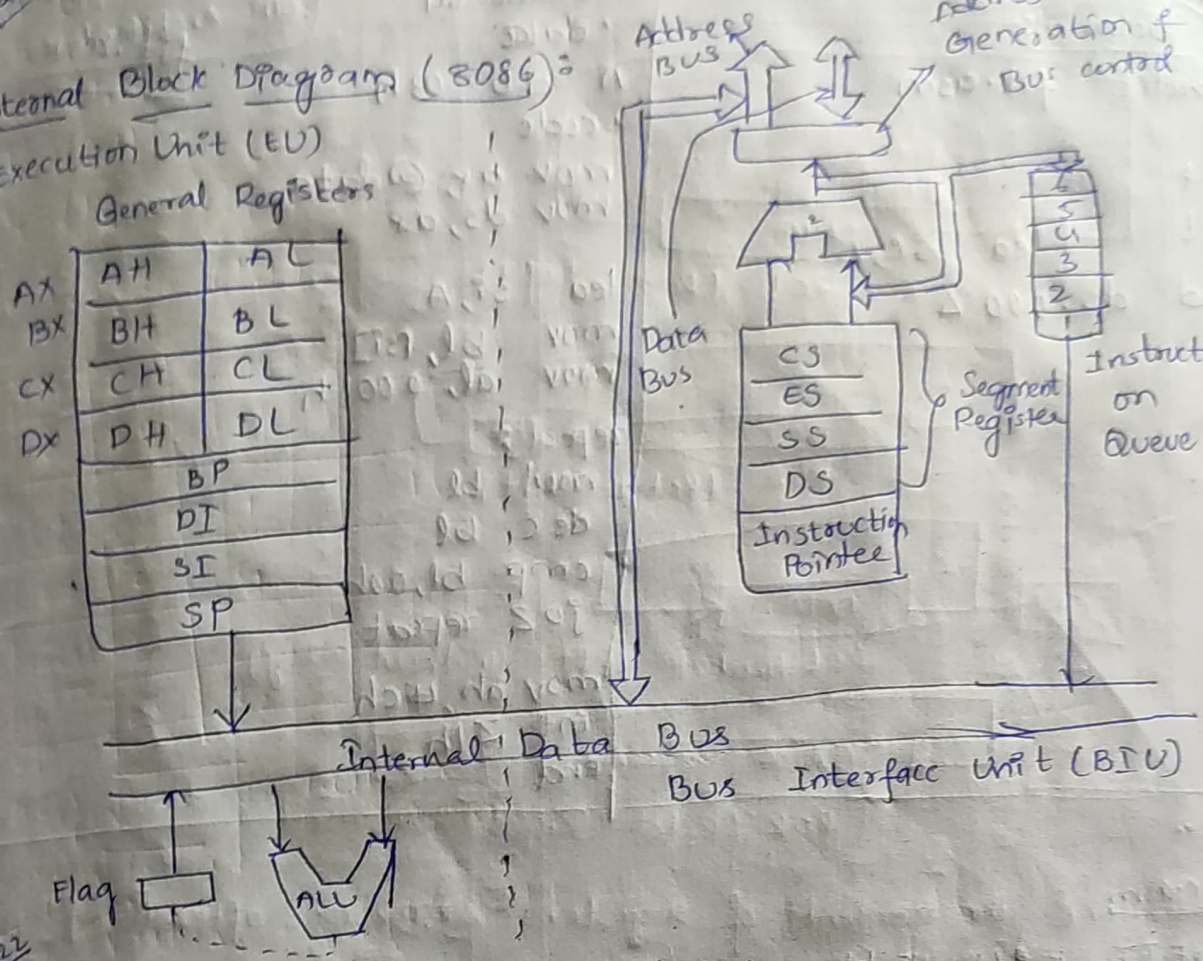
5/1/22

Internal Block Diagram (8086):



6/1/22

Open blank file: mount c c:\8086 try.asm

Z:\> c:

c:\>cd 8086

c:\8086>edit sam.asm
filename

>masm sam.asm

three files will be generated.

1) object file
2) source
3) Cross-reference

>link sam.obj
(exe file, list file, libraries)

>debug sam.exe
-t
-t
-quit

>edit sam.asm

sam.asm

```
.model small
.stack 64h
.data
a equ 05h
b equ 03h
.code
start: mov ax,@data
       mov ds,ax
       mov al,a
       mov bl,b
       add al,bl
       end
```

→ change add to sub

13/1/22

Factorial:-

>edit fit.asm

```
-t
-t
-d ds:000A
```

```
.model small
.stack 64
.data
A db 03h
.code
mov ax, @data
mov ds, ax

lea si, A
mov bl, [si]
mov al, 0001h

repeat:
mul bl
dec bl
cmp bl, 00h
jnz repeat

mov ah, 4ch
int 21h
end
```

Price.asm

```
PROFIT EQU 15H
ARRAYS SEGMENT
    COST DB 20H, 28H, 15H, 26H, 19H, 27H, 61H, 29H
    PRICES DB 8 DUP(0)
ARRAYS ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:ARRAYS
    START: MOV AX, ARRAYS
           MOV DS, AX
           MOV CX, 0008H
           MOV BX, 0000H
    DO_NEXT: MOV AL, COST[BX]
             ADD AL, PROFIT
             DAA
             MOV PRICES[BX], AL
             INC BX
             DEC CX
             JNZ DO_NEXT

CODE_ENDS
    END START
```

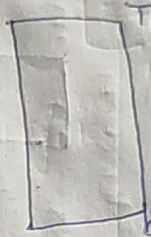
```
masm price.asm
link price.obj
debug price.exe

-t
-t
-d ds:0000
```

Write an ALP to find largest no. in an array

Input: count → 4

A[0] = 03
A[1] = 05
A[2] = 09
A[3] 08

We know that data are stored in data segment



lit.asm

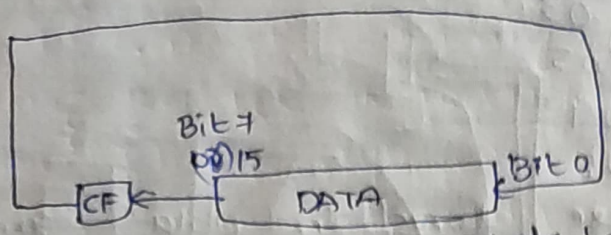
```
.model small
.stack 64
.data
.code
start: Mov ax, @data
       Mov ds, ax
       Mov si, 3000h
       Mov di, 3500h offset
       Mov cl, [si]
       mov ch, 00h
       Inc si
       Mov al, [si]
       dec cl
       Inc si
next: Cmp al, [si]
      Jc/Jnc fit
      largest
        Mov al, [si]
fit: Inc si
     loop next
     Mov [di], al
     hlt
     end
```

commands:

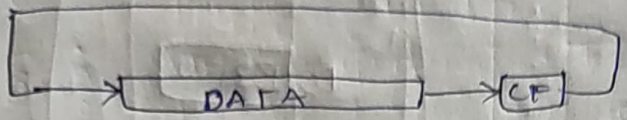
```
masm lit.asm
link lit.obj
debug lit.exe
-t
-t
mov cl, [si]
-e ds:3000
076C:3000 36.04 20.05 1F.00
   EF.09 36.06
-d ds:3000 (display data)
-e ds:3005 (To change value of 3005)
CX=0000
-d ds:3500
076A - segment register
```

10M - Control flow
20M - Architecture 8086

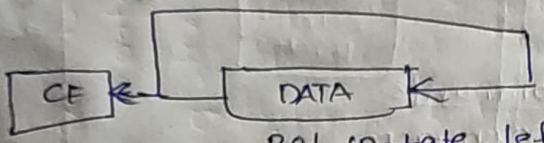
Rotate Instructions



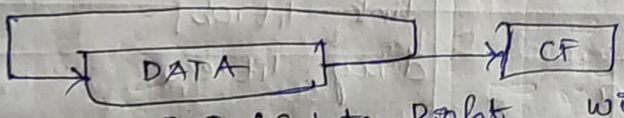
RCL (Rotate left)



RCR (Rotate Right)



ROL (Rotate left without carry)



ROR (Rotate Right without carry)

Determine the contents of registers AX, BX, CX after the following program is run.

	AX	BX	CX
MOV CL,3	----	----	--03
MOV AX,7FFH	007FF	----	--03
MOV BX,0505H	007F	0505	--03
ROL AX,CL	03F8	0505	--03
AND AH,BH	01F8	0505	--03
OR BL,AL	01F8	05F8	--03

AX - 16 bit

0000 0000 0111 1111

CF 0 ← 0000 0000 1111 1110 - 1st rotation (CL = 3 so 3 rotations)

0 ← 0000 0001 1111 1100 - 2nd rotation

0 ← 0000 0011 1111 1000 - 3rd rotation

0 3 F 8

AH = 03 = 0000 0011

BH = 05 = 0000 0101

(AND) 0000 0001

0 1

AL = F8 = 1111 1000

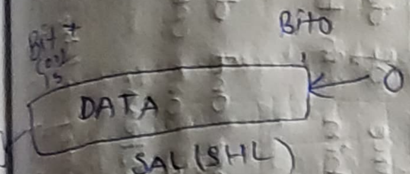
BL = 05 = 0000 0101

(OR) 1111 1101

F D

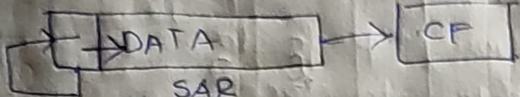
0 1 2 3 4 5 6 7 8 9 10-A 11-B 12-C 13-D 14-E 15-F

3) Shift Instructions:

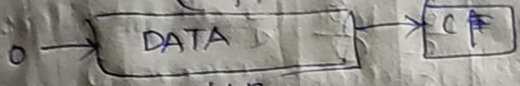


SAL (SHL)

word - 16 bit
nibble - 4 bit



SAR
(Shift Arithmetic Right)



SHR

C = 12 (1100

1 - (-ve)
0 - (+ve)

Consider a data byte (8 bit) C_0H

7	6	5	4	3	2	1	0
(1)	1	0	0	0	0	0	0

sign ← (SAL)

7	6	5	4	3	2	1	0
(1)	0	0	0	0	0	0	0

← 0

1's complement = 0111 1111
+1
1000 0000
128 0
$2^7 = 128$
⇒ -128

C_0H → 00111111
+1
01000000
-64 0
$2^6 = 64$

⇒

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0

(SAR)
(division by 2)
-64(D)

7	6	5	4	3	2	1	0	CF
1	1	1	0	0	0	0	0	0

0 0 0 1 1 1 1 1
+1
0 0 1 0 0 0 0 0 (2^5
= -32 D

*What is the content of register BL after the following instructions are executed?

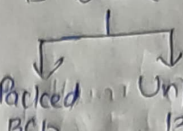
Instructions	BX	CX
Mov BL, B2H	B2	—
Mov CL, 2	B2	02
SAR BL, CL	ECH	02

B2 = 1 0 1 1 0 0 1 0 → 0
1st = 1 1 0 1 1 0 0 1 → 1
2nd = 1 1 1 0 1 1 0 0
E C (12)
H

2/2/22

Arithmetic Adjust Instructions

* Mostly Hexadecimal input is used
* In some, the data operated is BCD (or) ASCII



Ex: 99 → 1001 1001 (Packed BCD) | 09 → 0000 1001 (Unpacked BCD)

A → 1010 (10)
B → 1011 (11)
C → 1100 (12)

```
MOV AL, 7      AL (0007)
MOV BL, 5      (BL) (0005)
ADD AL, BL     AL = 0CH
```

7+5=12

Unpacked BCD: AX = 01 | 02 (AH | AL)

AAA – ASCII Adjust for Addition

```
MOV AH, 0
MOV AL, 7
MOV BL, 5
ADD AL, BL  ⇒ AX = 00 | 0C (AH | AL)
AAA         AX = 01 | 02
ADD AX, 3030H → AX = 3132H
```

- ✓ AAS
- ✓ AAM (ASCII adjust for Multiplication)
- AAD (ASCII adjust for division)

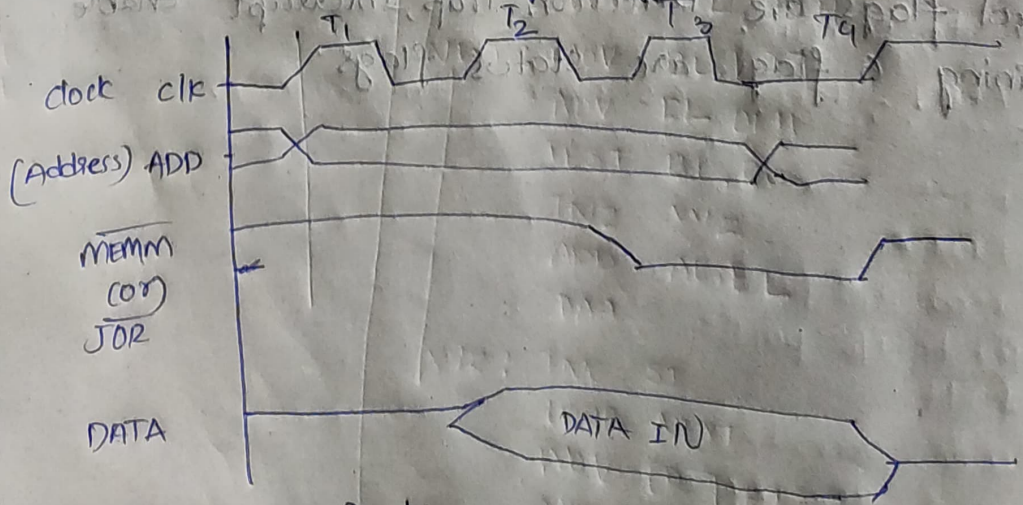
7/2/22 Qn1 3×10=30M

1 a) Internal Block Diagram of 8086 (EU, BIU)

b) Timing diagram for MEMR [3 bus architecture]

(or)

Min. Mode in 8086 Timing Diagram



2 a) Write ALP for finding smallest number from an array

b) Write ALP to find sum of n number

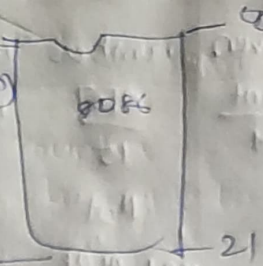
3a) i) Signal Pin description - TEST (5m)

Test PIN

INTA

ALE

ii) 8-bit data, apply SAL (5m)



b i) Repeat until Flowchart (5m)

ii) Apply SAL for 8-bit data (5m)

Extra: *Addressing modes

*Assembler Directives

*Logical to Physical address translation

*~~NOT~~ NEG instruction DAA

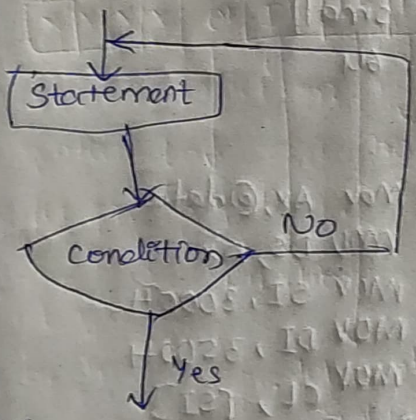
Rotate Instructions

*Ascii adjust instructions

*Memory Banking scheme.

For

Repeat Until Flowchart:-



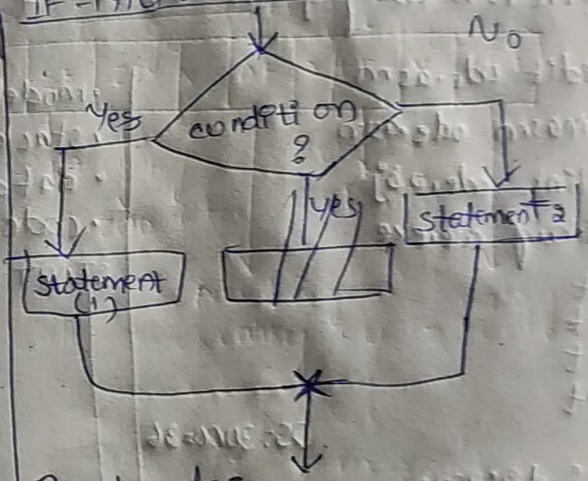
Pseudocode:

```
REPEAT
    STATEMENT(S)
UNTIL CONDITION
```

Example:

```
REPEAT
    GET DATA SAMPLE
    ADD 7
    STORE THE RESULT IN MEMORY
    WAIT 1 HR
UNTIL 24 SAMPLES TAKEN
```

IF-THEN Flowchart:-



Pseudocode:-

```
IF condition THEN
    STATEMENT(S) 1
ELSE
    STATEMENT(S) 2
```

IF Room Temperature ~~LESSTHAN~~

IF Room Temperature LESSTHAN SET-POINT THEN

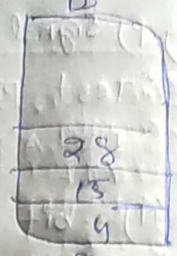
TURN ON FURNACE

ELSE

TURN OFF FURNACE

c) Sum of even numbers in a given series

Input	count	A[0]	A[1]	A[2]	A[3]
	4	15	28	07	08
		Not	Even	Not	Even




```
  0001 0101
  0000 0000
  0000 0001
  ZF = 0
  (Odd No)
```

```
  0010 1000
  0000 0000
  0000 0000
  ZF = 1
```

28+08 =

```
  0010 1000
  0000 1000
  0011 0000
     3    0
```

= 30 H

```
  0011 0000
  0000 0110
  0011 0110
```

There is a carry from D_3 to D_4

Even - JNZ

```
-edit ode.asm
-masm ode.asm
-link ode.obj
-debug ode.exe
-t
-t
-t
-t
```

DS:3000=36

```
-e ds:3000
36.04 20.15 17.28 FF.07 36.08
-t
-t
-t
```

```
.model small
.stack 64
.data
.code
START: MOV AX,@data
       MOV DS,AX
       MOV SI,3000H
       MOV DI,3500H
       MOV CL,[SI]
       INC SI
       MOV CH,00H
       MOV AL,00H
NEXT:  MOV BL,[SI]
       MOV DL,01H
       TEST BL,DL
       JNZ XYZ
       ADD AL,BL
       DAA
XYZ:   INC SI
       LOOP NEXT
       MOV [DI],AL
       MOV AH,4CH
       INT 21H
       END
       END
```