COMPUTER ORGANIZATION

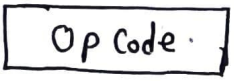
D.Neha
245319733015
CSE-A

ASSIGNMENT -5

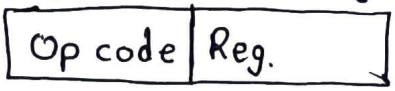
1) Explain the different types of instruction formats used in 8086 microprocessor.

A) The 8086 instruction format varies from 1-6 bytes in length. The instruction formats for 1-6 bytes instructions. The displacement and operands maybe either 8-bits or 16-bits long depending on instruction. The op code and the addressing mode is specified using the first two bytes of an instruction.

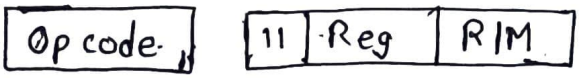
One byte instruction



One byte instruction register mode

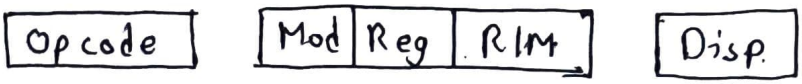


Register to register



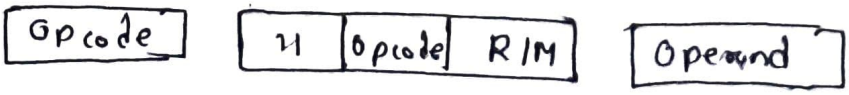
Register to/from memory with no displacement

Register to/from memory with displacement (8-bit)

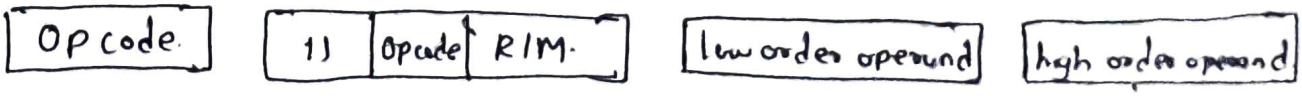


Register to/from memory with displacement (16-bit)

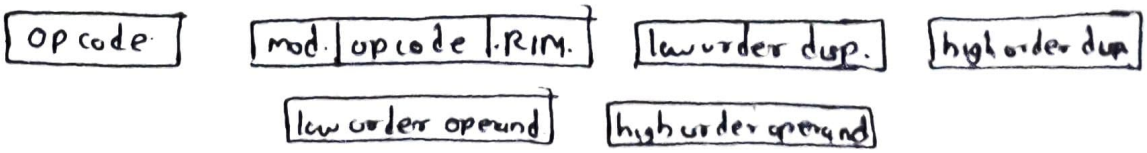
Immediate operator to Register (8 bit)



Immediate operand to register (16 bit)



Immediate operand to memory with 16 bit displacement



Instruction format in 8086 microprocessor:

For every instruction that is executed in 8086 microprocessor, an instruction format is available that is the binary representation of the instruction

This instruction format can be coded from 1–6 bytes depending upto the addressing modes used for instructions.

The general instruction format that is most of the instructions of 8086 microprocessor follow is

Opcode (6 bits)	D (1-bit)	W (2-bits)	MOD (2-bits)	Reg (3-bits)	R/M (3-bits)	lower order bits of displacement	higher order bits of displacement

2) Write a program using call instruction

A) Call Instruction: It calls a procedure

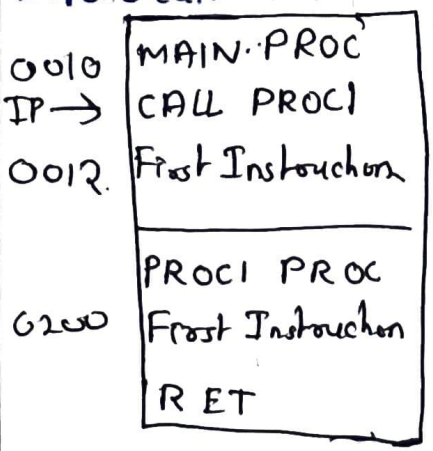
* pushes offset of next instruction on stack
* copies the address of the called procedure into IP (instruction pointer)

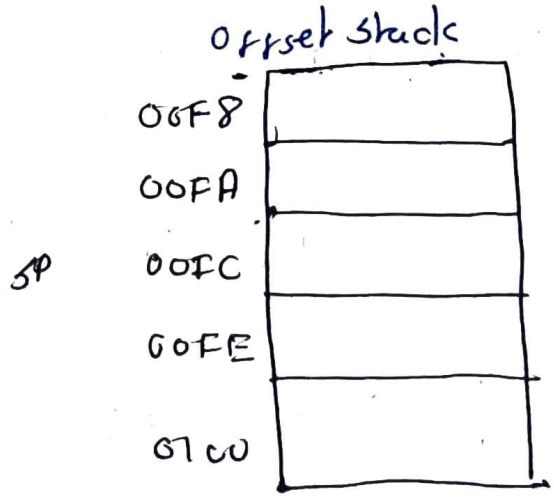
The call instruction is required to call subroutines programs from main program. The call ins resides in call area of program.

Procedure call

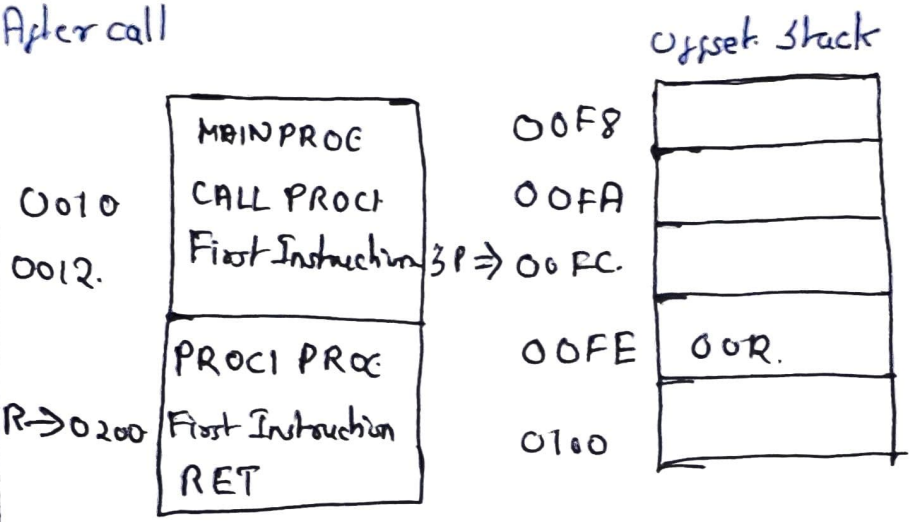
```
main proc
CALL PROC1
First Instruction
```

Before call





After call



Program:

Write an ALP to make a procedure print that print the following shape.

```
.
. .
. . .
. . . .
. . . . .
```

☆ data.

```
str1 db "*$"
str2 db 13, 10, "**$"
str3 db 13, 10, "***$"
str4 db 13, 10, "****$"
str5 db 13, 10, "*****$"
```

★ Code : code segment

```
main proc                : main procedure
mov ax, @data
mov ds, ax
call print
call exit
end p                    : end of main procedure
print proc
lea dx, str1
mov ah, 09h
int 21h
lea dx, str2.
```

```
int 21h
dea dx, str3.
int 21h
dea dx, str4.
int 21h
dea dx, str5
int 21h
ret
print endp
exit proc
mov ah, 4ch
int 21h
ret
```

Process of Call instruction

* The address of next instruction that exists in the caller program is stored in the stack.
* The instruction queue is emptied for accomodating the instructions of the procedure.
* Then the contents of instruction pointer (IP) is always changed with the address of first instruction of procedure
* The subsequent instructions of procedure are stored in the instructions queue of execution.

Syntax of CALL Instruction

CALL subprogram - name

Mainline or calling program

↓

Procedure → Procedure instruction

↓

Next machine ← RET instruction

Instruction ↓

4). Give the classification of 8086 instructions.

A) An instruction is binary pattern designed inside a microprocessor to perform a specific function

→ The entire group of instruction that a microprocessor supports called instruction set.

→ 8086 has more than 20,000 instructions

Classification of Instruction set:

→ data transfer instructions
→ Arithmetic instructions
→ Bit manipulation instructions
→ Program execution transfer instructions
→ string instructions
→ Processor control instructions

1) Data Transfer Instructions:

⇒ These instructions are used to transfer data from source to destination.
⇒ The operand can be constant, memory location, register or I/O port address.

→ MOV, Des, Src
→ PUSH operand
→ POP Des
→ XCHG Des, Sr
→ IN Accumulator, Port Address
→ OUT Port Address, Accumulator
→ LEA Register, Src;
→ LDS Des, Src
→ LES Des, Src;

- LAHF
- SAHF
- PUSH F
- POPF

2) Arithmetic Instruction:

ADD Des, Src :-

It adds a byte to byte or word to word

It affects AF, CF, OF, PF, SF, ZF flags

ex:- ADD AL, 74H
ADD Dx, Bx.
ADD Ax, [Bx]

SBB, Des, Src: It subtracts the two operands and also borrow from result

ex:- SBB AC, 74H
SBB Dx, Dx
SBB Ax, [Bx]

→ INC SRC
→ DEC SRC
→ AAA (ASCII Adjust After addition)
→ AAS (ASCII Adjust after sub)
→ AAM (ASCII Adjust After Multiplication)
→ AAD (ASCII Adjust After Division)
→ DAA (Decimal Adjust After Addition)
→ DAS (Decimal Adjust After subtraction)
→ NEG Src
→ cmp Des, Src
→ MUL Src
→ Div Src.

3) Bit Manipulation Instructions:

These instructions are used at bit level.
These instruction can be used for

* Testing a zero bit
* Set or reset a bit
* Shift bit across registers

→ NOT Src
→ AND Des, Src
→ OR Des, Src
→ XOR Des, Src
→ SHL Des, Count

→ SHR Des, Count
→ ROL Des, Count
→ ROR Des, Count

4) Program execution Transfer Instructions

* These instructions cause change in sequence of the execution instruction.
* This change can be through a condition or sometimes unconditional).
* These conditions are represented by flags

→ CALL Des
→ RET
→ JMP Des
→ Jxx Des
→ LOOP Des

Condition Jump Table:

Mnemonic	Meaning	Jump Cond
JA	Jump If Above	CF=0, ZF=0
JAE	Jump if above/equal	CF=0
JB	Jump if below	CF=1
JBE	Jump if below/equal	CF=1, ZF=1
JC	Jump if carry	CF=1
JE	Jump if equal	ZF=1
JNC	Jump if not carry	CF=0
JNE	Jump if not equal	ZF=0
JNZ	Jump if not zero	ZF=0
JPE	Jump if parity even	PF=1
JPO	Jump if parity odd	PF=0
JZ	Jump if zero	ZF=1

5) String Instructions:

* String is assemble language is just a sequentially stores bytes or words.
* These are very strong set of string instruction of 8086

→ CMPS Des, Src

→ SCAS String

→ MOV / MOV SB / MOVSB

→ REP (Repeat)

6) Processor Control Instructions:

* These instructions control the processor itself
* 8086 allows to control certain control flags.
  - causes the processing in certain direction
  - processor synchronization if more than one microprocessor attached

→ STC → It sets the cherry flag to 1.

→ CLC → It clears the carry flag to 0.

→ CMC → It complements the carry flag.

→ STD → It sets the direction flag to 1

→ CLD → It clear the direction flag to 0.

3) Write a program for a given arithmetic operation.

$Y = A * (B + C) / D$ where A, B, C, D are memory locations used to store input data and starts from 5000H. Y is the memory location used to store the result which is 5010

Sol] $Y = (A \oplus B) * (C - D)$

AL. CL

A → 5000
B → 5001
C → 5002
D → 5003
Y → 5004, 5005
(A+B) = AL

```
MOV AL, [A]
MOV BL, [B]
ADD AL, BL
MOV CL, [C]
MOV DL, [D]
SUB CL, DL      - (CL - DL) = CL
MUL CL          (AL * BL = AX)
MOV [Y], AX.
```

5) Write a assembly program to sort 100 numbers which are stored in 5000 location

Sol

```
      MOV  DL, 03
L3:   MOV  CL, DL
      MOV  SI, 5000
L2:   MOV  AL, [SI]
      CMP  AL, [SI+1]
      JC   L1
      XCHG AL, [SI+1]
      MOV  [SI], AL
L1:   INC  SI
      DEC  CL
      JNZ  L2
      DEC  DL
      JNZ  L3.
```