WDR PC-8

Projet 2023

UNIVERSITÉ DE NANTES

Noms:	Rideau Germain, Quemeneur Armel
Groupe:	584
Module:	Architecture des Ordinateurs

1 Instructions Machine

Instruction	Opcode	Commentaire	Format
inc i	000	Incrémente ri	R
dec i	001	Décrémente ri	R
jmp adr	010	Saute à l'addresse adr	I
isz i	011	Saute l'instruction suivante si ri == 0	R
stp	100	Arrête le programme	X

les instructions machines sont représentées sous forme R ou I selon si elles prennent en paramètre une addresse dans la RAM ou bien vont faire effet sur un registre.

- R : la valeur i donnée est le code d'un des 8 registre sur lequel cette opération va s'effectuer
- I : la valeur adr donnée est une addresse dans la RAM

1.1 Somme

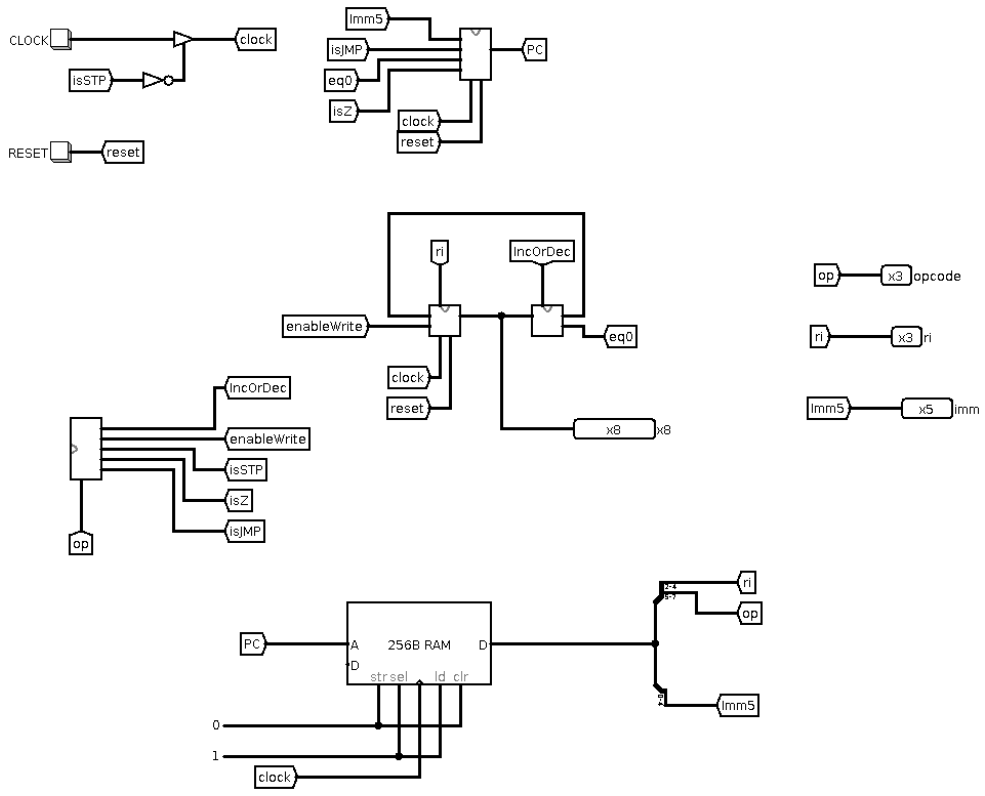
ligne	instruction	binaire	hexadécimal
0	jmp 3	010 00011	0x43
1	inc 0	000 00000	0x00
2	dec 1	001 00100	0x24
3	isz 1	011 00100	0x64
4	jmp 1	010 00001	0x41
5	stp	100 00000	0x80

Table 1: somme(r0, r1)

2 Fonctionnement du CPU

le cpu communique avec une RAM qui contient les instructions du code en hexadécimal, de manière contigue en commençant à l'adresse 0x00. A chaque tic d'horloge, l'opération pointée par le registre de l'unité nextAddLogic est effectuée. Puis le registre est actualisé et pointe vers l'adresse de la prochaine instruction. le système possède aussi un banc de 8 registres dans lesquels sont inscrits les valeurs obtenues par les instructions d'incrémentation ou de décrémentation. Pour connaître l'opération à effectuer en fonction de l'opcode donné par l'instruction courante, on a une unité Controller qui gère des bits ou flags qui sont levés si le code correspond à cette opération.

2.1 circuit principal



3 ALU

Le circuit d'unité arithmétique logique contient 2 types d'instruction, l'incrémentation et la décrémentation d'un registre. On remarque que ces instructions sont codées de la même manière : 00X avec X à 0 pour l'incrémentation et 1 pour la décrémentation. On a donc besoin que d'un bit de sélection entre les 2 opérations, ce bit ou flag est généré par l'unité controller. La valeur en sortie est gérée par l'unité regFile qui peut inscrire la valeur dans un registre

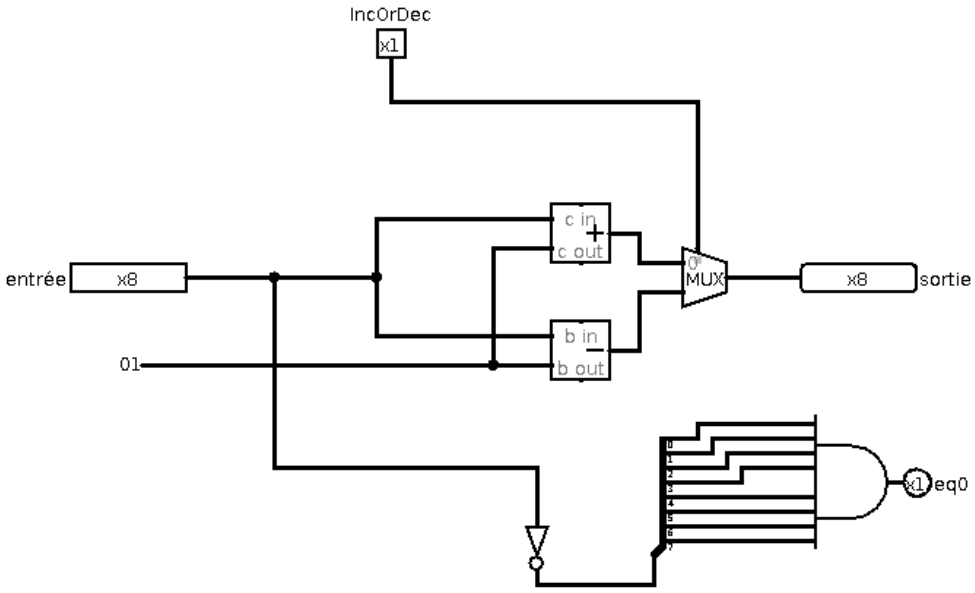
Entrées de L'ALU

1. Valeur du registre courant
2. flag IncOrDec venant du controlleur, 0 pour Incrémentation, 1 pour Décrémentation

Sorties de L'ALU

1. Valeur après opération
2. flag eq0, si la valeur en entrée est 0

Valeur de sortie : cette valeur est choisie par un multiplexeur dont le bit de sélection est le flag IncOrDec.
eq0 : calculé en faisant un ET logique sur tous les bits de la valeur en entrée.



Circuit ALU

4 RegFile

L'unité RegFile gère les registres, dans quel registre écrire, quel registre doit avoir sa valeur exposée en sortie. L'unité utilise la valeur i donnée par l'unité nextAddrLogic

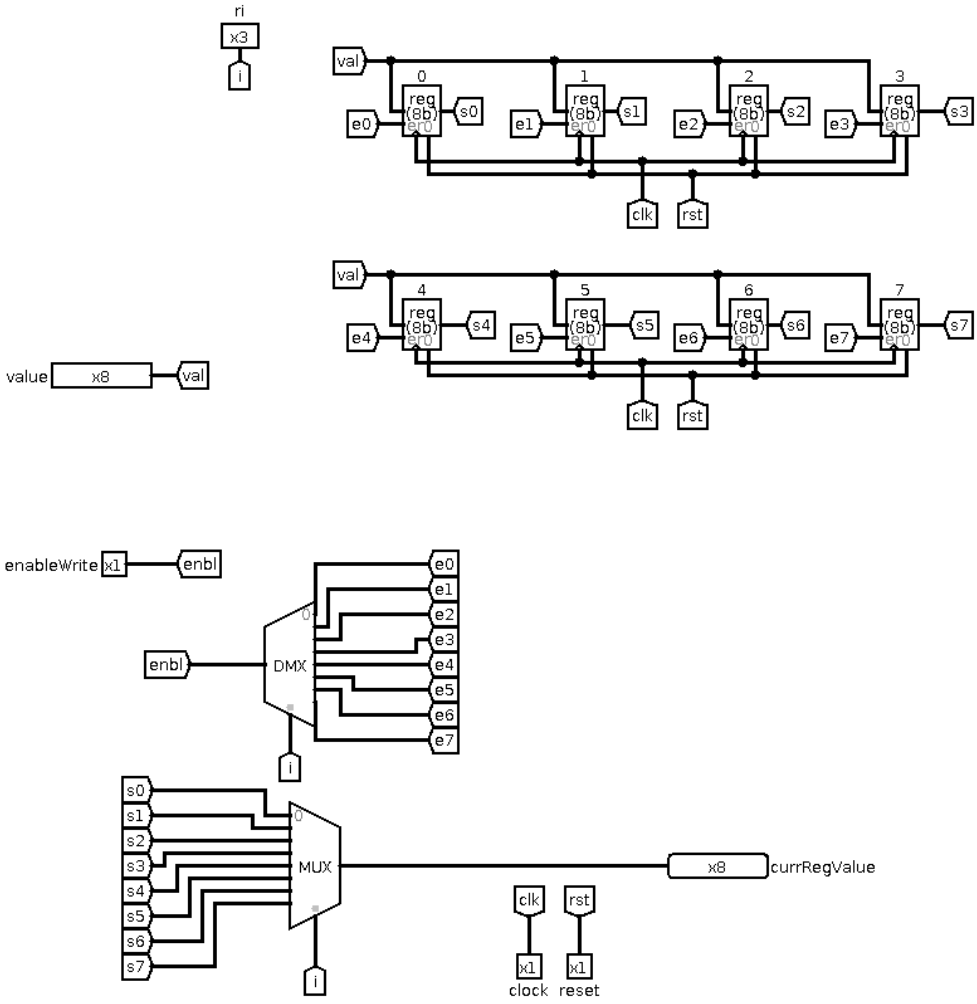
Entrées :

- **valeur** donnée par l'ALU sur 8bits
- **enableWrite** flag donné par le controller
- **ri** code sur 3 bits du registre courant
- **clock**
- **reset**

Sorties :

- valeur du registre courant

la valeur donnée par l'ALU est donnée en entrée pour chaque registre, c'est le flag enableWrite qui permet l'écriture dans le registre selectionné par ri dans un démultiplexeur. De la même manière, la valeur donnée en sortie est la valeur du registre sélectionné par ri dans un multiplexeur. L'horloge permet l'actualisation des registres et de la sortie en parallèle. le reset remet à 0 tous les registres.



5 Controller

L'unité Controller est l'unité qui gère les flags selon l'Opcode qu'elle reçoit en entrée.

Flags:

- IncOrDec : *sélectionne l'opération à effectuer*
- enableWrite : *autorise l'écriture dans les registres*
- isSTP : *arrêt du programme*
- isZ : *s'utilise avec le flag eq0*
- isJMP : *saute à l'addresse en mémoire*

q0	q1	q2	Opération
0	0	0	inc
0	0	1	dec
0	1	0	jmp
0	1	1	isZ
1	0	0	stp
1	0	1	x
1	1	0	x
1	1	1	x

Table 2: codage opcode

5.1 IncOrDec

q0\q1q2	00	01	11	10
0	0	1	0	0
1	0	0	0	0

Table 3: Karnaugh IncOrDec

On choisit arbitrairement 0 pour incrémentation et 1 pour décrémentation ainsi que l'état 0 par défaut.

Formule logique : -q0-q1q2

5.2 Jump

Formule logique : -q0q1-q2

5.3 isZ

Formule logique : -q0q1q2

5.4 stp

Formule logique : q0-q1-q2

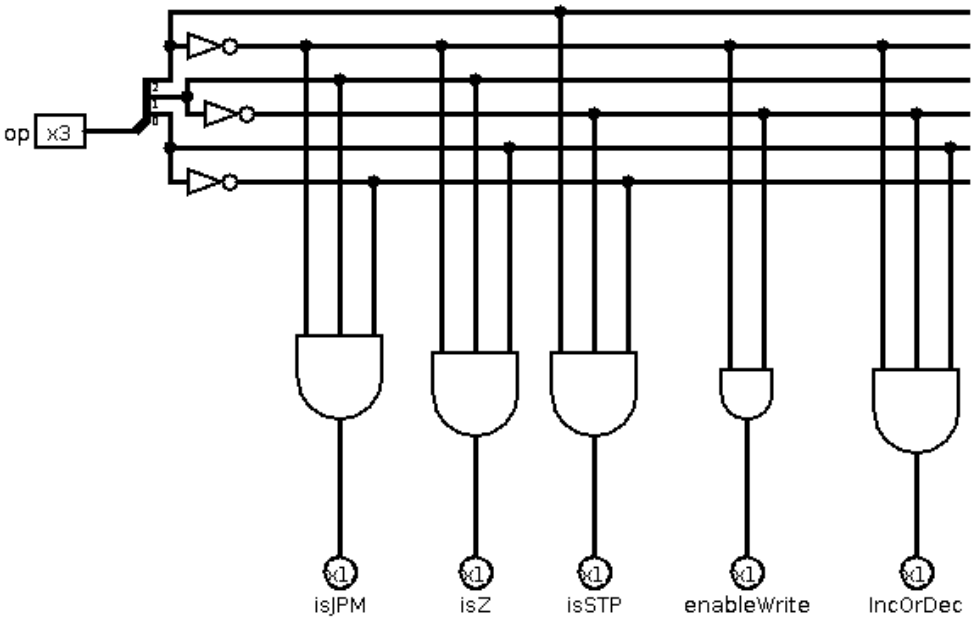
5.5 enableWrite

enableWrite doit être vrai lorsque l'opération courante doit écrire dans un registre, c'est à dire incrémentation ou décrémentation.

q0\q1q2	00	01	11	10
0	1	1	0	0
1	0	0	0	0

Table 4: Karnaugh enableWrite

Formule logique : -q0-q1



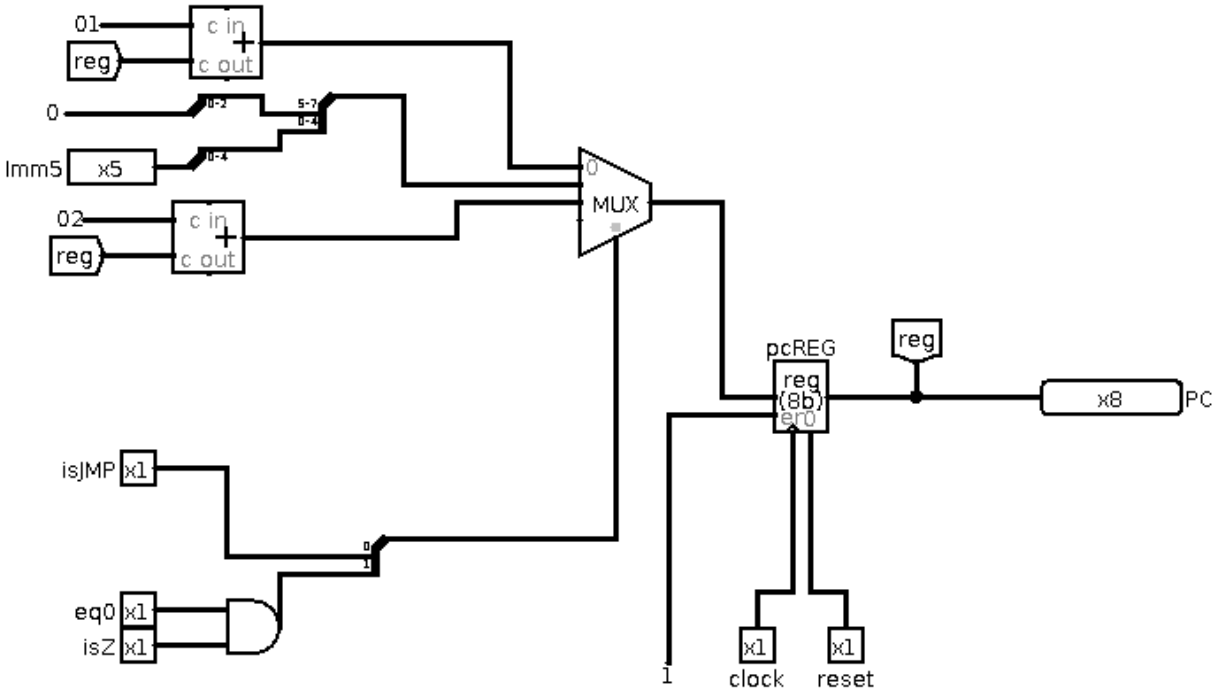
6 Next Address Logic

Cette unité contient un registre 8bits qui donne l'adresse mémoire de la prochaine instruction à exécuter. le comportement par défaut est d'incrémenter de 1 l'adresse à chaque tic d'horloge, cependant si il y a un jump ou un isZ et le flag eq0 levé, ce comportement doit changer.

- **Défaut** : regValue += 1.
- **isZ et eq0** : regValue += 2 (on saute l'instruction suivante)
- **jmp** : regValue = Imm5 donnée par l'instruction jmp.

Pour les 2 premiers, assez simple : on utilise un additionneur avec une constante.
Pour Imm5, on doit passer d'une valeur codée sur 5 bits à une valeur codée sur 8bits(le registre). On ajoute donc des 0 sur les 3 bits de poids fort.
La sélection de l'opération à effectuer se fait par un code sur 2 bits donné par les flags jmp en q0 et isZeq0 en q1.

q0	q1	instruction
0	0	add1
0	1	add2
1	0	jmp imm5
1	1	**erreur**



7 Algorithmes

ligne	instruction	code binaire	code hexa
0	inc 1	000 00100	04
1	inc 1	000 00100	04
2	inc 1	000 00100	04
3	inc 1	000 00100	04
4	isZ 1	011 00100	64
5	jmp 7	010 00111	47
6	stp	100 00000	80
7	dec 1	001 00100	24
8	inc 2	000 01000	08
9	inc 2	000 01000	08
10	jmp 4	010 00100	44

Table 5: TwoTimes(4)

ligne	instruction	code binaire	code hexa
0	isZ 0	011 00000	60
1	jmp 3	010 00011	43
2	stp	100 00000	80
3	dec 0	001 00000	20
4	inc 1	000 00100	04
5	jmp 0	010 00000	40

Table 6: move(r0,r1)

On remarque que pour ces 2 algorithmes, on doit programmer les paramètres de la ”fonction” dans le code. Une idée d'amélioration est de pouvoir charger une valeur en mémoire dans un registre afin de pouvoir utiliser ces algorithmes de manière polyvalente sans avoir à coder en dur la valeur des paramètres. Cela implique de modifier le circuit de gestion des registres et de devoir choisir entre la valeur du registre courant ou bien la valeur donnée par la RAM. Il faut donc avoir un nouveau multiplexeur qui prends en entrée la valeur donnée par l'ALU ainsi que la valeur chargée dans la RAM, un bit de sélection devra choisir quelle valeur inscrire dans le registre ri.

8 BONUS

Création d'une nouvelle instruction pour le processeur WDR PC-8. Plusieurs idées sont possibles et faisables en parallèle, mais on a une limitation à 8 instructions machines car l'opcode est codé sur 3 bits

- **jmpZ adr** : jump si eq0 est levé (si le registre courant est à 0). type I
- **load ri** : charge dans un registre du banc la valeur du registre a0.
- **set imm5** : met la valeur de a0 à imm5.

8.1 jmpZ

implémentation :

1. créer un nouveau flag (isJMPZ) dans le controller avec l'opcode choisi : *101*
2. modifier l'unité nextAddrLogic pour avoir un saut à Imm5 si eq0 ET isJMPZ sont levés.

8.2 load ri et set imm5

implémentation :

1. **set :** charger dans a0 la valeur imm5 lorsque le flag isSet est levé.
2. **load :** charger dans ri la valeur du registre a0 lorque le flag isLA est levé.
3. multiplexer dans regFile entre la valeur de l'ALU et de a0 pour load avec le flag isLA comme sélecteur (ALU par défaut)

résumé : utiliser un registre a0 temporaire avec 2 instructions, set Imm5 et load ri, un registre dont la valeur peut changer avec un imm5 et dont on peut charger la valeur dans un des registres du banc. Load doit aussi activer enableWrite. on change donc d'équation pour enableWrite dans le controller : on ajoute un ou logique avec l'instruction load.

8.3 Exemple

Instruction	Opcode	Commentaire	Format
inc i	000	Incrémente ri	R
dec i	001	Décrémente ri	R
jmp adr	010	Saute à l'addresse adr	I
isz i	011	Saute l'instruction suivante si ri == 0	R
stp	100	Arrête le programme	X
la ri	111	charge la valeur de A0 dans ri	R
set imm5	110	met la valeur de A0 à imm5	I
jmpz adr	101	jump à adr si registre courant à 0	I

Table 7: nouveau jeu d'instructions

ligne	instruction	code binaire	code hexa
0	set 4	110 00100	C4
1	la 1	111 00100	E4
2	jmpz 7	101 00111	A7
3	dec 1	001 00100	24
4	inc 2	000 01000	08
5	inc 2	000 01000	08
6	jmp 2	010 00010	42
7	stp	100 00000	80

Table 8: TwoTimes(4) avec nouvelles instructions