CA 2019 : Travaux Dirigés 1

On considère dans ce TD les codes assembleur donnés en annexe du sujet. Ces codes correspondent à la compilation en assembleur MIPS via *gcc* de deux fichiers C.

Exercice 0 : mémento MIPS

Prenez le temps de regarder le document avec l'ensemble des instructions MIPS, afin de savoir vous en servir.
Repérez les différentes classes d'instructions (arithmétique et logique, mémoire, saut), les mnémoniques de chaque classe. Ce cours vous semblera plus facile si vous parlez un peu le MIPS.

Pour information :

- r0 ou $0 vaut toujours 0
- %sp = $29 ou r29 est le pointeur de pile
- %fp = $30 ou r30 est le pointeur de frame (partie de la pile dédiée aux informations d'une fonction)
- Pour les conventions d'appel :
- La fonction appelante alloue de la place (4 octets par paramètres) sur la pile pour tous les paramètres puis elle utilise r4 – r7 pour faire passer la valeur des 4 premiers, les valeurs des paramètres suivants sont mises sur la pile (dans leur emplacement)
- La fonction appelante fait passer l'adresse de retour dans r31 ou $31, ce registre est implicitement écrit par les instructions finissant par « `al` » comme de `jal`, `jalr`, `bltzal`
- r2 ou $2 contient le résultat d'une fonction après un appel
- Une fonction alloue de la place sur la pile pour sauvegarder des registres (%fp, au minimum sans optimisation et $31 en cas d'appel de fonction dans son corps), de la place pour les variables locales. Dans une fonction, pointeur de frame vaut le haut de la pile après ces allocations (et après sauvegarde du contenu qu'il avait en entrée de la fonction).

Exercice 1 : analyse des fonctions d'un fichier assembleur

Pour les deux codes donnés, délimitez sur la feuille les lignes correspondant à des fonctions. Combien de fonctions composent ces deux codes ? Quels sont leurs noms ?

Quelles directives permettent de trouver le début et la fin d'une fonction ? (de telle sorte que l'on ait bien toutes les instructions et toutes les étiquettes de la fonction)

Exercice 2 : bloc de base

Trouvez la définition d'un bloc de base (cours). En MIPS l'instruction qui suit un branchement fait partie du même bloc de base que le branchement, c'est l'instruction du ***delayed slot*.**

Appliquez l'algorithme donné en cours pour déterminer les entêtes et les blocs de base simultanément pour les fonctions `max` du fichier *ex_asm.s* et `max_min_tab` du fichier *test_asm32.s*.
Idem pour les fonctions `mat_mul` et le `main` du fichier *test_asm32.s* et vérifiez que cela donne ce que vous souhaitez.

Suivez bien l'algorithme, car s'il est facile de voir les blocs de base à l'œil nu, vous aurez à programmer l'algorithme et c'est plus facile lorsqu'on l'a bien assimilé.

Exercice 3 : CFG

Numérotez les blocs de base dans leur ordre d'apparition en commençant par 0 pour les 4 fonctions analysées à la question précédente.
Combien de successeurs peut avoir un bloc de base ? Comment le déterminer ?

Pour chacune, des 4 fonctions précédentes, donnez pour chaque bloc de base précédentes ses blocs successeurs.
À partir de cette information, construisez le CFG associé aux fonctions en représentant le graphe sur votre feuille.

Exercice 4 : dominants

Calculez, pour les CFG associés aux fonctions `max`, `max_min_tab` et `mat_mul` les blocs dominants en appliquant l'algorithme donné en cours.

Donnez le bloc dominant immédiat de chacun des blocs s'il existe, puis dessinez l'arbre des blocs dominants avec la relation de dominance immédiate.

Exercice 5 : arcs retour et boucles naturelles

Pour les fonctions `max_min_tab`, `mat_mul`, quels sont les arcs retour du CFG ?

Donnez la liste des blocs de base compris dans les boucles naturelles associées.

Que pouvez vous dire des boucles ? Sont elles disjointes ? Imbriquées ? Si oui, peut on savoir quelle est la boucle interne ?

Esquissez un algorithme permettant de déterminer les boucles d'une fonction et les blocs qui composent chaque boucle, en supposant que vous connaissez

1) Les blocs de base de la fonction (liste sur laquelle on peut itérer)
2) Les prédécesseurs et successeurs de chaque bloc (leur nombre et on peut y accéder)
3) Les blocs dominants de chaque bloc de base de la fonction