Faculté d'Ingénieurs en Informatique, Multimédia, Systèmes, Télécommunication et Réseaux

Master en Génie Logiciel

TP Assembleur

Préparé par Elie MATTA

TP1(1)

```
//ce programme affiche 8

#include "stdafx.h"
#include "conio.h"

void ecrire(int i){
	printf("%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
	__asm{
		mov eax,8;
		push eax;
		call ecrire;
		pop eax;
	}
	getch();
	return 0;
}
```

TP1 (2)

```
// analyser le resultat affiche suite a l'execution de ce programme
#include "stdafx.h"
#include "conio.h"
void ecrire(int i){
	printf("%d ",i);
}

int _tmain(int argc, _TCHAR* argv[])
{
	__asm{
		push esp;
		push esp;
		push esp;
		call ecrire;
		pop eax;
		call ecrire;
		pop eax;
		call ecrire;
		pop eax;
	}

  getch();
	return 0;
}

//on a empiler la valeur 1244828 sur 4 octets car esp est code sur 4 octets car 32 bits
// on a empiler la valeur 1244824 et esp devient 1244820.
// on empile la valeur 1244820
//on applique push et ecrire sur les 3 valeurs donc on obtient :
// 1244820   1244824   1244828
```

TP1(3)

// analyser le resultat affiche suite a l'execution de ce programme

```
#include "stdafx.h"
#include "conio.h"

void ecrire(int i){
        printf("%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
        __asm{

inst1:  push inst1;     //inst adresse de l'instruction
inst2:  call ecrire;
inst3:  pop eax;
inst4:  push inst2;
                call ecrire;
                pop eax;
                push inst3;
                call ecrire;
                pop eax;
                push inst4;
                call ecrire;
                pop eax;
        }
        getch();
        return 0;
}
//resultat:4264974    4264979     4264984     4264985
//4264974 :adresse de l'instruction push inst1
//4264979 :adresse de call ecrire codee sur 5 octets
//4264984 :adresse pop eax
//4264985 :adresse push inst2
```

TP2

```
// tp2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "conio.h"
const char Hexad[17]="0123456789ABCDEF";
void EcrireHex(unsigned int i)
{
	int car[8];
	unsigned int j=i;
	for(int k=0;k<8;k++)
	{
		j=j/16;
		car[k]=Hexad[i-j*16];
		i=j;
	}
	for(int k=7;k>=0;k--)
		printf("%c",car[k]);
	printf(" ");
}
int _tmain(int argc, _TCHAR* argv[])
{
		__asm{
inst1:  push inst1;//inst1 est l'adresse de l'instruction push inst1
inst2:  call EcrireHex;// ecrire l'adresse de inst1
inst3:   pop eax;//depiler
inst4:   push inst2;//empilere l'adresse inst2
	call EcrireHex;//ecrire
		pop eax;// depiler
		push inst3;//empilere l'adresse inst3
		call EcrireHex;//ecrire
		pop eax;//depiler
		push inst4;//empilere l'adresse inst4
		call EcrireHex;//ecrire
		pop eax;//depiler
	}
```

```
        getch();
        return(0);
}

// cette procedure nous donne les adresse mais sous froem hexadecimal.
```


TP3(1)

---

faire une procedure qui afffiche EFLAGS

```
// tp2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "conio.h"
const char Hexad[17]="0123456789ABCDEF";
void EcrireHex(unsigned int i)
{
	int car[8];
	unsigned int j=i;
	for(int k=0;k<8;k++)
	{
		j=j/16;
		car[k]=Hexad[i-j*16];
		i=j;
	}
	for(int k=7;k>=0;k--)
		printf("%c",car[k]);
	printf(" ");
}

void EcrireBin(unsigned __int32 i)
{
	int car[32];
	unsigned int j=i;
	for(int k=0;k<32;k++)
	{
		j=j/2;
		car[k]=Hexad[i-j*2];
		i=j;
	}
	for(int k=31;k>=0;k--)
		printf("%c",car[k]);
```



```
        printf(" ");
}
void EcrireEFLAGS(unsigned __int32 i){
        printf("\n--------------------VDI-SZ-A-P-C\n");
        EcrireBin(i & 0xED5);
        printf("\n");
}
int _tmain(int argc, _TCHAR* argv[])
{
        __asm{
                mov eax,0xFFFFFF05;
                push eax;
                pushfd;
                add eax,0xFF;
                pushfd;
                push eax;
                call EcrireHex;
                pop eax;
                call EcrireEFLAGS;
                pop eax;
                call EcrireEFLAGS;
                pop eax;
                call EcrireHex;
                pop eax;
        }
   getch();
        return(0);
}
```

TP3(2)

```
// tp2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "conio.h"
const char Hexad[17]="0123456789ABCDEF";
void EcrireHex(unsigned int i)
{
	int car[8];
	unsigned int j=i;
	for(int k=0;k<8;k++)
	{
		j=j/16;
		car[k]=Hexad[i-j*16];
		i=j;
	}
	for(int k=7;k>=0;k--)
		printf("%c",car[k]);
	printf(" ");
}
int _tmain(int argc, _TCHAR* argv[])
{
		__asm{
inst1:  push inst1;//inst1 est l'adresse de l'instruction push inst1
inst2:  call EcrireHex;// ecrire l'adresse de inst1
inst3:   pop eax;//depiler
inst4:   push inst2;//empilere l'adresse inst2
	call EcrireHex;//ecrire
		pop eax;// depiler
		push inst3;//empilere l'adresse inst3
		call EcrireHex;//ecrire
		pop eax;//depiler
		push inst4;//empilere l'adresse inst4
		call EcrireHex;//ecrire
		pop eax;//depiler
	}
```

```
        getch();
        return(0);
}

// cette procedure nous donne les adresse mais sous froem hexadecimal.
```


TP4(1)

```
#include "stdafx.h"
#include "conio.h"

void EcrireChaine(char*a){
                printf("%s\n",a);
}
int _tmain(int argc, _TCHAR* argv[])
{
char string[6]="texte";
char string2[6]="abcde";
char*a;
char*b;
a=string;
b=string2;
char string3[6]="egaux";
char string4[10]="different";
char*c;
char*d;
c=string3;
d=string4;
printf("\n");
__asm{
        push a;
        call EcrireChaine;
        pop eax;
        push b;
        call EcrireChaine;
        pop eax;
        mov esi,a;
        mov edi,b;
        mov ecx,5;
        inc ecx;//la chaine se termine par zero "0" c-a-d dans cet exmeple la chaine a
        cld;    // est comme ca : texte0 et la chaine b est comme ca abcde0
        repe cmpsb;
        cmp ecx,0;
        je adr1;
push d;
```



```
call EcrireChaine;
pop eax;
jmp fin;
adr1: push c;
        call EcrireChaine;
        pop eax;
fin:
}
getch();
return 0;
}
```


TP4(2)

```
#include "stdafx.h"
#include "conio.h"

void EcrireChaine(char*a){
                printf("%s\n",a);
}
int _tmain(int argc, _TCHAR* argv[])
{
char string[6]="texte";
char*a;
a=string;
char string3[7]="existe";
char string4[13]="n'existe pas";
char*c;
char*d;
c=string3;
d=string4;
printf("\n");
__asm{
        mov AL,'v';
        mov edi,a; //on utilise edi mais non pas esi pour la comparaison dans scasb car
        mov ecx,5; //scasb compare edi a al, ax ou eax
        inc ecx;//la chaine se termine par zero
        cld;
        repne scasb;

        cmp ecx,0;
        ja adr1;
        push d;
        call EcrireChaine;
        pop eax;
        jmp fin;
adr1: push c;
        call EcrireChaine;
        pop eax;
fin:
}
```



```
        getch();
        return 0;
}
```


TP5(1)

Trouver les initiales d'une chaine:
Exemple: "Faculte de genie" -> "Fdg"

```
// tp5.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "conio.h"
void EcrireChaine(char*a){
                        printf("%s\n",a);
}

int _tmain(int argc, _TCHAR* argv[])
{
char string[17]="Faculte De Genie";
char*a;
a=string;
char string1[4]="   ";
char*b;
b=string1;
__asm{
mov esi,a; //on pointe esi au debut de la chaine de caracter a
mov edi,b; //on pointe edi au debut de la chaine de caracter b
mov al,[esi]; //F dans al
stosb; // al dans edi donc F dans edi
push edi; //F dans la pile
mov edi,a; //on pointe edi au debut de la chaine de caracter a
mov ecx,16;
repeter: mov al,' ';
repne scasb; //comparaison entre le contenu de edi et le contenu de al en decrementant ecx
mov al,[edi]; //repne saute lorsque [edi]=' ' donc on est arrive a un space donc on le stock dans
al
mov edx,edi; //pour ne pas perdre la place de edi
pop edi; //on met ce qu'il y a dans la pile dans edi
stosb; //on met al dans edi
push edi; //puis on met edi dans la pile
mov edi,edx; //on reprend le chemin de edi et on continue la comparaison
```



```
cmp ecx,0; //si on est arrive a la fin de la chaine de caracter a
je fin; //on saute a fin
jmp repeter; //sinon on repete repeter
fin:
pop edi; //on met ce qui est dans la pile dans edi
xor al,al; //al =0
stosb;// pour avoir 0 a la fin de la chaine de caractere
push b; //on met le resultat dans b
call EcrireChaine;// b = FDG
pop eax;
}
        getch();
        return 0;
}
```


TP5(2)

Trouver la taille d'une chaine donee (sachant que la taille maximale est fixee a 255)

code par Elie Matta
==========

```
#include "stdafx.h"
#include "conio.h"

void ecrire(int i){
        printf("%d ",i);
}

int _tmain(int argc, _TCHAR* argv[])
{
char string[15]="facultedegenie";
char*a;
a=string;
__asm{
mov edi,a; //on pointe edi au debut de la chaine de caracter a
mov ecx,255;

repeter:
mov al,'\0';
scasb; //comparaison entre le contenu de edi et le contenu de al et incrementation de edi
je adr1;
dec ecx;
jmp repeter;

adr1:
sub ecx,255;
neg ecx;
push ecx;
call ecrire;
pop ecx;
jmp fin;

fin:
```



```
}
        getch();
        return 0;
}
```


TP5(3)

Compter le nombre de fois qu'un caractere donne apprait dans une chaine : Bonjour et o

```
#include "stdafx.h"
#include "conio.h"

void ecrire(int i){
	printf("%d ",i);
}


int _tmain(int argc, _TCHAR* argv[])
{
	char string[5]="aboc";
char*a;
a=string;
__asm{

xor edx,edx; //edx=0
mov edi,a;
mov ecx,255; //on met ecx = 255 car on assume qu'on ne sait pas la longueur de la chaine de
caracter
mov al,0;
cld;
repne scasb; //en decrementant ecx de 255 jusqu'a arrive a 0
neg ecx; //le complement du resultat (ici ecx =250, mais on le met a -250 pour l'ajouter a 255)
add ecx,255; //on l'ajoute 255 pour avoir la taille de la chaine de caractere
mov edi,a;
mov al,'o';
repeter:
		repne scasb;
		cmp ecx,0;
		je fin;
		inc edx;
		jmp repeter;

fin:
```



```
                push edx;
                call ecrire;
                pop edx;
}
        getch();
        return 0;
}
```

==========

code par Elie Matta
==========

```
#include "stdafx.h"
#include "conio.h"

void ecrire(int i){
        printf("%d ",i);
}

int _tmain(int argc, _TCHAR* argv[])
{
char string[8]="bonjour";
char*a;
a=string;
__asm{
xor ebx,ebx;
mov ecx,7;
mov al,'o';
mov edi,a;
repeter:
repne scasb;
cmp ecx,0;
je adr1;
inc ebx;
jmp repeter;

adr1:
push ebx;
```



```
call ecrire;
pop ebx;

fin:
}
        getch();
        return 0;
}
```


TP5(4)

Trouver si une chaine de deux caracteres existe dans une autre chaine

==========

```
// tp.assembleur.B.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "conio.h"

void ecrirechaine (char* a)
{
	printf("%s\n",a);
}

void existe()
{
	printf("Existe\n");
}

void existepas()
{
	printf("Existe Pas\n");
}

int _tmain(int argc, _TCHAR* argv[])
{
	char string[10]="Bonjour";
	char *a;
	a=string;
	printf("\n");
	__asm{
				push a;
				call ecrirechaine;
				pop eax;
				mov edi,a;
				mov ecx,8;
				cld;
```

```
        boucle:
                        mov al,'j';
                        repne scasb;
                        mov al,'o';
                        cmp [edi],al;
                        je trouver;
                        cmp ecx,0;
                        je pastrouver;
                        jne boucle;
        trouver:
                        call existe;
                        jmp fin;
        pastrouver:
                        call existepas;
                        jmp fin;
        fin:
        }
        getch();
        return 0;
}
```


TP5(5)

Supprimer un caractere donne d'une chaine donnee
==========

```
// assembleur.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "conio.h"

void EcrireChaine( char* a)
{
	printf("%s \n",a);
}

int _tmain(int argc, _TCHAR* argv[])
{
	char string[8]="bonjour";
	char *a;
	char *b;
	a=string;
	b=string;
	__asm{
		mov esi,a;
		mov edi,b;
		push edi;
		mov edi,a;
		mov ecx,8;
repeter:
		mov al,'o';
		repe scasb;
		mov al,[edi-1];
		mov edx,edi;
		pop edi;
		stosb;
		push edi;
		mov edi,edx;
		cmp ecx,0;
```



```
                je fin;
                jmp repeter;
fin:
                pop edi;
                xor al,al;
                stosb;
                push b;
                call EcrireChaine;
                pop eax;
        }
        getch();
        return 0;
}
```


TP5(6)

Ecrire le programme permettant d'afficher les 50 premiers nombres premiers

==========

```
#include "stdafx.h"
#include "conio.h"

void ecrire(int i){
	printf("%d ",i);
}


int _tmain(int argc, _TCHAR* argv[])
{
	char string[5]="aboc";
char*a;
a=string;
__asm{
mov bl,1; //on met bl=1
movzx eax,bl; //extension de bl a eax (de 8 bit a 32 bits)
push eax; //on met eax dans la pile
call ecrire; //on affiche 1
pop eax; //on desempile ce qu'il y a dans la pile et on le met dans eax
inc bl; //on increment bl de 1
movzx eax,bl; //extension de 8 bit a 32 bits
push eax;
call ecrire; //on affiche 2
pop eax;
inc bl; //bl =3
movzx eax,bl;
push eax;
call ecrire; //on affiche 3
pop eax;
mov ecx, 50-3; //ecx = 47

Tester: add bl,2; //on ajoute 2 a bl
		mov dl,bl; //on met bl dans dl
		shr dl,1; //on divise dl par 2 (SHR on divise par 2, SHL on multiplie par 2)
```



```
modulos: movzx ax,bl; //on a mit bl dans ax pour
                div dl; //la diviser sur dl car div divise ax sur le registre qui suit la commande div
(le nombre ici est dl)
                cmp ah,0; //comparaison entre ah et 0 car le reste sera mit and ah
                je PasPremier; //s'il ya un reste donc ce n'est pas un nombre premier
                cmp dl,3; //comparaison entre dl et 3
                jbe NbPremier; //si dl est plus petit ou egale a 3 donc c'est un nombre premier
                sub dl,2; //sinon on retranche 2 de dl et on retourne a modulos
                jmp modulos;

NbPremier:      movzx eax,bl;
                push ecx;
                push eax;
                call ecrire; //affichage du nombre premier
                pop eax;
                pop ecx;
                loop Tester;
                jmp fin;

PasPremier: jmp Tester;

fin:



}
        getch();
        return 0;
}
```

