البرمجة بلغة التجميع

Lec 7,8,9,10

تعليمات لغة التجميع:-

- يتم تحويل برنامج لغة التجميع للغة الآلة بواسطة برنامج يسمى Assembler وبالتالي يجب كتابة التعليمات بصوره محدده حتى يتعرف عليها الـ Assembler، وفى هذا الجزء سنتناول الشكل العام للأوامر المستخدمة.
- يتكون البرنامج من مجموعه من التعليمات أو الأوامر بحيث يحتوى كل سطر على أمر واحد فقط كما أن هنالك نوعين من التعليمات.
- الأوامر أو التعليمات Instructions والتي يقوم الـ Assembler بتحويلها إلي لغة الآلة والإيعازات Assembler-Directives وهى إيعازات للـ Assembler للقيام ببعض العمليات المحددة مثل تخصيص جزء من الذاكرة لمتغير محدد وتوليد برنامج فرعى.

- كل الأوامر في لغة التجميع تأخذ الصورة
- ***NAME OPERATION OPERAND(S) COMMENT***

1. **يتم الفصل بين الحقول بواسطة مفتاح الـ TAB أو المسطرة(SPACE) أي يكون هناك فراغ واحد على الأقل بين كل حقل والحقل التالي.**
2. **يتم استخدام الاسم NAME في حالة حدوث عملية تفريع لهذا الأمر (لهذا السطر من البرنامج) في جزء ما من البرنامج وهو حقل اختياري.**
3. **الحقل Operation يحتوى على الأمر المطلوب تنفيذه.**
4. **الحقل Operation(s) يحتوى على المعامل أو المعاملات المطلوب تنفيذها بواسطة الأمر المحدد ويعتمد على نوع الأمر. (لاحظ أن هناك بعض الأوامر لا تتطلب وجود هذا الحقل).**
5. **حقل الملحوظات الـ Comments يستخدم عادة للتعليق على الأمر الحالي وهو يستخدم لتوثيق البرنامج.**

- كمثال للتعليمات

```
Srart: MOV   CX , 5 ; initialize counter
```

- هذه الأمر ذو عنوان Start والأمر المستخدم MOV والمعاملات هي CX والـرقم 5 ومعنى ذلك هو وضع الـرقم 5 في المسجل CX وحقـل الملاحظـات يوضـح أن 5 هـي القيمة الابتدائية للعداد.
- ومثال للإيعازات:
-

```
Proc   Main
```

- وهـذا الإيعـاز يقـوم بتعريـف برنـامج فرعـي (إجـراء) باسـم Main. فيمـا يلـي سنتحدث عن الحقول المختلفة بالتفصيل:

NAME FIELD حقل العنوان

- يتم استخدام هذا الحقل لإعطاء عنوان لأمر محدد أو لإعطاء اسم لبرنامج فرعي كذلك لإعلان أسماء المتغيرات، يتم تحويل هذا الحقل إلي عناوين في الذاكرة.
- يمكن أن يكون هذا الحقل بطول حتى **31** حرف وغير مسموح وجود مسافات بداخل الحقل كذلك لا يستخدم الحرف "." إلا في بداية الاسم ولا يبدأ برقم ولا يتم التفريق بين الحروف الكبيرة والصغيرة فيه.
- أمثلة لأسماء مقبولة:

- **start – counter - @character – sum_of_digits - $1000 – done? -.test**

- أمثلة لأسماء غير مقبولة:
  - **two words** يحتوي علي فراغات
  - **2abc** يبدأ برقم
  - **a45.ab** يحتوي علي الحرف (.) في منتصفه

Operation Field (الأمر) حقل التعليمة

- يحتوي هذا الحقل علي الأمر OpCode المطلوب تنفيذها في هذا السطر ويجب أن تكون إحدى التعليمات المعروفة للبرنامج الذي سيقوم بمعالجة البرنامج وهو الـ Assembler حيث سيقوم بتحويلها إلي لغة الآلة كمثال لذلك التعليمات Sub و Add و Mov وكلها تعليمات معرفة وسيتم الحديث عنها بالتفصيل لاحقاً.
- أما إذا كانت إيعازاً Pseudo-Op فلا يتم تحويلها للغة الآلة ولكنها لإخطار الـ Assembler ليقوم بشيء محدد مثلاً Proc تستخدم لتعريف برنامج فرعي Procedure

Operand Field حقل المعاملات

- يحتوي هذا الحقل علي المعاملات من مسجلات ومتغيرات وثوابت والتي سيتم تنفيذ الأمر الحالي عليها (مثل عملية الجمع مثلاً) ويمكن لهذا الحقل أن يحتوي علي قيمتين أو قيمة واحدة أو لا يحتوي علي أي قيمة علي الإطلاق وذلك حسب نوع الأمر المستخدم والأمثلة التالية توضح ذلك

الأمر	المعاملات
NOP	لا توجد معاملات
INC CX	يوجد معامل واحد وهو المسجل CX
ADD Word1 , 2	يوجد معاملان وهما المتغير Word1 والرقم 2

- في حالة الحقول ذات المعاملين يكون المعامل الأول هو الذي سيتم تخزين النتيجة فيه ويسمى بالمستودع **destination Operand** وهو يكون إما أحد المسجلات أو موقع محدد في الذاكرة (لاحظ أن بعض الأوامر لا تقوم بتخزين النتيجة أصلاً) أما المعامل الثاني فيحتوي علي المصدر **Source Operand** وعادة لا يتم تغيير قيمته بعد تنفيذ الأمر الحالي.

أما بالنسبة للإيعازات فيحتوي المعامل عادة علي معلومات إضافية عن الإيعاز.

Comment Field حقل التعليقات والملاحظات

- يحتوي هذا الحقل علي ملاحظات من المبرمج وتعليقات علي الأمر الحالي وهو عادة ما يقوم بتوضيح وظيفة الأمر وأي معلومات إضافية قد تكون مفيدة لأي شخص قد يقرأ البرنامج وتساعده في فهمه. يتم بدء هذا الحقل بالفاصلة المنقوطة ";" وأي عبارة تقع بعد هذه الفاصلة المنقوطة يتم تجاهلها علي أنها ملاحظات.
- رغم أن هذا الحقل اختياري ولكن لأن لغة التجميع تحتاج التعليمات فيها لبعض الشرح فإنه من الأفضل أن يتم وضع تعليقات علي أي أمر غير واضح أو يحتاج لتفسير وعادة ما يتم وضع تعليق علي كل سطر من أسطر البرنامج ويتم اكتساب الخبرة بمرور الزمن عن كيفية وضع التعليق المناسب. فمثلاً التعليق التالي غير مناسب:

```
MOV CX , 0        ; move 0 to CX
```

- وكان من الأفضل أن يتم كتابة التعليق التالي:

```
MOV CX , 0        ; CX counts terms, initialized to 0
```

- كما يتم أحياناً استخدام سطر كامل علي أنه تعليق وذلك في حالة شرح فقرة محددة كما في المثال التالي:

```
;
; Initialize Registers
MOV CX,0
MOV     BX, 0
```

البيانات المستخدمة في البرنامج PROGRAM DATA

- يقوم البرنامج بالتعامل مع البيانات في صورة أرقام ثنائية وفي برامج لغة التجميع يتم التعامل مع الأرقام في الصورة الثنائية أو السداسية عشر أو العشرية أو حتى في صورة حروف.

الأعداد Numbers

1. يتم كتابة الأرقام الثنائية في صورة 0 و1 وتنتهي الحرف B أو b للدلالة علي أن الرقم ثنائي Binary
2. مثل 01010111B أو 11100011b
3. الأرقام العشرية يتم كتابتها في الصورة المعتادة وبدون حرف في النهاية، كما يمكن أن تنتهي بالحرف D أو الحرف d دلالة علي أنها عشرية Decimal مثل 1234 و 1345d و 234D–.
4. الأرقام السداسية عشر يجب أن تبدأ برقم وتنتهي بالحرف H أو الحرف h للدلالة علي أنها سداسية عشر Hexadecimal مثل 0abh أو 56H. (السبب في استعمال 0 في المثال الأول لتوضيح أن المطلوب هو الرقم السداسي عشر ab وليس المتغير المسمى ab).

الجدول التالي يوضح بعض الأمثلة

الرقم	ملحوظات
10011	عشري
10011b	ثنائي
6455	عشري
-456h	سداسي عشر
FFFFh	خطأ (لا يبدأ برقم)
1,234	خطأ (يحتوي على حرف غير رقمي)
0ab	خطأ (لم ينتهي بالحرف h أو H)

الحروف Characters

- يتم وضعِ الحروف والجمل داخل علامات التنصيص مثلاً 'A' أو 'SUDAN' ويتم داخلياً تحويل الحروف إلي الأرقام المناظرة في كود الـ ASCII بواسطة الـ Assembler وبالتالي تخزينها في الذاكرة وعلى ذلك لا يوجد فرق بين الحرف 'A' والرقم 41h (وهو الرقم المناظر للحرف A في الجدول) وذلك داخل البرنامج أو من ناحية التخزين في الذاكرة.

المتغيرات VARIABLES

- تلعب المتغيرات في لغة التجميع نفس الدور الذي تلعبه في البرامج باللغات ذات المستوى العالي High Level Programming Languages مثل لغة الباسكال والسي. وعلى ذلك يجب تحديد أسماء المتغيرات المستخدمة في البرنامج ونوع كل متغير حيث سيتم حجز مكان في الذاكرة لكل متغير وبطول يتناسب مع نوع المتغير وذلك بمجرد تعريف المتغير. ويتم استخدام الجدول التالي لتعريف المتغيرات في لغة التجميع حيث يشير كل إيعاز لنوع المتغير المطلوب تعريفه.

الايعـــاز	المعـــــــنـى
DB (Define Byte)	لتعريف متغير حرفي يشغل خانة واحدة في الذاكرة
DW (Define Word)	لتعريف متغير كلمة يشغل خانتين متتاليتين في الذاكرة
DD (Define Double Word)	لتعريف متغير يشغل أربعة خانات متتالية في الذاكرة
DQ (Define Quad Word)	لتعريف متغير يشغل ثمان خانات متتالية في الذاكرة
DT (Define Ten Bytes)	لتعريف متغير يشغل عشر خانات متتالية في الذاكرة

في هذا الجزء سنقوم بالتعامل مع المتغيرات من النوع DB و DW.

- **المتغيرات الحرفيــة Byte Variables:**
  - يتم تعر يف المتغيرات الحرفية بالصورة التالية:
    - **Name DB Initial_Value**
    - مثلاً
    - **Alpha DB 4**
  - يقوم هذا الإيعاز بتعريف متغير يشـغل خانــه واحـدة في الـذاكرة واسمـه **Alpha** ويتم وضع قيمه ابتدائيــة مقدارها **4** في هذا المتغير.
  - يتم استعمال علامة الاستفهام (؟) في حالة عدم وجود قيمه ابتدائية للمتغيــر.
  - مثال: **Byte DB ?**

- **متغيرات الجمل Word Variables**
- يتم تعريف المتغير علي أنه من النوع Word ويتم تخزينه في خـانتين مـن الـذاكرة Two Bytes وذلك باستخدام الصيغة
- name DW initial_value
- مثلاً التعريف التالي
- DW WRD -2
- يتم فيه تعريف متغير باسم WRD ووضع قيمة ابتدائية (الرقم –2) فيه
- كما في حالة المتغيرات الحرفية يتم وضع العلامة ؟ في حالة عدم وجود قيمة ابتدائية للمتغير.

- **الرسائل والنصوص Character Strings**
  - يتم تخزين النصوص علي أنها سلسلة من الحروف ويتم وضع القيمة الابتدائية في صورة حروف أو القيم المناظرة للحروف في جدول الحروف ASCII Table فمثلا التعريفان التاليان يؤديان إلي نفس النتيجة وهي تعريف متغير اسمه Letters ووضع القيمة الابتدائية "ABC' فيه
    - 1 - Letters db 'ABC'
    - 2 – Letters db 41h, 42h,43h
  - ويمكن دمج القيمة الابتدائية لتحوي الحروف والقيم المناظرة لها كما في المثال التالي
    - msg db 0dh,0ah,'Sudan$'
  - ويتم هنا بالطبع التفرقة بين الحروف الكبيرة Capital Letters والحروف الصغيرة Small Letters.

- **الثوابت**
- يتم عادة استخدام الثوابت لجعل البرنامج أسهل من حيث القراءة والفهم وذلك بتعريف الثوابت المختلفة المستخدمة في البرنامج. يتم استخدام الإيعاز **EQU (EQUate)** لتعريف الثوابت علي النحو التالي:
-
-
- **name EQU Constant**
- حيث **name** هو اسم الثابت. مثلاً لتعريف ثابت يسمى **LF** بقيمة ابتدائية **0Ah** نكتب
- **LF EQU 0Ah**
- وبالتالي يمكن استخدام الثابت **LF** بدلاً عن الرقم **0Ah** كالآتي **MOV AL , LF** بدلاً عن استخدام الآتي **MOV AL,0Ah**. حيث يقوم الـ **Assembler** بتحويل الثابت **LF** داخل البرنامج إلي الرقم **0Ah**.

- كذلك يمكننا استخدام المثال التالي
  - Prompt EQU ‘Type your Name’
  - Msg DB prompt
- لاحظ أن EQU عبارة عن إيعاز وليس تعليمه أو أمر وبالتالي لا ينتج عنه تعريف متغير ووضعه في الذاكرة.

بعض الأوامر الأساسية

- في هذا الجزء سنتعرف علي بعض الأوامر الأساسية وكيفية استخدامها والقيود المختلفة علي استخدامها وسنفترض أن لدينا متغيرات حرفية باسم Byte1 و Byte2 ومتغيرات كلمة باسم Word1 و Word2

1 – الأمر MOV

- يستخدم الأمر MOV في نقل البيانات من مكان لآخر وهذه الأماكن هي المسجلات العامة أو المسجلات الخاصة أو المتغيرات في الذاكرة أو حتى في نقل (وضع) قيمة ثابتة في مكان محدد من الذاكرة أو علي مسجل. والصورة العامة للأمر هي

MOV Destination , Source

- حيث يتم نقل محتويات المصدر Source إلي المستودع Destination ولا تتأثر قيمة المصدر بعد تنفيذ الأمر مثلا

MOV AX , Word1

- حيث يتم نسخ محتويات (قيمة) المتغير Word1 إلي المسجل AX. وبالطبع يتم فقد القيمة الأولية للمسجل AX بعد تنفيذ الأمر. كذلك الأمر

MOV AL, ‘A’

- يقوم بوضع الرقم 041h (وهو الرقم المناظر للحرف A في جدول الـ ASCII) في المسجل AL.

- الجدول التالي يوضح قيود استخدام الأمر MOV

المصدر	المستودع			
	مسجل عام	مسجل مقطع	متغير (موقع في الذاكرة)	ثابت
مسجل عام	مسموح	مسموح	مسموح	غير مسموح
مسجل مقطع	مسموح	غير مسموح	مسموح	غير مسموح
متغير (موقع في الذاكرة)	مسموح	مسموح	غير مسموح	غير مسموح
ثابت	مسموح	غير مسموح	مسموح	غير مسموح

الحالات المستثناة من تعليمة MOV

1-لا تستطيع تعليمة MOV أن تنقل المعطيات بشكل مباشر بين حجرتي ذاكرة لذلك لا نرى في الجدول المجاور الحالة التالية : Mem → Memو لحل هذه المشكلة فإن المعطيات المرغوب بنقلها يجب نقلها أولاً في مسجل داخلي بواسطة تعليمة MOV ، و من ثم تنقل محتويات هذا المسجل إلى حجرة جديدة في الذاكرة بواسطة تعليمة MOV أخرى.

2-لا يمكن وضع قيمة فورية في مسجل مقطع مباشرة. أي أن التعليمة التالية غير مسموح بها MOV DS,1000 و لحل هذا المشكلة نستخدم التعليمتين التاليتين :

```
MOV AX,1000
MOV DS,AX
```

3-لا يمكن نقل محتويات أحد مسجلات المقاطع إلى مسجل مقطع آخر مباشرة، أي أن التعليمة التالية غير مسموح بها MOV DS,ES و لحل هذه المشكلة نقوم بـ

```
MOV AX,ES
MOV DS,AX
```

2- الأمر XCHG (Exchange)

- يستخدم الأمر XCHG لاستبدال قيمة مسجلين أو لاستبدال قيمة مسجل مع موقع محدد في الذاكرة (متغير). والصيغة العامة للأمر هي:

XCHG Destination, Source

- **مثال:**

XCHG AH, BL

- حيث يتم تبادل قيم المسجلين AH, BL (تصبح قيمة AH تساوى قيمةBL وBL تساوى AH).

- **مثال:**

- الأمر التالي يقوم باستبدال قيمة المسجل AX مع المتغير WORD1

XCHG AX, WORD1

- الجدول التالي يوضح قيود استخدام الأمر XCHG

	المستودع	
المصدر	مسجل عام	موقع في الذاكرة
مسجل عام	مسموح	مسموح
موقع في الذاكرة	مسموح	غير مسموح

- لاحظ عدم السماح للتعليمتين **MOV** أو **XCHG** بالتعامل مع موقعين في الذاكرة في أمر واحد مثل **MOV Word1,Word2**
- ولكن يمكن تفادي هذا القيد باستخدام مسجل وسيط فيصبح الأمر كما يلي:

```
Mov   AX , Word2
Mov   Word1 , AX
```

العمليات الحسابية ADD, SUB, INC, DEC, NEG:

- يتم استخدام الأمرين ADD و SUB لجمع أو طرح محتويات مسجلين أو مسجل وموقع في الذاكرة أو موقع في الذاكرة مع مسجل أو مسجل مع موقع في الذاكرة والصيغة العامة للأمرين هي :-
  - **ADD Destination, Source**
  - **SUB** Destination, Source
- **مثلاً الأمر**
  - **ADD WORD1, AX**
- يقوم بجمع محتويات المسجل AX إلي قيمة المتغير WORD1 ويتم تخزين النتيجة في المتغير WORD1 (لا يتم تغيير قيمة محتويات المسجل AXبعد تنفيذ الأمر) كذلك الأمر
  - **SUB AX, DX**

حيث يتم طرح محتويات المسجل DX من المسجل AX ويتم تخزين النتيجة في المسجل AX (لاحظ أن محتويات المسجل DX لا تتغير بعد تنفيذ الأمر)

الجدول التالي يبين قيود استعمال الأمرين ADD و SUB

	المستودع	
المصدر	مسجل عام	موقع في الذاكرة
مسجل عام	مسموح	مسموح
موقع في الذاكرة	مسموح	غير مسموح
ثابت	مسموح	مسموح

- لاحظ أنه غير مسموح بالجمع أو الطرح المباشر بين مواقع في الذاكرة في أمر واحد وبالتالي فإن الأمر ADD BYTE1, BYTE2 غير مسموح به ولكن يمكن إعادة كتابته على الصورة:

- MOV AL, BYTE2 ; حيث يتم قيمة المتغير إلى مسجل قبل عملية الجمع
- ADD BYTE1, AL

- الأمر ADD BL,5 يقوم بجمع الرقم 5 إلى محتويات المسجل BL وتخزين النتيجة في المسجل BL.
- كملاحظة عامه نجد انه يجب أن يكون المتغيرين لهما نفس الطول بمعني أن الأمر التالي غير مقبول

- MOV AX ,BYTE1

- وذلك لأن طول المتغير BYTE هو خانه واحدة أما المسجل AX فان طوله هو خانتين 2-BYTE. (أي أن المتغيرات (المعاملات) يجب أن تكون من نفس النوع)

- بيــنما نجــد الـ ASEMBLER يستـقبل الأمر
  - MOV AH, ‘A’ (مادام AH بايت فإن المصدر يجب أن يكون كذلك بايت)
- حيث يتم وضع الرقم 41h في المسجلAH ويقوم أيضا بتقبل الأمر
  - MOV AX ,’A’ (مادام AX كلمة فإن المصدر يجب أن يكون كذلك كلمة)
- حيـث سيــتم وضــع الرقــم0041h في المسجـل AX.

الأوامر INC (Increment) , DEC (Decrement) , NEG

- أما الأمرين INC ,DEC يتم فيها زيادة أو نقصان قيمه مسجل أو موقع في الذاكرة بمقدار 1 والصيغة العامة لها هي:
  - INC Destination ; Destination = Destination +1
  - DEC Destination ; Destination = Destination - 1
- فمثلا الأمر INC WORD1 يقوم بجمع 1 إلى محتويات المتغير WORD1
- بينما الأمر DEC WORD2 يقوم بإنقاص الرقم 1 من محتويات المتغير WORD2.

- أخيراً نتحدث عن الأمر **NEG(Negate)** والذي يستعمل لتحويل إشارة الرقم الموجب إلي رقم سالب والرقم السالب يتم تحويله إلي رقم موجب وذلك بتحويله إلى المكمل لاثنين **Complement 2'S** والصيغة العامة للأمر هي:

- **NEG Destination**

- حيث يتم التعامل مع أحد المسجلات أو موقع في الذاكرة

- مثال:

- **NEG BX ; BX = -BX**
- **NEG BYTE ; BYTE = -BYTE.**

تحويل العبارات إلى صورة برامج التجميع:-

- لكي يتم التعامل مع الأوامر السابقة سنقوم في هذا الجزء بتحويل بعض العمليات من لغات البرمجة العليا High Level Programming Languages إلي تعليمات بلغة التجميع.
- إذا افترضنا أن المتغيرين A و B عبارة عن متغيرين من النوع WORD.
- لتحويل العبارة B=A
- لأنه لا يمكن نقل محتويات لمتغير في الذاكرة إلي متغير آخر في الذاكرة مباشرةً يلزم تحويل العبارة إلي نقل قيمة المتغير إلي مسجل ثم نقل قيمة المسجل إلي الرقم المطلوب
  - انقل محتويات A الي المسجل AX قبل نقلها الى B

```
MOV AX , A
MOV B , AX
```

- أما الأمر A= 5 - A يتم تحويلة إلى الأوامر
  - ضع 5 في AX
  - AX تحتوي علي 5-A
  - ضعها في A

```
MOV AX , 5
SUB AX , A
MOV A , AX
```

- أو إلى الأوامر

```
NEG A
ADD A,5
```

- وأخيراً الأمر A=B-2*A يتم تحويلة إلى الأوامر

```
MOV AX,B
SUB AX,A
SUB AX, A
MOV A,AX
```