COA

MD SAIF
2100090144

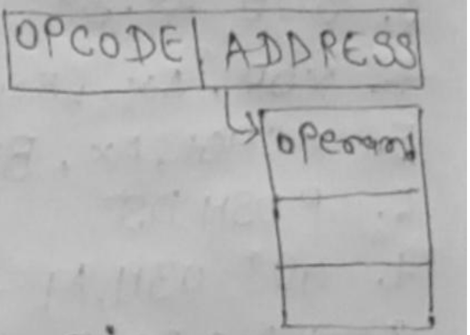
Home Assignment-1

1.) Illustrate the Direct, Register indirect implied addressing modes with suitable examples

Ans) Direct Addressing :- A very simple form of addressing in which the address field contain the effective address of the operand : EA = A

* It requirs only one Memory reference and no special calculation

Exp:- MOV AX [1592, H]
Mov BL [0300, H]

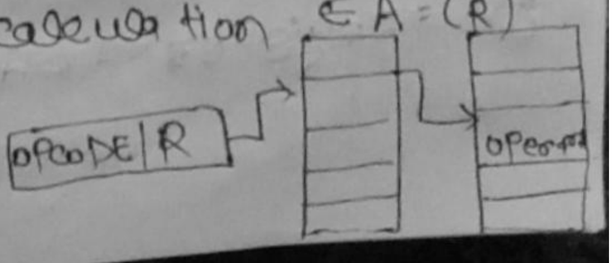


Direct Addressing mode memory

Register Indirect Addressing :-
Register Indirect Addressing is similar to indirect addressing except that address field refers to a register insted of a memory location

* It required only one Memory referen and no special calculation EA = (R)

Exp:- ADD AL, [BX]
Mov AX, [BX]



Implied Addressing Mode : Md Saif
2100090144

It refers to instructions that comprise only on opcode without on operand. The operands are specified implicity in the definition of a instruction.

* mainly used for Zero - address (stack-origized) and one-address (Accumulator origanized)

Q2) Identify which addressing modes do the following instruction belong.

1. MOV AX, [SI]
2. INC [3000H]
3. SUB AX, BX
4. XOR, AX, [5000H]
5. XCHG, AX, BX
6. PUSH DS
7. OUT 03H, AL
8. INC [BX]

-> 1) Register Indirect Addressing mode
2) Direct Addressing mode.
3) Register Addressing Mode.
4) Direct Addressing Mode.
5) Register Addressing mode
6) Register Addressing Mode.
7) Register Addressing Mode.
8) Register Indirect Addressing mode.

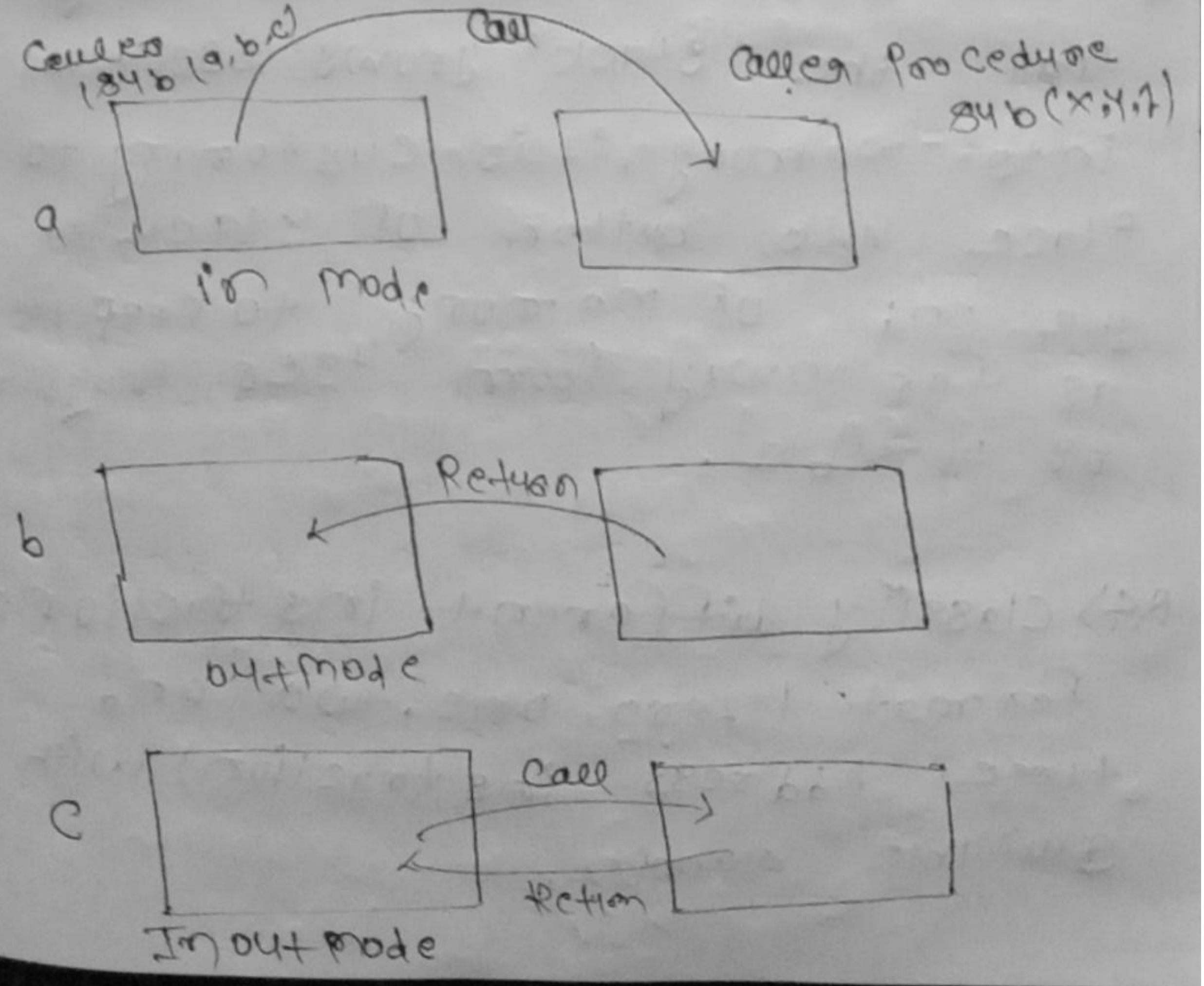
Md. Saif
2100090144

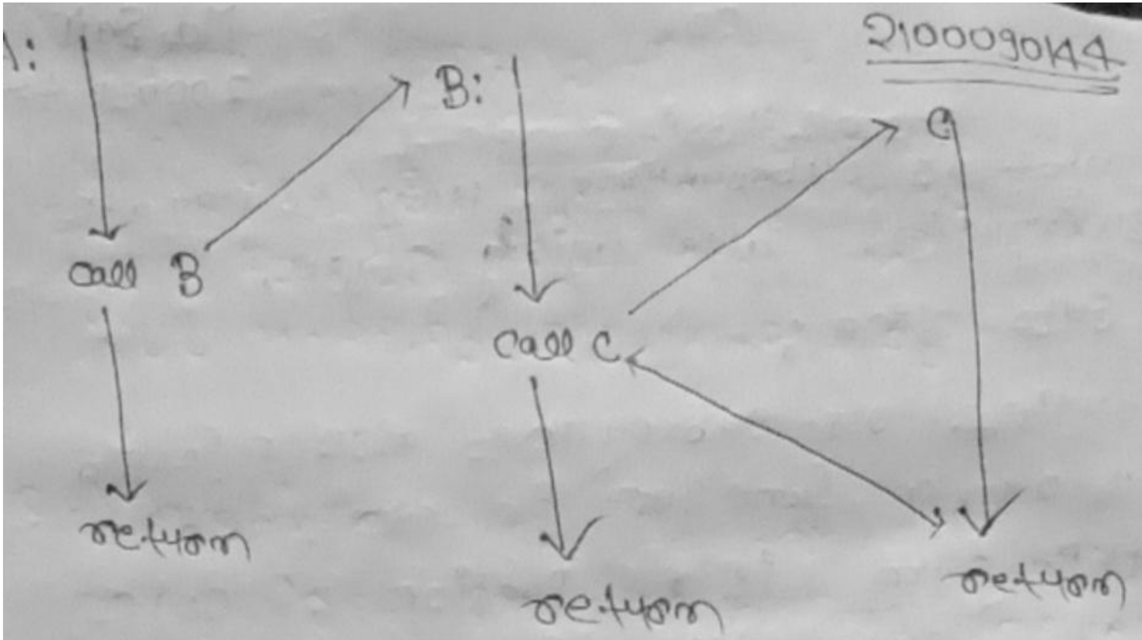
3) What is subrouttine? How to handle subroutine calls using stack.

→ Subrontine :-

Aroutine or subroutine also referred to as a function procedure and subprogram is a portion of code that may be called executes only -where in a program.

The executive program maintains a stack to heep control the flow of instructions. A program is often divides up into a main loop that calls a number of subroutines. Each subroutine has a specific task





* Above A calls B which call c

* Must even work when B is A

* The stack is a area of memory identifies by the programmer for temporary storage of information.

* The stack is LIFO structure given that the stack grows backwards into memory, it is customary to place the botton of stack at the end of Memory to keep it as far away from user programs as possible.

Q4> Classify different instructions format (zero, one, two, and three Address instruction) with suitable example.

→Ⓐ Zero Address Instructions·

Expression 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
MUL	TOS← (C+D)
DIV X	M[X]← TOS

PUSH A
PUSH B
ADD
PUSH C
PUSH D
ADD
MUL
STORE

(B) One Address Instructions

LOAD	A	A ← M[A]
ADD	B	A ← 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

C) Two Address Instructions :

MOV	R_1, A	R_1 ← M[A]
APP	R_1, B	R_1 ← R_1 + M[B]
MOV	R_2, C	R_1 ← M[C]
ADD	R_3, D	R_1 ← R_2 + M[D]
MUL	R_1, R_2	R_1 ← R_1 * R_2
MOV	X, R_1	M[X] ← R_1

b) Three Address Instructions:

OPCODE	ADDRESS1	ADDRESS2	ADDRESS3

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

Q.5) Illustrate the Arithematic, Logical, Branch and control instructions with suitable examples.

→ (A) Arithematic Instructions:

- Add : compute sum of two operands

  eg :- add al, 07h

  add ax, bx

- subtract : compute difference of two operands.

  eg :- sub, ah, 05h

  sub, ah, al

- Multiply : Compute Product at two operands.

  e.g :- MUL ax, 1234h

  MUL bx, 100h

  MUL bx,

- Divide: compute Quotient.

  eg :- mov ax, 800~~Quotient~~

  Mov Cx, 100h

- Absolute:- Replace operands by its original value.

  eg:- abs, RT, RA

- Negate:- Change sign of operand.

  eg:- neg RT, RA

- Increment:- Add 1 to operand

  eg:- inc A

- Decrement:- subtract 1 from operan

  eg:- dec A.

Logical Instructions:-

* AND:- Performs the logical operation And bitwise

eg:- AND, AL, 0Fh

AND, AL, 01h

* OR:- Performs the logical operation OR bitwise

Eg:- OR AX, 08h

OR AM, 05h

* NOT:- Performs logical Not bitwise

Eg:- operand : 0101.0011

After NOT:→ operands 1:1010 1100

* Exclusive OR :- Performs the specified logical operation Exclusive-OR bitwise

eg:- Operand 1 : 0101, Operand 1 : 0110

After XOR → operand 1 : 0110

* Test :- Test specified condition sets flag (s) based on outcome.

eg:- Test AL, 01H

JZ - Even - Number

* compare :- Make logical or arithmetic comparision flag (s) based on outcome

eg:- CMP destination. source

CMP DX, 00 :- compare the DX value with zero J E L7 :- if yes, then Jump to lbel 17.

* set control variables :- class of instructions ~~[illegible]~~ to set controls for Protection purposes, interrupt handing timer control etc.

* shift :- left (Right) shift operand introducing constanct at end.

exp:- SHR. AX, 2

SHL, AX, 2

- Rotate :- left (Right) shift operation with wrap around end.

  eg:- ROR AH, L1
  ROL AH, L1

System control :-

system control instructions are those which are used for system setting it can be used only in privileged state.

- Typically these instructions are reserved for the use of operating system.

Transfer of control :-

The most common transfer of control operations found in instruct set are :

1) Branch
2) skip
3) Procedure call.

BRPX : Branch to location x if result is Positive.

BRNX : Branch to location x if result is negative.

BRZ : Branch to location x if result is zero

BRDX : Branch to location if overflow occurs