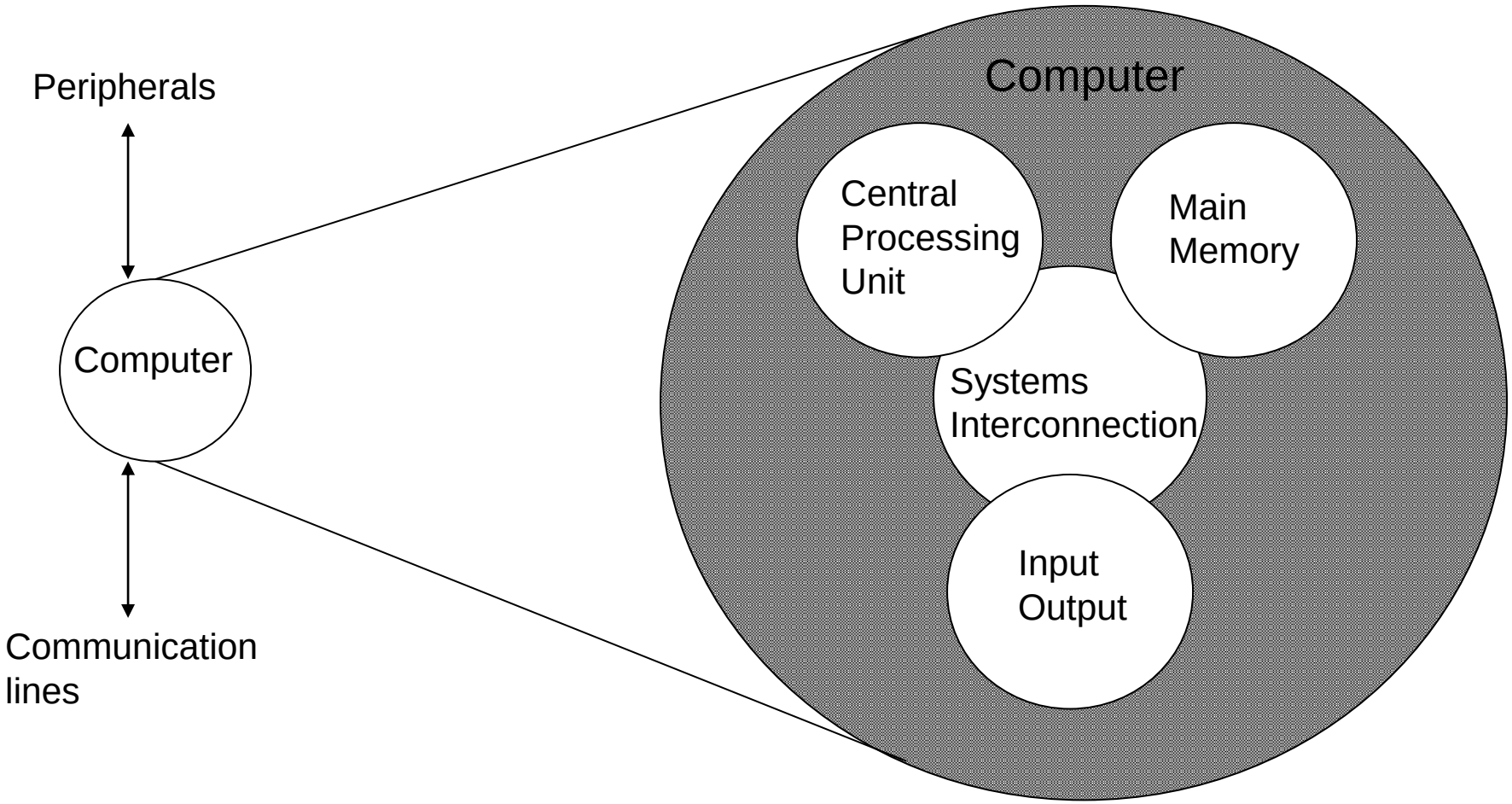
Introduction to Computer Organization & Microprocessors Architecture

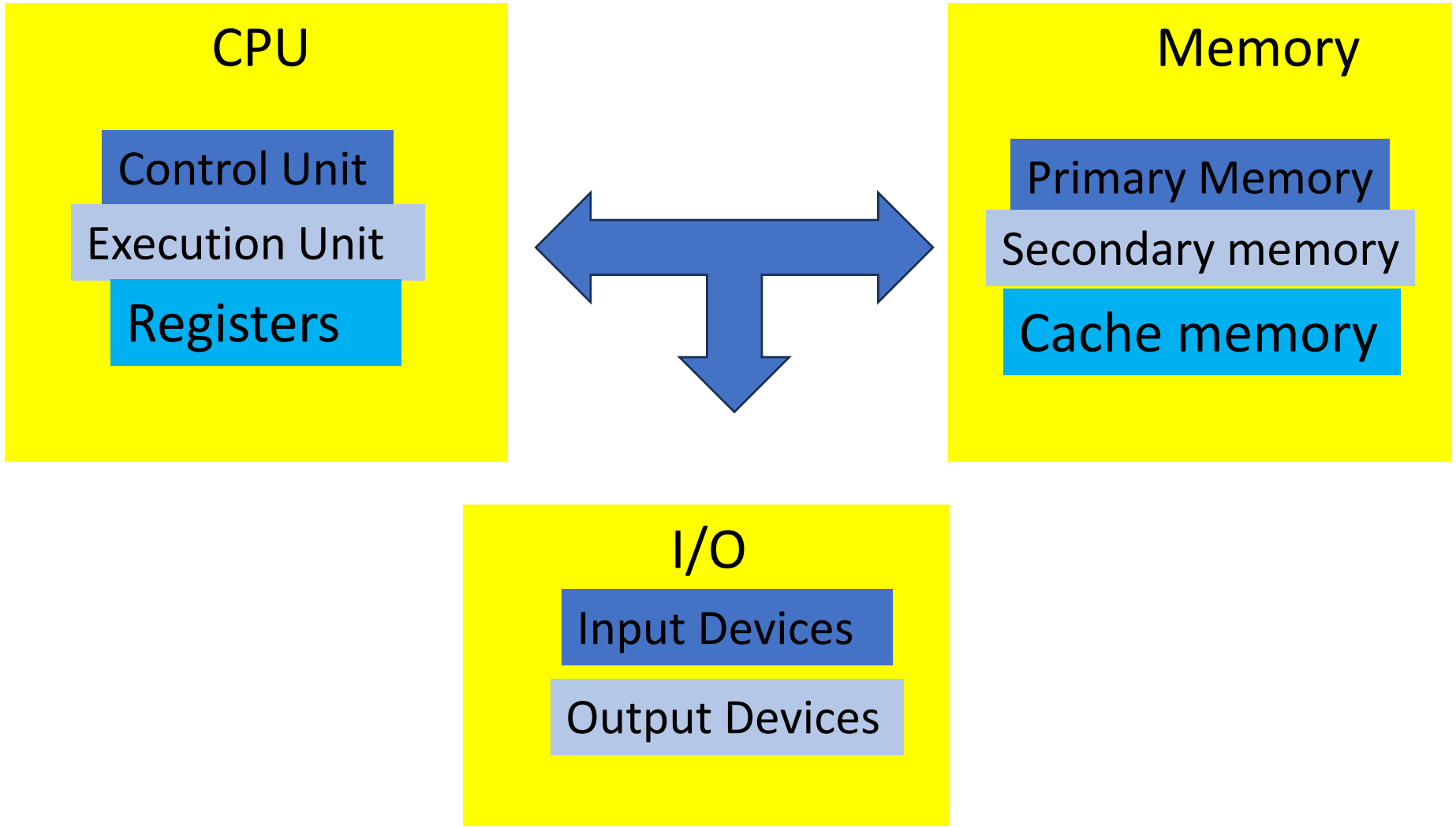
Introduction to Computer Architecture

- A computer is a combination of system software and a set of hardware
- Learning principals of software connected with hardware is the main outcome of computer architecture
- A Computer has main three elements which is known as
    - CPU – Central processing unit
    - Memory
    - Input and output
- These set of main parts are connected Via a system bus to perform communication with each other
- `These main elements also has special features to perform particular tasks

Architecture of a Computer System.

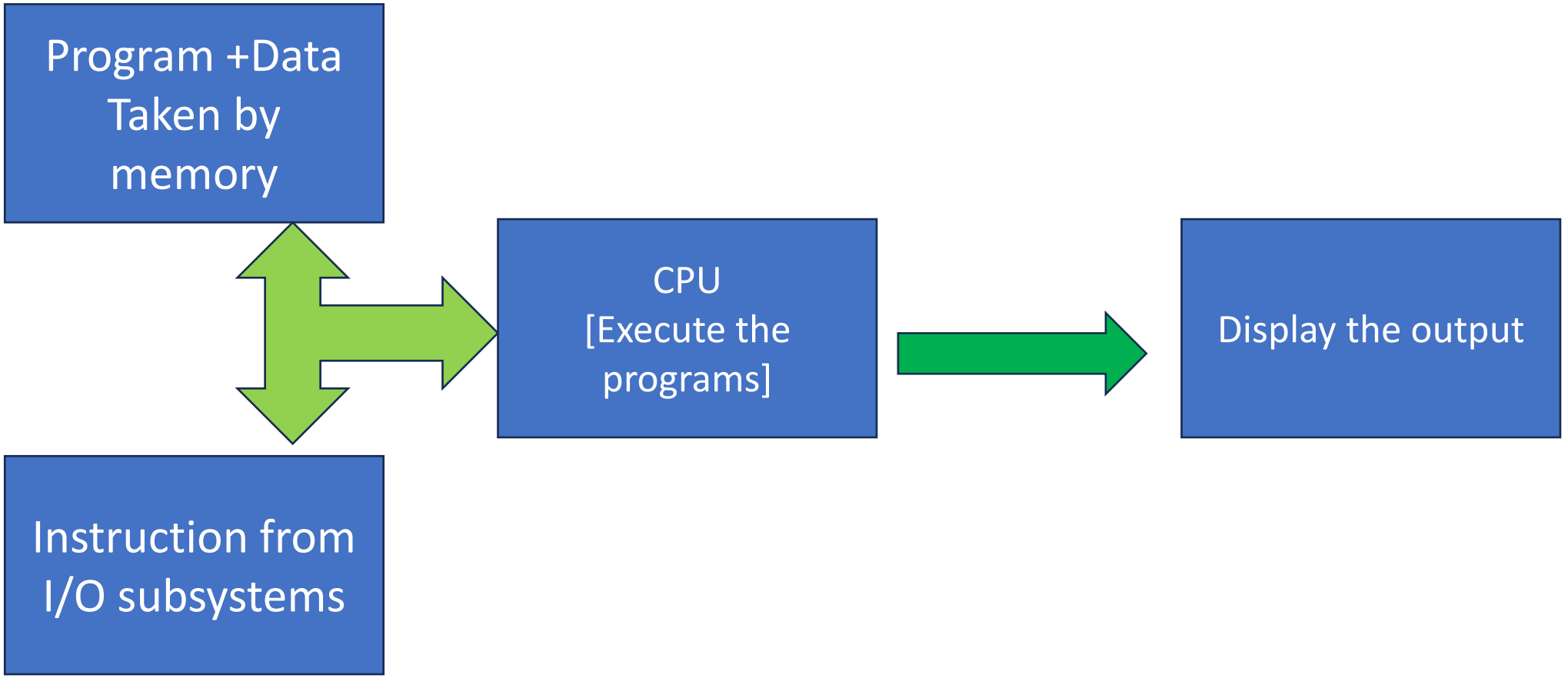


Internal Organization of Elements

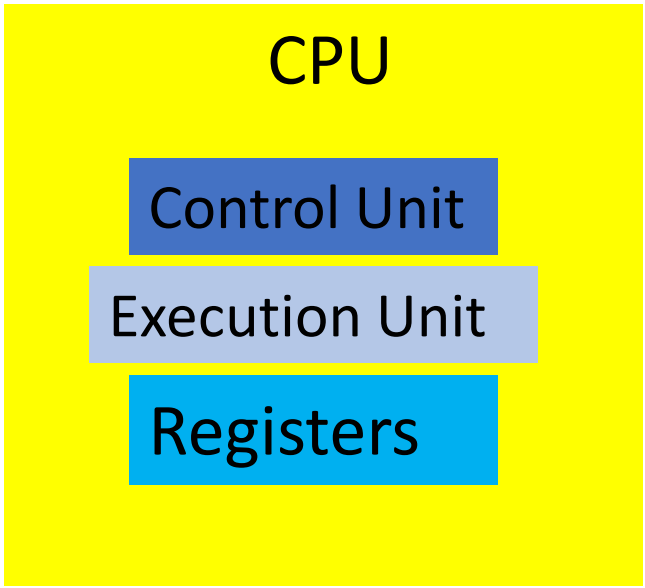


CPU (Central Processing Unit)

- The Central Processing Unit (CPU) is the primary component of a computer system
- Responsible for executing instructions and performing calculations
- processing data and instructions received from software applications and coordinating operations among various hardware components is the key tasks performed by a CPU
- The CPU consists of several key elements, including the Control Unit, Arithmetic Logic Unit (ALU), Registers, and Cache Memory….etc

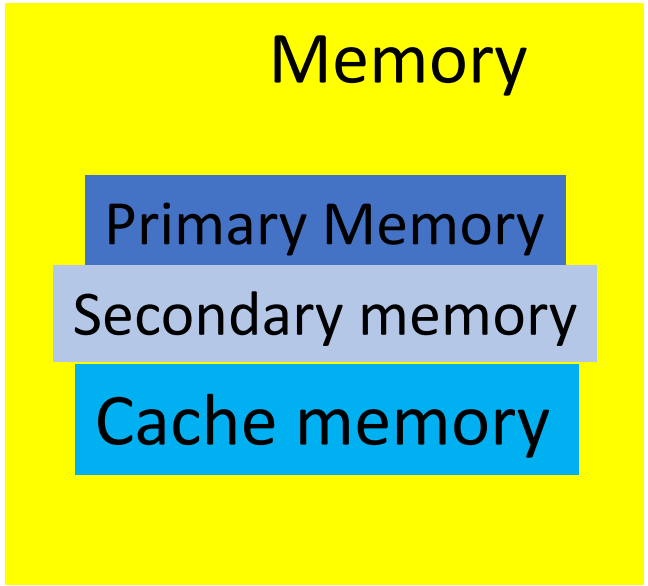

Program +Data Taken by memory
CPU [Execute the programs]
Display the output
Instruction from I/O subsystems

- Control unit- provides all control signals
- CPU will execute the programs one by one and programs are set of instructions
- Executing set of instructions one by one will complete overall execution of a program
- The set of instructions execute one by one means that
- Execution unit – the instructions or data executes three main ways
  - ❑Fetch – Taking instruction from computer memory to CPU
  - ❑Decode-Identifying or understanding which operation should perform with given instruction (eg: Addition, Multiplying)
  - ❑Execute –Performing the operations with ALU
- Registers- hold or Stroe data values



Memory

- In computer architecture, memory refers to the physical devices and components that store data and instructions temporarily or permanently for processing by the CPU (Central Processing Unit).
- Three types can be mentioned under the memory
  - Primary memory
  - Secondary memory
  - Cache memory

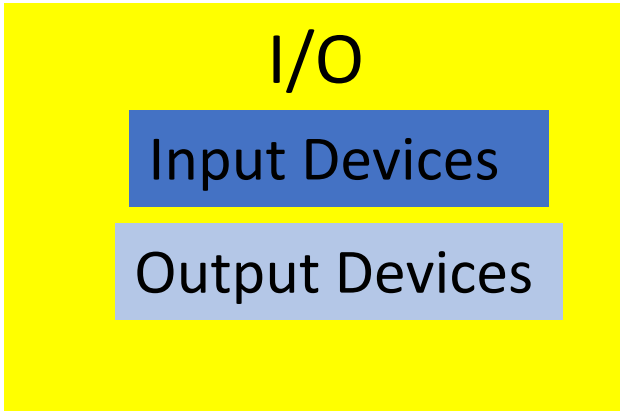


Memory

1. **Primary Memory (Main Memory):** This type of memory is directly accessible to the CPU and is used for storing data and instructions that are actively being processed. It includes components such as registers memory, RAM (Random Access Memory).
2. **Secondary Memory (Storage):** Secondary memory refers to storage devices that hold data and instructions for long-term or permanent storage. As examples hard disk drives (HDDs), solid-state drives (SSDs), optical drives, and flash drives.
3. **Cache Memory:** Cache memory is a small, high-speed type of memory located on the CPU or between the CPU and main memory. Its primary function is to store frequently accessed data and instructions to reduce the average time taken to access memory.

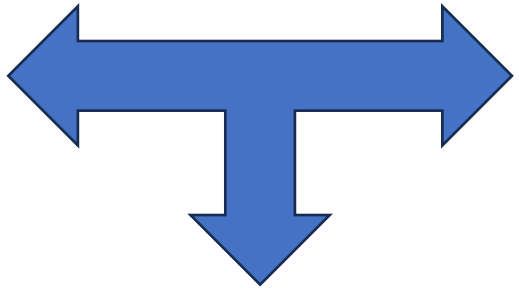
Input output sub systems

- Provides the means of communication between the computer system and the outside world
- Keyboard, mouse are input subsystems where monitor displays and printers are output subsystems and universal cable are communication sub systems that can be mentioned as I/O subsystems



CPU

Used to execute programs



Memory

Programs are in the memory

Program

Data

I/O

Data ?

Program ?

A Program

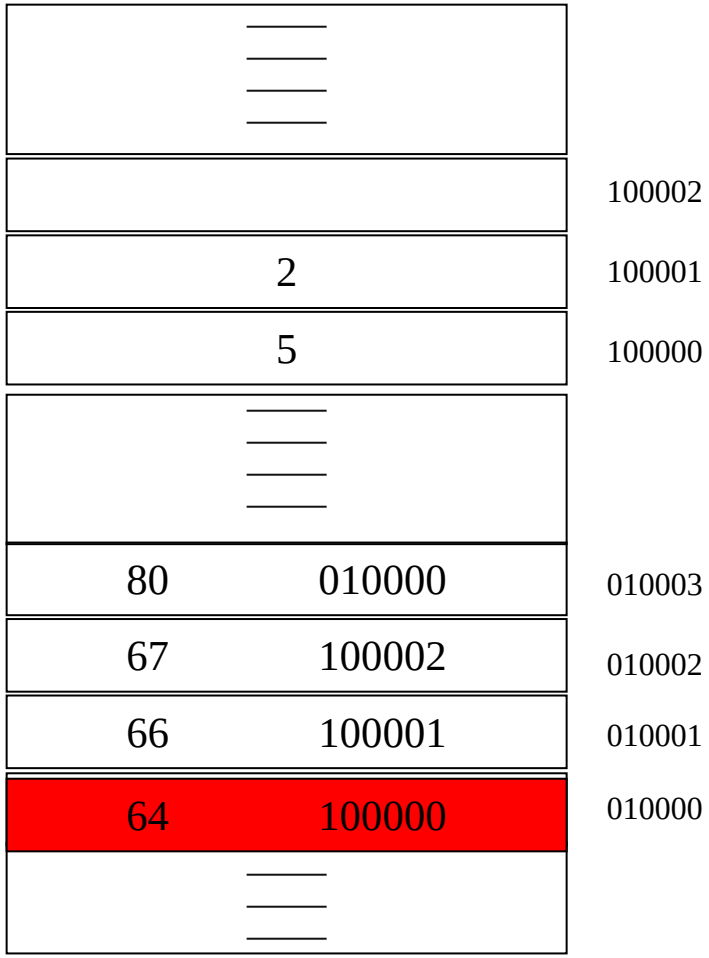
Program is a set of instruction arranged to achieve a specific task.

These instruction are stored in memory.

Microprocessor fetches instruction from memory, decode and execute them one by one. That means one after the other.

Therefore basically it involves two phases FETCH & EXECUTE, for each and every instruction.. It is possible to have many phases more than two.

Main memory



- Lets consider about **listing to a song using a computer**
- Steps how it process

St01
- Input devices such as key board will give the instruction to CPU

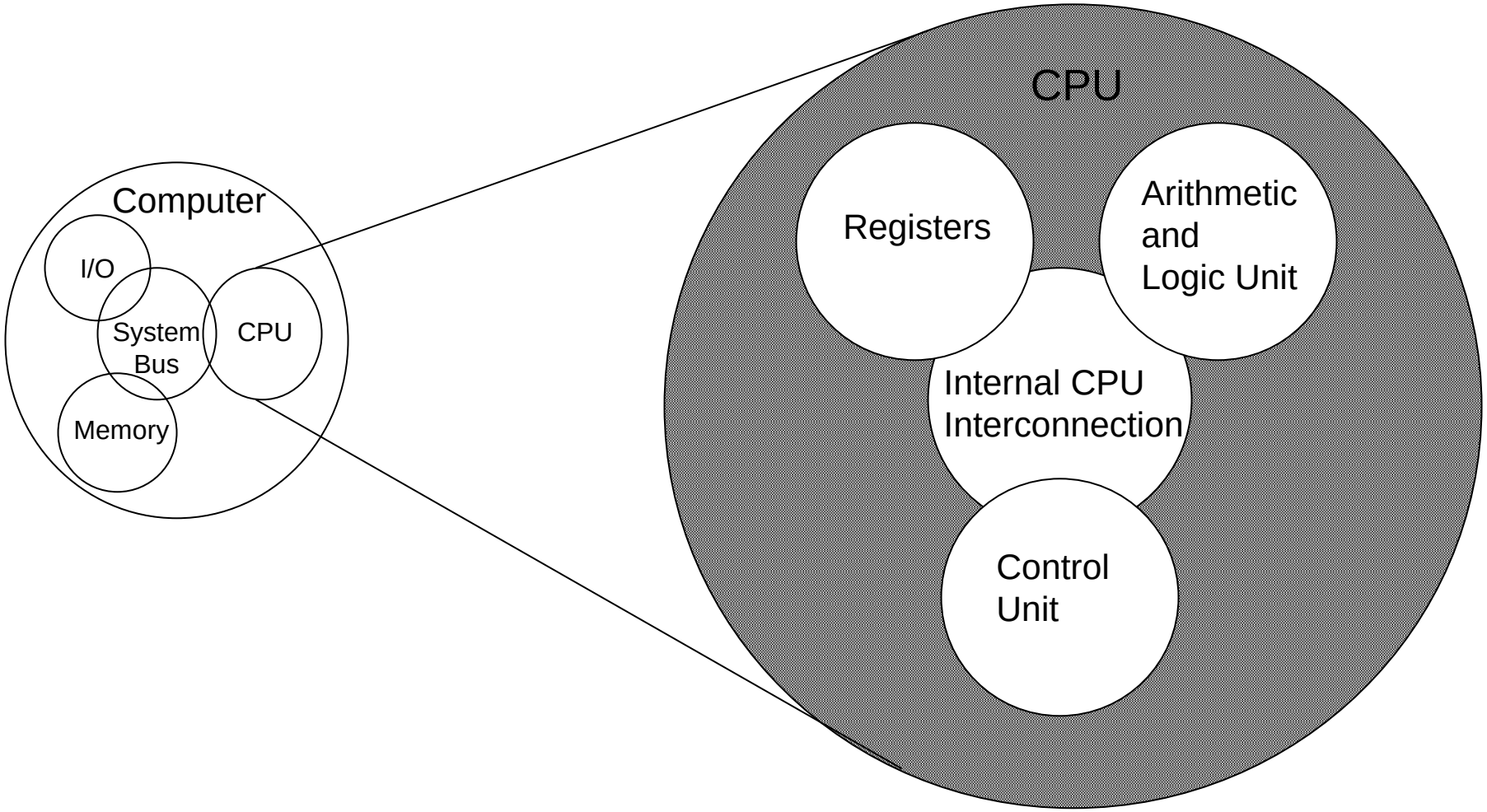
St02
- The Song is the Data which stored I the memory

St03
- Mp3 player will active and start playing the song
- Mp3 player is the program now. CPU will execute program

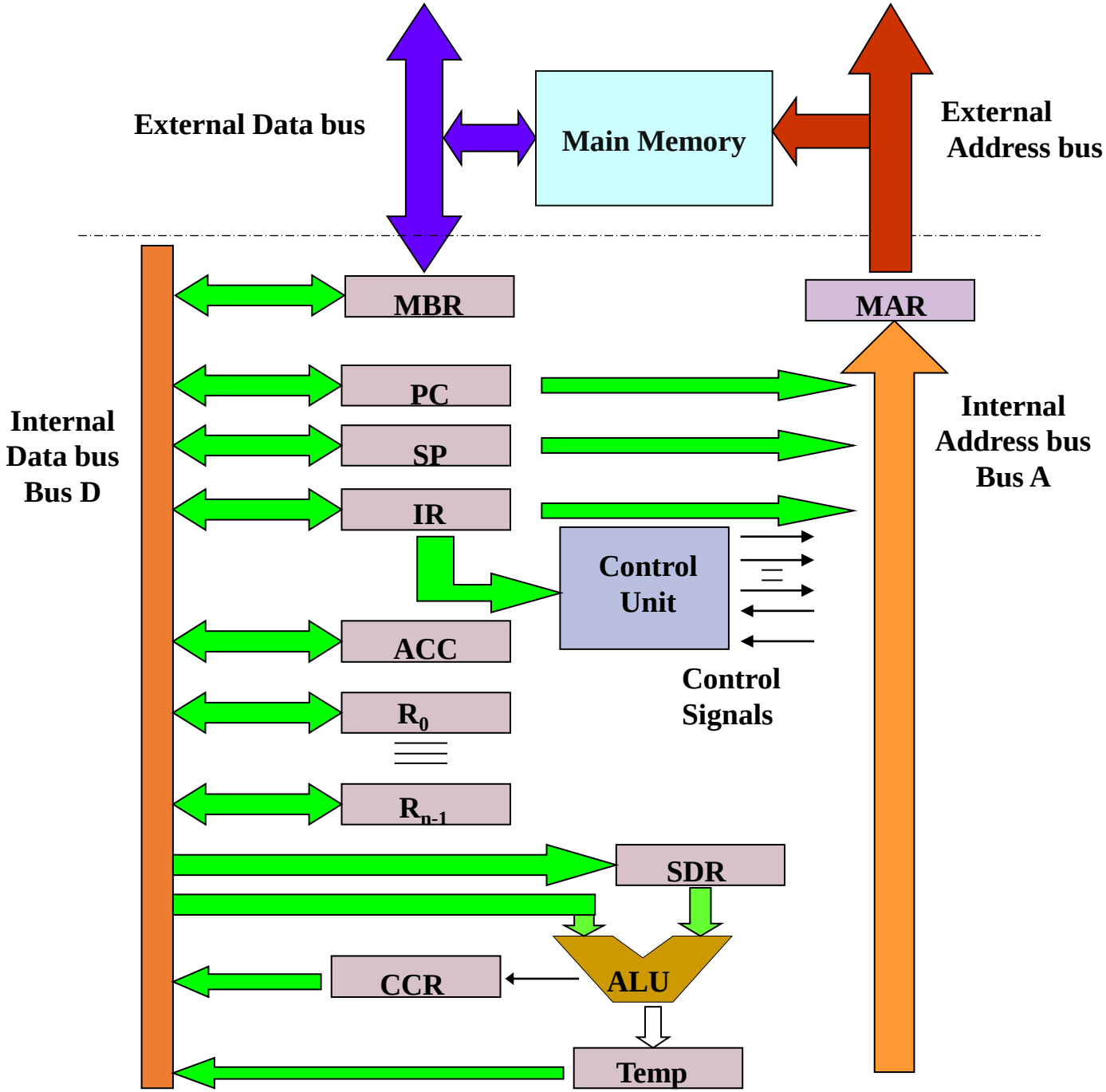
St04
- By using output devices, the output can listen

St05
- All communication signals passes through I/O subsystem buses

Microprocessor Architecture



Internal structure of a Typical Microprocessor


External Data bus
Main Memory
External
Address bus
MBR
MAR
PC
SP
IR
Internal
Data bus
Bus D
Internal
Address bus
Bus A
Control
Unit
ACC
Control
Signals
R_0
R_{n-1}
SDR
CCR
ALU
Temp

Registers

High-speed memory locations,

Hold instructions, data, variables, flag, etc temporarily.

Basically there are two categories of registers inside a microprocessor

I) General purpose registers

These are used to hold data, variables, results of an operation.

II) Special purpose registers

These are used to hold addresses, instructions, flags etc.

A Typical Microprocessor

- MBR Memory Buffer Register
- MAR Memory Address Register
- PC Program Counter
- SP Stack Pointer
- IR Instruction Register
- ACC Accumulator
- R0..Rn-1 General purpose registers
- SDR Store Data Register
- CCR Condition Code Register (Status Register)
- Temp Temporary Register
- ALU Arithmetic & Logic Unit

Purpose of Registers of a Micro Processor

- Registers are used to store or hold data and instruction in the CPU
- It will take the data or instruction from memory, store them and perform the instructions
- These instructions has one format
- One particular instruction has Opcode and an Operand
- Both parts carries different information

Structure of an Instruction

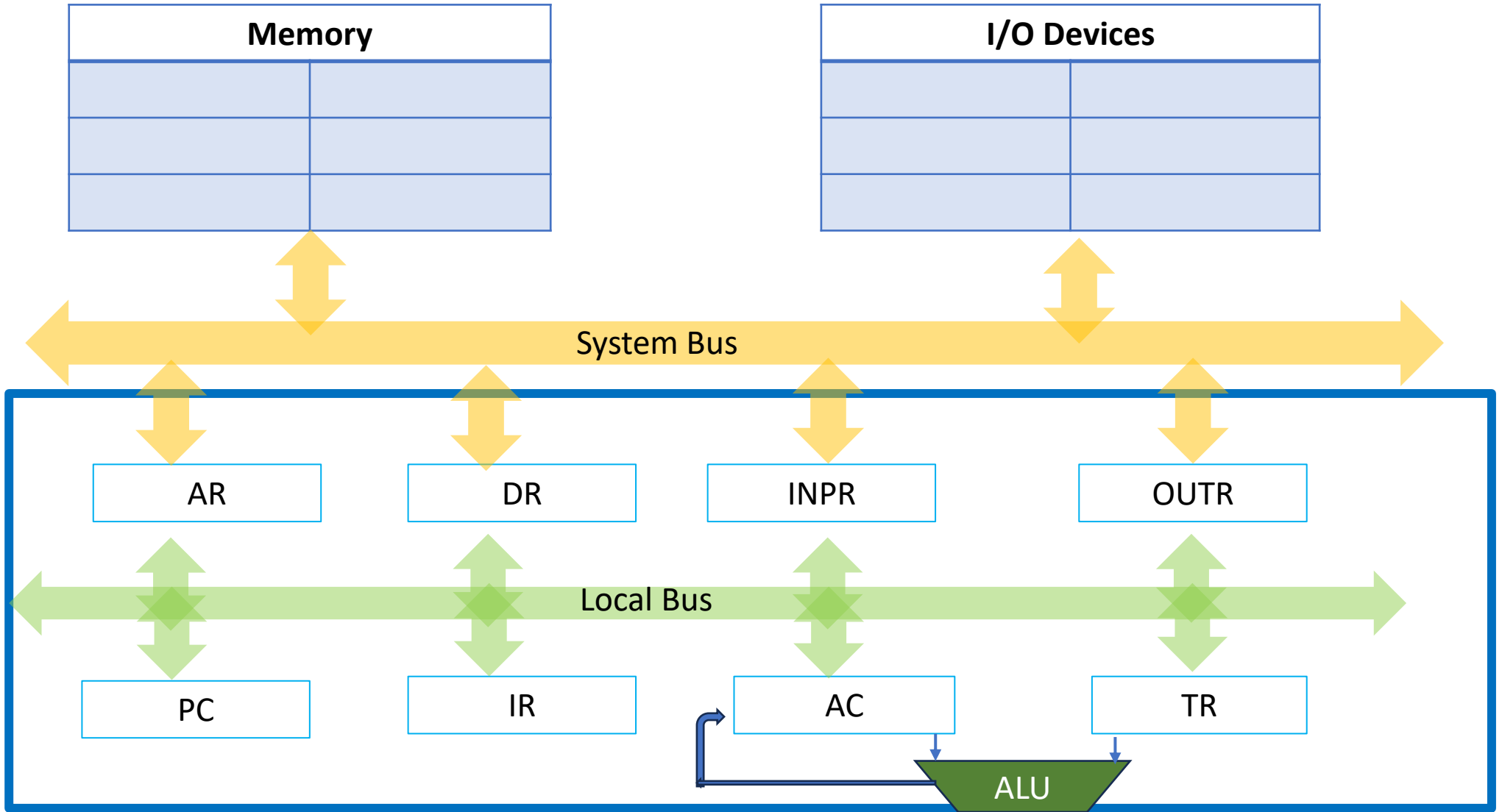
Opcode : Carries the instruction to the control unit
Eg: Instruction such as Add, Multiply, store, move

Operand: Provides the memory location, where should the instruction store
Eg: In the instruction JUMP 100, the 100th memory location will be the Operand

Types of registers and Application of them

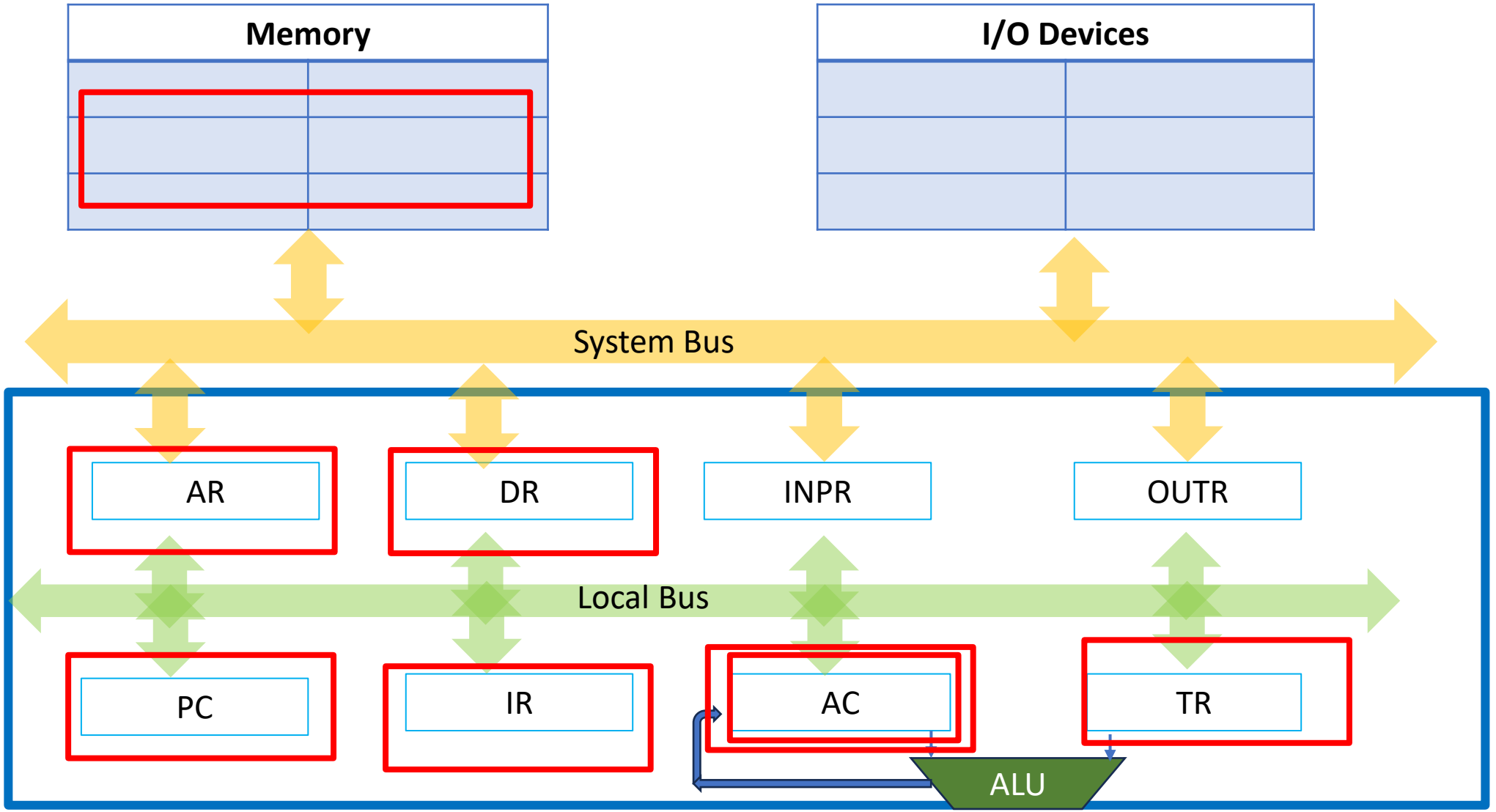
- AR/MAR - Address Register - It holds address of the memory
- PC -Program Counter – It hold the address of next instruction
- IR - Instruction Register – After fetching instruction from the memory it will store in the IR register
- DR - Data register – It has data operands from the memory
- AC - Accumulator – It is process register for ALU. ALU use it for all arithmetic and logical operations
- TR - Temporary Register – Hold data temporary while instruction execution

Register Organization



Executing Single Instruction from the memory

1. Address of the instruction is in program counter (PC)
   - PC → Give the Address to AR (Address Register)
2. Address register connect memory through system bus
   - Data from memory → Data Register
3. Data Register → IR register
   - IR register will identify the instruction Using Opcode and
4. Operand address again take information from memory and provide to TR register
5. According to the Opcode ALU perform Arithmetic and logical task and value passes to AC (Accumilator)


Memory
I/O Devices
System Bus
AR
DR
INPR
OUTR
Local Bus
PC
IR
AC
TR
ALU

ALU - Arithmetic and Logic Unit

This performs all the arithmetic ,logical and comparative operations and put the result into a register.

Arithmetic operations: addition, subtraction. (multiplication and division done by repeated addition and shift)

Logical operations: AND, OR, NOT,NAND, NOR, XOR, Complement, shifting(R/L), swapping.

Comparisons operations: greater than , less than , Greater than or equal , less than or equal ,not equal .

Control Unit

This is responsible for controlling everything internal and external to the microprocessor.

Generates all the required control and timing signals

Critical part of the microprocessor

There are different approaches to design the control unit

In the next section we will learn in detail about two major approaches

Assembly language

Instructions in that format is called mnemonics.

Programs written in that format is called Assembly Language programs

Still assembly language programs are difficult to writ and debug

Therefore, peoples developed high level Languages.

These are very close to natural languages (most of the time English)

Then it becomes easy to write and debug programs

Micro-Operations

Why we learn assembly language?

Programs written in assembly language runs very fast compared to programs written in HLLs.

Programmer has a greater control over the hardware resources.

Therefore, code can be fully optimized.

It is easy to learn the internal operation of a microprocessor if we are familiar with the assembly language

So, we will learn little bit about assembly language

A simple Assembly Language Program

LOAD A,(100000)

64	100000
Opcode	Operand

Load the Accumulator (ACC) with the content of the memory location 100000

ADD A,(100001)

66	100001

Add the Accumulator with the content of the memory location 100001. Result place back in ACC

STORE A,(100002)

67	100002

Store the content of Accumulator at the memory location 100002.

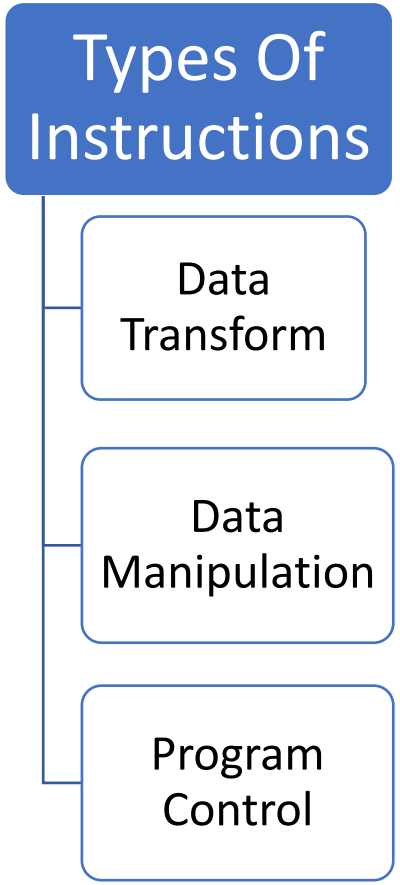
JUMP 010000

80	010000

Jump to the memory location 010000 and execute the instruction stored in that memory location.

Note: Numbers in Decimal. Opcodes are represented by example numbers.

Type Of Instructions



Mov (move instruction) **LD** (load instruction) **ST** (Store instruction)
IN **Out** (input output instruction)
Push **Pop** (with stack pointer)

+, -, X,%
AND, OR, NOT, XOR
Shift, Rotate

Jump
Call

Instruction Cycles

- Mainly there are three types of instruction cycles found in Computer Architecture
  - Fetch Instruction cycles
  - Decode Instruction cycles
  - Execute Instruction cycles

  Are the main types

Micro operations for the

LOAD A,(100000)

FETCH cycle

PC → Bus A

Bus A → MAR

MAR → Address bus

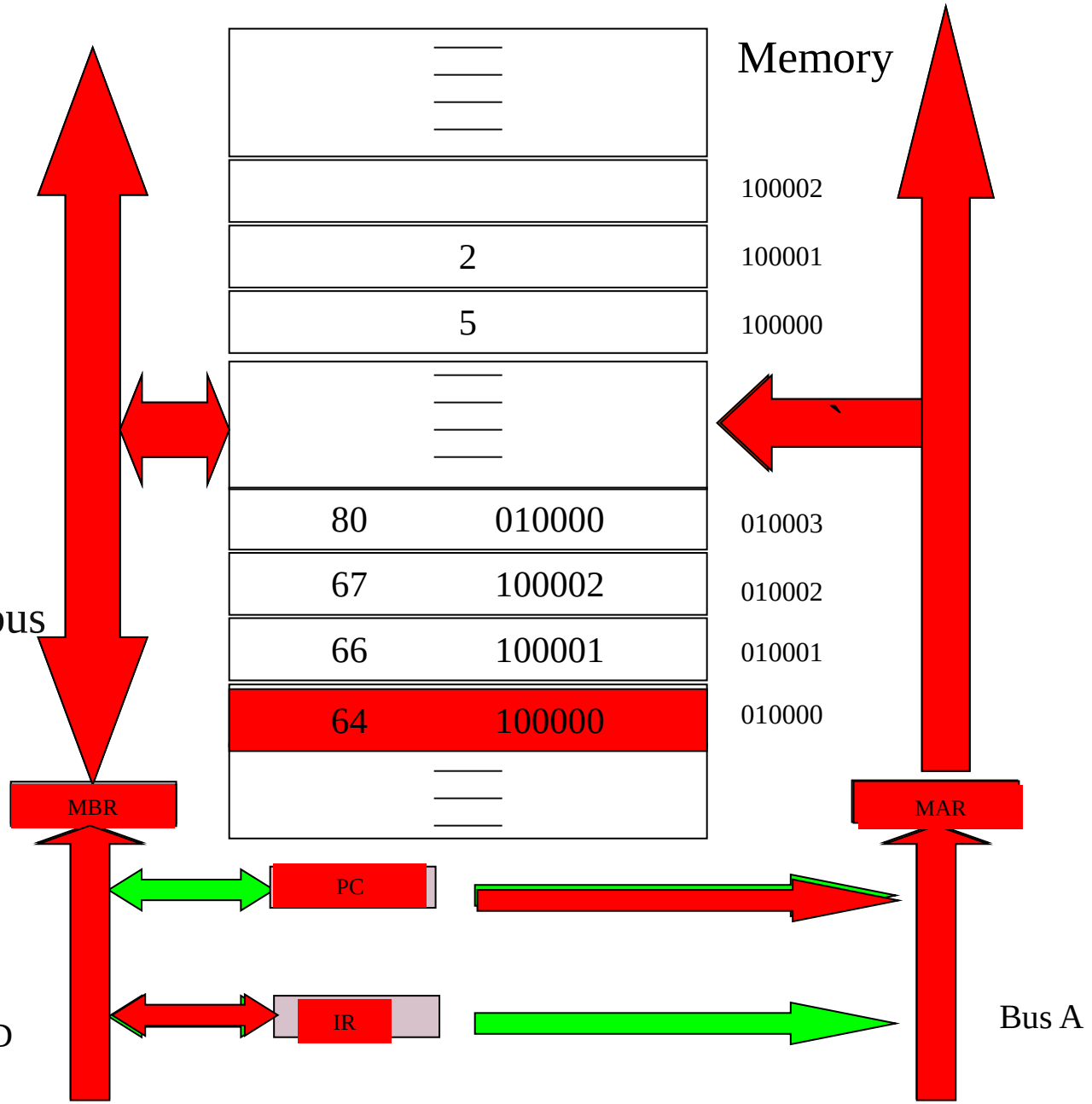
Set memory to read

Data bus → MBR

MBR → Bus D

Bus D → IR

Increment PC



Write down corresponding control signals to be activated for each and every micro operation

Micro operation	Controls to activate
PC → Bus A	(PC)out A
Bus A → MAR	
MAR → Address bus	
Set memory to read	
Data bus → MBR	
MBR → Bus D	
Bus D → IR	
Increment PC	

Micro operations for the

LOAD A,(100000)

Execution cycle

IR → Bus A

Bus A → MAR

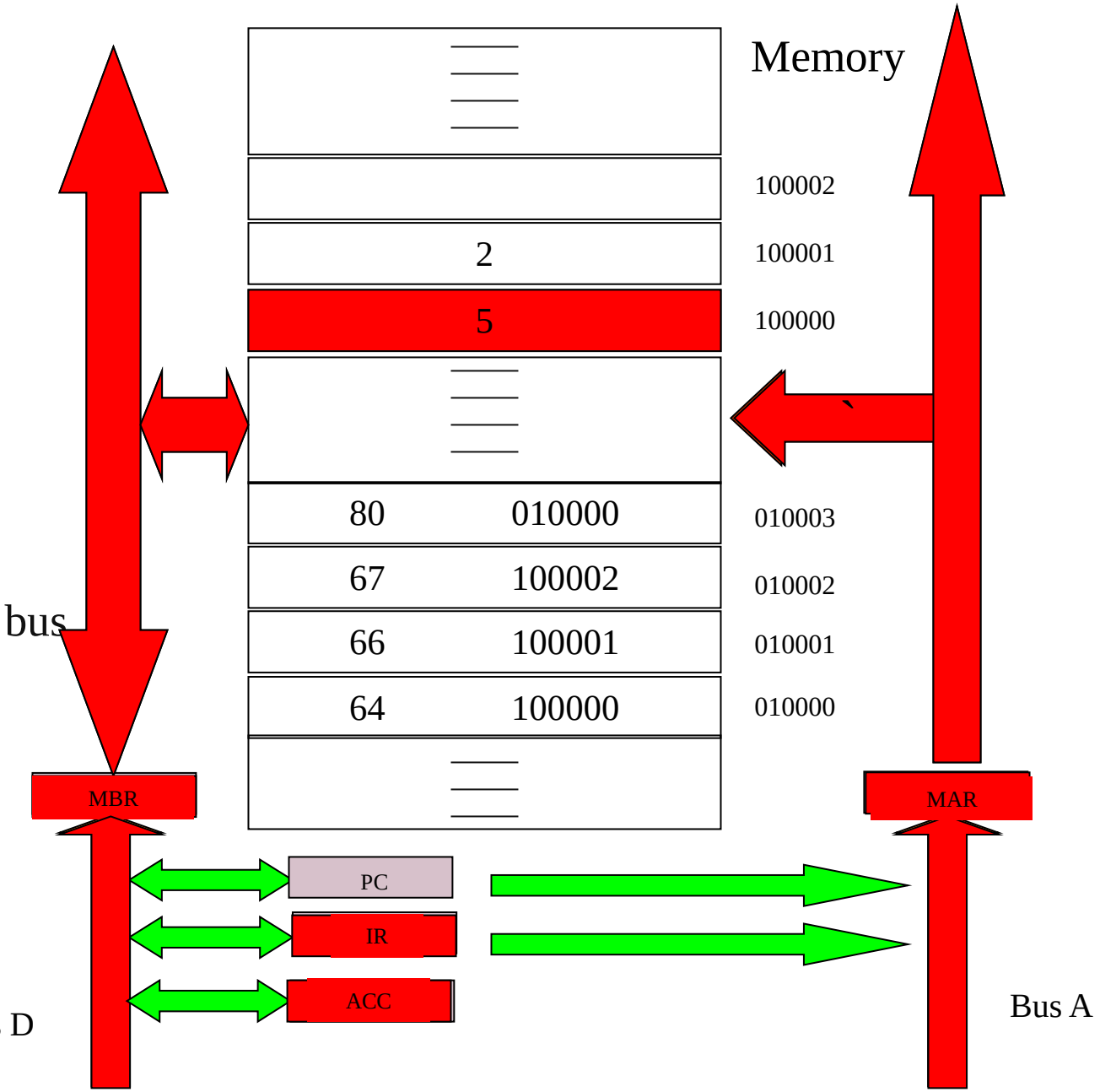
MAR → Address bus

Set memory to read

Data bus → MBR

MBR → Bus D

Bus D → ACC



Write down corresponding control signals to be activated for each and every micro operation

Micro operation	Controls to activate
IR → Bus A	(IR)out A
Bus A → MAR	
MAR → Address bus	
Set memory to read	
Data bus → MBR	
MBR → Bus D	
Bus D → ACC	

Micro operations for the

ADD A,(100001)

FETCH cycle

PC → Bus A

Bus A → MAR

MAR → Address bus

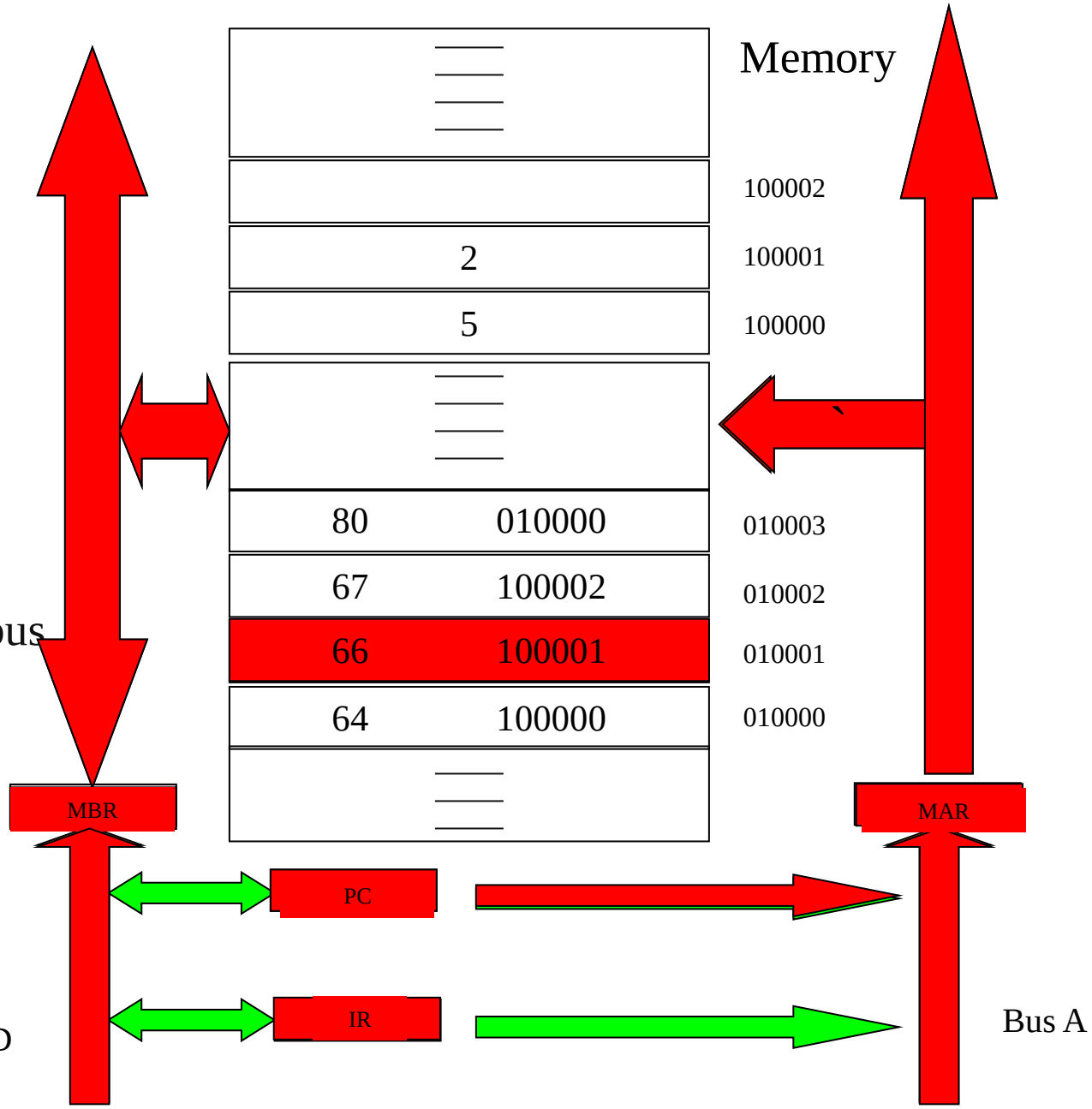
Set memory to read

Data bus → MBR

MBR → Bus D

Bus D → IR

Increment PC



Micro operations for the

ADD A,(100001)

Execution cycle

IR ⟶ Bus A

Bus A ⟶ MAR

MAR ⟶ Address bus

Set memory to read

Data bus ⟶ MBR

MBR ⟶ Bus D

Bus D ⟶ SDR

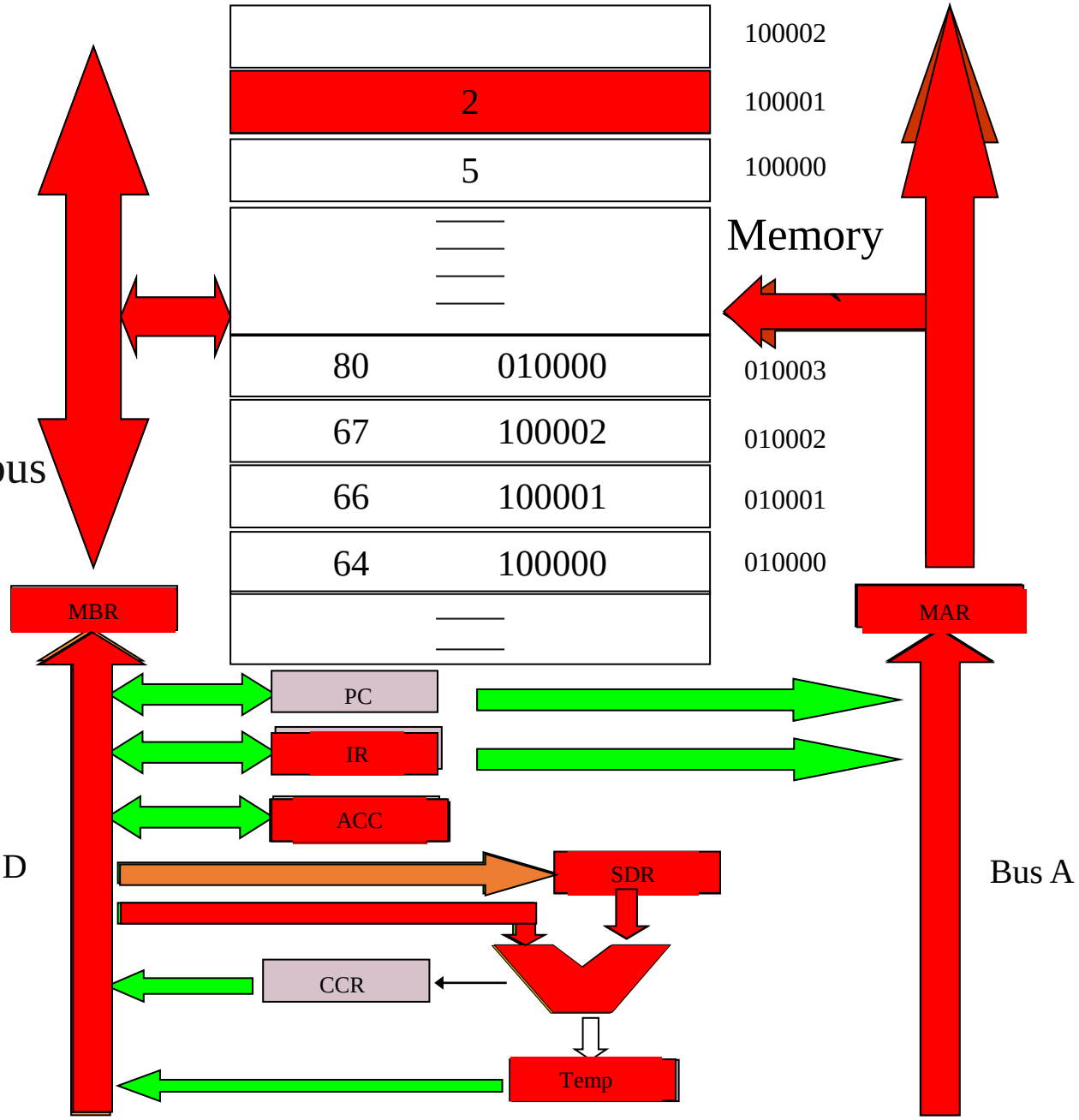
ACC ⟶ Bus D

Set ALU to ADD

(result goes to TEMP)

TEMP ⟶ Bus D

Bus D ⟶ ACC



Micro operations for the

STORE A,(100002)

FETCH cycle

PC → Bus A

Bus A → MAR

MAR → Address bus

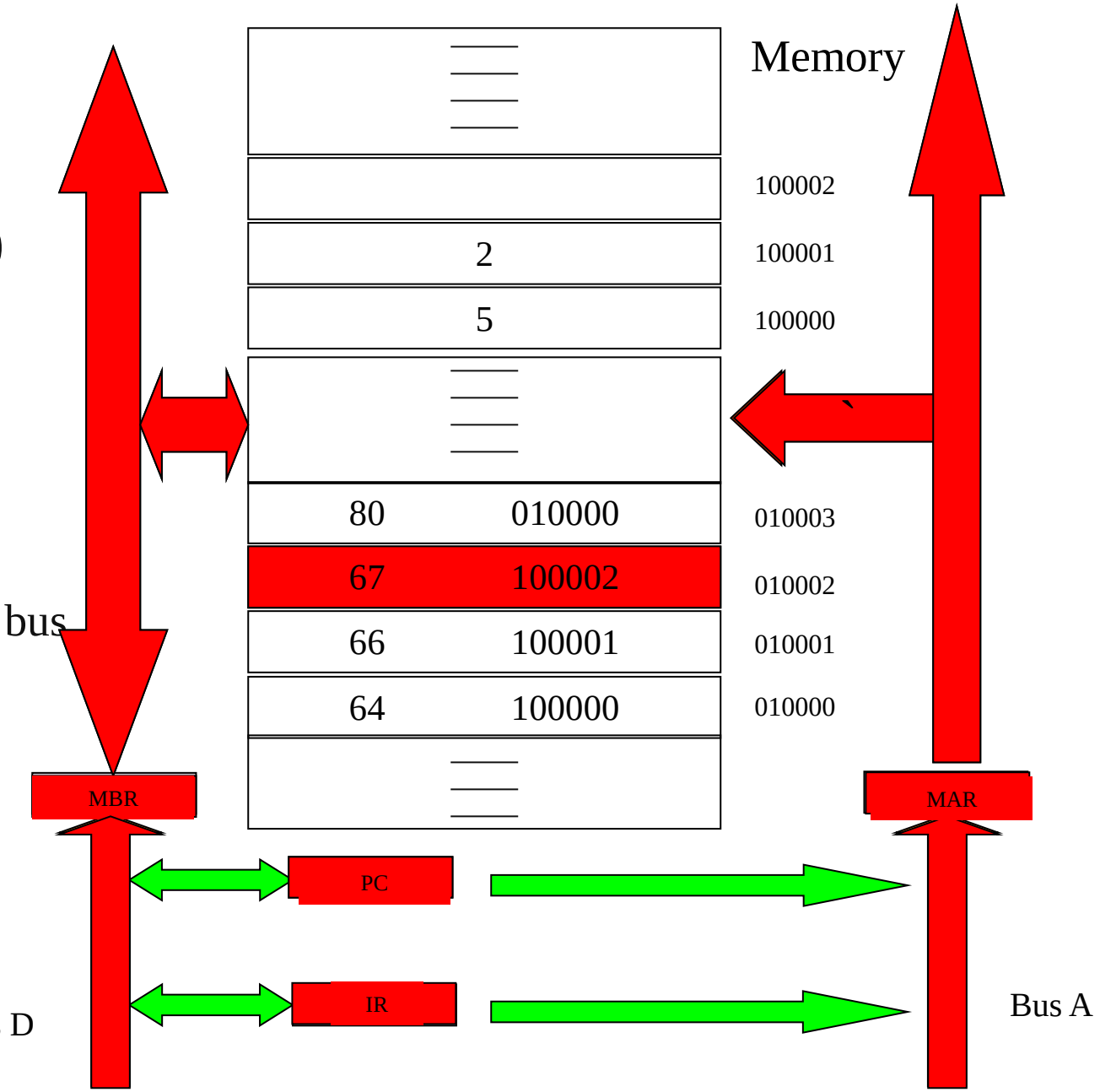
Set memory to read

Data bus → MBR

MBR → Bus D

Bus D → IR

Increment PC



Bus D BR

Micro operations for the

STORE A,(100002)

Execution cycle

IR → Bus A

Bus A → MAR

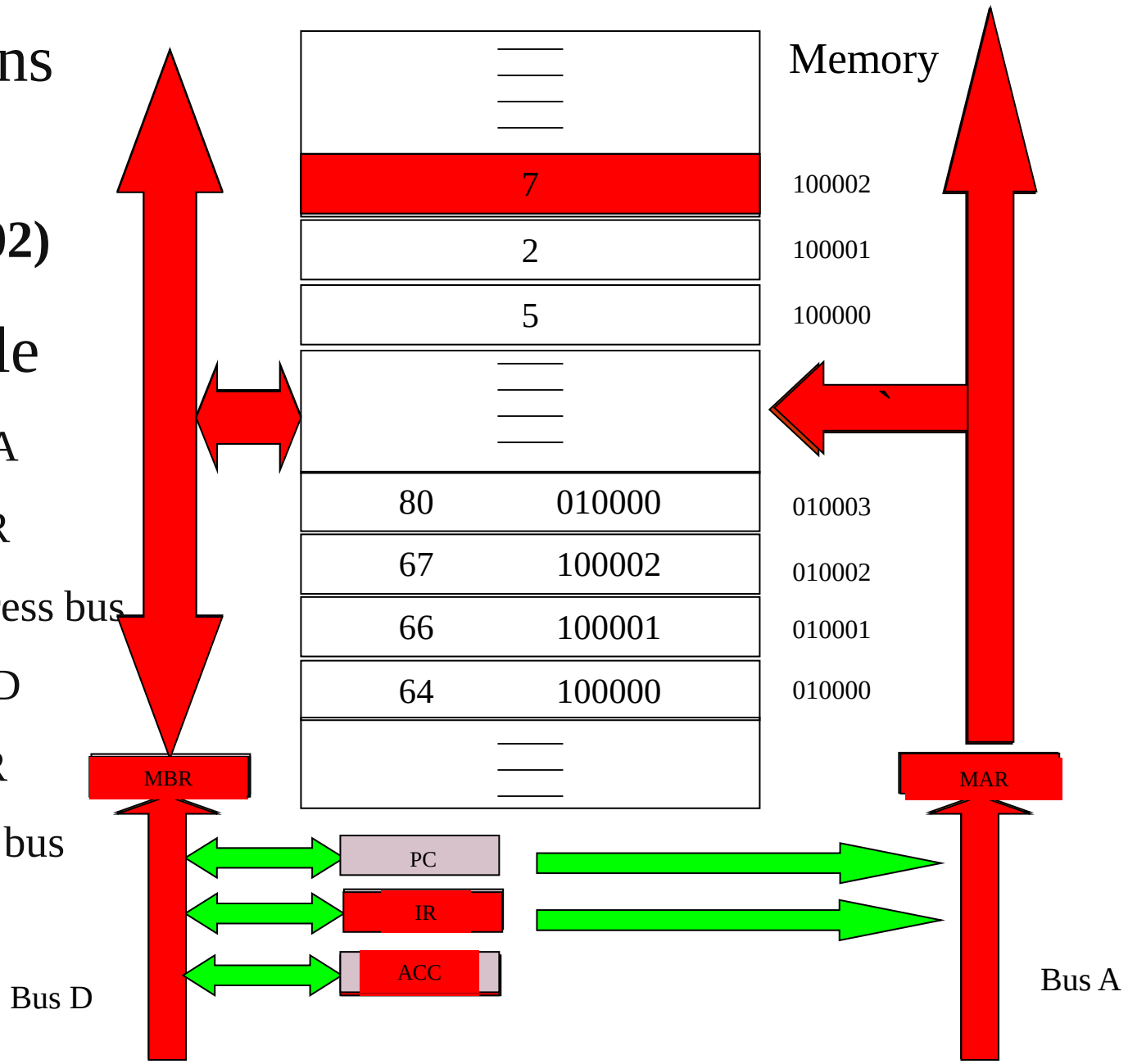
MAR → Address bus

ACC → Bus D

Bus D → MBR

MBR → Data bus

Set memory to Write



Micro operations for the

JUMP 010000

FETCH cycle

PC → Bus A

Bus A → MAR

MAR → Address bus

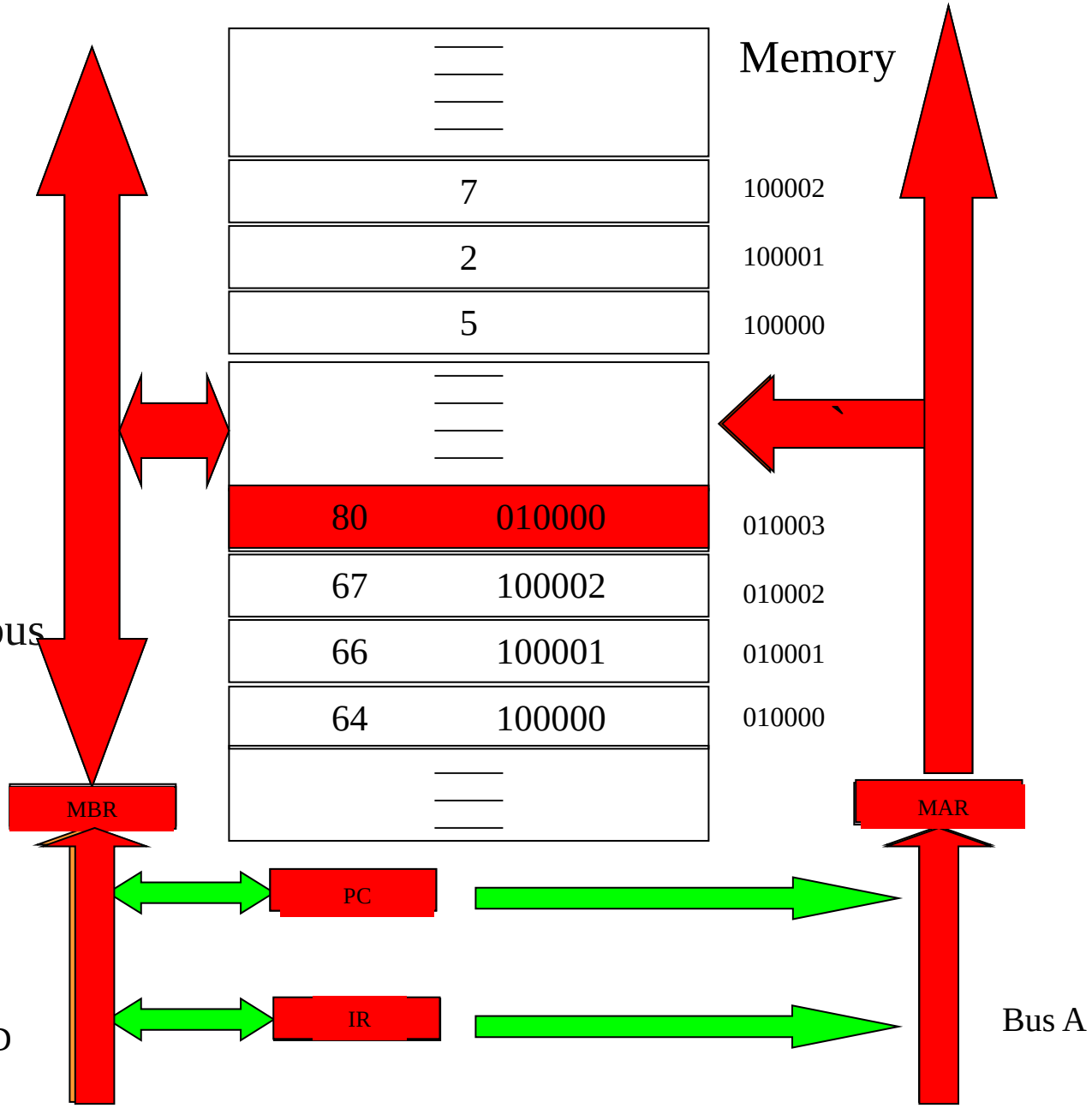
Set memory to read

Data bus → MBR

MBR → Bus D

Bus D → IR

Increment PC



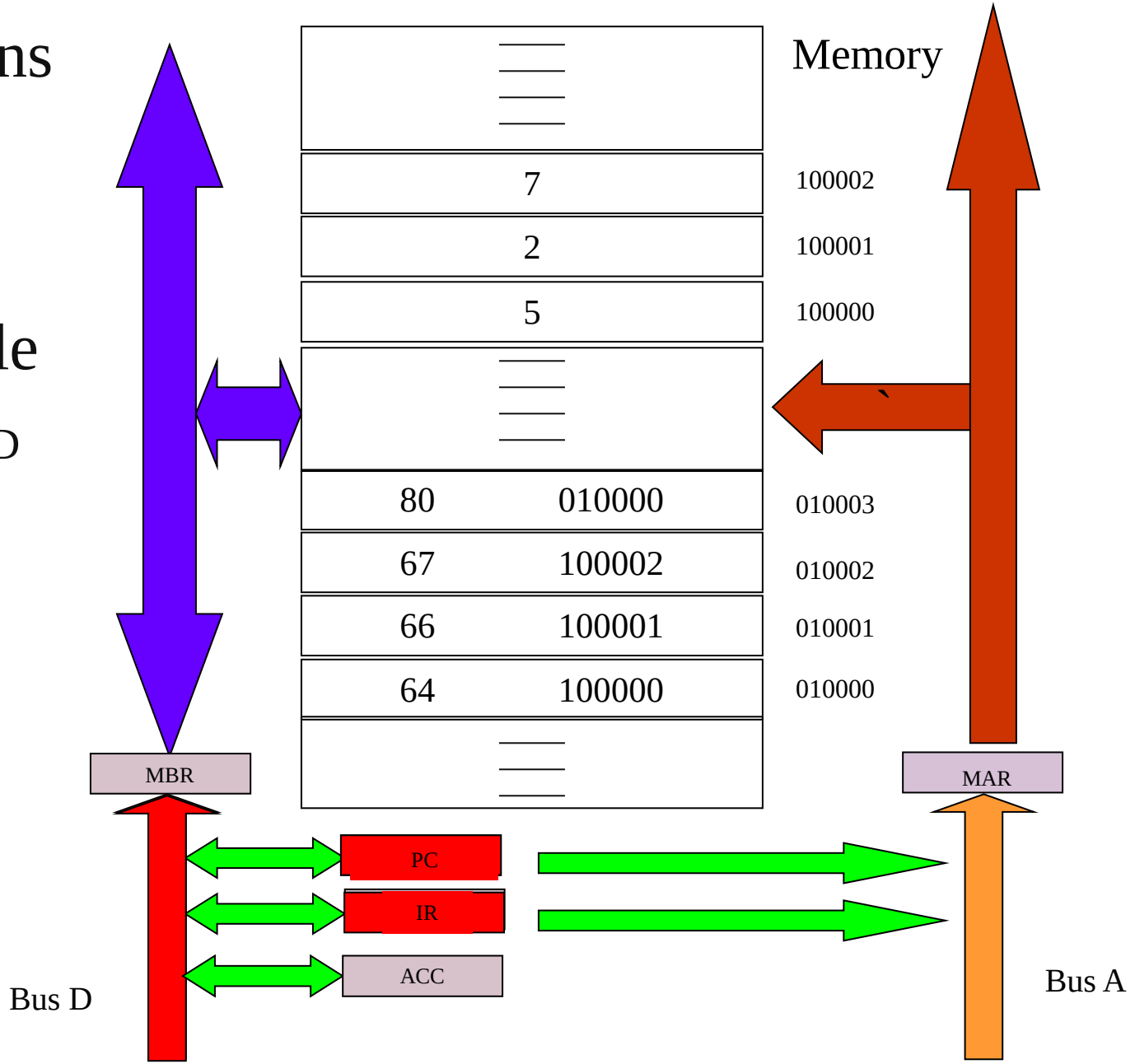
Micro operations for the

JUMP 010000

Execution cycle

IR → Bus D

Bus D → PC



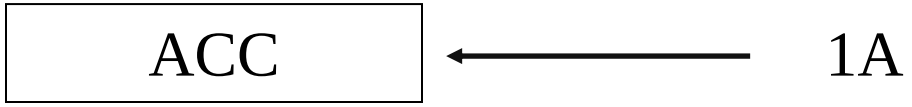
Addressing modes

Machine code instructions can have an operand part specifying the required operands. The way particular operands are referred to is known as the addressing mode. A particular CPU can support several addressing modes. Some of the examples are given below.

1. Immediate addressing mode

   In this mode the operand part of the machine code instruction itself contains the real data items.

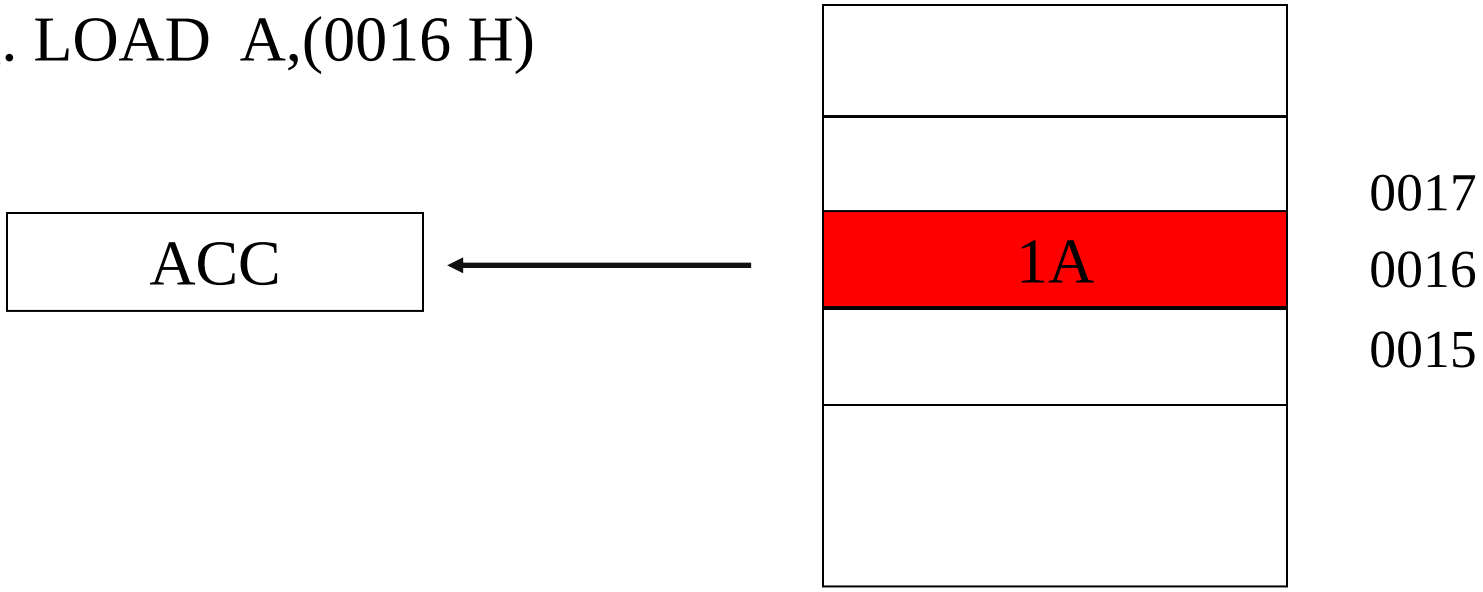
   Eg. LOAD A,1A H (H to denote Hexadecimal)



2. Direct Addressing mode.

in this mode the operand part gives the address of the memory location where the real operand data is stored

E.g. LOAD A,(0016 H)

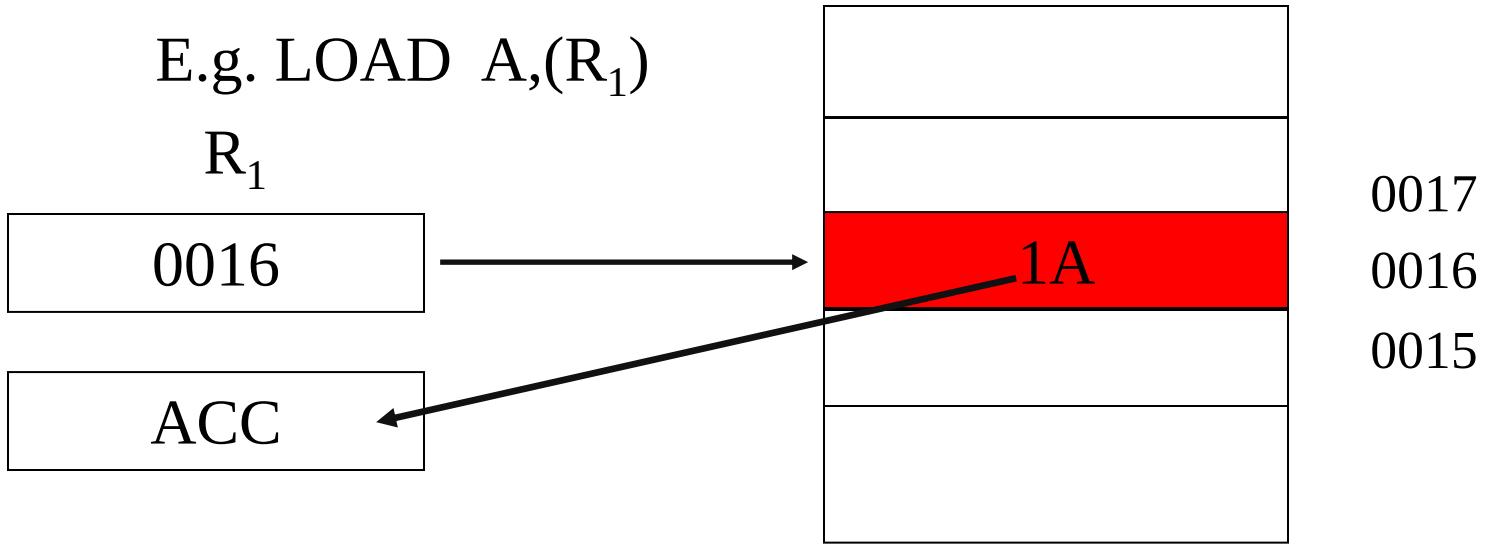


In this case Accumulator is loaded with the content of the memory location 0016H (ie by ‘1A’).

3. Indirect Addressing mode.

3.1 Register indirect addressing mode.

In this mode the operand part specifies an internal register which contains the address of a memory location where the real data item is stored.



In this case also Accumulator is loaded with the content of the memory location 0016H (ie by ‘1A’). But the address is obtained from R_1

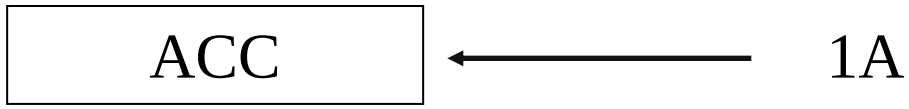
Addressing modes

Machine code instructions can have an operand part specifying the required operands. The way particular operands are referred to is known as the addressing mode. A particular CPU can support several addressing modes. Some of the examples are given below.

1. Immediate addressing mode

   In this mode the operand part of the machine code instruction itself contains the real data items.

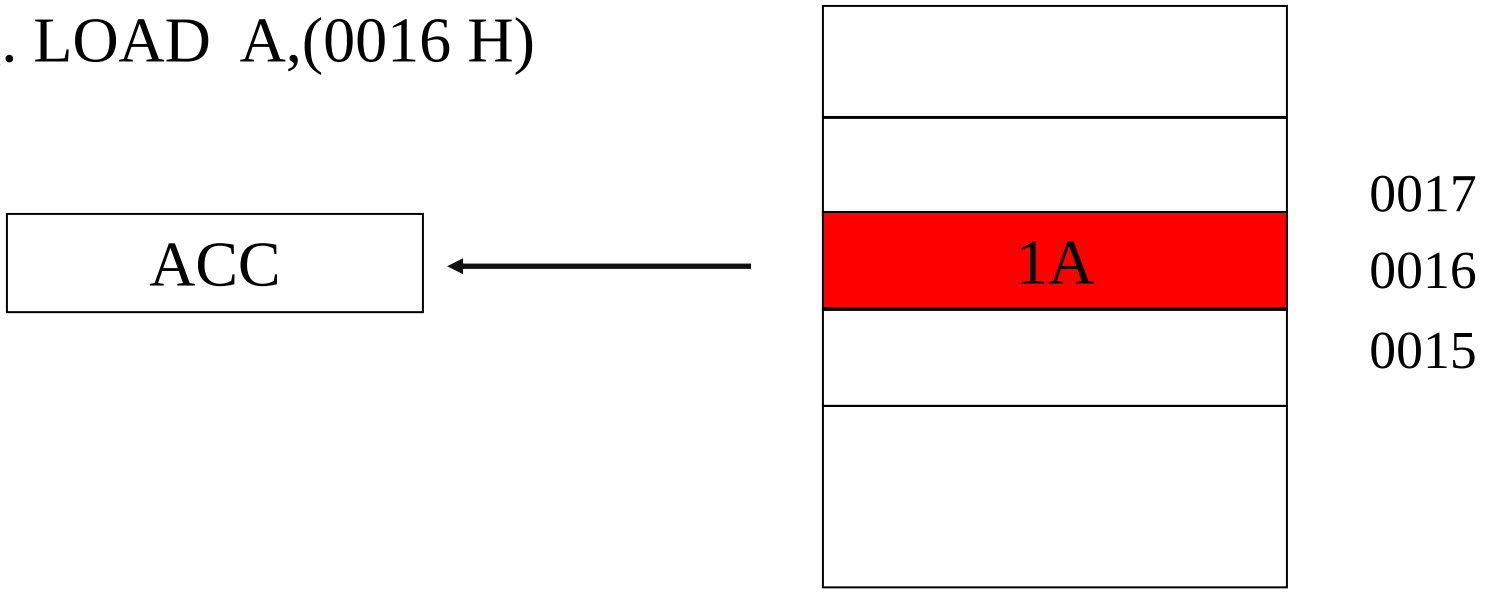
   Eg. LOAD A,1A H (H to denote Hexadecimal)



2. Direct Addressing mode.

in this mode the operand part gives the address of the memory location where the real operand data is stored

E.g. LOAD A,(0016 H)

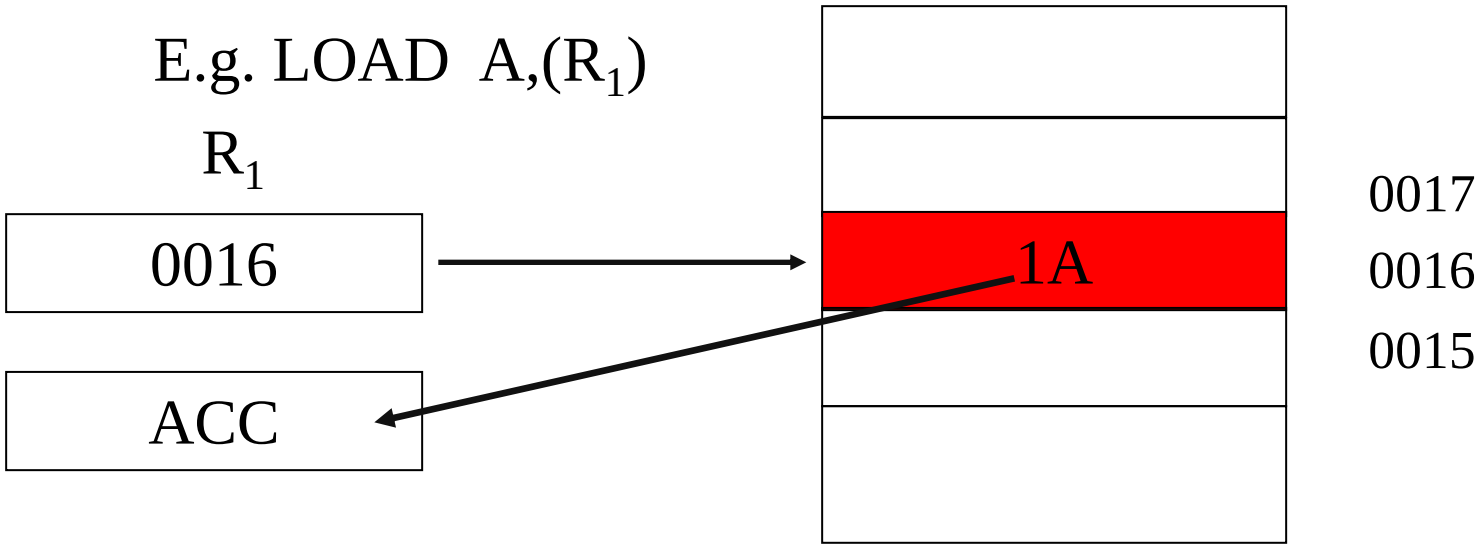


In this case Accumulator is loaded with the content of the memory location 0016H (ie by ‘1A’).

3. Indirect Addressing mode.

3.1 Register indirect addressing mode.

In this mode the operand part specifies an internal register which contains the address of a memory location where the real data item is stored.

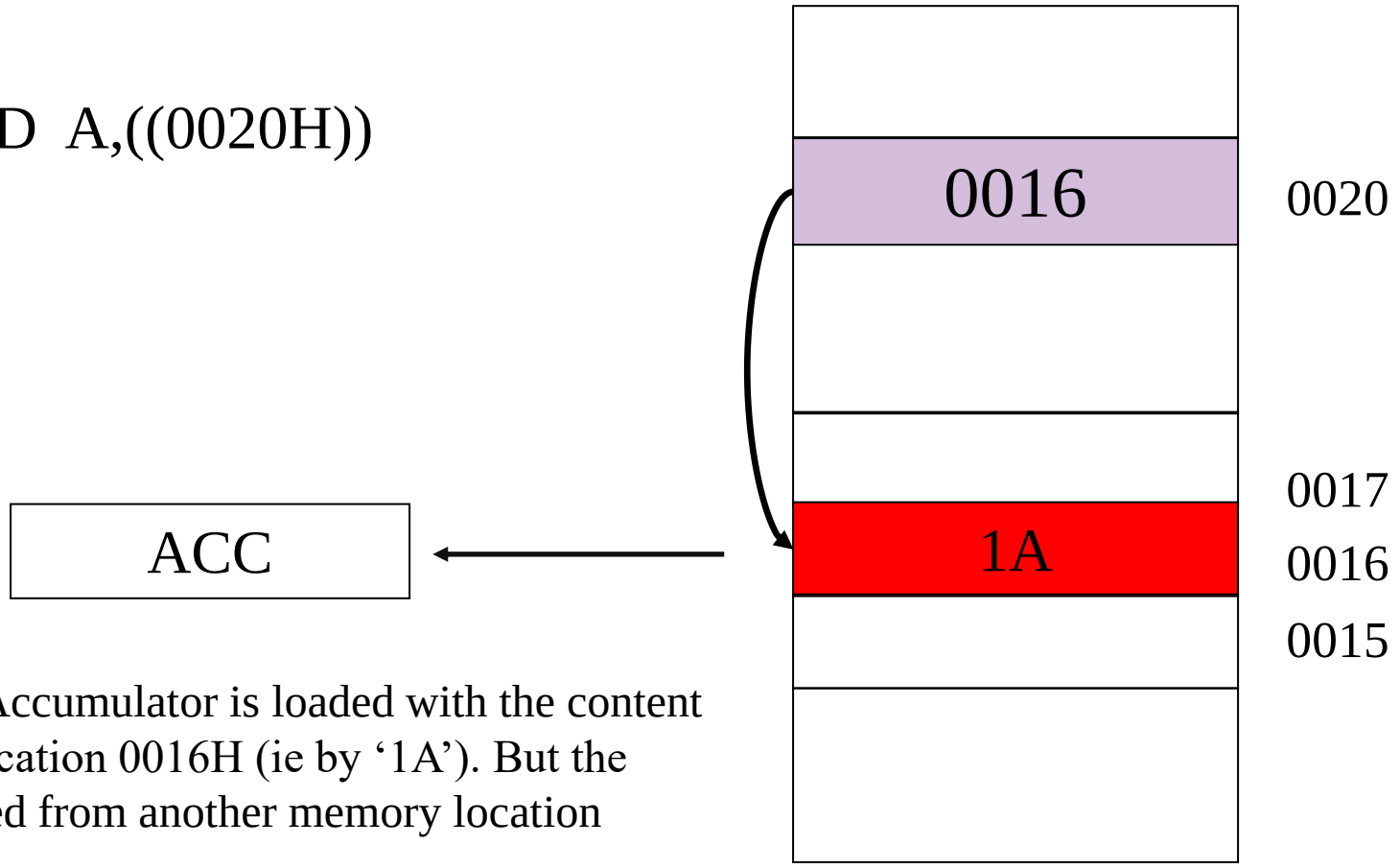


In this case also Accumulator is loaded with the content of the memory location 0016H (ie by ‘1A’). But the address is obtained from R_1

3.2 Memory indirect addressing mode.

In this mode, the operand part gives the address of a memory location which contains the address of the real data item in memory

E.g. LOAD A,((0020H))

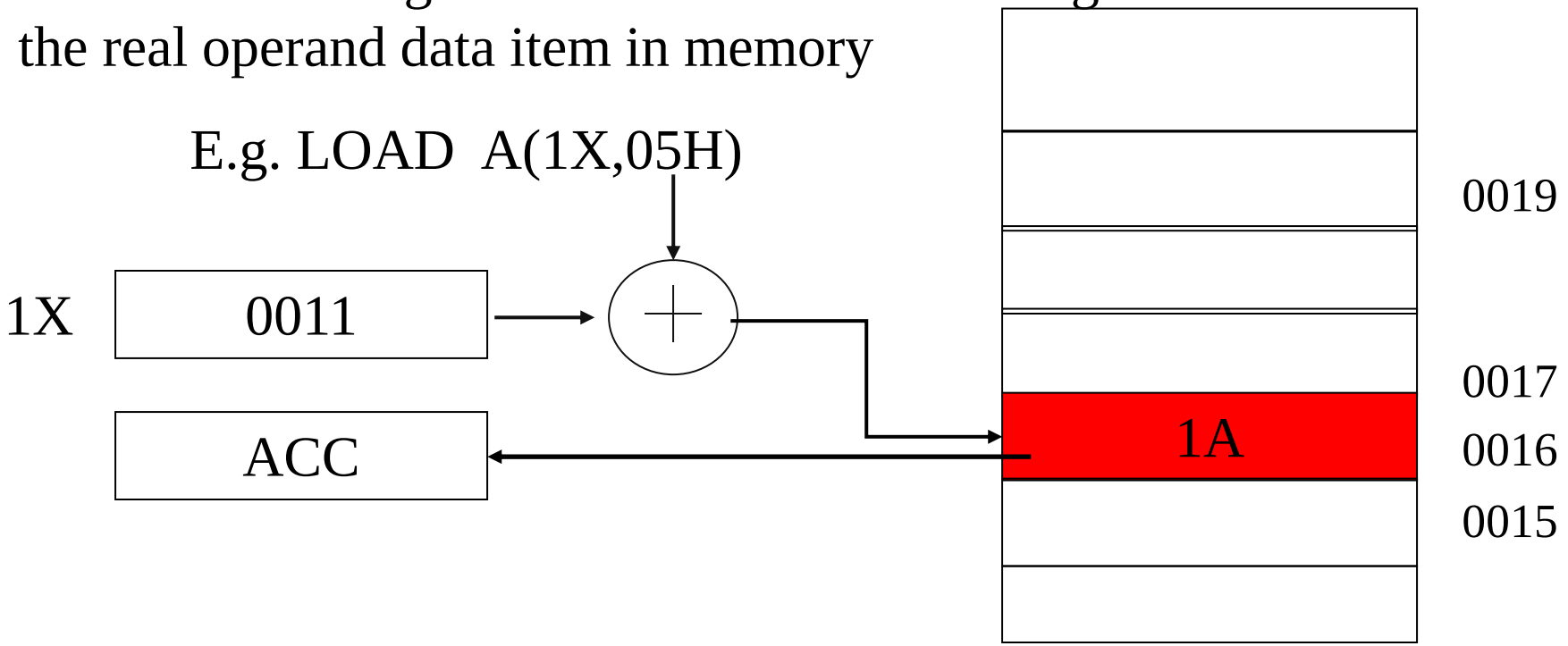


In this case also Accumulator is loaded with the content of the memory location 0016H (ie by ‘1A’). But the address is obtained from another memory location

4. Indexed Addressing mode.

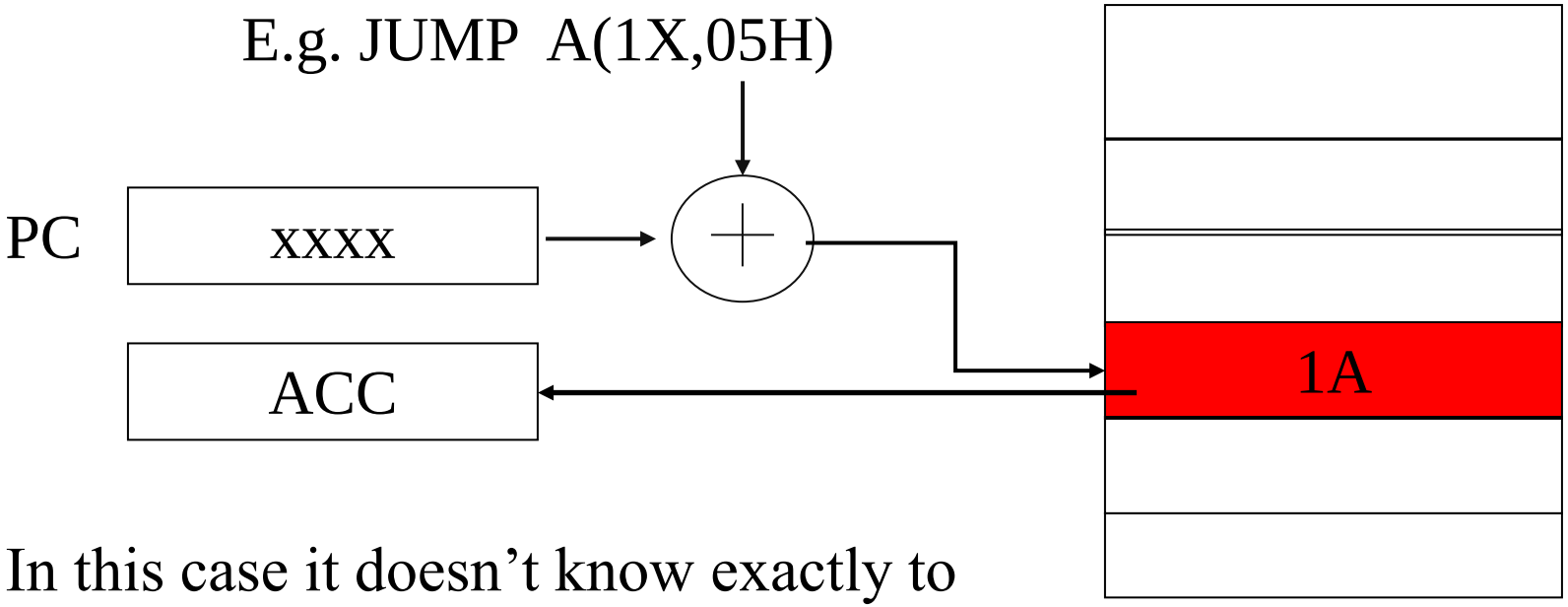
In this mode the operand part specifies a register called an indexed register. (if it is not specified then it is implied) the operand part contains a displacement or an off-set. The addition of the indexed register content and the off-set gives the address of the real operand data item in memory

E.g. LOAD A(1X,05H)



5. Relative Addressing mode.

Very much similar to the indexed addressing mode, but PC value is added to a given off-set (instead of indexed register value) to determine the addresses of the operand in memory

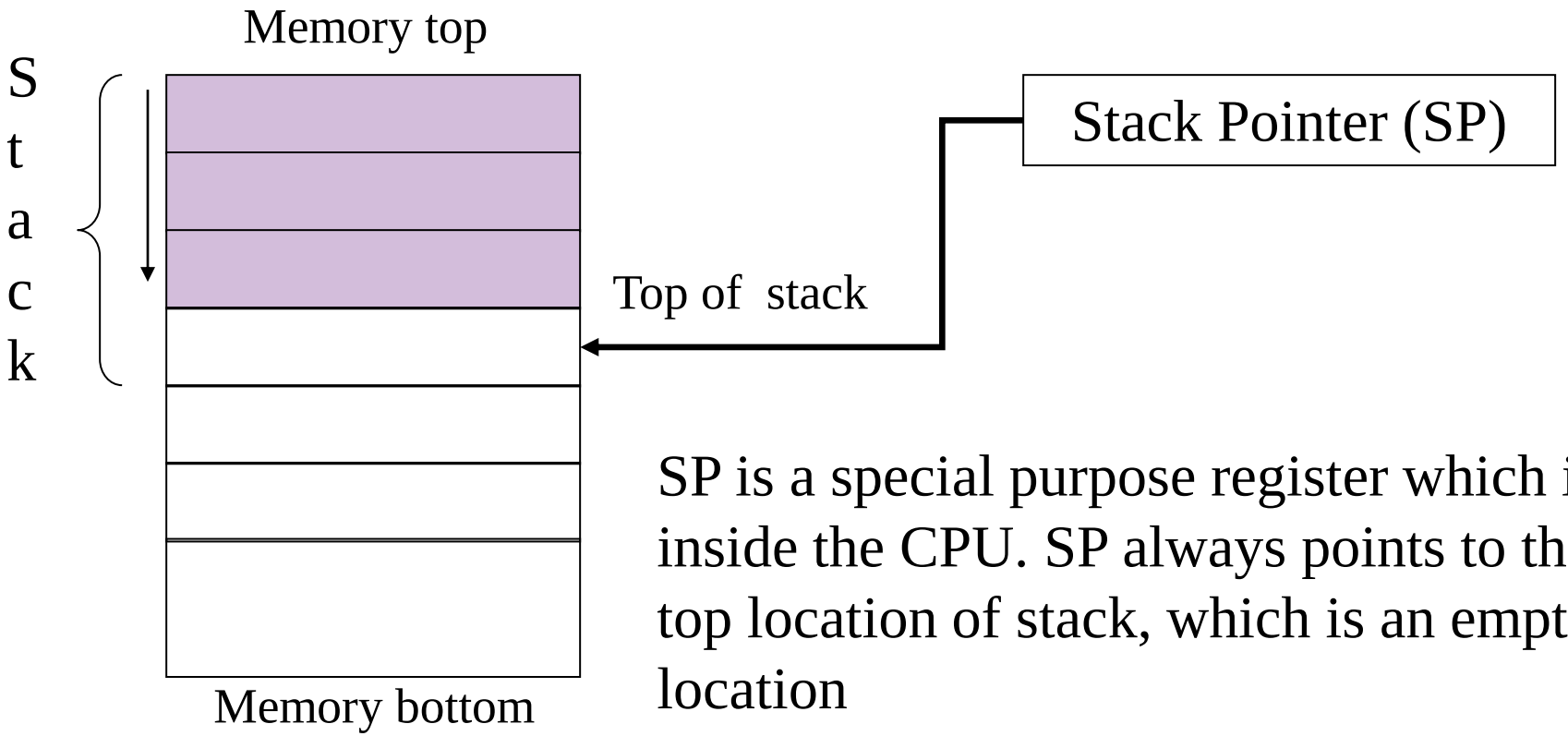


In this case it doesn’t know exactly to which location it should jump. But knows the length ,which it should jump.

Implementation of Stack

Stack is a reserved memory area used by the CPU for special purposes.

Ex. To save the return address of the CALL instruction.



SP is a special purpose register which is inside the CPU. SP always points to the top location of stack, which is an empty location

Two machine code instructions as **PUSH** & **POP** are usually provided to deal with the stack.

PUSH instruction this stores data items in the stack

E.g. PUSH A

This stores the accumulator contents in the stack and adjust the stack pointer to point to the next free location of the stack.

Fetch cycle is same as earlier

Micro operations			Controls to activate
SP	→	Bus A	(SP)outA
Bus A	→	MAR	
MAR	→	Address bus	
ACC	→	Bus D	
Bus D	→	MBR	
MBR	→	Data bus	
Set memory to write			
Decrement SP			

POP instruction this reads the data item from the stack to an internal CPU register

E.g. POP A

This reads the data item from the top of the stack to the accumulator.

Fetch cycle is same as earlier

Micro operations	Controls to activate
Increment SP	(SP)outA
SP → Bus A	
Bus A → MAR	
MAR → Address bus	
Set memory to read	
Data bus → MBR	
MBR → Bus D	
Bus D → ACC	