TP n°1, I45 – Projet *Emmanuel Bruno*
L2 Sciences et Technologies
Université du Sud Toulon-Var
Année universitaire 2005 – 2006

La Machine RISC

Objectif

L'objectif de ce TP est de programmer une machine RISC capable d'exécuter le code machine qui sera produit par notre compilateur. La structure du programme est imposée et une partie du code est fournie. Copier les fichiers (algo_risc.h, algo_risc.c, test_algo_risc.c et makefile) dans votre répertoire de travail :

Pour ce TP, vous travaillerez en éditant le fichier `algo_risc.c` qui vous est fourni en complétant les zones `/* A COMPLETER */` figurant dans le code. La fonctio principale (le "main") est contenue dans le fichier test_algo_risc.c, pour le compiler, il faut d'abord crée le fichier objet algo_risc.o. Cette procédure est automatisée avec le Makefile.

Bien sûr, avant de modifier le fichier source, lisez bien son contenu ainsi que celui du fichier `algo_risc.h` afin d'en comprendre le fonctionnement ! Pour chaque procédure que vous aurez à compléter sn fonctionnement est expliqué dans un commentaire. Ainsi, par exemple, vous verrez que la procédure `put` ajoute des instructions à un programme assembleur (dans le tableau `code`), la procédure `load` charge le code en mémoire, la procédure `execute` exécute ce code.

Attention : la mémoire est adressable en octets (cf. le TD) et composée de cellules de 4 octets, elle est simulée par un tableau d'`int (cf. algo_risc.h)` ; les indices du tableau `M` ne correspondent donc pas **directement** aux adresses indiquées dans le programme assembleur.

Fonctionnement général du programme (bien relire le TD décrivant la machine RISC)

- Le programme est encodé en binaire dans le tableau code
- Le programme est chargé (copié) dans le tableau M
- La machine commence à exécuter le programme à l'adresse `PC0`.
- `PC` indique l'adresse de l'instruction à exécuter et `IR` contient cette instruction.
- la machine exécute des instructions tant qu'elle ne rencontre pas l'instruction i_hrs.

Création d'une instruction à partir des codes mnémoniques

Réalisation

Vous commencerez par compléter la fonction put() qui permet de créer un programme assembleur (dans le tableau code) à partir des codes mnémoniques et des paramètres. Un exemple de création de programme est donné dans la fonction main().

Test

Dès que vous avez écrit cette fonction, vous pouvez la tester en utilisant la fonction fournie int2bin() qui permet d'afficher sous forme binaire un entier.

Décodage d'une instruction

Réalisation

La première fonction de la machine est de décoder l'instruction contenue dans `IR`.
Écrire la fonction decodeinst() qui décode une instruction et affiche le code opération et les paramètres.

Test

Compléter la fonction decode() qui decode un programme contenu dans un tableau et affiche sur l'écran le codage binaire et le décodage.
Écrire un programme assembleur avec quelques instructions et vérifier qu'elles soient bien décodées.
Lister et tester les cas qui peuvent poser un problème. (Format F2 : penser à tester avec des valeurs de `c` négatives, ...).

Chargement du programme assembleur dans la mémoire de la machine.

Le tableau code qui contient le programme crée par les appels à la fonction put() simule un fichier. Ecrire la fonction load() copie le contenu de ce fichier dans le tableau M qui simule la mémoire.

Exécution de l'instruction

Réalisation

Une fois que les instructions sont copiées en mémoire, il ne reste plus qu'à programmer leurs execution. Compléter la fonction execute().

Test : écriture de programmes assembleur

Ecrire quelques programmes en assembleur pour cette machine et les exécuter (addition d'entiers, PGCD, MIN/MAX,…).