TD sur l'architecture du MIPS

1 - Codage des instructions assembleur

En utilisant les documents sur le codage des instructions du MIPS, donner le codage de chacune des instructions du programme assembleur suivant sachant que l'étiquette Exit correspond à l'adresse 0x400020 (et main à l'adresse 0x400004).

```
main:    add  $t1, $s3, $s3
         add  $t1, $t1, $t1
         lw   $t0, 0($t1)
         j    Exit
         . . .
Exit :
```

2 - Désassemblage

Retrouver les instructions assembleur correspondant au code machine ci-dessous :

1 0x00AF8020
2 0x23BD000C
3 0X0085402A

3 – Compilation

On considère le bout de code C suivant :

```
for (i = 0; i < 100; i++)
     h = h + c;
```

On suppose que le registre $t0 est associé à la variable i, que le registre $s0 est associé à la variable c et $s1 à la variable h. Réécrire cette instruction en C avec une boucle while et do-while. Donner le code assembleur équivalent pour le MIPS de ces trois programmes. Combien d'instructions sont exécutées pour cette boucle ? Combien y a-t-il d'accès à la mémoire ? Quelle est la forme la plus intéressante.

4 – Architecture du MIPS (extrait Exam Oct. 2002)

On considère l'architecture du chemin de données du processeur MIPS étudiée en cours et donnée sur le document joint en annexe. Sur ce document, les lettres A .. P sont associées à certains bus. Le but de cet exercice est d'indiquer les valeurs présentes sur ces bus pour 3 instructions assembleur.

Pour déterminer ces valeurs, on supposera que l'instruction présente sur le bus de sortie de la mémoire d'instructions (bus repéré avec la lettre A) reste aussi longtemps qu'on le souhaite, pour obtenir une valeur constante et stable sur les bus. On laissera donc le temps à toutes les unités de cette architecture d'effectuer leur opération.

On considère les 3 instructions suivantes (avec $a1 = 0x0000FF00 et $t7=0) :

```
Pgm:    add  $s0, $a1, $t7
        andi $s1, $s0, 15
        beq  $s1, $zero, Exit
        …
Exit:
```

L'étiquette Pgm est à l'adresse 0x00400024 et l'étiquette Exit est à l'adresse 0x00400040.

On pourra remplir le tableau ci-dessous. On respectera les notations conseillées, soit hexadécimale pour des valeurs de 16 à 32 bits, et binaire pour les autres. Vous indiquerez sur votre copie les différents calculs si nécessaire. Les valeurs sans importance pour l'instruction sont notées *inutile* dans le tableau.

	add$s0, $a1, $t7	andi $s1, $s0, 15	beq $s1, $zero, Exit
A (hexa)			
B (binaire)			
C (binaire)			
D (binaire)			
E (binaire)			
F (hexa)			
G (binaire)			ind
H (hexa)			
I (hexa)			
J (hexa)			
K (hexa)			
L (binaire)			
M (hexa)			ind
N (hexa)			
O (hexa)	inutile	inutile	
P (hexa)			