**University of Al- Mustaqbal**

**College of Health and Medical Techniques**

**Intelligent Medical Systems Department**

# Microprocessors

*Lecture 1*

By

Reyam Thair Ahmed

# *Microprocessor*

هو عبارة عن شريحة ذات اطراف متعددة قادرة على تنفيذ البرنامج يطلق عليها باسم وحدة المعالج المركزية (CPU) وظيفته هي جلب الاوامر من الذاكرة وتنفيذها واحدة بعد الاخرى هذه العملية تتم عن طريق خطوط او مسارات تربط مكونات الكومبيوتر تسمى الناقلات (Buses).. المايكروبروسيسور يتحكم بالذاكرة واجهزة الأدخال والاخراج عن طريق هذه الناقلات.

**Fig.1 8086 Microprocessor**

**يقوم المعالج بتنفيذ مهام رئيسية:**

1. يجب ان يكون قادر على احضار المعلومات من الذاكرة هذه المعلومات قد تكون بيانات يحتاجها في عملية تنفيذ الاوامر.
2. يجب ان يكون لديه القدرة على اجراء العمليات الحسابية والمنطقية على البيانات التي يتم احضارها.
3. تنفيذ البرنامج المخزون بالذاكرة وفقا لسياق اوامر البرنامج.
4. جدولة المهام يجب ان يكون هناك مكان مناسب داخل المعالج لحفظ المعلومات لحين الحاجة اليها.

**من الجدير بالذكر بأن هنالك ثلاثة ممرات في الحاسب و هي:**

1. **ممر المعطيات DATA BUS** : و يصل بين المعالج و الذاكرة وظيفته نقل المعطيات من و إلى الذاكرة.
2. **ممر العناوين ADDRESS BUS** : و يصل بين المعالج و الذاكرة أيضاً و وظيفته نقل العناوين من المعالج إلى الذاكرة.
3. **ممر التحكم CONTROL BUS** : لتنسيق عمل الممرين السابقين(Synchronization) .

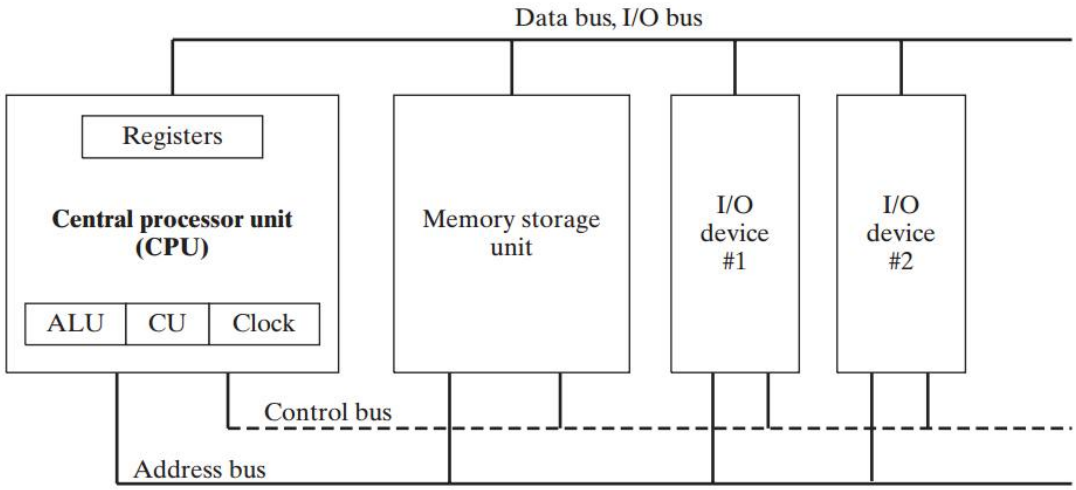



**Fig.2 Basic Microprocessor design**

- **Memory (الذاكرة)**

هو المكان او الوعاء لحفظ المعلومات لحين الحاجة اليها يكون بهيئة رقاقة او شريحة تتكون من مجموعة من الخلايا كل وحدة منها تخزن 1 بت يمكن لجهاز الكومبيوترالذي يحتوي على 1 ميغابايت ان يخزن 1 مليون من المعلومات , الذاكرة تعرض بشكل مرتب ومتسلسل يمكن تحديد كل بايت عن طريق رقم خاص يبدأ من 0كما في الشكل ادناه.

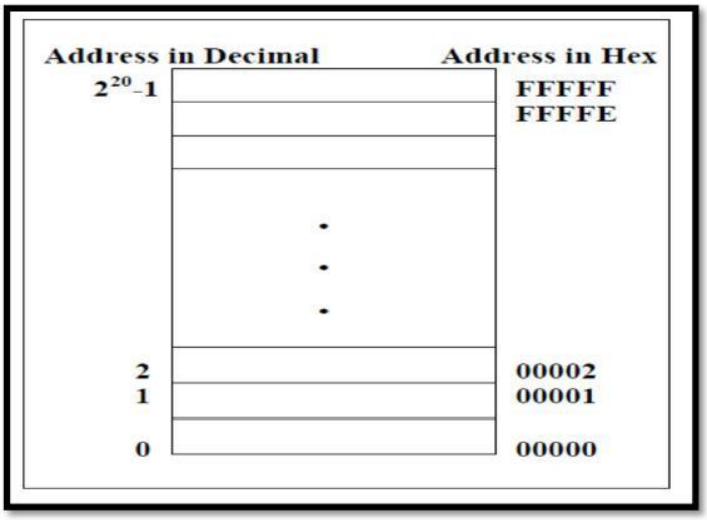


**Fig. 3 Logical view of the system memory**

يوجد بين المعالج و الذاكرة ممران هما ممر المعطيات (Data bus) بعرض 16 بت و ممر العناوين (Address bus) بعرض 20 بت. إن كون ممر العناوين (Address bus) ذو عرض 20 بت ( 20 خط نقل) هذا يعني أنه يستطيع نـقل رقم ثنائي ذو 20 خانة أي أن أكبر قيمة يمكن وضعها على ممر العناوين هي:

$$2^{20} = 1048576 \approx 1MB$$

و بذلك يستطيع المعالج 8086 عنونة واحد ميغا من الذاكرة فقط.

## *Types of memory*

### 1- Read Only Memories (ROM):

تقوم هذه الذاكرة بحفظ البيانات والمعلومات التي يقوم المستخدم بإجراء العمليات المختلفة عليها، حيث تصلها البيانات والمعلومات من وحدات الإدخال المختلفة، كما تصلها النتائج من وحدات المعالجة بعد إجراء المطلوب عليها، وتقوم RAM بتخزين كل هذه البيانات بشكلٍ مؤقت، وتتأثر هذه الذاكرة بانفصال التيار الكهربائي عن جهاز الكمبيوتر أو إغلاق الكمبيوتر، حيث تختفي كل المعلومات التي خزَّنتها، وتعتبر هذه الذاكرة أسرع من الROM.

### 2- Read/Write Memory (RAM):

تقوم هذه الذاكرة بتخزين برامج التشغيل والبرامج الأساسيّة التي تقوم بتشغيل جهاز الكمبيوتر، وهذه الذاكرة لا يمكن التعديل عليها أو محوها لأنها مخزَّنة من الشركة المصنّعة الرئيسة، ولا يمكن لجهاز الكمبيوتر أن يعمل من دون هذه الذاكرة، كما أنّها لا تتأثر بانقطاع التيار الكهربائي عن الجِهاز، بل يمكن استعادة بياناتها عند تشغيل الجِهاز مرةً أخرى.

## Registers المسجلات

هي عبارة عن مواقع تخزين مؤقته توجد داخل المعالج بهيئة خانات ثنائية تحتفظ بالبيانات لحين الحاجة اليها وتنفيذها بسرعة كبيرة.

يملك المعالج 8086 أربعة مجموعات من المسجلات ذات 16 بت يستطيع المبرمج الوصول إليها و هي:

- **أربعة(Segment Registers) مسجلات مقاطع CS,DS,SS,ES**
- **أربعة(Data Registers) مسجلات معطيات AX,BX,CX,DX .**

- **أربعة (Index Registers)مسجلات تأشير و فهرسة SI,DI,BP,SP .**
- **مؤشر التعليمة IP**

بالإضافة إلى ذلك يوجد مسجل آخر هو مسجل الأعلام **Flag Register** و يدعى أيضاً مسجل الحالة و هو مسجل ذو 16 بت و لكن نستخدم منه 9 خانات فقط.

سنشرح كلٍ من هذه المسجلات بالتفصيل

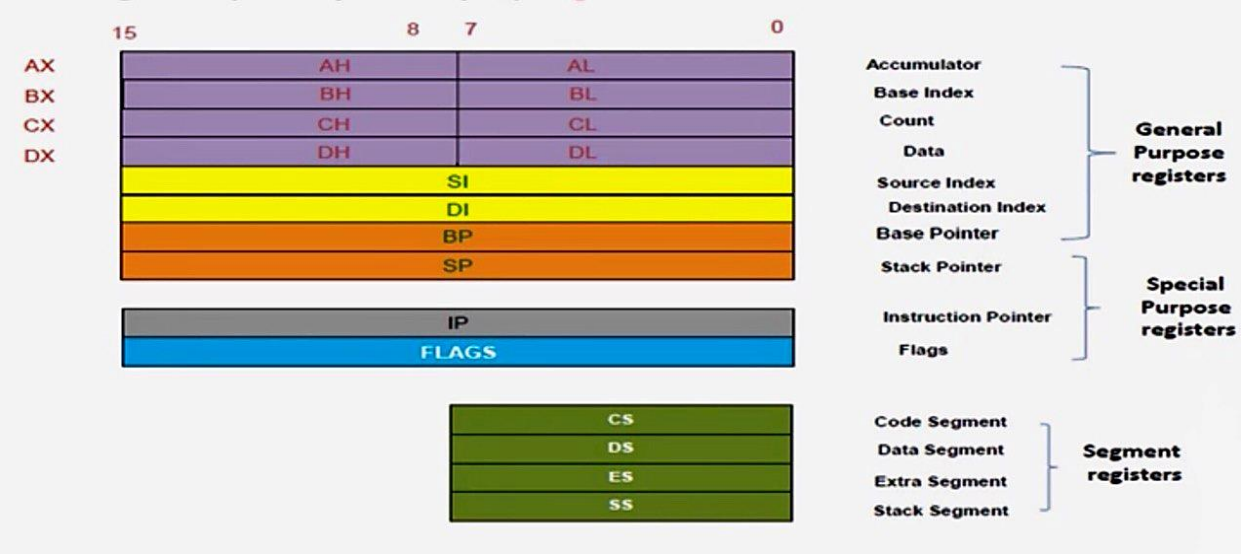



**Fig.4 Registers Type**

- **مقاطع الذاكرة (Segment Registers):**

هي عبارة عن مناطق مخصصة للبرنامج في الذاكرة هذه المسجلات تكون للاغراض العامة اي يمكن للمبرمج ان الوصول اليه وتغيير محتوياته.

يتعامل المعالج كما ذكرنا مع واحد ميغا من الذاكرة، و يمكن أن نقتطع من هذه الميغا أربعة مقاطع أساسية يتعامل معها برنامجنا بشكل مباشر (أي أنه لا تتم الاستفادة من كل الذاكرة بآن واحد) و هذه المقاطع الأربعة هي:

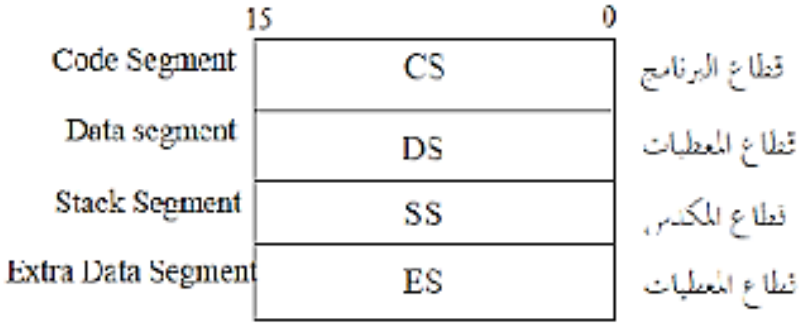



**Fig. 5 Segment Registers**

1- **مقطع الشيفرة Code Segment CS**

يخصص هذا المقطع من الذاكرة –كما هو واضح من تسميته- لتخزين شيفرة البرنامج. (نضع فيه البرنامج المراد تنفيذه، اي مكان مخصص بالذاكرة للكودات).

2- **مقطع المعطيات Data Segment DS**

يخصص هذا المقطع من الذاكرة لتخزين المعطيات و المتحولات التي يتعامل معها البرنامج. هذه البيانات يمكن ادخالها عن طريق اجزاء طرفية مثل (Keyboard).

3- **مقطع المكدس Stack Segment SS**

يخصص هذا المقطع للحفظ المؤقت لبعض المعلومات الضرورية و التي يخشى أن تضيع أو تتغير أثناء تنفيذ برنامج ما. بمعنى اخر يستخدم لتخزين البيانات التي نستخدمها بإكثر من عملية داخل البرنامج.

- **(آلية عمل Stack-** Last In First Out LIFO ) آخر ما يدخل أول ما يخرج.

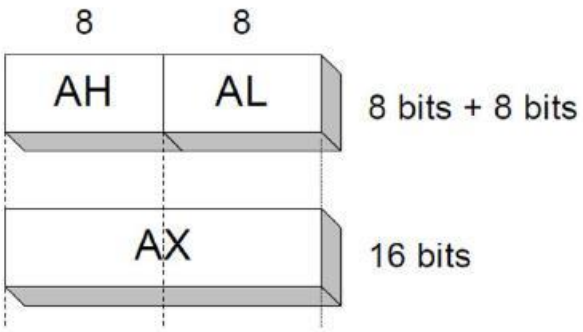
4- **مقطع المعطيات الإضافي Extra Segment ES**

هو عبارة عن مقطع اضافي يستخدم عند الحاجة (اذا كان عندنا برنامج يوجد فيه برنامج فرعي نستطيع ان ننفذه بمقطع خارجي حتى لا يأخذ مكان من البرنامج الاصلي) و بذلك نستطيع الاستفادة من مساحة أكبر في الذاكرة.

## Data Registers (مسجلات البيانات):

تستخدم هذه المسجلات من أجل التخزين المؤقت للنتائج المرحلية أثناء تنفيذ البرنامج حيث أن تخزين المعطيات في هذه المسجلات يمكننا من الولوج إلى تلك المعطيات بشكل أسرع مما لو كانت في الذاكرة. تقسم المسجلات إلى(Ax, Bx, Cx, Dx) هذه المسجلات ضمن المعالج هي الوحيدة التي يمكن ان تقسم الى جزأين.

يمكن التعامل مع كل من هذه المسجلات على أنه وحده واحدة بحجم 16-bits أو على وحدتين كل واحدة بسعة 8-bits إحداهما العليا HIGH والثانية المنخفضة LOW مثلا يمكن التعامل مع المسجل AX على انه مسجل بحجم 16-bitsأو التعامل مع النصف العلوي AH (HIGH) على انه مسجل 8-bits والمسجل المنخفض AL (LOW) على أنه مسجل 8-bitsوبالمثل مع المسجلاتC، D، Bوبالتالي يصبح لدينا 8 مسجلات من النوع 8-bits أو أربعة مسجلات من النوع 16-bits.




**1- المسجل AX (Accumulator)**

يعتبر المسجل AX هو المسجل المفضل للاستخدام في عمليات الحساب والمنطق ونقل البيانات والتعامل مع الذاكرة وموانئ الإدخال والإخراج. واستخدامه يولد برامج أقصر ويزيد من كفاءة البرنامج. حيث يجب مثلا في عمليه ضرب رقمين وضع أحد الرقمين فيه مع وضع القيمة المطلوب إخراجها إلي ميناء خروج محدد فيه ثم تتم قراءه القيمة التي يتم إدخالها من ميناء خروج محدد فيه دائما. وعموما يتم التعامل مع المسجل AX على أنه أهم المسجلات الموجودة في المعالج.

**2- المسجل BX (Base Register)**

يستخدم المسجل BX في عنونه الذاكرة حيث تتطلب بعض العمليات التعامل مع الذاكرة بمؤشر محدد ويتم تغيير قيمه المؤشر لإجراء عمليه مسح لجزء محدد من الذاكرة كما سنرى فيما بعد.

**3- المسجل CX (Count Register)**

يتم استخدام المسجل CX كعداد للتحكم بعدد مرات تكرار مجموعه محدده من التعليمات.كذلك يتم استخدامه في تكرار عمليه دوران مسجل لعدد محدد من المرات.

**4- المسجل DX (Data Register )**

يتم استخدامه في حفظ البيانات الخاصة بعمليات الضرب والقسمة كذلك يتم استخدامه كمؤشر لموانئ الإدخال والإخراج عند استخدام عمليات الإدخال والإخراج.

- **مسجلات المؤشرات Index Registers**

و هي عبارة عن منطقة مخصصة بالذاكرة للتعامل مع البرنامج ، تخزن فيها كودات وبيانات البرنامج او المكدس (stack). (مؤشر يحتوي على ازاحة البيانات (offset) والتعليمات (Instruction)).

أربعة مسجلات مساعدة تساعد في إيجاد العنوان الفيزيائي بالتعاون مع مسجلات المقاطع، و طول هذه المسجلات 16 بت، و هي :

1- **مسجل دليل المصدر Source Index SI** : يخزن فيه عنوان يدل على الإزاحة ضمن مقطع المعطيات DS و بمعنى آخر يستخدم للتعامل مع بيانات البرنامج المخزنة في (Data Segment).
2- **مسجل الوجهة Destination Index DI** : يخزن فيه عنوان يدل على الإزاحة ضمن مقطع المعطيات الإضافي ES ، و بمعنى آخر يستعمل مسجل دليل الهدف DI من أجل استنتاج العنوان الفيزيائي الذي يحدد حجرة متحول الهدف.
3- **مسجل مؤشر المكدس Stack Pointer SP** : يسمح مؤشر المكدس بوصول سهل للحجرات في مقطع المكدس الموجود في الذاكرة حيث أن القيمة في SP تمثل العنوان الفعال لحجرة المكدس التالية التي يمكن الوصول إليها نسبة إلى العنوان الحالي الموجود في مسجل مقطع المكدس SS و يحتفظ SP دوماً بقيمة تدل على قمة (Top) المكدس (stack)، هذا و إن قيمة هذا المسجل تتعدل تلقائياً عند وضع أو سحب معلومة بالمكدس.
4- **مسجل مؤشر القاعدة Base Pointer BP** : يستخدم كمؤشر اساسي لتحديد البيانات المراد قرائتها داخل المكدس (stack) عند استدعائها من البرنامج.

- **Status and Control Registers مسجلات الحالة والتحكم**

1. **مسجل مؤشر التعليمة Instruction Pointer IP**

هذا المسجل يحدد موقع التعليمة التالية التي ستنفذ في مقطع الشيفرة و بعد جلب شيفرة التعليمة من الذاكرة فإن BIU تعدل قيمة IP بحيث تشير إلى التعليمة التالية في الذاكرة ( التعديل يتم آلياً ).

2. **مسجل الأعلام Flags Register**

هو مسجل ذو 16 بت موجود في وحدة التنفيذ فائدته يوضح حالة العملية بعد التنفيذ كما هو واضح بالشكل :

Fig.6 Flag Register

و كما نلاحظ من الشكل السابق أنه يوجد ستة أعلام للحالة هي CF, PF, AF, ZF, SF, OF ، و كذلك يوجد ثلاثة أعلام للتحكم DF, IF, TF .

# ايعازات المعالج الدقيق 8086

## 1- ايعاز النقل Mov

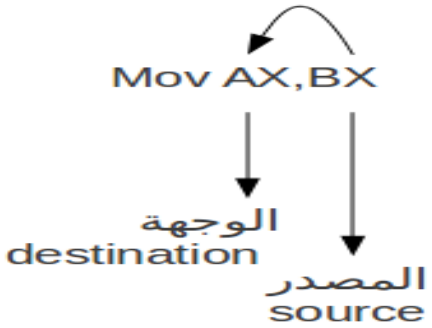
يقوم هذا الايعاز بنقل البيانات من مكان الى اخر دون اجراء اي تعديل او تغيير عليها (نسخ البيانات). يقوم هذا الايعاز بنقل البيانات من الذاكرة الى السجل او من السجل الى الذاكرة او من سجل الى سجل اخر.

REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

SREG: DS, ES, SS, and only as second operand: CS.

Memory: [BX], [BX+SI+7], variable, etc..

Immediate: 5, -24, 3Fh, 10001101b, etc...




- يجب ان يكون المصدر والوجهة بنفس الحجم اما كلاهما 8 بت او كلاهما 16 بت.

امثلة:

| | | |
|---|---|---|
| Mov CL,CH | √ | نقل محتويات السجل (CH) الى السجل (CL) |
| Mov CL,[2000H] | √ | نقل محتويات الذاكرة التي عنوانها (2000) الى السجل (CL) |
| Mov [100H],BL | √ | نقل محتويات السجل (BL) الى موقع الذاكرة (100) |
| Mov ES,DS | ✖ | Segment to segment |
| Mov BL,DX | ✖ | Mixed size |
| Mov CS,AX | ✖ | Code segment register may not be the destination register |

## 2- الايعازات الحسابية : تشمل هذه الايعازات الجمع والطرح والضرب والقسمة.

### 1. ايعاز الجمع ADD:

في هذا الايعاز يتم جمع المصدر Source مع الوجهة Destination وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

ADD Destination, Source

امثلة:

ADD AL,0FH يجمع القيمة 0FH مع محتويات السجل ويخزن النتيجة في السجل

ADD AX, BX AX+BX وتخزن النتيجة في AX

**2. ايعاز ADC الجمع مع carrier:**

في هذا الايعاز يتم جمع المصدر Source مع الوجهة Destination مع قيمة ال CF وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

ADC Destination, Source

امثلة:

```
STC       ; set CF = 1          يجعل قيمة ال carrier واحد
MOV AL, 5 ; AL = 5
ADC AL, 1 ; AL = 7              AL+CF+1 ⟶ AL
```

**3. ايعاز الطرح SUB:**

في هذا الايعاز يتم طرح المصدر Source مع الوجهة Destination وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

SUB Destination, Source

امثلة:

SUB AL,0FH تطرح القيمة 0FH مع محتويات السجل ويخزن النتيجة في السجل

SUB AX, BX AX-BX وتخزن النتيجة في AX

**4. ايعاز الطرح SBB:**

في هذا الايعاز يتم طرح المصدر Source مع الوجهة Destination مع استعارة (CF) من معامل الوجهة وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

SBB Destination, Source

امثلة:

```
STC       ; set CF = 1          يجعل قيمة ال carrier واحد
MOV AL, 5     ; AL = 5
SBB AL, 3     ; AL = 1          AL- CF- 3=1 ⟹ AL
```

**5. ايعاز الضرب MUL:**

يقوم هذا الايعاز بضرب المصدر في المسجل (AL)اذا كان المصدر 8-Bit او AX اذا كان المصدر 16-bit وتوضع النتيجة دائما في السجل AX اذا كان 8بت، واذا كان 16 بت يضع النتيجة في AXDX.

الصيغة العامة لهذا الايعاز هو:

MUL Source

امثلة:

```
MUL CL:       AX ⇐ CL*AL
MUL CX:       AXDX ⇐ CX*AX
MUL 002F      ✖
```

هذه الصيغة غير صحيحة يجب وضع القيمة في سجل اولا ثم نستخدم السجل في عملية الضرب فتصبح:

```
MUL CX, 002FH
MUL CX
```

الصيغة الصحيحة

**6. ايعاز القسمة DIV:**

يقوم هذا الايعاز بقسمة السجل AX على المصدر في حالة 8-bit او السجلين AX DX على المصدرفي حالة 16-bit. اي دائما يكون المصدر هو المقسوم عليه.

الصيغة العامة لهذا الايعاز هو:

DIV Source

$$\frac{AX}{Source\ (8bits)} = \underset{\text{الناتج}}{AL} \quad \underset{\text{باقي القسمة}}{AH}$$

$$\frac{AXDX}{Source\ (16bits)} = \underset{\text{الناتج}}{AX} \quad \underset{\text{باقي القسمة}}{DX}$$

امثلة:

DIV BL: AL ⟸ AX/BL AH ⟹ الباقي

DIV BX: AX ⟸ AXDX/BX DX ⟹ الباقي

DIV 002F ✖

هذه الصيغة غير صحيحة يجب وضع القيمة في سجل اولا ثم نستخدم السجل في عملية الضرب فتصبح:

DIV CX, 002FH

DIV CX الصيغة الصحيحة

**7. ايعاز الزيادة INC:**

يعمل هذا الايعاز على زيادة محتويات المصدر بمقدار واحد.

الصيغة العامة لهذا الايعاز هو:

INC Source

امثلة:

INC BL: BL + 1

INC [SI]: [SI] + 1

**8. ايعاز النقصان DEC:**

يعمل هذا الايعاز على نقصان محتويات المصدر بمقدار واحد.

الصيغة العامة لهذا الايعاز هو:

DEC Source

امثلة:

DEC BL:  BL - 1

DEC [SI]:  [SI] - 1

**9. ايعاز المقارنة CMP:**

يستخدم هذا الايعاز للمقارنة بين رقمين حيث يقوم بطرح المصدرSource من الوجهة Destination واذا كان الناتج :

Source = Destination ⟶ Zero Flag=1

Source > Destination ⟶ Carry Flag=1

Source < Destination ⟶ Carry Flag=0

الفائدة من هذا الايعاز هو تأثر Flag بنتيجة الطرح.

## 3- الايعازات المنطقية (Logical)

1- **OR**: في هذا الايعاز تتم عملية OR بين المصدر Source والوجهة Destination وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

OR Destination, Source

1 OR 1 = 1
1 OR 0 = 1
0 OR 1 = 1
0 OR 0 = 0

امثلة:

MOV CL, 0A ; (00001010)

OR CL, F5 يجمع القيمة (1111 0101) مع محتويات السجل ويخزن النتيجة في السجل

CL=0A= 0000 1010

F5= 1111 0101

11111111

2- **NOT** : يعكس كل بت (اذا كان البت 0 يحوله الى 1 والعكس صحيح).
الصيغة العامة لهذا الايعاز هو:

NOT Destination, Source

امثلة:

MOV AL, 00011011b  يقوم بنقل القيمة الى السجل AL
NOT AL ; AL = 1110010  يقوم بقلب كل بت من البتات ويخزن النتيجة في السجل

3- **AND**: في هذا الايعاز تتم عملية AND بين المصدر Source والوجهة Destination وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

AND Destination, Source

1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0

امثلة:

MOV AL, A2  يقوم بنقل القيمة 1010 0010 الى السجل
AND BL, A2 ;  BL =02= 0000 0010
AL=A2= 1010 0010
---
00000010

4- **XOR**: في هذا الايعاز تتم عملية XOR بين المصدر Source والوجهة Destination وتوضع النتيجة في الوجهة. الصيغة العامة لهذا الايعاز هو:

XOR Destination, Source

1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0

امثلة:

MOV AL, A2  يقوم بنقل القيمة 1010 0010 الى السجل

XOR BL, A2 ;

BL =02= 0000 0010
AL=A2= 1010 0010
---
10100000