काशी हिन्दू विश्वविद्यालय

BANARAS HINDU UNIVERSITY

Functioning of CPU

Guided by, ***Dr. Sarvesh Pandey***

Presented by,

Anjali Kumari (20229CMP001)
Anjali Verma (20229PHY002)
Anshika (20229PHY003)
Jaya Chaudhary (20229PHY006)
Suchita Srivastava (20229MAT006)

Contents:

- Introduction
- Instruction Formats
- Instruction types
- Addressing Modes
- General Register Organization
- Control Words
- Stack Organization

- Multicore architecture
- Multiprocessor
- Multicomputer
- Reverse Polish Notation

Introduction—

A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

A CPU comprises three major components, they are:
- Register Set
- ALU
- Control Unit (CU)

Register Set

The register set differs from one system to another. The register set comprises many registers which include general purpose registers and special purpose registers. The general purpose registers do not perform any specific function. They store the temporary data that is required by a program.

The special purpose registers perform specific functions for the CPU.
Example: Instruction Register (IR) is a special purpose register that stores the instruction that is currently being executed.

ALU

The ALU performs all the arithmetic, logical, and shift operations by providing necessary circuitry that supports these computations.

Control Unit

The control unit fetches the instructions from the main memory, decodes the instructions, and then executes it.

Instruction Formats—

An Instruction is composed of Operation Code (OPCODE) and Operand.

The first part of the instruction specifies the task to be perform called OPCODE and second part of the instruction is data to be operand on, and it is called Operand. Now, operand of instruction can be in various forms such that 8-bit or 16-bit data, 8-bit or 16-bit address, register or memory address etc. So, instruction format may be different in different instructions. **An instruction format is the layout of the bits of an instruction, in terms of constituent fields.**

Therefore, generally an instruction is of the following form-

OPCODE-FIELD	Address-Field

Generally, based on the address field, four common instruction formats are Available and they are-

⇨ Zero- Address Instruction

OPCODE- FIELD	NIL

For example,
PUSH A i.e. , insert the content of A into the stack

One- Address Instructions

For example,
ADD B i.e., Accumulator ← Accumulator + B

Two- Address Instructions

For example,
ADD A, B i.e., A ← A+B

⇨ Three- Address Instructions

OPCODE- Field	Address 1	Address 2	Address 3

For example,
ADD A, B, C i.e., A ← B + C

Instruction Length: Instruction length depends upon the memory size, Memory organization, bus structure, processor complexity and processor speed.
Some of the factors for determining number of bits for addressing are–
~ Number of addressing modes
~ Number of operands
~Number of Register in the register set
~Address range of referencing memory

To illustrate the influence of the number of address on computer programs, we will evaluate the arithmetic statement X=(A+B)*(C+D) using Zero, one, two, or three address Instructions.

1. Three-Address Instructions:

```
ADD   R1,A,B      R1< M[A] + M[B]
ADD   R2, C, D    R2< M[C] + M[D]
MUL   X, R1,R2    M[X]< R1 * R2
```

2. Two-Address Instructions:

```
MOV    R1, A      R1< M[A]
ADD    R1, B      R1< R1 + M[B]
MOV    R2, C      R2< M[C]
ADD    R2, D      R2< R2 + M[D]
MUL    R1, R2     R1< R1 * R2
MOV    X, R1      M[X]< R1
```

3. ***One-Address Instruction:***

```
LOAD    A    Ac< M[A]
ADD     B    Ac< Ac + M[B]
STORE   T    M[T]< Ac
LOAD    C    Ac< M[C]
ADD     D    Ac< Ac + M[D]
MUL     T    Ac< Ac * M[T]
STORE   X    M[X]<  Ac
```

Here, T is the temporary memory location required for storing the intermediate result.

*4. **Zero-Address Instructions:***

```
PUSH      A      TOS< A
PUSH      B      TOS< B
ADD              TOS< (A + B)
PUSH      C      TOS< C
PUSH      D      TOS< D
ADD              TOS< (C + D)
MUL              TOS< (C + D) * (A + B)
POP       X      M[X]< TOS
```

TOS stands for top of stack.

Addressing Modes—

1. Immediate Addressing

The simplest form of addressing is immediate addressing, in which the operand value is present in the instruction i.e.,

Operand = A

The advantage of immediate addressing is that no memory reference other than the instruction fetch is required to obtain the operand. The disadvantage is that the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length.

2. Direct Addressing Mode

In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.

EA=A

3. Indirect Addressing

With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range. One solution is to have the address field refer to the address of a word in memory, which in turn contains a full-length address of the operand. This is known as indirect addressing:

$$EA = (A)$$

There does not appear to be any particular advantage to this approach, and its disadvantage is that three or more memory references could be required to fetch an operand.

4. Register Addressing

Register addressing is similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address:

$$EA = R$$

The advantages of register addressing are that
(1) only a small address field is needed in the instruction
(2) no time-consuming memory references are required.

The disadvantage of register addressing is that the address space is very limited.

5. Register Indirect Addressing

Just as register addressing is analogous to direct addressing, register indirect addressing is analogous to indirect addressing. In both cases, the only difference is whether the address field refers to a memory location or a register. Thus, for register indirect address,

$$EA = (R)$$

The advantages and limitations of register indirect addressing are basically the same as for indirect addressing.

6. Displacement Addressing

A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing. It is known by a variety of names depending on the context of its use, but the basic mechanism is the same. We will refer to this as displacement addressing:

$$EA = A + ®$$

7. Relative Addressing Mode

It is a version of Displacement addressing mode. In this the contents of PC(Program Counter) is added to address part of instruction to obtain the effective address.

$$EA = A + (PC)$$

8. Stack Addressing Mode

In this mode, operand is at the top of the stack. For example: ADD, this instruction will POP top two items from the stack, add them, and will then PUSH the result to the top of the stack.

9. Base Register Addressing Mode

It is again a version of Displacement addressing mode. This can be defined as

$$EA = A + (R)$$

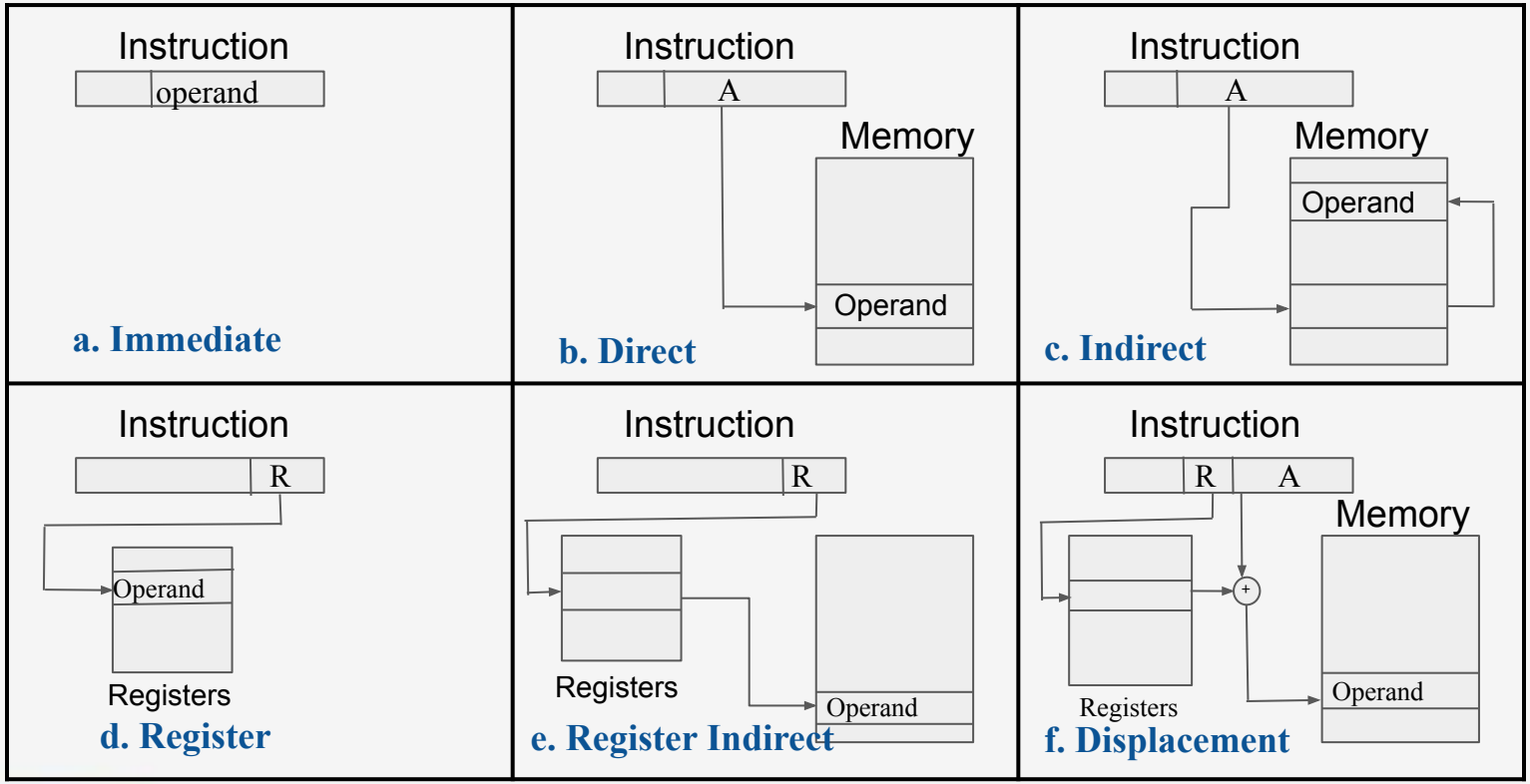
where A is displacement and R holds pointer to base address.

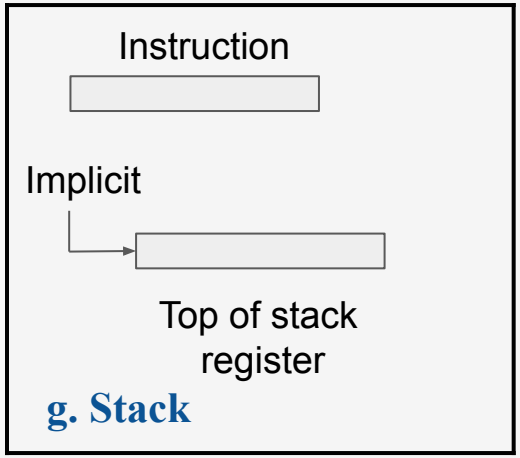
10. Indexed Addressing Mode

In this mode the content of an index register (XR) is added to the address part of the instruction to obtain the effective address.

The index register is a special CPU register that contains an index value.

$$EA = XR + A$$


Instruction
operand
a. Immediate
Instruction
A
Memory
Operand
b. Direct
Instruction
A
Memory
Operand
c. Indirect
Instruction
R
Operand
Registers
d. Register
Instruction
R
Registers
Operand
e. Register Indirect
Instruction
R
A
Memory
Registers
Operand
f. Displacement



g. Stack

Instruction Types—

Computer manufacturers regularly group instructions into the following Categories -

- Data Movement Instructions
- Arithmetic Instructions
- Boolean Instructions
- Bit Manipulation Instructions
- I/O Instructions
- Control Instructions
- Special Purpose Instructions

1. Data Movement Instructions

These are most frequently used instructions. Through this instructions data
Is moved from memory to register, from register to register, and from registers to memory, and many machines provide different instructions
depending upon the source and destination.
For example: MOVE, LOAD etc.

2. Arithmetic Instructions:

These include those instructions that use integers and floating point numbers. These instructions are used to perform arithmetic operations.
For example: ADD etc.

3. Boolean Logic Instructions:

These instructions perform Boolean operations, much in the same way that arithmetic operation. For example: NOT, OR, AND etc.

4. Bit Manipulation Instructions

These instructions are used for setting and resetting individual bits within a given data word. These include both arithmetic and logical shift instructions and rotate instructions, both to the left and to the right.

5. I/O Instructions

An instruction which is used to print something on the screen or any other output device is known as output instruction. Similarly, an instruction which is used to input information from the user is known as input instruction.

6. Special Purpose Instructions

These includes those used for string processing, high level language support, protection, flag control, and cache management.

7. Control Instructions

These includes branches, skips, and procedure calls.
Branching can be unconditional or conditional. Skip instructions are basically Branch instructions with implied address. Because no operand is required, skip instructions often use bits if the address field specify different situations. Procedure calls are branch instruction that automatically save the return address.

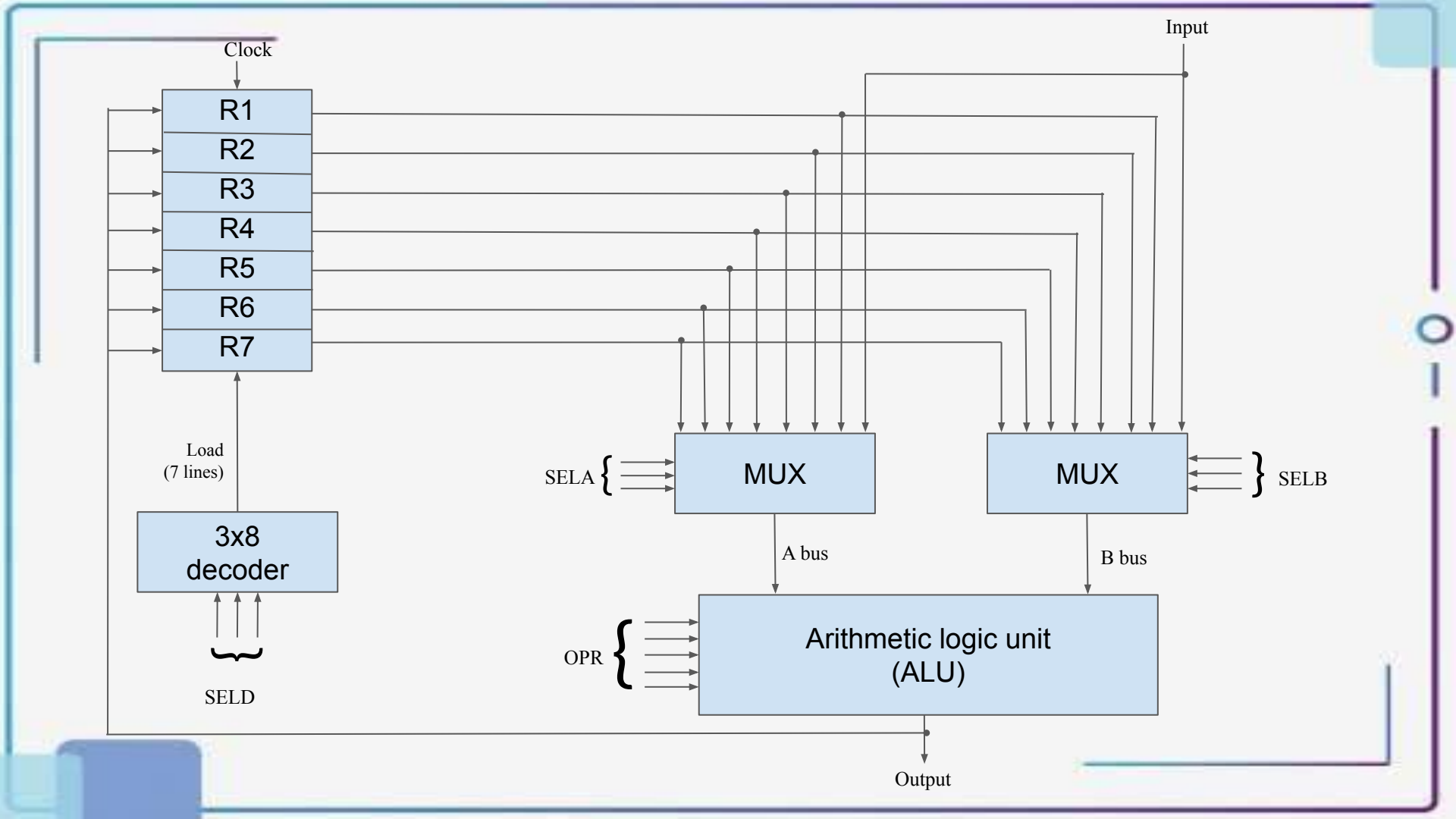
General Register Organization—

The design of a CPU is a task that involves choosing the hardware for implementing the machine instructions.

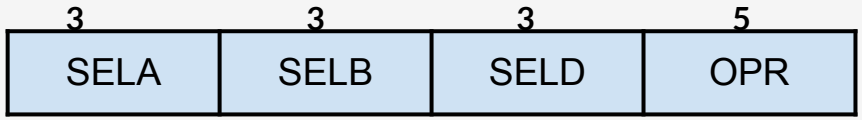
Let's describe how the registers communicate with the ALU through buses and explain the operation of the memory stack-

A bus organization for seven CPU registers is shown in Fig. The output of each register is connected to two multiplexers (MUX) to form the two buses A and B. The selection lines in each multiplexer select one register or the input data for the particular bus. The A and B buses form the inputs to a common arithmetic logic unit (ALU). The operation selected in the ALU determines the arithmetic or logic micro operation that is to be performed.

The result of the micro operation is available for output data and also goes into the inputs of all the registers. The register that receives the information from the output bus is selected by a decoder. The decoder activates one of the register load inputs, thus providing a transfer path between the data in the output bus and the inputs of the selected destination register.


Clock
Input
R1
R2
R3
R4
R5
R6
R7
Load
(7 lines)
3x8
decoder
SELD
SELA
MUX
MUX
SELB
A bus
B bus
OPR
Arithmetic logic unit
(ALU)
Output

Control Words—



A control Word (CW) is a word whose individual bits represent a various control signals. There are 14 binary selection inputs in the unit, and their combined value specifies a control word.

The 14-bit control word is defined in Fig. above. It consists of four fields. Three fields contain three bits each, and one field has
five bits.

- The three bits of SELA select a source register for the A input of the ALU.
- The three bits of SELB select a register for the B input of the ALU.
- The three bits of SELD select a destination register using the decoder and its seven load outputs.
- The five bits of OPR select one of the operations in the ALU.

The 14-bit control word when applied to the selection inputs specify a particular micro operation.

Stack Organization—

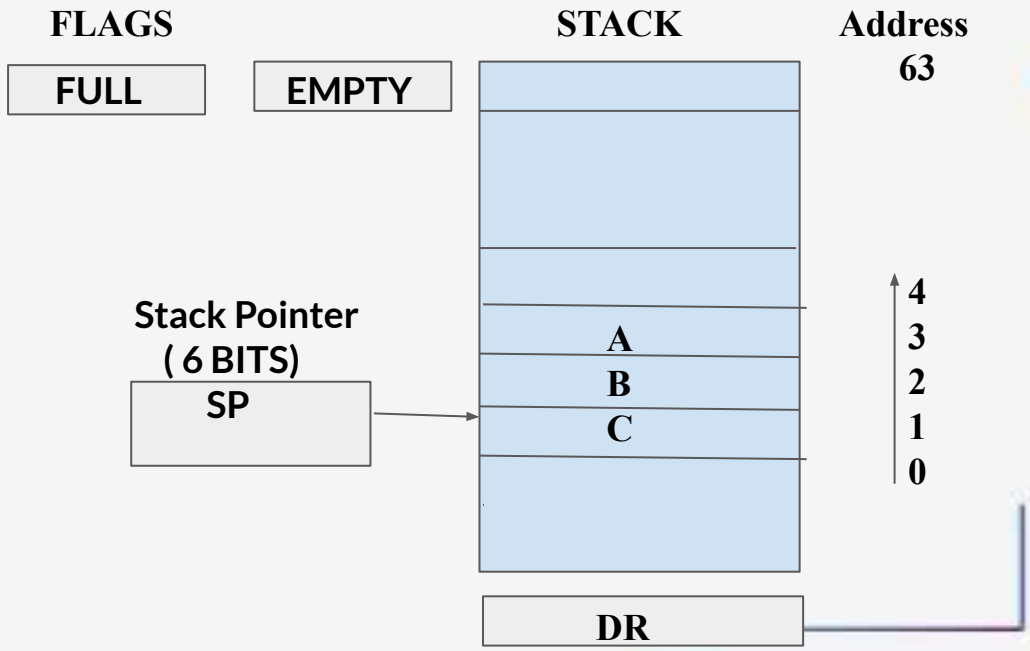
Stack is a storage device that stores information in a way that the item is stored last is the first to be retrieved (**LIFO**).
Stack in computers is actually a memory unit with address register (stack pointer SP) that can count only. SP value always points at points at top item in stack.

The two operations done on stack are:
PUSH (Push Down), operation of insertion of item into stack.
POP (Pop Up), operation of deletion item from stack.

1. Register stack

The stack can be arranged as a set of registers.
Here, 64 location stack unit with SP that stores address of the word that is currently on the top of stack.

3 items are placed in stack A, B, and C. Item C is in top of stack so that SP holds 3 which address of item C.



~To remove top item from stack we start by reading content of address 3 and decrementing the content of SP. Item B is now in top of stack holding address 2.
~To insert new item we start by incrementing SP then writing new word where SP now points top of stack.

In 64 word stack we need to have SP of 6 bits only (000000 to 111111). If 111111 is reached then at next push SP will be 000000, that is when the stack is **FULL.** Similarly when SP is 000001 then at next pop will go to 000000 that is when stack is **EMPTY.**

Initially, SP = 0
EMPTY=1
FULL=0

Procedure for pushing stacks~

SP ← SP + 1

M[SP] ← DR

IF (SP = 0) THEN (FULL = 1)

EMPTY ← 0

Procedure for popping stack~

DR ← M[SP]

SP ← SP - 1

IF (SP=0) THEN (EMPTY=1)

FULL ← 0

2. Memory Stack

Stack can be implemented in RAM memory attached to CPU. Only by assigning special part of it for stack operation.

Stack grows (pushed) with decreasing address and empties (pops) with increasing address.

New item is inserted with push operation by decrementing SP then write to SP address is done.

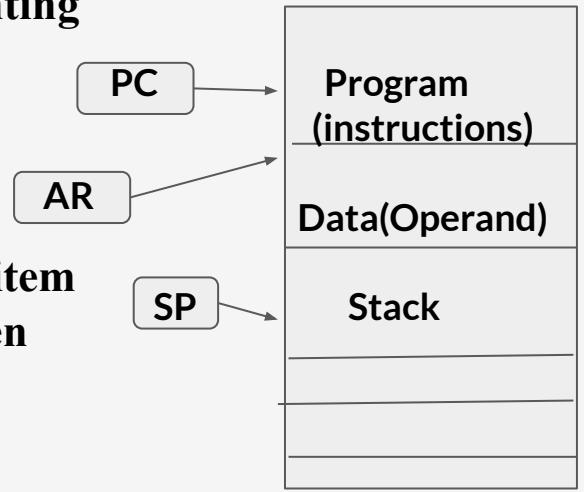
SP ← SP - 1

M[SP] ← DR

Last item is inserted with pop operation by removing item by reading from memory location addressed by SP then SP is incremented.

DR ← M[SP]

SP ← SP + 1



Multiprocessor—

- It is a collection of various processors which are arranged in a way which results in improved performance of the computer system.
- It provides faster speed in comparison to uniprocessors.
- It is system with two or more processors that share main memory and peripherals to process programs simultaneously.

Types of Multiprocessor :-

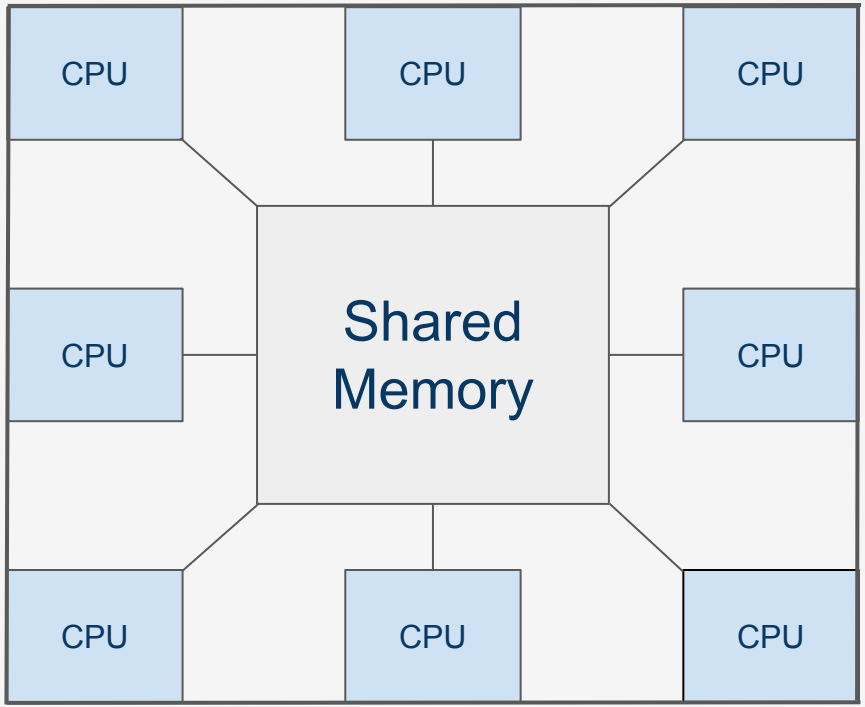
There are mainly two types of multiprocessors :

1. **Symmetric Multiprocessor System (SMP) -**

In this system, each processor has a similar copy of operating system and they all communicate with each other.
This is also known as Multiprocessor system with symmetry.
In this, all processors are in peer-to-peer arrangement.

2. **Asymmetric Multiprocessor system (ASMP) -**

In asymmetric system, each CPU is assigned a particular task/duty. Here, a master processor is in charge of giving instructions to other processors.
A master-slave relationship exists in this system.


CPU
CPU
CPU
CPU
Shared Memory
CPU
CPU
CPU
CPU

Advantages :-

- It is reliable because different processors share their work and so, work is completed with cooperation.
- It helps the system to achieve parallel processing.
- It enhances the performance of computer.
- It gives immediate response.

Disadvantages -

- Since, they work with different systems,so processors require memory space.
- These are very expensive.
- It creates deadlock if any processor already utilized the same input output device.
- It is complicated.
  -

Applications -

- It can be used as a uniprocessor , such as single instruction single data (SISD).
- It is used as Multiple instruction multiple data (MIMD) for executing multiple series of instructions.
- It can be used as Single instruction Multiple data (SIMD) which is used for vector processing.
- As Multiple instruction single data (MISD) used for describing pipeline processors.
  -
  -

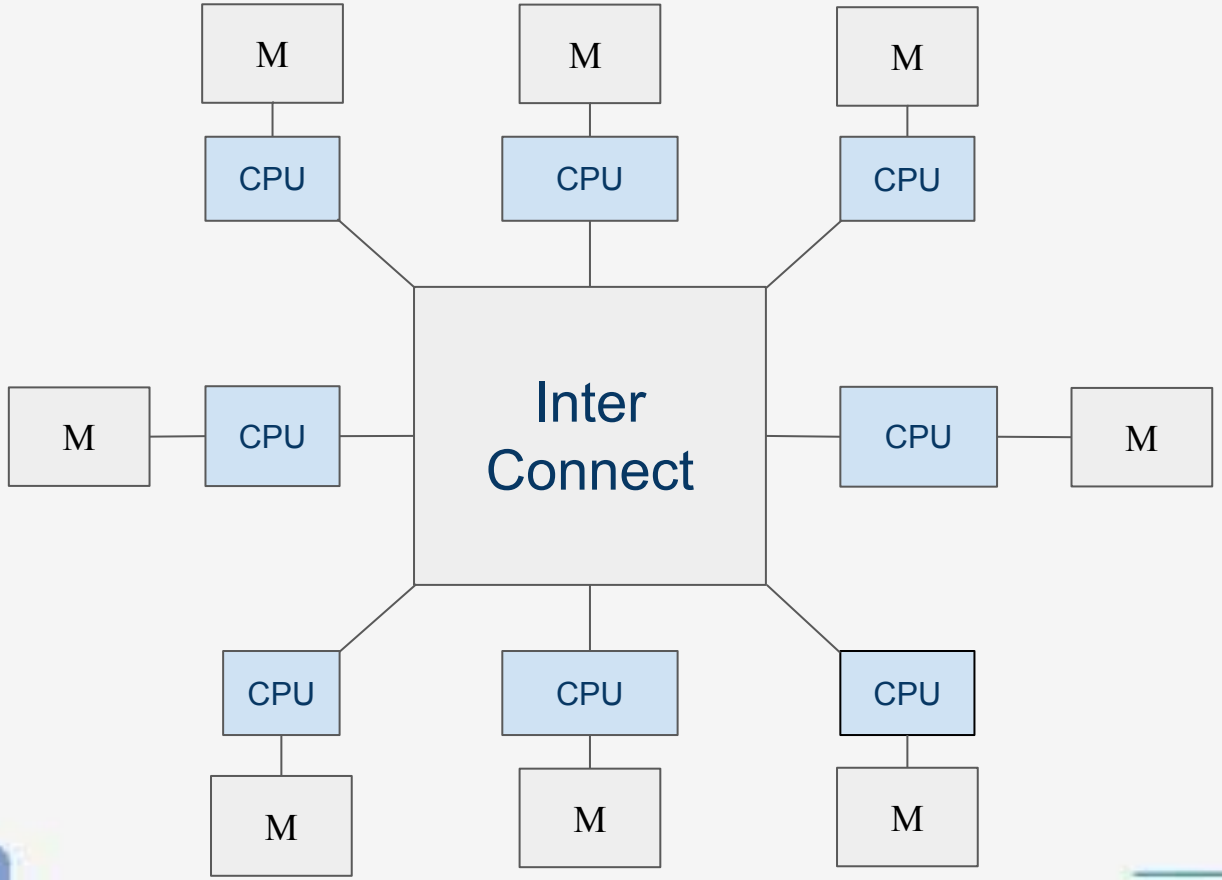
Multi computer—

- It is a computer system having multiple processors connected together to solve a problem.
- It is an interlinked structure of multiple autonomous computers.
- In this, memory of processing elements is distributed.
- It is constructed by interconnecting multiple autonomous computers through a network and each autonomous system has its own computing resource.

- In this, communication between different processing elements is not compulsory.
- An example of multicomputer is Message passing multicomputer.

Advantages -

- It is cost-effective.
- It is easier to build in comparison to Multiprocessor.
- It performs distributive computing.


M
M
M
CPU
CPU
CPU
M
CPU
Inter
Connect
CPU
M
CPU
CPU
CPU
M
M
M

Difference between Multiprocessor and Multicomputer :

Multiprocessor	Multicomputer
1. It is a system with two or more CPUs that allows simultaneous processing of program.	1. It is a set of processors connecting by communication network.
2. Programming of Multiprocessor is easier.	2. Programming of multicomputer is pretty difficult.
3. Parallel computing is used.	3. Distributive computing is used.
4. It is expensive.	4. Cost of system is less expensive.
5. It runs slower because it would be in one computer.	5. It runs faster.

Multi core Architecture—

Definition:

- A multi core architecture (or a chip multiprocessor) is a general purpose processor that consists of multiple cores on the same die and can execute programs simultaneously.
- Multi-core refers to an architecture in which a single physical processor incorporates the core logic of more than one processors. These single integrated circuit are known as a die.

- Multi-core architecture can be homogeneous or heterogeneous. Homogeneous processors are those that have identical cores on the chip. Heterogeneous processors have different type of cores on the chip..

Advantages -

- In can complete more work than single-center processors.
- Printed circuit board requires less space in case of utiliziing multi-core processors.
- It helps to complete complex work too.
- Can finish synchronous work as low recurrence.

Disadvantages -

- These are hard to oversee when compared with single-center processor.
- If great power is supplied then,they burn out..
- Very expensive.
- Become hot while accomplishing more work.

Reverse Polish Notation in Stack—

Very useful notation to utilize stacks to evaluate arithmetic expressions.
The 3 notations to evaluate expressions
1. A + B Infix notation
2. +AB Prefix notation (Polish notation)
3. AB+ Postfix notation (reverse Polish notation)

Reverse Polish Notation is in a form suitable for stack manipulation. Starts by scanning expression from left to right. When operator is found then perform operation with 2 operands in left of operator and replace result place of 2 operands and operator. Then we can continue this until you reach final answer.

Example: Expression A*B + C*D is written in RPN as AB*CD*+. And will be computed as
(A*B) CD *+
(A*B)(C*D)+

References—

1. ‘Computer System Architecture’, Morris M. Mano, 3rd edition,Prentice Hall India.
2. Computer Organization and Achitecture, William Stallings ,8th edition,PHI
3. Computer Organization, Carl Hamachar, Vranesic, McGraw Hill.
4. https://www.geeksforgeeks.org
5. https://www.researchgate.net
6. http://www.cambridgegcsecomputing.org
7. https://www.sciencedirect.com
8. https://dl.acm.org/profile/81344494349