TD 7 – Architecture logicielle

Branchements et boucles

Dans tout le TD, on considère le jeu d'instructions NIOS – II.

Format des instructions

Les branchements utilisent le format I. Quelle est l'amplitude des branchements ?
Les déplacements sont signés sur 16 bits. L'amplitude est donc de 2^{16} octets, soit CP+4 + déplacement, avec déplacement entre -2^{15} et $2^{15} – 4$ (car les adresses doivent être alignées)

Les instructions JMPI et CALL utilisent le format J. Quelle est l'amplitude des adresses atteignables par ces instructions ?
L'adresse est calculée de la manière suivante : CP31-28 | 26 bits|00, ce qui donne une amplitude de 2^{28} octets

Conditionnelles

a) Ecrire la séquence d'instructions qui réalise l'opération suivante : r3 = max (r1,r2)

```
     ADD r3,r0,r1
     BGT r1,r2, fin
     ADD r3,r0,r2
fin :
```

b) Ecrire une séquence d'instructions qui effectue la comparaison de deux entiers en complément à 2 sur 64 bits. Les opérandes sont contenus dans (r3,r2) et (r5,r4), les mots de poids fort étant respectivement dans r3 et r5. Le résultat est retourné dans r6, avec r6=1 si r3 :r2 ≤ r5 :r4 et r-=0 sinon.

Algorithme

Si R3 < R5 alors vrai
Sinon si R3> R5 alors faux
Sinon résultat = R2 ≤ R4

```
CMPLT r6,r3,r5      r6 = 1 si r3 < r5
BGT r6,r0, suite
CMPLE r6,r5,r3      r6 =0 si r3 > r5
BEQ r6,R0, suite
CMPLEU r6,r2,r4     r3=r5 et r6 = 1 si r2≤r4
```

Suite :

Boucle while

Ecrire le code assembleur correspondant au fragment de programme suivant

```
     while (a >b)
          a=a-b ;
```

om a et b sont des entiers non signés implantés consécutivement à partir de l'adresse 0x00001000.

```
     ORI r1,r0, 01000
```

```
     LDW r2,0(r1)
     LDW r3,4,(r1)
Boucle : BLEU r2,r3, suite
     SUB r2,r2,r3
     BEQ r0,r0, Boucle
```

Boucle Repeat

Boucle B1

```
     For (i=0 ; i<1000 ; i++)
          s= s + X[i] + Y[i] ;
```

où X et Y sont des vecteurs d'entiers implantés respectivement à partir des adresses 0x10000000 et 0x20000000 et s est un entier placé à l'adresse 0x00000100. On écrira d'abord le corps de la boucle, puis le test de sortie, puis les initialisations.

```
     XOR r5,r5,r5
     ORHI r1,r0, 0x1000
     ORHI r2,r0, 0x2000
     ADDI r6,r1, 3996
BCL :LDW r3,0(r1)
     LDW r4, 0(r2)
     ADD r5,r5, r3
     ADD r5,r5,r4
     ADDI r1,r1,4
     ADDI r2,r2,4
     BLE r1,r6,BCL
     STW r5, 0x100(r0)
```

Boucle 2

```
     For (i=1 ; i<1000 ; i++)
          Y[i-1]= X[i] + X[i-1] ;
```

où X et Y sont des vecteurs d'entiers implantés respectivement à partir des adresses 0x10000000 et 0x20000000 et s est un entier placé à l'adresse 0x00000100. On écrira d'abord le corps de la boucle, puis le test de sortie, puis les initialisations.

```
     ORHI r1,r0, 0x1000
     ORHI r2,r0, 0x2000
     ADDI r6,r1, 3992         ; déplacement du dernier élément
     LDW r3, 0(r1)            ; r3 = X[i-1]
BCL :LDW r4, 4(r1)            ; r4 = X[i]
     ADD r5,r4,r3             ; r5= X[i-1] + X[i]
     STW r5, 0(r2)            ; Y[i-1] = R5
     ADD r3,r0,r4             ; préparation itération suivante
     ADDI r1,r1,4
     ADDIr2,r2,4
     BLE r1,r6, BCL
```

Recherche de caractères

Ecrire un programme qui recherche la première occurrence de la valeur 0x41 dans un tableau de 10 caractères. Le résultat est l'indice (à partir de 0) de l'élément du tableau correspondant, et -1 si pas trouvé. Il est retourné dans le registre r1. Le tableau est implanté à partir de l'adresse 0x00001000

```
     XOR r1,r1,r1
     ORI r7,r0, 0x1000
     ORI r3,r0,0x41
     ADDI r5,r7,0x9
BCL :LDB r4,0(r7)
     BEQ r3,R4, trouve
     ADDI r1,r1,1
     ADDI r7,r7,1
     BLE r1,r5, BCL
     ADDI r1,r0,-1
trouve :
```