Projet : étude et réalisation d'un microprocesseur RISC

On considère un processeur RISC 32 bits possédant la structure suivante :

- un banc de 32 registres R0-R31 de 32 bits chacun avec 2 ports de lecture et 1 port d'écriture. Le registre R0 est câblé à 0, On peut tenter d'y écrire, mais sa lecture donnera toujours 0 ;

- une Unité Arithmétique et Logique (UAL) avec 2 entrées et une sortie de 32 bits. Elle permet de réaliser les opérations arithmétiques et logiques sur 32 bits du jeu d'instructions donné dans les tableaux ci-après ;

- une logique d'état composée de 4 bits de tests C (Carry), Z (Zero), V (oVerflow), N (Negative) permettant de donner une indication sur l'état des opérations réalisées et de pouvoir réaliser les branchements conditionnels ;

- un chemin de données pipeliné de 5 étages à cycle unique : EI / DI / EX / MEM / ER :

  - Le premier étage EI : réalise l'extraction de l'instruction réalise.
  - Le deuxième étage DI : réalise le décodage d'instruction et extraction des opérandes.
  - Le troisième étage EX : réalise l'exécution et calcul de l'adresse effective.
  - Le quatrième étage MEM : réalise l'accès à la mémoire.
  - Le cinquième étage ER : réalise l'écriture du résultat de l'instruction précédente.

- un registre : compteur de programme (CP) ;

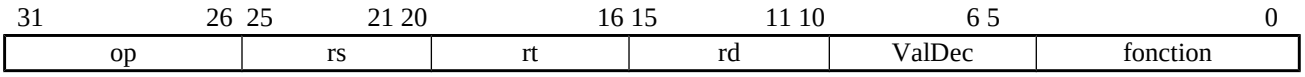
Les modes d'adressage sont de quatre types : registre à registre, immédiat, indirect et relatif. Toutes les instructions s'exécutent en un seul cycle. Les opérations d'accès mémoire peuvent être réalisées au niveau de l'octet, le demi-mot (16 bits) et le mot (32 bits) donc les adresses manipulées sont des adresses octet. Toutes les opérations, arithmétiques et logiques sont effectuées toujours sur des opérandes de 32 bits.

L'architecture interne du processeur est du type Harvard avec deux caches intégrés :

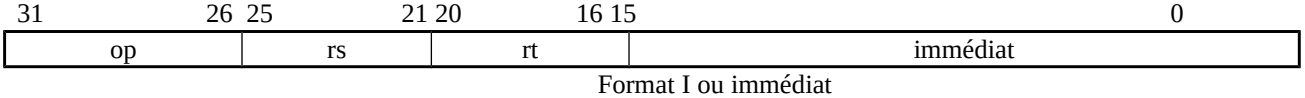
- cache d'instructions de 64 KO géré en application directe, la taille d'un bloc est de 16 mots de 32 bits (64 octets).
- cache de données de 64 KO géré en application associative par ensemble avec 4 blocs par ensemble. La taille d'un bloc est de 16 mots de 32 bits (64 octets).
  - La politique de remplacement est type LRU (Least Recently Used).
  - La politique de mise à jour de la mémoire centrale est du type WB (Write back) : mise jour différé.

Dans toute l'étude on ne considérera pas les exceptions.

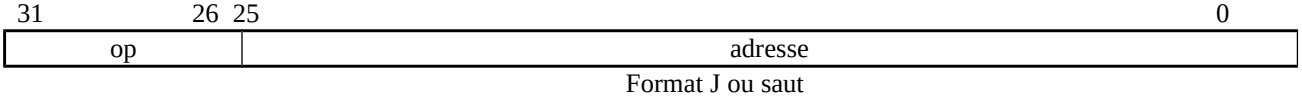
L'architecture RISC considérée ici possède 3 formats d'instructions fixes comme suit :



Format R ou registre



Format I ou immédiat



Format J ou saut

- Le champ **op** : code opération sur 6 bits, indique le format et la nature du traitement à effectuer.
- Le champ **rs** : formé de 5 bits, représente toujours un numéro du registre source
- Le champ **rt** : formé de 5 bits, représente un numéro de registre (source ou destination) ou une condition de branchement.
- Le champ **rd** : formé de 5 bits, représente un numéro de registre de destination.

- Le champ **ValDec** : formé de 5 bits, représente le nombre de décalage dans les instructions de décalage.
- Le champ **fonction** : formé de 5 bits, permet de différencier certaines instructions ayant le même code opération.
- Le champ **immédiat** : formé de 16 bits, représente soit un déplacement signé dans le cas des instructions de branchement, ou une constante signée immédiate dans le cas des opérations arithmétiques signées, ou une constante non signée immédiate dans le cas des opérations logiques et arithmétiques non signées .
- Le champ **adresse** : représente une adresse sur 26 bits.

Le jeu d'instruction est le suivant :

Instruction	op					fonction
lsl	000000	0	rt	rd	ValDec	000000
lsr	000000	0	rt	rd	ValDec	000010
jr	000000	rs	0	0	0	001000
add	000000	rs	rt	rd	0	100000
addu	000000	rs	rt	rd	0	100001
sub	000000	rs	rt	rd	0	100010
subu	000000	rs	rt	rd	0	100011
and	000000	rs	rt	rd	0	100100
or	000000	rs	rt	rd	0	100101
xor	000000	rs	rt	rd	0	100110
nor	000000	rs	rt	rd	0	100111
slt	000000	rs	rt	rd	0	101010
sltu	000000	rs	rt	rd	0	101011
jalr	000000	rs	0	rd	0	001001
bltz	000001	rs	00000	déplacement signé		
bgez	000001	rs	00001	déplacement signé		
bltzal	000001	rs	10000	déplacement signé		
bgezal	000001	rs	10001	déplacement signé		
j	000010	adresse				
jal	000011	adresse				
beq	000100	rs	rt	déplacement signée		
bne	000101	rs	rt	déplacement signée		
blez	000110	rs	0	déplacement signé		
bgtz	000111	rs	0	déplacement signé		
addi	001000	rs	rt*	constante signée (`sign_extend`)		
addiu	001001	rs	rt*	constante signée (`sign_extend`)		
slti	001010	rs	rt*	constante signée (`sign_extend`)		
sltiu	001011	rs	rt*	constante signée (`sign_extend`)		
andi	001100	rs	rt*	constante non signée (`zero_extend`)		
ori	001101	rs	rt*	constante non signée (`zero_extend`)		
xori	001110	rs	rt*	constante non signée (`zero_extend`)		
lui	001111	0	rt*	constante non signée		
lb	100000	Rs**	rt*	déplacement signé		
lh	100001	Rs**	rt*	déplacement signé		
lw	100011	rs**	rt*	déplacement signé		
lbu	100100	rs**	rt*	déplacement signé		
lhu	101000	rs**	rt*	déplacement signé		
sb	101000	rs**	rt	déplacement signé		
sh	101001	rs**	rt	déplacement signé		
sw	101011	rs**	rt	déplacement signé		

* Le registre rt joue le rôle de registre de destination

** Le registre rs joue le rôle de registre de base.

`sign_extend` extension à 32 bits avec recopie du bit de signe dans les 16 bits de poids fort.

`zero_extend` extension à 32 bits avec mise à 0 des 16 bits de poids fort.

Le tableau ci-après donne un exemple de chaque instruction avec son interprétation.

Catégorie	Instruction	Exemple	Signification	Commentaires
Arithm-étique	Addition	add R1,R2,R3	R1 = R2 + R3	3 opdes; excep. possible
	Soustraction	sub R1,R2,R3	R1 = R2 - R3	3 opdes; excep. possible
	Addition immédiat	addi R1,R2,100	R1 = R2 + 100	+ cste; excep. possible
	Addition non signé	addu R1,R2,R3	R1 = R2 + R3	3 opdes; pas d'exception
	Soustr. non signé	subu R1,R2,R3	R1 = R2 - R3	3 opdes; pas d'exception
	Add. imm. non signé	addiu R1,R2,100	R1 = R2 + 100	+ cste; pas d'exception
Logique	ET	and R1,R2,R3	R1 = R2 & R3	3 registres opérandes
	OU	or R1,R2,R3	R1 = R2 I R3	3 registres opérandes
	XOR	xor R1,R2,R3	R1 = R2 xor R3	3 registres opérandes
	Non OU	nor R1,R2,R3	R1 = R2 nor R3	3 registres opérandes
	ET immédiat	andi R1,R2,100	R1 = R2 & 100	2 registres opdes + cste
	OU immédiat	ori R1,R2,100	R1 = R2 I 100	2 registres opdes + cste
	XOR immédiat	xori R1,R2,100	R1 = R2 xor 100	2 registres opdes + cste
	Déc. logique gauche	lsl R1,R2,10	R1 = R2 << 10	Décalage à gauche
	Déc. logique droite	lsr R1,R2,10	R1 = R2 >> 10	Décalage à droite
Chart. imm.	Chart. des pds forts	Lui 100(R1)	R1 = 100 << 16	chargement poids forts
Transfert de Données	Chart. octet signé	Lb R1,100(R2)	R1 = $M[R2+100]_8$	Chart. avec ext. signe
	Chart. demi mot signé	Lh R1,100(R2)	R1 = $M[R2+100]_{16}$	Chart. avec ext. signe
	Chargement mot	lw R1,100(R2)	R1 = M[R2+100]	Chart. mot 32 bits
	Chart. octet non signé	Lbu R1,100(R2)	R1 = $M[R2+100]_8$	Chart. avec ext. des zéros
	Chart. demi non signé	Lhu R1,100(R2)	R1 = $M[R2+100]_{16}$	Chart. avec ext. des zéros
	Rangement octet	sb R1,100(R2)	$M[R2+100] = (R1)_8$	Rangt. Octet pf en mém.
	Rangement demi mot	sh R1,100(R2)	$M[R2+100] = (R1)_{16}$	Rangt. Demi pf en mém.
	Rangement mot	sw R1,100(R2)	M[R2+100] = R1	Rangt. mot en mém.
Branche-ment conditionnel	Branchement si =	beq R1,R2,100	si (R1==R2) aller en CP+4+100	Test d'égalité ; branch. relatif à CP
	Branchement si ≠	bne R1,R2,100	si (R1!=R2) aller en CP+4+100	Test d'inégalité ; brancht. relatif à CP
	Branchement si ≤ 0	blez R1,100	si (R1≤0) aller en CP+4+100	Test ≤ 0 compl. à 2 ; brancht. relatif à CP
	Branchement si > 0	bgtz R1,100	si (R1>0) aller en CP+4+100	Test > 0 compl. à 2 ; brancht. relatif à CP
	Branchement si < 0	bltz R1,100	si (R1<0) aller en CP+4+100	Test < 0 compl. à 2 ; brancht. relatif à CP
	Branchement si ≥ 0	bgez R1,100	si (R1≥0) aller en CP+4+100	Test ≥ 0 compl. à 2 ; brancht. relatif à CP
	Bt. Avec lien si < 0	Bltzal R1,100	si (R1<0) aller en CP+4+ 100 avec R31 <= CP+4	Test < 0 compl. à 2 ; brancht. relatif à CP
	Bt. Avec lien si ≥ 0	Bgezal R1,100	si (R1≥0) aller en CP+4+ 100 avec R31 <= CP+4	Test ≥ 0 compl. à 2 ; brancht. relatif à CP
	Positionner si <	slt R1,R2,R3	si (R2 < R3) alors R1=1 ; sinon R1=0	Test d'infériorité ; complément à 2
	Positionner si < immédiat	slti R1,R2,100	si (R2 < 100) alors R1=1 ; sinon R1=0	Test < constante ; complément à 2
	Positionner si < non signé	sltu R1,R2,R3	si (R2 < R3) alors R1=1 ; sinon R1=0	Test d'infériorité ; nomb. entiers naturels
	Positionner si < immédiat non signé	sltiu R1,R2,100	si (R2 < 100) alors R1=1 ; sinon R1=0	Test < constante ; nomb. entiers naturels
Saut Incondition-nel	Saut	j 10000	aller en 10000	Saut vers adresse de destination
	Saut par registre	jr R1	aller à l'adresse contenu dans R1	Pour switch ; retour de sous programme
	Saut avec lien	jal 10000	R31 = CP+4 ; aller en 10000	Pour appel de sous programme

	Saut avec lien par registre	Jalr R1,R3	R3 <= CP+4 ; Aller à l'adresse dans R1	Pour appel de sous programme

Travail demandé : Conception et simulation en VHDL du chemin de données et de l'unité de contrôle de l'ensemble du processeur pour le jeu d'instructions décrit dans le tableau ci-dessus, ainsi que les mémoires caches d'instructions et de données.

Le travail sera sanctionné par une présentation orale, une démonstration sur machine ainsi qu'un rapport décrivant l'essentiel de votre conception en justifiants vos choix et solutions. Les sources VHDL des différents modules seront fournies en annexe dans le rapport.

N.B. : Les rapports doivent rendus au plus tard le jour de la soutenance.