ADDRESSIN MODES أنماط العنونة

- **Every instruction of a program has to operate on a data.**
- **The different ways in which a source operand is denoted in an instruction are known as addressing modes.**

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

Group I : Addressing modes for register and immediate data

Group II : Addressing modes for memory data

Group III : Addressing modes for I/O ports

Group IV : Relative Addressing mode

Group V : Implied Addressing mode

تعمل كل تعليمة من تعليمات البرنامج على معطيا ت المعامل

أنماط العنونة يوجد عدة طرق للوصول الى معاملات المصدر

تصنف أنماط العنونة:

1- نمط العنونة الفوري ونمط العنونة السجلاتي

2- نمط العنونة بالوصول الى الذاكرة مباشرة Direct addressing mode

3- نمط العنونة Indirect addressing mode غيرالمباشرة

4- عنونة وحدات الادخال /الاخراج

2. Immediate Addressing

In immediate addressing mode, an 8-bit or 16-bit data is specified as part of the instruction

Example:

MOV DL, 08H

The 8-bit data (08_H) given in the instruction is moved to DL

(DL) ← 08_H

MOV AX, 0A9FH

The 16-bit data ($0A9F_H$) given in the instruction is moved to AX register

(AX) ← $0A9F_H$

Group I : Addressing modes for register and immediate data

1. Register Addressing

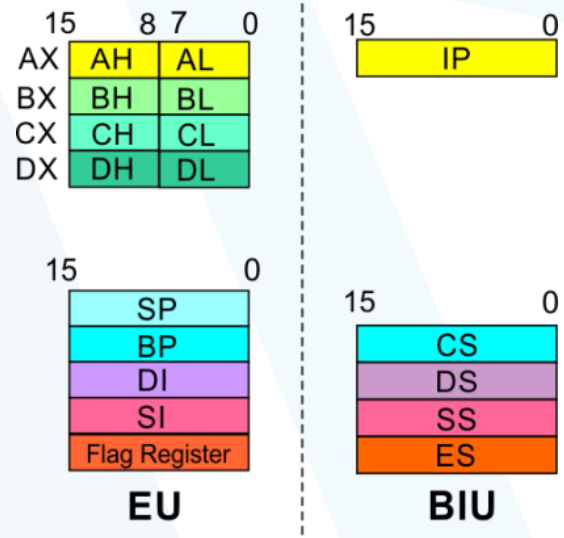
The instruction will specify the name of the register which holds the data to be operated by the instruction.

Example:

MOV CL, DH

The content of 8-bit register DH is moved to another 8-bit register CL

(CL) ← (DH)



Addressing Modes : Memory Access

- 20 Address lines ⇒ 8086 can address up to 2^{20} = 1M bytes of memory
- However, the largest register is only 16 bits
- Physical Address will have to be calculated **Physical Address : Actual address of a byte in memory. i.e. the value which goes out onto the address bus.**
- Memory Address represented in the form – **Seg : Offset** (Eg - 89AB:F012)
- Each time the processor wants to access memory, it takes the contents of a segment register, shifts it one hexadecimal place to the left (same as multiplying by 16_{10}), then add the required offset to form the 20- bit address

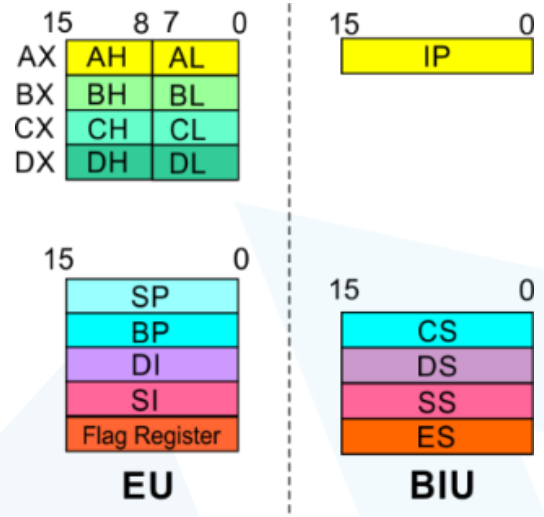
16 bytes of contiguous memory

89AB : F012 → 89AB → 89AB0 (Paragraph to byte → 89AB x 10 = 89AB0)
F012 → 0F012 (Offset is already in byte unit)
+ -------
98AC2 (The absolute address)

- To access memory we use these four registers: BX, SI, DI, BP
  Combining these registers inside [ ] symbols, we can get different memory locations (Effective Address, EA)
- Supported combinations:

[BX + SI] [BX + DI] [BP + SI] [BP + DI]	[SI] [DI] d16 (variable offset only) [BX]	[BX + SI + d8] [BX + DI + d8] [BP + SI + d8] [BP + DI + d8]
[SI + d8] [DI + d8] [BP + d8] [BX + d8]	[BX + SI + d16] [BX + DI + d16] [BP + SI + d16] [BP + DI + d16]	[SI + d16] [DI + d16] [BP + d16] [BX + d16]

BX BP	SI DI	+ disp



3. Direct Addressing نمط العنونة المباشرة

نمط العنونة المباشر يكون موقع العنوان الفعال للمعامل ضمن التعليمة
العنوان الفعال effective address رقم من 16 bit يكتب مباشرة ضمن التعليمة

Example

```
MOV   BX, [1354H]
MOV   BL,  [0400H]
```

تشير الأقواس المتوسطة حول الرقم 1354 الى محتوى موقع العنوان
وعند تنفيذ التعليمة يتم نقل محتوى العنوان الى المسجل BX

سمي نمط العنونة المباشرة لأن الازاحة للمعامل عن بداية المقطع ضمن التعليمة

Group II : Addressing modes for memory data

4. Register Indirect Addressing نمط العنونة غير المباشرة

In Register indirect addressing, name of the register which holds the effective address (EA) will be specified in the instruction.

Registers used to hold EA are any of the following registers:

BX, BP, DI and SI.

Content of the DS register is used for base address calculation.

العنونة غير المباشرة تستخدم المسجلات للاحتفاظ بالعنوان الفعال EA

BX, BP, DI and SI.

يستخدم المسجل DS لحساب عنوان القاعدة

Example:

MOV CX, [BX]

Note : Register/ memory enclosed in brackets refer to content of register/ memory

Operations:

EA = (BX) حساب العنوان الفعال

BA = (DS) x 16_{10} حساب عنوان القاعدة

MA = BA + EA حساب عنوان الذاكرة

(CX) ← (MA) or,

(CL) ← (MA)
(CH) ← (MA +1)

Group II : Addressing modes for memory data

When BX holds the base value of EA, 20-bit physical address is calculated from BX and DS.

When BP holds the base value of EA, BP and SS is used.

5. Based Addressing العنونة القاعدية

العنونة القاعدية يستخدم المسجلات BX , BP لتخزين العنوان الفعال والازاحة اذا كانت من 8 bit أو من 16 bit معرفة في التعليمة

اذا كانت الازاحة من 8 bit توسع لتصبح من 16 bitقبل الإضافة الى قيمة القاعدة

Example:

MOV AX, [BX + 08H]

Operations:

$0008_H \leftarrow 08_H$ (Sign extended) 16 bit توسيع الازاحة الى

$EA = (BX) + 0008_H$ حساب العنوان الفعال

$BA = (DS) \times 16_{10}$ حساب عنوان القاعدة

$MA = BA + EA$ حساب عنوان المعامل في الذاكرة

$(AX) \leftarrow (MA)$ or,

$(AL) \leftarrow (MA)$ نقل محتوى العنوان الى القسم الأدنى من المسجل AL

$(AH) \leftarrow (MA + 1)$ اضافة 1 الى العنوان التالي ووضع قيمته في المسجل AH

6. Indexed Addressing العنونة الدليلية(المفهرسة)

تستخدم المسجلات SI, DI لتخزين دليل قيمة المعطيات في الذاكرة

اذا كانت الازاحة من 8 BIT أو من 16 bit تكون ضمن التعليمة

يضاف مقدار الازاحة الى قيمة الدليل الموجود في أحد المسجلات SI , DI

Example:

MOV CX, [SI + 0A2H]

Operations:

$FFA2_H \leftarrow A2_H$ (Sign extended)

EA = (SI) + $FFA2_H$
BA = (DS) x 16_{10}
MA = BA + EA

(CX) ← (MA) or,

(CL) ← (MA)
(CH) ← (MA + 1)

case of 8-bit displacement, it is sign extended In to 16-bit before adding to the base value.

7. Based Index Addressing العنونة القاعدية المفهرسة

In Based Index Addressing, the effective address is computed from the sum of a base register (BX or BP), an index register (SI or DI) and a displacement.

Example:

MOV DX, [BX + SI + 0AH]

Operations:

$000A_H \leftarrow 0A_H$ (Sign extended)

EA = (BX) + (SI) + $000A_H$
BA = (DS) x 16_{10}
MA = BA + EA

(DX) ← (MA) or,

(DL) ← (MA)
(DH) ← (MA + 1)

العنونة القاعدية المفهرسة

يحسب العنوان الفعال من مجموع مسجل القاعدة BX, BP اضافة مسجل الدليل SI, DI ومقدار الازاحة

مثال احسب عنوان المعامل في الذاكرة بفرض

SI= 0020 , BX= 1234 , DS =0050

MOV DX, [BX + SI + 0AH]

الحل

000AH 0AH (Sign extended) ←

EA = (BX) +(SI) +000A = 1234 + 0020 + 000A = 125E

BA = 00500

MA = 00500 + 125E = 0175E

(DL) ←(MA)

(DH)← (MA + 1)

Employed in string operations to operate on string data.

The effective address (EA) of source data is stored in SI register and the EA of destination is stored in DI register.

Segment register for calculating base address of source data is DS and that of the destination data is ES

Note : Effective address of the Extra segment register

Example: **MOVS BYTE**

Operations:

Calculation of source memory location:

EA = (SI) BA = (DS) x 16_{10} MA = BA + EA

Calculation of destination memory location:

EA_E = (DI) BA_E = (ES) x 16_{10} MA_E = BA_E + EA_E

(MAE) ← (MA)

If DF = 1, then (SI) ← (SI) – 1 and (DI) = (DI) - 1
If DF = 0, then (SI) ← (SI) +1 and (DI) = (DI) + 1

9. Direct I/O port Addressing

These addressing modes are used to access data from standard I/O mapped devices or ports.

In **direct port addressing mode**, an 8-bit port address is directly specified in the instruction.

Example: IN AL, [09H]

Operations: $PORT_{addr} = 09_H$
(AL) ← (PORT)

Content of port with address 09_H is moved to AL register

10. Indirect I/O port Addressing

In indirect port addressing mode, the instruction will specify the name of the register which holds the port address. In 8086, the 16-bit port address is stored in the DX register.

Example: OUT [DX], AX

Operations: $PORT_{addr} = (DX)$
(PORT) ← (AX)

Content of AX is moved to port whose address is specified by DX register.

12. Implied Addressing

تعليمات لاتحتاج الى معاملات

Instructions using this mode have no operands. The instruction itself will specify the data to be operated by the instruction.

Example: **CLC** **carry** تصفير علم

This clears the carry flag to zero.

11. Relative Addressing

نمط العنونة Relative Addressing

يعرف العنوان الفعال في مسجل التعليمة **IP وتو سيع الازاحة الى 16 bit**

Example: **JZ 0AH**

Operations:

$000A_H \leftarrow 0A_H$ (sign extend)

If ZF = 1, then

$EA = (IP) + 000A_H$
$BA = (CS) \times 16_{10}$
MA = BA + EA

If ZF = 1, then the program control jumps to new address calculated above.

If ZF = 0, then next instruction of the program is executed.