# Projet de Conception de processeur

Université de Cergy-Pontoise

Benoît Miramond,Ghilès Mostafaoui

9 novembre 2012

## 1 Présentation générale du projet

### 1.1 Objectifs et évaluation du projet

L'objectif de ce Projet est de rassembler les différentes notions du cours pour réaliser un petit processeur rudimentaire. Ce Projet en binôme débutera durant les séances de TP encadrées et sera évalué par l'intermédiaire d'une soutenance orale et d'un rapport de projet d'environ 10 à 15 pages.

La soutenance orale sera présentée en 10 minutes par quelques planches powerpoint (pas plus de 10 ! ) suivant 4 parties :

- **Spécification du processeur**. Cette partie identifiera les caractéristiques du processeur en s'inspirant de ceux vus en cours.
- **Organisation du processeur**. Cette partie donnera les schémas en bloques du processeur de manière hiérarchique.
- **Jeu d'instruction du processeur**. Vous présenterez dans cette partie les différentes instructions machine permettant de piloter le processeur. Vous classifierez les instructions en fonction de leur type.
- **Améliorations apportées**. Vous présenterez l'objectif de vos modifications, le choix architectural proposé, les changements réalisés et les problêmes rencontrés.

Prévoyez une **Démonstration** de 5minutes. Il s'agira de faire la simulation du circuit conçu avec l'outil Logisim.

La présentation pourra être suivie de 5 minutes de questions.

### 1.2 Caractéristiques du processeur

Pour réaliser ce processeur, nous allons utiliser les composants déjà développés et les assembler autour d'un chemin de données. Evidemment, les caractéristiques finales de votre processeur ne seront pas très compétitives face à celles des processeurs actuelles qui mettent en jeu des mécanismes plus complexes comme le pipeline, la prédiction de branchement, ...
Pourtant les principes utilisés sont ceux de tous les processeurs actuels et sont donc la base primordiale à la construction de processeurs plus complexes. Ce processeur que nous appellerons le Cergy-Pontoise Unit Version 1 (CPU_V1) sera composé (voir la figure ci-dessous) :

- d'une unité de calcul (UAL),
- d'un banc de 4 registres généraux,
- d'une unité d'adressage,
- et d'un contrôleur.

Le CPU_V1 travaillera sur des mots de 4 bits.
Le CPU_V1 aura 2 bus de communication avec 2 mémoires externes :

- La mémoire de donnée,
- et la mémoire d'instructions

Ces deux mémoires sont les seuls composants que vous ne réaliserez pas par vous même, vous pourrez utiliser les composants RAM disponible sous Logisim.

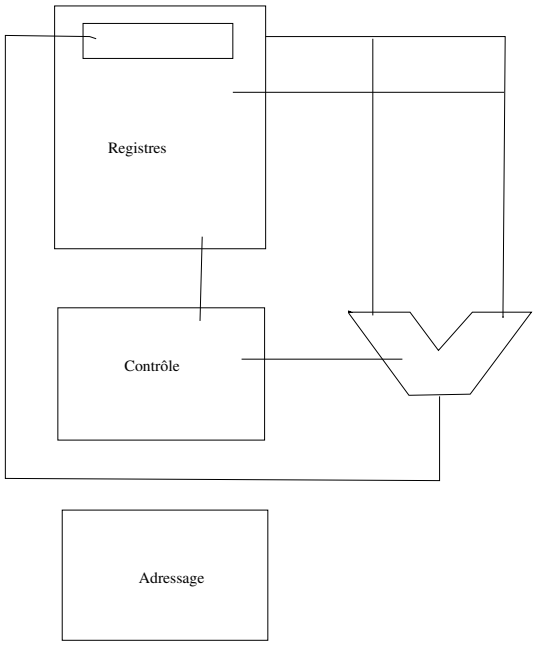


FIGURE 1 – Disposition des composants du chemin de données du CPU_V1.

## 1.3 Objectifs des séances

Il vous reste maintenant 2 séances de TP (TP 2 et TP 3) encadrées. Vous devrez donc terminer ce projet en utilisant les salles en libre service du Département (ou bien chez vous).
De manière à structurer votre travail, il vous est proposé un guide de travail qui vous indique comment découper votre conception en 4 étapes :

1. Séance TP 2 : Définir des interfaces des composants et débuter le développement des différents parties du CPU_V1 à partir de ceux déjà réalisés lors de la séance TP 1,
2. Séance TP 3 : Finir de développer les coeurs des composants,
3. Séance TP 3 : Commencer à connecter les composants,
4. Projet : validation des composants.

### 1.3.1 Détail des sous séances

L'objectif de la séance TP 2 est :
- de finir la réalisation de l'interface des 4 composants principaux du processeur pour comprendre son fonctionnement,
- de développer l'ALU (normalement déjà fait lors du TP 1 !),
- de concevoir le coeur du composant "Banc de registres" (au besoin à terminer chez soi)
- développer le coeur de l'unité d'adressage (sur la base des registres développés pour le banc de registre),

A la séance de TP 3, vous devrez :
- instancier et relier ces composants dans le coeur du processeur.
- développer les unités de décodage du contrôleur,
- réaliser l'incrémentation du compteur ordinal (PC),
- à instancier le contrôleur dans le coeur du processeur et à le relier aux autres composants.

Il vous restera durant votre projet à :
- ajouter des mécanismes de Debug dans le processeur (LEDs),
- instancier le processeur,
- le relier aux mémoires de données et d'instructions,

– vérifier les résultats obtenus sur un bench [1].

# 2 Interface des composants et chemin de données autour de l'ALU

Vous avez déjà réalisé une ALU durant la séance précédente. Vous allez maintenant intégrer cette ALU dans un environnement de travail complet. A savoir, dans un premier temps la connexion de l'ALU avec le banc des registres généraux.

**IMPORTANT** :
– Pour chaque composant à concevoir, vous utilisez une méthode de conception classique qui consiste à d'abord définir l'interface de ces composants avant de s'intéresser à l'implantation interne de chacune de ces parties (Comme les prototypes en C, ou les interfaces en Java)
– Il est fortement conseillé de lire le projet en entier (notamment la section : Implantation des composants) avant de réaliser vos interfaces afin de vous assurer que votre interface est en adéquation avec les spécifications du CPU_V1

Les composants à réaliser dans ce chemin de données sont décrits ci-dessous.

## 2.1 ALU

Définition d'une boîte ALU vide, mais dont l'interface est correcte. On supposera pour l'instant qu'elle ne peut faire que 8 opérations différentes

A noter que vous ajouterez 4 ports de sorties supplémentaires par rapport aux implantations étudiées de manière à ajouter les indicateurs ainsi que d'éventuelles extensions.

## 2.2 Banc de registres

Définition d'un modèle vide de registre 4 bits, dont l'interface est la suivante :
– 1 sortie sur 4 bits, notées $X_i$,
– 1 sortie sur 4 bits, notées $Y_i$,
– 1 entrée sur 4 bits, notées $R_i$,
– 1 signal de contrôle de lecture sur 1 bit, notées $selX_i$,
– 1 signal de contrôle de lecture sur 1 bit, notées $selY_i$,
– 1 signal de contrôle d'écriture sur 1 bit, noté $selR_i$, avec $i \in [0, 3]$
– plus 2 ports d'extension,
– et évidemment l'horloge.

## 2.3 Liaison entre l'ALU et le banc de registres

A la phase d'implantation, les boîtes registres contiendront des bascules D qui pourront écrire sur 2 bus en entrées de l'ALU (X et Y) et dans lesquels on pourra écrire à partir du bus de sortie de l'ALU : R.
Cette explication est nécessaire pour comprendre les interactions entre composants mais vous ne connecterez l'ensemble que lorsque l'implantation du coeur de chacun sera terminée. Notre chemin de données dispose de 4 registres de 4 bits.
Pour simplifier la connectique dans le banc de registre, vous disposerez les composants de la manière décrite dans la figure 2.

1. un jeu de données

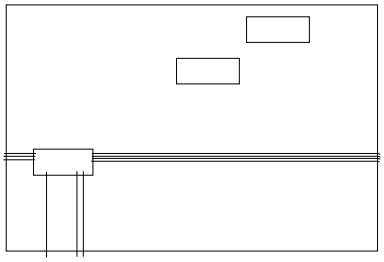

FIGURE 2 – Disposition des registres dans le banc de registre.

## 2.4 Unité d'adressage

Pour communiquer avec la mémoire, le processeur dispose d'une unité d'adressage.
L'interface de cette unité doit être déduite à partir de la figure 3 et de l'interface donnée du banc de registre.
Ajouter comme pour tous les composants 2 ports d'extension.

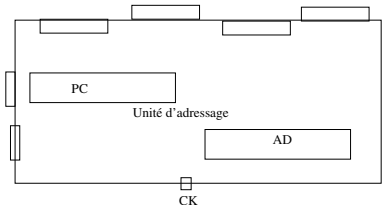


FIGURE 3 – Interface de l'unité d'adressage.

## 2.5 Unité de contrôle

### 2.5.1 Rôle et principe de fonctionnement

L'unité qui contrôle l'exécution des instructions machines par le processeur a pour rôle de placer les valeurs des différents signaux de commande de l'architecture à chaque cycle.

Le principe de fonctionnement de l'unité de contrôle est de lire le contenu du registre d'instruction, de décoder son contenu et d'en déduire le positionnement des différents signaux de l'architecture (ALU, sources, destinations, accès mémoire).
L'unité de contrôle est aussi responsable de l'exécution séquentielle des instructions en mémoire. Elle lit donc à chaque cycle d'horloge le contenu du PC (compteur ordinal), l'incrémente et renvoie vers l'unité d'adresse dans le registre PC.
Elle réagit également au signal RESET en recommençant l'execution à une adresse câblée d'initialisation. A vous de réfléchir à la réalisation de cette réaction au signal RESET.
Enfin, l'unité de contrôle peut gérer la présence d'un "immédiat" dans le corps de l'instruction. Cet immédiat peut être de deux types, soit directement une donnée (Ex : Add R1, Imm), soit une adresse à laquelle le processeur ira chercher une donnée (Ex : LOAD R1, @Imm).
Dans le second cas, il suffit de décoder le code opération de l'instruction, si celui si correspond à l'instruction LOAD, alors le contrôle renvoie l'immédiat vers l'unité d'adressage dans le registre AD. Pour réaliser cette fonctionnalité, référez vous à la table du jeu d'instruction du processeur (ci-dessous).
Nous ne gérons pas pour l'instant les immédiats de données.

### 2.5.2 Le contrôle du chemin de données

Pour exécuter une instruction dans ce processeur, il faut disposer les valeurs de commande pilotant :
– l'UAL par 3 signaux de commande

- le choix de l'opérande X de l'ALU parmi 4 registres, donc 4 bits de commande
- le choix de l'opérande Y de l'ALU parmi 4 registres, donc 4 bits de commande
- le choix de la destination du rangement parmi 4 registres, donc 4 bits de commande
- le choix d'un type d'accès mémoire (vers la mémoire) : Ecriture ou Lecture et le choix de faire un Fetch de l'instruction, soit 3 nouveaux signaux de commande

Le positionnement de ces signaux de contrôle dépend évidemment d'une donnée externe qui est l'instruction qui vient d'être chargée depuis la mémoire et qui doit apparaître donc en entrée de l'interface du processeur. L'unité de contrôle a également pour but de fournir l'adresse de la prochaine instruction à l'unité d'adressage. On doit donc avoir une entrée PC sur 4 bits et une sortie PC+ sur 4 bits. L'unité de contrôle réagit aussi au RESET (voir ci-dessus).

A vous d'identifier son interface en fonction des composants déjà placés, c'est-à-dire en relevant tous les signaux de commandes parvenant aux 3 autres composants plus les signaux de commande externes.

### 2.6 Ports externes de communication du processeur

Le processeur dispose de 2 Bus externes de communications avec la mémoire, l'un pour les données qui est en double sens de transfert (lecture/écriture) et l'autre qui est en lecture seul pour les instructions. Cette distinction entre bus données et bus instructions le classe comme une architecture dite de Harward. C'est donc dans l'unité de contrôle que l'on trouvera le registre instruction.
Vous devez donc disposez autant de ports que le nécessite l'interface externe du processeur, soit

- 4 bits de données entrantes,
- 4 bits de données sortantes,
- 4 bits d'adresse de données,
- 12 bits d'instructions,
- 4 bits d'adresses d'instructions,
- 1 signal de requête en lecture,
- 1 signal de requête en écriture,
- 1 signal de requête de Fetch,
- l'horloge,
- 1 signal RESET sur 1 bit.
- plus 2 ports d'extension

ATTENTION : on ne vous demande pas ici de créer un composant il s'agit juste de vous préciser les différents signaux de communication entre le processeur et les mémoires externes.

La connectique étant importante à ce niveau, la réelle difficulté lors de l'étape d'instanciation sera la rigueur de placement des composants que vous devrez adopter.
La lisibilité du schéma en bloques du processeur comptera dans la notation puisque les différents schémas des composants réalisés devront apparaître dans les planches de la soutenance (capture d'écran).

## 3 Implantation des composants

### 3.1 Introduction

Cette nouvelle spécification du projet correspond à la phase d'implantation de chacun des blocs disposés dans le chemin de données. L'objectif sera maintenant de développer le coeur des composants ALU, banc de registres, unité d'adressage et unité de contrôle. Les sections ci-après précisent le contenu et le comportement de ces blocs. A vous d'en faire le développement et de faire les choix d'architecture qui complèteront cette spécification **volontairement incomplète**.
Par ailleurs, vous devrez choisir une modification personnelle à ajouter à votre processeur parmi celles qui vous sont proposées. Pensez à choisir ces modifications au plus tôt dans votre conception de manière à directement prendre les bonnes décisions architecturales.

## 3.2 UAL

L'UAL que nous utiliserons dans ce processeur sera assez simple et correspond à celle déjà développée en TP. A savoir, cette UAL dispose de 3 signaux de commande pour choisir l'opération effectuée sur ses deux opérandes A et B. Les 8 configurations des signaux de commandes $F_0$, $F_1$, $F_2$ ne sont pas toutes utilisées puisque l'UAL développée ne réalise que l'addition (code 000), le OU (code 001), le ET (code 010) et la négation de l'entrée B (code 011). Le signal $F2$ sera éventuellement utilisé dans la section 7.
*Note : Vous donnerez dans le rapport le tableau indiquant la correspondance entre la valeur de ces signaux et l'opération effectuée.*
La réalisation de cette UAL sur 4 bits est donc la même que celle déjà réalisée sauf si vous choisissez de faire des modifications personnelles.

## 3.3 Registres généraux

Le chemin de données est composé de 4 registres généraux A, B, C, D de 4 bits. Ces registres sont accessibles soit en lecture soit en écriture. Le choix d'écriture dans un registre est commandé par un signal $wA$, $wB$, $wC$ ou $wD$ (appelé également $selX_i$ dans l'énoncé précédent).
On peut par ailleurs demander la lecture de chaque registre sur le Bus X ou sur le bus Y de façon séparée.
Le principe d'un bus est d'autoriser l'écriture sur un même signal par plusieurs sources. Pour éviter les

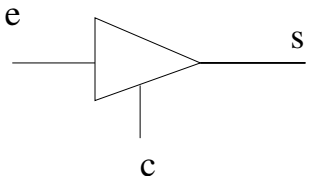


FIGURE 4 – Porte 3 états.

conflits d'écriture, chaque source est relié au bus par l'intermédiaire d'une porte appelée *Three-State* ou porte à trois états. Ces portes disposent de 2 entrées : la donnée, un signal de commande et une sortie. Vous pouvez instancier ce modèle de porte depuis la barre des tâches : figure 4. La sortie peut prendre comme son nom l'indique trois valeurs, ou trois états :

| Entrée | Commande | Sortie |
|---|---|---|
| 0 | 0 | HI |
| 0 | 1 | 0 |
| 1 | 0 | HI |
| 1 | 1 | 1 |

Lorsque le signal de commande est à 1 l'entrée est propagée sur la sortie, sinon, la sortie est dite en haute impédance ce qui correspond à une déconnexion du signal d'entrée. En activant un seul signal à écrire sur le bus à un instant donné, on évite les problème de conflit d'écriture.
Le schéma d'un registre est donc celui de la figure 5. Vous devez instancier 4 registres dans le banc de registres.
La sortie du registre D sera spécifiquement reliée directement à la sortie de données du processeur.

## 3.4 Unité d'adressage

L'unité d'adressage est uniquement composée de 2 registres 4 bits : PC et AD.
- PC est le registre d'adresse d'instructions,
- AD est le registre d'adresse de données.

Vous pouvez instancier à nouveau le modèle de registre que vous avez développé précédemment. Une seule des 2 sorties sera alors utilisée et directement reliée aux sorties PC_out et AD_out (cf. figure 2). Le signal de commande de lecture pour cette sortie (rPC, rAD) sera toujours actif. Les signaux de commande d'écriture seront reliés à l'interface de l'unité d'adressage (voir ci-dessous).

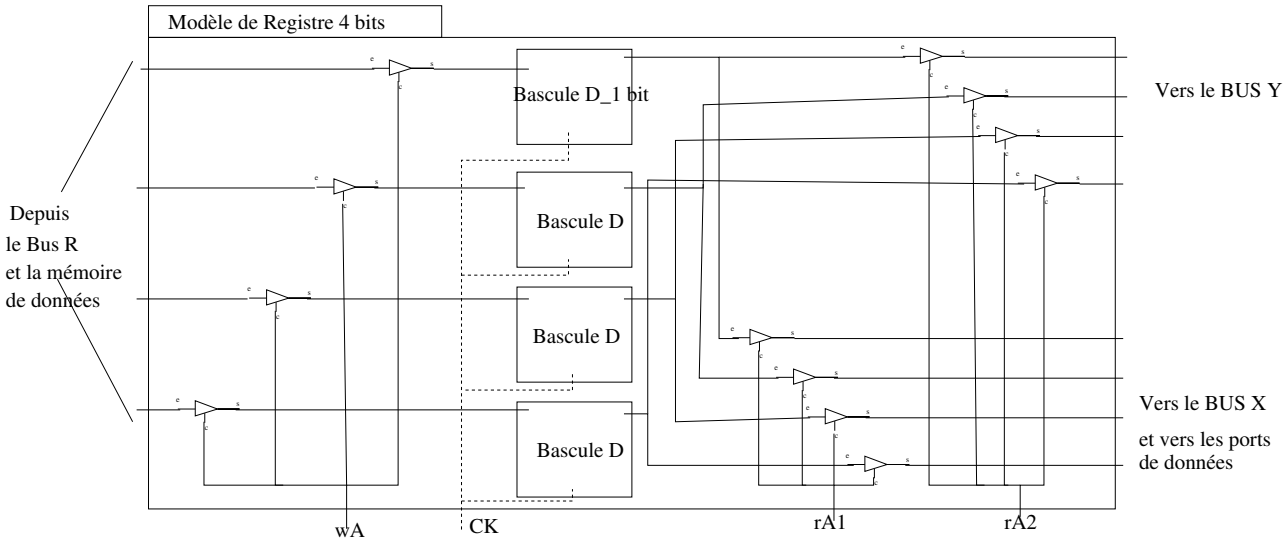


FIGURE 5 – Implantation d'un registre.

## 3.5 Unité de contrôle

L'unité de contrôle du processeur vient activer tous les signaux de commande du processeur en fonction de l'instruction qui a été chargée depuis la mémoire. Cette unité contient donc le registre d'instruction qui contient les différents champs correspondant aux catégories de commandes du processeur. Ces champs dépendent du format de l'instruction reçue, soit un format *Registres*, soit un format *Immédiat*.
Le format Registre est composé des champs suivants :

- Un champs CodOp, qui définit l'opération à réaliser. Ce champs sera sur 4 bits,
- Un champs de désignation de la source sur le bus X. Ce champs sera sur 2 bits puisque l'on peut choisir parmi A, B, C ou D.
- Un champs pour désigner la source écrivant sur le bus Y de même longueur que le précédent.
- Un champs pour désigner la destination du résultat, sur 2 bits également, les destinations possibles étant les mêmes que les sources.
- Un champs d'extension sur 2 bits.

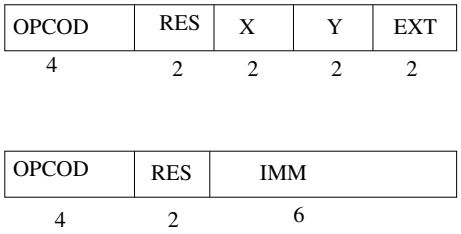


FIGURE 6 – Formats d'instruction.

Ce format est par exemple utilisé pour commander une opération ADD A, B, A qui fait A<= A+B.
Le format Immédiat est lui composé de 4 champs :

- Un champs CodOp, sur 4 bits,
- un champs registre qui désigne soit un registre source (STORE A, 0xA5), soit un registre destination (LOAD A, 0xA5).
- Un champs immédiat sur 4 bits.
- Un champs d'extension sur 2 bits.

Chaque instruction est donc codée sur 12 bits quelque soit le format employé.
Vous devez donc créer un modèle de registre sur 12 bits.
C'est ensuite le rôle de l'unité de contrôle de décoder cette instruction pour activer les signaux du processeur. Le schéma du coeur de cette unité est donné en figure 7. Le jeu d'instruction de ce processeur

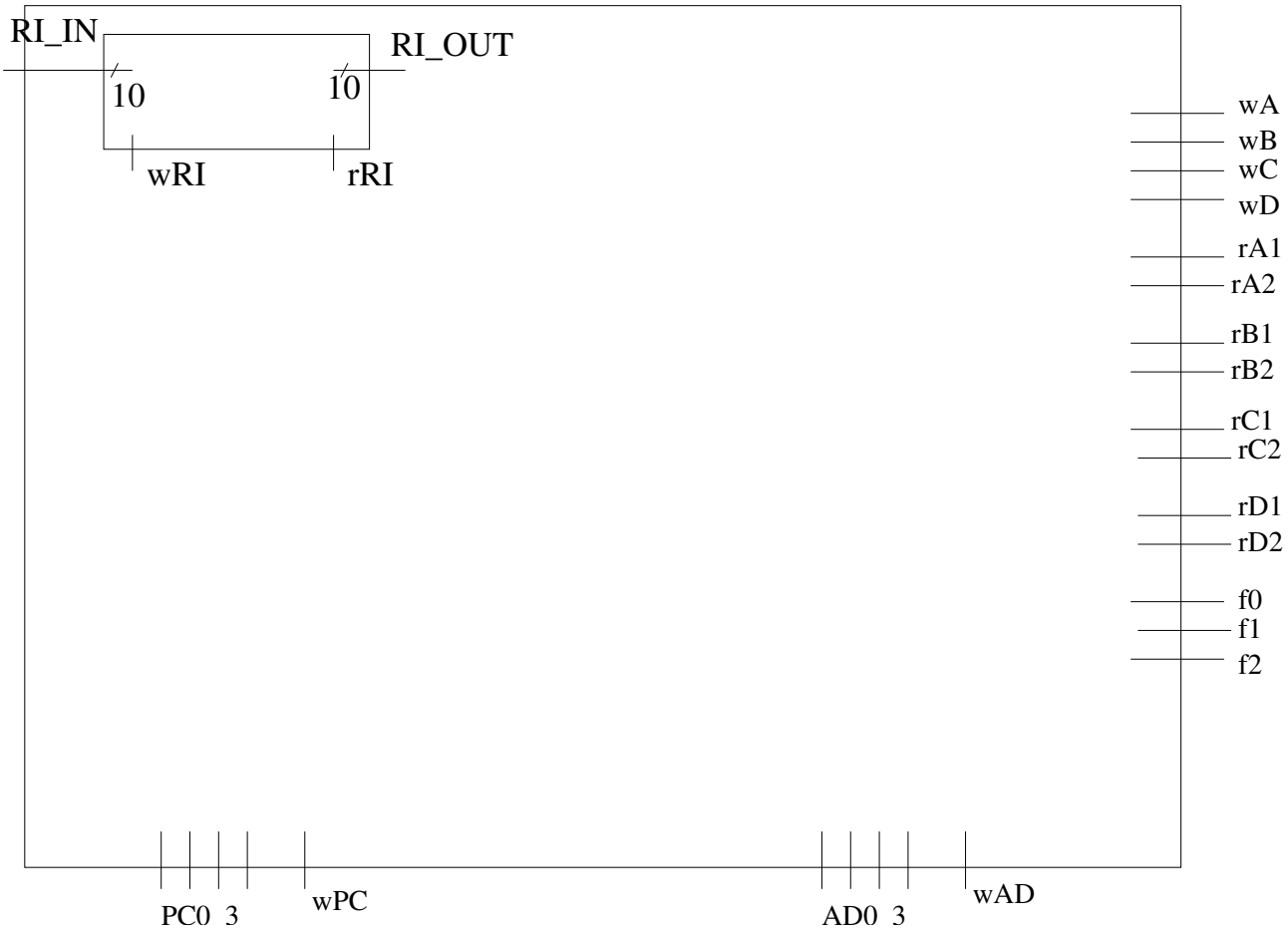


FIGURE 7 – Unité de contrôle du processeur.

est donné dans la table suivante :

| Mnémonique d'instruction | Code Opération | Format (**R**egistre ou **I**mmédiat) | Exemple |
|---|---|---|---|
| ADD | 0000 | R | ADD A, B, C |
| OR | 0001 | R | OR B, A, C |
| AND | 0010 | R | AND B, B, C |
| NOT | 0011 | R | NOT C, B, C |
| LOAD | 0100 | I | LOAD B, Ox67 |
| STORE | 1000 | I | STORE C, Ox2E |

Chaque instruction correspond à l'activation ou non des signaux de l'unité de contrôle. Votre rôle est de réaliser les 5 circuits qui décodent l'instruction ($ALU\_Function$, $REG\_ReadX$, $REG\_ReadY$, $REG\_Write$, $MEM\_Read/Write$) vers les signaux de commande selon les tableaux qui suivent.

Activation des signaux de commande de l'unité de contrôle (circuits $ALU\_Function$, $MEM\_Read/Write$) :

| Mnémonique | CodOp | $F_2$ | $F_1$ | $F_0$ | Read | Write | wAD | Fetch | wPC |
|---|---|---|---|---|---|---|---|---|---|
| ADD | 0000 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| OR | 0001 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| AND | 0010 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| NOT | 0011 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| LOAD | 0100 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| STORE | 1000 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Activation des signaux de lecture dans les registres généraux (circuits $REG_ReadX$ et $REG_ReadY$) :

| Source | Code dans l'instruction | $rA_i$ | $rB_i$ | $rC_i$ | $rD_i$ |
|---|---|---|---|---|---|
| A | 00 | 1 | 0 | 0 | 0 |
| B | 01 | 0 | 1 | 0 | 0 |
| C | 10 | 0 | 0 | 1 | 0 |
| D | 11 | 0 | 0 | 0 | 1 |

Activation des signaux d'écriture dans les registres généraux (circuit $REG_Write$) :

| Destination | Code dans l'instruction | $wA$ | $wB$ | $wC$ | $wD$ |
|---|---|---|---|---|---|
| A | 00 | 1 | 0 | 0 | 0 |
| B | 01 | 0 | 1 | 0 | 0 |
| C | 10 | 0 | 0 | 1 | 0 |
| D | 11 | 0 | 0 | 0 | 1 |

Le plus gros du travail de ce projet se situe sur la réalisation de l'unité de contrôle. A vous d'être particulièrement attentif sur sa réalisation.
A chaque cycle

- elle positionne les commandes de l'architecture de la faÁon décrite dans les tableaux précédents,
- elle lit le contenu du registre PC, l'incrémente et range le résultat dans le registre PC,
- elle indique s'il y a accès mémoire : Read, Write, Fetch et positionne les registres d'adresse à la bonne valeur.
- si le signal RESET est actif, elle positionne le PC à la valeur de Boot, constante enregistrée dans un registre 4bits.

## 3.6 Communication avec la mémoire

Pour tester votre processeur, vous devez enfin le faire communiquer avec 2 mémoires : une pour les données, et une pour le programme. Le composant mémoire est disponible dans la bibliothèque Memory sous les noms RAM et ROM. Vous pouvez paramétrer la taille des mots et la taille des adresses. La mémoire de données contient des mots de 4 bits adressables sur 4 bits. La mémoire d'instructions contient des mots de 12 bits adressables sur 4 bits.

Vous devez pour finir remplir la mémoire d'instruction d'une suite d'instruction en héxa ou en binaire.

## 3.7 Modifications personnelles

Pour personnaliser votre processeur vous devez choisir 1 des modifications proposées ci-dessous et explicitées plus loin. Pour chaque modification apportée, vous dicuterez dans votre rapport de la spécification précise de ces modifications, de leur réalisation et expliquerez vos choix architecturaux.

- Extension des opérations de l'UAL,
- Instruction de branchement,
- Instruction de saut conditionnel,
- Instructions arithmétiques et logiques sur registres et Immédiat,
- Indicateurs de l'UAL.

### 3.7.1 Extension des opérations de l'UAL

Vous devez ajouter l'opération de soustraction dans l'UAL et faire en sorte que cette nouvelle instruction du jeu d'instruction du $CPU\_VI$ puisse être exécutée. Vous ajouterez également un signal Z de sortie à l'ALU indiquant si le résulat des opérations de l'ALU est nul ou non.

### 3.7.2 Instruction de branchement

Ajouter à l'architecture du processeur les mécanismes nécessaires à l'ajout de l'opération de branchement JMP Address.

### 3.7.3 Instruction de saut conditionnel

Ajouter à l'architecture du processeur les mécanismes nécessaires à l'ajout de l'opération de branchement BEQ R, Adress. Cette instruction saute à l'adresse indiquée si le contenu du regitre R est nul.
L'ajout de cette instruction impose que vous ayez ajouté la modification 1.

### 3.7.4 Instructions AL sur Immédiats

Pour l'instant les opérations AL ne se font qu'entre registres (ex : Add A, B, A). Si vous choisissez cette amélioration, vous devez faire en sorte qu'une des 2 opérandes soit un Immédiat, par exemple permettre l'opération Add A, 0x3C qui additionne A et 0x3C et range le résultat dans A.
*Note : Une fois cette modification faite, elle sera appicable à toutes les instructions AL.*

### 3.7.5 indicateurs de l'UAL

Ajout d'un signal d'indication d'OVF, d'un signal de retenue sortante, d'un signal de résultat négatif dans l'ALU qui provoquent l'allumage d'une diode s'ils sont actifs.
Vous devez également ajouter 2 signaux d'entrée de l'UAL : enA et enB qui, s'ils sont actifs, autorisent la prise en compte des entrées A et/ou B. Dans le cas contraire, les entrées sont à 0.
Relier ces signaux à l'unité de contrôle de manière à ce qu'ils apparaissent dans l'instruction dans le champs EXT.