1.1) Computer Organization and Computer Architecture

refers to those attributes of a system visible to a programmer or those attributes that help direct impact on logical execution of program. Example of architectural attributes include instructions, instruction set, number of bits used to represent various data type, i/o mechanism and technique of addressing memory.

Computer organization refers to operational units(components) and their inter connections that realize the architectural specification. Organization attributes include those hardware details transferring to the programmer such as control signal, interfaces between computer and peripheral and memory technology used.

Computer organization is concerned with the way the hardware components operate and the way they are connected. is concerned with the behavior of the computer as seen by user.

Example: Von Neumann architecture, Harvard architecture

MIPs processor ISA

- **Fixed 32-bit instructions**
- **3 instruction formats**
- **3 operand, load-store architecture**
- **32 general purpose register**

- **registers are of 32 bits size**
- **addressing mode: register, immediate**

It is an architectural design issue whether a computer will have multiply instruction. It is an organizational design issue whether the multiply instruction will be implemented by a special multiply unit or by a mechanism that makes a repeated use of adder unit of system.

Why to study computer organization and architecture?

***** Computer- heart of computing ……without it ,most of the computing desciplines , theoretical mathematics are of no use.**

-cannot treated computer as an black box that execute a program only.

-should acquire some understanding and appreciation of a computer system functional components,their characteristics,their performance and interaction.

Reasons:-

1) **to select the most cost effective computer for use throughout a large organization(a larger cache or higher processor clock rate)**
2) **many processor are not used in pc or server but in embedded system (a designer may program a processor in C that is embedded in some real time or larger system such as intelligent automobile electronics controller.)**
3) **provides architectural support for programming languages and operating system**

Structure and Function:

A computer is a complex system contains millions of elementary electronic component. **Structure**: The way in which the component are interrelated. **Function**: The operation of each individual component is a part of structure.

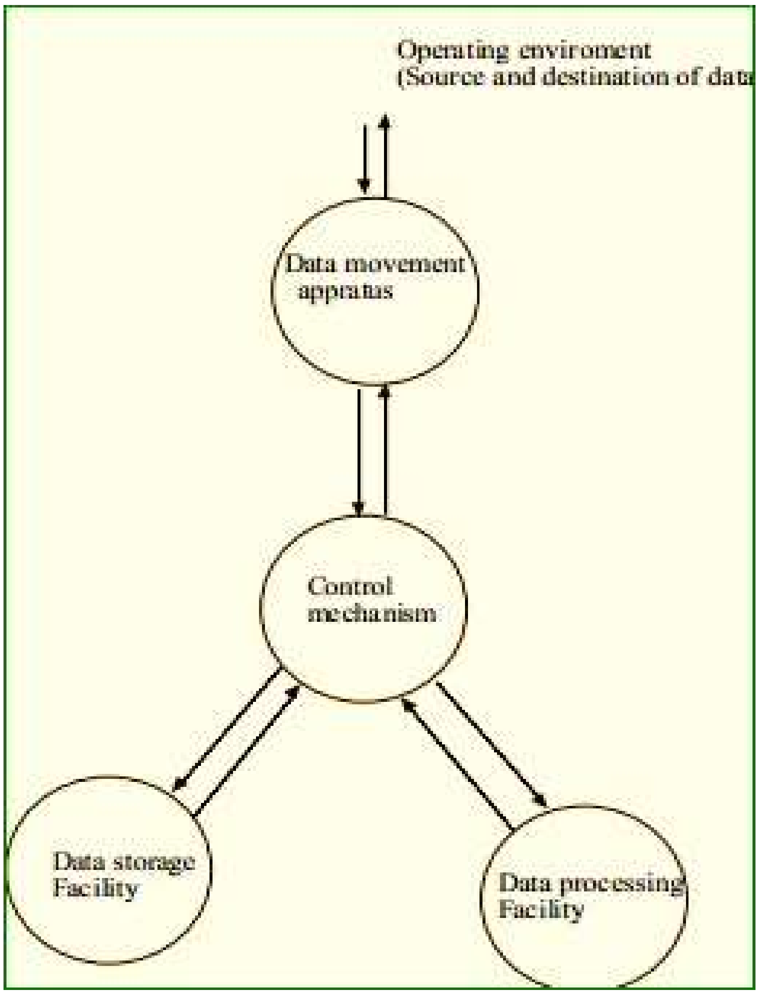


Fig. depicts the basic functions that a computer can perform.

In general terms, there are only four:

- Data processing.
- Data storage.
- Data movement
- Control

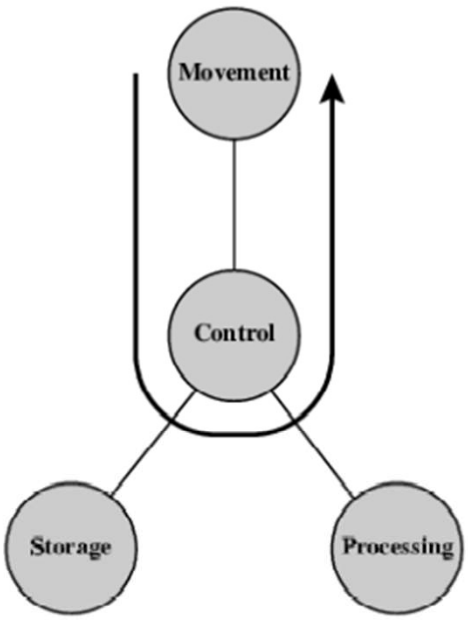


Fig: Data movement operation

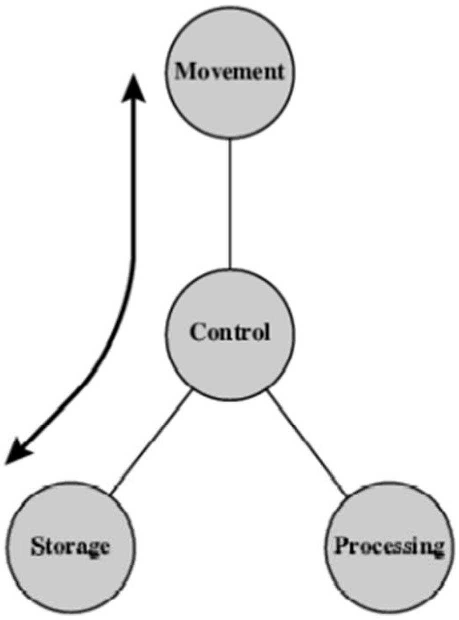
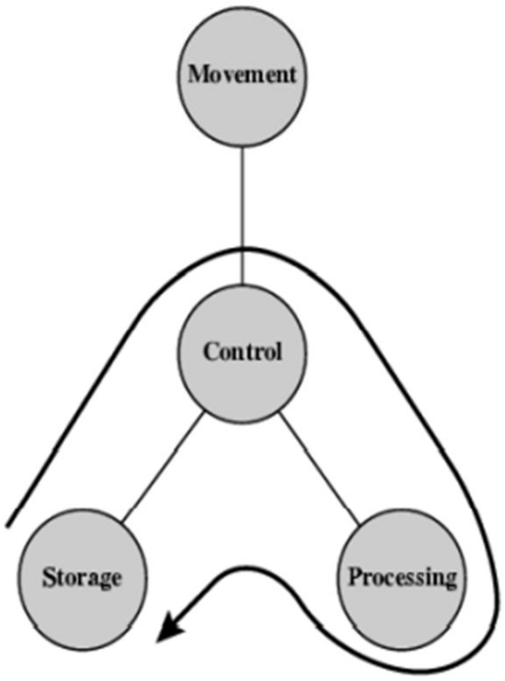


Fig: Storage Operation



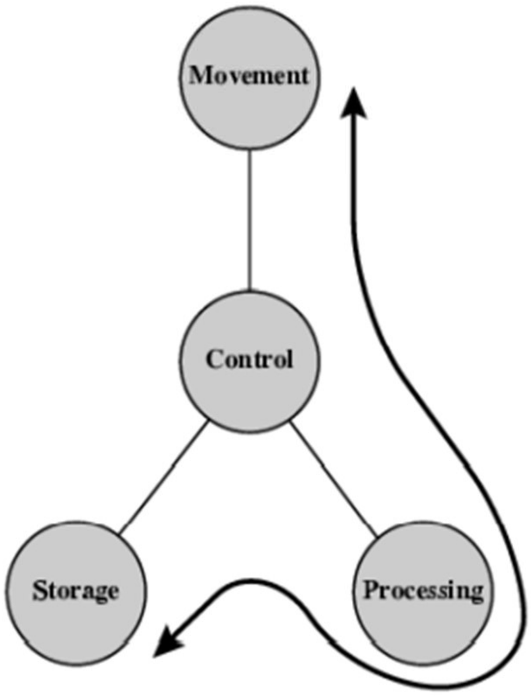


Fig: Data processing from I/O to storage   Fig: Control to all

1.3 Structure:

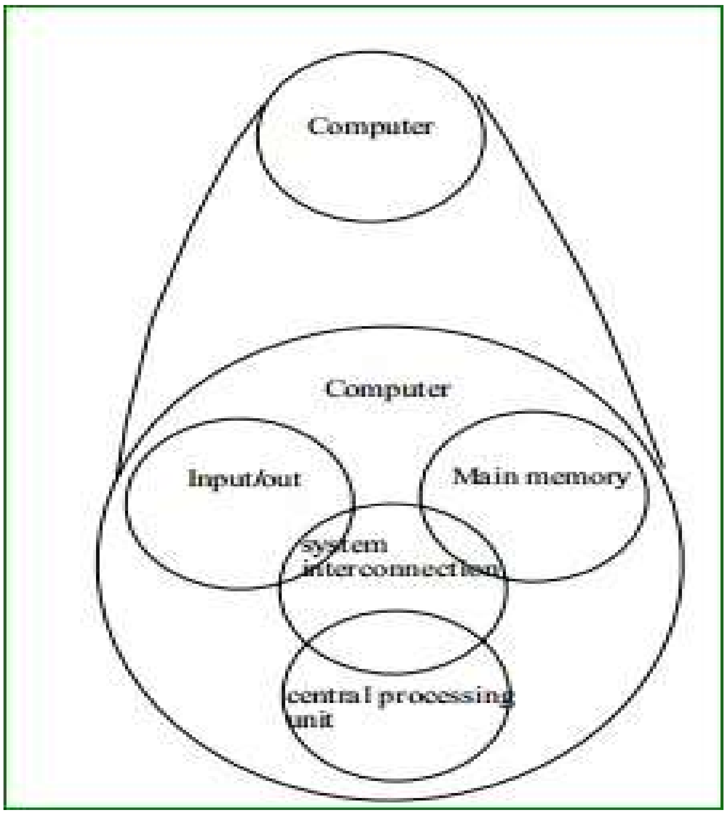


Fig: Computer: Top level structure.

There are four main structural components:

i)Central processing units: Controls the operation of computer and performs its data processing function.

ii) Main memory: Stores data.

iii)I/O : moves data between the computer and its external environment.

iv) System interconnection: Some mechanism that provides for communication among CPU, main memory and I/O.

Designing for performance

Design goals : The exact form of a computer system depends on the constraints and goals for which it was optimized. s usually trade off standards, cost, memory capacity, latency and throughput. Sometimes other considerations, such as features, size, weight, reliability, expandability and power consumption are factors as well.

The most common scheme carefully chooses the bottleneck that most reduces the computer's speed. Ideally, the cost is allocated proportionally to assure that the data rate is nearly the same for all parts of the computer, with the most costly part being the slowest. This is how skillful commercial integrators optimize personal computers.

Performance

Computer performance is often described in terms of clock speed (usually in MHz or GHz). This refers to the cycles per second of the main clock of the CPU. However, this metric is somewhat misleading, as a machine with a higher clock rate may not necessarily have higher performance. As a result manufacturers have moved away from clock speed as a measure of performance.

Computer performance can also be measured with the amount of cache a processor has. If the speed, MHz or GHz, were to be a car then the cache is like the gas tank. No matter how fast the car goes, it will still need to get gas. The higher the speed, and the greater the cache, the faster a processor runs.

Modern CPUs can execute multiple instructions per clock cycle, which dramatically speeds up a program. Other factors influence speed, such as the mix of functional units, bus speeds, available memory, and the type and order of instructions in the programs being run.

There are two main types of speed, latency and throughput. Latency is the time between the start of a process and its completion. Throughput is the amount of work done per unit time. Interrupt latency is the guaranteed maximum response time of the system to an electronic event (e.g. when the disk drive finishes moving some data). Performance is affected by a very wide range of design choices — for example, pipelining a processor usually makes latency worse (slower) but makes throughput better.

Computers that control machinery usually need low interrupt latencies. These computers operate in a real-time environment and fail if an operation is not completed in a specified amount of time. For example, computer-controlled anti-lock brakes must begin braking almost immediately after they have been instructed to brake.

The performance of a computer can be measured using other metrics, depending upon its application domain. A system may be CPU bound (as in numerical calculation), I/O bound (as in a webserving application) or memory bound (as in video editing). Power consumption has become important in servers and portable devices like laptops.

Benchmarking tries to take all these factors into account by measuring the time a computer takes to run through a series of test programs. Although benchmarking shows strengths, it may not help one to choose a computer. Often the measured machines split on different measures. For example, one system might handle scientific applications quickly, while another might play popular video games more smoothly. Furthermore, designers have been known to add special features to their products, whether in hardware or software, which permit a specific benchmark to execute quickly but which do not offer similar advantages to other, more general tasks.

A Functional Unit is defined as a collection of computer systems and network infrastructure components which, when abstracted, can be more easily and obviously linked to the goals and objectives of the enterprise, ultimately supporting the success of the enterprise‘s mission. From a technological perspective, a Functional Unit is an entity that consists of computer systems and network infrastructure components that deliver critical information assets,1 through network-based services, to constituencies that are authenticated to that Functional Unit.

1.6 Review of Instruction Sets, Addressing modes and Instruction Formats

INSTRUCTION FORMATS:

It defines the layout of bits of an instructions.

Instruction = opcode + operand or address field

OPCODE→operation code

It represents the operation to be performed

OPERAND→ Data values

Address field contains address.

Opcode	Operand or address

It represents the data in which the operation is to performed.

There is also a mode field that specifies the addressing mode (I) or effective address.

I	Opcode	Operand or address

TYPES: 4 types

A) Three Address Instructions:

In this format, address field specify either processor register or a memory operand. The advantage is it result in short programs. The disadvantage is that the binary coded instruction require too many bits to specify three addresses. e.g: cyber 170 commercial computer.

opcode	operand 1	operand 2	operand 3

X= (A+B) * (C+D) …..A,B,C,D,X are value variables

```
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
```

B) Two address Instructions:

In this format, the address field contains two address or operands.

opcode	operand1	operand2

X=(A+B) *(C+D)

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

C) One Address Instructions:

In this format, there is only one address field.

opcode	operand1

It uses the implied accumulator (AC) register for all data manipulation.

X=(A+B)* (C+D)

LOAD A	AC←M[A]	
ADD B	AC←AC+ M[B]	
STORE T	M[T]←AC	T=temporary memory location
LOAD C	AC←M[C]	
ADD D	AC←AC+ M[D]	
MUL T	AC←AC * M[T]	

```
STORE X    M[X] ←AC
```

D) ZERO Address Instructions:

$\boxed{\text{opcode}}$

This format requires no address field. PUSH and POP are zero address instruction which requires one address field to communicate with stack. ADD and MUL requires no address field.

X=(A+B)* (C+D)

```
 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
```

Q) Write down the code for the expression Y = (A-B/C) * (D +E*G) in 3,2,1 and 0 address instruction formats.

2.4) ADDRESSING MODES:

ADDRESSING MODES:

It is also known as memory addressing.

It is the technique of specifying the operands for any instructions. It specify the address of data to be operated

Advantages:

1) provide flexibility in programming (pointer to memory, counters for loop control, indexing of data, program relocation)
2) reduce the number of bits in the addressing field of the instructions

Types of Addressing Modes :

Each instruction of a computer specifies an operation on certain data. The most common addressing modes are:

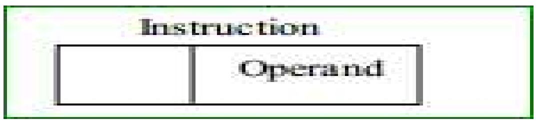
1. Immediate addressing mode
2. Direct addressing mode
3. Indirect addressing mode
4. Register addressing mode
5. Register indirect addressing mode
6. Displacement addressing mode
7. Stack addressing mode

To specify the addressing mode of an instruction several methods are used. Most often used are :

a) Different operands will use different addressing modes.

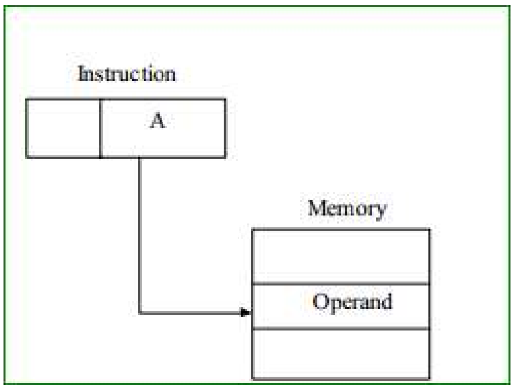
b)One or more bits in the instruction format can be used as mode field. The value of the mode field determines which addressing mode is to be used. The effective address will be either main memory address of a register.

Immediate Addressing:



This is the simplest form of addressing. Here, the operand is given in the instruction itself. This mode is used to define a constant or set initial values of variables. The advantage of this mode is that no memory reference other than instruction fetch is required to obtain operand. The disadvantage is that the size of the number is limited to the size of the address field, which most instruction sets is small compared to word length. e.g:-MVI A,32H

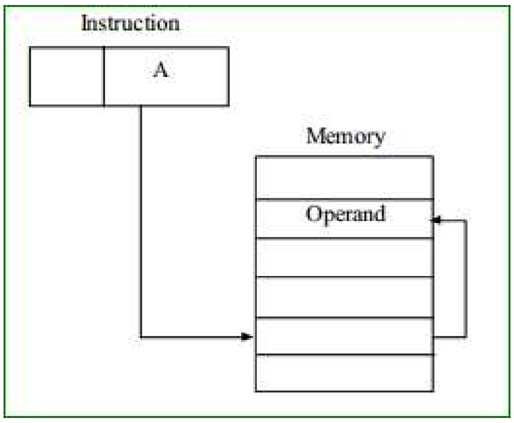
Direct Addressing:



In direct addressing mode, effective address of the operand is given in the address field of the instruction. It requires one memory reference to read the operand from the given location and provides only a limited address space. Length of the address field is usually less than the word length.

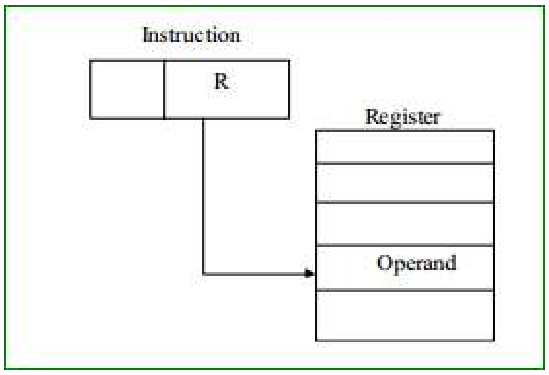
Eg: Mov AX, [5000H]

Indirect Addressing:



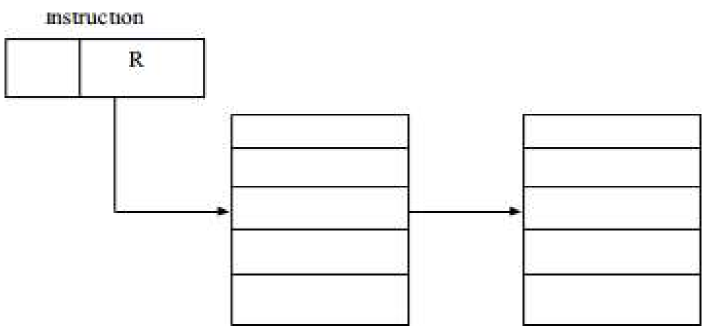
Indirect addressing mode, the address field of the instruction refers to the address of a word in memory, which in turn contains the full length address of the operand. The advantage of this mode is that for the word length of N, an address space of 2N can be addressed. Disadvantage is that instruction execution requires two memory reference to fetch the operand Multilevel or cascaded indirect addressing can also be used. Eg. ADD X , EA = M[X], AC= AC+[M[X]]; Used to implement pointers and passing parameters but high computation is needed.

Register Addressing:



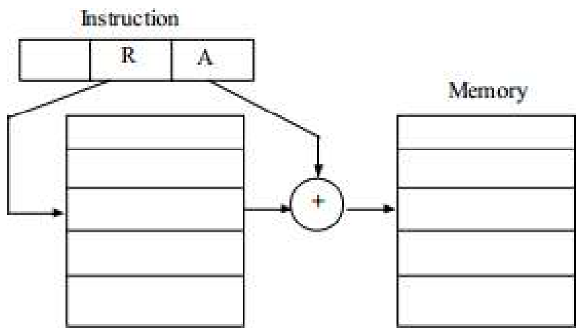
Register addressing mode is similar to direct addressing. The only difference is that the address field of the instruction refers to a register rather than a memory location 3 or 4 bits are used as address field to reference 8 to 16 generate purpose registers. The advantages of register addressing are Small address field is needed in the instruction. Eg. MOV AX,BX;

Register Indirect Addressing:



This mode is similar to indirect addressing. The address field of the instruction refers to a register. The register contains the effective address of the operand. This mode uses one memory reference to obtain the operand. The address space is limited to the width of the registers available to store the effective address. Eg. MOV AX, [BX];

Displacement Addressing:



In displacement addressing mode there are 3 types of addressing mode. They are : EA= A+ [R];

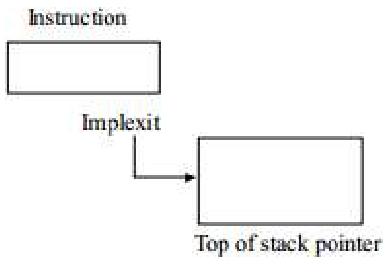
1) Resistor Relative addressing eg. MOV AX, 50H [BX];

2) Base plus index addressing eg. MOV AX, [BX], [SI;

3) Indexing addressing. Eg. MOV AX, [SI];

This is a combination of direct addressing and register indirect addressing. The value contained in one address field. A is used directly and the other address refers to a register whose contents are added to A to produce the effective address.

Stack Addressing:



Stack is a linear array of locations referred to as last-in first out queue. The stack is a reserved block of location, appended or deleted only at the top of the stack. Stack pointer is a register which stores the address of top of stack location. This mode of addressing is also known as implicit addressing.

# INSTRUCTION SETS

There are large number of instructions that a computer provide to perform various tasks.

Categories of instructions:

1) Data transfer instructions
2) Data manipulation instructions
3) Program control instructions

1) Data transfer instructions:

These instructions move data from one location to another without changing the content. The most common transfers are between memory and processor registers, between processor registers and input or output and between processor registers.

Name	**Mnemonics**
load	LOAD
Store	STOR
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
push	PUSH
pop	POP

2) Data manipulation Instructions

These instructions perform operations on data and provide the computational capabilities for the computer.

a) Arithmetic instructions

Addition , subtraction, multiplication and division are the four basic arithmetic operations. some of the typical arithmetic instructions are as follows:

NAME **MNEMONICS**

Increment → INC

Decrement→ DEC

add → ADD ADDI(add two binary integers) ,

Subtract→ SUB

multiply→MUL
divide→DIV
add with carry→ADDC
subtract with borrow→SUBB
negative→NEG

b) logical instructions

Logical instructions are the group of instructions that performs the binary operations on strings of bits stored in registers.
They are useful in manipulating individual bits or a group of bits that represent binary coded information

NAME	MNEMONICS
clear	CLR
complement	COM
and	AND
or	OR
exclusive- or	XOR
clear carry	CLRC
set carry	SETC
complement carry	COMC
enable interrupt	EI
disable interrupt	DI

c) shift instructions

Shift are the operations in which the bits of a word are moved to the left or right.
The bit shifted in at the end of the word determines the type of shift used. (logical shift, arithmetic shift, rotate)

NAME MNEMONICS
logical shift right →SHR
logical shift left→ SHL
arithmetic shift right→SHRA
arithmetic shift left→ SHLA
rotate right→ROR
rotate left→ROL
rotate right through carry→RORC

rotate left through carry→ROLC

3) Program control instructions

Program control instructions change the address value in program counter and cause the flow of control to be altered. They provide branching to different section of the program. They are conditional instructions and looping instructions.

NAME	MNEMONICS
branch→ BR	
jump→JMP	
skip→SKP	
call→CALL	
return→RET	
compare→CMP	

Conditional branch instructions

BZ	Branch if zero(z=1)
BC	Branch if carry(c=1)
BP	Branch if plus(s=0)

1. **Evaluate X = (A –B)/[C+(D*E)] Using three two one and zero address instruction formats.**

2. **Evaluate X = (A –B*F)*C+D/E Using three two one and zero address instruction formats.**