

## MIC : Instructions - Résumé

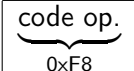
# Codes des registres

- Les registres généraux du processeur sont codés sur 3 bits :

AL, AX, EAX	000	AH, SP, ESP	100
CL, CX, ECX	001	CH, BP, EBP	101
DL, DX, EDX	010	DH, SI, ESI	110
BL, BX, EBX	011	BH, DI, EDI	111

# CLC

- ▶ Mise à zéro du Carry Flag

- ▶ Structure : 

# Le INC r32

- ▶ Incrémentation d'un registre

- ▶ Structure : 

$\underbrace{\text{code op.}}_{0x40 + \text{registre}}$
---

- ▶ Ex :  $\underbrace{INC\ ECX}_{0x41}$

# MOV r32, imm32

- Copie d'un immédiat dans un registre

- Structure : 

<u>code op.</u> 0xB8 + registre	<u>immédiat</u> 4 bytes
------------------------------------	----------------------------

- Ex :  $\underbrace{MOV\ ESI}_{0xBE}, \underbrace{8}_{0x08\ 00\ 00\ 00}$

# ADD EAX, imm32

- ▶ Addition d'un immédiat dans EAX

- ▶ Structure : 

code op. 0x05	immédiat 4 bytes
------------------	---------------------

- ▶ Ex :  $\underbrace{ADD\ EAX,}_{0x05} \underbrace{10}_{0x0A\ 00\ 00\ 00}$

# ADD r32/m32,r32

- ▶ Addition d'un registre vers un autre registre ou une mémoire

- ▶ Structure : 

code op. 0x01	byte ModR/M	..... voir plus bas
------------------	-------------	------------------------

- ▶ Plusieurs variantes, différenciées par le byte ModR/M

## ADD r32/m32,r32 : 1<sup>è</sup> variante

- Addition de registre à registre

- Structure : 

<u>code op.</u> 0x01	<u>byte ModR/M</u> 11 reg1 reg2
-------------------------	------------------------------------

- Ex :  $\underbrace{ADD}_{0x01} \underbrace{EAX, EBX}_{11\ 011\ 000}$  (registres inversés)



## ADD r32/m32,r32 : 2<sup>e</sup> variante

- ▶ Addition de registre à mémoire (adresse dans le 1<sup>er</sup> registre)

- ▶ Structure : 

<u>code op.</u> 0x01	<u>byte ModR/M</u> 00 reg1 reg2
-------------------------	------------------------------------

- ▶ Ex :  $\underbrace{ADD}_{0x01} \underbrace{[EAX], EBX}_{00\ 011\ 000}$  (registres inversés)

## ADD r32/m32,r32 : 3<sup>e</sup> variante

- Addition de registre à mémoire (adresse dans le 1<sup>er</sup> registre + déplacement sur 1 byte)

- Structure : 

<u>code op.</u> 0x01	<u>byte ModR/M</u> 01 reg1 reg2	<u>déplacement</u> 1 byte
-------------------------	------------------------------------	------------------------------

- Ex :  $\underbrace{ADD}_{0x01} \underbrace{[EAX + 10], EBX}_{01\ 011\ 000\ 00001010}$  (registres inversés)

## ADD r32/m32,r32 : 4<sup>e</sup> variante

- Addition de registre à mémoire (adresse dans le 1<sup>er</sup> registre + déplacement sur 4 bytes)

- Structure : 

<u>code op.</u> 0x01	<u>byte ModR/M</u> 10 reg1 reg2	<u>déplacement</u> 4 bytes
-------------------------	------------------------------------	-------------------------------

- Ex :  $\underbrace{ADD}_{0x01} \underbrace{[EAX + 512], EBX}_{10\ 011\ 000\ 0x00\ 02\ 00\ 00}$  (registres inversés)

## ADD r32/m32,r32 : 5<sup>e</sup> variante

- ▶ Addition de registre à mémoire (adresse fixe sur 4 bytes)

- ▶ Structure : 

<u>code op.</u> 0x01	<u>byte ModR/M</u> 00 reg1 EBP	<u>adresse</u> 4 bytes
-------------------------	-----------------------------------	---------------------------

- ▶ Ex :  $\underbrace{ADD}_{0x01} \quad \underbrace{[512], EBX}_{00\ 011\ 101\ 0x00\ 02\ 00\ 00}$  (inversion)

## ADD r32/m32,r32 : 6<sup>e</sup> variante

- Addition de registre à mémoire (avec index et base)

- Structure : 

<u>code op.</u> 0x01	<u>byte ModR/M</u> 00 reg1 ESP	<u>byte SIB</u> mult index base
-------------------------	-----------------------------------	------------------------------------

- Ex :  $\underbrace{ADD}_{0x01} \underbrace{[EAX * 2 + ECX], EBX}_{00\ 011\ 100\ 01\ 000\ 001}$  (inversion)

- Mult : 00=1x, 01=2x, 10=4x, 11=8x

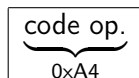
## ADD r32/m32,r32 : résumé

Variante	ModR/M	Restriction
ADD EAX, EBX	11	Aucune
ADD [EAX],EBX	00	Pas [ESP] ni [EBP]
ADD [EAX+10],EBX	01	Pas [ESP]
ADD [EAX+512],EBX	10	Pas [ESP]
ADD [512],EBX	00	[EBP]
ADD [EAX*2+ECX], EBX	00	[ESP]

# Le MOVSB

- Copie en mémoire un byte de [ESI] vers [EDI] et incrémente ESI et EDI

- Structure :



# Le INT

- ▶ Provoque une interruption

- ▶ Structure : 

<u>code op.</u> 0xCD	<u>n°d'interruption</u> 1 byte
-------------------------	-----------------------------------

- ▶ Ex :  $\underbrace{INT}_{0xCD} \underbrace{0x80}_{80}$



# Les préfixes

- ▶ Petit code se plaçant **avant** le code opératoire
- ▶ Exemples :
  - ▶ Préfixe de segmentation
  - ▶ Préfixe de répétition
  - ▶ Préfixe de taille

# Préfixes de segmentation

- ▶ Servent à spécifier le segment dans des adresses de type [ES :EAX]
- ▶ 0x2E pour CS, 0x3E pour DS, 0x26 pour ES, 0x64 pour FS, 0x65 pour GS et 0x36 pour SS
- ▶ Exemple :
  - ▶ ADD [EAX],EBX se codait 0x01 18
  - ▶ ADD [ES :EAX],EBX se code 0x26 01 18

# Préfixe de répétition REP

- ▶ Ex : REP MOVSB répète ECX fois MOVSB

- ▶ Structure : 

REP 0xF3	code op. 0xA4
-------------	------------------

## Préfixe de taille 0x66

- ▶ Le processeur 80386 encode en général de la même façon les instructions 16 bits et 32 bits :
  - ▶ INC ECX et INC CX sont tous deux codés 0x41
  - ▶ ADD EAX,EBX et ADD AX,BX sont tous deux codés 0x01 D8
- ▶ Comment le processeur fait-il la différence ?
- ▶ Le 80386 travaille par défaut en 32 bits donc 0x41 signifie par défaut INC ECX
- ▶ Pour utiliser un registre de 16 bits, on utilise un **préfixe de taille** (ici 0x66)
  - ▶ INC CX se code 0x66 41
  - ▶ ADD [AX],BX se code : 0x66 01 D8

# Préfixe de taille 0x66

test-instructions - KDbg

Fichier Affichage Exécution Points d'arrêt Configuration Aide

test-instructions.asm

```
+ 1 ; l_testEnonce.asm
+ 2
+ 3 global main
+ 4
+ 5 section .text
+ 6 main:
+ 7     inc ecx
+ 8     inc cx
+ 9     add eax, ebx
+10     add ax, bx
+11
+12     ; fin
+13     mov eax, 1
+14     mov ebx, 0
+15     int 0x80
+16
+17
+18
```

main

Adresse								
0x8048060 <main>	0x41	0x66	0x41	0x01	0xd8	0x66	0x01	0xd8
0x8048068 <main>...	0xb8	0x01	0x00	0x00	0x00	0xbb	0x00	0x00
0x8048070 <main>...	0x00	0x00	0xcd	0x80	0x1c	0x00	0x00	0x00
0x8048078	0x02	0x00	0x00	0x00	0x00	0x00	0x04	0x00
0x8048080	0x00	0x00	0x00	0x00	0x60	0x80	0x04	0x08
0x8048088	0x14	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x8048090	0x00	0x00	0x00	0x00	0x0e	0x00	0x00	0x00
0x8048098	0x03	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x80480a0	0x00	0x00	0x00	0x00	0x00	0x00	0x40	0x00
0x80480a8	0x00	0x00	0x02	0x00	0x00	0x00	0x00	0x00
0x80480b0	0x04	0x01	0x60	0x80	0x04	0x08	0x74	0x80
0x80480b8	0x04	0x08	0x00	0x00	0x00	0x00	0x74	0x65
0x80480c0	0x73	0x74	0x2d	0x69	0x6e	0x73	0x74	0x72
0x80480c8	0x75	0x63	0x74	0x69	0x6f	0x6e	0x73	0x2e
0x80480d0	0x61	0x73	0x6d	0x00	0x4e	0x41	0x53	0x4d
0x80480d8	0x20	0x32	0x2e	0x30	0x37	0x00	0x01	0x80

actif