Aide mémoire jeu d'instructions 80x86

Registres	
généraux 8 bits	ah,al,bh,bl,ch,cl,dh,dl
généraux 16 bits	ax,bx,cx,dx
base	bx,bp
index	si,di
pointeur de pile	sp

Instructions de transfert	
mov *dest, orig*	*dest* ← *orig*
dest/orig : R/R, R/M, R/I, M/R, M/I	
lea *dest, orig*	*dest* ← adresse de *orig*
dest/orig : R/M	

Instructions arithmétiques de base	
add *dest, op*	*dest* ← *dest* + *op*
adc *dest, op*	*dest* ← *dest* + *op* + C
sub *dest, op*	*dest* ← *dest* - *op*
sbb *dest, op*	*dest* ← *dest* - *op* - C
neg *dest* (R, M)	*dest* ← - *dest*
dest/op : R/R, R/M, R/I, M/R, M/I	

Instructions logiques	
or *dest, op*	*dest* ← *dest* | *op*
and *dest, op*	*dest* ← *dest* & *op*
xor *dest, op*	*dest* ← *dest* ^ *op*
not *dest* (R, M)	*dest* ← ! *dest*
dest/op : R/R, R/M, R/I, M/R, M/I	

Instructions arithmétiques avancées	
mul *op* (8 bits)	ax ← al * *op*
mul *op* (16 bits)	dx.ax ← ax * *op*
div *op* (8 bits)	al ← ax/*op*, ah ← reste
div *op* (16 bits)	ax←dx.ax/*op*, *dx* ←*reste*
op : R, M	
imul, **idiv** : variantes ***signées***	

Décalages et rotations	
rol *dest, op*	rotation gauche *dest*
ror *dest, op*	rotation droite *dest*
rcl *dest, op*	rot. gauche *dest* .C
rcr *dest, op*	rot. droite C.*dest*
shl *dest, op*	décalage gauche *dest.0*
shr *dest, op*	décalage droite 0.*dest*
dest : R, M	*op* : imm, cl
bit sortant → C	*op* = nbr. d'opérations

Gestion de la pile	
push *op* (16 bits)	sp ← sp - 2, [sp] ← *op*
pop *op* (16 bits)	*op* ← [sp], sp ← *sp + 2*
op : R,M	

O. Bailleux - 2008

Opérandes	
registre (R)	un des registres
immédiat (I)	une constante
mémoire (M)	une adresse mémoire

Opérandes mémoire
[base]
[index]
[base + déplacement]
[index + déplacement]
[base + index + deplacement]
variable
[adresse]

Incrémentation et décrémentation	
inc *dest*	*dest* ← *dest* + 1
dec *dest*	*dest* ← *dest* - 1
dest : M, R	

Sauts et branchements inconditionnels	
jmp *addr*	saut inconditionnel
call *addr*	appel sous-programme
ret	retour sous-programme
ret *op* (I)	idem, puis *sp* ← *sp* + *op*

Sauts conditionnels / carry	
jc *addr*	saut si C=1
jnc *addr*	saut si C=0

Comparaison	
cmp *op1* , *op2*	*op1* - *op2* → perdu
op1/op2 : R/R, R/M, R/I, M/R, M/I	
permet d'activer les flags avant saut conditionnel	

Sauts conditionnels / entiers non signés	
jb *addr*	saut si <
ja *addr*	saut si >
jbe *addr*	saut si <=
jae *addr*	saut si >=

Sauts conditionnels / entiers signés	
jl *addr*	saut si <
jg *addr*	saut si >
jle *addr*	saut si <=
jge *addr*	saut si >=

Sauts conditionnels / entiers	
je *addr*	saut si =
jne *addr*	saut si ≠
jz *addr*	saut si = 0
jnz *addr*	saut si ≠ 0