COMPUTER ORGANIZATION AND ARCHITECTURE

Fundamentals of Computer Architecture

Dr. Bore Gowda S B

Additional Professor

Dept. of ECE

MIT, Manipal

Introduction

- Why Computer Organization and Architecture?
  - Because you use it everyday
  - Because you will likely use it for the rest of your life
  - It focuses on the interface between hardware and software
  - It emphasizes the structure and behavior of the system
  - You can build your own computing device as per your requirement
  - Knowing how the hardware operates makes you a better coder
  - For a hardware engineer, designing the parts is a basic requirement

Introduction

- What is Computer Architecture?
- It refers to system attributes that are visible to the software programmer and have a direct influence on a program's logical execution, such as
  - the number of bits needed to represent distinct types of data,
  - the computer's instruction set,
  - the technique for addressing memory,
  - the method used for input and output, and so on.
- Computer architecture is a group of rules, orders, and processes that describe the functionality and performance of computer systems.
- Basically, it deals with the operational behaviour of computer systems.
- It involves decisions about the organization of the hardware, such as:
  - the instruction set architecture
  - the data path design
  - the control unit design.
- Computer architecture is concerned with optimizing the performance of a computer system and ensuring that it can execute instructions quickly and efficiently.

Introduction

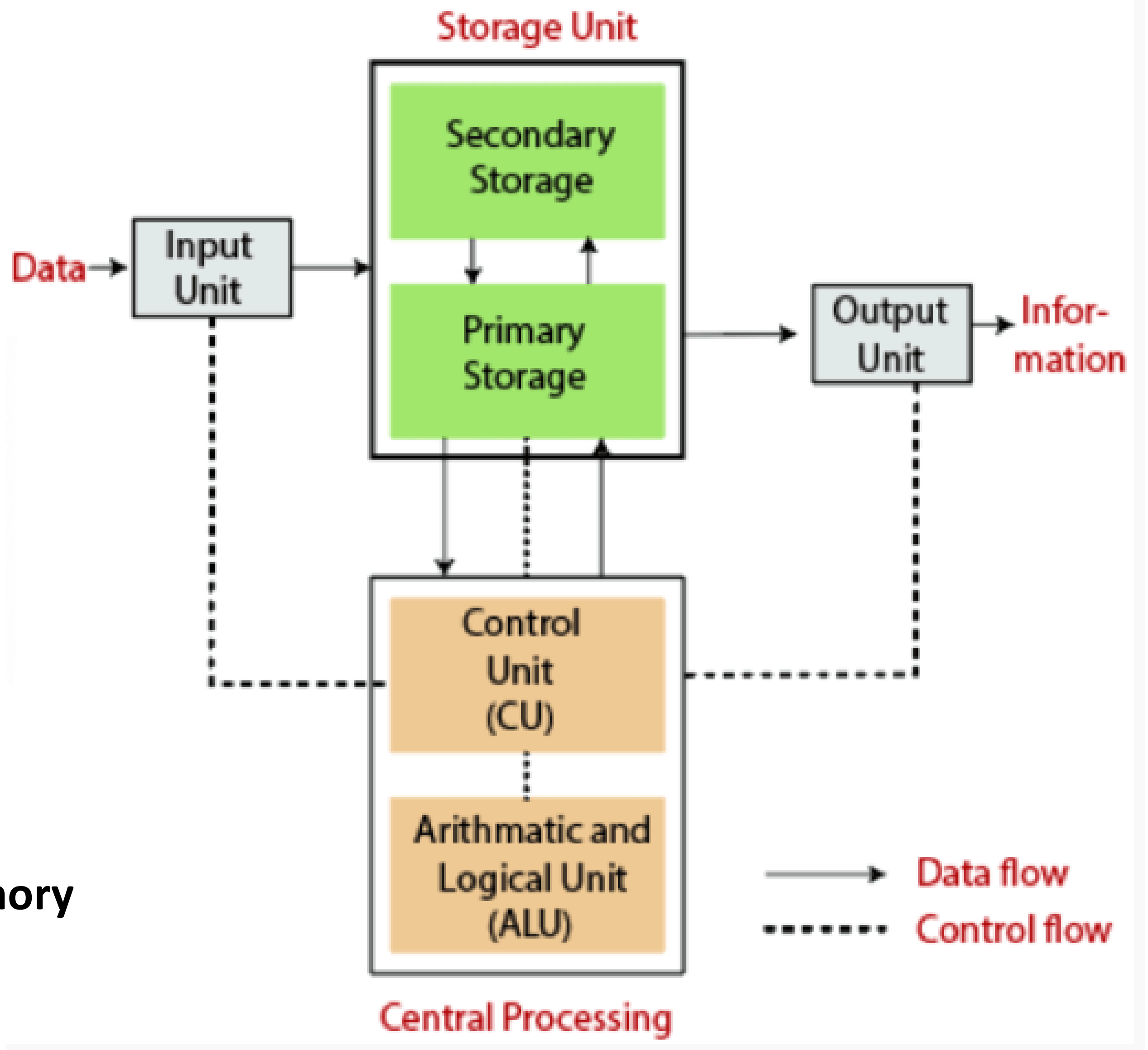
- What is Computer Organisation?
- Computer organisation is also known as Microarchitecture.
- What does microarchitecture include?
  - **Instruction implementation**: How instructions are carried out
  - **Data management**: How data is handled
  - **Control flow management**: How control flow is managed
  - **Techniques**: Techniques such as single-cycle, multicycle, pipelined, superscalar, and out-of-order processing
- It provides deep knowledge of ***functionality***, ***structuring***, ***internal working***, and ***implementation*** of a computer system.
- It refers to the operational units and their interconnections that realize the architectural specifications
- Examples of organizational attributes include those hardware details transparent to the programmer, such as
  - ✓ Control signals,
  - ✓ Interfaces between the computer and peripherals
  - ✓ Memory technology

Introduction

S.No	Computer Architecture	Computer Organization
1.	Explain what a computer does.	Explain how a computer actually does it.
2.	Majorly focus on the functional behaviour of computer systems.	Majorly focus on the structural relationship and deep knowledge of the internal working of a system.
3.	Deal with high-level design matters.	Deal with low-level design matters.
4.	It comes before computer organisation.	It comes after the architecture part.
5.	It is also called instruction set architecture.	It is also called microarchitecture.
6.	It covers logical functions, such as registers, data types, instruction sets, and addressing modes.	It covers physical units like peripherals, circuit designs, and adders.
7.	They coordinate between the hardware and software of the system.	They manage the portion of the network in a system.

Basic components of a computer

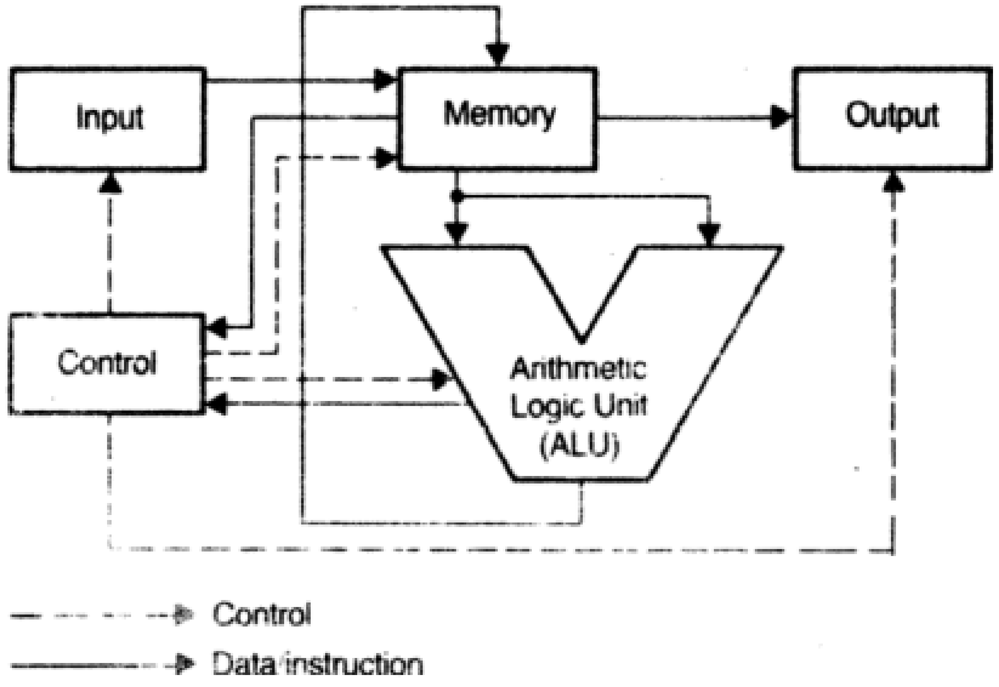
- To execute instructions, the computer performs the following steps:
  - the control unit reads or fetches an instruction from memory and decodes are translates it
  - for arthritic or logic type instructions, the control unit generates enables signals for the ALU to perform the required operations
  - for input and output(I/O) instructions, the control unit generates enable signals for the IO either to input data from or output data to external devices



- **CPU = program control unit + ALU + registers**
- **Memory = RAM + ROM + ERPROM etc+ secondary memory**

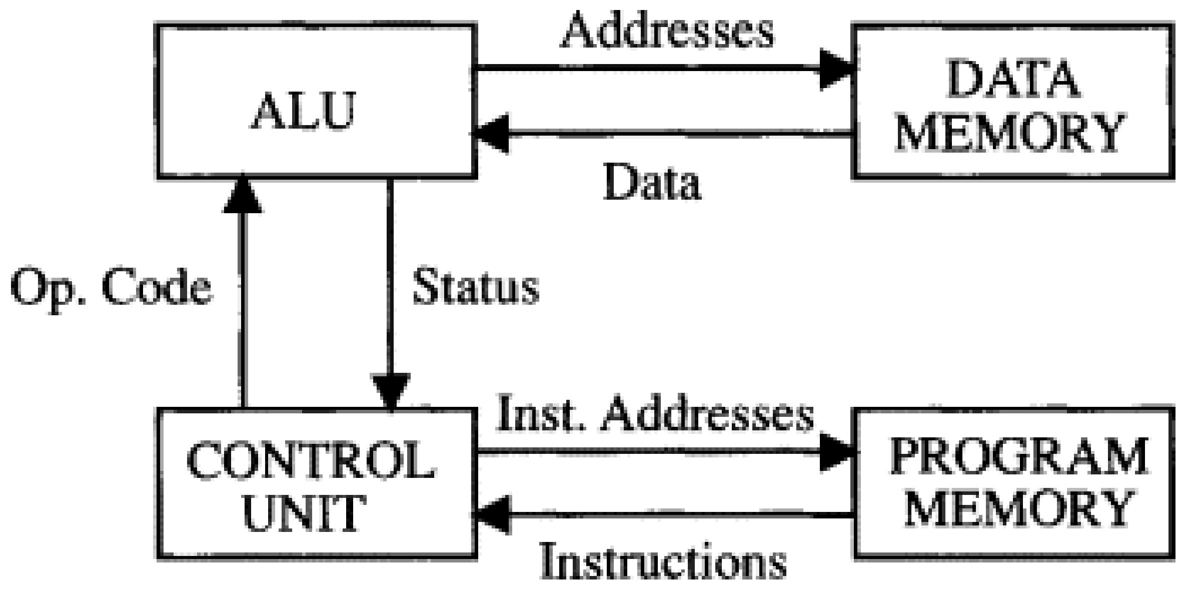
Von-Neumann computer architecture

- Von-Neumann proposed his computer architecture design in 1945 which was later known as Von-Neumann Architecture.
- The major components of it are:
  - ✓ a Control Unit
  - ✓ Arithmetic and Logical Memory Unit (ALU),
  - ✓ Registers and Inputs/Outputs.
- It is based on the ***stored-program computer concept***, where ***instruction data and program data*** are stored in the ***same memory***.
- This design is still used in most computers produced today.
- A Von Neumann-based computer:
  - ✓ Uses a single processor
  - ✓ Uses one memory for both instructions and data.
  - ✓ Executes programs following the fetch-decode-execute cycle



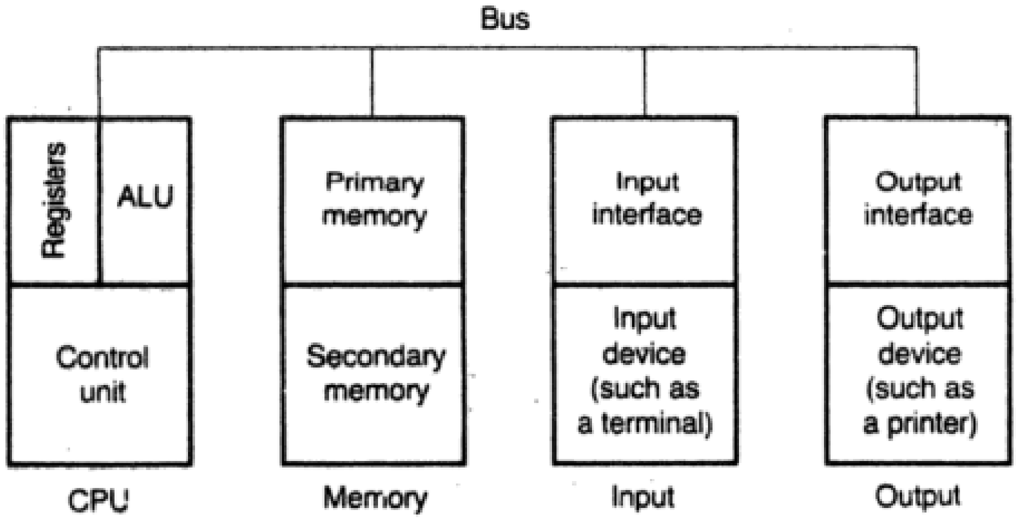
Harvard computer Architecture

- Harvard Architecture is the computer architecture that contains separate storage and separate buses (signal path) for instruction and data.
- It was basically developed to overcome the bottleneck of Von Neumann's Architecture.
- The main advantage of having separate buses for instruction and data is that the CPU can access instructions and read/write data at the same time.



Organization of Stored Program Computer System

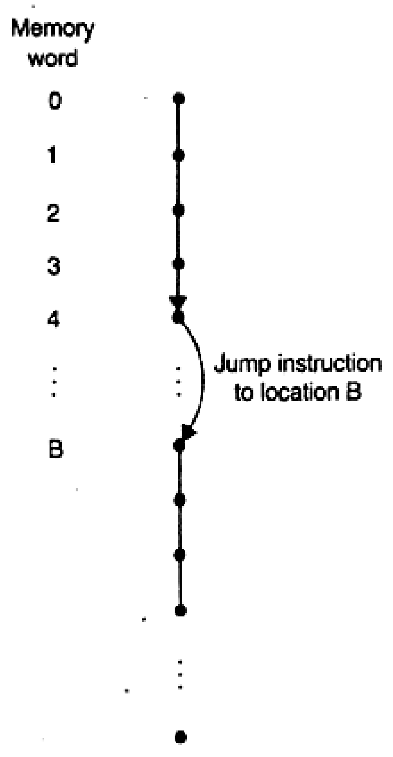
- Computer: device capable of processing information by executing a program consisting of several sequence of instructions stored in memory
- This kind is called a stored program computer, because the instructions and data are held in the same memory
- Instruction execution takes place within the CPU and is controlled by the control unit
- **Control Unit**
- Control unit has to perform the following task to carry out the instruction execution process:
  - ✓ instruction interpretation
    - Reads an instruction
    - recognizes the instruction operation
    - retrieves the required data
    - Perform desired operation by activating the ALU
  - ✓ instruction sequencing
    - Determines the address of the next instruction
    - programs are executed in a strictly sequential fashion
    - the next instruction to be executed stored in the consecutive memory locations



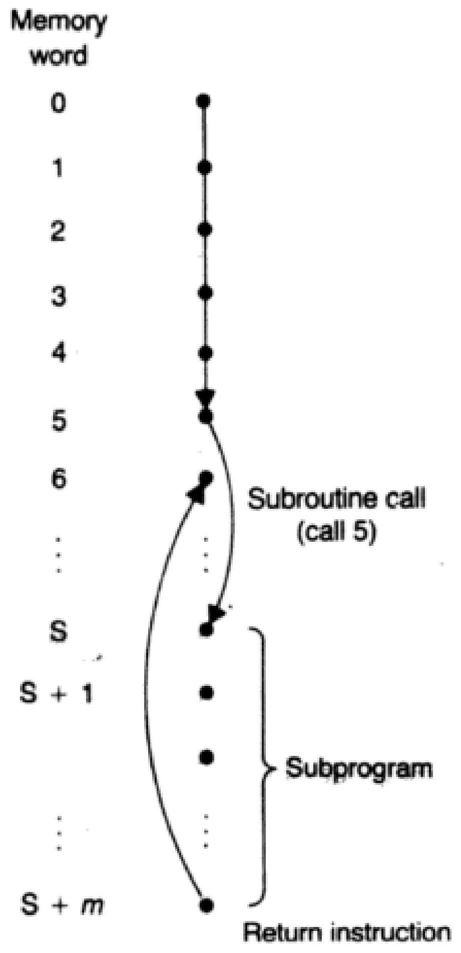
Organization of Stored Program Computer System

Typical control Flow sequences -

- Branch instruction
- Subroutine call



a. Branch Instruction



b. Subroutine Call

Organization of Stored Program Computer System

- **CPU**
  - includes set of high speed registers
  - registers may hold data, temporary results or memory address when a computation is in progress
- Common CPU registers
  - **Program counter(PC):** holds address of the instruction to be executed
  - **Instruction register(IR):** holds the instruction currently being executed
  - **Effective address register(EAR):** holds the address of the data to be retrieved from the memory
- Refer to as dedicated registers because they are available for the exclusive use of the control unit and cannot be accessed by a user program
- **Memory Unit**
- Divided into two sections: Primary memory and Secondary memory
- Primary memory
- designed using semiconductor technology
- it is of two types: read write memory and read only memory

Organization of Stored Program Computer System

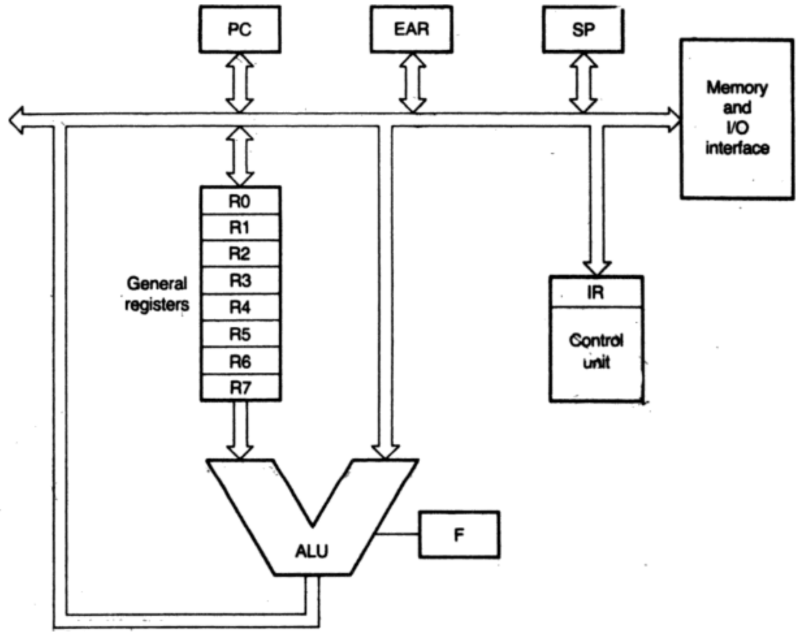
- **Memory Unit**
- Divided into two sections: Primary memory and Secondary memory
- **Primary memory**
  - designed using semiconductor technology
  - It is of two types: read write memory(RWM) and read only memory(ROM)
  - RWM AND ROM are random access memory
- Secondary memory
  - Used to store large amount of data
  - **Examples:** magnetic disks and tapes etc.
- **I/O device :** allow user to communicate with the computer
  - To facilitate communication between the CPU and an IO device we need interface circuitry between them
- System bus
  - The components of a stored program computer systems are connected to each other by a bus
  - three types of buses: ***address bus, data bus and control bus***

Computer structure(Organization)

- Classified into following three groups
  - General register-based processor
  - Accumulator based processor
  - Stack based processor

Computer structure(Organization)

- **General register-based processor**
  - 8 general registers : R0 through R7 can hold
    - Data
    - Memory addresses
    - Result of arithmetic or logic operation
  - **Program counter:** Holds address of the next instruction executed
  - **Effective address register**: Holds address of data to be retrieved from memory
  - **Instruction register:** Holds instruction currently being executed
  - **Stack Pointer:** Address of the top element of stack
  - **F flag register:** holds carry flag or zero flag
  - **Support two- and three-address instructions**



two-address instructions

INSTRUCTION	OPERATION
MOV x, y	y ← (x)
ADD x, y	y ← (x) + (y)
SUB y, x	x ← (x) − (y)
MUL y, x	x ← (x) * (y)
DIV y, x	x ← (x) / (y)

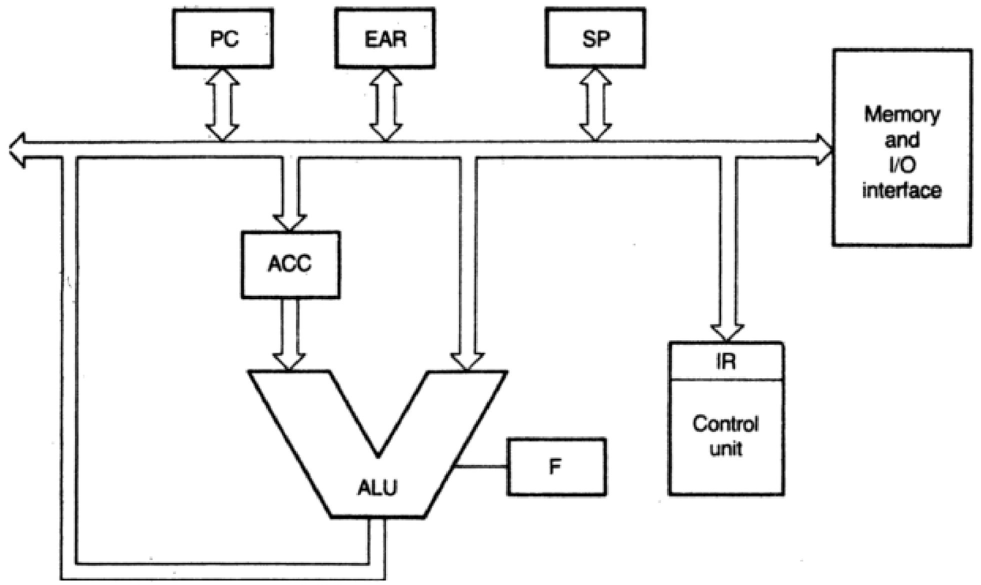
three-address instructions

INSTRUCTION	OPERATION
ADD x, y, z	z ← (x) + (y)
SUB y, x, z	z ← (x) − (y)
MUL x, y, z	z ← (x) * (y)
DIV y, x, z	z ← (x) / (y)

Computer structure(Organization)

Accumulator based processor

- One of the operands is assumed to be held in the accumulator register (ACC) for arithmetic or logic operation
- results of all arithmetic and logic operations are routed to the ACC
- one address instructions are very predominant in this organization



INSTRUCTION		OPERATION
LDA	x	Acc ← (x)
STA	x	x ← (Acc)
ADD	x	Acc ← (Acc) + (x)
SUB	x	Acc ← (Acc) − (x)
MUL	x	Acc ← (Acc) * (x)
DIV	x	Acc ← (Acc) / (x)

Implementation of assignment statement D = A + B*C using different addressing instruction format

THREE ADDRESS

MUL B,C,D ; D←(B)*(C)
ADD A,D,D ; D←(A)+(D)

TWO ADDRESS

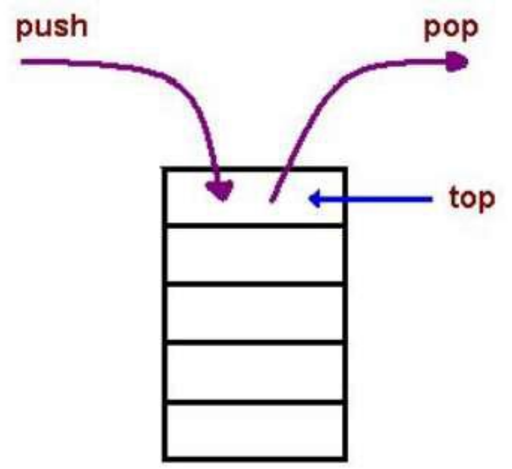
MOV B,D ; D←(B)
MUL C,D ; D←(C)*(D)
ADD A,D ; D←(A)+(D)

ONE ADDRESS

LDA B ; Acc←(B)
MUL C ; Acc←(Acc)*(C)
ADD A ; Acc←(Acc)+(A)
STA D ; D←(Acc)

Computer structure(Organization)

- Stack based processor
  - Stack
    - A stack is an ordered linear list in which all insertions and deletions are made at one end, called top.
    - It uses Last In First Out (LIFO) principle
    - A register is used to store the address of the topmost element of the stack which is known as Stack Pointer(SP)
    - The main two operations that are performed on the operands of the stack are: **Push and Pop.**
    - These two operations are performed from one end only.
    - **Push Operation:** The operation of inserting an item onto a stack
    - **Pop Operation:** The operation of deleting an item onto a stack



Introduction

- **Stack based processor**
  - **Stack**
  - Applications:
    - ✓ Evaluation of mathematical expressions using Reverse Polish Notation.
    - ✓ To implement subroutine calls and returns
    - ✓ to pass parameters from a main program to a subroutine
    - ✓ to handle interrupts
    - ✓ To reverse a word. A given word is pushed to stack-letter by letter-and then popped out letters from the stack.
    - ✓ An undo mechanism in text editors; this operation is accomplished by keeping all text changes in a stack.
    - ✓ **Backtracking:** this is a process when it is required to access the most recent data element in a series of elements.
    - ✓ Language processing
    - ✓ space for parameters and local variables is created internally using a stack.
    - ✓ The compiler's syntax check for matching braces is implemented by using stack.

Computer structure(Organization)

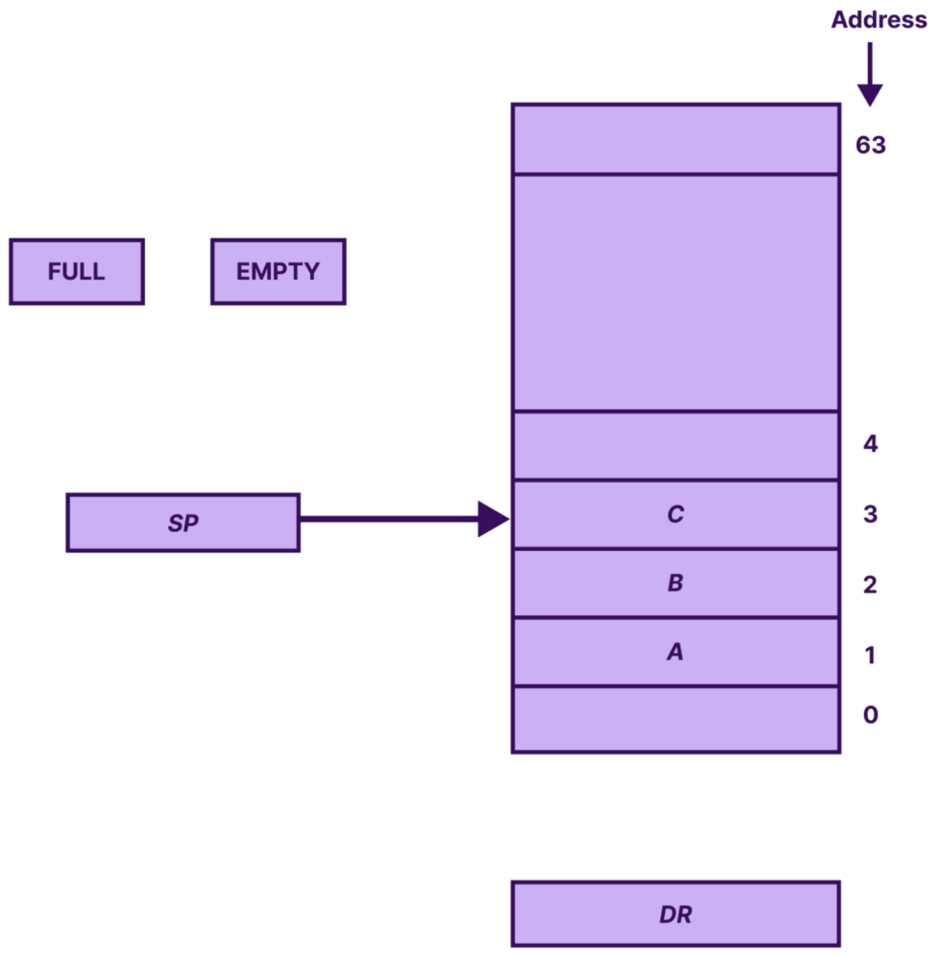
- **Stack based processor**
  - **Stack**
    - There are two types of stack organization which are used in the computer hardware:
    - **Register stack:** It is built using register
    - **Memory stack:** It is logical part of memory allocated as stack. The logically partitioned part of RAM is used to implement stack.

Computer structure(Organization)

Stack based processor

Register stack

- The stack can be arranged as a set of memory words or registers.
- Each time an element is added (pushed), the stack pointer is incremented.
- When an element is removed (popped), the pointer is decremented.
- Consider a 64-word register stack arranged as displayed in the figure.
- The stack pointer register includes a binary number, which is the address of the element present at the top of the stack.
- **Stack Pointer (SP):** Points to the top of the stack.
- **Data Register (DR):** Holds data being transferred.
- **Full:** The stack is at maximum capacity and cannot hold more data (stack overflow).
- **Empty:** The stack contains no data, and no items can be popped (stack underflow).



64 Word Stack

Computer structure(Organization)

Stack based processor

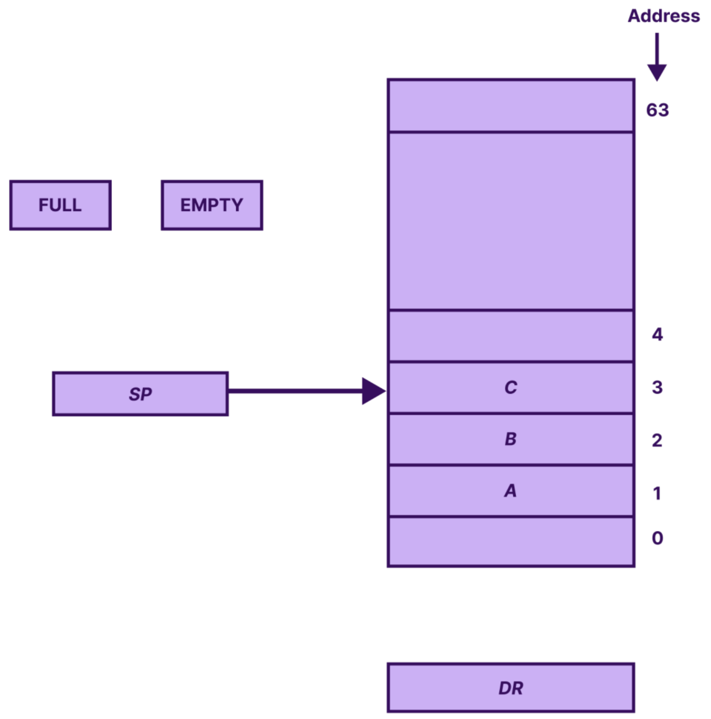
Register stack

The PUSH operation is implemented with the following sequence of microoperations:

SP ← SP+1	Increment stack pointer
M[SP] ← DR	Write item on top of the stack
If (SP=0) then (FULL←1)	Check if stack is full
EMTY←0	Mark the stack not empty

A new item is deleted from the stack if the stack is not empty (if EMTY=0). The POP operation consists of following sequence of microoperations:

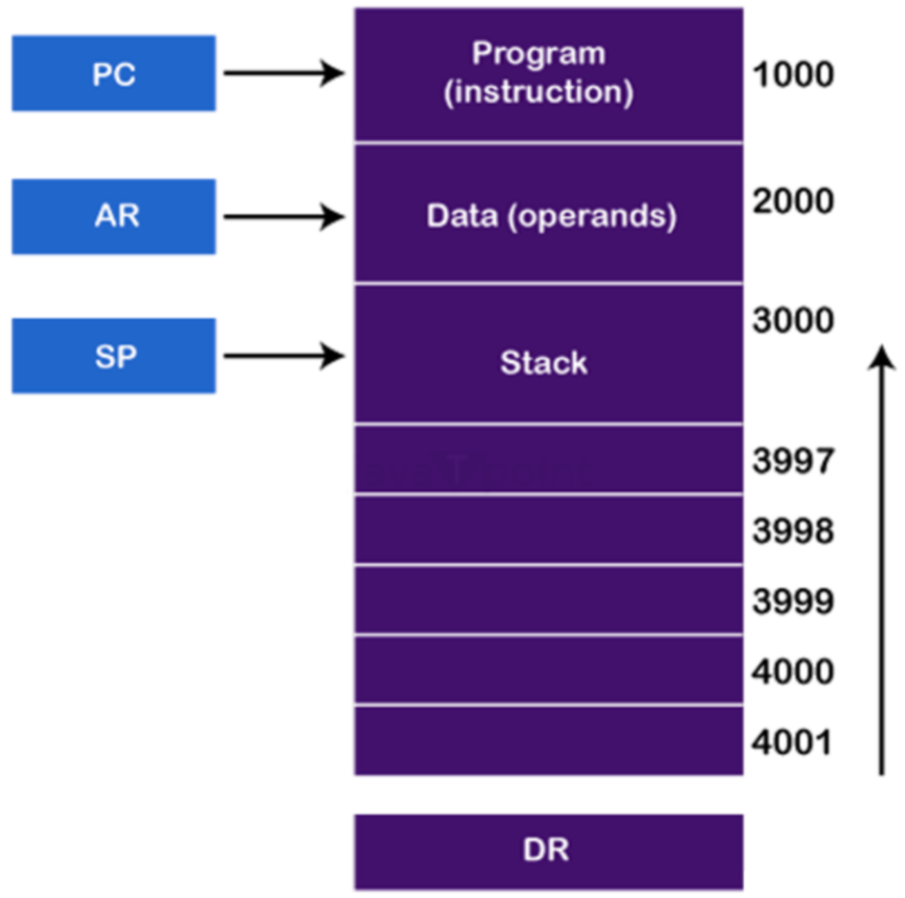
DR←M[SP]	Read item from the top of the stack
SP←SP-1	Decrement stack pointer
If(SP=0) then (EMTY←1)	Check if stack is empty
FULL ←0	Mark the stack not full



64 Word Stack

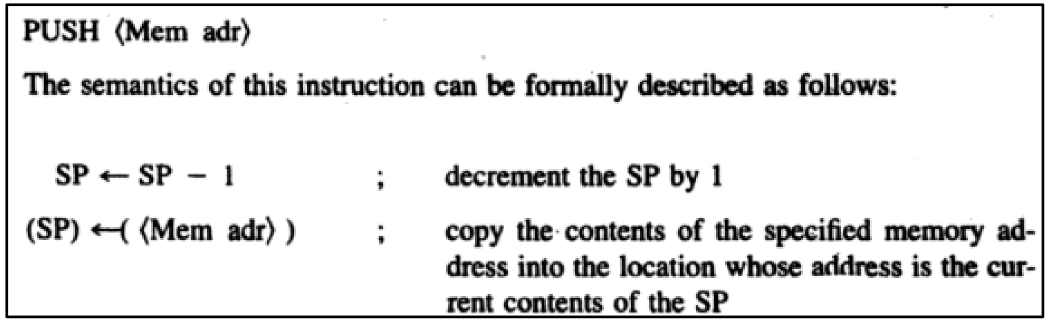
Computer structure(Organization)

- Stack based processor
  - Memory stack
    - can be implemented by configuring some portion of the read write memory
    - one of the CPU registers called the ***stack pointer (SP)*** is used to hold the address of the most recently entered item onto the stack
    - The stack pointer register keeps track of the top of the stack, and elements are inserted or removed using this pointer.
    - The stack typically grows downwards, with new elements being pushed at lower memory addresses.
    - **Program Counter (PC):** Holds the address of the next instruction.
    - **Address Register (AR):** Stores memory addresses.
    - **Stack Pointer (SP):** Points to the top of the stack.
    - **Data Register (DR):** Holds data being transferred.



Computer structure(Organization)

- Stack based processor
  - Memory stack
    - **PUSH – writing onto the stack**
    - Item can be inserted into the stack by executing PUSH instruction

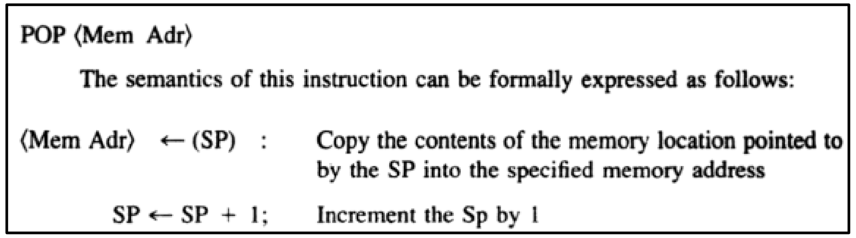


PUSH ⟨Mem adr⟩

The semantics of this instruction can be formally described as follows:

SP ← SP − 1	;	decrement the SP by 1
(SP) ←(⟨Mem adr⟩)	;	copy the contents of the specified memory address into the location whose address is the current contents of the SP

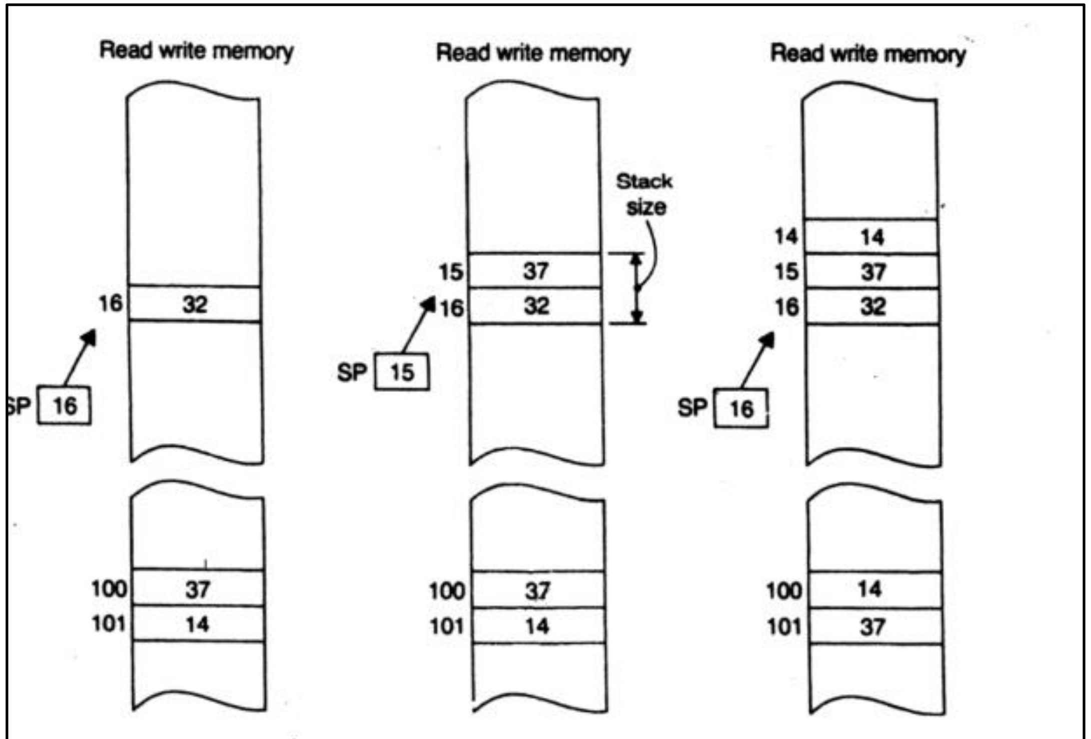
    - **POP – reading from the stack**
    - Item can be removed from the stack by executing PUSH instruction



POP ⟨Mem Adr⟩

The semantics of this instruction can be formally expressed as follows:

⟨Mem Adr⟩ ← (SP)	:	Copy the contents of the memory location pointed to by the SP into the specified memory address
SP ← SP + 1;		Increment the Sp by 1



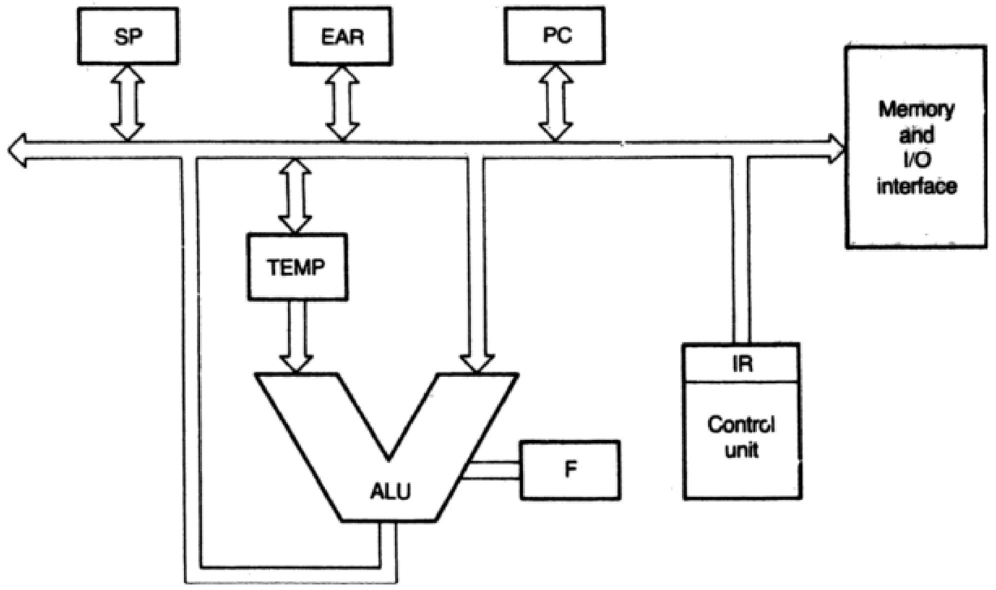
a. Initial Configuration
b. Configuration that Results after Executing the Instruction PUSH 100
c. Configuration that Results after Executing the Sequence of Instructions PUSH 100 PUSH 101 POP 100 POP 101

Computer structure(Organization)

Stack based processor/Stack machine

Stack

- In the event of binary arithmetic or logic operation, the first operand will be removed from the stack and held in register TEMP
- the second operand will be directly routed from the stack to the right input of the ALU
- the result of the operation is stored in the top of the stack



Some Typical Stack Machine Instructions

INSTRUCTION	*OPERATION*
PUSH X	(TOS) ← (X)
POP Z	Z ←—— ((TOS))
ADD	(NOS) ← ((TOS)) + ((NOS))
SUB	(NOS) ← ((TOS)) − ((NOS))
MUL	(NOS) ← ((TOS)) * ((NOS))
DIV	(NOS) ← ((TOS)) / ((NOS))

Note: TOS = top of stack
NOS = next on the stack

Implementation of assignment statement D = A + B*C using stack machine instructions

```
PUSH A
PUSH B
PUSH C
MUL     ; calculate B*C
ADD     ; Add A to B*C
POP D   ; save result.
```

Arithmetic Expression Evaluation

- Expression notation is a way to write mathematical expressions using operators and operands.
- The three most common types of expression notation are infix, prefix, and postfix.
- **Infix notation**
  - ✓ The most common way to write mathematical expressions
  - ✓ Operators are written between operands
  - ✓ **For example:** A + B
  - ✓ Parentheses are used to specify the order of operations
  - ✓ This is the most natural way to write math, but it can be ambiguous
- **Prefix notation**
  - ✓ Also known as **Polish notation (PN)**
  - ✓ Operators are written before operands
  - ✓ **For example:** + A B
  - ✓ Parentheses are not needed because the order of operations is clear
  - ✓ This notation is easy for computers to read, but it can be unintuitive for humans
- **Postfix notation**
  - ✓ Also known as **Reverse Polish notation (RPN)**
  - ✓ Operators are written after operands
  - ✓ **For example:** A B +
  - ✓ Parentheses are not needed because the order of operations is clear
  - ✓ This notation is useful for analyzing and manipulating mathematical formulas

Infix	Prefix	Postfix
A + B * C + D	+ + A * B C D	A B C * + D +
(A + B) * (C + D)	* + A B + C D	A B + C D + *
A * B + C * D	+ * A B * C D	A B * C D * +
A + B + C + D	+ + + A B C D	A B + C + D +

Arithmetic Expression Evaluation

- **Convert the following expressions into RPN:**
  1. X = (A + B + C) / (D –E * F)
  2. X = A * B + C * D + E * F
  3. X = (8 + 2 * 5) / (1 + 3 * 2 –4)
  4. A = (B + C) * D –E
  5. [(A + B) * C + D] / (E + F + G)

Evaluation of Postfix expression using stack

❑**The evaluation rule/algorithm for the postfix expression is stated as:**

1. Read the expression from left to right.
2. If it is an operand then push the element into the stack.
3. If the element is an operator except NOT operator, pop the two operands from the stack and evaluate them with the read operator and push back the result of the evaluation into the stack.
4. If it is the NOT operator then pop one operand from the stack and then evaluate it and push back the result of the evaluation into the stack.
5. Repeat it till the end of stack.

Introduction

- Evaluate the postfix expression: **10 8 4 + * 6 3 / -**

Sr. no.	Expression	Stack	Operations
0	10	10	Push onto stack
1	8	10 8	Push onto stack
2	4	10 8 4	Push onto stack
3	+	10 12	Pop top two elements i.e. 8 and 4 from the stack, add them, then push the result onto stack
4	*	120	Pop top two elements i.e. 10 and 12 from the stack, multiply them, then push the result onto stack
5	6	120 6	Push onto stack
6	3	120 6 3	Push onto stack
7	/	120 2	Pop top two elements i.e. 6 and 3 from the stack, divide them, then push the result onto stack
8	-	118	Pop top two elements i.e. 120 and 2 from the stack, substract them, then push the result onto stack