HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Romuald Mosqueron

Prof. Marina Zapater

Laboratoire d'architecture des ordinateurs
semestre printemps 2022

Microarchitecture PIPELINE - Partie 2

Informations générales

Le rendu pour ce laboratoire sera **par groupe de deux**, chaque groupe devra rendre son travail.

Le rendu du laboratoire ne sera pas évalué comme indiqué dans la planification des laboratoires, cependant il permet aux professeurs/assistants d'assurer un suivi de votre travail. Tout retard impactera l'évaluation du rapport final

⚠ **N'oubliez pas de sauvegarder et d'archiver votre projet à chaque séance de laboratoire**

NOTE : Nous vous rappelons que si vous utilisez les machines de laboratoire situées au niveau A, il ne faut pas considérer les données qui sont dessus comme sauvegardées. Si les machines ont un problème nous les remettons dans leur état d'origine et toutes les donnés présentes sont effacées.

Objectifs du laboratoire

L'objectif est de comprendre le fonctionnement d'un processeur dit pipeliné. Vous recevrez le processeur complet en version pipeliné et des programmes à exécuter. Cette version du processeur ne comprend aucun mécanisme de détection des aléas et d'arrêt du pipeline. A la fin du laboratoire, vous devrez modifier le circuit pour ajouter ces mécanismes. Ce laboratoire est noté. Vous devez rendre le projet Logisim et **tous les codes assembleur** que vous écrivés ou modifiés.

Outils

Pour ce labo, vous devez utiliser les outils disponibles sur les machines de laboratoire (A07/A09) ou votre ordinateur personnel avec la machine virtuelle fournie par le REDS.

⚠ **L'installation de la machine virtuelle doit se faire en dehors des séances de laboratoire afin que vous puissiez profiter de poser des questions pendant le laboratoire. L'installation n'est pas comptée dans les périodes nécessaires à la réalisation de ce laboratoire.**

Fichiers

Vous devez réutiliser et continuer de complèter les fichiers du laboratoire Pipeline - Partie 1.

⚠ **Le fichier Makefile ne doit pas être modifié !**

⚠ **Respectez l'architecture hiérarchique présentée dans le cours.**

Travail demandé

Ce laboratoire est divisé en deux parties. Chaque partie est notée séparément. Pour la première partie, vous avez du réaliser les points suivants :

— Analyse et test du processeur pipeliné
— Ajout de la détection d'aléas de contrôle

Pour la deuxième partie, vous devez réaliser :

— Ajout de la détection d'aléas de donnée
— Optimisation du pipeline avec la technique du pipeline forwarding

1 Aléas de donnée

Vous pouvez modifier les entrées/sorties des différents blocs si vous le désirez. Dans ce cas, vous devez rendre un fichier readme.txt qui décrit brièvement les différents changements effectués, en spécifiant les blocs et les signaux.

⚠ Si vous changez les entrées/sorties, la taille du bloc va changer et les signaux peuvent ne plus être connectés correctement. C'est votre responsabilité de contrôler que les blocs soient connectés correctement.

1.1 Circuit data_hazard

Ce circuit permet de détecter si une instruction en cours de décodage est dépendante du résultat d'une instruction qui n'est pas encore écrit dans un registre.

Vous devez compléter le contenu de ce bloc.

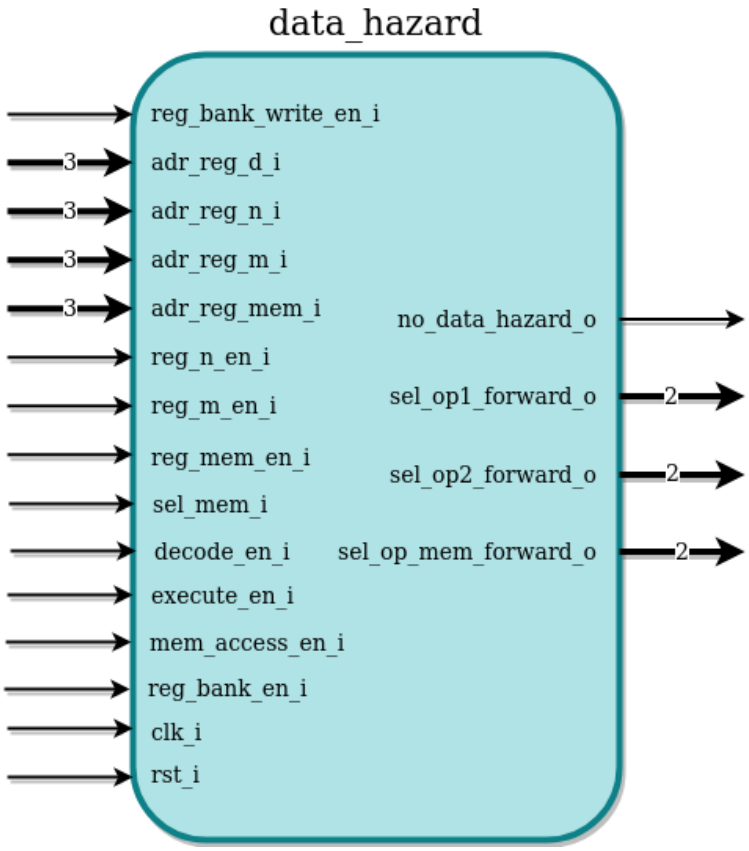

FIGURE 1 – Signaux du bloc data_hazard

Description des différents signaux du bloc :

Nom I/O	Description
reg_bank_write_en_i	Indique si l'instruction requiert l'écriture d'un registre
adr_reg_d_i	Registre à écrire
adr_reg_n_i	Registre à lire pour l'opérande 1
adr_reg_m_i	Registre à lire pour l'opérande 2
adr_reg_mem_i	Registre à lire pour l'opération mémoire
reg_n_en_i	Indique si l'opérande 1 est lue d'un registre
reg_m_en_i	Indique si l'opérande 2 est lue d'un registre
reg_mem_en_i	Indique si l'opération mémoire lit une valeur d'un registre
sel_mem_i	Indique si cette instruction est une instruction mémoire
decode_en_i	Enable du bloc DECODE
execute_en_i	Enable du bloc EXECUTE
mem_access_en_i	Enable du bloc MEMORY_ACCESS
reg_bank_en_i	Enable du bloc BANK_REGISTER
clk_i	Clock du système
rst_i	Reset asynchrone du système
no_data_hazard_o	Indique qu'il n'y **pas** d'aléa de donnée pour cette instruction
sel_op1_forward_o	Sélection d'une donnée forward-ée pour l'opérande 1
sel_op2_forward_o	Sélection d'une donnée forward-ée pour l'opérande 2
sel_op_mem_forward_o	Sélection d'une donnée forward-ée pour l'opérande des instructions mémoire

Pour le moment, nous nous intéressons uniquement à commander la sortie **no_data_hazard_o**. Répondez aux questions ci-dessous puis servez-vous de ces réponses pour créer le schéma Logisim permettant de détecter un aléa de donnée et de commander cette sortie.

Questions

1. Comment savoir si une instruction est dépendante d'une instruction qui est pour le moment dans le stage EXECUTE ? dans le stage MEMORY_ACCESS ? Dans le stage WRITE_BACK ?
2. Est-ce que ça pose un problème si une instruction dépend du résultat d'une instruction qui est au stage WRITE_BACK ?
3. Quels informations doivent être mémorisées pour chaque instruction ?
4. Quelles informations permettent de savoir si le registre D est utilisé ?

Indications

Un aléa de données est détecté si le ou les registres qui doivent être lus pour l'instruction courante correspond à un registre qui va être écrit par une instruction précédente. On peut donc séparer l'aléa de donnée en trois aléas distincts :

— Un aléa à cause du registre N

— Un aléa à cause du registre M

— Un aléa à cause du registre MEM

Il vous faudra comparer la valeur des signaux adr_reg_n_i, adr_reg_m_i, adr_reg_mem_i avec l'adresse du registre D de l'instruction qui est dans le bloc EXECUTE, MEMORY_ACCESS et WRITE_BACK. Pour se faire il faut mémoriser à l'aide de registres, l'adresse du registre D des instructions qui passent dans le pipeline.

Attention, certaines instructions n'utilisent pas de registre en lecture ou pas tous les registres en lecture. C'est pourquoi il faut contrôler si les valeurs dans adr_reg_n_i, adr_reg_m_i et adr_reg_mem_i contiennent la valeur d'un registre qui va être lu ou si leur valeur doit être ignorée.

De la même manière, certaines instructions ne font pas d'écriture dans le registre D. Ceci est aussi le cas si par exemple, certains blocs du pipeline sont désactivés.

1.2 Circuit hazard_detection

Ce circuit est instancié dans main_control_unit qui est instancié dans le bloc decode. La plupart des connections sont déjà faites. Vous devez ajouter dans le circuit main_control_unit les connections pour les signaux **reg_n_en_s**, **reg_m_en_s**, **reg_mem_en_s**. Les signaux du bloc hazard_detection sont décrits ici.

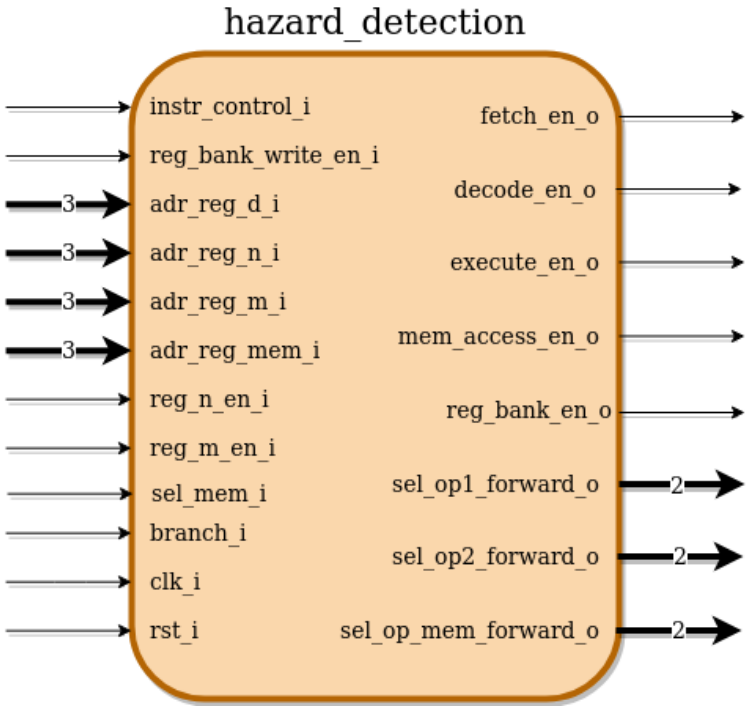

FIGURE 2 – Entrées/sorties du bloc hazard_detection

Description des différents signaux du bloc :

Nom I/O	**Description**
instr_control_i	Indique que l'instruction en cours de décodage est une instruction de contrôle.
reg_bank_write_en_i	Indique si l'instruction requiert l'écriture d'un registre
adr_reg_d_i	Registre à écrire
adr_reg_n_i	Registre à lire pour l'opérande 1
adr_reg_m_i	Registre à lire pour l'opérande 2
adr_reg_mem_i	Registre à lire pour l'opération mémoire
reg_n_en_i	Indique si l'opérande 1 est lue d'un registre
reg_m_en_i	Indique si l'opérande 2 est lue d'un registre
reg_mem_en_i	Indique si l'opération mémoire lit une valeur d'un registre
sel_mem_i	Indique si cette instruction est une instruction mémoire
branch_i	Indique si l'instruction en cours de décodage suit un saut qui a été pris
clk_i	Clock du système
rst_i	Reset asynchrone du système
fetch_en_o	Enable du bloc fetch
decode_en_o	Enable du bloc decode
execute_en_o	Enable du bloc execute
mem_access_en_o	Enable du bloc memory_access
reg_bank_en_o	Enable du bloc bank_register
sel_op1_forward_o	Sélection d'une donnée forward-ée pour l'opérande 1
sel_op2_forward_o	Sélection d'une donnée forward-ée pour l'opérande 2
sel_op_mem_forward_o	Sélection d'une donnée forward-ée pour l'opérande des instructions mémoire

Commande des signaux dans main_control_unit

Répondez aux questions ci-dessous puis transposez sur Logisim vos réponses et complétez le circuit.

Questions

1. Quelles informations permettent de savoir si le registre N, M ou mem sont utilisés ?

Commande des signaux dans hazard_detection

Dans le circuit hazard_detection, analyser comment les signaux **fetch_en_s**, **decode_en_s execute_en_s**, **mem_access_en_s** et **reg_bank_en_s** sont générés à partir du signal no_data_hazard_s. Ces signaux dépendent notamment (mais pas que) de **no_data_hazard_s**. Vous pouvez vous aider du schéma 3.

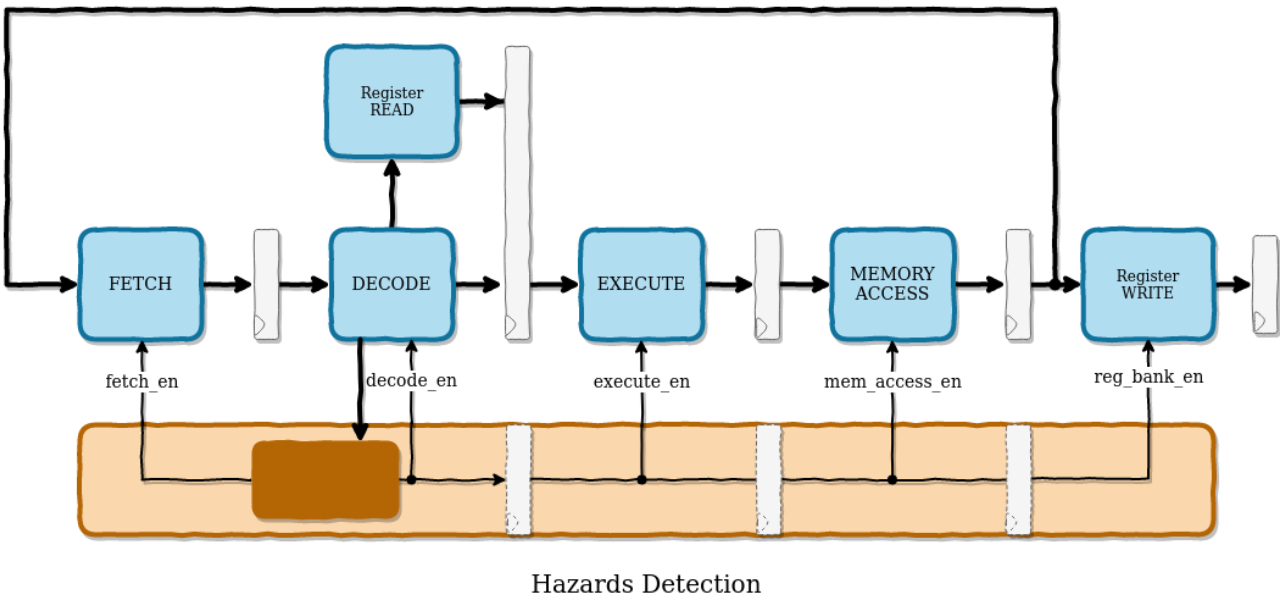

FIGURE 3 – Croquis du processeur pipeliné

Questions

1. Quelles informations permettent de savoir si le registre D est utilisé ?
2. Une détection d'aléa de donnée va influencer quel(s) enable(s) ? A quel moment ? Pourquoi ?

1.3 Test aléas de donnée

Ecriver un programme qui contient des aléas de donnée. Vous pouvez réutiliser et modifier le code que vous aviez réalisé pour les aléas de contrôle si vous le désirez. Utiliser l'instruction *BL* qui génère un aléa de donnée et un aléa de contrôle. Tester votre programme en faisant un chronogramme.

Questions

1. Est-ce que les valeurs dans les registres sont mises à jour correctement et au bon moment ?
2. Pourquoi l'instruction *BL* génère un aléa de contrôle et un aléa de donnée ?
3. Combien de cycles sont nécessaires pour résoudre les aléas de l'instruction *BL* ?
4. Quel est l'IPC pour votre programme ?

2 Pipeline Forwarding

⚠ **Faite une copie de votre workspace qui contient la gestion des aléas et travailler dans la copie du workspace.**

Vous devez rendre un circuit séparé pour le pipeline forwarding.

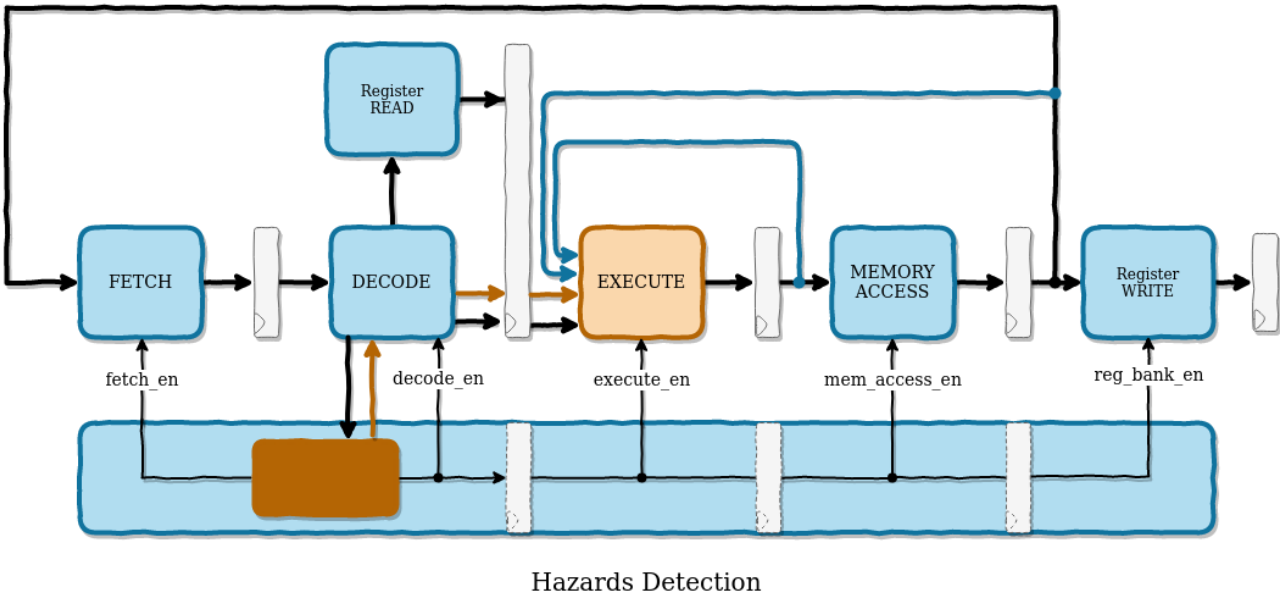

FIGURE 4 – Croquis du processeur optimisé avec du pipeline forwarding

Afin d'optimiser la gestion des aléas, deux chemins pour les données ont été ajoutés au circuit afin d'obtenir la nouvelle valeur des registres avant qu'elle soit écrite dans les registres comme sur la Figure 4.

2.1 Circuit data_hazard

Vous devez faire des modifications au circuit data_hazard pour créer les signaux **sel_op1_forward_s**, **sel_op2_forward_s** et **sel_op_mem_s**. Ces signaux doivent prendre les valeurs suivantes.

Valeur	Description
0	Il n'y pas de data forwarding possible ou nécessaire pour cette instruction
1	Data forwarding depuis le bloc EXECUTE
2	Data forwarding depuis le bloc MEMORY_ACCESS

sel_op1_forward_s fait la sélection pour l'opérande 1. **sel_op2_forward_s** fait la sélection pour l'opérand 2 et **sel_op_mem_forward_s** fait la sélection de la donnée pour les instructions mémoire.

Questions

1. A quoi sert le signal sel_mem_i ?
2. Est-il possible/utile de faire un data forwarding depuis le stage WRITE_BACK ? (l'écriture dans le registre dans la banque de registres). Comment pourrait-il être ajouté au circuit ?
3. Quelles sont les conditions pour que le forwarding puisse avoir lieu ? Quelles sont les conditions pour que le forwarding soit utile ?
4. Quelles sont les conséquences du forwarding sur la gestion des aléas de données ? Quelles sont les conséquences du forwarding sur la gestion des aléas de contrôle ?

Indications

Les valeurs qui proviennent d'une instruction mémoire ne peuvent pas être forwardées.

2.2 Circuit execute

Vous devez modifier le circuit Execute pour que **operand_1_s** et **operand_2_s** soient séléctionnés avec les signaux **sel_op1_forward_s** et **sel_op2_forward_s** respectivement. Ils doivent pouvoir garder leur valeur si le forwarding n'est pas effectué ou prendre la valeur **forward_execute_data_s** ou **forward_mem_data_s** dépendant d'où le forwarding est effectué.

La même chose doit être fait pour le signal **reg_mem_data_s** grâce au signal **sel_op_mem_forward_s**

Questions

1. Pourquoi doit-on faire ça ?
2. Pourquoi doit-on faire ça pour le signal **reg_mem_data_s** ?
3. Que devrait-on faire si on avait un data forwarding venant du WRITE_BACK ?

2.3 Test : pipeline forwarding

Vous pouvez réutiliser le programme que vous aviez fait pour la gestion des aléas pour tester votre pipeline avec le forwarding. Faites un chronogramme et regardez si tout est correct.

Questions

1. Est-ce que votre processeur fonctionne correctement ? Est-ce que les timings sont respectés ? Est-ce que les registres contiennent les bonnes valeurs si on regarde étape par étape l'exécution des instructions ?
2. Quel est l'IPC de votre programme ? et le throughput si on considère une clock à 4KHz ?
3. Combien de cycles sont nécessaires pour que l'instruction BL soit complétée ?
4. Avez-vous d'autres idées d'optimisation de ce processeur ?

3 Rendu

Votre rendu ne sera pas évalué comme indiqué dans la planification des laboratoires, cependant il permet aux professeurs/assistants d'assurer un suivi de votre travail. Tout retard impactera l'évaluation du rapport intermédiaire ou final.

Les fichiers à rendre pour l'évaluation sont les suivants :

— Le circuit processeur_ARO2.circ dans lequel la détéction des aléas de donnée a été ajoutée.

— Le circuit processeur_ARO2_forwarding.circ dans lequel le pipeline forwarding a été ajouté.

— Le code de test des aléas de donnée.