TP8

Pipeline

Objet du TP

L'objectif de ce TP est d'observer et de comprendre le fonctionnement d'un processeur pipeliné ainsi que la notion de parallélisme d'instructions.

Partie 1 : Présentation du simulateur

Dans ce TP, nous allons utiliser un simulateur permettant d'observer l'état du pipeline du NIOS. Le NIOS utilisé dans ce TP est un NIOS à 8 étages de pipeline (3 étages d'exécution et 2 étages d'accès à la mémoire).

- Copier le répertoire **share:\l3info\ILM\TP\TP8** dans votre répertoire local.
- Ouvrez l'archive **openNios.jar**. Ce logiciel permet d'exécuter un programme donné en langage assembleur (*nomfichier.s*).

Lorsque vous lancez le simulateur, la seule fenêtre présente à l'écran est celle de l'éditeur de texte. Dans cette fenêtre, choisissez l'option « **Load and run »** permettant de choisir un fichier de code assembleur et de l'exécuter. Sélectionnez le fichier que vous voulez exécuter *(nomfichier.s*).
Plusieurs fenêtres devraient alors s'ouvrir :

- Une fenêtre donnant l'état de la file de registres,
- Une fenêtre permettant de lire la mémoire,
- Une fenêtre de statistiques,
- Une fenêtre qui désassemble le code contenu en mémoire,
- Une fenêtre qui permettra de suivre l'état du pipeline.

Lorsque ces fenêtres sont apparues, cela signifie que le programme est prêt à être simulé. Pour la simulation, vous pouvez utiliser les raccourcis suivants :

- F6 pour une exécution complète du code en affichant les résultats,
- F7 pour exécuter un cycle,
- F8 pour exécuter N cycles,
- F9 pour exécuter jusqu'à une adresse donnée,
- F5 pour une exécution complète du code (affiche les résultats à la fin),
- F4 pour remettre le simulateur à zéro.

Partie 2 : Travail à réaliser

Deux programmes vous sont fournis à titre d'exemples.

- Une multiplication de deux matrices 4x4 (*matrixMultiply.s*)
- Un filtrage numérique (*Filtrage.s*)

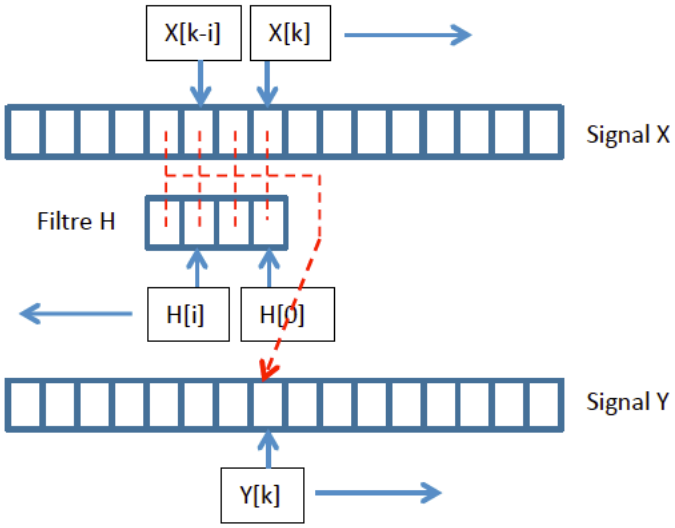
Vous pouvez choisir le code que vous préférez. Le filtrage numérique est plus rapide en temps de simulation.

Le programme (*Filtrage.s*) réalise un filtrage numérique, c'est-à-dire la convolution entre un signal X et un filtre H pour produire un signal filtré Y = X*H.
Le signal numérique X peut être vu comme un tableau de dimension 1 d'échantillons X[k] (des valeurs numériques sur 32 bits dans notre exemple). Le signal X de notre exemple comporte 16 échantillons.

Le filtre H est un tableau de dimension 1 contenant des coefficients H[i] (au nombre de 4 dans notre exemple) représentés sur 32 bits.

La relation permettant de calculer Y[k] en fonction de X et H est donnée par :
Signal en entrée : k = 0 à 15 : X[k]
Filtre : i = 0 à 3 : H[i]
Pour k = 3 à 15 : $Y[k] = \sum_{i=0}^{3}(X\,[k - i\,]\,H[i])$
(Y[0] = Y[1] = Y[2] = 0)

Le signal Y ne peut pas être calculé pour les 3 premiers échantillons car X est inconnu pour les indices négatifs.
Lors de la simulation, vous surveillerez particulièrement les registres r6 et r7 (valeurs de i et k) et le remplissage progressif de Y en mémoire.



1. Vous pouvez lire le code assembleur chargé dans l'éditeur de code du simulateur. Donnez une estimation du nombre de cycles nécessaires à l'exécution du code.
2. Observez l'exécution des premières instructions en utilisant F7. Qu'observez-vous ? Exécutez le code en entier (F6 ou F5) et donnez le nombre de cycles écoulés.
3. Tentez de réduire au maximum ce temps en utilisant les transformations de code qui vous semblent adaptées :
    a. Réordonnancez les instructions (pour cacher les dépendances).
    b. Déroulez des boucles.
    En pratique, est-ce qu'un code entièrement déroulé serait plus efficace ?