ElCoM
ELECTRICAL COMPUTER MECHATRONICS

تقدم لجنة ElCoM الاكاديمية

تلخيص لمادة:

مختبر لغة تجميع ومعالجة دقيقة

جزيل الشكر للطالبة:

مروة حلوة

Microprocessors Lab

Mid exam

exp 1 : Review

① How to enter debug :-

ⓐ 32 bit : start → programs → Run → ↵ → debug → ↵

② system commands :-

Base offset
0100 : 0000

⇒ 0100 : 0000 (00 at 00) (00 at 01) (11 at 02) (00 at 03) - (22 at 04) (66 at 05) (88 at 06) (00 at 07)

Memory Management commands :

① D (Dump) : displays memory on the screen
D add
D range

② E (edit/enter) :

E 10 , 20 , 40 , 50 , 30
(E → address; 10 → start; 20 , 40 , 50 , 30 → values)

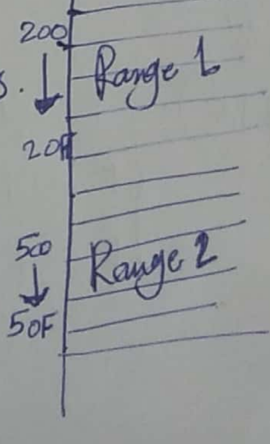
③ F (Fill) :

F 10, 20, 11, 22, 33, FF, 'A' ⇒ يكرر القيم .
(10, 20 → Range; 11, 22, 33, FF, 'A' → values)

④ C (compare) :

C 200, 20F , 500 ⇒ display the different values .
(200 → start range 1; 20F → end range 1; 500 → start range 2)

Range 1 Logical add	value1	value 2	Range 2 Log. add
073F : 0100	B4	00	073F : 0200
⋮	⋮	⋮	⋮



الممسوحة ضوئيا بـ CamScanner

⑤ M (Move): copies a block of data

M 100, 105, 110 ⇒ move DS: 100-105 to DS: 110

⑥ S (Search): يرجع العنوان الذي يبحث عنه address

S 100, 1000, CD, 20
Range — Search sequence.

Assembler Commands:

① A (Assemble):
A address. ⇒ it is assumed to be an offset from CS if only give offset

A 100 ⇒ CS: 100H.
A ⇒ current location.

② U (Unassemble): translate memory into assembly language machine code
(disassemble 32 bytes).
default

Program Control Commands: (code segment).

① G (Go): execute the program in memory.

G breakpoint.
G = startadd, endadd.

② T (Trace): execute number of instructions at either current CS:IP or optional add.

T count
T = add, count.
CS: add.

③ R (Register):
→ display content of one reg, allow edit
→ display all reg, flags, next inst, to be executed.
→ display all 8 flag setting.

CamScanner الممسوحة ضوئيا بـ

3

④ RF [Press enter] ZR → to change any single flag

→ display the flags.

Flag mnemonics :- (orderd).

set or clear	OV	DN	EI	NG	ZR	AC	PO	CY
	NV	UP	DI	PL	NZ	NA	PE	NC

R command ⇒ display next inst

machine inst

0F3F : 0100 41 INC CX

- 0F3F : CS value (seg)
- 0100 : IP value (offset)
- 41 : machine inst
- INC : Assembler mnemonic
- CX : operand

⑤ Q (Quit) : quit debug & return to DOS.

Addressing Mode

- Reg. operand
- immediate operand
- Direct
- Reg. indirect
- Based index
- Reg. relative
- Based relative plus index

Instruction Set :-

① Arithmatic Instruction :

① Add op1, op2 ⇒ op1 = op1 + op2

② sub op1, op2 ⇒ op1 = op1 - op2

③ Mul operand ⇒ byte: AX = AL * op / word: (DX AX) = AX * op

operand: Reg, mem

④ Div operand ⇒ byte: AL = AX / op, AH = remainder / word: AX = (DX AX) / op, DX = remainder

operand: Reg, mem (word / byte)

الممسوحة ضوئيا بـ CamScanner

② Data transfar Instructions :

① General purpose instructions :

MOV : copy data from source to dest

MOV D, S (S) → (D).

② stack Instructions :

ⓐ Push : write word into stack
Push

Push (16-bit reg) ⇒ ① SP = SP - 2
② 16-bit reg → SS : SP

ⓑ Pop : read word from stack to dest. reg.

Pop (16- bit reg) ⇒ ① SS : SP → 16-bit reg.
② SP = SP + 2.

③ string instructions : depend on direction flag.

CLD instruction ⇒ D = 0
STD instruction ⇒ D = 1

ⓐ LODS

LODB

Df=0 → AL = OS:[SI], SI = SI + 1

Df=1 → AL = DS:[SI], SI = SI - 1

LODW

Df = 0 → AL = DS:[SI], SI = SI + 2

Df = 1 → AL = DS:[SI], SI = SI - 2.

ⓑ STOS

STOSB

Df=0 → ES:[DI] = AL, DI = DI + 1

Df=1 → ES:[DI] = AL, DI = DI - 1.

STOSW

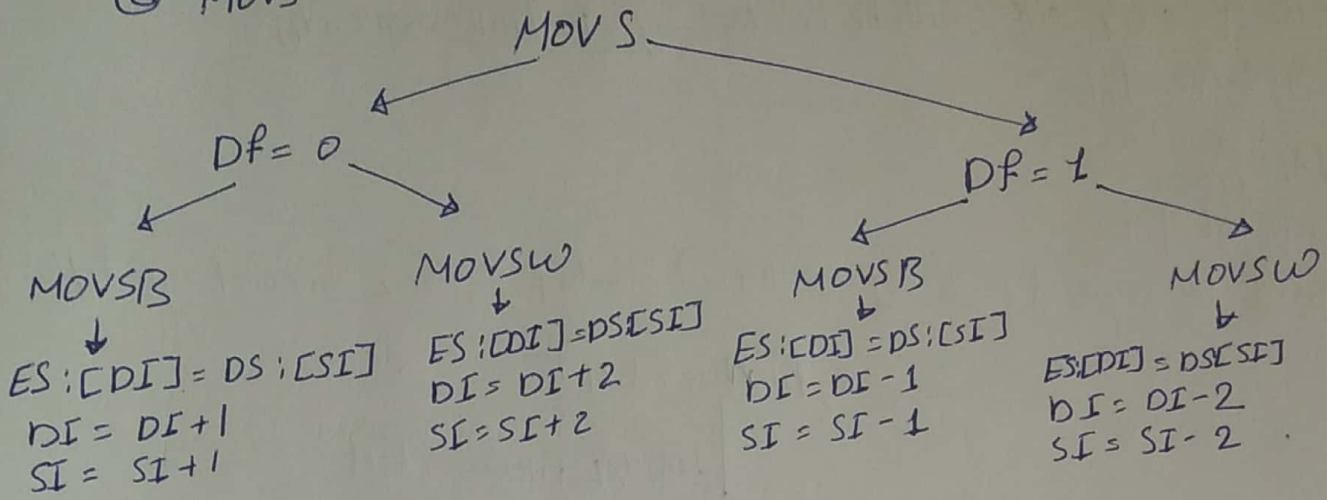
Df=0 → ES[DI] = AL, ES[DI+1] = AH, DI = DI + 2.

Df=1 → ES:[DI] = AL, ES:[DI+1] = AH, DI = DI - 2

الممسوحة ضوئيا بـ CamScanner

5

© MOVS



@ REPEAT (REP) : repeat instruction(s) until the contents of register CX become equal zero, useful in: MOVS & STOS

examples in manual.

exp 2 : Developing Assembly Language & executing using Emu & MASM

* MASM :-

① Assemble the source File (.ASM file).

open DOSbox → go to D:/8086> → MASM /L exp2.asm.

* (Assmble the source file into a machine language object file).
* (MASM checks the source file for syntax errors

② object file (.obj).

* before feeding the .obj file into link, all syntax errors must be corrected.

③ List file (.lst)

* optional file
* very useful for programmers ? it lists all the opcodes & offset add & errors that MASM detected.

EICoM
CamScanner الممسوحة ضوئيا بـ

④ executable file & MAP file (.exe & .map)

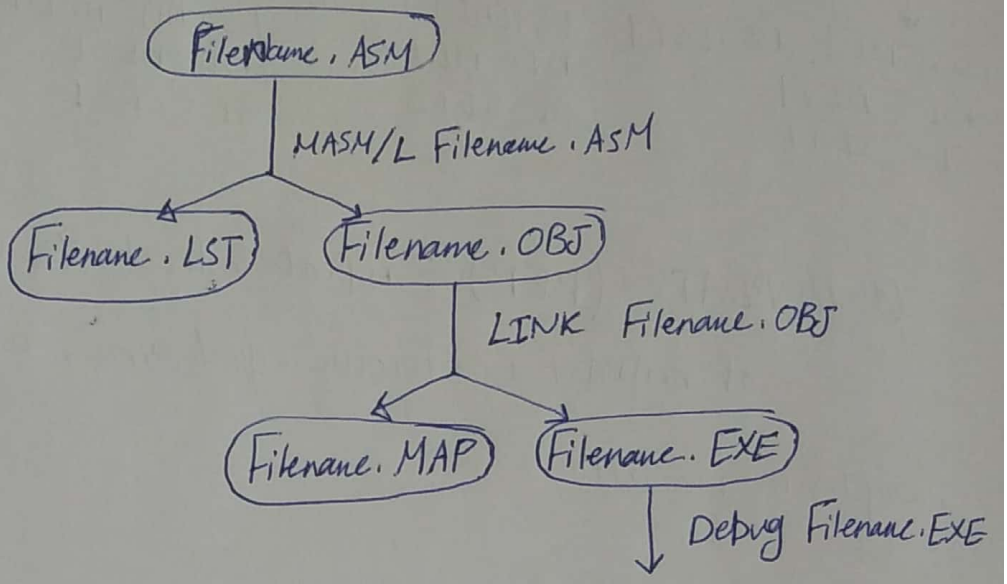
* .exe file & .map file will be created by ⇒ Link exp2.obj .

① executable file .

② MAP file

* optional file

* gives the name of each seg. , where it starts & stops , size of seg in bytes .



* Developing Assembly Language Program

① Data definition :

Sizes (DB, DW, DD--)

error1 the value assigned is greater than memory size allocated in to .

Illegal inst :

① Seg to seg data transfer MOV DS, ES / MOV DS, CS .. illegal

② code seg (CS) cannot be a dest op. MOV CS, Ax illegal

③ write an immediate value to seg MOV ES, 1422h illegal

④ Inst that have operands with diff. sizes VAR DB 5

MOV Ax, VAR error

الممسوحة ضوئيا بـ CamScanner

Directives

① use to tell assembler what to do

② Commands that are recognized & acted upon by the assembler
- Not part of inst. set
- Used to
  ① declare code
  ② Data areas
  ③ select memory model
  ④ declare procedures

③ Different assemblers have different directives.

NASM not equal MASM

Instructions

① Use to tell CPU what to do

② Assembled into machine code by assembler.

③ Executed at runtime by CPU

④ Member of Intel IA-32, IA-16 inst. set

* Segment definition:

Simplified seg

full seg.

① Simplified segment Definition

```
.Model memorymodel
.Data
  :
.Code
.startup
  !
.Exit
END
```

← values default → decimal

② Full segment Definition

```
Name1 Segment
  :                 ; Stack
Name1 ENDS
Name2 Segment
  :                 ; Data
Name2 ENDS
Name3 segment       ; code
Assume CS: Name3, DS: Name2, SS: Name1
MOV AX, Mydata
MOV DS, AX
  :
Name3 ENDS
```

الممسوحة ضوئيا بـ CamScanner

Notes :-

① Write number in hexa ⇒ AAHX ⇝ OAAH ✓

اذا كان الرقم يبدأ بـ A-F لازم اضع قبله صفر

② MOV CS, 500 → ممكن فيها غلطتين

① CS → dest op ما بيكون

② [redacted] reg لازم رقم ما بصير اتعامل

memory model table

③ Lea SI, Ax ≡ mov SI, offset Ax

④ examples of Directives ⇒ .①Model / .②Startup / .③Exit / .④Data / .⑤Code ⑥Segment / ⑦ENDS / ⑧ASSUME

⑤ you can define var. in code & stack seg. but it's more correct Programing practice to keep them in data segment

⑥ we have two problems :

① the assembler needs to know which segment addr. to put into the Data segment regestor (DS).

Sol
MOV Ax, MyData ; set up our own data seg. add. in DS
MOV DS, Ax ; Can't load seg. reg. directly from memory.

② which form of memory-addressing machine instructions to use the assembler doesn't know that this more took place.

Sol the ASSUME directive.

الممسوحة ضوئيا بـ CamScanner

تطبيق Sum of avarge code

ex①

```
.Model small
.data
V3 DB 1,2,3,4,5
.code
.startup
mov SI, offset V3
mov AL, 00
mov cx, 5
L1: Add AL,[SI]
   inc SI
Loop L1
```

→ تغيير طريقة الـ CX

```
L1: Add AL,[SI]
   inc SI
   dec cx
   cmp cx, 0   } ≡ Loop
   jne L1
End
```

ex②

```
seg1 Segment
V4 dw 2,3,4,5,0AAH,0FFH,20H.
Avg DB ?
Sum dw ?
seg1 endS

Seg2 Segment
Assume CS: seg2, DS: Seg1
mov Ax, offset seg1
mov DS, Ax
mov SI, offset V4
mov Ax,00
mov Cx, 7
L1: Add Ax, [SI]
   Add SI, 2
Loop L1
mov Sum, Ax
mov BL, 7;
Div BL   ; remander AH & result AL
mov Avg, AL .
```

الممسوحة ضوئيا بـ CamScanner

exp 3 : Programming Techniques

① Subroutines

Handling inst :

① CALL op
PUSH IP
IP = op (16 bit)
No flags affect.

② RET or RET op
pop IP
SP = SP + op
No flags affect.

```
. Model small
. code
. startup
label { Sub1  proc          FAR   { stack
                directive   NEAR  { inderflow
Sub1
        ≡
        ret
  Sub1 endp
call sub1
  ≡
End
```

② Macros → خلال assembler

is group of instructions that perform one task.

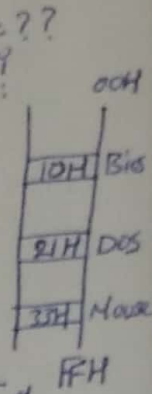
```
. Model small
. code
macro { m1  macro x1, x2, x3
        Local L1, L2 .
        L1 : mov AX, Bx
        jmp L1
        endm
≡
m1 5, 'A', 500
=
m1 6, 'B', 600 .
```

الممسوحة ضوئيا بـ CamScanner

11

Subroutine	Macro
① Perform one task	① Perform one task
② CALL & RET instructions	② MACRO & ENDM directives.
③ accessed via CALL inst.	③ inserted in the program at the point of usage
④ slower, becouse of call&ret inst	④ faster, placed macro in the program by assembler at the point they are invoked
⑤ Run	⑤ Assemble

exp4: BIOS Interrupts Programming



* INT XX ; the interrupt number XX can be 00-FF

① types 0-7 ⇒ reserved by Intel
0-4 ⇒ being predefined
5 ⇒ IBM uses for print screen
6&7 ⇒ not used.

② types 8-Fh ⇒ generated by hardware devices connected to 8259.

③ types 10h-1Fh ⇒ can be called by app programs to perform various I/O operations & status checking.

Text display

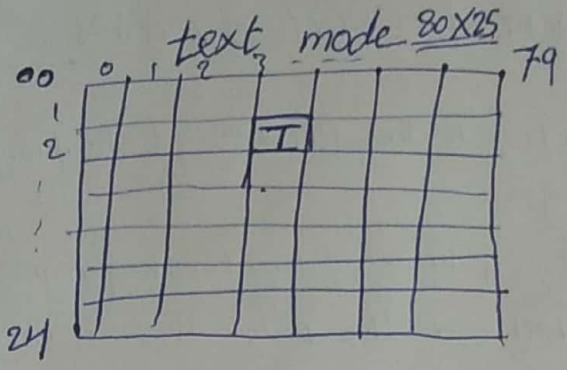
* Display mode: ⇒ text mode → display char.
⇒ Graphics mode → display pixels.

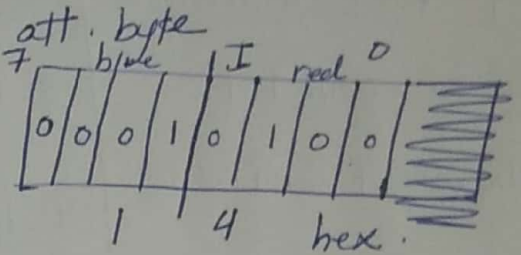
* video adapters: ⇒ MDA → only display text
⇒ CGA → display in color both text & graphics.

* Mode number: * Largest is colomns number.

الممسوحة ضوئيا بـ CamScanner

* attribute byte :- the high byte of the word that specifies a display character,





عكس I ← I → 0 نفس

```
.model small
.code
mov Ah,0
mov AL,3
int 10H
```
① display mode

text mode ⇒ default select display mode.

graphic mode ⇒ must to select display mode.

```
mov Ah, 2
mov DH, 2
mov DL, 3
mov BH, 0
int 10H
```
② move cursor

```
mov AH, 9
mov AL, 'T'
mov Cx, 1
mov BL, 14H
int 10H
```
③ display char

fun: 9 & Ah & Eh

Display char.

9 → cannot move cursor

Ah → يمكن تحريك المؤشر

Eh → لا يمكن تحريك المؤشر / معنى تجاهل

```
end
```

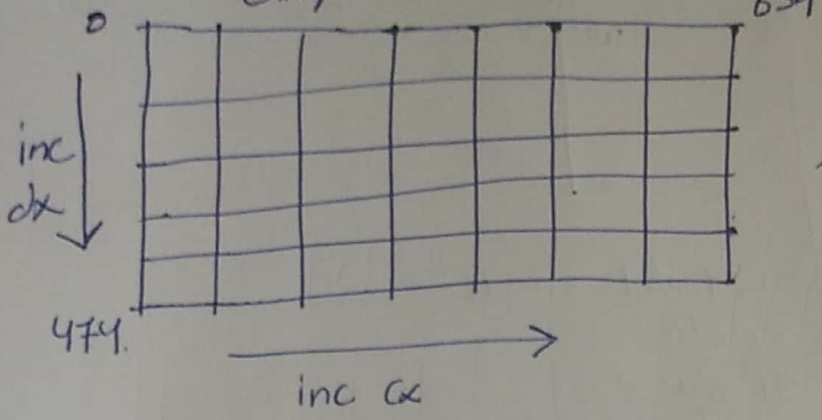
* Graphics Modes

* Pixel: picture element
* APA: all points addressable
* number of columns & rows gives the resolution of the graphic mode.

الممسوحة ضوئيا بـ CamScanner

13

Graphics mode



12H
640 x 480
columns x rows

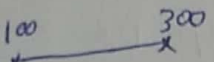
* التقاطع بخطين نقط

```
.model small
.code
mov Ah,0
mov AL,12H
int 10H
```
} select mode

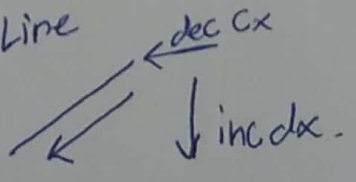
```
.data
str1 dB "Micro---"
.code
mov SI, offset str1
mov AL, [SI]
inc SI
```
} display

```
mov AH, 0CH
mov AL, 0AH
mov Cx, 100
mov Dx, 100
mov BH,0
int 10H
```
} write pixel.

```
mov AH, 0CH
mov AL, 0AH
mov Cx, 100
mov Dx, 100
mov BH, 0
L1: int 10H
inc Cx
cmp Cx, 300
jne L1
```
} draw horizontal Line

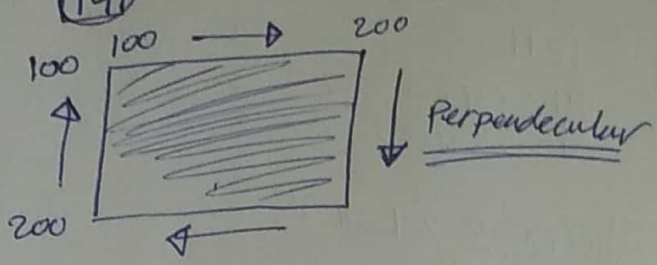



```
mov AL,0CH
mov Cx, 300
mov Dx, 100
mov AL,04
L1: int 10H
inc dx
dec Cx
cmp dx, 300 jne L1
```
} draw Line

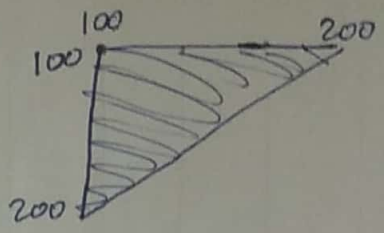


EICoM
ELECTRICAL ENGINEERING MECHATRONICS
الممسوحة ضوئيا بـ CamScanner

(14)



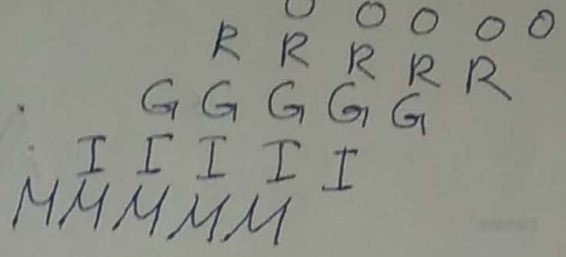

```
.Model small
.data
.code
mov Ah, 0
mov AL, 12H
int 10H
mov Ah, 0CH
mov AL, 0AH

mov Cx, 100
mov Dx, 100
L1 : int 10H
inc Cx
cmp Cx, 200
jne L1

inc Dx
mov Cx, 100
cmp Dx, 200
jne L1
```




```
.Model small
.data
V1 dw 0
.code
mov Ah, 0
mov AL, 12H
int 10H
mov Ah, 0CH
mov AL, 0AH
mov V1, 200
mov Cx, 100
mov Dx, 100
L1: int 10H
inc Cx
cmp Cx, V1
jne L1
dec V1
inc Dx
mov Cx, 100
cmp Dx, 200
jne L1 .
```

الممسوحة ضوئيا بـ CamScanner

Q1 display



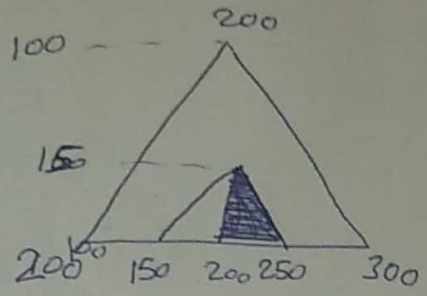
```
.model small
.data
v1 db 0
x1 db 20
x2 db 3
.code
.startup
mov ah,0
mov al,3
int 10h
M1 macro v1
mov ah,2
mov dh,x1
mov dl,x2
mov bh,0
int 10h
mov ah,0ah
mov bh,0
mov al,v1
mov cx,5
int 10h
M1 endm
mov v1,'M'
M1 v1
dec x1
inc x2
mov v1,'I'
M1 v1
dec x1
inc x2
mov v1,'G'
M1 v1
dec x1
inc x2
mov v1
mov v1,'R'
M1 v1
dec x1
inc x2
mov v1,'O'
M1 v1
ret.
```

الممسوحة ضوئيا بـ CamScanner

Q2

```
.Model small
.data V1 dw 0
.code
mov Ah,0
mov AL,12H
int 10H
```

```
mov Ah,0CH
mov AL,0AH
```




```
    mov Cx,200
    mov dx,100
L1: int 10H
    inc dx
    dec cx
    cmp dx,200
jne L1

    mov Cx,200
    mov dx,100
L2: int 10H
    inc dx
    inc cx
    cmp dx,200
jne L2.

    mov cx,100
    mov dx,200
L3: int 10H
    inc cx
    cmp cx,300
jne L3

    mov dx,150
    mov cx,200
    cmp dx,200
    jne L4
L4: int 10H
    inc dx
    dec cx
    cmp dx,200
jne L4
```

```
    mov dx,150
    mov cx,200
    mov V1,200
L5: int 10H
    inc cx
    cmp cx,V1
    jne L5
    inc V1
    inc dx
    mov cx,200
    cmp dx,200
    jne L5.
```

الممسوحة ضوئيا بـ CamScanner

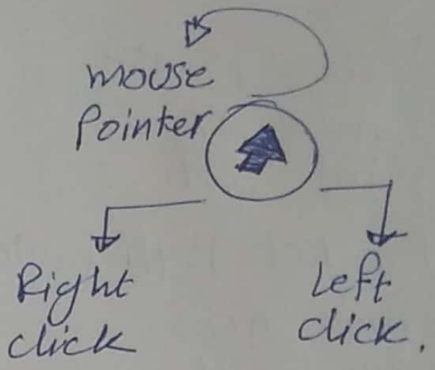
(1)

exp5

DOS & Mouse Interrupts Programming :-

DOS → 21H
mouse → 33H
} AH → function number

output in AL



* function 9:
  Print a string:
  - لازم ينتهي بـ $.
  - يخزنه في DS .

ex

```
.data
str db 'My name is Marwa$'
.code
.startup
 mov Ah,9
 mov Dx,offset str
 int 21H.
```

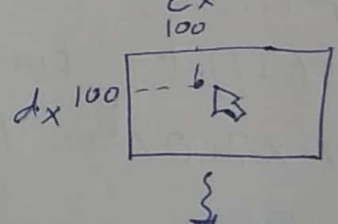
* function 2 :
  when DL = A ⇒ LineFeed ⇒ display result will be
  سطر فراغ .

| A
| B
| C
| ;

* function 00h :
  * ح تظهر الشاشة السوداء .

* شروط الـ MOUSE
  1. بشتغل على Graphic mode
  2. انها الـ fun بتوخذ AX كاملة محجوزة .

* function 03h :
* the value of Cx is doubled



Cx = 200
dx = 100

الممسوحة ضوئيا بـ CamScanner

(2)

(1) select mode (Graphic)

```
Mov AH,0
Mov AL, 13H
Int 10H.
```

(2) mouse initial

```
Mov Ax,00
int 33H
```

(3) show pointer

```
mov Ax,1
int 33H.
```

(4) select both buttons.

```
mov Ax,3
int 33H.
```

(5) work when click.

```
cmp Bx,1
JE left
cmp Bx,2
JE right
```

(6) infinit Loop for all of this (445..)

```
AGAIN:
  !
JMP AGAIN
```

(7) move carsure :-

(1) shift right → Cx (doubled value).

```
Shr Cx, 1
```

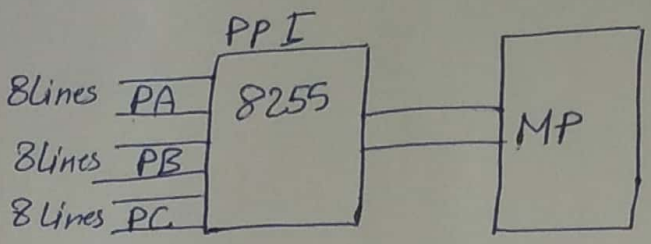
(2) shift right 3 times where we write a char

(13H) 320 x 200 ⟷ 40 x 25

الممسوحة ضوئيا بـ CamScanner

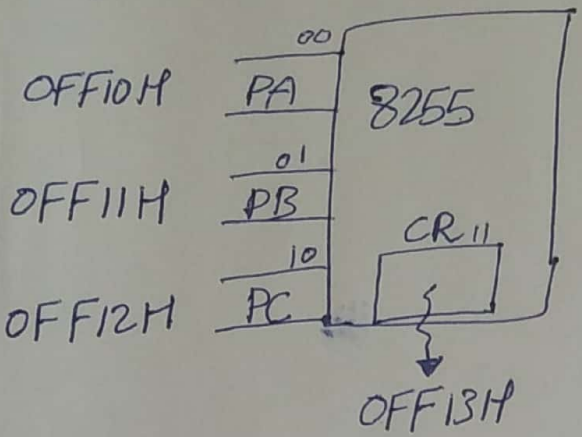
(4)

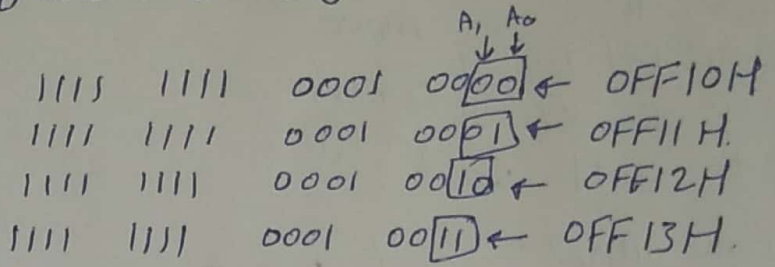
exp 6 Parallel data In/out.



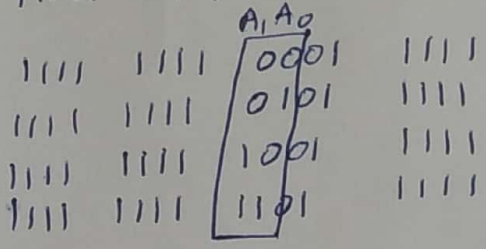
* adderess

16 line → each digit tack 4 lines.



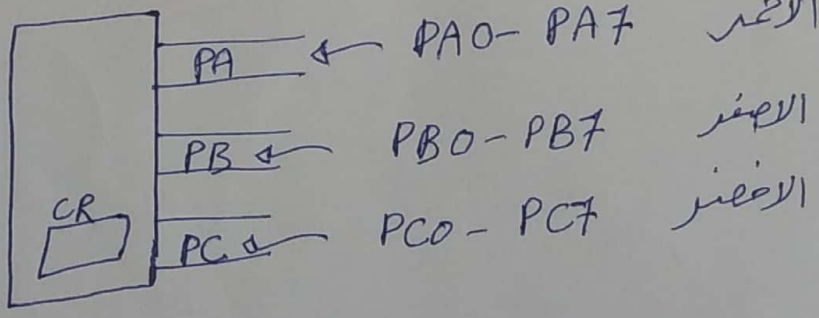


ex if addresses of ports is OFF1FH, OFF5FH, OFF9FH, OFFDFH.
find A_1 & A_0

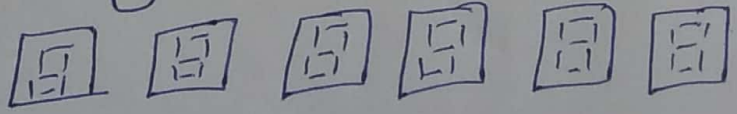


IDE Caple 25 Pin

* traffic light :



* Seven segments.



dot matrix

الممسوحة ضوئيا بـ CamScanner

(3)

* move cursor to write a char:-

```
shr dx, 1   ← doubled
shr cx, 1
shr cx, 1   } 8
shr cx, 1

shr Dx, 1
shr Dx, 1   } 8
shr Dx, 1

mov Ah, 2
mov DH, DL
mov DL, CL
int 10H     } move cursor in BIOS.

mov Ah, 2
mov DL, 'char'   } select char.
int 21H
```

* move cursor to write a pixel:-

```
shr cx, 1   ← doubled value.
mov AH, 0CH
mov AL, 0EH.   } draw pixel.
int 10H
```

example

الممسوحة ضوئيا بـ CamScanner

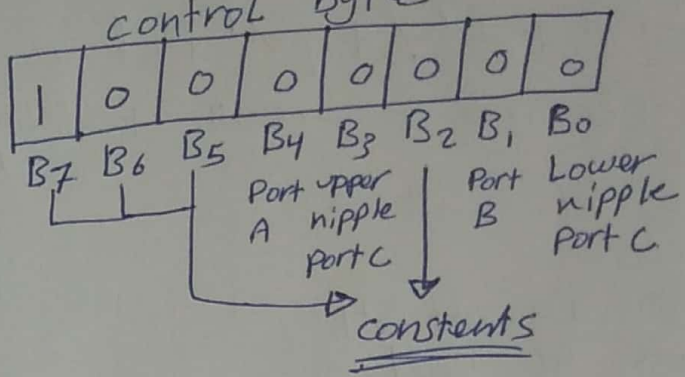
⑤ turn on red leds :-

```
• model small
• code
mov dx, off13H
mov AL, 80H
out dx, AL
```

address معنى

```
mov dx, AL  X
   ↑ syntax error
```

control byte




```
mov Dx, off10H   } ALL LEDS ON
mov AL, offH
out Dx, AL
```

Put: 0 → output
1 → input

Port C → الوحيد اللي ممكن ينقسم لجزئين

```
mov Dx, off10H   } واحد ON وواحد off
mov AL, 55H
out dx, AL
end
```

0101 1100
↓↓↓↓↓↓↓↓

1010 0100

1001 0010
↑ ↑ ↑ ↑

```
mov Dx, off10H   } جميع الالوان ON
mov AL, offH
out Dx, AL
inc DX
out Dx, AL
inc Dx
out Dx, AL
end
```

الممسوحة ضوئيا بـ CamScanner

⑥

PA 0000 0000 — PB 0000 0000 — PC 0000 0000

Red — Yallow — green

4MHZ speed

Need delay

① بدي اضوي Red كامل + delay + الاصفر + delay + الاخضر + بعيد من البداية :-

```
.model small
.code
mov  Dx, off13H
mov  AL, 80H
out  Dx, AL
Port A
Again:
mov Dx, off10H
mov AL, offH
out Dx, AL
; delay
call delay  =>
mov AL, 00
out Dx, AL
Port B
mov Dx, off11H
mov AL, offH
out Dx, AL
call delay
mov AL, 00
out Dx, AL
Port C
mov Dx, off12H
mov AL, offH
out Dx, AL
call delay
jmp Again.
```

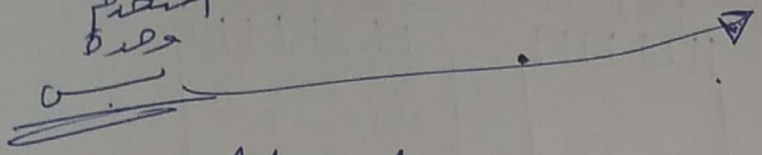
```
delay proc
mov cx, offffH
L1: Loop L1
ret
delay endp
End.
```

```
Loop L1 ≡ dec cx
          cmp cx, 0
          jne L1
          ret
```

الممسوحة ضوئيا بـ CamScanner

⑦ tern on seven segment :

counter on port B, 0 → F, 00 → FF



```
.model small
.code
mov Dx, 0ff13H
mov AL, 80H
out Dx, AL
Again:
mov Dx, 0ff11H
mov AL, 00
L1: out Dx, AL
call delay
inc AL
cmp AL, 10H
jne L1
jmp Again.
```

عشان تبين الـ F

Notes :

① down counter ⟹ dec ⟺ inc

mov AL, 01 ← 00 من

FF ≠ zero الى قبل الـ

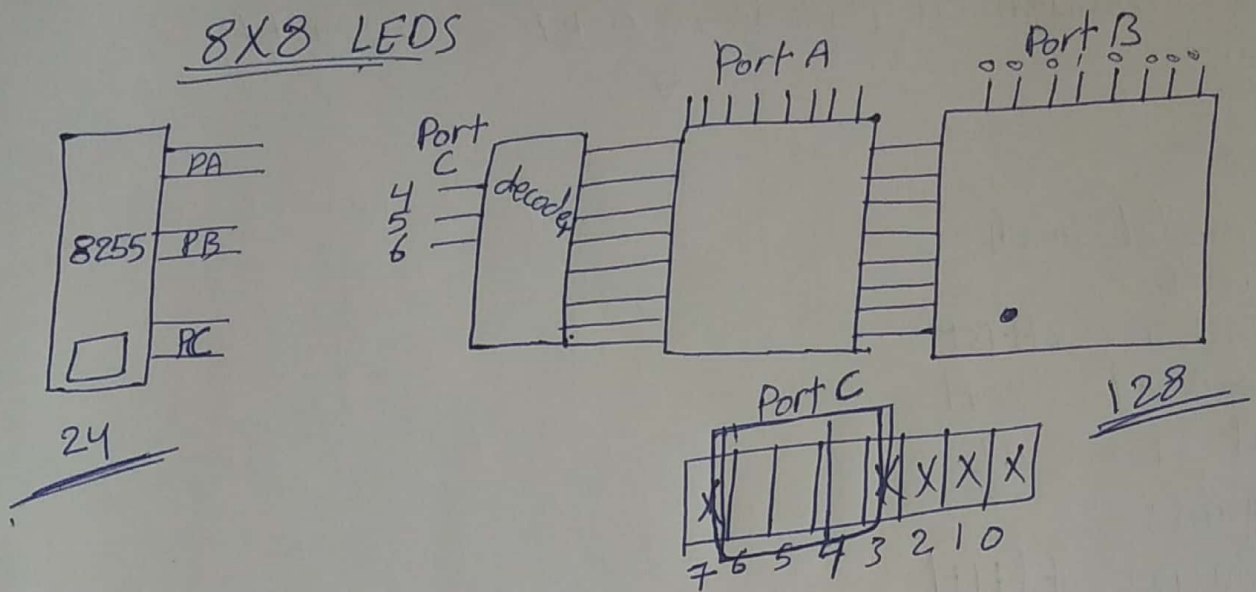
② odd → inc AL
cmp AL, 11H

③ even → inc AL
cmp AL, 10H.

الممسوحة ضوئيا بـ CamScanner

exp 7 I/O Apps: Dynamic Display.

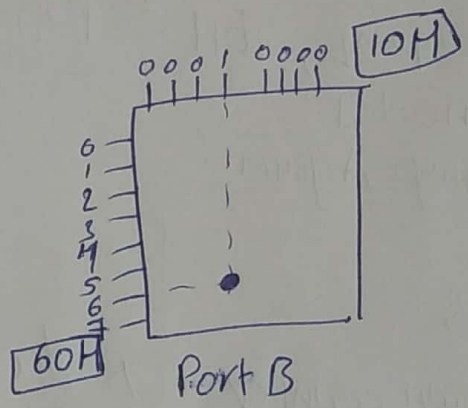
8X8 LEDS

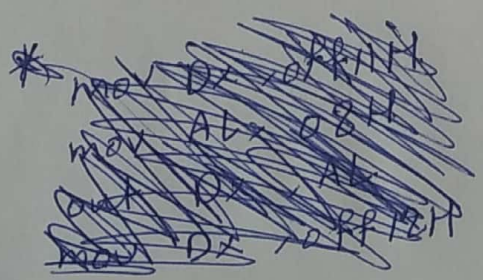


* display spicific dot.

- code

```
mov Dx, off13H   } initialization.
mov AL, 80H
out Dx, AL
mov Dx, off11H   } Port B
mov AL, 10H        00010000
out Dx, AL
mov Dx, off12H   } row 7
mov AL, 60H      } Port C → decoder
out Dx, AL
End
```



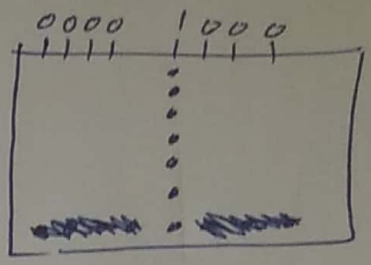


الممسوحة ضوئيا بـ CamScanner

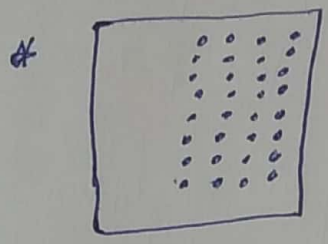
(9)

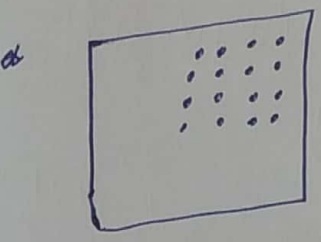
```
mov Dx, 0ff11H
mov AL, 08H
out Dx, AL
```

draw column




```
mov Dx, 0ff12H
Again: mov AL, 00
L1: out Dx, AL
Add AL, 10H
cmp AL, 80H
jne L1
jmp Again.
```





تغيير لون

```
mov al, 0F.
```

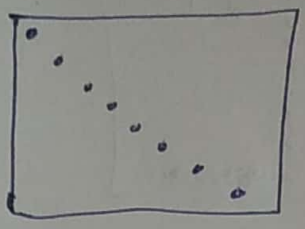
تغيير لون

```
cmp AL, 40.
```

+

الممسوحة ضوئيا بـ CamScanner

(10)



- code

```
mov Dx , off13H
mov AL , 80H
out Dx , AL
L2: mov BH , 80
    mov BL , 0
L1: mov Dx , off11H
    mov AL , BH
    out Dx , AL
    Shr BH , 1
    mov Dx , off12H
    mov AL , BL
    out Dx , AL
    Add BL , 10H
    cmp BL , 80H
    jne L1
    jmp L2
```

* work on computer
* عشان اشتغل البرنامج على الكمبيوتر

(1) ml Test.Asm ↵
⇓
Exe file

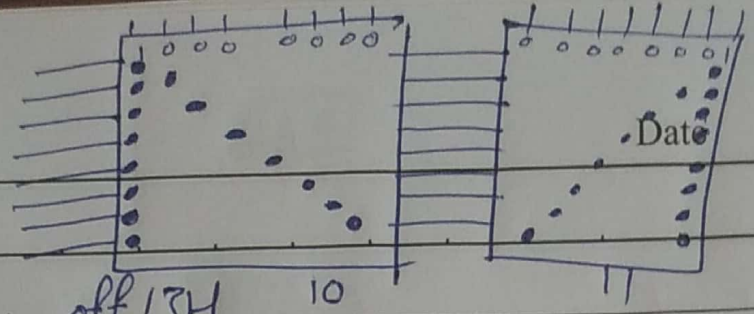
(2) exe2bin Test.exe ↵

(3) L 400 CR
(4) g = 400

} كل دي على الكمبيوتر

EICoM
ELECTRICAL COMPUTER MECHATRONICS

الممسوحة ضوئيا بـ CamScanner



Subject Date No.

```
mov dx, off13H
mov AL, 80H
out dx, AL
L2: mov bH, 80
    mov bL, 0
    mov cH, 01
    mov cL, 0
L1: mov dx, off10h
    mov AL, bH
    out dx, AL
    shr bh, 1
    mov dx, off11h
    mov al, ch
    out dx, al
    shL ch, 1
    mov dx, off12H
    mov AL, bL
    out dx, AL
    mov AL, cL
    out dx, AL
    add bL, 10H
    cmp bL, 80H
    Add cL, 10H
    cmp cL, 80h
    jne L1
    mov dx, off10h
    mov AL, 80H
    out dx, AL
    mov dx, off12H
    mov AL, 00
L3: out dx, al
    add al, 10H
    cmp al, 80H
    jne L3
    mov dx, off11h
    mov al, 01h
    out dx, al
    mov dx, off12H
    mov al, 00
L5: out dx, al
    add al, 10H
    cmp al, 80H
    jne L5
    jmp L2
    end
```


الممسوحة ضوئيا بـ CamScanner

Extra Code :-

Counters :-

① counter 00 → 0F (normal)

```
.model small
.code
mov dx, 0ff13H
mov AL, 80H
out Dx, AL
Again: mov Dx, 0ff11H → Port B
  mov AL, 00
L1: out Dx, AL
call delay
 inc AL
cmp AL, 10H → 0F + 01 = 10H
 jne L1
 Jmp again
```

② odd counter 01 → 0F

```
 mov AL, 01
L1: out Dx, AL
 call delay
 inc AL
 inc AL
cmp AL, 11H
```

③ odd counter 0F → 01

```
  mov AL, 0FH
L1: out dx, AL
call delay
dec AL
dec AL
cmp AL, 00H.
```

④ even counter 00 → 0F

```
mov AL, 00
L1: out dx, AL
call delay
 inc AL
 inc AL
cmp AL, 10H
```

⑤ even counter 0F → 00

```
mov AL, 0E H
L1: out dx, AL
call delay
dec AL
dec AL
cmp AL, FFH
```

and soon.

الممسوحة ضوئيا بـ CamScanner