Name: Anshika jain

Roll Number: 2K19/SE/012

Branch: SE

Tutorial Sheet – 2

Solution – 1:

X = (A + B * C) / (D – E / F)

a) **Three address instructions**

	Instructions	Comments
(i)	MPY X, B, C	X ← B x C
(ii)	ADD X, X, A	X ← X + A
(iii)	DIV T, E, F	T ← E / F
(iv)	SUB T, D, T	T ← D – T
(v)	DIV X, X, T	X ← X / T

b) **Two Address Instructions**

	Instructions	Comments
(i)	MOVE X, B	X ← B
(ii)	MPY X, C	X ← B x C
(iii)	ADD A, X	X ← X + A
(iv)	MOVE Y, E	Y ← E
(v)	DIV Y, F	Y ← Y / F
(vi)	SUB D, Y	Y ← D – T
(vii)	DIV X, Y	X ← X / T

c) **One Address Instructions**

	Instructions	Comments
(i)	LOAD E	AC ← E
(ii)	DIV F	AC ← AC / F
(iii)	STORE X	X ← AC
(iv)	LOAD D	AC ← D
(v)	SUB X	AC ← AC – X
(vi)	STORE X	X ← AC
(vii)	LOAD B	AC ← B
(viii)	MPY C	AC ← AC x C
(ix)	ADD A	AC ← AC + A
(x)	DIV X	AC ← AC / X
(xi)	STORE X	X ← AC

d) **Zero Address Instructions:**

	Instructions
(i)	PUSH A
(ii)	PUSH B
(iii)	PUSH C
(iv)	MUL
(v)	ADD
(vi)	PUSH D
(vii)	PUSH E
(viii)	PUSH F
(ix)	DIV
(x)	SUB
(xi)	DIV

Solution – 2:

a) **Implied addressing mode:**

   i. COM: Complement Accumulator
   Operand in AC is implied in the definition of the instruction. All register reference instruction that uses an accumulator are implied mode instruction.
   PUSH: Stack push ➔ Operand is implied to be on top of the stack. Zero address instruction in stack are implied mode since the operands are implied on top of stack.

b) **Immediate addressing mode:** To initialize register to a constant value

c) **Register addressing mode:** to implement variables.

d) **Register Indirect addressing mode:** to pass arrays as a parameter because array name is the base address and pointer is needed to point the address.

e) **Direct addressing mode:** to access static data.

f) **Indirect addressing mode:** to implement pointers because pointers are memory locations that store the address of another variable.

g) **PC Relative addressing mode:**

   i. For program relocation at run time i.e., for position independent code
   ii. To change the normal sequence of execution of instructions.
   iii. For branch type instructions since it directly updates the program counter.

h) **Indexed addressing mode:** for array implementation or array addressing, for record implementations

i) **Auto-Increment addressing mode:**

   i. For implementing loops
   ii. For stepping through arrays in a loop
   iii. For implementing stack as push and pop

j) **Auto-Decrement addressing mode:**

   i. For implementing loops
   ii. For stepping through arrays in a loop
   iii. For implementing stack as push and pop

k) **Base Address Register addressing mode:**

   i. For writing relocatable code i.e., for relocation of program in memory even at run time
   ii. For handling recursive procedures