

بسم الله الرحمن الرحيم

هذه الدروس من تأليف الاخ affef من الفريق العربي للبرمجة

<http://www.arabteam2000.com>

المعالج

المعالج عبارة عن القلب النابض للحاسوب فهو يقوم بتنفيذ الاوامر والبرامج المكتوبة بلغة الماكينة فقط. هنالك عدة شركات تنتج معالجات منها شركة انتل التي انتجت معالجات 8086 وغيرها. في هذه الدروس سوف يتم التعامل مع المعالج 8086.

الذاكرة

تستخدم الذاكرة لتخزين البيانات بها . وتقسم الذاكرة الى عدة اقسام اهمها ذاكرة القراءة فقط (ROM) بحيث يمكن القراءة منها ولا يمكن الكتابة عليها وعند انقطاع التيار الكهربائي تبقى البيانات مخزنة فيها. ملاحظة: هنالك انواع من هذه الذاكرة التي تسمح بالكتابة عليها بمساعدة اجهزة خاصة ومنها من يسمح بالكتابة عليها بمساعدة برامج خاصة.

ذاكرة الوصول العشوائي (RAM): هي ذاكرة للتخزين المؤقت وعند انقطاع التيار الكهربائي عنها تمحى البيانات الموجودة فيها.

تتكون الذاكرة من خلايا. لكل خلية عنوان (address) وفحوى وكل خلية تتكون من 8 بتات أي بطول بايت واحد ويمكن تخزين رمزا واحدا فقط في كل خلية. عنوان الخلية الاولى في الذاكرة هو صفر (0)

وحدات الذاكرة

1- **البت (bit)** : وهي اصغر وحدة تخزين بحيث تسمح بتخزين 1 أو 0 (0 , 1)

2- **النيبل (nibble)**: وهي 4 بتات

3- **بايت (byte)**: وهو عبارة عن 8 بتات ويمكن تخزين رمزا واحدا فقط . الرموز هي كل ما يمكن كتابته على لوحة المفاتيح ورؤيته على الشاشة. ولكل رمز يوجد قيمة عددية تقابله واذا اردنا معرفة قيمة رمز معين ننظر في جدول ASCII فهو يحتوي على الرموز والقيم العددية المقابلة لكل رمز

4- **كلمة (word)** : وهي عبارة عن 16 بتا (16 bits)

5-الكلمة المزدوجة (Double word) : وهي عبارة عن 32 بتا (32 bits)

6-كيلو بايت (kilo byte): وهي عبارة عن 1024 بايت (1024 byte)

7-ميغا بايت (mega byte): وهي عبارة عن 1024 كيلو بايت
وهناك وحدات اخرى ولكن لا نريد ان نتطرق اليها لانها ليست مهمة في تدريس هذا الموضوع

الناقل bus

الناقل: هو عبارة عن مجموعة من الاسلاك التي تقوم بنقل بيانات بين المعالج والذاكرة ووحدات الادخال والاخراج

وحدات الادخال: بمساعدتها يمكن ادخال البيانات للمعالج مثل لوحة المفاتيح الفارة وغيرها

وحدات الاخراج: بعد معالجة البيانات في المعالج يقوم المعالج باخراج النتائج على وحدات الاخراج ومنها الشاشة والطابعة وغيرها

1- **ناقل العناوين (Address bus):** بمساعدتها يختار المعالج خلية الذاكرة او وحدة الادخال او وحدة الاخراج المطلوبة .

2- **ناقل البيانات (Data bus):** بمساعدتها يتم ارسال او استقبال البيانات

3- **خطوط السيطرة (control lines):** تستخدم لتنظيم العمليات في الحاسوب مثل اختيار عملية القراءة او اختيار عملية الكتابة وغيرها

مكونات المعالج

1- **وحدة الحساب والمنطق Arithmetic –loic unit (ALU) :** وهي احدى مكونات المعالج وهي تقوم بجميع العمليات الحسابية (جمع ، طرح ، ضرب ، قسمة) وجميع العمليات المنطقية مثل او ، وايضا وغيرها (OR , AND) والعمليات الخاصة بالتعامل مع البتات

2- **مسجل الرايات (flags register):** هو من مكونات المعالج ويحتوي على 16 بتا وكل بت يحتوي على القيمة 0 او 1 وهذه القيمة تعطي معلومة اضافية عن العملية الاخيرة التي قام بها المعالج

3- **المسجلات (registers):** وظيفة المسجلات الحفظ المؤقت للبيانات لكل مسجل اسم ووظيفة خاصة به

ملاحظة : هنالك مكونات اخرى للمعالج مثل (command decoder) ، (instruction queue) ، (address buffer)

لغات البرمجة

يمكن تقسيم لغات البرمجة الى عدة اقسام منها

- **لغة الماكينة (machine language):** وهي اللغة الوحيدة التي يفهما المعالج وهي تتكون من (0 , 1) وتمتاز هذه اللغة بالسرعة ولكنها صعبة للبرمجة

- **لغة التجميع (assembly language):** وهي عبارة عن اختصارات خاصة يستعملها المبرمج لبرمجة المعالجات وتمتاز بالسرعة وايضا اقل صعوبة من لغة الماكينة. على المبرمج ان يعرف اقسام واسماء مكونات المعالج. لكل عائلة من المعالجات لها اختصارات خاصة بها.

- **لغات عليا (high level language):** وهي تتكون من كلمات قريبة على لغة الانسان وتمتاز بسهولة البرمجة ولكنها بطيئة بالنسبة للغة الماكينة والتجميع. مثل اللغات C , C++ , visual basic

ملاحظة: عند كتابة برنامج بلغة التجميع او بلغة عليا نحتاج الى مترجم (compiler) يقوم بترجمة البرنامج الى لغة الماكينة حتى يفهمها المعالج

لغة اسمبلي ، اسمبلر و اسمبلي

لغة اسمبلي (assembly language) : هي لغة التجميع

- **اسمبلر (assembler):** هو المترجم (compiler) وهو عبارة عن برنامج يقوم بتحويل البرامج التي تكتب بلغة التجميع الى لغة الماكينة

- **اسمبلي (assembly):** وهي عملية تحويل برنامج مكتوب بلغة التجميع بمساعدة الاسمبلر الى لغة الماكينة

الموازن

في لغة التجميع نتعامل مع 3 موازين

1-الميزان الثنائي: يتكون من الرموز 0 , 1 فقط ومن هنا الاسم ثنائي بقدر عدد الرموز التي يحق استعمالها

امثلة:

11011100 , 10001000

لكل بت توجد قيمة مثل الميزان العشري خانة الاحاد خانة العشرات خانة المئات وغيرها هنا القيم كما يلي:

البت	الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
------	--------	---------	---------	---------	---------	---------	---------	---------

القيمة	1	2	4	8	16	32	64	128
--------	---	---	---	---	----	----	----	-----

نرى ان قيمة البت في الميزان العشري تساوي 2 للقوة (مكان البت ناقص 1)
البت الموجودة في جهة اليمين هي البت ذات القيمة الصغرى والبت الموجودة في اقصى اليسار تدعى البت ذات القيمة الكبرى

2-الميزان العشري : يتكون من الرموز 0,1,2,3,4,5,6,7,8,9 اذا هنا نرى ان عدد الرموز هو 10 ومن هنا الاسم

امثلة: 123 , 234, 567

3- ميزان السادس عشر (hexa): يتكون من الرموز 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

امثلة:

1A2 , 111, DE1

قيم خانات العدد

الخانة	الاولى	الثانية	الثالثة
القيمة	1	16	256

القانون العام:

قيمة الخانة في الميزان العشري = 16 للقوة (مكان الخانة ناقص 1)
امثلة

الميزان العشري	الميزان الثنائي
136	10001000
5	101

جدول يربط بين الموازين

العشري	الثنائي	السادس عشر
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9

A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

اسئلة اكمل الجدول التالي:

الميزان الثنائي	الميزان العشري
10010	
	122

اكمل الجدول التالي:

الميزان العشري	الميزان السادس عشر
17	
	12F

كتابة الاعداد في لغة التجميع

1-الثنائي: العدد المسجل في الميزان الثنائي يكتب عن يمينه الرمز B
امثلة 110B , 10111B

2-السادس عشر: العدد المسجل في الميزان السادس عشر يكتب عن يمينه الرمز H
امثلة: 10H , 12AH , 11H

3-العشري: يكتب عن يمينه الرمز D او يترك بدون اضافات
امثلة: 100D , 102 , 11D

ملاحظة: اذا كانت هنالك حاجة لشرح طرق التحويل بين الموازين فاننا على استعداد لشرحها
الى اللقاء في الدرس القادم
الدرس القادم باسم المعالج 8086

المعالج 8086

- لهذا المعالج 16 ناقل بيانات (data bus) بحيث يسمح بقراءة/كتابة كلمة كاملة 16 بت وايضا يمكن قراءة/كتابة نصف كلمة 8 بت

- لهذا المعالج 20 ناقل عناوين (address bus) بحيث يسمح بالتوجه ل- 1 ميغا عناوين. وبما ان لكل خلية عنوان خاص اذا يمكن التوجه الى 1 ميغا خلايا ذاكرة

- يوجد في هذا المعالج 14 مسجلا. كل مسجل بطول 16 بت.

المسجلات

تقسم المسجلات الى مجموعات حسب الوظائف

المجموعة الاولى: المسجلات العامة وتستخدم هذه المسجلات في العمليات الحسابية والمنطقية والتخزين المؤقت للبيانات. تتكون هذه المجموعة من 4 مسجلات. طول كل مسجل 16 بت.

يمكن تقسيم كل مسجل منها الى قسمين بحيث يكون كل جزء بطول 8 بت. يمكن التوجه الى المسجل بأكمله او الى جزء من المسجل.

في حالة التوجه الى المسجل بأكمله فان المسجل بأكمله يتأثر واذا توجهنا الى جزء واحد منه فهو الذي يتأثر ولكن الجزء الاخر لا يتأثر.

المسجل الاول يدعى ACCUMULATOR ويرمز له في AX . يقسم هذا المسجل الى قسمين AH و AL. كل قسم بطول 8 بت

AH	AL
----	----

مثال : نفرض ان $AX=2030H$

اذا $AH=20H$ و $AL=30H$

لو عوضنا $50H$ في المسجل AL نحصل على $AX=2050H$ أي ان التأثير فقط على AL

ولكن لو عوضنا $402H$ في AX

نحصل على $AL=02H$ و $AH=04H$

هنا نرى ان التأثير كان على الجزئين لاننا توجهنا الى AX

هذه مسجلات المجموعة الاولى:

Accumulator : AX

Base : BX

Count : CX

Data : DX

ويقسم كل مسجل الى جزئين كالتالي:

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

المجموعة الثانية:

هي مجموعة تتكون من خمسة مسجلات وطول كل مسجل 16 بت . لا يمكن تقسيم المسجلات الى اجزاء بل نتعامل مع المسجل بأكمله والسبب ان هذه المسجلات تحتوي على عناوين وطول كل عنوان 16 بت

SP
Bp
SI
DI
IP

Sp: Stack pointer يؤشر على اخر معطى تم ادخاله الى ال **stack** . **stack** هو عبارة عن منطقة في ذاكرة القراءة والكتابة RAM تستخدم للتخزين المؤقت سوف نتطرق لها في دروس قادمة.

BP: Base pointer يستخدم للتأشير على معطى في داخل ال **stack**
SI: Source index يحتوي على عنوان معطى في مقطع البيانات

DI: Destination Index يحتوي على عنوان معين في مقطع البيانات

IP: Instruction Pointer يحتوي على عنوان الامر التالي الذي يجب ان ينفذه المعالج بعد انتهاء الامر الحالي أي الامر الذي ينفذه حالياً

ملاحظات:

- لا يمكن تقسيم أي من هذه المسجلات

- المقطع هو عبارة عن مكان في الذاكرة طوله الاكبر هو 64 كيلوبايت. يمكن ان يحتوي البرنامج على 4 مقاطع ويمكن اقل حسب الحاجة. المقاطع هي كما يلي:

1- مقطع ال **stack segment**

2- مقطع البيانات **Data segment** يحتوي على المتغيرات والثوابت

3- مقطع البيانات الاضافية **extra segment** يستخدم لتخزين البيانات الاضافية في حالة اذا لم يكن هنالك مكانا في مقطع البيانات ويمكن ايضا استخدامه حتى ولو كان هنالك اماكن فارغة في مقطع البيانات. سوف نرى ذلك في الدروس المقبلة

4- مقطع الاوامر **code segment** يحتوي على اوامر البرنامج

المجموعة الثالثة:

مجموعة مسجلات المقاطع وهي تحتوي على عناوين بداية المقاطع وطول كل مسجل 16 بت ولا يمكن تقسيمه أي يجب التعامل معه بأكمله. وهي 4 مسجلات (**registers**)

CS
DS
SS
ES

CS: code segment **DS**: data segment **ES**: extra segment **SS**: stack segment

المجموعة الأخيرة:

تحتوي على مسجل واحد وهو مسجل الرايات **flags registers**. وهو بطول 16 بت يمكن التوجه لكل بت على حدة. كل بت تعطي معلومة معينة عن اخر عملية قامت بها وحدة الحساب والمنطق في المعالج.

على كل مبرمج ان يفهم هذا المسجل بشكل جيد حتى يستطيع ان يبرمج بشكل جيد. تقسم الرايات الى ثلاثة اقسام:

- **رايات غير مستعملة:** أي انها لا تقيد في الحكم على اخر عملية وهي موجودة فقط في حالة تطوير المعالج ربما يحتاجون الى رايات اضافية فيمكن استغلال هذه الرايات. في الوقت الحالي نحن لسنا بحاجة اليها

- **رايات الوضع:** وهي الرايات التي تتأثر وتتغير حسب وضع العمليات التي تقوم بها وحدة الحساب والمنطق في المعالج

- **رايات السيطرة:** وهي رايات المبرمج يتحكم بوضعها فاذا وضع بداخلها القيمة 1 تبقى هذه القيمة حتى يغيرها المبرمج في البرنامج عن طريق اوامر برمجة خاصة بها

مسجل الرايات في المعالج 8086 يحتوي على 7 رايات غير مستخدمة 6 رايات وضع و 3 رايات سيطرة

X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF
---	---	---	---	----	----	----	----	----	----	---	----	---	----	---	----

كل راية تحتوي على الاشارة X فهي غير مستخدمة

الرمز	اسم الراية	النوع	ملاحظات
CF	Carry flag	وضع	يأخذ 1 اذا اعطت البت الاخيرة عن اليسار 1 الى الخارج او اخذت 1 من الخارج والا يأخذ صفر
PF	Parity flag	وضع	اذا كان عدد ال 1 في اول 8 بتات من النتيجة زوجي تأخذ هذه الراية 1 والا تأخذ صفر
AF	Auxiliary flag	وضع	اذا اعطت البت الرابعة البت الخامس 1 او اخذت البت الرابعة من البت الخامسة 1 تأخذ هذه الراية 1 والا صفر
ZF	Zero flag	وضع	اذا كانت النتيجة صفر تأخذ هذه الراية 1 والا تأخذ صفر
SF	Sign flag	وضع	اذا كانت النتيجة سالبة تأخذ هذه الراية القيمة 1 والا تأخذ القيمة صفر
TF	Trap flag	سيطرة	اذا وضع المبرمج بها القيمة 1 يتم تنفيذ امر واحد فقط من الاوامر وبعدها تأخذ القيمة صفر
IF	Interrupt flag	سيطرة	نضعه 1 للسماح لعمليات interrupt في مدخل interrupt للمعالج
DF	Direction Flag	سيطرة	لتحديد اتجاه الحركة في حالة قراءة النصوص
OF	Overflow flag	وضع	تأخذ 1 اذا كانت النتيجة اكبر من المكان المخصص لحفظها والا تأخذ صفر

الموازين

هنالك العديد من الموازين ولكن ما يهمنا هو

1-الميزان الثنائي: رموز هذا الميزان 0 , 1 وبمساعدة هذه الرموز يتم تكوين الاعداد في هذا الميزان

مثلا: هذه اربعة اعداد في الميزان الثنائي

1001	0101	1100110	101010
------	------	---------	--------

الاعداد التالية غير صحيحة لان بها رموز لا تنتمي لرموز الميزان

العدد	السبب
102	لان الرمز 2 ليس من رموز الميزان
11013	بسبب الرمز 3 فهذا الرمز لا يتبع لرموز الميزان

2-الميزان العشري : رموز هذا الميزان 0,1,2,3,4,5,6,7,8,9 وكما نلاحظ عددها 10 رموز
امثلة: 201 , 100 , 532 , 1234

الاعداد التالية غير صحيحة لان بها رموز لا تنتمي لرموز الميزان

العدد	السبب
10A2	لان الرمز A ليس من رموز الميزان
11013F	بسبب الرمز F فهذا الرمز لا يتبع لرموز الميزان

3-الميزان السادس عشر: رموز هذا الميزان 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
امثلة:

101 , 217 , 2A3 , 5FA1 , FF

الاعداد التالية غير صحيحة لان بها رموز لا تنتمي لرموز الميزان

العدد	السبب
102G	لان الرمز G ليس من رموز الميزان
1101K3	بسبب الرمز K فهذا الرمز لا يتبع لرموز الميزان

التحويل بين الموازين

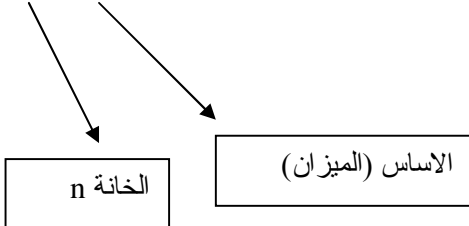
لتحويل أي عدد في أي ميزان الى الميزان العشري نعمل ما يلي:

1-نحول العدد المراد تحويله الى صورة مجموع القوى

2-نجمع القيم للحصول على القيمة المطلوبة

القانون العام لصورة قانون مجموع القوى

$$X = a_n \times b^{n-1} + a_{n-1} \times b^{n-2} + \dots + a_1 \times b^0$$



ملاحظة: أي عدد للقوة صفر = واحد

ما عدا الصفر

لذلك $b^0 = 1$

مثال: حول العدد (1101) في الميزان الثنائي الى الميزان العشري

الحل: هنا $b=2$ $n=4$

$$X=1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

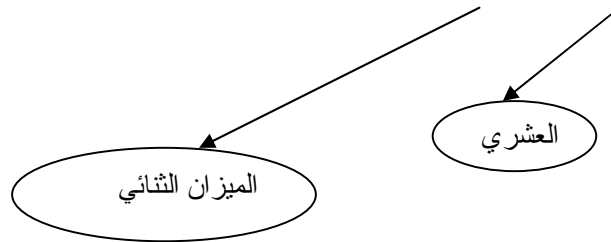
$$X=1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$

$$X=8+4+0+1$$

$$X=13$$

إذا

$$(1101)_2 = (13)_{10}$$



مثال اخر معطى العدد 1A في الميزان السادس عشر، جد قيمة هذا العدد في الميزان العشري.

$b=16$

هنا: $n=2$

$$X=1 \times 16^1 + A \times 16^0$$

$$X=1 \times 16 + 10 \times 1$$

$$X=16+10$$

$$X=26$$

ملاحظة:

العشري	السادس عشر
10	A
11	B
12	C
13	D
14	E
15	F

مثال اخر:

حول العدد 2F الى الميزان العشري

$$X=2 \times 16^1 + F \times 16^0$$

$$X=2 \times 16 + 15 \times 1$$

$$X=32+15$$

$$X=47$$

$$(2F)_{16}=(47)_{10}$$

التحويل من الميزان 16 الى الميزان 2

كل خانة في ميزان السادس عشر عبارة عن 4 خانات في الميزان الثنائي لفهم عملية التحويل يجب ان نعرف ما يلي:
لكل خانة يوجد وزن ووزنها هو الاساس للقوة المكان ناقص واحد أي b^{n-1}

في الجدول التالي قيم الخانات في الميزان الثنائي $b=2$

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
1	2	4	8	16	32	64	128

في الجدول التالي قيم الخانات في الميزان العشري $b=10$

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
1	10	100	1000	10000	10^5	10^6	10^7

في الجدول التالي قيم الخانات في الميزان السادس عشر $b=16$

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
1	16	256	4096	16^4	16^5	16^6	16^7

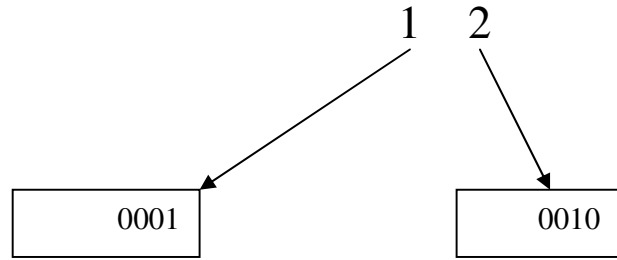
الجدول التالي يربط بين الميزان السادس عشر والثنائي

تذكر كل خانة في الميزان السادس عشر عبارة عن 4 خانات في الميزان الثنائي

العشري	الميزان الثنائي				السادس عشر
	الخانة الاولى	الخانة الثانية	الخانة الثالثة	الخانة الرابعة	
	1	2	4	8	
0	0	0	0	0	0
1	1	0	0	0	1
2	0	1	0	0	2
3	1	1	0	0	3
4	0	0	1	0	4
5	1	0	1	0	5
6	0	1	1	0	6
7	1	1	1	0	7

8	1	0	0	0	8
9	1	0	0	1	9
A	1	0	1	0	10
B	1	0	1	1	11
C	1	1	0	0	12
D	1	1	0	1	13
E	1	1	1	0	14
F	1	1	1	1	15

معطى العدد 12 في الميزان السادس عشر جد قيمته في الميزان الثنائي



إذا الجواب النهائي هو 00010010

$$(12)_{16} = (00010010)_2$$

الجدول السابق يصلح للتحويل من السادس عشر الى الميزان الثنائي وبالعكس مع مراعاة الملاحظة الاولى الا وهي ان كل 4 خانات في الميزان الثنائي عبارة عن خانة في الميزان السادس عشر

امثلة:

معطى العدد 10110011 في الميزان الثنائي جد العدد في الميزان السادس عشر

الحل:

نعمل مجموعة كل مجموعة عبارة عن 4 خانات نبدأ دائما من اليمين

المجموعة الاولى: 0011 وحسب الجدول هي الرقم 3

المجموعة الثانية: 1011 وحسب الجدول هي B

الحل النهائي:

$$(10110011)_2 = (B3)_{16}$$

ملاحظة مهمة: إذا كان عدد الخانات اقل من 4 نضيف اصفار على يسار العدد حتى نكمل المجموعة الى 4 خانات

مثال:

معطى العدد التالي في الثنائي 1010110 جد العدد في السادس عشر

المجموعة الاولى: 0110 وحسب الجدول هذه 6
المجموعة الثانية : 101 وهي تتكون من 3 خانات لذا علينا ان نضيف 0 على يسار المجموعة
لنحصل على 4 خانات
المجموعة الجديدة: 0101 وحسب الجدول هي 5
الحل النهائي
 $(1010110)_2 = (53)_{16}$

التحويل من العشري الى الميزان الثنائي

لفهم الطريقة علينا الاتفاق على الاشياء التالية:

- الخانة الاولى هي الخانة اليمنى وهي ذات القيمة الصغرى وقيمتها $2^0 = 1$
الخانة التي تليها هي الخانة الثانية وهي تقع على يسارها وهكذا

- عملية التحويل عن طريق القسمة نقسم العدد العشري على 2 والخانة التي عليها الدور تأخذ
باقي القسمة ونقسم الناتج الصحيح مرة ثانية على 2 وتأخذ الخانة التالية باقي القسمة نستمر في
هذه الطريقة حتى يكون المقسوم اقل من المقسوم عليه فتأخذ الخانة الحالية قيمة المقسوم وينتهي
التحويل.
نقوم الان بكتابة الالغوريثم

1. عوض $n=1$
2. عوض المقسوم عليه $b=2$
3. عوض العدد العشري في المقسوم x
4. طالما x اكبر او يساوي b نفذ
قسم x على b
ضع الباقي في الخانة n
ضع في x الناتج الصحيح
اضف 1 لرقم الخانة n
5. ضع X في الخانة n

مثال:

معطى العدد 12 عشري حول العدد الى الميزان الثنائي
الحل:

العدد	الناتج الصحيح	الباقى	هل $X \geq b$	العدد الثنائي	رقم الخانة n
					1
12			نعم		
6	6	0		0	2
6			نعم		
3	3	0		00	3
3			نعم		
1	1	1		100	4
1			لا		
				1100	

طريقة اخرى نقسم على 2 حتى يكون الناتج صفر
مثال:

معطى العدد العشري 57 حول العدد الى الثنائي

المقسوم	المقسوم عليه	الناتج الصحيح	الباقى
57	2	28	1
28	2	14	0
14	2	7	0
7	2	3	1
3	2	1	1
1	2	0	1

الان نكتب العدد نبدأ من الاسفل

111001

ملاحظة اذا اردنا تحويل عدد عشري الى الميزان السادس عشر نعمل نفس الشيء ولكن بدلا

من $b=2$ يكون $b=16$

مع مراعاة الجدول السابق

A=10 b=11 C=12 D=13 E=14 F=15

مثال :

حول العدد 254 الى الميزان السادس عشر

المقسوم	المقسوم عليه	الناتج الصحيح	الباقى
254	16	15	14
15	16	0	15

عند تسجيل العدد نتذكر الجدول $E=14$ $F=15$
 الجواب النهائي
 $(254)_{10}=(FE)_{16}$

طريقة اخرى هي الانتقال من الميزان العشري الى الثنائي وبعدها من الثنائي الى السادس عشر

طريقة اخرى لتحويل العدد من العشري الى الثنائي هي كما يلي:

نحن نعرف ان لكل خانة قيمة حسب قانون b^{n-1}

اذا نظرنا بتمعن نجد ان قيمة الخانة التالية هي دائما قيمة الخانة الحالية مضروبة بالعدد 2
 عند تحويل عدد عشري نبدأ بتسجيل القيم حتى الحصول على قيمة خانة تساوي العدد المطلوب
 او اكبر منه مثال:

حول العدد 104 الى الميزان الثنائي

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
1	2	4	8	16	32	64	128

نحن هنا بحاجة الى 8 خانات لان 104 اكبر من 64 واصغر من 128
 الحل الان كما يلي:

نبدأ من اليسار ونسأل السؤال هل قيمة الخانة اقل او تساوي العدد الذي معنا اذا كان الجواب نعم
 نضع القيمة 1 تحت الخانة ونطرح من العدد قيمة الخانة وننتقل الى الخانة التي عن يمينها والا
 نضع 0 تحت الخانة ولا نطرح شيء وننتقل الى الخانة التي عن يمينها وهكذا

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
1	2	4	8	16	32	64	128
0	0	0	1	0	1	1	0

-هل 128 اصغر او تساوي 104 الجواب كلا اذا نضع صفر

-هل 64 اصغر او تساوي 104 الجواب نعم اذا نضع 1 ونطرح 64 يبقى معنا 40

- هل 32 اصغر او تساوي 40 الجواب نعم اذا نضع 1 ونطرح 32 يبقى معنا 8

-هل 16 اصغر او تساوي 8 الجواب لا اذا نضع صفر

-هل 8 اصغر او تساوي 8 الجواب نعم اذا نضع 1 ونطرح 8 يبقى معنا صفر

-هل 4 اصغر او تساوي 0 الجواب كلا اذا نضع صفر

-هل 2 اصغر او تساوي 0 الجواب كلا اذا نضع صفر

-هل 1 اصغر او تساوي 0 الجواب كلا اذا نضع صفر

الحل النهائي

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الثامنة
1	2	4	8	16	32	64	128
0	0	0	1	0	1	1	0

الحل النهائي

$$(104)_{10} = (01101000)_2$$

ملاحظة: جميع الطرق التي تم شرحها في هذا الدرس تتاسب الاعداد الموجبة فقط في الدرس القادم سوف نتطرق الى الاعداد الموجبة (السالبة والموجبة) بالاخص الاعداد السالبة

الاعداد الموجبة (السالبة والموجبة)

نقسم الاعداد الى قسمين اعداد موجبة واعداد سالبة والسؤال هو كيف يمكن ان نفرق بين هذه الاعداد.

من المعروف ان العدد في الميزان الثنائي يتكون من بتات . البت الاولى الموجودة في الجهة اليمنى هي البت ذات القيمة الصغرى واما البت الموجودة في الجهة اليسرى فهي بت الاشارة . اذا كانت قيمتها 1 فالعدد سالب واذا كانت قيمتها 0 فالعدد موجب .

اذا لمعرفة نوع العدد يجب تحويله الى الميزان الثنائي . ولكن هنالك طرق اخرى يمكن بمساعدتها معرفة نوع العدد . في البداية نركز على الطريقة الاولى .

مثال: معطى المتغير X من نوع byte . $x = 7DH$ هل قيمة المتغير قيمة موجبة ام سالبة؟ الرمز H يدل على الميزان السادس عشر

الحل :

اولا علينا ان نحول العدد الى الميزان الثنائي

$$7DH = 01111101_2$$

اذا نظرنا الى البت الاخيرة في الجهة اليسرى نرى ان قيمتها 0 لذلك قيمة المتغير موجبة .

مثال اخر:

$$X = 92H$$

$$92H = 10010010$$

استنتاج: اكبر قيمة موجبة للمتغير x هي 01111111 وهي تساوي 7FH وهي تساوي 127 عشري

الاعداد السالبة

معطى ان المتغير $x = -6$ في الميزان العشري سجل العدد في الميزان الثنائي والميزان السادس عشر

الحل:

حسب المعطيات فان المتغير x من نوع byte اذا سوف نستخدم 8 بتات

الخطوة الاولى: نكتب القيمة المطلقة (العدد الموجب) للمتغير في الميزان الثنائي حسب ما تعلمناه في الدروس السابقة:

الاولى	الثانية	الثالثة	الرابعة	الخامسة	السادسة	السابعة	الاشارة
1	2	4	8	16	32	64	
0	1	1	0	0	0	0	0

اذا

$$(6)_{10} = (00000110)_2$$

الخطوة الثانية: نبدأ بكتابة العدد من جهة اليمين كما هو حتى ان نصادف خانة قيمتها واحد فنسجلها ونعكس كل الخانات التي بعدها (أي نسجل بدلا من 1 صفر وبدلا من صفر 1)

$$(-6)_{10} = 11111010_2 = (FA)_{16}$$

طريقة اخرى هي كما يلي:

-اولاً: نحول القيمة المطلق للقيمة السالبة في الميزان السادس عشر

$$6_{10} = (06)_{16}$$

لقد سجلنا 06 لاننا بحاجة الى 8 بتات

- الخطوة الثانية: نطرح 06 من الصفر والسبب هو

$$-6 = 0 - 6$$

اذا:

00

-

06

0 ناقص 6 لا يجوز تأخذ الصفر الاولى 16 من الصفر الثانية 16 ناقص 6 تساوي 10 عشري وهي A في الميزان السادس عشر

الصفر الثانية اعطت الصفر الاولى لهذا السبب هي الان 1- فتأخذ 16 الان 1-16 يبقى معها 15 الان -15

0=15 وهي تساوي F في الميزان السادس عشر

الجواب النهائي

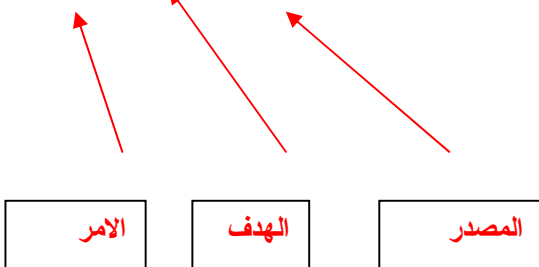
$$-6 = (FA)_{16} = (11111010)_2$$

الوامر

MOV الامر (1)

الصيغة الاولى: هنالك مجموعة من الاوامر التي بحاجة الى مصدر وهدف. المصدر لا يتغير وانما الهدف الذي يتغير

MOV OP1,OP2



الامر mov يعني ضع في op1 نسخة من قيمة op2 بدون التأثير على OP2 . بعد هذه العملية op2 لا يتغير بينما OP1 يتغير

مثال:

معطى AL=20H و BL=50H

MOV AL,BL

MOV BL,60H

نعمل جدول متابعة لمشاهدة التغييرات

الامر	AL	BL
	20H	50h
MOV AL,BL	50H	50H
MOV BL,60H	50H	60H

بعد تنفيذ المقطع نحصل على

AL=50H

BL=60H

ملاحظات

(1) الامر MOV لا يؤثر على الرايات flags أي ان قيم الرايات لا يتغير وانما تحافظ على القيم السابقة

(2) يجب ان يكون المصدر من نفس كبر الهدف (بايت، بايت) او (word , word)

مثال:

MOV AX ,CX

MOV DL,AL

MOV CH,20H

اما هذه الامثلة بعضها غير صحيح

الامر	الوضع	السبب
MOV AL,BX	غير صحيح	لان المصدر اكبر من الهدف
MOV AL,123H	غير صحيح	لا يمكن تخزين عدد من ثلاث منازل في الميزان 16 في مسجل بكبر 8 بتات. يمكن تخزين قيمة بكبر منزلة او منزلتين
MOV BL,120	صحيح	لان $120=78H$ أي انه يتكون من منزلتين
MOV DX,12345H	غير صحيح	لانه في مسجل بكبر 16 بت يمكن تخزين قيمة بكبر 4 منازل على الاكثر في الميزان السادس عشر

تمارين:

1) اكتب مقطع برنامج يبدل قيم CL ، DL فيما بينها. أي ان DL يأخذ قيمة CL و CL يأخذ قيمة DL

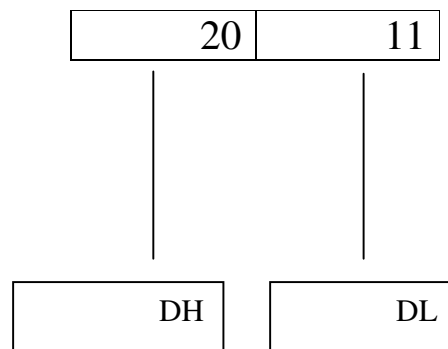
الحل: هنا يجب ان نستعين بمسجل مساعد

```
MOV AL,DL
MOV DL,CL
MOV CL,AL
```

2) معطى الامر التالي:

```
MOV DX , 2011H
```

ما هي القيم الموجودة في كل من DH ، DL



2) الامر ADD

ADD OP1, OP2
Op1=op1+op2

اجمع قيمة المصدر وقيمة الهدف واحفظ النتيجة في الهدف

مثال:

```
Mov dh,41H
Add dh,3
```

بعد تنفيذ المقطع المسجل dh يصبح 44H

مثال :

```
Mov AL , 6
MOV CL , 8
ADD AL , CL
```

جدول متابعة:

الامر	CL	AL
	??	??
Mov AL , 6	??	06H
MOV CL , 8	08H	06H
ADD AL , CL	08H	0EH

ملاحظات:

- ?? : تعني اننا لا نعلم ما بداخله
- علينا تسجيل القيم في الميزان السادس عشر على شكل منزلتين في المسجلات ذات 8 بت وعلى شكل 4 منازل في المسجلات ذات 16 بت.

مثال:

DX=1024H AL=06H
- كل قيمة عددية تبدأ بحرف A , B ,C ,D ,E ,F يجب ان نسجل 0 عن يسارها

مثال:

```
Mov DL , 0D1H
ADD CX , 0A123H
```

تمرين : اكتب مقطع برنامج ينفذ ما يلي: $CL=2*AL+BL$

الحل:

```
Mov CL,AL
ADD CL ,CL
ADD CL,BL
```

(3) الامر SUB

```
sub op1 ,op2
op1=op1-op2
```

اطرح المصدر من الهدف وخزن النتيجة في الهدف
تمرين:

اكتب مقطع برنامج يحسب حاصل الفرق بين DL والمسجل BL ويحفظ النتيجة في BH
الحل:

```
Mov BH , DL  
Sub BH , BL
```

4) الامر INC

هذا الامر بحاجة الى operand واحد فقط

```
Inc op1  
Op1=op1+1
```

كبر الهدف بـ 1

5) الامر DEC

هذا الامر بحاجة الى operand واحد فقط

```
Dec op1  
Op1=op1-1
```

اطرح من OP1 واحد

التعامل مع الذاكرة

يمكن القراءة من الذاكرة او الكتابة الى الذاكرة . عندما نريد التوجه الى الذاكرة للكتابة او القراءة علينا ان نقوم بما يلي:

- ضع في المسجل SI او DI او BX عنوان الخلية المطلوبة
- ضع القيم العددية التي تريد كتابتها في مسجل ملائم من حيث الكبر
- استخدم الامر MOV وضع المسجل الذي يحوي عنوان الخلية بين اقواس مربعة

مثال:

اكتب مقطع برنامج يخزن في الخلية 102H القيمة 30H
الحل:

```
Mov si,102H  
Mov AL, 30H  
Mov [SI],AL
```

في حال كتبنا بدلا من MOV [si],AL الامر MOV si,AL نحصل على خطأ لان المترجم يفهم باننا نريد ان نخزن قيمة AL في المسجل SI ونحن نعلم ان المسجل si 16 بت والمسجل AL 8 بت اذا لا توجد ملائمة بين المصدر والهدف

بينما MOV [si],AL ضع قيمة AL في الخلية التي عنوانها مخزن في المسجل si
ملاحظة مهمة جدا: لا يجوز ان يكون الهدف خلية في الذاكرة والمصدر ايضا خلية في الذاكرة
مثال: X اسم متغير

الامثلة التالية غير صحيحة

Mov [si],[bx]

add [di],X

تمرين : اكتب مقطع برنامج يقرأ فحوى الخلية 440H ويحفظها في المسجل DH
الحل:

```
Mov DI, 440H  
Mov DH, [DI]
```

تمرين: اكتب مقطع برنامج يجمع فحوى الخلية 10H وفحوى الخلية 11H ويخزن النتيجة في الخلية 12H
الحل:

```
Mov si,10H  
Mov ah,[si]  
Inc si  
Add ah,[si]  
Inc si  
Mov [si],ah
```

تمرين مهم جدا:

معطى المقطع التالي:

```
Mov si,5  
Mov ax,1122H  
Mov [si],ax
```

ماذا يعمل هذا المقطع
نحن نعرف ان كبر كل خلية في الذاكرة هو 8 بت اذا يمكن تسجيل قيمة تتكون من منزلتين في الميزان السادس عشر.

ولكن AX عبارة عن 16 بت أي لا يمكن تخزينها كلها في خلية واحدة لهذا السبب
يتم حفظ قيمة AL في الخلية التي يؤشر عليها SI ويحفظ قيمة ah في الخلية التي تليها.
اذا

في الخلية 5 يسجل 22H وفي الخلية 6 تسجل القيمة 11H

المقارنة

في بعض الاحيان نحتاج الى مقارنة بين المصدر والهدف . مثلا افحص اذا كان المسجل AL يحتوي على القيمة 6 اذا كان الجواب نعم اضع القيمة 4 للمسجل والا اطرح 3 من قيمة المسجل. من الصيغة السابقة يتبين لنا اننا بحاجة لامر ليقارن بين فحوى المسجل AL والقيمة العددية 6. الامر هو امر المقارنة CMP

ملاحظات هامة بالنسبة لامر المقارنة CMP:

1

(هذا الامر بحاجة الى مصدر وهدف CMP OP1,OP2

2) هذا الامر يطرح المصدر من الهدف ولكنه لا يؤثر على الهدف وانما يرسل معلومات الى مسجل الرايات ويغير قيمة الرايات (رايات الوضع) بحيث تتلائم قيم الرايات مع النتيجة

OP1-OP2 → flags

3) بعد عملية المقارنة سوف تكون النتيجة حالة من الحالات التالية:

الحالة	النتيجة	الاستنتاج
الاولى	صفر	المصدر والهدف متساويان
الثانية	موجبة	الهدف اكبر من المصدر
الثالثة	سالبة	الهدف اصغر من المصدر

للتذكرة

مسجل الرايات في المعالج 8086 يحتوي على 7 رايات غير مستخدمة 6 رايات وضع و 3 رايات سيطرة

X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF
---	---	---	---	----	----	----	----	----	----	---	----	---	----	---	----

كل راية تحتوي على الاشارة X فهي غير مستخدمة

الرمز	اسم الراية	النوع	ملاحظات
CF	Carry flag	وضع	يأخذ 1 اذا اعطت البت الاخيرة عن اليسار 1 الى الخارج او اخذت 1 من الخارج والا يأخذ صفر
PF	Parity flag	وضع	اذا كان عدد ال 1 في اول 8 بتات من النتيجة زوجي تأخذ هذه الراية 1 والا تأخذ صفر
AF	Auxiliary flag	وضع	اذا اعطت البت الرابعة البت الخامس 1 او اخذت البت الرابعة من البت الخامسة 1 تأخذ هذه الراية 1 والا صفر
ZF	Zero flag	وضع	اذا كانت النتيجة صفر تأخذ هذه الراية 1 والا تأخذ صفر
SF	Sign flag	وضع	اذا كانت النتيجة سالبة تأخذ هذه الراية القيمة 1 والا تأخذ القيمة صفر
TF	Trap flag	سيطرة	اذا وضع المبرمج بها القيمة 1 يتم تنفيذ امر واحد فقط من الاوامر وبعدها تأخذ القيمة صفر
IF	Interrupt flag	سيطرة	نضعه 1 للسماح لعمليات interrupt في مدخل interrupt للمعالج
DF	Direction Flag	سيطرة	لتحديد اتجاه الحركة في حالة قراءة النصوص
OF	Overflow flag	وضع	تأخذ 1 اذا كانت النتيجة اكبر من المكان المخصص لحفظها والا تأخذ صفر

معلومات مؤكدة عن الرايات

(1) عندما تكون النتيجة صفر تكون الراية $SF=0$, $ZF=1$
لانه عندما تكون النتيجة صفر لن تكون النتيجة سالبة (الصفر ليس عدد سالب)

(2) عندما تكون النتيجة سالبة تكون الراية $SF=1$, $ZF=0$
لانه عندما تكون النتيجة سالبة لا يمكن ان تكون صفر لانها عدد اصغر من صفر

(3) عندما تكون النتيجة موجبة تكون الراية $SF=0$, $ZF=0$
لانه عندما تكون النتيجة موجبة فهي اكبر من صفر وايضا اكبر من الاعداد السالبة
من المعلومات السابقة نستنتج

-اذا اردنا ان نعرف اذا كان المصدر (OP2) والهدف (OP1) متساويان علينا ان نسأل عن راية الصفر اذا كانت $ZF=1$. اذا كانت الاجابة نعم فهما متساويان واذا كانت الاجابة كلا فهما مختلفان
-اذا اردنا ان نعرف اذا كان الهدف (OP1) اصغر من المصدر (OP2) نسأل عن راية الاشارة $SF=1$ اذا كانت الاجابة نعم نعرف ان الهدف اصغر من المصدر
-اذا اردنا ان نعرف اذا كان الهدف (OP1) اكبر من المصدر (OP2) نسأل اذا كان $ZF=0$ وايضا $SF=0$
طريقة افضل نسأل عن الحالة $ZF=1$ اذا لم يتحقق نسأل عن الحالة الثانية $SF=1$ اذا لم يتحقق اذا بدون شك الهدف اكبر من المصدر

القفز

نحن نعلم ان البرنامج عبارة عن اوامر مخزنة في خلايا الذاكرة والمعالج ينفذ اوامر البرنامج بالترتيب مثال:

```
MOV AL,2  
ADD AL, BL  
INC DI
```

فهو ينفذ الامر الاول MOV
ثم ينتقل الى الامر الثاني ADD
ثم ينتقل الى الامر الثالث INC ويستمر هكذا الى نهاية البرنامج
فاذا اردنا ان نغير ترتيب التنفيذ او ان لا ننفذ بعض الاوامر فلا بد من استخدام اوامر القفز.
القفز في لغة الاسبلي يقسم الى قسمين من حيث الشروط

1- القفز المشروط: هنا تتم عملية القفز في حالة اذا تحقق الشرط المطلوب . مثلا اقفز اذا كانت النتيجة سالبة. او اقفز اذا كانت النتيجة صفر

2- القفز غير المشروط: هنا يتم القفز بدون أي شرط بمجرد وصول المعالج الى هذا الامر يتم القفز

اوامر القفز المشروط كثيرة نأخذ في كل مناسبة جزء منها

الامر	المعنى	الراية المطلوبة
JZ	اقفز اذا كانت النتيجة صفر	ZF=1
JNZ	اقفز اذا كانت النتيجة لا تساوي صفر. (النتيجة موجبة او سالبة)	ZF=0
JE	اقفز اذا كانت النتيجة صفر	ZF=1
JNE	اقفز اذا كانت النتيجة لا تساوي صفر. (النتيجة موجبة او سالبة)	ZF=0
JC	اقفز اذا البت الاخيرة من جهة اليسار اخرجت 1 للخارج او احتاجت لواحد من الخارج	CF=1
JNC	اقفز اذا البت الاخيرة من جهة اليسار لم تخرج 1 للخارج او لم تحتاج لواحد من الخارج	CF=0

ملاحظات :

- الامر JE يعمل مثل JZ نفس الشيء

- الامر JNE يعمل مثل JNZ

تذكر: جدول الجمع في الميزان الثنائي:

a,b: عبارة عن بتات

S: حاصل جمع البت a والبت b

C: carry انه هنالك 1 سوف ينتقل الى البت التي تليها

a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

امثلة: جد النتيجة وقيم الرايات SF CF ZF

62H+17H

في البداية نحول كل قيمة الى الميزان الثنائي:

62H=01100010

17H=00010111

					C	C		
العدد الاول	0	1	1	0	0	0	1	0
العدد الثاني	0	0	0	1	0	1	1	1
النتيجة	0	1	1	1	1	0	0	1

إذا نظرنا الى النتيجة باللون الازرق نرى انها لا تساوي صفر لانه يكفي ان يكون بالنتيجة على الاقل 1 حتى تكون النتيجة لا تساوي صفر اذا $ZF=0$
 إذا نظرنا الى البت الاخيرة في النتيجة في الجهة اليسرى نجد قيمتها صفر اذا النتيجة موجبة وليست سالبة اذا $SF=0$
 إذا نظرنا البت الاخيرة في العدد الاول + البت الاخيرة في العدد الثاني النتيجة انها لم ترسل Carry الى البت التاسعة ولم تأخذ borrow من التاسعة اذا $CF=0$
 النتيجة: كل اربع بتات تمثل منزلة في ميزان السادس عشر اذا
 $01111001=79H$

مهم جدا : تذكر

1 C: تعني 1

1+1+1-----s=1 c=1(2

تمرين: $87H+93H$

$87H=10000111$

$83H=10010011$

C					C	C	C	
العدد الاول	1	0	0	0	0	1	1	1
العدد الثاني	1	0	0	1	0	0	1	1
النتيجة	0	0	0	1	1	0	1	0

$ZF=0$ لان النتيجة ليست صفر

$SF=0$ لان النتيجة موجبة انظر اخر بت في اليسار للنتيجة .

ملاحظة: C الاخيرة التي نتجت عن اضافة اخر بت مع اخر بت في العددين لا تظهر في النتيجة لانه لا يوجد لها مكان لاننا قررنا ان النتيجة سوف تكون في 8 بتات فقط والذي يحدد ذلك كبر الهدف مثلا ADD 20H , BL نرى هنا ان الهدف مسجل بطول 8 بتات . ولكن كيف نعرف انه هنالك بت لم تظهر في النتيجة
 الراية $CF=1$

النتيجة: $00011010=1AH$

القفز غير المشروط

هنالك امر واحد فقط وهو jmp

ملاحظة مهمة جدا اوامر القفز بحاجة الى عنوان للقفز اليه أي يجب تسجيل العنوان بعد الامر
 مثال:

افحص اذا كان المسجل AL يحتوي على القيمة 6 اذا كان الامر كذلك ضع 7 في المسجل BL والا ضع 8 في المسجل DX
 الحل:

CMP AL,6

JE T1

MOV DX,8

JMP T2

T1:MOV BL ,7

T2:NOP

ملاحظات الامر NOP لا ينفذ شيئاً (لا تفعل شيئاً) (NO OPERATION)
الامر الاول الذي ينفذه هو امر المقارنة CMP وبعدها ينتقل الى الامر JE اذا تحقق ان الهدف والمصدر متاويان تتم عملية القفز الى العنوان T1 وهناك نرى انه سوف ينفذ الامر MOV BL, 7 ثم ينتقل الى الامر NOP
اما اذا لم يتحقق الشرط أي ان المصدر والهدف مختلفان فانه لا يقفز وانما ينفذ الامر الذي يأتي بعد امر القفز المشروط وهو MOV DX, 8 وبعدها ينتقل الى الامر التالي وهو القفز الغير مشروط JMP T2 فنتم عملية القفز الى العنوان T2

تمرين :

سجل الاوامر التي سوف ينفذها المقطع السابق في كل حالة من الحالات التالية:
AL=8

```
CMP AL,6
MOV DX,8
JMP T2
```

T2:NOP

AL=6

```
CMP AL,6
JE T1
T1:MOV BL,7
T2:NOP
```

تمرين:

اكتب مقطع برنامج يضع القيمة 20H في الخلايا الموجودة في المجال 1 حتى 8

الحل:

```
MOV BL,20H
MOV SI,1
T2:CMP SI,9
JE T1
MOV [SI],BL
INC SI
JMP T2
T1:NOP
```

ملاحظة: لتحديد عنوان في البرنامج حتى نستطيع ان نقفز اليه خلال البرنامج نسجل الاسم متبوعاً بنقطتين
مثلاً TO_NEXT:ADD SI,30H T10:INC SI
يحق استعمال الكلمات والارقام بشرط ان لا يكون الاسم عبارة عن كلمة محجوزة للغة الاسمبلي مثل NOP
INC

يجب ان تكون اسماء العناوين مختلفة فلا يجوز استعمال نفس الاسم اكثر من مرة واحدة

القفز الجزء الثاني

لقد تحدثنا في الدرس السابق عن انواع القفز وتطرقنا الى بعض اوامر القفز. في هذا الدرس سوف نتطرق الى اوامر جديدة.

في هذا الدرس سوف نتحدث عن الرايات (PF , AF , OF) للتذكرة:

الرمز	اسم الراية	النوع	ملاحظات
PF	Parity flag	وضع	اذا كان عدد ال 1 في اول 8 بتات من النتيجة زوجي تأخذ هذه الراية 1 والا تأخذ صفر
AF	Auxiliary flag	وضع	اذا اعطت البت الرابعة البت الخامس 1 او اخذت البت الرابعة من البت الخامسة 1 تأخذ هذه الراية 1 والا صفر
OF	Overflow flag	وضع	تأخذ 1 اذا كانت النتيجة اكبر من المكان المخصص لحفظها والا تأخذ صفر

الراية PF

هذه الراية تأخذ القيمة 1 اذا كان عدد البتات التي قيمتها 1 في اول ثمانية بتات عن اليمين زوجيا. و تأخذ القيمة صفر اذا كان عدد البتات التي قيمتها 1 فرديا.

ملاحظات مهمة: الصفر يعتبر زوجي
مثال:

جد وضع الراية PF بعد العملية الحسابية التالية

20H-20H

الحل :

تذكر: جدول الجمع في الميزان الثنائي:

a,b: عبارة عن بتات

D:حاصل الطرح $D=a-b$

b: borrow تعني ان هنالك حاجة لأخذ قيمة بقدر الميزان الذي نتعامل معه من البت التي تلي البت الحالية.

القصد في قيمة بالقدر الميزان في الميزان العشري نأخذ 10 وفي الميزان الثنائي نأخذ قيمة بقدر 2 وفي

الميزان السادس عشر نأخذ قيمة بقدر 16

a	b	D	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

اولا تحويل القيم الى الميزان الثنائي:

$$20H=0010000$$

العدد الاول	0	0	1	0	0	0	0	0
العدد الثاني	0	0	1	0	0	0	0	0
النتيجة	0	0	0	0	0	0	0	0

فهنا نرى ان النتيجة النهائية لا تحتوي على بتات قيمتها 1 أي ان عدد البتات التي قيمتها 1 هو صفر
PF=1

تمرين:

جد وضع الراية PF بعد العملية التالية:

$$2AH+71H=00101010+01110001$$

	C	C						
العدد الاول	0	0	1	0	1	0	1	0
العدد الثاني	0	1	1	1	0	0	0	1
النتيجة	1	0	0	1	1	0	1	1

عدد البتات التي قيمتها 1 في النتيجة هو 5 وهذا العدد هو فردي لذلك pf=0
جد وضع الراية PF بعد العملية التالية:

$$2FH-79H=00101111-01111001$$

b	b	b	b					
العدد الاول	0	0	1	0	1	1	1	1
العدد الثاني	0	1	1	1	1	0	0	1
النتيجة	1	0	1	1	0	1	1	0

PF=0

الملاحظة الثانية: لو كانت النتيجة اكبر من 8 بتات . مثلا 16 بت فاننا ننظر الى 8 بتات الاول من جهة اليمين فقط وان باقي البتات لا تؤثر على الراية PF
مثلا : لو كانت النتيجة

$$1001010011110000$$

فهنا عدد البتات التي قيمتها 1 في البتات الثمانية الاولى هو 4 اذا pf=1

الراية AF

تأخذ هذه الراية 1 اذا البت الخامسة اخذت من البت الرابعة Carry او اذا اخذت الرابعة من الخامسة borrow . باختصار ننظر الى البت الخامسة اذا كان فوقها C (carry) او b (borrow) فان $AF=1$ والا $AF=0$ امثلة:

$$2AH+6AH=00101010+01101010$$

	C	C		C		C		
العدد الاول	0	0	1	0	1	0	1	0
العدد الثاني	0	1	1	0	1	0	1	0
النتيجة	1	0	0	1	0	1	0	0

انظر الى C باللون الاحمر. البت الخامسة فوقها C لان البت الرابعة اعطت البت الخامسة CARRY لهذا السبب $AF=1$ تمرين:

$$23H+6AH=00100011+01101010$$

	C	C				C		
العدد الاول	0	0	1	0	0	0	1	1
العدد الثاني	0	1	1	0	1	0	1	0
النتيجة	1	0	0	0	1	1	0	1

هنا $AF=0$ لان البت الخامسة لم تأخذ من الرابعة C ولم تعطي الرابعة b

الراية OF

تعتبر هذه الراية مهمة جدا لانها تعطي المبرمج معلومة مهمة جدا عن النتيجة النهائية. فعندما لا يمكن اظهار النتيجة في المكان المحدد لها فان النتيجة تكون غير صحيحة وحتى يعرف المبرمج بانها غير صحيحة يتوجه الى هذه الراية. مثلا :

ADD AL,BL

المقصود هنا ان نجمع للمسجل AL فحوى المسجل BL ونحفظ النتيجة في المسجل AL . لو كانت النتيجة اكبر من بايت على سبيل المثال بحاجة الى اكثر من ثمانية بتات فان المسجل AL لن يستطيع حفظ النتيجة النهائية انما يحفظ جزء منها وهذه النتيجة تكون غير صحيحة هنا $OF=1$ باختصار: اذا كانت النتيجة غير منطقية فان $OF=1$ اليك بعض الحالات التي تأخذ OF القيمة 1

الحالة	العدد الاول	العملية الحسابية	العدد الثاني	النتيجة
الاولى	موجب	جمع(+)	موجب	سالبة
الثانية	سالب	جمع(+)	سالب	موجبة
الثالثة	موجب	طرح (-)	سالب	سالبة
الرابعة	سالب	طرح(-)	موجب	موجبة

امثلة:

$$71H+76H=01110001+01110110$$

	C	C	C					
العدد الاول	0	1	1	1	0	0	0	1
العدد الثاني	0	1	1	1	0	1	1	0
النتيجة	1	1	1	0	0	1	1	1

هنا $OF=1$ لان العدد الاول موجب والثاني موجب والنتيجة سالبة. لا يمكن ان نجمع عددين موجبين ونحصل على نتيجة سالبة.
شرح المشكلة:

$$71H=7 \times 16+1=113 \quad 76H=7 \times 16+6=118$$

$$113+118=231$$

وهذه النتيجة الموجبة لا يمكن ان تحفظ في مسجل من نوع بايت الذي يحتوي على 8 بتات لانه البت الاخيرة للاشارة اذا ببقى لدينا 7 بتات و اكبر قيمة يمكن حفظها في سبة بتات هي عندما تكون جميع البتات السبعة قيمتها 1. أي

$$01111111=7FH=7*16+15=127$$

اذا 231 لا يمكن ان تحفظ في 7 بتات فتستغل بت الاشارة لتضع بها بقية القيمة فتظهر النتيجة هنا سالبة .
مثال اخر:

$$B1H+96H=10110001+01000111$$

C		C	C					
العدد الاول	1	0	1	1	0	0	0	1
العدد الثاني	1	0	0	1	0	1	1	0
النتيجة	0	1	0	0	0	1	1	1

هنا $OF=1$ لان عدد سالب اضفنا له عدد سالب اخر فحصلنا على نتيجة موجبة وهذا غير منطقي . انظر الى البت الثامنة في كل عدد والنتيجة.
تحليل المشكلة:

العدد الاول هو 10110001 وهو عدد سالب انظر الى البت الحمراء التي تمثل الاشارة. لمعرفة قيمته علينا ان نجد قيمته المطلقة.

واللحصول عليه نعمل ما يلي:

-نعكس قيم كل البتات مكان 1 نسجل 0 ومكان 0 نسجل 1

نحصل على القيمة 01001110

نضيف للقيمة الجديدة 1
 $01001110+00000001$
 نحصل على
 01001111

$$01001111=4FH=4 \times 16+15=79$$

إذا العدد الاول هو 79 - عشري
 الان الى العدد الثاني: العدد 96H
 $96H=10010110$ وهو ايضا عدد سالب انظر الى بت الاشارة لمعرفة قيمته نستعين بالقيمة المطلقة
 -نعكس قيم البتات نحصل على العدد التالي

01101001
 نضيف القيمة 1 للعدد الجديد
 $01101001+00000001$
 نحصل على 01101010

$$01101010=6AH=6 \times 16+10=106$$

إذا العدد الثاني هو 106 - عشري
 إذا

$$-79+(-106)=-79-106=-185$$

هذه القيمة لا يمكن كتابتها في مسجل يتكون من من ثمانية بتات لان اصغر عدد سالب يمكن كتابته في المسجل هو 128-

مجال الاعداد السالبة التي يمكن تخزينه في مسجل معين هو دائما من 1- حتى العدد $(2^N-1)/2$. الرمز N هو عدد البتات للمسجل يشمل بت الاشارة مثلا AL له ثمانية بتات اذا

$$(2^8-1)/2=128$$

إذا المجال المسموح به هو
 $[-128..-1]$

القيمة 185- هي خارج المجال ولا يمكن تخزينها لهذا السبب $OF=1$

تمارين وحلول

جد قيم رايات الوضع بعد العملية الحسابية

$$4AH+77H=01001010+01110111$$

	C	C	C	C	C	C		
العدد الاول	0	1	0	0	1	0	1	0
العدد الثاني	0	1	1	1	0	1	1	1
النتيجة	1	1	0	0	0	0	0	1

قيم الرايات حسب النتيجة

OF	AF	PF	CF	ZF	SF
1	1	0	0	0	1

تمرين آخر
جد قيم رايات الوضع بعد العملية الحسابية

4AH-77H=01001010-01110111

b	b	b	b		b	b	b	
العدد الاول	0	1	0	0	1	0	1	0
العدد الثاني	0	1	1	1	0	1	1	1
النتيجة	1	0	0	1	0	0	1	1

قيم الرايات حسب النتيجة

OF	AF	PF	CF	ZF	SF
0	0	1	1	0	1

ملاحظة: OF=0 لان النتيجة منطقية عدد موجب ناقص عدد موجب النتيجة سالبة صحيح يمكن ان تكون اذا كان العدد الثاني اكبر من الاول وهنا

$$4AH-77H=4 \times 16+10-(7 \times 16+7)=74-119=-45$$

وهذه القيمة في المجال المذكور

اوامر القفز المشروط المتعلقة في الرايات الجديدة

الامر	المعنى	الراية المطلوبة
JP	JMP IF PARITY	PF=1
JNP	JMP IF NO PARITY	PF=0
JPE	JMP IF PARITY EVEN	PF=1
JPO	JMP IF PARITY ODD	PF=0
JO	JMP IF OVERFLOW	OF=1
JNO	JMP IF NOT OVERFLOW	OF=0

القائمة التالية تتعلق في القفز المشروط عند التعامل مع الاعداد الموجهة (اعداد سالبة واعداد موجبة)

CMP OP1 , OP2

الامر	وصف الامر
JG	JMP IF OP1>OP2 (JMP IF GREATER)
JNG	JMP IF OP1 <= OP2
JL	JMP IF OP1<OP2 (JMP IF LESS THAN)
JNL	JMP IF OP1>=OP2
JGE	JMP IF OP1>=OP2 (JMP IF GREATER THAN OR EQUAL)
JNGE	JMP IF OP1<OP2
JLE	JMP IF OP1<=OP2

JNLE	JMP IF OP1>OP2
------	----------------

هناك بعض اوامر القفز التي تتعامل فقط مع الاعداد الموجبة فسوف اتطرق اليها في الدروس القادمة.
بعض الاوامر الحسابية

(1) الجمع مع الراية cf

ADC (ADD WITH CARRY)

ADC OP1 ,OP2

OP1=OP1+OP2+CF*

ملاحظات:

-المقصود في CF* أي قيمة الراية CF التي كانت قبل عملية الجمع وليست القيمة الجديدة التي عملتها عملية الجمع الجديدة

مثال: معطى مقطع البرنامج التالي:

MOV AL,90H

ADD AL,85H

ADC AL,3

ملاحظات	CF	AL	الامر
	?	??	
MOV لا تؤثر على الرايات	?	90H	MOV AL,90H
	1	15H	ADD AL,85H
AL=15H+3+1	0	19H	ADC AL,3

ملاحظة: ان النتيجة غير منطقية لانها بحاجة الى اكثر من ثمانية بتات

(2) الطرح مع راية CF

SBB (SUBTRACT WITH CARRY)

SBB OP1,OP2

OP1=OP1-OP2-CF*

الى اللقاء في الدرس القادم

اوامر القفز المشروط الجزء الثالث

لقد تعلمنا في الدروس السابقة اوامر قفز متنوعة:

1- اوامر قفز مشروط التي تسأل عن راية من الرايات (jz , je , jc,jo.....)

2- اوامر قفز مشروط المتخصصة في الاعداد الموجهة (الاعداد السالبة والاعداد الموجبة) أي انها تعتبر

البت الاخيرة من الجهة اليسرى هي بت الاشارة

3- اوامر قفز مشروط المتخصصة في الاعداد الموجبة أي انها لا تعتبر البت الاخيرة من الجهة اليسرى بت

اشارة ولكنها تعتبرها جزء من العدد وهي ذات القيمة الكبرى.

مثال:

X=11111111

هذا العدد بالنسبة للقفز المشروط المتخصص في الاعداد الموجهة عبارة عن عدد سالب لان البت الاخير من الجهة اليسرى قيمتها 1. وهذا العدد هو عبارة عن 1- (ناقص واحد) اما بالنسبة لاوامر القفز المشروط المتخصصة في الاعداد الموجبة فقط. فبالنسبة لها لا توجد اعداد سالبة بل جميع الاعداد هي اعداد موجبة والبت الاخير من ضمن العدد. وهذا العدد قيمته 255 عشري

القائمة التالية تتعلق في القفز المشروط عند التعامل مع الاعداد الموجبة

CMP OP1 , OP2

الامر	وصف الامر
JA	JMP IF OP1>OP2 (JMP IF above)
JNA	JMP IF OP1 <= OP2
JB	JMP IF OP1<OP2 (JMP IF BELOW)
JNB	JMP IF OP1>=OP2
JAE	JMP IF OP1>=OP2 (JMP IF ABOVE OR EQUAL)
JNAE	JMP IF OP1<OP2
JBE	JMP IF OP1<=OP2
JNBE	JMP IF OP1>OP2

امثلة:

معطى المقطع التالي:

```
MOV AL , -1
CMP AL,0
JG T1
MOV BL,4
JMP T2
T1:MOV BL,6
T2:NOP
```

ما هو فحوى المسجل BL عند الوصول الى العنوان T2
الحل: BL=4 لان الامر JG من اوامر القفز المشروط التي تتعامل مع الاعداد الموجهة . فانها تعتبر ان المسجل AL يحتوي على القيمة 1- وهي قيمة سالبة اصغر من صفر . اذا لا يتم القفز وانما يقوم المعالج بالانتقال الى الامر الذي يليه وهو

MOV BL,4

وبعدها يقفز الى العنوان T2

تمرين: معطى المقطع التالي:

```
MOV AL , -1
CMP AL,0
JA T1
MOV BL,4
JMP T2
```

T1:MOV BL,6
T2:NOP

ما هو فحوى المسجل BL عند الوصول الى العنوان T2

الحل: المسجل BL=6 لان الامر JA يتعامل مع الاعداد الموجبة فقط فهو لا يعرف الاعداد السالبة.
بالنسبة له هذه قيمة 255 عشري وفعلا هذه القيمة اكبر من الصفر فيتم القفز الى T1 وهناك يضع المعالج القيمة 6 في المسجل AL وينتقل بعدها الى العنوان T2

من هنا نرى انه علينا الانتباه عند استعمال اوامر القفز المشروط أي امر نأخذ

اوامر القسمة والضرب

اوامر الضرب:

في لغة الاسبلي توجد اوامر لضرب الاعداد الموجبة فقط واوامر لضرب الاعداد الموجهة (أي الموجبة والسالبة)

ضرب الاعداد الموجبة:

نستعمل هنا امر الضرب MUL ولكن هنالك شروط يجب ان تتحقق عندما نريد ان نستعمل هذا الامر .

MUL OP1

ملاحظات مهمة:

- OP1 يمكن ان يكون مسجلا او اسم متغير او خلية في الذاكرة ويمكن ان يكون خليتين
- يمكن ان يكون OP1 بطول 8 بتات او 16 بتا
- اذا كان OP1 من نوع بايت عندها تتم العملية التالية

AX=AL*OP1

أي ان المعالج سوف يقوم بضرب OP1 بفحوى المسجل AL وفقط هذا المسجل ويضع النتيجة في المسجل AX. لهذا السبب يجب ان نضع احدى القيم في المسجل AL وان نحفظ قيمة المسجل AH في مكان اخر قبل عملية الضرب لانه بعد عملية الضرب سوف يقوم المعالج بحذف القيمة القديمة ويضع مكانها القيمة الجديدة

مثال:

اكتب مقطع برنامج يضرب فحوى الخلية 20H في القيمة 8 ويحفظ النتيجة في المسجل CX
الحل:

MOV SI,20H
MOV AL,[SI]
MOV BL,8
MUL BL
MOV CX,AX

-اذا كان OP1 من نوع كلمة أي 16 بت عندها تتم العملية التالية

DX:AX=AX*OP1

عندها يقوم المعالج بضرب OP1 بقيمة فحوى المسجل AX ويحفظ النتيجة في المسجلين DX و AX لهذا السبب يجب ان نضع احدى القيم في المسجل AX وان نحفظ القيمة الموجودة في المسجل DX في مكان اخر قبل عملية الضرب

مثال :

عليك ضرب العدد FFFFH بالعدد 4

الحل:

```
MOV AX,0FFFFH
```

```
MOV CX,4
```

```
MUL CX
```

ملاحظة: لقد اضفنا صفر امام العدد الذي وضعناه في المسجل AX لان القانون يقول كل عدد يبدأ بحرف من الحروف (A ,B ,C, D ,E , F) يجب اضافة صفر عن يساره

ضرب الاعداد الموجبة

نفس الشروط التي ذكرتها سابقا ولكن بدلا من الامر MUL نستخدم IMUL

مثال :

اكتب مقطع برنامج ينفذ التمرين التالي

```
CX=-10 * DL
```

الحل:

```
MOV AL,-10
```

```
IMUL DL
```

```
MOV CX,AX
```

القسمة

تقسم عمليات القسمة الى قسمين

1-قسمة الاعداد الموجبة فقط

2-قسمة الاعداد الموجبة(سالبة وموجبة)

قسمة الاعداد الموجبة

نستخدم في هذه الحالة الامر DIV

وهو بالشكل التالي:

```
DIV OP1
```

-اذا كان OP1 من نوع بايت عندها تتم العملية التالية:

```
AX/OP1
```

يقوم المعالج بتقسيم القيمة الموجودة في المسجل AX (فقط هذا المسجل)

على OP1 ويتم حفظ الناتج في المسجلات التالية:

AL: يحفظ الناتج الصحيح

AH: يحفظ الباقي الصحيح

مثال:

قسم القيمة 9 على 4

الحل:

```
MOV AX,9
MOV BL,4
DIV BL
```

بعد تنفيذ هذا المقطع

AH=1 AL=2

ملاحظة: لا يجوز الاختزال في الباقي الصحيح حتى ولو امكن ذلك
مثال:

```
MOV AX,8
MOV CL,6
DIV CL
```

AH=2 AL=1

صحيح انه $2/6 = 1/3$ الا انه لا يجوز التبسيط
- اذا كان OP1 من نوع 16 بت تتم العملية التالية

DX:AX/OP1

أي ان المعالج يتعامل مع المسجلين كأنهما مسجلا واحدا ويقسمهما على OP1 ويحفظ النتيجة بالشكل التالي:
AX: يحتوي على الناتج الصحيح
DX: يحتوي على الباقي الصحيح

قسمة الاعداد الموجبة

هنا نستخدم نفس الشروط مثل الاعداد الموجبة ولكن الفرق الوحيد هو بدلا من الامر DIV نستخدم

IDIV الامر

مثال:

احسب ما يلي

-120/12H

الحل:

```
MOV AX,-120
MOV BL,12H
IDIV BL
```

تمرين:

اكتب مقطع برنامج يحسب

1234H/-1010

الحل:

```
MOV DX,0
MOV AX,1234H
MOV CX,-1010
IDIV CX
```

ملاحظة: لقد وضعنا القيمة صفر في المسجل DX قبل عملية القسمة لان المعالج سوف يستخدم القيمة الموجودة فيه عند القسمة ويمكن ان تكون قيمة مختلفة عن الصفر وحتى نتأكد ان القيمة صفر وضعنا الصفر في DX

الحلقات

يمكن تكوين الحلقات في لغة الاسبلي في عدة طرق نذكر منها الطرق التالية:
 1_ الطريقة الاولى باستخدام المسجل CX والامر LOOP .
 نضع في المسجل CX عدد الدورات اللازمة
 سجل عنوانا عن يسار الامر الاول في الحلقة
 ننهي الحلقة في الامر LOOP مع عنوان بداية الحلقة.
 المبنى العام

عنوان الحلقة:

اوامر الحلقة
 LOOP عنوان الحلقة

مثال :

اكتب مقطع برنامج يقوم بتخزين القيمة 28H في الخلايا الموجودة في المجال [400H..407H]
 عدد الخلايا المطلوبة $407H - 400H + 1 = 8$
 اذا

```
MOV SI,400H
MOV AL,28H
MOV CX,8
T1:MOV [SI],AL ; بداية الحلقة
    INC SI
    LOOP T1      ; نهاية الحلقة
```

عندما يصل المعالج الى الامر LOOP يقوم بتنفيذ عمليتين وهما:
 يقوم بطرح 1 من المسجل CX
 $CX = CX - 1$

يقوم بقرنة القيمة الجديدة للمسجل CX مع الصفر اذا كانت تساوي صفر ا ينهي المعالج الحلقة ويخرج وينفذ الامر الذي يلي الامر LOOP . واذا كانت قيمته لا تساوي صفر تتم عملية القفز الى بداية الحلقة

الطريقة الثانية: بمساعدة CMP والامر JMP
 المبنى العام

اعطاء عداد الحلقة قيمة بدائية تمثل عدد الدورات المطلوبة
 - عنوان الحلقة

اوامر الحلقة

1 - عداد الحلقة=عداد الحلقة

0, عداد الحلقة CMP

عنوان الحلقة JNZ

مثال: اكتب مقطع برنامج يضيف القيمة 4 لكل خلية في المجال [0..5]
الحل:

نأخذ المسجل BL كعداد للحلقة

```
MOV SI,0
MOV BH,4
MOV BL,6
T1:ADD [SI],BH
INC SI
DEC BL
CMP BL,0
JNZ T1
```

العمليات المنطقية

في لغة الاسمبلي هنالك عمليات حسابية مثل (الجمع ، الطرح ، الضرب ، القسمة) وهنالك العمليات المنطقية

التي تتعامل مع البتات مثل (OR , AND, NOT ,XOR ,TEST)
النتيجة من هذه العمليات تكون اما صدق TRUE والتي نعبر عنها في القيمة 1 . او تكون النتيجة من نوع كذب FALSE والتي نعبر عنها بالقيمة 0.

الامر الاول: OR

هنا يكفي شرطا واحدا ان يتحقق فنحصل على نتيجة صدق

اليكم جدول الامر OR

النتيجة	B	A
0	0	0
1	1	0
1	0	1
1	1	1

ملاحظات :- A,B عبارة عن بتات .

- يأخذ هذا الامر مصدر وهدف OR OP1 , OP2

- يجب تحويل قيم المصدر والهدف الى الميزان الثنائي حتى نتمكن من التعامل مع القيم

مثال: معطى

AL=2FH

BL=52H

جد قيمة كل مسجل من المسجلات بعد تنفيذ العملية المنطقية التالية:

OR AL , BL

الحل:

علينا ان نحول الهدف والمصدر الى الميزان الثنائي

AL=2FH=00101111

BL=52H=01010010

العدد الاول	0	0	1	0	1	1	1	1
العدد الثاني	0	1	0	1	0	0	1	0
النتيجة	0	1	1	1	1	1	1	1

الحل :

بما ان المسجل BL هو المصدر فان قيمته لا تتغير وتبقى BL=52H

المسجل AL هو الهدف اذا التأثير يتم عليه حسب النتيجة AL=7FH

الامر الثاني: AND

هنا يكفي ان لا يتحقق شرطا واحدا حتى نحصل على نتيجة كذب

اليكم جدول الامر AND

النتيجة	B	A
0	0	0
0	1	0
0	0	1
1	1	1

ملاحظات :- A,B عبارة عن بتات .

- يأخذ هذا الامر مصدر وهدف AND OP1 , OP2

- يجب تحويل قيم المصدر والهدف الى الميزان الثنائي حتى نتمكن من العمل مع القيم

مثال: معطى

AL=27H

BL=56H

جد قيمة كل مسجل من المسجلات بعد تنفيذ العملية المنطقية التالية:

AND AL , BL

الحل:

علينا ان نحول الهدف والمصدر الى الميزان الثنائي

AL=27H=00100111

BL=56H=01010110

العدد الاول	0	0	1	0	0	1	1	1
العدد الثاني	0	1	0	1	0	1	1	0
النتيجة	0	0	0	0	0	1	1	0

الحل :

بما ان المسجل BL هو المصدر فان قيمته لا تتغير وتبقى BL=56H
المسجل AL هو الهدف اذا التأثير يتم عليه حسب النتيجة AL=06H

الامر الثالث: TEST

هنا يكفي ان لا يتحقق شرطا واحدا حتى نحصل على نتيجة كذب

اليكم جدول الامر TEST

النتيجة	B	A
0	0	0
0	1	0
0	0	1
1	1	1

ملاحظات :- A,B عبارة عن بتات .

- يأخذ هذا الامر مصدر وهدف TEST OP1 , OP2

يجب تحويل قيم المصدر والهدف الى الميزان الثنائي حتى نتمكن من العمل مع القيم
يعمل هذا الامر مثل الامر AND ولكن مع فرق بسيط وهو انه لا يؤثر على الهدف ولا على

المصدر وانما يؤثر فقط على الرايات ويقوم بتغيير بعض الرايات

مثال: معطى

AL=27H

BL=56H

جد قيمة كل مسجل من المسجلات بعد تنفيذ العملية المنطقية التالية:

TEST AL , BL

الحل:

علينا ان نحول الهدف والمصدر الى الميزان الثنائي

AL=27H=00100111

BL=56H=01010110

العدد الاول	0	0	1	0	0	1	1	1
العدد الثاني	0	1	0	1	0	1	1	0
النتيجة	0	0	0	0	0	1	1	0

الحل :

بما ان المسجل BL هو المصدر فان قيمته لا تتغير وتبقى BL=56H
المسجل AL هو الهدف ولكن الامر TEST لا يؤثر على الهدف المسجل يحافظ على قيمته السابقة

AL=27H

اذا ما هي الفائدة منه؟

الجواب : في بعض الاحيان نحتاج الى فحص بت معينة دون ان نؤثر على القيمة الاصلية. فنستخدم هذا الامر

مثال: اكتب مقطع برنامج يفحص اذا كان المسجل DL يحتوي على قيمة فردية اذا كان يحتوي على قيمة فردية يضع القيمة 6 المسجل AL واذا كانت القيمة زوجية يضع في المسجل AL القيمة 12H
الحل: يعتبر العدد فرديا اذا كانت البت الاولى قيمتها 1 والسبب هو ان قيم جميع البتات في أي عدد زوجية ما عدا البت الاولى التي قيمتها 1 . اذا نفحص البت الاولى

الرمز B يدل على قيمة ثنائية ; TEST AL,00000001B

JZ T1

MOV AL,6

JMP FINISH

T1:MOV AL,12H

FINISH:

الامر الرابع: XOR

هنا اذا كانت البتات مختلفة النتيجة 1 واذا كانت متشابهة النتيجة 0

اليكم جدول الامر XOR

النتيجة	B	A
0	0	0
1	1	0
1	0	1
0	1	1

ملاحظات :- A,B عبارة عن بتات .

- يأخذ هذا الامر مصدر وهدف AND OP1 , OP2

يجب تحويل قيم المصدر والهدف الى الميزان الثنائي حتى يتمكن من العمل مع القيم

مثال: معطى

AL=A7H

BL=76H

جد قيمة كل مسجل من المسجلات بعد تنفيذ العملية المنطقية التالية:

XOR AL , BL

الحل:

علينا ان نحول الهدف والمصدر الى الميزان الثنائي

AL=A7H=10100111

BL=76H=01110110

العدد الاول	1	0	1	0	0	1	1	1
العدد الثاني	0	1	1	1	0	1	1	0
النتيجة	1	1	0	1	0	0	0	1

الحل :

بما ان المسجل BL هو المصدر فان قيمته لا تتغير وتبقى BL=76H

المسجل AL هو الهدف اذا التأثير يتم عليه حسب النتيجة AL=D1H

الامر الخامس: XOR
هنا يتم عكس البتات البت التي قيمتها 1 تأخذ القيمة 0 والبت التي قيمتها 0 تأخذ القيمة 1
اليكم جدول الامر NOT

النتيجة	A
0	0
0	1

ملاحظات :- A عبارة عن بت .
- يأخذ هذا الامر OPERAND واحد فقط NOT OP1
يجب تحويل قيمته الى الميزان الثنائي حتى نتمكن من التعامل معه

مثال: معطى

AL=A7H

جد قيمة كل المسجل من بعد تنفيذ العملية المنطقية التالية:

NOT AL

الحل:

علينا ان نحول الى الميزان الثنائي

AL=A7H=10100111

العدد الاول	1	0	1	0	0	1	1	1
النتيجة	0	1	0	1	1	0	0	0

الحل :

المسجل AL هو الذي يتأثر حسب النتيجة AL=58H
الى اللقاء في الدرس القادم

المتغيرات

المتغير: هو عبارة عن مكان في الذاكرة له عنوان وقيمة وحجم. المقصود في الحجم عدد الخلايا التي يحجزها المتغير

تعرف المتغيرات في مقطع البيانات او في مقطع الكود (الاوامر) ولكن من المفضل ان تكون في مقطع البيانات الا اذا اردنا دمج مقطع البيانات مع مقطع الكود لتكوين مقطع واحد بدل مقطعين
حجز خلايا في الذاكرة

المتغير يحجز خلايا ولكن كيف نحدد عد الخلايا المطلوبة. اليكم هذا الجدول الذي يبين عدد الخلايا المطلوبة للحجز

عدد الخلايا(عدد البايت)	المعنى	الاختصار
1	Define byte	db
2	Define word	dw
4	Define double	dd

	word	
dq	Define quad word	8
dt	Define ten bytes	10

ملاحظة: عندما نريد ان نحجز اماكن في الذاكرة حتى نضع بها قيم معينة خلال تنفيذ البرنامج. يمكن ان نحجز بدون ان نسجل اسم متغير ولكن هنالك صعوبة ستواجهنا خلال البرنامج عندما نحاول ان نتوجه الى هذه الاماكن. لانه علينا ان نحسب العنوان لوحدنا بينما عندما نعطي الاماكن المحجوزة اسم معين فهذا يسهل عملية التوجه اليها لان المعالج يقوم بحسابة العنوان بدلا منا

ملاحظة: يمكن اعطاء المتغير قيمة أولية ويمكن عدم اعطاء المتغير قيمة أولية. اذا اردنا ان لا نعطي المتغير قيمة أولية نسجل مكان القيمة الرمز ؟ (علامة سؤال)

امثلة:

المعنى	صورة الكتابة
هنا نعلن عن متغير باسم X وهو من نوع بايت أي انه حجز خلية واحدة وقيمتة الاولى 4	X db 4
هنا نعلن عن متغير باسم Y من نوع word ولكن بدون قيمة أولية.	Y dw ?
هنا نعلن عن حجز خلية بدون عنوان ونضع بها القيمة 5	Db 5

X db 2,3,4,5,1

هنا تم تعريف مصفوفة تتكون من خمسة حدود وكل حد عبارة عن بايت أي ان هذه المصفوفة قد حجزت 5 خلايا. لاحظ انه يجب ان نفصل بين الحدود بمساعدة الفاصلة

X1 dw 2,3,4,5,1

هنا تم تعريف مصفوفة باسم x1 تتكون من خمسة حدود وكل حد من نوع word أي ان المصفوفة تحجز 10 خلايا.

ملاحظة: عدد الخلايا التي تحجزها مصفوفة هو بقدر عدد الحدود ضرب ما يحجزه كل حد

Dt 1234

الرموز والنصوص

نعرف متغيرات من نوع رمز بالشكل التالي:

X db 'A'

هنا تم تعريف متغير باسم X وهو من نوع رمز ويحجز خلية واحدة وقيمه الاولى A ولكن تذكر ان المعالج يخزن اسكي كود للرمز وهو 65 عشري

```
X db 'my name is afeef '
```

هنا تم تعريف متغير من نوع نص باسم x وقيمه الجملة المكتوبه في داخل الشطائر

النسخ طبق الاصل dup (duplicate)
اذا اردنا مثلا حجز 6 خلايا ووضع القيمة 20 في كل خلية
هنالك عدة طرق للحل نذكر منها

```
X db 20,20,20,20,20,20
```

الطريقة الثانية استعمال dup

```
X db 6 dup (20)
```

لاحظ القيمة التي عن يسار ال dup تحدد عدد الحدود المطلوبة والقيمة التي بين الاقواس تحدد القيمة البدائية لكل حد .

ملاحظة مهمة: يمكن تسجيل كود اسكي بدلا من الرموز. مثال

```
X db 'AB'
```

كود اسكي للرمز A هو 65 عشري وللرمز B هو 66 عشري اذا

```
X db 65,66
```

لاحظ عندما نسجل كود اسكي للرمز لا نضع الشطائر
مثال اخر

```
Y db '123'
```

يمكن تسجيل كود اسكي للرمز 1 والرمز 2 والرمز 3

```
Y db 31H, 32H, 33H
```

من الاشياء المريحة جدا في لغة التجميع انه يمكن تعريف متغير يحتوي على رموز ونصوص وقيم عديدة
مثال:

```
X db 1, 2, 3,'AB','C', 4 dup (5)
```

هنا تم تعريف متغير باسم X ويحجز هذا المتغير 10 خلايا . يمكن كتابة المصفوفة بشكل اخر

```
X db 1,2,3,65,66,67,5,5,5,5
```

ملاحظة مهمة جدا:

الحد الاول في المصفوفة او النص او من حدود المتغير رقمه التسلسلي صفر
مثال: اذا اردنا ان نجمع الحد الرابع في المصفوفة السابقة للمسجل AL نسجل الاوامر التالية

```
Mov si,3
```

```
Add AL,x[si]
```

او يمكن كتابة الرقم التسلسلي مباشرة

```
Add AL,x[3]
```

تسجيل الملاحظات في لغة التجميع
تستخدم الملاحظات لتوضيح البرنامج للقارئ اذا اردنا ان نسجل ملاحظات نستخدم الفاصلة المنقوطة (;)
كل الكلمات التي تقع عن يمين الفاصلة المنقوطة لا ينفذها المترجم

مثال:

Mov BL,6 ; put 6 in BL

كتابة البرامج

اكتب برنامجا ينفذ $Y = 2 \times AL - 5$. يضرب المسجل al في القيمة 2 ويخزن النتيجة في المتغير y
الحل:

المتغير Y يجب ان يكون من نوع word أي ان يحجز خليتين نحصل على نتيجة اكبر من بايت واحد
بسبب عملية الضرب

النتيجة يمكن ان تكون سالبة لاننا لا نعرف ماذا يوجد في المسجل al يمكن مثلا ان تكون قيمة سالبة عندما
نستخدم امر الضرب يجب ان نستخدم امر الضرب المتخصص في الاعداد الموجهة
الحل:

Sseg segment stack

Db 10 dup (0)

Sseg ends

Dseg segment

Y dw 0

Dseg ends

Cseg segment

Assume cs:cseg,ds:dseg,ss:sseg

Start:mov ax,dseg

Mov ds,ax

Mov BL,2

Imul BL

Mov Y,AX

Sub Y,5

Mov ah,4ch

Int 21h

Cseg ends

End start

```
Sseg segment stack
    Db 10H dup (0)
Sseg ends
```

هنا تم تعريف مقطع المكس stack سوف نتطرق اليه في الدروس القادمة عنوان بداية المكس sseg وهي كلمة اختيارية أي يمكن اختيار كلمة غيرها ولكن يجب ان لا تكون كلمة محفوظة أي من كلمات لغة الاسمبلي. Segment stack هذه اجبارية وهي تدل على ان هذا مقطع المكس

```
    Db 10H dup (0)
```

هنا نعلن ان المكس حجز 10 خلايا وفي كل خلية القيمة صفر

```
Sseg ends
```

هنا تم الاعلان عن نهاية تعريف منطقة المكس

```
Dseg segment
```

```
Y dw 0
```

```
Dseg ends
```

هنا تم الاعلان عن تعريف مقطع البيانات وقد قمنا بتعريف متغير من نوع word ووضعنا به القيمة الاولى صفر

```
Cseg segment
```

```
Assume cs:cseg,ds:dseg,ss:sseg
```

هنا تم الاعلان عن بداية مقطع الكود والاعلان عن سجلات المقاطع وعن العناوين التي سوف تأخذها خلال البرنامج

```
Start:mov ax,dseg
```

```
    Mov ds,ax
```

هنا تم وضع عنوان بداية البيانات في مسجل المقطع ds .
ملاحظة لا يمكن تسجيل الامر بالشكل التالي: mov ds,dseg
يجب الاستعانة بمسجل كامل مثل bx,cx,dx,si,di,ax

```
Mov BL,2
```

```
    Imul BL
```

```
    Mov Y,AX
```

```
    Sub Y,5
```

هذه هي الاوامر التي تنفذ المهمة السابقة

```
Mov ah,4ch
```

```
    Int 21h
```

هذه تنهي البرنامج وترجع السيطرة والتحكم لنظام التشغيل

Cseg ends

هنا نعلن عن انتهاء مقطع الكود (الوامر)

End start

اغلق البرنامج لاحظ اننا سجلنا في بداية البرنامج start:

وفي النهاية

End start

ملاحظة هنالك طريقة اخرى لكتابة البرامج

وامر الدوران

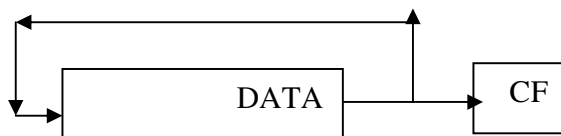
هنالك عدة اوامر للدوران في لغة الاسبلي

1. دوران نحو اليمين: فهنا تتحرك البتات من اليسار الى جهة اليمين اي ان البت الاولى تنتقل مكان البت

الاخيرة والاخيرة الى قبل الاخيرة وهكذا. صورة من البت الاولى التي انتقلت الى البت الاخيرة تدخل الى

راية CF

ROR OP,1



مثال:

MOV AL,3DH

ROR AL,1

0	0	1	1	1	1	0	1
---	---	---	---	---	---	---	---

AL

1	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---

CF

1

ملاحظة:المصدر هنا عبارة عن قيمة عددية ويجب ان تكون 1 فقط واحد وفي حالة اذا اردنا تدوير اكثر من بت واحدة فهناك حلان

الحل الاول ان نسجل الامر بقدر عدد مرات البتات المراد تدويرها مثلا:

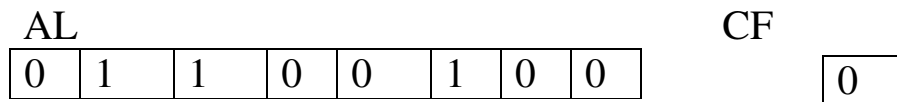
MOV AL,23H; AL=00100011

ROL AL,1

ROL AL,1

ROL AL,1

فهنا تم تدوير ثلاث بتات .



حل اخر: نضع في CL عدد مرات الدوران ثما نسجل الامر
مثال:

```
MOV AL,26H
MOV CL,3
ROR AL,CL
```

لاحظ انه البت التي تتحرك تخرج من ال DATA تضع صورة من قيمتها في الراية CF وتنتقل البت الى
اخر ال DATA وان القيمة المحفوظة في الراية CF هي قيمة اخر بت خرجت من ال DATA

تمرين: اكتب مقطع برنامج يحفظ في المسجل BL عدد البتات التي قيمتها 1 في المسجل AX

الحل:

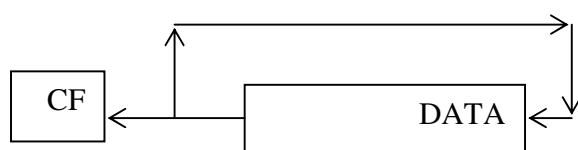
فكرة الحل هي تدوير بتات المسجل AX 16 دورة وفي كل دورة نسأل عن قيمة الراية CF اذا كانت 1
نضيف للعداد BL القيمة 1. نسأل عن الراية cf لان كل بت تنتقل من البداية الى النهاية تضع صورة منها
في داخل هذه الراية

```
MOV CX,16
MOV BL,0
T1:ROR AX,1
JC CF_1
JMP CF_0
CF_1:ADD BL,1
CF_0:LOOP T1
```

(2) الامر الثاني المتخصص في الدوران هو الدوران لليسا والامر هو

ROL OP,1

فهنا البت الموجودة في نهاية المسجل تضع صورة لها في الراية CF وتنتقل الى البداية. كما هو مبين في
الشكل التالي:



مثال: لو فرضنا ان المسجل BL يحتوي على قيمة 6CH ونفذنا الامر التالي:

ROL BL,1

0	1	1	0	1	1	0	0
---	---	---	---	---	---	---	---

عندها نحصل على ما يلي:

CF

BL

0

1	1	0	1	1	0	0	0
---	---	---	---	---	---	---	---

السؤال اذا اردنا ان نقوم بتدوير عدة بتات مرة واحدة فاننا نستعمل المسجل CL ونضع فيه عدد الدورات
مثال:

ROL DL,1

ROL DL,1

ROL DL,1

ROL DL,1

يمكن ان نسجل بدلا من هذه الاوامر ما يلي:

MOV CL,4

ROL DL,CL

اوامر الازاحة

هنالك اوامر ازاحة بتات تتخصص في الاعداد الموجبة وهنالك اوامر ازاحة تتخصص في الاعداد الموجبة
1.ازاحة نحو اليمين الاعداد الموجبة.

SHR OP,1

هنا يتم اضافة صفر في نهاية ال OP ونقوم بدفع البتات الاخرى الى اليمين وفي هذه الحالة تخرج البت
الاولى الى راية CF
كما هو مبين في الشكل التالي:



مثال: معطى BL=75H اذا نفذنا الامر التالي:

SHR BL,1

اننا نحصل على الشكل التالي:
المسجل BL قبل تنفيذ الامر

0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

بعد عملية التنفيذ نحصل على

0	0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---	---

BL

CF

ملاحظة: هذه الامر يقوم بعملية قسمة القيمة على 2 ولكن دون باق أي انه يحفظ الناتج الصحيح من خارج القسمة والباقي يذهب الى الراية CF
لو اردنا ان نعمل الامر عدة مرات فاننا سوف نسجل الامر عدة مرات او نستعين في بالمسجل CL
مثال:

SHR AL,1
SHR AL,1
SHR AL,1

يمكن نسجل بدلا من هذه الاوامر ما يلي:

MOV CL,3
SHR AL,CL

ملاحظة: هنا لا يمكن تخزين الباقي في الراية CF لانه يمكن ان نخزن بت واحدة فقط وهي اخر بت خرجت من المسجل او المتغير. هنا تمت قسمة المسجل عل القيمة 8 لانه في كل ازاحة نقسم على 2 وهنا عملنا ثلاث ازاحات اذا قسمنا على 8

2. ازاحة نحو اليسار.

SHL OP,1

هنا يتم اضافة صفر في بداية ال OP ونقوم بدفع البتات الاخرى الى اليسار وفي هذه الحالة تخرج البت الاخيرة الى راية CF
كما هو مبين في الشكل التالي:



مثال: معطى BL=75H اذا نفذنا الامر التالي:

SHL BL,1

اننا نحصل على الشكل التالي:
المسجل BL قبل تنفيذ الامر

0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

بعد عملية التنفيذ نحصل على

CF

BL

0

1	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---

ملاحظة: هذه الامر يقوم بعملية ضرب القيمة بالعدد 2 البت الاخيرة تنتقل الى الراية CF
لو اردنا ان نعمل الامر عدة مرات فاننا سوف نسجل الامر عدة مرات او نستعين في المسجل CL
مثال:

```
SHL AL,1
SHL AL,1
SHL AL,1
```

يمكن ان نسجل بدلا من هذه الاوامر ما يلي:

```
MOV CL,3
SHL AL,CL
```

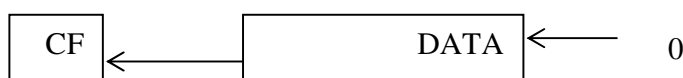
هنا تم ضرب العدد بالقيمة 8 والسبب هو ان كل ازاحة عبارة عن ضرب العدد بالقيمة 2 وهنا عملنا ثلاث ازاحات أي $2 \times 2 \times 2 = 8$ المهم ان ننتبه جيدا لانه في بعض الاحيان المسجل لا يمكنه ان يستوعب النتيجة الجديدة وهنا نخسر بتات من النتيجة وتكون النتيجة الموجودة في المسجل غير صحيحة.
الحل ان نستعين عندها في مسجل كامل مثلا

```
MOV AH,0
MOV CL,3
SHL AX,CL
```

3. ازاحة نحو اليسار للاعداد الموجهة (السالبة والموجبة).

```
SAL OP,1
```

هنا يتم اضافة صفر في بداية ال OP ونقوم بدفع البتات الاخرى الى اليسار وفي هذه الحالة تخرج البت الاخيرة الى راية CF
كما هو مبين في الشكل التالي:



مثال: معطى BL=75H اذا نفذنا الامر التالي:

```
SAL BL,1
```

اننا نحصل على الشكل التالي:
المسجل BL قبل تنفيذ الامر

0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

بعد عملية التنفيذ نحصل على

CF

BL

0

1 1 1 0 1 0 1 0

ملاحظة: هذه الامر يقوم بعملية ضرب القيمة بالعدد 2 البت الاخيرة تنتقل الى الراية CF
لو اردنا ان نعمل الامر عدة مرات فاننا سوف نسجل الامر عدة مرات او نستعين في المسجل CL
مثال:

SAL AL,1
SAL AL,1
SAL AL,1

يمكن ان نسجل بدلا من هذه الاوامر ما يلي:

MOV CL,3
SAL AL,CL

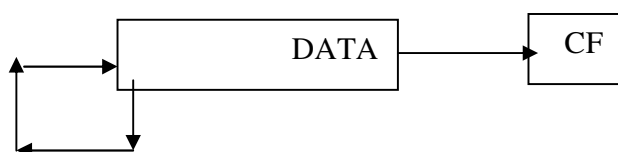
هنا تم ضرب العدد بالقيمة 8 والسبب هو ان كل ازاحة عبارة عن ضرب العدد بالقيمة 2 وهنا عملنا ثلاث
ازاحات أي $2 \times 2 \times 2 = 8$ المهم ان ننتبه جيدا لانه في بعض الاحيان المسجل لا يمكنه ان يستوعب النتيجة
الجديدة وهنا نخسر بتات من النتيجة وتكون النتيجة الموجودة في المسجل غير صحيحة.
الحل ان نستعين عندها في مسجل كامل مثلا

MOV AH,0
MOV CL,3
SAL AX,CL

4. ازاحة نحو اليمين للاعداد الموجبة والسالبة والموجبة.

SAR OP,1

هنا يتم نسخ البت الاخيرة من جهة اليسار في نهاية ال OP ونضعها في نهاية ال OP ونقوم بدفع البتات
الاخرى الى اليمين وفي هذه الحالة تخرج البت الاولى الى راية CF
كما هو مبين في الشكل التالي:



مثال: معطى BL=F5H اذا نفذنا الامر التالي:

SAR BL,1

اننا نحصل على الشكل التالي:
المسجل BL قبل تنفيذ الامر

1 1 1 1 0 1 0 1

بعد عملية التنفيذ نحصل على

1	1	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---	---

BL

CF

ملاحظة: هذه الامر يقوم بعملية قسمة القيمة على 2 ولكن دون باق أي انه يحفظ الناتج الصحيح من خارج القسمة والباقي يذهب الى الراية CF
مثال: معطى BL=75H اذا نفذنا الامر التالي:

SAR BL,1

اننا نحصل على الشكل التالي:
المسجل BL قبل تنفيذ الامر

0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

بعد عملية التنفيذ نحصل على

0	0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---	---

BL

CF

لو اردنا ان نعمل الامر عدة مرات فاننا سوف نسجل الامر عدة مرات او نستعين في بالمسجل CL
مثال:

SAR AL,1

SAR AL,1

SAR AL,1

يمكن نسجل بدلا من هذه الاوامر ما يلي:

MOV CL,3

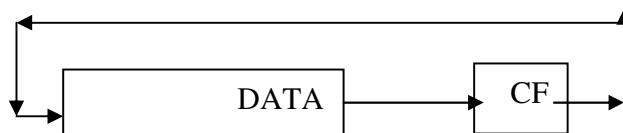
SAR AL,CL

ملاحظة: هنا لا يمكن تخزين الباقي في الراية CF لانه يمكن ان نخزن بت واحدة فقط وهي اخر بت خرجت من المسجل او المتغير. هنا تمت قسمة المسجل على القيمة 8 لانه في كل ازاحة نقسم على 2 وهنا عملنا ثلاث ازاحات اذا قسمنا على 8

5. التدوير الى اليمين مع الراية CF

RCR OP,1

هنا يتم اعتبار الراية CF من ضمن البتات التي يجب تدويرها انظر الشكل التالي:



مثال: معطى $CF=1$ $AL=3DH$
 اذا نفذنا الامر التالي:

`RCL AL,1`

قبل التنفيذ

AL CF

0	0	1	1	1	1	0	1
---	---	---	---	---	---	---	---

1

بعد التنفيذ:

AL	1	0	0	1	1	1	1	0
----	---	---	---	---	---	---	---	---

CF

1

ملاحظة: المصدر هنا عبارة عن قيمة عددية ويجب ان تكون 1 فقط واحد وفي حالة اذا اردنا تدوير اكثر من بت واحدة فهناك حلان
 الحل الاول ان نسجل الامر بقدر عدد مرات البتات المراد تدويرها مثلا:

`RCL AL,1`

`RCL AL,1`

`RCL AL,1`

فهنا تم تدوير ثلاث بتات .

حل اخر: نضع في CL عدد مرات الدوران ثما نسجل الامر
 مثال:

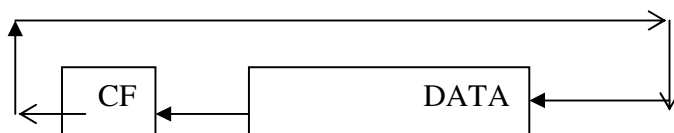
`MOV CL,3`

`RCL AL,CL`

6. التدوير الى اليسار مع الراية CF

`RCL OP,1`

هنا يتم اعتبار الراية CF من ضمن البتات التي يجب تدويرها انظر الشكل التالي:



مثال: معطى $CF=0$ $AL=3DH$
 اذا نفذنا الامر التالي:

`RCL AL,1`

قبل التنفيذ

AL

0							
0	0	1	1	1	1	0	1

بعد التنفيذ:

AL

0							
0	1	1	1	1	0	1	0

ملاحظة: المصدر هنا عبارة عن قيمة عددية ويجب ان تكون 1 فقط واحد وفي حالة اذا اردنا تدوير اكثر من بت واحدة فهناك حلان
الحل الاول ان نسجل الامر بقدر عدد مرات البتات المراد تدويرها مثلا:

RCL AL,1
RCL AL,1
RCL AL,1

فہنا تم تدویر ثلاث بتات .

حل اخر: نضع في CL عدد مرات الدوران ثم نسجل الامر
مثال:

```
MOV CL,3
RCL AL,CL
```

الاختصار	الامر
ROTATE RIGHT	ROR
ROTATE LEFT	ROL
ROTATE RIGHT WITH CARRY	RCR
ROTATE LEFT WITH CARRY	RCL
SHIFT ARITHMETIC LOGICAL LEFT	SAL
SHIFT ARITHMETIC LOGICAL RIGHT	SAR

النصوص

النص: هو عبارة عن رمز او اكثر تحفظ عادة تحت اسم في مكان معين في الذاكرة وتحفظ الرموز بجانب بعضها البعض
مثال:

```
Str1 db 'this is my first string'
Str2 db '12345'
Str3 dw 'ad'
Str4 dw 'v'
```

ملاحظة هامة : اذا كان النص من نوع word يجب ان يكون عدد رموزه اصغر من ثلاثة رموز
مثلا: str4 dw 'abc' خطأ لان عدد الرموز هنا 3 رموز

اوامر النصوص

-اوامر النصوص تعمل على منطقتين في الذاكرة
المنطقة الاولى : تدعى منطقة المصدر وهذه المنطقة معرفة بمساعدة DS:SI اين ان المصدر موجود في منطقة data segment والمؤشر الذي نستخدمه هو si
المنطقة الثانية: تدعى منطقة الهدف وهذه المنطقة معرفة بمساعدة ES:DI أي انها موجودة في منطقة extra segment والمؤشر هنا هو di
ملاحظة: يمكن ان تكون المنطقة الثانية في منطقة data segment بشرط ان نعطي المسجل es نفس قيمة المسجل ds

يجب ان يؤشر si على نص المصدر والمؤشر di على نص الهدف قبل تنفيذ عمليات النصوص
-اذا كان DF=0 المؤشرات تكبر بعد كل عملية واذا كان DF=1 المؤشرات تصغر بعد كل عملية
-الامر الذي يضع القيمة صفر في الراية DF هو الامر CLD أي (clear direction flag)
-الامر الذي يضع القيمة 1 في الراية DF هو STD أي (set direction flag)

- قيمة المؤشر si والمؤشر di تتغير بشكل تلقائي بعد كل عملية نقوم بها على النصوص بواسطة امر من اوامر النصوص.

السؤال هو بكم تتغير ؟

الجواب حسب قيمة الراية DF ونوع الامر (هنالك اوامر تتم على بايت وهنالك اوامر تتم على word)
الاور

الامر MOVSB : هنا يتم نسخ بايت من نص المصدر الى نص الهدف . ويتم تغيير المؤشرات si و di حسب وضع DF

مثال:

معطى:

```
Cld
Lea si,string1
Lea di,string2
Movsb
```

هنا تم نسخ الرمز الاول في نص المصدر (string1) الى نص الهدف (string2) وايضا تم تكبير كل واحد من المؤشرات si,di بواحد

المقطع التالي تنفذ نفس المهمة التي عملها المقطع السابق

```
Lea si,string1
Lea di,string2
Mov al,[si]
Mov [di],al
inc si
inc si
```

اذا اردنا ان ننسخ مثلا 10 رموز ماذا نعمل نسجل الامر السابق 10 مرات ام هنالك طريقة ابسط واسهل .
الجواب يمكن ان نستخدم الامر rep . (repeat)
الاستعمال كما يلي :

ضع في المسجل cx عدد الرموز المراد نسخها
نسجل rep movsb
الحل:

```
Cld
Lea si,string1
Lea di,string2
Mov cx,10
Rep Movsb
```

الامر movsw : يعمل مثل الامر السابق ولكن هنا يتم نسخ رمزين في كل عملية والمؤشرات تتقص او تزيد
ب 2 حسب الراية DF

اذا كان DF=0 عندها SI=SI+2

DI=DI+2

اذا كان DF=1 عندها SI=SI-2

DI=DI-2

الامر LOADSB : هنا يتم وضع نسخة من الرمز الذي يؤشر عليه المؤشر si في المسجل AL ويتم تغيير المؤشر SI بواحد حسب وضع DF .

اذا كان DF=0 عندها SI=SI+1

اذا كان DF=1 عندها SI=SI-1

الامر LOADSW : هنا يتم نسخ رمزين من نص الهدف الى المسجل AX ويتم تغيير المؤشر SI ب 2 حسب وضع الراية DF

اذا كان DF=0 عندها SI=SI+2

اذا كان DF=1 عندها SI=SI-2

الامر STOSB هنا يتم وضع نسخة من الرمز الموجود في المسجل AL في نص الهدف في المكان الذي
يؤشر عليه المؤشر DI ويتم تغيير المؤشر DI بواحد حسب وضع DF .

إذا كان $DF=0$ عندها $DI=DI+1$

إذا كان $DF=1$ عندها $DI=DI-1$

الامر STOSW هنا يتم نسخ المسجل AX الى نص الهدف ويتم تغيير المؤشر DI بـ 2 حسب وضع
الراية DF

إذا كان $DF=0$ عندها $DI=DI+2$

إذا كان $DF=1$ عندها $DI=DI-2$

الامر SCASB هنا يتم مقارنة AL مع الرمز الذي يؤشر عليه المؤشر DI في نص الهدف ويتم تغيير
المؤشر DI بواحد حسب وضع DF .

إذا كان $DF=0$ عندها $DI=DI+1$

إذا كان $DF=1$ عندها $DI=DI-1$

ملاحظة: هنا المصدر والهدف لا يتأثران ولكن الرايات تتأثر
أي ان فحوى المسجل AL والرمز الذي تمت مقارنته لا يتغيران

الامر SCASW هنا يتم مقارنة AX مع الرمز الذي يؤشر عليه المؤشر DI والرمز الذي يليه في نص
الهدف ويتم تغيير المؤشر DI بـ 2 حسب وضع DF .

إذا كان $DF=0$ عندها $DI=DI+2$

إذا كان $DF=1$ عندها $DI=DI-2$

ملاحظة: هنا المصدر والهدف لا يتأثران ولكن الرايات تتأثر
أي ان فحوى المسجل AX والرموز الذي تمت مقارنتها لا تتغير

الامر CMPSB هنا يتم مقارنة الرمز الذي يؤشر عليه المؤشر SI في نص المصدر مع الرمز الذي يؤشر
عليه المؤشر DI في نص الهدف ويتم تغيير المؤشر DI والمؤشر SI بواحد حسب وضع DF .

إذا كان $DF=0$ عندها $DI=DI+1$

$SI=SI+1$

إذا كان $DF=1$ عندها $DI=DI-1$

$SI=SI-1$

ملاحظة: هنا المصدر والهدف لا يتأثران ولكن الرايات تتأثر
ملاحظة: هنا يتم طرح الهدف من المصدر بعكس الامر CMP الذي يطرح المصدر من الهدف

الامر CMPSW هنا يتم مقارنة الرمز الذي يؤشر عليه المؤشر SI والرمز التالي في نص المصدر مع
الرمز الذي يؤشر عليه المؤشر DI والرمز التالي في نص الهدف ويتم تغيير المؤشر DI والمؤشر SI بـ 2
حسب وضع DF .

إذا كان $DF=0$ عندها $DI=DI+2$

$SI=SI+2$

إذا كان $DF=1$ عندها $DI=DI-2$

$SI=SI-2$

ملاحظة: هنا المصدر والهدف لا يتأثران ولكن الرايات تتأثر

ملاحظة: هنا يتم طرح الهدف من المصدر بعكس الامر `CMP` الذي يطرح المصدر من الهدف

الحلقات والنصوص

إذا اردنا تنفيذ امر من اوامر النصوص عدة مرات يمكن استخدام اوامر الحلقات
الامر `rep`

ينفذ امر نصوص عددهم المرات بقدر القيمة المخزنة في المسجل `CX`
مثلا:

`Mov cx,20`

`Rep stosb`

تمرين:

معطى نص باسم `str3` بطول 15 رمزا اكتب برنامجا ينسخ الرمز `C` في خلايا النص
الحل:

`.MODEL SMALL`

`.STACK 20H`

`.DATA`

`STR3 DB 15 DUP ('0')`

`.CODE`

`START:`

`MOV AX,@DATA`

`MOV DS,AX`

`MOV ES,AX ;ES=DS` هذا مهم حتى نقول للمعالج ان

المقطع التالي للفحص سوف نقوم بطباعة حدود النص `STR3` ;

`MOV CX,15`

`MOV SI,0`

`T11:MOV DL,STR3[SI]`

`MOV AH,2`

`INT 21H`

`INC SI`

`LOOP T11`

المقطع التالي لنسخ الرمز `C` في حدود النص `STR3` ;

`CLD`

`MOV CX,15`

`MOV AL,'C'`

`LEA DI,STR3`

REP STOSB

المقطع التالي للفحص سوف نقوم بطباعة حدود النص STR3 ;

MOV CX,15

MOV SI,0

T1:MOV DL,STR3[SI]

MOV AH,2

INT 21H

INC SI

LOOP T1

انهاء البرنامج واعطاء السيطرة لنظام التشغيل ;

MOV AH,4CH

INT 21H

END START

الامر REPZ : نفذ الحلقة كل الوقت الذي يتحقق به ما يلي:
CX<>0 وايضا ZF=1

يستخدم هذا الامر عادة للبحث عن رمز مختلف في النص.
لمعرفة اذا كان الخروج من الحلقة بسبب CX=0 أي ان النص قد انتهى نستخدم الامر JCXZ هذا يعني
اقفز اذا كان CX=0 اذا لم يتم القفز نعرف ان انتهاء الحلقة كان بسبب ZF=0 أي اننا وجدنا رمز مختلف
مثلا اذا اردنا ان نفحص اذا كان نصا معيناً يحتوي على رمز يختلف عن الرمز D

.MODEL SMALL

.STACK 20H

.DATA

STR3 DB 'DDDDADDD'

NO DB 'NOT FOUND\$'

YES DB ' FOUND\$'

.CODE

START:

MOV AX,@DATA

MOV DS,AX

MOV ES,AX ;ES=DS هذا مهم حتى نقول للمعالج ان

المقطع التالي للفحص اذا كان هنالك رمز مختلف عن D ;

CLD

MOV CX,9

MOV AL,'D'

LEA DI,STR3

REPZ SCASB

```
JCZX T1; JUMP IF CX=0 هذا يعني ان النص انتهى ولم نجد رمزا مختلفا
LEA DX,YES هنا يعني ان ايقاف عملية البحث بسبب اننا وجدنا رمزا مختلفا ;
MOV AH,9
INT 21H
JMP FINISH
T1:LEA DX,NO
MOV AH,9
INT 21H
```

انهاء البرنامج واعطاء السيطرة لنظام التشغيل ;

```
FINISH:MOV AH,4CH
INT 21H
```

```
END START
```

الامر REPNZ

نفذ الحلقة كل الوقت الذي يتحقق به ما يلي:
ZF=0 وايضا CX<>0

يستخدم هذا الامر عادة للبحث عن رمز معين في النص. مثلا اذا اردنا ان نفحص اذا كان نصا معيناً يحتوي على الرمز D.

اوامر متفرقة

هنالك بعض الاوامر المتفرقة في لغة الاسمبلي التي لا بد من التطرق اليها
الاول seg

يستخدم هذا لمعرفة عنوان المقطع الموجود فيه المتغير (variable) او العنوان المطلوب (lable)
مثال:

```
.model small
.data
x1 db 10
student db 5 dup (?)
.code
start:
    mov ax,@data
    mov ds,ax
```

هنا نرى تم تخزين عنوان بداية مقطع البيانات data في مسجل المقطع ds
ملاحظة: @data عبارة عن عنوان اي قيمة عددية تدل على بداية مقطع البيانات data segment وكما
نعلم لا يحق ان نخزن قيمة عددية بصورة مباشرة في مسجل مقطع لذلك استخدمنا المسجل ax وضعنا القيمة
العددية في داخله ثم نسخنا فحوى المسجل ax الى المسجل ds

يمكن ان نحصل على نفس النتيجة بمساعدة seg

```
.model small
.data
x1 db 10
student db 5 dup (?)
.code
start:
    mov ax,seg x1
    mov ds,ax
```

لقد حصلنا على عنوان مقطع البيانات عن طريق اسم متغير في داخله و seg

الامر xlat

في بعض الاحيان نريد ان نترجم جدول معين الى جدول اخر (مثلا لو اردنا ان نغير قيمة الاسكي كود في
الرمز الذي يلائمه) في هذه الحالات نستخدم الامر XLAT
قبل استخدام الامر XLAT علينا مراعاة الشروط التالية:
1. علينا تخزين البايت المراد ترجمته في المسجل AL وفقط في هذا المسجل
2. المسجل BX يجب ان يحتوي على عنوان جدول الترجمة

ثم نسجل الامر XLAT عندها يقوم هذا الامر بالامور التالية:
1. يجمع فحوى المسجل AL والمسجل BX وعندها نحصل على عنوان داخل جدول الترجمة (اي اننا
نؤشر على بايت معين في جدول الترجمة والذي عنوانه عبارة عن BX+AL
2. يضع فحوى البايت الذي عنوانه BX+AL في المسجل AL

مثال: في هذا المثال يوجد جدولين (مصفوفتين) فسوف نقوم بترجمة حدود المصفوفة الاولى (الجدول الاول)
عن طريق جدول الترجمة الجدول الثاني:

```
. model small
.stack 10h
.data
table1 db 1,13,4,1,2,8,13,14,11,8,13,3,5,5,8,7,10,4,6,13,4,3,9,12,0,3,12,8
table2 db 'u','i','k','a','l','s','y','m','e','n','b','h','g',' ','t'
.code
start:mov ax,@data
    mov ds,ax
```

هنا سوف نقوم بمسح الشاشة;

```

mov ax,3
int 10h

```

نضع في المسجل bx عنوان جدول الترجمة

```

lea bx,table2

```

المسجل si سوف نستعمله كمؤشر على حدود الجدول الاول للانتقال بين الحدود

```

mov si,0

```

هنا المسجل cx يحتوي على عدد حدود الجدول الذي نريد ان نترجمه;

```

mov cx,28

```

هنا سوف تتم عملية الترجمة لكل حد ومن ثم طباعة الحد الذي يلائمه في الجدول الثاني

```

next_item:mov al,table1[si]
xlat
mov dl,al
mov ah,2
int 21h
inc si
loop next_item

```

انهاء البرنامج واعطاء نظام التشغيل السيطرة

```

mov ah,4ch
int 21h
end start

```

الامر in (input) والامر out (output)

من المعروف انه يمكن ربط المعالج 8086 مع 64 كيلو اجهزة ادخال واخراج داخلية. لكل واحد يوجد عنوان حتى يتمكن المعالج من التمييز بينها.

يمكن ادخال واخراج بيانات فقط عن طريق المسجل AL او المسجل AX

عند استخدام الامر in او الامر OUT يجب ان نسجل عنوان الجهاز المطلوب

امثلة:

In AL,50H

هذا يعني ادخل الى المسجل AL فحوى جهاز الادخال الذي عنوانه 50H

OUT 50H,AL

هذا يعني اخرج فحوى المسجل AL لجهاز الاخراج الذي عنوانه 50H

IN AX,80H

هذا يعني ادخل فحوى جهاز الادخال صاحب عنوان 80H للمسجل AL وفحوى جهاز الادخال صاحب عنوان 81H للمسجل AH

OUT 90H,AX

هذا يعني اخرج فحوى المسجل AL لجهاز الاخراج صاحب العنوان 90H وفحوى المسجل AH لجهاز الاخراج صاحب عنوان 91H

في حالة اذا كان عنوان الجهاز المطلوب اكبر من 255 اي FFH عندها نستخدم المسجل DX كمؤشر مثال:

```
MOV DX,301H
IN AL,DX
```

هنا يتم ادخال فحوى جهاز الادخال صاحب عنوان 301H الى المسجل AL

الامر PUSHF والامر POPF
الامر PUSHF : يقوم بتخزين فحوى مسجل الرايات في مقطع المكعدة (STACK SEGMENT)

الامر POPF : يقوم هذا الامر بنسخ الكلمة الموجودة في راس المكعدة الى مسجل الرايات
يمكن استخدام هذه الاوامر عندما نريد ان نسأل عن قيمة راية معينة بدون ان نعمل عملية تؤثر على الراية وبقية الرايات
مثال:

```
PUSHF
POP AX
```

والان يمكن التوجه الى اي راية نريدها عن طريق المسجل AX
وايضا يمكن استخدام هذه الاوامر اذا اردنا ان نعطي الرايات قيمة بدائية. نضع مثلا في المسجل BX القيم المطلوبة ثم نخزنها في المكعدة وبعدها ننسخها الى مسجل الرايات
مثلا:

```
MOV BX,0000000001010001B
PUSH BX
POPF
```

هنا تم التحكم في الرايات $CF=AF=ZF=1$ وباقي الرايات قيمتها صفر

الى اللقاء في الدرس القادم

السلام عليكم

الماكرو

الماكرو : هو عبارة عن مجموعة اوامر لها اسم ويمكن استدعاء هذه مجموعة الاوامر عن طريق تسجيل اسم الماكرو. يمكن استدعاء الماكرو اكثر من مرة في البرنامج. فيمكن القول ان لاستخدام الماكرو عدة فوائد منها:

1. تقليل الاخطاء فيمكن تسجيل اسم الماكرو بدلا من كتابة الاوامر مرة ثانية وهذا يقلل من الوقوع في اخطاء املائية

2. زيادة البرنامج وضوحا . فبدلا من كتابة اوامر فاننا نسجل كلمات عادية مفهومة وهي اسماء الماكرو وتدل على ما يعملها هذا الماكرو

3. سرعة التغيير: اذا اردنا ان نضيف او نحذف او امر معينة فبدلا من اجراء هذه التغييرات في جميع الاماكن التي تتواجد بها هذه الاوامر فعند استخدام الماكرو يكفي ان نغير في الماكرو.

4. زيادة سرعة البرنامج لان الماكرو يزرع الاوامر التي يحتويها في المكان الذي نستدعيه فيه ملاحظة: الماكرو يكبر حجم البرنامج وهذا يعد صفة سلبية للماكرو

انواع الماكرو

يمكن تقسيم الماكرو حسب عدة مواصفات نأخذ التقسيم الاول

1. هنالك انواع لا تأخذ برامترات

2. هنالك انواع تأخذ برامترات

ماكرو بدون برامترات

المبنى العام للماكرو الذي لا يأخذ برامترات هو كما يلي:

Macro name macro

الاوامر

Endm

مثال : اكتب ماكرو يقوم بمسح الشاشة:

الحل:

Clear_screen macro

Mov ax,3

Int 10h

Endm

مثال اخر

اكتب ماكرو يطبع يستقبل من لوحة المفاتيح رمزا

الحل:

Read_char macro

Mov ah,1

Int 21h

Endm

مثال اخر

اكتب ماكرو يطبع نجمة على الشاشة

الحل:

Print_star macro

Mov dl,'*'

Mov ah,2

Int 21h

Endm

ملاحظات: عليك الانتباه الى ما يلي:

1. عليك ان تحفظ قيم المسجلات التي سوف تستخدمها في الماكرو اذا كانت تحتوي على قيم مهمة بالنسبة لك وانت بحاجة اليها لان اوامر الماكرو سوف تغير القيم. هنالك عدة طرق لحفظ قيم المسجلات نذكر منها:

أقبل استدعاء الماكرو نقوم بتخزين القيم في مسجلات ثانية او نخزنها في الذاكرة وبعد الانتهاء من الماكرو نرجع القيم

مثال:

```
Mov bx,ax
Clear_screen
Mov ax,bx
```

نلاحظ هنا اننا قمنا بحفظ قيمة المسجل ax في المسجل bx قبل استدعاء الماكرو وبعد استدعاء الماكرو تمت عملية ارجاع القيمة الى المسجل ax
ب. يمكن ان نحفظها في داخل الماكرو قبل القيام في الاوامر الرئيسية مثلاً

```
Clear_screen macro
Push ax
Mov ax,3
Int 10h
Pop ax
Endm
```

نلاحظ في التمرين السابق اننا حفظنا الماكرو في المكعدة (stack segment) قبل ان نقوم بعملية مسح الشاشة وقبل الخروج من الماكرو تمت عملية ارجاع القيم من المكعدة
2. يمكن استدعاء ماکرو من داخل ماکرو آخر
مثال:

طور ماکرو يقوم بمسح الشاشة واستقبال رمز من لوحة المفاتيح ثم يطبع الرمز
الحل:

```
Print_mac macro
Clear_screen
Read_char
Mov dl,al
Mov ah,2
Int 21h
Endm
```

هنا تم استدعاء الماكرو clear_screen والماكرو readl_char
3. عندما نستخدم LEBLE (عنوان) في الماكرو يجب ان نعرفها بمساعدة الكلمة المحفوظة LOCAL أي اننا نقول لبرنامج الترجمة بان هذا العنوان هو عبارة عن عنوان محلي معروف في الماكرو فقط .
ملاحظة: عند استخدام عنوان والتوجه اليه اكثر من مرة عن طريق اوامر الماكرو او استدعاء الماكرو اكثر من مرة فان برنامج الترجمة سوف يعلن عن خطأ
مثال:

اكتب ماکرو يطبع الارقام من 0 الى 9
الحل:

```
Print_numbers macro
Local t1
Mov cx,10
Mov dl,'0'
T1:mov ah,2
Int 21h
```

```
Inc dl
Loop t1
Endm
```

ملاحظة: اذا كان لدينا اكثر من عنوان محلي في الماكرو نسجل العناوين ونفصل بين العناوين بمساعدة فاصلة

```
Local t1,t2,t3
```

4. للخروج من الماكرو من أي نقطة نريدها أي قبل نهاية الماكرو نستخدم الامر exitm مثال:

```
Check macro
Local t1,t2
Mov cx,10
T1: Mov ah,1
     Int 21h
     Cmp al,'B'
     Jz t2
     Inc dh
     exitm
T2: dec dh
     loop t1
endm
```

حل اخر بدون استخدام exitm

```
Check macro
Local t1,t2,t3
Mov cx,10
T1: Mov ah,1
     Int 21h
     Cmp al,'B'
     Jz t2
     Inc dh
     Jmp t3
T2: dec dh
     loop t1
t3: nop
endm
```

5. اذا اردنا ان نسجل ملاحظات في داخل الماكرو نستخدم الفاصلة المنقوطة مرتين مثال:

```
Print_char macro
Mov ah,1 ;;put 1 in ah
Int 21h
```

```

Mov dl,al ;;put the ascii_code of the char in dl
Mov ah,2
Int 21h
Endm

```

اكتب برنامجا ينفذ ما يلي:

1. ينظف الشاشة
2. يطبع على الشاشة الجملة enter a char
3. يستقبل رمزا من لوحة المفاتيح
4. ينزل سطر جديد
5. اذا كان الرمز عبارة عن رقم [0..9] عندها يطبع الرمز وينتهي البرنامج والا ينظف الشاشة ويطبع نجمة ويرجع الى الفرع 2

الحل:

```

.model small
.stack 20h
.data
NL db 10,13,'$'
msg db 'enter a char$'
clear_screen macro
mov ax,3
int 10h
endm
print_char macro
mov dl,al
mov ah,2
int 21h
endm
read_char macro
mov ah,1
int 21h
endm
new_line macro
lea dx,NL
mov ah,9
int 21h
endm
print_msg macro
lea dx,msg
mov ah,9

```

```

int 21h
endm
.code
start: mov ax,@data
      mov ds,ax
clear_screen
t2:print_msg
   read_char
   mov bl,al
   new_line
   sub bl,30h
   cmp bl,9
   ja t1
   add bl,30h
   mov al,bl
   print_char
   jmp finish
t1:clear_screen
   mov al,'*'
   print_char
   jmp t2
finish:mov ah,4ch
       int 21h
       end start

```

السلام عليكم
هذا الدرس الجزء الثاني من درس الماكرو
الماكرو الذي يأخذ برامترات
المبنى العام للماكرو مع برامترات

Macro name macro p1,p2,....,pn
الاوامر
Endm

P1,p2,...pn : هي البرامترات
مثال :

اكتب ماكرو يتلقى برامترين من نوع byte ويحفظ بالمسجل al البرامتر الاكبر

Max_num macro x,y
Local t1,finish
Mov al,x
Cmp y,al

```

Jg t1
Jmp finish
T1:mov al,y
Finish:
Endm

```

ملاحظة: عندما نريد استخدام الماكرو في موقع معين في البرنامج نسجل اسم الماكرو ونسجل البرامترات المطلوبة.

مثلا في حالة استدعاء الماكرو السابق نسجل

```
Max_num DH,DL
```

هنا تم اعطاء الماكرو المسجلات DH و DL كبرامترات

```
Max_num 7,9
```

هنا تم اعطاء الماكرو القيم 7 ، 9 كبرامترات

```
Max_num x1,x2
```

هنا تم اعطاء الماكرو متغيرين x1 , x2 كبرامترات

في كثير من الحالات يفضل استخدام الماكرو مع برامترات لانه يعطينا امكانية استخدام الماكرو لعدة عمليات.

مثال: معطى في مقطع البيانات

```

.data
NL db 10,13','$'
Msg1 db 'enter the first number....$'
Msg2 db 'enter the second number..$'
Msg3 db 'your number is out the range$'

```

الان نسجل الماكرواات المطلوبة ولكن بدون برامترات
هذا الماكرو مسؤول عن النزول الى سطر جديد

```

New_line macro
Lea dx,NL
Mov ah,9
Int 21h
Endm

```

هذا الماكرو مسؤول عن طباعة الجملة الاولى

```

Print_msg1 macro
Lea dx,msg1
Mov ah,9
Int 21h
Endm

```

هذا الماكرو مسؤول عن طباعة الجملة الثانية

```

Print_msg2 macro
Lea dx,msg2
Mov ah,9

```

```
Int 21h
Endm
```

هذا الماكرو مسؤول عن طباعة الجملة الثالثة

```
Print_msg3 macro
Lea dx,msg3
Mov ah,9
Int 21h
Endm
```

وفي داخل البرنامج نسجل

```
New_line
Print_msg1
Print_msg2
Print_msg3
```

نلاحظ من العمليات السابقة اننا بحاجة الى ماكرو لكل عملية
الحل الامثل هو استخدام ماكرو مع برامترات
الحل:

```
Print_mac macro x
Lea dx,x
Mov ah,9
Int 21h
Endm
```

وفي داخل البرنامج نسجل

```
Print_mac NL
Print_mac msg1
Print_mac msg2
Print_mac msg3
```

كما نلاحظ انه بمساعدة ماكرو واحد من نوع ماكرو مع برامترات تم اجراء جميع العمليات السابقة.
الان ننتقل الى نوع جديد وهو الماكرو المتكرر
وهو ماكرو يقوم بتنفيذ مجموعة الاوامر الموجودة فيه عدد مرات معروفة مسبقا
يسجل هذا الماكرو بشكل عام داخل مقطع البيانات
المبنى العام لهذا الماكرو

```
Rept n
الوامر
Endm
```

بحيث n عبارة عن قيمة عددية او تعبيراً يرجع قيمة عددية او اسم ثابت من الثوابت المعرفة في البرنامج او
امر يرجع قيمة عددية . باختصار المهم ان يكون قيمة عددية

مثال:
اكتب ماكرو يطبع النجمة 5 مرات

```
Rept 5  
Mov dl,'*'  
Mov ah,2  
Int 21h  
Endm
```

سؤال: ماذا ينفذ الماكرو التالي:

```
Rept 4  
Add ax,2  
Endm
```

الجواب: الماكرو السابق يمكن تغييره بالاوامر التالية

```
Add ax,2  
Add ax,2  
Add ax,2  
Add ax,2
```

أي انه يجمع 8 للمسجل ax
المثال التالي يقوم بطباعة الاحرف الانجليزية على نفس السطر بمساعدة الماكرو المتكرر.

```
.model small  
.stack 20h  
.data  
Clear_screen macro  
    Mov ax,3  
    Int 10h  
Endm  
.code  
start: mov ax,@data  
    mov ds,ax  
    clear_screen  
    mov dl,'A'  
    rept 26  
    mov ah,2  
    int 21h  
    inc dl  
    endm  
finish:mov ah,4ch  
    int 21h  
end start
```

الان ننتقل الى نوع اخر من الماكرو وهو ايضا ماكرو متكرر ولكن هنا يتم استخدام جميع البرامترات في كل دورة نستخدم برامتر الذي يصل اليه الدور

```
Irp name1, <arg1,arg2,arg3,....,argN>  
مجموعة الاوامر المراد تكرارها  
Endm
```

مثال:

```
Irp num, <10,20,30>  
Add ax,num  
Endm
```

الماكرو السابق ينفذ الاوامر التالية

```
Add ax,10  
Add ax,20  
Add ax,30
```

بالاختصار تم اضافة 60 للمسجل ax
مثال اخر

```
Irp push_reg, <ax,bx,cx,dx>  
Push push_reg  
Mov push_reg,0  
End
```

الماكرو السابق ينفذ الاوامر التالية:

```
Push ax  
Mov ax,0  
Push bx  
Mov bx,0  
Push cx  
Mov cx,0  
Push dx  
Mov dx,0
```

الدوال

تستخدم الدوال لتسهيل عملية البرمجة واجراء التغييرات السريعة وتقليل نسبة الوقوع في الازغاء الكتابية .
انظر مادة الماكرو

الفرق بين الدالة والماكرو

1. الماكرو يؤدي الى سرعة البرنامج بينما الدوال تخفف من سرعة البرنامج لان المعالج يقوم بالتوجه الى اوامر الدالة بعد استدعائها بينما الماكرو يزرع اوامره في الاماكن المطلوبة قبل ان يتوجه اليه المعالج لتنفيذ اوامره أي انها جاهزة للتنفيذ

2. الماكرو يكبر من حجم البرنامج لانه يزرع الاوامر من البداية بينما الدالة فقط عند الاستدعاء تقوم بزرع الاوامر

3. الدالة بحاجة الى مكس لحفظ عنوان العودة بينما الماكرو ليس بحاجة الى المكس

الامر jmp هو امر يؤدي الى القفز لعنوان معين في البرنامج بدون رجوع ولكن عند استدعاء دالة معينة فهنا يتم القفز الى مجموعة اوامر بحيث يقوم المعالج بتنفيذها وبعد الانتهاء من تنفيذ هذه الاوامر يتم الرجوع الى الامر الموجود بعد الامر الذي ادى الى استدعاء الدالة.

الدالة: هي عبارة عن مقطع برنامج يتم الانتقال اليه عندما نقوم باستدعاء الدالة وفي نهاية المقطع يجب ان يكون هنالك امر انتهاء للمقطع الذي يرجع المعالج الى الامر الذي يلي امر الاستدعاء لذلك يجب على المعالج ان يحفظ عنوان الامر الذي يجب الرجوع اليه قبل ان يقوم بعملية القفز. يخزن عنوان الرجوع في المكس stack segment .

الامر : call بمساعدة هذا الامر يتم استدعاء الدالة.
المبنى العام للامر:
Call اسم الدالة

الامر call يضع عنوان الرجوع في المكس (stack segment). عنوان الرجوع بطول 16bit او 32bit حسب نوع الدالة.

المسجلات ip أو cs ip تأخذ عنوان الدالة المراد تنفيذها عند استدعاء الدالة
ملاحظة: الامر call لا يؤثر على مسجل الرايات

الامر ret
الامر ret يستخدم هذا الامر للرجوع من الدالة.
مبنى الامر ret

Ret n
الرمز n يمثل عدد البايتات (bytes) التي يجب تنظيفها في المكس عند الرجوع من الدالة.
المقصود بعملية التنظيف هو القفز عن البايتات وذلك عن طريق تغيير رأس المكس المسجل SP
ملاحظة: اذا اردنا ان لا نقوم بعملية تنظيف بايتات في المكس نسجل الامر ret بدون n
ملاحظة: الامر ret لا يؤثر على مسجل الرايات

انواع الدوال

هنالك نوعان من الدوال

1. دالة قريبة: وهي دالة الموجودة في نفس مقطع الاوامر (code segment) الرئيسي
2. الدالة البعيدة: وهي الدالة الموجودة في مقطع اوامر اخر .

المبنى العام لدالة قريبة

Function_name proc near

اوامر الدالة

Function_name endp

ملاحظة: يمكن الاستغناء عن الكلمة near

Function_name proc

اوامر الدالة

Function_name endp

مثال:

في 1 وتفحص اذا كانت النتيجة سالبة اذا تحقق الشرط تضع al للمسجل 10 أكتب دالة قريبة التي تضيف b1 والا فانها تضع صفر في المسجل b1 المسجل
الحل:

```
Add_num proc near
Add al,10
Cmp al,-1
Jg t1
Mov bl,1
Jmp t2
T1:mov bl,0
T2: Ret
Add_num endp
```

مثال اخر

اذا تحقق الشرط 50H وتفحص اذا كانت النتيجة اكبر من 10 في العدد b1 أكتب دالة التي تضرب المسجل من 4 bytes عليك تنظيف . dl في المسجل 0 والا فانها تعوض dl في المسجل 2 تضع في القيمة في نهاية الدالة stack المكس
الحل:

```
mul_num proc near
mov al,bl
mov cl,10
imul cl
cmp ax,50H
Jg t1
Mov dl,0
Jmp t2
T1:mov dl,2
T2: Ret 4
mul_num endp
```

المبنى العام لدالة بعيدة

```
Func_name proc far
اوامر الدالة
Func_name endp
```

اعطاء البرامترات للدوال

يمكن اعطاء الدالة برامترات في شكلين

في هذه الحالة نعطي الدالة صورة من المتغير وتقوم الدالة باجراء العمليات الحسابية : by value أ. والتغييرات على النسخة وليس على الاصل وفي هذه الحالة قيمة المتغير الاصلية لا تتغير

في هذه الحالة نعطي الدالة عنوان المتغير فعندما تقوم الدالة باجراء تغييرات على : by reference ب. المتغير فانها تؤثر على قيمة المتغير الاصلية ايضا

طرق ارسال القيم للدالة

يمكن ارسال قيم للدالة بطريقتين

في هذه الحالة قبل استدعاء الدالة نقوم بحفظ القيم في مسجلات معينة وبعدها نستدعي : بمساعد مسجلات. 1. الدالة وتتوجه الدالة الى هذه المسجلات وتأخذ القيم المطلوبة

مثال:

BX اكتب دالة تقوم بضرب قيمتين وحفظ النتيجة في المسجل

الحل:

```
mul_numbers proc near
mov al,bl
imul cl
mov bx,ax
Ret
mul_numbers endp
```

bl , cl هنا نرى ان الدالة سوف تأخذ القيم من المسجلات
في البرنامج الرئيسي نسجل مثلا ما يلي

```
Mov bl,5
Mov cl,8
Call mul_numbers
```

وتم ارسال البرامترات للدالة عن طريق المسجلات 8 بالقيمة 5 فهنا استخدمنا الدالة السابقة لضرب القيمة

هنا قبل استدعاء الدالة نقوم بارسال القيم الى المكس وبعدها نستدعي الدالة ونقوم : (stack) المكس. 2. mov والامر bp الدالة باخذ القيم من المكس بمساعدة المسجل

يقوم باستدعاء الدالة المطلوبة وايضا يقوم بادخال عنوان الرجوع ايضا في call تذكر ان الامر : ملاحظة
اذا القيم موجودة في الخلايا التي (دالة بعيدة) بت 32 او (دالة قريبة) بت 16 المكس وهذا العنوان بطول
تحت الخلايا التي تحتوي على عنوان العودة او الرجوع

تمرين:

عن طريق المكس تقوم الدالة بجمع هذه (word) اكتب دالة قريبة التي تتلقى قيمتين كل قيمة بكبر كلمة
DX . القيم وتحفظ الدالة النتيجة في المسجل

على الدالة ان تخرج هذه القيم من منطقة المكس في نهاية الدالة : ملاحظة

الحل:

```
Test_proc proc near
```

```

Mov bp,sp
Add bp,2
Mov ax,[bp]
Add bp,2
Mov dx,[bp]
Add dx,ax
Ret 4
Test_proc endp

```

تمرين:

اكتب برنامجا كاملا يحتوي على دالتين الدالة الاولى لتنظيف الشاشة الدالة الثانية تتلقى قيمتين عن طريق
 .DX المكس وتقوم بجمع القيمتين وحفظ النتيجة في المسجل
 :الحل

```

.model small
.stack 20h
.Data
.code
    Start:mov ax,@data
           Mov ds,ax
           call clean
           Mov bx,3
           Push bx
           Mov bx,6
           Push bx
           Call test_proc

           Mov ah,4ch
           Int 21h
clean proc near
mov ax,3
int 10h
ret
clean endp
Test_proc proc near
Mov bp,sp
Add bp,2
Mov ax,[bp]
Add bp,2
Mov dx,[bp]
Add dx,ax

```

Ret 4
Test_proc endp
End start

العمليات الحسابية على النصوص

كود BCD

BCD-CODE: binary coded decimal code

معظم المدخلات العددية هي من نوع نص والتي تمثل اعداد عشرية. احدى الطرق لمعالجة هذه المدخلات هي ترجمة هذا النص لعدد في الميزان الثنائي وبعدها القيام بالعمليات الحسابية المطلوبة واذا اردنا ان نعرض النتيجة نقوم بتحويل النتيجة من قيمة عددية الى نص ثم نعرض النص على الشاشة مثلاً. عملية الترجمة المزدوجة من نص الى قيمة عددية ومن قيمة عددية الى نص بحاجة الى وقت أي اننا نخسر وقتاً طويلاً في عمليات الترجمة. طريقة ثانية لمعالجة النصوص هي ان نقوم باجراء العمليات الحسابية على النصوص وبعدها نقوم بتصحيح النتيجة بمساعدة امر معين للحصول على شكل قيمة عشرية. حتى يكون التصحيح بسيطاً قاموا بانشاء كود BCD ما يميز هذا الكود هو ان:

الرموز المستعملة في هذا الكود هي 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 هنالك نوعان لتمثيل لهذا الكود

1. packed BCD : في هذه الحالة كل بايت يحتوي على خانتين أي كل خانة بحاجة الى 4 بتات

2. unpacked BCD : في هذه الحالة كل بايت يحتوي على خانة واحدة فقط أي كل خانة بحاجة الى 8 بتات

اليكم الجدول التالي الذي يظهر بعض الاعداد في الميزان العشري وكود BCD

الميزان العشري	Packed BCD	Unpacked BCD
0	0H	00H
1	1H	01H
2	2H	02H
3	3H	03H
4	4H	04H
5	5H	05H
6	6H	06H
7	7H	07H
8	8H	08H
9	9H	09H

0101H	11H	11
0102H	12H	12
0508H	58H	58
020105H	215H	215

عمليات تصحيح النتيجة

الان ننظر الى العمليات الحسابية التالية:

الميزان العشري	BCD-code
12	12H
+11	+11H
---	----
23	23H

النتيجة السابقة صحيحة اذا في هذه الحالة النتيجة ليست بحاجة الى تصحيح
ننظر الى العملية التالية:

الميزان العشري	BCD-code
12	12H
+19	+19H
---	----
31	2BH

اذا نظرنا الى النتيجة فان النتيجة تحتوي على رمز B وهذا الرمز غير مسموح فيه في BCD-
CODE
الحل:

اذا اضعنا القيمة 6 الى الخانة الغير منطقية في النتيجة تتم عملية التصحيح انظر الحل:
 $2BH + 06H = 31H$

انظر الى التمرين التالي:

الميزان العشري	BCD-code
18	18H
+19	+19H
---	----
37	31H

إذا نظرنا إلى النتيجة نرى أن خانة النتيجة من الرموز المسموح فيها ولكن النتيجة بحاجة إلى تعديل للحصول على الشكل المطلوب.
الحل أن نضيف للنتيجة 6

$$31H + 06H = 37H$$

السبب هنا هو أن الخانة الأولى بعد عملية الجمع أعطت الخانة الثانية القيمة 1. أي أن $AF=1$ (انظر مسجل الرايات)

بالاختصار هنالك حالتان يجب أن نصحح النتيجة فيهما

1. إذا كانت النتيجة تحتوي على رمز ليس من الرموز المسموح فيها في BCD-CODE
بالاختصار إذا كانت إحدى الخانات أكبر من 9

2. إذا كان $AF=1$

لتصحيح النتيجة نستخدم الأوامر التالية:

ملاحظة: من الأفضل أن نطلب تصحيح النتيجة بعد كل عملية لأننا لا نعرف إذا كنا بحاجة أو لا ولكن المعالج يمكنه أن يميز إذا نحن لسنا بحاجة إلى فحص النتيجة وبعدها نقرر وإنما نطلب من المعالج أن يصحح بعد كل عملية وهو الذي يقرر إذا كانت هنالك حاجة أم لا

1) AAA

ASCII adjust after addition

يقوم هذا الأمر بتصحيح النتيجة الموجودة في المسجل AL إلى الشكل العشري في كود

unpacked BCD وتضع النتيجة الجديدة في المسجل AX

يستخدم هذا الأمر بعد عملية جمع بين قيمتين من نوع unpacked BCD أو بكود ASCII

مثال:

MOV ax,0

Mov al,5

Add al,8

AAA

بعد هذه العملية

AX=0103H

تمرين آخر:

Mov ax,0108H

add al,'7'

AAA

بعد هذه العملية:

AX=0205H

2)DAA

decimal adjust after addition

يقوم هذا الامر بتصحيح النتيجة الموجودة في المسجل AL لخانتين في packed BCD .
نستخدم هذا الامر بعد عملية جمع بين قيمتين من نوع packed BCD
مثال:

```
Mov ax,0  
Mov al,45H  
Add al ,28H  
DAA
```

بعد هذا المقطع

AX=0073H

3)AAS

ASCII ADJUST AFTER SUBTRACTION

هذا الامر يقوم بتصحيح النتيجة في المسجل AL لعدد عشري في هيئة unpacked BCD
نستخدم هذا الامر بعد عملية طرح بين قيمتين من نوع
ASCII-code او unpacked BCD
مثال:

```
Mov ax , 0105H  
Sub al,08H  
AAS
```

بعد هذا المقطع

AX=0007

4)DAS

decimal adjust after subtraction

يقوم هذا الامر بتصحيح النتيجة الموجودة في المسجل AL لخانتين عشرية في هيئة packed BCD .
نستخدم هذا الامر بعد عملية طرح بين قيمتين من نوع packed BCD
مثال:

```
Mov ax,0  
Mov al,45H  
sub al ,28H  
DAS
```

بعد هذا المقطع

AX=0017H

5)AAM

ASCII adjust after multiply

هذا الامر يقوم بتصحيح القيمة الموجودة في المسجل AL الى خانتين في هيئة unpacked BCD . نتيجة التصحيح تخزن في المسجل AX كله بحيث خانة العشرات تخزن في المسجل AH وخانة الاحاد في المسجل AL .
نستخدم هذا الامر بعد عملية الضرب

Mov al,04H

Mov CL , 08H

MUL CL

AAM

بعد هذا المقطع

AX=0302H

ملاحظة:نستخدم هذا الامر احيانا بدون أي صلة للضرب لتحويل عدد من الميزان السادس عشر الى عدد في كود unpacked BCD بشرط ان لا يكون العدد اكبر من 99
مثال:

Mov ax,0013H

AAM

هنا نحصل على

AX=0109H

6)AAD

ASCII adjust for Division

يقوم هذا الامر بتصحيح القيمة الموجودة في المسجل AX والتي هي على unpacked BCD قبل عملية القسمة الى قيمة في الميزان السادس عشر hex .
مثال:

Mov CL,06H

Mov ax,0109H

AAD

DIV CL

بعد الامر AAD تصبح قيمة المسجل AX=0013H
وبعد عملية القسمة DIV CL
AL=03 , AH=01

أي ان AX=0103H

ملاحظة: نستخدم هذا الامر احيانا بدون أي صلة للقسمه لتحويل عدد في كود unpacked BCD الى عدد في الميزان السادس عشر بشرط ان لا يكون العدد اكبر من 99 مثال:

Mov ax,0201H

AAD

هنا نحصل على

AX=0015H

Record

في بعض المرات تكون الاجابة على سؤال معين نعم او لا امثلة:
هل انت متزوج؟

هل يجب ان تلبس النظارات اثناء السياقة؟

فلتخزين الجواب تكفي بت واحدة فقط اذا كانت مثلا القيمة لهذه البت 1 تمثل الجواب نعم واذا كانت القيمة صفر فالجواب كلا

واحيانا تكون القيمة المراد تخزينها اقل من بايت (8 بتات) مثلا

كم عدد الاولاد في الصف ففي بايت واحد يمكن تخزين 255 ولا يمكن ان يكون مثلا في مدرسة ابتدائية او اعدادية او ثانوية عدد طلاب صف بهذا العدد

اذا عندما نريد ان نخزن بيانات معينة ويمكن تخزينها كل واحدة منها بعدد معين من البتات نستعمل ال record

في لغة الاسمبلي هنالك نوعان من ال record من ناحية الكبر

1. record بطول 8 بتات (8 bits)

2. record بطول 16 بت (16 bits)

اهمية استخدام ال record

1. توفير ذاكرة

2. تقليل حجم البيانات المخزنة مما يؤدي الى سرعة تحميل (upload) قاعدة البيانات المخزنة فيها هذه البيانات

3. تسريع عمليات البحث والاستفسارات وجلب المعطيات لصغر حجم قاعدة البيانات
المبنى العام لتعريف record

قبل استخدامه لتعريف متغيرات من نوع RECORD يجب تعريف النمط المطلوب

Record_name record field1_name:bits_num,field2_name:bits_num.....

تعريف متغير من نوع record

Var_name record_name <قيمة الحقل الثاني , قيمة الحقل الاول>

مثال:

لدينا مجموعة من الطلاب تتكون من 20 طالبا نريد ان نخزن المعلومات التالية عن كل طالب.

1. كود الطالب (رقم خاص بالطالب)

2. هل لديه رخصة سياقة ام لا

3. من أي شعبة هو (تاريخ، فيزياء، كيمياء، حاسوب ، علم اجتماع) أي انه يوجد في المدرسة 5 تخصصات. ملاحظة: كل طالب يمكن ان ينتمي الى شعبة واحدة فقط

4. عدد اللغات التي يتقنها من بين هذه اللغات (انجليزية، عربية، فرنسية، اسبانية)

5. عدد المواضيع التي يدرسها اذا علم ان الطالب يدرس من 8 الى 11 موضوعا

تحليل قبل الحل:

لو اردنا استخدام الطريقة السابقة أي بدون استخدام record فنحن بحاجة الى ما يلي:

المهمة	عدد البايتات (bytes)
كود الطالب	1
رخصة سياقة	1
الشعبة	1
عدد اللغات	1
عدد المواضيع	1

اذا نحن بحاجة هنا الى 5 byte تخزين معلومات عن كل طالب

الحل بمساعدة الطريقة الجديدة:

هنا يجب ان نفكر قبل ان نشتغل:

كود الطالب:

في الفرقة يوجد 20 طالب . يجب ان نعطي لكل طالب رقم خاص ويختلف عن ارقام الطلاب حتى نستطيع

ان نميز الطالب. يمكن ان نعمل ما يلي:

الطالب الاول نعطيه الرقم صفر

الطالب الثاني نعطيه الرقم 1

الطالب الثالث نعطيه الرقم 2

اذا الطالب الاخير (الطالب العشرون) نعطيه الرقم 19 السؤال التالي هو كم بتا (bits) نحن بحاجة لكتابة

اكبر كود لدينا وهو الرقم 19 الجواب علينا ان نحول القيمة 19 في الميزان العشري الى الميزان الثنائي

$$19 = (10011)_2$$

اذا نحن بحاجة الى 5 بتات (bits)

الان بالنسبة لرخصة السياقة تكفي بت واحدة (صفر تعني لا يوجد لديه رخص 1 تعني نعم يوجد لديه

رخصة)

بالنسبة للشعبة نعمل ما يلي:

نرمز لكل شعبة برقم

الشعبة	الرقم
تاريخ	0
فيزياء	1
كيمياء	2
حاسوب	3
علم اجتماع	4

اذا اكبر قيمة نريد تسجيلها حسب الجدول هي 4 لمعرفة عدد البتات المطلوبة لتسجيل اكبر رقم نحوله الى

الميزان الثنائي

$$4=(100)_2$$

أي اننا بحاجة هنا الى 3 بتات
عدد اللغات:

يمكن ان لا يتقن هذا الطالب أي لغة . او يمكن ان يتقن لغة واحدة او لغتين او ثلاث لغات او اربع لغات
اذا اكبر عدد هو 4 وكما شاهدنا في الفرع السابق لتسجيل القيمة 4 نحن بحاجة الى 3 بتات
بالنسبة لعدد المواضيع التي يدرسها اكبر عدد هو 11 لمعرفة عدد البتات المطلوبة لتسجيل العدد 11 نحوله
الى الميزان الثنائي:

$$11=(1011)_2$$

اذا نحن بحاجة الى 4 بتات
اذا الحل:

نحن بحاجة الى 5 بتات للكود وبت واحدة لرخصة السياقة و 3 بتات للشعبة و 3 بتات لعدد اللغات التي
يتقنها و 4 بتات لعدد المواضيع اذا العدد النهائي هو 16 بتا اذا يمكن استخدام record من نوع 16 بت أي
2 بايت (byte)

اذا نظرنا الى الجدول السابق والحل النهائي نرى اننا نوفر 3 بايت (byte) لتخزين المعلومات عن كل طالب
سؤال: هل يجب ان يكون عدد البتات المطلوبة 16 او 8 بالضبط ام ماذا
الجواب: اذا كنا بحاجة الى عدد بتات اصغر من 9 تخزن البيانات على شكل record بطول 8
اما اذا كنا بحاجة الى عدد بتات اكبر من 8 واصغر من 17 بت تخزن البيانات على record بطول 16 بت
سؤال: من يخبر الاسمبلر بذلك؟
الجواب: يقوم الحاسوب بجمع عدد البتات لكل حقل المسجلة في bits_num لكل حقل ويعرف العدد
المطلوب

الان نرجع الى حل السؤال السابق

نعرف نمط جديد من نوع record بطول 16 بت

اريد ان اعطي كل حقل اسما كما هو مبين في الجدول التالي:

اسم الحقل	الرمز	عدد البتات
كود الطالب	Code1	5
رخصة	k	1
الشعبة	D	3
اللغات	L	3
المواضيع	S	4

اسم النمط الجديد مثلا student_info

Student_info record code1:5,K:1,D:3,L:3,S:3

ننظر الى كيفية التخزين . نحن نعلم اننا بحاجة الى 16 بتا التخزين يتم بالشكل التالي:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Code1					K	D			L			S			

من الشكل السابق نرى انه لو كنا بحاجة الى 12 بتا فان البرنامج سوف يأخذ record بطول 16 بتا لان
الرقم 12 اكبر من 8 السؤال ماذا مع البتات الاربعة الزيادة
الجواب: البرنامج يحجزها ويضع فيها القيمة صفر
السؤال أي اربعة بتات هل من 1-4 او من 13-16

الجواب: اذا نظرنا الى الجدول السابق نرى انه يبدأ بادخال القيم من الجهة اليسرى أي انه سوف يحجز 4-1 ويضع بها القيمة صفر
تعريف متغيرات من نوع النمط الجديد
مثال:

Std1 student_info < , , , , >

هنا تم تعريف متغير باسم std من نوع student_info ولكن بدون ان نعطيه قيم بدائية

Std2 student_info <10,1,2,3,9>

هنا تم تعريف متغير من نوع student_info مع المعلومات التالية:

الحقل	القيمة البدائية	ملاحظات
كود	10	
رخصة سياقة	معه	لان قيمة الحقل الثاني 1
التخصص	كيمياء	رقم 2 تعني كيمياء
عدد اللغات	3	
عدد المواضيع	9	

ملاحظة: يمكن اعطاء جزء من الحقول قيم بدائية ولكن يجب ان يكون عدد الفواصل بين اشارات اكبر واصغر بقدر عدد الحقول ناقص 1

Std1 student_info <12,0, , , 10>

هنا تم تعريف متغير جديد باسم std1 رقم كوده 12 لا يوجد لديه رخصة سياقة وعدد المواضيع التي يدرسها 10 اما الحقول الاخرى لم نعطيها قيم بدائية.

يمكن اعطاء القيم او تغيير القيم خلال عمل البرنامج

مثلا يمكن ان نكتب برنامجا يستقبل من كل طالب اسم كود الطالب وهل يوجد لديه رخصة ام لا والشعبة وعدد اللغات التي يتقنها وعدد المواضيع التي يتعلمها ونخزنها في الحاسوب.

ويمكن ايضا اجراء عمليات تغيير وبحث وغيرها

طبعا سوف نؤجل هذه العملية حتى نتعلم عن كيفية انشاء ملفات وتخزين البيانات فيها

اوامر ال record

للتعامل مع هذا النمط الجديد نحن بحاجة الى ثلاثة اوامر رئيسية وهي

الامر الاول: اسم الحقل يرجع عدد الازاحات التي يجب ان عملها حتى يصبح هذا الحقل في الجهة اليمنى
امثلة:

Mov cx,code1

بعد هذا الامر تصبح قيمة المسجل

Cx=11

لان الحقل code موجود في البتات من 13-16 وحتى يصبح في البتات من 1-5 يجب ازاحته 11 بتا

Mov bx,L

Bx=4

لان هذا الحقل موجود في البتات من 5-7 وحتى يصبح في البتات من 1-3 يجب ازاحته 4 بتات الى اليمين
بالاختصار عليك عدد عدد البتات التي تقع عن يمينه

```
Mov cx,K
Cx=10
Mov bx,D
Bx=7
```

```
Mov ax,S
Ax=0
```

لان هذا الحقل موجود في جهة اليمين

الامر الثاني: mask field_name : يرجع قيمة عددية كلها اصفار ما عدا البتات الموجودة في نفس مكان الحقل المذكور.

```
Mov ax,mask code
Ax=(1111100000000000)2=F800H
```

```
Mov ax, mask K
Ax=(0000010000000000)2=0400H
```

```
Mov ax,D
Ax=(0000001110000000)2=0380H
```

```
Mov ax, mask L
Ax=(0000000001110000)2=0070H
```

```
Mov ax,mask S
Ax=(0000000000001111)2=000FH
```

الامر الثالث: width field_name يرجع عدد البتات التي يحجزها هذا الحقل امثلة:

```
Mov ax,width code1
Ax=5
```

```
Mov ax, width K
Ax=1
```

```
Mov ax,width D
Ax=3
```

```
Mov ax,width L
Ax=3
```

```
Mov ax,width S
Ax=4
```

الان كيف يمكن تعريف 20 طالب؟

الجواب: اذا اردنا ان لا نعطي الطلاب قيم بدائية نسجل ما يلي:

```
Students student_info 20 dup (< , , , , >)
```

هنا تم تعريف مصفوفة من نوع student_info بدون ان نعطي الحدود قيم بدائية

اما اذا اردنا نعطي الطلاب قيم بدائية مختلفة نعمل مصفوفة ونعطي 20 قيمة مختلفة لكل طالب قيمة خاصة به .نكتفي في المثال التالي بتعريف مصفوفة تحتوي على ثلاثة طلاب
Students student_info <0,1,2,2,9>, <5,0,1,1,11>,<3,1,3,4,10>

مثال محلول:

لدينا فرقة من الطلاب تتكون من 6 طلاب . في الذاكرة مخزنة معلومات عن كل طالب . كود الطالب ، نوع الامتحانات (امتحانات شفوية، امتحانات كتابية، امتحان عملي)، عدد الامتحانات على الطالب ان يقدم على الاكثر 3 امتحانات
بما ان عدد الطلاب 6 نعطي كل طالب رقم
الاول رقم صفر الثاني 1 الثالث 2
اذا الطالب السادس يأخذ الرقم 5 وهو اكبر قيمة تظهر في الكود

$$5=(101)_2$$

اذا نحن بحاجة الى ثلاثة بتات (bits)

نوع الامتحان

نعطي امتحان شفهي الرقم صفر

نعطي امتحان كتابي الرقم 1

نعطي امتحان عملي الرقم 2

اذا اكبر قيمة 2 وهي بحاجة الى 2 بتات لانه

$$2=(10)_2$$

عدد الامتحانات

بما ان اكبر عدد امتحانات سيكون 3

اذا نحن بحاجة الى 2 بتات لانه

$$3=(11)_2$$

اذا عدد البتات الكلي هو 7 بتات اذا سوف نأخذ record بطول 8 لان العدد الكلي اقل من 9
اليك الحل التالي:

البرنامج يطلب من المستعمل ادخال كوده في المجال [0..5] ويفحص ما ادخله الطالب اذا كان الكود المدخل خارج المجال يطبع له جملة تخبره انه ادخل كودا خارج المجال ويعطيه فرصة جديدة . اذا ادخل المستعمل كودا ضمن المجال يطبع له البرنامج نوع الامتحان . ويطبع له جملة تطلب منه ادخال رمز Y (حرف كبير) او رمز y (حرف صغير) لانهاء البرنامج والا يعود البرنامج من جديد

```
.model small
.stack 20h
.data
student_info record code1:3,kind1:2,count1:2
students student_info <0,1,1>,<1,2,2>,<2,3,0>,<3,2,1>,<4,2,2>,<5,1,2>
nl db 10,13,'$'
msg db 10,13,'enter your code in the range [0..5]....$'
msg2 db 10,13,'to stop the program press y or Y....$'
msg3 db 10,13,'your code is out of the range, remember [0..5]..$'
msg4 db 'your kind is....$'
print macro x
lea dx,x
```

```

mov ah,9
int 21h
endm
.code
start:mov ax,@data
      mov ds,ax
      print nl
next:print msg
      mov ah,1
      int 21h
      sub al,30h
      cmp al,5
      jna t1
      print msg3
      jmp next
t1:mov ah,0
     mov si,ax
     print nl
     print msg4
     mov bl,students[si]
     and bl,mask kind1
     mov cl,kind1
     shr bl,cl
     mov dl,bl
     add dl,30h
     mov ah,2
     int 21h

     print msg2
     mov ah,1
     int 21h
     cmp al,'y'
     jz finish
     cmp al,'Y'
     jz finish
     mov ax,3
     int 10h

     jmp next
finish:Mov ah,4ch

```

الامر struc

بمساعدة هذا الامر يمكن انشاء نمط جديد منتركيبة البيانات الغير موجودة في لغة الاسبلي. ومن ثم يمكن ان يستعمل هذا النمط الجديد لتعريف متغيرات في البرنامج.

مثال: معطى البيانات التالية:

- كود الطالب في المجال [1..1000] أي ان عدد الطلاب في المعهد 1000 طالب ولكل طالب رقم خاص فيه حتى يتمكن البرنامج من التفريق بين الطلبة.
- عدد الامتحانات التي يقدمها الطالب 3 امتحانات
- علامة الطالب في كل امتحان في المجال [0..100]
- معدل الطالب هو عبارة عن معدل العلامات الثلاث بعد عملية التقريب
- اسم الطالب الشخصي
- اسم الاب
- اسم العائلة

لون اردنا ان نعرف البيانات السابقة بمساعدة نمط من الانماط التي تعرفها لغة الاسبلي (db , dw , dd , dq,dt) فهذا صعب لان كل معلومة من المعلومات السابقة سوف تعرف لوحدها وعند التوجه والتعامل معها في البرنامج سوف يكون ايضا صعبا.

الحل الافضل هو تعريف تركيبة بيانات جديدة تضم البيانات السابقة. لتعريف تركيبة جديدة علينا ان نحدد نوع وطول البيانات التي نريد ان نعرفها ضمن التركيبة الجديدة

نبدأ بالبيانات السابقة:

كود الطالب : حسب المعطيات السابقة اكبر كود هو 1000 في الميزان العشري أي 3E8 في الميزان السادس عشر أي اننا بحاجة الى ثلاث خانات لتخزين القيمة الكبرى وكل خانة بحاجة الى اربع بتات أي نصف بايت اذا لا يمكن تخزين هذه القيمة في متغير من نوع بايت وانما يجب ان يكون متغير من نوع

word

Student_Code dw ?

عدد الامتحانات: اكبر عدد من الامتحانات التي يمكن ان يقدمه الطالب هو 3 اذا يمكن ان نخزن هذه القيمة

في متغير من نوع byte

Exam_num db 0

علامات الطالب

علامة الطالب في كل امتحان هي في المجال [0..100] اذا اكبر علامة هي 100 وهذه القيمة يمكن ان

تخزن في byte

Mark1 db 0

Mark2 db 0

Mark3 db 0

معدل الطالب

اكبر معدل يمكن ان يحصل عليه الطالب هو 100 ونفرض ان المعدل لا يحتوي على فواصل أي انه بعد حساب المعدل سوف نقوم بعملية تقريب وبعدها يتم تخزين المعدل النهائي.
اذا يكفي هنا ان نستخدم بايت لتخزين المعدل

Student_average db 0

ملاحظة: لو اردنا ان نحفظ المعدل مع فواصل يفضل ان يكون نوع المتغير المسؤول عن المعدل من نوع نص

مثلا معدل مع خانتين عن يمين الفاصلة اكبر معدل هو 100.00

Stud_avg db '000.00'

اسم الطالب:

نفرض ان اطول اسم هو بطول 15 رمزا

اذا نعرف نصا بطول 15 رمزا

Student_name db 15 dup ('0')

هنا تم حجز 15 مكانا لاسم الطالب

اسم الاب:

نفرض ان اطول اسم هو بطول 15 رمزا

اذا نعرف نصا بطول 15 رمزا

Student_father db 15 dup ('0')

هنا تم حجز 15 مكانا لاسم الاب

اسم العائلة:

نفرض ان اطول اسم هو بطول 15 رمزا

اذا نعرف نصا بطول 15 رمزا

Student_family db 15 dup ('0')

هنا تم حجز 15 مكانا لاسم العائلة

لتسهيل عمليات البحث والتعديل والتوجه الى القيم المطلوبة بسهولة ومرونة اكثر يفضل ان نعرف تركيبية جديدة تحتوي على جميع التعاريف السابقة. يمكن ان نقوم بهذه العملية بمساعدة الامر struc

المبني العام للتركيبية

Structure_name struc

Fields of the structure

Structure_name ends

حسب المعطيات السابقة

Student_information struc

Student_Code dw ?

Exam_num db 0

Mark1 db 0

Mark2 db 0

Mark3 db 0

Student_average db 0

Student_name db 15 dup ('0')

Student_name db 15 dup ('0')
 Student_family db 15 dup ('0')
 Student_information ends

ملاحظات هامة:

- عند تعريف تركيبة جديدة لا يحجز هذا التعريف مكانا في الذاكرة انما عند استخدام التركيبة الجديدة في البرنامج يتم الحجز.
- يمكن اعطاء حقل او اكثر قيم بدائية عند تعريف متغير من نمط التركيبة الجديدة

تعريف متغيرات من نوع التركيبة الجديدة

يمكن تعريف متغيرات من نوع التركيبة الجديدة student_information مع امكانية اعطاء الحقول قيم بدائية (أولية)

1. تعريف متغيرات جديدة من نوع التركيبة student_information بدون تعويض قيم في الحقول أي ان المتغير الجديد يأخذ نفس القيم التي عرفت في التركيبة

Students student_information < >

هنا تم تعريف متغير باسم students يحتوي على الحقول التي تحتوي عليها التركيبة student_information بدون تعويض قيم في الحقول وانما تأخذ الحقول نفس القيم الموجودة في تعريف التركيبة student_information

2. تعريف متغيرات جديدة مع اعطاء قيم للحقول

Student1 student_information <20,3,100,80,90,90,'akram','malek','mohsen'>

هنا تم تعريف متغير باسم student1 من نوع student_information وتم تعويض قيم أولية في الحقول.

مثلا: كود الطالب 20 , قدم الطالب 3 إمتحانات ، العلامات التي حصل عليها هي 100 ، 80 ، 90 معدل

الطالب 90 اسم الطالب akram اسم الاب malek اسم العائلة mohsen

3. تعريف متغير من نوع التركيبة student_information وتعويض قيم أولية في جزء من الحقول وباقي الحقول تأخذ القيم الاولية التي عرفت في التركيبة

Student2 student_information <10,2,100,90, , , 'samee', 'kasem', 'hamed'>

هنا تم تعريف متغير باسم student2 مع القيم التالية

اسم الحقل	القيمة الاولية	السبب
كود	10	انظر التعريف student2
عدد الامتحانات	2	انظر التعريف student2
العلامة الاولى	100	انظر التعريف student2
العلامة الثانية	90	انظر التعريف student2
العلامة الثالثة	0	انظر تعريف التركيبة student_information
المعدل	0	انظر تعريف التركيبة student_information
اسم الطالب	samee	انظر التعريف student2
اسم الاب	kasem	انظر التعريف student2
اسم العائلة	hamed	انظر التعريف student2

ملاحظة: يمكن خلال البرنامج ان نقوم بتغيير القيم التي تم تعويضها عند تعريف المتغير
3. تعريف متغير مع اعطاء حقل واحد فقط قيمة اولية

```
Student3 student_information <5>
```

هنا تم اعطاء الكود القيمة 5 وباقي الحقول تأخذ القيم الافتراضية أي القيم التي عوضناها في الحقول عند تعريف التركيبة student_information
4. تعريف متغير مع اعطاء حقل واحد فقط قيمة اولية

```
Student4 student_information < , 2>
```

هنا تم اعطاء الحقل عدد الامتحانات القيمة 2 وباقي الحقول تأخذ القيم الافتراضية

```
Student5 student_information < 22,1,20>
```

هنا الكود 22 عدد الامتحانات التي قدمها 1 العلامة الاولى 20 وباقي الحقول تأخذ القيم الافتراضية التي اعطيت في التركيبة student_information

التوجه الى الحقول المطلوبة

هنالك عدة طرق للتوجه الى حقل معين
مثال معطى المتغير التالي:

```
Student10 student_info < >
```

للوصول الى العلامات

```
Mov al,student10.mark1
```

```
Mov bl,student10.mark2
```

```
Mov cl,student10.mark3
```

اذا بمساعدة اسم المتغير واسم الحقل يمكن ان نصل الى القيمة
الطريقة الثانية

نؤشر على بداية اسم المتغير بمساعدة المسجل bx ثم نتوجه الى الحقل المطلوب
مثلا اذا اردنا ان نتوجه الى معدل الطالب

```
Lea bx , student10
```

```
Mov ch , [bx].average
```

التركيبات والمصفوفات

عندما نعرف مصفوفة من نوع تركيبة معينة لا يجوز ان نعطي اسم المصفوفة التي تتكون من عدة عناصر
وكل عنصر عبارة عن تركيبة قيم اولية.
مثلا لو اردنا ان نعرف صف يتكون من 30 طالبا .

```
Class10_students student_information 30 dup (< >)
```

لا يجوز ان نعطي قيم اولية للحقول مثلا

```
Class10_students student_information 30 dup (<2,10>)
```

هذا التعريف خاطئ

