

Objet du TP

Ce TP porte sur la manipulation de tableaux en assembleur Nios.

Démarrage

Comme pour le premier TP, un squelette de projet est disponible sur le *share*.

1. Copiez le répertoire **share:\I3info\ILM\TP\TP2** dans votre espace personnel, par exemple à l'emplacement **H:\ILM\TP\TP2**
2. Lancez Altera Monitor en exécutant le fichier : **share:\I3info\ILM\TP\monitor.bat**
3. Allez dans le répertoire que vous avez précédemment copié, et ouvrez le projet **TP2\TP2.amp** dans Altera Monitor.
4. Lorsqu'un dialogue vous propose de charger le processeur Nios II sur votre carte, vérifiez que celle-ci est branchée et allumée et appuyez sur **Oui**.

Partie 1 : Lecture / affichage d'un tableau

Vous allez écrire deux fonctions, *lectureTableau()* et *affichageTableau()*, dont les prototypes sont :

```
void lectureTableau(int *tableau, int taille);
void affichageTableau(int *tableau, int taille);
```

Ces fonctions permettront respectivement de lire et d'afficher le contenu d'un tableau d'entiers, étant donnés son adresse et le nombre d'éléments.

Question 1 :

Complétez la fonction *main()* pour invoquer ces deux fonctions :

```
int tableau[10]; // Déclaré dans le segment de données
void main() {
    print_string("Lecture du tableau.\n");
    lectureTableau(tableau, 10);
    print_string("Affichage du tableau.\n");
    affichageTableau(tableau, 10);
    exit();
}
```

Rappel : Par convention, les arguments sont passés via les registres **r4-r7**.

Question 2 :

Implémentez la fonction *lectureTableau()* en assembleur. Vous utiliserez la pile pour sauver les valeurs des registres utilisés selon les conventions vues en cours.

```
void lectureTableau(int *tableau, int taille) {
    int i;
    for (i=0; i<taille; i++) {
        print_string("Entrez un nombre:\n");
        tableau[i] = read_int();
    }
}
```

```
}
```

Question 3 :

Testez le bon fonctionnement de votre fonction en inspectant la mémoire après l'ajout de chaque élément. Pour ce faire, déterminez l'adresse du tableau puis utilisez l'onglet *Memory* de la zone principale pour inspecter la mémoire à cette adresse. Vous pouvez passer en affichage décimal en sélectionnant **Number format > Decimal** dans le menu contextuel (clic-droit).

Question 4 :

Implémentez la fonction ***affichageTableau()*** en assembleur :

```
void affichageTableau(int *tableau, int taille) {  
    int i;  
    for (i=0; i<taille; i++) {  
        print_int(tableau[i]);  
    }  
}
```

En exécutant votre programme, vérifiez que le tableau affiché est identique au tableau lu.

Partie 2 : Inversion de tableau

Question 1 :

Implémentez en assembleur la fonction suivante, permettant d'inverser le contenu d'un tableau :

```
void inversionTableau(int *tableau, int taille) {  
    int tmp;  
    int i=0, j=taille-1;  
    while (i<j) {  
        tmp = tableau[i];  
        tableau[i] = tableau[j];  
        tableau[j] = tmp;  
        i++; j--;  
    }  
}
```

Question 2 :

Modifiez votre programme pour invoquer cette fonction entre la lecture et l'affichage du tableau.

Question 3 :

Vérifiez le bon fonctionnement de votre programme.

Compte-rendu

Vous adresserez un compte-rendu de votre travail. Pour cela, vous devrez créer une archive contenant un seul répertoire nommé tp2. Cette archive inclura votre code assembleur commenté et un compte-rendu permettant d'apprécier votre travail :

- Un état de votre programme (fonctionnel ou non),
- En cas de difficultés, une description de celles-ci et de vos pistes de résolution,
- Une justification de vos choix d'implémentation.

Le nom de l'archive doit inclure les noms du binôme.

Pensez à inclure la mention ILM dans le champ « sujet » de votre mail.