TP1 : Représentation des données en mémoire et Opérations arithmétiques

Introduction

Ce TP utilise l'ordinateur (Processeur et Mémoire) implanté sur une carte FPGA pour exécuter des programmes écrits en langage assembleur. Ces programmes permettront d'observer

- L'implantation des données en mémoire
- L'exécution d'instructions arithmétiques par le processeur NIOS II et l'observation des cas de débordement

Mode d'emploi

1. Connecter la carte DE2 au terminal via le câble USB (entrée Blaster du côté de la carte)
2. Lancer le programme Quartus.
3. Via l'onglet Tool, ouvrir le programme « programmer ». Via la commande Open (sous l'onglet File), charger le fichier config.sof. Lorsque le fichier apparaît, cliquer sur Program/Configure. Cliquer sur l'onglet « Hardware Setup » et vérifier que « USB-Blaster » apparaît dans le fenêtre.
4. Cliquer sur Start pour charger la configuration sur le FPGA. Cette étape charge «le processeur NIOS et sa mémoire » sur la carte.
5. Le programme Quartus peut alors être fermé.
6. Lancer le programme « Altera Debug Client
7. Via l'onglet NIOS II, cliquer sur « Configure System » puis cliquer sur Load pour charger le fichier « nios_system.ptf ». Puis cliquer OK. Après cette phase, les logiciels permettant de compiler des programmes C ou des programmes assembleur sont chargés sur la carte DE2.
8. Via l'onglet NIOS II, avec Assembly pour Program Type, on peut ajouter (add) un programme assembleur.s.
9. Via l'onglet Action, cliquer sur « Compile&Load. ». Vérifier que les deux étapes s'exécutent correctement (pas d'erreur signalée).
10. L'onglet « Dissassembly » permet de voir le programme assembleur généré. Ce programme peut s'exécuter instruction par instruction via la commande « Single Step » ou commande F2. On peut sortir de l'exécution du programme assembleur par la commande « Disconnect ».
11. L'onglet « Memory » permet de visualiser le contenu des cases mémoire. Il est possible en cliquant sur les adresses de changer le mode d'affichage :1) « Number of words to display »(il est conseillé de choisir 8 pour faciliter la lecture des adresses hexadécimale).2° « View as » qui permet de voir le contenu des cases mémoire par mots de 8 bits (octet), 16 bits (2 octets) ou 32 bits (4 octets).

Représentation des données en mémoire

Exécuter successivement les programmes TP1_memoire.s, TP1_memoire1.s et TP1_memoire2.s

1. TP1_memoire : voir l'implantation mémoire. Quel ordre utilise le processeur NIOS (big ou little endian) ?
2. TP1_memoire1 : quel est le problème ?
3. TP1_memoire2 : en mode « dissassembly », exécuter pas à pas et voir le résultat d'exécution des différentes instructions Load.

Instructions arithmétiques

1. Exécuter le programme TP1_add.s pas à pas et voir le résultat d'exécution des instructions arithmétiques.
2. Modifier le programme TP1_add.s en utilisant des instructions de décalage sur un registre contenant un entier positif et un registre contenant un entier négatif. Les instructions de décalage sont définies en annexe.

Annexe 1.

Programme TD1_memoire.s

```
.include "nios_macros.s"
.text
.global _start
_start:
STOP:
      br          STOP
.org 0xe00
.word 1, 2, 3, 4, 5
.byte 'a', 'b', 'c', 'd', 'A', 'B', 'C', 'D','0','1'
.short 10, 11, 12
.float 1, 2, 3
.end
```

Le programme commence à l'adresse start. Dans ce cas, il ne fait rien.
Les données sont placées en mémoire à partir de l'adresse spécifiée par org (0xe00)

Annexe 2

Instructions assembleur de décalage

Instruction assembleur	Effet	
slli rC, rA, IMM5	rC ←rA << IMM5	Décalage logique gauche
srai rC, rA, IMM5	rC ←rA >> IMM5	Décalage arithmétique droite
srli rC, rA, IMM5	rC ←rA >> IMM5	Décalage logique droite

IMM5 = constante sur 5 bits.