**TP Architecture des ordinateurs**

**4 TP2 : le corrigé**

**Le fichier Adr.c**

#define constante 2

typedef struct {

int champ1;

char champ2, champ3;

} rec\_t;

int variable1, variable2, variable3, \*p1, i;

rec\_t enreg;

unsigned char tab\_byte[9];

rec\_t tab\_enr[9];

void proc() {

asm {

**// AX = 1; (adressage immédiat)**

mov ax,1

**// AX = constante; (adressage immédiat)**

mov ax,constante

**// AX = variable1; (adressage direct)**

mov ax,[variable1]

**// AX = variable2; (adressage basé sur BX)**

mov bx,offset variable2 // bx = &variable2;

mov ax,[bx] // adressage basé sur BX

**// p1 = &variable3; (via SI)**

mov si,offset variable3

mov [p1],si

**// AX = \*p1; (via BX)**

mov bx,[p1]

mov ax,[bx]

**// AH = enreg.champ2; (adressage basé sur BX avec déplacement)**

mov bx,offset enreg // bx = &enreg;

mov ah,[bx+2] // champ2 : déplacement de 2 octets

**// AX = i; (adressage indexé par SI)**

mov si,offset i // si = &i

mov ax,[si] // adressage basé sur si

**// AL = enreg.champ3; (adressage indexé par SI avec déplacement)**

mov si,offset enreg // si = &enreg

mov al,[si+3] // champ3 : déplacement de 3 octets

**// AH = tab\_byte[ i ]; (adressage basé sur BX et indexé par SI)**

mov bx,offset tab\_byte // bx = &tab\_byte[0]

mov si,[i] // si = i

mov ah,[bx+si]

**// AL = tab\_enr[ 4 ].champ3; (adressage basé, indexé avec déplacement)**

mov bx,offset tab\_enr // bx = &tab\_enr[0]

mov si,16 // si = 4\*4

mov al,[bx+si+3]

**// AH = tab\_enr[ i ].champ3; (adressage basé, indexé avec déplacement)**

mov si,[i] // si = i

add si,si // si = 2\*si

add si,si // si = 2\*si

mov ah,[bx+si+3]

}

}

int main() {

**// initialiser les variables**

variable1 = 3;

variable2 = 4;

variable3 = 5;

enreg.champ2 = 6;

i = 7;

enreg.champ3 = 8;

tab\_byte[ i ] = 9;

tab\_enr[ 4 ].champ3 = 10;

tab\_enr[ i ].champ3 = 11;

**// appeler la procédure**

proc();

return 0;

}

* Que fait l'instruction ADD SI,SI ? ***une multiplication par 2 donc deux* add si,si *réalisent une multiplication par 4***