



اسم المادة : هيكلية الحاسوب ولغة أسمبلي

تجمع طلبة كلية التكنولوجيا والعلوم التطبيقية - جامعة القدس المفتوحة

acadeclub.com

وُجد هذا الموقع لتسهيل تعلمنا نحن طلبة كلية التكنولوجيا والعلوم التطبيقية وغيرها من خلال توفير وتجميع **كتب وملخصات وأسئلة سنوات سابقة** للمواد الخاصة بالكلية, بالإضافة لمجموعات خاصة بتواصل الطلاب لكافة المواد:

ل للوصول للموقع مباشرة اضغط **هنا**

وقفكم الله في دراستكم وأعانكم عليها ولا تنسوا فلسطين من الدعاء

الوحدة النووية
مقدمة إلى هيكلة الحاسوب

* مجالات المقامع:

FFFFFF H



يكون من 16 مقطع يستقيم 4 منها. كل
قطاع 64 KB.

ADD AL, BX ← 8 بتة خاطئة

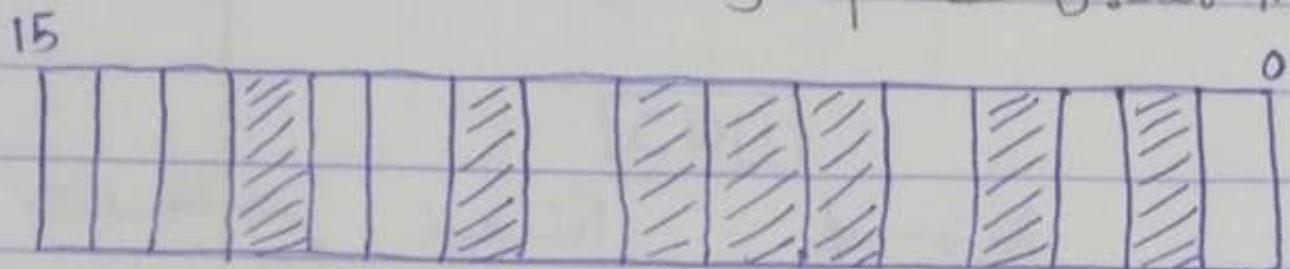
↓ 16 بتة
↑ 8 بتة

mov 0FH, Ax

mov Ax, 0FH

Des. Source انتقل المصدر إلى الوجهة Ax

* مسجل العالم [Flags]:

[illegible]

حساب العناوين

Physical Address = عبارة عن عنوان القطاع + مقدار

الازاحة + عنوان أحد المجلات.

لـ عنوان الـ EA

$$PA = EA + \text{عنوان أحد القطاعات}$$

- حرفا العنونة:

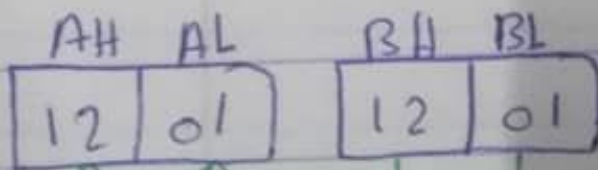
mov Destination, Source

mov DS, SS عبارة خاطئة

mov Ax, B

mov Ax, SS ✓

mov DS, Ax ✓



mov Array1, Array2 X عبارة خاطئة

✓ mov Ax, Array2

✓ mov Array1, Ax تصحيحا

	15	7	0
AX	AH	AL	
BX	BH	BL	
CX	CH	CL	
DX	DH	DL	

علامات
الفرق

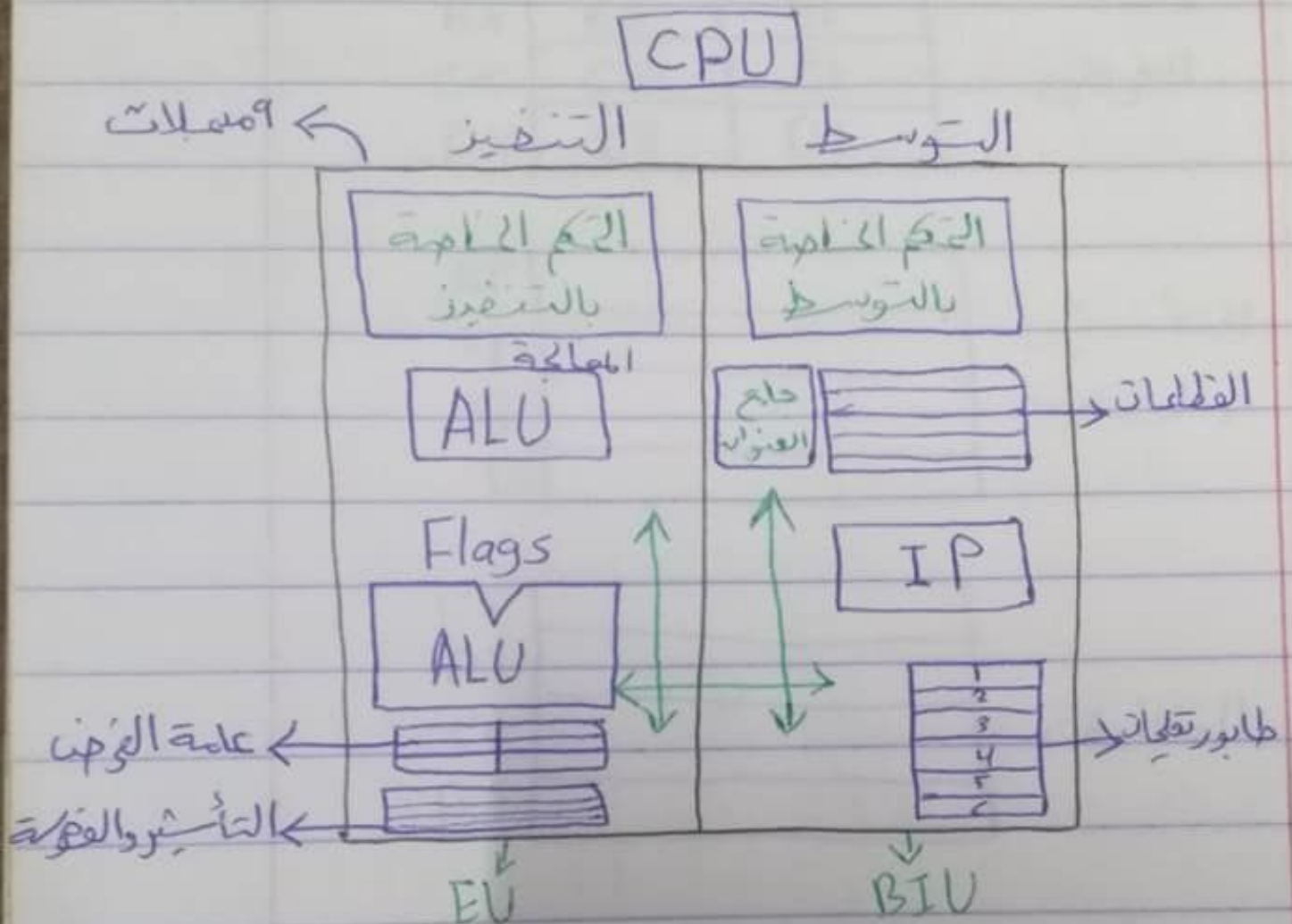
BP	
SP	
SI	
DI	

التأشير
والفرقة

CS	
DS	
SS	
ES	

القطاعات

هيكلة الـ 8086



$$2^{20} \text{ byte} = 2^{10} \text{ KB} = 1 \text{ MB} = \text{سعة الذاكرة}$$

Extra

code

← Data تعريف البيانات

← stuck تعريف مصفوفة

* أدوات العمليات [Operators]:

اجمع Bx ل Ax ADD Ax, Bx
 mov نقلية

- دائماً في مقدمة التعليمة.

- أدوات تستخدم لربط جملة أو متغيرات مع بعضها.

[÷ / * / - / + / NOT / OR / AND]

- pop / push Ax ← وحاصل واحد

(← احفظ قيمتها المتغيرة)

mov Sum, Ax

[1] الأدوات الحسابية: (العمليات الأربعة [÷ / x / - / +], mod)

← (1101001) + (1001111)

← mod ← يعيد باقي الرقم للرقم الثاني

[2] الأدوات المنطقية: (SHL / SHR / OR / AND / Not)

← (10011101) = Not 01100010

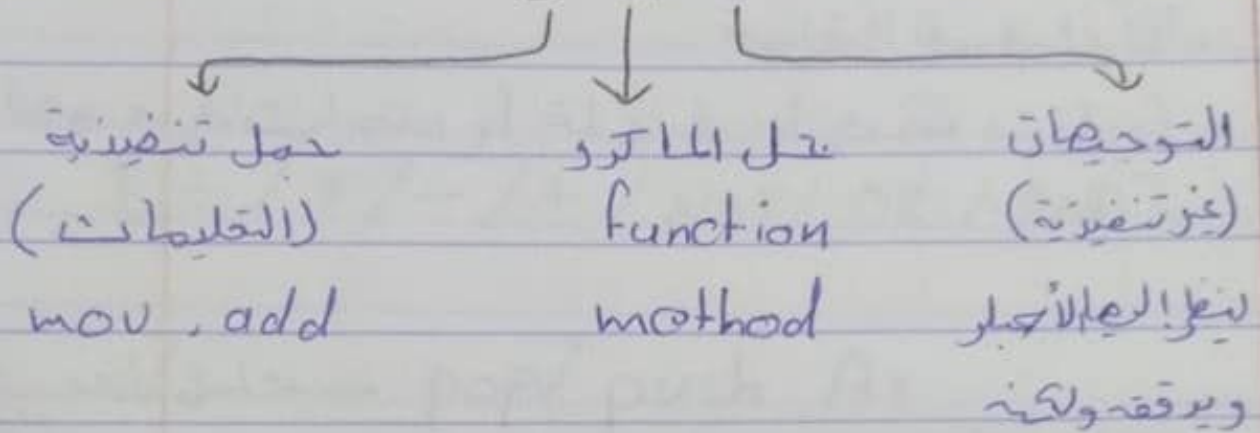
← (0101) SHR (1001111^x) = 01001111

← (0101) SHL (0011111^x) = 00111110

الوحدة الثالثة مقدمة إلى لغة التجميع

- مجموعة الجمل ← البرنامج

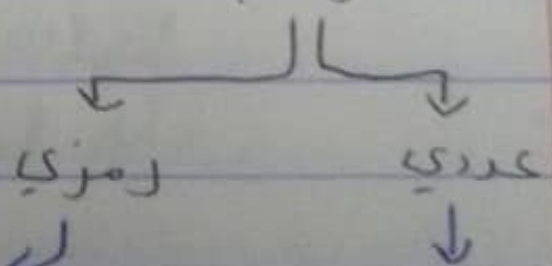
عمل البرنامج



عند التنفيذ يهاها (المسوارات البرنامج).

- يسمح بكتابة الأكواد سواء مخرجة أو كيرة.

- الثوابت



لأن كل ما يكتب بين الحاصرين " "

10011 b

1130

0A13H

1456Q

← الأول ا في رمزاً

← لا يجوز كلمة مخرجة

- حرق العنوان:

mov Ax, Bx

① المسجلات ←

mov Ax, 13H

② الفورية ←

AH	AL
00	13

mov Ax, [1234]

③ المباشرة ←

mov Ax, Array

AL	1234	13H
AH	1235	20H

mov Ax, [Bx]

④ غير المباشرة ←

mov Ax, [SI]

mov Ax, [DI]

⑤ الفهرية

⑥ القاسية

⑦ القائمة الفهرية

⑧ المتناظر

⑨ ال mov string ← الرمزية

movs

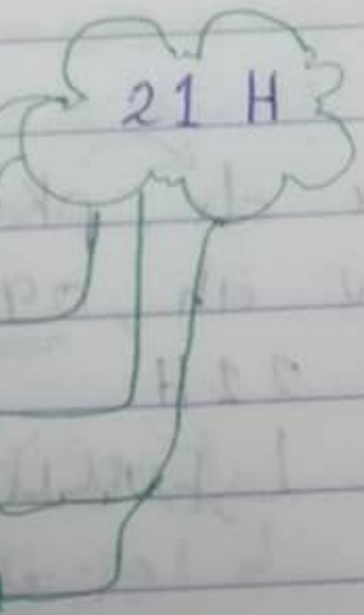
* الاعتراضات :

- الاعتراض 21 H : مجموعة من الحالات - AH

□ حاجة بعض البرامج : 09H

عنوة البرنامج

حاجة نص	AH = 09H
ادخال حرف	AH = 01H
قراءة رقم	AH = 02H
exit	AH = 4CH
	AH = 05H
	AH = 0AH



- mycode.asm mycode.obj mycode.exe

Source file $\xrightarrow{\text{(myasm)}}$ object file $\xrightarrow{\text{link}}$ exc. file

\rightarrow يعني نقل البرمجة لـ .obj

- far = أي يمكن استدعائه من برامج أخرى.

* \swarrow Leax dx, pkey ^{الصف}

mov ah, 09H ^{القيمة}

int 21H

الاعتراض هنا لمراقبة ما يكتبه المستخدم

- الديناميكية في معالجة اذخالات in.

تقنية اذخالات out أو استخدام الاعتراض هنا.

- int = معالجة لستدعي رقم من 0 \rightarrow 255.

توضيح: القطعة

→ Sta Segment البيان للرجوع stack _
 نقطة قاي stack Data _
 DB 64 Dup ("stack") Code _
 → sta ends

البيان عن وجود مجموعة في ذاكرة 64 غير والقيمة التلقائية
 لكل عنصر في stack 63 0 1

stack	stack	stack	stack	stack
-------	-------	-------	-------	-------

T1 DB 64 Dup ('0')

له اجزى في الذاكرة مجموعة T1 تكون من 64 عنصر

عنايت ارجاعي القيمة

SEG T1 ← SEG
 Offset ← مقدار الازاحة
 Type
 Length
 (المتغير)

$$\text{Type} \times \text{Length} = \text{Size}$$

procedure [اجراء]: قریب near

Far کہ دور

macro [ماکرو] •

int 10H -

int 17H -

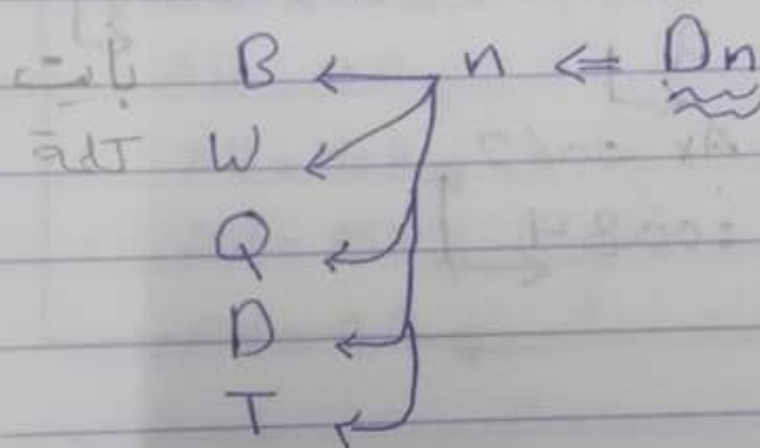
int 16H -

المهام الرئيسية Dx و Dy, 0
وارتباط دھن ای ال طالبہ حقیقی

- عرض عنوان الناقدة $\leftarrow D$

- عرض عنوان الناقدة من الموضع 200 $\leftarrow D$ 200

Notes *



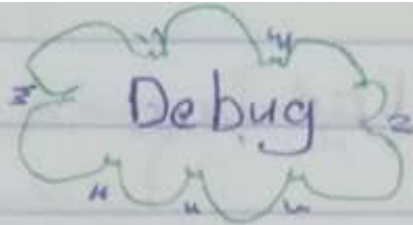
Array DW 64 Dup(?)

- Type Array = 2

- length = 64

- size = Type x length

= 2 x 64 = 128



Windows XP / Windows 7

cmd

```
C:\ADMIN\users> debug  
R  
Ax 0062  
:0084
```

- R ← اعرضا تلك عتويات المجلات

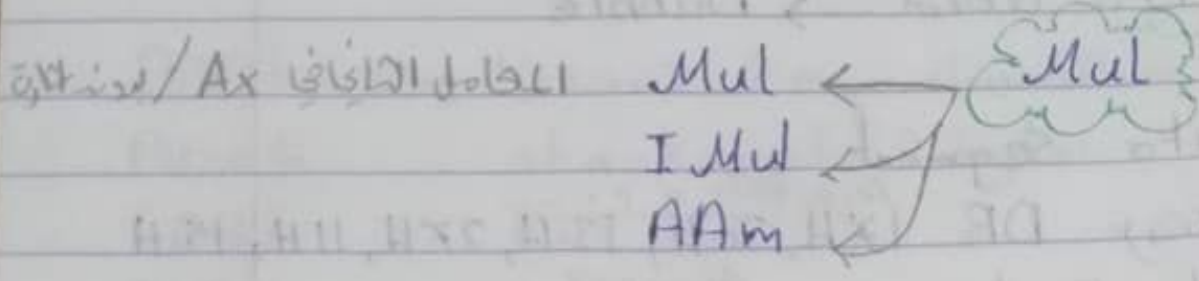
- R + اسم الملف ← اعرضا عتويات الملف
R Ax 0062

: لتغيير عتوة Ax نكتب :-

: تصبح عتوة ال Ax 0084 بدل 0062
: 0084 ←

- RF ← تغيير عتوة F

\overline{Ax} المتعكس



```
mov Ax, 13H
mov Bx, 17H
Mul Bx.
```

الرجوع إلى Bx في السجل الثاني
والرجوع إلى Ax في السجل الأول

CBW

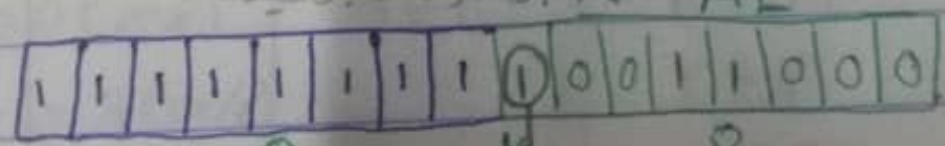
→ Ax ←

CWD

أعمل على حماية واحد الأحرف
حواله 16 بت والثاني 8 بت مثانه انا
أحد الأحرف في سجل حول الثاني

```
CBW ADD Bx, AL
```

AL ← أكبره دونه تأثير قيمته



بناءً على هاي القيمة يعني الثاني

AL + Bx
التي قنا بإضافة

mov Ax, 32H

mov Bx, 14H

* ADD Ax, Bx

$Ax \leftarrow Ax + Bx$ (جاء)

$Ax \leftarrow 32H + 14H$

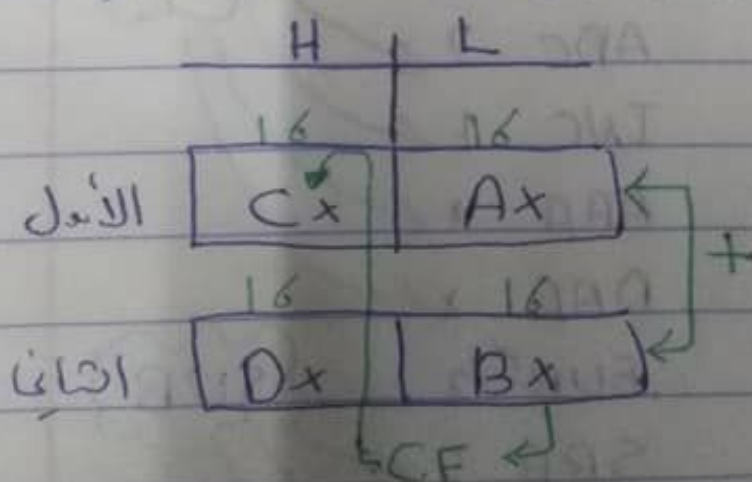
* SUB Ax, Bx

$Ax \leftarrow Ax - Bx$ (جاء)

* ADC Ax, Bx

$Ax \leftarrow (Ax + CF) + Bx$

جمع قيمته الأولى في 32 بيت



ADD Ax, Bx

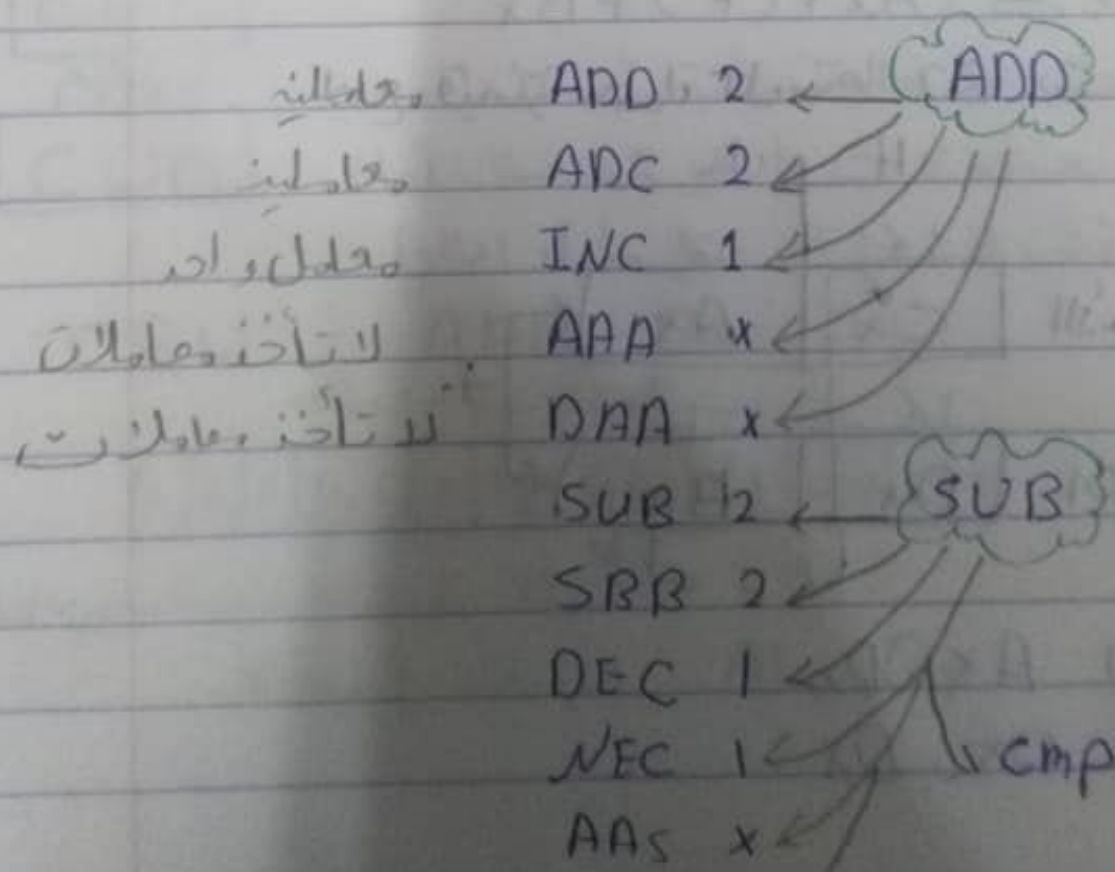
ADC, Cx, Dx

الوحدة الرابعة

تعليمات لغة الآسكي

- 117 تعليمة رئيسية كل تعليمة يتفرع منها مجموعة من التعليمات.

تعليمات حسابية	تعليمات على مستوى الودان الثمانية (البتات)
ADD	1 - تعليمات منطقية.
SUB	2 - تعليمات اراحة.
Mul	3 - تعليمات الدوران [ROL/ROR]
Div	[RCL / RCR]



الوحدة الخامسة
الماتر واستعدادها

* مثال: اكتب برنامج بلغة اسمبلي لاستعداد ماتر و
ناتج المعادلة 6^3 ؟

م

Fact
Exp

Taroub macro F, E

XOR Dx, Dx ← لتصفير المعاملات

mov CX, E

mov BL, F

mov AL, 1 ← لتزنيته الناتج

jcxz L2

L1: mul BL

loop L1

L2: endm

stack segment

DB 64 Dup('stack')

stack ends

Data segment

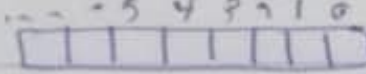
Fact DB 6H

Exp DB 3H

* التعليمات الخاصة بتسجيلات الذاكرة.

POPF : خذ المكرر ولضعه في 2 بايت وحط في ال Flag

PUSHF : stack gets 2 bytes

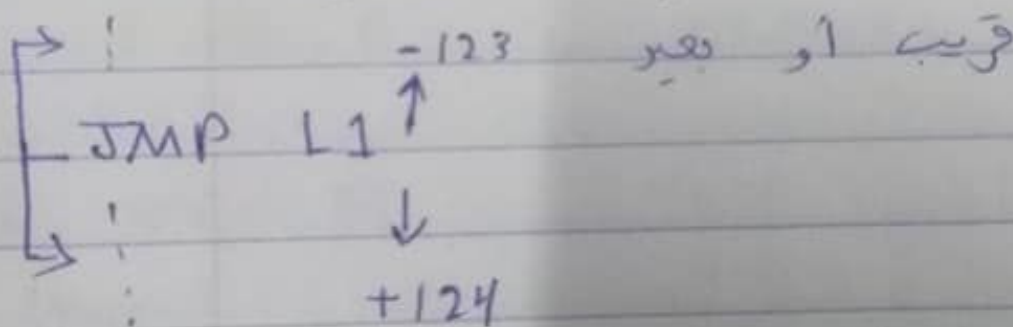
LAHF : 

SAHF

* تعليمات التحكم:

JMP - غير مشروطة :- go to

Code segment : قفز أمالي أو خلفي



* تعليمات التكرار:

تكرار الزيادة	تكرار السلاسل الثابتة	تكرار التكرار
Loop(CX)	REP	REPT
LoopZ(CX/Z)		IRP
LoopNZ		IRPC
(CX, -ZF)		

- CBW Bit \rightarrow Word
- CWD Word \rightarrow Double

Data Segment

Array DB 17H, 20H, 13H, 2FH, 11H, 15H
Data Ends

mov AL, 2

mov BI, Offset Array

LEA BL, Array

xlat Array

Mul 5

mov BL, 5

mul BL

* توجيهات المطبعة :

- SALL → لا قطع شياً
- XALL → قطع لكل التفرقة فقط
- LALL → قطع جميع الكل [ملاحظة / ما تكرر / توصيل والتفرقة]

Allocate macro Length

X = 0

if Length GT OFFH

EXITM

endif

REPT Length

X = X + 1

DB X

endm

enalm

* Taroub macro F, E

Array Label Byte

x = 0

REPT 28

DB 'a' + x

x = x + 1

endm

endm

* Taroub macro F, E

Array Label Byte

IRP A, <0,1,2,3,4,5,6,7,8,9>

DB A

endm

endm

Local L₁, L₂

إذا استخدمنا label في برنامج

ما نكرر أكثر من مرة نعلم

أنه يستخدم التسمية

Local

Data ends
code segment

Assume

pr proc far

karoub fact, Exp

karoub 2, 3

* التوجيهات الخاصة بالماكرو:

1- علامة الفرع [Local / Endm / macro]

2- التكرار [IRPC / IRP / REPT]

العلامة تكرر حتى - endm

3- المطابقة

4- الشرطية if

5- EXITM

Fact! orje 1 -

$$n! = n(n-1)!$$

$$(n-1)! = (n-1)(n-2)!$$

$$Ax \quad 5 \quad 6! = 6 \times 5 \times 4 \times 3 \times 2 \times$$

$$0! = 1$$

دالة "العدد أكبر من صفر"

Fact macro N

mov CX, N

mov AX, 1

mov BX, N

L1: mul BX

Dec BX

loop L2

* ماتر و لسطاعة حرف A دى لسطاعة ؟

print macro

```
move ah, 2
```

```
mov dl, 'A'
```

int 21 H

endom

code segment

assume cs:code

main proc bar

print

```
mov ah, 4ch
```

~~int 21~~

main endp

code ends

end main

البراءات العلية

استاد اجراء في لطيفة من a a a a a

code segment

assume cs:code

main proc far

call print

main endp

print proc near

ret

print endp

code ends

end main

البراءات العلية + البراءات

البراءات العلية + البراءات