هيكلية البرنامج بلغة masm32

```
.386
.model flat,stdcall
option casemap:none
include\masm32\include\windows.inc
include\masm32\include\kernel32.inc
include\masm32\include\masm32.inc
include\masm32\include\debug.inc
includelib\masm32\lib\kernel32.lib
includelib\masm32\lib\masm32.lib
includelib\masm32\lib\debug.lib
```

ملاحظة // هذه الملفات والمكتبات يجب تضمينها في كل برنامج.

.data

في هذا الجزء يتم التصريح عن المتغيرات (مواقع الذاكرة) في البرنامج ويشبه حقل (var) في لغة باسكال

.code

start:

في هذا الجزء يتم كتابة الايعازات الخاصة بالبرنامج ويشبه (begin) في لغة باسكال

يجب وضع هذه العبارة قبل نهاية البرنامج. **invoke ExitProcess,0**

تمثل نهاية البرنامج وتشبه (.end) في لغة باسكال. **end start**

أنواع المتغيرات

توجد لدينا ثلاث أنوع من المتغيرات أو مواقع الذاكرة وحسب المساحة الخزنية وتقاس بالبت والأنواع هي:-

1- **db** وهي مختصر لعبارة (data byte) وحجمها **8 bit .**
2- **dw** وهي مختصر لعبارة (data word) وحجمها **16 bit .**
3- **dd** وهي مختصر لعبارة (data double) وحجمها **32 bit .**

كيفية التصريح عن المتغيرات

لقد تعلمنا في بعض اللغات كيفية التصريح عن المتغيرات فمثلاً في لغة باسكال عندما نريد أن نصرح عن متغير يكون بحقل (var) وبالصيغة التالية

```
var
x:integer;
```

أي أن الصيغة تكون **; النوع : أسم المتغير**

أما بلغة (masm32) يكون التصريح عن المتغيرات في حقل (data) كما ذكرنا سابقاً وبالشكل التالي:-

أسم المتغير النوع القيمة الابتدائية

وكما في الأمثلة التالية:-

```
.data
r1 dd 30h
r2 dw 20
k  db 00111011b
```

ملاحظة//لغة (masm32) تتعامل مع نظام (**hexadecimal**) ففي التعريفات أو التصريحات السابقة لاحظنا وجود ثلاث صيغ في حالة ذكر الرقم فقط معنى ذلك أن الرقم بالنظام العشري (اللغة تحوله إلى نظام 16) أما أذا جاء بعد الرقم حرف (**h**) معنى ذلك أن الرقم مكتوب بنظام 16 أما أذا كان الرقم عبارة عن (0,1) وفي نهايته حرف (**b**) معنى ذلك أن الرقم مكتوب بالنظام الثنائي .

ملاحظة// للتحويل من النظام العشري إلى النظام ال16 نقسم على العدد 16 فمثلاً كيف تخزن قيمة r2 **r2 dw 20**

16	**20**	**4**
16	**1**	**1**
	0	

يكتب الرقم من الأسفل إلى الأعلى ومن اليسار إلى اليمين أي أن قيمة **r2** هي **14h**
أما التحويل من الثنائي إلى ال16 نأخذ كل أربع bits ونضع بمكانها الرقم الذي يقابلها بنظام ال16 فمثلا

k db 00111011b

1011 يمثل الرقم (11 أي **B**)
0011 يمثل الرقم (**3**)

أي أن قيمة k هي 3B

أيعاز الطباعة (PrintHex)

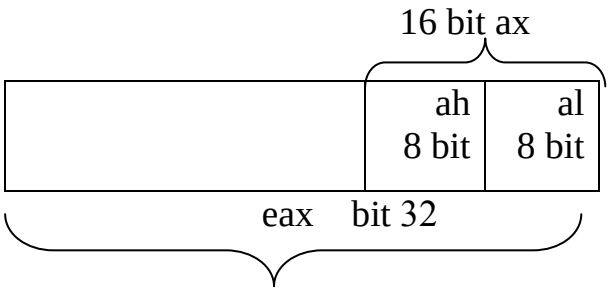
يستخدم هذا الإيعاز للطباعة على شاشة التنفيذ وتكون الطباعة للمسجلات أو مواقع الذاكرة (المتغيرات) ذات الحجم 32bit وتكون النتيجة بنظام hexadecimal حيث أن حرف(P,H) يجب أن تكتب كبيرة.

أنواع المسجلات (Register)

المسجل (register):- عبارة عن مكان خزني معروف حجمه لدى اللغة وهنالك ثلاث أنوع من المسجلات:-

1- مسجلات ذات حجم 32 bit وهي :- eax,ebx,ecx,edx,esi,edi,esp,ebp
2- مسجلات ذات حجم 16 bit وهي :- ax , bx , cx , dx , si , di , sp , bp
3- مسجلات ذات حجم 16 bit وهي :- ah , al , bh , bl , ch , cl , dh ,dl

شكل توضيحي يمثل **eax register**

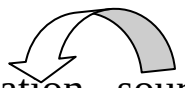


أيعازات النقل (Transfer Instruction)

1- mov instruction

يستخدم هذا الإيعاز لنسخ محتويات (source إلى destination) حيث أن :-
source :- المكان الذي تخرج منه القيمة.
destination:- المكان الذي تستقر فيه القيمة .

والصيغة العامة لهذا الإيعاز هي :-



mov destination , source

ملاحظة//1- قيمة ال source لاتتغير.
2-يجب أن يكون source=destination متساوين بالحجم .

الصيغ المقبولة لإيعاز mov

1- mov register, register ex\\ mov eax,ebx
2- mov register, memory ex\\ mov ax,var1
3- mov memory, register ex\\ mov y,edx
4- mov register,قيمة مباشرة ex\\ mov ah,70h
5- mov memory,قيمة مباشرة ex\\ mov k1,55h

ملاحظات

1- لايجوز النقل بين موقعي ذاكرة (متغيرين)

mov memory, memory
ex\\ mov x, r1 false

2- cs, eip, ip لا يمكن أن يكونوا destination

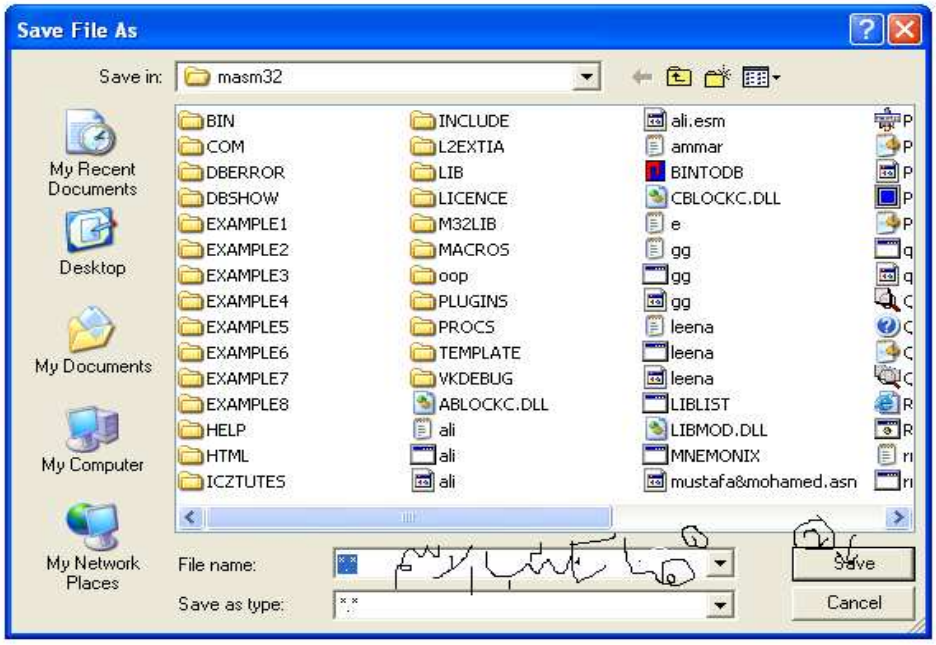
mov eip , eax false

س// أكتب برنامج بلغة masm32 لنسخ محتويات موقع ذاكرة وقيمته (10h) إلى مسجل (register) مثل eax؟

```
.386
.model flat,stdcall
option casemap:none
include\masm32\include\windows.inc
include\masm32\include\kernel32.inc
include\masm32\include\masm32.inc
include\masm32\include\debug.inc
includelib\masm32\lib\kernel32.lib
includelib\masm32\lib\masm32.lib
includelib\masm32\lib\debug.lib
.data
r1 dd 10h
.code
start:
mov eax,r1
PrintHex eax
invoke ExitProcess,0
end start
```

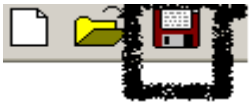
كيفية خزن البرنامج

من قائمة file نختار save as أو save سوف تظهر لنا النافذة التالية:-



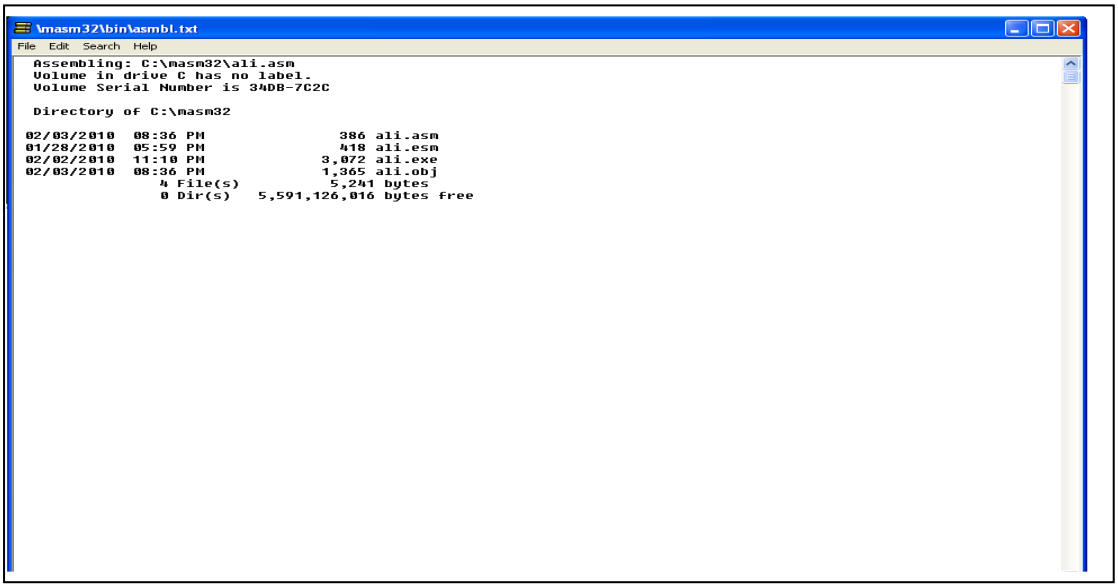
في حقل file name نكتب الاسم ويجب كتابة**asm.** بعده مثلاً أسم البرنامج ali نكتب في حقل file name **ali.asm** وبعدها نضغط على save أو على (Enter) من (Keyboard) .

ملاحظة// بعد كل تعديل أو إضافة يجب الضغط على أداة save الموجودة في شريط الأدوات لحفظ التغيرات .



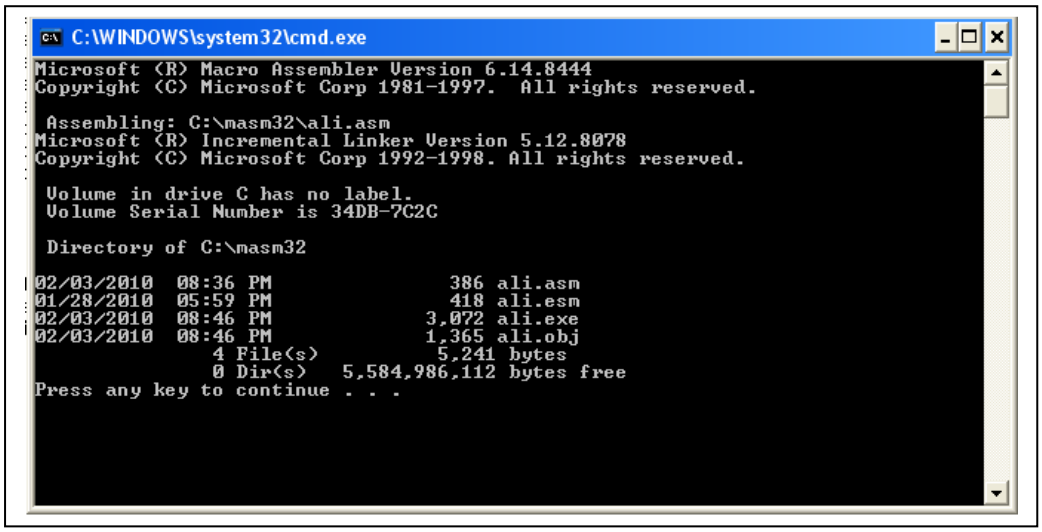
كيفية تنفيذ البرنامج

1- من قائمة Project نختار Assemble ASM file سوف تظهر لنا النافذة التالية :-



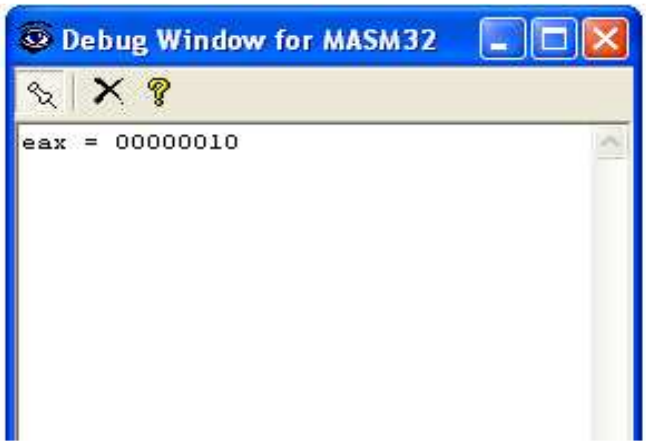
أذا ظهرت بعد عبارة Assembling مباشرةً عبارة Volume in drive معنى ذلك أن التنفيذ في الخطوة الأولى صحيح ننتقل إلى الخطوة الثانية، أما أذا لم تظهر معنى ذلك أن البرنامج فيه خطأ نصححه ونضغط على الأداة save ثم نعيد التنفيذ .

2- من قائمة Project نختار Assemble & Link سوف تظهر لنا النافذة التالية:-



أذا لم تظهر قبل عبارة Press any key to continue كلمة error معنى ذلك أن التنفيذ في الخطوة الثانية صحيح أما أذا ظهرت كلمة error نغلق النافذة ونصحح البرنامج ونضغط على الأداة save ونعيد الخطوة الأولى والثانية.

3- من قائمة Project نختار الخيار الأخير وهو Run program سوف يظهر لنا ناتج التنفيذ كما في النافذة التالية :-



ملاحظة// نظام hexadecimal كل عدد فيه يحتاج إلى **4 bit** لذلك يكون عدد الأرقام التي يأخذها موقع الذاكرة (المتغير) أو المسجلات من نوع 8 bit **رقمين** أما ذات حجم 16 bit **أربع** أرقام و ذات حجم 32 bit **ثمانية** أرقام .
ملاحظة//في حالة وجود عدد أرقام أقل من الأرقام المحددة نضع **أصفار** بقدر العدد المطلوب من جهة اليسار وكما في المثال الآتي:-

```
mov  ax,16h
```

في هذه الحالة تكون قيمة ax هي **0016h** لان ax ذات حجم 16 bit ويحتاج إلى أربع أرقام .

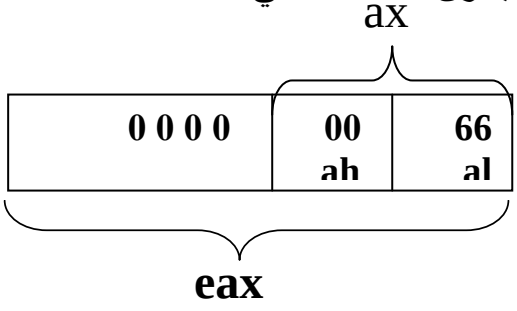
```
mov  eax,2h
```

في هذه الحالة تكون قيمة eax هي **00000002h** لان eax ذات حجم 32 bit ويحتاج إلى ثمانية أرقام .

مجموعة أمثلة :-

1- mov al,66h

الرقم يخزن بالشكل التالي



eax=00000066
ax=0066
ah=00
al=66

2- mov ebx , 40

في هذه الحالة يجب تحويل الرقم إلى نظام (hexadecimal)
تكون القيمة المقابلة للرقم 40 هي 28h

16	40	
16	2	8
	0	2

ebx=00000028
bx=0028
bh=00
bl=28

3- mov dh,9h
edx=00000900
dx=0900
dh=09
dl=00

كيفية التخلص من السالب

1- يجب أن يكون الرقم بنظام (Hexadecimal) وإذا لم يكن نحوله كما ذكرنا سابقاً.
2- تحويل الرقم المعطى بالسالب إلى النظام الثنائي (binary system) وحسب عدد الأرقام المطلوبة لكل حجم.
3- نستخرج المتمم (2'complement) وذلك بقلب كل 0 إلى 1 وكل 1 إلى 0.
4- نجمع مع 1 (0+0=0 ، 0+1=1 ، 1+0=1 ، 1+1=0 وباليد 1).
5- نحول العدد الناتج إلى (hexadecimal).

مجموعة أمثلة :-

1- mov ax,-30h

1- العدد بنظام السادس عشر ويكتب 0030 لان ax ذات حجم 16bit ويحتاج إلى أربع أرقام
2- نحول العدد إلى النظام الثنائي فيصبح:-

0000 0000 0011 0000

3- المتمم هو 1111 1111 1100 1111

4- نجمع مع الرقم + 1

1111 1111 1101 0000

5- الناتج هو F F D 0

eax=0000FFD0
ax=FFD0
ah=FF
al=D0

2- mov eax,-3h
بتطبيق نفس الخطوات السابقة يكون الناتج هو:-
eax=FFFFFFFD
ax=FFFD
ah=FF
al=FD

3-mov eax,-1
eax=FFFFFFFF
ax=FFFF
al=FF
ah=FF

س// أكتب برنامج بلغة masm32 للتبديل بين محتويات two register الأول eax وقيمته 60h والثاني ebx وقيمته 90h؟

```
.386
.model flat,stdcall
option casemap:none
include\masm32\include\windows.inc
include\masm32\include\kernel32.inc
include\masm32\include\masm32.inc
include\masm32\include\debug.inc
includelib\masm32\lib\kernel32.lib
includelib\masm32\lib\masm32.lib
includelib\masm32\lib\debug.lib
.data
.code
start:
mov eax,60h
mov ebx,90h
mov ecx,eax
mov eax,ebx
mov ebx,ecx
PrintHex eax
PrintHex ebx
invoke ExitProcess,0
end start
```

مجموعة واجبات (H.W)

س[1]// أكتب برنامج بلغة masm32 يقوم بالتبديل بين موقعي ذاكرة الأول هو k وقيمته 30h والثاني n وقيمته 10h باستخدام أيعاز mov؟

س[2]// نفذ الخطوات التالية وأعطِ النتائج ؟

1- mov ax,9Bh
eax=?
ax=?
ah=?
al=?
2- mov bh,1001b
ebx=?
bx=?
bh=?
bl=?
3- mov edx,-19
edx=?
dx=?
dh=?
dl=?
4-mov ah,10h
mov bx,ax
ebx=?
bx=?
bh=?
bl=?
5- mov dx,-30
mov bh,dh
ebx=?
bx=?
bh=?
bl=?

س3//أذا علمت أن المتغيرات معرفة بالشكل التالي :-

```
.data
r1  db  00000010b
r2  dw  20h
r3  dd  10h
```

فأي من العبارات التالية صحيحة وأي منها خاطئة (أذا كانت صحيحة أعطِ الناتج وأما أذا كانت خاطئة أعطِ سبب الخطأ)؟

```
1- mov  ax,r1
2- mov  ax,r2
3- mov  eax,r3
4- mov  r2,r1
5- mov  r2, dx
```

2- Xchg Instruction

يستخدم هذا الإيعاز للتبديل بين محتويات معاملين (two operand) والصيغة العامة له هي :-

Xchg destination , source

الصيغ المقبولة لهذا الإيعاز هي :-

```
1- Xchg  reg , reg          ex\\xchg  eax,ebx
2- Xchg  reg , mem          ex\\xchg  ax,var1
3- Xchg  mem , reg          ex\\xchg  r1,ah
```

ملاحظات//

1- يجب أن تكون المعاملات متساوية بالحجم.
2- لايمكن التبديل بين موقعي ذاكرة.

س// أكتب برنامج يقوم بتبديل محتويات مسجلين هما eax وقيمته 66h و edx وقيمته 39h باستخدام أيعاز Xchg؟

```
.code
   start:
     mov  eax,66h
     mov  edx,39h
     PrintHex  eax
     PrintHex  edx
    Xchg  eax,edx
    PrintHex  eax
    PrintHex  edx
    invoke  ExitProcess,0
 end  start
```

مجموعة واجبات(H.W):-

س[1]// أكتب برنامج بلغة masm32 يقوم بالتبديل بين موقعي ذاكرة الأول هو r1 وقيمته 220h والثاني r2 وقيمته 100h باستخدام أيعاز Xchg؟

س[2]// نفذ الخطوات التالية وأعطِ النتائج ؟

```
1- mov  eax,-40
   xchg  ah,al

2- mov  ebx,0ffh
   mov edx,-2Ah
   xchg  bx,dx

3-mov ah,-7
  mov ax,-30
  xchg  eax ,ebx
```

3- movzx {Mov with Zero Extend}

يقوم هذا الإيعاز بنسخ محتويات إل (source) إلى (destination) ويقوم بتوسيع القيمة بإضافة أصفار إلى جهة اليسار ليصبح الحجم أما 16 bit أو 32 bit

أي أن :- **destination > source**

والصيغة العامة له هي:-

movzx destination , source

والصيغ المقبولة لهذا الإيعاز هي :-

1- movzx r32,r8 or m8 ex\\ movzx eax,bl
2- movzx r32,r16 or m16 ex\\ movzx edx,r
3- movzx r16,r8 or m8 ex\\ movzx bx,al

ملاحظة// قيمة إل(source) أقل من (destination) ولا تتغير .

مثال// أعطِ النتائج التالية :-

```
mov bx,0A69Bh
movzx eax,bx ⟶ eax =0000A69B
movzx edx,bl ⟶ edx =0000009B
movzx cx,bh ⟶ cx = 00A6
```

ملاحظة// رقم 0 الذي يسبق الرقم في المثال السابق دلالة على أن حرف A هو رقم.

4- movsx {Mov with sign Extend}

يقوم هذا الإيعاز بنسخ محتويات إل (source) إلى (destination) ويقوم بتوسيع القيمة بإضافة **أصفار** إلى جهة اليسار في حال كانت **آخر bit صفر** وإذا كان **آخر bit هو 1 يضيف (s)1** أي أن مقابل كل **أربع واحدات يضيف الحرف F** ليصبح الحجم أما 16 bit أو 32 bit

أي أن :- **destination > source**

والصيغة العامة له هي:-

movsx destination , source

والصيغ المقبولة لهذا الإيعاز هي :-

1- movsx r32,r8 or m8 ex\\ movsx eax,bl
2- movsx r32,r16 or m16 ex\\ movsx edx,r
3- movsx r16,r8 or m8 ex\\ movsx bx,al

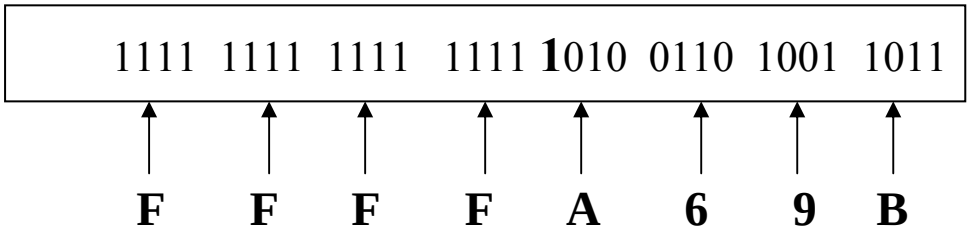
ملاحظة// قيمة إل(source) أقل من (destination) ولا تتغير .

مثال// أعطِ النتائج التالية :-

```
mov bx,0A69Bh
movsx eax,bx ⟶ eax =FFFFA69B
movsx edx,bl ⟶ edx =FFFFFF9B
movsx cx,bh ⟶ cx = FFA6
```

توضيح

قيمة bx هي A69B نلاحظ أن آخر رقم هو A نحول الرقم A إلى النظام الثنائي 1010 ⟵ نلاحظ أن آخر bit من جهة اليسار هو واحد أذاً نضيف واحدات ليصبح الحجم 32 bit كما بالشكل التالي :-



مجموعة واجبات(H.W):-

1- mov bl,-7
   movzx ax,bl ax=?
2- mov ax,901Ch
   movzx edx,ah edx=?
3-mov ax,-30
  movsx ebx,ax ebx=?
4- mov ah,-fh
   movsx bx,ah bx=?

Arithmetic Instruction

1- Inc Instruction

يقوم هذا الإيعاز **بإضافة 1** إلى (operand) .

والصيغة العامة لهذا الإيعاز هي :-

1- inc reg ex\\ inc ax
2- inc mem ex\\ inc r

مجموعة من الأمثلة :-

1- mov ax,30h
inc ax ⟶ ax =0031h

2- mov ax,30
inc ax

في هذه الحالة يجب تحويل الرقم 30 إلى نظام (hexadecimal)

16	30	
16	1	E
	0	1

أي أن قيمة ax =1E وبعد تطبيق أيعاز inc تصبح القيمة ax = 001F

مجموعة واجبات(H.W):-

1- mov ax ,-30
inc ax ax=?
2- mov bx ,7
inc bx bx=?
3- mov al,17
inc al al=?
4- mov ah,-1
inc ax ax=?
5- mov al,FF
inc ax ax=?

2- dec Instruction

يقوم هذا الإيعاز **بإنقاص 1** من ال (operand) .

والصيغة العامة لهذا الإيعاز هي :-

1- dec reg ex\\ dec ax
2- dec mem ex\\ dec r

مجموعة من الأمثلة :-

1- mov al,1
dec al ⟶ al = 00
2- mov ah,7
dec ah ⟶ ah =06
3- mov ax,40
dec ax

في هذه الحالة يجب تحويل الرقم 40 إلى نظام (hexadecimal)

16	40	
16	2	8
	0	2

أي أن قيمة ax =28h وبعد تطبيق أيعاز dec تصبح القيمة ax = 27h

4- mov al,-7
dec al

نتخلص من السالب

0000 0111
1111 1000
1+
1111 1001
1-
1111 1000

F 8

al=F8

مجموعة واجبات(H.W):-

1- mov al,-30
dec al al=?
2- mov bx,-79
dec bh bh=?

3-Add Instruction

يقوم هذا الإيعاز بإضافة محتويات ال(source) إلى (destination) والصيغة العامة له هي :-

add destination, source

ملاحظات//

1- لايجوز جمع كميتين غير متساويتين بالحجم.
2- لايجوز الجمع بين موقعي ذاكرة.
3- عمليات الجمع(add) والطرح(sub) والزيادة والنقصان بمقدار واحد (inc&dec) تؤثر على (flag register).
4-(0=0+0) ، (1=1+0) ، (1=0+1) ، (1+1=0 وباليد 1).

مجموعة من الأمثلة :-

1- mov r1,20h
mov eax,40h
add eax,r1 ⟶ eax = 00000050

```
0010 0000
0100 0000  +
0110 0000
  ↓    ↓
  6    0
```

2- mov bl,40
mov ah,32
add ah,bl

16	40	
16	2	8
	0	2

bl= 28h

```
0010 1000
0010 0000
0100 1000
  ↓    ↓
ah =  4    8
```

16	32	
16	2	0
	0	2

ah=20h

مجموعة واجبات(H.W):-

```
1- mov  bl,-20
   mov  al,20
    add  al,bl
2- mov  b,-56
    mov eax,0ACD2h
     add  ah,bh
3- mov  bl,5
   mov ah,-3
   add  eax,ebx
```

4-sub Instruction

يقوم هذا الإيعاز بطرح محتويات ال(source) من ال(destination) والصيغة العامة له هي :-

sub destination, source

ملاحظة//تطبق عليه نفس شروط الإيعاز add.

كيفية أجراء عملية الطرح

1- نحول ال (source) و ال(destination) إلى النظام الثنائي .
2- نستخرج متمم ال(source) وذلك بقلب 0 إلى 1 و1 إلى 0.
3- نجمع ال(destination) مع متمم ال(source).
4- نجمع الناتج مع ال (carry) أو مع الواحد.
5- نحول النتيجة النهائية إلى نظام ال(hexadecimal).

مجموعة من الأمثلة :-

```
1- mov  al,20h
   sub  al,20h
    al=00
```

```
(source)  0010  0000
(destination)  0010  0000
(source) متمم  1101  1111+
-----------
1111 1111
          1+
-----------
0000  0000
 ↓     ↓
 0     0
```

2- mov eax,70h
mov ebx,30h
sub eax,ebx
eax = 00000040

(source) 0011 0000
(destination) 0111 0000
(source) متمم 1100 1111+

1 0011 1111
1+
0100 0000
4 0

3- mov al,66h
mov bh,50h
sub al,bh ⟶ al=16h

4- mov eax,80h
mov ebx,60h
sub eax,ebx ⟶ eax = 00000020

ملاحظات//

1- في حالة (**destination >= source**) لاتوجد مشكلة ونحل حسب الخطوات السابقة.

2- يجب مراعاة أن يكون العدد مكتوب بنظام (hexadecimal) وأن يكون موجب وأن لم يكن نحوله إلى نظام (hexadecimal) ونتخلص من السالب كما بينا سابقاً.

3- - في حالة (**destination < source**) نطبق نفس الخطوات السابقة ولكن يعتبر العدد الناتج بالسالب نطبق عليه خطوات كيفية التخلص من السالب وكما في الأمثلة التالية:-

1- mov ah,30h
mov bh,70h
sub ah,bh
ah = C0

(source) 0111 0000
(destination) 0011 0000
(source) متمم 1000 1111+

1011 1111
1+
0100 0000
– (4 0)

0100 0000
1011 1111
1+
1100 0000
C 0

2- mov bx,-30h
mov cx,20h
sub bx,cx

في هذه الحالة يجب التخلص من السالب (-30)

0030 ← 0000 0000 0011 0000
1111 1111 1100 1111
+1

(destination) bx ← 1111 1111 1101 0000
(source) cx ← 0000 0000 0010 0000
(source) متمم ← 1111 1111 1101 1111

نجمع (destination) مع متمم (source)

(destination) 1111 1111 1101 0000
(source) متمم 1111 1111 1101 1111
1 1111 1111 1010 1111
+1

1111 1111 1011 0000

bx = F F B 0

مجموعة واجبات(H.W):-

1- mov al,79h
sub al,-1
al=?

2- mov eax,-64
mov bx,-13
sub ebx,eax
ebx=?

3- mov bx,0A332h
mov ax,60
sub ah,bh
ah=?

4- mov eax,-22
sub ax,-3
ax=?

5-Neg Instruction

يستخدم هذا الإيعاز لعكس أشارة العدد المدخل (يعمل له 2'complement)
والصيغة العامة لهذا الإيعاز هي :-

1- neg reg ex\\ neg ax
2- neg mem ex\\ neg r

ملاحظة// استخدامه نفس خطوات **التخلص من السالب** وكما في الأمثلة التالية:-

1-mov al,5h
neg al
al=FB

0000 0101
1111 1010
+1
1111 1011
F B

2-mov al,-5h
neg al
al=05

3- mov eax,30h
neg ax
ax=0000FFD0

4- mov ah, 30h
neg ah
eax=0000D000

مجموعة واجبات(H.W):-

1- mov ax,-33
neg al
al=?
2- mov eax ,88
neg ax
ax=?
3- mov bh,40
neg bh
neg bx
bx=?

س[1]// أكتب برنامج بلغة masm32 لحل المعادلة التالية مع أعطاء النتائج في كل خطوة:-

r = -x +(y-z)

علماً أن x =10h وy=30h و z=60h والمتغيرات ذات حجم 32 bit.

```
.data
x  dd  10h
y  dd  60h
z  dd  30h
r  dd  00h
.code
   start:
     mov  eax,x
     neg  eax
     PrintHex  eax⟶ eax=FFFFFFF0
     mov  ebx,y
     sub  ebx,z
     PrintHex  ebx⟶ebx=00000030
     add  eax,ebx
     mov  r,eax
     PrintHex  r⟶⟶ r=00000020
     invoke  ExitProcess,0
   end  start
```

س[2]// أكتب برنامج بلغة masm32 لحل المعادلة التالية مع أعطاء النتائج في كل خطوة:-

r = - (x -(y-z)+n)

علماً أن x =10h وy=20h و z=30h و n=40h والمتغيرات ذات حجم 32 bit.

```
  .data
x  dd  10h
y  dd  20h
z  dd  30h
n  dd  40h
r  dd  00h
.code
   start:
     mov  eax,y
     sub  eax,z
```

```
 PrintHex  eax→ eax=FFFFFFF0
mov  ebx,x
sub  ebx,eax
 PrintHex  ebx→ ebx=00000020
 add  ebx,n
  neg   ebx
 mov  r,ebx
 PrintHex  r————→ r=FFFFFFA0
 invoke  ExitProcess,0
end  start
```

مجموعة واجبات(H.W):-

1- أكتب برنامج بلغة masm32 لحل المعادلة التالية مع أعطاء النتائج في كل خطوة:-

k = - (x +y-(z-n))

علماً أن x=22 وy=20h و z=3 و n=40h والمتغيرات ذات حجم 16 bit.

2- أكتب برنامج بلغة masm32 لحل المعادلة التالية وبدون استخدام أيعاز sub مع أعطاء النتائج في كل خطوة:-

r = x-y-z

علماً أن x=44 وy=50 و z=30h و المتغيرات ذات حجم 32 bit.

3- أكتب برنامج بلغة masm32 لحل المعادلة التالية مع أعطاء النتائج في كل خطوة:-

z=(x+y)-(r-n)

علماً أن x =11h وy=100h و r=30 وn=55h والمتغيرات ذات حجم 32 bit.

4- أكتب برنامج بلغة masm32 لحل المعادلة التالية مع أعطاء النتائج في كل خطوة:-

z=-y+r-(n+w)

علماً أن y =10h وr=606h و r=30 وn=55h و w=40 والمتغيرات ذات حجم 32 bit.

5- أكتب برنامج بلغة masm32 لحل المعادلة التالية مع أعطاء النتائج في كل خطوة:-

z=(2x+2y)-(r+(-n))

علماً أن x =1h وy=10h و r=30h وn=5h والمتغيرات ذات حجم 32 bit.