Q.) What is the meaning of normalization and how to calculate the bias of the exponent value.

Ans: Normalization consists of shifting significand digits left until the most significant bit is nonzero. **a normalized floating-point number's mantissa has no non-zero digits to the left of the decimal point and a non-zero digit just to the right of the decimal point**. Normalization is done by adding A normalized floating point number

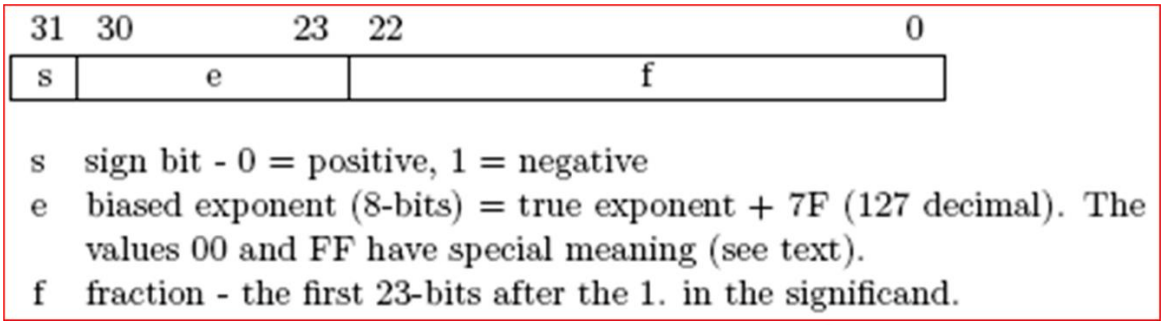
has the form:

1.sssssssssssssss × $2^{eeeeeee}$

We add binary of 127 to true exponent to calculate the the bias of exponent value and normalize our answer, because 23 bit to 30 bit that contain exponent should be positive.

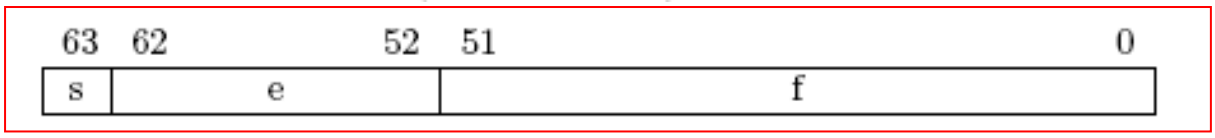
Q.) What is the IEEE floating point representation.

Ans: The IEEE defines two different formats with different precisions: single and double precision. Single precision is used by float variables in C and double precision is used by double variables.



s sign bit - 0 = positive, 1 = negative
e biased exponent (8-bits) = true exponent + 7F (127 decimal). The values 00 and FF have special meaning (see text).
f fraction - the first 23-bits after the 1. in the significand.

$e = 0$ and $f = 0$	denotes the number zero (which can not be normalized) Note that there is a +0 and -0.
$e = 0$ and $f \neq 0$	denotes a *denormalized number*. These are discussed in the next section.
$e = \text{FF}$ and $f = 0$	denotes infinity (∞). There are both positive and negative infinities.
$e = \text{FF}$ and $f \neq 0$	denotes an undefined result, known as *NaN* (Not a Number).



Q.) Differentiate between fixed point and floating-point number representation.

Ans: Integer Representation: (Fixed-point representation): An eight bit word can be represented the numbers from zero to 255 including 00000000 = 0 00000001 = 1 - - - - - - - 11111111 = 255 In general if an n-bit sequence of binary digits an-1, an-2 …..a1, a0; is interpreted as unsigned integer A.

A = $\sum_{i=0}^{n-1} 2^i a_i$

The floating point is always interpreted to represent a number in the following form ±M × R±E . Only the mantissa M and the exponent E are physically represented in the register (including their sign). The radix R and the radix point position of the mantissa are always assumed. A floating-point binary no is represented in similar manner except that it uses base 2 for the exponent. The fraction has zero in the leftmost position to denote positive. The floating-point number is equivalent to M × 2E.

Q.) Explain types of instructions organization and instruction format.

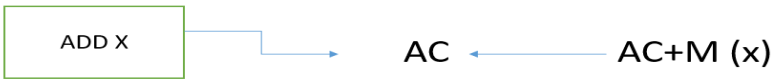
- Computers may have instructions of several different addresses. The no. of address field in the instruction format of a computer depends on internal organization of its registers. Main 3 organizations are as follows:

1. Single Accumulator Organization.
2. General Register Organization.
3. Stack Organization.

Ans:

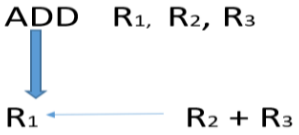
1. Single Accumulator Organization

- In this type of organization that specifies the arithmetic addition is by an assembly language instruction as ADD X, where X is address of operand. The ADD instruction in this case, result in the operation.



2. General Register Organization

- The instruction format in this type of computer needs 3 register address field. Thus, instruction for an arithmetic addition may be written as.



3. Stack Organization

- The stack organization CPU uses stack which uses PUSH and POP operation. These defined operation requires the address field.

For eg. PUSH X
POP X

For the above organization. There are 4 types of instructions-

Types of Instructions

1. Three Address Instruction
2. Two Address Instruction
3. One Address Instruction
4. Zero Address Instruction

1. Three Address Instruction

- Computers with three address instruction format can use each address field to specify either a processor register or a memory operand.
- The program in assembly language that evaluate X= (A+B)*(C+D) as follows:

ADD R_1, A, B = R_1 ← M [A] + M [B]

ADD R_2, C, D = R_2 ← M [C] + M [D]

Mul X, R_1, R_2 = M[X] ← R_1 * R_2

It is assumed that the computer has two processor register. The symbol M[A] denotes the operand at memory address symbolized by A.

- The advantage of three address format is that it results in short program when evaluating arithmetic operations. The disadvantage is that the binary coded instruction require to many bits to specify three address.

3. One Address Instruction

- One address instruction uses an accumulator register for all data manipulation for the purpose of multiplication and division. There is a need for second register and assume that the AC contains the result of all operation. The program is to evaluate: X= (A+B)*(C+D) in 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

2. Two Address Instruction

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

Mov R_1, A = R_1 ← M [A]

ADD R_1, B = R_1 ← R_1 + M [B]

Mov R_2, C = R_2 ← M [C]

ADD R_2, D = R_2 ← R_2 + M [D]

MUL R_1, R_2 = R_1 ← R_1 * R_2

MOV X, R_1 = M[X] ← R_1

4. Zero Address Instruction

- Step organized computes and do not use an address field for the instruction addition and multiplication.
- The PUSH and POP instructions however need an address field to specify the operand that communicate with the stack.
- The program to evaluate the same operation in Zero address instructions as follows:-

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	←	(A+B) * (C+D)
POP X	M[X]	←	TOS

Q.13) Define following terms:

a. Control Memory

b. Micro instruction

Ans:

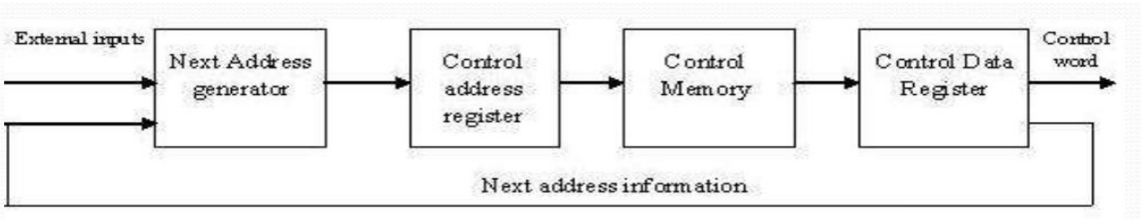


Figure 4: Micro-programmed Control Unit

- The address of micro-instruction that is to be executed is stored in the control address register (CAR).
- Micro-instruction corresponding to the address stored in CAR is fetched from control memory and is stored in the control data register (CDR).
- This micro-instruction contains control word to execute one or more micro-operations.
- After the execution of all micro-operations of micro-instruction, the address of next micro-instruction is located.

COA Question Bank Solutions

Ans 1 ⇒

Computer Architecture	Computer Organization
① Functional description of requirements and design implementation.	about operational attributes are linked together and contribute to realize architectural specification.
② describe what computer does	describe how computer does.
③ Deals with high-level design issues.	deals with low-level design issues.
④ Indicates its hardware	Indicates its performance

Computer Architecture	Computer organization
① For designing computer its architecture is fixed first	Decided after set up of architecture.
② comprises logical functions such as instruction sets, register, data types, and addressing modes.	consists of physical units like circuit designs, peripherals, and adders.
③ coordinates the hardware and software of the system.	handles the segments of the network in a system.

Ans 2⇒ Control unit controls the flow of data between processor and memory and peripherals also, it directs entire system to carry out stored program instructions by generating relevant timing and control signals to all operations in the computer.

It communicates with ALU and memory to instruct which operation is performed and co-ordinates the activities of every devices linked, peripherals, auxiliary storage and other two units of CPU.

Attributes	Hard wired control	Micro-programmed Control
Speed	Fast	Slow
Cost of implementation	More	Cheaper
Flexibility	Not flexible, difficult to modify for new instruction	Flexible, new instructions can easily be added.
Ability to handle complex Instructions	Difficult	Easier
Decoding	Complex	Easy
Applications	RISC Microprocessor	CISC Microprocessor
Instruction set Size	Small	Large.
Control Memory	Absent	Present
Chip Area Required	Less	More

(Pg. 7.07)

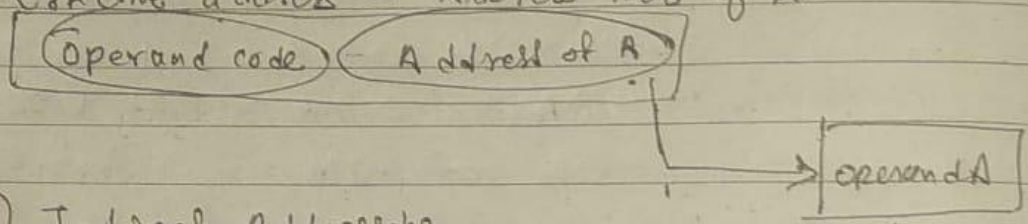
Ans 3⇒ The different ways in which location of operand is specified in an instruction are referred to as addressing modes.

1) Immediate

Operand given explicitly in instruction to accumulator, i.e. no reference to memory.

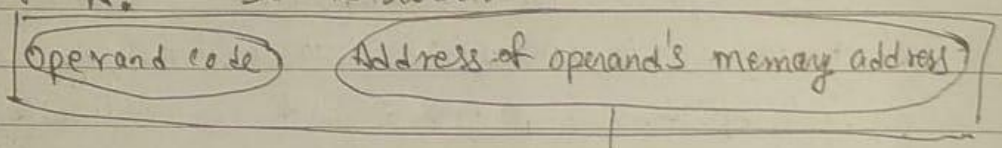
2) Direct Addressing

Effective address = Address field of A.



3) Indirect Addressing

Memory cell pointed by instruction contains address of A. It is slower



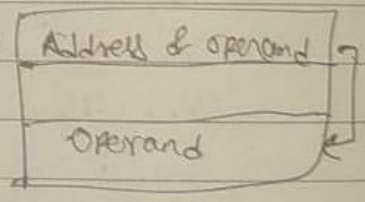
4) Register Addressing

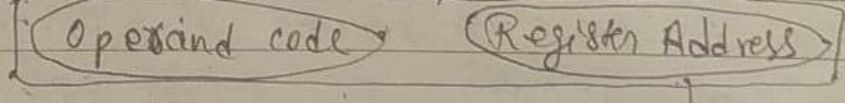
Operand present in register named in address field

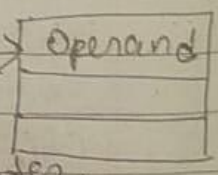
Limited number of registers, very small address field needed.

No memory access, fast execution

Shorter instruction, Faster instruction fetch.



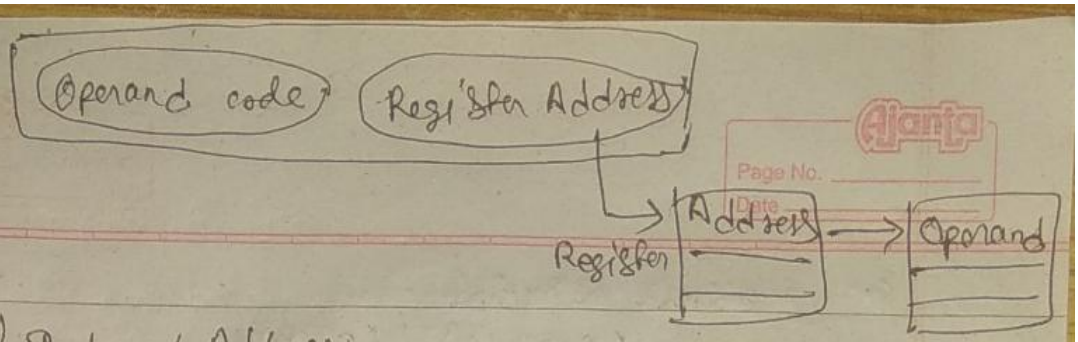




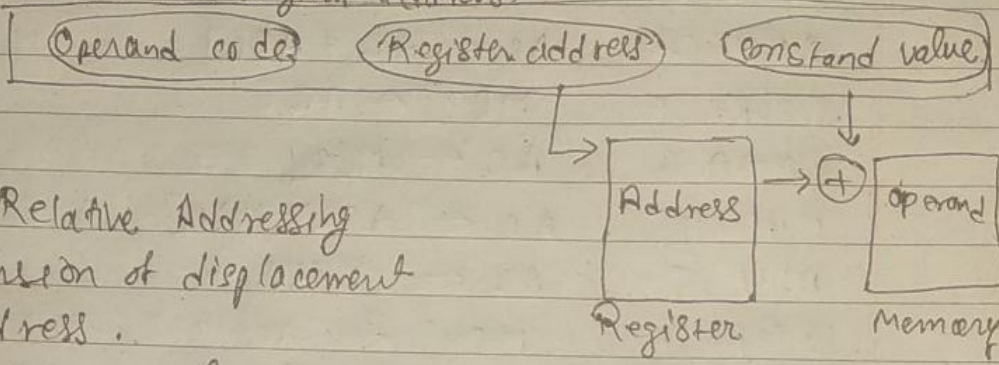
5) Register indirect addressing

Operand is in memory cell pointed by content of register.

Large address space. Fewer memory access than direct addressing.



6) Indexed Addressing
Effective address of operand is allocated after constant intervals from a given address.



7) Relative Addressing
version of displacement address.
Get operand from x byte away from current location pointed by program counter.

Ans 4⇒ $X = (A - B + C) / (G * H)$

a. 3 address instructions.

SUB R_1, A, B = $R_1 \leftarrow M[A] - M[B]$
ADD R_1, R_1, C = $R_1 \leftarrow R_1 + M[C]$
MUL R_2, G, H = $R_2 \leftarrow M[G] * M[H]$
DIV X, R_1, R_2 = $M[X] \leftarrow R_1 / R_2$

b. 2 address instructions

MOV R_1, A = $R_1 \leftarrow M[A]$
SUB R_1, B = $R_1 \leftarrow R_1 - M[B]$
ADD R_1, C = $R_1 \leftarrow R_1 + M[C]$
MOV R_2, G = $R_2 \leftarrow M[G]$
MUL R_2, H = $R_2 \leftarrow R_2 * M[H]$
DIV R_1, R_2 = $R_1 \leftarrow R_1 / R_2$
MOV X, R_1 = $M[X] \leftarrow R_1$

c. 1 address instruction

$X = (A - B + C) / (G * H)$

LOAD A	AC ← M[A]
SUB B	AC ← AC − M[B]
ADD C	AC ← AC + M[C]
STORE T	M[T] ← AC
LOAD G	AC ← M[G]
MUL H	AC ← AC * M[H]
DIV T	AC ← T/AC
STORE X	M[X] ← AC

d. zero address instruction

PUSH A	TOS ← A
PUSH B	TOS ← B
SUB	TOS ← A − B
PUSH C	TOS ← C
ADD C	TOS ← A − B + C
PUSH G	TOS ← G
PUSH H	TOS ← H
MUL	TOS ← G * H
DIV	TOS ← (A − B + C) / (G * H)
POP X	M[X] ← TOS

Ans ⇒ $X = A + B / C * (D + E) - F$

a. 3 address instructions

DIV R_1, B, C = $R_1 \leftarrow M[B] / M[C]$

ADD R_2, D, E = $R_2 \leftarrow M[D] + M[E]$

MUL X, R_1, R_2 = $M[X] \leftarrow R_1 * R_2$

ADD R_1, A, X = $R_1 \leftarrow M[X] + M[A]$

SUB Z, R_1, F = $M[Z] \leftarrow R_1 - F$

b. 2 address instructions

MOV R_1, B = $R_1 \leftarrow M[B]$

DIV R_1, C = $R_1 \leftarrow R_1 / M[C]$

~~ADD~~ MOV R_2, D = $R_2 \leftarrow M[D]$

ADD R_2, E = $R_2 \leftarrow R_2 + M[E]$

MUL R_1, R_2 = $R_1 \leftarrow R_1 * R_2$

ADD R_1, A = $R_1 \leftarrow R_1 + M[A]$

SUB R_1, F = $R_1 \leftarrow R_1 - M[F]$

MOV X, R_1 = $M[X] \leftarrow R_1$

c. 1 address instructions

LOAD B	$AC \leftarrow M[B]$
DIV C	$AC \leftarrow AC / M[C]$
STORE T	$M[T] \leftarrow AC$
LOAD D	$AC \leftarrow M[D]$
ADD E	$AC \leftarrow AC + M[E]$
MUL T	$AC \leftarrow AC * M[T]$
ADD A	$AC \leftarrow AC + M[A]$
SUB F	$AC \leftarrow AC - M[F]$
STORE X	$M[X] \leftarrow AC$

d. Zero address instructions

PUSH B	TOS ← B
PUSH C	TOS ← C
DIV	TOS ← B/C
PUSH D	TOS ← D
PUSH E	TOS ← E
ADD	TOS ← D+E
MUL	TOS ← B/C * (D+E)
PUSH A	TOS ← A
ADD	TOS ← A + B/C * (D+E)
PUSH F	TOS ← F
SUB	TOS ← A + B/C * (D+E) - F
POP X	M[X] ← TOS