Unix : concepts et outils avancés
TD no 11 : mémoire partagée, sémaphores

Olivier Aumage
LIP, bureau 302
E-mail : `Olivier.Aumage@ens-lyon.fr`

13 décembre 2001

1 Préambule

Finir les TD numéro 9 et 10 s'ils ne sont pas terminés.

2 Processeurs

2.1 Objectif

Le but de cet exercice est de modéliser une partie des mécanismes de synchronisation intervenant dans un processeur superscalaire, c'est-à-dire un processeur disposant de plusieurs unités de calcul. Dans cet exercice, vous utiliserez des sémaphores pour synchroniser les différentes entités actives.

Les commandes de contrôle des sémaphores sont masquées par les fonctions du fichier **`semaphore.c`** du répertoire **`/home/oaumage/Unix/Semaphores`** qu'il faudra recopier dans votre répertoire de même que le fichier **`semaphore.h`**, qui contient la déclaration du type **`semaphore`** et les déclarations des prototypes des fonctions de manipulation. Vous pourrez utiliser les commandes shell **`ipcs`** et **`ipcrm`** pour respectivement observer et détruire les sémaphores utilisés par votre programme.

Note : le *nom* des sémaphores doit être un nom de fichier (ex : **`/var/tmp/semaphore_1`**), car la fonction **`ftok`** est utilisée en interne pour générer la clé d'accès au sémaphore.

Note 2 : l'énoncé de ce TD est une adaptation d'une partie du sujet d'examen de l'année 2000-2001.

2.2 Cas simple

Description Le processeur superscalaire en question est constitué d'une unité d'alimentation (**`UA`**), d'une unité de *prefetching* (**`UP`**) et de plusieurs unités de calcul (UC). Ce processeur dispose de R registres communs à l'ensemble des unités, d'un tampon de prefetching d'une taille **`MAX_FETCHED`** instructions et d'un tampon d'instructions décodées de taille **`MAX_READY`**. Pour des raisons de simplicité, on se limite exclusivement à des flots d'instructions strictement séquentiels (donc sans opérations de saut).

Fonctionnement L'unité de prefetching est chargée de parcourir le tampon de prefetching à la recherche d'une *instruction exécutable*. Lorsqu'une telle instruction est trouvée, elle est transférée dans le tampon d'instructions décodées et le processus est itéré. Au fur et à mesure que le tampon de prefetching est vidé par l'unité correspondante, l'unité d'alimentation rajoute des instructions en provenance de la mémoire. Les unités de calcul exécutent les instructions : dès qu'une unité a terminé l'exécution d'une instruction, elle extrait la première instruction du tampon d'instructions décodées et l'exécute.

Une instruction **i** utilise un ensemble R_i de registres quelconques ($0 \leq |R_i| \leq R$). Les unités de calcul fonctionnant en parallèle, il est indispensable que deux unités de calcul n'utilisent pas simultanément les mêmes registres (dans un premier temps, on ne se préoccupe pas de distinguer les accès en lecture des accès en écriture). De plus, la sémantique du programme doit être respectée. En conséquence, une *instruction exécutable* du point de vue de l'**UP** est une instruction ayant les propriétés suivantes :

– toutes les instructions dont elle dépend (c'est-à-dire l'ensemble des instructions **j** qui la précèdent et pour lesquelles $R_i \cap R_j = \emptyset$) ont été exécutées ;
– elle n'utilise aucun des registres utilisés par les instructions présentes dans le tampon d'instructions décodées.

Si l'instruction considérée par l'**UP** n'est pas exécutable, l'**UP** teste l'instruction suivante, jusqu'à l'obtention d'une instruction exécutable ou l'arrivée en fin de tampon.

Implémentation Les tampons de prefetching et d'instructions décodées ainsi que les données de synchronisation autres que les sémaphores seront disposés dans une zone de mémoire partagée de taille suffisante. Le format de codage des instructions est libre (la seule information strictement nécessaire concerne les registres impliqués).

Question 1 Écrivez un programme **processeur.c** qui initialise les différentes parties de la zone de mémoire partagée et les sémaphores. Lorsque les programmes implémentant les unités seront prêts, le programme **processeur.c** sera également chargé de les lancer.

Question 2 Écrivez un programme **alim.c** implémentant l'unité d'alimentation et un programme **prefetch.c** qui implémente l'unité de prefetching (uniquement l'interaction avec l'unité d'alimentation pour l'instant).

Question 3 Complétez la simulation en programmant **calc.c**, le programme qui gère une unité de calcul, et la partie manquante du programme **prefetch.c**. Il est important qu'il n'y ait pas d'attente active lorsqu'aucune instruction n'est disponible dans le tampon d'instructions décodées. Pour l'unité de prefetching en revanche, il n'est pas demandé de chercher à éviter l'attente active.

Augmentation du parallélisme Le but d'une architecture superscalaire étant de paralléliser l'exécution des instructions, il est indispensable d'éviter de reporter abusivement le traitement d'une instruction donnée. On distingue donc désormais l'usage qu'une instruction fait d'un registre : une instruction **i** utilise un ensemble IN_i de registres ($0 \leq |IN_i| \leq R$) quelconques en lecture et un ensemble OUT_i de registres ($0 \leq |OUT_i| \leq R$) quelconques en écriture. Les deux ensembles ne sont pas nécessairement disjoints.

Pour pouvoir placer une instruction **i** dans le tampon des instructions décodées, l'**UP** doit donc vérifier la propriété suivante pour chaque instruction **j** précédente en attente ou en cours d'exécution :

$$IN_i \cap OUT_j = \emptyset \quad \wedge \quad OUT_i \cap IN_j = \emptyset \quad \wedge \quad OUT_i \cap OUT_j = \emptyset$$

Question 4 Quelles modifications et/ou ajouts sont nécessaires au niveau des structures de données de synchronisation pour autoriser l'exécution simultanée d'instructions effectuant des accès en (uniquement) lecture à des registres communs ? Modifiez les programmes correspondants en conséquence.

Barrière de synchronisation Il est parfois nécessaire de s'assurer que les **UC** ont terminé l'exécution d'une série d'instructions avant d'exécuter une nouvelle série.

Question 5 Faites les modifications nécessaires pour permettre l'utilisation d'une instruction de synchronisation.

— * —