École de technologie supérieure
Département de génie électrique

Dernière mise à jour : Hiver 2022

ELE344 Conception et architecture de processeurs
Projet 3 : Le processeur MIPS avec pipeline

(Durée 3 semaines)

I. Introduction

Ce projet a pour objectif de concevoir un processeur MIPS avec pipeline. À la fin de ce projet, vous devriez comprendre les aspects architecture du processeur MIPS avec pipeline. À la fin de ce projet, vous devriez être en mesure de :

1. Comprendre les principes du pipeline dans un processeur et des impacts sur la performance;
2. Comprendre les aléas de données et de contrôle;
3. Ajouter des étages de pipeline au processeur MIPS mono-cycle;
4. Implémenter la logique pour l'unité d'envoi pour minimiser l'impact des aléas de données;

Deux préparations doivent être soumises respectivement **AVANT** le début de la première et la deuxième séance de laboratoire (voir les descriptions des PRÉPARATIONS 1 et 2). Tout retard implique une note zéro pour cet élément d'évaluation.

II. Préparation

Le processeur MIPS contient deux composants : le contrôleur et le chemin de données (*Datapath*) tel que présenté à la figure 1. Le chemin de données (*Datapath*) contient cinq étages de pipeline (IF, ID, EX, MEM, WB) et une unité d'envoi (*Forwarding Unit*) est ajoutée au chemin de données du MIPS mono-cycle réalisé dans le projet 2, tel que présenté à la figure 2. Le contrôleur reste tel que décrit dans le projet 2.

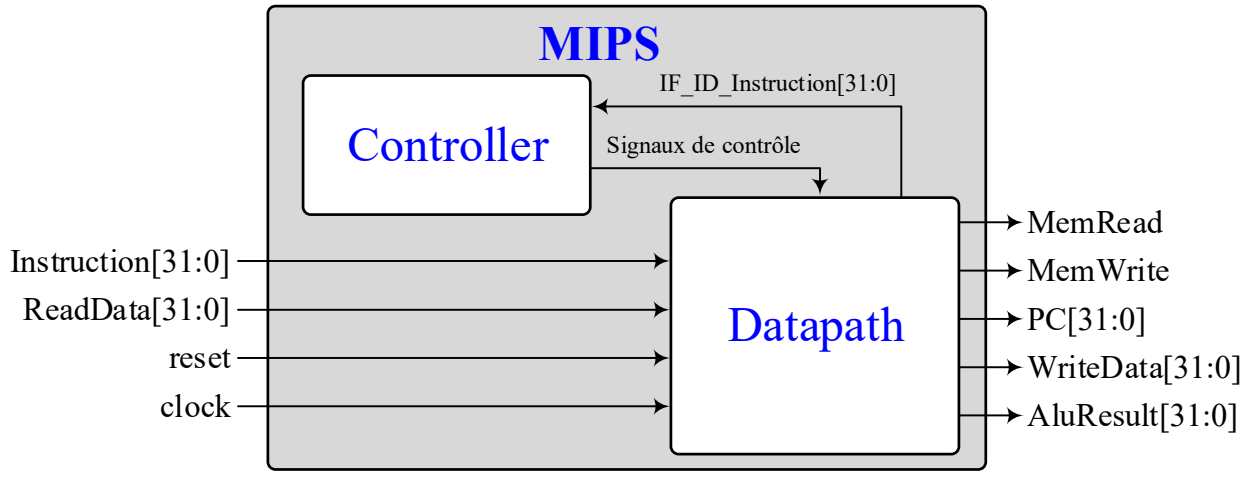

Figure 1 : Le composant MIPS avec pipeline

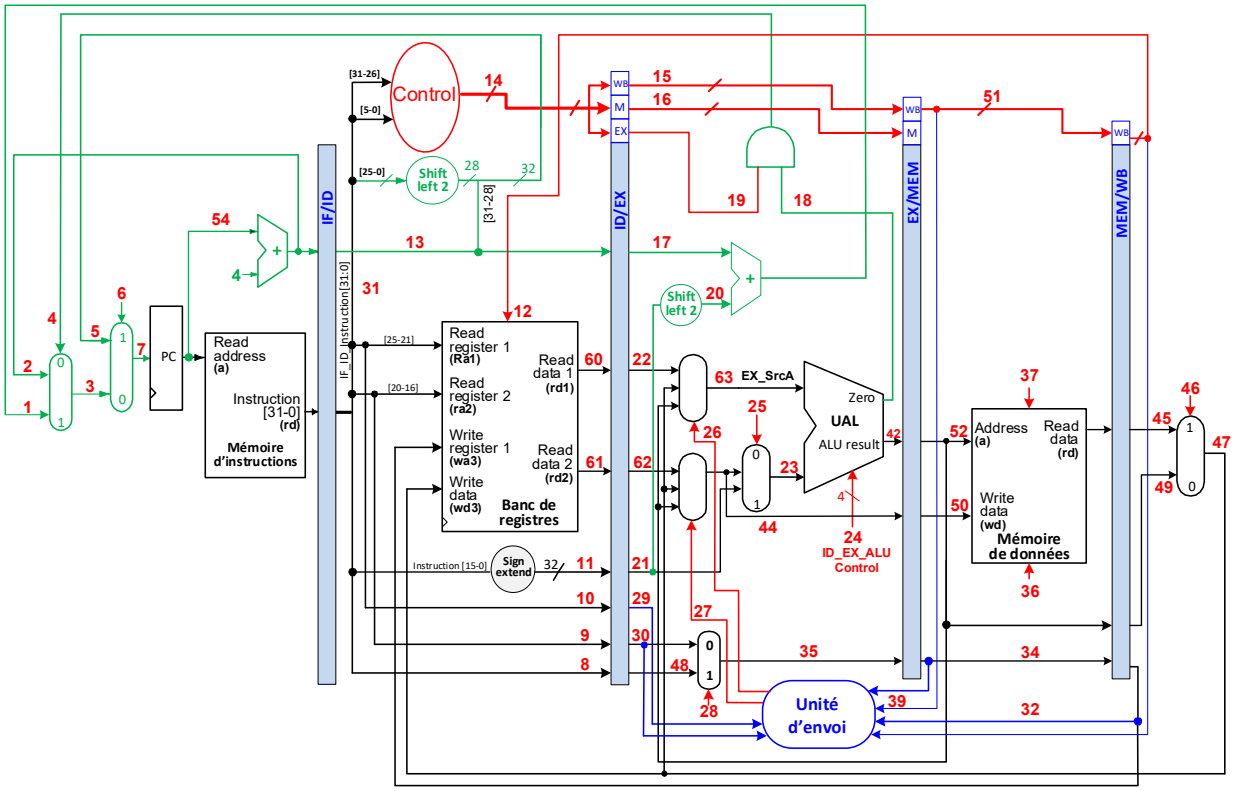

Figure 2 MIPS complet avec cinq étages de pipeline, avec l'unité de renvoi (Figure 4.65 tirée de Hennessy & Patterson « Computer Organisation and Design – The HW/SW Interface», 5e ed.)

PRÉPARATION 1 : Compléter le tableau 1 du fichier Excel fourni avec l'énoncé et le déposer sous « Dépôt de la préparation 1 » sous Projet 3 dans Moodle. Ce fichier doit être déposé **AVANT LE DÉBUT DE LA PREMIÈRE SÉANCE DU PROJET 3**. NE PAS CHANGER L'ORDRE DES SIGNAUX DANS LE FICHIER.

Concernant le tableau 1, pour chaque signal, vous devez :

1. Identifier leur type (std_logic ou std_logic_vector) et leur taille (la valeur de N);
2. Dans la première colonne, identifier son numéro (#) sur la figure 2, sachant que
   a. Les signaux associés aux étages registres (IF/ID, ID/EX) de pipeline ont les préfixes : IF_ID, ID_EX, EX_MEM, MEM_WB.
   b. Les préfixes IF_, ID_, EX_, MEM_, WB_ identifient l'étage pipeline auquel le signal est généré;
   c. N/A (non applicable) est assigné si le signal n'existe pas dans la figure 2.
3. Ajouter une très courte description sur son utilisation et sa fonctionnalité;
4. Déterminer le chemin dont le signal fait partie, parmi : signal de contrôle (C), signal du chemin de données (D), signal du chemin du calcul du PC (PC);
5. Déterminer le ou les cycles pipeline (IF, ID, EX, MEM, WB) auxquels l'information portée par le signal est destinée. Par exemple, déterminer le cycle auquel une donnée est mémorisé, traité par l'UAL, décodé; déterminer le cycle auquel un signal de contrôle est utilisée comme commande. Ajouter le symbole T si le signal est transmis au prochain étage pipeline.

III. Spécifications du processeur MIPS avec pipeline

Concevoir le processeur MIPS avec pipeline tel que représenté à la figure 2 tout en respectant les spécifications suivantes :

1. Les étages de registre dans le chemin de données et pour le calcul du PC doivent être insérés dans le fichier **`datapath.vhd`** créé dans le Projet 2 par l'insertion d'un seul process synchrone par étage (voir l'annexe 2) et selon le Tableau 1 du fichier Excel;
2. L'unité de contrôle est décrite dans le même fichier **`controller.vhd`** créé dans le Projet 2 (rien à changer);
3. Une unité d'envoi (***forwarding unit***) doit être ajouté dans le fichier **`datapath.vhd`** pour réduire les aléas de données. Cette unité anticipe l'écriture dans le banc de registres du résultat d'opération des instructions précédentes et envoie les données intermédiaires comme opérandes à l'UAL. L'UAL utilise ainsi le contenu du registre pipeline du résultat de l'opération logique et arithmétique des instructions précédentes plutôt que d'attendre son écriture dans le banc de registres;
4. Les sauts (J) et les branchements (BEQ) se décident et sont effectifs aux étages ID et EX respectivement. Aucune unité de détection d'aléas de contrôle est ajoutée pour bloquer (*stall*) ou vider le pipeline (*flush*). Le compilateur doit donc considérer un ou deux cycles de retard (*jump et branch delay slots*) pour assurer l'intégrité de l'exécution. Un nombre minimum de NOp doit être insérer au besoin dans ces espaces (un pour J, deux pour BEQ);
5. Un NOp pourrait également être requis lors de lecture en mémoire (LW) (*load delay slot*) s'il y a dépendance de données avec l'instruction précédente au LW;
6. La fréquence d'opération après synthèse et optimisation sur FPGA doit être maximisée.
7. **BONUS :**

   Si réussi, ces bonus s'ajoutent à l'ensemble des projets. Notez que la note finale pour l'ensemble des projets ne peut pas dépasser 100 %.

   a. Unité de détection d'aléas de données pour LW. Cette unité détecte lorsque l'adresse d'une opérande d'une instruction est égale à celle du registre de destination de son instruction LW précédente. Cette unité doit alors suspendre (*stall*) l'exécution pour un cycle. La suspension fige le PC et le registre d'instruction pour un cycle et les signaux de contrôle à l'étage EX sont mis à zéro pour ne pas écrire dans la mémoire de données, ni dans le banc de registre (équivalent à un NOp). (10 points)

   b. Réduction du nombre de cycles de retard. La logique pour le branchement doit être devancé d'un étage pour avoir un seul cycle de retard plutôt que deux. Si possible, les instructions doivent être réordonnées pour minimiser les cycles de retard en remplissant les *delay slot* avec des instructions utiles plutôt que des NOp (5 points).

IV. Implémentation

Importer le fichier de design **`datapath.vhd`** créé dans le projet 2. Vous devez le compléter en respectant les **contraintes de conception suivantes** :

1) Le circuit doit être commandé par une et une seule horloge;
2) Tous les éléments synchrones doivent être sensible à la **transition montante** de l'horloge sauf le banc de registre qui est sensible à la transition descendante (fichier regfile.vhd fourni);
3) Lors de la synthèse dans Quartus, l'outil ne doit générer aucun message d'erreur (*error*) ou d'avertissement (*warning*) [sauf si justifié];
4) Vous devez simuler votre design avec le programme MIPS utilisé au projet 2 en prenant soin d'ajouter des NOp (utiliser l'instruction « addi $0,$0,0 » pour le NOp) pour assurer l'intégrité

de l'exécution du programme.

A. Étape 1

1) Faire un copié-collé de la 2e colonne de votre tableau 1 de la préparation 1 dans votre fichier datapath.vhd dans la partie déclarative de son architecture;
2) Ajouter les étages de registre pipeline et établir les bonnes connexions entre les étages avec les signaux (voir l'annexe 1);
3) Ajouter l'unité d'envoie (*forwarding unit).*

VÉRIFICATION 1 : Déterminer les équations de deux sorties ForwardA et ForwardB en fonction de leurs entrées en utilisant un encodeur de priorité.

B. Vérification fonctionnelle par simulation

Cette étape vise à vérifier la fonctionnalité de votre processeur MIPS avec pipeline par simulation. Le programme utilisé pour valider le MIPS monocycle dans le Projet 2 doit être utilisé. Prenez soin d'ajouter des NOp (**si nécessaire**) pour assurer l'intégrité de l'exécution du programme.

PRÉPARATION 2 : Compléter le tableau 2 du fichier Excel qui indique la valeur de signaux pour les 20 premier cycles d'exécution (de droite à gauche) du processeur MIPS avec pipeline si le programme fourni dans votre projet 2 (modifié avec les NOp) est exécuté. Déposer le fichier Excel sous « Dépôt de la préparation 2 » sous Projet 3 dans Moodle (**AVANT LE DÉBUT DE LA DEUXIÈME SÉANCE DU PROJET 3**). NE PAS CHANGER L'ORDRE DES SIGNAUX DANS LE FICHIER.

B.1. Simulation de l'entité *top*

Un banc d'essai VHDL pour l'entité *top* est inutile car le code assemblé est déjà dans la ROM `imem.vhd` (modifié avec l'insertion de NOp, si nécessaire). Simulez l'entité *top* avec ModelSim en exécutant les commandes suivantes (attention à la polarité du signal reset selon votre implémentation et veuillez écrire ces commandes (plutôt que de faire un copié=collé) :

```
vsim Top
force clk 1,0 5 ns -repeat 10 ns
force reset 1,0 25 ns
add wave –r *
run 500 ns
```

Comparer les valeurs obtenues avec les valeurs théoriques et déboguer.

VÉRIFICATION 2 : Lorsque votre vérification du bon fonctionnement de votre design par simulation avec ModelSim est terminée, faire valider votre simulation par votre chargé de laboratoire. La simulation doit être claire, lisible et doit faire apparaître (dans l'ordre sur une fenêtre « *wave* ») le résultat des signaux CLK, RESET, PC (9:2), DATAADRESS et WRITEDATA. La simulation doit exécuter avec succès tout le programme.

C. Vérification de la synthèse

Synthétiser le composant `top.vhd` avec Quartus et analyser les rapports de synthèse en prenant soin d'extraire la fréquence maximale, les ressources utilisées, les délais et position du chemin critique. Comparer les données extraites avec celles du projet 2 et expliquer les différences.

VÉRIFICATION 3 : Lorsque votre vérification des rapports de synthèse est faite, faire valider par votre chargé de laboratoire. Le *slack* doit être positif. Vous aurez à expliquer les différences de performance par rapport à celles obtenues dans le projet 2. Le barème de correction sous la section « Description HDL pour la synthèse » du document `grille_evaluation_fichier_VHDL.pdf` sera entre autres utilisé.

D. Documents à remettre (Un seul fichier zip déposé sous l'onglet « Dépôt du rapport » sous le Projet3 du site Moodle du cours)

À la complétion du projet, remettre les documents suivants dans un et un seul fichier zip et le déposer sous l'onglet « Dépôt du projet 3 » du site Moodle du cours (ena.etsmtl.ca) avec comme nom : `ELE344_PROJET3_KABI_GOUM.zip`

où KABI et GOUM sont les lettres du code permanent des membres de l'équipe.

Le fichier **Zip** doit contenir :

1) Le fichier **Tableaux.xlsx** contenant les tableaux 1 et 2, corrigés au besoin.
2) Les fichiers **VHDL**
3) Les fichiers **.do** (si utilisé)
4) Pour un projet non fonctionnel à 100 % selon les vérifications 2 et 3, un rapport obligatoirement nommé : **Rapport_Projet3_KABI_GOUM.docx.** Les détails sont présentés à la section « Détails du rapport ». Utilisez le fichier `rapport_gabarit.docx` pour la rédaction de ce rapport.

D.1. Sections du rapport pour un projet non fonctionnel

Un rapport devra être remis au cas où le projet ne serait pas complètement fonctionnel lors de la vérification 2 ou 3 et doit contenir quatre sections. Ce rapport permettra d'obtenir jusqu'à 75 % des points non obtenus aux Vérifications 2 et 3. Ce rapport doit contenir les sections suivantes :

Introduction et objectifs du projet

Description du processeur MIPS avec pipeline:

Descriptions détaillées des étages et de la logique associée.

Localisation et causes d'erreurs :

Explications des causes ayant pu mener aux erreurs de conception s'étant produites dans votre design. Ces explications doivent être appuyer par des chronogrammes des parties ne fonctionnant pas correctement. Ces chronogrammes sous forme de captures d'image doivent inclure :

- ✓ Les signaux représentés en hexadécimal.
- ✓ Les formes d'ondes doivent inclurent entre autres les signaux suivants et dans cet ordre : CLK, Reset, PC, WRITEDATA, AluResult.

Discussion et amélioration possibles:

Discussion mettant en lien les causes d'erreurs et les améliorations pouvant être apportées. Les pistes de solutions possibles devront y être expliquées.

Annexe 1 Ajout de registres en VHDL

L'ajout d'un registre en VHDL sur un signal consiste à créer un signal interne assigné dans un process synchrone et qui sera retardé d'un cycle d'horloge. Dans l'exemple suivant, le contenu des signaux `ID_EX_a` et `EX_MEM_a` sont égaux à celui de `IF_ID_a` mais retardé d'un et deux cycles d'horloge respectivement:

```
ARCHITECTURE exemple1_arc OF exemple1 IS
  SIGNAL IF_ID_a, ID_EX_a, EX_MEM_a: STD_LOGIC_VECTOR(n-1 DOWNTO 0);
BEGIN
   PROCESS (clock) BEGIN
     IF rising_edge(clock) THEN
        ID_EX_a  <= IF_ID_a;  -- retard d'un cycle d'horloge
        EX_MEM_a <= ID_EX_a; -- retard de deux cycles d'horloge
     END IF;
   END PROCESS;
END exemple1_arc;
```

Annexe 2 Barème d'évaluation du Projet 3

Critères	Description	Note	Poids (%)
Préparation 1	Tableau 1 Description des signaux		15
Préparation 2	Tableau 2 : Valeurs des signaux ou registres pour les 20 premier cycles		20
Vérification 1	Équations de l'unité d'envoi (*Forwarding unit*)		10
Vérification 2	Résultats de simulation		30
Vérification 3	Rapports de synthèse du top.vhd avec Quartus (Fréquence maximale, ressources utilisées, les délais et leurs chemins critiques) et explication des différences par rapport au projet 2		10
Rapport final	Rapport_Projet3_XXX.docx*		
VHDL	Fichiers VHDL**		15
	Total :		**100**
	Bonus* : Unité de détection d'aléas**		10
	Bonus* : Réduction du nombre de cycles de retard**		5

* Voir la description du contenu à la section D.1. Ce rapport permet d'obtenir jusqu'à 75 % des points non obtenus aux Vérifications 2 et 3.

** La forme et le contenu des fichiers VHDL sont évalués selon les barèmes spécifiés dans le fichier `grille_evaluation_fichier_VHDL.pdf`

*** Si réussi, ces bonus s'ajoutent à l'ensemble des projets. Notez que la note pour l'ensemble des projets ne peut pas dépasser 100 % de leur poids dans la note finale de l'étudiant.