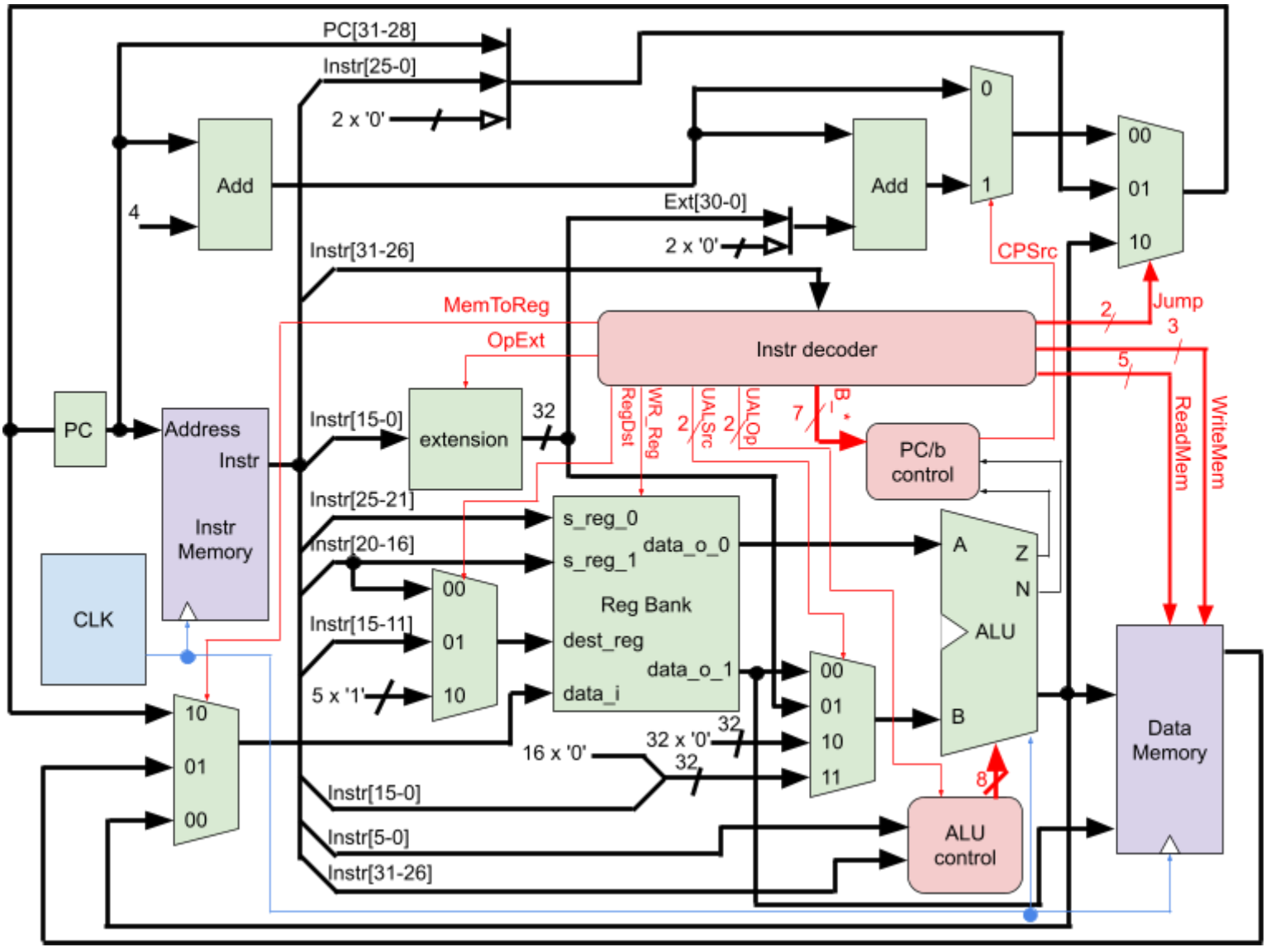
TP CSM - M2 SIAME

Nous allons désormais réaliser la logique de contrôle du processeur, puis assembler tous les éléments développés jusqu'à présent pour réaliser le processeur complet.



I. Introduction - Retour sur les instructions MIPS

bits	31...26	25...21	20...16	15...11	10...6	5...0
I type	op	rs	rt	immediate		
J type	op	target				
R type	0	rs	rt	rd	shamd	funct

Les instructions embarquant un immédiat sont codées avec le type I (y compris les branchements), les sauts (jump) sont codées avec le type J et le reste des instructions est codé avec le type R (Register).
Les bits 31 à 26 servent toujours à encoder l'opcode de l'instruction. Pour les types I et R, les indices de deux registres opérandes sont codés sur les bits 25 à 21 et 20 à 16. Pour le type I, l'immédiat est codé sur les bits 15 à 0. Pour le type R, les bits 10 à 6 encodent un éventuel shift (0 si pas de shift).
On note en particulier que les instructions de type R ont toutes leur opcode à 0. La différence entre ces instructions est encodée dans les 6 bits de poids faible (les bits de fonction).
Les instructions de saut (jump) encodent la cible du saut sur les bits 25 à 0 (adresse absolue - les 4 bits de poids forts étant pris sur le PC) -- les branchements quant à eux permettent de faire des comparaisons entre registres, ce qui laisse moins de place pour encoder une adresse => les branchements (type I) encodent dans le champ immédiat une adresse relative au PC actuel.
Une particularité supplémentaire est que les instructions jr (jump to register) et jalr (jump and link to register) sont des instructions de saut nécessitant un registre en entrée => elles sont encodées comme des instructions de type R et non de type J.

II. Décodeur

Les bits 31 à 26 de l'instruction courante (l'opcode) sont envoyés dans un décodeur qui génère les signaux de contrôle correspondants.
Pour cela on envoie le signal d'entrée **in** dans un décodeur 6 -> 2^6, puis on combine les différentes sorties du décodeur (minterms m_i) pour générer les signaux de contrôle.

Port	Taille	I/O	Formule	Rôle
in	6 bits	Entrée		Bits 31...26 de l'instruction courante
MemToReg	2 bits	Sortie	bit_1: m_0+ m_1+ m_3 bit_0: m_{32}+ m_{33}+ m_{35}+ m_{36}+ m_{37}	Contrôle du multiplexeur de source du banc de registre
OpExt	1 bit	Sortie	m_1+m_4+m_5+m_6+ m_7+m_8+m_{10}+m_{32} +m_{33}+m_{35}+m_{36}+ m_{37}+m_{40}+m_{41}+ m_{43}	Contrôle de la logique d'extension des immédiats
RegDst	2 bits	Sortie	bit_1: m_1 + m_3 bit_0: m_0	Contrôle du multiplexeur du registre destination
WR_Reg	1 bit	Sortie	m_0+m_1+m_3+m_8+ m_9+m_{10}+m_{11}+m_{12} +m_{13}+m_{14}+m_{15}+ m_{32}+m_{33}+m_{35}+ m_{36}+m_{37}	Contrôle de l'écriture dans le registre
UAL_Src	2 bits	Sortie	bit_1: m_1 + m_{15} bit_0: m_8+m_9+m_{10}+m_{11} +m_{12}+m_{13}+m_{14}+ m_{15}	Contrôle du multiplexeur du 2ème opérande de l'ALU
UAL_Op	2 bits	Sortie	bit_1: m_0+m_8+m_9+m_{10}+ m_{11}+m_{12}+m_{13}+m_1 $_4$+m_{15} bit_0: m_1+m_4+m_5+m_6+ m_7+m_8+m_9+m_{10}+ m_{11}+m_{12}+m_{13}+ m_{14}+m_{15}	Sélection du type d'opération de l'ALU
B_eq	1 bit	Sortie	m_4	
B_ne	1 bit	Sortie	m_5	
B_lez	1 bit	Sortie	m_6	
B_gtz	1 bit	Sortie	m_7	

B_bltz	1 bit	Sortie	m_1	
B_gez	1 bit	Sortie	m_1	
B_gez_Al & B_ltz_Al	1 bit	Sortie	m_1	
EcrireMem_W	1 bit	Sortie	m_{43}	Contrôle d'écriture de la mémoire de données (mot)
EcrireMem_H	1 bit	Sortie	m_{41}	Contrôle d'écriture de la mémoire de données (demi-mot)
EcrireMem_B	1 bit	Sortie	m_{40}	Contrôle d'écriture de la mémoire de données (octet)
LireMem_W	1 bit	Sortie	m_{35}	Contrôle de lecture de la mémoire de données (mot)
LireMem_SH	1 bit	Sortie	m_{33}	Contrôle de lecture de la mémoire de données (demi-mot signé)
LireMem_UH	1 bit	Sortie	m_{37}	Contrôle de lecture de la mémoire de données (demi-mot non signé)
LireMem_SB	1 bit	Sortie	m_{32}	Contrôle de lecture de la mémoire de données (octet signé)
LireMem_UB	1 bit	Sortie	m_{36}	Contrôle de lecture de la mémoire de données (octet

				non signé)
Jump	2 bits	Sortie	bit_1: m_0 bit_0:m_2 + m_3	Contrôle du multiplexeur de choix du prochain PC

III. Signaux de sélection de l'opération ALU

Pour générer les signaux Sel, Slt_slti et Enable_V qui contrôlent l'ALU, on va utiliser les signaux de contrôle UAL_Op (généré à la section précédente), les codes fonctions des instructions R (bits 5 à 0) et les bits de l'opcode des instructions arithmétiques I (bits 31 à 26).

$Sel_0 = UALOp1.\overline{UALOp_0}.(\overline{F_5}.\overline{F_3}.\overline{F_2}.\overline{F_1}.\overline{F_0} + F_5.\overline{F_3}.F_2.\overline{F_1}.F_0 + \overline{F_5}.\overline{F_3}.F_2.F_1.\overline{F_0} + \overline{F_5}.F_3.\overline{F_2}.F_1) + UALOp_1.UALOp_0.(Op_1 + Op_2.\overline{Op_1}.\overline{Op_0})$

$Sel_1 = \overline{UALOp_1} + UALOp_1.\overline{UALOp_0}.(\overline{F_3}.\overline{F_2}.\overline{F_0} + \overline{F_5}.\overline{F_3}.F_2.\overline{F_1}.\overline{F_0} + \overline{F_5}.F_3.\overline{F_2}.\overline{F_1}.F_0 + F_5.\overline{F_3}.\overline{F_2}.F_0 + \overline{F_5}.F_3.\overline{F_2}.F_1) + UALOp_1.UALOp_0.\overline{Op_2}$

$Sel_2 = UALOp_1.\overline{UALOp_0}.(\overline{F_5}.\overline{F_3}.\overline{F_2}.\overline{F_0} + \overline{F_5}.\overline{F_3}.F_2.F_1) + UALOp_1.UALOp_0.Op_2.Op_1.\overline{Op_0}$

$Sel_3 = \overline{UALOp_1}.UALOp_0 + UALOp_1.\overline{UALOp_0}.(F_5.\overline{F_3}.\overline{F_2}.F_1 + \overline{F_5}.F_3.\overline{F_2}.F_1) + UALOp_1.UALOp_0.\overline{Op_2}.Op_1$

$Slt\_slti = UALOp_1.\overline{UALOp_0}.F_3.\overline{F_2}.F_1.\overline{F_0} + UALOp_1.UALOP_0.\overline{Op_2}.Op_1.\overline{Op_0}$

$Enable\_V = UALOP_1.\overline{UALOp_0}.F_5.\overline{F_2}.\overline{F_0} + UALOp_1.UALOp_0.\overline{Op_2}.\overline{Op_0}$

IV. Sélection du prochain PC CPSrc

Le signal CPSrc contrôle un multiplexeur qui permet de sélectionner le prochain PC entre:

- La valeur du PC courant + 4 (signal à 0)
- La valeur résultant d'une instruction de branchement conditionnel (signal à 1 si la condition est vraie)

Ce signal doit donc être mis à 1 si et seulement si l'instruction courante est un branchement conditionnel et que sa condition est vraie.

Instruction	minterm	rt4	rt0	condition
beq	m4	X	X	Z
bne	m5	X	X	$\overline{Z}$
blez	m6	X	X	N+Z
bgtz	m7	X	X	$\overline{N}.\overline{Z}$
bltz	m1	0	0	$N.\overline{Z}$
bgez	m1	0	1	$\overline{N} + Z$
bltzal	m1	1	0	$N.\overline{Z}$
bgezal	m1	1	1	$\overline{N} + Z$

On obtient donc:

CPSrc = $m_4.Z + m_5.\overline{Z} + m_6.(N + Z) + m_7.\overline{N}.\overline{Z} + m_1.(N.\overline{Z}.\overline{rt_0} + (\overline{N} + Z).rt_0$

V. Assemblage du processeur complet

En utilisant les composants implémentés au cours de toutes les séances de TP et en utilisant le schéma donné en haut de ce sujet, réalisez le processeur complet.