

Laboratoire d'architecture des ordinateurs semestre printemps 2023 - 2024

Microarchitecture - Combinatoire - Instructions

Informations générales

Cette partie est à faire de manière **individuelle**. Vous devez rendre individuellement votre travail sur Cyberlearn (Circuit Logisim et le programme assembleur).

A l'issue de ce laboratoire, nous considérons que chaque étudiant possède tous les outils nécessaires à la réalisation de tous les laboratoires de ce cours.

 **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 données présentes sont effacées.


Objectifs du laboratoire

L'objectif de ce laboratoire sera, dans un premier temps, de prendre en main les outils nécessaires à la réalisation correcte des différentes étapes des prochains laboratoires. Lors de ces laboratoires vous aller créer un processeur complet simplifié. Le but étant que vous réalisiez chaque étape vous-mêmes afin de pouvoir mettre en pratique la théorie vue en cours.

Ce laboratoire n'est pas noté mais sert d'introduction à la méthodologie et à la prise en main des outils dont nous aurons besoin tout au long du semestre.

Outils


Pour ce laboratoire, 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 télécharger à partir du site Cyberlearn un .zip contenant un répertoire «workspace» où vous trouverez :

- **labo_processeur.circ** : Le fichier de travail Logisim
- **main.S** : fichier source du code assembleur
- **Makefile** : fichier contenant les directives d'assemblage

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

Workspace fourni

Ce workspace sera utilisé uniquement lors de ce laboratoire d'introduction.

Vous allez recevoir un circuit qui contient :

- Mémoire d'instructions
- Processeur_ARO2 (très incomplet, contient seulement le PC)

1 Laboratoire instructions

Travail à effectuer

Entité du bloc pour la mémoire d'instructions



Nom I/O	Description
addr_i	Adresse de l'instruction courante
instr_o	Intruction courante à envoyer au processeur

Le processeur doit avoir accès à une mémoire d'instructions. C'est le processeur qui sera chargé de donner l'adresse courante afin que la mémoire lui retourne l'instruction à traiter.

Dans cette première partie, vous allez instancier 2 types de mémoires et vérifier laquelle est la plus adaptée au processeur implémenté dans les laboratoires d'ARO2.

Ouvrir le circuit Logisim qui vous a été fourni.

Etape 1 : Implémenter un composant ROM sur 8 bits comme mémoire d'instructions

Dans le composant *instr_mem_8b*, instancier une ROM qui a les caractéristiques ci-dessous, puis connecter les entrées/sorties de la ROM aux entrées/sorties du composant.

- Taille du bus d'adresse : **15 bits**
- Taille du bus de donnée : **8 bits**

Etape 2 : Implémenter un composant ROM sur 16 bits comme mémoire d'instructions

Dans le composant *instr_mem_16b*, instancier une ROM qui a les caractéristiques ci-dessous, puis connecter les entrées/sorties de la ROM aux entrées/sorties du composant.

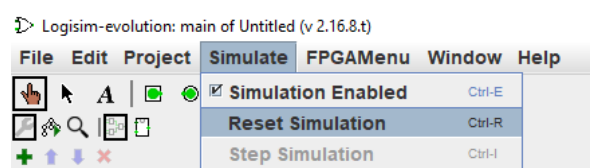
- Taille du bus d'adresse : **15 bits**
- Taille du bus de donnée : **16 bits**

Ces valeurs vous sont fournies car elles sont définies par rapport au contexte et au processeur réalisé. Il s'agit ici des contraintes qu'il a fallu définir.

Pour information : La taille du bus d'adresse permet directement de définir la taille du programme que nous pouvons exécuter dans le système.

Pour information : La taille du bus de donnée permet directement de définir, de manière indirecte, le nombre de registres ainsi que le nombre d'instructions supportés par le système.

NOTE : Il se peut que les adresses et les données de la ROM ne se mettent pas à jour. Si cela vous arrive, faites un reset de la simulation.



Etape 3 : Configurer l'incrémentation du PC dans le bloc Processeur_ARO2

Dans le composant *processeur_ARO2*, modifier la constante d'incrémentation du PC afin d'avoir un incrément de 1 à chaque cycle d'horloge.


Etape 4 : Compiler le programme assembleur fourni

Exemple d'un programme assembleur (fichier main.S) :

```
mov r1 , #5
add r0 , r2 , r3
```

Vous allez compiler le programme qui vous a été fourni pour ce laboratoire. Pour cela vous avez deux options :

- **Geany** : Avec cet éditeur de texte, vous pouvez compiler avec le menu : Build -> Make.
- **terminal** : Placez-vous dans le dossier qui contient votre programme et tapez : make

 **Il ne faut absolument pas modifier le fichier Makefile**

Après la compilation, vous devez obtenir les fichiers "*main.lss*" et "*main.raw*".

Etape 5 : Charger le programme compilé dans les mémoires d'instructions

Dans le composant *instr_mem_8b*, aller sur la mémoire d'instructions :

1. clic-droit->clear contents : efface le contenu
2. clic-droit->load image : charge une image en mémoire
3. sélectionner le fichier main.raw

Vous devriez voir des valeurs dans la mémoire ROM d'instructions.

Faites la même chose dans le composant *instr_mem_16b*.

Etape 6 : Comparer le contenu dans les mémoires avec les valeurs d'instructions compilées

A l'aide du fichier main.lss, comparer les valeurs écrites dans les mémoires ROM d'instructions 8 bits et 16 bits avec le programme fourni.

Les codes d'instructions correspondent-ils ?

Quelle taille de ROM semble le plus adaptée ?

Etape 7 : Tester pas à pas afin d'observer le fonctionnement

Simuler le circuit afin de vérifier son fonctionnement, pour ceci vous allez utiliser l'outil «**Assembly viewer**» de Logisim qui permet d'afficher le fichier *main.lss* avec l'instruction courante automatiquement surlignée :

- Se placer dans le circuit main pour le simuler, puis utiliser le bouton "show simulation hierarchy" en haut à gauche de la fenêtre. Vous pouvez ensuite vous déplacer dans l'arborescence du circuit pendant la simulation.
- Depuis le menu **Simulate** ouvrir l'outil «**Assembly viewer**».
- Puis charger le fichier *main.lss* : File -> Open lss file.
- Ensuite se placer dans le composant *processeur_ARO2* et sélectionner le registre PC à l'aide du bouton **Get Registers**.
- Revenir dans le circuit main et visualiser les sondes sur les adresses et données, ainsi que l'instruction courante dans le fichier *main.lss*.
- Initialiser le PC (ctrl+r dans logisim ou alors simulate->reset simulation).
- Avancer pas à pas dans le programme en utilisant la touche F2 (un appuie) pour avancer d'un cycle d'horloge. Assurez-vous que F2 fait des sauts d'une période, et pas d'une demi période en allant dans Project > Options > Duration of Main tick (F2) Full period.
- Vérifier le comportement des adresses (vision processeur et mémoire) et des données en sorties des mémoires d'instructions pour chaque cycle d'horloge.

Observer la différence entre la valeur de l'adresse avec la vision depuis le processeur et depuis la mémoire. Quelle opération est faite entre ces 2 valeurs ?

Ceci a-t-il un lien avec l'incrément du PC ? Si oui, qu'elle doit être la valeur de l'incrément ?

Pour un bon fonctionnement, à chaque cycle d'horloge on doit accéder à une nouvelle instruction, qui doit être l'instruction suivantes dans le programme.

Le programme fonctionne-t-il correctement ?

Etape 8 : Modifier l'incrément du PC dans le bloc Processeur_ARO2

Dans le composant *processeur_ARO2*, modifier la constante d'incrément du PC afin d'avoir un incrément de 2 à chaque cycle d'horloge.

Etape 9 : Tester pas à pas afin d'observer le fonctionnement

Avec le nouvelle incrément, simuler pas à pas le circuit comme décrits dans l'étape 7.

Le programme fonctionne-t-il correctement ? Quelle taille de ROM est le plus adapté à notre processeur ?

Etape 10 : Ecrire un programme simple qui utilise quelques instructions supportées

Modifier le fichier `main.S` et utiliser les fonctions *lsr* (Logical Shift Right), *mov* et *add* afin de réaliser un algorithme qui effectue l'opération suivante :

$$R = (A + B + C + D)/4$$

Votre programme doit comporter au début, une partie déclarative qui permet de changer simplement les valeurs à tester (utilisation de *mov*) pour les valeurs A, B, C et D.

NOTE : Les différentes valeurs seront considérées comme des valeurs non-signées sur 16 bits. Considérer cette information lorsque vous choisissez les valeurs de tests. L'objectif de ce laboratoire n'est pas de devoir observer ce qui se passe en cas de dépassement mais que vous compreniez bien comment s'exécute le programme.

NOTE : Un manuel des instructions ARM que nous allons supporter se trouve sur Cyberlearn. Il sert de référence quant à l'utilisation des différentes instructions.

Etape 11 : Compiler le programme assembleur écrit à l'étape 10

Compiler le programme assembleur que vous venez d'écrire afin d'obtenir les fichiers "`main.lss`" et "`main.raw`".

Etape 12 : Tester avec l'émulateur python du processeur

Une version simplifiée du processeur a été implémentée en python afin que vous puissiez observer l'effet sur les registres. Le but étant de vérifier que vous arriviez correctement à réaliser les opérations demandées. Valider le fonctionnement de votre programme avec l'émulateur. Voici les quelques étapes à suivre pour lancer cette partie :

- Télécharger l'émulateur python sur la page Cyberlearn.
- Installer les packages python nécessaires en rentrant les commandes suivantes dans un terminal :

```
$ pip3 install pysimplegui
$ sudo apt-get update
$ sudo apt-get install python3-tk
```

- Lancer l'émulateur depuis un terminal avec la commande :

```
$ python3 proc_aro_emu.py
```

Accepter la période d'évaluation de 30 jours concernant l'outil PySimpleGUI.

Il faut maintenant fournir à l'émulateur le fichier "`main.raw`" que vous avez généré à l'étape 11. Ensuite vous pourrez exécuter séquentiellement les instructions afin de voir l'effet sur les registres.

Conclusion

Lorsque vous arrivez à cette étape, nous considérons que vous avez correctement pris en main les outils nécessaires pour les laboratoires.