2016-2017

TP 1 -

On utilise le simulateur QtSpim présenté en annexe.
Les programmes sont disponibles sur le site du cours. Il est fortement conseillé de les imprimer et de les conserver : tous les documents de l'UE sont disponibles au partiel et à l'examen.

1. Prise en main de QtSpim

Charger le programme TP1Ex1.s. Le début du programme, jusqu'au syscall, correspond à la gestion de l'appel de procédure et ne sera pas étudié.

a. Exécuter le programme. Donner le contenu en hexadécimal des registres $t2, $t3 et $t4 après l'exécution.

$t2	
$t3	
$t4	

b. Comment sont traduites les instructions li $t0, 0x12345678 et li $t1, 0x00000001 ?

Li $t1, 0x12345678	
Li $t1, 0x00000001	

Justification

c. Modifier le code pour initialiser $t0 à 0x80000000 et $t1 à 0xFFFFFFFF. Donner le contenu en hexadécimal des registres $t2, $t3 et $t4 après l'exécution.

$t2	
$t3	
$t4	

Que constate-t-on en remplace ADDU par ADD et pourquoi ?

2. Comparaisons

Charger le programme TP1Ex2.s.

a. Donner une trace d'exécution pour $t0 = 0x12345678 et $t1 = 0x00000027, puis en inversant les initialisations.

	$t0=0x12345678 **$t1=0x00000027**			**$t0=0x00000027** **$t1= 0x12345678**		
Instruction MIPS	**$t2**	**$t3**	**$t4**	**$t2**	**$t3**	**$t4**
sub t2,t0,t1						

b. Que calcule ce programme ? (penser à écrire sur une feuille les différentes étapes du programme, et se rappeler que le XOR est associatif : a XOR (b XOR c) = (a XOR b) XOR c, (éventuellement, comment montrer que le XOR est associatif?)).

c. Donner une expression C correspondant à ce programme.

3. Moyenne de deux entiers

La moyenne entière de deux entiers naturels a et b peut se calculer ainsi :

$$\lfloor (a+b)/2 \rfloor = a\&b + (a\hat{}b)>>1$$

où &, ^, + et >> sont les opérateurs ET bit à bit, XOR bit à bit, additionnage et décalage.

a. Ecrire le code assembleur correspondant. Donner une trace d'exécution pour : a=7 et b= 3 ; a = 7 et b = 2 ; a = 0xFFFFFFFF et b = 1

Instruction MIPS	**Commentaire**
	Init a
	Init b
Jr $ra	Retour jr (fin de programme)

	a=7, b=3	
Instruction MIPS	**Registre modifié**	**Valeur**
	a=7, b=2	

Instruction MIPS	Registre modifié	Valeur
	a=0xFFFFFFFF, b=1	
Instruction MIPS	**Registre modifié**	**Valeur**

4. Addition en complément à un

Ecrire un code en MIPS32 qui réalise l'addition lorsque les nombres relatifs sont codés en complément à un. On ne s'intéressera pas aux problème sde dépassement (overflow).

Rappel

- en CCà1, pour écrire un nombre négatif on calcule d'abord sa représentation positive sur n-1 bits et on « retourne » tous les bits (0 → 1 et 0 → 1)
- l'addition en CCà1 entre un nombre positif et un nombre négatif est fausse lorsqu'il y a une retenue. Pour obtenir le bon résultat il faut sommer la retenue à la somme précédemment calculée.

Tester votre programme, notamment avec la représentation « négative » du zéro !

Annexe : QtSpim

QtSpim est un logiciel de simulation de l'exécution de programmes assembleur MIPS.

1. Présentation

Au lancement de QtSpim, une fenêtre s'ouvre.
Les menus permettent de sélectionner les affichages et de charger et exécuter une simulation. La fenêtre est divisée en différentes sections:

1. L'onglet *Int Registers* affiche le contenu de tous les registres entiers.
2. L'onglet *Text* affiche les instructions MIPS chargées en mémoire pour être exécutées. De gauche à droite, l'adresse mémoire de l'instruction, l'instruction en hexadécimal, les instructions MIPS réelles (pas de pseudo-instructions), l'assembleur MIPS que vous avez écrit ainsi que les commentaires.
3. L'onglet *Data* affiche les données et leurs valeurs dans les segments de données et la pile en mémoire.
4. La console information répertorie les actions effectuées par le simulateur.

Pour une meilleure lisibilité, décochez tout ce qui correspond à *Kernel* dans les menus *Text Segment* et *Data Segment*, et décochez *FP Registers* dans le menu *Window*.
Par un clic droit sur un registre ou une adresse mémoire, vous pouvez modifier son contenu dynamiquement.

4. Utilisation

Pour exécuter le programme dans QtSpim:

1. Utilisez un éditeur de texte pour créer un programme (ou utilisez un programme existant). Le programme doit avoir le suffixe ''.s ''. Votre code doit utiliser les noms de registres, précédés du signe '$'. Utilisez les registres $t0-$t7[1].
2. Utiliser le menu *File>Reinitialize and Load File* pour charger le programme en mémoire. Le simulateur inclut un programme d'assemblage, qui effectue l'encodage en binaire et la traduction des pseudo-instructions et des labels. Il vérifie la correction syntaxique du programme. Si le programme est incorrect, une fenêtre signalant la première erreur s'affiche et le programme ne peut pas être pas chargé.
3. Vous pouvez ensuite exécuter le programme, suivant l'un des trois modes suivants :
    a. Simulator>Run (ou touche F5) - toutes les instructions seront exécutées, et le contenu final de la mémoire et des registre seront visibles dans la fenêtre QtSpim.
    b. Simulator>Single Step (ou touche F10) – exécution pas à pas. Après l'exécution de chaque instruction, l'affichage est mis à jour.
    c. Création d'un point d'arrêt : click droit sur une instruction, et *Set Breakpoint*. Pour supprimer le point d'arrêt, click droit sur une instruction, et *Clear Breakpoint*
4. Pour modifier le programme, il faut revenir dans un éditeur de texte, effectuer les modifications et recharger le programme.

[1] Ceci sera expliqué dans la partie Procédures du cours.

5. Où obtenir QtSpim?

- Le simulateur est installé sur les machines de TP. Vous pouvez l'installer facilement sur un ordinateur personnel : des versions Linux, Windows et mac sont disponibles sur
  http://spimsimulator.sourceforge.net
- Le menu Help fournit
  - Une documentation sur l'usage du simulateur : *Help>QtSpim Manual*
  - La documentation complète : *Help>Assemblers, Linkers and the SPIM Simulator*