



اسم المادة : هيكلية الحاسوب ولغة أسمبلي

تجمع طلبة كلية التكنولوجيا والعلوم التطبيقية - جامعة القدس المفتوحة

acadeclub.com

وُجد هذا الموقع لتسهيل تعلمنا نحن طلبة كلية التكنولوجيا والعلوم التطبيقية وغيرها من خلال توفير وتجميع **كتب وملخصات وأسئلة سنوات سابقة** للمواد الخاصة بالكلية, بالإضافة لمجموعات خاصة بتواصل الطلاب لكافة المواد:

للوصول للموقع مباشرة اضغط **هنا**

وفقكم الله في دراستكم وأعانكم عليها ولا تنسوا فلسطين من الدعاء

جامعة القدس المفتوحة (فرع شمال غزة)



كلية التكنولوجيا والعلوم التطبيقية

هيكلية الحاسوب ولغة اسمبلي

١٣٨١

الوحدة الرابعة: **تعليمات لغة التجميع اسمبلي**

الفصل الأول ١٢٢١

اعداد : أ. مصلح منير مصلح

طاقم تعليمات المعالج ٨٠٨٦

- يوجد في معالج ٨٠٨٦ عدد من التعليمات **الاساسية ١١٧**
- تعليمة mov الاساسية تمتد إلى ٢٦ تعليمة مختلفة وتنفذ على مستوى لغة الآلة.

❖ تعليمات نقل البيانات والمعطيات

وظيفة هذه التعليمات تقوم في نقل البيانات والمعطيات من المسجلات القطاعات ومسجلات المعطيات وحجرة في الذاكرة .

تعليمات نقل البيانات والمعطيات

تعليلة mov

- تستخدم في نقل بايت أو كلمة من مكان لمكان آخر أي من مصدر إلى الهدف

الرمز	المعنى	مثال
acc	المراكم	ax
Mem	حجرة في الذاكرة أو موقع في الذاكرة	متغير
Reg	مسجل	Ax,bx,cx,dx,al,ah,...
IMM	متحول فوري	ثابت
Seg-reg	متحول مقطع	CS,DS,ES,ES
Reg16	تعامل مسجل ٢ بايت أو كلمة	للاشارة للمسجل SI ,DI,SP,...
Mem16	حجرتين في الذاكرة أو موقع في الذاكرة يتعامل مع ١٦ بت أو ٢ بايت أو كلمة	متغير يتعامل مع ١٦ بت أو ٢ بايت أو كلمة

تعليلة mov

- قيود استخدام تعليلة mov
- مسجل عام ax,cx,dx,bx,ah,al,bh,bl....
- مسجل مقاطع CS,DS,ES,SS
- متغير يحجز موقع في الذاكرة او حجرة في الذاكرة

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

الحالات المستثناة من تعليمة mov

١. لا تستطيع تعليمة MOV نقل البيانات والمعطيات بشكل مباشر بين حجرتين في الذاكرة

Mov x,y

- لحل هذه المشكلة نستخدم مسجل داخلي والنقل يتم من خلال هذه المسجلات

Mov al,x

Mov ah,y

٢. لا يجوز التعامل بشكل مباشر مع مسجل القطاع لذلك نتعامل بطريقة غير مباشرة مثال
لا يجوز وضع قيمة ثابتة او فورية في مسجل القطاعات

Mov ds,1000

- لحل هذه المشكلة نستخدم المسجلات الداخلية أي مسجلات المعطيات

Mov ax,1000

Mov ds ,ax

الحالات المستثناة من تعليمة mov

٣. لا يمكن نقل محتويات مسجلات القطاعات CS,DS,ES,SS الى مسجل قطاعات اخرى بشكل مباشر لذلك نحتاج الى مسجلات المعطيات حتى يتم نقل محتويات سجل مقاطع مع بعض

Mov DS,ES

لحل هذه المشكلة

MOV AX,ES

MOV DS,AX

مثال عام

MOV AL,[SI]

معناه انقل عنوان او حجرة مسجل SI كمان تم شرحه سابق هذه المسجل المساعد يدل على قطاع DS وذلك من خلال قوسين [] وهذا العنوان يتم احتسابه من خلال قانون الفيزيائي $PA = DS * 01 + OFFSET(SI)$ وهذا العنوان يتم نقله إلى مسجل AL

تعليلة XCHG

- تستخدم هذه التعليلة لاستبدال محتويات المصدر إلى الهدف ومحتويات الهدف إلى محتويات المصدر.
- يستخدم لعملية تبديل القيم أو عكس القيم في مسجلين أو مسجل مع متغير
- الصيغة العامة : **XCHG Destination, Source**
- مثال : **XCHG AX,DX**
- معناه تبديل قيم مسجلان AX,DX بحيث تصبح قيمة AX هي قيمة DX وقيم DX هي قيمة AX
- مثال : **XCHG CHA,SSA** هذه الحالة لا تجوز وذلك لان نقل محتوى متغير مع متغير لا يصلح يجب اختيار مسجل وسطي

الحالات المستثناة من هذه التعليمة:

- الحالات المستثناة

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

- XCHG [SUM],BX تعني تبديل قيم كل من مسجل BX ووضع هذه القيمة في الذاكرة او موقع في الذاكرة ونقل بيانات في هذا الموقع في مسجل BX

تعليلة XLAT

- تتعامل هذه التعليلة مع **AL فقط**
- تستخدم هذه التعليلة مع الجداول أي المصفوفات من اجل تغيير عنصر محل عنصر
- بحيث ان يتم تخزين الجدول في قطاع DS
- ويوضع في مسجل **BX** ازاحة بداية الجدول بالنسبة الى بداية قطاع DS
- ويوضع في مسجل AL ازاحة العنصر بالنسبة إلى بداية الجدول
- تعمل هذه التعليلة على جمع محتويات مسجل AL مع محتويات مسجل BX بالاضافة على قيمة بداية قطاع DS
- **$AL = [AL + BX] + DS * 10 = القيمة$**
- الناتج يتم تخزينه في مسجل AL وهذا الناتج يكون عبارة عن الازاحة بالنسبة لقطاع DS

مثال ص ٨١

- على فرض لديك مصفوفة او جدول يتكون من ٩ حجلات في اول حجرة ١ والحجرة الثانية ٤ وهكذا وازاحة هذا الجدول عن قطاع DS تساوي 100 غير قيمة الحجرة الثانية بدل ال 4 الى 16

التعليمات LDS,LES,LEA

- تستخدم هذه التعليمات لنقل البيانات او المعطيات بحيث يتم تحميلها بشكل مباشر من عنوان ذاكرة إلى مسجل مقطع او مسجل
- **تعليمة LEA :load affective address**
 - وظيفتها تحميل قيمة الموقع الفعال EA على مسجل SI,DI مثل .
 - الصيغة LEA REG16,EA
 - REG16 هي SI,DI,BX,CX
 - بمعنى يحمل قيمة EA وليس محتوياته أي يحمل مقدار الازاحة
 - مثال LEA SI,[BX+DI]
 - EA=26 ,DI=14 ,BX=12 يتم تخزين قيمة ٢٦ في SI
- **تعليمة LDS:Load data register Segment**
 - عملية النقل تتم فيها : نقل البيانات إلى مسجل ومن المسجل تتم نقل البيانات إلى قطاع DS **تحميل الى مسجل والمسجل DS**
 - عملية النقل هنا تكون ٢ بايت أي كلمة ومن ثم يتم نقل ٢ بايت اخرى بحيث ان اول ٢ بايت يتم نقلها إلى REG و ٢ بايت الاخرى تنتقل الى DS

LDS reg16,mem32

Mem32 → reg16
Mem32+2 → DS

التعليمات LDS,LES,LEA

تعليمة LES:Load EXtra register Segment

➤ عملية النقل تتم فيها : نقل البيانات إلى مسجل ومن المسجل تتم نقل البيانات إلى قطاع ES

تحميل الى مسجل والمسجل ES

➤ عملية النقل هنا تكون ٢ بايت أي كلمة ومن ثم يتم نقل ٢ بايت اخرى بحيث ان اول ٢ بايت يتم نقلها إلى REG و ٢ بايت الاخرى تنتقل الى ES

LES reg16,mem32

Mem32 → reg16
Mem32+2 → ES

Mnemonic	Meaning	Format	Operation	Flags affected
LEA	Load Effective address	LEA REG, EA	(EA) → (REG)	None
LDS	Load register and DS	LDS REG, MEM	(PA) → (REG) (PA+2) → (DS)	None
LES	Load register and ES	LES REG, MEM	(PA) → (REG) (PA+2) → (ES)	None

مثال

• اذا علمت ان $BX=20H$ $DI=1000H$ $DS=1200H$

اوجد كلا من

1. $LEA\ SI, [DI+BX+5]$

2. $LDS\ SI, [200H]$

3. $LES\ SI, [200H]$

محتويات هذه العناوين	عناوين الذاكرة
11	12200
BB	12201
EF	12202
CC	12203
33	12204

1. $EA=1000+20+5=1025$ اذا سيتم تخزين قيمة العنوان الفعال في $SI=1225$

2. $PA=1200*01+200=12200$

أي SI المسجل الاول يتم فيه تخزين ١٦ بت أي ٢ بايت $SI=BB11$
ومن ثم يتم اضافة ٢ بايت أي حجتين وتصبح قيمة $DS=CCEF$

التعليمات الرياضية

أولاً: تعليمات الجمع

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
ADD	جمع	ADD D,S	$S + D \rightarrow D$ Carry \rightarrow CF	أعلام الحالة
ADC	جمع مع أخذ الانزياح بعين الاعتبار	ADC D,S	$S + D + CF \rightarrow D$ Carry \rightarrow CF	أعلام الحالة
INC	الزيادة بمقدار واحد	INC D	$D + 1 \rightarrow D$	أعلام الحالة
AAA	تصحيح ناتج جمع عددين بشيفرة الأسكي	AAA	سيتم شرحها لاحقاً	AF,CF
DAA	تصحيح ناتج جمع عددين بشيفرة BCD	DAA	سيتم شرحها لاحقاً	كل أعلام الحالة ماعدا OF

تعليمات الجمع

- يوجد ستة أعلام للحالة هي OF, SF, ZF, AF, PF, CF
- تعليمتي ADD و ADC إن المتحولات المسموحة:

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

المصدر S	الهدف D
Reg	Reg
Mem	Reg
Reg	Mem
Imm	Reg
Imm	Mem
Imm	Acc

➤ بالنسبة للمتحولات المسموحة في تعليمة INC فهي، تعليمة تحتوي على معامل واحد قد يكون :

الهدف D
Reg8
Reg16
Mem

مثال

- مثال: بفرض $AX=4F3DH$ و $BX=FD81H$ و $CF = 1$ فما هي نتيجة تنفيذ التعليمة $ADC AX, BX$ مبيناً حالة أعلام الحالة بعد تنفيذ عملية الجمع هذه .
- الحل: كونه السؤال طلب حالة الأعلام لابد من تحويله على الأعداد الثنائية من أجل توضيح حالة الأعلام

$$\begin{array}{r}
 \begin{array}{ccccccc}
 & 1111 & 111 & & & & 1 \\
 AX = & 0100 & 1111 & 0011 & 1101 & b \\
 BX = & 1111 & 1101 & 1000 & 0001 & b \\
 CF = & & & & 0001 & b & + \\
 \hline
 \boxed{1} & 0100 & 1100 & 1011 & 1111 & b \\
 \hline
 CF & & & & & &
 \end{array}
 \end{array}$$

تابع حل المثال

• و الآن أعلام الحالة

- هي : $PF = 0$ لأن عدد الواحدات فردي في البايت الأول والبايت الاول يكون في هذا الاعلام نقطة الاهتمام من ناتج الجمع
- $AF = 0$ لأنه لا يوجد انزياح من الخانة ٣ إلى الخانة ٤ في البايت الأول من ناتج الجمع (حيث يتم ترقيم الخانات بدءاً من الصفر) للبايت الاول
- $SF = 0$ و هي آخر خانة من نتيجة الجمع (الناتج موجب) لآخر بت
- $CF = 1$ بسبب وجود انزياح خارجي لآخر بت
- $OF = 0$ لأنه يوجد إنزياح داخلي و إنزياح خارجي لآخر بت
- ملاحظة: $OF = 1$ إذا وجد انزياح داخلي فقط أو وجد انزياح خارجي فقط

0 1

0 1

بدون الاهتمام ان الحمل هو السبب بوضع الناتج 1 $1 = 1 \quad 0$

علیمة التصحيح DAA

Decimal Adjust for Addition

- تستخدم هذه التعليمة لإنجاز عملية تصحيح لناتج جمع عددين بشيفرة BCD
- وهذه الشيفرة تمثل العدد في ٤ بت
- يجب أن يكون ناتج الجمع حتماً في AL أي في النصف السفلي من المراكم AX
- **خطوات الحل :**
- نحول الأعداد إلى ثنائية بحيث يتم تمثيل كل رقم في ٤ بت وإجراء عملية الجمع
- إذا كان لدينا ناتج عملية الجمع يوجد به انزياح $AF=1$ من بت ٣ إلى ٤ أو الجزء الأدنى من البتات $Bit(0,1,2,3)$ أو عند تحويلها لشيفرة BCD أكبر من ٩ نعمل على إضافة ٦ إلى الناتج بمعنى إذا تحقق أي شرط من هذان الشرطين
- إذا كان البتات العليا أي البتات $Bit(4,5,6,7)$ أكبر من ٩ أو إذا كان $CF=1$ يتم إضافة ٦٠ إلى المراكم إلى الناتج

if Bit3 Bit2 Bit1 Bit0 of **AL > 9 or AF = 1**

then **AL = AL + 6 , AF = 1**

if **AL > 9 h or CF = 1** then

AL = AL + 60h , CF = 1

مثال : على فرض ان $BL = 68$ BCD و $AL = 28$ BCD

اوجد ما يلي : $ADD AL, BL$

DAA

↓

1

$$\begin{array}{r} 28 \text{ BCD} = 0010 \ 1000 \text{ b} \\ 68 \text{ BCD} = 0110 \ 1000 \text{ b} \quad + \\ \hline 1001 \ 0000 \rightarrow \text{AL} \\ \text{CF} = 0 \quad \quad 0110 \quad + \\ \hline 1001 \ 0110 \Rightarrow \text{AL} = 96 \text{ BCD} \\ \text{AF} = 1 \end{array}$$

الحل : إن نتيجة تنفيذ هاتين التعليمتين هي

حيث 0110 تمثل الرقم ستة

تعليلة تصحيح AAA

ASCII Adjust for Addition

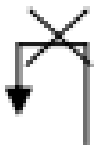
- تستخدم هذه التعليلة لتصحيح ناتج جمع عددين بشيفرة آسكي (و هنا أيضاً يجب أن يكون ناتج الجمع في المسجل AL)
- قاعدة التصحيح في هذه التعليلة هي:
 1. بالبداية نحول كل خانة الى 4 بت ونجري عملية الجمع
 2. اذا كان الناتج Bit(0,1,2,3) اكبر من 9 أو حمل متوسط يساوي 1 $AF=1$
 3. نعمل على إضافة 6 الى الناتج ومن ثم كذلك نظيف 1 الى $AH=AH+1$
 4. بحيث تصبح قيمة كل من $AF = 1$ $CF = 1$
 5. الرجوع إلى المثال السابق يكون الناتج عبارة عن A6
 6. غير ذلك ليس أكبر من 9 أو عدم وجود حمل منتصف لا نظيف 6 ونعمل على تصفير الجزء الاعلى الايسر وقيمة $AH=00H$

مثال

- مثال: بفرض أن $AL=32H=2ASCII$ و $BL=34H=4ASCII$ ما هو ناتج تنفيذ التعليمتين التاليتين :

ADD AL,BL

AAA


$$\begin{array}{r} AL = 0011 \ 0010 \\ BL = 0011 \ 0100 \ + \\ \hline 0110 \ 0110 \ \rightarrow AL = 66h \\ AL = 06h, AH = 00 \end{array}$$

تعليمات الطرح

- هناك مجموعة واسعة من تعليمات الطرح كما هو واضح من الجدول التالي

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	$D - S \rightarrow D$ $\text{borrow} \rightarrow CF$	$SUB\ D,S$	طرح	SUB
أعلام الحالة	$D - S - CF \rightarrow D$ $\text{Carry} \rightarrow CF$	$SBB\ D,S$	الطرح مع الاستعارة	SBB
أعلام الحالة	$D - 1 \rightarrow D$	$DEC\ D$	الإنقاص بمقدار واحد	DEC
أعلام الحالة	$0 - D \rightarrow D$ $1 \rightarrow CF$	$NEG\ D$	المتمم الثنائي	NEG

الثاني

تعليمات الطرح

DAS	تصحیح ناتج طرح عددين بشيفرة BCD	DAS	سيتم شرحها لاحقاً	كل أعلام الحالة عدا OF
AAS	تصحیح ناتج جمع عددين بشيفرة الأسكي	AAS	سيتم شرحها لاحقاً	AF, CF

بالنسبة لتعليمة **NEG** فالمتحولات المسموحة هي **Mem16, Mem, Reg16, Reg.**

مثال

- مثال: بفرض أن **DS=2F00H,SI=0018H** و العنوان الفيزيائي المتولد عنهما هو **2F018h** و بفرض كانت محتويات الحجرة التي يشير إليها العنوان الفيزيائي **[2F018]=0400H**
- اوجد ناتج تنفيذ التعليمة **SUB [SI],03F8H**
- ملاحظة الحاسوب لا يفهم عملية الطرح يأتي بالطرح من خلال عملية الجمع وذلك بإيجاد المتمم الحسابي الثاني للمطروح منه ومن ثم اجراء عملية الجمع
- بالبداية نحول الاعداد الى ثنائية
- ومن ثم نعمل على ايجاد المتمم الحسابي الثاني للعدد 03F8
- 0000 0011 1111 1000
- المتمم الثاني نبدأ من الشق الايمن ننزّل للصفر واول بت 1 يتم تنزيله من ثم عكس ما بعده يصبح 1111 1100 0000 1000

تابع المثال

نجري عملية الجمع



$$\begin{array}{r} \text{Destination} = 0400h = 0000\ 0100\ 0000\ 0000\ b \\ \text{Source} = 03F8h = 1111\ 1100\ 0000\ 1000\ b \quad + \\ \hline \square \quad 0000\ 0000\ 0000\ 1000\ b \end{array}$$

لاحظ أن :

✓ $PF = 0$ لأن عدد الواحدات فردي في البايت الأول من الناتج

✓ $AF = 1$ لأنه لا يوجد معنا حمل (انزياح) عند الانتقال من الخانة الثالثة إلى الخانة الرابعة (عكس حالة الجمع).

✓ $ZF = 0$ لأن النتيجة ليست صفرية .

✓ $SF = 0$ و هي قيمة آخر خانة من الناتج . MSB.

✓ $CF = 0$ لأن هناك انزياح خارجي (عكس حالة الجمع).

✓ $OF = 0$ لحصول انزياح داخلي و انزياح خارجي بأن واحد

□ في الطرح لا يكون حمل يكون استعارة

تعليلة DAS

- تستخدم هذه التعليلة لتصحيح ناتج طرح عددي ن بشيفرة BCD حيث يكمن ناتج طرح هذين العددين في المسجل AL وقاعدة التصحيح هي :

if Bit3 Bit2 Bit1 Bit0 of AL > 9 or AF = 1

then AL = AL – 06 , AF = 1

if AL > 9Fh or CF=1

then AL = AL – 60h , CF = 1

مثال DAS

مثال: بفرض أن $AL = 86$ BCD و $AH = 07$ BCD، بين نتيجة التعليمتين التاليتين: $SUB\ AL, AH$

DAS

$$\begin{array}{r} \overline{AL} = 1000\ 0110\ b \\ \overline{AH} = 1111\ 1001\ b \quad + \\ \hline \boxed{1}0111\ 1111\ b \Rightarrow AL = 7Fh \end{array}$$

و الآن :

$AF = 1$ بسبب عدم وجود انزياح من الخانة الثالثة إلى الخانة الرابعة.

$CF = 0$ لوجود انزياح خارجي

و بتطبيق الشرط 1 من قاعدة التصحيح نجد أن $AF = 1$ ، $AL = 79h$

تعليلة AAS

- تستخدم هذه التعليلة لتصحيح ناتج طرح عددين بالشفرة ASCII حيث يكمن ناتج الطرح في AL، و قاعدة التصحيح هي :

if Bit3 Bit2 Bit1 Bit0 of AL > 9 or AF = 1

then AL = AL – 06h , AL = AL and 0Fh

AH = AH – 01 , AF = 1 , CF = 1

Else AL = AL and 0Fh , AH = 00

مثال تعليمة AAS

- بفرض أن $AL=38 = 8 \text{ ASCII}$ و $BL = 35H = \text{ASCII}$ ، ما هو ناتج تنفيذ التعليمتين التاليتين :

SUB AL,BL

AAS

الحل:

$$\begin{array}{r} \text{AL} = 0011\ 1000\ \text{b} \\ \text{BL} = 1100\ 1011\ \text{b} \quad + \\ \hline \boxed{1} 0000\ 0011\ \text{b} \Rightarrow \text{AL} = 03\text{h} \end{array}$$

$AF = 0$ بسبب وجود انزياح من الخانة الثالثة إلى الخانة الرابعة

$CF = 0$ بسبب وجود انزياح خارجي

و بعد تطبيق قاعدة التصحيح نجد $AL = 03\text{h}$, $AH = 00$

تعليلة NEG

- تستخدم هذه التعليلة لايجاد المتمع الحسابي الثاني لمسجل او متغير أي موقع في الذاكرة
- مثال

MOV AL,22H

NEG AL

الحل نحول العدد من النظام السادس عشر الى ثنائي ومن ثم نوجد المتمع الثاني

1. 0010 0010

2. 1101 1110

□ يؤثر على مسجل الحمل ومسجل الاشارة كونه نوجد المتمع الاول + 1 ويتم فيه عكس البت الأخير أي تتحول من ٠ الى ١ او العكس

□ CF,SF يتأثر في هذه التعليلة

تعليمات الضرب والقسمة

- يوجد لدينا تعليمات ضرب وقسمة خاصة في الاشارة او غير مهتمة في الاشارة.

MUL	ضرب بدون إشارة	MUL S	AL.S8 → AX AX.S16 → DX,AX	أعلام الحالة
DIV	تقسيم بدون إشارة	DIV S	Q[AX/S8] → AL ¹⁶ R[AX/S8] → AH ¹⁶ Q[(DX,AX)/S16] → AX ³² R[(DX,AX)/S16] → DX ³²	أعلام الحالة

متحول مصدر عبارة عن بايت

- أما الرمز R فيعني باقي القسمة و الرمز Q ما هو إلا حاصل قسمة .

- لا يمكن ان يكون المصدر ثابت
- المضروب والمقسوم يجب تخزينه في AX
- جميع اعلام الحالة الستة تتأثر

تعليمات الضرب والقسمة

- عملية الضرب mul
- العنصر أو الرقم أو المسجل المراد ضربه يجب ان يكون في خاتمه al في حالة نريد ان نتعامل في ١ بايت
- الرقم أو السجل المرافق للتعليمية يتم ضربه في مسجل al
- `mov al,02`
- `mov bl,04`
- `mul bl`
- تعني ذلك اضرب قيمة المسجل bl في al وخزن الناتج في al بما انه نتعامل مع ١ بايت ملاحظة يجب ان تكون قيمة المصدر متوافقة مع الهدف
- ملاحظة مهمة البرنامج يعمل على تحويل الاعداد من النظام السادس عشر إلى النظام الثنائي ومن ثم يقوم في عملية الضرب والناتج يتم تحويله الى نظام سادس عشر .
- ماذا لو كان عملية الضرب لكلمة أي ٢ بايت ولا يتسع مسجل ax إلى الناتج يعتبر مسجل dx هو مكمل لمسجل ax لتخزين الناتج

تعليمات الضرب والقسمة

مثال

```
MOV AX,500  
MOV BX,500  
MUL BX
```

الحل

- 250000 في النظام العشري نريد تحويله الى نظام سادس عشر من خلال غلى نظام ثنائي ومن ثم غلى سادس عشر .
- الناتج في الثنائي 00001001000011110000
- 3D090 الناتج في نظام سادس عشر يخزن في مسجل AX=D090 ويتبقى لدينا رقم 3 يتم تخزينه في مسجل
- DX =0003

MOV AX,0100H

MOV BX,1234H

MUL BX

• كونه الارقام في السادس عشر نضرب بشكل مباشر

الناتج 0123400

• سيتم AX=3400 DX=0012

تعليمات الضرب و القسمة

- في حالة القسمة اذا كان التعامل مع ١ بايت يكون الناتج في مسجل AL وباقي القسمة في مسجل AH بالنسبة غلى مسجل AX
 - اذا كان التعامل مع مسجل ٢ بايت يكون ناتج القسمة في مسجل AX وباقي عملية القسمة في مسجل DX
 - **اذا اردنا ان نأخذ الاشارة بعين الاعتبار في عملية الضرب و القسمة نستخدم التعليمات الآتية**
- IMUL هي تعليمة الضرب مع أخذ الإشارة بعين الاعتبار
 - IDIV هي تعليمة القسمة مع أخذ الإشارة بعين الاعتبار
 - البت الأخير يدل على اشارة الرقم اذا ١ تكن اشارة الرقم سالبة واذا كان ٠ تكن اشارة الرقم موجبة تسمى آخر خانة في MSB

تعليمات الضرب والقسمة مع اخذ بعين الاعتبار الاشارة

• IDIV,IMUL

- نجري عملية الضرب بشكلها المعتاد كون الحاسوب لا يتعامل مع الطرح والاشارة السالبة نوجد **المتعم الثنائي الثاني** لنظام السادس عشر بالرجوع للأمثلة السابقة:

- `mov al, 02H`
- `mov bl,-03H`
- `Imul bl` **AL=-6 = 0000 0110=1111 1010**
- **=FA=AL**

مثال ضرب بالاشارة

مثال

```
MOV AX,-500  
MOV BX,500  
IMUL BX
```

الحل

- **250000-** في النظام العشري نريد تحويله الى نظام سادس عشر من خلال غلى نظام ثنائي ومن ثم الى سادس عشر .
- **3D090** الناتج في نظام سادس عشر يخزن في مسجل كون الاشارة سالبة **3D090-** نوجد متمم الثاني
- **AX=2F70h** كونه الاشارة سالبة تصبح **AX=D090**
- **DX =0003** في حال الاشارة السالبة تصبح قيمة **DX=FFFC**

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	$Q[AL/10d] \rightarrow AH$ $R[AL/10d] \rightarrow AL$	AAM	تصحيح الناتج في AL من ضرب عددين BCD أو عددين ثنائيين	AAM
SF, ZF, PF	$AH.10d + AL \rightarrow AL$ $00 \rightarrow AH$	AAD	<p>تصحيح AX من أجل القسمة حيث AX ليس ناتج القسمة وإنما هو متحول الهدف في عملية القسمة. لذلك نطبق هذه التعليمة قبل تعليمة القسمة على عكس باقي تعليمات التصحيح</p>	AAD
لا يوجد	$MSB \text{ of } AL \rightarrow$ $All \text{ bits of } AH$	CBW	تحويل بايت إلى كلمة	CBW
لا يوجد	$MSB \text{ of } AX \rightarrow$ $All \text{ bits of } DX$	CBW	تحويل كلمة إلى كلمة مضاعفة	CWD

R: فيعني باقي القسمة و الرمز
Q: ما هو إلا حاصل قسمة

تعليمات الضرب و القسمة

- إن المتحولات المسموحة في تعليمات الضرب و القسمة هي بالنسبة للمصدر **S** Reg8, Reg16, Mem8, Mem16
- و بالنسبة إلى الهدف **D** فالمتحول الوحيد المسموح هو المراكم دوماً.
- **تمديد الاشارة المقسوم في حالة استخدام ارقام ٨ بت :**
- تعليمة **CBW** تستخدم لتمديد الاشارة من سجل **AL** الى **AH**
- بحيث ان يصبح سجل **AX** يحتوي على ١٦ بت
- بحيث يكون تمديد الاشارة اذا كان ٨ بت في **AH**
- اذا كان العدد موجب المراد تمديده أي آخر بت في **AL** يكون 0 بحيث يتم تمديد سجل **AH** بـ ٨ بت من الأصفار 00H=00000000
- اذا كان العدد سالب المراد تمديده أي آخر بت في **AL** يكون 1 بحيث يتم تمديد سجل **AH** بـ ٨ بت من الأحاد FF H = 11111111

تعليمات الضرب و القسمة

- تمديد الاشارة المقسوم في حالة استخدام ارقام ١٦ بت

CWD:

➤ في القسمة ١٦ بت اذا كان الناتج يمكن استيعابه في AX يوضع في DX قيمة 0

➤ عند استخدام امر IDIV يخزن الناتج في AX وقيمة DX تكون مساوية لخانة الاشارة مثلا لو كانت آخر بت في AX قيمته 1 وهذا يوحي الاشارة سالبة لو تم تمديدها تكون قيمة DX=11111111_11111111=FFFFh

تعليمات تعديل الاسكي للضرب AAM:

- إن التعليمة AAM تستخدم لتصحيح ناتج ضرب عددين غير مجمعين لأنه عند ضرب عددين غير مجمعين نحصل على نتيجة مجمعة و النتيجة يجب أن تكون غير مجمعة، لذلك نصححها بواسطة التعليمة AMM
- تأتي هذه التعليمة بعد عملية الضرب
- مثال: بفرض أن $BL = 09$ و $AL = 07$ فما هي نتيجة تنفيذ التعليمات التالية :

MUL BL

AAM

حل المثال

- النتيجة في العشري 63 ويقسم على 10
- آلية العمل الناتج يتم تحويله الى ثنائي ومن ثم إلى عشري ومن ثم نقسم على 10.

Q[AL/10d] → AH
R[AL/10d] → AL

- R فيعني باقي القسمة و الرمز Q ما هو إلا حاصل قسمة

MUL BL
AAM

AX = 00 07
BX = 00 09

MUL	00 3F	AX
AAM	06 03	AX

مثال على عملية القسمة

Mov al,6

Mov bl,3

Div bl

الحل يخزن الناتج في al وباقي القسمة في ah

$$6/3=2$$

Ax=00 02 h

نلاحظ المقسوم عليه عبارة عن ٨ بت المرافق لتعليمة div

لكن ل كان المقسوم عليه عبارة عن ١٦ بت المرافق لتعليمة div
يكون ناتج القسمة في ax وباقي القسمة dx

مثال على عملية القسمة

Mov ax,9

Mov bx,4

Div bx

Ax=0002h

Dx=0001h

نلاحظ ان المسجل بجانب التعليمة ١٦ بت

تعليمة AAD

- قاعدة التصحيح في تعليمة AAD هي : إن التقسيم بالنسبة إلى الأعداد غير المجمعة يؤدي إلى الحصول على نتائج خاطئة و لذلك يجب تجميع الأعداد قبل قسمتها .

- تعليمة التصحيح AAD (و التي يتم تطبيقها قبل عملية التقسيم)

- مثال على فرض $AX = 0604$ طبق تعليمة ADD

- القانون :
 $AH.10d + AL \rightarrow AL$
 $00 \rightarrow AH$

$$\begin{aligned} AL &= 06 \times 10d + 04h = 64d = 40h \\ AH &= 00h \end{aligned}$$

$$\left. \begin{aligned} AL &= 06 \times 10d + 04h = 64d = 40h \\ AH &= 00h \end{aligned} \right\} \Rightarrow AX = 0040h$$

التعليمات المنطقية

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	$S.D \rightarrow D$	AND D,S	AND المنطقي	AND
أعلام الحالة	$S + D \rightarrow D$	OR D,S	OR المنطقي	OR
أعلام الحالة	$S \oplus D \rightarrow D$	XOR D,S	XOR المنطقي	XOR
لا يوجد	$\overline{D} \rightarrow D$	NOT D	NOT المنطقي	NOT

في هذه التعليمات يجب ان يكون هناك تناسق بيت كلا من المصدر والهدف

تعليلة AND

MOV AX,4211H

MOV BX,2221H

AND AX,BX

0100 0010 0001 0001

0010 0010 0010 0001

=0000 0010 0000 0001=0201H=AX

حالة واحدة يكون فيها الناتج $1 \cdot 1 = 1$

وباقى الاحتمالات تساوى 0

تعليلة OR

MOV AX,4211H

MOV BX,2221H

OR AX,BX

0100 0010 0001 0001

0010 0010 0010 0001

=0110 0010 0011 0001=6231H=AX

حالة واحدة يكون فيها الناتج $0+0=0$
وباقى الاحتمالات تساوى 1

تعلیمه XOR

MOV AX,4211H

MOV BX,2221H

XOR AX,BX

0100 0010 0001 0001

0010 0010 0010 0001

=0110 0000 0011 0000 = 6030H=AX

0 **XOR** 1 = 1 يكون ناتج يساوي 1 اذا كان البت مختلفان

1 **XOR** 0 = 1

تعلیمة NOT

- هذه التعلیمة تعمل على عكس قيمة البت من 0->1 او 1->0
- بمعنى ايجاد المتمم الحسابي الاول
- وبطريقة مباشرة اذا كان العدد يتكون من بايت أي ٨ بت نطرحه من FF
- واذا كان العدد يتكون من ٢ بايت ١٦ بت نطرحه من FFFF

MOV AX,4756H

NOT AX

AX=0100 0111 0101 0110

-AX= 1011 1000 1010 1001 = **B8A9H**

بطريقة أخرى مباشرة **B8A9** = FFFF-4756=

تعليمات الإزاحة

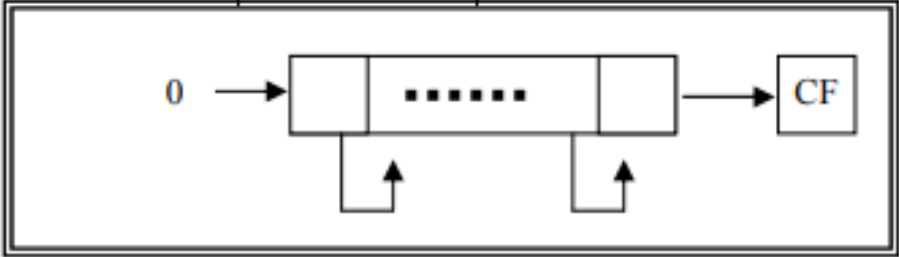
- هناك نوعان من تعليمات الإزاحة هما الإزاحة المنطقية و الإزاحة الرياضية كما هو واضح في الجدول التالي :

العلامة المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
OF, CF		SAL/SHL D, count	إزاحة رياضية/إزاحة منطقية و كلاهما نحو اليسار	SAL/SHL

- العملية هنا هي إزاحة محتويات D نحو اليسار باتجاه CF عدداً من الخانات مساوياً لقيمة count و ملء جميع الخانات اليمنى المفرغة بأصفار .
- و بالنسبة لتأثير هذه التعليمات على علم OF : إذا تبدلت خانة الإشارة نتيجة الإزاحة فإن $OF. = 1$

تعليمات الإزاحة

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
SHR	إزاحة منطقية نحو اليمين	SHR D,count		OF,CF



- العملية هنا هي إزاحة منطقية لمحتويات D نحو اليمين باتجاه CF عدد من الخانات مساوياً لقيمة count و ملء جميع الخانات اليسرى المفرغة بأصفار .
- و بالنسبة لتأثير هذه التعليمات على العلم : OF إذا تبذلت خانة الإشارة نتيجة الإزاحة فإن $Of = 1$

تعليمات الإزاحة

العلامة المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة		SAR D,count	إزاحة رياضية نحو اليمين	SAR

- العملية هنا هي إزاحة محتويات D نحو اليمين باتجاه CF عدداً من المرات مساوياً لقيمة count و ملء الخانات جميع الخانات اليسرى بقيمة الخانة (MSB)خانة الإشارة أو آخر خانة

تعليمات الإزاحة

- ملاحظة: بالنسبة للتعليمتين : SAL, SHL إذا طبقنا هاتين التعليمتين من أجل الإزاحة بعدد من الخانات $N = \text{count}$ فهذا يعني ضرب متحول الهدف بـ 2^N والذي هو مضاعف العدد 2^N .
- بمعنى على فرض هناك سلسلة من البتات ونريد الإزاحة بمقدار 2 الناتج بعد إجراء الإزاحة عبارة عن المعطى مضروب بـ 4
- مثال 16 0000 0001
- الناتج 0100 0000 64 كأنه $16 * 4 = 64$ لماذا 2 لانه 2 قوة 2 = 4
- على فرض نريد إزاحة الرقم 5 بمقدار 1 = 0101
- الناتج 1010 يساوي 10 وهذا يعني كل واحد إزاحة ضرب العدد بـ 2 بالعشري الإزاحة بمقدار 2 تعني ضرب العدد بـ 4
- يتم تعويض المراتب الفارغة في الإيعازات SHL, SHR, SAL بأصفار
- في الإيعاز SAR يتم تكرار البت الأخير بت الإشارة حسب مرات التكرار

تعليمات الإزاحة

- إن المتحولات المسموحة بالنسبة لتعليمات الإزاحة هي :

D	Count
Reg	1
Reg	CL
Mem	1
Mem	CL

- يمكن ان تكون الازاحة في مسجل بحيث وتكون للاشارة الى عدد مرات الازاحة ولا تتغير قيمة السجل
- عندما Count لا يساوي الواحد فعندئذ يجب تحميل قيمة count في المسجل CL ثم كتابة تعليمات الإزاحة
- لأن مسجل cx هو مسجل خاص في العداد ولا يجوز استخدام مسجل آخر

MOV CL,count

كون مسجل c| خاص في العد وهو يحتوي على ٨ بت قيمة العداد تتراوح بين (1-ff)

SAL AX,CL

تعليلة SAL,SHL

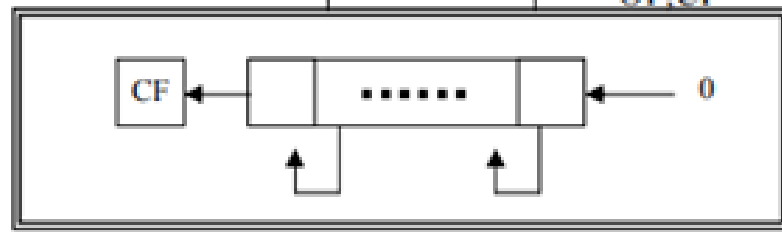
- الازاحة المنطقية لليسار.

MOV DH, 8Ah

MOV CL , 2

SHL DH,CL

DH = 1000 1010



- يمكن الحصول على DH الجديد من خلال مسح اخر خانتان في اقصى اليسار وتعويض بدلا منهن صفران اقصى اليمين
- قيمة DH=00101000
- قيمة CL لا تتغير
- قيمة CF=0 لانه اخر بت تم ازاحته
- قيمة OF=1 تكن 1 اذا كانت اخر عملية ازاحة أدت إلى 1
- الامر (SAL) Shift Arithmetic Left نفس آلية عمل SHL لكنه يستخدم في العمليات الحسابية

تعليلة SHR

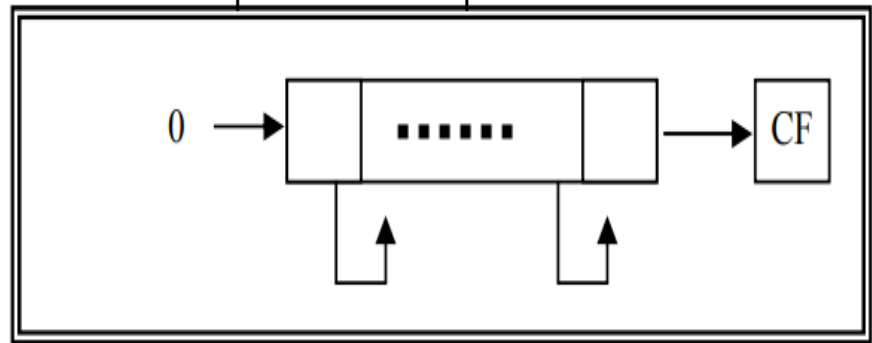
- تعليلة الازاحة لليمين المنطقية

MOV DH, 8Ah

MOV CL, 2

SHR DH, CL

DH = 1000 1010



- يمكن الحصول على DH الجديد من خلال مسح اول خانتان في اقصى اليمين وتعويض بدلا منهن صفران اقصى اليسار

• قيمة DH=00100010

• قيمة CL لا تتغير

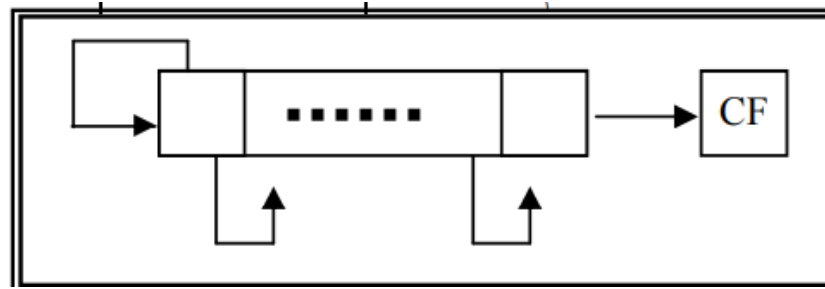
• قيمة CF=1 لانه اخر بت تم ازاحته

• قيمة OF=1 تكن 1 اذا كانت اخر عملية ازاحة أدت إلى 1

• إذا تبدلت خانة الإشارة نتيجة الإزاحة فإن OF = 1

تعليلة الازاحة الرياضية لليمين SAR

- ملاحظة ان اشارة الرقم بعد اجراء الازاحة باستخدام تعليلة الازاحة لليمين للعمليات الحسابية لا تتغير



- مثال

MOV DH, 8Ah

SAR DH,1

DH = 1000 1010

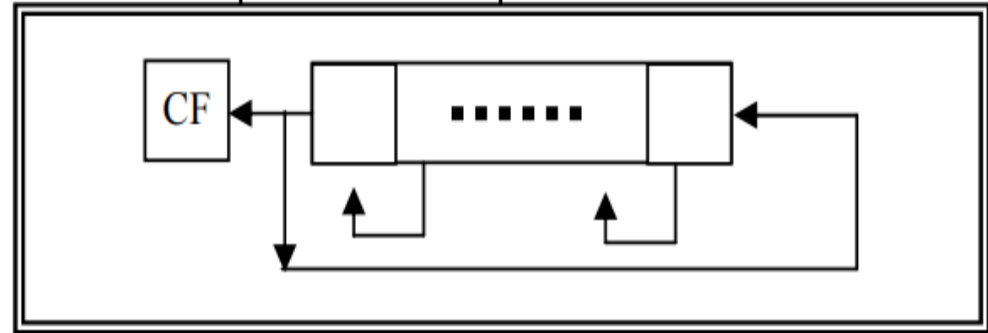
- تصبح قيمة DH=11000101

- قيمة CF=0

- OF=0

تعليمات التدوير اليسار ROL

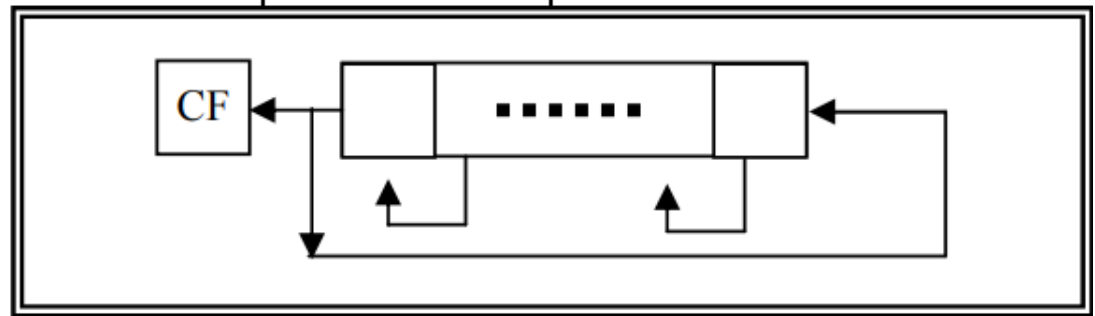
الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
ROL	تدوير نحو اليمين	$ROL D, count$		OF, CF



- العملية هنا هي تدوير محتويات D من اليسار عدداً من المرات مساوياً لقيمة count . كل خانة تتراوح خارج الـ MSB وتوضع في الخانة LSB و في CF.
- و بالنسبة لتأثير هذه التعليمات على العلم OF إذا تبذلت خانة الإشارة نتيجة الإزاحة فإن $OF = 1$

تعليمة التدوير لليسار ROL

مثال



• قيمة DH=00101010

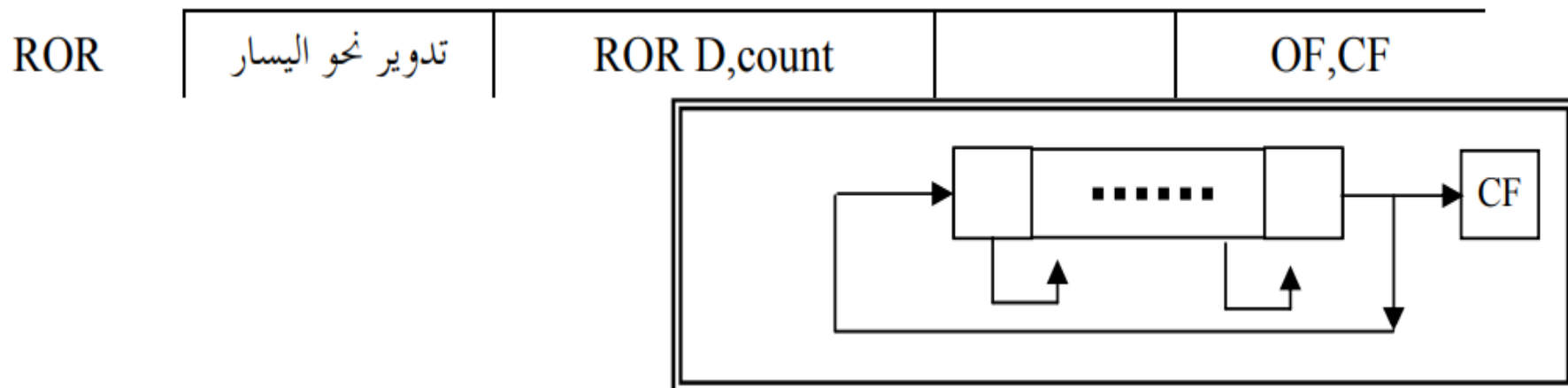
• قيمة CL لا تتغير

• قيمة CF=0 لأنه آخر بت تم تدويره هو 0

• قيمة OF=1 إذا تبديلت خانة الإشارة نتيجة التدوير فإن 1 = OF

التدوير لليمين ROR

- لعملية هنا هي تدوير محتويات D من اليمين عدداً من المرات مساوياً لقيمة count. كل خانة تزاح خارج ال-LSB وتوضع في الخانة MSB و في CF
- و بالنسبة لتأثير هذه التعليمة على العلم OF إذا تبدلت خانة الإشارة نتيجة الإزاحة فإن $OF = 1$



التدوير لليمين ROR

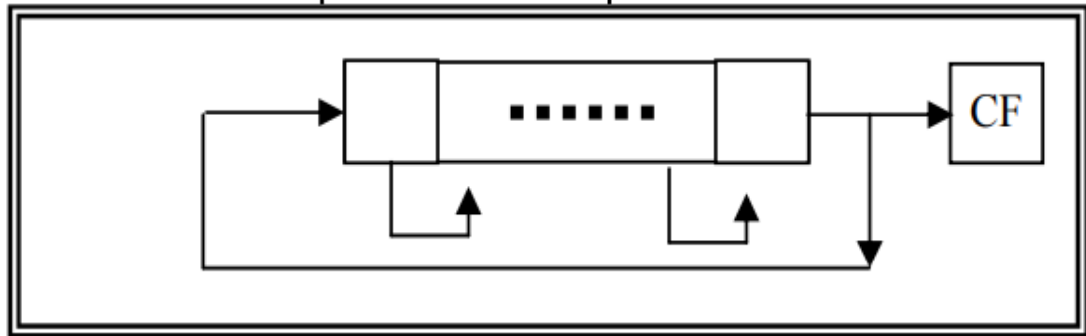
MOV DH, 8Ah

MOV CL, 2

ROR DH, CL

DH = 1000 1010

مثال



- قيمة DH=10100010

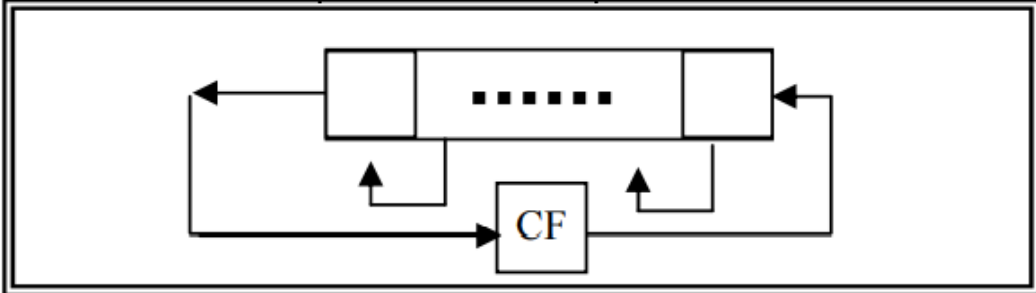
- قيمة CL لا تتغير

- قيمة CF=1 لأنه آخر بت تم تدويره هو ١

- قيمة OF=0 إذا تبديلت خانة الإشارة نتيجة التدوير فإن ١ OF=

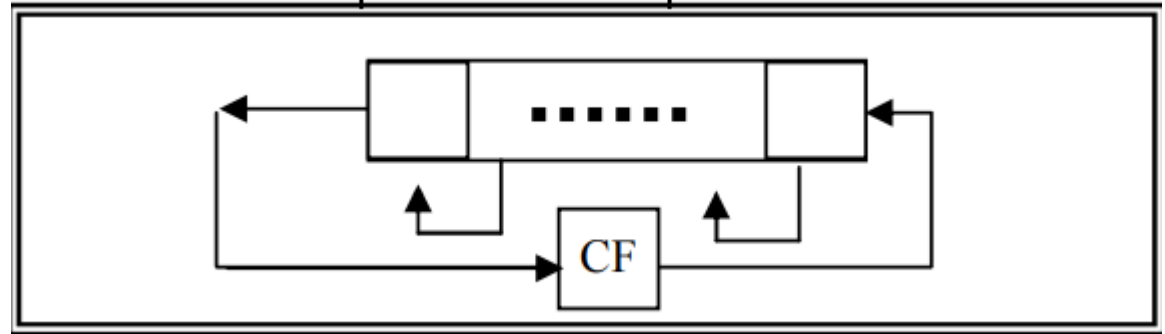
تعليلة تدوير اليسار RCL عبر CF

- العملية هنا مشابهة لتعليلة ROL ما عدا أن المحتوى الأصلي لـ CF يوضع في الخانة LSB أما الخانة المزاحة خارج الـ MSB فتوضع في CF

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
OF,CF		RCL D,count	تدوير نحو اليسار عبر الـ CF	RCL
<div></div> <div>تدوير من اليسار نحو اليمين عبر cf</div>				

تعليلة تدوير اليسار RCL عبر CF

في هذا التدوير تكن قيمة CF محدد مسبقاً



• قيمة DH=00010101

• قيمة CL لا تتغير

• قيمة CF=1 لانه اخر بت تم تدويره هو ١

هذا ازاحة واحدة ازاحة الثانية نكرر العملية

قيمة DH=00101011

قيمة CF=0

قيمة OF=1 إذا تبذلت خانة الإشارة نتيجة التدوير فإن ١ OF=

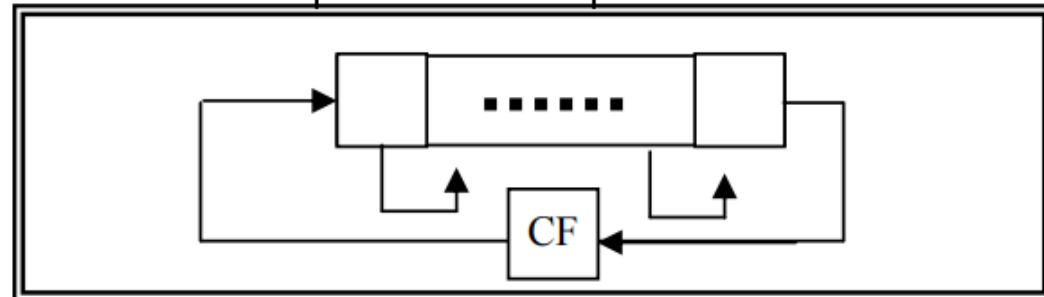
تدوير اليمين RCR عبر الـ cf

- عملية هنا مشابهة لتعليمة ROR ما عدا أن المحتوى الأصلي لـ CF يوضع في الخانة MSB أما الخانة المزاحة خارج الـ LSB فتوضع في CF
- قيمة $OF=1$ إذا تبدلت الإشارة نتيجة التدوير فإن ١

$OF=$

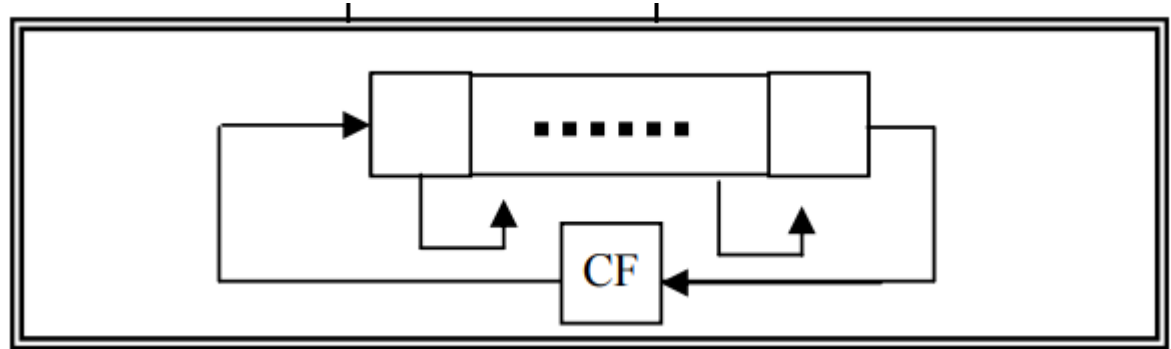
الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
RCR	تدوير نحو اليمين عبر الـ CF	RCR D, count		OF, CF

تدوير من اليمين نحو اليسار عبر CF



تدوير نحو اليمين عبر الـ cf

في هذا التدوير تكن قيمة CF محدد مسبقاً



• قيمة DH=11000101

• قيمة CL لا تتغير

• قيمة CF=0 لأنه آخر بت تم تدويره هو 0

هذا ازاحة واحدة ازاحة الثانية نكرر العملية

قيمة DH=01100010

قيمة CF=1

قيمة OF=1 إذا تبديلت خانة الإشارة نتيجة التدوير فإن 1 OF=

;CF=1

MOV DH, 8Ah

MOV CL, 2

ROR DH, CL

DH = 1000 1010

لا تكون $TF = 1$ تكون قمتها إلى عندما نريد تنفيذ البرنامج خطوة
خطوة وتكون صنف في عملية الكسوف الأخطاء لا تصححها أدل بأدلى

ثانياً علم المقاطعة Interrupt flag IF
× هي أخبار المخرج أنه العملية الآتية من IF الأربعة قدما وقاطع كفة
التدريج

لا تكون قمتها $IF = 1$ عندما لا نرغب في تنفيذ أي مقاطعة
× - - - $IF = 0$ تكون المقاطعة ممتدة

ثالثاً علم الاتجاه Direction flag DF
× $DF = 1$ اتجاه سير العملية يكون من العنوان الأعلى إلى العنوان الأدنى
× $DF = 0$ اتجاه سير العملية يكون من العنوان الأدنى إلى العنوان الأعلى

تعليمات مسجلات الأعلام

- نلاحظ ان هذه التعليمات لا يوجد فيها معاملات

هو مسجل ذو 16 بت موجود في وحدة التنفيذ كما هو واضح بالشكل :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

الكلمة المختارة	المعنى	الصيغة	العملية	الأعلام المتأثرة
LAHF	تحميل AH من مسجل الأعلام	LAHF	Flags → AH النصف الأول من مسجل الأعلام يوضع في AH	لا يوجد
SAHF	تخزين قيمة AH في مسجل الأعلام	SAHF	AH → Flags يوضع AH في النصف الأول من مسجل الأعلام	أعلام الحالة عدا OF

تعليمات مسجلات الأعلام

- يمكن من خلال هذه التعليمات التغيير في محتوى مسجل
الاعلام CF

CLC	تنظيف الـ CF	CLC	$0 \rightarrow CF$	CF
STC	توضيع الـ CF	STC	$1 \rightarrow CF$	CF
CMC	متمم أحادي لـ CF	CMC	$\overline{CF} \rightarrow CF$	CF
CLI	تنظيف الـ IF	CLI	$0 \rightarrow IF$	IF
STI	توضيع الـ IF	STI	$1 \rightarrow IF$	IF

تعليمات المقارنة CMP

compare

- تسمح تعليمة المقارنة CMP بمقارنة عددين ب ٨ بت أو ١٦ بت و هي مشروحة بالجدول التالي

الأعلام المتأثرة	العسلية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	$D - S$ تتأثر الأعلام	CMP D,S	مقارنة عددين	CMP

- يقوم البرنامج في عملية المقارنة من خلال طرح الهدف من المصدر ولا يتم تخزين النتيجة لكن **مسجل الأعلام تتأثر** أي أي تبقى كلاً من محتويات المصدر S و محتويات الهدف D على حالها.
- تستعمل هذه التعليمة لجعل أعلام الحالة تأخذ قيمة واحد منطقي أو صفر منطقي وإذا كان **نتيجة ١ نفذ الذي يلي هذه التعليمة** بناء على الأمر التالي يتأثر في أي من الرايات

تعليمات المقارنة CMP

- إن المتحولات المسموحة لهذه التعليمة مبينة في الجدول التالي
- نلاحظ ان لا يمكن ان يكون الهدف عدد ثابت
- نلاحظ كذلك لا يمكن ان يكون كلا من الهدف والمصدر موقع في الذاكرة
- تعليمة CMP هي عبارة عن تعليمة SUB لكن لا تخزن نتيجة

MOV AL,1EH

CMP AL,0DH

JNE @REPEAT2

..

..

@REPEAT2: INC AX

D	S
Reg	Reg
Reg	Mem
Mem	Reg
Reg	Imm
Mem	Imm
Acc	Imm

تعليمات القفز

- الغاية من تعليمة القفز هي تعديل طريق تنفيذ التعليمات في البرنامج .
- هناك نوعان من تعليمات القفز، وهي : القفز المشروط و القفز غير المشروط .
- في القفز غير المشروط لا يوجد أي شروط من أجل حدوث القفز.
- القفز المشروط فإن الحالات الشرطية الموجودة في لحظة تنفيذ تعليمة القفز تتخذ القرار فيما إذا سيحدث القفز أم لا، ففي حال تحقق الحالات الشرطية فإنه يتم القفز، و إلا يتابع التنفيذ بالتعليمة التي تلي تعليمة القفز في البرنامج

تعليلة القفز غير المشروط JMP

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	القفز إلى العنوان المحدد	JMP operand	قفز غير مشروط	JMP

- هناك نوعان أساسيان من القفز غير المشروط:

1. القفز ضمن المقطع الجزئي:

- القفز ضمن المقطع الجزئي فإنه يتطلب منا تعديل قيمة ال IP فقط

1. القفز ضمن مقاطع البرنامج كالقفز من مقطع الكود الى مقطع آخر

- يتطلب منا تعديل محتويات كل من مقطع ال CS و مسجل مؤشر التعليم IP

➤ لا تؤثر هذه التعليمات على حالة الاعلام

تعليلة القفز غير المشروط JMP

```
@BEGIN: MOV AH , 2
```

```
..
```

```
...
```

```
....
```

```
JMP @BEGIN
```

تعليلة القفز غير المشروط

- المتحولات المسموحة لتعليلة القفز غير المشروط هي :

	Operand		
متحول الالفة القصيرة	Short_Label	}	للقفز ضمن المقطع الجزئي
متحول الالفة القريبة	Near_Label		
متحول مؤشر ذاكري 16 بت	Memptr16		
متحول مؤشر مسجلي 16 بت	Regptr16		
متحول الالفة البعيدة	Far_Label	}	للقفز بين المقاطع الجزئية
متحول مؤشر ذاكري 32 بت	Memptr32		

القفز ضمن المقطع الجزئي

- إن متحولات اللافتة ال قصيرة و اللافتة القريبة تحدد القفز النسبي لعنوان تعليمة القفز نفسها
- اللافتة القصيرة أي القفز ضمن نفس الاجراء
- الفتة القريبة القفز من اجراء على اجراء أي برنامج فرعي لكن ضمن نفس مقطع الكود
- القيمة التي يحملها المعامل مع تعليمة القفز هي عبارة عن قيمة فورية تشير الى الاذاحة للتعليمة المراد القفز اليها
- من الطبيعي ان المترجم عندما يصل تعليمة القفز يكن له قيمة IP اما وعند القفز ينتقل او يتغير عنوان ال IP
- إن متحولات اللافتة ال قصيرة و اللافتة القريبة تحدد القفز النسبي لعنوان تعليمة القفز نفسها فمثلاً في تعليمة القفز باللافتة القصيرة يتم تشفير العدد ذي ٨ بت كمتحول فوري لتحديد الإزاحة (Disp) ذات الإشارة التي تشير إلى التعليمة التالية التي سيتم تنفيذها من حجرة تعليمة القفز، و عندما تنفذ تعليمة القفز يعاد شحن ال IP بقيمة جديدة موضحة كما يلي

القفز ضمن المقطع الجزئي

- كيف يتم حساب قيمة IP الجديدة :

قيمة IP الجديدة = [قيمة IP + طول شيفرة تعليمة القفز) + مقدار الإزاحة ذات الإشارة بعد تمديدها بجعل متحول 8 بتات بالشكل 16 بت]

- إن القيمة الجديدة لـ IP هي التي تحدد القفز

مثال

- مثال : ليكن لدينا

IP = 0112h

JMP disp ; disp = 0F2h =0112+2-F2

= IP + 2 + disp (بعد تمديد إشارتها) = 0112 + 2 + FFF2 = 0106h (أهملنا خانة الحمل)

- بما أن العنوان الناتج أصغر من عنوان تعليمة القفز فهذا يعني أننا نقفز إلى تعليمة تسبق تعليمة القفز أي القفز نحو الوراء . ٠١٠٦ < ٠١١٢ >

- مثال آخر : IP = 0112h JMP 04 Address = 0112 + 2 +

0004 = 0118h

- نلاحظ أن ٠١١٢ < ٠١١٨ فهذا يعني أن القفز نحو الأمام

- ملاحظة: بما أن متحول اللافتة القصيرة ذو ٨ بتات فهو يسمح بالقفز و سبب ذلك أنه إذا أضفنا طول شيفرة تعليمة القفز و هو ٢ بايت
- اللافتة القريبة فهو متحول فوري ذو ١٦ بت و لذلك يسمح بالقفز ضمن مجال يساوي ٣٢ KB نحو الخلف أو نحو الأمام من عنوان تعليمة القفز
- يمكن تحديد القفز إلى عنوان بشكل غير مباشر بواسطة محتويات حجرة ذاكرة أو محتويات مسجل أي ب استخدام متحول مؤشر ذاكري ١٦ بت أو متحول مؤشر مسجلي ١٦ بت و هنا أيضاً يتم القفز ضمن مجال ٣٢ KB

استخدام متحول مؤشر ذاكري ١٦ بت أو متحول مؤشر مسجلي ١٦ بت

JMP BX في هذه التعليمة يستعمل مضمون المسجل BX من أجل الإزاحة و هذا يعني أن قيمة BX يتم تحميلها في IP ثم كونه في نفس القطاع فإن قيمة القيمة الجديدة لـ IP هي من تحدد مكان القفز

BX = 0200h

DISP= 0100h

JMP BX

- يمكن استخدام مختلف أنواع أنظمة العنونة لتحديد المتحول المستعمل كمؤشر ذاكري فمثلاً `JMP[SI]` في هذه التعليمة تستعمل محتويات `SI` كعنوان حجرة الذاكرة التي تحتوي على العنوان الفعال، هذا العنوان يتم تحميله في `IP` و ومع مقدار الازاحة `+3` تحدد التعليمة التي سيتم القفز إليها و عادة في هذه الحالة تستخدم المسجلات التالية `BX, SI, DI`

القفز بين المقاطع الجزئية أو القفز خارج المقطع الجزئية

- تستعمل اللافتة البعيدة متحولاً فورياً ذات ٣٢ بت لتحديد القفز إلى عنوان ما. حيث يتم تحميل الـ ١٦ بت الأولى من هذا المتحول في IP أو تكون هي العنوان الفعال نسبة لمحتويات المسجل CS أما الـ ١٦ بت الثانية فيتم تحميلها في المسجل CS و التي تحدد مقطع الشيفرة الجديد

JMP farlabel

حيث farlabel هو متحول بـ ٣٢ بت (الكلمة الأول تشحن في IP أو الكلمة الثانية تشحن في الـ CS)

القفز بين المقاطع الجزئية أو القفز خارج المقطع الجزئية

- إن الطريقة غير المباشرة لتحديد العنوان الفعال و عنوان مقطع الشيفرة من أجل القفز بين المقاطع الجزئية هي باستعمال متحول مؤشر ذاكري ب ٣٢ بت . و في هذه الحالة فإن أربع بايتات من الذاكرة متتابعة اعتباراً من العنوان المحدد تحتوي على العنوان الفعال و عنوان مقطع الشيفرة الجديد على الترتيب
- `JMP farseg[DI]` ففي هذه التعليمة تستعمل محتويات `DI, DS` لحساب عنوان حجرة الذاكرة التي تتضمن الكلمة الأولى للمؤشر الذي يعرف الحجرة التي سيتم القفز إليها..

$$\left\{ \begin{array}{l} DI = 0200h \\ DS = 0100h \end{array} \right. \quad \text{إن العنوان الفيزيائي للمؤشر هو :}$$
$$PA = DS \times 10h + DI = 01000 + 0200 = 01200h$$

القفز بين المقاطع الجزئية أو القفز خارج المقطع الجزئية

و لتكن محتويات هذه الحجرة و الحجرات التي تليها كما هو واضح في الشكل التالي:

Address (h)	Content
01200	10
01201	30
01202	00
01203	04

قيمة IP الجديدة هي $IP = 3010h$

قيمة CS الجديدة هي $CS = 0400h$

إذن العنوان الفيزيائي للتعليمية التي سيتم القفز إليها هو:

$$PA = CS \times 10h + IP = 07010h$$

تعليمة القفز المشروط

- هي مشروحة في الجدول التالي :

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	إذا تحقق الشرط CC فإنه يتم القفز إلى العنوان المحدد بواسطة المتحول و إلا فيتم تنفيذ التعليمة التالية لتعليمة القفز	متحول Jcc	قفز مشروط	Jcc

تعليمة القفز المشروط

- هناك ١٨ من تعليمات القفز المشروط و هي مشروحة في الجدول التالي

المعنى	الكلمة المختزلة
القفز إذا كان $CF = 1$	JC
القفز إذا كان $CF = 0$	JNC
القفز إذا كان $OF = 1$	JO
القفز إذا كان $OF = 0$	JNO
القفز إذا كان $SF = 1$	JS
القفز إذا كان $SF = 0$	JNS
القفز إذا كان $CX = 0000$	JCXZ

تعليمة القفز المشروط

- هناك ١٨ من تعليمات القفز المشروط و هي مشروحة في الجدول التالي

JE/JZ	القفز في حالة التساوي/أو إذا كان الناتج يساوي الصفر
JGE/JNL	القفز إذا كان أكبر أو يساوي/القفز إذا لم يكن أصغر
JA/JNBE	القفز إذا كان فوق/القفز إذا لم يكن تحت أو يساوي
JAE/JNB	القفز إذا كان فوق أو يساوي/القفز إذا لم يكن تحت
JB/JNAE	القفز إذا كان تحت/القفز إذا لم يكن فوق أو يساوي
JBE/JNA	القفز إذا كان تحت أو يساوي/القفز إذا لم يكن فوق
JG/JNLE	القفز إذا كان أكبر/القفز إذا لم يكن أصغر أو يساوي
JLE/JNG	القفز إذا كان أصغر أو يساوي/القفز إذا لم يكن أكبر
JNE/JNZ	القفز إذا لم يكن يساوي/القفز إذا كان الناتج يساوي قيمة غير صفرية
JNB/JBO	القفز إذا كانت خانة Parity غير موجودة/القفز إذا كان $PF = 0$
JP/JPE	القفز في حالة وجود خانة Parity/القفز إذا كان $PF = 1$

تعليمة القفز المشروط

- ملاحظة : للتمييز بين مقارنة الأعداد ذات الإشارة و الأعداد بدون إشارة فإن هناك اسمين مختلفين يبدو أما نفس الشيء في تعليمات القفز و هما فوق (A) و تحت (B) من أجل مقارنة الأعداد بدون إشارة.
- و أصغر (L) و أكبر (G) من أجل مقارنة الأعداد ذات الإشارة

البرامج الفرعية

- هي إجراءات مكتوبة بشكل مستقل عن البرنامج الرئيسي.
- متى وجب على البرنامج الرئيسي أن ينجز الوظيفة المحددة بواسطة البرنامج الفرعي فإنه يستدعي البرنامج الفرعي إلى العمل و من أجل هذا يجب أن يتحول التحكم من البرنامج الرئيسي إلى نقطة البداية في البرنامج الفرعي.
- حيث يستمر تنفيذ البرنامج الفرعي، و عند اكتمال التنفيذ يعود التحكم إلى البرنامج الرئيسي بالتعليمة التالية لتعليمة مناداة البرنامج الفرعي

NEXT PROC FAR, NEAR

.....

.....

.....

NEXT ENDP

بين العمل لمناداة البرنامج الفرعي و القفز

- ملاحظة : إن الفرق بين العمل لمناداة البرنامج الفرعي و القفز هو أن مناداة البرنامج الفرعي لا تنتج قفزاً فقط إلى العنوان المناسب في ذاكرة تخزين البرنامج و لكنها أيضاً تملك تقنية من أجل حفظ المعلومات مثل IP او CS التي تكون مطلوبة للعودة إلى البرنامج الرئيسي .

تعليمات المناداة و العودة

- كلاً هاتين التعليمتين معاً تزودان تقنية من أجل استدعاء البرنامج الفرعي إلى العمل و إعادة التحكم إلى البرنامج الأساسي لمتابعة تنفيذه.

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	يُتابع التنفيذ في البرنامج الفرعي من العنوان المحدد بواسطة المتحول operand الموجود في تعليمة المناداة. و المعلومات المطلوبة من أجل العودة مثل IP و CS تُحفظ في المكس	CALL operand	مناداة برنامج فرعي	CALL

تعليمات المناداة و العودة

- هناك ٥ أنواع للمتحويلات المسموح باستخدامها مع تعليمة المناداة و هي :

OPERAND	
Near_pro	}
Memptr16	
Regptr16	
Far_proc	}
Memptr32	

للمناداة ضمن المقطع الجزئي

للمناداة خارج المقطع الجزئي

تعليمات المناداة و العودة

- مناداة ضمن المقطع الجزئي للبرنامج الفرعي (أي البرنامج الرئيسي و البرنامج الفرعي يقعان في نفس مقطع ال شيفرة)
- حيث أن تنفيذ تعليمة المناداة يسبب حفظ محتويات IP في المكس
- لأنه سوف يتم تعديل قيمة IP آلياً لتتلاءم مع البرنامج الفرعي وعندئذ ينقص مؤشر المكس بمقدار ٢
- إن القيمة المحفوظة في IP ضمن المكس هي عنوان التعليمة التي تلي تعليمة المناداة

المناداة خارج المقطع الجزئي

- فهو يسمح للبرنامج الفرعي بأن يكمن في مقطع شيفرة آخر، و في هذه الحالة تستخدم المتحولات التالية `pro_Far`، `MeMTR32`
- تحدد هذه المتحولات كلاً من العنوان الجديد لـ `IP` و عنوان المقطع الجديد لـ `CS`
- كلتا الحالتين فإن تنفيذ تعليمة المناداة يسبب حفظ محتويات المسجلات `CS` ثم `IP` في المكس و من ثم تحميل القيم الجديدة المحددة بالمتحول `operand` في `IP` و `CS`
- إن القيم المخزنة لـ `CS` و `IP` في المكس تسمح بالعودة إلى البرنامج الرئيسي من مقطع شيفرة آخر
- إن المتحول `proc_Far` يمثل متحولاً فورياً بـ ٣٢ بت و الذي يكون مخزناً في البايتات الأربعة
- `CALL01234321` حيث أن هاتان الكلمتان يتم تحميلهما مباشرة من ذاكرة تخزين البرنامج في `IP` و `CS` حيث `CS` هو مقطع الشيفرة للبرنامج الفرعي. إن عنوان التعليمة الأولى في البرنامج الفرعي يكون محدداً بالكلمة الأولى بعد تعليمة `CALL` أي يخزن ضمن `IP`..

تعليلة العودة RET

- إن كل برنامج فرعي يجب أن ينتهي بتنفيذ التعليلة التي تعيد التحكم إلى البرنامج الرئيسي و هذه التعليلة هي تعليلة العودة RET

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		RET/RET operand	العودة إلى البرنامج المستدعي	RET

- العودة إلى البرنامج المستدعي عن طريق إعادة تخزين قيم IP فقط أو IP أو CS معاً (حسب نوع تعليلة المناداة أي ضمن المقطع الجزئي أو خارجه) من أجل المتحول pro_F
- و إذا كان المتحول operand موجوداً في تعليلة العودة RET فيجب إضافته إلى محتويات SP هذا و إن المتحول إذا وجد في تعليلة العودة فهو عبارة عن متحول إزاحة ب ١٦ بت

تعليمات قطاع المكس POP,PUSH

- إن التعليمة المستخدمة لحفظ البارامترات في المكس هي تعليمة الدفع PUSH و التعليمة المستخدمة لاسترجاعها هي تعليمة POP وتكن سحب البيانات بطريقة الداخل آخر ا خارج أولا LIFO
- ملاحظة: يتعامل المكس مع كلمات و ليس مع بايتات

POP MM

PUSH MM

- يستخدم المكس لحفظ البيانات بشكل مؤقت اثناء تنفيذ البرنامج التي قد نحتاج لها .

PUSH, POP

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	$S \rightarrow ((SP))$	PUSH S	دفع <u>كلمة</u> إلى المكس	PUSH
لا يوجد	$((SP)) \rightarrow D$	POP D	سحب <u>كلمة</u> من المكس	POP

- أثناء عمليات المقاطعة ومناداة البرنامج الفرعي يتم دفع محتويات المسجلات الداخلية المعاينة بالمعالج إلى قسم من الذاكرة يدعى بالمكدس حيث تبقى هذه المحتويات هناك بشكل مؤقت
- فمثلاً عندما تحدث المقاطعة فإن المعالج و بشكل أوتوماتيكي يدفع بمسجل الأعلام if معناه خلصت المقاطعة، القيمة الحالية في CS، و القيمة الحالية في IP إلى المكس.

PUSH, POP

- و العنوان المشتق من محتويات SS و SP هو العنوان الفيزيائي لحجرة التخزين الأخيرة في المكس (قمة المكس) التي تمّ دفع المعطيات إليها.
- إن القيمة في مؤشر المكس تبدأ ب FFFFh بشكل معكوس
- بما أن المعطيات المنقولة من و إلى المكس عادة هي كلمات فإننا نتصور المكس على شكل حجرات ذات ٢ بايت، كما أنه من الضروري أن تكون جميع حجرات المكس في حدود الكلمات الزوجية و ذلك لإنقاص عدد دورات الذاكرة المطلوبة لدفع أو سحب المعطيات من المكس .
- يقوم المعالج بدفع المعطيات و العناوين إلى المكس كلمة في كل مرة، و في كل مرة يتم دفع قيمة مسجل ما إلى قمة المكس فإن القيمة في مؤشر المكس أولاً تنقص بمقدار ٢

PUSH, POP

- في هذه الطريقة فإن المكس ينمو نحو الأسفل في الذاكرة انطلاقاً من قاعدة المكس FFFFH الى 0000H أي كلما اتجهنا لأسفل يزيد حيز الاستعاب للمكس
- . إن العنوان الفيزيائي المعروف بواسطة SS و SP دائماً يشير إلى حجرة القيمة الأخيرة المدفوعة إلى المكس حيث أن محتوياته تسحب أولاً من المكس إلى المسجل المعني ضمن المعالج ثم يزداد SP بمقدار ٢

MOV AX,4422H

PUSH AX

POP BX

تعليمات الحلقات الدوران

- هناك ثلاث تعليمات مصممة بشكل خاص لتحقيق عملية الحلقة . و هذه التعليمات يمكن استعمالها بدلاً من تعليمات القفز الشرطي
- عند استخدام الدوران يجب استخدام مسجل العداد CX لتحديد عدد مرات التكرار حيث ينقص بشكل تلقائي بمقدار ١ دون التأثير على الأعلام ثم القفز إلى الحجرة المعرفة بواسطة الالفة القصيرة إذا كان CX لا يساوي الصفر و إلا يتم تنفيذ التعليمة التالية لتعليمة الحلقة.

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختلة
لا يوجد	{	LOOP short_label	حلقة	LOOP

تعليمات الحلقات الدوران

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LOOPE/ } لافتة قصيرة LC	حلقة طالما يساوي/ أو طالما صفر	LOOPE/ LOOPZ

إنقاص CX بمقدار واحد دون التأثير على الأعلام ثم القفز إلى الحجرة المعرفة بواسطة الالفتة القصيرة إذا

كان CX لا يساوي الصفر و ZF يساوي الصفر و إلا يتم تنفيذ التعليمة التالية لتعليمة الحلقة.

و هنا جسم الحلقة فقط هو الذي يؤثر على الأعلام.

ZF = 1
CX NOT 0

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LOOPNE/ } لافتة قصيرة LOOP	حلقة طالما لا يساوي/ أو طالما ليس صفراً	LOOPNE/ LOOPNZ

إنقاص CX بمقدار واحد ثم القفز إلى الحجرة المحددة بواسطة الالفتة القصيرة إذا كان CX لا يساوي الصفر و

ZF = 0
CX NOT 0

ZF يساوي الصفر و إلا يتم تنفيذ التعليمة التالية لتعليمة الحلقة.

و هنا أيضاً جسم الحلقة فقط هو الذي يؤثر على الأعلام.

تعليمات الحلقات الدوران

- جسم الدوران

```
MOV CX, 5  
Nxt: -----  
      -----  
      -----  
      LOOPNE Nxt
```

} جسم الحلقة

تعليمات السلسلة Moving String

- نقصد بكلمة السلسلة أن بايتات أو كلمات معطيات تكمن في حجرات متعاقبة للذاكرة .
- إن تعليمات السلسلة تسمح للمبرمج بتنفيذ عمليات مثل نقل المعطيات من بلوك ذاكرة إلى بلوك آخر في الذاكرة، مسح أو كنس SCAN سلسلة من عناصر المعطيات المخزنة في الذاكرة و البحث عن قيمة معينة
- مقارنة عناصر سلسلتين لتحديد فيما إذا كانا متطابقتين أو مختلفتين و تعليمات السلسلة الأساسية.

تعليمات السلسلة

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		MOVS operand	نقل عنصر من سلسلة	MOVS

العملية: العنصر المحدد بواسطة DS:SI يتم نقله إلى الحجرة المحددة بواسطة القيمة ES:DI ثم:

$SI \pm 1 \text{ or } 2 \rightarrow SI$
 $DI \pm 1 \text{ or } 2 \rightarrow DI$

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	نفس العملية السابقة و مقدار التزايد هو 1	MOVSB	نقل عنصر بايت من سلسلة	MOVSB
لا يوجد	نفس العملية السابقة و مقدار التزايد هو 2	MOVSW	نقل عنصر كلمة من السلسلة	MOVSW
أعلام الحالة		CMPS operand	مقارنة عنصر سلسلة	CMPS

تعليمات السلسلة

العملية: يتم طرح متحول الهدف من متحول المصدر و لا تُخزن النتيجة إنما تُعدل أعلام الحالة فقط، أي:

$$((DS \times 10h) + SI) - ((ES \times 10h) + DI) \rightarrow \text{أعلام الحالة}$$

$$SI \pm 1 \text{ or } 2 \rightarrow SI$$

$$DI \pm 1 \text{ or } 2 \rightarrow DI$$

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
SCAS(B or W)	مسح عنصر سلسلة	SCAS operand		أعلام الحالة

العملية :

$$(AL \text{ or } AX) - ((ES \times 10h) + DI) \rightarrow \text{أعلام الحالة}$$

$$DI \pm 1 \text{ or } 2 \rightarrow DI$$

Cmp al or ax, [di]

تعليمات السلسلة

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LODS operand	تحميل عنصر سلسلة	LODS (B or W)
<div>العملية :</div> $((DS \times 10h) + SI) \rightarrow (AL \text{ or } AX)$ $SI \pm 1 \text{ or } 2 \rightarrow SI$				<div>عناصر وليس عنصر</div>
الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		STOS operand	تخزين عنصر سلسلة	STOS(B or W)
<div>العملية :</div> $(AL \text{ or } AX) \rightarrow ((ES \times 10h) + DI)$ $DI \pm 1 \text{ or } 2 \rightarrow DI$				<div>عناصر وليس عنصر</div>

تعليمات تكرار السلسلة repit

- في معظم التطبيقات يجب تكرار العمليات الأساسية للسلسلة من أجل معالجة جميع عناصرها . و يتم إنجاز هذا العمل بواسطة إدخال تعليمات التكرار قبل التعليمة الأساسية للسلسلة التي سوف تتكرر

الكلمة المختزلة	المعنى	الاستخدام
REP	التكرار طالما لم نصل إلى نهاية السلسلة أي $CX \neq 0$	MOVS, STOS
REPE/REPZ	التكرار طالما لم نصل إلى نهاية السلسلة و السلسلتان متساويتان أي $ZF=1, CX \neq 0$	CMPS, SCAS
REPNE/REPNZ	التكرار طالما لم نصل إلى نهاية السلسلة و السلسلتان غير متساويتان أي $ZF=0, CX \neq 0$	CMPS, SCAS

مثال:

بفرض أن :

SI = 0100h	DS = 0200h
DI = 0110h	ES = 0400h

فإن نتيجة تنفيذ التعليمتين التاليتين :

```
MOV CX,20h  
REP MOVSB
```

هي أن التعليمة الأولى تقوم بتحميل المسجل CX بالقيمة ٣٢ d = 20h أما التعليمة الثانية فتنقل ٣٢ بايت من حجرات ذاكرة المصدر المحددة بواسطة DS و SI إلى بلوك حجرات ذاكرة الهدف المحددة بواسطة ES و DI

movsb

```
x db "musleh"
y db 6 dup (?)

.code

mov ax,@data
mov ds,ax

mov si,offset x
mov di,offset y

mov cx , 6

movsbb : mov al,[si]
          mov [di],al

          inc si
          inc di

          loop movsbb

ret
```

```
org 100h ; set location counter to 100h

.data

x db "musleh"
y db 6 dup (?)

.code

mov ax,@data
mov ds,ax

mov si,offset x
mov di,offset y

mov cx , 6

movsbb :

movsb

    loop movsbb

ret
```

يمكن الاستغناء عن الجزء
المحدد باستخدام movsb

آلية العمل movsw, movsb

- آلية العمل بالبداية نستخدم مؤشر si للإشارة إلى العنصر الذي نريد نقله ومؤشر di للمكان الذي نريد نقل العنصر له
- ومن ثم يتم زيادة مؤشر si بمقدار ١ أو ٢ على حسب اذا تم تعريف عملية نقل السلسلة أو المصفوفة ١ بايت أو ٢ بايت
- ومن ثم يمكن انقاص مؤشر si بمقدار ١ أو ٢ على حسب اذا تم تعريف عملية نقل السلسلة أو المصفوفة ١ بايت أو ٢ بايت.
- كيف يتم تحديد هل نزيد أو ننقص من خلال مسجل التحكم DF
 - اذا $DF=0$ هذا يعني اتجاه النقل أو القراءة من اليسار بمعنى أي نزيد
 - اذا كان $DF=1$ هذا يعني اتجاه القراءة يبدأ من اليمين وهذا بمعنى نطرح ١ أو ٢
 - كيف يمكن التحكم في DF من خلال التعليمات التي تم شرحهم أي
 - ✓ $CLD=DF=0$
 - ✓ $STD=DF=1$
- ❖ الفرق بين البايت والكلمة هو ان نزيد او نطرح بمقدار ٢ أي ان si,di تزداد بمقدار ٢ او تنقص

آلية العمل movsb

في حال نبدأ النقل من الجزء الايمن

```
x db "musleh"  
y db 6 dup (?)
```

```
.code
```

```
mov ax, @data  
mov ds, ax
```

```
mov si, offset x+5  
mov di, offset y+5
```

```
mov cx, 6
```

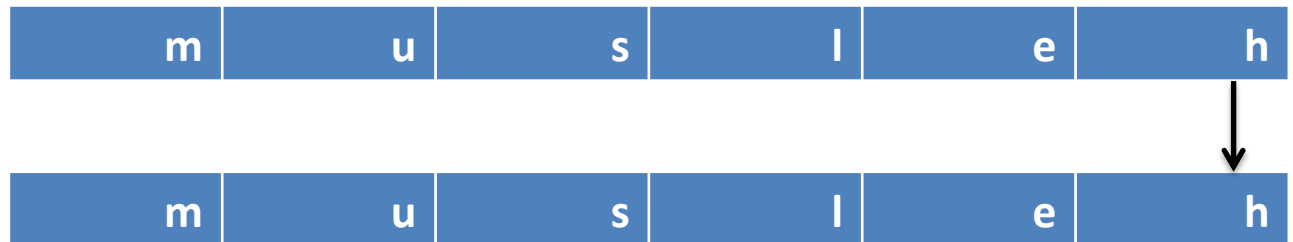
```
movsbb :
```

```
STD ; DF=1
```

```
movsb
```

```
loop movsbb
```

```
ret
```



نلاحظ ان df=1 بمعنى نبدأ من الشق
الايمن في عملية النقل

Load string byte

- نلاحظ تم استخدام التكرار الخاص
- في معالجة السلاسل بالاضافة
- بالإضافة الى تعليمة نقل محتويات
- Si إلى al

```
x db "1","2","3"
```

```
.code
```

```
mov bx,@data
```

```
mov ds,bx
```

```
mov si,offset x
```

```
mov cx,3
```

```
rep lodsb
```

```
hlt
```

storb

```
x db "1","2","3"

.code

mov bx, @data
mov ds, bx
|
mov di, offset x+3
mov al, "3"

    cld

    stosb

    hlt
```

أكتب برنامج يعمل على مقارنة سلسلتين إذا كانت متساويتين يطبع «y» إذا كانت مختلفتين يخرج

- علما أن السلسلة الأولى x="musleh"
- والسلسلة الثانية y="muslhh"

```
.data
x db "muslhh"
y db "musleh"

.code

mov bx, @data
mov ds, bx

mov si, offset x
mov di, offset y

mov cx, 6
cld

REPE CMPSB
      JNZ MM
      MOV AX, "Y"

MM: hlt
```

جامعة القدس المفتوحة (فرع شمال غزة)



كلية التكنولوجيا والعلوم التطبيقية

هيكلية الحاسوب ولغة اسمبلي

١٣٨١

الوحدة الخامسة: **الماكرو**

الفصل الأول ١٢٢١

اعداد : أ. مصلح منير مصلح

```
01  
02 org 100h ; set location counter to 100h  
03  
04  
05  
06  
07  
08 mov al,02h  
09 mov bl,05h  
10 ; .....  
11  
12  
13 call mult  
14 hlt  
15  
16  
17  
18 ; .....  
19 mult proc  
20  
21     mul bl  
22     ret  
23  
24 mult endp  
25  
26  
27  
28 ;
```

```
01 org 100h  
02  
03 main macro x,y  
04  
05     mov al,x  
06     add al,y  
07  
08  
09  
10 endm  
11  
12  
13  
14 .data    |  
15 x db ?  
16 y db ?  
17  
18  
19 .code  
20  
21 mov bx,@data  
22 mov ds, bx  
23  
24  
25 main 2,2  
26  
27 hlt  
28  
29  
30  
31  
32  
33  
34
```

مقدمة عن ماكرو

- بعض الاحيان عند كتابة برنامج ما تكرر مجموعة من الاكواد البرمجية والتعليمات داخل برنامج واحد .
- وهذا يؤدي الى صدور اخطاء وزيادة الجهد والوقت
- لذلك توفر لغة اسمبلي شيء **يسمى الماكرو او البرنامج الفرعي** او الجزئي الذي يحتوي على مجموعة من التعليمات يمكن استدعائه من أي مكان من البرنامج
- يمكن استدعائه من أي مكان في البرنامج الرئيسي يمكن كذلك استدعائه بشكل ذاتي للماكرو او من داخل ماكروا آخر دون الحاجة لتكرار الجمل واستدعائه اكثر من مرة
- الماكرو هو عبارة عن مجموعة من جمل لغة اسمبلي يتم تعريفه في بداية البرنامج

ماهي الإضافة التي قدمها الماكرو او ماهي فوائد الماكرو

- المترجم الذي يعالج او يترجم الماكرو **يعرف بإسم الماكرو**
أسمبلر macro Assembler

1. تقليل عدد التعليمات المدخلة.
2. تجزئة البرنامج إلى مجموعة من الماكرو والإجراءات
3. تقليل الجهد اللازم لإدخال التعليمات لكونها قد قلت.
4. تقليل إمكانية الوقوع في الخطأ الناتج عن عملية إدخال التعليمات.

من أهم خصائص الماكرو .

- **سهولة التتبع والفهم والتعديل** : إذا كان الماكرو صحيح دون اخطاء مهما بلغت عدد مرات استدعاءها تبقى النتيجة صحيحة
- **انتشار الماكرو macro expansion** يقصد بها عند استدعاء الماكرو نستدعي اسم الماكرو فإنه يقوم باستبدال هذه الجملة بمجموعة من الجمل المكونة لمتن او جسم الماكرو .

أمثلة على استخدام الماكرو

- عمليات الإدخال والإخراج لتخزين قيم ابتدائية في المسجلات
تمهيداً لتنفيذ عمليات الاعتراض المسجلات Interrupts
- تحويل البيانات بين النظامين الاسكي والثنائي.
- في العمليات الحسابية عند إجرائها على كلمات عدة بدال بايت أو كلمة وفي معالجة السلاسل الرموز وانجاز عمليات القسمة والطرح

□ تعريف الماكرو يسبق عادة تعريف أي من قطاعات برنامج
لغة اسمبلي

مقارنة بين الماكرو والبرنامج الفرعي او الاجراء

- الفكرة العامة للماكرو والاجراء واحدة في كتابة مجموعة التعليمات المتكررة في البرنامج مرة واحدة ثم استدعاءها عند الحاجة :

الماكرو	البرنامج الفرعي او الاجراء
<ul style="list-style-type: none">■ يبدأ بتعريف الماكرو في توجيهه■ Macro■ تعريفه يتطلب معاملات يتم تعريفها■ ببداية الماكرو	<ul style="list-style-type: none">■ يبدأ بتعريف الاجراء في توجيهه proc■ لايجوز استخدام المعاملات مع توجيهه■ proc■ NEAR, FAR
<ul style="list-style-type: none">■ تتميز في خصائص انتشار الماكرو■ لا توجد جملة في شيفرة الهدف مقابل كل جملة استدعاء للماكروا في لغة المصدر كما هو الحال في الاجراء	<ul style="list-style-type: none">■ لا تتميز في خاصية انتشار الماكرو

مقارنة بين الماكرو والبرنامج الفرعي او الاجراء

البرنامج الفرعي والاجراء	الماكرو
تعتبر جمل استدعاء الاجراء من تعليمات يتم معالجتها من قبل المعالج	تعتبر جمل استدعاء الماكرو من التوجيهات الاسمبلر
تستدعي بواسطة كلمة call	يتم استدعاءها بذكر اسمها فقط او اسمها مع معاملات
تحتوي على ret التي توهي انتهاء الاستدعاء وارجاء المعالج الى البرنامج	لا تحتوي على ret
الاجراء يحتاج الى وقت اكبر اثناء التنفيذ وكذلك للعودة ret	يعتبر الماكرو الذي يكون جسمه عدد اقل من التعليمات اكثر كفاءة في التنفيذ
لا يمكن	يمكن كتابة الماكرو في مكتبة ويمكن استدعاءه وقت الحاجة
لا يمكن أي تعديل على الحمل يتطلب ترجمته من جديد	اكثر مرونة في تعديل طريقة عمله

مكونات الماكرو

- جمل الماكرو يجب ان يكتب في بداية البرنامج قبل تعريف القطاعات

- يتكون الماكرو من

1. جملة بداية الماكرو

2. جسم او متن الماكرو : يمكن ان يحتوي على تعليمات او توجيهها

3. جملة نهاية الماكرو ENDM

جملة بداية الماكرو

- تستخدم لتحديد بداية الماكرو
- الصيغة العامة لبداية الماكرو

NAME MACRO Parameter1,parameter2

- Name اسم الماكرو الذي من خلاله هذا الاسم نستدعيه بكتابة اسمه يجب ان يخضع لشروط الاسماء .
- يجب ان يكون هذا الاسم فريد ممنوع نسمى متغير او او بنفس الاسم
- Parameter هي المعاملات عبارة عن متغيرات يحددها المستخدم عند الحاجة وهي اختيارية

جملة بداية الماكرو

- يفضل ان اسم الماكرو ان يدل على العمل الذي يقوم به أي على وظيفته مثل خاص في ضرب عددين نسميه
multnum
- جدول اسماء الماكرو macro name table الاسمبلر عند ملاحظته اسم للماكرو يخزنه في هذا الجدول
- عندما نريد استدعاء الماكرو نكتب فقط اسمه اذا كان يوجد له معاملات نحدد قيم لهذه المعاملات

مثال على بداية الماكرو

Multnum **macro** var1,var2

- اسم الماكرو mulnum يمكن استدعائه بكتابه هذا الاسم فقط
- او استدعائه بكتابة هذا الاسم مع تحديد قيم للمعاملات
mulnum 5,2
- ملاحظة الماكرو يمكن ان يحتوي على معاملات او لا

مكونات الماكرو

جملة نهاية الماكرو

- تستخدم لتحديد جمل نهاية الماكرو
- يستخدم توجيهه **endm** للدلالة عليها

متن او جسم الماكرو **macro body**

- قد يكون متن الماكرو توجيهات او تعليمات
- أي ان الماكرو يحتوي على تعليمات أخرى غير تعليمة البداية والنهاية

التوجيهات الخاصة بالماكرو:

■ ويمكن تقسيم التوجيهات الخاصة بالماكرو إلى خمسة مجموعات رئيسية على النحو التالي:

أ- التوجيهات عامة الغرض OPS – Pseudo Purpose
General

ب- توجيهات التكرار loop

ج- التوجيهات الشرطية Operations – Pseudo

د- توجيهه خروج من الماكرو EXITM

هـ- توجيهات الطباعة

التوجيهات الخاصة بالماكرو:

١- التوجيهات عامة الغرض OPS – GeneralPurpose Pseudo

■ توجيهية بداية الماكرو MACRO

■ توجيهه نهايته ENDM

■ توجيهه LOCAL

✓ تستخدم هذه التوجيه داخل اسم الماكرو لتعريف الرموز داخله

✓ تعني عدم اعادة استخدام نفس الرموز او الوسم في أكثر من موقع

✓ بمعنى المتغير الموجود بجانب هذه التعليمة عند كل مرة يتم فيه

استدعاء الماكرو اعطيه اسم مغاير مثل var1 عند اول استدعاء وثاني

استدعاء var2

توجيهه LOCAL

- الصيغة العامة لهذه التوجيهة

LOCAL symbol, symbol,...

- **symbol** : يمثل اسم الرمز الموضعي المؤقت والذي سيتم استبداله باسم رمز فريد في كل مرة يتم فيها نشر الماكرو مكان إحدى جمل الاستدعاء
- يجب ان تحتوي هذه التوجيهة على الاقل لو رمز واحد
- لو اردت استخدام اكثر من رمز الفصل بينهم من خلال فاصلة
- تأتي هذه الجملة بعد جملة تعريف الماكرو مباشراً لان الرموز الملازمة لهذه التوجيه يمكن استخدامها من قبل متن الماكرو
- لايجوز استخدام أي جملة قبلها حتى لو كانت ملاحظة سيتم اصدار خطأ

ملاحظات

- ملاحظة عند كتابة المعاملات بجانب local عدم كتابة مثلاً متغير var1 أي عدم استخدام الأرقام لأنه هو تلقائي عند كل استدعاء سيتم إعطاء متغير رقم مثلاً عند الاستدعاء الأول للماكرو سيعطي var11 وبهذا يحدث خطأ
- سيتم تخزين الناتج في ax ولو ما اتسع سيتم تكملة تخزين الناتج في سجل dx
- :: الفارزتين منقوطين تعني في الماكرو ملاحظات على خلاف في لغة الاسمبلي المعتاد عليها تكن فارزة مفردة

توجيهات التكرار

يوجد ثلاث توجيهات للتكرار

■ REPT

■ IRP

■ IRPC

□ توجيهة REPT

- تستخدم هذه التوجيهة لإعادة تكرار مجموعة من الجمل عدة مرات
تحدد بقية المعامل الحقيقي عند استدعاء الماكرو
- يوجد نوعان من المعاملات في الماكرو
 - **معامل حقيقي** : معامل مع جملة استدعاء الماكرو
 - **معامل شكلي** : يستخدم عند تعريف الماكرو وفي متن الماكرو

توجيهية REPT

- نهاية توجيهية REPT مماثلة لتوجيهية نهاية الماكروا ENDM
- الجمل التي يتم تكرارها هي التي تقع بين توجيهية REPT ونهاية ENDM

```
REPT expression  
Statements  
ENDM
```

- Expression هي عبارة عن معامل حقيقي يجب الا يزيد طوله عن ١٦ بت او ثنائية يجب ان يكون ثابت وهو يعبر عن عدد مرات التكرار.
- ويمكن ان تكن جمل التكرار بشكل منفصل أي خارج الماكرو او داخل الماكرو

• مثال

```
mov ax,2  
REPT 2  
ADD AX,2  
endm
```

X = 0 ;;	صفر الرمز X
REPT 26 ;;	حيث أن الرقم يمثل عدد مرات
DB 'a' + X ;;	خزن شيفرة آسكي لحرف من الحروف
X = X + 1 ;;	انتقل إلى الحرف التالي
ENDM ;;	توجيه نهاية التكرار

توجيهية IRP

- تستخدم عند الحاجة لتكرار جملة أو أكثر من جمل لغة أسمى حيث أن عدد التكرارات والمعاملات لكل واحدة من التكرارات تحددها مجموعة من القيم تسمى . Arguments
- نفس عمل rept ولكن يتم التكرار حسب قيم معينة يتم إعطاؤها في الجملة.

والصيغة العامة لهذه التوجيهية هي :

```
IRP Parameter, < argument [, argument] . . . >
```

```
Statements
```

```
ENDM
```

- ومبدأ عمل هذه التوجيهية هو أن الجمل **Statements** الواقعة بين IRP و ENDM تكرر مرة لكل قيمة من القيم . Arguments.

توجيهة IRP

• تحتوي هذه على معاملان

- معامل الشكلي Parameter ويعبر عن اسم لموقع سيتم استبداله بالقيمة الحالية لل argument
- موضوعة بين <> : مجموعة القيم التي سيتم تعويضها واحدة تلو الاخرى لكل مرة يتم فيها تكرار الجمل
- القيم بين الاقواس يمكن ان تكون قيم ثابت او رمز <> عند استخدامك اكثر من رمز او ثابت يفصل بينهم فاصلة
- Parameter يمكن استخدامه أكثر من مرة ضمن الجمل الواقعة بين بداية التعليق ونهايتها

توجيهة IRP

- عندما يصل الاسمبلر لهذه التعليمية فإنه يعمل نسخة من الجمل المكونة لمتن هذه التعليمية مع تعويض قيم المحددة بين الأقواس $\langle \rangle$
- في حال كتابة التعليمية كالتالي بدون قيم $\langle \rangle$, IRP M نلاحظ قيم المعامل الشكلي بمعنى المتغير قيم فارغة أي يحجز في الذاكرة قيمة فارغة للمتغير M أي يعوض عنها بلا شيء
- في حال كتابة هذه التعليمية IRP M كأنك تكتب هذه التعليمية

توجيهية IRP

• مثال

في هذا المثال يتم تنفيذ التوجيهية DB عشرة مرات وتخزين القيم من 0 إلى 9 في المواقع التي تم حجزها باستخدام التوجيهية IRP وإعطاء الموقع الأول الاسم Byte_Table.

```
Byte_Table LABEL BYTE
            IRP A, < 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 >
            DB A
            ENDM
```

تعليمة IRPC

- الصيغة العامة

```
IRPC Parameter, String  
Statements  
ENDM
```

- يتم تكرار مجموعة من الجمل الموجودة بين التعليمة بناء على عدد حروف النص **STRING**
- يتم استبدال الحرف الاول من حرف النص مكان المعامل الشكلي حتى يحل الحرف الاخير محل المعامل الشكلي البارميتر
- يمكن لل **string** ان تكون عبارة عن ارقام او رموز او حروف
- وفي حال احتواء **string** على فراغات او فواصل لفصل الاعداد او الحروف عن بعضها البعض يجب استخدام الاقواس **<>**

تعليمة IRPC

- عند مصادفة الاسمبلر تعليمة IRPC فانه يكرر متن هذه التعليمة على حسب عدد الرموز او الحروف او الارقام بحيث كل عملية تكرار يستبدل الرمز الحالي ب البارميتر

IRPC character, 0123456789

DB character

ENDM

في هذا المثال يتم تنفيذ التوجيه DB عشر مرات ، مرة لكل رمز من الرموز العشرة في النص "0123456789" . لذلك فإن تنفيذ هذه التوجيه يؤدي إلى حجز عشر مواقع من النوع بايت تخزن فيها القيم من 0 إلى 9 . في هذا المثال تعامل رموز النص "0123456789" على أنها أرقام من 0 إلى 9 .

تعليمة IRPC

- إذا اردنا معاملتها على أنها رموزا وليس عددا فانه يتم استبدال تعليمة DB character يتم استبدالها في تعليمة

DB '&character'

- & من عمليات الماكرو والتي سيتم معالجتها فيما بعد

التوجيهات الشرطية

- مثل توجيهات التكرار يمكن ان تكون ضمن الماكرو او منفصلة.
- يتم تنفيذ الجمل بناء على فحص تحقق الشرط المتحقق اذا لم يتحقق يتم تتبع البرنامج

□ توجيهية: EXITM

- تستخدم لجعل الأسمبلر ينهي عملية انتشار الماكرو مكان جملة استدعائه بحيث يستثنى من عملية الانتشار هذه جميع الجمل الواقعة بين التوجيهية EXITM وتوجيهية نهاية الماكرو ENDM عند معالجة الأسمبلر لجملة الاستدعاء

التوجيهات الشرطية

- الصيغة العامة: EXITM

- تستخدم هذه التوجيهية مع التوجيهات الشرطية وذلك من خلال القفز عن جمل الاسمبلي في نهاية الماكرو عند تطبيق شرط معين

```
Allocate MACRO Length
```

```
X = 0
```

```
IF LENGTH GT 0FFH
```

```
    EXITM
```

```
ENDIF
```

```
    REPT Length
```

```
        X = X + 1
```

```
        DB X
```

```
    ENDM
```

```
ENDM
```

التوجيهات الشرطية

• المثال السابق

في هذا المثال تستخدم التوجيهية EXITM مع التوجيهية الشرطية IF حيث يقوم الأسمبرل بإنشاء جدول فقط عند إستدعاء الماكرو بمعامل حقيقي قيمته لا تزيد عن 255 . وفي هذه الحالة يتم تخزين القيم من 0 إلى 255 في هذه المواقع . تستخدم التوجيهية IF لفحص قيمة Length حيث يجب أن لا تزيد عن القيمة 0FFH بالنظام السادس عشري (أي

توجيهات الطباعة

- تستخدم هذه التوجيهات للتحكم بكمية الجمل التي ينتجها الأسمبلير من الماكرو في البرنامج الرئيس نتيجة عملية التجميع والتي تخزن في ملف من النوع LST
 - عند استخدام هذه التوجيهات يمكن عرض او عدم عرض جمل الماكرو عند استدعاءها في البرنامج الرئيسي أي الشرح بجانب المحاكى
 - وتشمل توجيهات الطباعة التوجيهات التالية:
- .XALL
 - .LALL (List ALL)
 - .SALL (Suppress ALL)
 - النقطة قبل التعليمة تعتبر جزء من التعليمة بدونها يصدر خطأ
 - استخدام احد هذه التوجيهات يبقى مفعولها قائم على كل ماكرو مالم تستخدم توجيهه اخرى
 - تعتبر .XALL المستخدمة ضمناً إذا لم يتم استخدام واحدة من التوجيهات السابقة

توجيهة XALL.

- تستخدم لعرض التعليمات التنفيذية فقط التي تولد شيفرة هدفية
- حيث جمل الملاحظات والتوجيهات لا يتم عرضها في ملف LST والناج عند تنفيذ MASM
- الصيغة العامة لهذه التعليمات XALL.

تعليلة LALL . (List ALL)

- تستخدم هذه التعليلة لعرض كافة جمل الماكرو بما فيها الملاحظات والتوجيهات
- يجب ان تسبق هذه التعليلة او التوجيهة جملة استدعاء الماكرو او انتشار الماكرو
- الصيغة العامة LALL.

تعليلة SALL.

■ استخدام هذه التوجيهية يؤدي إلى عدم نشر تكرار الجملة المكونة لمتن الماكرو عند استدعائها من داخل البرنامج الرئيس.

• ومعنى ذلك أن الملف من نوع LST والناج من عملية تنفيذ الأسمبلير MASM لا يحتوي على الجملة المكونة لمتن الماكرو.

• الصيغة العامة: SALL.

❖ أي توجيهات الطباعة ليس لها أي تأثير على تنفيذ البرنامج أي وظيفتها فقط عرض الجملة المكونة لمتن الماكرو أو لا

جملة استدعاء الماكرو

□ يجب ان يكون معرف الماكرو في البداية حتي يتم استدعاءه

□ عند تعريف الماكرو يجب مراعاة :

1. يكون اسم الماكرو معبر عن وظيفته.

2. تحديد المتن

3. توجيهه انتهاء الماكرو

□ عندما نريد استدعاء الماكرو نذكر اسمه فقط واذا كان هناك معاملات للماكرو أي نضع الثوابت بجانب اسم الماكرو المراد استدعاءه

□ مثال 2, 3 NUM على فرض استدعاء ماكرو NUM

- المعاملات الحقيقية هي متغيرات يتم تعريف قيمتها في البرنامج الرئيس حيث يتم تمرير هذه القيم الي القيم المناظرة لها في الماكرو
- المعاملات الشكلية هي متغيرات غير معرفة القيمة يتم احتساب قيمها داخل الماكرو
- عند استخدام الماكرو وجمل استدعائه بمعاملات يجب ان تكون كلا من المعاملات الشكلية والحقيقة متطابقة من ناحية النوع والعدد

أدوات الماكرو:

1. أداة التعويض &
2. أداة النص الحرفي < >
3. أداة الرمز الحرفي !
4. أداة التعبير %
5. أداة الملاحظة داخل الماكرو ::

أداة التعويض &

- تستخدم هذه الأداة لإخبار الأسملي بأن يستبدل معامل شكليا لمعامل الحقيقي المناظر له عند استدعاء الماكرو.
- تستخدم هذه الاداة لو سبق المعامل او المتغير نص لنميزه عن النص او اذا كان بين خاصرتين

```
XXX MACRO Parameter1, Parameter2
```

```
PUBLIC Err&Parameter1
```

```
Err&Parameter1 DB 'Error &Parameter1 : &Parameter2'
```

```
ENDM
```

```
XXX    MACRO  Parameter1, Parameter2
PUBLIC  Err&Parameter1
Err&Parameter1      DB  'Error &Parameter1 : &Parameter2'
ENDM
```

● ولو فرضنا انه تم استدعاء هذا الماكرو عن طريق الاستدعاء :

Xxx 5,<Unreadable Disk>

■ فان نشر الماكرو سيؤدي إلى الجملة التالية فقط في مكان جملة الاستدعاء

Err5 DB 'Error 5: Unreadable Disk'

■ يلاحظ في هذا المثال أن الأسمبلير سوف يستبدل ¶meter1

■ بالقيمة 5 ويسبدل ¶meter2 بالقيمة "Unreadable Disk"

أداة النص الحرفي < > :

- تستخدم أداة النص الحرفي وهي على شكل قوسين < > لإخبار السمبر بأن يعامل مجموعة العناصر الواردة بينها على أنها نص متكامل وليس مجموعة قيم منفصلة.
- الصيغة العامة: <text>
- يمكن استخدام هذه الأداة لإخبار المعالج بأن يعامل بعض الرموز الخاصة مثل & والفارزة المنقوطة; على أنها حروف كما هي ال تحمل أي معنى
- يمكن كذلك استخدام هذه الأداة مع توجيهات IRP التكرار

مثال على اداة النص الحرفي <>

- اذا اعطيت هذا الماكرو

```
TEST MACRO X  
IRP Y,<X>  
DB Y  
ENDM  
ENDM
```

فإذا كانت جملة الاستدعاء على النحو التالي:

Test < 0, 1, 2, 3, 4 >

فإن الأسمبلير يستبدل هذه الجملة بالجملة التالية نتيجة لنشر الماكرو:

```
IRP Y, < 0, 1, 2, 3, 4 >  
DB Y  
ENDM  
DB0  
DB1  
DB2  
DB3  
DB4
```

عند المعالجة تصبح الجملة

أداة الرمز الحرفي !:

- تستخدم هذه الأداة لإخبار الأسمبليير بأن يعامل الحرف التالي مباشرة إلى من رموز من رموز الآسكي وليس رمزاً جوار هذه الأداة على أنه رمزاً معيناً لغة أسمبلي يحمل معنا .
- الصيغة العامة: **Character!**
- حيث أن **Character** يمثل رمزاً من رموز شيفرة آسكي.
- **&!** تعامل على انها حروف من حروف الآسكي ولا تجبر المعالج القيام في شيء
- يعامل مع معناه الحرفي
- أي وضع أي رمز من الرموز داخل <> مشابه تماماً لوضعه بعد !
< > مثل !!
- استدعاء ماكرو اسمه <205!>Expression, 100MMM
- هذا الاستدعاء صحيح لانه يعبر عن رمز > بعد علامة التعجب لو لم يوجد علامة تعجب يعتبر خاطيء

أداة التعبير %

- تستخدم هذه الأداة لإخبار الأسمبليز بأن يعامل ما بعد هذه الأداة على أنه تعبيراً Expression وليس نصاً Text
- الصيغة العامة: %text
- حيث يقوم المعالج عند مصادفة هذه الأداة بحساب قيمة text ومن ثم يستبدل المعامل text بالقيمة المحسوبة.
- ويمكن للمعامل text أن يكون إما تعبيراً عددياً أو نصياً معروفاً باستخدام التوجيه EQU
- إذا تم استخدام أكثر من معامل بجانب % يجب الفصل بينهم في فاصلة ولا بمسافة يصدر خطأ إذا ترك مسافة

أداة الملاحظة داخل الماكرو ;;

❖ تستخدم هذه الأداة لإخبار الأسمبلير بأن ال يقوم بنسخ
جمل الملاحظات والتعليقات الموجودة في الماكرو عند
نشر الماكرو مقابل كل جملة من جمل الاستدعاء.

● الصيغة العامة: text;;

● حيث أن text يمثل تعليقاََ معيناً أو ملاحظات تظهر
فقط داخل الماكرو.

■ عند استدعاء الماكرو لا يتم نشر الملاحظات داخل
الماكرو

مكتبات الماكرو macro

- الهدف من مكتبة الماكرو هو حفظ جميع البرامج الفرعية المكونة بأسلوب الماكرو بحيث يتم تعريفها مرة واحدة واستخدامها في أي برنامج يحتاج إليها المبرمج.
- بدلاً من تعريف برامج الماكرو في بداية كل برنامج رئيس يمكن تخزين جميع برامج الماكرو في مكتبة الماكرو حيث يمكن الرجوع إليها باستخدام توجيهية: INCLUDE
- في حال كانت مكتبة الماكرو مخزنة على وحدة القرص الصلب C: باسم MACRO.LIB فإنه يمكن استخدام جميع الماكرو في هذه المكتبة عن طريق توجيهية:
- INCLUDE C: MACRO.LIB

هل يمكن ربط لغة اسمبلي بلغات أخرى عالية المستوى

- نظراً لسهولة برمجة بعض العمليات في لغة أسمبلي وصعوبتها في اللغات عالية المستوى، وسهولة برمجة بعض العمليات الأخرى في اللغات عالية المستوى وصعوبتها في لغة أسمبلي.
- رأى مصممو لغات البرمجة الاستفادة من ميزات كل لغة من اللغات.
- لذلك نجد أنه بالإمكان ربط برامج لغة أسمبلي مع برامج لغات عالية المستوى مثل لغة باسكال ولغة بييسك ولغة C.