TP ASSEMBLEUR

TP1

Introduction:

On propose la structure suivante pour insérer un programme assembleur en C++:

```
#include "stdafx.h"
#include "conio.h"
void Ecrire (int i)
{
printf ("i=%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
//Ecrire(12);
__asm {
      mov eax,12;
      push eax;
      call Ecrire;
      pop eax;
}
getch();
return 0;
}
```

La procédure Ecrire en C++ permet de contourner, au début, les difficultés de programmer les commandes d'Entrées/sorties en assembleur.
Les 4 instructions en assembleur permettent d'appeler Ecrire en remplaçant la fonction "Ecrire(12);" de C++.
Analyser le résultat affiché suite à l'exécution du programme suivant.

La pile:

Analyser le résultat affiché suite à l'exécution du programme suivant:

```
#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;
}
```

Les codes opération (opcodes)

En analysant le résultat de l'exécution du programme suivant:

```
#include "stdafx.h"
#include "conio.h"
void Ecrire (int i)
{
printf ("%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
__asm {
inst1: push inst1;
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;
}
```

a. Trouver, en nombre d'octets, la taille de chacune des 3 premières instructions.
b. En déduire la longueur de l'instruction 4
c. Afficher les adresses en Hexadécimal en se servant de la procédure suivante:

```
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 (" ");
}
```

d. Sachant que la séquence suivante permet d'afficher les 4 octets de l'instruction1

Pointée par inst1 : `inst1 :push inst1` et instruction2 pointée par inst2 : `inst2 :push inst2` et que le stockage des données s'effectue en commençant par le poids faible, écrire le code et interpréter le résultat pour déduire le code opération de l'instruction push. Dresser le contenu de la pile et de la memoire:

En utilisant le code suivant:

```
inst1 :push inst1
inst2 :push inst2
call EcrireHex;
pop eax;
call EcrireHex;
pop eax;

  mov eax,inst1;
  mov eax,[eax];
  push eax;
  call EcrireHex;
  pop eax;

  mov eax,inst2;
  mov eax,[eax];
  push eax;
  call EcrireHex;
  pop eax;
```

deduire le contenu de la memoire.

Interpreter les resultats du code suivant :

1-

```
  mov eax,inst1;

  //-------------------------------------------
  //mov eax, [eax+4];

  //-------------------------------------------

  //-------------------------------------------


  push eax;
  call EcrireHex;
  pop eax;
```

2-

```
 mov eax,inst1;
 movzx ebx, byte ptr [eax+4];
  push ebx;
  call EcrireHex;
  pop eax;
```

3-

```
  mov eax,inst1;
  movzx edx, byte ptr [eax+5];
  push edx;
  call EcrireHex;
  pop eax;
```

4-

```
 mov eax,inst1;
 movzx ebx, word ptr [eax+4];
  push ebx;
```

```
        call EcrireHex;
        pop eax;
```

e- Determiner la longuere des instructions suivantes add eax,10, add eax,15 et determiner le contenu de la mémoire ainsi que le code operation de l'addition. Repeter avec add al,10 ;add al,15. Conclure.

TP2

Introduction:

On rappelle la structure du squelette nécessaire pour insérer un programme assembleur en C++:

```
#include "stdafx.h"
#include "conio.h"
void Ecrire (int i)
{
printf ("%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
//Ecrire(12);
__asm {
mov eax,12;
push eax;
call Ecrire;
pop eax;
}
getch();
return 0;
}
```

On rappelle aussi la procédure d'affichage d'un nombre hexadécimal:

```
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 (" ");
}
```

Et la procédure permettant d'afficher un nombre positif ou négatif est la suivante :

```
void EcrireAvecSigne(int i,int j)
{
    if (j==0)
    {
    printf("i=%d\n",i);
    }
    else
    {
        printf("i=%d\n",-i); }}
```

Le registre indicateur 1:

Pour afficher le registre indicateur on l'empile d'abord dans la pile à l'aide de L'instruction *"pushfd"* et puis on utilise les procédures suivantes pour l'affichage:

```
void EcrireBin(unsigned __int32 i)
{
int car[32];
unsigned __int32 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--------------------ODI-SZ-A-P-C\n");
EcrireBin(i & 0xED5);
printf("\n");
}
```

a. En utilisant la procédure `EcrireAvecSigne`, écrire le programme qui affiche la valeur `0xFFFFFF05.(-251). Pourquoi a-t-on besoin de la fonction EcrireAvecSigne. Proposer autres méthodes utilisant la fonction Ecrire.`

b. Ecrire le programme suivant et montrer que *"mov"* n'affecte pas le registre indicateur. Utiliser une pile afin d'interpréter les indicateurs après l'exécution de l'addition et de la soustraction:

```
0-pushfd
1-mov eax,0xFFFFFF05; (-251)
2-push eax;
3-pushfd;
4-add eax,0x000000FF;(255)
5-pushfd;
6-push eax;
7-sub eax,4;//eax=eax-4
8-pushfd;
9-push eax;
10- Add eax, 0Xffffff05;eax=0+(-251)=-251
11- pushfd;
12- Push eax

13- call Ecrire;
14- pop eax;
15- call EcrireEflags;
16- pop ebx;
17- call Ecrire;
18- pop eax;
20- call EcrireEflags;
21- pop eax;
```

```
22- call Ecrire ;
23- pop eax;
24- call EcrireEflags;
25- pop eax;
26- call EcrireEflags;
27- pop eax;
28- call Ecrire;
29- pop eax;
30- call EcrireEflags;
```

c. Ajouter les instructions permettant d'additionner 0x FFFFFF05 à la valeur finale de eax et d'afficher le registre indicateur. Interpréter le résultat affiché.

NB : Dans les questions b et c il est intéressant de dresser le contenu de la pile

Le registre indicateur 2:

En utilisant le contenu du registre eflag, interpréter les codes suivant :

a-

```
mov eax,0;
add eax,0;
pushfd;
call EcrireEflags;
popfd;
```

b- Donner un autre exemple permettant de mettre l'indicateur P à 1.

c- <0+<0=>0

```
mov eax,0x80000006;
//push eax;
//call Ecrire;
//pop eax;
mov ebx,0x80000001;
//push ebx;
//call Ecrire;
//pop ebx
add eax,ebx;
push eax;

//call disp;

call Ecrire;
pop edx;
jmp fin;
disp:
pushfd;
call EcrireEflags;
popfd
ret;
```

```
   fin:
```

d- >0+>0=<0

```
                    mov eax,0x80000006;
                    neg eax;
                    //push eax;
                    //call Ecrire;
                    //pop eax;
                    mov ebx,0x80000001;
                    neg ebx;
                    //push ebx;
                    //call Ecrire;
                    //pop ebx
                    add eax,ebx;
                    //push eax;
                    call disp;
                    //call Ecrire;
                    //pop eax;
                    jmp fin;
   disp:
          pushfd;
          call EcrireEflags;
          popfd;
          ret;
```

e-

```
                    mov eax, 0;
                    mov al, 0xf9;(-7, 249) 11111001
                    cmp al,0;
                    push eax;
                    pushfd;
                    call EcrireEflags;(S=1)
                    popfd;
                    pop eax;
                    push eax;
                    call EcrireHex; //ou bien call ecrire
                    pop eax;

                    mov eax, 0;
                    mov ax, 0xf9;
                    cmp ax,0;
                    push eax;
                    pushfd;
                    call EcrireEflags;
                    popfd;
                    pop eax;
                    push eax;
                    call EcrireHex;
                    pop eax;
                    jmp fin;
               fin:
```

- demonter dans la premiere partie du code comment les ruptures conditionnelle seront considerees.(JA, JAE, **JB**, JBE en ANS; JG, **JL**, JGE, JLE en AS)

)

f- En utilisant l'instruction

```
mov bl,253;
```

Interpreter cette instruction ?
Quelles instructions faut-il ajouter pour afficher correctement le contenue du registre eflag ?

g- Determiner les indicateurs C,**V**,**S**,**Z** et P et les valeurs des registres dans des instruction suivantes

```
mov dh,3 ;
add dh,2 ;
add dh,1;
sub dh,0x73;
add dh,0x79;
add dh,2;
add dh,0xFA;
add dh,3
```

TP3

Introduction:

On rappelle la structure du squelette nécessaire pour insérer un programme assembleur en C++:

```
#include "stdafx.h"
#include "conio.h"
void Ecrire (int i)
{
printf ("%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
//Ecrire(12);
__asm {
mov eax,12;
push eax;
call Ecrire;
pop eax;
}
getch();
return 0;
}
```

On rappelle aussi la procédure d'affichage d'un nombre hexadécimal:

```
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 (" ");
}
```

Et du nombre binaire:

```
void EcrireBin(unsigned __int32 i)
{
int car[32];
unsigned __int32 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 (" ");
}
```

Et du registre indicateur

```
void EcrireEflags (unsigned __int32 i)
{
printf("\n--------------------VDI-SZ-A-P-C\n");
EcrireBin(i & 0xED5);
printf("\n");
}
```

La rupture inconditionnelle de séquence:

Exécuter le programme suivant:

```
      mov eax,addret;
      push eax;
      call EcrireHex;
      pop eax;
      mov eax,200;
      call rout1;
addret:
      mov eax,10;
      jmp fin1;
rout1:
      push eax;
      //call Ecrire;
      pop eax;
      mov eax,[esp];
      push eax;
      //call EcrireHex;
      pop eax;
      ret;
fin1:
```

a. Interpréter le résultat tout en dressant le contenu de la pile avec les données stockées. Indiquer aussi les adresses de la pile auxquelles les données sont stockées.
b. Dans rout1, commenter *call EcrirHex*, et, dans addret commenter *mov eax,10* et *jmp fin* et les remplacer par **push eax** et **ret**. Interpréter le résultât.
c. Ecrire une procédure en assembleur permettant d'appliquer le complément à 1 (not …) au nombre à afficher et d'appeler la procédure rout1. Le transfert de données entre le programme principal et la procédure rout1 doit s'effectuer à travers la pile(c-a-d empiler le résultat du not eax avant d'appeler rout1). Utiliser **ebp pour regarder à l'intérieur de la pile la valeur du not eax**, et utiliser EcrireHex pour la afficher. N.B.:dans le cas d'appelle de plusieurs sous programme, rajouter **push ebp;** au début et **pop ebp;** à la fin pour que l'exécution du programme appelant Ass/ C++ ne soit pas perturbée. Faites attention et ajouter les instructions de désempilation nécessaires

La rupture conditionnelle de séquence:

Soit le code suivant:

```
                    mov al, 0xf9;
                    cmp al,0;


plg:
    push eax;
    call EcrireHex;
    pop eax;
    jmp fin;
```

1- Expliquer pourquoi les deux instructions JL plg et JA plg effectuent un branchement à plg. ?
2- Que ce passe-t-il si on rempalce al par ax ou eax ?. Expliquer.

Génération en assembleur du code C++ compilé:

Ouvrir un nouveau projet. Dans les propriétés du projet sous le volet *C/C++ output files* rajouter dans *assembler output* la directive /FA. Ecrire le programme C++ suivant:

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[])
{
      static __int8 i=12;
      int j=14;

      __asm
      {
        mov al,10;
        add al,20;
        add al,i;
        //operand size conflict
        //mov bl,j;
        mov  bl,byte ptr j;

        mov ebx,j;
        mov bl,15;
        add al,bl;
      }
return 0;
}
```

1- Regarder le fichier ".asm" généré dans le dossier *Debug* du projet. Essayer de trouver le code C++ et interpréter.

2- Que veut-il dire *xor eax,eax*?

TP4

Introduction:

On rappelle la structure du squelette nécessaire pour insérer un programme assembleur en C++:

```
#include "stdafx.h"
#include "conio.h"
void Ecrire (int i)
{
printf ("%d ",i);
}
int _tmain(int argc, _TCHAR* argv[])
{
//Ecrire(12);
__asm {
mov eax,12;
push eax;
call Ecrire;
pop eax;
}
getch();
return 0;
}
```

On considère la procédure suivante pour afficher une chaîne de caractères:

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

La manipulation des chaînes de caractères:

Exécuter le programme suivant:

```
#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;
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 chaîne se termine par zéro
cld;
rep movsb;
push a;
call EcrireChaine;
pop eax;
push b;
call EcrireChaine;
pop eax;
}
getch();
return 0;
}
```

1.Interpréter le code précédent

2. Ecrire le programme permettant de comparer deux chaînes de même taille.
Indications:

a- Comparer leur longueur. Si égale on continue.
b-Utiliser repe ou repne et cmpsb().
c- Selon la valeur finale de z on décide si les 2 chaînes sont égales.
d- répéter le même exercice tout en utilisant loope ou loopne

3. Rechercher un caractère dans une chaîne.
Indications:

a- Mettre le caractère dans AL.
b- Pointé par EDI sur le 1er caractère de la chaîne et mettre la taille dans ecx.
c- Utiliser repe ou repne avec scasb().
d- Décider selon ecx.

TP5

Les instructions arithmétiques.

1 :

Ecrire le code assembleur qui recherche tous les nombres entre 100 et 999 y compris ces deux valeurs, et détecte si ce nombre est égal à la somme des cubes de ses chiffres.
Par exemple 153=1^3+5^3+3^3.

Indications :

1- Essayer le programme pour un seul nombre et par suite rechercher tous les autres en utilisant une boucle.
2- Vous devez décomposer tout d'abord le nombre en ses composants.

2 :

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

TP6 suite chaines de caractères

1 (test2 1206)

Trouver les initiales d'une chaîne. Exemple: « Faculté de génie »

a- Les initiales sont à être affichées caractère par caractère, chacun sur une ligne comme suit :

F
d
g

b- Les initiales sont à être affichées sur la même ligne : Fdg

2 :

Compter le nombre de fois qu'un caractère donné apparaît dans une chaîne :
« oBonjouro » et 'o' => 4

3:

Trouver si une chaine de deux caractères existe dans une autre chaîne :
« Bonjour » et « on » alors afficher "existe"
Ajuster le programme pour qu'il recherche si les caractères de la chaine 'on' existent ailleurs dans la chaine (Bojnour),(Bojourn)

4:

Supprimer un caractère donné d'une chaîne donnée dans une autre chaine:
« Bonjour » et « j » alors le résultat est «Bonour »
Bonus : réaliser la suppression dans la même chaine