Université de Bordeaux
Licence STS

ARCHITECTURE DES ORDINATEURS

TP : **08**

Construction du chemin de données d'un processeur (partie 1)

L'outil que vous allez utiliser est le simulateur de circuits SimcirJS, dont une copie adaptée au cours est installée sur :

`http://dept-info.labri.fr/ENSEIGNEMENT/archi/circuits/blank.html`

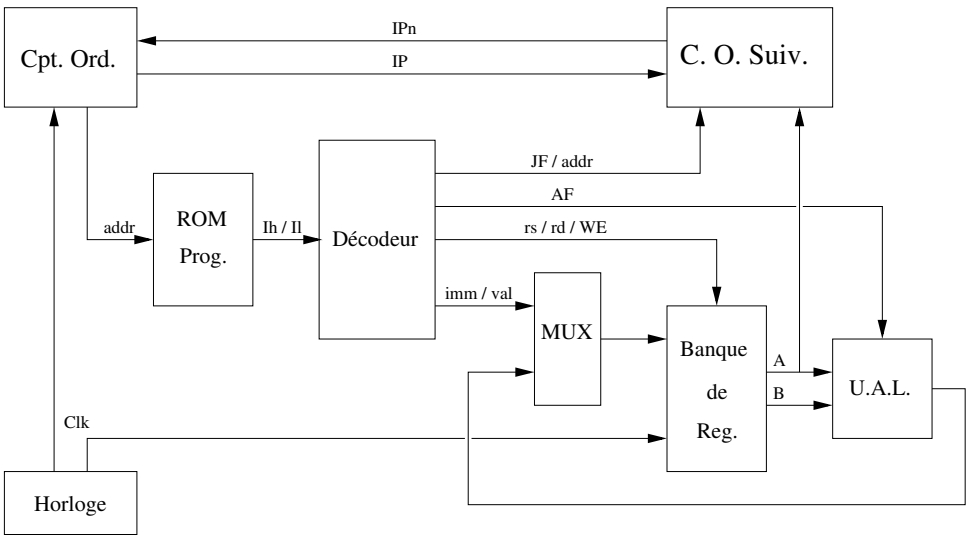
(ce lien est disponible depuis le site du cours).

On veut construire un processeur sur 4 bits, à l'image du processeur historique « 4004 » d'Intel. Le processeur à construire possède les caractéristiques suivantes :

- Il dispose de 4 registres, numérotés de 0 à 3, c'est-à-dire adressés sur 2 bits ;
- Le compteur ordinal est sur 3 bits, permettant l'existence de $2^3 = 8$ instructions dans la mémoire de programme (oui, c'est peu !) ;
- Chaque instruction est formatée sur un octet, à savoir 2 mots de 4 bits. Le tableau suivant recense les instructions constituant ce langage machine, ainsi que leur codage :

Instruction	7	6	5	4	3	2	1	0
`halt`	0	0	—					
`li` $\texttt{val}_{(4)}$`,`$\texttt{rd}_{(2)}$	0	1	$\texttt{rd}_{(2)}$		$\texttt{val}_{(4)}$			
`add` $\texttt{rs}_{(2)}$`,`$\texttt{rd}_{(2)}$	1	0	0	0	$\texttt{rs}_{(2)}$		$\texttt{rd}_{(2)}$	
`dec` $\texttt{rd}_{(2)}$	1	0	1	1	—		$\texttt{rd}_{(2)}$	
`jmp` $\texttt{addr}_{(3)}$	1	1	—		0	$\texttt{addr}_{(3)}$		
`jz` $\texttt{rs}_{(2)}$`,`$\texttt{addr}_{(3)}$	1	1	$\texttt{rs}_{(2)}$		1	$\texttt{addr}_{(3)}$		

L'architecture du processeur susceptible d'exécuter les instructions de ce langage est la suivante :



Exercice 1 : Codage d'un programme

Quelle est la fonction du programme codé ci-dessous ?

Adresse	Code	Instruction
0	6 x	`li` $x_{(4)}$`,r2`
1	5 y	`li` $y_{(4)}$`,r1`
2	4 0	`li 0,r0`
3	E F	`jz r2,7`
4	8 4	`add r1,r0`
5	B 2	`dec r2`
6	C 3	`jmp 3`
7	0 0	`halt`

Exercice 2 : Décodage des instructions

Pour alimenter les différentes unités fonctionnelles du processeur, il est nécessaire de décoder les instructions afin d'en extraire les données pertinentes et de les fournir aux autres unités fonctionnelles selon un format uniforme.

— Considérez le circuit suivant :
`http://dept-info.labri.fr/ENSEIGNEMENT/archi/circuits/blank.html#wpVWw5tuw5pAEH3Cj8KUf0DDuwzCk`
`Cg8OBG1LDh2fCgSZodsKLw57ChxrCtngEw4Bvw7PDmTnDuDO7w6wXw5vCrcOBwpDCnMOjS8K5w5vDhcOiHA8Ww4tzcBoAI`
`w49eQcKQwoDCqEkiw4cBwofCmXjCgsOUSxhUw7rCmsOFw7EOBMOrCMKkAl3Cpklow4ExwrBMcB/Dpn/Cs1UMK8KFcMKEwrt`
`CoTfDsiNKVcOew7cRw6RwwrLClcKOwoXDrXd1w61Cw7Evw5rCiAN4w63ChigKw7NPOcOFw4k/w6cXw5EbwqIjwpoQw47Chi`
`DlcKawo/Dr8KcRcOXw63DqjrDniXCunDDuV5dw77CuMOSw4TClMO3LMK/fcKOGznCv8KSKOfCmcOLwrUrwrfDnsOvw5PCr1`
`J3rDq1BvZMKSwojCjcKSacO7Vm9Tw7lUZ3ETJcKFwpbCksO1G8K+wpHCmXJaw6EbKsOpT8KlGsKVwpvCqcKsP3XDnMKfw4c`
`w5vCksKLYcKuPlxLaVjCn8KuwqU2wqzCj8KXWsOyw5XDp0stw7nDqgPCpsKWfMO1CcObw7rCq8KPwpjCmmsjw7osWMKtWz7`
À la gauche de votre plan de travail se trouvent deux `RotaryEncoderBus`, qui vont vous permettre de représenter tous les codes d'instructions possibles. Pour accéder individuellement aux bits de chacun des mots de 4 bits, vous disposez de deux `BusIn`. Le `2to4decoder` permet quant à lui d'extraire le type général de l'instruction à partir des deux bits de poids le plus fort.
Étudiez bien la structure de ce circuit, et à quels bits ou ensembles de bits correspond chaque « plot » de droite.

— Les instructions `add` et `jz` possèdent un champ de registre source `rs`, mais celui-ci n'est pas placé au même endroit dans chacune de ces instructions. Câblez la fonction `rs` (pour « *register number (source)* »), renvoyant la valeur `rs` contenue dans chacune des instructions qui la possèdent.
Pour des raisons de commodité, cette sortie sera sur quatre bits, afin de pouvoir utiliser les connexions de bus. Les deux valeurs binaires seront donc agrégées en sortie au moyen d'un `BusOut`.
Pour les instructions qui n'ont pas de champ `rs`, la valeur à renvoyer n'est pas pertinente. Il n'est donc pas pertinent de complexifier la fonction pour gérer une valeur par défaut.
Testez votre fonction avec chacune des instructions en question. Pour faciliter vos tests, vous pourrez afficher, au moyen d'un `4bit7segBus`, la valeur calculée.

— Même question pour le champ `rd`.

— Câblez la fonction `WE` (« *write enable* »), qui renvoie 1 lorsque l'instruction doit effectivement modifier la valeur du registre de numéro `rd` dans la banque de registres. Cette fonction doit donc identifier les instructions qui possèdent un champ `rd`.

— Câblez la fonction `imm`, qui renvoie 1 lorsqu'une valeur immédiate doit être prise en compte dans l'instruction.

— Câblez la fonction `val`, qui renvoie sur 4 bits la valeur immédiate présente dans l'instruction.

— Câblez la fonction `AF`, qui renvoie sur 4 bits le code de fonction arithmétique et logique présent dans l'instruction.

— Câblez la fonction `JF` (« *jump function* »), qui indique les modalités de passage à l'instruction suivante :

Valeur	Signification
0	Même instruction (cas de `halt`)
1	Passage à l'instruction suivante
2	Branchement inconditionnel
3	Branchement conditionnel

Pour coder cette fonction, considérez chacun des deux bits de la valeur à retourner comme une fonction indépendante, calculée à partir des bits 7, 6 et 3 de l'instruction. Vous agrègerez ensuite ces deux bits au moyen d'un `BusOut`.

— Câblez la fonction `addr`, qui renvoie, étendue sur 4 bits, la valeur de l'adresse de destination contenue dans l'instruction.

Toutes les fonctionnalités décrites ci-dessus sont encapsulées au sein du composant `CpuDecoder`, que vous pourrez utiliser par la suite.

Exercice 3 : Horloges

Question 1

Câblez un circuit d'horloge, alimenté par une entrée d'horloge `Clk`, qui fournit deux signaux d'horloge de sortie, `Clk1` et `Clk2`, tels que, sur quatre périodes de temps consécutives, on ait `Clk1` et `Clk2` alternativement à `1` une fois sur deux, lorsque l'entrée d'horloge `Clk` est à `1`.
N.B. : On ne veut pas de dérive de l'horloge.

Question 2

Ajoutez à votre circuit un bouton d'activation (`Toggle`), tel que le circuit d'horloge ne fonctionnera que lorsque ce bouton sera activé.

Question 3

Ajoutez à votre circuit un bouton de réinitialisation (`PushOn`), qui fait que, lorsque votre circuit sera désactivé, réinitialisé, puis réactivé, c'est toujours `Clk1` qui s'allumera en premier, avant `Clk2`.
N.B. : On acceptera une dérive de l'horloge.