Unit – 3 Design and Organization of Basic Computer, CPU

Topics covered into this unit

- ❖ **Basic Computer Organization and Design**
- ➢ **Instruction Codes**
  - **Stored Program Organization**
  - **Indirect Address**
- ➢ **Computer Register**
  - **Common Bus System**
- ➢ **Computer Instructions**
  - **Instruction Set Completeness**
- ➢ **Timing and Control**
- ➢ **Instruction Cycle**
  - **Fetch and Decode**
  - **Determine the Type of Instruction**
- ➢ **Register Reference Instruction**
- ➢ **Memory Reference Instruction**
- ➢ **Input-Output and Interrupt**
- ➢ **Complete Computer Description**
- ➢ **Design of Basic Computer**
- ➢ **Design of Accumulator Logic**

- ❖ **Central Processing Unit**
- ➢ **Introduction**
- ➢ **General Register Organization**
- ➢ **Stack Organization**
- ➢ **Instruction Formats**
- ➢ **Addressing Modes**

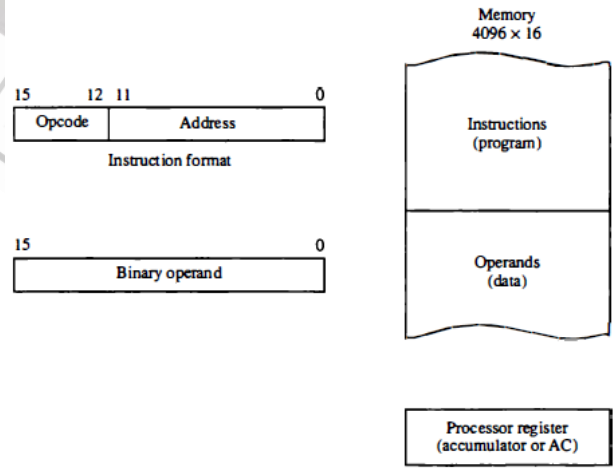
❖ **Basic Computer Organization and Design**

➢ **Instruction Codes**

- The organization of the com-puter is defined by its Internal registers, the timing and control structure, and the set of instructions that it uses.
- The Internal organization of a digital system is defined by the sequence of microoperations it performs on data stored in its register.
- The user of a computer can control the process by means of a program.
- *A program is a set of Instructions that specify the operations, operands, and the sequence by which processing has to occur.*
- The data processing task may be altered by specifying a new program with different instructions or specifying the same instructions with different data.
- A computer instruction is a binary code that specifies a sequence of microoperations for the computer.
- Instruction codes together with data are stored in memory.
- The computer reads each instruction from memory and places it in a control register.
- The control then interprets the binary code of the instruction and proceeds to execute it by issuing a sequence of microoperations.
- *"An instruction code is a group of bits that instruct the computer to perform a specific operation".*
- It is usually divided into parts, each having its own particular interpretation.
- The most basic part of an instruction code is its operation part. The operation code of an instruction is a group of bits that define such operations as add, subtract, multiply, shift, and complement.
- An operation is part of an instruction stored in computer memory. It is a binary code that tells the computer to perform a specific operation.
- This operation must be performed on some data stored in processor registers or in memory.
- An instruction code must therefore specify not only the operation but also the registers or the memory words where the operands are to be found, as well as the register or memory word where the result is to be stored.

• **Stored Program Organization**

- The simplest way to organize a computer is to have one processor register and an instruction code format with two parts.
- The first part specifies the operation to be performed and the second specifies an address.
- The memory address tells the control where to find an operand in memory. This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.

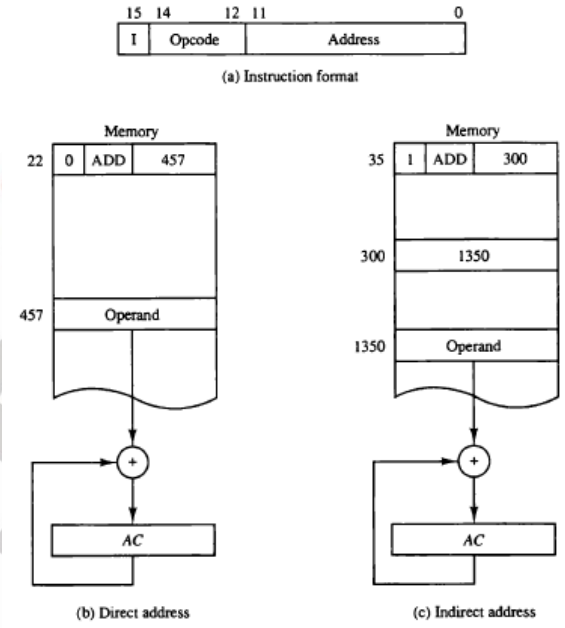


Stored Program Organization

- Instructions are stored in one section of memory and data in another.
- For a memory unit with 4096 words we need 12 bits to specify an address since 212 = 4096.
- If we store each instruction code in one 16-bit memory word, we have available four bits for the operation code (opcode) to specify one out of 16 possible operations, and 12 bits to specify the address of an operand.
- The control reads a 16-bit instruction from the program portion of memory.
- It uses the 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory.
- It then executes the operation specified by the operation code.

- **Indirect Address**

- Sometimes convenient to use the address bits of an instruction code not as an address but as the actual operand.
- When the second part of an instruction code specifies an operand, the instruction is said to have an immediate operand.
- When the second part specifies the address of an operand, the instruction is said to have a direct address.
- This is in contrast to a third possibility called indirect address, where the bits in the second part of the instruction designate an address of a memory word in which the address of the operand is found.
- One bit of the instruction code can be used to distinguish between a direct and an indirect address.

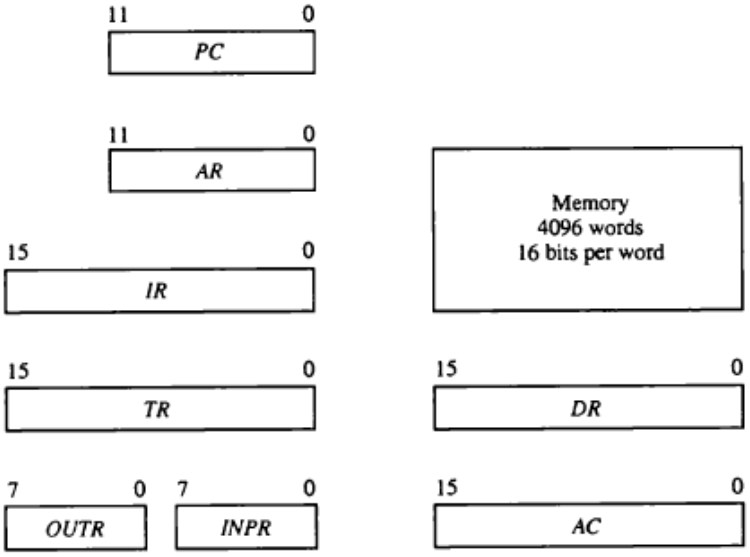


Direct and Indirect Address

- **Computer Register**

- Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time.
- The control reads an instruction from a specific address in memory and executes it.
- It then continues by reading the next instruction in sequence and executes it, and so on.
- It is also necessary to provide a register in the control unit for storing the instruction code after it is read from memory.
- The computer needs processor registers for manipulating data and a register for holding a memory address.

- These requirements dictate the register configuration shown in below figure:



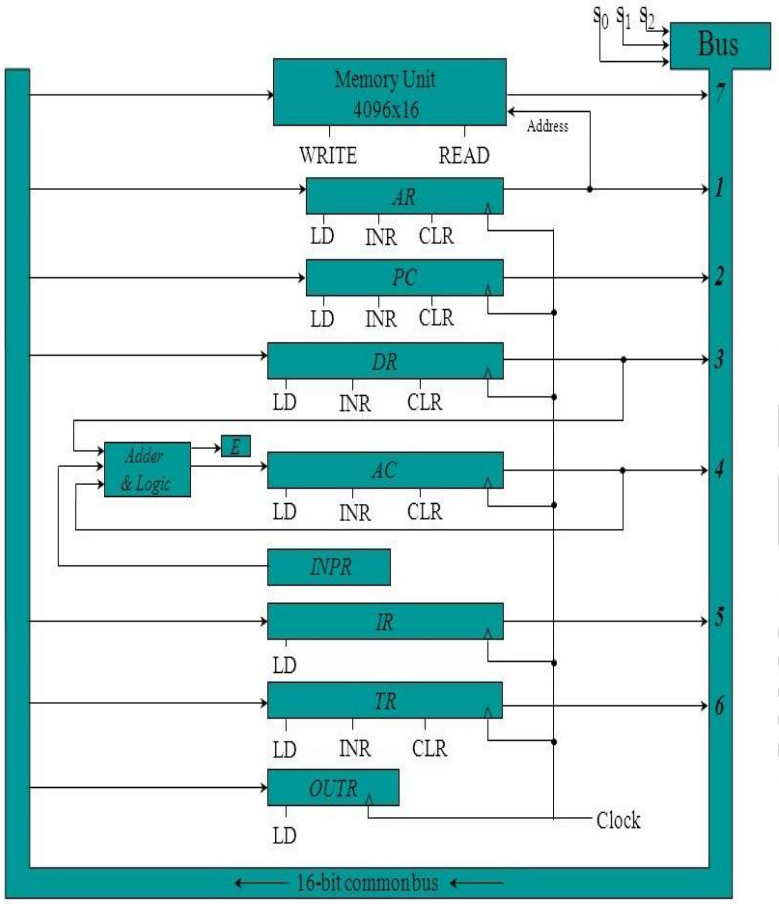
Basic Computer Register and Memory

- The registers are also listed with a description of their function and the number of bits that they contain in below table:

Register Symbol	No. of Bits	Register Name	Function
DR	16	Data Register	Holds memory operands.
AR	12	Address Register	Holds address for memory.
AC	16	Accumulator	Processor Register.
IR	16	Instruction Register	Holds Instruction Code.
PC	12	Program Counter	Holds address of Instructions.
TR	16	Temporary Register	Holds Temporary Data.
INPR	8	Input Register	Holds Input Character.
OUTR	8	Output Register	Holds Output Character.

- **Common Bus System**

- The basic computer has eight registers, a memory unit, and a control unit.
- Paths must be provided to transfer information from one register to another and between memory and registers.
- The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers.
- A more efficient scheme for transferring information in a system with many registers is to use a common bus.
- The 16 lines of the common bus receive information from six registers and the memory unit.
- The bus lines are connected to the inputs of six registers and the memory. Five registers have three control inputs: LD (load), INR (increment), and CLR (clear).
- INPR is connected to provide information to the bus but OUTR can only receive information from the bus.
- INPR receives a character from an input device which is then transferred to AC. OUTR receives a character from AC and delivers it to an output device.
- There is no transfer from OUTR to any of the other registers.



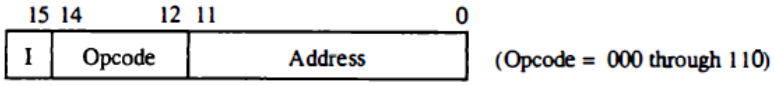
➢ **Computer Instructions**

- The basic computer has three instruction code formats which are as follows:

1. **Memory-Reference Instruction**
2. **Register-Reference Instruction**
3. **Input-Output Instruction**

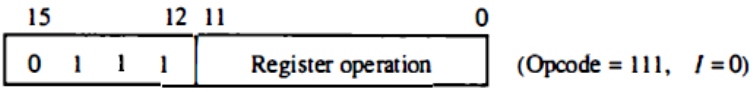
- Each format has 16 bits. The operation code (opcode) part of the instruction contains three bits and the meaning of the remaining 13 bits depends on the operation code encountered.

1. Memory-Reference Instruction



- A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I.
- I is equal to 0 for direct address and to 1 for indirect address.

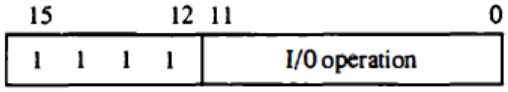
2. Register-Reference Instruction



- The register reference instructions are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction.
- A register-reference instruction specifies an operation on or a test of the AC register.

- An operand from memory is not needed; therefore, the other 12 bits are used to specify the operation or test to be executed.

3. Input-Output Instruction



(Opcode = 111, $I = 1$)

- An input-output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction.
- The remaining 12 bits are used to specify the type of input-output operation or test performed.

- **Instruction Set Completeness**

- A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable.
- The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:
  1. Arithmetic, logical, and shift instructions
  2. Instructions for moving information to and from memory and processor registers
  3. Program control instructions together with instructions that check status conditions
  4. Input and output instructions
- Arithmetic, logical, and shift instructions provide computational capabilities or processing the type of data that the user may wish to employ.
- The binary information in a digital computer is stored in memory, but all computations are done in processor registers. Therefore, the user must have the capability of moving information between these two units.
- Program control instructions such as branch instructions are used to change the sequence in which the program is executed.
- Input and output instructions are needed for communication between the computer and the user.
- Programs and data must be transferred into memory and results of computations must be transferred back to the user.

➢ **Instruction Cycle**

- A program residing in the memory unit of the computer consists of a sequence of instructions.
- The program is executed in the computer by going through a cycle for each instruction.
- Each instruction cycle in turn is subdivided into a sequence of subcycles or phases.
- In the basic computer each instruction cycle consists of the following phases:
  1. Fetch an instruction from memory.
  2. Decode the instruction.
  3. Read the effective address from memory if the instruction has an indirect address.
  4. Execute the instruction.
- Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction.

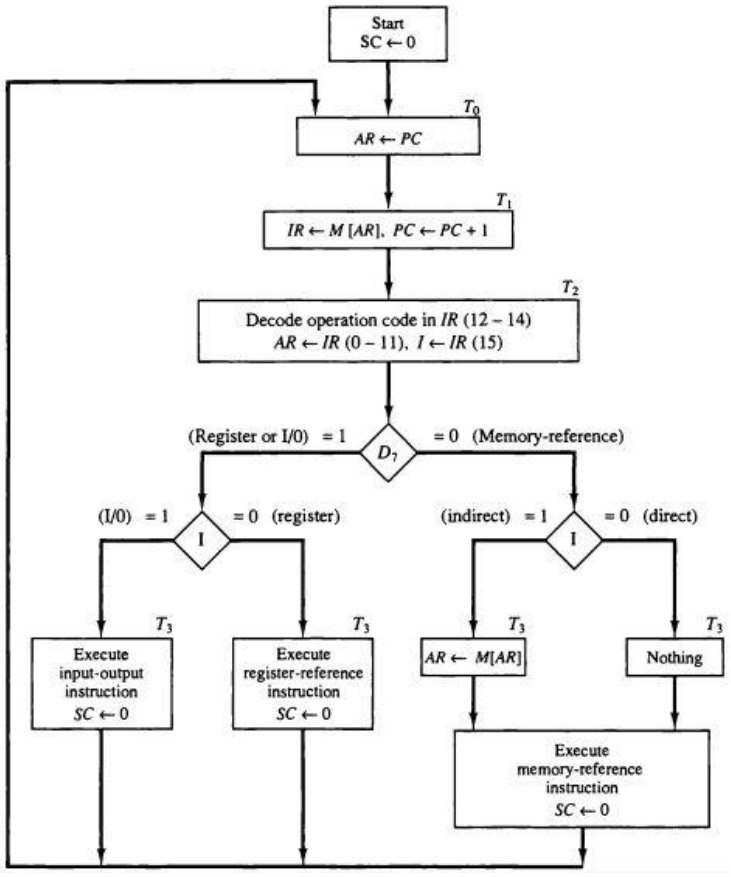
- **Fetch and Decode**

- Initially, the program counter PC is loaded with the address of the first instruction in the program.
- The sequence counter SC is cleared to 0, providing a decoded timing signal T_0.
- After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on.

- The microoperations for the fetch and decode phases can be specified by the following register transfer statements.
  T_0: AR←PC
  T_1: IR←M[AR], PC←PC + 1
  T_2: D_0,…, D_7←Decode IR(12-14), AR←IR(0-11), I←IR(15)
- Since only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associ-ated with timing signal T_0.
- The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T_1. At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.
- At time T_2, the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.
- Note that SC is incremented after each clock pulse to produce the sequence T_0, T_1, and T_2.

- **Determine the Type of Instruction**

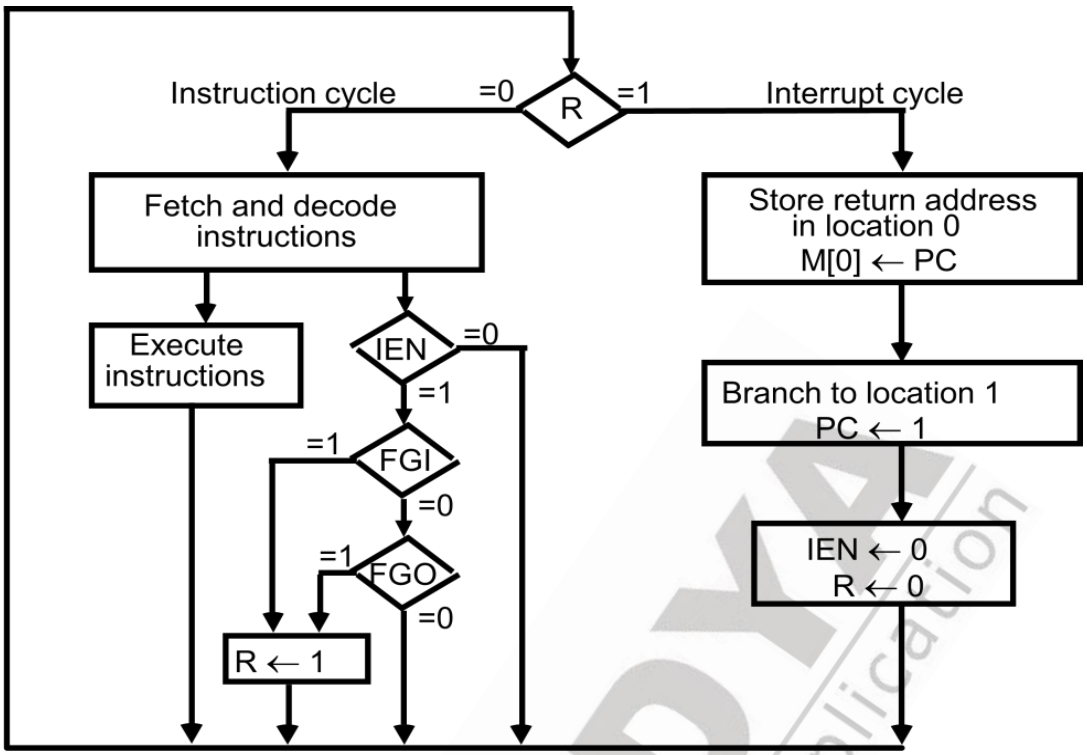
- The timing signal that is active after the decoding is T_3. During time T_3, the control unit determines the type of instruction that was just read from memory.



Flowchart of Instruction Cycle

- The flowchart presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after the decoding.
- The three possible instruction types available in the basic computer.
- Decoder output D, is equal to 1 if the operation code is equal to binary 111.
- If D_7 = 1, the instruction must be a register-reference or input-output type.
- If D_7 = 0, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.

- **Interrupt Cycle with Flow Chart:**



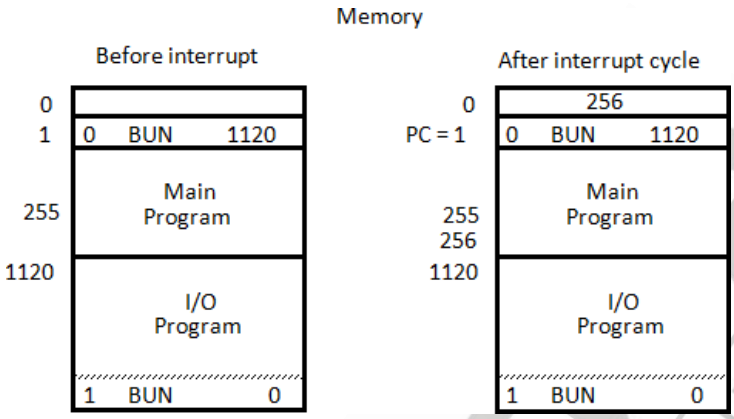
Flowchart for interrupt cycle

- The way that the interrupt is handled by the computer can be explained by means of the flowchart shown in figure.
- An interrupt flip-flop R is included in the computer.
- When R = 0, the computer goes through an instruction cycle.
- During the execute phase of the instruction cycle IEN is checked by the control.
- If it is 0, it indicates that the programmer does not want to use the interrupt, so control continues with the next instruction cycle.
- If IEN is 1, control checks the flag bits.
- If both flags are 0, it indicates that neither the input nor the output registers are ready for transfer of information.
- In this case, control continues with the next instruction cycle. If either flag is set to 1 while IEN = 1, flip-flop R is set to 1.
- At the end of the execute phase, control checks the value of R, and if it is equal to 1, it goes to an interrupt cycle instead of an instruction cycle.

- **Interrupt Cycle**

- The interrupt cycle is a hardware implementation of a branch and save return address operation.
- The return address available in PC is stored in a specific location where it can be found later when the program returns to the instruction at which it was interrupted. This location may be a processor register, a memory stack, or a specific memory location.
- Here we choose the memory location at address 0 as the place for storing the return address.

- Control then inserts address 1 into PC and clears IEN and R so that no more interruptions can occur until the interrupt request from the flag has been serviced.
- An example that shows what happens during the interrupt cycle is shown in Figure



Demonstration of the interrupt cycle

- Suppose that an interrupt occurs and R = 1, while the control is executing the instruction at address 255. At this time, the return address 256 is in PC.
- The programmer has previously placed an input-output service program in memory starting from address 1120 and a BUN 1120 instruction at address 1.
- The content of PC (256) is stored in memory location 0, PC is set to 1, and R is cleared to 0.
- At the beginning of the next instruction cycle, the instruction that is read from memory is in address 1 since this is the content of PC. The branch instruction at address 1 causes the program to transfer to the input-output service program at address 1120.
- This program checks the flags, determines which flag is set, and then transfers the required input or output information. Once this is done, the instruction ION is executed to set IEN to 1 (to enable further interrupts), and the program returns to the location where it was interrupted.
- The instruction that returns the computer to the original place in the main program is a branch indirect instruction with an address part of 0. This instruction is placed at the end of the I/O service program.
- The execution of the indirect BUN instruction results in placing into PC the return address from location 0.

➢ Design of basic Computer

The basic computer consists of the following hardware components:

1. A memory unit with 4096 words 16 bits each.
2. Nine registers: AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC.
3. Seven flip-flops: I, S, E, R, IEN, FGI, and FGO.
4. Two decoders: a 3 * 8 operation decoder and a 4 * 16 timing decoder.
5. A 16-bit common bus.
6. Control logic gates.
7. Adder and logic circuit connected to the input of AC.

- **instruction formats:**
  - Computer may have instructions of several different lengths containing varying number of address.
  - The number of address fields in the instruction format of a computer depends on the internal organization of its registers.
  - Most computer fall into one of three types of CPU organizations.

1. Single accumulator organization
2. General register organization
3. Stack organization.

- The number of address fields available in instruction defines the format of instruction. There are different format for the instructions like zero,one, two, three address fields.
- Let the arithmetic operation be: X = (A + B) * (C + D)
- **List of Instruction formats:**

1. Zero address instruction.
2. One address instruction.
3. Two address instruction.
4. Three address instruction.
5. RISC instruction.

1. Zero address instruction:

- A stack organized computer does not use an address field for the instructions ADD and MUL.
- The PUSH and POP instructions, however, need and address field to specify the operand that communicates with the stack
- The following program shows how X = (A + B) * (C + D) will be written for a stack organized computer.

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) 8 (A + B)
POP X	M[X]←TOS

- To evaluate arithmetic expression in a stack computer, it is necessary, to convert the expression into reverse polish notation.
- The name 'Zero-address' is given to this type of computer because of the absence of an address field in the computational instruction.

2. One-Address Instructions:

- One-Address Instructions use an implied accumulator (AC) register for all data manipulation.
- For multiplication and division there is a need for a second register.
- However, here we will neglect the second register and assume that the AC contains the result of all operations.

- The program to evaluate X= (A + B) * (C + d) is:

```
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
```

- All operations are done between the AC register and a memory operand.
- T is the address of a temporary memory location required for storing the intermediate result.

3. **Two-Address Instruction:**

- Two-Address Instruction are the most common in commercial computers.
- Here, again each address field can specify either a processor register or a memory word.
- The program to evaluate X=(A + B) * (C + D) is as follows:

```
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
```

- The MOV instruction moves or transfers the operands to and from memory and processor registers.
- The first symbol listed in instruction is assumed to be both a source and the destination where the result of the operation is transferred.

4. Three-Address Instructions:

- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.
- The program is assembly language that evaluates X=(A + B) * (C + D) is shown below, together with comments that explain the register transfer operation of each instruction.

```
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
```

- It is assumed that the computer has two processor registers, R1 and R2.
- The symbol M[A] denotes the operand at memory address symbolized by A.
- The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions.
- The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

5. RISC Instruction:

- RISC stands for reduced instruction set computer
- The instruction set of a typical RISC processor is restricted to the use of load and store instructions when communicating between memory and CPU.
- All other instructions are executed with in the registers of the CPU without referring to a memory.
- A program for a RISC type CPU consists of LOAD and STORE instructions that have one memory and one register address, and computational-type instructions that have three addresses with all three specifying processor registers.
- The following is a program to evaluate X=(A + B) * (C + D)

LOAD	R1, A	R1←M[A]
LOAD	R2, B	R2←M[B]
LOAD	R3, C	R3←M[C]
LOAD	R4, D	R4←M[D]
ADD	R1, R1, R2	R1 ←R1 + R2
ADD	R3, R3, R4	R3←R3 + R4
MUL	R1, R1, R3	R1←R1 * R3
STORE	X, R1	M[X]←R1

- The load instructions transfer the operands from memory to CPU registers.
- The add and multiply operations are executed with data in the registers without accessing memory.
- The result of the computations is then stored in memory with a store instruction.

❖ **Addressing Modes**

- The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.
- Computers use addressing mode techniques for the purpose of accommodating one or both the following provisions:

1. To give programing versatility to use the user by providing such facilities as pointers to memory, counters for loop control, indexing of data and program relocation.
2. To reduce the number of bits in the addressing field of the instruction.

- **List of Addressing Modes**

1. Implied Mode
2. Immediate Mode
3. Register Mod
4. Register Indirect Mode
5. Autoincrement or Autodecrement Mode
6. Direct Address Mode
7. Indirect Address Mode
8. Relative Addressing Mode
9. Index Addressing Mode
10. Base Register Addressing Mode

1. Implied Mode:

- Operands are specified implicitly in the definition of the instruction.
- For example, the instruction "Complement accumulator (CMA)" is an implied mode instruction because the operand in the accumulator register is implied in the definition of the instruction.

- In fact, all register reference instructions that use an accumulator and zero address instruction.

2. Immediate Mode:

- Operand is specified in the instruction itself.
- In other words, an immediate mode instruction has an operand field rather than an address field.
- The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.
- Immediate mode of instructions is useful for initializing register to constant value.

E.g. MOV R1, 05H

Instruction copies immediate number 05H to R1 register.

3. Register Mode:

- Operands are in registers that reside within the CPU.
- The particular register is selected from a register field in the instruction.

E.g. MOV AX, BX

Move value from BX to AX register.

4. Register Indirect Mode:

- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.
- Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.
- The advantage of this mode is that address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

E.g. MOV[R1], R2

Value of R2 is moved to the memory location specified in R1.

5. Autoincrement or Autodecrement Mode:

- This is similar to the register indirect mode expect that the register in incremented or decremented after (or before) its value is used to access memory.
- When the Address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be achieved by using the increment or decrement instruction.

6. Direct Address 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.

E.g. ADD 457

7. Indirect Address Mode:

- In this mode the address field of the instruction gives the address where the effective address is stored in memory.
- Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.
- The effective address in this mode is obtained from the following computational:

Effective address = address part of instruction + content of CPU register.

8. Relative Address Mode:

- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually signed number which can be either or negative.

Effective address = address part of instruction + content of PC

9. Index Addressing Mode:

- In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.
- The indexed register is a special CPU register that contain an index value.
- The address field of the instruction defines the beginning address of a data array in memory.
- Each operand in the array is stored in memory relative to the begging address.

Effective address=address part of instruction + content of index register.

10. Base Register Addressing Mode:

- In this mode the content of a register is added to the address part of the instruction to obtain effective address.
- A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.
- The base register addressing mode is used in computers to facilitate the relocation program in memory.

Effective address=address part of instruction + content of base register