

5. Instructions de base du Pentium

Notation et abréviations

Dans les tableaux d'indicateurs :

Indicateurs	O	D	I	T	S	Z	A	P	C
	?				?	?	*	?	*

* indique un bit positionné, i.e. mis à 0 ou à 1 par l'instruction

? indique un bit indéfini suite à l'instruction

indique que le bit n'est pas affecté par l'instruction

Dans les opérandes :

reg8 désigne un des registre de 8 bits : al, bl, cl ou dl
 reg16 désigne un registre de 16 bits : ax, bx. cx. dx, di, si, bp, ou sp
 reg32 désigne un registre de 32 bits : eax, ebx, ecx, edx, edi, esi, ebp ou esp.
 accum désigne l'accumulateur qui peut être al, ax ou eax selon la taille de l'opération.

mem16 désigne un emplacement mémoire de 16 bits
 mem32 désigne un emplacement mémoire de 32 bits
 etc.

immed8 désigne une valeur immédiate de 8 bits
 immed16 désigne une valeur immédiate de 16 bits
 immed32 désigne une valeur immédiate de 32 bits

Dans le temps d'exécution (cycles)

mp mode protégé
 mv mode virtuel

AAA ASCII Adjust After Addition

Ajuste la somme de deux digits BCD non compactés pour créer un résultat BCD non compacté. Après une instruction ADD, si la somme dépasse 9, AAA ajoute 6 au résultat, sinon le laisse inchangé.

Si cette addition de 6 cause une retenue, AH est incrémenté de 1 et les indicateurs CF et AF sont mis à 1, sinon, CF et AF sont mis à 0 et AH est inchangé. Dans les deux cas, les bits 4 à 7 de AL sont mis à 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				?	?	*	?	*

Syntaxe

aaa

cycles

3

Exemple

```

mov al, 07
mov bl, 05
add al, bl           ; AL = 0x0C
aaa                  ; AX = 0x0102 = 12 en BCD non compacté

```

AAD ASCII Adjust Before Division

Convertit en binaire deux digits BCD non compactés dans AH:AL et place le résultat dans AL, i.e. remplace AL par $AL + (10 * AH)$, puis met AH à 0. Cette instruction précède habituellement une instruction DIV. Après cette division, le quotient sera dans AH et le reste dans AL.

La version généralisée de cette instruction permet l'ajustement de deux digits non compactés dans n'importe quelle base. Le paramètre immédiat de 8 bits contient la base en question, 0x08, 0x10, etc. Toutefois, cette version n'a pas de mnémonique et doit être codée à la main : D5 immed8.

Indicateurs

O	D	I	T	S	Z	A	P	C
---	---	---	---	---	---	---	---	---

?	*	*	?	*	?
---	---	---	---	---	---

Syntaxe

aad

Cycles

10

Exemple

```

mov ax,0x0704      ; 74 en BCD
aad                ; AL = 0x4A = 74 en binaire
mov bh,09
div bh              ; AH = 08 , AL = 02

```

AAM ASCII Adjust After Multiply

Effectue la conversion binaire-BCD non compacté du nombre situé dans AX. Tout se passe comme si AAM divisait AX par 0x0A et plaçait le quotient dans AH et le reste dans AL. Cette instruction est utile après une instruction MUL qui a effectué la multiplication binaire de deux valeurs BCD non compacté.

La version généralisée de cette instruction peut convertir AX en n'importe quelle base. Un paramètre immédiat de 8 bits contient la base désirée. Cette version doit être codée à la main : D4 immed8.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				*	*	?	*	?

Syntaxe

aam

cycles

18

Exemple

```

mov al, 05
mov dl, 07
mul dl              ; AX = 0x0023
aam                 ; AX = 0x0305 = 35 en BCD

```

AAS ASCII Adjust After Subtraction

Ajuste la différence de deux digits en BCD non compacté pour créer un résultat BCD non compacté. Après une instruction SUB, si la différence dépasse 9, ou si AF = 1, AAS soustrait 6 au résultat, sinon le laisse inchangé.

Si cette soustraction de 6 nécessite un emprunt, le registre AH est décrémenté de 1 et CF et AF sont mis à 1, Sinon, CF et AF sont mis à 0 et AH est inchangé. Dans les deux cas, les bits 4 à 7 de AL sont mis à 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				?	?	*	?	*

Syntaxe

aas

Cycles

3

Exemple

```

mov al, 0x0105      ; 15 en BCD
mov bl, 06
sub al, bl           ; AL = 0x00FF
aas                  ; AX = 0x0009

```

ADC Add with Carry

Additionne l'opérande source, l'opérande destination et CF. Le résultat est inscrit dans l'opérande destination. Cette instruction est utilisée pour additionner les portions les plus significatives de nombres s'étendent sur plusieurs mots ou plusieurs doubles mots.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

```

adc reg, reg
adc mem, reg
adc reg, mem

```

Cycles

1
3
2

adc reg, immedi	1
adc mem, immedi	3

Exemple

Soit deux nombres de 64 bits débutant aux adresses A et B. On veut les additionner et placer la réponse dans A.

```

mov     eax, A      ; double mot de poids faible
add     eax, B      ; somme
mov     A, eax      ; écriture du résultat
mov     eax, A+4    ; double mot de poids fort
adc     eax, B+4    ; au cas où il y aurait une retenue de A + B
mov     A+4, eax    ; écriture du résultat

```

ADD**Add**

Additionne l'opérande source et l'opérande destination et place la somme dans l'opérande destination.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe**Cycles**

add reg, reg	1
add mem, reg	3
add reg, mem	2
add reg, immedi	1
add mem, immedi	3

AND**Logical AND**

Effectue le ET logique bit par bit entre les opérandes source et destination, et enregistre le résultat dans l'opérande destination. Chaque bit du résultat est 1 si les bits correspondants des opérandes sont tous deux 1. Sinon, le bit du résultat est 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Syntaxe

and reg, reg
 and mem. reg
 and reg, mem
 and reg, immedi
 and mem, immedi

Cycles

1
 3
 2
 1
 3

Exemple

mov ax, 0x3435 ; 45 en ASCII
 and ax, 0x0F0F ; conversion ASCII-BCD
 ; AX = 0x0405 = 45 en BCD non compacté

BOUND**Check Array Bounds**

Détermine si le premier opérande (indice dans un tableau) est à l'intérieur des limites du tableau spécifiées par le second opérande. L'indice dans le tableau est un entier signé dans un registre. Si le registre est de 16 bits, le second opérande est un emplacement mémoire de 32 bits contenant dans son mot de poids faible la limite inférieure et dans son mot de poids fort la limite supérieure de l'indice. Si le registre est de 32 bits, le second opérande est un emplacement mémoire de 64 bits avec les deux limites exprimées sur 32 bits.

Si la valeur du registre est inférieure à la limite du bas ou supérieure à la limite du haut, il y a déclenchement d'une exception #BR (no 5 dans IVT).

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

bound reg16, mem32
 bound reg32, mem64

Cycles

8 - 32
 8 - 32

Exemple

Supposons que bx est un index dans un tableau de 1000 éléments numérotés de 1 à 1000. On place 0x000103E8 à l'adresse Limit. Si bx n'est pas entre 1 et 1000 (0x03E8) inclusivement, il y aura interruption.

bound bx, Limit

BSF/BSR**Bit Scan**

Balaye un opérande mot ou double mot pour trouver le premier bit dont la valeur soit 1. Si un bit 1 est trouvé, ZF est mis à 0 et le registre destination contient l'indice du premier bit 1 trouvé. Si aucun bit 1 n'est trouvé, ZF est mis à 1 et la destination est indéfinie. BSF (Bit Scan Forward) balaye du bit 0 au bit le plus significatif. BSR (Bit Scan Reverse) balaye du bit le plus significatif au bit 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
					*			

Syntaxe

```
bsf  reg16, reg16
bsf  reg32, reg32
bsf  reg16, mem16
bsf  reg32, mem32
bsr  reg16, reg16
bsr  reg32, reg32
bsr  reg16, mem16
bsr  reg32, mem32
```

Cycles

```
6-34
6-42
6-35
6-43
7-39
7-71
7-40
7-72
```

BSWAP**Byte Swap**

Échange le premier octet avec le quatrième et le second avec le troisième dans un registre de 32 bits. Cette instruction ne change aucun bit à l'intérieur des octets et sert à passer du

mode little endian au mode big endian et vice-versa. Pour échanger deux octets dans un mot (registre de 16 bits), utiliser plutôt l'instruction XCHG.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

bswap

Cycles

1

Exemple

mov eax, 0x12345678

bswap eax ; EAX = 0x78 563412

BT/BTC/BTR/BTS Bit Tests

Copie la valeur d'un bit d'un champ de bits vers CF où il pourra être testé au moyen d'une instruction subséquente jc ou jnc. L'opérande source spécifie la position du bit dans le champ. L'opérande destination spécifie le champ de bits. Il peut s'agir d'un registre ou d'un emplacement mémoire. S'il s'agit d'un registre, l'instruction utilise le modulo 16 ou 32 selon la taille du registre. S'il s'agit d'un emplacement mémoire, celle-ci désigne l'adresse du début du champ de bits. L'opérande source spécifie alors un offset signé entre 2^{-31} et $2^{31}-1$ s'il est un registre, entre 0 à 31 s'il est une valeur immédiate.

BT copie simplement le bit dans CF. BTC copie le bit dans CF puis le complémente dans la destination. BTR copie le bit dans CF puis le met à 0 dans la destination. BTS copie le bit dans CF puis le met à 1 dans la destination.

Indicateurs

O	D	I	T	S	Z	A	P	C
								*

Syntaxe

bt reg16, imm8/32

bt mem16, imm8/32

bt reg16, reg16/32

bt mem16, reg16/32

btc reg16, imm8/32

Cycles

4

4

4

9

7

btc	mem16, immed8/32	8
btc	reg16, reg16/32	7
btc	mem16, reg16/32	13

idem pour btr et bts

Exemple

```
mov    ax, 0x2467
bt     ax, 3           ; CF = 0
```

CALL Call Procedure

Appelle une procédure. L'instruction empile l'adresse de l'instruction suivante, puis saute à l'adresse spécifiée par l'opérande.

Lors d'un appel «near», l'offset IP est poussé sur la pile et l'offset de la destination est chargé dans IP.

Lors d'un appel «far», le segment CS est empilé et le segment de la destination est chargé dans CS. Ensuite, l'offset IP est empilé et le nouvel offset est chargé dans IP. Une instruction RET subséquente dépilera l'adresse de retour, de sorte que l'exécution continuera avec celle qui suit l'appel.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

Cycles

call	étiquette	1
call	éloigné	4
call	reg	2
call	mem32	2
call	r/m32	5

CBW Convert Byte to Word

Convertit un octet signé dans AL en un mot signé dans AX en répliquant le bit de signe de AL dans tous les bits de AH.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

cbw

Cycles

3

Exemple

```

mov    al, 0x92
cbw                      ; AL = 0xFF92
mov    al, 0x84
cbw                      ; AL = 0x0084

```

CDQ Convert Double to Quad

Convertit un double mot signé dans EAX en un quadword signé dans la paire EDX:EAX en répliquant le bit de signe de EAX dans tous les bits de EDX.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

cdq

Cycles

2

Exemple

```

mov    eax, 0x92070A25
cdq                      ; EDX = 0xFFFFFFFF

```

CLC Clear Carry Flag

Met le bit CF à 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
								0

Syntaxe

clc

Cycles

2

CLD Clear Direction Flag

Met le bit DF à 0. Toutes les instructions de chaîne subséquentes procéderont vers le haut (des adresses basses aux adresses hautes) en incrémentant les registres d'index appropriés.

Indicateurs

O	D	I	T	S	Z	A	P	C
								0

Syntaxe

cld

Cycles

2

CLI Clear Interrupt Flag

Met IF à 0. Quand IF est 0, les interruptions masquables ne sont pas reconnues tant que IF n'est pas remis à 1 au moyen de l'instruction STI. En mode protégé, CLI ne met IF à 0 que si le niveau de privilège de la tâche courante est inférieur ou égal à la valeur des bits IOPL. Sinon, une erreur de protection générale se produit.

Indicateurs

O	D	I	T	S	Z	A	P	C
								0

Syntaxe

cli

Cycles

7

CMC Complement Carry Flag

Complémente l'indicateur de retenue CF.

Indicateurs

O	D	I	T	S	Z	A	P	C
								*

Syntaxe

cmc

Cycles

2

CMP Compare Two Operands

Compare deux opérandes en prévision d'une instruction subséquente de saut conditionnel ($j_{\text{condition}}$) ou de *set* conditionnel ($\text{set}_{\text{condition}}$). Cette opération est effectuée en soustrayant l'opérande source de l'opérande destination et en positionnant les indicateurs en fonction du résultat.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

cmp reg, reg
 cmp mem, reg
 cmp reg, mem
 cmp reg, immediat
 cmp mem, immediat

Cycles

1
 2
 2
 1
 2

CMPS/CMPSB/CPMSW/CMPSD Compare String

Compare l'octet, le mot ou le double mot source dont l'adresse est dans ESI (DS:SI) avec l'octet, le mot ou le double mot destination dont l'adresse est dans EDI (DS:DI). La comparaison est effectuée en soustrayant l'élément destination de l'élément source et les indicateurs sont mis à jour selon le résultat (qui n'est pas enregistré). Remarquer que c'est le sens inverse de l'instruction CMP. Après la comparaison, EDI et ESI sont ajustés selon la taille des opérandes et l'état de DF. Ils sont incrémentés si DF avait préalablement été mis à zéro avec CLD ou décrémentés si DF avait été mis à 1 avec STD.

Avec la forme CMPS de l'instruction, des opérandes doivent être fournis pour indiquer la taille des éléments de données à traiter. Une dérogation de segment peut être donnée pour la source (mais pas pour la destination). Lorsqu'on utilise l'une des formes CMPSB (octets), CMPSW (mots) ou CMPSD (doubles mots), c'est le choix d'instruction qui détermine la taille des éléments à traiter.

CMPS et ses variantes sont souvent utilisées avec un préfixe de répétition. On utilise REPNE (ou REPNZ) pour trouver le premier accord entre deux chaînes et REPE (ou REPZ) pour trouver le premier désaccord. Avant l'instruction, ECX doit contenir le nombre maximum d'itérations. Après REPNE CMPS, ZF est 0 si aucun accord n'a été trouvé. Après REPE CMPS, ZF est 1 si aucun désaccord n'a été trouvé. ZF est mis à 1 ou 0 selon le résultat de la dernière comparaison, et non selon la valeur de ECX même si ECX décroît jusqu'à 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

cmps	src, dest
cmpsb	[[src]], [[dest]]
cmpsw	[[src]], [[dest]]
cmpsd	[[src]], [[dest]]

Cycles

5
5
5
5

CMPXCHG Compare and Exchange

Compare l'opérande destination à AL, AX ou EAX. S'il est égal, l'opérande source est copié dans la destination. Sinon, la destination est copiée dans AL, AX ou EAX. L'instruction positionne les indicateurs selon le résultat de la comparaison.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

```
cmpxchg    mem, reg
cmpxchg    reg, reg
```

Cycles

6
6

CWD Convert Word to Double

Convertit le mot signé dans AX en un double mot signé dans la paire DX:AX en répliquant le bit de signe de AX dans tous les bits de DX.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

```
cwd
```

Cycles

2

Exemple

```
mov    ax, 0xAF04
cwd                      ; DX = 0xFFFF
```

CWDE Convert Word to Extended Double

Convertit un mot signé dans AX en un double mot signé dans EAX en répliquant le bit de signe de AX dans les 16 bits de poids fort de EAX.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

cwde

Cycles

3

Exemple

mov ax, 0x9ABC

cwde ; EAX = 0xFFFF9ABC

DAA Decimal Adjust After Addition

Ajuste le résultat d'une addition en un nombre BCD compacté inférieur à 100 (décimal). L'instruction d'addition précédente devrait placer sa somme de 8 bits (binaire) dans AL. DAA convertit cette somme binaire en format BCD compacté avec le digit décimal le moins significatif dans les 4 bits de poids faible et le digit le plus significatif dans les 4 bits de poids fort. Si la somme est supérieure à 0x99 après l'ajustement, CF et AF sont mis à 1. Sinon, CF et AF sont mis à 0.

Si demi-octet de poids faible de AL est supérieur à 9 ou que AF = 1, alors DAA ajoute 6 à AL et AF = 1 et CF = (CF | retenue de AL+6). Sinon AF = 0. Ensuite, si le demi-octet de poids fort est supérieur à 9, alors DAA ajoute 0x60 à AL et CF = 1. Sinon, CF = 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

daa

Cycles

3

Exemple

mov al, 0x35

; 35 en BCD compacté

mov bl, 0x27

; 27 en BCD compacté

add al, bl

; AL = 0x5C

daa

; AL = 0x62 = 62 en BCD compacté

DAS Decimal Adjust After Subtraction

Ajuste le résultat d'une soustraction en un nombre BCD compacté inférieur à 100 (décimal). L'instruction de soustraction précédente devrait avoir placé son résultat de 8 bits dans AL. DAS convertit cette différence binaire en format BCD compacté avec le digit décimal le moins significatif dans les 4 bits de poids faible et le digit le plus significatif dans les 4 bits du haut. Si la différence est supérieure à 99h après l'ajustement, CF et AF sont mis à 1. Sinon, CF et AF sont mis à 0.

Si le demi-octet de poids faible de AL est supérieur à 9 ou si AF = 1, alors DAS retranche 6 à AL et AF = 1 et CF = (CF | retenue de AL - 6). Sinon, AF = 0. Si le demi-octet de poids fort est supérieur à 9 ou si CF = 1, alors on retranche 0x60 à AL et CF = 1, sinon CF = 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				*	*	*	*	*

Syntaxe

das

Cycles

3

Exemple

```

mov  al, 62          ; 62 en BCD compacté
mov  bl, 35          ; 35 en BCD compacté
sub                      ; AL = 0x2D
das                      ; AL = 0x27 = 27 en BCD compacté

```

DEC Decrement

Retranche 1 de l'opérande destination. Comme l'opérande est traité comme un entier non signé, l'instruction DEC n'affecte pas CF. Pour détecter un effet sur CF, utiliser plutôt l'instruction SUB.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

dec	reg8	Cycles
dec	mem	1
dec	reg16	3
dec	reg32	1
dec	reg32	1

DIV**Unsigned Divide**

Effectue la division non signée de deux nombres. Si l'opérande source (le diviseur) est de 8 bits, l'opérande destination implicite (le dividende) est la paire de registres AH:AL. Le quotient va dans AL et le reste dans AH. Si la source est de 16 bits, le dividende implicite est la paire de registres DX:AX. Le quotient va dans AX et le reste dans DX. Si la source est de 32 bits, le dividende implicite est la paire de registres EDX:EAX; le quotient va dans EAX et le reste dans EDX.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				?	?	?	?	?

Syntaxe

div	reg8	Cycles
div	reg16	17
div	reg32	25
div	mem	41
div	mem	17 - 25 - 41

Exemple

```

xor    dx, dx
mov    eax, 0x9564    ; EAX = 38244 = dividende
mov    bx, 0x64        ; BX = 100 = diviseur
div    bx              ; DX = 0x002C (44), AX = 0x017E (382)

```

ENTER**Make Stack Frame**

Crée un bloc de pile pour une procédure qui reçoit des paramètres passés par la pile. Quand *immed16* est 0, ENTER est équivalent à push bp, suivi de mov bp, sp. Le premier opérande de l'instruction ENTER spécifie le nombre d'octets à réserver pour les variables locales. Le second opérande spécifie le niveau d'imbrication de la procédure. Le niveau d'imbrication devrait être 0 pour les langages qui ne supportent pas l'accès aux variables locales des procédures de plus haut niveau (C, Basic et FORTRAN). Voir l'instruction complémentaire LEAVE pour la façon de quitter la procédure.

Cette instruction n'est pas conseillée car elle est plus lente que les instructions individuelles qu'elle remplace.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

enter immed16,0
 enter immed16, 1
 enter immed16, immed8

Cycles

11
 15
 15+2imm8

HLT**Halt**

Arrête le CPU jusqu'à ce qu'une interruption redémarre l'exécution à l'instruction suivant HLT. En mode protégé, c'est une instruction privilégiée.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

hlt

Cycles

12

IDIV Signed Divide

Effectue la division signée de deux nombres. Si l'opérande source (le diviseur) est de 8 bits, l'opérande destination implicite (le dividende) est la paire de registres AH:AL. Le quotient va dans AL et le reste dans AH. Si la source est de 16 bits, l'opérande destination implicite est la paire de registres DX:AX. Le quotient va dans AX et le reste dans DX. Si la source est de 32 bits, le dividende est EDX:EAX; le quotient va dans EAX et le reste dans EDX.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				?	?	?	?	?

Syntaxe

idiv	reg8	22
idiv	reg16	30
idiv	reg32	46
idiv	mem	22 - 30 - 46

Exemple

```

mov    dx, 0xFFFF
mov    ax, 0x9564    ; DX:AX = -27292
mov    bx, 0x3E8     ; BX = 100
idiv    bx           ; DX = 0xFFA4 (-92), AX = 0xFE0 (-272)

```

IMUL Signed Multiply

Effectue la multiplication signée de deux nombres. Il y a trois syntaxes possibles : un opérande, deux opérandes, et trois opérandes.

S'il n'y a qu'un opérande de 8 bits, la destination implicite est AL et le produit va dans AX. S'il n'y a qu'un opérande de 16 bits, la destination implicite est AX et le produit va dans la paire de registres DX:AX. S'il n'y a qu'un opérande de 32 bits, le produit va dans la paire EDX:EAX. Les indicateurs CF et OF sont mis à 1 si le produit est étendu en signe dans AH pour des opérandes de 8 bits, DX pour des opérandes de 16 bits, ou dans EDX pour des opérandes de 32 bits.

Avec la forme à deux opérandes, un registre de 16 ou de 32 bits contient l'un des facteur et sert de destination pour le résultat; l'autre facteur peut être soit un registre, soit une donnée immédiate de même taille que le registre destination.

Avec la forme à trois opérandes, le premier est un registre de 16 ou de 32 bits dans lequel le résultat sera enregistré, le deuxième est un registre de même taille ou un opérande mémoire contenant le premier facteur, et le troisième est une valeur immédiate représentant l'autre facteur. Dans les deux cas, OF et CF sont mis à 1 si le résultat est trop grand pour entrer dans le registre destination. Comme la moitié de poids faible du produit est la même pour la multiplication signée et la multiplication non signée, ces syntaxes peuvent être utilisées pour des nombres signés ou non signés. Avec la forme à trois opérandes, on peut spécifier les opérandes source et destination. La source peut être n'importe quel opérande mémoire ou registre général de 16 ou de 32 bits. La destination peut être n'importe quel registre général de même taille. Les indicateurs OF et CF sont mis à 1 si le produit n'entre pas dans la destination.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				?	?	?	?	*

Syntaxe

Cycles

imul	reg	11
imul	mem	11
imul	reg16, immed	10
imul	reg32, immed	
imul	reg16, reg16, immed	10
imul	reg32, reg32, immed	10
imul	reg16, mem16, immed	
imul	reg32, mem32, immed	
imul	reg16, reg16	10
imul	reg32, reg32	
imul	reg16, mem16	
imul	reg32, mem32	

Exemple

```

mov    eax, 0x9FFFFFFF
mov    ebx, 3
imul   ebx    ; EDX = 0xFFFFFFFF, EAX = 0xDFFFFFFD

```

IN Input from Port

Copie un octet, un mot ou un double mot d'un port vers AL, AX ou EAX respectivement. L'adresse du port est spécifiée dans l'opérande source, qui peut être DX, ou une valeur immédiate de 8 bits si le numéro de port est inférieur à 255. En mode protégé, une erreur de protection générale se produit si IN est utilisée alors que le niveau de privilège courant est supérieur à la valeur de l'indicateur IOPL.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

in accum, immed
in accum, dx

Cycles

7, mp=4, 21, mv=19
7, mp=4, 21, mv=19

INC Increment

Ajoute 1 à l'opérande destination. L'opérande est traité comme un nombre non signé, de sorte que l'instruction INC n'affecte pas la retenue CF. Si on a besoin de détecter une retenue, on utilise plutôt l'instruction ADD.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

inc reg
inc mem

Cycles

1
3

INS/INSB/INSW/INSD Input from Port to String

Copie les données provenant d'un port spécifié dans l'opérande source vers une chaîne destination spécifiée dans l'opérande destination. L'adresse de la chaîne doit être dans EDI (ES:DI) même si un opérande est fourni. Le port d'entrée est spécifié dans DX. À chaque itération, EDI est ajusté selon la taille de l'opérande et l'état de DF. EDI est incrémenté si DF avait été mis à zéro avec CLD et décrémenté si DF avait été mis à 1 avec STD.

Avec la forme INS de l'instruction, un opérande destination doit être fourni pour indiquer la taille des éléments de données à transférer. La dérogation de segment n'est pas permise. Lorsque l'une des formes INSB (octets) INSW (mots) ou INSD (doubles mots) est utilisée, c'est le choix d'instruction qui détermine la taille des éléments de données à transférer.

INS et ses variations sont normalement utilisées avec le préfixe REP. Avant l'exécution de REP, ECX doit contenir le nombre d'itérations désiré. En mode protégé, une erreur générale de protection se produit si le niveau de privilège courant est supérieur à la valeur de l'indicateur IOPL.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

ins dest, dx
 insb
 insw
 insd

Cycles

9, mp=6, 24, mv=22
 9, mp=6, 24, mv=22
 9, mp=6, 24, mv=22
 9, mp=6, 24, mv=22

INT Interrupt

Déclenche un appel à un sous-programme de traitement d'interruption logicielle. Un opérande immédiat de 8 bits (0 à 255) spécifie la procédure d'interruption à appeler. L'appel est effectué en indexant le numéro d'interruption dans la Table de vecteurs d'interruption (IVT) débutant à l'offset 0 du segment 0. En mode réel, la IVT contient des pointeurs de 4 octets (CS et IP) vers les procédures d'interruption. En mode privilégié, la IVT contient des pointeurs de 8 octets.

Quand une interruption est appelée en mode réel, les indicateurs, puis CS et IP sont empilés, puis les indicateurs de trappe (TF) et d'interruption (IF) sont mis à 0. STI pourra être utilisé ultérieurement pour restaurer les interruptions. Pour le retour d'une interruption, on utilise l'instruction IRET.

Indicateurs

O	D	I	T	S	Z	A	P	C
		*	*					

Syntaxe

int immed8

Cycles

16, mp=31, 48

INTO Interrupt on Overflow

Déclenche un appel au sous-programme de traitement de l'interruption logicielle 4 si OF est 1. Le comportement par défaut de MS-DOS pour l'interruption 4 est de retourner sans rien faire. Pour que INTO fonctionne correctement, il faut donc définir une procédure d'interruption pour l'interruption 4.

Indicateurs

O	D	I	T	S	Z	A	P	C
		*	*					

Syntaxe

into

Cycles

13, mp=27, 44

IRET/IRETD**Interrupt Return**

Retourne le contrôle d'une procédure d'interruption au code interrompu. En mode réel, l'instruction IRET dépile IP, CS et les indicateurs (dans cet ordre) et poursuit l'exécution. Sur le Pentium, on doit utiliser IRETD, pour dépiler un pointeur d'instruction de 32 bits quand on retourne d'une interruption appelée par un segment de 32 bits. Le suffixe F empêche que du code d'épilogue soit généré quand on termine un bloc PROC. On l'utilise pour terminer des procédures de traitement d'interruption.

Indicateurs

O	D	I	T	S	Z	A	P	C
*	*	*	*	*	*	*	*	*

Syntaxe

```
iret
iretd
iretf
iretdf
```

Cycles

8, 10, mp=27
10, mp=27

Jcondition**Jump Conditionally**

Continue l'exécution à l'étiquette spécifiée si la condition donnée par les indicateurs est vraie. Si la condition est fausse, il n'y a pas de saut et l'exécution continue à l'instruction suivante. Les sauts «short» sont de -128 à 127 octets par rapport au début de l'instruction suivante, et les sauts «near» de -32768 à 32767. L'assembleur génère le plus petit saut possible à moins qu'on spécifie explicitement la taille du saut.

Quand le processeur est en mode mémoire plate, les sauts courts sont de -128 à +127 octets, et les sauts «near» sont de -2 à +2 gigaoctets. Il n'y a pas de sauts «far».

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```
jcondition etiquette
```

Cycles

1

JCXZ/JECXZ Jump if CX is Zero

JCXZ continue l'exécution du programme à l'étiquette spécifiée si CX = 0. On utilise JECXZ pour effectuer un saut si ECX = 0. Si CX ou ECX n'est pas 0, l'exécution continue à l'instruction suivante. L'étiquette doit être à moins de -128 à +127 octets de l'instruction qui suit le saut.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

jcxz étiquette

Cycles

6, 5

JMP Jump Unconditionally

Continue l'exécution du programme à l'étiquette spécifiée par l'opérande destination. Les sauts sont «short» (entre -128 et 127 octets par rapport à l'instruction suivant le saut), «near» (entre -32768 et 32767 octets) ou «far» (dans un segment de code différent). À moins que la distance ne soit spécifiée explicitement, l'assembleur choisit le saut le plus petit possible. Avec des sauts «short» et «near», l'opérande spécifie une nouvelle adresse IP. Avec les sauts «far», l'opérande spécifie de nouvelles adresses IP et CS. Les différentes conditions sont présentées en annexe.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

jmp étiquette
 jmp reg16
 jmp reg32
 jmp mem16
 jmp mem32

Cycles

1

jmp mem48

LAHF Load Flags into AH Register

Transfère les bits 0 à 7 du registre FLAGS dans AH. Ceci comprend SF, ZF, AF, PF et CF mais pas TF, IF ni OF.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

lahf

Cycles

2

LAR Load Access Rights

Copie les droits d'accès d'un sélecteur dans le registre spécifié. L'opérande source doit être une opérande registre ou mémoire contenant un sélecteur. L'opérande destination doit être un registre qui recevra les droits d'accès si le sélecteur est valide et accessible avec le niveau de privilège courant.

Indicateurs

O	D	I	T	S	Z	A	P	C
					*			

Syntaxe

lar reg16, reg16

lar reg32, reg32

lar reg16, mem16

lar reg32, mem32

Cycles

8

8

LDS/LES/LFS/LGS/LSS**Load Far Pointer**

Lit et enregistre le «far pointer» spécifié par l'opérande mémoire source. L'instruction place la valeur de segment du pointeur dans DS, ES, FS, GS ou SS, dépendant de l'instruction. Ensuite elle place la valeur de l'offset du pointeur dans l'opérande destination.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

lds reg, mem
 les reg, mem
 lfs reg, mem
 lgs reg, mem
 lss reg, mem

Cycles

4
 4
 4
 4
 4, mp=8

LEA**Load Effective Address**

Calcule l'adresse effective (offset) de l'opérande mémoire source et enregistre le résultat dans le registre destination. Si l'opérande source est une adresse mémoire directe, l'assembleur code l'instruction comme si c'était la forme plus performante MOV reg, imméd (équivalente à MOV reg, OFFSET mem).

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

lea reg, mem

Cycles

1

LEAVE High Level Procedure Exit

Met fin au bloc de pile d'une procédure. LEAVE renverse l'action d'une instruction ENTER précédente en restaurant SP et BP aux valeurs qu'ils avaient avant que le bloc de pile de la procédure soit initialisé. LEAVE est équivalent à `mov sp, bp` suivi de `pop bp`. Cette instruction est déconseillée parce qu'elle est plus lente que les instructions qu'elle remplace.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

`leave`

Cycles

3

LOCK Lock the Bus

Interdit l'accès du bus à d'autres processeurs durant l'exécution de la prochaine instruction. C'est un préfixe. Elle doit précéder une instruction qui accède à une région de la mémoire à laquelle un autre processeur pourrait essayer d'accéder en même temps.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

`lock instruction`

Cycles

1

LODS/LODSB/LODSW/LODSD Load Accumulator from String

Charge AL, AX ou EAX avec un élément d'une chaîne en mémoire. ESI doit pointer vers l'élément source, même si un opérande est fourni. Pour chaque élément chargé, ESI est

ajusté selon la taille de l'opérande et l'état de DF. ESI est incrémenté si DF avait été mis à 0 avec CLD ou décrétementé si DF avait été mis à 1 avec STD.

Avec la forme LODS de l'instruction, un opérande doit être fourni pour indiquer la taille des éléments de données à traiter. Une dérogation de segment peut être fournie. Lorsque l'une des formes LODSB (octets), LODSW (mots) ou LODSD (doubles mots) est utilisée, c'est le choix d'instruction qui détermine la taille des éléments de données à traiter, et si l'élément sera chargé dans AL, AX ou EAX.

LODS et ses variations ne sont pas utilisées avec un préfixe de répétition, car il n'est pas utile de charger des données mémoire à un même registre.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

lods src
 lodsb
 lodsw
 lodsd

Cycles

2
 2
 2
 2

LOOP/LOOPW/LOOPD Loop

Boucle de façon répétitive vers une étiquette donnée. LOOP décrémente ECX en mode 32 bits ou CX en mode 16 bits (sans changer aucun indicateur) et si le résultat n'est pas 0, continue l'exécution à l'adresse spécifiée par l'opérande. On peut utiliser LOOPW pour forcer le test de CX ou LOOPD pour forcer celui de ECX. Quand CX est 0 après avoir été décrétementé, l'exécution continue à l'instruction suivante. L'opérande doit spécifier une étiquette «short» (entre -128 et +127) à partir de l'instruction suivant le LOOP.

Cette instruction est déconseillée parce qu'elle est plus lente que les instructions qu'elle remplace.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

loop	etiquette
loopw	etiquette
loopd	etiquette

Cycles

5, 6

LOOPcondition/LOOPconditionW/LOOPconditionD**Loop Conditionally**

Boucle de façon répétitive vers une étiquette donnée si la condition est vraie et si ECX (CX en mode 16 bits) n'est pas 0. On peut forcer l'utilisation de CX avec la forme W de l'instruction, ou celle de ECX avec la forme D. L'instruction décrémente ECX (sans changer aucun indicateur) et teste si ZF a été mis à zéro par une instruction précédente telle que CMP. Avec LOOP et LOOPZ (elles sont synonymes), l'exécution se poursuit à l'étiquette si l'indicateur ZF est 1 et que ECX n'est pas 0.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

loope	etiquette
loopew	etiquette
looped	etiquette
loopz	etiquette
loopzw	etiquette
loopzd	etiquette

Cycles

7, 8

LSL**Load Segment Limit**

Charge la limite de segment d'un sélecteur dans un registre spécifié. L'opérande source doit être un opérande registre ou mémoire contenant un sélecteur. L'opérande destination doit être un registre qui recevra la limite de segment si le sélecteur est valide et visible au

niveau de privilège courant. L'indicateur ZF est mis à 1 si la limite de segment est transférée, et mis à zéro sinon.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

lsl reg16, reg16
 lsl reg32, reg32

Cycles

8

lsl reg16, mem16
 lsl reg32, mem32

8

MOV**Move Data**

Copie la valeur de l'opérande source dans l'opérande destination. Si l'opérande destination est SS, les interruptions sont interdites jusqu'à l'exécution de l'instruction.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

mov reg, reg
 mov mem, reg
 mov reg, mem
 mov mem, immed
 mov reg, immed
 mov segreg, reg16
 mov segreg, mem16
 mov reg16, segreg
 mov mem16, segreg

Cycles

1

1

1

1

1

2, 3

2, 3

1

1

MOV Move to/from Special Registers

Copie une valeur d'un registre spécial de ou vers un registre général de 32 bits. Les registres spéciaux incluent les registres de contrôle CR0, CR2 et CR3, les registres de déboguage DR0, DR1, DR2, DR3, DR6 et DR7, et les registres de test TR3, TR4, TR5, TR6 et TR7.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				?	?	?	?	?

Syntaxe

Cycles

mov	eax, cr0	22
mov	eax, cr2	12
mov	eax, cr3	21, 46
mov	eax, cr4	14
mov	controlreg, reg32	4
mov	reg32, debugreg	11, 12
mov	debugreg, reg32	11, 12
mov	reg32, testreg	11, 12
mov	testreg, reg32	11, 12

MOVS/MOVSb/MOVSd/MOVSq Move String Data

Copie une chaîne d'une partie de la mémoire à une autre. EDI (DS:SI) doit pointer vers la chaîne source et EDI (ES:DI) vers l'adresse destination, même si des opérandes sont fournis. Pour chaque élément déplacé, EDI et ESI sont ajustés selon la taille des opérandes et l'état de DF. Ils sont incrémentés si DF avait été mis à 0 avec CLD auparavant, ou décrémentés s'il avait été mis à 1 avec STD.

Avec la forme MOVS de l'instruction, des opérandes doivent être fournis pour indiquer la taille des éléments de données à traiter. Une dérogation de segment peut être donnée pour l'opérande source mais pas pour la destination. Lorsqu'on utilise l'une des formes MOVSb (octets), MOVW (mots) ou MOVSd (doubles mots), c'est le choix d'instruction qui détermine la taille des éléments de données à traiter.

MOVS et ses variantes sont normalement utilisées avec le préfixe REP.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```
movs    reg, reg
movsb
movsw
movsd
```

Cycles

4
4

MOVSX Move with Sign-Extend

Copie et étend en signe la valeur de l'opérande source et place le résultat dans le registre destination. **MOVSX** est utilisée pour copier un opérande source signé de 8 bits ou 16 bits dans un registre plus grand de 16 bits ou de 32 bits.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```
movsx    reg, reg
movxs    reg, mem
```

Cycles

3
3

Exemple

```
mov    al, 0x9E
movsx  bx, al    ; BX = 0xFF9E
```

MOVZX Move with Zero-Extend

Copie et étend en zéros la valeur de l'opérande source et place le résultat dans le registre destination. **MOVZX** est utilisée pour copier un opérande source non signé de 8 bits ou 16 bits dans un registre plus grand de 16 bits ou de 32 bits.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```
movzx    reg, reg
movzx    reg, mem
```

Cycles

3
3

Exemple

```
mov      al, 0x9E
movzx    bl, al          ; BL = 0x009E
```

MUL Unsigned Multiply

Multiplie l'opérande destination implicite par un opérande source spécifié. Les deux opérandes sont traités comme des nombres non signés. Si un seul opérande de 8 bits est donné, la destination implicite est AL et le produit va dans AX. Si un seul opérande de 16 bits est donné, la destination implicite est AX et le produit va dans la paire de registres DX:AX. Si l'opérande est de 32 bits, le produit va dans la paire de registres EDX:EAX. CF et OF sont mis à 1 si AH n'est pas 0 pour des opérandes de 8 bits ou si DX n'est pas 0 pour des opérandes de 16 bits.

Indicateurs

O	D	I	T	S	Z	A	P	C
*					?	?	?	*

Syntaxe

```
mul      reg
mul      mem
```

Cycles

8, 11, 10

Exemple

```

mov    eax, 0x9FFFFFFF
mov    ebx, 3
mul    ebx          ; EDX = 0x00000001; EAX = 0xDFFFFFFD

```

NEG Two's Complement Negation

Remplace l'opérande par son complément à 2. NEG effectue cette opération en soustrayant l'opérande de 0. Si l'opérande est 0, l'indicateur de retenue est mis à 0, sinon, il est mis à 1. Si l'opérande contient la valeur négative la plus grande possible (-128 pour 8 bits ou -32768 pour 16 bits), la valeur ne change pas, mais les bits de retenue et de dépassement de capacité sont mis à 1.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

```

neg    reg
neg    mem

```

Cycles

1
3

NOP No Operation

N'effectue aucune opération. NOP peut être utilisée pour réaliser des délais ou l'alignement des instructions.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```

nop

```

Cycles

1

NOT One's Complement Negation

Inverse chaque bit de l'opérande en mettant à 0 les bits qui sont 1 et à 1 les bits qui sont 0.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

not reg
not mem

Cycles

1
3

OR Inclusive OR

Effectue l'opération OU bit par bit entre les opérandes source et destination et enregistre le résultat dans l'opérande destination. Pour chaque position de bit dans les opérandes, si l'un ou l'autre des deux bits est 1, le bit correspondant du résultat est 1. Sinon, le bit correspondant du résultat est mis à 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Syntaxe

or reg, reg
or mem, reg
or reg, mem
or reg, immedi
or mem, immedi

Cycles

1
3
2
1
3

Exemple

mov ax, 0x0405 ; AX = 45 en BCD non compacté
or ax, 0x3030 ; AX = 45 en ASCII

OUT Output to Port

Copie un octet, un mot ou un double mot de l'accumulateur vers un port. L'adresse du port est spécifiée par l'opérande destination, qui peut être DX ou une constante de 8 bits.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

out immed8, accum
out dx, accum

Cycles

12, mp=9, 26, mv=24
12, mo=9, 25, mv=24

OUTS/OUTSB/OUTSW/OUTSD Output String to Port

Envoie une chaîne à un port. La chaîne est considérée comme la source et doit être pointée par ESI (DS:SI) même si un opérande est fourni. Le port de sortie est spécifié dans DX. Pour chaque élément transmis, ESI est ajusté selon la taille de l'opérande et l'état de l'indicateur de direction DF. ESI est incrémenté si DF a été mis à zéro avec **CLD**, ou décrémenté s'il a été mis à 1 avec **STD**.

Avec la forme **OUTS** de l'instruction, un opérande doit être fourni pour indiquer la taille des éléments à transmettre. Une dérogation de segment peut être donnée. Lorsque l'une des formes **OUTSB** (octets) **OUTSW** (mots) ou **OUTSD** (doubles mots) est utilisée, c'est le choix d'instruction qui détermine la taille des éléments de données à transmettre.

OUTS et ses variations sont normalement utilisées avec le préfixe **REP**. Avant l'exécution de l'instruction, CX doit contenir le nombre d'éléments à transmettre. En mode protégé, une faute générale de protection se produit si **OUTS** est utilisé quand le niveau de privilège courant est supérieur à la valeur de l'indicateur IOPL.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

outs dx, src
 outsb
 outsw
 outsd

Cycles

13, pm=10, 27, vm=24

POP**Pop**

Dépile le sommet de la pile dans l'opérande destination. La valeur à ESP (SS:SP) est copiée dans l'opérande destination et ESP est incrémenté de 2 ou 4 selon la taille de l'opérande. L'opérande destination peut être un emplacement mémoire, un registre général ou un registre de segment autre que CS. Utiliser RET pour dépiler CS.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

pop reg16
 pop reg32
 pop mem16
 pop mem32

Cycles

1

3

POPA/POPAD**Pop All**

Dépile des 16 octets au sommet de la pile dans les 8 registres généraux. Les registres sont dépilés dans l'ordre suivant :DI, SI, BP, SP, BX, DX, CX, AX. Les valeurs pour le registre SO est supprimée plutôt que copiée dans SP. POPA dépile toujours des registres de 16 bits. Utiliser POPAD pour dépiler des registres de 32 bits.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

popa
popad

Cycles

5

POPF/POPFD**Pop Flags**

Dépile la valeur située au sommet de la pile dans le registre d'indicateurs. POPF dépile toujours dans FLAGS. Utiliser POPFD pour dépiler dans le registre de 32 bits EFLAGS..

Indicateurs

O	D	I	T	S	Z	A	P	C
*	*	*	*	*	*	*	*	*

Syntaxe

popf
popfd

Cycles

6, pm=4

PUSH/PUSHW/PUSHD**Push**

Empile l'opérande source sur la pile. SP est décrémenté de 2 ou de 4 selon la dimension de l'opérande, et la valeur de la source est copiée à l'adresse ESP (SS:SP). L'opérande peut être une adresse mémoire, un registre général ou un registre de segment. Il peut aussi être une constante. **PUSH SP** sauve la valeur de SP après l'empilement. **PUSHW** et **PUSHD** empilent respectivement un mot et un double mot.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

push reg16
push reg32
pushw reg16
pushd reg32

Cycles

1

push	mem16	2
push	mem32	
push	segreg	1
pushw	segreg	
pushd	segreg	
push	immed	1
pushw	immed	
pushd	immed	

PUSHA/PUSHAD Push All

Empile les huit registres généraux. Ces derniers sont empilés dans l'ordre suivant : AX, CX, DX, BX, SP, BP, SI, DI. La valeur empilée pour SP est la valeur avant l'instruction **PUSHA** empile toujours des registres de 16 bits. Utiliser **PUSHAD** pour empiler des registres de 32 bits.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

pusha
pushad

Cycles

5

PUSHF/PUSHFD Push Flags

Empile le registre d'indicateurs. PUSHF empile toujours le registre d'indicateurs de 16 bits FLAGS. Utiliser PUSHFD pour empiler le registre de 32 bits EFLAGS.

Indicateurs

O	D	I	T	S	Z	A	P	C
---	---	---	---	---	---	---	---	---



Syntaxe

pushf
pushfd

Cycles

4, mp=3

RCL/RCR/ROL/ROR Rotate

Effectue une rotation des bits de l'opérande destination du nombre de bits spécifié dans l'opérande source. **RCL** et **ROL** font une rotation vers la gauche; **RCR** et **ROR**, vers la droite.

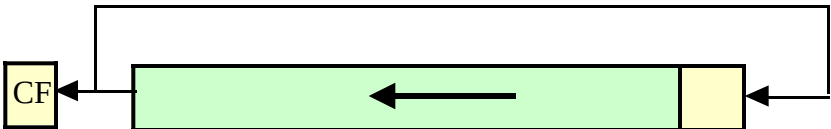
ROL et ROR font une rotation du nombre de bits dans l'opérande. Pour chaque rotation, le bit de gauche ou le bit de droite est copié dans CF en plus d'effectuer une rotation. RCL et RCE font une rotation incluant CF. CF devient ainsi une extension de l'opérande, de sorte qu'une rotation sur 9 bits est effectuée pour des opérandes de 8 bits, de 17 bits pour des opérandes de 16 bits, et de 33 bits pour des opérandes de 32 bits.

L'opérande source peut être CL ou une constante de 8 bits. Les nombres de rotations supérieurs à 31 sont tronqués modulo 32. OF n'est modifié que par les rotations sur un seul bit et vaut MSB(dest) XOR CF. Pour des rotations de plusieurs bits, OF est indéfini.

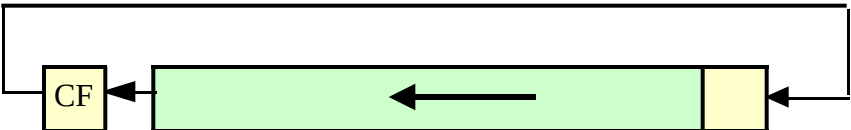
Indicateurs

O	D	I	T	S	Z	A	P	C
*								*

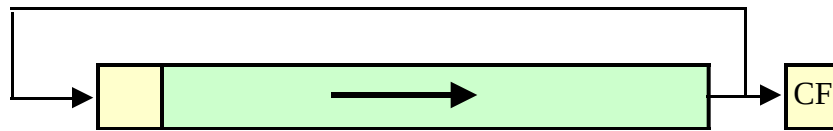
ROL



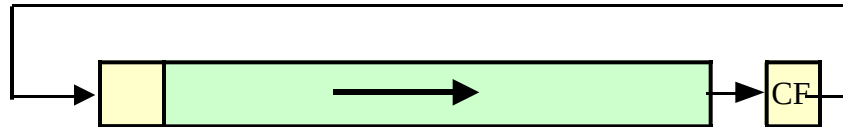
RCL



ROR



RCR



Syntaxe

Cycles

rol	reg, 1	1
ror	reg, 1	1
rcl	reg, 1	1
rcr	reg, 1	1
rol	mem, 1	3
ror	mem, 1	3
rcl	mem, 1	3
rcr	mem, 1	3
rol	reg, cl	4
ror	reg, cl	4
rcl	reg, cl	7-24
rcr	reg, cl	7-24
rol	mem, cl	4
ror	mem, cl	4
rcl	mem, cl	9-26
rcr	mem, cl	9-26
rol	reg, immed8	1
ror	reg, immed8	1
rcl	reg, immed8	8-25
rcr	reg, immed8	8-25
rol	mem, immed8	3
ror	mem, immed8	3
rcl	mem, immed8	10-27
rcr	mem, immed8	10-27

Exemples

```

mov  eax, 0x8FFFFFFF
rol  eax, 1           ; EAX = 0x1FFFFFFF; CF = 1; OF = 1;
clc
mov  eax, 0x8FFFFFFF
rcl  eax, 1           ; EAX = 0x1FFFFFFE; CF = 1; OF = 1;

```

RDTSC Read from Time Stamp Counter

Copie le contenu du registre de 64 bits TSC (Time Stamp Counter) dans la paire de registres EDS:EAX. Ce registre est incrémenté à chaque cycle d'horloge

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

rdtsc

Cycles

6, 11

REP Repeat String

Répète une instruction de chaîne le nombre de fois indiqué par CX. CX est d'abord comparé à 0; s'il est 0, l'exécution passe à l'instruction suivante. Sinon, CX est décrémenté, l'instruction de chaîne est exécutée et la boucle continue. REP est utilisée avec MOVS, STOS, INS et OUTS.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```

rep movs dest, scr
rep movsb rep

```

Cycles

6, 13n

rep movsw	
rep movsd	
rep stos dest	6, 9+3n
rep stosb rep stosw	
rep stosw	
rep stosd	
rep ins dest, dx	11+3n, pm=(8, 25)+3n
rep insb	
rep insw	
rep insd	
rep outs dx, src	13+4n, pm=(10, 27)+4n
rep outsb rep outsd	
rep outsw	
rep outsd	

REPcondition Repeat String Conditionally

Répète une instruction de chaîne tant que la condition est vraie et que ECX $\neq$ 0 (CX en mode 16 bits). REPE et REPZ (synonymes) répètent tant que ZF est 1. REPNE et REPNZ (synonymes) répètent tant que ZF est 0. Ces instructions ne sont utilisées qu'avec SCAS et CMPS, car ce sont les deux seules instructions de chaîne qui affectent ZF. Avant l'exécution de l'instruction, ECX devrait contenir le nombre maximum désiré de répétitions et par la suite est décrémenté à chaque itération.

Indicateurs

O	D	I	T	S	Z	A	P	C
*								

Syntaxe

```
repe cmps src, dest
repe cmpsb
repe cmpsw
repe cmpsd
```

Cycles

7, 9+4n

repe scas dest	7, 9+4n
repe scasb repz scasw	
repe scasw	
repe scasd	
repne cmps src, dest	7, 9+4n
repne cmpsb	
repne cmpsw	
repne cmpsd	
repne scas dest	7, 9+4n
repne scasb	
repne scasw	
repne scasd	

RET/RETN/RETF Return from Procedure

Retourne d'une procédure en continuant l'exécution à l'adresse dépilée du sommet de la pile. Une valeur immédiate peut être fournie pour indiquer un nombre d'octets supplémentaires à dépiler. Cette valeur sert à ajuster la pile pour des arguments empilés avant l'appel de la procédure.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```
ret
retn
ret    immed16
retn   immed 16
```

Cycles

2

4, 23

ROL/ROR Rotate

Voir **RCL** / **RCR**.

SAHF Store AH into Flags

Copie AH dans les bits 0 à 7 du registre EFLAGS. Copie SF, ZF, AF, PF et CF mais pas OF, DF, IF ni TF.

Indicateurs

O	D	I	T	S	Z	A	P	C
				*	*	*	*	*

Syntaxe

sahf

Cycles

2

SAL/SAR Shift

Voir **SHL** / **SHR** / **SAL** / **SAR**.

SBB Subtract with Borrow

Ajoute CF à l'opérande source, puis soustrait cette valeur de l'opérande destination. Le résultat est enregistré dans l'opérande destination. **SBB** est utilisée pour soustraire la partie la moins significative de nombres de précision multiple.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

sbb reg, reg
sbb mem, reg
sbb reg, mem
sbb reg, immed

Cycles

1
3
2
1

sbb mem, immedi

3

Exemple

Soit deux nombres de 64 bits débutant aux adresses A et B. On veut les soustraire B de A et placer la réponse dans A.

```

mov     eax, A      ; double mot de poids faible
sub     eax, B      ; différence
mov     A, eax      ; écriture du résultat
mov     eax, A+4    ; double mot de poids fort
sbb     eax, B+4    ; au cas où il y aurait un emprunt de A - B
mov     A+4, eax    ; écriture du résultat

```

SCAS/SCASB/SCASW/SCASD Scan String Flags

Parcourt une chaîne dont l'adresse est dans EDI (ES:DI) pour trouver la valeur spécifiée dans AL, AX ou EAX. Pour chaque élément, l'élément destination est soustrait de la valeur de l'accumulateur et les indicateurs sont positionnés en fonction du résultat, mais ce dernier n'est pas enregistré. EDI est ajusté selon la taille des opérandes et l'état de l'indicateur de direction DF. ESI est incrémenté si DF avait été mis à zéro avec CLD, ou décrémenté si DF avait été mis à 1 avec STD.

Avec la forme SCAS de l'instruction, un opérande doit être fourni pour indiquer la taille des éléments de données à traiter. La dérogation de segment n'est pas permise. Lorsque l'une des formes SCASB (octets), SCASW (mots) ou SCASD (doubles mots) est utilisée, c'est le choix d'instruction qui détermine la taille des éléments de données à traiter, et si l'élément recherché est dans AL, AX ou EAX.

SCAS et ses variantes sont normalement utilisées avec un préfixe de répétition. REPNE (ou REPNZ) est utilisé pour trouver le premier élément d'une chaîne qui corresponde à la valeur de l'accumulateur. REPE (ou REPZ) est utilisée pour trouver le premier désaccord. Avant la recherche, ECX (CX en mode 16 bits) doit contenir le nombre maximum d'itérations désiré. Après l'instruction REPNE SCAS, ZF est 0 si la chaîne ne contient pas la valeur spécifiée. Après un REPE SCAS, ZF est 1 si la chaîne ne contient rien d'autre que la valeur spécifiée.

À la fin de l'instruction, EDI (ES:DI) pointe sur l'élément qui suit (si DF = 0) ou qui précède (si DF = 1) l'accord ou le désaccord. Si ECX atteint 0, EDI pointe à l'élément qui suit ou précède la dernière comparaison. L'indicateur de zéro reflète le résultat de la dernière comparaison, et non la valeur de ECX.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

scas dest
scasb
scasw
scasd

Cycles

4

SETcondition Set Conditionally

Met l'octet spécifié à 1 (0xFF) si la condition est vraie, ou à 0 (0x00) si la condition est fausse.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

set_{condition} reg8
set_{condition} mem8

Cycles

1

2

SGDT/SIDT/SLDT Store Descriptor Table Register

Copie un registre de table de descripteurs dans un opérande mémoire spécifié. SGDT enregistre le registre de Table Globale de Descripteurs (GDTR); SIDT, le registre de Table de Vecteurs d'Interruption (IVT), et SLDT, le registre de Table Locale de Descripteurs. Ces instructions ne sont utiles qu'en mode privilégié.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

sgdt	mem48
sidt	mem48
sldt	reg16
sldt	mem16

Cycles

4
4
2
2

SHL/SHR/SAL/SAR**Shift**

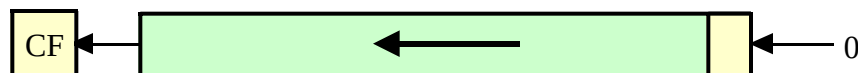
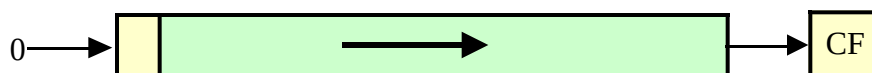
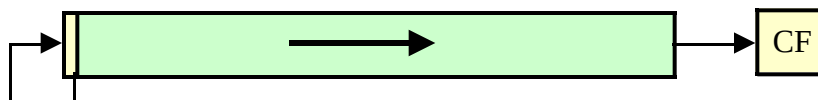
Décale les bits de l'opérande destination le nombre de fois spécifié par l'opérande source. SAL et SHL décalent vers la gauche, SAR et SHR vers la droite.

Avec SHL, SAL et SHR, le bit décalé à la fin de l'opérande est copié dans CF et le bit le plus à gauche ou le plus à droite libéré par le décalage est remplacé par 0. Avec SAR, le bit décalé à la fin de l'opérande est copié dans CF et le bit le plus à gauche libéré par le décalage conserve sa valeur antérieure, préservant ainsi le signe de l'opérande. SAL et SHL sont synonymes.

Le nombre de décalages peut être dans CL ou une valeur immédiate de 8 bits. Les nombres de décalages supérieurs à 31 sont tronqués modulo-32. Le bit OF n'est affecté que pour un décalage de 1 bit. Pour des décalages multiples, OF est indéfini.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	?	*	*

SAL et SHL**SHR****SAR**

Syntaxe		Cycles
	sar reg, 1	1
	sal reg, 1	
	shr reg, 1	
	sal reg, 1	
	sar mem, 1	3
	sal mem, 1	
	shl mem, 1	
	shr mem, 1	
	sar reg, cl	4
	sal reg, cl	
	shl reg, cl	
	shr reg, cl	
	sar mem, cl	4
	sal mem, cl	
	shl mem, cl	
	shr mem, cl	
	sar reg, imm8	1
	sal reg, imm8	
	shl reg, imm8	
	shr reg, imm8	
	sar mem, imm8	3
	sal reg, imm8	
	shl reg, imm8	
	shr reg, imm8	
Exemples	mov ax, 0x1234	
	sar ax, 2	; AX = 0x048D
	mov bx, 0x9106	
	sar bx, 3	; BX = 0xF220
	mov cx, 0x9106	
	shr cx, 3	; CX = 0x1220

SHLD/SHRD Double Precision Shift

Décale les bits de l'opérande source vers l'opérande destination. Le nombre de bits de décalage est spécifié par le troisième opérande. SHLD décale le premier opérande vers la gauche du nombre de positions spécifié dans le troisième opérande. Les positions libérées par le décalage sont remplies par les bits les plus significatifs du deuxième opérande.

SHRD décale le premier opérande vers la droite du nombre de positions spécifié dans le troisième opérande. Les positions libérées par le décalage sont remplies par les bits les moins significatifs du deuxième opérande.

Le nombre de décalages peut être dans CL ou une valeur immédiate de 8 bits. Si un nombre de décalages supérieur à 31 est fourni, il est tronqué modulo 32.

Indicateurs

O	D	I	T	S	Z	A	P	C
?				*	*	?	*	*

Syntaxe

Cycles

shld	reg16, reg16, immed8	4
shld	reg32, reg32, immed8	
shld	mem16, reg16, immed8	4
shld	mem32, reg32, immed8	
shrd	reg16, reg16, immed8	4
shrd	reg32, reg32, immed8	
shrd	mem16, reg16, immed8	4
shrd	mem32, reg32, immed8	
shld	reg16, reg16, cl	4
shld	reg32, reg32, cl	
shld	mem16, reg16, cl	5
shld	mem32, reg32, cl	
shrd	reg16, reg16, cl	4
shrd	reg32, reg32, cl	

shrd mem16, reg16, cl 5
shrd mem32, reg32, cl

STC Set Carry Flag

Met l'indicateur de retenue CF à 1.

Indicateurs

O	D	I	T	S	Z	A	P	C
								1

Syntaxe

stc

Cycles

2

STD Set Direction Flag

Met l'indicateur de direction DF à 1. Toutes les instructions de chaîne subséquentes procéderont vers le bas, i.e. ESI et/ou EDI seront décrémentés.

Indicateurs

O	D	I	T	S	Z	A	P	C
								1

Syntaxe

std

Cycles

2

STI Set Interrupt Flag

Met l'indicateur d'interruption IF à 1, ce qui autorise les interruptions matérielles. Si les interruptions avaient été interdites par une instruction CLI antérieure, les interruptions en attente ne seront pas traitées immédiatement; mais seulement après l'instruction qui suit STI.

Indicateurs

O	D	I	T	S	Z	A	P	C
1								

Syntaxe

sti

Cycles

7

STOS/STOSB/STOSW/STOSD Store String Data

Copie la valeur de AL, AX ou EAX dans un emplacement mémoire dont l'adresse est contenue dans EDI (ES:DI). Ensuite, EDI est ajusté selon la taille de l'élément transféré et l'état de l'indicateur de direction. EDI est incrémenté si DF avait été mis à 0 avec CLD ou décrémenté si DF avait été mis à 1 avec STD.

Avec la forme STOS de l'instruction, un opérande doit être fourni pour indiquer la taille des éléments à copier. La dérogação de segment n'est pas permise. Lorsque l'une des formes STOSB (octets), STOSW (mots) ou STOSD (doubles mots) est utilisée, c'est le choix d'instruction qui détermine la taille des éléments de données à traiter et si l'élément provient de AL, AX ou EAX.

STOS et ses variantes sont souvent utilisées avec le préfixe REP pour remplir une chaîne avec une même valeur. Avant d'utiliser l'instruction REP, ECX doit contenir la nombre d'itérations désiré.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

```

stos    dest
stosb
stosw
stosd

```

Cycles

3

SUB**Subtract**

Soustrait l'opérande source de l'opérande destination et enregistre le résultat dans l'opérande destination..

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

```
sub    reg, reg
sub    mem, reg
sub    reg, mem
sub    reg, imméd
sub    mem, imméd
```

Cycles

```
1
3
2
1
3
```

TEST**Logical Compare**

TEST effectue une opération AND bit par bit entre les deux opérandes mais n'enregistre pas le résultat. Les indicateurs sont modifiés en conséquence. Cette instruction est utilisée pour positionner les indicateurs en vue d'une instruction conditionnelle subséquente telle que `jcondition` ou `setcondition`.

Indicateurs

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Syntaxe

```
test    reg, reg
test    mem, reg
test    reg, mem
test    reg, imméd
test    mem, imméd
```

Cycles

```
1
2
2
1
2
```

Exemple

```
test    ax, ax ; ZF = 1 si AX = 0, sinon ZF = 0
```

VERR/VERW Verify Read or Write

Détermine si le sélecteur de segment spécifié est atteignable et si on peut y lire (VERR) ou y écrire (VERW) avec le niveau de privilège courant. Si oui, Z = 1, sinon, Z = 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
					*			

Syntaxe

verr reg16

verr mem16

verw reg16

verw mem16

Cycles

7

7

7

7

WAIT Wait

Suspend l'exécution du processeur jusqu'à ce que ce dernier reçoive via la broche BUSY# l'indication que le coprocesseur a terminé une opération simultanée. Cette instruction était utilisée avec le coprocesseur 80x87.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

wait

Cycles

1

XADD Exchange and Add

Additionne les opérandes source et destination, et place la somme dans la destination. En même temps, la valeur originale de la destination est placée dans la source. Les indicateurs sont ajustés selon le résultat de l'addition.

Indicateurs

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Syntaxe

xadd mem, reg

xadd reg, reg

Cycles

3

4

Exemple

mov al, 0x08

mov bl, 0x03

xadd al, bl ; AL = 0x0B, BL = 0x08

XCHG**Exchange**

Échange les valeurs des opérandes source et destination.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

xchg reg, reg

xchg reg, mem

xchg mem, reg

Cycles

3

3

3

XLAT/XLATB**Translate**

Change la valeur de AL d'un index de table en entrée dans la table. Avant l'instruction, AL est un index non signé dans une table dont l'adresse est dans EBX (DS:BX). Après l'instruction, AL contient la valeur de la table à l'index spécifié. Aucun opérande n'est nécessaire, mais on peut en fournir un à titre de documentation. XLATB est un synonyme de XLAT.

Indicateurs

O	D	I	T	S	Z	A	P	C

Syntaxe

xlat
xlatb

Cycles

4

Exemple

```
Table db '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
lea    ebx, Table
mov    al, 0x0E
xlatb                                     ; AL = 'E'; conversion binaire-ASCII
```

XOR**Exclusive OR**

Effectue le ou disjonctif bit par bit entre l'opérande source et l'opérande destination et enregistre le résultat dans l'opérande destination. Pour chaque position de bit des opérandes, si les deux bits sont différents, le bit correspondant du résultat est 1. Sinon, le bit correspondant du résultat est 0.

Indicateurs

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Syntaxe

xor reg, reg
xor mem, reg
xor reg, mem
xor reg, immediat
xor mem, immediat

Cycles

1
3
2
1
3

Exemple

```
xor    eax, eax          ; EAX = 0x00000000
xor    al, 0xFF          ; AL = NOT AL
```

Conditions pour les sauts et les sets

Mnémonique	Indicateurs vérifiés	Description
B/NAE	$CF = 1$	Vrai si inférieur/pas supérieur ou égal (comparaison non signée)
AE/NB	$CF = 0$	Vrai si supérieur ou égal / pas inférieur (comparaison non signée)
BE/NA	$CF = 1$ ou $ZF = 1$	Vrai si inférieur ou égal / pas supérieur (comparaison non signée)
A/NBE	$CF = 0$ et $ZF = 0$	Vrai si supérieur / pas inférieur ou égal (comparaison non signée)
E/Z	$ZF = 1$	Vrai si égal (zéro)
NE/NZ	$ZF = 0$	Vrai si pas égal (pas zéro)
L/NGE	$SF \neq OF$	Vrai si plus petit / pas plus grand (comparaison signée)
GE/NL	$SF = OF$	Vrai si plus grand ou égal / pas plus petit (comparaison signée)
LE/NG	$ZF=1$ ou $SF \neq OF$	Vrai si plus petit ou égal / pas plus grand (comparaison signée)
G/NLE	$ZF=0$ et $SF=OF$	Vrai si plus grand/pas plus petit ou égal (comparaison signée)
S	$SF=1$	Vrai si négatif
NS	$SF= 0$	Vrai zéro ou positif
C	$CF=1$	Vrai si retenue (ou emprunt)
NC	$CF=0$	Vrai si pas de retenue (ou d'emprunt)
O	$OF = 1$	Vrai si débordement de capacité
NO	$OF=0$	Vrai si pas débordement de capacité
P/PE	$PF=1$	Vrai si parité/parité paire
NP/PO	$PF=0$	Vrai si pas de parité/parité impaire

6. Mots réservés

Les identificateurs que vous utiliserez dans vos programmes ne peuvent pas figurer parmi les mots réservés.

Les noms des registres

AH, AL , AX, EAX, BH, BL, BX, EBX, CH, CL, CX, ECX, DH, DL, DX, EDX, SI, ESI, DI, EDI, BP, EBP, SP, ESP, CS, DS, ES, FS, GS, IP, EIP,

Les noms des instructions

AAA, AAD, AAM, etc.

Les directives

ALIGN, .ALPHA, ASSUME, BYTE, .CODE, COMM, COMMENT, .CONST, .CREF, .DATA, .DATA?, DB, DD, DF, DOSSEG, DQ, DT, DW, DWORD, ELSE, .ELSE, END, ENDIF, ENDM, ENDP, ENDS, .EQU, ERRnn, EVEN, EXITM, EXTRN, EXTERN, .FARDATA, .FARDATA?, FWORD, GROUP, IF, .IF, IF1, IF2, IFB, IFDEF, IFDIF, IFE, IFDN, IFNB, IFNDEF, INCLUDE, INCLUDELIB, IRP, IRPC, LABEL, .LALL, .LFCOND, .LIST, LOCAL, MACRO, .MODEL, NAME, ORG, &OUT, PAGE, PROC, PUBLIC, PURGE, QWORD, .RADIX, RECORD, REOT, .SALL, SEGMENT, .SEQ, .SFCOND, .STACK, STRUC, STRUCT, SUBTTL, .TFCOND, TITLE, TWORD, UNION, WORD, .XALL, .XCREF, .XLIST

Les opérateurs

AND, BYTE, COMMENT, CON, DUP, EQ, FAR, GE, GT, HIGH, LE, LENGTH, LINE, LOLW, LT, MASK, MOD, NE, NEAR, NOT, NOTHING, OFFSET, OR, PTR, SEG, SHL, SHORT, SHR, SIZE, SIZEOF, STACK, THIS, TYPE, WHILE, WIDTH, WORD, XOR.