TD8 : – Architecture logicielle :
Conditionnelles et boucles

Dans tout le TD, on considère le jeu d'instructions MIPS32. (On supposera que les branchements et les sauts sont « normaux », c'est-à-dire non retardés).

1. Format des instructions

Les branchements utilisent le format I. Quelle est l'amplitude des branchements ?
L'instruction J utilise le format J. Quelle est l'amplitude des adresses atteignables par ces instructions ?

Type	-31- format (bits) -0-					
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	C (16)		
J	opcode (6)	address (26)				

Jump	j C	PC =PC[31:28] | C*4	J	0x2	-	Unconditionally jumps to the instruction at the specified address.
Jump register	jr $s	goto address $s	R	0x0	0x8	Jumps to the address contained in the specified register
Branch on equal	beq $s,$t,C	if ($s == $t) go to PC+4+4*C	I	0x4	-	Goes to the instruction at the specified address if two registers are equal.
Branch on not equal	bne $s,$t,C	if ($s != $t) go to PC+4+4*C	I	0x5	-	Goes to the instruction at the specified address if two registers are not equal.
Branch on greater than zero	bgtz $s,C	if ($s > 0) go to PC+4+4*C	I	0x7		Goes to the instruction at the specified address if the register content is > 0.
Branch on greater or equal zero	bgez $s,C	if ($s ≥0) go to PC+4+4*C	I	0x1	Rt=1	Goes to the instruction at the specified address if the register's content is ≥ 0.
Branch on less than zero	bltz $s,C	if ($s < 0) go to PC+4+4*C	I	0x1	Rt=0	Goes to the instruction at the specified address if the register's content is < 0.
Branch on greater or equal zero	blez $s,C	if ($s ≥0) go to PC+4+4*C	I	0x6		Goes to the instruction at the specified address if the register's content is greater than 0.
Set on less than	slt $d,$s,$t	$d = ($s < $t)	R	0	0x2A	Tests if one register is less than another.

Set on less than immediate	slti $t,$s,C	$t = ($s < C)	I	A_{16}	-	Tests if one register is less than a constant.

Move conditional on Not Zero	movn $d,$s,$t	if $t≠0, $d=$s	R	0	0x13	MIPS IV
Move conditional on Zero	movz $d,$s,$t	if $t=0, $d=$s	R	0	0x12	MIPS IV

2. Conditionnelles

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

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 sir r3 :r2 ≤ r5 :r4 et r-=0 sinon.

3. Boucle while

Ecrire le code assembleur correspondant au fragment de programme suivant

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

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

4. Boucle For

Boucle 1

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

a) 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.

b) Reprendre a) en considérant que les deux vecteurs X et Y sont rangés successivement en mémoire à partir de l'adresse 0x10000000

Boucle 2

```
For (i=0 ; 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.

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.