كلية التكنولوجيا

والعلوم التطبيقية

هيكليـــــة الحاســـــوب

ولغـــــة التجميـــع

ASSEMBLY

رقم المقرر 1381

الدليل العملي

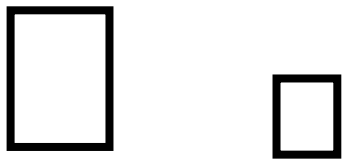
اعداد

د. يوسف ابوزر

2016

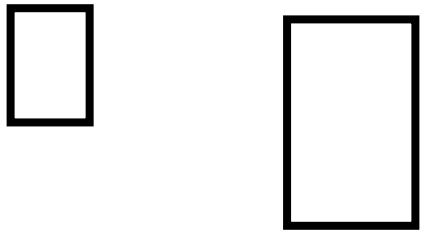
جامعـــة القـــدس المفتوحة

محتويات الدليل



الوحدة الاولى

مقدمة الى برنامج المنقح Debug



1. المقدمة

1.1 تمهيد

عزيزي الطالب، أهلاً بك إلى الوحدة الأولى من التطبيق العملي لمقرر هيكلية الحاسوب ولغة التجميع اسمبلي وهي بعنوان مقدمة الى برنامج المنقح Debug.

تهدف هذه الوحدة إلى إعطائك معلومات أساسية فيما يتعلق بالتعامل مع برنامج المنقح Debug من حيث العريف والتعامل مع الأوامر والتعليمات لمعالجة المسجلات والذاكرة من حيث استعراضها والتخزين بها وكيفية كتابة وتنفيذ برامج لغة التجميع اسمبلي، وتتبع تعليمات برنامج اسمبلر وأخيرا الوصول إلى الذاكرة وعرض محتوياتها والتعديل عليها

وقد تم تزويد هذه الوحدة بالعديد من الأمثلة التوضيحية، والتدريبات، إضافة إلى بعض أسئلة التقويم الذاتي والإشكال التي توضح بعض المفاهيم.

مرة أخرى نرحب بك في هذه الوحدة، آملين أن تستمتع بدراستك للمادة التي نعرضها لك في هذه الوحدة.

2.1 أهداف الوحدة

ينتظر منك، عزيزي الطالب، بعد دراسة هذه الوحدة أن تكون قادراً على:

1. تتعرف على برنامج Debug وكيفية تشغيله والتعامل معه.
2. تتمكن من استعراض قيم المسجلات في معالج 8086.
3. تتمكن من تخزين برنامج اسمبلي في مكان معين في الذاكرة.
4. تتمكن من تنفيذ برنامج اسمبلي.
5. تتمكن من تتبع تعليمات برنامج اسمبلي، وماذا يحدث في المسجلات بعد كل تعليمة.
6. تتمكن من الوصول إلى الذاكرة وعرض محتوياتها والتعديل عليها.
7. تتأكد من صحة وامكانية استخدام التعليمات.

3.1 أقسام الوحدة

تتكون هذه الوحدة من ستة أقسام رئيسة. فالقسم الأول يناقش مقدمة للوحدة، اما القسم الثاني فيناقش برنامج المنقح وطرق تشغيله والتعامل مع المسجلات والذاكرة وكيفية ادخال برنامج لغة التجميع وتنفيذه. القسم الثالث فيوضح كيفية ادخال برنامج لغة التجميع وتنفيذه. في القسم الرابع تم مناقشة تتبع تنفيذ برنامج لغة التجميع اسمبلي. القسم الخامس يناقش أوامر التعامل مع الذاكرة واخيرا القسم السادس يناقش أوامر أخرى لبرنامج المنقح.

4.1 ما تحتاج إليه لدراسة الوحدة

لقد تضمنت هذه الوحدة العديد من المفاهيم الأساسية، وبذلك فان طبيعة المادة المعروضة، عزيزي الطالب، تحتاج إلى جو هادئ خاص ومريح حتى تستطيع التركيز على المفاهيم المعطاة، وتستوعبها بالشكل المناسب. وكل ما تحتاجه بعد ذلك بعض القرطاسية كقلم رصاص وورق أبيض لتقوم بتطبيق بعض المفاهيم وتحل الأسئلة والتدريبات المعطاة في ثنايا الوحدة. ويمكنك عزيزي الطالب استخدام الانترنت للبحث عن بعض المفاهيم والمواضيع المتعلقة برنامج Debug. كما ان هناك حاجة الى:

1. معرفة بأساسيات المعالج 8086 ومسجلاته الرئيسية. (راجع الوحدة الثانية من الكتاب المقرر).
2. معرفة بأساسيات لغة الاسمبلي. (راجع الوحدة الرابعة من الكتاب المقرر).

2. مدخل الى برنامج المنقح Debug

1.2 برنامج المنقح Debug

مصلح debug تشير إلى عملية تتبع الأخطاء في برامج الحاسوب وتصحيحها، وكل لغة برمجة توفر محرر لاكتشاف وتصحيح الأخطاء

ما هو برنامج debugger المنقح؟ هو برنامج يتيح لك عرض المسجلات والمكدس والأوامر بلغة الاسمبلي وأشياء أخرى كثيرة. ويقوم كذلك بعمل تتبع tracing للبرنامج وذلك لاكتشاف الأخطاء ان وجدت.

برنامج Debug هو إحدى الأدوات البرمجية المستخدمة لتنفيذ وتتبع برنامج اسمبلي. وهو مخصص للمعالج Intel-8086.

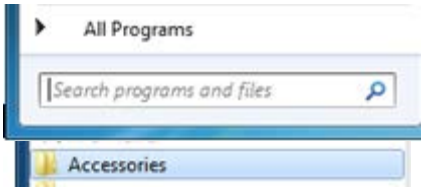
برنامج Debug موجود ضمن غالبية أنظمة تشغيل DOS و WINDOWS. بالإضافة إلى امكانية كتابة وتنفيذ برنامج اسمبلي، يحتوي برنامج Debug على العديد من الاوامر التي يمكن استخدامها للوصول الى الذاكرة وعرض محتواها والقراءة منها او الكتابة عليها، بالإضافة الى امكانية عرض وتعديل مسجلات المعالج.

2.2 تشغيل برنامج Debug

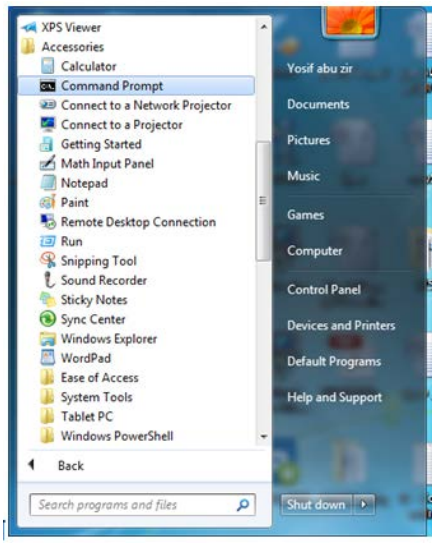
سنبدأ في هذا الشرح عزيزي الطالب بتوضيح كيفية تشغيل برنامج Debug. ونظرا لان لغة الاسمبلي تحتاج إلى معرفة معلومات عن مسجلات المعالج الذي سيتم كتابة البرنامج له، فسيتم استعراض ملخص عن مسجلات المعالج المستخدم هنا وهو معالج 8086. بعد ذلك سيتم استعراض أوامر برنامج Debug والاطلاع على امثلة وتدريبات تبيّن استخدام هذه الاوامر.

يمكنك التعامل مع المنقح debug وذلك بدخولك الى نظام الدوس وذلك باتباع الخطوات التالية

1. النقر على ايقونة أبدأ  ثم الضغط على كافة البرامج All Programs



2. ثم اختيار الإكسسوارات فتظهر لنا القائمة التالية:

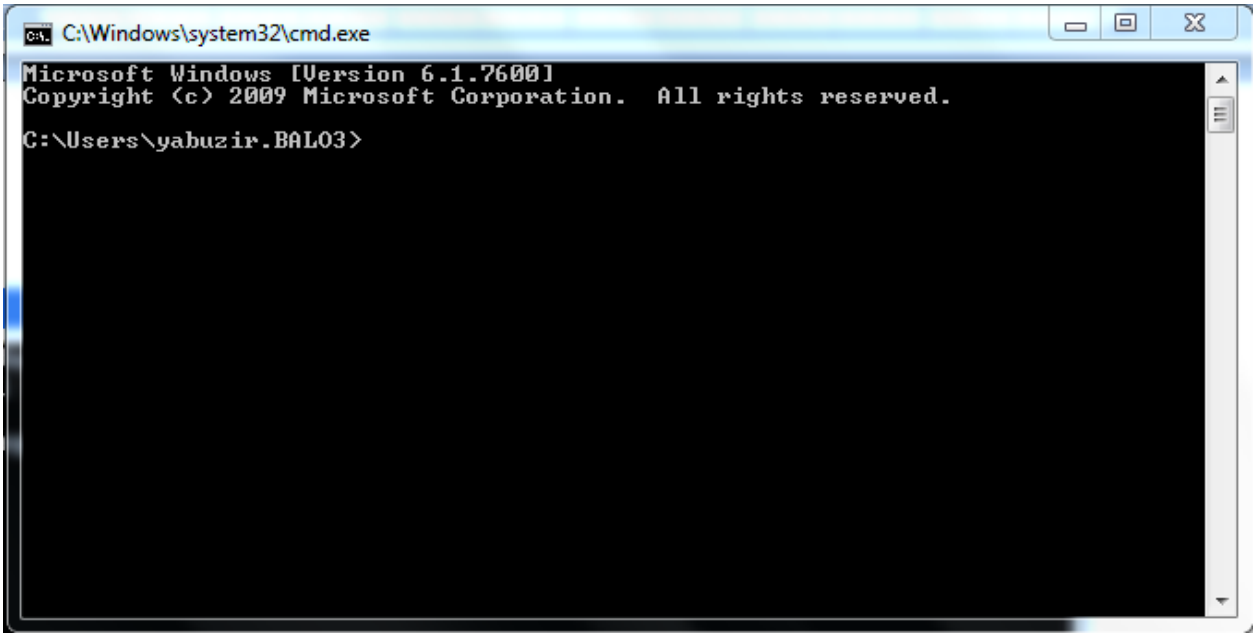


نختار منها محث الدوس Command Prompt

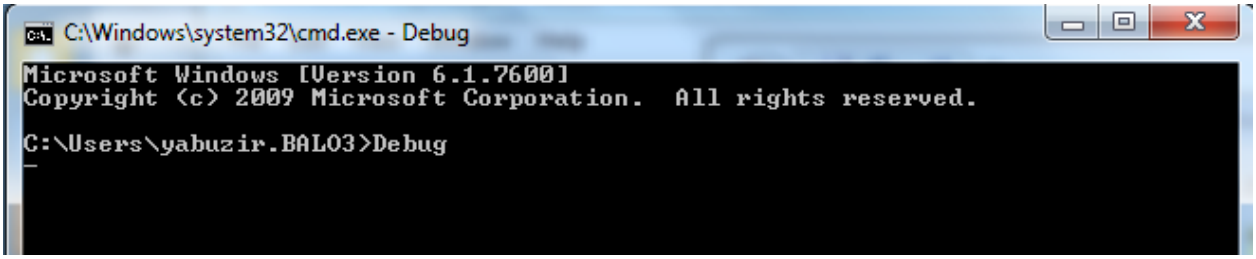
او

1. النقر على ايقونة أبدأ 
2. ندخل cmd ثم Enter

فتظهر لنا نافذة الدوس **DOS** كما في الشكل التالي:



للدخول على برنامج المنقح ندخل الكلمة debug ثم مفتاح الادخال Enter. فتظهر لنا اشارة السالبة للدلالة على اننا في نظام المنقح. حيث يمكننا ان ندخل الاوامر الخاصة بالمنقح



3.2 خطوات تشغيل برنامج ال Debug على أجهزة الحاسوب الحديثة

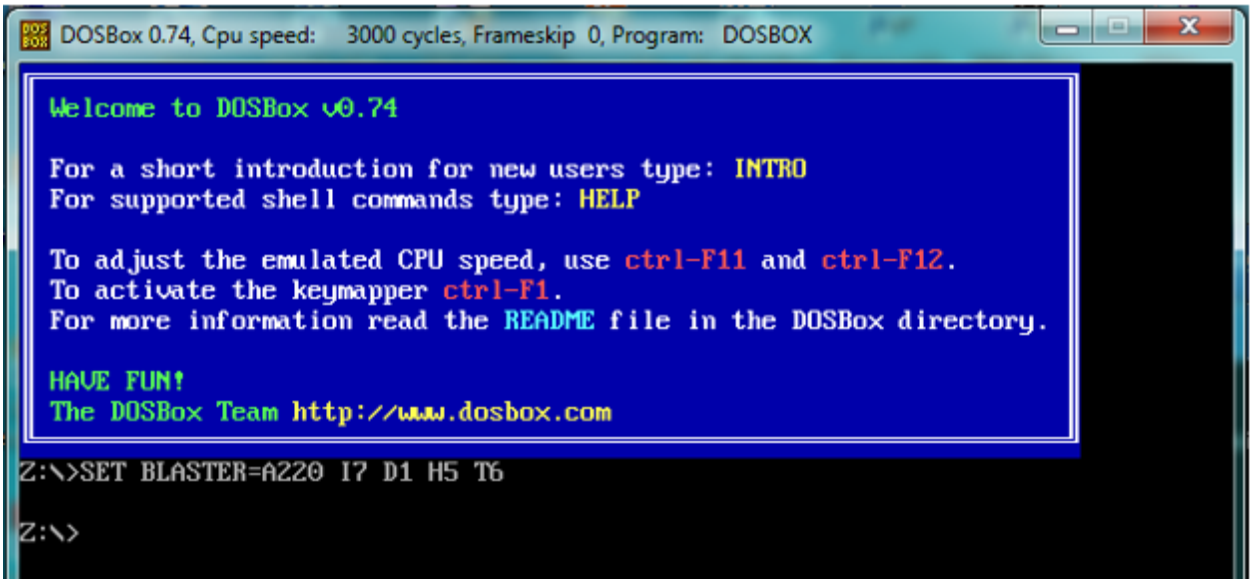
هناك العديد من الأجهزة الحديثة لا يعمل عليها برنامج ال Debug، لتشغيل البرنامج لا بد من استخدام البرامج التالية اتباع الخطوات التالية:

1- تنزيل برنامج dosbox من هذا الرابط : (مساحته 1.38 ميجابايت) او اطلبه من الفني المختبر
http://www.dosbox.com/download.php?main=1

اطلب ملف المنقح Debug من فني المختبر وخزن الملف على القرص الصلب لحاسوبك لنفرض على القرص C

2- مرحلة الاعداد : اضغط على الأيقونة اضغط next اضغط next اضغط install اضغط close ستجد هذه الأيقونة على سطح المكتب

3- انقر فوق الايقونة لتبدئ بالتشغيل تظهر لك الشاشة التلية:



4- لتشغيل برنامج المنقح Debug نكتب الأوامر التالية:

1- نكتب Z:>**mount c c:** ، ثم نضغط على مفتاح الادخال، كما في الشكل التالي:

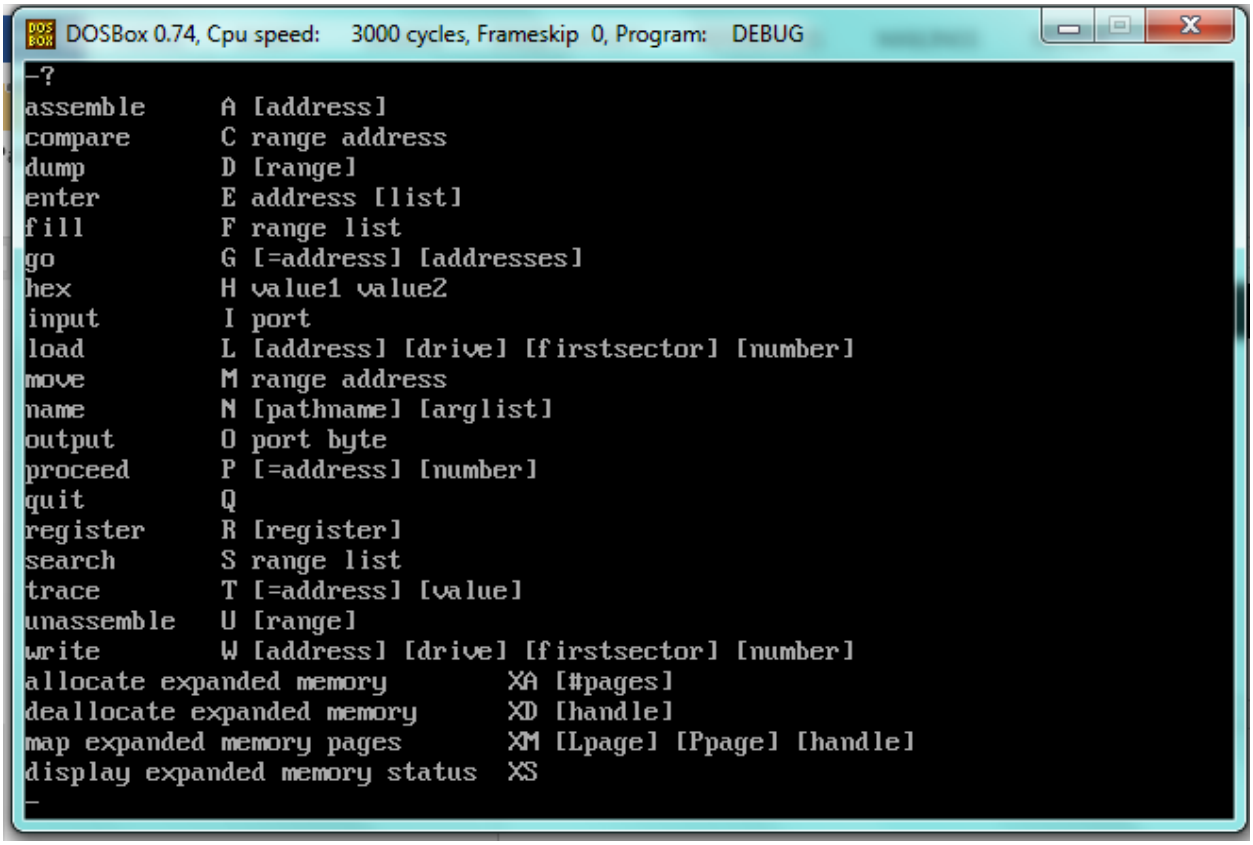
```
Z:\>mount c c:\
Mounting c:\ is NOT recommended. Please mount a (sub)directory next time.
Drive C is mounted as local directory c:\

Z:\>
```

2-ننتقل الى المحرك الذي تم تخزين برنامج Debug علية وفي حالتنا كان القرص الصلب C بإدخال الأمر التالي **C:** ثم نضغط على مفتاح الادخال.

```
Z:\>C:

C:\>
```

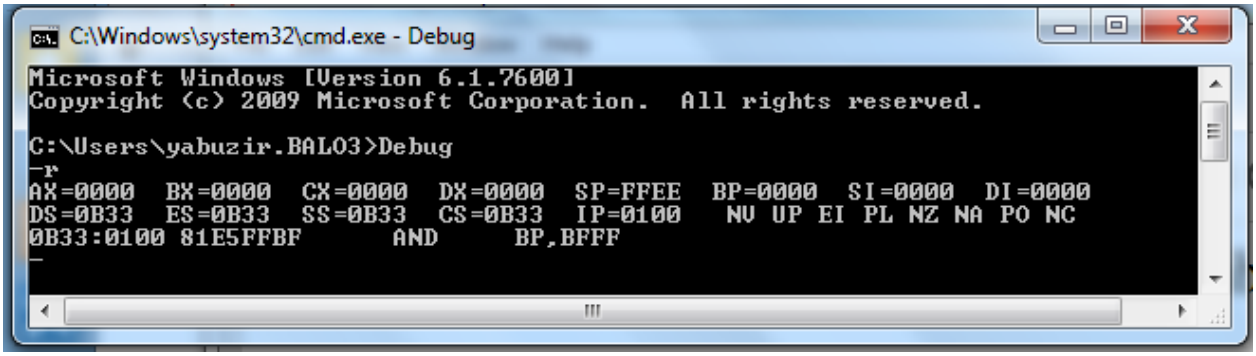
```
Z:\>c:

C:\>_
```

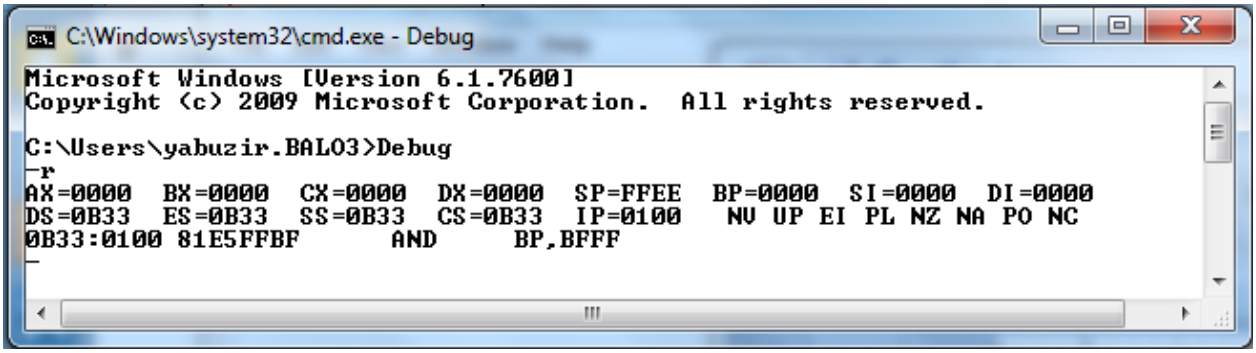
3- نكتب كلمة Debug `C:\>debug` ثم نضغط على مفتاح الادخال ، فتظهر لنا شاشة برنامج Debug بوجود إشارة السالب، حيث يمكننا ان نبدئ بإدخال أوامر Debug، فمثلا لعرض أوامر Debug نطبع الرمز إشارة الاستفهام ؟ ونضغط على مفتاح الادخال، فتظهر لنا شاشة برنامج Debug التالية:



4.2 عرض المسجلات

للتعرف على المسجلات ندخل الحرف R ثم نضغط على مفتاح الادخال، فتظهر لنا قائمة بأسماء المسجلات والقيم المخزنة فيها.





نشاط (1)

1. ما أسماء المسجلات التي تظهر على الشاشة؟

.....

.....

2. ما هي القيم في كل مسجل؟

.....

.....

صنف المسجلات حسب أنواعها

.....

.....

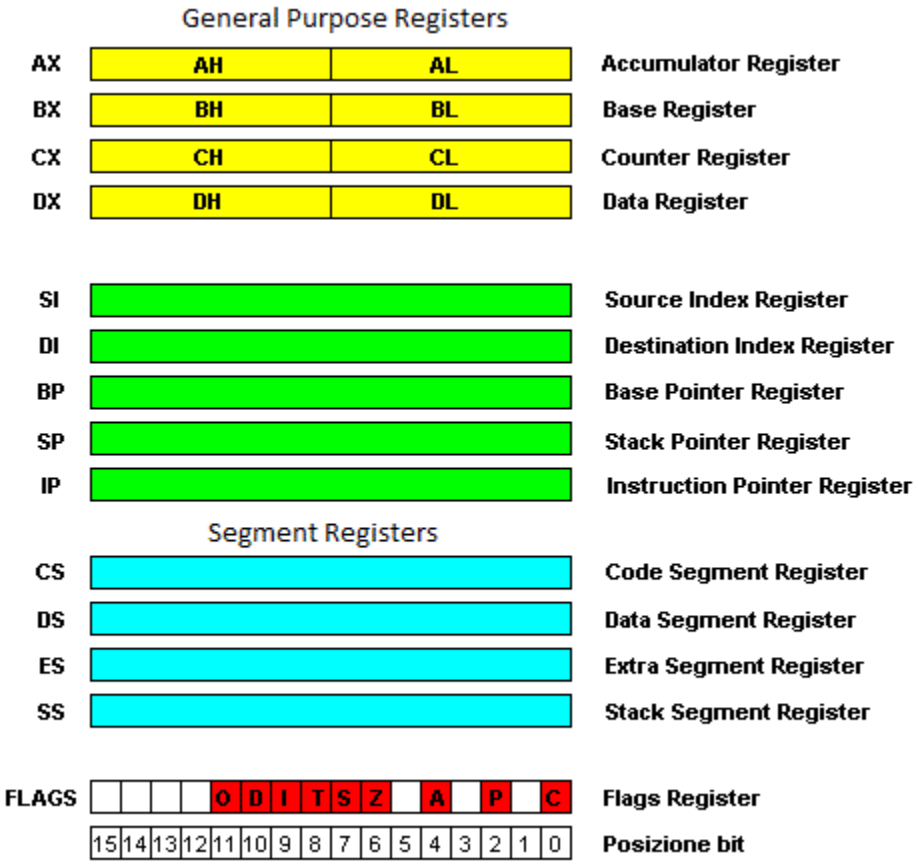
.....

.....

.....

قبل الخوض بمزيد من الأوامر، لنتعرف بشكل سريع على مسجلات المعالج 8086، والتي يعمل معها برنامج Debug، ولا يمكن أن نتعامل مع هذا البرنامج بدون الاطلاع على هذه المسجلات ومعرفة وظيفتها.

يحتوي معالج 8086 على 14 مسجل يمكن للمستخدم أن يتعامل معها. ويمكن تصنيف هذه المسجلات كما يلي شكل (1):



شكل (1) أنواع المسجلات

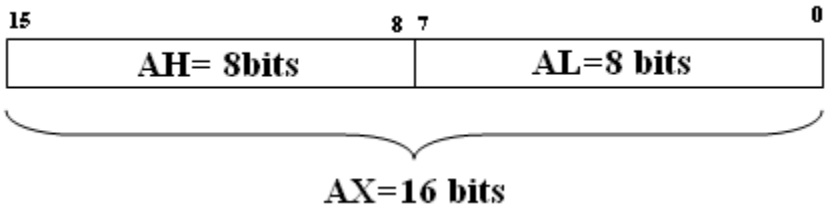
1. مسجلات المعطيات (عامة الاستخدام):

و عددها أربعة وهي: **AX, BX, CX, DX**.

حجم كل منها 16 بت. وسميت عامة الاستخدام لان المستخدم يمكنه التعامل معها كيف ما شاء؛ يمكنه تحميلها بأي قيمة، نقل محتوياتها إلى بعض المسجلات الأخرى أو إلى بعضها البعض أو إلى الذاكرة.

هذه المسجلات بها خاصية إمكانية تقسيمها إلى جزأين؛ كل جزء حجمه 8 بت. مثلا مسجل AX يمكن تقسيمه إلى جزأين؛ الجزء الأعلى و حجمه 8 بت يسمى AH (حرف H من كلمة High)، و الجزء الأسفل و حجمه ايضا 8 بت و يسمى AL (حرف L من كلمة Low).

أي أن المسجل الكامل والذي حجمه 16 بت يضاف إلى اسمه X(مثلا AX)، والجزء الأعلى منه يضاف إلى اسمه H (مثلا AH). وهذا ينطبق على المسجلات AX، BX، CX، و DX. الشكل(2) يبين هذا التقسيم بالنسبة للمسجل AX.



شكل(2): مسجل AX

كما هو مبين في لشكل(2) أعلاه، البتات الثمانية على اليمين هم الجزء الأسفل (AL)، و البتات الثمانية على اليسار هم الجزء الأعلى (AH).

2. مسجلات القطاعات:

وعددها أربعة وهي:

- مسجل قطاع البيانات DS: يخزن عنوان بداية قطاع البيانات. وهو القطاع الذي يخزن به جميع المتغيرات والثوابت و DS يستخدم لتخزين بداية عنوان هذا القطاع.
- مسجل قطاع التعليمات CS: يخزن عنوان بداية قطاع التعليمات. وهو القطاع الذي يخزن به تعليمات برنامج الاسمبلي. و CS به بداية عنوان هذا القطاع.
- مسجل قطاع المكدس SS : يخزن عنوان بداية المكدس.
- مسجل قطاع البيانات الإضافي ES

3. مسجلات الفهرسة و التأشير:

وعددها خمسة وحجم كل منها 16 بت وهي:

- مسجل مؤشر التعليمات :IP ويخزن عنوان التعليمة القادمة التي سيتم تنفيذها.
- مسجل مؤشر المكدس SP: ويخزن عنوان العنصر الموجود في أعلى المكدس.
- مسجل BP: ويمكن أن يؤشر إلى أي مكان في المكدس ايضا.
- مسجل SI: يستخدم مع سلسلة الرموز
- مسجل DI : يستخدم مع سلسلة الرموز ايضا

4. مسجل الرايات (حالة البرنامج):

يحتوي على 16 بت، 9 منها هي بتات للرايات والسبعة المتبقية غير مستخدمة. هذه الرايات تتبدل قيمتها بشكل مستمر بعد تنفيذ كل تعليمة. ومن الأمثلة على هذه الرايات:

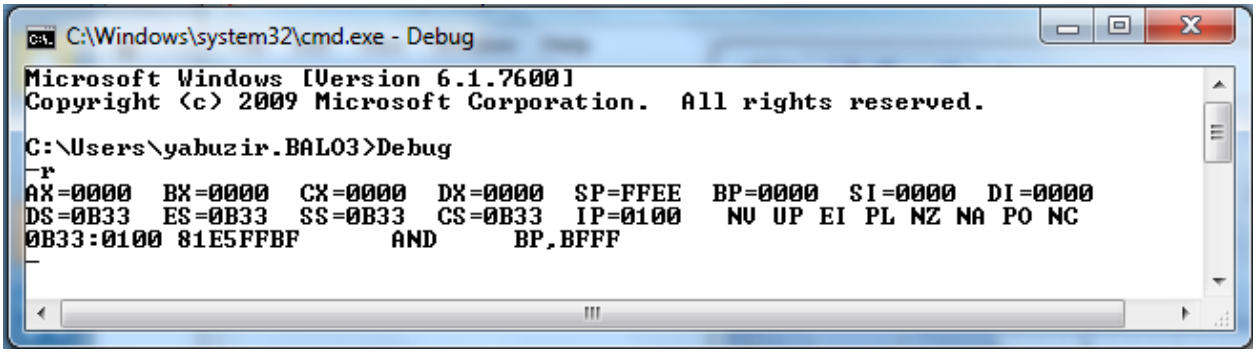
- بت راية الصفر:ZF يصبح قيمة هذا البت 1 إذا كانت نتيجة العملية تساوي صفرا.
- بت راية الإشارة SF : يصبح قيمة هذا البت 1 إذا كانت نتيجة العملية سالبة.

والجدول التالي (1) يلخص هذه الرايات

جدول (1) رايات مسجل الحال

Flags	Description
ov | no	Overflow | No overflow
dn | up	Decrement | Increment
ei | di	Input enable | Input disable
ng | pl	Sign: Negative | Positive
zr | nz	Zero | Not zero
ac | na	Auxiliary carry | No auxiliary carry
pe | po	Parity even | Parity odd
cy | nc	Carry | No carry

4.2 معالجة قيم المسجلات

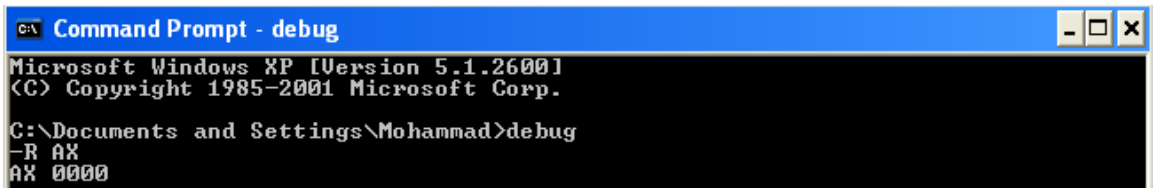


شكل (3) محتويات المسجلات

بالرجوع الى الشكل (3) السابق نرى ان جميع القيّم التي تظهر هي بالنظام السادس عشري Hexadecimal . فمثلا القيمة المخزنة في مسجل AX هي 0000 – أربعة أرقام بالنظام السادس عشري –أي 16 بت بالنظام الثنائي وهو حجم جميع مسجلات المعالج 8086.

كما اشارنا سابقا يمكننا استخدام الأمر **R** لإظهار محتويات المسجلات، لإظهار محتويات أحد المسجلات ندخل **R** ثم اسم المسجل، فمثلا لإظهار محتويات المسجل AX فقط نطبع R بعدها اسم المسجل AX الذي تريد معرفة محتوياته كما يلي:

اكتب **R AX ثم اضغط مفتاح الادخال** ليظهر لك محتويات AX فقط كما هو مبين في الشكل (4) التالي.



شكل (4) محتويات المسجلات

الأمر R: سبق أن تحدثنا عن هذا الأمر وأنه يعرض قيمة المسجلات كافة أو قيمة مسجل معين.

بالإضافة الى ما تم ذكره سابقا هناك إمكانية تعديل قيمة المسجل الى قيمة جديدة بعد عرض قيمته الحالية باستخدام الامر R ، و ذلك بكتابة القيمة الجديدة بعد النقطتين كما يلي:

R CX

CX 0000

: 500

في المثال اعلاه، تم عرض القيمة الحالية للمسجل CX وهي 0000، ثم قمنا بكتابة القيمة الجديدة وهي 500. وإذا كنا لا نريد تعديل القيمة يجب الضغط على مفتاح Enter لإنهاء خيار تعديل القيمة.

مثال1:

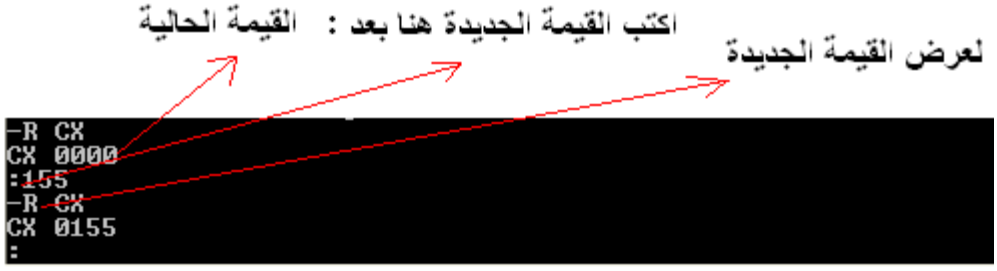
1-اكتب الامر الذي يعرض محتوى المسجل CX ثم قم بتغيير قيمته إلى 155، ثم تأكد من القيمة الجديدة.

2- تغيير محتوى المسجل IP الى القيم 100، حيث ان هذه العملية ضرورية قبل تنفيذ برامج لغة التجميع باستخدام برنامج debug لتأكد من ان مؤشر البرنامج IP يشير الى اول موقع في البرنامج.

3- تغيير محتوى المسجل CS الى القيم 2000، حيث ان هذه العملية ضرورية قبل ادخال برامج لغة التجميع باستخدام برنامج debug .

الحل:

1-



2-

لمعرفة محتوى مسجل التأشير IP قبل تشغيل البرنامج، اكتب: R IP. بعد ذلك، سوف نرى العنوان الذي يشير اليه المسجل IP.

ندخل الرقم 2000 بعد الرمز الخاص : ثم نضغط مفتاح الادخال.

R IP

:100

3-

لمعرفة محتوى مسجل التعليمة CS قبل تشغيل البرنامج، اكتب: R CS. بعد ذلك، سوف نرى العنوان الذي يشير اليه المسجل CS .

ندخل الرقم 2000 بعد الرمز الخاص : ثم نضغط مفتاح الادخال.

R CS

:*2000*

أسئلة التقويم الذاتي (1)

1–ما هو برنامج Debug ؟

2–كيف يتم تشغيل برنامج Debug ؟

3–اذكر أسماء المسجلات الأربعة للمعطيات Data Registers، وبيّن لماذا سميت بمسجلات عامة الاستخدام؟

4–ما هي وظيفة المسجل CS ؟

5–ما هي وظيفة المسجل IP ؟

6–ما هي وظيفة مسجل الرايات (مسل حالة البرنامج)؟

7–بين عمل راية الصفر في مسجل الرايات؟

8– لماذا نغير قيم المسجلات CS و IP ؟

9– ما ناتج تنفيذ الأوامر والتعليمات التلية:

```
-r cs
CS 0B11
:2000
-a cs:0
2000:0000 mov ax,2
2000:0003 add ax,3
2000:0006 hlt
2000:0007
```

3 ادخال برنامج لغة التجميع اسمبلي

الأمر A يسمح بإدخال برنامج اسمبلي إلى الذاكرة. ويمكن كتابة رقم بعد حرف A يحدد العنوان الذي سيتم كتابة التعليمات به. فمثلا لإدخال برنامج يبدأ من العنوان 100 في قطاع التعليمات، ما عليك عزيزي الطالب إلا أن تكتب:

- A 100

ولنأخذ المثال التالي والذي يبين خطوات إدخال برنامج في قطاع التعليمات ابتداء من العنوان 100 (تذكر عزيزي الطالب، ان جميع الأرقام المستخدمة هنا هي بالنظام السادس عشري). هذا البرنامج يضع قيمة 5 في المسجل AX، وقيمة 3 في المسجل BX ثم يجمع قيّم المسجلين AX و BX ويخزن التعليمة في:AX

مثال2:

```
-A 100
0B58:0100 MOV AX,5
0B58:0103 MOV BX,3
0B58:0106 ADD AX,BX
0B58:0108
-
```

ملاحظة: للخروج من كتابة التعليمات، فقط اضغط Enter مرتين بعد آخر تعليمة تكتبها.

يمكن ملاحظة بعض النقاط إذا ما نظرنا إلى المثال أعلاه.

أولا، بعد كتابة **A 100** والضغط **Enter** يظهر السطر التالي:

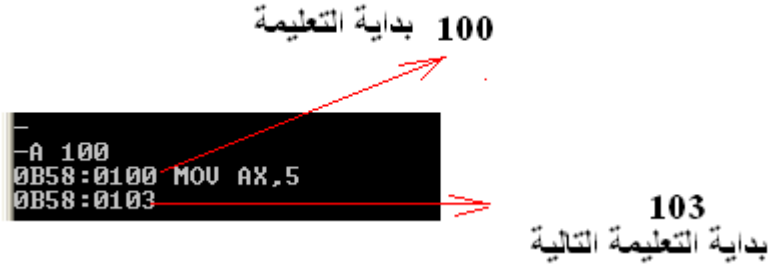
0B58: 0100

هذا السطر يبين قيمة مسجل قطاع التعليمات CS الحالية وهي **0B58** ، بالإضافة إلى الإزاحة التي سيتم إدخال التعليمات بها وهي **100** والتي حددناها عندما كتبنا **A 100** (إي ادخل تعليمة ابتداء من الإزاحة **100**).

بعد كتابة أول تعليمة والتي هدفها تحميل AX بقيمة 5، والضغط على **Enter**، يظهر السطر الثاني وهو:

0B58: 0103

نستنتج من هذا السطر أن طول التعليمة الأولى وهي MOV AX,100 هو 3 بايت.



كما هو موضح في الشكل أعلاه فان أول التعليمة تبدأ في الموقع 100، والتعليمة التالية ستبدأ في الموقع 103، أي أن التعليمة الأولى احتلت 3 بايت.

4 تتبع برنامج لغة التجميع اسمبلي الامر T

هذا الامر مفيد جدا وكثير الاستخدام في برنامج Debug. ويستخدم لتتبع البرنامج تعليمة بعد تعليمة، ويبيّن ما الذي يحدث في المسجلات والرايات بعد كل تعليمة. ونستخدم الحرف T هنا من كلمة "Trace" أي تتبع.

المثال التالي يوضح طريقة عمل الامر T.

مثال3:

باستخدام Debug اكتب برنامج بلغة الاسمبلي يعمل كالتالي:

- يحمل في AX قيمة 5
- يحمل في BX قيمة 3
- يضيف BX إلى AX
- ثم يصفر BX

الحل:

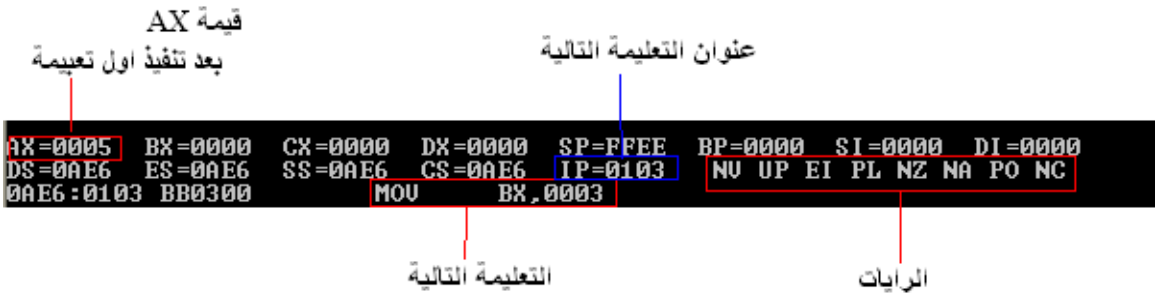
نكتب البرنامج التالي من الإزاحة 100 (نستخدم A100)، ثم ندخل التعليمات التالية:

```
MOV AX,5
MOV BX,3
ADD AX,BX
XOR BX,BX
```

بعدها نكتب T=100 أي ابدأ التتبع من الإزاحة 100.

```
-A100
0AE6:0100 MOV AX,5
0AE6:0103 MOV BX,3
0AE6:0106 ADD AX,BX
0AE6:0108 XOR BX,BX
0AE6:010A
-T=100
```

بعد كتابة T=100 والضغط Enter يظهر التالي



في الشكل السابق أعلاه، لاحظ عزيزي الطالب أن المعلومات التي ظهرت هنا تبين قيم المسجلات ووضعية الرايات بعد تنفيذ التعليمة MOV AX,5. من هذه المعلومات يظهر لنا ما يلي:

- أن قيمة AX أصبحت 5،
- وأن قيمة المسجل IP أي عنوان التعليمة التالية هو 103،
- وأن التعليمة التالية هي MOV BX,3، وهي التعليمة التالية التي قمنا نحن بإدخالها.
- بالنسبة للرايات: هناك 8 ثمان رايات ممكن أن تتغير بعد كل تعليمة. وعلى سبيل المثال، NZ تعني أن نتيجة آخر تعليمة لا تساوي صفراً، أي أن راية الصفر تساوي صفرا. وايضا PL تعني أن نتيجة آخر تعليمة هي موجبة، أي أن راية الإشارة تساوي صفرا. وجدول 2 يبين بعض الرايات ومعناها في برنامج Debug.

الراية	إذا كانت قيمتها تساوي صفر	إذا كانت قيمتها تساوي واحد
راية الصفر	NZ	ZR
راية الإشارة	PL	NG
راية الحِمل	NC	CY

جدول 2 : بعض الرايات و معناها في برنامج Debug

بالعودة إلى المثال السابق والذي أدخلنا فيه برنامج ثم بدأنا بالتتبع من موقع 100 باستخدام الامر T=100. لنرى الآن التعليمة التالية وما التغييرات التي احدثتها. نكتب هنا T فقط، ولا يلزم تحديد العنوان لهذه التعليمة. فقط يلزم تحديد عنوان أول تعليمة، بعدها نكتب T فقط.

```
-T

AX=0005  BX=0003  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=0106   NV UP EI PL NZ NA PO NC
0AE6:0106 01D8          ADD     AX,BX
```

لاحظ أن قيمة BX تغيرت لأننا نفذنا التعليمةMOV BX,3 . لاحظ ايضا عنوان التعليمة التالية (IP)، و نص التعليمة (ADD AX,BX).

التعليمة التالية (استمر بكتابة T):

```
-T

AX=0008  BX=0003  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=0108   NV UP EI PL NZ NA PO NC
0AE6:0108 31DB          XOR     BX,BX
```

لاحظ أن قيمة AX أصبحت 8 وذلك لأننا اضفنا BX (3) الى AX (5) باستخدام ADD AX,BX، فأصبحت النتيجة 8.

التعليمة التالية (استمر بكتابة T):

```
-T

AX=0008  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=010A   NV UP EI PL ZR NA PE NC
0AE6:010A D8BB0000            FDIVR   DWORD PTR [BP+DI+0000]      SS:0000=CD
```

لاحظ اننا صفرنا قيمة BX باستخدام XOR BX,BX، و لاحظ ايضا أن قيمة راية الصفر اصبحت ZR. أي أن قيمة راية الصفر أصبحت 1.

تدريب 1

قم بإجراء العمليات التالية باستخدام Debug:

1. اعرض المحتويات الحالية لجميع المسجلات.

تتم هذه الخطوة باستخدام الامر R

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=0100   NV UP EI PL NZ NA PO NC
0AE6:0100 31C0          XOR     AX,AX
```

2. اعرض الوضعية الحالية للرايات

هنا نستخدم R F (F تعني Flags أي الرايات)

```
-R F
NV UP EI PL NZ NA PO NC  -
-
```

3. ادخل البرنامج التالي في الإزاحة 255،

MOV CX,5

MOV AX,0

LABEL: ADD AX,CX

LOOP LABEL

(ملاحظة: كلمةLABEL لا تكتب في برنامج DEBUG لان هذا البرنامج يتعامل مع تعليمات و لا يتعامل مع أسماء)

```
-A155
0AE6:0155 MOV CX,5
0AE6:0158 MOV AX,0
0AE6:015B ADD AX,CX
0AE6:015D LOOP 015B
0AE6:015F
```

لاحظ تعليمة LOOP أتى بعدها العنوان الذي تعود إلية وهو 015B وهي الإزاحة المخزن بها التعليمة

ADD AX,CX

4. تتبع ماذا يحدث بالمسجلات بعد تنفيذ كل تعليمة:

تعليمة MOV CX,5

```
-T=155

AX=0000  BX=0000  CX=0005  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=0158   NV UP EI PL NZ NA PO NC
0AE6:0158 B80000        MOV     AX,0000
```

تعليمة MOV AX,0

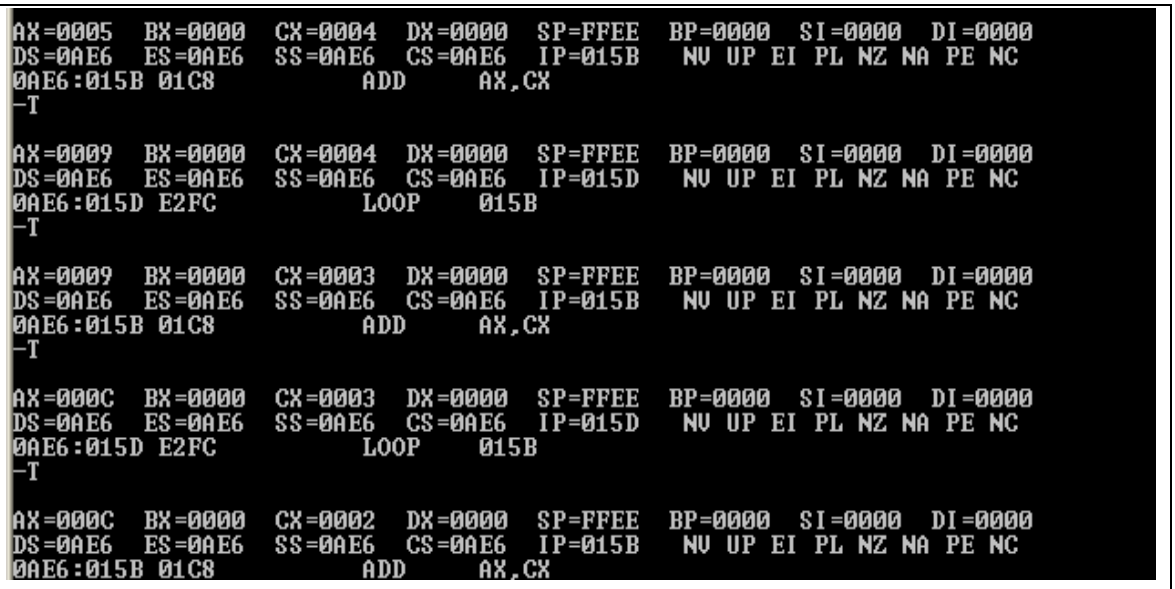
```
-T

AX=0000  BX=0000  CX=0005  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=015B   NV UP EI PL NZ NA PO NC
0AE6:015B 01C8          ADD     AX,CX
```

تعليمة ADD AX,CX

```
-T

AX=0005  BX=0000  CX=0005  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0AE6  ES=0AE6  SS=0AE6  CS=0AE6  IP=015D   NV UP EI PL NZ NA PE NC
0AE6:015D E2FC          LOOP    015B
```

تعليمة LOOP

لاحظ أن هذه التعليمة ستنفذ خمس مرات ، لان قيمة CX=5 ، والقيمة المخزنة في CX هي التي تحدد عمليات التكرار. و في كل مرة تعود إلى التعليمة ADD AX,CX و تنقص من CX واحد إلى أن يصبح CX يساوي صفرا.



تم تنفيذ LOOP مرتين فقط في الشكل أعلاه. يمكنك عزيزي الطالب أن تكمل تنفيذ هذا البرنامج. لاحظ محتوى CX قد بدأ من 5، ثم أخذ يتناقص مع تنفيذ LOOP كل مره.

كما يمكنك تنفيذ عدد معين من الجمل باستخدام الامر T، و ذلك بكتابة عدد الجمل بعد الامر T ، مثلا لتنفيذ ثلاث جمل نكتب الامر التالي:

T3

الامر G

يعني GO ويستخدم هذا الامر لتنفيذ برنامج أو مجموعة من التعليمات. ويجب الإزاحة لأول تعليمة ولآخر تعليمة عند كتابة هذا الامر.

مثال 4:

ادخل ونفذ البرنامج التالي باستخدام الامر G.

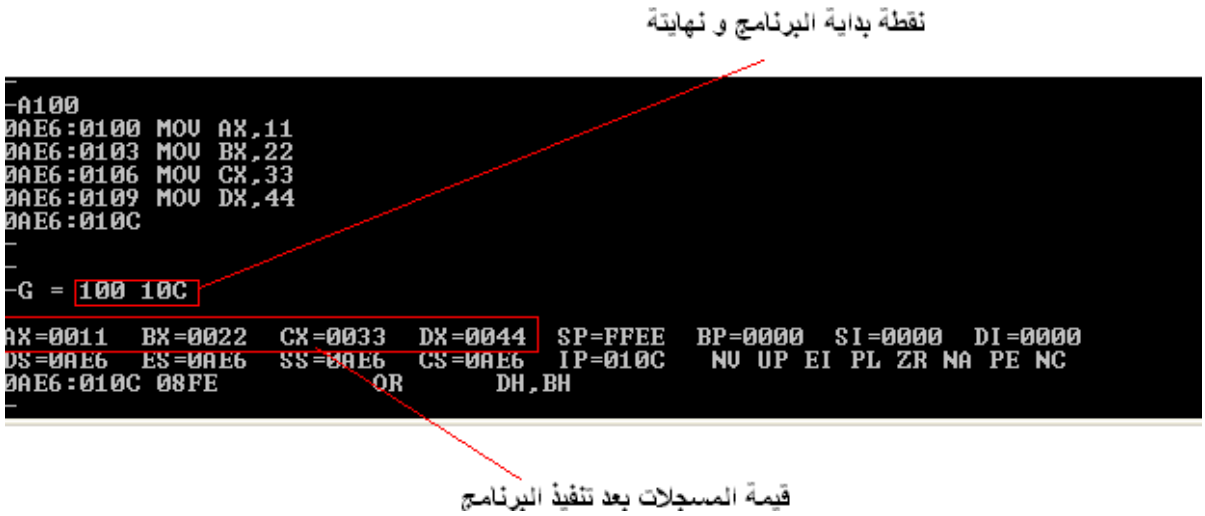
MOV AX,11

MOV BX,22

MOV CX, 33

MOV DX,44

الحل: الشكل. التالي يوضح الحل



تم إدخال البرنامج ابتداء من الإزاحة 0100 و انتهى بالإزاحة 010C. لذلك استخدمنا:

G=100 10C

لتنفيذ الأوامر من الإزاحة 0100 إلى 010C. لاحظ ايضا قيم المسجلات AX، BX، CX، و DX و التي تغيرت إلى القيم الموجودة في البرنامج.

5. التعامل مع الذاكرة

1.5 الامر D

يستخدم هذا الامر لعرض محتويات الذاكرة. يجب تحديد المواقع التي نريد عرضها. على سبيل المثال الامر:

D DS: 100 105

هذا الامر يعرض محتويات المواقع في قطاع البيانات من الموقع 100 إلى الموقع 105.

لاحظ أن محتويات الموقع 100 هي B8

محتويات الموقع 102 هي 11

محتويات الموقع 103 هي 00

محتويات الموقع 104 هي 22

محتويات الموقع 105 هي 00

مع العلم عزيزي الطالب، ان الامر D يعرض محتويات قطاع البيانات إذا لم يتم تحديد القطاع، فمثلا:

D	يعرض الأمر أول 128 بايت من قطاع البيانات
D 100	يعرض الأمر أول 128 بايت من العنوان 100
D ES:0000	يعرض الأمر أول 128 من القطاع الإضافي

2.5 الامر E

يستخدم هذا الامر للكتابة على موقع أو مواقع معينة في الذاكرة.

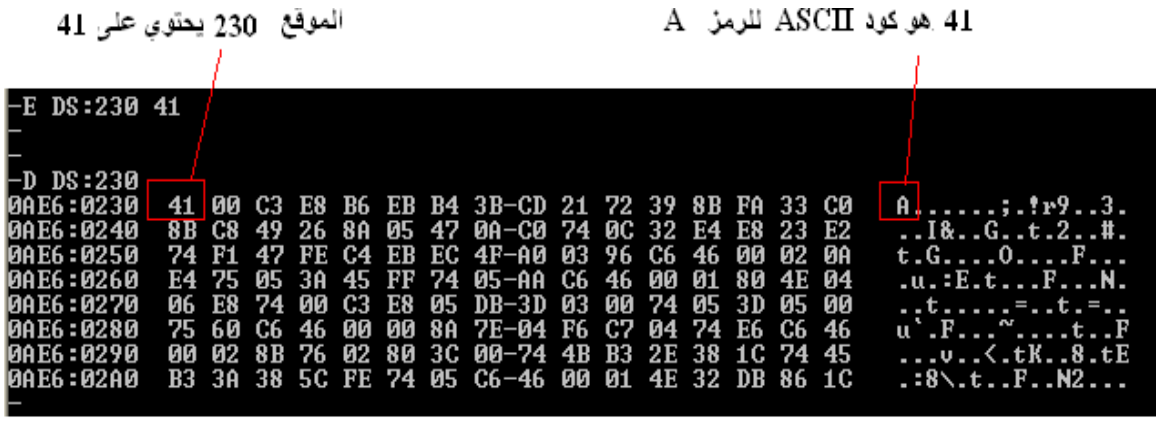
مثلا لكتابة القيمة 41 على الإزاحة 230 من قطاع البيانات:

E DS: 230 55

يمكن لك أن تتأكد من قيمة هذا الموقع باستخدام الامر:

D DS:230

انظر الشكل التالي، ولنلاحظ بعض الأمور.



لاحظ في الشكل السابق انه عند كتابة الامر D DS:230، ظهرت مجموعة من المواقع التخزينية من ضمنها الموقع 230 والذي يحتوي على القيمة 41. نلاحظ ايضا على الجهة اليمنى من الشكل مجموعة من

الرموز تمثل الرموز المقابلة لكود ASCII في كل موقع. مثلا الحرف A كود ASCII له هو 41 (بالنظام السادس عشري).

عزيزي الطالب، لنفترض انك تريد إدخال قيم في مواقع متتالية، فان الامر E يمكنك من فعل ذلك. أي انه لا يقتصر فقط على الكتابة في موقع واحد.

مثال 5:

اكتب على المواقع: 504,503,502,501,500 بالتتابع القيم 44,33,22,11,00

الحل:

نبدأ بالإدخال إلى الموقع الأول كالتالي

E DS:500 ثم مفتاح الادخال

لاحظ أننا لم نكتب القيمة. بعد الضغط Enter ، تظهر القيمة القديمة لهذا الموقع، يمكننا كتابة القيمة الجديدة مباشرة بجانبها، ثم نضغط على المسطرةاو العارضة الطويلة (space bar) من لوحة المفاتيح، ليعطينا إمكانية تعديل الموقع الذي يليه و هكذا حتى ننتهي نضغط Enter.

تدريب 2

اكتب الجملة

I'M STUDYING IN QOU

في المواقع التخزينية ابتداء من الموقع 1000 في قطاع البيانات. ثم تأكد من وجودها في الذاكرة

الحل:

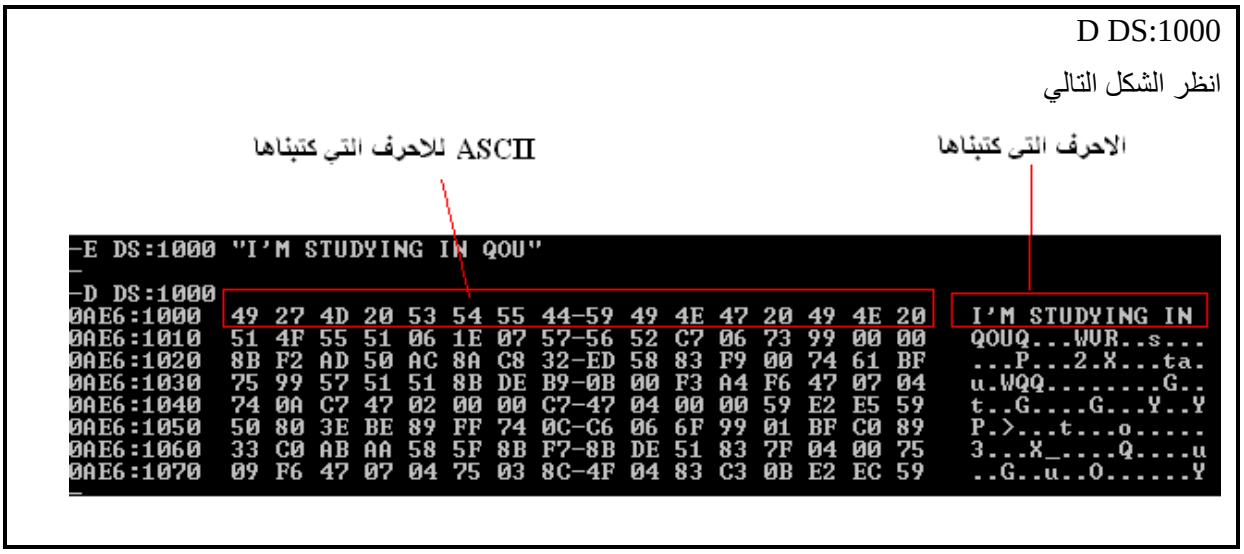
E DS:1000 " I'M STUDYING IN QOU"

الموقع 1000 كتبنا فيه I، الموقع 1001 كتبنا فيه '، الموقع 1003 كتبنا فيه M، و هكذا...

للتأكد، اكتب

D DS:1000

انظر الشكل التالي



3.5 الامر M

يستخدم هذا الامر لنسخ محتوى موقع أو مواقع في الذاكرة الى مكان معين في الذاكرة. مثلا الامر التالي:

M CS:100 110 CS:500

هذا الامر ينسخ محتوى المواقع من 100 الى 110 في قطاع التعليمات CS و يضعها ابتداءا من الموقع 500 في نفس القطاع في الذاكرة، أي انه ينسخ محتوى الموقع 100 الى 500، و محتوى 101 الى 501، و هكذا...

4.5 الامر F

F من Fill أي املأ. يستخدم هذا الامر لمليء سلسلة من المواقع التخزينية بقيمة واحدة. مثلا الامر التالي:

F 100 150 33

يقوم هذا الامر بتعبئة المواقع من 100 إلى 150 بالقيمة 33. عادة ما يستخدم هذا الامر لتصفير سلسلة من المواقع.

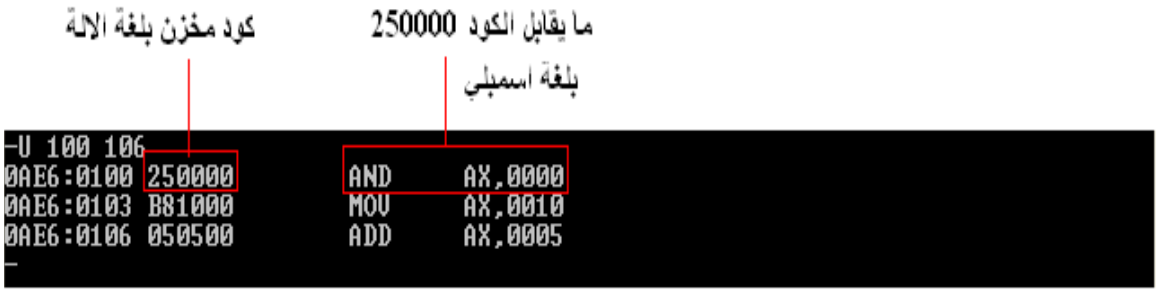
6. أوامر أخرى لبرنامج المنقح Debug

1.6 الامر U

U من UnAssemble. هذا الامر أوامر مكتوبة بلغة الآلة إلى شكلها بلغة الاسمبلي. مثلا الامر:

U 100 106

يعرض الأرقام المخزنة في المواقع من 100 إلى 106، وما يقابلها من تعليمات اسمبلي. الشكل التالي يبين عمل هذا الامر.



لاحظ في الشكل السابق أن الموقع 100 يحتوي على القيمة 250000 وهي كود الة يقابلها بلغة الاسمبلي التعليمة AND AX,0000 .

2.6 الامر Q

هذا الامر يستخدم للخروج من البرنامج. فمثلا بعد إدخال:

- Q

يخرج البرنامج و يعود إلى صفحة أوامر DOS

إلى هنا عزيزي الطالب تم استعراض جميع الأوامر المهمة في برنامج Debug. مع العلم بوجود بعض الاوامر التي لم يتم الحديث عنها

نشاط:

تعرف على كيفية استخدام الاوامر التالية، انظر الملحق في الكتاب المقرر:

- الامر C : (Compare) يستخدم للمقارنة
- الامر S: (Search) يستخدم للبحث
- الامر W: (Write)
- الامر L : (Load)
- الامر N : (Name)

أسئلة التقويم الذاتي (2)

1) ما هو الامر اللازم في برنامج Debug لعرض الوضعية الحالية للرايات؟
2) ما هو استخدام الامر T؟
3) كيف يتم عرض محتويات الموقع 1FA2 في الذاكرة؟ اكتب الامر اللازم.
4) كيف يتم كتابة القيمة 23 في الموقع 234 باستخدام أوامر Debug؟
5) استخدم الامر U لتحويل كود الآلة المخزن في المواقع 120 إلى 133 إلى ما يقابله بلغة الاسمبلي؟

7. نشاط تدريبي

هذا النشاط خاص ببرنامج اكتشاف الأخطاء المنقح DEBUG, إذا احتجت إلى أي استفسار عن هذا التطبيق يمكنك الرجوع إلى المادة السابقة للتعرف على أوامر أل DEBUG (ويفضل أن يتم تنفيذه أمام عضو هيئة التدريس في مختبر الحاسوب)

الفرع الأول

انتقل إلى نظام تشغيل أل DOS وتأكد من وجود البرنامج DEBUG . لتشغيل البرنامج ادخل DEBUG أمام إشارة الحث في نظام الدوس ثم اضغط ENTER
تظهر بعد ذلك إشارة سالب – للدلالة على أن البرنامج مستعد لاستقبال الأوامر

أطبع ما يلي أمام إشارة السالب - R AX ثم ENTER R AX-
ثم ادخل الرقم 13 ثم ENTER
أطبع ما يلي أمام إشارة السالب - R BX ثم ENTER R BX-
ثم ادخل الرقم 3 ثم ENTER
أطبع ما يلي أمام إشارة السالب - E 100 ثم ENTER E 100-
بعد ذلك ادخل 01 ثم اضغط على العارضة الطولية SPACE BAR
ادخل ألان الرقم D8 ثم ENTER

1) **ما هي القيم او محتويات كل من**
   i. **المسجل AX .....**
   ii. **المسجل BX .....**
   iii. **المسجل IP .....**
2) **ما هي التعليمة التي تم إدخالها في الموقع 100 .....**
3) **للحصول على النتيجة ادخل الأمر T والذي يعني TRACE وهو أمر يستخدم لتنفيذ تعليمات لغة التجميع واحدة تلو الأخرى**

ملاحظة:
1- قبل استخدام الأمر T تأكد من أن المسجل IP = 100 فإذا كانت قيمته غير ذلك نفذ الخطوات التالية:
ادخل الأمر التالي R IP ثم ENTER -R IP
بعد ذلك ادخل 100 ثم اضغط على ENTER
2- جميع الأرقام بالنظام السادس عشر

4) **ما هي النتيجة المخزنة في المسجل AX بعد تنفيذ التعليمة .....**

الفرع الثاني

1- تأكد من أن المسجل IP = 100 ثم ادخل ما يلي

```
-E 200
41 4C 51 55 44 53 20 4A 50 45 4E 20 55 4E 49 56 45 52 53 49 54 59 24
```

لا تنسى أن تضغط على مفتاح ENTER بعد إدخال جميع القيم السابقة. القيم السابقة يمكن الحصول عليها من المقرر صفحة 338 شيفرة أسكي بالنظام السادس عشر

ادخل التعليمات التالية, لإدخال التعليمات استخدم الأمر A كما يلي –A 100

بعد ذلك ادخل التعليمات التالي كل تعليمة على سطر متبوعة بمفتاح الإدخال ENTER

```
MOV  AH,09
MOV  DX,200
INT  21
INT  20
```

ثم مفتاح الإدخال ENTER حتى تظهر الإشارة - . يمكن تنفيذ التعليمات باستخدام الأمر G أو الأمر T

ما هي النتيجة التي حصلت عليها

.....

.....

2- أعتمد على الخطوات السابقة وبالرجوع إلى الصفحة رقم 338 من المقرر، اكتب محتويات الذاكرة وتعليمات لغة التجميع لطباعة العبارة ASSEMBLY LANGUAGE

.....

.....

.....

.....

.....

1- تأكد من أن IP = 100 ثم ادخل التعليمات التالية كما يلي:

```
-A 100
MOV  CX,0A
MOV  AX,0
ADD  AX,CX
LOOP 106
INT  20
```

نفذ التعليمات السابقة, ويفضل أن تستخدم الأمر T لمتابعة مكونات المسجلات

بين ما هي محتويات المسجل AX .....

ما هي الوظيفة التي تؤديها التعليمات السابقة .....

الفرع الثالث

باستخدام البرنامج DEBUG والخوارزمية التالية اكتب تعليمات لغة التجميع لطباعة الحروف الانجليزية من A..L الخوارزمية :

1- اسند قيمة الرمز A وهي 41H إلى المسجل DX

2- اسند القيمة 02 إلى المسجل AH لطباعة الرمز

.....

.....

.....

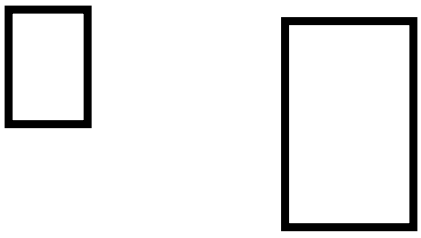
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

8. المراجع

1. مقرر هيكلية الحاسوب ولغة التجميع 1381، جامعة القدس المفتوحة، 2003.
2. http://thestarman.pcministry.com/asm/debug/debug.htm
3. http://www.armory.com/~rstevew/Public/Tutor/Debug/debug2.htm
4. https://www.princeton.edu/ssp/trips/data/debug_tutorial.pdf

الوحدة الثانية

مقدمة الى لغة التجميع اسمبلي



1. المقدمة

1.1 تمهيد

عزيزي الطالب، أهلاً بك إلى الوحدة الثانية من التطبيق العملي لمقرر هيكلية الحاسوب ولغة التجميع اسمبلي وهي بعنوان مقدمة الى لغة التجميع اسمبلي.

تهدف هذه الوحدة إلى إعطائك معلومات أساسية فيما يتعلق بالتعامل مع برامج الاسمبلي وبرنامج المجمع من حيث لمراحل التي يمر بها هذا البرنامج باستخدام نظام الأسمبلي، البرامج اللازمة استخدام نظام الأسمبلي وخطوات التنفيذ

وقد تم تزويد هذه الوحدة بالعديد من الأمثلة التوضيحية، والتدريبات، إضافة إلى بعض أسئلة التقويم الذاتي والإشكال التي توضح بعض المفاهيم.

مرة أخرى نرحب بك في هذه الوحدة، آملين أن تستمتع بدراستك للمادة التي نعرضها لك في هذه الوحدة.

2.1 أهداف الوحدة

ينتظر منك، عزيزي الطالب، بعد دراسة هذه الوحدة أن تكون قادراً على:

1. تتعرف على البرامج اللازمة لتنفيذ برامج الاسمبلي.
2. تتعرف على برنامج الاسمبلي وكيفية تشغيله والتعامل معه.
3. تتعرف على مراحل معالجة برنامج التجميع اسمبلي.
4. تتمكن من كتابة برنامج الاسمبلي.
5. تتمكن من تخزين برنامج اسمبلي باستخدام محرر النصوص.
6. تتمكن من تنفيذ برنامج اسمبلي.
7. تتمكن من تصحيح الأخطاء في برنامج لغة التجميع اسمبلي.

3.1 أقسام الوحدة

تتكون هذه الوحدة من ثلاثة أقسام رئيسة. فالقسم الأول يوضح. المراحل التي يمر بها هذا البرنامج باستخدام نظام الأسمبلي، اما القسم الثاني فيعرض البرامج اللازمة استخدام نظام الأسمبلي من حيث ادخال البرامج وعملية التجميع بوساطة المجمع MASM وعملية الربط بوساطة LINK. القسم الثالث فيوضح كيفية ادخال برنامج لغة التجميع وتنفيذه.

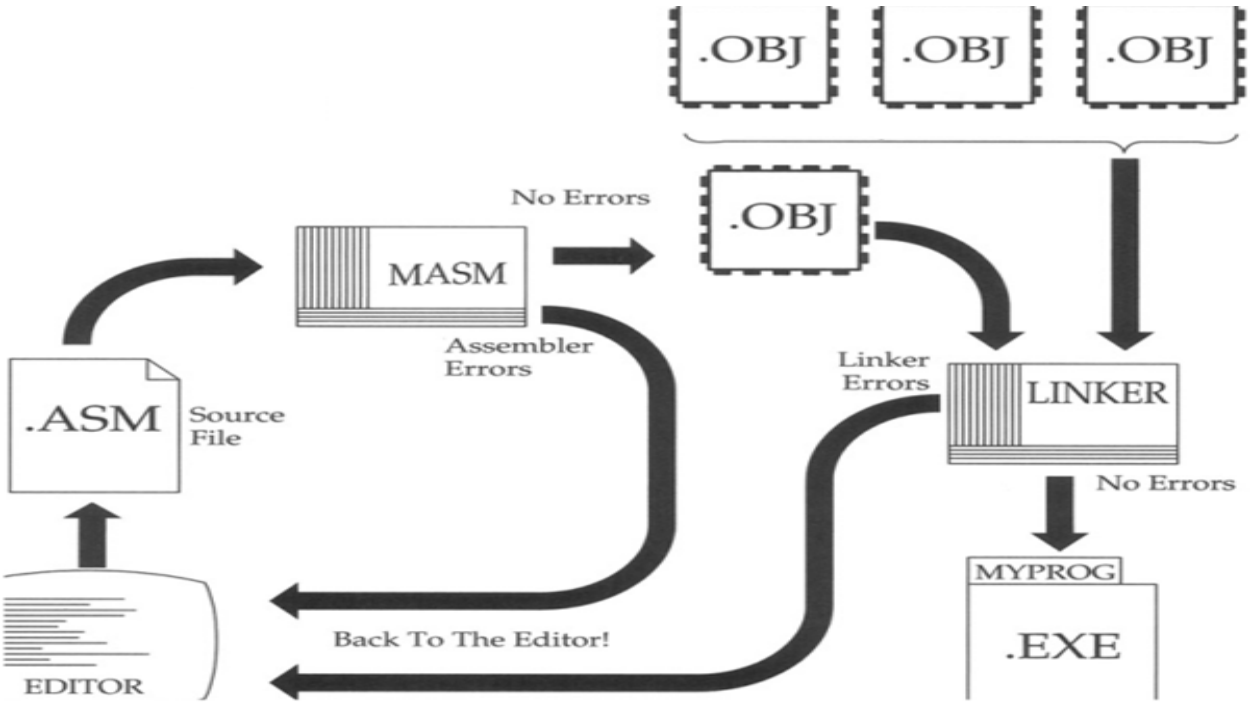
4.1 ما تحتاج إليه لدراسة الوحدة

لقد تضمنت هذه الوحدة العديد من المفاهيم الأساسية، وبذلك فان طبيعة المادة المعروضة، عزيزي الطالب، تحتاج إلى جو هادئ خاص ومريح حتى تستطيع التركيز على المفاهيم المعطاة، وتستوعبها بالشكل المناسب. وكل ما تحتاجه بعد ذلك بعض القرطاسية كقلم رصاص وورق أبيض، حاسوب وبرمجيات نظام الاسمبلي (المجمع) لتقوم بتطبيق بعض المفاهيم وتحل الأسئلة والامثلة والتدريبات المعطاة في ثنايا الوحدة. ويمكنك عزيزي الطالب استخدام الانترنت للبحث عن بعض المفاهيم والمواضيع المتعلقة برنامج Debug. كما ان هناك حاجة الى:

3. معرفة بأساسيات مراحل تنفيذ برامج لغة التجميع اسمبلي. (راجع الوحدة الرابعة من الكتاب المقرر).
4. معرفة بأساسيات لغة الاسمبلي لمعالجة تعليمات الاخراج. (راجع الوحدة السبعة من الكتاب المقرر).

1. المراحل التي يمر بها هذا البرنامج باستخدام نظام الأسمبلي

سنتعرف عزيزي الطالب على المراحل التي يمر بها البرنامج لغة التجميع اسمبلي باستخدام نظام الأسمبلي MASM، علماً بأن هذه المراحل تطبق لأي برنامج بلغة الأسمبلي شكل (1) .



شكل (1) **مراحل معالجة برنامج الأسمبلي**

1.1 المرحلة الأولى :ادخال البرنامج Program input

تهدف هذه المرحلة إلى نقل البرنامج إلى الحاسوب وتخزينه على الذاكرة الثانوية) القرص الصلب أو القرص المرن او USB .)وعادة نستخدم لوحة المفاتيح من خلال أحد برامج معالجة النصوص Word processor أو محرر النصوص Text editor يتم ادخال البرنامج ابتداء من الرمز الأول في السطر الأول حتى نهاية السطر، ومن ثم الرمز الأول في السطر الثاني، ... وهكذا.

كما نود ان نذكرك عزيزي الطالب ان البرنامج المكتوب بلغة الأسمبلي يحب أن يخزن ايضاً في ملف .يتم اختيار أسماء الملفات التي تحتوي برامج لغة الأسمبلي حسب قواعد نظام التشغيل .(DOS) بحيث يحتوي الشق الثاني دائماً الرموز .(ASM) وعلى سبيل المثال نفرض أن البرنامج الذي تم إدخاله من قبل قد خزن في ملف اسمه. SOURC1. ASM

2.1. المرحلة الثانية :ترجمة البرنامج Program Compilation

تهدف مرحلة الترجمة إلى:

-اكتشاف وتصحيح الأخطاء المطبعية والقواعدية، أي الأخطاء الناجمة عن مخالفة قواعد اللغة.

-تحويل البرنامج المصدري إلى برنامج هدفي.

تنفذ مرحلة الترجمة بواسطة مترجم الاسمبلر الذي يحمل اسم MASM وكما هو مبين في الشكل) 1 (فإن الاسمبلر لا يكون البرنامج الهدفي إلا في حالة خلو البرنامج المصدري من كافة الأخطاء المطبعية والقواعدية .وفي حالة وجود مثل هذه الأخطاء يصدر الأسمبلر قائمة تتضمن رقم الخطأ، ورقم السطر الذي يحتوي الخطأ، ووصف قصير للخطأ .إن وجود اخطاء قواعدية يتطلب من المبرمج الرجوع إلى مرحلة ادخال البرنامج لإجراء التصحيحات والتعديلات اللازمة وتستمر عملية الانتقال بين مرحلتي الإدخال والترجمة لحين الحصول على البرنامج الهدفي .
لا تنسى، عزيزي الطالب، أن تخزن البرنامج على الذاكرة الثانوية بعد إجراء أي تغيير عليه، فالتغيير الذي تجريه عن طريق لوحة المفاتيح يتم على البرنامج المخزن في الذاكرة الرئيسة في حين أن الأسمبلر يعالج البرنامج المخزن في الذاكرة الثانوية.

3.1. المرحلة الثالثة :الربط والتحرير Program Linking

أن البرنامج الهدفي الناتج من مرحلة الترجمة لا يكون جاهزاً للتنفيذ بسبب احتوائه على رموز لا يستطيع برنامج الأسمبلر ترجمتها إلى لغة الآلة في مرحلة الترجمة .فمثلاً، يمكن أن يحتوي البرنامج على استدعاء لبرامج مكتبية أو برامج أخرى للمستخدم تمت ترجمتها سابقاً بشكل منفصل عن البرنامج الحالي .أن ترجمة مثل هذه الاستدعاءات تتم في مرحلة الربط والتحرير وتتطلب البحث عن هذه البرامج ودمجها مع البرنامج الحالي لتكوين برنامج تنفيذي واحد.

تنفذ مرحلة الربط والتحرير بوساطة برنامج الربط والتحرير Linker ، الذي يستقبل برنامجاً برنامجاً هدفياً واحداً (أو مجموعة برامج هدفية)مخزن على الذاكرة المساعدة ويحوله إلى برنامج تنفيذي .يلاحظ أن اسم البرنامج الهدفي يتضمن عبارة .(Obj) في حين أن اسم البرنامج التنفيذي يتضمن عبارة EXE .

يتكون البرنامج التنفيذي فقط في حالة خلو البرنامج الهدفي(البرامج الهدفية)من الأخطاء .وفي حالة وجود اخطاء في عملية الربط والتحرير يجب الرجوع إلى مرحلة الادخال لإجراء التعديلات اللازمة، واجراء عملية الترجمة، ومن ثم تنفيذ عملية الربط والتحرير.

وتستمر عملية الانتقال بين مراحل الإدخال والترجمة والربط والتحرير لحين تكوين البرنامج التنفيذي

4.1.المرحلة الرابعة :تنفيذ البرنامج Program Execution))

أن البرنامج الناتج بعد مرحلة الربط والتحرير يصبح جاهزاً للتنفيذ .تهدف مرحلة التنفيذ إلى الحصول على النتائج المطلوبة.ويوجد طريقتان لتنفيذ البرامج هما:

-الطريقة المباشرة من خلال نظامDOS

-استخدام برنامج» مكتشف الأخطاءDebug .

تستخدم الطريقة المباشرة عادة في الحالات التي يتضمن فيها البرنامج وسائل لإدخال المعطيات واخراج النتائج إلى وحدات الإخراج المناسبة .تستخدم الطريقة الثانية في الحالات التي لا يتضمن البرنامج فيها مثل هذه الوسائل أو الحالات التي تؤدي إلى نتائج خاطئة أو لا تعطي أي نتائج سبب وجود اخطاء منطقية في البرنامج نفسه.

وتبعاً للطريقة المباشرة يتم تنفيذ البرنامج بكتابة اسم البرنامج فقط SOURCE1 بعد الحصول على اشارة جاهزية الحاسوب لاستقبال الأمر التالي .وهنا يبدأ الحاسوب فوراً بتنفيذ البرنامج طبقاً لتسلسل التعليمات المكونة له.

وتبعاً للطريقة الثانية يتم تنفيذ البرنامج من خلال استدعاء برنامج (Debug) الذي بدوره يطلب من المستخدم طباعة الأوامر عن طريق لوحة المفاتيح .يستخدم برنامج Debug لتنفيذ العمليات الآتية:

-تنفيذ البرنامج دفعة واحدة أو تعليمة بعد تعليمه.

-عرض محتويات المسجلات ومواقع الذاكرة الرئيسة حسب الطلب.

-تعديل محتويات المسجلات ومواقع الذاكرة الرئيسة حسب الحاجة.

-تحويل التعليمات المصدرية (Source Instructions) إلى لغة الآلة وبالعكس.

-البحث في الذاكرة الرئيسة، مليء مجموعة مواقع بقيمة معينة، ...وغيرها.

2.البرامج اللازمة استخدام نظام الأسمبلي

- برنامج تحرير النصوص edit او Notepad
- برنامج مترجم الاسمبلر MASM
- برنامج التحرير والربط LINK
- برنامج اكتشاف الأخطاء DEBUG

3. خطوات تنفيذ برامج التجميع أسمبلي

- ننتقل الى الدليل الفرعي الذي يحتوي على مجمع برنامج أسمبلي.
- ادخال نص البرنامج المصدري بوساطة أحد معالجات النصوص
- تخزينه في ملف اسمه SOURCE1.ASM فمثلا hello.asm
- يستخدم برنامج الأسمبلر MASM لتحويل البرنامج المصدري إلى برنامج هدفي بواسطة الأمر التالي، متبوع بفاصلة منقوطة ;

C> MASM
C> MASM SOURCE1.ASM;

Source filename [.ASM]: SOURCE1
Object filename [SOURCE1.OBJ]: SOURCE1
Source listing [NUL.LST]: SOURCE1
Cross- reference [NUL.CRF]: SOURCE1

- بعد إكمال عملية الترجمة يلزم تحويل البرنامج الهدفي الناتج إلى برنامج تنفيذي بوساطة برنامج الربط والتحريك الذي يستدعى من خلال الامر:

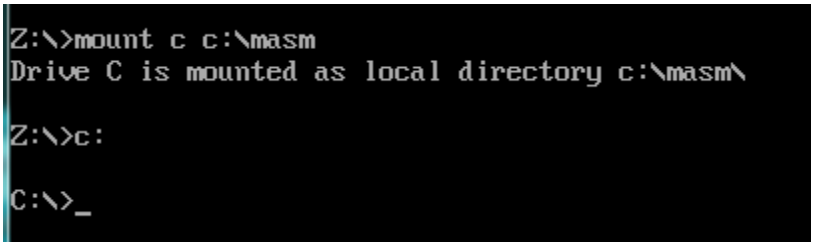
C> LINK
C> LINK SORCE1;
Object Modules [.OBJ]: SOURCE1
Run File [SOURCE1.EXE]: SOURCE1
List File [NUL.MAP]: SOURCE1
Libraries [.LIB] :

- ندخل اسم البرنامج SOURCE1 لتنفيذه

4.المثال الاول

لنتبع خطوات كتابة برنامج ليطبع العبارة "Hello World" على الشاشة

1- ننتقل الى الدليل الفرعي الذي يحتوي على مجمع برنامج أسمبلي وفي مثالنا تم استخدام دليل قرعي باسم **MASM**. باستخدام الامر التالي من الدوس DOS **CD MASM**



2- ادخال نص البرنامج المصدري بوساطة أحد معالجات النصوص . توجيهات وتعليمات لغة أسمبلي التالية تقوم بطباعة العبارة "**Hello World**" على الشاشة.

```
hellostk  SEGMENT BYTE STACK 'STACK'    ;Define the stack segment
          DB 100h DUP(?)    ;Set maximum stack size to 256 bytes (100h(
hellostk  ENDS

hellodat  SEGMENT BYTE 'DATA' ;Define the data segment
dos_print EQU 9                    ;define a constant via EQU
strng     DB  'Hello World',13,10,'$'  ;Define the character string
hellodat  ENDS

hellocod  SEGMENT BYTE 'CODE' ;Define the Code segment. list
          assume cs:hellocod,ds:hellodat,ss:hellostk
START:    mov ax, SEG hellodat       ;ax <-- data segment start address
          mov ds, ax                 ;ds <-- initialize data segment register
          mov ah, dos_print          ;ah <-- 9 DOS 21h string function
          mov dx,OFFSET strng        ;dx <-- beginning of string
          int 21h                    ;DOS service interrupt
```

```
        mov ax, 4c00h           ;ax <-- 4c DOS 21h program halt function
        int 21h                 ;DOS service interrupt
hellocod  ENDS
        END     START           ; ?END label? defines program entry
```

3- تخزين الملف باسم hello.asm في نفس الدليل الفرعي الذي يحتوي على مجمع لغة اسمبلي

4- نستخدم برنامج الأسمبلر MASM لتحويل البرنامج المصدري إلى برنامج هدفي بواسطة الأمر التالي، متبوع بفاصلة منقوطة ;

C\ MASM> **MASM hello.asm;**

```
C:\>masm hello.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51716 + 464828 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>_
```

5- بعد إكمال عملية الترجمة بدون أخطاء يلزم تحويل البرنامج الهدفي الناتج إلى برنامج تنفيذي بوساطة برنامج الربط الذي يستدعى من خلال الامر:

C\ MASM> **LINK hello;**

```
C:\>link hello;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>
```

6- بعد إكمال عملية التحويل بدون أخطاء ندخل اسم البرنامج **hello** لتنفيذه ثم مفتاح الادخال. فيطبع لنا البرنامج العبارة **Hello World** على الشاشة.

```
C:\>hello
Hello World

C:\>_
```

بالرجوعالى البرنامج السابق نرى انه أن لدينا سلسلة من الرموز في قطاع المعطيات تبدأ بالبايت المسمّى MES وتنتهي بالبايت الذي يحتوي رمز إشارة الدولار .$ المطلوب عرض سلسلة الرموز على الشاشة.

```
strng    DB  'Hello World',13,10,'$'  ;Define the character string
```

وبالرجوع إلى الوحدة السابعة نجد أن عرض سلسلة رموز على الشاشة يمكن أن تنفد بإستخدام المهمة رقم 9 في الإعتراض رقم 21H كما في سلسلة التعليمات الآتية:

```
        mov ah, dos_print          ;ah <-- 9 DOS 21h string function
        mov dx,OFFSET strng        ;dx <-- beginning of string
        int 21h                    ;DOS service interrupt
```

نوضح كل تعليمة على النحو التالي:

1- تستخدم تعليمة MOV لتخزين رقم المهمّة (9) في المسجل AH حيث ان الثابت dos_print يحتوي القيمة 9.

```
dos_print EQU 9                    ;define a constant via EQU
```

2- تستخدم تعليمة mov dx,OFFSET strng ;dx <-- beginning of string لتخزين قيمة الإزاحة لسلسلة الرموز strng في المسجل DX. ويمكن ان نستخدم تعليمة LEA لتخزين قيمة الإزاحة لسلسلة الرموز strng في المسجلDX. LEA dx, strng ;dx <-- beginning of string

3-تستخدم التعليمة INT لتنفيذ مهمة عرض سلسلة الرموز.

وتجدر الإشارة هنا إلى أن سلسلة الرموز strng المطلوب عرضها على الشاشة يجب أن تنتهي بإشارة .$ وفي حالة عدم وجود هذه الإشارة في نهاية سلسلة الرموز فإن عملية عرض الرموز من الذاكرة الرئيسية على الشاشة سوف تستمر لحين الوصول إلى بايت يحتوي إشارة.$

تدريب (1)

اعد كتابة البرنامج السابق لطباعة اسمك ,رقمك الجامعي على الشاشة

الحل

```
hellostk  SEGMENT BYTE STACK 'STACK'    ;Define the stack segment
          DB 100h DUP(?)    ;Set maximum stack size to 256 bytes (100h(
hellostk  ENDS

hellodat  SEGMENT BYTE 'DATA' ;Define the data segment
dos_print EQU 9                    ;define a constant via EQU
str1     DB  'Yousef Abuzir',13,10,'$'  ;Define the character string
str2     DB  '0123012002120',13,10,'$'  ;Define the character string
hellodat  ENDS

hellocod  SEGMENT BYTE 'CODE' ;Define the Code segment. list
         assume cs:hellocod,ds:hellodat,ss:hellostk
START:    mov ax, SEG hellodat       ;ax <-- data segment start address
          mov ds, ax                 ;ds <-- initialize data segment register
          mov ah, dos_print          ;ah <-- 9 DOS 21h string function
          mov dx,OFFSET str1         ;dx <-- beginning of string
          int 21h                    ;DOS service interrupt

          mov dx,OFFSET str2         ;dx <-- beginning of string
          int 21h                    ;DOS service interrupt

          mov ax, 4c00h              ;ax <-- 4c DOS 21h program halt function
          int 21h                   ;DOS service interrupt
hellocod  ENDS
          END      START             ; ?END label? defines program entry
```

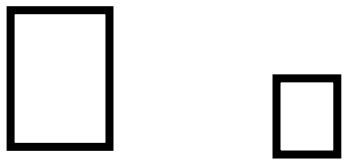
بعد تنفيذ البرنامج اجب عن الأسئلة التالية:

1- ما هو التغيير على البرنامج السابق؟

2- كيف خزنت اسمك ورقمك الجامعي في البرنامج؟

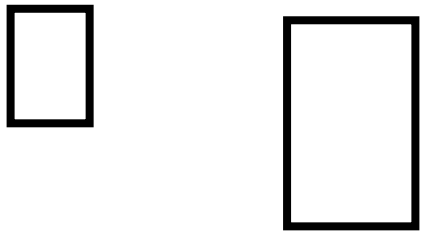
3- ماهي تعليمات الطباعة؟

4- كيف تعرف البرنامج على البيانات التي سيتم طباعتها؟



الوحدة الثالثة

العمليات الحسابية في لغة التجميع اسمبلي



1.مقدمة

أما بالنسبة الوحدة الثالثة من العملي فهي مرتبطة بالوحدة السادسة من الكتاب وتعالج طاقم التعليمات الخاصة بالمعالج الدقيق من نوع . 8086 / 8088 وستناول في هذه الوحدة الصيغة العامة لتعليمات لغة أسمبلي والتعليمات المختلفة من عليمات نقل البيانات، التعليمات الحسابية،. وسيتم الحديث عن تعليمات نقل التحكم، تعليمات معالجة البيانات الثنائية، تعليمات معالجة سلاسل الرموز في الوحدة التالية.

وستجد في ثنايا الوحدة أمثلة وتدريبات وأسئلة تقويم ذاتي على المواضيع المختلفة في الوحدة لمساعدتك على فهم مادة المقرر.

أهلاً بك في رحاب هذه الوحدة وأرجو أن تنتفع بموضوعاتها

.

2.أهداف الوحدة

بعد دراسة هذه الوحدة وتنفيذ التطبيق العملي على الحاسوب ينتظر منك عزيزي الطالب أن تكون قادراً على:

1-تعرف طرق تمثيل البيانات الرقمية والرمزية المختلفة

2-تبين طاقم تعليمات المعالج الدقيق

3-تذكر التعليمات المختلفة وكيفية استخدامها

4-تكتب برامج بلغة أسمبلي لحل مسائل مختلفة

5-تحلل برامج لغة أسمبلي عن طريق تتبع تعليماتها

3.المثال الأول: ADD32.asm

اكتب برنامج في لغة التجميع اسمبلي لجمع عددين صحيحين طول كل منهما 32 بت 32 bit (القيم بالنظام السادس عشر)، حيث ان القيم الأدنى مستوى من العدد الأول والثاني سيتم جمعها وتخزينها في المسجل DX و القيم الاعلى مستوى ن العدد الأول والثاني في المسجل CX

ملاحظة (ضرورة مراجعة عمليات الجمع مع الحمل ADC)

val2	val1	العدد الأول
2222h	1111h	

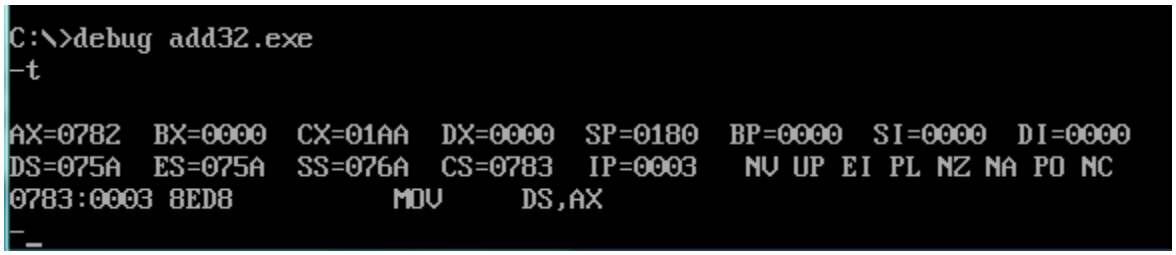
val4	val3	العدد الثاني
4444h	3333h	

```
data segment
   val1 dw 1111h
   val2 dw 2222h
   val3 dw 3333h
   val4 dw 4444h
data ends

code segment
   assume cs:code, ds:data
start :
   mov ax, data
   mov ds, ax

   mov dx, val1
   add dx, val3

   mov cx, val2
   adc cx, val4
```

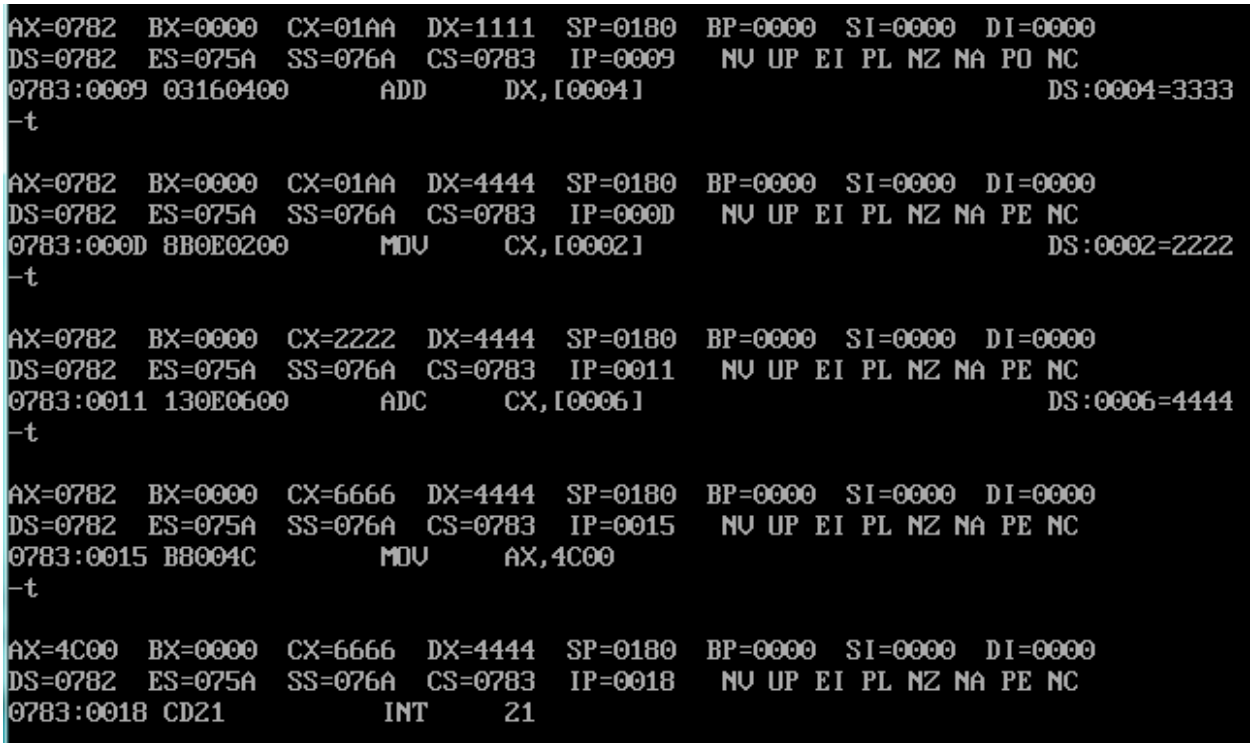
```
    mov ax, 4c00h
    int 21h
code ends
    end start
```

للتأكد من النتائج يمكن استخدام برنامج DEBUG لمتابعة خطوات تنفيذ البرنامج والقيم في كل من مواقع الذاكرة والمسجلات. للقيام بذلك

Debug add32.exe ثم مفتاح الادخال



نستخدم الامر T لتنفيذ جمل البرنامج تعليمة تعليمه ومتابعة القيم في المسجلات كما يلي:



أسئلة

1- ما هو العدد الأول .....

و ما هو العدد الثاني .....

2- ما نتيجة جمع العددين .....

3- اين تم تخرين النتيجة او حاصل جمع العددين .....، بين ذلك بالرسم

4- لماذا استخدمنا كلا ADD و ADC لجمع العددين .....

5- ما هي وظيفة التعليمات التالية

```
mov ax, 4c00h
int 21h
```

المثال الثاني (nxchang.asm)

اكتب برنامج في لغة التجميع اسمبلي لتبديل محتويات الموقعين او المتغيرين A و B (القيم بالنظام السادس عشر)، حيث ان الموقع الأول يحتوي على قيمة A = 10 الموقع الثاني يحتوي على قيمة B = 20

ملاحظة (ضرورة مراجعة تعليمات النقل)

```
stk  SEGMENT BYTE STACK 'STACK'    ;Define the stack segment
        DB 100h DUP(?)    ;Set maximum stack size to 256 bytes (100h)
stk  ENDS
DATA SEGMENT
A DB 10
B DB 20
DATA ENDS
CODE SEGMENT
```

```
     ASSUME DS:DATA,CS:CODE, ,ss: stk
START:
     MOV AX,DATA
     MOV DS,AX
     MOV AL,A
     MOV AH,B
     MOV A,AH
     MOV B,AL
     MOV BL,A
     MOV BH,B
     MOV AH,4CH
     INT 21
CODE ENDS
END START
```

خطوات التنفيذ

```
C:\>masm nxchng.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51746 + 464798 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link nxchng;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>nxchng

C:\>
```

المتابعة باستخدام برنامج المنقح Debug

```
C:\>debug nxchng.exe
-r
AX=FFFF  BX=0000  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076A  CS=077A  IP=0002   NV UP EI PL NZ NA PO NC
077A:0002 B87A07        MOV     AX,077A
-t

AX=077A  BX=0000  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076A  CS=077A  IP=0005   NV UP EI PL NZ NA PO NC
077A:0005 8ED8          MOV     DS,AX
-
```

```
AX=077A  BX=0000  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=0007   NV UP EI PL NZ NA PO NC
077A:0007 A00000        MOV     AL,[0000]                       DS:0000=0A
-t

AX=070A  BX=0000  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=000A   NV UP EI PL NZ NA PO NC
077A:000A 8A260100      MOV     AH,[0001]                       DS:0001=14
-t

AX=140A  BX=0000  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=000E   NV UP EI PL NZ NA PO NC
077A:000E 88260000      MOV     [0000],AH                       DS:0000=0A
-
```

```
AX=140A  BX=0000  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=0015   NV UP EI PL NZ NA PO NC
077A:0015 8A1E0000      MOV     BL,[0000]                       DS:0000=14
-t

AX=140A  BX=0014  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=0019   NV UP EI PL NZ NA PO NC
077A:0019 8A3E0100      MOV     BH,[0001]                       DS:0001=0A
-t

AX=140A  BX=0A14  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=001D   NV UP EI PL NZ NA PO NC
077A:001D B44C          MOV     AH,4C
-
```

```
AX=0A14  BX=140A  CX=0121  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=077A  ES=075A  SS=076A  CS=077A  IP=001D   NV UP EI PL NZ NA PO NC
077A:001D B44C          MOV     AH,4C
```

تدريب (1)

اكتب برنامجا بلغة التجميع اسمبلي لجمع عددين وإيجاد معدلهما ثم تخزين النتيجة في الذاكرة.

الحل:

```
; Program to save average of 2 numbers in Assembly Language


stack segment
stack ends
data segment
  temp1 dw 6h
  temp2 dw 8h
  avg1 db 2h
  temp3 db ?
data ends
code segment
   assume cs:code,ds:data
   start:
      mov ax,data
      mov ds,ax


      mov ax,temp1
      add ax,temp2
      div avg1
      mov temp3,al


      mov ax,4c00h
      int 21h
code ends
  end start
```

ادرس البرنامج او الحل السابق وحاول الاجابة على الأسئلة التالية، يمكنك إعادة تنفيذ البرنامج بالتعديلات او الأسئلة المطلوبة:

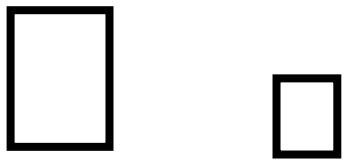
1- هل يمكن ان نستبدل avg1 ب القيمة 2 في التعليمية div avg1، مع بيان السبب .....

2- لماذا لم نجمع محتويات المتغيرات temp1 و temp2 مباشرة .....

3- هل يمكن تخزين قيمة المتغير temp1 في المسجل AL ولماذا .....

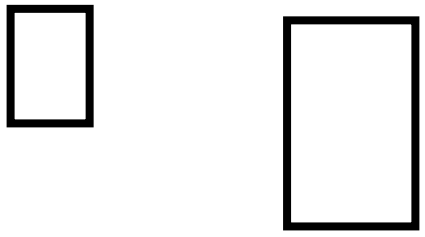
4- لماذا استخدم المتغير temp3 في البرنامج .....

5- ماذا تعني إشارة السؤال في التوجيهة التالية temp3 db ? .....



الوحدة الرابعة

عمليات التحكم في لغة التجميع اسمبلي



1.مقدمة

إن الشكل الاعتيادي لسير تنفيذ تعليمات أي برنامج من البرامج هو الشكل التسلسلي، أي أن التعليمات تنفذ حسب موقعها في البرنامج .إلا أنه وفي بعض الأحيان قد يتطلب الأمر تغيير هذا التسلسل والانتقال بسير تنفيذ البرنامج إلى تعليمة معينة وفقاً للتسلسل المنطقي للبرنامج وليس وفقاً للتسلسل الموقعي.

تمتلك لغة أسمبلي مجموعة من التعليمات المخصصة لهذه الغاية، حيث تستخدم هذه التعليمات لنقل التحكم وتغيير سير تنفيذ البرنامج.

يمكن تصنيف تعليمات نقل التحكم إلى أربع مجموعات هي:

-تعليمات القفز غير المشروط واستدعاء البرنامج الفرعي والعودة منهJMP, CALL, RET

-تعليمات نقل التحكم المشروطConditional Transfer Instructions

-تعليمات التكرارIteration Control Instructions

-تعليمات الاعتراضInterrupt Instructions

وفيما يأتي امثلة توضح وشرح بعض هذا التعليمات: من خلال تطبيقات برمجية على الحاسوب.

2.المثال الأول (compXY.as)

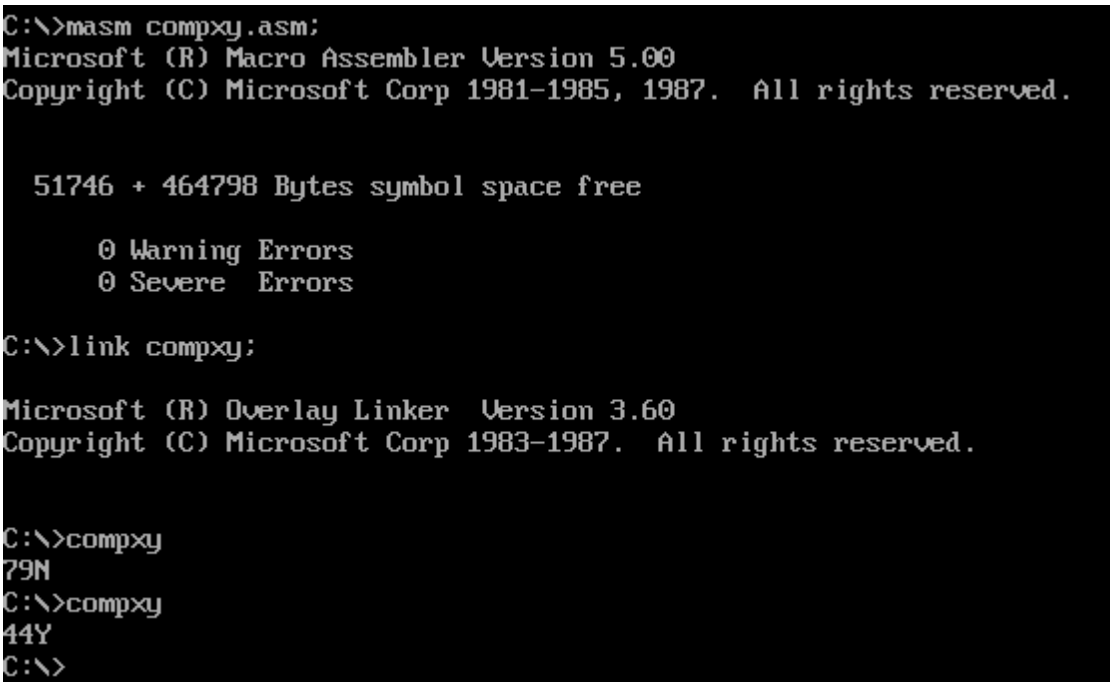
اكتب برنامج لإدخال عددين وطباعة Y في حالة كون العددين متساوين ويطبع N في حالة عدم مساواة العددين.

```
STACK  SEGMENT PARA STACK 'STACK'
     DB  64 DUP('STACK('
STACK  ENDS

DATA SEGMENT PARA PUBLIC 'DATA'
*****************************;
        xx  db  0
```

```
        yy  db  0

;*****************************
DATA  ENDS
CODE SEGMENT PARA PUBLIC 'CODE'
;*****************************
    ASSUME CS:CODE,DS:DATA,SS:STACK
    PROG PROC FAR
        PUSH DS
        MOV  AX,0H
        PUSH AX
        MOV  AX,DATA
        MOV  DS,AX
;************************
         mov cx,1
next:    mov ah,1
         int 21h

         mov xx,al
         ;mov ah,21h
         mov ah,1
       int 21h
         mov yy ,al
         mov bh,xx
         mov bl,yy
         cmp bh,bl
         jne notequal
eqlual:
         mov dl,'Y'
         mov  ah,2    ; Writes one character to the standard output (screen). DOS 1.x function (obsolete(
         int 21h
         jmp continue;
```

```
notequal:
        mov dl,'N'
        mov ah,2
        int 21h
continue:
        loop next

*************************;
      RET
PROG    ENDP
CODE    ENDS
      END  PROG
```

خطوات التنفيذ والنتائج



تدريب 1

اكتب برنامج بلغة التجميع اسمبلي لمقارنة عددين A= 10 , B= 20 وطباعة احد العبارات التالية متساوين او اكبر او اصغر،

1- بحيث تكون النتائج كما يلي:

يطبع العبارة التالية اذا العددين متساويين BOTH A AND B ARE EQUAL

يطبع العبارة التالية إذا العدد الأول A أكبر من العدد الثاني A IS GREATER THAN B B

يطبع العبارة التالية إذا العدد الأول A اصغر من العدد الثاني B B IS GREATER THAN A

2-اسند قيم جديدة للمتغيرات A= 20 , B= 20 ما ناتج التنفيذ .....

3-اسند قيم جديدة للمتغيرات A= 70 , B= 50 ما ناتج التنفيذ .....

4- لماذا اسنادنا قيم المتغير A الى المسجل AL .....

ادرس الحل "البرنامج" الموجود للتدريب ثم اجب عن الأسئلة التالية:

5- ما هي جملة المقارنة المستخدمة في البرنامج وكيف تعمل .....

6- متى ننتقل الى الإشارة LAB1 .....

7- متى ننتقل الى الإشارة LAB2 .....

8-متى نطبع محتوى العبارة MSG3 .....

9- ما هي وظيفة التعليمات التالية :

..... MOV AH,09H

..... JMP END1

..... JZ LAB1

..... LEA DX,MSG3

..... INT 21H

البرنامج:

```
; A PROGRAM TO COMPARE TO MEMORY LOCATION in Assembly Language
DATA SEGMENT
    A DB 10H
```

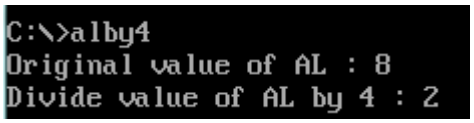
```
        B DB 20H
        MSG1 DB 10,13,"BOTH A AND B ARE EQUAL$"
        MSG2 DB 10,13,"A IS GREATER THAN B$"
        MSG3 DB 10,13,"B IS GREATER THAN A$"
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        START:
                MOV AX,DATA
                MOV DS,AX


                MOV AL,A
                CMP AL,B
                JZ LAB1
                JA LAB2
                MOV AH,09H
                LEA DX,MSG3
                INT 21H
                JMP END1
        LAB1:
                MOV AH,09H
                LEA DX,MSG1
                INT 21H
                JMP END1
        LAB2:
                MOV AH,09H
                LEA DX,MSG2
                INT 21H
                JMP END1


         END1:   MOV AX,4C00H
                INT 21H
```

```
CODE ENDS
      END START
```

```
OUTPUT
**********
B IS GREATER THAN A
```

3.المثال الثاني (ALby4.asm)

اكتب برنامج بلغة التجميع اسمبلي لقسمة محتويات المسجل AL على القيمة 4 وطباعة النتائج بعناوين واضحة ، بحيث تكون النتائج كمل يلي:



استخدم تعليمات الازاحة لليمن SHR instruction

```
stk  SEGMENT BYTE STACK 'STACK'    ;Define the stack segment
        DB 100h DUP(?)    ;Set maximum stack size to 256 bytes (100h)
stk  ENDS

dat  SEGMENT BYTE 'DATA' ;Define the data segment
   PROMPT_1  DB  'Original value of AL : $'
   PROMPT_2  DB  0DH,0AH,'Divide value of AL by 4 : $'
dat  ENDS

cod  SEGMENT BYTE 'CODE' ;Define the Code segment
assume cs: cod,ds: dat,ss: stk
START: mov ax, SEG dat        ;ax <-- data segment start address
```

```
 mov ds, ax                ;ds <-- initialize data segment register     MOV DS,

AX


    LEA DX, PROMPT_1           ; load and print PROMPT_1
    MOV AH, 9
    INT 21H


    MOV AL, 38H                ; place 8 in the AL
    MOV BL, AL                 ; save Al in to BL


    MOV AH, 2                  ; print the original value of AL
    MOV DL, AL
    INT 21H


    LEA DX, PROMPT_2           ; load and print PROMPT_2
    MOV AH, 9
    INT 21H


    AND BL, 0FH                ; convert ascii to decimal code


    MOV CL, 2                  ; divide AL by 4
    SHR BL, CL


    OR BL, 30H                 ; convert decimal to ascii code


    MOV AH, 2                  ; print the processed value of AL
    MOV DL, BL
    INT 21H
```

```
    MOV AH, 4CH              ; return control to DOS
    INT 21H
  cod  ENDS
END START
```

خطوات التنفيذ والحل:

```
C:\>alby4
Original value of AL : 8
Processed value of AL : 2
C:\>masm alby4.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51716 + 464828 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link alby4;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>alby4
Original value of AL : 8
Divide value of AL by 4 : 2
C:\>
```

4.المثال الثالث (eono.asm)

اكتب برنامج بلغة التجميع اسمبلي لإيجاد عدد الأرقام الفردية وعدد الأرقام الزوجية في مصفوفة من الأرقام قائمة الاعداد هي من 1 الى 9 . استخدم برنامج debug للتأكد من النتائج.

```
;Program to find the total no. of even and odd numbers. from an array in Assembly
DATA SEGMENT
A DW 1,2,3,4,5,6,7,8,9,10
DATA ENDS
```

```
CODE SEGMENT
      ASSUME DS:DATA,CS:CODE
START:
    MOV AX,DATA
    MOV DS,AX
    LEA SI,A
    MOV DX,0000
    MOV BL,02
    MOV CL,10
  L1:MOV AX,WORD PTR[SI]
    DIV BL
    CMP AH,00
    JNZ L2
    INC DH
    JMP L3
  L2:INC DL
  L3:
    ADD SI,2
    DEC CL
    CMP CL,00
    JNZ L1
    MOV AH,4CH
    INT 21H
CODE ENDS
END START
```

```
;OUTPUT:->

;AX=0005  BX=0002  CX=0000  DX=0505  SP=0000  BP=0000  SI=0014  DI=0000
```

```
;DS=0BF4  ES=0BE4  SS=0BF4  CS=0BF6  IP=0029   NV UP EI PL ZR NA PE NC
;0BF6:0029 B44C          MOV     AH,4C
```

ادرس البرنامج جيدا واجب عن الأسئلة التالية.

1–نفذ البرنامج على الحاسوب واستخدم برنامج debug لمتابعة وتنفيذ البرنامج وما هي القيمة المخزنة في المسجلات CX و AX و DX وما هي دلالتها لك. .....

2–اين يتم تخزين عدد الأرقام الزوجية .....

3–اين يتم تخزين عدد الأرقام الفردية .....

4–لماذا اسنادنا للمسجل القيمة 10 عن طريق التعليمة MOV CL,10

5–هل تتغير قيمة المسجل CX في البرنامج .....

- اذا كانت اجابتك بنعم فكيف واي تعليمة تغير قيمة المسجل .....
- وهل تتغير بالزيادة ام بالنقصان ولماذا .....

6–ما أهمية التعليمة CMP CL,00 .....

7–لماذا نضيف العدد 2 الى المسجل SI في التعليمة التالية: ADD SI,2 .....

8–اذا اردنا ان نضيف 1 الى المسجل SI ويبقى البرنامج يعمل بنفس الأسلوب ما هو التعديل المطلوبة على البرنامج .....

9–هل يمكن ان نستعمل التعليمة INC بدلا من ADD SI,1 .....

10– ما هي وظيفة التعليمة MOV AX,WORD PTR[SI] .....

<u>5.المثال الرابع (2s.asm)</u>

اكتب برنامج بلغة التجميع اسمبلي يقوم بإيجاد المتم الثنائي لأي عدد ثنائي مخزن في متغير طوله 8 ثنائيات. استخدم البرنامج debug لمتابعة خطوات الحل والتأكد من النتيجة.

الحل

```
; Program to find 2's Complement of given binary  umber in Assembly Language
Data Segment
```

```
  num db 00000010B
Data Ends
Code Segment
  Assume cs:code, ds:data
  Begin:
    mov ax, data
    mov ds, ax
    mov es, ax
    mov ah, 0000h
    mov al, num
    NOT al
    mov bl, al
    adc al, 00000001B
    mov bl, al
  Exit:
    mov ax, 4c00h
    int 21h
Code Ends
End Begin
```

```
C:\>masm 2s.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51766 + 464778 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors

C:\>link 2s;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment
```

متابعة البرنامج باستخدام البرنامج debug

```
C:\>debug 2s.exe
-t

AX=076A  BX=0000  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076B  IP=0003   NV UP EI PL NZ NA PO NC
076B:0003 8ED8          MOV     DS,AX
```

```
AX=0002  BX=0000  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=076A  SS=0769  CS=076B  IP=000C   NV UP EI PL NZ NA PO NC
076B:000C F6D0          NOT     AL
-t

AX=00FD  BX=0000  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=076A  SS=0769  CS=076B  IP=000E   NV UP EI PL NZ NA PO NC
076B:000E 8AD8          MOV     BL,AL
-t

AX=00FD  BX=00FD  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=076A  SS=0769  CS=076B  IP=0010   NV UP EI PL NZ NA PO NC
076B:0010 1401          ADC     AL,01
-t

AX=00FE  BX=00FD  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=076A  SS=0769  CS=076B  IP=0012   NV UP EI NG NZ NA PO NC
076B:0012 8AD8          MOV     BL,AL
-t

AX=00FE  BX=00FE  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=076A  SS=0769  CS=076B  IP=0014   NV UP EI NG NZ NA PO NC
076B:0014 B8004C        MOV     AX,4C00
```

تدريب (2)

اكتب برنامج بلغة التجميع اسمبلي لإيجاد اكبر واصغر عدد في مجموعة من الاعداد في مصفوفة مكونة من ستة ارقام مخزن كل عدد منها في بايت واحد (8 ثنائيات) .

الحل

```
;Program to find the largest and smallest number from an array of n 8 bit numbers
DATA SEGMENT
A DB 5,2,5,6,4,3
B DB ?
```

```
DATA ENDS
CODE SEGMENT
    ASSUME DS:DATA,CS:CODE
START:
    MOV AX,DATA
    MOV DS,AX
    MOV CX,0000
    MOV CL,06
    LEA BX,A
    MOV AL,00
    MOV AH,BYTE PTR[BX]
  L1:CMP AL,BYTE PTR[BX]
    JNC L2
    MOV AL,BYTE PTR[BX]
  L2:CMP AH,BYTE PTR[BX]
    JC L3
    MOV AH,BYTE PTR[BX]
  L3:INC BX
    DEC CL
    CMP CL,00
    JNZ L1
    MOV AH,4CH
    INT 21H
CODE ENDS
END START
```

```
;OUTPUT:->
;-G CS: 0025
;
;AX=0206  BX=0006  CX=0000  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
;DS=0BF4  ES=0BE4  SS=0BF4  CS=0BF5  IP=0025   NV UP EI PL ZR NA PE NC
;0BF5:0025 B44C        MOV     AH,4C
```

6.المثال الخامس (nooc.asm)

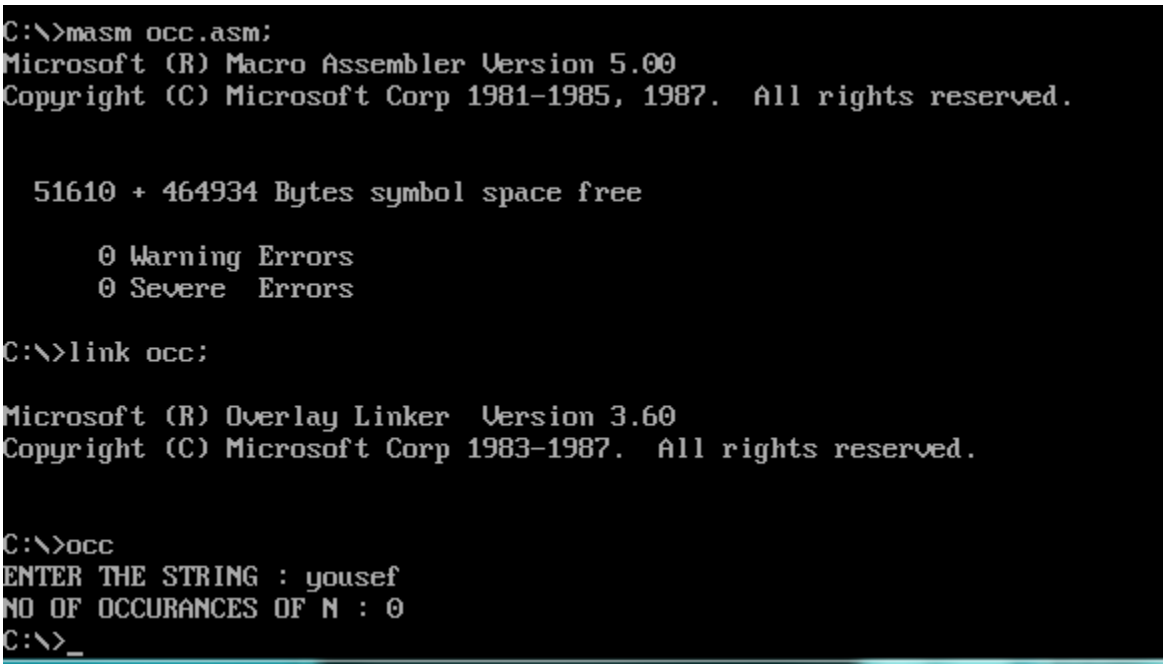
أكتب برنامج بلغة التجميع اسمبلي لإيجاد عدد مرات تكرار الحرف "N" في سلسلة رمزية يتم إدخالها من لوحة المفاتيح.

البرنامج

```
;Program to find the no. of occurrences of character 'N' in the input;string
stk  SEGMENT BYTE STACK 'STACK'    ;Define the stack segment
        DB 100h DUP(?)    ;Set maximum stack size to 256 bytes (100h)
stk  ENDS
DATA SEGMENT
      MSG1 DB "ENTER THE STRING : $"
      MSG2 DB "NO OF OCCURANCES OF N : $"
      NEWLINE DB 10,13,'$'
      STR1 DB 80 DUP('$')
      CNT DB 0
DATA ENDS
CODE SEGMENT
      ASSUME DS:DATA,CS:CODE, SS:stk
      START :
            MOV AX,DATA
            MOV DS,AX

            MOV AH,09H
            LEA DX,MSG1
            INT 21H

            MOV AH,0AH
            LEA DX,STR1
            INT 21H

            MOV AH,09H
            LEA DX,NEWLINE
```

```
        INT 21H

        LEA SI,STR1+1
        MOV CL,BYTE PTR[SI]
        MOV CH,00H

   L1   :  INC SI

        CMP BYTE PTR[SI],'N'

        JNZ L2
        INC CNT
     L2 : LOOP L1

        ADD CNT,'0'

        MOV AH,09H
        LEA DX,MSG2
        INT 21H

        MOV AH,02H
        MOV DH,00H
        MOV DL,CNT
        INT 21H

        MOV AH,4CH
        INT 21H
CODE ENDS
    END START
```

```
;OUTPUT
;
;       ENTER THE STRING : Nablus
;       NO OF OCCURANCES OF N : 1
```



عدل البرنامج لإيجاد عدد مرات تكرر الحرف n بغض النظر اذا كان صغير ام كبير.

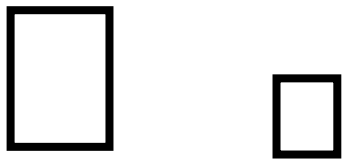
تدريب (3) (nooc.asm)

أكتب برنامج بلغة التجميع اسمبلي لإيجاد عدد مرات تكرار الأرقام السالبة والأرقام الموجبة في سلسلة من الأرقام مخزنة في مصفوفة باسم num وخزن النتائج في متغيرين الأول p لعدد تكرار الاعداد الموجبة و n لتخزين عدد مرات تكرار الاعداد السالبة

البرنامج

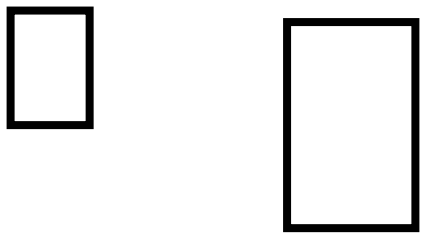
```
;Program to count the number of +ve ( positive ) and _ve ( negative ) numbers
data segment
        num db -3,1,-5,6,-7,9,'#'
        p_cnt db 0h
        n_cnt db 0h
data ends
code segment
        assume ds:data,cs:code
start:
```

```
        mov ax,data
        mov ds,ax

        lea si,num
main:
        cmp num[si],0h
        jg pos
        inc si
        add n_cnt,01h
        cmp num[si],'#'
        je exit
        jmp main

pos:
        add p_cnt,01h
        inc si
        cmp num[si],'#'
        je exit
        jmp main

exit:
        mov bl,p_cnt
        mov cl,n_cnt

        mov ax,4c00h
        int 21h

code ends
        end start
```



الوحدة الخامسة

برمجة عمليات الإدخال والإخراج في لغة التجميع اسمبلي



1.مقدمة

عزيزي الطالب، أهلاً بك في رحاب الوحدة الخامسة من دليل التطبيق العملي لمقرر هيكلية الحاسوب ولغة الأسمبلي .« تشكل هذه الوحدة برمجة عمليات الإدخال والإخراج وتعتبر جزءً مهماً في لغة الأسمبلي وذلك لأن معظم التطبيقات تتطلب استخدام عمليات الإدخال والإخراج وبشكل مكثف.

سوف نقدم لك عزيزي الطلب بعض الأمثلة التطبيقية لنوضح أساليب برمجة عمليات الإدخال/الإخراج وبالتحديد :عمليات الإدخال/الإخراج المبرمجة وعمليات الإدخال/الإخراج الموجهة بالاعتراضات. وسوف نناقش وننفذ امثلة عملية لتوضيح مهام الاعتراضات21H, 17H, 16H, 10H ودورها في برمجة تلك العمليات

2.أهداف الوحدة

بعد انتهائك من دراسة هذه الوحدة وتنفيذ التطبيق العملي والبرنامج على الحاسوب ينتظر منك عزيزي الطالب أن تكون قادراً على أن:

1-تكتب برامج لتنفيذ عمليات الإدخال بواسطة لوحة المفاتيح.

2-تكتب برامج لتنفيذ عمليات الإخراج إلى الشاشة والآلة الطابعة

3.المثال الأول

اكتب برنامج لطباعة العبارة HELLO WORLD على الشاشة باستخدام امر الاعتراض INT 21H والوظيفة رقم 09H

الحل

```
PROGRAM TO DISPLAY HELLO WORLD ON THE CONSOLE in Assembly Language
DATA SEGMENT
   MSG DB "HELLO WORLD$"
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
START:    MOV AX,DATA
   MOV DS,AX
```

```
    MOV AH,09H
    MOV DX,OFFSET MSG
    INT 21H
    MOV AH,4CH
    INT 21H
CODE ENDS
END START
```

اجب عن الأسئلة التالي

1- ما وظيفة التعليمة التالية MOV AH,09H .....

2- ما وظيفة التعليمة التالية INT 21H

3- ما وظيفة التعليمة التالية MOV DX,OFFSET MSG

4- ما أهمية إشارة الدولار الأمريكي في $ في نهاية السلسة الرمزية MSG DB "HELLO WORLD$"

4.المثال الثاني (cls.asm)

اكتب برنامجا بلغة التجميع اسمبلي لمسح الشاشة

الحل والتنفيذ

```
;PROGRAM TO CLEAR SCREEN in Assembly Language
STACK SEGMENT
STACK ENDS

DATA SEGMENT
DATA ENDS
CODE SEGMENT BYTE 'CODE' ;Define the Code segment
   ASSUME CS:CODE, ds:data, ss:stack
START:
```

```
    MOV AX,0600H    ;06 TO SCROLL & 00 FOR FULLJ SCREEN
    MOV BH,71H    ;ATTRIBUTE 7 FOR BACKGROUND AND 1 FOR FOREGROUND
    MOV CX,0000H    ;STARTING COORDINATES
    MOV DX,184FH    ;ENDING COORDINATES
    INT 10H       ;FOR VIDEO DISPLAY
    MOV AH,4CH    ;RETURN TO DOS MODE
    INT 21H
CODE ENDS
END START
```

ادرس المثال السابق واجب على الأسئلة

1- ما هي وظيفة التعليمات التالية:

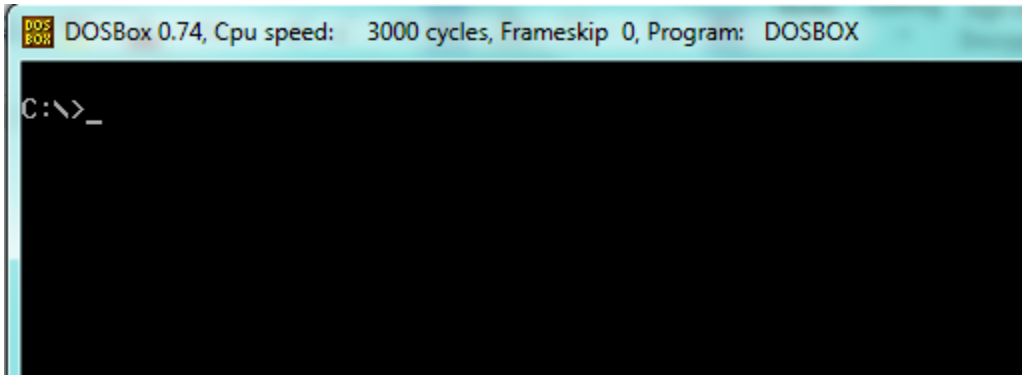
```
    MOV AX,0600H    ;06 TO SCROLL & 00 FOR FULLJ SCREEN
    MOV BH,71H    ;ATTRIBUTE 7 FOR BACKGROUND AND 1 FOR FOREGROUND
    MOV CX,0000H    ;STARTING COORDINATES
    MOV DX,184FH    ;ENDING COORDINATES
```

2-ما هي وظيفة الاعتراض INT 10H

تنفيذ البرنامج

```
C:\>masm cls.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51770 + 464774 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link cls;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

C:\>cls_
```

وعند تنفيذ البرنامج cls يتم مسحح الشاشة



5.المثال الثالث (3-19.asm)

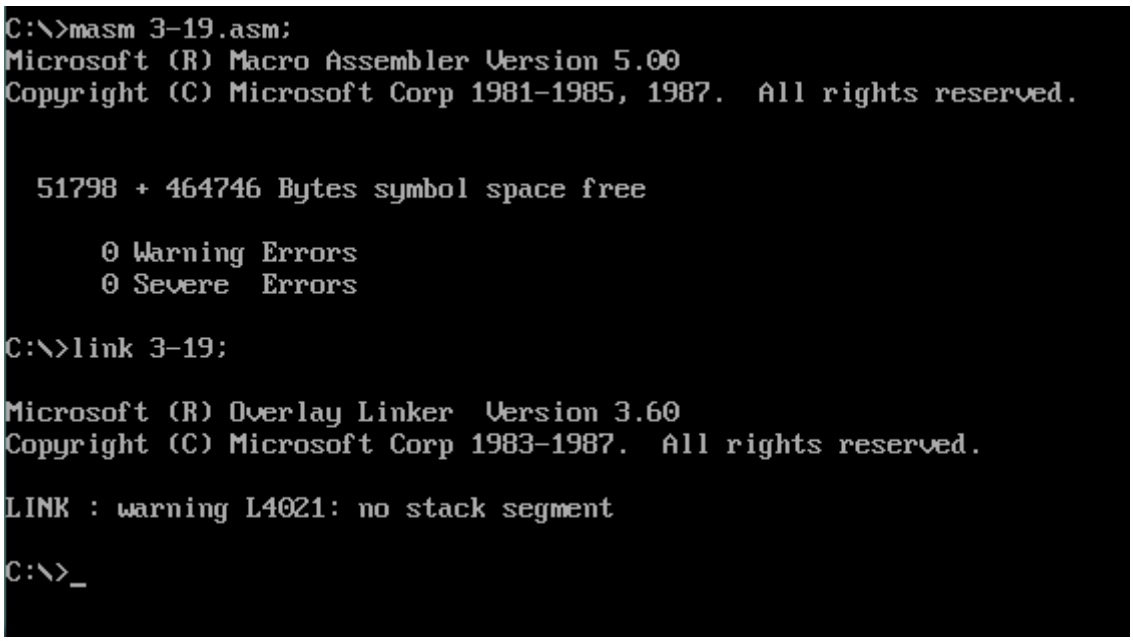
اكتب برنامجا بلغة التجميع اسمبلي لوضع المؤشر في السطر الثالث العمود 19 على الشاشة

الحل والتنفيذ

```
Code for Program to set the cursor on row-3 and column-19 in Assembly Language
STACK SEGMENT
STACK ENDS
DATA SEGMENT
DATA ENDS
CODE SEGMENT BYTE 'CODE' ;Define the Code segment
      assume cs:code,ds:data,ss:stack


 MOV AX,data
 MOV DS,AX


 MOV AH,02H
 MOV BH,00
 MOV DH,3
 MOV DL,19
 INT 10H
```

```
  MOV Ax,4C00H
  INT 21H
code ends
END
```

تنفيذ البرنامج السابق



ادرس البرنامج السابق واجب عن الأسئلة الاتية:

1–ما هي التعليمات **بلغة التجميع اسمبلي لوضع المؤشر في السطر الثالث العمود 19 على الشاشة**

.....

.....

.....

2– ما الاعتراض المستخدم للتعامل مع الشاشة.

.....

6.المثال (4) (border.asm)

اكتب برنامجا بلغة التجميع اسمبلي لوضع اطار ملون على الشاشة

الحل:

```
;Program to Print a Color Border on the Screen in Assembly Language
stack segment
stack ends
data segment
data ends
code segment BYTE 'CODE' ;Define the Code segment
   assume cs:code,ds:data,ss:stack


  mov ah,0bh
  mov bx,01
  int 10h

  mov ah,4ch
  int 21h
code ends
END
```

ادرس البرنامج السابق واجب عن الأسئلة الاتية:

1–ما هي التعليمات **بلغة التجميع اسمبلي** لوضع اطار ملون على الشاشة

.....

.....

2– ما الاعتراض المستخدم للتعامل مع الشاشة.

.....

تنفيذ البرنامج السابق

```
C:\>masm border.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51788 + 464756 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link border;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment
```

7.المثال (5) (blue.asm)

اكتب برنامجا بلغة التجميع اسمبلي لمسح الساسة وجعل لون نمط الخط الظاهر على الخلفية ازرق

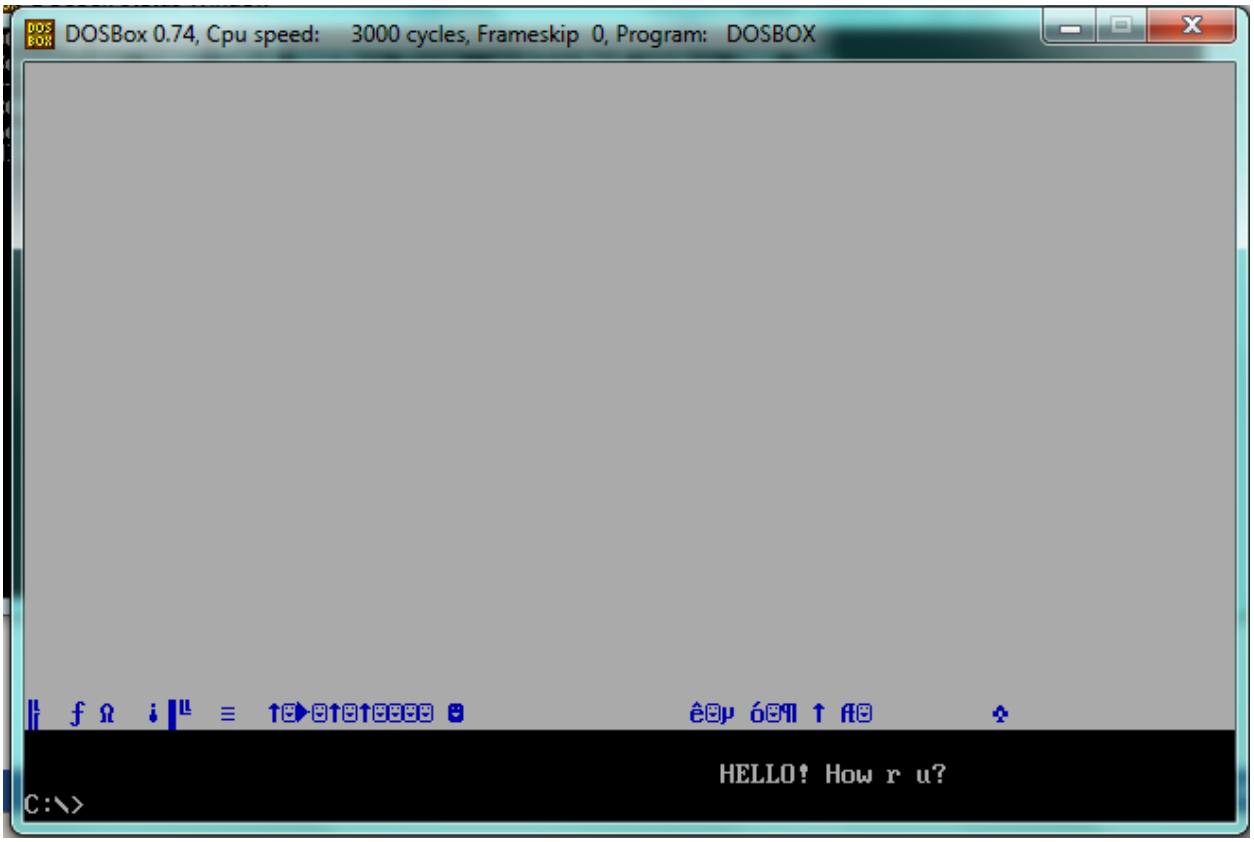
الحل:

```
PROGRAM TO CLEAR THE SCREEN AND MAKE BLUE FOREGROUND
stack segment
stack ends
data segment
    MSG1 DB 'HELLO! How  are you?$'
    MSG2 DB ' Hi!!!!!!!!!!!!!!!!!!$'
data ends
code segment BYTE 'CODE' ;Define the Code segment
  assume cs:code,ds:data,ss:stack


  MOV AX, DATA
  MOV DS,AX


  MOV AX,0600h          ;06 isfor row and 00 isfor column
```

```
  MOV BH,71H          ;7 is used for white background
  MOV CX,0000H         ;and 1 is used for blue foreground
  MOV DX,184FH
  INT 10H
                       ; Upper Left corner to Lower right corner
  MOV dx,OFFSET MSG1
  mov ah,09h
  int 21h


  mov ah,4ch
  int 21h
code ends
END
```

```
C:\>masm blue.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51760 + 464784 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link blue;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment
```

لتنفيذ البرنامج اطبع blue ثم مفتاح الادخال فتكون نتيجة التنفيذ كما يلي:



تدريب (1) (1-9.asm)

اكتب برنامج لطباعة الاعداد من 0 الى 9 على الشاشة

الحل

```
;Code for Program to print the digits 0, 1… 9 in Assembly Language
stack segment
stack ends
DATA SEGMENT
DATA ENDS
CODE SEGMENT BYTE 'CODE' ;Define the Code segment
      ASSUME DS:DATA,CS:CODE,ss:stack
      START :
            MOV AX,DATA
            MOV DS,AX
```

```
    MOV BL,00H
    MOV CH,00H
    MOV CL,0AH

 L1  :  MOV DH,00H
    MOV DL,BL
    ADD DL,'0'
    MOV AH,02H
    INT 21H
    INC BL
    LOOP L1
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

خطوات الحل والنتيجة

```
C:\>masm 1-9.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51770 + 464774 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link 1-9;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

C:\>1-9
0123456789
C:\>_
```

تدريب (2)(atoa.asm)

اكتب برنامج يقوم بإدخال سلسلة رمزية ويقوم بتحويل الاحرف الصغيرة او الكبيرة الى أحرف صغيرة او كبيرة وطباعة الاحرف

الحل

```
stack segment BYTE STACK 'STACK'    ;Define the stack segment
        DB 255h DUP(?)    ;Set maximum stack size to 256 bytes (100h)
stack ends
DATA SEGMENT
           MSG  DB  0DH,0AH, ' ENTER THE STRING :-----> :  $'
           MSG2 DB  0DH,0AH, ' YOUR STRING IS  :-----> :  $'
           STR1 DB  255 DUP(?)
           ONE  DB ?
           TWO  DB ?
DATA ENDS
CODE SEGMENT BYTE 'CODE' ;Define the Code segment
     ASSUME DS:DATA,CS:CODE,ss:stack
START :
       MOV AX,DATA
       MOV DS,AX


       LEA DX,MSG
       MOV AH,09H
       INT 21H


       LEA SI,STR1
       MOV AH,01H
```

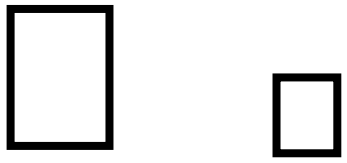
```
READ:
        INT 21H
        MOV BL,AL

        CMP AL,0DH
        JE  DISPLAY

        XOR AL,20H
        MOV [SI],AL
        INC SI

        ;CMP BL,0DH
        JMP READ

DISPLAY:

        MOV AL,'$'
        MOV [SI],AL

        LEA DX,MSG2
        MOV AH,09H
        INT 21H

        LEA DX,STR1
        MOV AH,09H
        INT 21H

        MOV AH,4CH
       INT 21H
```

```
CODE ENDS
END START
```

نتيجة التنفيذ

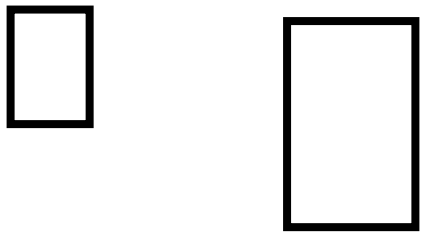
```
C:\>masm atoa.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51716 + 464828 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link atoa;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>atoa

 ENTER THE STRING :-----> :  yousef

 YOUR STRING IS  :-----> :  YOUSEF
C:\>
```

```
C:\>atoa

 ENTER THE STRING :-----> :  yousef

 YOUR STRING IS  :-----> :  YOUSEF
C:\>atoa

 ENTER THE STRING :-----> :  YOUSEF

 YOUR STRING IS  :-----> :  yousef
```



الوحدة السادسة

الماكرو واستخدامها في لغة التجميع اسمبلي



1.مقدمة

أهلاً بك عزيزي الطالب في رحاب الوحدة السادسة والأخيرة في دليل التطبيق العملي من مقرر هيكلية الحاسوب ولغة أسمبلي ، وهي تعالج موضوع الماكرو واستخدامها .

هذه الوحدة تعالج مواضيع أساسية عن الماكرو وهذه المواضيع هي:

-مكونات الماكرو حيث يتكون الماكرو من جملة البداية وجملة النهاية ومتن الماكرو.

-جملة استدعاء الماكرو.

وتم تزويد هذه الوحدة ب أمثلة وتدريبات على المواضيع المختلفة في الوحدة لمساعدتك على فهم مادة المقرر.

2.المثال الأول (mactoc.asm)

اكتب برنامجا بلغة التجميع اسمبلي لطباعة عنوان او اسم على وسط الشاشة، استخدم ماكرو باسم PRNSTR لطباعة العنوان على الشاشة

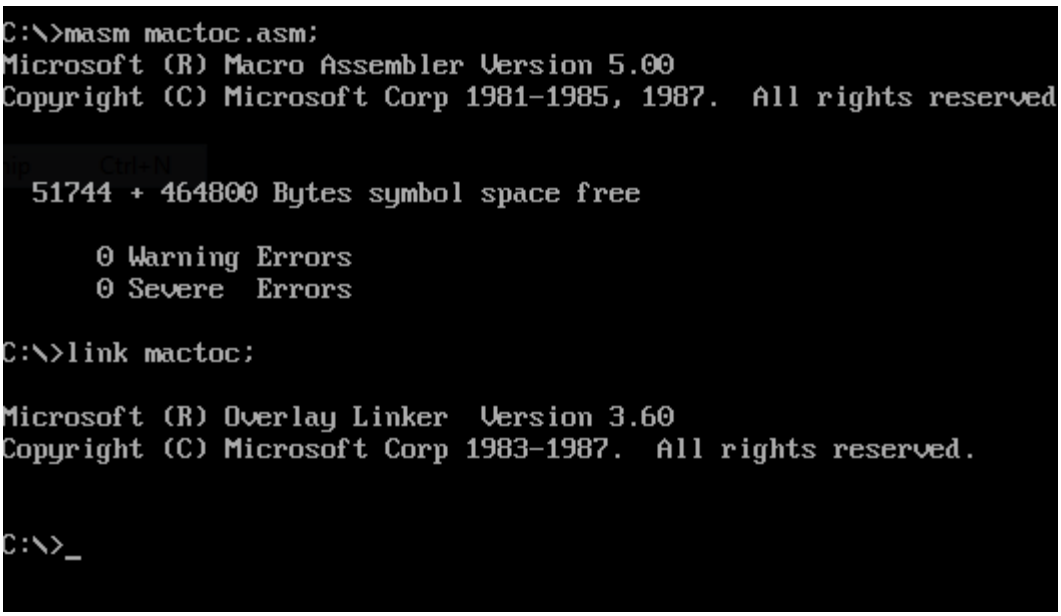
الحل:

```
;PROGRAM TO DISPLAY NAME IN CENTER OF SCREEN
PRNSTR MACRO MSG
   MOV AH,09H
   LEA DX,MSG
   INT 21H
ENDM
stack segment BYTE STACK 'STACK'    ;Define the stack segment
         DB 255h DUP(?)    ;Set maximum stack size to 256 bytes (100h)
stack ends
DATA SEGMENT
   MSG1 DB "Al-Quds Open University $"
DATA ENDS


CODE SEGMENT BYTE 'CODE' ;Define the Code segment
   ASSUME CS:CODE,DS:DATA ,ss:stack
```

```
START:
   MOV AX,DATA
   MOV DS,AX
   MOV AX,0600H
   MOV BH,71H
   MOV CX,000H    ;UPPER LEFT ROW,COLUMN
   MOV DX,184H    ;LOWER RIGHT ROW,COLUMN
   INT 10H
   MOV AH,02H
   MOV BH,00H
   MOV DH,0CH
   MOV DL,23H
   INT 10H
   PRNSTR MSG1
   MOV AH,4CH
   INT 21H
CODE ENDS
END START
```

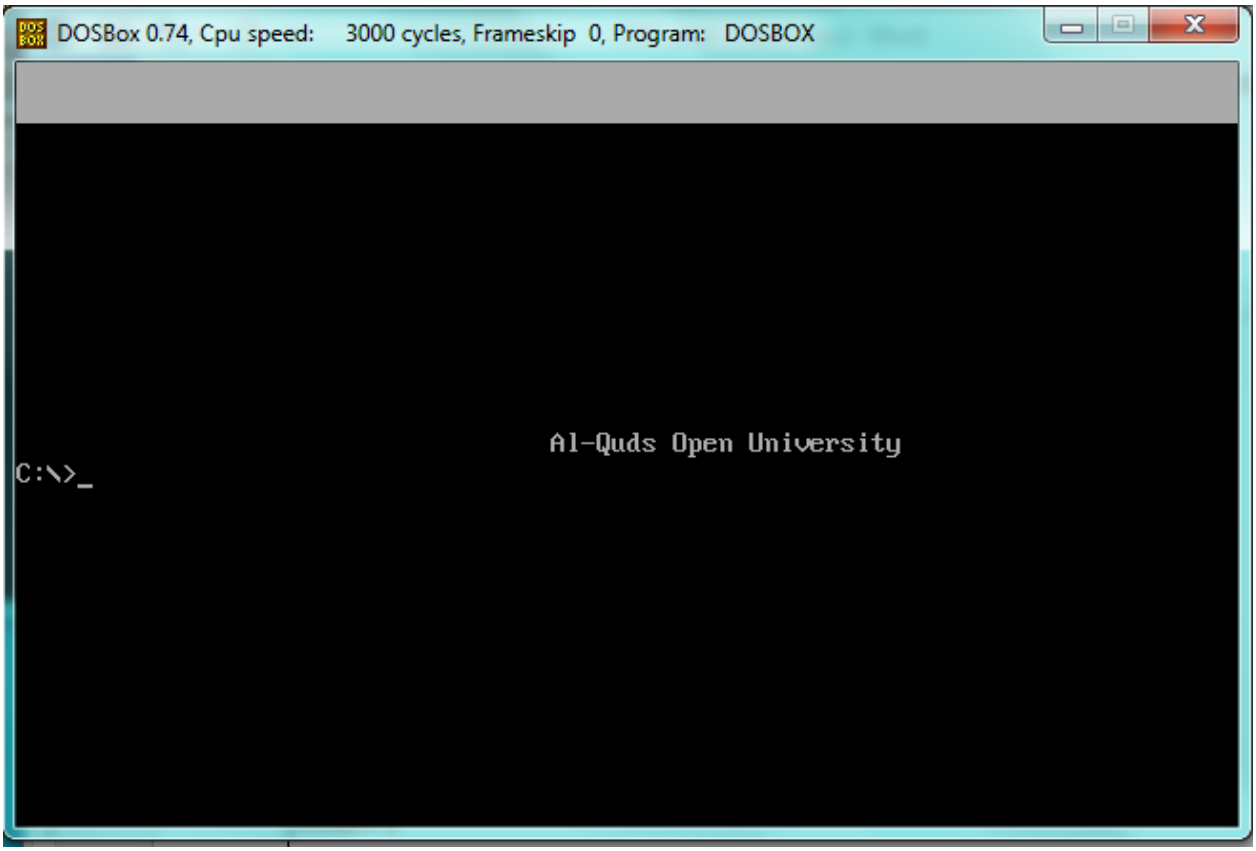
خطوات ونتيجة التنفيذ

الشكل التالي يبين خطوات تنفيذ البرنامج



النتيجة

نتيجة التنفيذ، يفضل تنفيذ البرنامج cls الذي كتبناه سابقا قبل تنفيذ البرنامج بهدف مسح الشاشة



أسئلة:

ادرس البرنامج السابق جيد ثم اجب عن الأسئلة التالية:

ما اسم الماكرو .....

كيف اعلنا عن انتهاء الماكرو .....

كيف نستدعي الماكرو ..... وما هي جملة الاستدعاء .....

في أي مكان من البرنامج يكتب الماكرو .....

3.المثال الثاني (REVERSE.asm)

اكتب برنامج بلغة التجميع أسمبلي يستدعي ماكرو لإظهار عبارة لإدخال النص وماكرو اخر لطباعة النتائج، بحيث يقوم البرنامج بقراءة نص وطباعة بالمعكوس

بحيث تكون النتائج كما يلي:

ENTER THE STRING :abc
cba REVERSE STRING IS

الحل

```
;Code for PROGRAM TO FIND THE REVERSE OF A STRING USING MACRO;OUTPUT
;**********
;C:\masm\reverse
;ENTER THE STRING :abc
;cba REVERSE STRING IS

GETSTR MACRO STR
MOV AH,0AH
LEA DX,STR
INT 21H
ENDM

PRINTSTR MACRO STR
MOV AH,09H
LEA DX,STR
INT 21H
ENDM

DATA SEGMENT
   STR1 DB 80,80 DUP('$')
```

```
   STR2 DB 80,80 DUP('$')
   MSG1 DB 10,13,'ENTER THE STRING :$'
   MSG2 DB 10,13,'THE REVERSE STRING IS :$'
   STORE DB 2 DUP('$')
DATA ENDS

CODE SEGMENT

      ASSUME CS:CODE,DS:DATA
      START:

            MOV AX,DATA
            MOV DS,AX

            PRINTSTR MSG1
            GETSTR STR1
            PRINTSTR MSG2

            LEA SI,STR1+2
            LEA DI,STR2+2

            MOV CL,STR1+1    ;FOR STORING THE LENGTH OF THE STRING
            MOV CH,00H
            MOV BL,CL
            MOV BH,00H

       LAB1:
            INC SI         ;FOR GOING TO THE END OF THE STRING
            LOOP LAB1
```

```
        MOV CX,BX

    LAB2:
        MOV AL,DS:BYTE PTR[SI] ;FOR COPYING CONTENTS OF STR1 TO AL
        MOV DS:BYTE PTR[DI],AL ;FOR COPYING CONTENTS OF AL TO STR2
        INC DI
        DEC SI
        LOOP LAB2

    PRINTSTR STR2+2
    MOV AH,DS:BYTE PTR[SI]
    MOV STORE,AH
    PRINTSTR STORE



    MOV AX,4C00H
    INT 21H

CODE ENDS
    END START
```

خطوات ونتيجة التنفيذ

الشكل التالي يبين خطوات تنفيذ البرنامج والنتيجة

أسئلة:

ادرس البرنامج السابق جيد ثم اجب عن الأسئلة التالية:

1. ما عدد الماكرو في البرنامج.....
2. ما اسماء الماكرو .....
3. ما هي وظيفة كل ماكرو .....
4. كيف تم استدعاء كل الماكرو ..... وما هي جمل الاستدعاء .....
5. ما هي نتيجة تنفيذ البرنامج .....

تدريب (1)(printer.asm)

باستخدام الاعتراض رقم INT 17h والماكرو اكتب برنامج بلغة التجميع اسمبلي لطباعة نص يتم ادخالة من لوحة المفاتيح على الطابعة

الحل

```
; Program to Print a given string on printer Using INT 17h in Assembly

Language

prnstr macro msg
        mov ah, 09h
        lea dx, msg
        int 21h
ENDM
hellostk  SEGMENT BYTE STACK 'STACK'    ;Define the stack segment
          DB 100h DUP(?)    ;Set maximum stack size to 256 bytes (100h)
hellostk  ENDS

data segment
        msg1 db "Enter string to be printed : $"
        msg2 db 0dh, 0ah, "I/O Error or Paper out...$"
        msg3 db 0ah, "Printing string...$"
        buf db 80
            db 0
            db 80 dup(' ')
data ends

code segment
        assume cs:code, ds:data, es:data,ss:hellostk
start :
        mov ax, data
        mov ds, ax
        mov es, ax

        prnstr msg1
```

```
        mov ah, 0ah
        lea dx, buf
        int 21h
        mov si, offset buf + 2
        mov ch, 00h
        mov cl, byte ptr [si-1]

        mov dx, 0000h
again :
        mov ah, 02h
        int 17h

        test ah, 00101001b
        jz cont
        prnstr msg2
        jmp again
cont :
        mov ah, 00h
        mov dx, 0000h
next :
        mov ah, 00h
        mov al, [si]
        int 17h
        inc si
        loop next

        mov ax, 4c00h
        int 21h
code ends
        end start
```

خطوات التنفيذ