UNIVERSITÉ
de Cergy-Pontoise

TP/Projet : Conception du processeur

Université de Cergy-Pontoise

J. Lorandel

- **Présentation générale du projet**

1. Objectifs et évaluation

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 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 celles vues en cours.

–**Organisation du processeur**. Cette partie donnera les schémas en blocs 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 problèmes rencontrés.

Prévoyez une démonstration de 5 minutes. Il s'agira de faire la simulation du circuit conçu avec l'outil Logisim. La présentation pourra être suivie de 5 minutes de questions.

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.

2. Caractéristiques du processeur

Pour réaliser ce processeur, nous allons utiliser les composants déjà développés durant la séance précédente 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 actuels 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 des éléments suivants (voir la figure ci-dessous) :

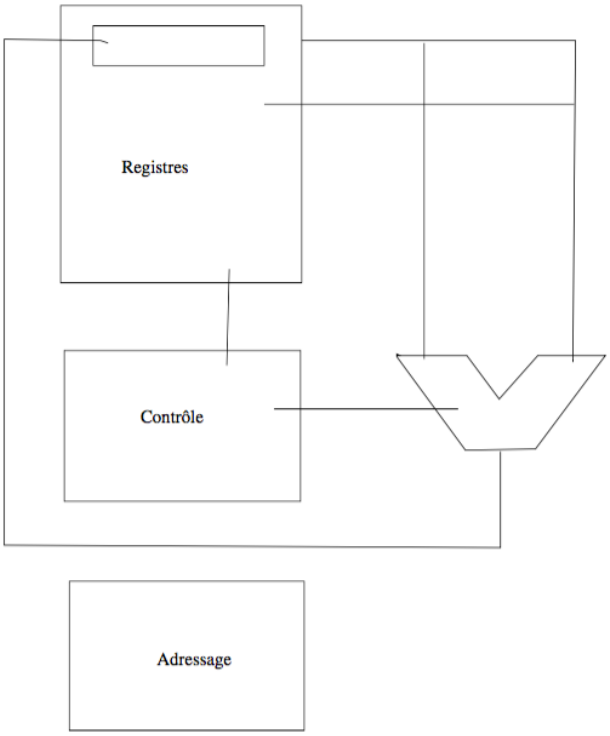

Figure 1 : Disposition des éléments du chemin de données du CPU_V1

– 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.

- **Jeu d'instructions**

Le Cergy-Pontoise Unit supportera des instructions sur 12 bits. Deux formats d'instructions sont possibles, soit un format *Registres*, soit un format *Immédiat*, illustrés sur la figure ci-dessous.

Le format Registre est composé des champs suivants :

– Un champ OPCOD, qui définit l'opération à réaliser. Ce champ sera sur 4 bits,

– Un champ de désignation de la source sur le bus X. Ce champ sera sur 2 bits puisque l'on peut choisir parmi A, B, C ou D.

– Un champ pour désigner la source écrivant sur le bus Y de même longueur que le précédent.

– Un champ 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.

Ce format est par exemple utilisé pour commander une opération ADD A, B, A qui fait A<= A+B.

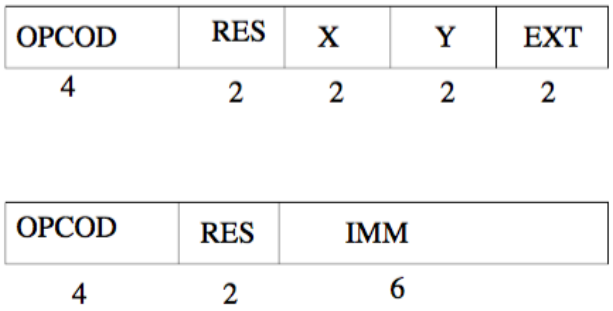

Figure : Formats d'instruction

Le format Immédiat est lui composé de 4 champs :

– Un champ OPCOD, sur 4 bits,

– Un champ registre qui désigne soit un registre source (STORE A, 0xA5), soit un registre destination (LOAD A, 0xA5).

– Un champ immédiat sur 4 bits + Un champ d'extension sur 2 bits.

Chaque instruction est donc codée sur 12 bits quel que soit le format employé. On pensera donc à créer un modèle de registre sur 12 bits pour le registre d'instruction.

- **Communication avec les mémoires**

Un processeur seul ne fonctionne pas. Il est nécessaire de le connecter à deux mémoires. Une mémoire de données et une mémoire d'instruction. La mémoire de données sera une RAM qui permettra de sauvegarder des variables. Ensuite, la mémoire d'instruction contiendra votre programme, c'est-à-dire la suite d'instruction à réaliser. L'élément fondamental pour l'interfaçage avec ses mémoires et l'unité d'adressage, que l'on détaillera plus tard dans le document.

Question : Comment communiquent les mémoires données et instructions avec le processeur ? Quel est le

format d'instruction utilisé ? Quels sont les bus mis en jeu ?

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 de suivre les étapes détaillées ci-dessous pour être efficace.

a. Détails des séances

L'objectif de la séance de TP 2 est :

- de développer l'ALU (normalement déjà fait lors du TP 1 !),
- de concevoir le cœur du composant "Banc de registres" (au besoin à terminer chez soi)
- de développer le registre d'instruction sur 12 bits
- développer le cœur 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 cœur du processeur.
- développer les unités de décodage du contrôleur,
- réaliser l'incrémentation du compteur ordinal (PC),
- instancier le contrôleur dans le cœur du processeur et 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]

• Travail à faire

La première chose à faire est de prendre une feuille de papier et de dessiner les connexions entre les différents éléments du processeur. Vous veillerez à bien identifier le rôle des différents éléments, préciser les bus de communication, à ajouter les mémoires et les bus de communications avec le processeur. Vous pouvez repartir du schéma global (mais volontairement incomplet) en figure 1.

Une fois terminé et validé par le professeur encadrant, vous commencerez la conception du processeur par l'ALU.

1.1 ALU

Normalement, vous avez déjà réalisé et validé le cœur de l'ALU durant la séance précédente. Il ne vous reste plus qu'à concevoir l'interface. On supposera pour l'instant qu'elle ne peut faire que 8 opérations différentes. A savoir, cette UAL dispose de 3 signaux de commande pour choisir l'opération effectuée sur

[1] Un jeu de données

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. 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.

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. Il faut donc anticiper si vous souhaitez faire des modifications (ajout d'opérations, etc).

1.2 Banc de registres

1.2.1 Registre général

Définition d'un modèle vide de registre 4 bits, dont l'interface est la suivante :

- 1 sortie sur 4 bits, notée X_i,
- 1 sortie sur 4 bits, notée Y_i,
- 1 entrée sur 4 bits, notée R_i,
- 1 signal de contrôle de lecture sur 1 bit, noté selX_i,
- 1 signal de contrôle de lecture sur 1 bit, noté 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.

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 conflits d'écriture, chaque source est reliée 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.

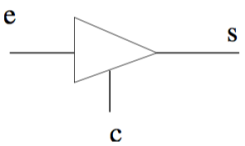

Figure 4 : Porte 3 états

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 problèmes 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.

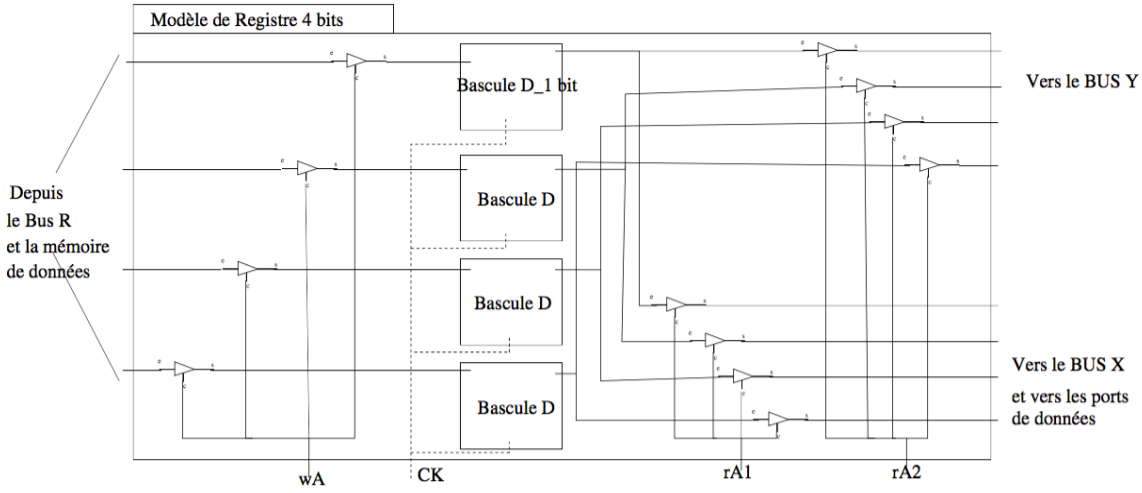

Figure 5 : Implantation d'un registre 4 bits

1.2.2 Banc de registres

Une fois le registre conçu, il vous reste à créer 4 registres 4 bits et de les relier dans un nouveau module que vous appellerez « Banc_De_Registres ». Vous veillerez à avoir une disposition claire sous Logisim des éléments dans ce module (registres, signaux de contrôle, etc.).

1.2.3 Registre d'instruction (RI)

L'unité de contrôle aura besoin du registre d'instruction pour stocker l'instructions à traiter, provenant de la mémoire d'instruction. Vous réaliserez donc ce registre sur le même modèle que précédemment en faisant attention à sa taille, ses signaux de contrôle.

Question : Quelle est la taille en bits de ce registre ?

1.3 liaison entre l'ALU et le banc de registre

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 cœur de chacun sera terminée. Normalement, à ce stade vous avez compris comment sont reliés les différents éléments à partir du schéma réalisé sur papier. Finalement, notre chemin de données dispose de 4 registres de 4 bits.

1.4. 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.

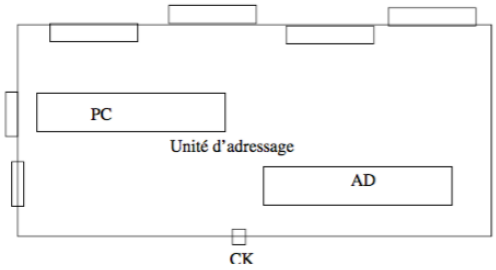

Figure 3 : Interface de l'unité d'adressage

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. 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 figure du registre 4 bits).

Question : A quoi sert le registre AD ? Que contient-il ? Même question pour le PC.

1.5 Unité de contrôle

1.5.1 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). On retrouve les trois grandes étapes du cycle d'exécution Fetch-I, Decode et Execute.

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'exécution 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.

1.5.2 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.

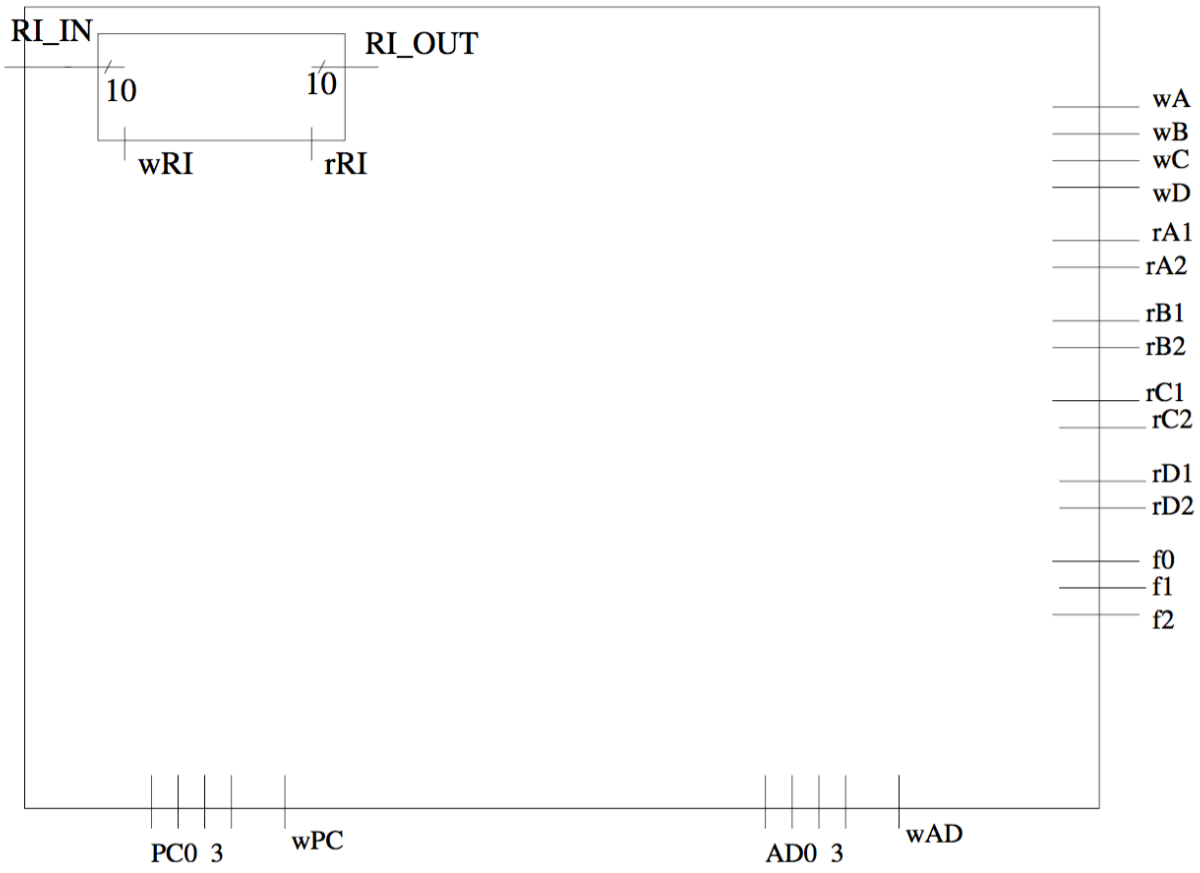

Figure 7 : Unité de contrôle du processeur

Le jeu d'instruction de ce 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 façon 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.

1.6 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 Harvard. C'est

donc dans l'unité de contrôle que l'on trouvera le registre instruction. Vous devez donc disposer 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 blocs du processeur comptera dans la notation puisque les différents schémas des composants réalisés devront apparaître dans les planches de la soutenance (capture d'écran).

1.7 Communication avec la mémoire

Une fois tous les développements réalisés, il reste à tester votre processeur. Pour cela, vous devez 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 hexa 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 discuterez 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 résultat des opérations de l'ALU est nul ou non.

3.7.2 Instruction de branchements

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 registre R est nul. L'ajout de cette instruction impose que vous ayez ajouté la modification indiquée en 3.7.1.

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 applicable 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.