MICROARCHITECTURE NON PIPELINEE

1.1 Introduction

Dans ce chapitre, nous présentons les chemins de données et la partie contrôle de l'implémentation non pipelinée du processeur PCLM-2012. Le jeu d'instructions de ce processeur est présenté en annexe.

1.2 Les transferts élémentaires et commandes.

Le transfert élémentaire le plus simple recopie le contenu d'un registre dans un autre registre à travers un bus. Pour la simplicité de l'exposé, nous supposerons que les registres sont opaques : l'opacité peut être réalisée avec des bascules à commande par flanc d'horloge (*edge triggered flip flop*), ou des bascules maître-esclave. L'opacité des bascules permet que la sortie d'un registre soit connectée sur l'entrée du même registre à travers un organe combinatoire. C'est ce qui est nécessaire, par exemple, pour réaliser l'incrémentation du contenu d'un registre.

La Figure 1 illustre les éléments d'un transfert élémentaire. Elle montre trois registres R_i, R_j et R_k. Chaque registre de n bits reçoit en entrée les n bits d'un bus, et est connecté en sortie sur le même bus à travers une barrière de bus. Une barrière de bus établit la connexion entre la sortie du registre R_i et le bus si la commande de barrière associée T_i est active. A chaque registre est associée une commande d'écriture W qui écrit les n bits du bus dans le registre lorsque la commande d'écriture est active.

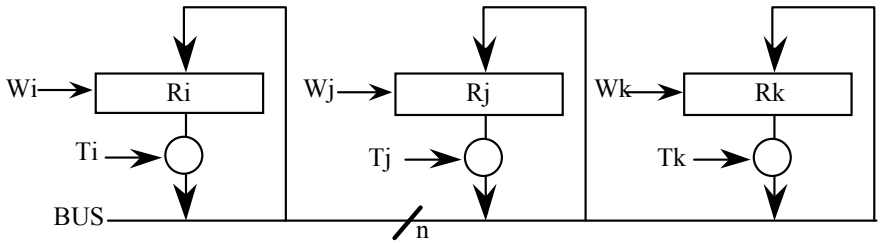

Figure 1 : Transfert de registre à registre.

Le transfert d'un registre R_i vers un registre R_k à travers un bus met donc en jeu deux commandes : T_i (du registre vers le bus) et W_k (du bus vers le registre).

1.3 Ressources et horloges

Les ressources matérielles du processeur sont

- l'unité arithmétique et logique (UAL). Elle permet, sous contrôle de bits de l'instruction, d'effectuer l'ensemble des opérations arithmétiques et logiques du jeu d'instructions, plus les décalages. Elle effectue les opérations suivantes : addition sur ses deux entrées e1 et e2, addition sur ses entrées e1, e2 et la retenue d'entrée, incrémentation de l'entrée e1, fonctions logiques et, ou et ou exclusif (xor), concaténation de la partie haute de e1 et de la partie basse de e2 (habb), et l'ensemble des décalages suivants : décalage logique droite et gauche, décalage arithmétique droite, et rotation à droite à travers la retenue, de 0 à 15 positions.
- le banc de registres. On peut, dans un cycle donné, lire simultanément deux registres. Les numéros des registres lus sont fournis par des bits de l'instruction contenue dans RI (cf. format des instructions en annexe) Cette caractéristique permet d'envoyer simultanément deux opérandes à l'entrée de l'UAL. On peut, dans un autre cycle, écrire dans un registre. Le numéro de registre écrit est fourni également par des bits de l'instruction.
- les registres spécifiques du processeur : CP, RI, RAdM.

La mémoire, qui contient les instructions et les données, est connectée aux bus internes de la machine. Dans ce chapitre, on ne considère pas de hiérarchie mémoire. Le registre RAdM contient l'adresse mémoire d'une instruction Load et Store. Il n'y a pas de registre RDM, car la donnée en écriture vient directement d'un registre, et la donnée en lecture est directement rangée dans un registre.

Il est important de spécifier le mode de fonctionnement des registres et les signaux de contrôle (horloge) associés. Tous les registres utilisés sont des registres non transparents utilisant des bascules D à commande par transition d'horloge. Pour des raisons de simplicité, on suppose une horloge symétrique (50% à l'état haut et 50% à l'état bas). La période de l'horloge est Tc : on l'appellera temps de cycle de la machine (figure 1). Tous les registres sont écrits en fin de cycle sur la transition montante du signal d'horloge.

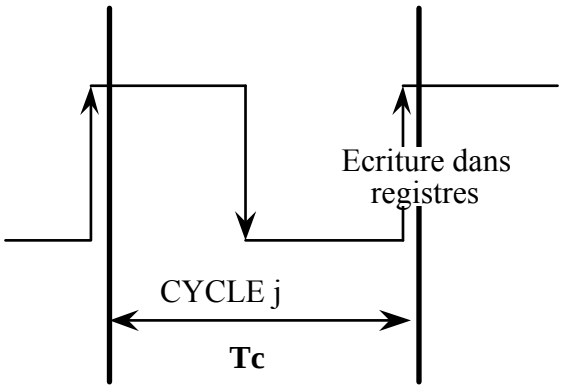

Figure 2 : Cycle d'horloge

On suppose que les caractéristiques temporelles des différentes ressources sont les suivantes :
Le temps nécessaire pour l'opération la plus longue de l'UAL est T_{UAL}. C'est généralement le temps nécessaire pour l'opération arithmétique la plus longue, à cause des temps de propagation de la retenue.
Le temps de cycle mémoire est T_{CM}. C'est le temps minimum entre deux opérations mémoire successives (lecture ou écriture). Le temps d'accès mémoire est T_{AM}. C'est le temps minimum entre l'envoi d'une adresse et la réception de la donnée en lecture.
Tout transfert de registre à registre introduit un temps T_{reg}, qui est la somme du temps de propagation du registre de sortie (retard horloge-sortie du registre) et du temps d'établissement du registre d'entrée (établissement du signal avant l'horloge du cycle suivant). Un transfert registre à registre à travers un bus introduit un temps T_{REG} qui est la somme de T_{reg} et du temps de traversée de la barrière de bus.
Les étapes de chaque instruction en fonction de chaque type d'instructions sont données par la table 1 :

PHASE	UAL	Sauts et Branchements	Mémoire
LECTURE INSTRUCTION	RI ←M(CP)		
DÉCODAGE	Décodage et CP := CP + 1		
EXÉCUTION	Rd := Ra op Rb ou Rb := Ra op IMM	R7 := CP (Call) CP := Ra (JMP et Call) Si condition CP := CP + dépl. (branch)	RAdM := Ra + IMM. RDM := Rb
MÉMOIRE			Rb := M(RAdM) (Lec) M(RAdM) := Rb (Ecr)

Table 1 : Phases d'exécution d'une instruction

1.4 Actions élémentaires à chaque étape.

Le format des instructions UAL étant un format à trois opérandes (deux sources et une destination), contenu dans des registres, le chemin de données est organisé autour de trois bus (deux sources et une destination) autour de l'UAL. Nous examinons les actions élémentaires à chaque étape d'exécution de l'instruction. Le chemin de données correspondant est présenté dans la Figure 3.
On utilise les abréviations suivantes : $RI_{11-9} \equiv a$, $RI_{8-6} \equiv b$, $RI_{5-3} \equiv d$. a, b et d sont des numéros de registres spécifiés dans l'instruction, en fonction des formats d'instructions (cf. annexe). RI_{5-0} + imm ou dep selon le format. On appelle IMM ou DEP les versions étendues sur 16 bits de imm et dep..

1.4.1 Lecture de l'instruction (LI)

RI := M(CP)

Cette phase correspond à la lecture de l'instruction. Elle est réalisable en un cycle d'horloge si la condition suivante est réalisée :
$Tc > T_{AM} + T_{REG}$

1.4.2 Décodage et incrémentation de CP

CP := CP + 1

Dans cette phase, on incrémente le compteur de programme pour préparer l'instruction suivante et on décode l'instruction pour déterminer les actions suivantes. L'incrémentation de CP utilise l'UAL.

La contrainte temporelle pour une réalisation en un cycle est la suivante :
$Tc > T_{UAL} + T_{REG}$

1.4.3 Exécution

La phase exécution correspond à la partie opération de toutes les instructions. Compte tenu de la structure générale des instructions (3 opérandes), elle consiste à déterminer les sources des entrées de l'UAL via les bus A et B, l'opération à effectuer, et la destination de la sortie de l'UAL via le bus R.
Le bus A est alimenté par Ra pour les instructions UAL et sauts et par CP pour les instructions branchements.
Le bus B est alimenté par Rb pour les instructions UAL de format RR, par IMM pour les instructions UAL de format RI, les instructions de branchement et les instructions mémoire.
L'UAL effectue une des opérations suivantes : +, OU, XOR, sur Bus A et Bus B, , transfert du bus A, +1 sur Bus A, habb (transfert de la partie haute du bus A et de la partie basse du bus B).
Le bus R alimente un des registres généraux, spécifié sous forme Rd ou Rb pour les instructions UAL, le registre RAdM seul pour les instructions mémoire, et le registre CP dans le cas des instructions de saut et branchement qui modifient CP. Dans le cas de branchement conditionnel, il y a écriture de l'adresse de branchement dans CP.
Le registre RAdM est chargé à partir du bus R dans le cas d'une instruction mémoire.
L'instruction Call nécessite un traitement particulier. Il faut effectuer la sauvegarde de l'adresse de retour, que contient CP après la phase Décodage, dans le registre R7. Il faut donc effectuer le transfert Rb := CP en parallèle avec le transfert CP := Ra pour sauvegarder l'adresse de retour dans Rb. L'utilisation de registres non transparents autorise ces transferts simultanés. Il suffit d'imposer b = 111 pour que l'adresse de retour soit dans R7. Seul le registre R7 peut recevoir une autre valeur que le contenu du bus R, d'où la commande MUX R, qui indique si le banc de registres est chargé à partir du bus R ou de CP.
La table 2 rassemble les actions élémentaires nécessaires dans la phase Exécution pour chacune des instructions.

CODE OP	MNE	BUS A	BUS B	OP	MUX R7	BUS R	Test cond
0000	ADD	Ra	Rb	+	Bus R	Rd	
0000	ADC	Ra	Rb	+	Bus R	Rd	
0000	AND	Ra	Rb	et	Bus R	Rd	
0000	XOR	Ra	Rb	xor	Bus R	Rd	
0000	SUB	Ra	Rb	-	Bus R	Rd	
0000	OR	Ra	Rb	or	Bus R	Rd	
0001	ANI	Ra	IMM	et	Bus R	Rb	
0001	XOI	Ra	IMM	xor	Bus R	Rb	
1001	CALL	Ra	R0	+	CP	CP	
1001	JMP	Ra	R0	+	x	CP	
1010	LIM	Ra	RI	ha,bb	Bus R	Rb	
1100	DEC	Ra		dec	Bus R	Rb	
1011	BC	CP	IMM	+	x	CP	X
1110	LD	Ra	IMM	+	x	RAdM	
1111	ST	Ra	IMM	+	x	RAdM	

Table 2 : Actions élémentaires de la phase Exécution.

La contrainte temporelle sur la phase Exécution est
$Tc > T_{UAL} + T_{REG} + 2\ T_{MUX}$

Contrôle de l'UAL
Dans la phase LI, l'UAL n'intervient pas. Dans la phase DI, elle fait +1 sur l'entrée e1.
Les commandes de l'UAL dans la phase EX découlent des codes opérations et de leur extension.
La table 3, déduite de la table 2, donne les différentes opérations de l'UAL en fonction des codes opérations

+	ADD, ADC, JMP, CALL, tous les branchements conditionnels, LD et ST
-	SUB
And	AND, ANI
Or	OR
Xor	XOR
dec	Toutes les instructions de décalage

ha,bb	LIM

Table 3 : Instructions et commandes de l'UAL

Contrôle des branchements conditionnels
Il faut calculer la condition, qui est un booléen vrai ou faux.
Le booléen s'obtient selon l'expression logique suivante :
COND = (Instr BEQ et Ra=0) ou (Inst BNE et Ra≠0) ou (Inst BGT et Ra>0) ou (Inst BGE et Ra ≥0) ou (Inst BLT et Ra < 0) ou (Inst BLE et Ra ≤ 0)
Les six instructions correspondent à un code op commun 1011 avec six extensions différentes du code op.
Les relations Ra cond 0 s'obtiennent en transférant Ra sur le bus A et en examinant les bits a15 à a0 du bus A.
La condition Ra ≠0 s'obtient à la sortie d'une porte Ou à 16 entrées (a15 à a0) et la condition Ra = 0 à la sortie d'un Nor à 16 entrées. La condition Ra <0 correspond au bit a15, et la condition Ra ≥ 0 au complément de a15, etc.
Si COND est vrai, on écrit la sortie de l'UAL dans CP, sinon on inhibe le signal d'écriture.

1.5 Chemin de données

Le chemin de données est présenté en Figure 3 ;

1.6 Examen détaillé des commandes

L'UAL et la mémoire aliment le bus D. Les connections de ces opérateurs au bus D sont les commande de barrière de bus (interrupteurs) T_{UAL} et $T_{mémoire}$. Les registres R1 à R7, RI, CP, RadM reçoivent des données du bus D. L'écriture dans ces registres se fait via les commandes d'écriture W_{R1} à W_{R7}, W_{CP}, W_{RI}, W_{RadM}
Les registres R1 à R7, CP et RAdM alimentent le bus A. Les transferts sur le bus A sont commandées par les barrières de bus T_{MUX-A}, T_{CP} et T_{RAdM}.s
Les registres R1 à R7 et RI alimentent le bus B. Les transferts sur le bus B sont commandées par les barrières de bus T_{MUX-B} et T_{RI}
La Table 4 donne les commandes pour le tronc commun des instructions. La table détaille les opérations effectuées par l'UAL et la mémoire, les registres source des bus A et B, les registres destination du bus R, ainsi que l'entrée active du multiplexeur qui est connecté à CP. Les mêmes informations sont fournies par les tables suivantes pour les instructions arithmétiques et logiques (Table 5), les instructions mémoire (Table 6), les instructions de saut et branchement (Table 7), et les instructions de décalage et rotation (Table 8).

Action	Source bus A	Source bus B	Opération UAL	Opération mémoire	Source bus R	Mux CP/Bus R	Destination Bus R
RI ← Mem(CP)	CP			Lecture	Mémoire	Bus	RI
CP ← CP + 1	CP		+1		UAL	Bus	CP

Table 4: Commandes pour le tronc commun des instructions

Mnémonique	Source bus A	Source bus B	Opération UAL	Opération← mémoire	Source bus R	Mux CP/Bus R	Destination Bus R
ADD	Ra	Rb	+		UAL	Bus	Rd
ADC	Ra	Rb	+ ret		UAL	Bus	Rd
SUB	Ra	Rb	-		UAL	Bus	Rd
OR	Ra	Rb	OR		UAL	Bus	Rd
AND	Ra	Rb	AND		UAL	Bus	Rd
XOR	Ra	Rb	XOR		UAL	Bus	Rd
ANI	Ra	IMM(RI)			UAL	Bus	Rd
XOI	Ra	IMM(RI)			UAL	Bus	Rd

Table 5 : Commandes des instructions UAL

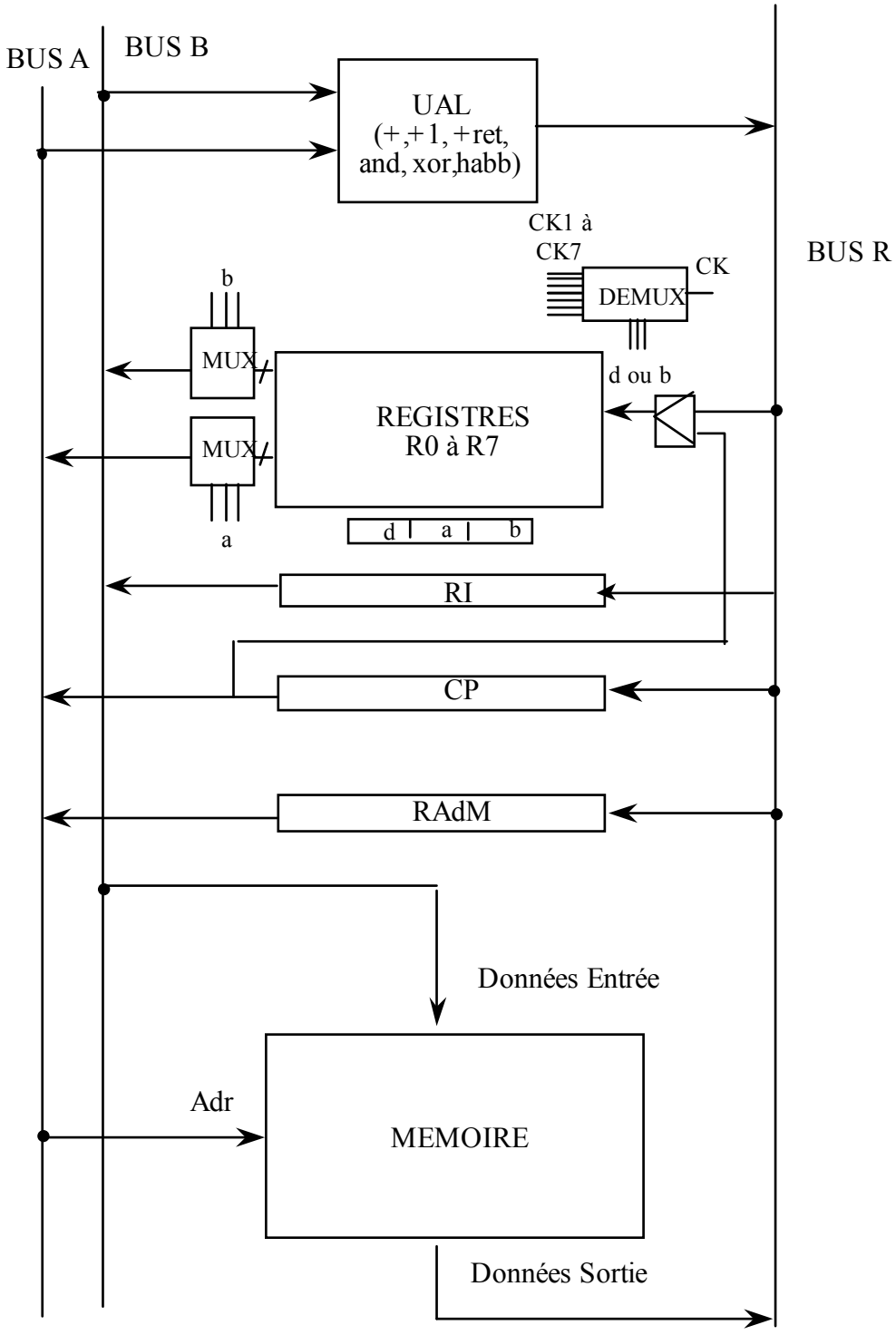

Figure 3 : chemin de données, version non pipelinée.

Mnémonique	Source bus A	Source bus B	Opération UAL	Opération mémoire	Source bus R	Mux CP/Bus R	Destination Bus R
LD	Ra	DEP(RI)	+		UAL	Bus	RAdm
	RAdM			Lecture	Mémoire	Bus	Rd
ST	Ra	DEP(RI)	+		UAL	Bus	RAdm
	RAdM	Rb		Ecriture	Mémoire	Bus	Rd

Table 6 : Commandes des instructions mémoire

Mnémo	Source bus A	Source bus B	Opération UAL	Opération mémoire	Source bus R	Mux CP/Bus R	Destination Bus R	Test condition
JMP	Ra		Transfert		UAL	Bus	CP	
CALL	Ra		Transfert		UAL	CP	CP	
BC	CP	DEP(RI)	+		UAL	Bus	CP	OUI

Table 7 : Commandes des instructions de saut et branchement

Dans la Table 7, le test de la condition détermine s'il y a ou non écriture dans le registre CP. En fait, W_{CP} = Cond . C_{LK} où cond est la condition du branchement conditionnel.

Mnémo	Source bus A	Source bus B	Opération UAL	Opération mémoire	Source bus R	Mux CP/Bus R	Destination Bus R
LIM	R0	IMM	+		UAL	Bus	Rd
SLL	Ra		sll		UAL	Bus	Rd
ROR	Ra		ror		UAL	Bus	Rd
SLR	Ra		slr		UAL	Bus	Rd
SAR	Ra		sar		UAL	Bus	Rd

Table 8 : commandes des instructions de rotation et de décalage

1.7 Séquencement des instructions

1.7.1 Automate de la partie contrôle

L'automate de la partie contrôle de la version non pipelinée est représenté dans la Figure 4. Le codage des états est indiqué. (Q1=0 et Q0=0 indique l'état LI).

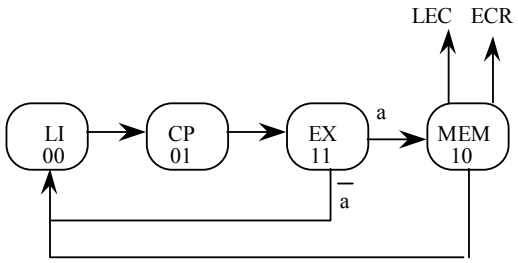

Figure 4 : automate d'états de la version non pipelinée.

1.7.2 Implantation avec des bascules D

La Table 9 donne les entrées des bascules D en fonction des sorties et de l'entrée a, pour implanter l'automate de la Figure 4

a	Q1	Q 0	D1	D0
	0	0	0	1
	0	1	1	1
0	1	1	0	0
1	1	1	1	0
	1	0	0	0

Table 9

$D_1 = \overline{Q_1} Q_0 + a.Q_0$

$D_0 = \overline{Q_1}$

L'entrée a de l'automate s'exprime en fonction du contenu du registre instruction RI.
$a = RI_{15}. RI_{14}.RI_{13}$

1.7.3 Génération détaillée des commandes

On rappelle la liste des commandes nécessaires dans le processeur PCLM-2012 avec le chemin de données utilisé.

- Vers bus A (barrières de bus) : T_{RA} + Mux a, T_{CP}, T_{RAdm}
- Vers bus B (barrière de bus) : T_{RB} + Mux b, T_{RI}
- UAL : +, -, +ret, et, ou, xor, transfert, +1
- Mémoire : Lecture, Ecriture
- Vers bus R (barrière de bus) : T_{UAL}, T_{Mem}
- Multiplexeur : Bus R/CP
- Depuis bus R (écriture registre) : W_{CP}, W_{RI}, W_{RAdM}, W_{R1} à W_{R7}

Pour la lecture de l'instruction, la suite de commandes est T_{CP}, Lecture, T_{mem},W_{RI}
Pour la mise à jour de CP, la suite des commandes est T_{CP}, +1, T_{UAL}, W_{CP}
Pour la phase exécution UAL et calcul d'adresse, les commandes suivantes interviennent en fonction des instructions. Nous listons les instructions qui activent les commandes citées

- T_{RA} : ADD, ADC, SUB, OR, AND, XOR, ANI, ADI, LD, ST, Jump, Call, SRA, ROR, SLA, SLL
- T_{RB} : ADD, ADC, SUB, OR, AND, XOR
- T_{RI} : ANI, ADI, LD, ST, BC, LIM
- T_{CP} : BCond
- T_{RadM} : LD,ST
- UAL
  - + : ADD, ADC, LD, ST, ADI
  - - : SUB
  - et : AND, ANI
  - ou : OR
  - ou exclusif : XOR
  - transfert : LIM, JMP, Call
- T_{UAL} : pour toutes les instructions
- MUX CP/Bus : CP si Call, Bus autrement
- Signaux d'écriture
  - WR1 à WR7 (selon d ou b) : ADD, ADC, SUB, OR, AND, XOR, ANI, XOI, SRA, ROR, SLA, SLL, LIM
  - W_{CP} : BC (avec condition vraie), JMP, Call
  - W_{RAdM} : LD, ST

Pour la phase Mémoire, les commandes suivantes sont activées par les instructions citées.

- T_{RAdM} : LD, ST
- T_{RB} : ST
- Lecture : LD
- Ecriture : ST
- T_{mem} : pour LD et ST
- MUX CP/BUS : BUS
- Ecriture registre : W_{R1} à W_{R7} : LD

On peut maintenant définir les commandes en fonction de l'état de l'automate et des bits du registre d'instructions RI.

Exemples

- Lecture mémoire : Cycle LI + Cycle MEM et LD. Seules les instructions LD et ST peuvent arriver dans l'état mémoire. Le bit RI_{12} est suffisant pour les distinguer.

  Soit $\overline{Q_0}.(\overline{Q_1} + \overline{RI_{12}})$.
- Ecriture mémoire : Cycle MEM et ST

Soit $Q_1\overline{Q_0}RI_{12}$

- Barrière de bus T_{RA} : Cycle EX pour toutes les instructions sauf BC

  Soit $Q_1.Q_0.(\overline{RI_{15}} + RI_{14} + \overline{RI_{13}} + \overline{RI_{12}})$

- Barrière de bus T_{RAdM} : Cycle MEM (pour LD et ST, seules instructions dans ce cycle).

  Soit $Q_1.\overline{Q_0}$

- Barrière de bus T_{CP} : Cycle LI + Cycle CP + Cycle EX pour BC

  Soit $\overline{Q_1} + Q_0RI_{15}.\overline{RI_{14}}.RI_{13}.RI_{12}$

- Ecriture dans le registre R_1 (commande W_{R1}) : (Cycle EX et ADD, ADC, SUB, OR, AND, XOR, ANI, XOI, SRA, ROR, SLA, SLL, LIM et cycle MEM et LD) et registre 1.

  Soit $Q_1Q_0(\overline{RI_{15}}.\overline{RI_{14}} + RI_{15}RI_{14}.\overline{RI_{13}}) + Q_1\overline{Q_0}\,\overline{RI_{12}}$ avec $\overline{RI_{11}}.\overline{RI_{10}}.RI_9$ comme commande de contrôle du démultiplexeur d'horloge ou à inclure dans l'expression si commande d'horloge individuelles.

1.8 Efficacité

Dans la version non pipelinée que nous venons de présenter, les instructions load et store utilisent 4 cycles et toutes les autres instructions utilisent 3 cycles. Si le rapport des instructions load et store sur l'ensemble des instructions est m, alors le nombre moyen de cycles par instruction (CPI) est donné par
CPI = 3 (1-m) + 4 m = 3 + m.
On obtient donc un nombre CPI supérieur à 3. La valeur m = 0.2 conduit à CPI = 3.2.
Diminuer CPI implique de pipeliner les instructions.

1.9 ANNEXE

1.9.1 FORMAT DES INSTRUCTIONS

Les formats d'instruction sont données Figure 13. On utilise les abréviations suivantes :
$RI_{11\text{-}9}$ +a, $RI_{8\text{-}6}$ + b, $RI_{5\text{-}3}$ + d.
Les adresses et les mots mémoire sont 16 bits.
Les registres R0 à R7 sont accessibles sous la forme Ra, Rb ou Rd, où a, b, d représentent un numéro de registre. On rappelle que R0 =0 (accessible en lecture, non modifiable en écriture).
Immédiat et déplacement (bits $RI_{5\text{-}0}$) sont en complément à deux. Il y a extension de signe pour les ajouter à des valeurs sur 16 bits.

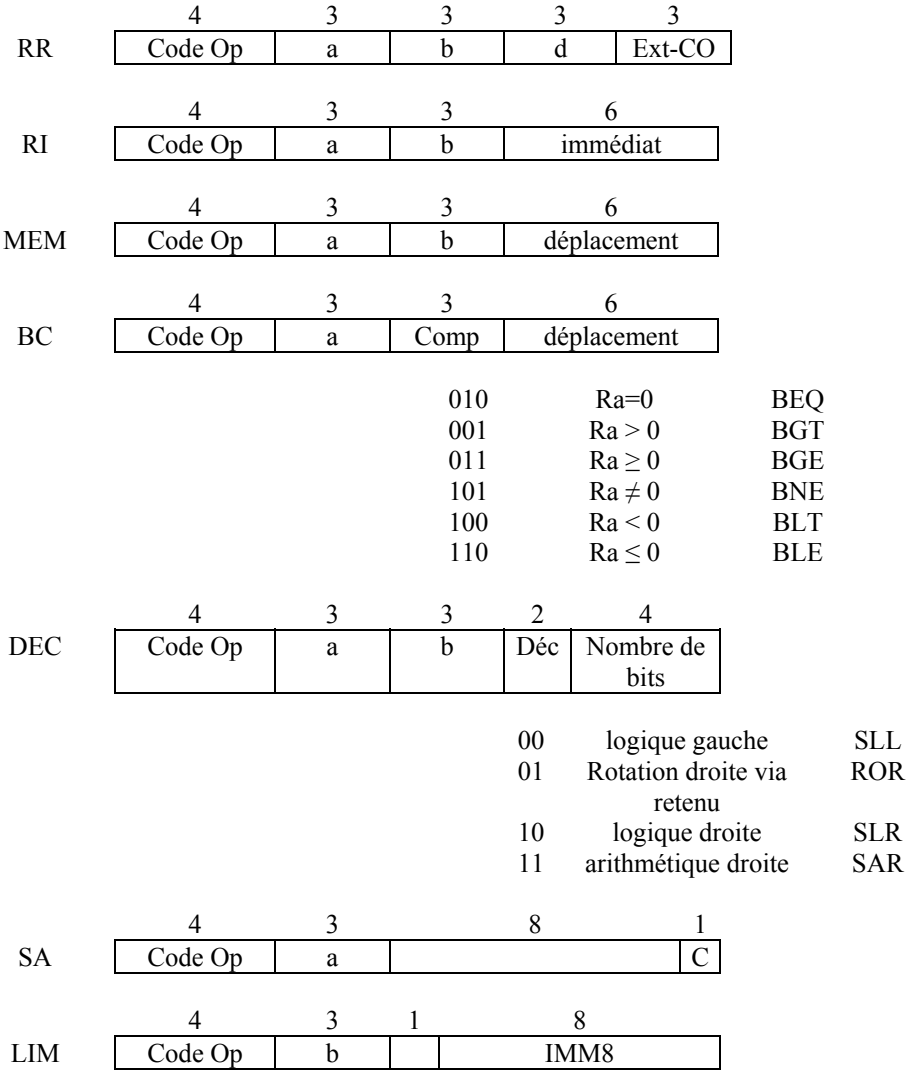

Figure 5 : Formats d'instructions

1.9.2 .JEU D'INSTRUCTIONS

CODE OP	EXT CO	MNÉMO NIQUE	MODE	EFFET
0000	000	ADD	RR	Rd ← Ra + Rb
0000	001	ADC	RR	ret, Rd ← Ra + Rb + ret
0000	100	AND	RR	Rd ← Ra and Rb
0000	110	XOR	RR	Rd ← Ra xor Rb
0000	010	SUB	RR	Rd ← Ra - Rb
0000	101	OR	RR	Rd ← Ra or Rb
0001		ANI	RI	Rb ← Ra and imm
0010		ADI	RI	Rb ← Ra xor imm
1001	0	JMP	SA	CP ←Ra
1001	1	CALL	SA	R7←CP ; CP← Ra
1010		LIM	LIM	Rab_{7-0} ← RI_{7-0}; Rb_{15-8} inchangé
1011	010	BEQ	BC	Si Ra = 0, CP ← CP + déplacement
1011	001	BGT	BC	Si Ra > 0, CP ← CP + déplacement
1011	011	BGE	BC	Si Ra ≥ 0, CP ← CP + déplacement
1011	101	BNE	BC	Si Ra ≠ 0, CP ← CP + déplacement
1011	100	BLT	BC	Si Ra < 0, CP ←CP + déplacement
1011	110	BLE	BC	Si Ra ≤ 0, CP ← CP + déplacement
1100	00	SLL	DEC	Rb ← Ra décalé gauche n bit
1100	01	ROR	DEC	Rb ← Ra rotation droite via retenue n bit
1100	10	SLR	DEC	Rb ← Ra décalé logique droite n bit
1100	11	SAR	DEC	Rb ← Ra décalé arithmétique droite n bit
1110		LD	MEM	Rb ← M(Ra + depl)
1111		ST	MEM	M(Ra + depl) ←Rb

Table 10: Jeu d'instructions