

الهيكل العام لبرنامج مكتوب بلغة التجميع في بيئة visual studio 2012 :

تمتلك البرامج المكتوبة بلغة التجميع هيكل بسيط وكما موضح أدناه:

Include irvine32.inc

.data

(Insert variable here)

.code

Main proc

(insert executable instruction here)

Exit

Main endp

End main

وفي ما يلي توضيح لهيكل البرنامج العام:

١. العبارة الموجهة **include** تقوم بنسخ تعريفات ومعلومات التنصيب من ملف نصي اسمه

irvine32.inc

٢. تشير العبارة **(.data)** إلى بداية مقطع البيانات حيث يتم في هذا الجزء التصريح عن جميع المتغيرات التي سوف تستخدم في البرنامج لاحقاً.

٣. تشير عبارة التوجيه **(.code)** إلى بداية مقطع الـ **code** حيث توجد كل العبارات القابلة للتنفيذ.

٤. تشير الخطوة **(main proc)** إلى بداية الإجراء **(procedure)** ويتم إعطاء إي اسم لهذا الإجراء على سبيل المثال كلمة **main** بشرط إن يكتب نفس الاسم في المواقع التي تم كتابتها باللون الأحمر.

٥. تقوم العبارة **(Exit)** باستدعاء إحدى دوال **Microsoft windows** التي تعمل على إيقاف البرنامج لإنهاء التنفيذ ، نلاحظ إن العبارة **Exit** ليست كلمة مفتاحية وإنما هي عبارة عن **Macro command** معرف في ملف **irvine32.inc** والذي يوفر طريقة لإنهاء البرنامج .

٦. تشير عبارة **(Endp)** إلى نهاية الإجراء.

٧. يشير الإجراء **(End)** إلى آخر سطر في برنامج المجمع.

ايعازات نقل البيانات Data Transfer :**أولاً:- الإيعاز mov:**

يقوم الإيعاز mov بنسخ البيانات مع معامل المصدر Source operand إلى معامل الهدف Destination operand والصيغة العامة له هي :

mov destination, source

حيث إن الإيعاز mov يحتاج إلى معاملين ، المعامل الأول من جهة اليمين هو الهدف والمعامل الثاني هو المصدر، من الممكن إن تتغير محتويات الهدف إما المصدر فلا تتغير محتوياته وعمل هذا الإيعاز هو مشابه لعبارة الإحلال في لغة و C++ .

ملاحظة:

يجب مراعاة القواعد الآتية عند استعمال الإيعاز mov:

١. كلا المعاملين يجب إن يكونا بنفس الحجم.
٢. لا يمكن لكلا المعاملين إن يكونا معاملات ذاكرة.
٣. لا يمكن للمسجلات CS,EPI,IP إن تكون معاملات هدف.
٤. إي قيمة ثابتة لا يمكن نقلها إلى مسجلات القطع (DS,SS,ES.CS)
٥. المعامل المصدر ممكن إن تكون حجم بياناته اقل من أو يساوي حجم معامل الهدف والأمثلة التالية توضح هذه النقطة :

mov dl,45h	مقبولة
mov dx,456h	
mov dl ,al	
mov dl,456h	غير مقبولة

ملاحظة :

أدناه قائمة بالحالات المختلفة والمقبولة لإيعاز الـ **mov**:

Mov Reg,Reg
Mov mem,Reg
Mov Reg,Mem
Mov Mem,Imm
Mov Reg,Imm

حيث إن الاختصارات الآتية تعني :

- **Mem**: معامل ذاكرة مثل ... ,x , a .
- **Reg** : مسجل مثل .. ax , ebx .
- **Imm** : قيمة ثابتة مثل ... 5 ,6,10 .

مثال ١:

اكتب برنامج يقوم بنقل الرقم 5 إلى المسجل ax .

The screenshot shows a Visual Studio IDE with two windows. The left window displays assembly code in a file named 'pe'. The code is as follows:

```

1  Include Irvine32.inc
2  .code
3  main proc
4  mov ax,5h
5  call dumpregs
6  exit
7  main endp
8  end main
9

```

The right window is a command prompt titled 'C:\Windows\system32\cmd.exe'. It displays the output of the 'call dumpregs' instruction, showing the current state of the registers and flags:

```

EAX=763D0005 EBX=7FFDF000 ECX=00000000 EDI=00F11005
ESI=00000000 EDI=00000000 EBP=001DF7C4 ESP=001DF7BC
EIP=00F11019 EFL=00000026 CF=0 SF=0 ZF=1 OF=0 AF=0 PF=1
Press any key to continue . . .

```

الإيعاز **call dumpregs**: يستخدم هذا الإيعاز لطباعة قيم المسجلات علماً إن النتيجة تظهر بالنظام السادس عشري وذلك نتيجة تأثير الحرف **h** الذي يتم وضعه بعد كل رقم .

خطوات تنفيذ البرنامج المكتوب بلغة التجميع:

- ١- عملية التجميع (بمعنى **compiler**): عن طريق الضغط على القائمة **Build** نختار **Rebuild solution** حيث إن هذه الخطوة تستخدم لاكتشاف الأخطاء الموجودة بالبرنامج.
- ٢- التنفيذ: من القائمة **Debug** نختار **Start without debugging** .

ثانياً:- الإيعاز **movsx (mov with sign-extends)** : يقوم هذا الإيعاز بنسخ البيانات من معامل مصدر أقل إلى معامل هدف أكبر ويقوم بتوسعة القيمة ويستعمل مع الإعداد ذات الإشارة (الإعداد السالبة) وينفذ بصيغة **fff** وله ثلاث حالات :

1.	Movsx Reg32 , Reg/Mem(8)
2.	Movsx Reg32 , Reg/Mem(16)
3.	Movsx Reg16 , Reg/Mem(8)

ثالثا:- الإيعاز **movzx (mov with zero extends)** : يقوم هذا الإيعاز بنسخ البيانات من معامل مصدر اقل إلى معامل هدف اكبر ويقوم بتوسعة القيمة صفريا (**zero extend**) ويستعمل هذا الإيعاز مع الإعداد الصحيحة التي لاتمتلك إشارة (**unsigned integer**) وله ثلاث حالات :

1.	Movzx Reg32 , Reg/Mem(8)
2.	Movzx Reg32 , Reg/Mem(16)
3.	Movzx Reg16 , Reg/Mem(8)

مثال :

اكتب برنامج يقوم بنقل القيمة **A96BH** إلى المسجل **bx** ومن ثم نقله إلى المسجل **eax**.

```

1 Include Irvine32.inc
2 .code
3 main proc
4 mov bx,0a96bh
5 movzx eax,bx
6 call dumpregs
7 exit
8 main endp
9 end main
10

```

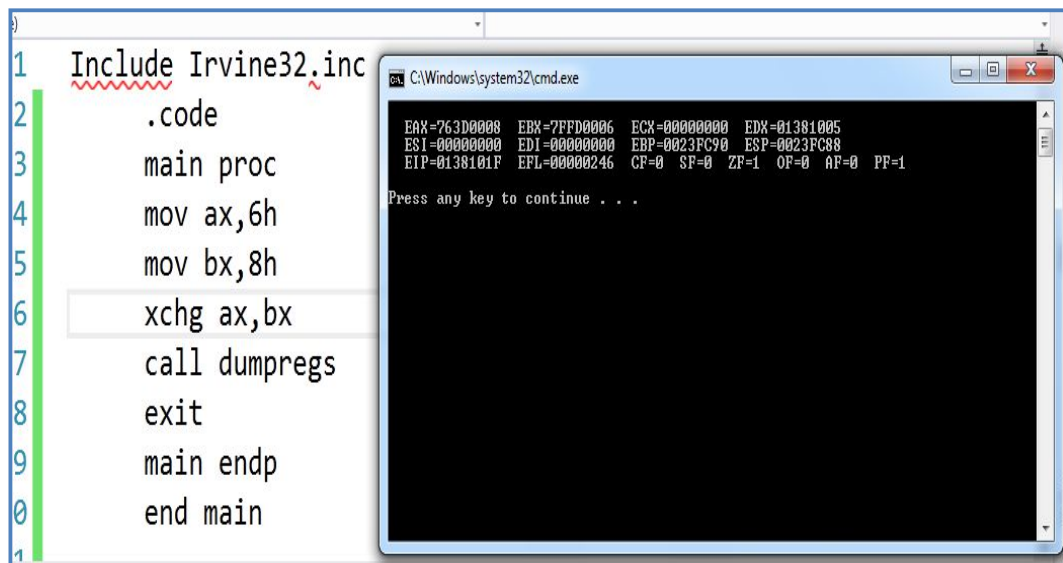
EAX=0000A96B EBX=7FFDA96B ECX=00000000 EDI=01241005
 ESI=00000000 EDI=00000000 EBP=0012FF54 ESP=0012FF4C
 EIP=0124101C EFL=00000024 CF=0 SP=0 ZF=1 OF=0 AF=0 PF=1
 Press any key to continue . . .

رابعا:- الإيعاز **xchg(exchange data)** : يقوم هذا الإيعاز بتبديل محتويات معاملين إي إن عمله مشابه لعملية (**swap**) ولكن بخطوة واحدة وله ثلاث حالات :

1.	Xchg Reg , Reg
2.	Xchg Reg , Mem
3.	Xchg Mem , Reg

مثال :

اكتب برنامج يقوم بتبديل محتوى المسجل **ax** مع محتوى المسجل **bx** .



The image shows a screenshot of an assembly program and its execution. On the left, the assembly code is displayed in a text editor with line numbers 1 through 10. The code includes Irvine32.inc, defines a main procedure, moves 6h to ax and 8h to bx, swaps them using xchg, calls dumpregs, and exits. On the right, a command prompt window shows the output of the program, displaying the state of various registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, EIP, EFL) and flags (CF, SF, ZF, OF, AF, PF) before and after the swap operation. The output shows that the values of ax and bx have been swapped.

```
1 Include Irvine32.inc
2 .code
3 main proc
4 mov ax,6h
5 mov bx,8h
6 xchg ax,bx
7 call dumpregs
8 exit
9 main endp
10 end main
```

Output from command prompt:

```
EAX=763D0008 EBX=7FFD0006 ECX=00000000 EDX=01381005
ESI=00000000 EDI=00000000 EBP=0023FC90 ESP=0023FC88
EIP=0138101F EFL=00000024 CF=0 SF=0 ZF=1 OF=0 AF=0 PF=1
Press any key to continue . . .
```