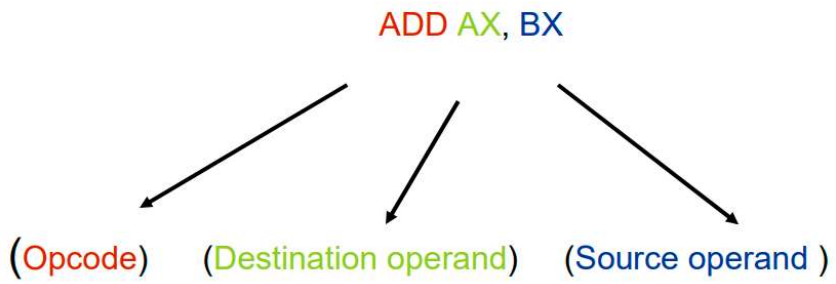
المحاضرة الأولى

توليد ترميز التعليمات

عند تنفيذ التعليمات في المعالج فإن هذه التعليمة تقسم إلى مجموعة من الأقسام كما يلي:



يتم تمثيل التعليمات في المعالج وفقاً لمجموعة من البتات:

Opcode - 6	D - 1	W -1	1st byte
MOD - 2	Reg - 3	R/M - 3	2nd byte
Displacement or data (optional) up to 4 bytes			

يمثل البايتين الأول والثاني تعريف التعليمة بشكل عام حيث يتألف البايت الأول من ثلاثة أقسام وهي ترميز التعليمة opcode بمقدار ستة بتات و D يحدد نمط التعامل مع مسجل أو ذاكرة بمقدار 1 بت وحجم الكلمة W بمقدار 1 بت. في حيث يحدد البايت الثاني نمط عمل التعليمة MOD في حال ذاكرة أو سجل و Reg بمقدار 3 بت تحدد أي مسجل يتم التعامل معه. و R/M بمقدار 3 بت تحدد هدف التعليمة.

تحدد D عملية المسجل في حقل REG في البايت الثاني في حال كونها مصدر أو هدف حيث تأخذ القيمة 1 في حال كان هدف و 0 في حال كان مصدر. يحدد البت W حجم التعليمة حيث يأخذ القيمة 0 في حال كان التعامل مع عمليات من حجم 8 بت و1 في حال 16 بت.

- The d (direction) field specifies the direction of data movement:
  - d = 1 destination is operand specified by REG field
  - d = 0 destination is operand specified by R/M field

يتم الحصول على قيم بتات REG من الجدول التالي:

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

ويتم الحصول على قيم MOD من الجدول التالي:

CODE	EXPLANATION
00	Memory Mode, no displacement follows*
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

حيث 00 التعامل فقط مع موقع ذاكرة و 01 التعامل مع موقع ذاكرة بإزاحة 8 بت و 10 التعامل مع موقع ذاكرة بإزاحة 16 بت و 11 التعامل فقط مع مسجلات.

يتم الحصول على قيمة R/M من الجدول:

MOD = 11			EFFECTIVE ADDRESS CALCULATION			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16

مثال:

لنأخذ التعليمة MOV BL,AL

ترميز تعليمة MOV هو 100010 , بما أن المصدر هو مسجل AL أي D=0 , بما أن المسجل AL وليس AX وبالتالي فإن التعامل مع تعليمات بطول 8bit وبالتالي W=0. بما أننا ننقل من مسجل إلى مسجل أي MOD=11 بما أن المصدر هو AL أي REG=000 وبما أن الهدف هو BL أي R/M=011.

MOV BL,AL **($88C3_{16}$)**

Opcode for MOV = 100010

D = 0 (AL source operand)

W bit = 0 (8-bits)

Therefore byte 1 is $10001000_2=88_{16}$

- MOD = 11 (register mode)
- REG = 000 (code for AL)
- R/M = 011 (destination is BL)

Therefore Byte 2 is $11000011_2=C3_{16}$

أمثلة عملية:

MOV reg/mem to/from reg/mem

- This instruction has the structure:

  `100010dw  modregr/m  Disp-lo  Disp-hi`

- Where 0, 1 or 2 displacement bytes are present depending on the MOD bits
- **`MOV AX,BX`**

  w = 1 because we are dealing with words

  MOD = 11 because it is register-register
- if d = 0 then REG = source (BX) and R/M = dest (AX)

  `= 1000 1001  1101 1000 (89 D8)`
- if d = 1 then REG = source (AX) and R/M = dest (BX)

  `= 1000 1011  1010 0011 (8B C3)`

MOV reg/mem to/from reg/mem

- **`MOV [BX+10h],CL`**

  w = 0 because we are dealing with a byte

  d = 0 because we need R/M Table 2 to encode [BX+10h]
- therefore first byte is **`10001000 = 88H`**
- since 10H can be encoded as an 8-bit displacement, we can use

  `MOD=01 REG=001 and R/M=111  = 0100 1111 = 4FH`
- and the last byte is 10H

  `result: 88 4F 10`

  `Note: MOV [BX+10H],CX = 89 4F 10`
- since 10H can also be encoded as a 16-bit displacement, we can use

  `MOD=10 REG=001 and R/M=111  = 1000 1111 = 8FH`
- and the last bytes are 00 10

  `result: 88 8F 00 10`

MOV reg/mem, imm

- This instruction has the structure:

```
1100 011w   MOD 000 R/M disp1    disp2
```

- Where 0, 1 or 2 displacement bytes are present depending on value of MOD
- **`MOV BYTE PTR [100h],10h`**

  w = 0 because we have byte operand
  MOD = 00 (R/M Table 1) R/M = 110 (Direct Addr)
  bytes 3 and 4 are address; byte 5 immediate data
- Result

```
C6 06 00 01 10
```

- **`MOV WORD PTR [BX+SI],10h`**

  w = 1 because we have byte operand
  MOD = 00 (R/M Table 1) R/M = 000 ([BX+SI])
  bytes 3 and 4 are immediate data
- Result

```
C7 00 10 00
```

MOV imm to reg

- This instruction is optimized as a 4-bit opcode, with register encoded into the instruction

  1011wreg
- Examples

```
MOV bx, 3          1011 w=1 reg=011=BX
10111011 imm       BB 03 00
MOV bh, 3          1011 w=0 reg=111=BH
10110111 imm       B7 03
MOV bl, 3          1011 w=0 reg=011=BL
10110011 imm       B3 03
```

MOV direct mem to/from accumulator

- Another optimized instruction

```
101000dw     addr
```

- Example `mov al, [34F4]`

```
d = 0 because dest is REG
w = 0 because AL is 8 bits
10100000 addr = A0 F4 C4
```

- Example `mov [34F4], ax`

```
d = 1 because dest is REG
w = 1 because AX is 8 bits
10100011 addr = A3 F4 C4
```

Immediate Mode Instructions

- Immediate mode instructions have only one register or memory operand; the other is encoded in the instruction itself
  - The Reg field is used an "opcode extension"
  - The addressing mode byte has to be examined to determine which operation is specified

```
add imm to reg/mem    1000 00sw      mod000r/m
or imm to reg/mem     1000 00sw      mod001r/m
```

- In instructions with immediate operands the "d" bit is interpreted as the "s" bit
- When the s bit is set it means that the single byte operand should be sign-extended to 16 bits

ADD imm to reg/mem

- **add dx, 3 ;Add imm to reg16**

  1000 00sw mod000r/m

  w=1 (DX is 16 bits)

  mod = 11 (use REG table) r/m = 010 =DX

- With s bit set we have

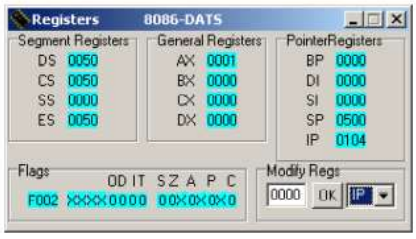
  1000 0011 11 000 010 operand = 83 C2 03

- With s bit clear we have

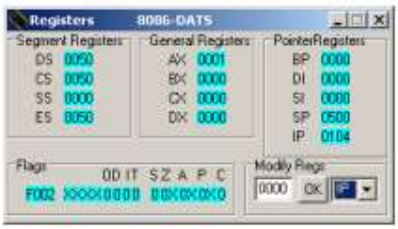
  1000 0001 11 000 010 operand = 81 C2 03 00

اختبار الإجراء البرمجي:

بعد توصيل اللوحة إلى جهاز الحاسب وتشغيل الطاقة يتم تشغيل تطبيق 8086-DATS حيث سيقوم التطبيق باكتشاف اللوحة:



يقوم البرنامج بالتأكد من توصيل اللوحة والبحث عن جميع الاعدادات وفي حال كانت التوصيلات صحيحة ستظهر النافذة التالية:



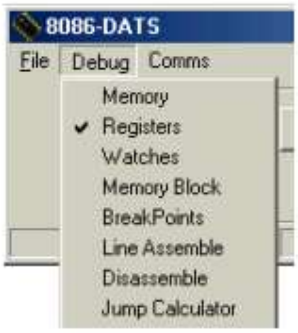
في حال عدم اكتشاف اللوحة يمكن إجراء Reset لها ومن ثم اختيار Detect من القائمة comms. يمكن تفحص الوصلات بأي وقت من خلال Check monitor وفي حال عم التطبيق واللوحة بشكل صحيح سيعطينا الرسالة monitor ok.

يتضمن البرنامج الواجهات التالية:

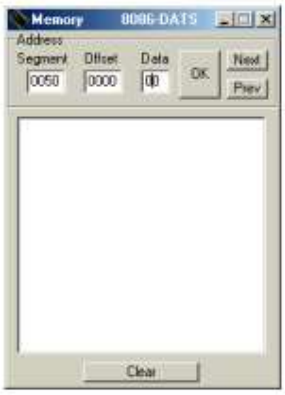
تسمح القائمة EDIT للمستخدم بكتابة الكود البرمجي على ملف نصي يمكن استخدامه مع تطبيقات Cross-Assemblers والتي يتم اعدادها في الملف 86DATS.INI . في حين تسمح القائمة Assemble بفتح نافذة لاختيار ملف assembler المطلوب بناءه . تسمح القائمة Link بعمليات الربط حيث يتم إدخال ملف من نمط object اليها وإخراج ملف من نمط Intel Hex File في حال الرغبة بتحميله للوحة. تسمح اللائحة Download بتحميل ملف Intel Hex File إلى اللوحة البرمجية.

قائمة Debug :

تسمح هذه القائمة بالتحكم بعرض الواجهات التي تعرض حالة الذاكرة والسجلات والوصلات المختلفة:

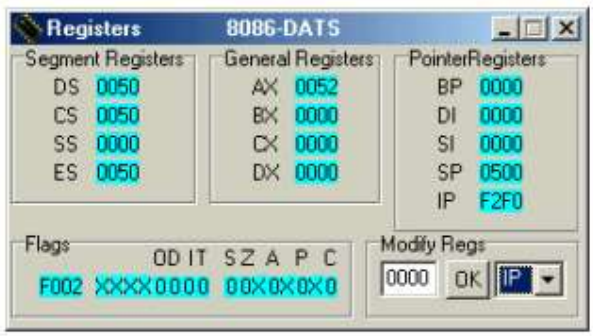


تسمح الواجهة Memory للمستخدم بتفحص أو تحديث محتويات الذاكرة حيث عند النقر على هذا الخيار تظهر النافذة التالية:



تعرض عناوين البيانات في أقسام المقطع والإزاحة 'Segment' and 'Offset' في حين تخزن البيانات في قسم 'Data'. يتم وضع البيانات في هذه الصناديق بالقيمة السداسية عشر ومن ثم نقر ok. يمكن التنقل بين السجلات ببساطة بنقر الأزرار Next, Previous.

بنفس الطريقة تعرض قائمة السجلات حالة السجلات كما في الشكل التالي:

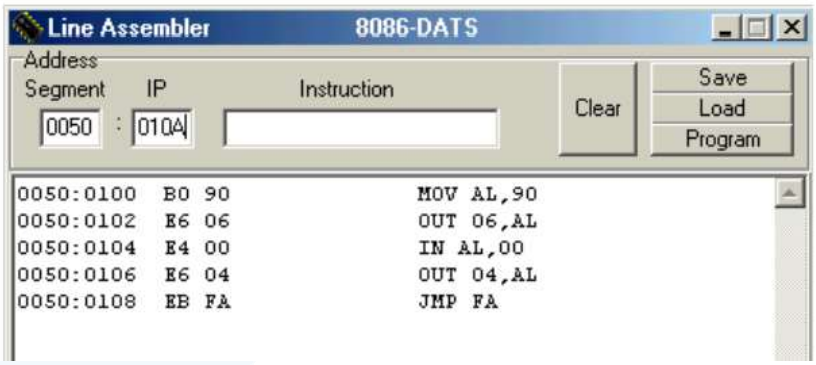


تعرض السجلات في ثلاث اقسام مستقلة, Segment registers, General registers and Pointer registers في حين تعرض الاعلام في القسم السفلي بصورة مستقلة. يمكن التعديل على كل هذه السجلات باستثناء الاعلام من خلال القسم 'Modify Regs' حيث يتم اختيار السجل وإدخال القيمة السداسية عشر ومن ثم نقر ok.

تسمح قائمة Watches بإدخال لائحة من مواقع الذاكرة والتي نريد ادارتها خلال عمليات debug حيث يتم النقر في صناديق segment and offset وإدخال قيمة سداسي عشرية مقبولة واختيار Add وتستمر العملية لكل العناوين المرغوبة.

Line Assembler:

يسمح هذا القسم بكتابة القسم البرمجي للوحة على شكل قيم سطرية, يجب إدخال العنوان المطلوب تخزين البرنامج له في القسم Segment and Instruction Pointer , في حال كان البرنامج المدخل صحيحاً عندها يتم تحديث العرض لعرض عنوان البدء ويتم ترجمة الإجراء إلى النط hex إلى اللوحة.



يجب الانتباه إلى حجم القيم المدخلة للبرنامج وفقاً للشروط المختلفة.