.

The result generated is stored in main memory or sent to an output device.

The cycle is then repeated by fetching the next instruction.

Thus in this way the instruction cycle is repeated continuously.

Instruction execution and straight line sequencing

- Instruction execution needs the following steps, which are
- PC (program counter) register of the processor gives the address of the instruction which needs to be fetched from the memory.
- If the instruction is fetched then, the instruction opcode is decoded. On decoding, the processor identifies the number of operands. If there is any operand to be fetched from the memory, then that operand address is calculated.
- Operands are fetched from the memory. If there is more than one operand, then the operand fetching process may be repeated (i.e. address calculation and fetching operands).
- After this, the data operation is performed on the operands, and a result is generated.
- If the result has to be stored in a register, the instructions end here.
- If the destination is memory, then first the destination address has to be calculated. Then the result is then stored in the memory. If there are multiple results which need to be stored inside the memory, then this process may repeat (i.e. destination address calculation and store result).
- Now the current instructions have been executed. Side by side, the PC is incremented to calculate the address of the next instruction.
- The above instruction cycle then repeats for further instructions.

Straight line sequencing

- Straight line sequencing means the instruction of a program is executed in a sequential manner(i.e. every time PC is incremented by a fixed offset).
- And no branch address is loaded on the PC.

Example 1:

•programs and data are stored in the same memory, i.e. von Neumann architecture.

•First instruction of a program is stored at address i.

•PC gives address i and instruction stored at that address i is fetched from the memory and then decoded and then operand

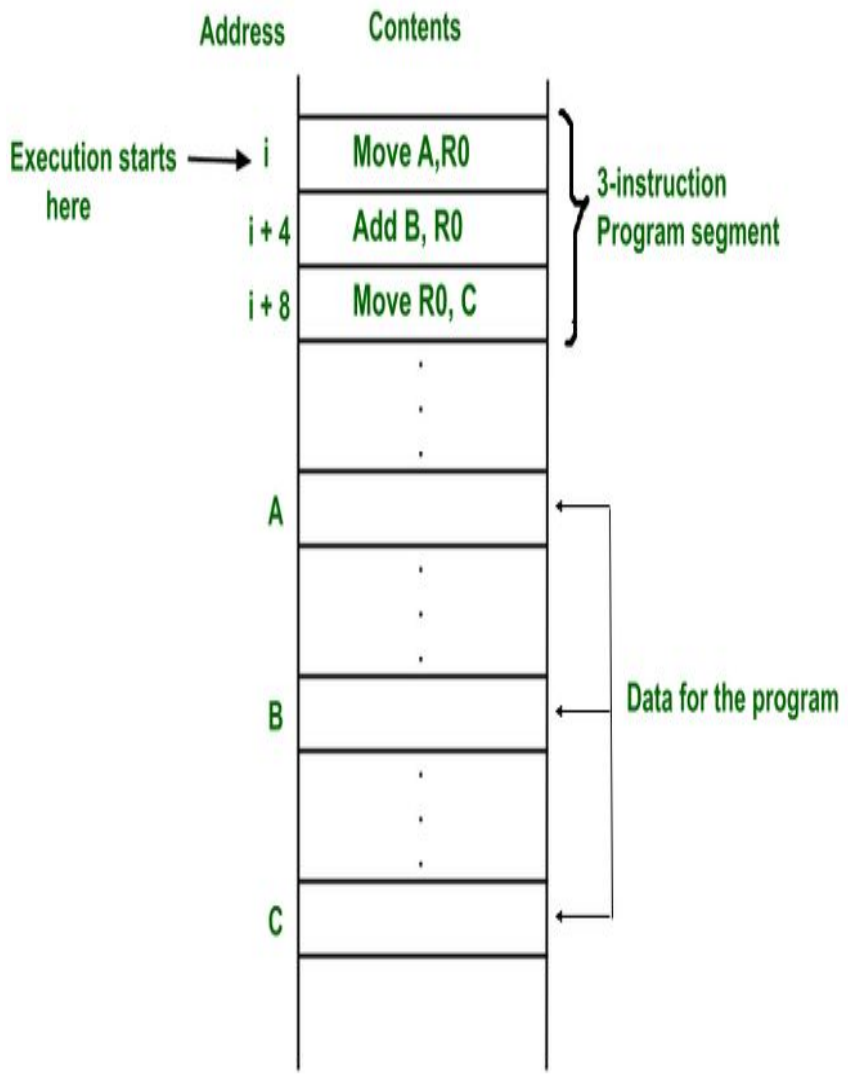
•A is fetched from the memory and stored in a temporary register and then the instruction is executed(i.e. content of address A is copied into processor register R0).

•the PC gets incremented by 4(i.e. it contains the address of the next instruction) because the instruction and memory segment is of 4 bytes.

•So the instruction at address i is executed.

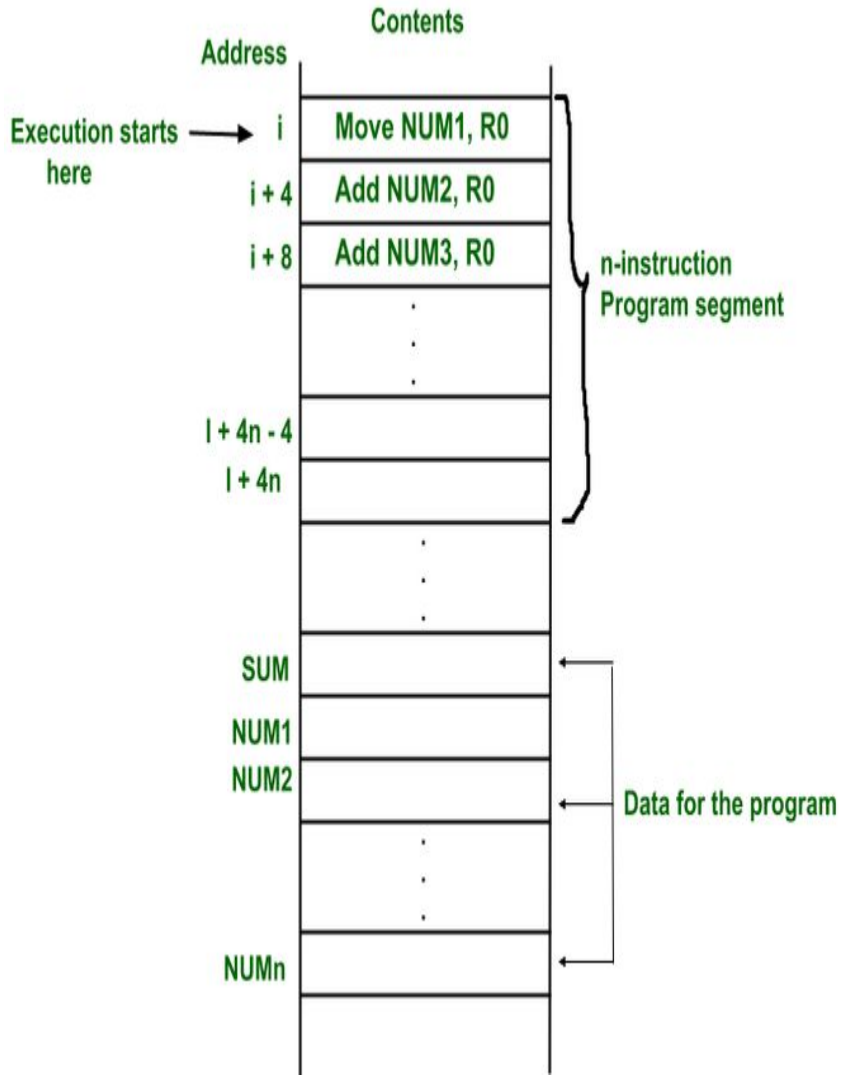
•So every time, the PC is incremented by 4.

Therefore, the program is executing in a sequential manner. And this process is called straight line sequencing.



•The addresses of the memory locations containing the n numbers are represented as NUM1,NUM2…..NUMn(i.e. NUM1 address includes first number).

•The first number is stored into processor register R0. And every other number is added to register R0. Finally, when the program ends(i.e. n numbers are added, the result is placed in memory location SUM



•The second way is to use a loop to add n number. But here straight line sequencing is not used because every time loop iteration ends, PC has to load the branch address and program execution starts from that address.

•Here the location N stores the value of n.

•Processor register R1 is used as a counter to determine the number of times the loop gets executed.

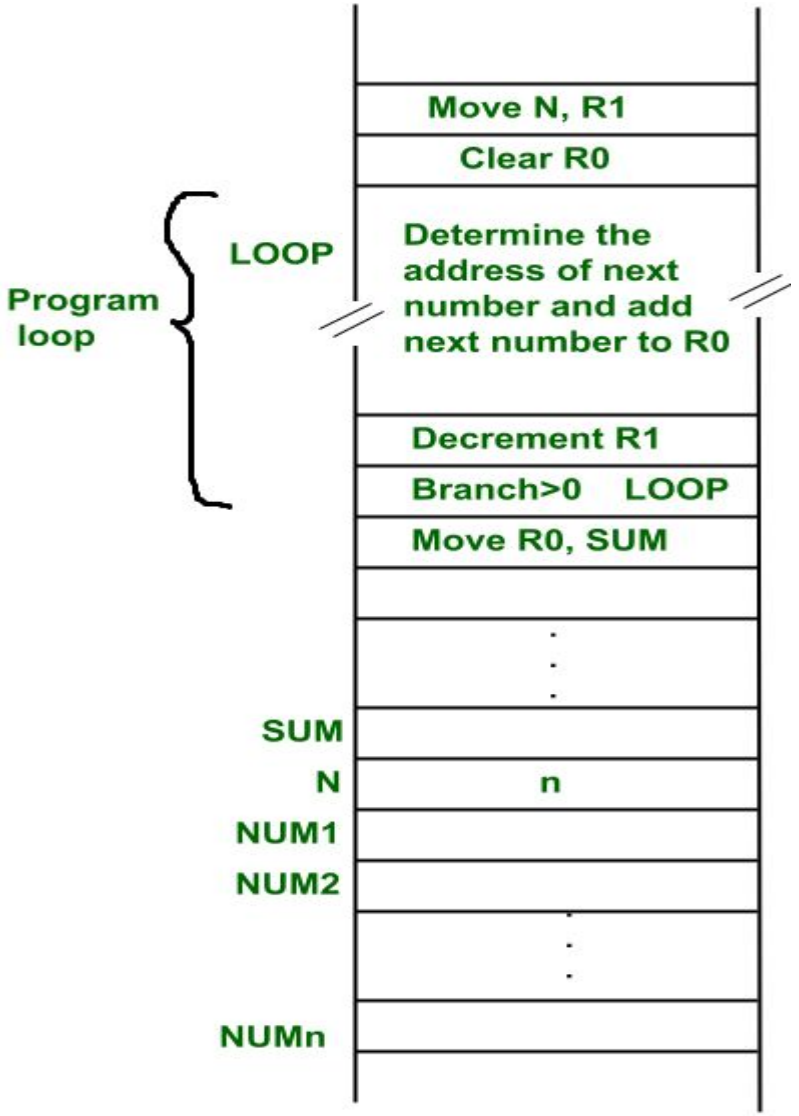
•The contents of the location N are moved into R1 at the start of program execution.

•After that, register R0 is cleared.

•The address LOOP is reloaded again and again until R1 becomes 0.

•Every time a number is added, then the R1 value is decremented.

•When R1 becomes 0, we come out of the loop and the result which is stored at R1 is copied into memory location SUM.



Condition Codes

The processor keeps track of information about the results of various operations for use by subsequent conditional branch instructions

N – Negative	1 if results are Negative
	0 if results are Positive
Z – Zero	1 if results are Zero
	0 if results are Non zero
V – Overflow	1 if arithmetic overflow occurs
	0 non overflow occurs
C – Carry	1 if carry and from MSB bit
	0 if there is no carry from MSB bit

Addressing Modes

Addressing Modes

Different ways in which the location of the operand is specified in an instruction is referred as addressing modes. The purpose of using addressing mode is:

To give the programming versatility to the user.

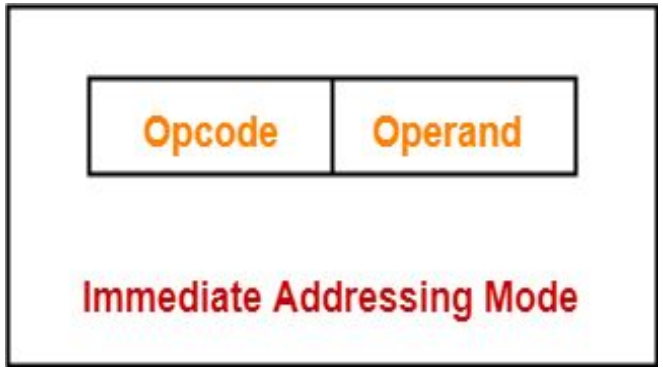
To reduce the number of bits in addressing field of instruction.

Types of Addressing Modes

- Immediate Addressing
- Direct Addressing
- Indirect Addressing
- Register Addressing
- Register Indirect Addressing
- Relative Addressing
- Indexed Addressing
- Auto Increment
- Auto Decrement

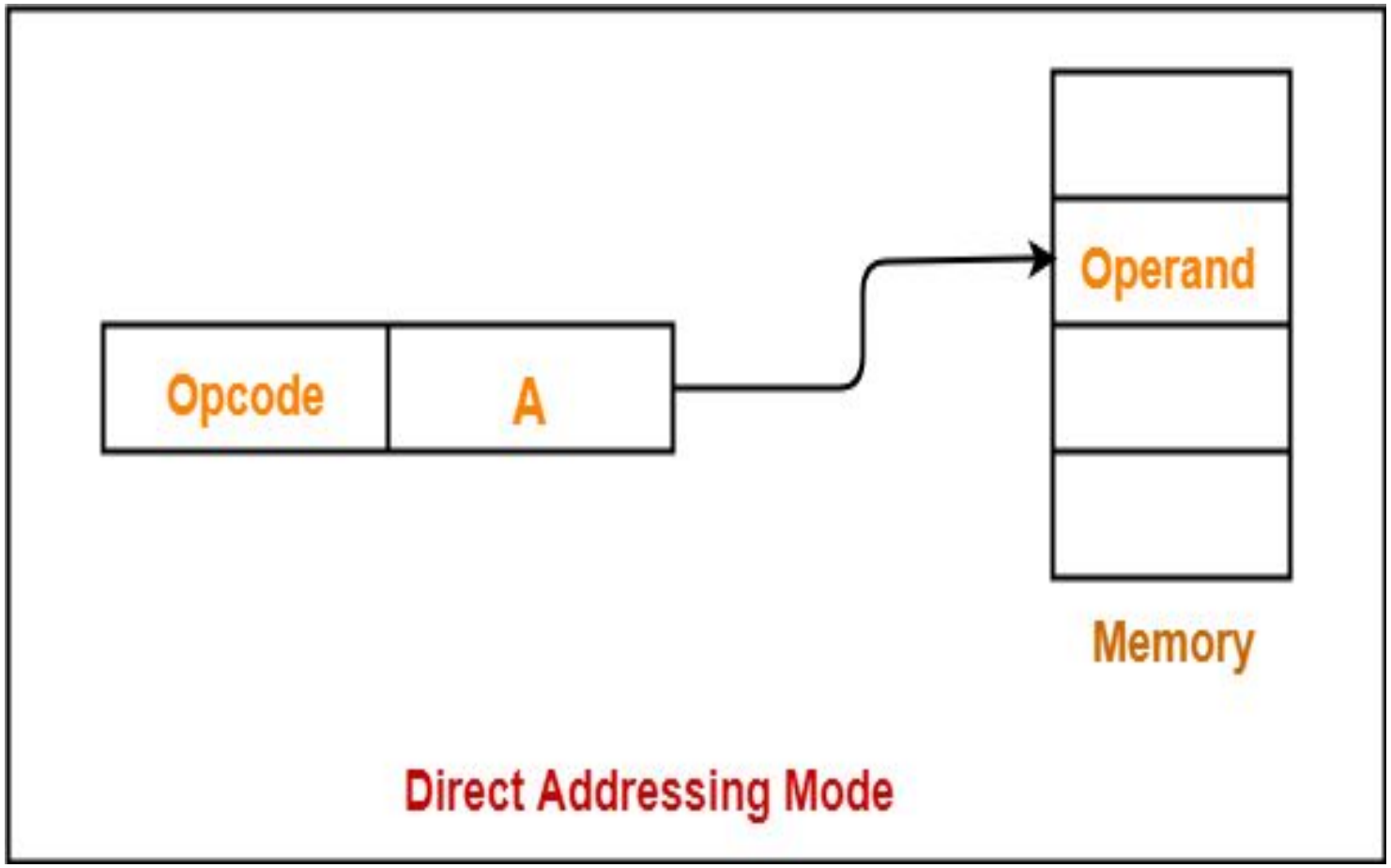
Immediate Addressing

- Operand is given explicitly in the instruction
- e.g. ADD 5
  - Add 5 to contents of accumulator
  - 5 is operand
- No memory reference to fetch data
- Fast
- Limited range
- MOV AL,25H ; Immediate addressing AL=25
- MOV AX,2345H ; AX=2345 AX=> AH=23 AL=45



Direct Addressing

- Address field contains address of operand
- Effective address (EA) = address field (A)
- e.g. ADD A
  - Add contents of cell A to accumulator
  - Look in memory at address A for operand
- Single memory reference to access data
- No additional calculations to work out effective address
- Limited address space
- MOV AL,DATA1 ; Direct Addressing AL=23
- MOV AX,DATA2 ; AX=1234
- MOV DATA3,AL ; DATA3=23
- MOV DATA4,AX ; DATA4=1234

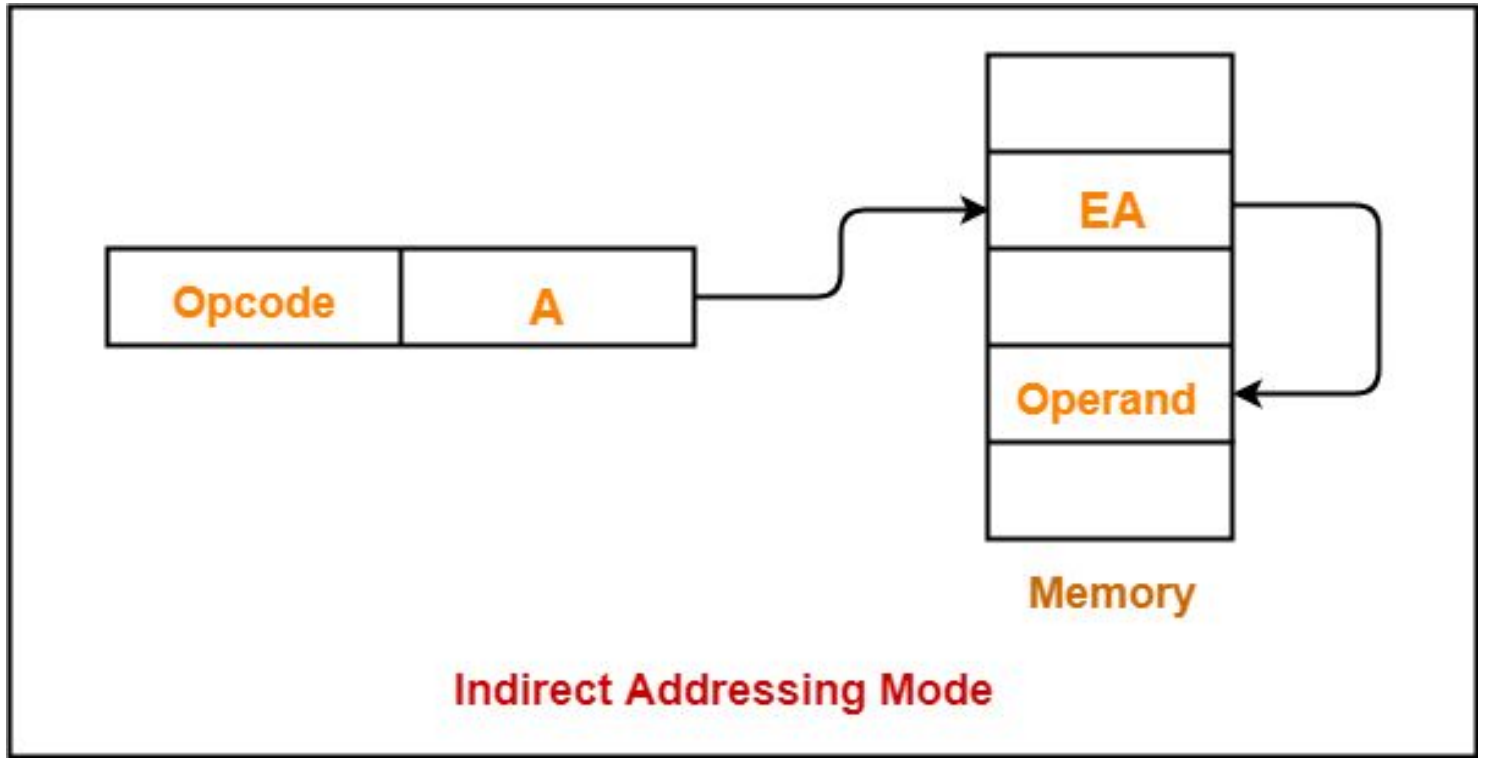

Opcode
A
Operand
Memory
Direct Addressing Mode

Indirect Addressing

- Memory cell pointed to by address field contains the address of (pointer to) the operand

  Two references to memory are required to fetch the operand.

- Effective Address = [A]
  - Look in A, find address (A) and look there for operand
- e.g. ADD (A)
  - Add contents of cell pointed to by contents of A to the accumulator


Opcode
A
EA
Operand
Memory
Indirect Addressing Mode

Register Direct Addressing

In this addressing mode,

- The operand is contained in a register set.
- The address field of the instruction refers to a CPU register that contains the operand.
- No memory access
- Very fast execution
- Very limited address space
- Limited number of registers
- Very small address field needed
    - Shorter instructions
    - Faster instruction fetch

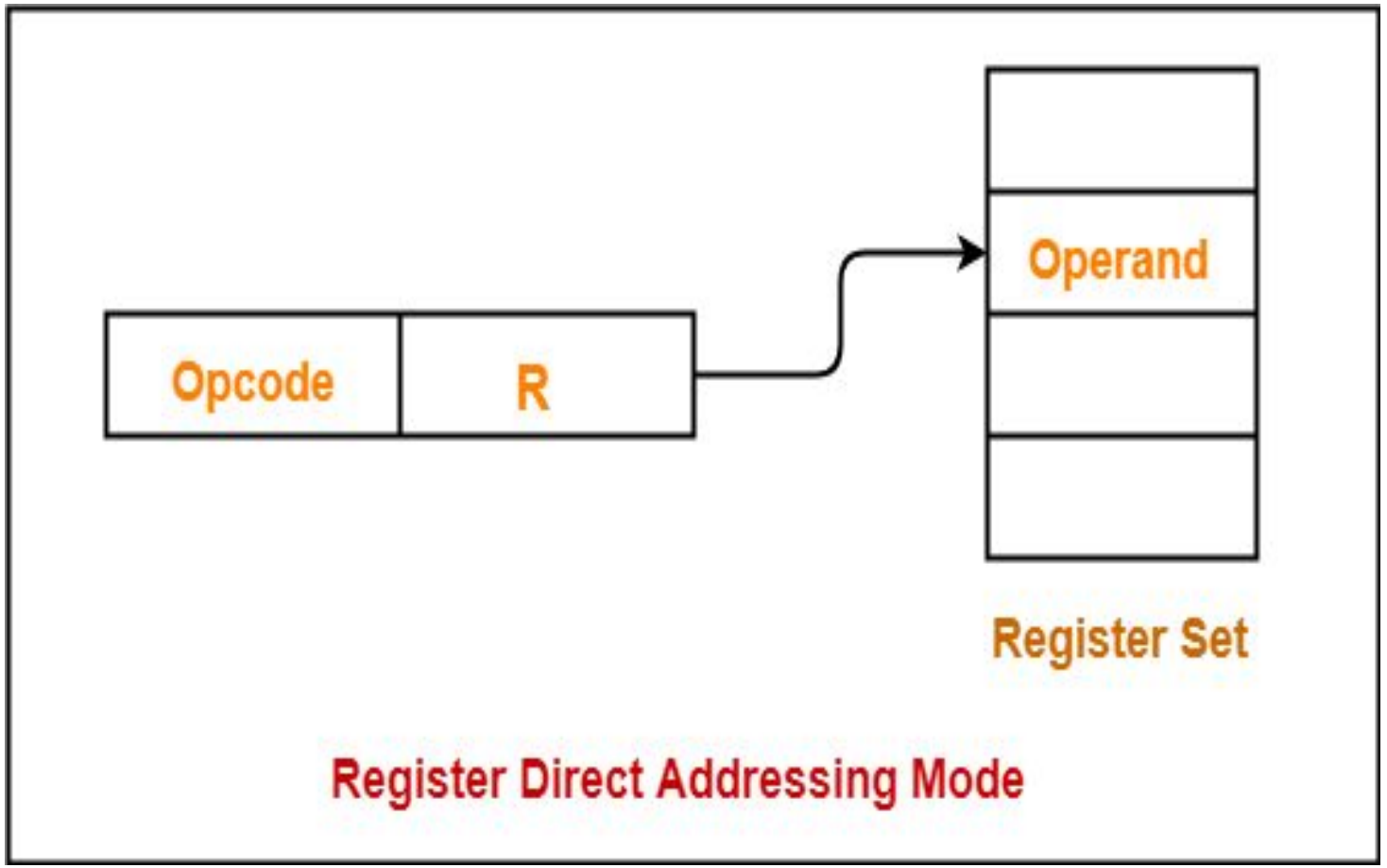
Register Direct Addressing

Eg:

ADD R will increment the value stored in the accumulator by the content of register R.

$$AC \leftarrow AC + [R]$$

- This addressing mode is similar to direct addressing mode.
- The only difference is address field of the instruction refers to a CPU register instead of main memory.


Opcode
R
Operand
Register Set
Register Direct Addressing Mode

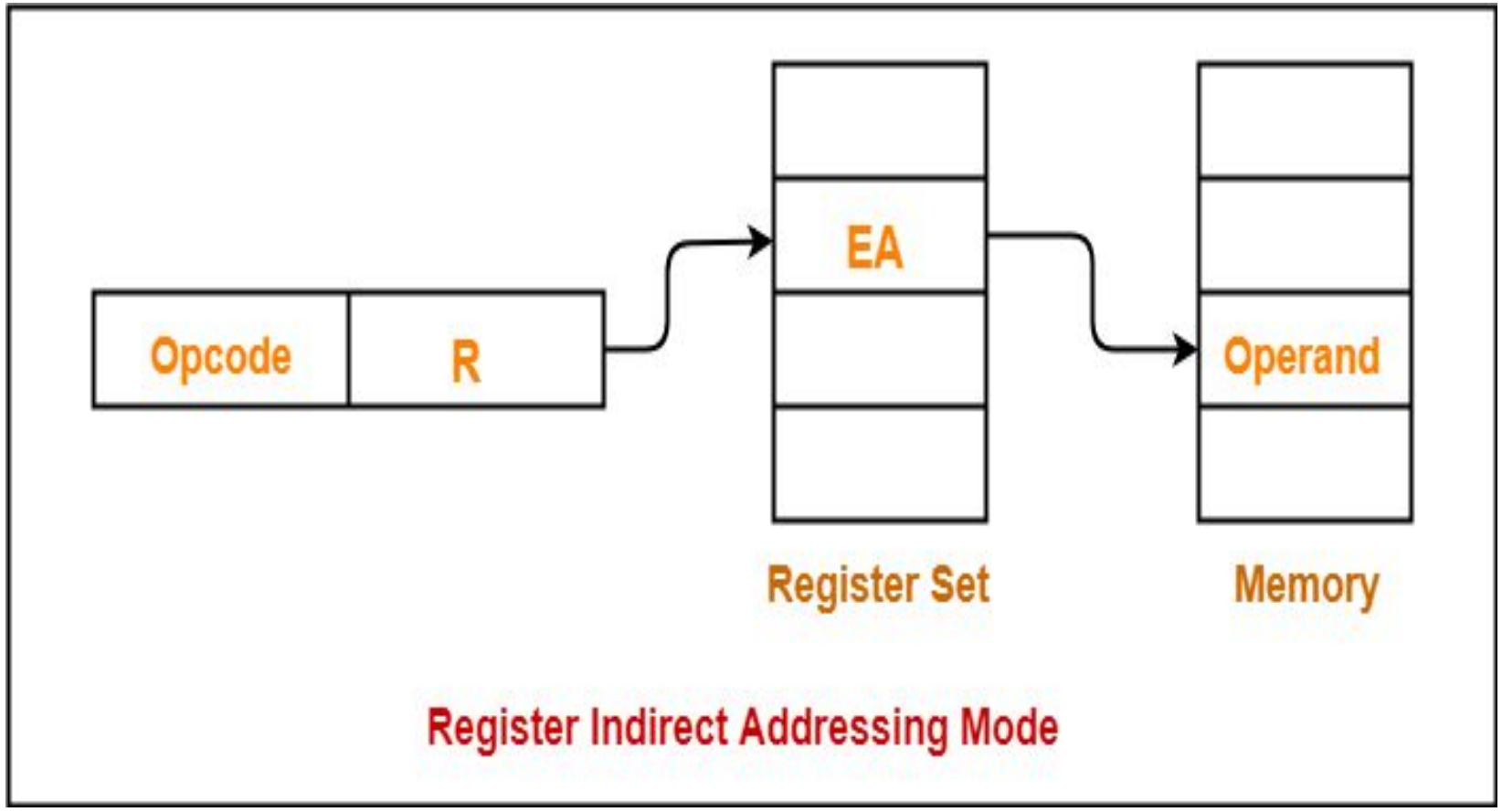
Register Indirect Addressing

- The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand

Eg:

ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.

$$AC \leftarrow AC + [[R]]$$

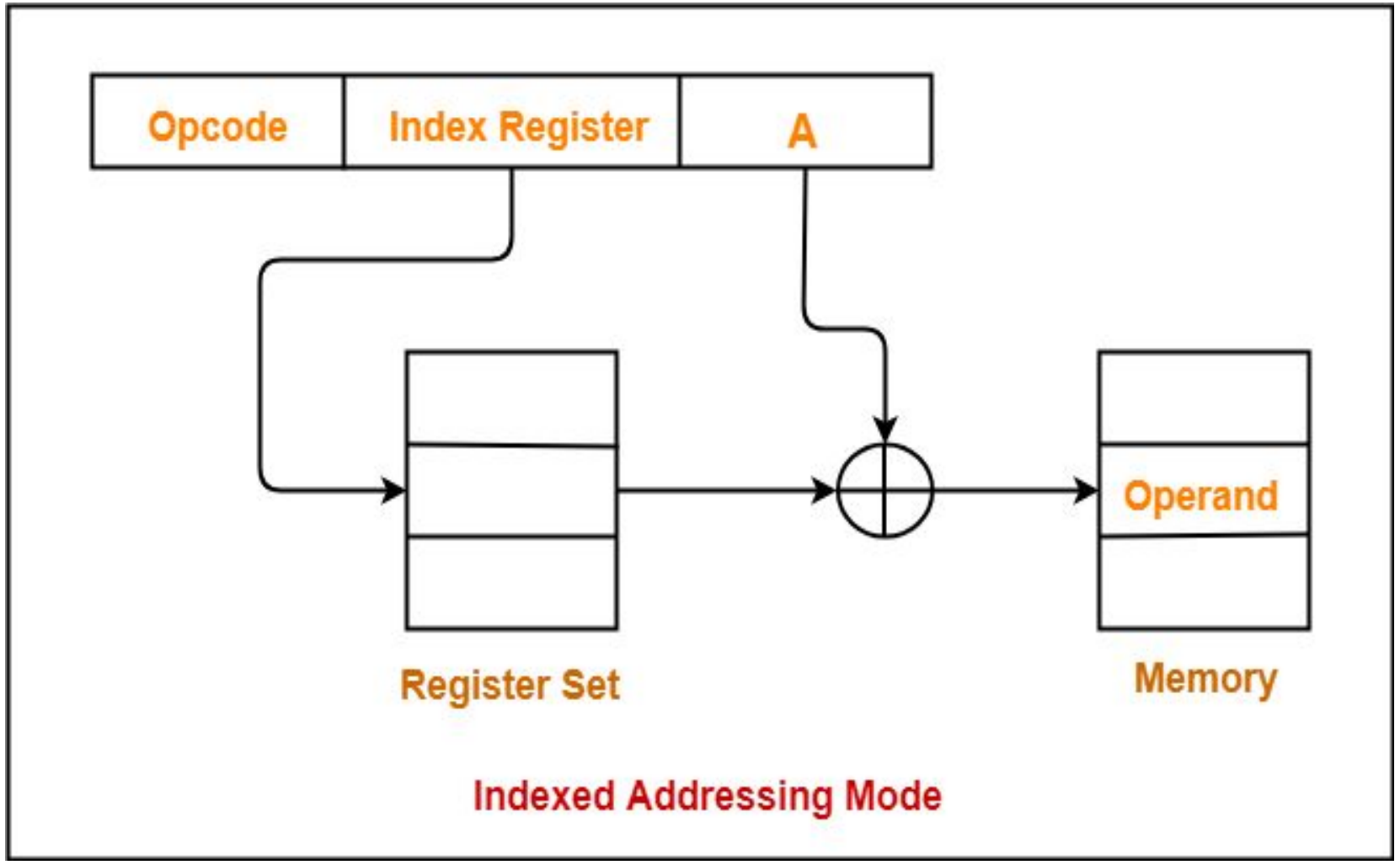

Opcode
R
EA
Operand
Register Set
Memory
Register Indirect Addressing Mode

Indexed Addressing

In this addressing mode,

- Effective address of the operand is obtained by adding the content of index register with the address part of the instruction.

Effective Address = Content of Index Register + Address part of the instruction


Opcode
Index Register
A
Operand
Register Set
Memory
Indexed Addressing Mode

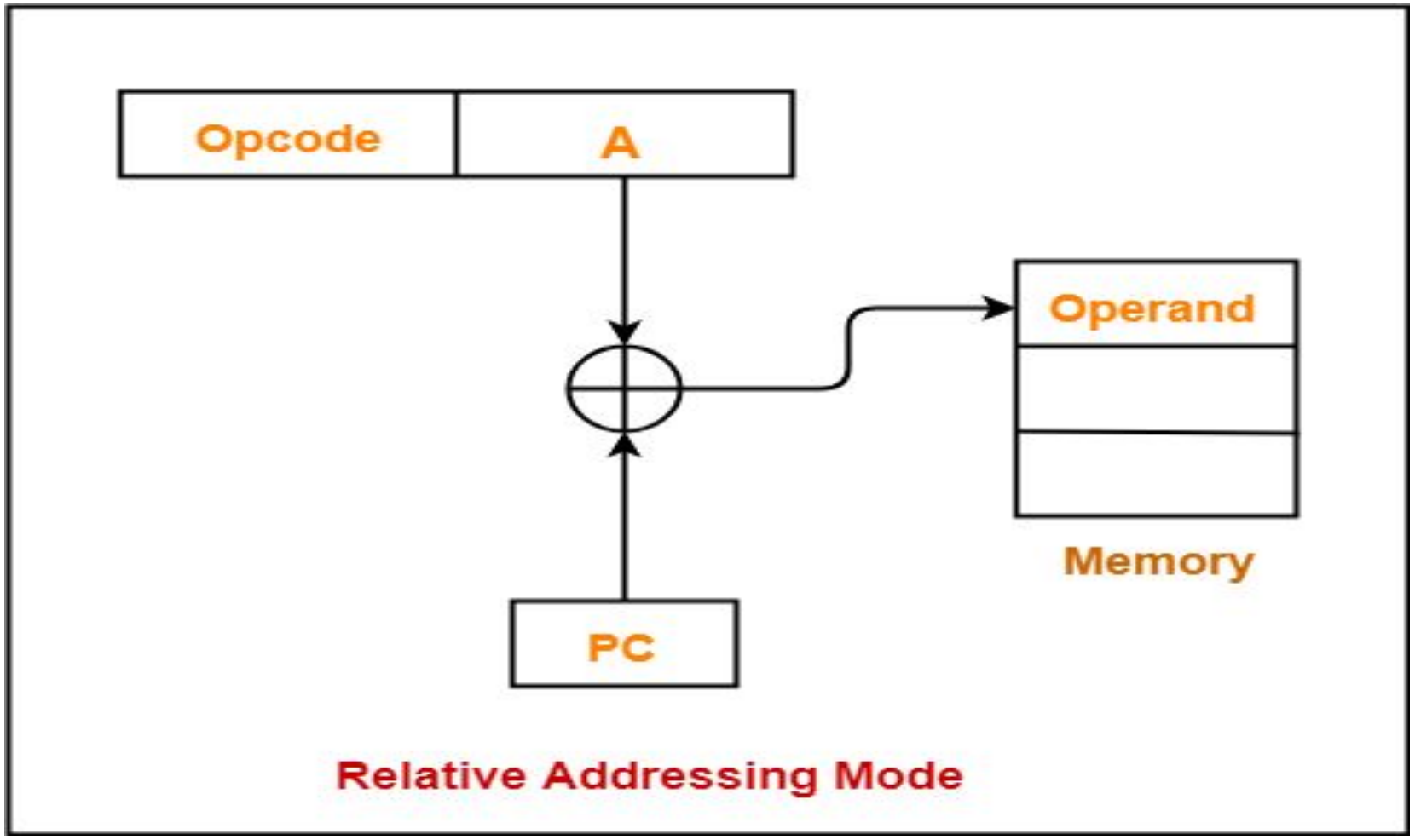
Relative Addressing

A version of displacement addressing

In this addressing mode,

- Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.

Effective Address = Content of Program Counter + Address part of the instruction


Opcode
A
Operand
Memory
PC
Relative Addressing Mode

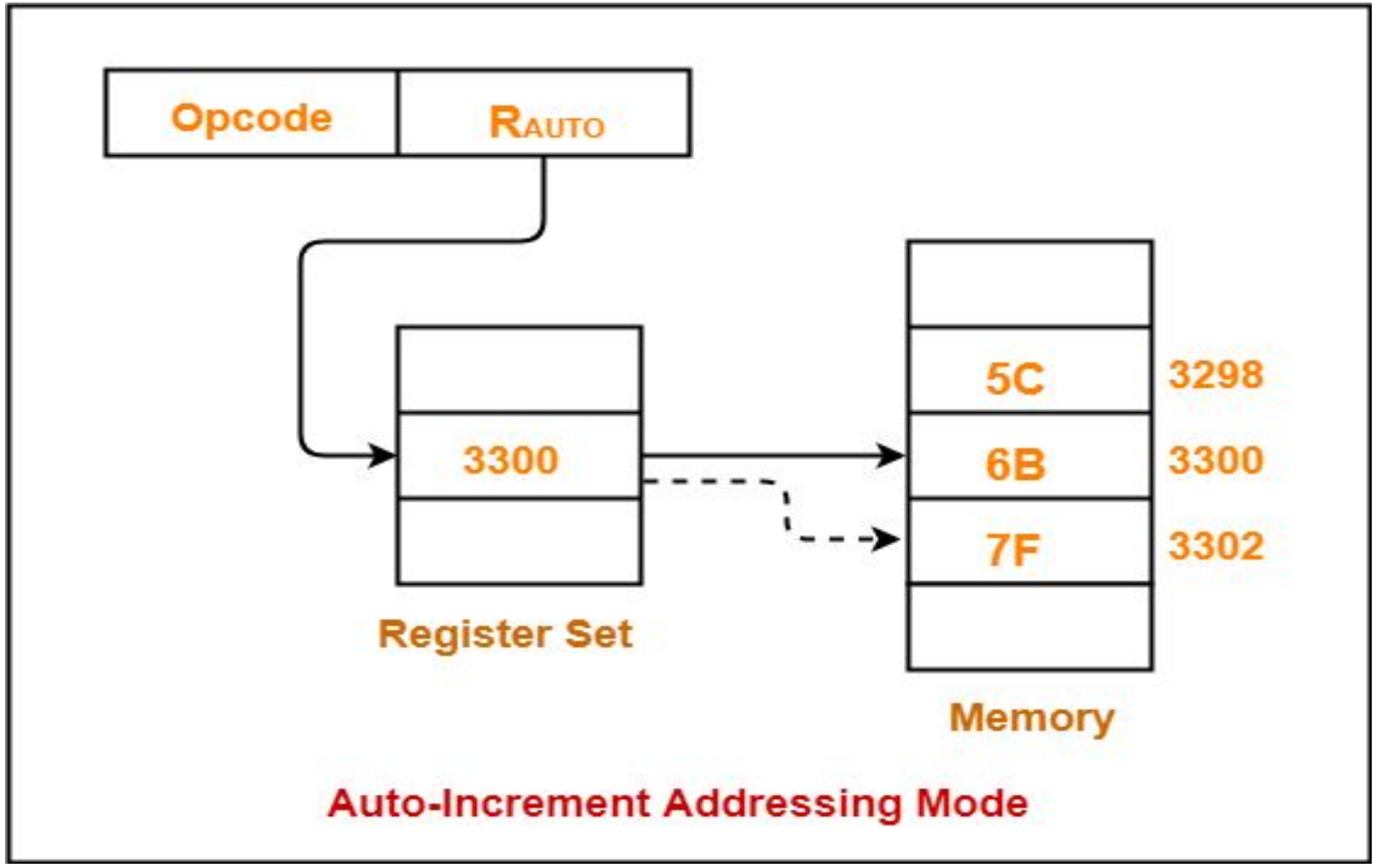
Auto Increment Mode

A special case of Register Indirect Addressing Mode where

Effective Address of the Operand = Content of Register

In this addressing mode,

- After accessing the operand, the content of the register is automatically incremented by step size ‘d’.
- Step size ‘d’ depends on the size of operand accessed.
- Only one reference to memory is required to fetch the operand.


Opcode
RAUTO
3300
Register Set
5C
6B
7F
3298
3300
3302
Memory
Auto-Increment Addressing Mode

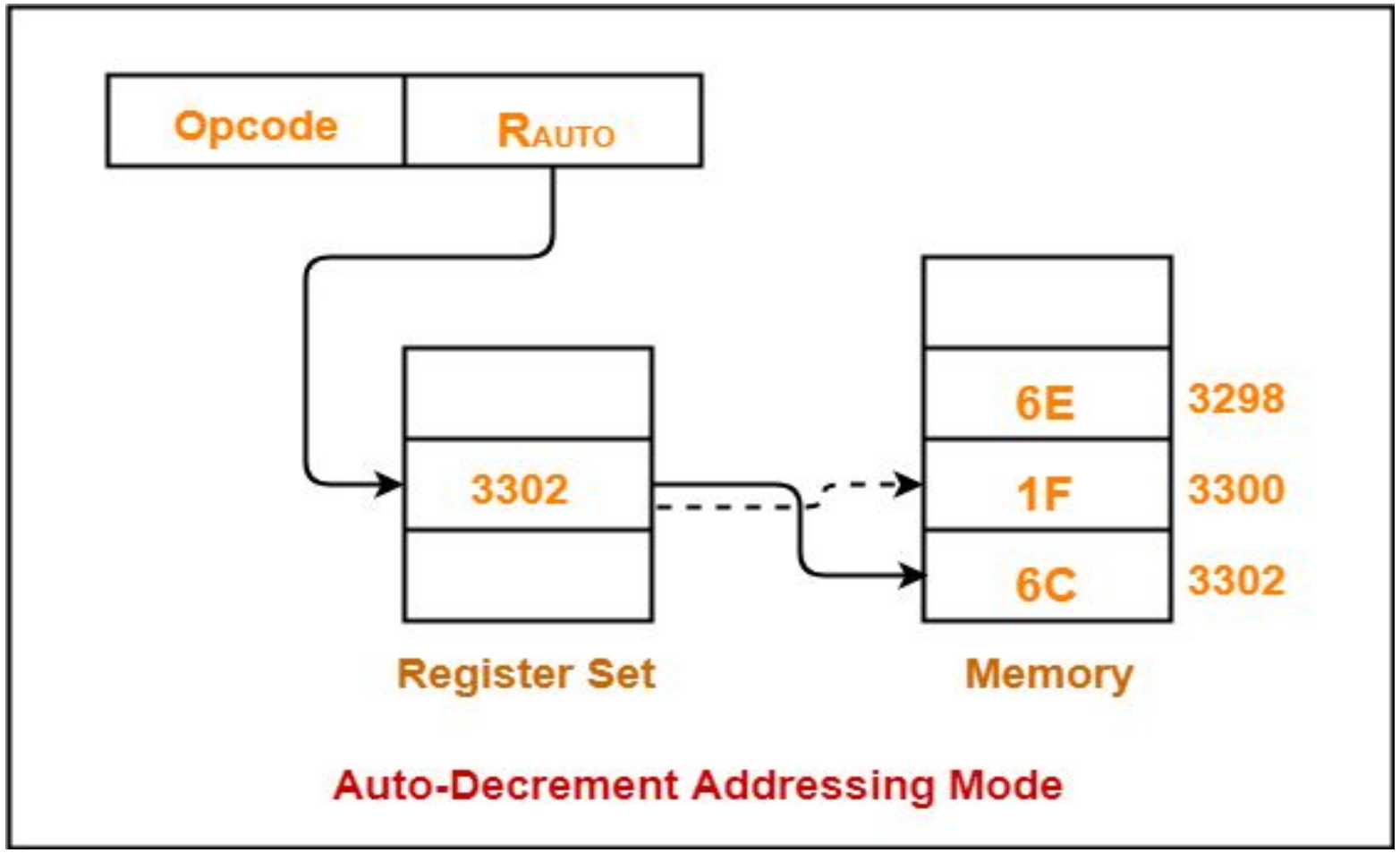
Auto Decrement Mode

- A special case of Register Indirect Addressing Mode where

  Effective Address of the Operand = Content of Register – Step Size

In this addressing mode

- First, the content of the register is decremented by step size ‘d’.
- Step size ‘d’ depends on the size of operand accessed.
- After decrementing, the operand is read.
- Only one reference to memory is required to fetch the operand.


Opcode
RAUTO
3302
6E
1F
6C
3298
3300
3302
Register Set
Memory
Auto-Decrement Addressing Mode

Case Study: 8086
Introduction to Microprocessor

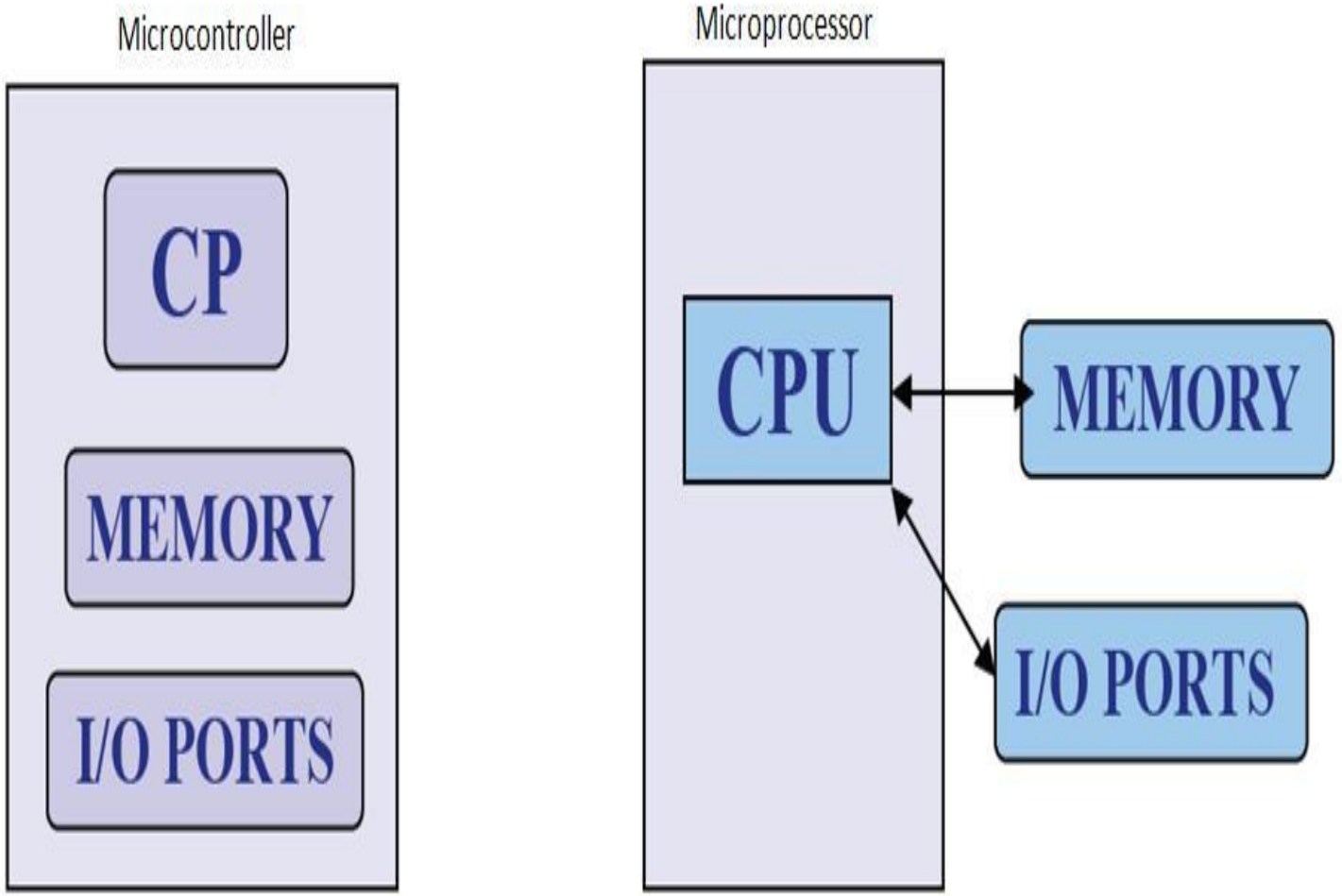
Microprocessors

Microprocessor : is a CPU on a single chip.

Microcontroller: If a microprocessor, its associated support circuitry, memory and peripheral I/O components are implemented on a single chip, it is a microcontroller.

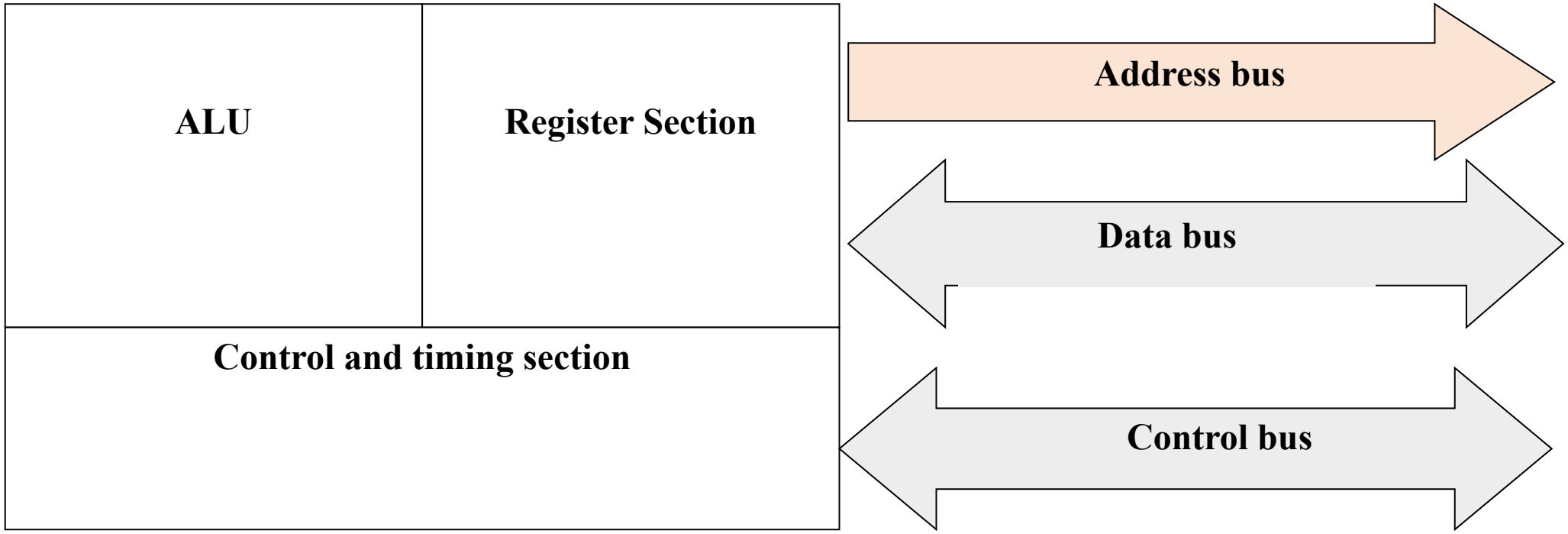
- We use AVR microcontroller as the example in our course study

What is Microprocessor and Microcontroller?



Sl No	Microprocessor	Microcontroller
1	CPU is stand-alone, RAM, ROM, I/O, timer are separate	CP,RAM,ROM,I/O and timer are all on single chip
2	Designer can decide on the amount of ROM,RAM and I/O ports	Fix amount of on-chip ROM, RAM, I/O ports
3	Expansive	Not Expansive
4	General purpose	Single purpose
5	Microprocessor based system design is complex and expensive	Microcontroller based system design is rather simple and cost effective
6	The instruction set of microprocessor is complex with large number of instructions.	The instruction set of a Microcontroller is very simple with the less number of instructions.

Internal structure and basic operation of microprocessor



Block diagram of a Microprocessor

- Microprocessor performs three main tasks:
  - data transfer between itself and the memory or I/O systems
  - simple arithmetic and logic operations
  - program flow via simple decisions

Microprocessor types

- Microprocessors can be characterized based on
  - the word size
    - 8 bit, 16 bit, 32 bit, etc. processors
  - Instruction set structure
    - RISC (Reduced Instruction Set Computer), CISC (Complex Instruction Set Computer)
  - Functions
    - General purpose, special purpose such image processing, floating point calculations
  - And more …

Evolution of Microprocessors

- The first **microprocessor** was **introduced** in 1971 by Intel Corp.
- It was named Intel 4004 as it was a 4 bit processor.

Categories according to the generations or size

First Generation (4 - bit Microprocessors)

- could perform simple arithmetic such as addition, subtraction, and logical operations like Boolean OR and Boolean AND.
- had a control unit capable of performing control functions like
  - fetching an instruction from storage memory,
  - decoding it, and then
  - generating control pulses to execute it.

Second Generation (8 - bit Microprocessor)

- The second generation microprocessors were introduced in 1973 again by Intel.
- the first 8 - bit microprocessor which could perform arithmetic and logic operations on 8-bit words.

Third Generation (16 - bit Microprocessor)

- introduced in 1978
- represented by **Intel's 8086, Zilog Z800 and 80286**,
- 16 - bit processors with a performance like minicomputers.

Fourth Generation (32 - bit Microprocessors)

- Several different companies introduced the 32-bit microprocessors
- the most popular one is the **Intel 80386**

Fifth Generation (64 - bit Microprocessors)

- Introduced in 1995
- After 80856, Intel came out with a new processor namely Pentium processor followed by **Pentium Pro CPU**
- allows multiple CPUs in a single system to achieve multiprocessing.
- Other improved 64-bit processors are **Celeron, Dual, Quad, Octa Core processors**.

Typical microprocessors

- Most commonly used
  - 68K
    - Motorola
  - x86
    - Intel
  - IA-64
    - Intel
  - MIPS
    - Microprocessor without interlocked pipeline stages
  - ARM
    - Advanced RISC Machine
  - PowerPC
    - Apple-IBM-Motorola alliance
  - Atmel AVR

8086 Microprocessor

- designed by Intel in 1976
- 16-bit Microprocessor having
- 20 address lines
- 16 data lines
- provides up to 1MB storage
- consists of powerful instruction set, which provides operations like multiplication and division easily.

supports two modes of operation

Maximum mode : suitable for system having multiple processors

Minimum mode : suitable for system having a single processor.

Features of 8086

- Has an instruction queue, which is capable of storing six instruction bytes
- First 16-bit processor having
  - 16-bit ALU
  - 16-bit registers
  - internal data bus
  - 16-bit external data bus

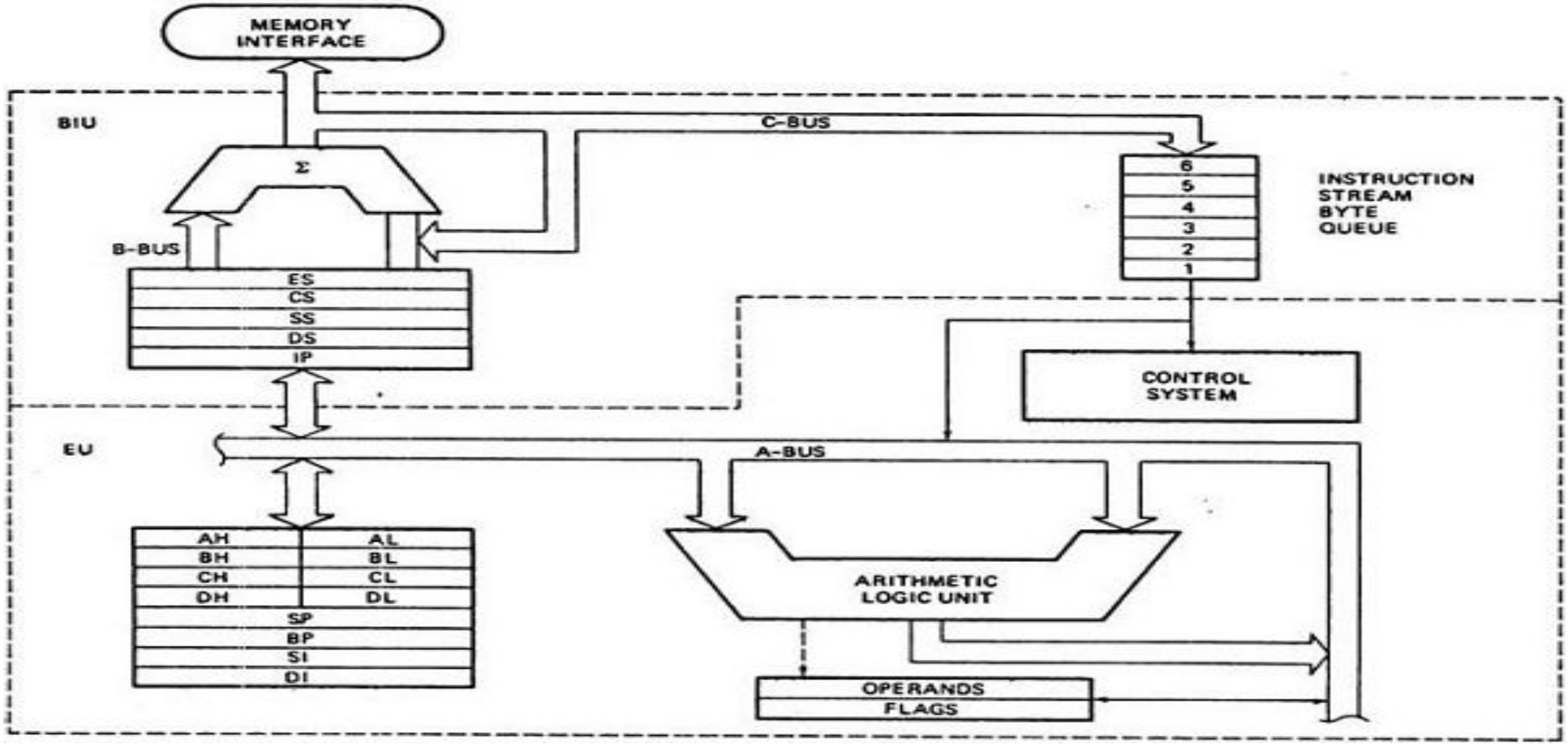
  uses two stages of pipelining
    1. Fetch Stage and
    2. Execute Stage

  which improves performance.

Fetch stage : can pre-fetch up to 6 bytes of instructions and stores them in the queue.

Execute stage : executes these instructions.

Architecture of 8086



Segments in 8086

memory is divided into various sections called segments

Code segment : where you store the program.

Data segment : where the data is stored.

Extra segment : mostly used for string operations.

Stack segment : used to push/pop

General purpose registers

used to store temporary data within the microprocessor

AX – Accumulator

16 bit register

divided into two 8-bit registers AH and AL

to perform 8-bit instructions also

generally used for arithmetical and logical instructions

BX – Base register

16 bit register

divided into two 8-bit registers BH and BL

to perform 8-bit instructions also

Used to store the value of the offset.

CX – Counter register

16 bit register

divided into two 8-bit registers CH and CL

to perform 8-bit instructions also

Used in looping and rotation

DX – Data register

16 bit register

divided into two 8-bit registers DH and DL to

perform 8-bit instructions also

Used in multiplication an input/output port addressing

Pointers and Index Registers

SP – Stack pointer

16 bit register

points to the topmost item of the stack

If the stack is empty the stack pointer will be (FFFE)H

It's offset address relative to stack segment

BP –Base pointer

16 bit register

used in accessing parameters passed by the stack

It's offset address relative to stack segment

SI – Source index register

16 bit register

used in the pointer addressing of data and

as a source in some string related operations

It’s offset is relative to data segment

DI – Destination index register

16 bit register

used in the pointer addressing of data and

as a destination in string related operations

It’s offset is relative to extra segment.

IP - Instruction Pointer

16 bit register

stores the address of the next instruction to be executed

also acts as an offset for CS register.

Segment Registers

CS - Code Segment Register:

user cannot modify the content of these registers

Only the microprocessor's compiler can do this

DS - Data Segment Register:

The user can modify the content of the data segment.

SS - Stack Segment Registers:

used to store the information about the memory segment.

operations of the SS are mainly Push and Pop.

ES - Extra Segment Register:

By default, the control of the compiler remains in the DS where the user can add and modify the instructions

If there is less space in that segment, then ES is used

Also used for copying purpose.

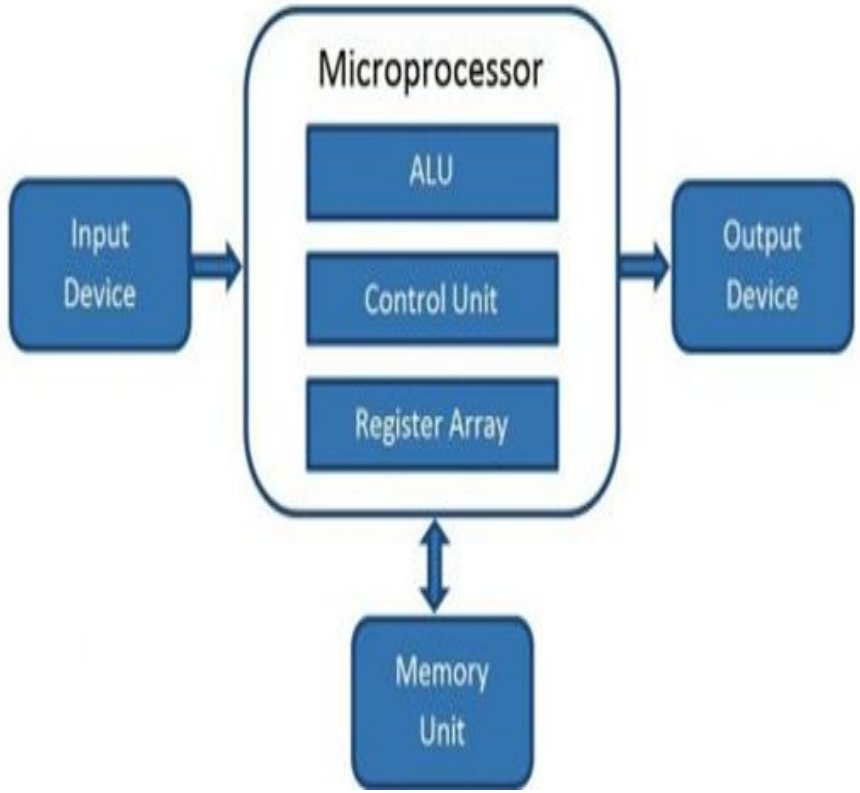
Flag or Status Register

- 16-bit register
- contains 9 flags
- remaining 7 bits are idle in this register
- These flags tell about the status of the processor after any arithmetic or logical operation
- IF the flag value is 1, the flag is set, and if it is 0, it is said to be reset.

Microcomputer

- A digital computer with one microprocessor which acts as a CPU
- A complete computer on a small scale, designed for use by one person at a time
- called a personal computer (PC)
- a device based on a single-chip microprocessor
- includes laptops and desktops

Block Diagram



Introduction to 8086 Assembly Language

Assembly Language Programming

EXAMPLE : Adding two 8 bit numbers

```
DATA SEGMENT                    ; Data Segment
N1
3n2 DB 12H
N2 DB 21H
RES DB ?
DATA ENDS
CODE SEGMENT                    ; Code segment
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
MOV DS, AX
MOV AL, N1
MOV BL, N2
ADD AL, BL
MOV RES, AL
INT 21H
CODE ENDS
END START
```