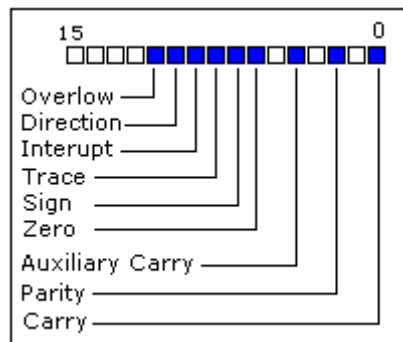


الفريق العربي للهندسة العكسية

دورة ATRE لتعليم الأسس للمبتدئين من الصفر
الدرس الرابع - التعليمات الحسابية والمنطقية
allko

سنتكلم اليوم عن التعليمات الحسابية (Arithmetic) والمنطقية (Logic).
سنقسم الدرس الى أربع مجموعات بناء على المعاملات (operands) لكل تعليمة .

لكن قبل البدء دعنا نفصل الحديث قليلا عن الـ flags .



كما ترى فإن هذا المسجل هو بحجم 16 بت ، كل بت يسمى flag ويأخذ القيمة 0 أو 1 .
كنت أريد ترجمة الفقرة التالية لكن رأيت أنها مكونة من مصطلحات ولا يوجد ما يحتاج الترجمة...

- **Carry Flag (CF)** - this flag is set to **1** when there is an **unsigned overflow**. For example when you add bytes **255 + 1** (result is not in range 0...255). When there is no overflow this flag is set to **0**.
- **Zero Flag (ZF)** - set to **1** when result is **zero**. For none zero result this flag is set to **0**.
- **Sign Flag (SF)** - set to **1** when result is **negative**. When result is **positive** it is set to **0**. Actually this flag take the value of the most significant bit (MSB).
- **Overflow Flag (OF)** - set to **1** when there is a **signed overflow**. For example, when you add bytes **100 + 50** (result is not in range -128...127).

باقي الرايات ليس من المهم معرفتها (على الأقل بالنسبة لمبتدئ)

First group: ADD, SUB, CMP, AND, TEST, OR, XOR

هذه التعليمات لها المعاملات (operands) التالية

REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate

REG: EAX, AX, AL, AH, EBX, etc...

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

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

- بعد هذه التعليمات ، فالنتيجة تخزن دائما في المعامل الأول . لكن تعليمة **CMP** و **TEST** تؤثر على الرايات فقط ولا تخزن أي شيء .
- الرايات التي تتأثر بهذه التعليمات هي **CF, ZF, SF, OF, PF, AF**
- عندما يكون هناك two operands فيجب أن يكونا من نفس الحجم ، مثلا لا يجوز أن تكتب
mov ecx,ax لانهما ليسا من نفس الحجم.
- أيضا مما يجب ذكره هو أنه اذا رايت قيمة ما ك operand (معامل) لإحدى التعليمات ، فهذه القيمة هي بالنظام العشري إذا لم يكن هناك لاحقة ، مثلا mov ebx,30 تعني أننا سننقل القيمة 30 عشري ، بينما mov ebx,30h تعني نقل القيمة 30 بالعكس وهكذا.
- في حالة عنوان الذاكرة فيجب أن تحدد الحجم الذي تريده . تذكر أننا قلنا ان two operands يجب أن يكونا من نفس الحجم ، لذا في حالة عنوان الذاكرة فإننا نبدأ ال operand بالحجم المطلوب أما dword أو word أو حتى byte ، ثم ptr وهي اختصار لـ Pointer أي مؤشر ، ثم عنوان الذاكرة (DS:[BX] مثلا) . أنظر الى هذه التعليمة مثلا :
ADD DWORD PTR DS:[ECX],EAX
أو هذه SUB AX , WORD PTR DS:[ECX]

- 1- تعليمة ADD تقوم بجمع المعامل الثاني الى المعامل الأول .
- 2- تعليمة Sub تطرح المعامل الثاني من الأول .
- 3- تعليمة cmp تطرح المعامل الثاني من الأول (لكن النتيجة يتأثر بها الرايات فقط) .
- 4- دعنا نرى تعليمة AND . هذه تعليمة منطقية . كل عملية منطقية لها ما يسمى بـ Truth Table أو جدول الحقيقة. بالنسبة لتعليمة AND فال Truth Table هو : (هذه العملية هي عملية ضرب منطقي لكن بين الصفر والواحد فقط)

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

- 5- تعليمة TEST تقوم بنفس ما تقوم به AND لكن النتيجة تؤثر على الرايات فقط.

6- تعليمة OR تقوم بعملية منطقية ، ال Truth Table هو كالتالي :

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

7- تعليمة XOR تقوم بعملية منطقية ، ال Truth Table هو كالتالي :
(إذا كان المعاملان متشابهين فالنتيجة 0 ، وإذا كانا مختلفين فالنتيجة 1)

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

Second group: MUL, IMUL, DIV, IDIV

هذه التعليمات لها المعاملات (operands) التالية

REG
memory

- تعليمة MUL مسئولة عن الضرب بدون إشارات ، IMUL للضرب بإشارات . نفس الشيء ينطبق على DIV و IDIV .
- تعليمة MUL و IMUL تؤثران على الرايات التالية فقط : **CF , OF**
فعندما تكون النتيجة أكبر من حجم المعاملات ، عندها تحمل هاتين الرايتين القيمة 1 (set to 1) فيما عدا ذلك فتحملان القيمة 0 (set to 0) . أما بالنسبة لتعليمة DIV و IDIV فالرايات غير معروفة القيمة (undefined) .
- القواعد التي يتم على أساسها الضرب في تعليمتي MUL و IMUL هي :

- A **byte** operand is multiplied by **AL**; the result is left in **AX**.
- A **word** operand is multiplied by **AX**; the result is left in **DX:AX**. DX contains the high-order 16 bits of the product..
- A **doubleword** operand is multiplied by **EAX** and the result is left in **EDX:EAX**. EDX contains the high-order 32 bits of the product.

-2 * -4 = 8 h = 1000 b

```
MOV AL, -2  
MOV BL, -4  
IMUL BL  
; AX = 8 h = 1000 b
```

1700 * 520 = 0D7D20 h = 11010111110100100000 b

```
MOV AX, 1700
MOV BX, 520
MUL BX
; DX = 000D h = 0000000000001101 b
; AX = 7D20 h = 0111110100100000 b
```

**12345678h * 99999999 h = AEC33E18EAD65B8 h =
1010111011000011001111100001100011101011010110010110111000 b**

```
MOV EAX, 12345678h
MOV EBX, 99999999h
MUL EBX
; EDX = 0AEC33E1 = 00001010111011000011001111100001
; EAX = 8EAD65B8 = 10001110101011010110010110111000
```

لاحظ أنه إذا كتبت العدد كما هو فهذا يعني أنه بالعشري ، اما إذا أضفت اللاحقة h فهذا يعني أنه بالعكس ، بالمثل فإن إضافة اللاحقة b تعني أن العدد بالبايناري.

أما بالنسبة لتعليمة IDIV و DIV فلهما الجدول التالي الذي يوضح الحالات الثلاث للقسمة (تبعاً لحجم ما تريد قسمته)

Size	Quotient (الناتج)	Reminder (الباقى)	Dividend (المقسوم عليه)
Byte	AL	AH	AX
Word	AX	DX	DX:AX
dword	EAX	EDX	EDX:EAX

11CCCEEE44BBBAAA h / 33A33A33 = 583EF4DF h = 10110000011111011110100110111111 b

```
MOV EDX, 11CCCEEE h
MOV EAX, 44BBBAAA h
MOV ECX, 33A33A33 h
IDIV ECX
; EAX = 583EF4DF h = 10110000011111011110100110111111 b
; EDX = 15B96C3D h
```

Third group: INC, DEC, NOT, NEG

هذه التعليمات لها المعاملات (operands) التالية

REG
memory

- تعليمة INC, DEC تؤثران على الرايات التالية : ZF, SF, OF, PF, AF
- تعليمة NOT لا تؤثر على أي من الرايات.
- تعليمة NEG تؤثر على الرايات التالية : CF, ZF, SF, OF, PF, AF.

- 1- تعليمة INC تقوم بزيادة المعامل بمقدار واحد
- 2- تعليمة DEC تقوم بإنقاص المعامل بمقدار واحد
- 3- تعليمة NOT تقوم بعكس كل بت من بتات المعامل (ال 0 يصبح 1 والعكس صحيح)
- 4- تعليمة NEG تقوم باستبدال القيمة بال two's complement لها .

Fourth group : MOV instruction

هذه التعليمة لها المعاملات (operands) التالية

```
MOV REG, memory
MOV memory, REG
MOV REG, REG
MOV memory, immediate
MOV REG, immediate
```

أما في حالة كان أحد المعاملات segment register ، فهذا هو المسموح فقط :

```
MOV SREG, memory
MOV memory, SREG
MOV REG, SREG
MOV SREG, REG
```

هذه التعليمة تقوم بنسخ المعامل الثاني (المصدر source) إلى المعامل الأول (الوجهة destination)

والآن الى القليل من الأمثلة... لن نكتب البرامج كاملة فنحن لا نعرف بعد كيف يمكن عمل ذلك ، ما سنقوم به هو كتابة الكود الذي ينفذ ما نريده فقط وليس كل البرنامج.

الأمثلة التالية تشمل جميع الأوامر السابق ذكرها

9553h+35h=9588h

```
mov eax,9553h
add eax,35h
```

111011101b * 100100001b=100001101001111101b

```
mov eax,111011101b
mul 100100001
```

01100001 AND 11011111 = 01000001

```
mov cx,01100001b
and cx,11011111b
```

ملاحظة : يمكنك التعامل مع الأحرف ، أي أن تنقل قيمة الـ ascii لحرف ما الى مسجل ما وذلك بوضع الحرف بين علامتي تنصيص مفردتين هكذا : 'a' كما في المثال التالي :

01100001 AND 11011111 = 01000001

```
mov al, 'a' ; al = 01100001b
and al, 11011111b ; al = 01000001b ('A')
```

203 / 4 = 50.75

MOV AX, 203 ; AX = 00CBh

MOV BL, 4

DIV BL ; AL = 50 (32h), AH = 3

NEG 5

MOV AL, 5 ; AL = 05h = 0000 0101

NEG AL ; AL = 0FBh (-5) = 1111 1011

NEG AL ; AL = 05h (5)

NOT 00011011b

MOV AL, 00011011b

NOT AL ; AL = 11100100b

في هذا المثال ، لنفرض ان القيمة 00011011b موجودة في الذاكرة عند العنوان DS:[2156] عدنها يمكن عمل التالي :

NOT 00011011b

NOT [2156] ; [2156] = 11100100b

الى هنا ينتهي درس اليوم...آمل أن أكون قد وفقت في إيصال المعلومة ببساطة وبدون تعقيد...

Assignment:

Write a code that perform the following arithmetic operation : (with minimum instruction)
[(555AAA h + 10001000100010001 b) * 666777888] / 0E777C h

References:

1- <http://www.emu8086.com/>

2- <http://www.arabteam2000.com/>

Arab Team for Reverse Engineering

αλλκο
August - 2006