EICoM
ELECTRICAL COMPUTER MECHATRONICS

مختبر معالجات دقيقة

للطالبة المبدعة

نتالي الكايد

إرادة - ثقة - تغيير

Subject EXP(1) Date No.

* نزل برنامج DosBox ، بتظهرلنا الشاشة بنكتب mount d d:\ ثم enter ثم d: enter

* عشان نشغل الـ debug في commands تبعت البرنامج ← بنزل الملف اللي عالمودل ، بنفك الضغط وبنغير اسمه إلى 8086 وبنخله على الـ local D drive .

* هلا بزجع لنفس الشاشة ، عشان أشغل الـ debug commands بكتب ← cd 8086 ثم enter ثم debug .

* شو معنى (debug system) عبارة عن monitor Programme براقبلي الريجسترات الموجودة جوا الميموري في المايكرو بروسيسور .. وبساعدني أشوف قيم الريجسترات وأعدل عليها وأعمل display أو مقارنة أو move وكتير من التغييرات وكمان بقدر من خلاله أعدل على الـ code segment يعني اني أكتب instruction set ويتحولوا إلى machine code ويتخزنوا في الـ code segment وبعديها أعملهم run .

* ليش بنشتغل على debug system ؟ لأنه الكيت اللي بالدار اسمها (Mts Ket) عبارة عن microprocessor training system تبعت المايكروبروسيسور الـ IC عليها وهي فيها 40 bins .

لما بدي أفتح الكيت كهاردوير .. بلاقي في Program debug تنزل عليه الـ (O.S) تبعها اللي هو عبارة عن monitring فقط ، فهاي الكيت ما بتفهم علي إلا عن طريق الـ Command .

فهاد الـ debug system زي أني بشغل على الكيت الحقيقة.

* بفوت ع my computer ← C drive ← عالبرنامج ← batch file ← بنزل الأخير اسمه كومنت [autoexec Lec]

بكتب ← mount d d:\ ← save .

d:
cd 8086

NOTEBOOK

Subject Date No.

* ال system commands اللي ح نشرحهم بهاي التجربة هي: memory managment commands

D : Display the contents of memory / dumps.

لو رجعنا لتجربة ٢ وكتبنا كوماند D ح يطلعلي مجموعة من ال addresses وجميعهم مجموعة من الريجسترات ، أول آدريس بكون هو ال (default addr) اللي بتكون من جزأين

- segment address
- offset address

} logical address التشيفرة مع بعض بتحكيلهم

073F : 0100

segment offset

logical

بما انه هاد ال logical address ، فكيف بتحسب ال physicle address ؟ بنعمل shift لل segment address + ال offset .

وبعين هيك (073F0) + (0100) ، هاد الفيزيكال آدريس هو أول 2 digits فقط (00) باقي طالميه عالتجربة .

فهاد بعض الفيزيكال آدريس بهاي ال 2 digits ⇐ 074F0

0100 ⇐ // // ال offset // //

0101 ⇐ (00) 2 digits التاني // // //

0102 ⇐ (00) 3 digits التالت // // //

⋮

* كل ما نوصل 0107 بكون مشينا كل ال 8 ريجسترات ، كل ريجستر بحوي فقط على (2digits) مكتوبة بالهيكسا سواء كـ addresses أو كـ values .

* لما كتبنا كوماند D لمرة واحدة ، كم ريجستر أعطاني ؟ كل ريجستر بوسع من (00-FF)

0000 0000 ⇐ 1111 1111

N O T E B O O K

CamScanner

Subject Date No.

بكل سطر في 16 value

073F : 0100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
073F : 010C	00 00 00 00 ----	صار ٢ تكون فيها
073F : 0120		الادريس تبعها = 10F
// : 0130		فهلا لما نزيدها بمقدار
// : 0140		واحد عشان نروح عالسطر
// : 0150		التاني بعبي النتائج
// : 0160	00	
// : 0170		

أو بنكتبها 16 location

10 hexa يعني عشان نصل كل سطر بنزله بزيد عالادريس 10

0110 0111

0110 (طبعا كله بال hexa).

* آخر لوكيشن هاد عنوانه 17F.

* من الادريس 100 إلى الادريس 17F، كم لوكيشن عنا؟ 17F + 1 = $17F

لأنه حسبنا من عند الصفر

decimel بال ، وعددهم [80 hexa Location] فبطلعوا

يساوي 128 byte في ال D command الواحدة.

* لو كتبنا كمان D command ٢ يكمل بنفس النمط ويعبي الادريس الأول صر تكملة عالادريس الأخير من ال D command الأولى، اذن يساوي 073F : 0180.

* لنفترض انه بدي أعرض عدد معين من الlocations، مش كل ال 128؟

D address ← بعطيه من أي ادريس يعطيني لوكيشن

مثال:- D 300 ← D ٢ يظهر النتائج كالتالي ← 073F : 0300 ← برضو ٢ يعرض 128 location

D range

مثال:- D 300-305 ← ٢ يعرض 6 location مش 5 لأنه اللوكيشن 300 محسوب و٢ تطلع النتائج كالتالي

073F : 0300 00 00 00 00 00 00

300 301 302 303 304 305

Subject Date No.

* دائماً الـ default segment هي الـ data segment ، لكن اذا بدي أروح على segment معينة فلازم أكتبها مثل لو بدي أروح على stack segment بكتب SS: 40,45

↳ بروح على الـ stack segment الحالي ← 073F:0040 05 00 00 00 00 00

073F: عشان بهاد البرنامج انه الـ stack والـ data والـ extra كلهم ماخدين نفس الـ segment address ، بس على الكيبت بكون لكل واحد رقم غير.

ES: 30,40 → 073F:0030 00 00 14 00 18 00 3F - - - -

073F:0040 05

عدد اللوكيشن بيساوي $10_h + 1 = 11_h$ ، يعني 17 لوكيشن بالديسيمال و 11 بالهيكا.

ES: 30,3F → 073F:0030 00 00 14 00 18 00 - - - -

بروح تعطيني سطر واحد يحوي على 16 لوكيشن.

لو بدنا نعمل display لـ segment عناياها وكان مثلاً الـ segment address بدها بيساوي 700 ، بكتب

700: 20,26 → 0700:0020 00 00 00 00 00 00 00

أعطتني 7 locations.

CamScanner

Subject Date No.

2 Enter/Edit Command (E) :- هاد بساعدني أني أعدل على قيمة المخزنة داخل الريجستر

- E address
- E address , List

e 20 ← هاد الـ offset address

073F : 0020 FF ← هاي معناها انه على آدريس 20 ، القيمة هنا FF ، وبيتش مني أدخل القيمة الجديدة مبكشب مثلاً 55 وبكبس Enter.

073F : 0020 FF.55

d 20,21 ← عشان أعرض قيم الريجستر بعد التعديل

073F : 0020 55 FF

55 ← قيمة الريجستر الذي عنوانه 20

FF ← ما بتهمني حالياً ، قيمة الريجستر الذي عنوانه 21

e 30, 11,22,33,44,55 ← start address, List

d 30,3F

073F :0030 11 . 22 33 44 55

e SS:44, aa, bb, cc, dd, ee, ff ← يعني بدي أعدل على السكاتر اللي عنوانه 44 وأحط فيه هاي القيم

d SS : 44,4F

073F : 0044 AA, BB, CC, DD, EE, FF

CamScanner

Subject Date No.

3 Fill Command (F) :-

↘ F range , List

F 30,40 , 22, 23, ff
d 30,40
073F : 0030 22 33 ff 22 33 ff ----
073F: 0040 33

* عبّا القيم الموجودة وضلّه يكررهم لحد ما خلص عدد اللوكيشن .

F 200 : 20,30 20 80 40 50
d 200 : 20,30
073F : ----

F es: 300 ,310 , 55, 33, dd
d es : 300, 310
073F : 0300 55 33 dd --- بضلّه يكررهم

F 30,40 , "hello in micro Lab" → بدي أعبي الريجستر بـ String
d 30,40 → هاي عبارة عن الـ assci number
073F : 0030 68 65 6c 6F 20 → للحروف وهي --- hello in micro La ← وقف لحد 16 لوكيشن
073F : 0040 61 → الفراغ الـ قيمة b ← لحرف آخر لوكيشن هون

CamScanner

Subject Date No.

4 Compar command (c) :- بعمل مقارنة لـ 2 blocks في الميموري

f 60,70,"hello on roof" بدنا نقارن بين هاد وبين المثال في الصفحة السابقة.

c 30,40,60,70 عشان أقدر بينهم لازم يكون إلهم نفس الطول

range block(1) range block(2) بعيد نفس عدد اللوكيشن.

بقدر أكتب بس 30,40,60 لأنه لازم يكون البلوكين إلهم نفس الطول.

073F:0036 69 6F 073F:0066 ← ج بقارن بس اللوكيشن المختلفين

073F:0037 6D 72 073F:0069 ← لقا عند هاد اللوكيشن في اختلاف

5 move command (m) :- بعمل copy لقيمة بلوك وبنقلها إلى لوكيشن تاني.

m 30,40,80 بدي أنقل محتوى اللوكيشن من (30-40) إلى اللوكيشن

d 80,90 (80-90) وأكيد كالة ج بفهم انه من 80 لأنه بجز إلهم نفس الطول.

القيم الموجودة في 30-40 ما ج تروح بعد الـ move.

073F:0080 68 65 6C ... hello in micro La

073F:0090 61 b

CamScanner

Subject Date No.

[6] Search command (S) :-

بدور على قيم داخل رينج معينه

S 60,70,"O"] بدي يدور على حرف O ضمن هاد الرينج

073F : 0064
073F : 0066
073F : 006A
073F : 006B
] بعرض اللوكيشنز اللي لقا فيها حرف O ، يعني لقاه ٤ مرات

S 60,70,"OO"] بدي بادور على OO

073F : 006A] لقاها عنوان واحد بس

مهم جداً :- لو كان في OO لكن بينهم space ما رح يوخدها .

بس يوخدهم حسب المعطى (OO) .

[7] assemble command (A) :-

mov aX,bX → هاد عبارة عن ما assembly instruction

error بقدر أنفذ مباشرة ورح يعطيني ايرور

a فعشان أقدر أنفذه لازم أخزن هاي الانتركشن وأستعملها ، فبتتخزن في

073F : 0100 ال (code seg---) ، والكومانند اللي

a 200 بساعدني على هاد الاشي هو (A)

073F : 200 هاد الانتركشن تحول إلى machin code وتخزن و القيمة

073F : 200 mov ax, bx] تبعته ما بتكفي لواحد byte ، لازم

073F : 202 ← يعني بدي أروح على الأدرس 200 و أبلش أحط الانتركشن عندها

تحجز (2 bytes) فعشان هيك بنلاحظ انه الأدريس اللي بعده 202 مش 201 .

Subject Date No.

073F : 0202 mov ax,4 ← 3 byte جزء

073F : 0205 mov bx,9 ← 3 bytes جزء

073F : 0208 add ax,bx ← 3 bytes جزء

073F : 020A ret ← 1 byte جزء

073F : 020B

بكبس enter مرتين عشان
أطلع من الوكيشن 020A

[8] Unassemble (U) Command :-

```
u 200
073F:0200  89D8      MOV   AX,BX     صون بعرضلي الـ machin code
073F:0202  B80400    MOV   AX,0004   للانستركشز اللي خزناها
                                     في المثال السابق .
```

[9] Go Command (g) :- بستعمله عشان أنفذ الـ assembly instruction اللي خزنتهم

g = 200 ,20A ← صون رح ينفذ من آدرس 200 لحد آدرس 208 مهم الانتباه :-

AX=000D BX=0009 ... لكن بنكتب آدرس 20A عشان يوقف لحد

~~BX=0009~~ رح يقوم بعرض قيم عنده (بس بدون ما ينفذه) وأنا ما بدي أنفذه

الرجسترات بعد عملية لأنه فيه (ret) ممكن تعملي stack over flow ,

التنفيذ.

CamScanner

Subject Date No.

Ax = ---- Bx = ---- Cx = ---- ----

Es = ---- ss = ---- IP = 020A ← هاد الـ (instruction Pointer) بأشر على

073F : 020A C3 RET ← الأدريس يلي وقفنا عنده لكن ما نفذناه

↳ يلي لسا ما تنفذه

10 Trace command (T) :- بشبه كوماند go بس الفرق انه بينفذ instruction بالترتيب

↘ T → trace the next instruction. T

↘ T5 → trace the next five instructions. T count

↘ T=105:10 → trace 16 instructions starting at CS:105 . T address, count

11 Register command (R) :- بيعرضلي قيم الريجسترات كاملة

↘ R وبعدّل على قيم الريجسترات الخاصة

↘ R register زي Ax , Bx والخ ...

الفرق بين R , D

R بعدل على ريجسترات Special Regs

D بعدل على ريجسترات Data segment

r

Ax = --- Bx = --- Cx = ---- ----

Ds = ---- Es = ---- ss = ---- ----

r Ax

Ax 000D

: 1F ← هون بدّخل القيمة الجديدة يلي بدي أحطها مكان الريجستر Ax

r

Ax = 001F Bx = --- ----

القيمة بعد التعديل

NOTEBOOK

Subject Date No.

rf صورة ازا بدي أعرض الـ flag Reg

NV	UP	EI	PL	NZ	NA	PO	NC
↓	↓	↓	↓	↓	↓	↓	↓
set	clear	set	Clear	clear	clear	set	Clear

وبستناني أعدل عليهم ، هاذا مابدي أعدل بس بكبس enter .

flags:

Set	Clear
OV = overflow	NV = No overflow
DN = Direction down	UP = Direction UP
EI = interrupts enabled	DI = interrupts disabled
NG = Sign flag Negative	PL = sign flag positive
ZR = Zero	NZ = Not Zero
PO = odd Parity	PE = even Parity
AC = Auxiliary carry	NA = No Auxiliary carry
CY = carry	NC = No carry

rf

NV UP EI PL NZ NA PO NC OV DN CY

هيدول التعديلات

rf

OV DN EI PL NZ NA PO CY بلي بدي ياها

بعد التعديل بعد التعديل

12 Quit (Q) :- عشان أطلع من الـ debug system

CamScanner

Subject Part (2) [exp(1) :- Instruction set] Date No.

* شو الفرق بين الـ system command والـ instruction set ؟

الـ system command بتشتغل على الـ debuge system ، يعني على الـ moniter program تبع الكيبت أو على الـ Dos box

يلي بتشتغل عليه ، وبتتكون من حرف واحد فقط يعبر عنها .

بكون الـ instruction set بتتكون من حرفين فأكثر ، وهي ~~مجلوطة من جنو~~ خاصة بالمايكروبروسيسر .

* the addressing modes are categorized into the categories :- (the way in which the operand of an instruction is specified)

[1] Register operand addressing mode :-

مثال :- Mov Ax , Bx — يعني الداتا عم jes من الريجسترات

XoR Ax , Dx — عليها

ADD AL, BL — ففجرد ما أشوف انه السورس هو ريجستر بعرف انه Reg mod بـ .

[2] immediate mode :-

مثال :- Mov Ax, 2000 — يعني الداتا جاية من

Mov cL , 0A — رقم constant

ADD AL , 45 — فالسورس هي imm .

AND Ax , 0000

[3] displacement mode :- (direct mode)

مثال :- Move Ax , [DISP] — لما بشوف [ ] وجواتها

Mov Ax , [0500] — رقم هاد يعني دايركت مود ، فهاد الرقم هو الـ offset address , أو بتحكيله effective ,

للاستركشن يلي بدي أجيب منه داتا .

فبروح على الاستركشن دايركت .

NOTEBOOK

Subject Date No.

4 Register indirect mode :-

مثال:- Mov Ax, [DI] صون بكون جوا اسم ريجستر داخل [ ]

ADD AL, [Bx] صار معناه انه خزن آدرس انتركشن معين

Mov Ax, [SI] داخل ريجستر ، يعني الحالة لازم أروح على الريجستر أخذ منه الآدرس ثم أروح على الانتركشن المطلوب ، فهو حطلي الآدرس بصورة indirect.

5 Based Indexed mode :-

مثال:- Base register : Bx, BP → صون بستعمل base Reg

Index register : SI, DI → index Reg // // //

Mov AL, [BP+SI]

Mov Ax, [Bx+DI]

6 indexed mode :-

مثال:- Mov Ax, [SI+2000] بستعمل Index Reg ويضيفعليهم

Mov AL, [DI +3000] رقم displacment

صون رح يجمع القيمة يلي بالريجستر مع الرقم بيكتبوه (offset) ويروح يجيب الداتا ويخزنها في Ax

7 based mode :-

مثال:- Mov AL, [BP + 0100] بستعمل based Reg مع الرقم

8 Based indexed displacment mode :-

مثال:- Mov AL, [SI + BP+2000] index Reg + based Reg + رقم displacment

CamScanner

ملاحظة :- بالنسبة لـ direction flag يجعلها CLD يعني كلما كان auto increment أعملها

auto decrement كان أعملها set يعني STD ويجعلها

Subject Date No.

* Instruction Set :-

1. arithmetic instruction :-
   - ADD operand1 , operand2
   - SUB op // //
   - MUL operand [mem أو Reg لازم يكون op الـ]
   - DIV // [// //]

2. Data transfer instruction :-
   - Mov → general instruction → Mov Ax , 00FF ← ينقل البيانات من مكان لمكان
   - PUSH , POP → stack instruction . → PUSH Ax → SP = SP-2 (يرجع لها العنوان (SP) ويوضع فيه البيانات)
   - LODS . STOS . MOVS → string instruction .

بهمني قبل ما أنفذ أي انتركشن load أو store ، انتبه لاسمه ، لو فيه i بايكون auto increment ، لو فيه d بايكون auto decrement ، بس بتكون قيمته مهمة بايكون directional flag

POP Ax ← ينقل البيانات من SP ثم SP = SP+2

مثال :-

SP = B3F
SS = 0040
SS : B3A = 00
SS : B3B = 00
SS : B3C = 00
SS : B3D = 00
SS : B3E = 00
SS : B3F = 00

(رجع 4 ، رجع 3)

Mov CX , 88AA
Mov DX , BB99
PUSH CX
PUSH DX

اكمل :-

1. CX 88AA
2. BB99 DX
3. SP = SP-2 = رجع أرجع خطوتين = SP = B3D
4. SP = SP-2 = رجع أرجع كمان خطوتين = SP = B3B

و B3D رجع يعمل PUSH للقيمة بتعت CX

رجع أعمل PUSH لقيمة DX لعند B و C .

CX 88AA ← D AA . E 88

B 99 . C BB

Subject Date No.

مثال:-

قبل		بعد
SP = B3B	POP BX	SP = B3F
SS = 0040	POP W0[80]	SS = 0040
SS: B3A = 00		SS: B3A = 00
SS: B3B = 22		SS: B3B = 22
SS: B3C = 66		SS: B3C = 66
SS: B3D = CC		SS: B3D = CC
SS: B3E = 88		SS: B3E = 88
SS: B3F = 00		SS: B3F = 00
		BX = 6622
		DS:80 = CC
		DS:81 = 88

3 String instruction :-

↘ Load :- يعني أحمّل داتا من الميموري إلى الرجستر وهاي الداتا إما تكون byte واحد أو تنين.

بعد ما أحمّل الداتا من الميموري فإن SI بتتقل بمقدار 1 أو 2 حسب كم نقلت بايتات

SI: source index

↘ STORE :- يعني بدي أخزن داتا من الرجيسترات (data seg) إلى الميموري

بعد ما أخزن الداتا في الميموري فإن DF بتزيد بمقدار 1 أو 2 عكسب عدد الـ bytes.

قبل	instruction	بعد
DF=0	LOD SB بايت	SI = SI + 1
DF=1	LOD SB	SI = SI - 1
DF=0	LOD SB ورد	SI = SI + 2
DF=1	LOD SB W	SI = SI - 2

قبل	instruction	بعد
Df=0	STOSB	DI = DI + 1
Df=1	STO SB	DI = DI - 1
Df=0	STO SW	DI = DI + 2
Df=1	STO SW	DI = DI - 2

CamScanner

Subject Date No.

Movs :-

هاي بستعمل الـ source index والـ destination index .

قبل	instruction	بعد
DF=0	MovSB	DI = DI + 1 , SI = SI + 1
DF=1	MovSB	DI = DI - 1 , SI = SI - 1
DF=0	Mov SW	DI = DI + 2 , SI = SI + 2
DF=1	Mov SW	DI = DI - 2 , SI = SI - 2

REPEAT :-

لما بكتبها جنب اتركن معيه معناته لازم أعيد تنفيذه مرة تانية .

مثال

قبل	instruction	بعد
Ds:7F = C2	STD	DF = 1
Ds:80 = 30	Mov SI, 83	SI = 7F
Ds:81 = 24	LOD SW	DX = FC71
Ds:82 = 8B	Mov DX, AX	AX = 8B 24
Ds:83 = 71	LOD SW	
= Fc		
= 90		
= A0		
= F9		

أول اشي عطاني اللوكيشن يلي بدي أجيب منه داتا وهو 83 ، بروح عليه وبجيب منه الداتا

تاني اشي بده أجمّل word كاملة ، معناته رح أجيب داتا من العنوان 83 و 84 .

فالداتا يلي باللوكيشن 83 بحطها على الجزء low من الريجستر والداتا يلي باللوكيشن 84 بحطها على الجزء H من الريجستر ، فصارت قيمة AX = FC71 ، وصرت بعمل SI - 2 لأنه نقلنا word ولسيت byte .

تالت اشي بدي انقل الداتا المخزنة من AX إلى DX فبتعطيني قيمة DX = FC71

رابع اشي بدي أجمّل داتا من اللوكيشن 81 و 82 ، لأنه احنا نقصنا SI بمقدار 2 فصرت عند 81 بدل 83 ، فبعطيك بتصير قيمة AX = 8B 24 ، وصار SI - 2 .

فبتصير قيمتها 7F .

Subject Date No.

مثال:- CLD
Mov DI, 70
Mov Ax, 927c
STOSW

بعد
Es : 70 = 7c
Es : 71 = 92
DI = 72
DF = 0

مثال:- STD
Mov SI, 87
Mov DI, 98
Mov cx, 3
Rep Movsw

	قبل	بعد
	Ds : 7E = 14	DF = 1
	Ds : 7F = c2	SI = 81
	DS : 80 = 30	DI = 95
	: 81 = 24	Es : 97 = 71
	: 82 = 8B	: 98 = Fc
	: 83 = 71	: 99 = 90
	: 84 = Fc	: 9A = A0
	: 85 = 90	: 9B = F9
	: 86 = A0	: 9C = 8c
	: 87 = F9	
	: 88 = 8c	

هاي يعني بدي أكرر أمر الـ Mov ٣ مرات

* أول اشي يروح على الـ Source index اللي قيمة 87 فياخد 87 و 88 ويودي الداتا تبعتهم على الـ destin.- index اللي قيمتها أو بالأحرى عنوانها 98 و 9c .
ويتقل الداتا عليهم .
* هلا بتقل DF-2 و فرح نفس عند العنوان 85 بالورس و عند 99 بال des-- .
* هلا بجيب الداتا التانية من 85 و 86 إلى 99 و 9A .
* كما انه في تكرار وهالة بترجع بتعيد نفس العملية ٣ مرات فبتنقص SI بمقدار 2 و DI بمقدار 2 . وهكذا ---

CamScanner

Subject Date No.

ملاحظات مهمة :-

MUL 20 ⇒ رح تعطي ايرور

Div 20 ⇒ رح تعطي ايرور

لازم أحط الرقم بريجستر مو رقم .

لازم أرجع لمادة بالسيمبلي عشان نشوف طريقة كتابة الانتركشن زي mov ---

Subject Experiment (2): Date No.

developing assembly Language Programs and executing using Emu8086 and MASM.

* في التجربة دي هنتعلم نكتب برنامج بلغة الاسيمبلي ، ونعمله execution باستخدام الـ emulater وباستخدام MASM .

* بالنسبة لبرنامج MASM بكون موجود في فولدر 8086 ، بنفتح برنامج DosBox ونروح على الباث D:\8086 .

* لما كنا نكتب انتركشن في Dosbox كان يطبعلي ايرور ، لكن ما أكتب كوماند A ويعديني بقدر أكتب assembly Program .

فلو مثلاً كتبنا كود كامل ولاحظنا انه نسينا واحد من الانتركشنز فلحتى نوصل للانتركشن ونضيفه عالكود رح نيجي نطبعلي فعشان نحل هالمشكلة .. بنستخدم الـ MASM .

* بالنسبة لـ MASM والايميوليتر ، عشان أقدر أكتب assembly Program لازم يكون عنا assembly source file ، بنعمله على الـ desktop كالتالي :-

New → text document → Lab2.ASM → بنفتح الملف open with (Notebad) .

وبنكتب عليه الكود .. الكود لازم يكون مرتب عشان ما يطلعلي عليه إيرور وتنسيق الكود بنحكيه بأي Segment defintion أهميتها أعرف من خلالها وين الـ code segm... تبعي عشان أحط فيها انكود وأعرف وين الـ data segm... عشان أحط فيها الداتا .

* في عنا طريقتين لـ Segment defintion :-

; → Comment

Subject | Date | No.

الطريقة الأولى :-

; Simplified segment definition

.model small

.Code

.startup

.exit

end

هون بعرّف الداتا و بكتب .data

* هدول كلهم السهم directives والدايركتف بختلفوا عن الانتركشن العادي بإنهم ما بقولوا لـ machine code

هون بكتب الكود

لو سألنا .startup لازم أكتب محلها
mov ax,@data
mov ds , ax

* هاي الدايركتف ما بفهمها إلا الاسمبلر ، أما المايكروبروسيسر ما رح يشوفوا ، هو بس رح يشوف machine code

* نوع أسمبلر عن التاني ، ممكن تختلف الدايركتيف المستعملة فيه .

* الاسمبلر المستعمل في هاد اللاب هو MASM .

.Model :- هاد معناه انه رح استخدم مودل لتعريف السيغمنت ، وبهاد المثال اسم المودل small .

وفي مودل غيرها بنسميهم memory models مثل

Tiny	→	1	
Compact	→	1	→ many
Small	→	1	→ 1
medium	→	many	→ 1
large	→	many	→ many
Flat	→	1	

سيتم توضيحه لاحقا

الطريقة الثانية :-

; full segment definition

data1 segment

data1 ends

code1 segment

assume cs : code1 , ds : data1 , es : ---

هاد منخلاله بعرّف الاسمبلر وين الداتا سيغمنت ووين الكود سيغمنت ووين الستاك ، الخ ...

لو كتبنا زي هيك رح يعطيني إيرور ، لأنه همة code1 و data1 عبارة عن Lable والليبل همة offset address هو رقم وما بعرف أنقله مباشرة على سيغمنت رجستر . فلذلك غلط نكتب هيك فبنكتب زي السطرين يلي إجوه .

mov ds, data1
mov ax, data1
mov ds , ax

هدول التنين همة بدل الـ .startup

code1 ends

end

بكتب الكود هون

NOTEBOOK

CamScanner

Subject Date No.

يعني الكود والداتا مرتبطات ببعض .

memory model :-	# of code segment	# of data segment	Data and code Combined	seg size
Tiny	1	1	yes	64K
Compact	1	many	No	64K
Small	1 (يعني مسموح أستعمل كود سيغمنت وحدة)	1	No	64K
Meduim	many	1	No	64K
large	many	many	No	64k
flat			yes	32bitOS 4GB

يعني بقدر أستعمل أكثر من داتا سيغمنت زي كود سيغمنت زي مثلاً سيغمنت

- Small model صدول مو خطر ، وبالاردعادة بنستعمل

*

CamScanner

Subject Date No.

; Simplified segment definition

.model small

.data

V1 db 0000 0000 إلى 1111 1111

define byte
يعني حجز بايت كامل داخل data seg الـ V1 وسميتها وبدي أخزن فيها داتا.

* لو بدي أعرّف لوكيشن فاضي ، ممكن أحط فيه قيمة زيرو أو ممكن أكتب

V1 db ?

يعني قيمة غير محددة

* بقدر أخزن رقم محصور بين 00 إلى FF .

، طب لو كتبت 300

ع يقرأها كـ decimal

ويعطي ايرور لأنه البايت الواحد في الديسمل بوسع لحد 255 فقط .

فعشان أتخطى الايرور بدل ما أكتب db بكتب dw

هيك بدل ما أحجز بايت واحد ، بحجز word.

V1 db 200

V2 db "A"

.code

.startup

mov al , V1

inc V1

add V1 , 8

mov bl , V2

* لو كنا معرفين dw

بدل db ، فهون ع يعبر ايرور لأنه al بتقبل توسع ممكن أحل المشكلة بإني أكتب .al بدل ax

* لما نعرف db "A" V2 يعني

خزن الـ assci code تبع حرف A داخل V2 .

* بقدر نعرّف آريه فبكتب بالداتا سيغمنت

arr1 db 0 , 5 , 6 , 0Fh , 1ah

decimal Hexa

هاد الآريه ع تخزن ، وأول لوكيشن ع يأخذ offset address of arr .

يتبع ...

NOTEBOOK

Subject Date No.

* بدنا ننتبه انه لما كتبنا arr1 db 0,5,6,Fh,1ah ↓ صوت رح يعطي ايرور لأنه لما آجي أعرّف حرف لازم أحط قبله رقم صفر ، فهيك بكتبه 0Fh ، أحسن ما يفكو ليل.

* db بهاي الحالة مخزنة byte كامل للرقم 0 و بايت كامل للرقم 5، وبايت كامل للرقم 6 وهكذا.

* لو كان في ضمن الداتا FFh رح يعطي ايرور لأنه لازم أكتب رقم 0 قبله = 0FFh

أما بالنسبة للـ FF رح توسع في بايت ، لأنه أصلاً البايت الواحد بوسع من 00-FF.

* لو كنا كاتبين arr1 db 0,5,6,Fh,1ah صوت رح يوسّع word كحالة بيتشغل 2byte لكل لوكيشن فيصير مثلاً الرقم 5 مخزن بالبايت الأول و 00 بالبايت الثاني ... FF بالبايت الأول و 00 بالبايت الثاني كونه مافي اشي يعبيه.

* الآن لو بدنا نجمع كل قيم الداتا فبنكتب هيك بالكود بيختصر:-

```
mov al , arr1      ← ما بعبي تكتب هيك رح يعطيه ايرور
mov al , arr1[0]   ←  //  //
```

هاد الداتا اله offset add بتاع 2bytes

ما رح يوسع داخل al.

```
mov si , offset arr1   ← هيك صحيح ومعناه انه بدي الأوفست
mov al , [si]          آدريس بتاع arr1 وخزنه داخل
mov al , [si+1]        الـ si اللي جوه [sourse index]
```

or

```
mov al , [si]          1 word = 2 byte = 16 bit
inc si                 يعني رح يوسع الـ offset address.
add al , [si]
inc si
add al , [si]
```

NOTEBOOK

Subject Date No.

or

```
mov cx,4
mov si,offset arr1
mov al,[si]
L1: inc si
add al,[si]
Loop L1
```

بدل ما أضيف القيم يلي في الآريه وحدة وحدة
بقدر أعمله بـ loop ففي البداية بعرّف
كاونتر اسمه cx وبكتب 4 لأنه
عدد العناصر غير العنصر الأول = 4

```
DEC cx
CMP cx,0
JNE L1
```

هدول = مكافئات
لـ Loop L1

لو كانت الآريه معرفة على أساس ال
هو db فالاختلافات يلي رح تصير على الكود
كالتالي :-

```
mov cx,4
mov si,offset arr1
mov ax,[si]
L1:
inc si
inc si
or
add si,2
add ax,[si]
Loop L1
```

عشان توسع
الأوفست
آدريس.

مرتين

عشان توسع

```
arr1 dw 0,5,6,...
```

رح ياخد أول
offset
address
لنفرض انه
00

offset ال
address
اللي 04
وهكذا...

بما انه حاجزين word
فإن ال offset add
رح نزيد بمقدار 2
فبصير 02

NOTEBOOK

Subject Date No.

* لو بدي أعرّف آريه بتحتوي على حروف وكلمات هيك بكتب :-

str1 db " hello in micro Lab " أو str1 db 'h','o','l'----

= ↓ بنقدر نكتبها

الـ byte بتكفي انها تخزن واحد char لأنه احنا بنخزن هون الـ ascii code تبع الحرف وطبعاً الـ ascii code هو رقمين بس يعني 2 digits فبوسعوا داخل 1 byte

رح يعطوا نفس النتيجة.

* اذا بدي أعرف طول هاي الآريه ، بعدّ عدد الحروف ومن ضمنهم الفراغات بين الكلمات لأنهم بحجزوا مكان والهم ascii code .

* لو مثلاً بدنا ندور على حرف معين في الآريه ، هيك بكتب بالكود بسيقمنت :-

```
mov cx,0
mov si, offset str1
L2: cmp [si], 'm'
    jne continue
    inc count
Continue:
    inc si
    inc cx
    cmp cx,18
    jne L2
```

أو بقدر أكتبه بطريقة تانية ← lea si, str1 (load effective address)

يقارن أول اندكس اذا كان فيه حرف m فإذا لقيته بزيد على الكاونتر واذا ما لقيته بروح jump على Continue

بعرفه بالداتا سيقمنت
count db 0

لأنه في 18 حرف بالآريه

Subject Date No.

* هلا احنا كملنا كتابة الكود هينا نفتح على MASM ، فأول اشي بنروح
this PC → D → 8086 → [redacted] بنحط الملف (LAB2.ASM)
اللي اشتغلنا عليه بعاد امكان

هلا بنفتح على شاشة الـ DosBox بنكتب أول كوماند MASM ↓
LAB2.asm

رح ينشأ فايل اسمه Lab2.obj واللي هو عبارة عن الماشين كود للبرنامج تبعي.

أو بنكتب كوماند تاني : MASM/L Lab2.ASM

رح ينشأ عنه فايلين Lab2 List ، Lab2.obj

لو فتحناه بنلاقي نفس الانستركشنز اللي كتبناهم بالكود
وبنلاقي الـ offset address والـ machine code - وازا في غلط بالكود أو انه تمام.

هلا بنكتب كوماند جديد :- Lab2.obj هلا بطلعلي انه الـ run file رح يكون اسمه Lab2.exe وبيسئلني اني أعين الاسم ، ازا ما بدي أعين بكبس Enter.

هلا الـ List file بكون مكتوب جنبه nul ، يعني مش رح يطلعلي ، لأنه هو optional file
ملو بدي ياه يطلعلي بكتب Lab2.map .

هلا رح يطلعلي map file ، exe file .

map file: في معلومات عن البرنامج اللي كتبته ومن أي آدرس رح يبلش يعمل run للكود

exe file: عشان أفتحه بكتب على الـ dosbox : Lab2.exe
بس ما رح أشوف قداامي ، لأنه الشغل كله عم يصير على الرانا فيموري.

NOTEBOOK

CamScanner

Subject Date No.

* هيك بنكون كملنا الـ MASM هاد رح نوخد عن الاسمبلي... بعد ما ننزله عالجهاز بنفتحه مباشرة ، بنكبس على quick start tutorial ثم على 8086 instruction set وبكبس على أي انتركشن بعديني كيف بشتغل هذا الانتركشن ---

* لو كبسنا على example رح يعطيني كودات جاهزة

* لما نكتب كود بنكبس على emulate بيطلع شاشتين وحدة منهم فيها الريجسترات والتانية original source code ، بكون نفسه الكود بس في مثلاً خط ممنوع هاد بدلني على الانتركشن يلي لسا ما تنفذ ، وهاد رح يتنقل ، ازا بدي أنفذه بكبس على run يلي تحت الريجسترات

* لو شفنا شاشة الريجسترات ، فيها الانتركشن والماشين كود تبعها والستاك والريجسترات ---

* لو كبسنا على view ← stack رح يعرضلي الستاك وال top of the stack ويقدر يلي عليه لون أزرق

أفتح شاشة ال variable منتفض بتكون وال flags والميموري.

* لو كبسنا على view ← memory ← بتقدر تعدل على الداتا سيغمنت.

Subject Date No.

مثال :- في عندي آريه وبدي أحسب المعدل لكل العناصر اللي فيها ؟

```
Model small
.Data
Arr DW 50,60,70,80,90 ---   فيها 20 عنصر
avg DB ?
.code
start up
mov SI, offset Arr,
mov cx , 19
mov Ax ,[SI]
L1: add SI ,2
    add Ax ,[SI]
    Loop L1
mov BL, 20
Div BL
mov AVG, AL
End
```

Ah = reminder
AL = result

CamScanner

Subject exp(3):- Programming techiques. Date No.

* رح نوخذ بهاي التجربة عن الـ Subroutine وكيف نعرفه وكيف نفرّق الـ macro وكلهم بكود بلغة الاسيمبلي.

* نحنا بنعرف انو الـ defoalt case لأي برنامج مكتوب بلغة الاسيمبلي انه يكون بمشي على الانتركشن بشكل sequintionally ، إلا اذا حُكنا Jump أو call أو return .

* أحيانا بحتاج أنفذ task معينة خلال تنفيذ البرنامج وكانت هاي التاسك عبارة عن أكثر من انتركشن واحد ، ومحتاجة أعيد هاي التاسك كذا مرة ، فحسان ما يطلع الكود كتير طويل. بستعمل كونسبت اسمه [modular Programming] .

macro و Subroutine اسمه ← بقدر أعمل implementation إلو باستخدام اشي اسمه ← هو عبارة عن 2 type of special segment بعرّفها وبعين استدعيها خلال الكود.

* Subroutine :- aslo known procedure, is a special segment of program that can be called for execution from any Point in a program. the process of tranferring control from the main program to a subroutine and return control back to the main program is achieved by what is known as subroutine handling instruction.

بستعمل فيها انتركشن معينة همة :- CALL , RET

CamScanner

Subject Date No.

* كيف يتم تعريف الـ Subroutine ؟

```
.model small
.data
.code
.startup
Subl proc far
near او
; مجموعة الانتركشن
ret
Subl endp
.exit
end
```

اذا ما في far ولا near فإن الديفولت هو near .

مثال:-

بدنا نعمل ننكشن يعمل جمع لمجموعة من المتغيرات (2 variable) .

```
.model small
.data
Vara db 30
Varb db 40
.code
.startup
Subl proc
    mov al, Vara
    mov bl, Varb
    add al, bl
    mov cl, al
    ret
Subl endp
```

exit

NOTEBOOK

CamScanner

Subject Date No.

main :

mov al,4 هاد الكود رح يمشي سيكونشل ، فأول اشي بدخل على

add bl, al الـsubroutine بنزل انتركشن ورا التاني ، هلا لما

Call Sub1 يوصل لعند ret ، ret فجأة هي بتعمل pop

mov dl, cl لقيمة IP يلي بالستاك [آدريس الانتركشن يلي عملناله push

.exit لما عملنا call بس خنا حالياً دخلنا على الـ subro.. بدون

end ما نوصل الـ call أصلاً ، فهون بهاي الحالة رح يلاقي الستاك

[بنقل الـ Sub routine لهون] فاضي وهاد معناه (Logical error) وهاد الايرور اسمه

(stack underflow) .

فالغلط هون انه مكان الـ subroutine غلط ، لازم نحطه تحت الـ main

فبعد ما أنقل مكان الـ subroutine ، أول اشي رح ينقل قيمة 4 إلى al

ثم يجمعها مع القيمة يلي داخل bl ويخزن الناتج في bl ثم يستدعي الـ subrot

و لما نوصل عن الـ call رح تعمل push للـ IP على الستاك ، حيث أن

IP بكون بالشر على next instruction يلي بده ينفذ .

يعني لو فرضنا في انتركشن مكتوب تحت call ، هاد الانتركشن لازم

نرجعله بعد ما نكمل الـ subrot ، فكيف بنرجعله ... من خلال الآدريس

تبعه يلي تخزن بالستاك قبل ما نروح على الـ subrot .

هاد هو معنى الـ call بتعمل شغلتين :- Jump to subrot / Push IP .

هلا بروح على الـ sub بنفذ كلشي فيه لحد ما يوصل لعند الـ Ret يلي بتعمل

pop للـ IP حيث أرجع لورا الانتركشن وأنفذه بعد الـ subrota.. .

وبما انه الكود بمشي سيكونشلي معنا بعد ما نكمل انتركشن mov dl,cl

رح يرجع يدخل الـ sub مرة تانية لأنه جاي تحته ، فراح يطلعلي logical error

اسمه [stack overflow] ، فعشان اتخلص من هالايرور وما أرجع أدخل على الـ sub مرة تانية .

Subject Date No.

للتمرين 2 كتب انتركشن Jump يودينى لنهاية البرنامج بدون ما يرجع يدخل على الـ subrot .

فيصير الكود النهائي كالتالي :-

```
.model small
.data
  vara db 30
  varb db 40
.code
.startup
  main:
  mov al,4
  add bl, al
  call subl
  mov dL, cl
  jump fin
  subl Proc
  mov al, vara
  mov bl, varb
  add aL, bl
  mov cl, al
  ret
subl endb
fin :
.exit
end
```

* هنا بنكبس على emulate عشان ينفذ البرنامج ، عشان أشوف الـ variable بكبس على veiw ثم variable .

* ازا بدنا نشوف كل انتركشن لحال كيف بتنفذ ، بكبس على كبسة single step .

CamScanner

Subject Date No.

* هون برنا نشوف كيف بتم تعريف الـ macro :-

```
.model small
.data
vara db 12
varb db 20
.code
.startup
M1 macro
mov al, vara
mov bl, varb
endm
main :
mov al,bl
M1
add AL,bl
M1
M1
.exit
end
```

أول اشي بمشي الكود بالترتيب فبدخل على الماكرو وبنفذه
وبعديه بدخل على الـ main و بنفذه بالترتيب ع لما بهل
عند M1 بعمل copy-paste للفنكشن تبع الماكرو وبنفذه
ثم بنفذ الانستركشن add Al,bl ثم برجع بستدعي الماكرو
مرتين وهكذا .
بنلاحظ هون ما في ret فا في أي غلط أثناء التنفيذ.

CamScanner

Subject Date No.

* أخدنا اشي اسمه Prameter list حيث ما استدعي الفنكشن
بقدر أ بعت معه متغيرات مثل الأنبوت والأبتوت إلخ ...

مثال :-

```
.model small
.data
Vara db 12
Varb db 20
.code
.startup
M1 macro V1,V2,V3
mov al, V1
mov bl, V2
mov cl, V3
endm
main:
M1 12, Boo, 'A'
M1 Vara, 300, 'A'   ] صحيح ولا يعطي error
M1 Vara, 300, 'A'
```

* أول اشي لازم تنتبه انه ما بقدر اعرّف Prameter باسم متغيرات عادية ، لازم أعرفها عن طريق تريجرات .

* هدول الـ prameters مو شرط تكون معرفينهم داخل الداتا سيجمنت ، مثلاً انا عرّفت داخل الداتا سيجمنت Vara و Varb لكن استخدمت Prameter باسم V1, V2 فهدول المتغيرات بنسميها local Var لأنه ما بتعرف عليها إلا الماكرو . [همة اسم لـ value مش لوكيشن حجزناه بالداتا سيجمنت] .

همون الرقم 12 رح تروح على V1
والرقم 300 رح تروح على V2
والحرف A رح يروح على V3

الآن 12 رح توسع بـ V1 ورح ننقلها إلى al تمام
لكن 300 لما ننقلها من V2 إلى AL ما رح توسع
لأنه AL ما بتوسع إلا لـ 255 .

لو كان مكتوب Labels داخل الماكرو كالتالي :-

```
L1: add AL,Cl
cmp AL, 0
JNE L1
L2:
mov Ax, Bx
endm
```

NOTEBOOK

CamScanner

Subject Date No.

* لو قمنا باستدعاء الماكرو أكثر من مرة ، فإن الاسيمبلر أي ليبل بشوفه ر2 يشيله ويحط offset address.
فمثلاً لو استدعينا الماكرو مرتين ، عند التنفيذ ر2 يتكرر ١٠ مرتين وهما مرتين وهكذا ...
فهاي تعتبر مشكلة لأنه مابجي نفس الليبل يكون موجود بأكثر من مكان بنفس الكود ، الليبل لازم يكون unique
إله offset address بالنسبة للأمبولتر. [هاد الايرور اسمه dublicate tables].

* فتى نتغلب على هالمشكلة بكتب local directive بعرفه بعد الماكرو مباشرة عشان نفهم الاسيمبلر انه هدول الليبلز ر2 يتكرروا ... فزي كأنه بفهم انه انا تكرر الماكرو لا ترجع تكرر الليبل بنفس الاوفست آدريس ، فبجي الاسيمبلر هلا كسبلهم offset address مختلفة.
فبجي الكود النهائي للكود كالتالي :-

```
.model small
.data
  Vara db 12
  Varb db 20
.code
.start up
M1 macro V1, V2, V3
LOCAL L1, L2, L3
mov aL, V4
mov bX, V2
mov CL, V3
L1 : add AL, cl
CmP AL, 0
JNE L1
L2:
mov Ax, BX
L3:
```

NOTEBOOK

CamScanner

Subject Date No.

* لو عنا بتعريف الـ main قبل الماكرو ، فإن الاسيمبلر رح يمشي بالترتيب وطبعاً رح يعطي ايرور لأنه لما يوصل لـ Vara,300,'A' ما رح يفهم شو يعني M1 لانهي لسا M1 ولا هي Subrot وهي ماكرو وما بتوديني على آي آدريس ، الخ ....
فعشان هيك الـ sub دائماً نعرفه قبل الـ main لكن الماكرو قبل الـ main .

* شو هي الاختلافات بين الـ Subrotine والـ macro ؟

MacRo	Procedrre
* macro definination contains a set of instruction to support moduler Programming.	* Procedure contains a set of instructions which can be called repetitive which can perform a specific task.
* it is used for small set of instruction mostly less than ten instruction.	* it is used for large set of instructions mostly more than ten instructions.
* in case of macro memory requirement in high.	* in case of procedure memory req. is less.
* Call and RET instruction arnt required.	* Call and RET are required.
* assembler directive (macRo) is used to define macro and (ENDM) is used to indicate the body is over.	* assembler directive (Proc) is used to define Pro Cedure and (ENDP)
* execution time for macre is less as it executes faster than Procedure.	* execution time is high as it executes slower than macro.
* Prometer is passed as Part of statement	* Parameter is passed as Regs and memory locations
* overhead time doesnt take place as there is no calling and returning.	* overhead time takes place during calling and returning.

N O T E B O O K

Subject Date No.

السؤال :

write a macro that takes 2 variables (A,B) and find the result of following equation :- $Bx = (A*4 - B*2)/7$

invake the macro 3 times in the main Program withe different values of A and B.

الحل :- [كود الماكرو فقط]

```
mov   aL , a
mov   bL , 4
mul   bL
mov   bl , aL

mov   aL , b
mov   dL , 2
mul   dL
mov   dL , aL

sub   bL , dL
mov   aL , bL
mov   dL , 7
div   dL.
```

السؤال :- write a program in assembly language that define a subrutine , this subrutine will convert a string array of character of 20 characters defined in the data segment to lower case String.

فكرة انه الأحرف capital الأسكي كود يكون إلها أقل بـ 20 من الأحرف small فكل حرف نزيد عليه 20 عشان يتحول الأسكي تبعه و يتحول من حرف كابيتل إلى small .

CamScanner

Subject Experiment (4) : BIOS interrupts programming. Date No.

* في ٣ أنواع لـ interrupt -: Bios interrupt / mouse interrupt / DOS interrupt .

* interrupt :- هي عبارة عن سيجنال بنقل tent للكومبيوتر ، عشان يوقف الـ run بالبرنامج الحالي ، عشان ينفذ الفنكشن يلي صار بسببه الـ interrupt .

* الـ interrupt ممكن يكون هاردوير وممكن يكون سوفت وير .

↳ master clear [shut down] ↳ برنامج بكتبه عشان يعالج الـ interrupt .

وهي إلها الـ highestPraiorty يعني لو كان في مية برنامج شغال واجت كبست على الـ shut down ، رح يوقف كل البرامج ويطفي الجهاز.

* الانتركشن الخاص بالـ interrupt بالاسمبلي هو :-

INT XX

↳ interrupt number (00 - FF) .

* لما نعمل رح يستدعي اشي اسمه [interrupt service routine] ، وهو عبارة عن subroutine لكنه خاص بهاد الـ interrupt [الرقم يلي رح نشتغل عليه XX] .

* بهاد التجربة رح نتعامل مع [interuppt 10h] وهو اسمه Bios interr.- ، ووهو إله أكتر من subroutine.

* كيف برنا نعرف شو الفنكشن يلي رح يتنفذ ؟ هاد بحدد على الزكستران يلي اسمه AH ↓ بكون مخزن فيه رقم الفنكشن اللي رح يتنفذ .

* Bios : Basic input / output system.

keyboard screen

↳ إلها فنكشن معين بشغله من خلال الأنتربت

Subject Date No.

* بعاي التجربة رح نحكي عن الـ BIOS interrupt اللي هي الـ bais output system مركز على الـ

* الـ operating system بقسم الشاشة إلى نوعين :- graphic mode / text mode

graphic mode ↓ الشاشة أيضا مقسمة إلى صفوف وأعمدة ، تقاطع كل صف مع عمود يعطي Pixel.

text mode ↓ الشاشة بالعالمة بتكون مقسمة إلى صفوف وأعمدة وتقاطع كل صف مع عمود يعطي charachter

* text mode :- في أكثر من نوع بتشتغل عليه الشاشة وإلهم أرقام مختلفة

قص حفظ

mode number	Description	adapaters
0	40X25 16-color text (color burst off)	CGA, EGA, MCGA, VGA
1	40X25 //	CGA. EGA . MCGA, VGA
2	80X25 // (color burst off)	//
3	80 X 25 //	//
7	80 X25 monochrome text	MDA , EGA , VGA

لو مثلاً بدي اشتغل على الـ mode(3) فإن الديسكبرشن إله من الجدول :- 80 X 25 16 colore

فالرقم الكبير هو دائماً يعبر عن الأعمدة والصغير عن الصفوف ، فهاد الـ mode ،

2- يقسم الشاشة لـ 80 عمود و 25 صف ويقدر أستخدم 16 لون مختلف .

جدول الألوان :-

I	R	G	B	color
0	0	0	0	Black
0	0	0	1	blue
0	0	1	0	green
0	0	1	1	cyan
0	1	0	0	Red
0	1	0	1	magenta
0	1	1	0	brown
0	1	1	1	white
1	0	0	0	gray
1	0	0	1	light blue
1	0	1	0	light ~~cyan~~ green
1	0	1	1	light cyan
1	1	0	0	// red
1	1	0	1	// magenta
1	1	1	0	yellow
1	1	1	1	intense white .

NOTEBOOK

CamScanner

Subject Date No.

* عشان نشغل الـ mode 3 اللي هو تابع لـ text mode التابع للـ screen interrupt التابع لـ bios بنكتب كالتالي :-

Function 0] ← رقم الفنكشن محجوز

وظيفة function [Select display mode] ← AH لـ reg

Input : AH = 0 ← هو اللي بحدد رقم نوع المود ورقمه.

AL = mode number ( )

output : none

بنفتح على الديبيوليق ، بنجيب ، وبنبلش نكتب :-

أول اشي بدنا نحدد نوع الـ mode

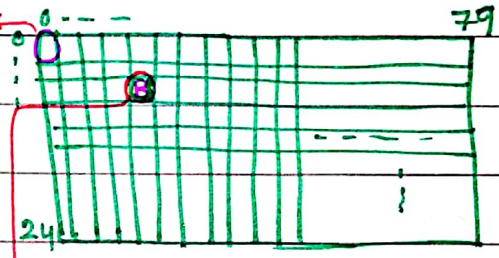
- model
- code

mov ah, 0 ← لأنه شغالين ع فنكشن 0

mov al, 3 ← رقم المود

int 10h ← نوع الإنتربت bios ورقمه 10

المكان الافتراضي للكيرسر.



mode (3) : بقسم الشاشة لـ 80 عمود و 25 صف

بدي أطبع حرف B بلون أحمر مع خلفية صفراء بالصف رقم = 7 والعمود رقم = 12 فبدنا نكتب كود يطبق هاد الكلام.

هلق بدنا نحرك الـ cursor من المكان الافتراضي إلى المكان المطلوب ، فبنستعمل Function 2 [حتى حفظ ، موجود كله في الماتول

mov ah, 2

mov dh, 7 ← رقم الصف

mov dl, 12 ← رقم العمود

mov bh, 0 ← ثابتة لأنه الـ page اللي بتعاملنا دايما هي 0 .

int 10h ← (هاد اسمه) page number

Subject Date No.

هلا بدنا نكتب الفنكشن اللي بطبع Character

وهو function 9 [من العناوان]

```
mov ah, 9
mov bh, 0
mov al, 'B'
mov cx, 1        ] عدد مرات طباعة الحرف
mov BL, 0e4h
int 10h
```

BL: 0e4h
- 1110 لون أصفر (من جدول الألوان) الخلفية
- 0100 لون أحمر الحرف
- 0 لازم نكتب قبل الحرف

هون بنزوع على الجدول attribute byte اللي موجود بالمانوال.

Bit	7	6	5	4	3	2	1	0
	background color					foreground color		
att	Blinking	Red	green	blue	intensity	Red	green	blue

كثافة اللون

نص البايت يعبر عن لون الخلفية | نص البايت يعبر عن لون الحرف

هلا بدنا نعرض النتائج بنكبس على emulate → run.

لو كتبنا زي هيك (على أساس طبع 5A) (3D و

```
mov ah, 9
mov bh, 0
mov al, 'A'
mov cx, 5
mov BL, 0e4h
int 10h
```

رح يطبع AAAAA DDD

ما طبع DDDAA

رح طبع DDD فوق أول ٣ A

والسبب اني ماصرتلي الكيرسر

فلازم بعد ما أطبع 5A أكتب كود تحريك الكيرسر كالتالي

```
mov ah, 9
mov bh, 0
mov al, 'D'
mov cx, 5
mov BL, 0e4h
int 10h

mov ah, 2
mov dh, 7
mov dL, 17
mov bh, 0
int 10h
```

بعد التحريل.

AAAAA DDD

NOTEBOOK

Subject Date No.

* عرفنا انه function 9 بنستعمله عشان نطبع الحرف ، لكن في غيره فنكشن بيشتغلوا نفس الاشي مثل function Ah بس الفرق انه ما بحط ال attribute بال bl ، هو بيشوف شو آخر attribute طالبة بفنكشن 9 ويعمل متله.

```
mov  ah , 0ah
mov  al , 'D'
mov  cx , 3
int  10h
```

ع يطبع الحرف D ثلاث مرات ← لكن صفات الحرف ع تكون بناءً على الخطوة السابقة (فإحنا ما طبعناه بفنكشن 9 كان لونه أحمر وخلفية صفراء ع يكمل هون عنفس النمط.

ملاحظة مهمة: هاد الفنكشن ما بحرك الكيرسر تلقائياً ، فلو طبعت ٣ حروف بدون تحريك الكيرسر ع يطبعهم فوق بعض و يطلع حرف واحد فقط. لو طبعت A,B,C ع الأخير بس C.

* أما بالنسبة للفنكشن e (function Eh) ، انه بعمل أدڤانس للكيرسر ، يعني بعد طباعة حرف الـ A خمس مرات فما في داعي بعدها أحرّك الكيرسر ، هو لحاله بحركه لكنه شكلة ما بكرر بعين ما في أحط عدد مرات تكرار الحرف بـ cx ، ودائماً بطبع أبيض وأسود.

```
mov , ah , 2
mov , dh , 3
mov , dl , 3
mov , bh , 0
int  10h
mov , 0eh
mov al , 'A'
int  10h
```

مكان الطباعة من = 3 صف و عمود = 2

لو طبعت ٣ حرف ورا بعض ع يعرضهم كلهم لأنه بحرك الكيرسر تلقائياً.

ع يطبع أبيض وأسود كونه function e

N O T E B O O K

Subject Date No.

* هلا لوبدنا نطبع عبارة كاملة ، فلازم بالأول أعرفها بالداتا سيغمنت متل هيك :-

```
.model small
.data
  str1 db 'welcome in ex 4'
.code
.startup
mov ah , 0
mov al , 3
int 10h
```

تحديد رقم ونوع mode الـ function (0) .

```
mov ah , 2
mov dh , 3
mov dl , 15
mov bh , 0
int 10h
```

تحديد مكان الطباعة (صف ٣ وعمود ١٥) فنكشن (2)

```
mov cx , 15
mov si , offset str1
L4:
mov ah , 0eh
mov al , [si]
int 10h
inc si
loop L1
```

بما انهم 15 صرف فلازم أعمل لوب عليهم كلهم .

* إحنا هون حبينا على فنكشن (e) لكن لوبدي هاي الحروف تكون ملونه ، فبظطر أتعامل مع فنكشن 9 كالتالي مع لوب .

الكود اللي فوق نفسه

```
mov cx , 15
mov si , offset str1
L1:
mov ah , 2
mov dh , 3
mov dl , 15
mov bh , 0
int 10h
inc dl
mov
```

كل مرة بطبعه لازم أصرح الكيرسر فبعمل ~~dec~~ inc dl

بكتب هون فنكشن 9

Subject Date No.

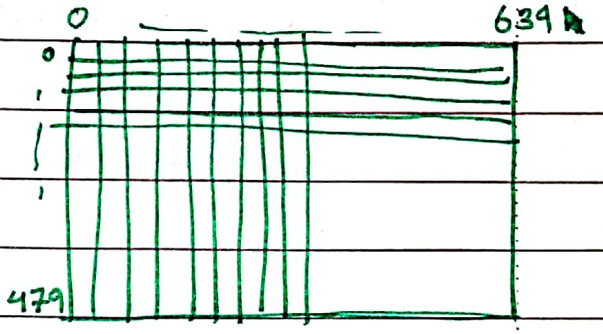
* هون بدنا نتعامل مع graphical mode -: أسهل من text mode حيث انه ما في cursor .

* تقاطع الأعمدة مع الصفوف هون أكتر بكتير من الـ text

* احنا رح نتعامل مع (model2) اللي فيه

640 X 480 16 color .

عدد الصفوف عدد الأعمدة



100X200

مثال :- بدنا نرسم خط مستقيم بيلش من البيكسل (صف = 100) (وعمود 100) وينتهي عند

، لونه الخط أخضر .

a (100,100) عمود صف

b (200,200) عمود صف

• made

• Code

```
mov   ah , 0      ] تحديد المود والاتربيت فنكشن (0)
mov   al , 12     ] رقم المود
int   10h
```

```
mov   ah , 0Ch    ] نكتب البيكسل فنكشن (C)
mov   al , 0Ah    ] لون أخضر
mov   Bh , 0      ] هاد رح يرسم بكسيل خضرا واحدة فقط
mov   CX , 100    ] العمود
mov   DX , 100    ] الصف
int   10h         واحنا بدنا خط فيه 100 . بدنا loop فبنعمل bixel
```

~~End~~

بس نوع استعمل bh , dx , cx لانهم محجوزين .

N O T E B O O K

لفن ريكورد
اسابعه

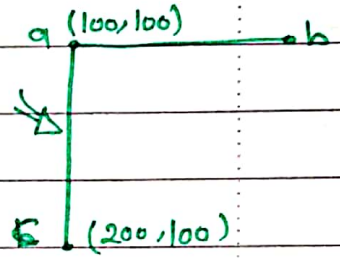
Subject: Date No.

again:
int 10h

inc cx ← لأنه اللي بزيد عندي هو العمود cx ولكن الصف ثابت.
cmp cx,200 ← بدي أضل أزيد بالأعمدة لحد ما توصل للبكسل 200
jne again ← لحد لما ما وصل 200 بترجع تعيد اللوب

يدنا نسبته وين بتخلي اللوب ، اذا حطيته فوق رح يصير infint loop لأنه رح يضل يزيد على الرجيستر للما لانهاية ، فبحطه قبل int 10h ، بحيث بعد ما يزيد قيمة cx بمقدار واحد يروح يطبع البكسل.

* مثال :- لو بدنا نرسم خط عمودي بعد الشكل (a-c) ، لونه أحمر.




```
.model small
.code
mov ah,0
mov al,12h
int 10h
mov ah,0ch
mov al,0
mov bh,0
mov cx,100
mov dx,100
again:
int 10h
inc dx
cmp dx,200
jne again
end
```


CamScanner

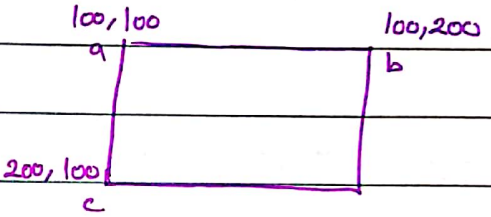
Subject Date No.

مثال:- لو بدنا نعمل مربع كامل، ونفيه باللون الأخضر.

[الفكرة هون انه احنا بنكرر الصف الأخضر بمقدار 100 مرة ، أو بنكرر العمود الأصفر 100 مرة].

بنكمل على الفرع السابقه وبنكتب :-

بدنا نضل نزيد بعدد الأعمدة cx لحد ما نوصل للعمود 200 ولازم ننتبه انه الصف ما يتغير أبدا




```
.model small
.code
mov ah,0
mov al,12h
int 10h

mov ah,0Ch
mov al,0ah
mov bh,0
mov cx,100
mov dx,100
again:
int 10h
inc dx
cmp dx,200
jne again
mov dx,100
inc cx
cmp cx,200
jne again
end
```

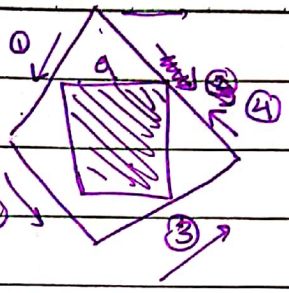
عشان بس يخلص رسم العمود يرجع الجديد

لو ما كتبنا هيك رح يطلع

عشان يزيد على الأعمدة

Subject Date No.

سؤال :- نرسم هاد الشكل ؟



نرسم المربع اللي بنعرف :-

```
.model small
.code
mov ah,0
mov al,#12h
int 10h

mov ah,0ch
mov al,0ah        لون
mov bh,0
mov cx,           صف البدء
mov dx,           عمود البدء

L1:
int 10h
inc dx
cmp dx,           الصف الهدف اللي رح نوصله
jne L1
mov dx,           الصف الهدف اللي بدأنا عنده
inc cx
cmp cx,           الهدف اللي رح نوصله
jne L1
end
```

```
mov ah,0ch
mov ah,           لون
mov cx,           العمود اللي رح نبدأ منه
mov dx,           الصف اللي رح نبدأ منه

L2:
int 10h
dec cx
inc dx
cmp dx,           الصف اللي رح نوصله
jne L2                                    ①

mov ah,0ch
mov ah,           لون
mov cx,           عمود
mov dx,           صف

L3:
int 10h
inc cx
inc dx
cmp dx,
jne L2                                    ②

dec dx  نفس الكود لكن                    ③
inc cx

dec dx  نفس الكود لكن                    ④
dec cx
```

NOTEBOOK

CamScanner

Subject Date No.

مثال :- لو بدنا نرسم هاد المثلث :-



لونه أخضر .

```
.model small
.code
mov ah , 0
mov al , 12h
int 10h

mov ah , 0Ch
mov ah , 0Ah
mov cx , 200h
mov dx , 100

again :
int 10h
dec cx
inc dx
cmp dx , 200
jne again
```

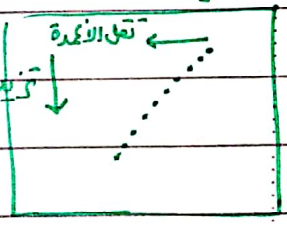
الفكرة انه بدنا نرسم العمود الأول ثم الثاني والثالث وهكذا ، وكل عمود جديد بنقص عن اللي قبله بمقدار pixel .

[فالعمود بنقص بمقدار 1 والصف بمقدار 1] .

فكل من العمود والصف ليس رقم ثابت هو متغير بنعبر عنهم V1 , V2

هيك بتكون رسمنا الخ

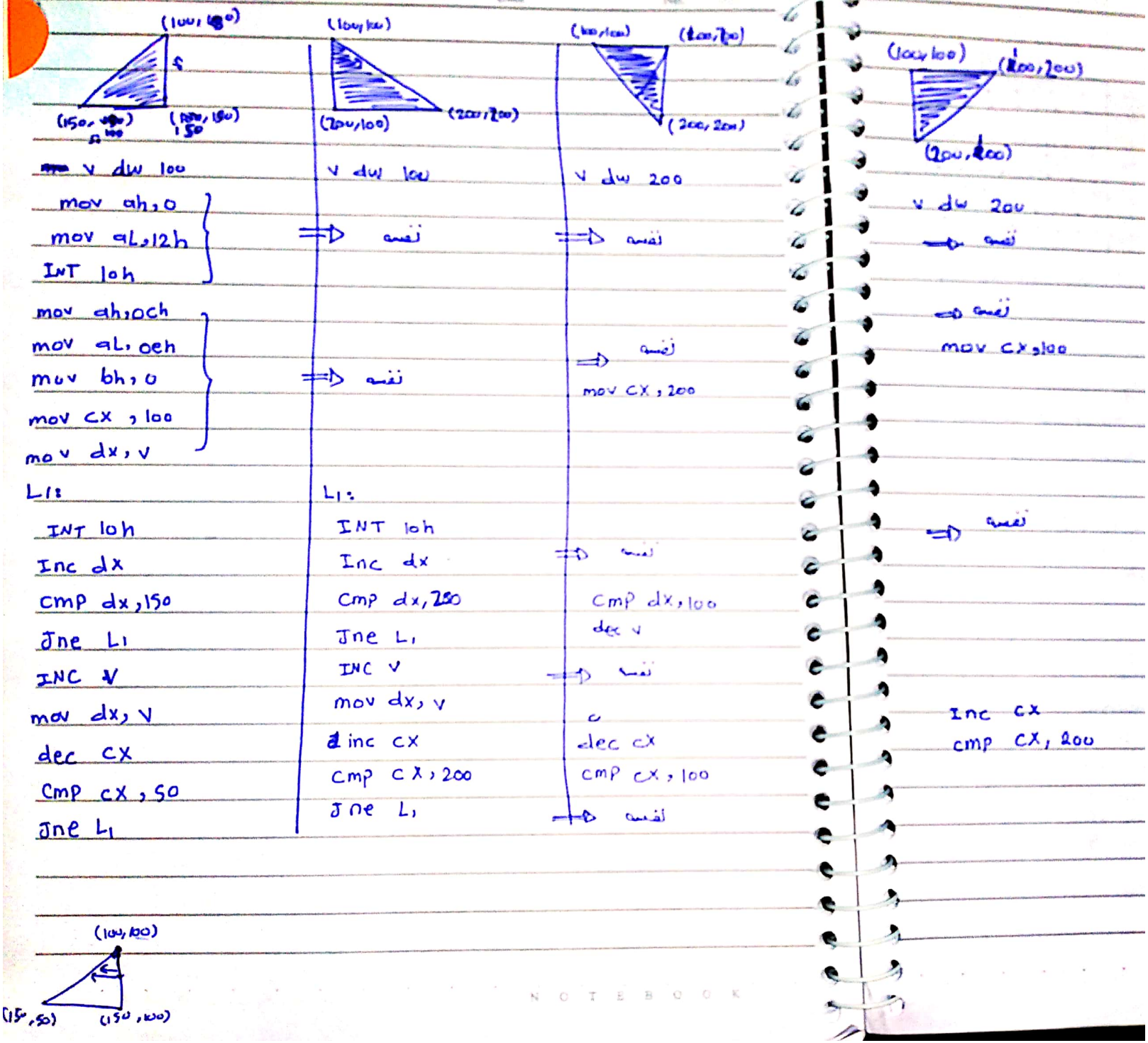
للتوضيح :-



قلل الأعمدة

وزاد الصفوف

NOTEBOOK


(100,100)
(150,150)
(150,100)
v dw 100
mov ah,0
mov al,12h
INT 10h
mov ah,0ch
mov al,0eh
mov bh,0
mov cx,100
mov dx,v
L1:
INT 10h
Inc dx
cmp dx,150
Jne L1
INC v
mov dx,v
dec cx
Cmp cx,50
Jne L1
(100,100)
(200,100)
(200,200)
v dw 100
نفسه
نفسه
L1:
INT 10h
Inc dx
Cmp dx,250
Jne L1
INC v
mov dx,v
inc cx
cmp cx,200
Jne L1
(200,200)
v dw 200
نفسه
نفسه
mov cx,200
نفسه
Cmp dx,100
dec v
نفسه
dec cx
cmp cx,100
نفسه
(100,100)
(200,200)
(200,100)
v dw 200
نفسه
نفسه
mov cx,100
نفسه
Inc cx
cmp cx,200
(100,100)
(150,50)
(150,100)

Subject ex(5):- mouse interrupts Programming. Date No.

* بهاي التجربة رح ندرس DOS interrupt ، mouse interrupt ، استخدامهم بشبه استخدام الـ Bios interrupt .

* بدل ماالتعامل مع 10h بعمل تعامل (21h) .

Function 1 :

Single key input

input AH = 1

output AL = ASCII

```
.model small
.code
mov AH, 1
int 21h
end
```

هاد الفنكشن لما أعمله run ، بتمشي من اليوزر انه يدخل حرف ، فمثلاً دخلنا حرف الـ D .
هاد رح يوخد الـ ascii تبع حرف الـ D ويخزنه في الرجستر AL ... ويطبع الحرف D تلقائياً .
[نستخدم هاد الفنكشن لما يكون بدي أدخل قيم من الكيبورد عن طريق اليوزر] .

Function 2 :

display a character or execute a control function.

input : AH = 2
DL = ascii

output : AL = ascii

```
.model small
.code
mov ah, 2
mov dl, 'Q'
int 21h
```

رح يطبع الحرف Q على الشاشة ، لأنه احنا بال text mode تلقائياً ... لو اذا حولنا لـ graphic mode كي رح ياخد الاسكي تبع الحرف .

Subject Date No.

* الفرق بين هاد الفنكشن و الفنكشن اللي بنستعمله بال Blos interr.. انه هاد بطبع بس أبيض وأسود بدون ألوان ولا ألوان خلفية ... وما بيدرك الكيرسر وما بتكرر طباعة الحرف.

* فائدتها انه بتعمل executer of control character ⇐ المقصود فيها :

Function	symbol	ascii code
Beep (sound)	BEL	7
back space	BS	8
tab	HT	9
line feed (new line)	LF	A
Carriage return	CR	D

```
.model small
.code
mov ah,2        خزنت الأسكي كود تبع الـ Beep
mov dL,7 ⇒      بدي نعمل رن فالكود رح
INT 21h         يطلع صوت .
```

```
.model small
.code
mov ah,2
mov dL,'A'
inT 21h

mov ah,2
mov dL,8
int 21h

mov ah,2
mov dL,'B'
```

B

أول اشي رح يطبع حرف الـ A بعدين رح يرجع لورا ويطبع حرف الـ B .

فهيك بين حرف الـ B وكان حرف الـ A .

N O T E B O O K

Subject Date No.

```
.model small
.code

mov ah,2
mov al, 'A'
int 21h
end
```

A B

⇒ طباعة حرف A

```
mov ah,2
mov al, 9
int 21h
```

⇒ tab function.

```
mov ah,2
mov al, 'B'
int 21h
end
```

⇒ طباعة حرف B

```
mov ah,2
mov dl, 10
int 21h
```

A
B
C
D

بنلاحظ انه
بنزل سطر بس
ما بطبع تحت A مباشرة
بتحرك جنبها ثم بنزل سطر جديد.

```
mov ah,2
mov al, 0DH
int 21h
```

A
B
C
D

بطبع أول حرف، ثم يرجع لبداية
السطر وبطبع تاني حرف ثم
يرجع لبداية السطر وبطبع الحرف
التالت وهكذا، فبيضل بالنهاية
الحرف الأخير هو الظاهر فقط.

NOTEBOOK

Subject Date No.

Function 9 : * الفرق بينه وبين الموجود في int 10 انه ضمان بطبع صرف حرف ، أما هون بطبع الجملة كاملة.

display a string

input : AH = 9

Dx = offset address of string

the string must end with '$'

```
.model
.data
.code    str1 db "welcome to micro lab exp 5 $"
.start up
mov ah, 9
```

لما يوصله رح يوقف طباعة ، لحتى مارح يطبع $.

```
mov dx, offset str1
int 21h
```

كمان بدي أطبع هاي الجملة بمكان معين.

```
mov ah, 2
```

فبكتب الفنكشن 2 بين أخدناه بال int 10

```
mov al, 40
```

بين كتبة الكيرسر ... لكن قبل فنكشن الطباعة.

```
mov dh, 12
mov bh, 0
int 10h
```


CamScanner

Subject Date No.

* mouse interrupt driver :- . 4 functions فيها فقط

[1] graphic mode على بشتغل ما

[2] . كاملة للفنكشن Ax بتوخد

Function 00h :

mouse initializiagtion ⇒ يلي بتعمله انها بتبين الماوس على الشاشة السودا وبتحطه على الموقع 00 .

input : Ax = 0D

output : if succesful : Ax = 0FFFFh and Bx = unmber of mouse buttons.

if faild : Ax = 0 .

int 33h

Function 01h :

show mouse pointer المقصود بال mouse Pointer هو الـ arrow (↗) عشان أعرضه داخل الشاشة السودا .

input : Ax = 01

output : display mouse pointer

Function 02h :

hide visible mouse pointer. يعمل اخفاء لل mousePointer .

Input : Ax = 02 .

output : hide mouse Pointer

CamScanner

Subject Date No.

Function 03h

get mouse position and status of its buttons.

input : Ax = 03

output : if left button is down, Bx = 1

if right button is down, Bx = 2

if both buttons are down, Bx = 3

Cx = x column number.

Dx = y row number.

Note: cx value is doubled.

بهرضلي بأي عمود وهنك كان الـ arrow موجود لما اليوزر عمل click ، فإذا اليوزر كبس عل right click بتكون قيمة Bx = 2 واذا كبس على الـ left بتكون قيمة Bx = 1

واذا كبس عجلة وسط موبيل ، بتكون Bx = 3 .

وهاي خلفة انه قيمة cx بترجع double ، فلو كان اليوزر عند العمود 100 ، رح تكون (cx = 200)

example: write a code to cheack the mouse click, if the right button is clicked, draw a yellow pixel in the mouse location, else if the left button is clicked, draw a red pixel in the mouse location.

CamScanner

Subject Date No.

```
.model small
.code
mov Ah,0
mov aL,12H
int 10h
```

graphic mode
بما انه سؤالي فيه ماوس

```
mov ax,1
Int 33H
mov AX,1
INT 33h
```

mouse initilization
show mouse Pointer

```
again:
mov Ax,3
int 33H
CMP BX,1
J ELEFT
CMP BX,2
JE right
Jmp agian.
```

get mouse Position
بتفقد قيمة Bx ، اذا كانت 1
معناته كابس عل اليسير ، اذا كانت 2 ....

```
LEFT:
ShR cx,1
mov ah,0Ch
mov aL,0Eh
int 10h
Jmp again
```

بما انه قيمة cx بترجع مدبلة فعشان أرجعها زي أصلها بقسم ٥ عل ٢ من shift right .
هاد الفنكشن تبع طباعة البكسل الأصفر.

NOTEBOOK

Subject Date No.

Right :

```
shl cx,1
mov ah,0ch
mov al,04h
int 10h
jmp again
end
```

* سؤال :- مطلوب نرسم ⊿ لونه أخضر بالكبس على right click ، ومثلث أصفر بالكبس على left click

حيث طول المثلث = 15 .

```
v1 db 0
```

فكرته انه بتعرف v1 قيمته صفر عند الدخول سيخزنت

وعند الكود بالنزل للرسم بيكتب

```
Loop:
int 10h
inc cx
inc v1
cmp v1,15
jne L1
```

ونفس الشي بالرسم dx .

الكود صحيح ✓

[Right و left تقس الكود السابقه بس بتحول عند ال]

CamScanner

Subject Date No.

```
LEFT:
SHR CX,1
mov AX,0CH
mov AL,03H
mov bh,0
L1:
int 10h
inc cx
inc v1          بعرفه بالناظر
                لتحديث = 0
cmp V1,15
jne L1
mov v1,0
L2:
Int 10h
inc dx
inc v1
cmp v1,15
jne L2
mov v1,0
L3:
int 10h
dec dx
dec cx
inc v1
cmp v1,15
jne L3
jmp again
```

```
Right:
ShR Cx,1
mov Ah,0ch
mov AL,0eh
mov bh,0
L4:
Int 10h
inc cx
inc v1
cmp v1,15
jne L4
mov v1,0
L5:
int 10h
inc dx
inc v1
cmp v1,15
jne L5
mov v1,0
L6:
int 10h
dec dx
dec cx
inc v1
cmp V1,15
jne L6
jmp again
```

NOTEBOOK

CamScanner

Subject exp(6) :- Date No.

* ح نبلش ندرس عن الكيت اللي بالمختبر واسمها (NTS Ket) وهي الكيت عليها microprocessor Ic و ICs مختلفة ، الـ ICs اللي نبغى عنها بالتجارب المتبقية هي (Ica255) اللي هي Programm Prerefaral interface.

ح نشتغل على هاي الـ (I.C) على mode واحد فقط.

* أهمية الـ (Ica255) انها تزيد عدد I/O lines الموجودة على المايكروبروسيسر ، فبالتالي يقدر أعمل applications أكثر.

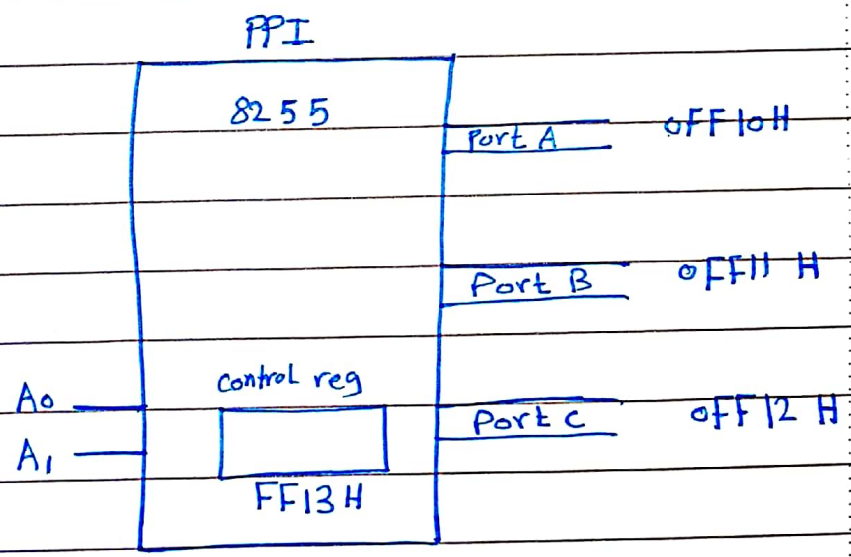
* إلها (3ports) وكل بورت عبارة عن (8 Lines) بتبرمجوا على أساس input/output Lines.

وبما انه عنا ports فلازم يكون الهم (addresses) يوضدوهم عن طريق (A1,A0).

يكونوا مشبوكين مع المايكروبروسيسر

ويصير في (address decoding) لما نقل microprocessar

الاتصال مع الوسط الخارجي ، يوضدوا addresses.



NOTEBOOK

Subject Date No.

كل من Port A , Port B , Port C لازم توخد آدريس ، والآدريس موضح على الصورة في الصفحة السابقة .

* بالنسبة لـ (control register) ، إله آدريس قيمته (FF13 H) ، أهميته انه بتحكم بباقي البورتات ، وهو عبارة عن رجستر فيه [8 bits] من bit 0 لـ bit 7 بعطيه Value يا هيض يا واحد عشان تتحكم ببورت من البورتات ... اذا حطينا 0 معناته هاد البورت اوتبوت ، واذا حطيناه 1 فبكون انبوت بورت .

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	Port A	upper Port C	0	Port B	Lower Port C

- bit 7 = 1 : مسؤولين عن الـ mode اللي بنشتغل عليه في الـ IC . وهو ثابت بهاد اللاب .
- bit 5 bit 6 = 0 0 : مسؤولين عن الـ mode اللي بنشتغل عليه في الـ IC . وهو ثابت بهاد اللاب .
- bit 4 : مسؤول عن تحديد بورت A إذا انبوت أو أوتبوت
- bit 3 : مسؤول عن الـ upper lines من بورت C وهو البورت الوحيد اللي بقدر أقسمه لقسمين ، فيمكن نخلي قسم منه مبرمج كانبوت بورت والقسم التاني مبرمج كأوتبوت بورت .
- bit 2 : دايما صفر
- bit 1 : مسؤول عن تحديد بورت B .
- bit 0 : مسؤول عن تحديد القسم Lower من بورت C .

Subject Date No.

example :- if we connect 7-segment display to port A (output), 8 Push buttons to port B (input), and 8 LEDs to port C (output), the value of control register should be $10000010_B = 82H$.

الحل :-

```
mov DX, 0FF13H   → أول اشي بنحط عنوان ثابت للـ control reg
mov aL, 82H      → تاني اشي بنحط قيمة داخل الـ control reg وقيمتها بالسؤال 82H.

mov DX, 0FF10H   → عشان اقرأ من أي عنوان وبما انه هون بقرأ من بورت A فبحط قيمة العنوان تبعه.
IN AL, DX        → هيك بيصير نقلنا القيمة المخزنة من البورت إلى AL

mov DX, ——       → عشان أكتب على أي عنوان.
mov AL, ——       → بنقل القيمة إلى هذا البورت
OUT DX, AL       → بعد تنفيذ out instruction، تم نقل الداتا إلى البورت المطلوب.
```


CamScanner

Subject Date No.

* هاد الكيت يلي رح نشتغل عليها باللاب ، بتكون مشبوكة مع جهاز الكومبيوتر ومع الأنترفيس كيت .

- 7-segment والليدات موجود عليها

أحمر — مشبوكين مع بورت A . وهي عبارة عن 8 LEDs 8 lines مشبوكين مع بورت A

أصفر — مشبوكين مع بورت B

أخضر — مشبوكين مع بورت C

* هلا بدنا نكتب كود بمثل [traffic light] ، بدنا نضوي الليدات الأحمر والأصفر والأخضر .

```
.model small
.code
mov DX, 0FF13H
mov AL, 80H
OUT DX, AL
```

* أول اشي بنربط أوامر الـ control register

80H → 1000 0000 لأني اريد اشغل 3 LEDs بس أشغل 3 Ports .

```
mov DX, 0FF10H
mov aL, 00
OUT DX, aL
mov DX, 0FF11H
OUT DX, aL
mov DX, 0FF12H
OUT DX, aL
```

بما انه بدنا نضوي الثلاث ليدات ، وعشانه لازم نبرمجهم كلهم على أساس انهم أوتبوت .

بكل البورتات turn off لليدات لأنه مبدئياً مابنعرف شو فيهم قيم

N O T E B O O K

Subject Date No.

```
again :
mov   DX , 0FFF10H
mov   AL , 0FFH  ──→ بما إني بدي أضوي البورت A وجناته بدي أضوي
OUT   DX , AL        كل الليدات فبحطهم 1111 1111 = FF
Call  delay
mov   aL , 00   ┐
OUT   DX , AL   ┘──→ رجعة طفية البورت

mov   DX , 0FFF11 H
mov   aL , 0FF H      نفس العملية على بورت B
OUT   Dx , aL
Call  delay
mov   aL , 00
out   Dx , aL

mov   DX , 0FFF12 H   نفس العملية على بورت C
mov   aL , 0FF H
OUT   DX , AL
Call  delay
mov   AL , 00
OUT   Dx , AL
Jmp   again
```


CamScanner

Subject Date No.

```
Delay Proc
mov cx, 0FFFFH
L1 : Loop L1
ret
delay endp
```

or

```
Delay Proc
mov cx, 0FFFFH
L1: dec cx
cmp cx,0
jne L1
ret
delay endp
```

* بعد كتابة الكود وتخزينه على الكومبوتر هاد برنا نشغله مع الكيت لكن الكيت ما بتفهم غير فايل (AsM) فأول خطوة لازم نحول الكود إلى binary file (file.bin).

* بنفتح على Dosbox بنعمل masm try.Asm ، بطلعلي object file ، بعدين Link بعمل ، List file اسم الملف للأوبجيكت فايل.

بعدين بعمل enter ثم ~~tasm exe~~ (exe2bin) هاد بتكتب ، exe بطلعلي file

Binary file على .

* الكيت اللي بنستخدمها متوصلة بين الكيت والكومبوتر ... وكل الاجهزة متوصلة على serialy rs2-32 communication . com1

* بنفتح على برنامج اسمه data transfer .

* هاد برنا ، com file ثم بعمل browse للفايل اللي طلع معنا اللي هو bin file وبنعمله debug command وبكتب ... download قبل ما أعمله داونلود لازم أجهز الكيت فبطوبها وبكتب L للتعبير عن load ثم N 200 ثم بكبس كبسة الانتر مرتين

أوفيست آدرس 200 code segment يعني رح أنزل على ال ثم بكبس انتر مرة واحدة .

بس اتأكد انه كلشي صح ، بكتب كوماند ال run وهو G=200 ثم Enter

على dos box بكتب وبنتج

```
masm Led.Asm
Link Led.obj
```

⇓

Led.exe

بكتب exe2bin Led.~~obj~~ exe

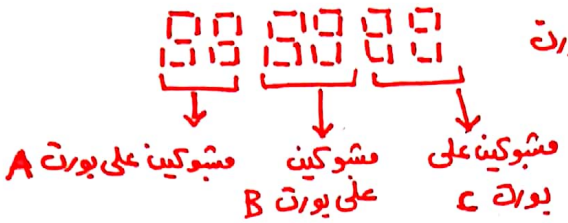
⇓

وبنتج Led.bin

NOTEBOOK

* هلا بدنا نطبق مثال تاني عشان نشغّل الـ 7-segment .

فيه عنا ٦ من الـ seven segment كل تنتين شابكين على بورت



* الـ seven segment الواحدة بتيجي على شكل chip حقيرة بتحتوي على 10 Legs ، بحيث كل سبعتة إلها

common anode ← common عشان الـ 2bins و 8 bins
common cathode ←

* من الأبليكيشنز اللي بنطبقها على الـ 7-seg ، مثل الـ timer ، elevator ، stop wach ، micro wave ...

* بتعرض الأرقام من 0 إلى F .



بما انه إلها [8 input lines] فكل وحدة من الـ 7-seg مشبوكة على بورت كامل .. كل 2 7-seg بشغلهم بورت واحد عشان نشغل ثنتين على نفس البورت بنستخدم multiplexer عشان يختار واحد منهم حسب الحاجة .

هلا بدنا ناخد مثال انه نشغل رقم (12) [~~واحد واثنين حيث يعرض على الـ 7-seg الموجود على~~] .

بالنسبة للرقم ١ بكون هيك !! فبنكتبه على بورت A 0001 يساوي ١ ..

أما بالنسبة للرقم 2 فبنبعته 0010 = 2 للبورت A ... هلا هاد الرقم كامل 0001 0010 حيث يساوي 12h

~~بالكود~~ .

```
.model small
.code

mov dx,0FF13h
mov al,80h
out dx,al

mov dx,0FF10h
mov al,12h
out dx,al

inc dx
mov al,34h
out dx,al

inc dx
mov al,56h
out dx,al
```

رح يعرض هاد الرقم الثابت

123456

CamScanner

هلا بدنا ناخد مثال لعرض رقم وتغيير هذا تابت :-

بدنا نعرض الأرقام من صفر - 9 على 7-seg واحدة.

```
.model small
.code
mov DX, 0FF13H
mov AL, 80H
OUT DX, AL
mov DX, 0FF10H
mov AL, 1
DISPLAY:
    OUT DX, AL
    Call Delay
    INC AL
    CMP AL, 10
JNE DISPLAY
JMP FINISH
DELAY PROC
mov CX, 0FFFFh
L1: LOOP L1
RET
Delay ENDP
FINISH:
END
```

مثال:- اطلب خانالاب تعمل عدد الأرقام هي الفردية على 7-segment

فاكل كان نبلش عندّ من الرقم 1 لحد الرقم F+2 يعني 11 = B

واطلب بوّال تاني نعرضها 3C على بورت A، B، C ثم

C3 على بورت A، B، C فاكل كونا نكتب كود

زي اللي مكتوب بالدفتر فقط مع تغيير الصيغة يعني

0011 1100 = 3C

1100 0011 = C3

وهكذا ...

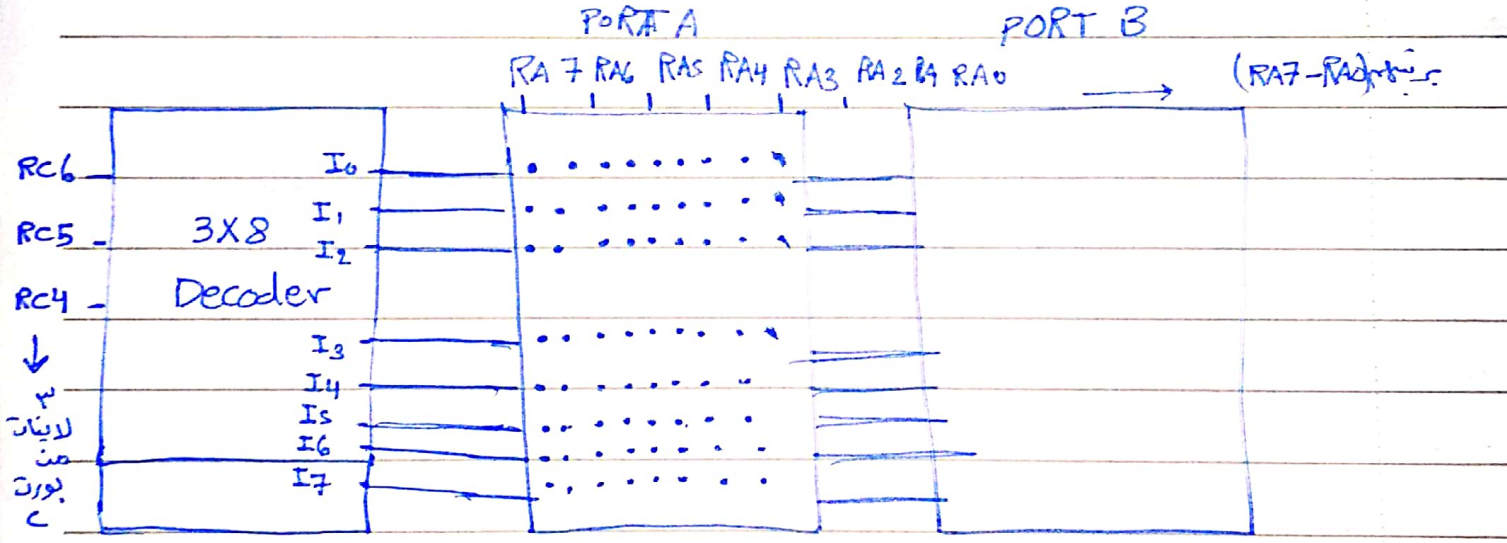
Subject exp(7): dot matrix Date No.

* الدون ماتريكس مشوكلة على نفس الكيفية اللي شرحنا عليها التجربة الماضية ، ح نتعامل مع نفس البورتات A ، B ، C ، Control reg .

* في عنا تنين dotmatrix على الكيفية ، كل وحدة منهم (8X8 Leds) ... يعني كل وحدة منهم فيها (46 Led) .

* بحاج 8 Ports عشان أقدر أ dot matrix (الواحدة) ... بما انه عندي تنين فهيك صرت بحاج 16 Ports .

* هاي الـ dot matrix بتنبرمج زي المصفوفة ، فعشان أقدر أتحكم بكل dot واحدة لازم أفعّل الصف والعمود اللي بتقعوا بهاي النقطة .

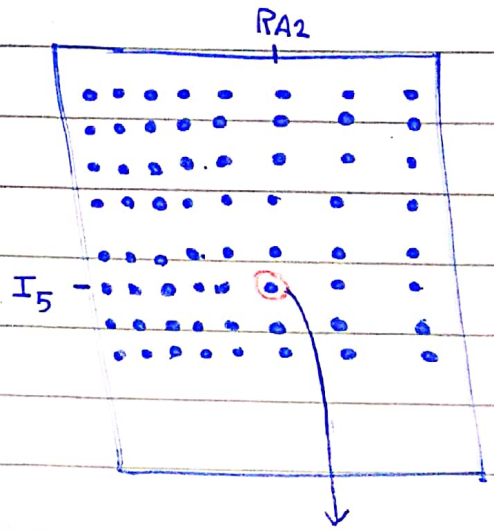


✓ الصف الأول من بورت A والبورت الأول من بورت B ح يضووا مع بعض . كال فعلت ما

✓ وهكذا باقي الصفوف .

NOTEBOOK

Subject Date No.



لو بدنا نضوي هاد الليد I_5 و RA2 لازم نضوي

فالقيمة يلي لازم نحطها على بورت A = 0000 0100 → 0 4

القيمة يلي لازم نحطها على بورت C = 0101 0000 → 5 0

PortC = RC_7 --- RC_2 RC_1 RC_0

0101 0000

5 = 101. → RC_4, RC_6 RC_5

```
.model small
.code
mov   DX , 0FF13 H
mov   AL , 80H
OUT   DX , aL
```

controlreg يعرف على البورتات الابتوت والاوتبوت

```
mov   DX , 0FF10H
mov   aL , 04 H
OUT   DX , aL
```

بعرّف على بورت A شو اللي لازم نعوضي.

```
mov DX , 0FF12 H
mov aL , 50 H
out DX , AL
end
```

NOTEBOOK

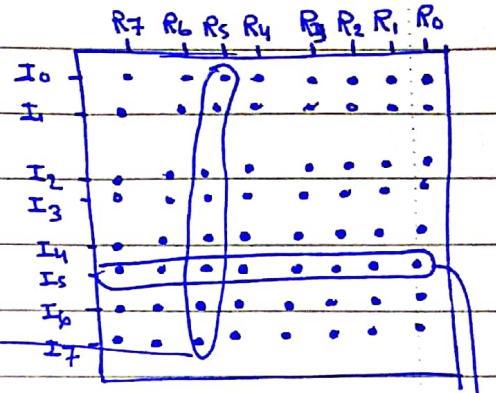
Subject Date No.

```
.model small
.code
mov   DX , 0FF13H
mov   aL , 80H
OUT   DX , aL

mov   DX , 0FF10H
mov   aL , 0FF H
OUT   DX , aL

mov   DX , 0FF12H
mov   aL , 50H      → لو كان بدي أضوي عا الحرف التاني يعني I1
OUT   DX , AL
END                 يعني قيمة البورت C

                    0001  0000
                      1     0
```



لو بدنا نضوي هاد الصف كامل. (1111 1111)

لو بدنا نضوي العمود الثالث هاد كامل بس عل بورت B

```
.model small
.code
mov   DX , 0FF 13 h
mov   AL , 80 h
OUT   DX , AL

mov   DX , 0FF11 h
mov   aL , 20 h
OUT   DX , aL

again: mov   DX , 0FF12h
       mov   aL , 00
L:     OUT   DX , aL
       add   aL , 10H
       cmp   aL , 80 h
       Jne   L
       Jmp   again
END
```

NOTEBOOK

CamScanner