معالجات قسم د. تاج

المعالج 8086: يحتوي مقاطعات وسجلات

• المقاطعات:

1 CS : Code segment

مقاطعة العمليات الأساسية: يتم فيها إجراء العمليات على المتغيرات لذلك هي أساسي (موجود في كل برنامج)

2 SS : Stack segment

مقاطعة المكدس: تستخدم عندما يذكر "مكدس" و لعمليات الـ pop (سحب كلمة من المكدس) و push إضافة كلمة إلى المكدس.

3 ES : Extra segment : البيانات الإضافي

4 DS : Data segment : مقاطعة البيانات

الـ Es و الـ DS: يتم فيهم تعريف المتغيرات و السلاسل الحرفية و الجداول أي كل ما يتم إجراء عمليات عليه و لكن هناك تعليمات مخصصة لـ Es و هناك تعليمات مخصصة لـ DS

• سجل الغرض الخاص: يخزن عنوان التعليمة التالية بالجلب (لا يستخدم بالمسائل)

• سجلات الأغراض العامة:

	16 bit	
	8bit	8bit
AX: المراكم	AH الباية الأعلى	AL الباية الأدنى
BX: سجل القاعدة	BH	BL
CX: سجل العداد	CH	CL
DX: سجل البيانات		
مؤشرات	SP: stack pointer	
	DI: دليل الوجهة	
	SI: دليل المصدر	
	Bp مؤشر القاعدة	

* AX: السجل الراكم يستخدم للعمليات الحسابية و المنطقية أي نقط تخزين (لا مؤشر ولا عداد)

* BX: هو سجل عام أي ← التخزين
← كمؤشر
← عداد

"عند استخدامه كعداد يستخدم كامل أي ليس BL أو BH بل BX"

* CX: العداد الأساسي وهو عداد تنازلي دائماً وقد نستخدم BX أو DX كعداد ثاني، و يستخدم بتعليمات الإزاحة والدوران بأكثر من بت

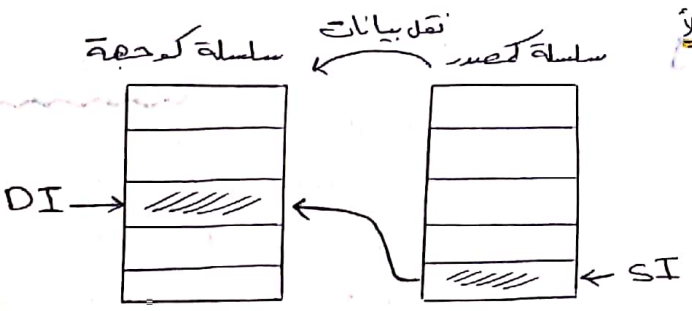
* DX: عنونة بوابات الدخل والخرج و كمساعد في عمليات الضرب والقسمة المؤشرة وغير المؤشرة.

ملاحظة: جميع السجلات و المقاطعات طولها 16 bit

الربط بين المقاطعات والأغراض العامة والخاصة:

CS	SS	DS	ES
IP	SP BP	SI: سلاسل كمصدر BX	DI: سلاسل كوجهة BX

مثلاً



• المؤشرات مهمتها تدل على مكان المحتوى الذي نحتاجه في السلسلة

• نستطيع وضع أكثر من مؤشر على السلسلة

• المؤشر إما يتحرك تصاعدياً أو تنازلياً أي لا نستطيع استخدام نفس المؤشر ليتحرك تصاعدياً و تنازلياً ← لذلك عند الحاجة نستخدم مؤشرين حيث BX مؤشر عام

المتغيرات في البرنامج :

- DB : المتغير بايت واحد 1Byte=8bit
- DW : المتغير من بايتين (كلمة) ← 16bit

ولتعريف المتغيرات في البرنامج :

١ تعريف الثوابت :

اسم الثابت EQU قيمة الثابت

٢ تعريف السلسلة أو الجدول :

القيمة DUP عدد العناصر نوع العناصر إما DB أو DW

القيمة: إنا لم يذكر نضع ؟ / قيمة مذكورة

نوع العناصر أو محتوى السلسلة !

أشهر التعليمات المستخدمة بالبرامج :

١ المؤشرات تصاعدية ← CLD

المؤشرات تنازلية ← STD

توضع غالباً في بداية البرنامج وتدل أن جميع مؤشرات البرنامج تصاعدية أو تنازلية ، وليس لهم معاملات

٢ تعليمة النسخ Mov مصدر ، وجهة (المعاملات)

أي يتم نسخ محتوى المصدر إلى الوجهة بحيث محتوى المصدر لا يتغير .

- محظورات الـ Mov : أسئلة :

* Mov AX , BL (X)

لأن طول المصدر لا يساوي طول الوجهة

* Mov AX , CS (X)

لا يمكن أن تكون معاملات الـ Mov مقاطعة وسجل (يجب إما سجلين أو سجل وموقع في الذاكرة)

* Mov AX , CS:[IP] (✓)

مجاورة

* Mov R1 , R2 (X)

لا يمكن النسخ بين موقعين في الذاكرة (أي رسو غير رسو) السجلات والمقاطعات هي مواقع بالذاكرة) فنضطر وسيط مثل AX

* Mov [BP] , [BX] (X)

لأن الوجهة والمصدر مؤشرات تشير لمحتوى في الذاكرة

* Mov CS , DS (X)

لا يمكن النسخ بين مقاطعتين .

* Mov 100 , CX (X)

لأن الوجهة قيمة فورية

* Mov CX , 100 (✓)

* Mov IP , AX (X)

ممنوع استخدام سجل IP في تعليمات Assembly

* Mov [AX] , DX (X)

لأن المترجم لا يستخدم للسنوتة أي لا يشير للذاكرة

* Mov BX (X)

لا يمكن معامل وحيد

٣ ADD تعليمة الجمع مصدر ، وجهة

SUB تعليمة الطرح مصدر ، وجهة

محظورات الـ SUB والـ ADD نفس محظورات الـ Mov .

٤ LEA تحميل المؤشر على سلسلة

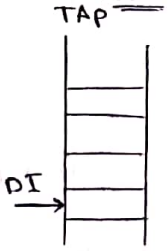
16bit حصراً

LEA DI , TAP

DI: اسم المؤشر

TAP: اسم السلسلة (الجدول)

مثلاً:



تعليمات الزيادة والانقاص :

لها معامل وحيد

INC الوجهة (زيادة الوجهة بمقدار 1)

DEC الوجهة (إنقاص الوجهة بمقدار 1)

الوجهة غالباً تكون اسم المؤشر أو اسم السجل ، إذا كانت الوجهة مؤشر عندئذ INC ← تصاعدي ، DEC ← تنازلي

الفرق بين (INC, DEC) و (CLD و STD) :

- الـ CLD تجعل الزيادة تلقائية أي تضع بأول البرنامج فتجعل كل المؤشرات تصاعدية وبالمثل STD
- إذا احتجنا مؤشرين أحدهم تصاعدي والآخر تنازلي نستخدم خلال البرنامج INC التصاعدي و DEC تنازلي

تعليمة المقارنة

CMP مصدر ، وجهة

تقوم بعملية المقارنة عن طريق طرح المصدر والوجهة

← إذا الناتج صفر فهما متساويين

← الناتج ≠ الصفر فهما غير متساويين .

التبديل بين المصدر والوجهة

XCHG مصدر ، وجهة

XCHG وجهة ، مصدر

- ولها نفس محظورات Mov

تعليمات السلاسل المحرفية :

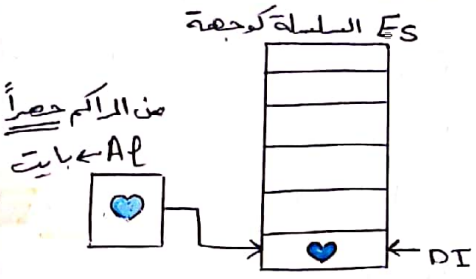
تستخدم فقط عند استخدام السلاسل والجداول وهي لاتحتاج معاملات .

1 تعليمات التخزين (الإدخال - الكتابة) بالسلسلة بالذاكرة

- STOSB (STore string Byte)

أي نخزن في السلسلة بايتات

- STOSW (STore string Word)

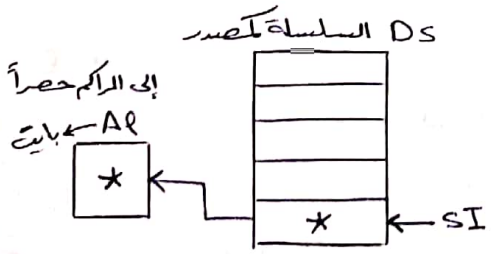


- نأخذ القيمة من الراكم حصراً
- السلسلة كوجهة القاطعة ES المؤشر DI

* إذا أجبرنا على مقاطعة ما مثلاً DS فلا يمكن استخدام تعليمة الـ STOSB بل نستخدم تعليمة بديلة هي تعليمة النسخ Mov

2 تعليمات (التحميل - القراءة - الإظهار) من السلسلة بالذاكرة

LODSB / LODSW



* إذا أجبرنا على مقاطعة ما مثلاً Es فلا يمكن استخدام هذه التعليمة بل نستخدم تعليمة بديلة هي تعليمة النسخ Mov

أهم التعليمات المنطقية :

- تستخدم التعليمات المنطقية عندما نطلب تغيير بت من شعاع .
- كل التعليمات المنطقية لها معاملين عدا الـ NOT لها معامل وحيد

AND	*	
OR	+	
NOT	-	(عكس)
XOR	مقارنة	(تشابه 0 / اختلاف 1)
Test	*	

مثال :

AL = 1111 1111 (7 6 5 4 3 2 1 0)

المطلوب تغيير البت الخامس (ذو الوزن 4) ؟

نستخدم تعليمة الـ OR حيث نقلب (0 ← 1) حيث نقابل الخانة المطلوبة بواحد وبقية الخانات أصفار

```
1111 1111
0001 0000   أي OR AL, 00010000b  ثنائي
1111 1111   أو OR AL, 10h  سادس عشر
```

مثال

AL = 1111 1111

المطلوب تغيير البت الخامس (ذو الوزن 4) ؟

نستخدم تعليمة AND التي تقلب (1 ← 0) حيث نقابل الخانة المطلوبة بصفر وباقي الخانات بواحد

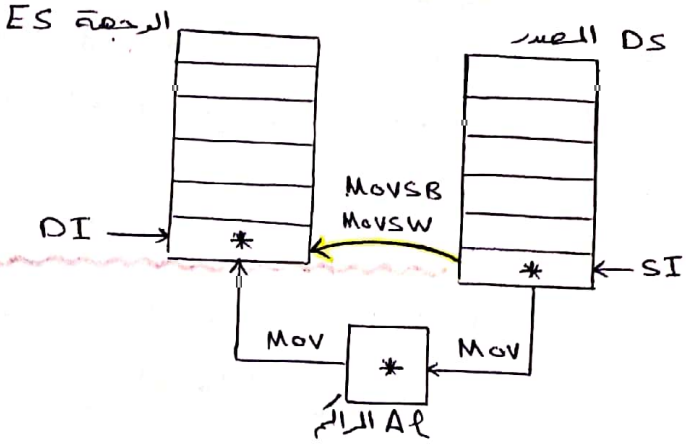
```
1111 1111
1110 1111   أي AND AL, 11101111b
1110 1111
```

الفرق بين الـ AND والـ Test

- AND تخزن الناتج في الوجهة
- Test لاتخزنه في الوجهة فقط تؤثر على سجل الأعلام

٣ النسخ بين السلاسل

MOVSB / MOVSW

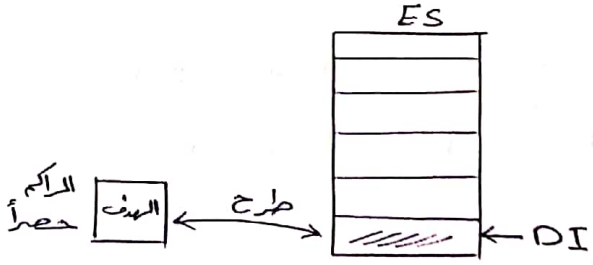


- عند النسخ باستخدام الـ MOVSB / MOVSW لا نحتاج سواكم بل نسخ مباشرة
- إذا ذكر المصدر في ES والوجهة في DS فالتعليمات السابقة لا تعمل بل نحتاج تعليمة بديلة هي Mov ونحتاج لوسيط هو الراكم
  Mov AL , ES:[DI] (تجاوز)
  Mov DS:[SI] , AL

٤ تعليمات المسح (المرور على السلسلة):

SCASB / SCASW

يتم المرور على السلسلة بهدف إيجاد Byte أو Word ما إذا كانت السلسلة وجهة ونضع الهدف المطلوب إيجاده في الراكم للمقارنة مع عناصر السلسلة.



ناتج الطرح صفر ← العنصر في السلسلة يماثل الهدف ← المطلوب
ناتج الطرح ≠ 0 ← العنصر في السلسلة لا يماثل الهدف
* لو ذكر في السؤال السلسلة مخزنة في DS مثلاً عندئذٍ التعليمة البديلة هي CMP المقارنة
CMP AL , DS:[SI]

ملاحظة: لم نستخدم تعليمة الطرح لأن الـ SUB تطرح وتخزن الناتج في الوجهة ← (الهدف في الراكم سيتغير) لذلك نستخدم التعليمة CMP

٥ المقارنة بين سلسلتين: CMPSB / CMPSW

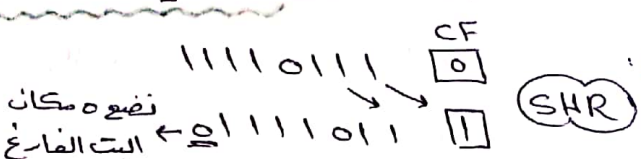
يتم المقارنة بين سلسلة مصدر ← DS و SI
سلسلة وجهة ← ES و DI
لو غير فالبديل هو CMP

تعليمات الإزاحة والدوران:

لها معاملين مصدر ووجهة حيث المصدر هو عدد مرات الإزاحة أو الدوران.

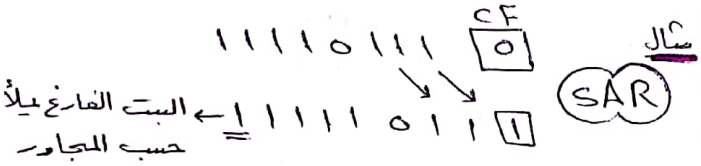
- SHR إزاحة منطقية نحو اليمين
- SHL إزاحة منطقية نحو اليسار

مثال: SHR



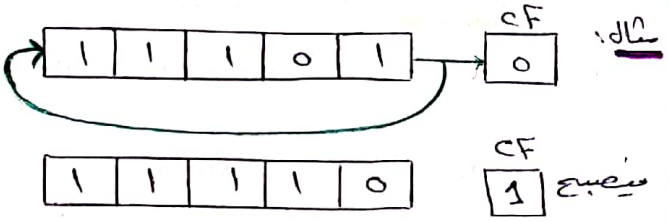
- SAR إزاحة حسابية نحو اليمين
- SAL إزاحة حسابية نحو اليسار

مثال: SAR



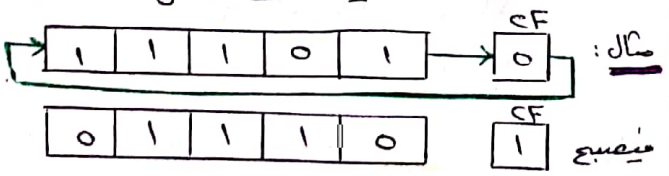
- ROR دوران نحو اليمين بدون حامل
- ROL دوران نحو اليسار بدون حامل

مثال:



- RCR دوران نحو اليمين مع حامل
- RCL دوران نحو اليسار مع حامل

مثال:



ملاحظة: عند الإزاحة أو الدوران بمقدار بت مثلاً

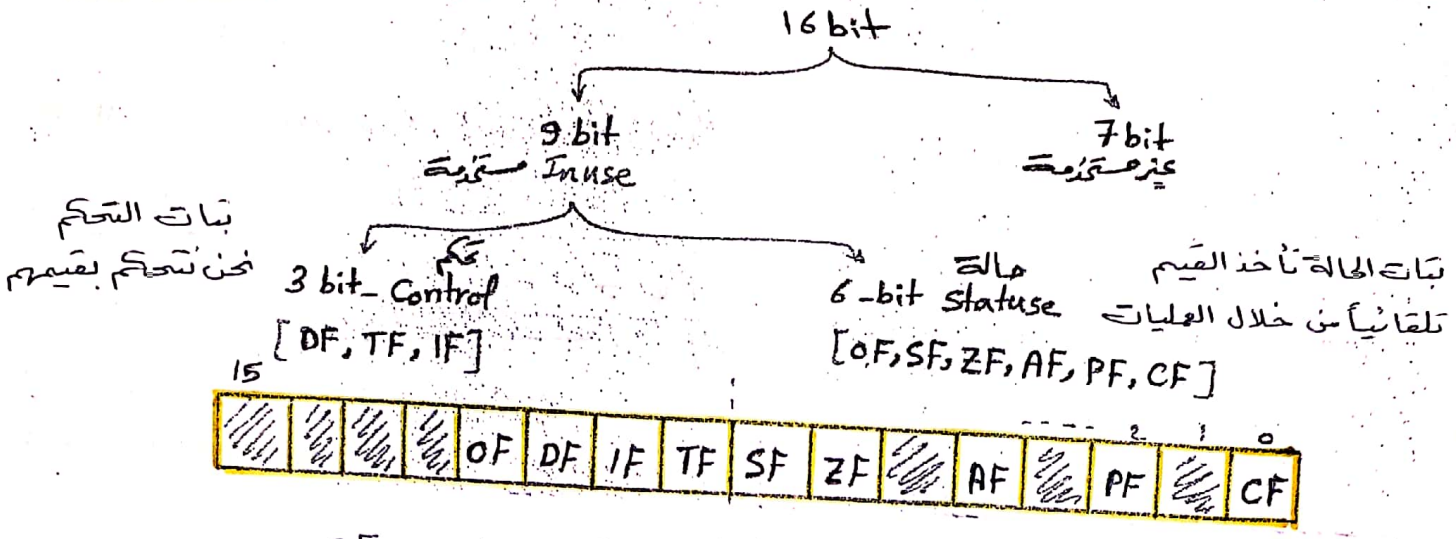
SHR AL , 1

- إذا أردنا أن نزيح أكثر من بت نستخدم السجل CX

Mov CL , 3
SHR AL , CL

مسجل الأعلام (FR: Flags Register) «دورات هام»

هو سجل طوله 16bit يوجد فيه 7 أعلام غير مستخدمة و 6 أعلام حالة و 3 أعلام تحكم.



* CF : علم الحامل: يتفعل عندما يظهر منقول نهائي CF=1 وإلا CF=0
* AF : علم الحامل المساعد : يتفعل عندما يظهر منقول من منتصف البايت أو الكلمة AF=1 وإلا AF=0
* ZF : علم الصفر : إذا كان ناتج العملية صفر يتفعل ZF=1 وإلا ZF=0
* SF : علم الإشارة : SF=1 أي البت الأعلى من ناتج العملية يساوي 1 ⇐ الناتج سالب
SF=0 أي البت الأعلى من ناتج العملية يساوي 0 ⇐ الناتج موجب
* PF : علم الشفية : إذا عدد الواحدات في الناتج زوجي ⇐ PF=1
إذا عدد الواحدات في الناتج فردي ⇐ PF=0
* O : علم الطفحان : OF=1 يوجد طفحان
OF=0 لا يوجد طفحان

سؤال دورة : (نقطبات الحالة أما بتات التحكم لا تتأثر بها)

1) ما الذي سيحدث لحالات الأعلام التالية (CF , PF , AF , ZF , SF , OF) اذا وردت التعليمة

ADD AL,1 وكان محتوى السجل AL = 7FH

2) ما هو محتويات السجل AL وحالة بتات الستة المذكورة سابقاً في الطلب السابع بعد تنفيذ التعليمات:

```
MOV AL , 6DH
MOV BH , 40H
AND AL , BH
```

① AL = 0111 1111

+ 1

1000 0000

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

② AL = 6DH = 0110 1101 b

BH = 40H = 0100 0000 b

* (AND)

AL = 0100 0000 b

OF=0 ZF=0
CF=0 AF=0
DF=0 SF=0

تعليمات القفز: لها معامل وحيد (الموقع المطلوب)

II القفز غير المشروط

JMP اسم الموقع

أي اقفز إلى الموقع ونفذ التعليمات التي تليه.

I القفز المشروط: أي القفز بناء على شرط ما

* JZ / JE

JZ: اقفز عندما الناتج صفر أي ZF=1

JE: اقفز عندما القيمتين متساويتين Equal

* JNZ / JNE

عكس السابقات حيث N: Not

* JS / JNS

JS: اقفز عندما الناتج سالب SF=1

JNS: اقفز عندما الناتج موجب SF=0

ولمعرفة إشارة عدد ما نجمعه مع الصفر ينتأثر العلم SF بعد العملية

* JC / JNC

JC: اقفز عندما CF=1

JNC: اقفز عندما CF=0

* JCXZ

اقفز عندما العداد CX=0

* JA / JNBE

JA: اقفز عندما الوجهة أكبر من المصدر

JNBE: اقفز عندما الوجهة ليس أصغر ولا يساوي المصدر

وهاتان التعليمات متكافئتان ← JA أسهل

تعليمات الإدخال والاستخراج من الذاكرة من خلال بوابة:

IN الإدخال رقم البوابة ، المراكم

OUT الإخراج المراكم ، رقم البوابة

وعند الدوشي ^^

⑦ برنامج (دورة التطبيقي / 2017) : / 25 درجة /

في ترميز (ASCII) يعطى الرمز 20H للحرف الفراغ blank ويعطى الرمز 2AH للحرف (*) ، اكتب برنامجاً نعرف فيه السلسلة (I am good at assembly) ثم يقوم البرنامج بإجراء مسح SCAN السلسلة ويستبدل الفراغ بين الكلمات بالحرف (*) لتصبح السلسلة (I*am*good*at*Assembly) مع العلم أنه علم الصفر ZF

لا تهم في سجل الأعلام هو بت ذو الوزن (6) ..

*الطريقة الثانية (مع سلاسل رمزية):

عمود الأسماء

```
EXTRA   Segment   ← الكلمة المفتاحية للقاطعة
String  DB 'I am good at Assembly'     20 bit
           0 1 2 3 ...
           ↑ DI
        (DB ← بايتات)
EXTRA   ENDS  → اختصار segment
                                    } تعريف المتغيرات

CODE    Segment
        Assume CS:CODE , ES:EXTRA  → للربط مع الكتوبة لكي يفهم المعالج الأسماء القاطعات
        (Assume: اعتبر أن)
        CLD  → المؤشرات تصاعدية
        MOV  Cx, 21d  → عداد بعدد مرات المسح = عدد المحارف
        LEA  DI, String      21d وهو عداد زماني ← بدأ من عشري
        تحميل المؤشر على السلسلة
```

فحص SCASB → DI , ES → String

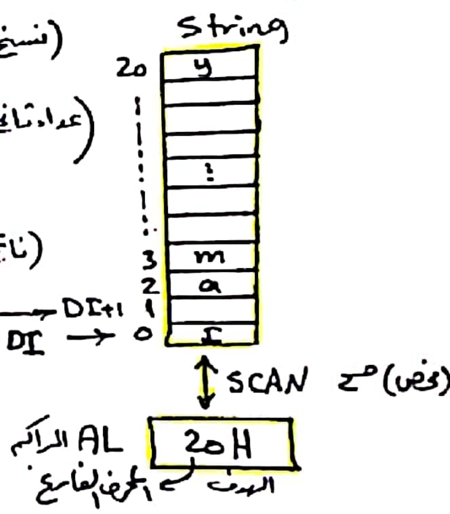
كل حرف هو بايت حتى حرف الفراغ

السلسلة موجهة ← القاطعة ES ← EXTRA (الاسم) إلا إذا ذكر الدكتور اسم آخر

أسماء المواقع نضعها :

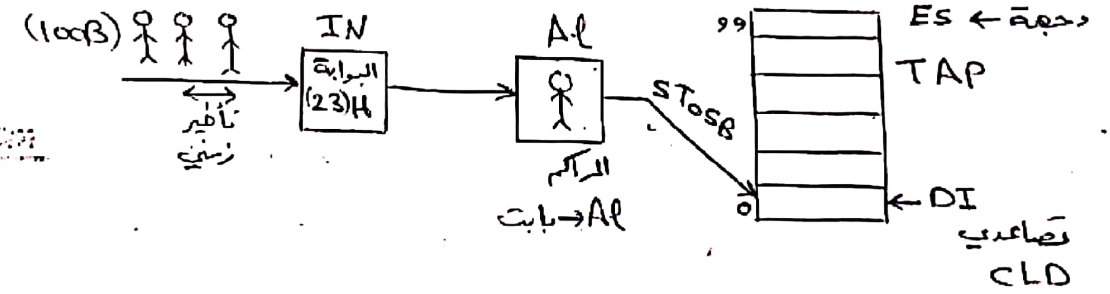
```
        MOV  AL , 20H    // سجل نضع فيه ترميز الحرف الفارغ الهدف (ننسخ الهدف في AX  AL)
        MOV  DL , 00H    // سجل مسؤول عن عدد المحارف المستبدلة (عداد ثاني DL للمحارف المستبدلة يبدأ من 0)
back:   SCASB
        JNZ  nextchar    (ناتج طرح السلسلة والهدف ≠ 0 أي) إذا لم يجد الهدف
وجدنا الهدف ← MOV [DI-1], 2AH *   // لأن المؤشر تلقائياً ينتقل [DI-1] عند بداية التعليمة الجديدة للمكان التالي ← نرجعه للمكان المطلوب
        INC  DL  → زيادة العداد        // ZF=1
nextchar: Loop back     // ZF=0 & ZF=1
CODE    ENDS
```

ارجع للموقع back ودائماً نستخدم التعليمة Loop مع CX حيث إنقاص CX يعتمد على Loop



② البرنامج الثاني: اكتب برنامج بلغة (Assembly) لادخال (100 Byte) لجدول اسمه TAP عن طريق بوابة (23H) مباشرة ذاكرة تصاعدية مع ترك تأخير زمني بين كل بايت وبايت عبر البوابة.

OSAMA Al-kassem




```
Time    EQU 147
EXTRA   segment
TAP     DB 100 DUP (?)
EXTRA   ENDS
CODE    segment
        ASSUME CS:Code , ES:EXTRA
        CLD //   عداد تصاعدي   Direction Flag = 0
        MOV CX, 100
        LEA DI, TAP //
Back:   IN Al , 23H
        STOSB //   ادخال بايت للذاكرة من الراكم
        CALL Delay //   تطلب التعليمة Call الاجرائية Delay المسؤولة عن التأخير
        Loop Back   ← تنقص الـ CX
        HLT
Delay   PROC Near   نوع الاجرائية      (اسم الاجرائية)
        MOV BX , Time ; 147H   } تأخير زمني داخلي
Count:  Loop Count              }
        RET   (الرجوع لتعليمات البرنامج)   تنقص السجل BX وتعوضه إذا كانت BX=0 ليتوقف عن التأخير.
Delay   ENDP   (نهاية الاجرائية)
CODE    ENDS
```

استخدمنا السجل BX بدلاً من CX لأن CX له وظيفة في البرنامج.

عداد ثاني.

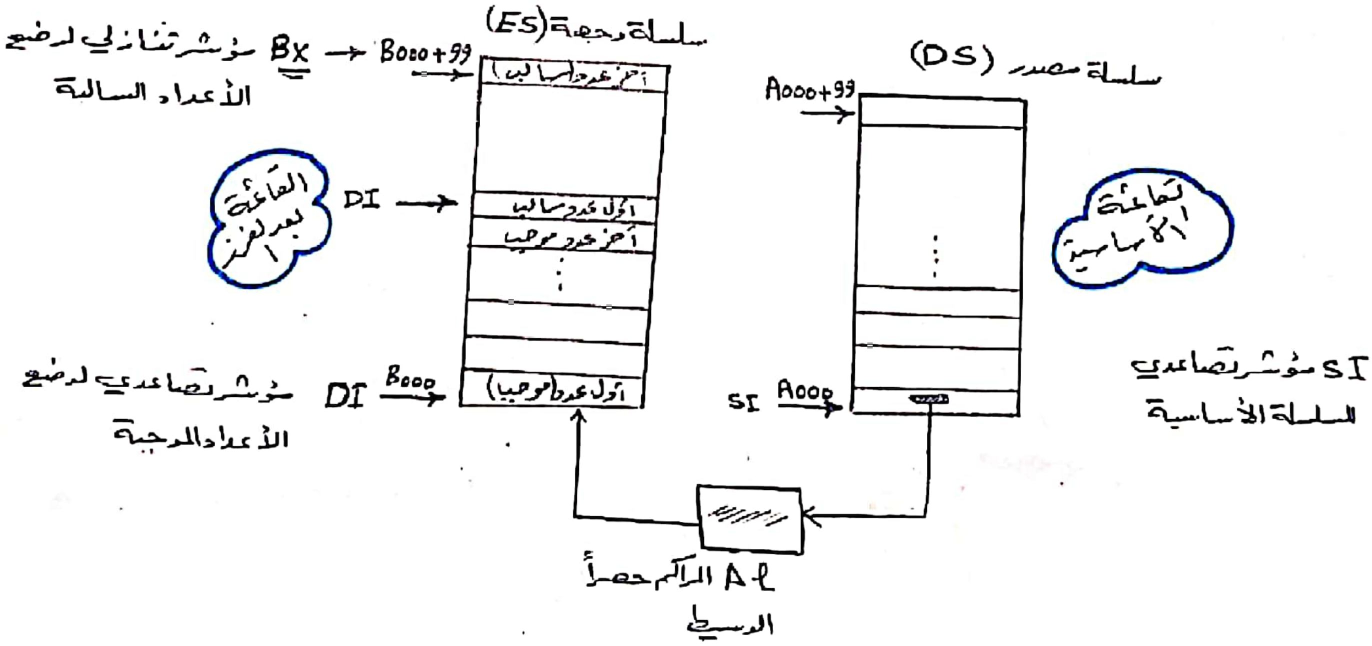
(زمن التأخير) الزمني يلي يعطي التأخير الزمني المطلوب

تستخدم الـ HLT عندما يكون تحت الـ Loop عمليات بدي تنفذها طالما الـ Loop شغالة ولكن لما تخلص الـ Loop أتجاوز التعليمات وأخرج من البرنامج ← HLT تلبي ذلك

البرنامج الثالث : دورة (2018)

بفرض لدينا قائمة مكونة من مئة بايت تحتوي مئة عدد صحيح مؤشر . اكتب إجرائية اسمها INTSORT من النوع (NEAR) مهمتها إعادة كتابة الأعداد المؤشرة بحيث تكتب الأعداد الموجبة بدءاً من العنوان المنطقي (B000) تليها الأعداد السالبة والتي يجب أن تنتهي عند العنوان B000+99d وذلك بفرض أن القائمة الأساسية تبدأ بالعنوان المنطقي (A000) وتنتهي بالعنوان المنطقي (A000+99d) .

الحل:




```
INTSORT   PROC  NEAR
          MOV  SI , A000H //
          MOV  DI , B000H //
          MOV  BX , DI
          ADD  BX , 99
          MOV  CX , 100
START     MOV  AL, [SI]
          INC  SI  → تصاعدي
          ADD  AL , 00H //
          JS   AGEN          ← إذا العدد سالب
          MOV  [DI], AL      ← العدد موجب
          INC  DI
          LOOP START
AGEN:     MOV  [BX], AL
          DEC  BX
          LOOP START
          RET
INTSORT   ENDP
```

ابدأ القائمة الرئيسية بالعنوان المنطقي A000H لمؤشر القائمة الأساسية SI وعنوان الأعداد الموجبة B000H لمؤشر القائمة الثانية .

تم استخدام السجل BX في عنونة الذاكرة حيث يتطلب البرنامج التعامل مع الذاكرة بعنوانه (مؤشر) محدد .

جمع العدد في المراكم مع الصفر لمعرفة إشارة العدد في المراكم فيما بعد وبعدها عن طريق (JS) حيث سيتم القفز للوسم (AGEN) إذا كان (SF=1) أي العدد AL سالب .

- تحميل المؤشرات على السلاسل من خلال العناوين المنطقية لأن لم يعطي اسم الجدول لاستخدم LEA
- عند تحميل BX خطأ نكتب Mov BX , B000+99 ليس بنفس الطول فنتم ذلك :

  Mov BX , DI ← تحوي B000H
  ADD BX, 99