Phelma 2A
Filière Systèmes Electroniques Intégrés

Architectures de processeurs et sécurité matérielle

TD APSM23-1 – Architecture pipeline

Partie 1

On considère un processeur RISC construit autour d'une partie opérative "de base" dont la structure est représentée en figure 1.1. Ce processeur est basé sur une architecture chargement-rangement, et exécute des instructions de calcul arithmétique ou logique de type "Rx op Ry -> Rz", où Rx, Ry et Rz sont trois registres banalisés appartenant au banc de 32 registres illustré sur la figure. Seules les instructions de calcul seront prises en compte dans cet exercice ; le matériel utilisé pour les instructions d'accès mémoire n'est pas détaillé. Tous les registres de la partie opérative (registres banalisés et registres tampons T1 à T3) sont sensibles au même front d'horloge. L'unité de traitement UAL permet de réaliser les opérations A + B, A – B et décalage à gauche de A (A étant positionné sur l'entrée marquée +, et B étant positionné sur l'entrée marquée +/-). Le banc de registres permet de réaliser deux lectures et une écriture par cycle.

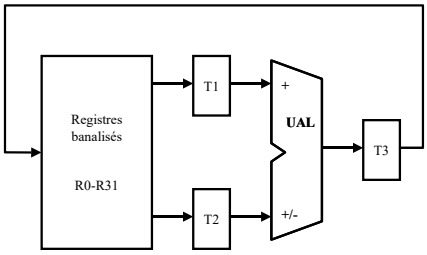

Figure 1.1 – partie opérative de base

En l'absence d'aléa d'exécution, une instruction peut être lue à chaque cycle depuis la mémoire cache instructions. Le pipeline est divisé en quatre étages, correspondant aux tâches suivantes pour une instruction de calcul :

- lecture de l'instruction (mémoire -> registre d'instruction et incrémentation du compteur ordinal)
- décodage de l'instruction (détermination du type d'instruction, gestion des aléas, transfert des opérandes)
- exécution (calcul et transfert de la sortie de l'UAL vers T3)
- rangement (transfert T3 -> registre banalisé de destination)

1.1. Dans le cas idéal où aucun aléa d'exécution ne survient, combien faut-il de cycles d'horloge au total pour exécuter cinq instructions avec le processeur proposé (depuis le chargement de la première, jusqu'au rangement du résultat de la dernière) ? Combien en faut-il pour exécuter cinquante mille instructions ?

1.2. On souhaite exécuter la série d'instructions suivante, décrite en "pseudo-assembleur" :

Instruction I1 : R1 – R7 -> R2
Instruction I2 : R2 + R4 -> R8
Instruction I3 : R2 – R1 -> R23
Instruction I4 : R6 – R9 -> R18
Instruction I5 : R8 + R2 -> R8

Combien de cycles d'horloge vont être nécessaires pour l'exécution de cette partie de programme (depuis la lecture de I1, jusqu'au rangement du résultat de I5), en supposant que les instructions avant I1 s'exécutent sans aléa et qu'elles ne modifient pas les registres utilisés par les instructions considérées ? On précisera les opérations internes exécutées à chaque cycle (par exemple, "R3 -> T1" si le contenu du registre R3 est transféré vers le tampon T1).

1.3. Sans modifier la fonctionnalité, un simple ré-ordonnancement des instructions pourrait-il permettre d'améliorer les performances du programme de la question 1.2. ?

1.4. Quatre modifications de la partie opérative par ajout de multiplexeurs sont proposées en figures 1.4.1 à 1.4.4. Combien de cycles peuvent être gagnés, lors de l'exécution de la séquence d'instructions de la question 1.2., par chacune de ces modifications ? Le détail des opérations internes exécutées à chaque cycle dans les quatre cas n'est pas demandé (mais une explication du raisonnement est nécessaire !).

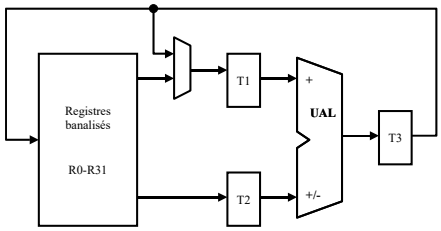

Figure 1.4.1 – modification 1

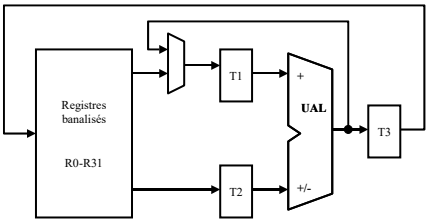

Figure 1.4.2 – modification 2

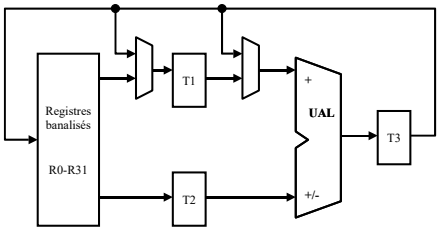

Figure 1.4.3 – modification 3

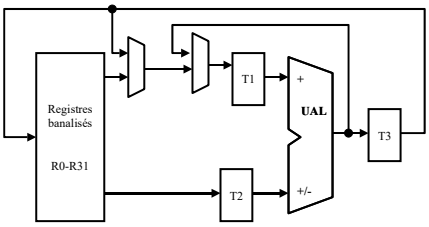

Figure 1.4.4 – modification 4

1.5. Quelle structure de partie opérative choisiriez vous, parmi les deux dernières (figures 1.4.3 et 1.4.4), pour optimiser l'exécution d'un programme ? Justifiez votre réponse.

Partie 2

La figure 2.1 résume la structure du pipeline de l'unité entière d'un processeur d'architecture Sparc v8 nommé "Leon 2". Ce pipeline à cinq étages permet de charger une instruction par cycle.

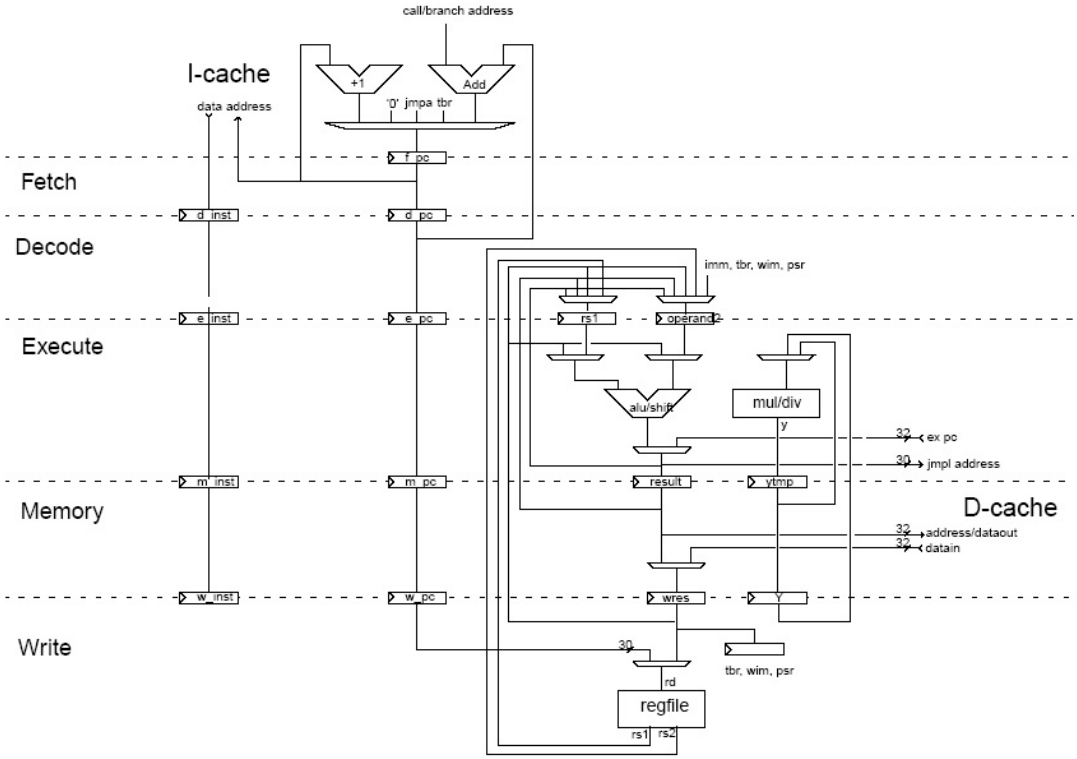

Figure 2.1 – structure d'une unité entière avec pipeline

2.1. A quoi sert le rebouclage de la sortie du registre wres vers les étages précédents ?

2.2. En fonction de la structure proposée, et en l'absence d'aléa, expliquez les étapes du déroulement d'une instruction de type "LOAD Rx, $D" (LOAD en adressage basé), permettant de lire une donnée à l'adresse hexadécimale $Ad et de la charger dans le registre Rx. L'adresse $Ad correspond à la somme du contenu de l'un des registres internes (prédéfini – par exemple R31) et du déplacement $D fourni en valeur immédiate.

2.3. Même question pour une instruction de type branchement retardé ("delay slot" de 1 cycle).