
TP n° 1
Assembleur

Pour l'ensemble du TP :

- on utilise l'assembleur nasm
- on travaille avec des entiers codés sur 32 bits
- les résultats en retour des fonctions sont stockés dans `eax`

Exercice 1 - Traduire le programme C suivant en assembleur Pentium (`nasm`) en faisant appel à la librairie C pour l'affichage des résultats. On supposera que les variables `a` et `b` ont été initialisées préalablement :

```
int main() {
    int a, b;

    if (a<b) {
        printf("a inferieur à b\n");
    } else {
        printf("a superieur ou egal à b\n");
    }
    return 0;
}
```

L'entête du programme `nasm` commencera par :

```
global main
extern printf

; ===== DATA SECTION =====
section .data

        a dd 1 ; a est un entier sur 32 bits
        b dd 2 ; b est un entier sur 32 bits
        msg_inf db "a inferieur à b", 10, 0
        msg_sup_egal db "a superieur ou egal à b", 10, 0

; ===== CODE SECTION =====
section .text

main:
    push ebp
    mov  ebp,esp
    pushad
    ....<code a traduire>...

    popad
    mov  esp,ebp
    pop  ebp
    xor  eax,eax
    ret
```

```
$ nasm -f elf -g tp1_1.nasm
$ gcc -ggdb -o tp1_1.exe tp1_1.o
```

On compile le programme grâce aux commandes suivantes :

- modifier le programme précédent afin d'afficher les valeurs de *a* et *b*
- modifier le programme précédent afin de demander à l'utilisateur de saisir les valeurs de *a* et *b* au clavier

Exercice 2 - Traduire le programme C `tp1_2.c` en assembleur nasm (`tp1_2.nasm`) :

```
// tp1_2.c
int main() {
    int i, sum=0, n;
    n=15;
    for (i = 0; i < n; ++i) {
        sum = sum + i;
    }
    printf("la somme est %d\n", sum);
    return 0;
}
```

Exercice 3 - Traduire le programme C `tp1_3.c` en assembleur nasm (`tp1_3.nasm`) :

```
// tp1_3.c
int tab[20];

int calc_sum(int t[], int n) {
    int i, sum = 0;

    for (i = 0; i < n; ++i) {
        sum = sum + t[i];
    }
    return sum;
}

int main() {
    int i;

    for (i = 0; i < 20; ++i) tab[i] = 2*i;
    printf("la somme des elements de tab est %d\n", calc_sum(tab,20));
    return 0;
}
```