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) ;

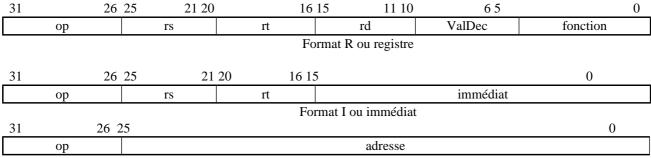
Les modes d'adressage sont de quatre types : registre à registre, immédiat, indirect et relatif. Toutes les instructions s'exécutent en un seul cycle. Les opérations d'accès mémoire peuvent être réalisées au niveau de l'octet, le demi-mot (16 bits) et le mot (32 bits) donc les adresses manipulées sont 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 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	dé	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	100101	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

 $sign_extend$ $\[\]$ 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.

^{**}Le registre rs joue le rôle de registre de base.

Arithm fétique Addition immédiat Aver R1, R2, R3 R1 = R2 + R3 A podes pas d'exception B = R2 + 100 + este; pas d'exception + este; pas d'exception Addition immédiat Are R1, R2, R3 R1 = R2 + R3 A speistres opérandes Are R1, R2, R3 R1 = R2 + R3 A speistres opérandes Are R1, R2, R3 R1 = R2 + R3 A sregistres opérandes A sregistres	Catégorie	Instruction	Exemple	Signification	Commentaires
Addition immédiat addi R1,R2,100 R1 = R2 + 100 + cste; excep, possible addu R1,R2,R3 R1 = R2 + R3 3 opdes; pas d'exception addi R1,R2,100 R1 = R2 + R3 3 opdes; pas d'exception addi R1,R2,R3 R1 = R2 + R3 3 opdes; pas d'exception addit R1,R2,R3 R1 = R2 R3 3 opdes; pas d'exception addit R1,R2,R3 R1 = R2 R3 3 registres opérandes R0 R1,R2,R3 R1 = R2 R3 3 registres opérandes R0 R1,R2,R3 R1 = R2 R3 3 registres opérandes R0 R1,R2,R3 R1 = R2 R3 R3 3 registres opérandes R0 R1,R2,R3 R1 = R2 R3 R3 3 registres opérandes R0 R1,R2,R3 R1 = R2 R3 R3 3 registres opérandes R0 R1,R2,R3 R1 = R2 R3 R3 3 registres opérandes R1 R1,R2,100 R1 = R2 R2 R1 R0 R3 3 registres opérandes R1 R1,R2,100 R1 = R2 R2 R1 R0 R3 3 registres opérandes R1 R1,R2,100 R1 = R2 R2 R1 R0 R3 3 registres opérandes R1 R1,R2,R3 R1 = R2 R3 R1 = R2 R1 R3 3 registres opérandes R1 R1,R2,R3 R1 = R2 R3 R1 R2 R1 R3 3 registres opérandes R1 R1,R2,R3 R1 = R2 R3 R1 R2 R1 R3 3 registres opérandes R1 R1,R2,R3 R1 = R2 R3 R1 R2 R1 R3 3 registres opérandes R1 R1,R2,R3 R1 = R2 R3 R1 R2 R1 R3 3 registres opérandes R1 R1,R2,R3 R1 R2,R3 R1 R2 R3 R1 R3 R3 R1 R2 R3 R1 R2 R3 R1 R3 R3 R3 R2 R3 R3 R2 R3 R3 R2 R3 R3 R3 R2 R3		Addition	add R1,R2,R3	R1 = R2 + R3	3 opdes; excep. possible
étique Addition non signé addu R.I.R.2.R.3 R.I. = R.2 + R.3 3 opdes; pas d'exception Add. imm. non signé aux R.I.R.2.R.3 R.I. = R.2 + R.3 3 opdes; pas d'exception ET and R.I.R.2.R.3 R.I. = R.2 + R.0 + cste; pas d'exception OU or R.I.R.2.R.3 R.I. = R.2 L.R.3 3 registres opérandes NOO NOR NOR R.I.R.2.R.3 R.I. = R.2 k.R.3 3 registres opérandes NOO NOR NOR R.I.R.2.R.3 R.I. = R.2 nor R.3 3 registres opérandes NOO NOR NOR R.I.R.2.R.3 R.I. = R.2 nor R.3 3 registres opérandes Logique F.I. middiat and R.I.R.2.100 R.I. = R.2 nor R.3 3 registres opérandes Logique OU immédiat ori R.I.R.2.100 R.I. = R.2 nor R.3 3 registres opérandes Logique Logique droite Is R.I.R.2.100 R.I. = R.2 nor R.3 3 registres opérandes Logique droite Is R.I.R.2.100 R.I. = R.2 nor R.3 3 registres opérandes Chart. cotte in mémdiat nor R.I.R.2.100 R.I. = R.2.100 2 celaige dardie Chart. d	Arithm-	Soustraction	sub R1,R2,R3	R1 = R2 - R3	3 opdes; excep. possible
Soust. non signé Add. imm. non signé addiu R.I.R.2.100 R1 = R2 + 100 + cste; pas d'exception Add. imm. non signé addiu R.I.R.2.100 R1 = R2 & R3 3 registres opérandes COU or R.I.R.2.R3 R1 = R2 I R3 3 registres opérandes R1 = R2 & R3 R3 = R2 & R3 3 registres opérandes R1 = R2 & R3 R3 = R2 & R3 3 registres opérandes R1 = R2 & R3 R3 = R2 & R3 3 registres opérandes R1 = R3 & R3 & R3 = R2 & R3 R3 = R3 & R3 = R3 & R3 = R3 & R3 R3 = R3 & R3 &		Addition immédiat	addi R1,R2,100	R1 = R2 + 100	+ cste; excep. possible
Add, imm, nor signé Addi in R.I.R.2.100 R.I. = R.2 + 100 stetic; pas d'exception	étique	Addition non signé	addu R1,R2,R3	R1 = R2 + R3	3 opdes; pas d'exception
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	_	Soustr. non signé	subu R1,R2,R3	R1 = R2 - R3	3 opdes; pas d'exception
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		Add. imm. non signé	addiu R1,R2,100	R1 = R2 + 100	+ cste; pas d'exception
XOR		ET	and R1,R2,R3	R1 = R2 & R3	3 registres opérandes
Non OU		OU	or R1,R2,R3	R1 = R2 I R3	3 registres opérandes
Logique ET immédiat		XOR	xor R1,R2,R3	R1 = R2 xor R3	3 registres opérandes
Logique OU immédiat ori R1,R2,100 R1 = R2 I 100 2 registres opdes + este XOR immédiat xori R1,R2,100 R1 = R2 xor 100 2 registres opdes + este Déc. logique dauche Ls R1,R2,10 R1 = R2 xor 100 2 registres opdes + este Chart. dem Déc. logique droite Ls R1,R2,10 R1 = R2 >> 10 Décalage à gauche Chart. des pds forts Lui 100(R1) R1 = 100 << 16		Non OU	nor R1,R2,R3	R1 = R2 nor R3	3 registres opérandes
XOR immédiat Xori R1,R2,100 R1 = R2 xor 100 2 registres opdes + cste Déc. logique gauche Isi R1,R2,10 R1 = R2 << 10 Décalage à gauche Déc. logique droite Isi R1,R2,10 R1 = R2 << 10 Décalage à gauche Déc. logique droite Isi R1,R2,10 R1 = R2 >> 10 Décalage à gauche Décalage à forite Chart. timm. Chart. des pds forts Lui 100(R1) R1 = 100 << 16 chargement poids forts Chart. cotte signé Lb R1,100(R2) R1 = M[R2+100]s Chart. avec ext. signe Chart. dem mot signé Lb R1,100(R2) R1 = M[R2+100]s Chart. avec ext. signe Chart. det non signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Chart. det non signé Chart. loctet non signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Rangement octet sb R1,100(R2) M[R2+100] = (R1)s Rangt. Octet pf en mém. Rangement demi mot sh R1,100(R2) M[R2+100] = (R1)s Rangt. Demi pf en mém. Rangement mot sw R1,100(R2) M[R2+100] = (R1)s Rangt. Demi pf en mém. Rangt.		ET immédiat	andi R1,R2,100	R1 = R2 & 100	2 registres opdes + cste
Déc. logique gauche Déc. logique droite Ist R1,R2,10 R1 = R2 <> 10 Décalage à gauche Décalage à forte Chart. imm. Chart. dep 4s forts Lui 100(R1) R1 = R2 >> 10 Décalage à droite Chart. octet signé Lb R1,100(R2) R1 = M[R2+100]s Chart. avec ext. signe Chart. demi mot signé Lb R1,100(R2) R1 = M[R2+100]s Chart. avec ext. signe Chart. demi mot signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Chart. demi mot signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Rangement demi mot signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Rangement demi mot sh R1,100(R2) M[R2+100] = (R1)s Rangt. Octet pf en mém. Rangement si = beq R1,R2,100 M[R2+100] = (R1)s Rangt. Demi pf en mém. Branchement si = beq R1,R2,100 si (R1=R2) aller en CP+4+100 Test d'égalité ; branch. relatif à CP Branchement si > 0 bgtz R1,100 si (R150) aller en CP+4+100 Test d'inégalité ; branch. relatif à CP Branchement si > 0 bgtz R1,100 si (R1>0) aller en CP+4+100 Test <0 compl. à 2 ; branch. relatif à CP	Logique	OU immédiat	ori R1,R2,100	R1 = R2 I 100	2 registres opdes + cste
Chart. imm. Déc. logique droite Isr R1,R2,10 R1 = R2 >> 10 Décalage à droite Chart. des pds forts Lui 100(R1) R1 = 100 << 16 chargement poids forts Chart. octet signé Lb R1,100(R2) R1 = M[R2+100]s Chart. avec ext. signe Chart. demi mot signé Lb R1,100(R2) R1 = M[R2+100]s Chart. avec ext. signe Chart. octet non signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Chart. demi non signé Lbu R1,100(R2) R1 = M[R2+100]s Chart. avec ext. des zéros Rangement octet sb R1,100(R2) M[R2+100] = (R1)s Rangt. octet pf en mém. Rangement demi mot sb R1,100(R2) M[R2+100] = (R1)s Rangt. Demi pf en mém. Rangement mot sw R1,100(R2) M[R2+100] = R1 Rangt. mot en mém. Branchement si = beq R1,R2,100 si (R1=R2) aller en Test d'égalité ; branch. CP+4+100 relatif à CP Test d'égalité ; branch. Branchement si ≤ 0 blez R1,100 si (R1≤0) aller en Test d'ocompl. à 2 ; CP+4+100 rest d'égalité ; branch. relatif à CP Br		XOR immédiat	xori R1,R2,100	R1 = R2 xor 100	2 registres opdes + cste
Chart. imm. Chart. des pds forts Lui 100(R1) R1 = 100 << 16		Déc. logique gauche	lsl R1,R2,10	R1 = R2 << 10	Décalage à gauche
Transfert de Données Transfert de Données Transfert de Données Example de Données Transfert de Données Transfert de Données Example de Données Transfert de Données Example de Données Transfert de Données Example de Sous Ex		Déc. logique droite	lsr R1,R2,10	R1 = R2 >> 10	Décalage à droite
Transfert de Données Chart. demi mot signé Chart. octet non signé Chart. octet non signé Chart. demi non signé Sh R1,100(R2) R1 = M[R2+100] = (R1)s Chart. avec ext. des zéros Chart. demi non signé MR2+100] = (R1)s Rangt. Demi pf en mém. Rangement oxi Rangement oxi Rangement oxi sh R1,100(R2) M[R2+100] = (R1)s Rangt. Demi pf en mém. Rangt. Deli pa mém. Rangt. Deli pa mém. Pest d'inégalité : brancht. relatif à CP Test d'ospalité : brancht. relatif à CP Test d'infériorité : compl. à 2 : brancht. relatif à CP Test d'infériorité : nomb. signé Sinon R1=0 Positionner si < slitu R1,R2,R3 si (R2 < R3) alors R1=1 : Test d'infériorité : nomb. entiers naturels Saut Incondition- Rangement ox sh R1,1000 Rangement demi mot sh R1,10000 Rangement demi mot sh R1,10000 Ralti R2+1001 = (R1)s Rangt. Deli pa mém. R	Chart. imm.	Chart. des pds forts	Lui 100(R1)	R1 = 100 << 16	chargement poids forts
Transfert de Données Chart. demi mot signé Chart. octet non signé Chart. octet non signé Chart. demi non signé Sh R1,100(R2) R1 = M[R2+100] = (R1)s Chart. avec ext. des zéros Chart. demi non signé MR2+100] = (R1)s Rangt. Demi pf en mém. Rangement oxi Rangement oxi Rangement oxi sh R1,100(R2) M[R2+100] = (R1)s Rangt. Demi pf en mém. Rangt. Deli pa mém. Rangt. Deli pa mém. Pest d'inégalité : brancht. relatif à CP Test d'ospalité : brancht. relatif à CP Test d'infériorité : compl. à 2 : brancht. relatif à CP Test d'infériorité : nomb. signé Sinon R1=0 Positionner si < slitu R1,R2,R3 si (R2 < R3) alors R1=1 : Test d'infériorité : nomb. entiers naturels Saut Incondition- Rangement ox sh R1,1000 Rangement demi mot sh R1,10000 Rangement demi mot sh R1,10000 Ralti R2+1001 = (R1)s Rangt. Deli pa mém. R			, , ,	$R1 = M[R2+100]_8$	<u> </u>
Transfert de Données Chart. cottet non signé Lbu R1,100(R2) R1 = M[R2+100] Chart. mot 32 bits		_			
Transfert de Données Chart. demi non signé Rangement octet Rangement octet Rangement demi mot Rangement demi mot Rangement mot Rangement mot Rangement mot Rangement mot Rangement si = beq R1,R2,100 Branchement si ≤ 0 Branchement si ≤ 0 Branchement si ≤ 0 Branchement si > 0 Branchement si > 0 Branchement si > 0 Branchement si < 0 Brancheme			lw R1,100(R2)		Chart. mot 32 bits
de Données Rangement demi non signé Rangement octet Rangement demi mot Rangement demi mot Rangement demi mot Rangement mot Rangement mot Rangement si = Branchement si = Branchement si ≤ 0 Branchement si > 0 Branchement si < 0 Br	Transfert				Chart. avec ext. des zéros
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $					
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $					Rangt. Octet of en mém.
Rangement mot $sw R1,100(R2)$ $M[R2+100] = R1$ Rangt. mot en mém. Branchement $si = beq R1,R2,100$ $si (R1=R2)$ aller en $CP+4+100$ relatif à CP $CP+4+100$ relatif à CP $CP+4+100$ relatif à CP $CP+4+100$ relatif à CP $CP+4+100$ $CP+4+100$ relatif à CP $CP+4+100$			1 1 1		
Branchement si = beq R1,R2,100 si (R1==R2) aller en CP+4+100 relatif à CP Branchement si ≠ bne R1,R2,100 si (R1!=R2) aller en CP+4+100 relatif à CP Branchement si ≠ bne R1,R2,100 si (R1!=R2) aller en CP+4+100 relatif à CP Branchement si ≤ 0 blez R1,100 si (R1≤0) aller en CP+4+100 brancht. relatif à CP Branchement si > 0 bltz R1,100 si (R1>0) aller en CP+4+100 brancht. relatif à CP Branchement si < 0 bltz R1,100 si (R1>0) aller en CP+4+100 brancht. relatif à CP Branchement si < 0 bltz R1,100 si (R1≥0) aller en CP+4+100 brancht. relatif à CP Branchement si ≥ 0 bgez R1,100 si (R1≥0) aller en CP+4+100 brancht. relatif à CP Branchement si < 0 bltz R1,100 si (R1≥0) aller en CP+4+ Test ≥ 0 compl. à 2 ; brancht. relatif à CP Bt. Avec lien si < 0 Bltzal R1,100 si (R1≥0) aller en 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+ Test ≥ 0 compl. à 2 ; brancht. relatif à CP Bt. Avec lien si < 0 Bgezal R1,100 si (R1≥0) aller en CP+4+ Test ≥ 0 compl. à 2 ; brancht. relatif à CP Branchement si < sit R1,R2,R3 si (R2 < R3) alors R1=1 ; rest d'infériorité ; complément à 2 Positionner si < slti R1,R2,R3 si (R2 < R3) alors R1=1 ; Test d'infériorité ; nomb. entiers naturels Positionner si < sltiu R1,R2,100 si (R2 < R3) alors R1=1 ; Test d'infériorité ; nomb. entiers naturels Saut sut vers adresse de destination sin R1 = 0 Pour switch ; retour de sous programme Pour switch ; retour de sous programme Pour appel de sous					
Branchement si ≠ bne R1,R2,100 si (R1!=R2) aller en CP+4+100 relatif à CP Branchement si ≤ 0 blez R1,100 si (R1!=R2) aller en CP+4+100 relatif à CP Branchement si ≤ 0 blez R1,100 si (R1≤0) aller en CP+4+100 brancht. relatif à CP Branchement si > 0 bgtz R1,100 si (R1>0) aller en CP+4+100 brancht. relatif à CP Branchement si < 0 bltz R1,100 si (R1>0) aller en CP+4+100 brancht. relatif à CP Branchement si ≥ 0 bgez R1,100 si (R1≥0) aller en CP+4+100 brancht. relatif à CP Branchement si ≥ 0 bgez R1,100 si (R1≥0) aller en CP+4+100 brancht. relatif à CP Branchement si ≥ 0 bgez R1,100 si (R1≥0) aller en 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+ brancht. relatif à CP Bt. Avec lien si ≥ 0 Bgezal R1,100 si (R1≥0) aller en CP+4+ brancht. relatif à CP Bt. Avec lien si ≥ 0 sti R1,R2,R3 si (R2 < R3) alors R1=1; Test d'infériorité ; sinon R1=0 complément à 2 Positionner si < slt R1,R2,R3 si (R2 < R3) alors R1=1; Test d'infériorité ; nomb. signé sinon R1=0 entiers naturels Positionner si < slt R1,R2,R3 si (R2 < R3) alors R1=1; Test d'infériorité ; nomb. entiers naturels Saut sinon R1=0 saut par registre jr R1 aller à l'adresse contenu dans R1 programme Saut avec lien jal 10000 R31 = CP+4; aller en Pour appel de sous			+		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$,,		,
Branchement si ≤ 0 blez R1,100 si (R1≤0) aller en CP+4+100 brancht. relatif à CP si (R1<0) aller en CP+4+100 brancht. relatif à CP si (R1<0) aller en CP+4+100 brancht. relatif à CP brancht. relati		Branchement si ≠	bne R1,R2,100	si (R1!=R2) aller en	Test d'inégalité ; brancht.
Branchement si > 0 bgtz R1,100 si (R1>0) aller en CP+4+100 brancht. relatif à CP si (R1<0) aller en CP+4+100 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4+100 avec R31 <= CP+4 brancht. relatif à CP cP+4 sinon R1=0 complément à 2 complémen				CP+4+100	relatif à CP
Branchement si > 0 bgtz R1,100 si (R1>0) aller en CP+4+100 brancht. relatif à CP Branchement si < 0 bltz R1,100 si (R1<0) aller en CP+4+100 brancht. relatif à CP Branchement si ≥ 0 bgez R1,100 si (R1≥0) aller en CP+4+100 brancht. relatif à CP Branchement si ≥ 0 bgez R1,100 si (R1≥0) aller en CP+4+100 brancht. relatif à CP Bt. Avec lien si < 0 Bltzal R1,100 si (R1≥0) aller en 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+ Test < 0 compl. à 2 ; brancht. relatif à CP Bt. Avec lien si ≥ 0 Bgezal R1,100 si (R1≥0) aller en CP+4+ Test ≥ 0 compl. à 2 ; brancht. relatif à CP Bt. Avec lien si ≥ 0 Bgezal R1,100 si (R2 < R3) alors R1=1 ; complément à 2 Positionner si < slt R1,R2,R3 si (R2 < R3) alors R1=1 ; complément à 2 Positionner si < non signé sinon R1=0 sinon R1=0 rest < constante ; complément à 2 Positionner si < slt R1,R2,R3 si (R2 < R3) alors R1=1 ; Test d'infériorité ; nomb. signé sinon R1=0 entiers naturels Positionner si < slt R1,R2,100 si (R2 < 100) alors R1=1 ; Test < constante ; nomb. entiers naturels sinon R1=0 aller en 10000 Saut vers adresse de destination Pour switch ; retour de sous dans R1 programme Pour appel de sous		Branchement si ≤ 0	blez R1,100	si (R1≤0) aller en	
Branchement si < 0 bltz R1,100 si (R1<0) aller en CP+4+100 brancht. relatif à CP rest < 0 compl. à 2 ; branchement si ≥ 0 bgez R1,100 si (R1<0) aller en CP+4+100 brancht. relatif à CP rest ≥ 0 compl. à 2 ; brancht. relatif à CP rest ≥ 0 com					
Branchement si < 0		Branchement $si > 0$	bgtz R1,100		_
BranchementBranchement si ≥ 0bgez R1,100si (R1≥0) aller en CP+4+100Test ≥ 0 compl. à 2; brancht. relatif à CPconditionnelBt. Avec lien si < 0					
Branchement mentBranchement si ≥ 0bgez R1,100si (R1≥0) aller en CP+4+100Test ≥ 0 compl. à 2 ; brancht. relatif à CPconditionnelBt. Avec lien si < 0		Branchement si < 0	bltz R1,100		I
ment conditionnelBt. Avec lien si < 0					
ConditionnelBit. Avec lien si < 0		Branchement $si \ge 0$	bgez R1,100	, , ,	
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$					
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	conditionnel	Bt. Avec lien si < 0	Bltzal R1,100		_
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		D: 4 1: '> 0	D 1 D1 100		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		Bt. Avec lien si ≥ 0	Bgezal R1,100	, ,	
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		Positionnar si <	clt D1 D2 D2		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		rositionner si <	Sit K1,K2,K3		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		Positionner si <	slti R1 R2 100		*
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$			Std 1(1,1(2,100		-
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$			sltu R1 R2 R3		-
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$			5144 141,142,143		
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$			sltiu R1,R2,100		
SautSautj 10000aller en 10000Saut vers adresse de destinationIncondition-Saut par registrejr R1aller à l'adresse contenu dans R1Pour switch ; retour de sous programmenelSaut avec lienjal 10000R31 = CP+4 ; aller enPour appel de sous					
Incondition-Saut par registrejr R1aller à l'adresse contenu dans R1Pour switch ; retour de sous programmenelSaut avec lienjal 10000R31 = CP+4 ; aller enPour appel de sous	Saut		j 10000	\	
nel Saut avec lien jal 10000 R31 = CP+4; aller en Pour appel de sous	Incondition-	Saut par registre	jr R1	aller à l'adresse contenu	Pour switch; retour de sous
				dans R1	
	nel	Saut avec lien	jal 10000	R31 = CP+4; aller en	Pour appel de sous
10000 programme				10000	programme

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