ARCHI – Architecture des ordinateurs

Sylvain Brandel
2022 – 2023
sylvain.brandel@univ-lyon1.fr

CM 11

ÉBAUCHE D'UN PROCESSEUR

Unité centrale

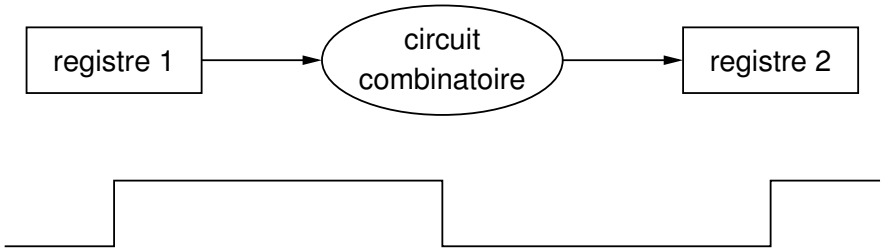
- Briques de base ?
  - Processeur :
    - UAL : calculs
    - Registres : pour calcul, banc avec contrôleur
  - Mémoire centrale : pour instructions et données, banc avec contrôleur

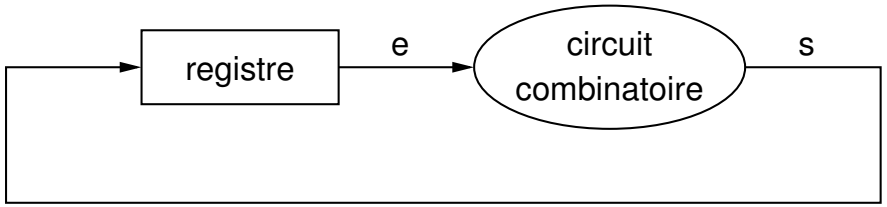
- Chemin de données
  - Circuits pour le traitement des instructions et des données
  - Pas contrôle

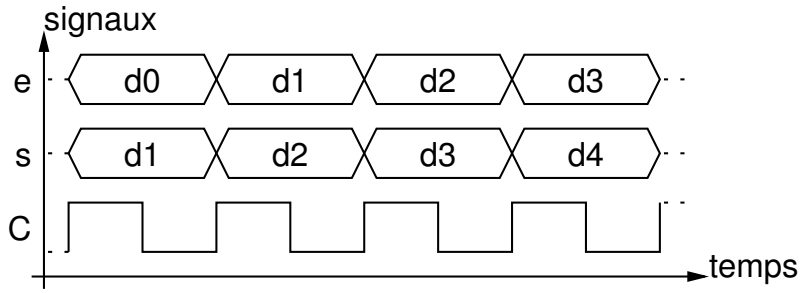
- Quel chemin ? (pour l'archi)
- Quel chef d'orchestre ?

Unité centrale

- Qui dit chef d'orchestre …
- Synchronisation — Pas de conflit
  - Circuits (bascules) synchrones au max
  - ICI transitions registres sur front descendant — Comment ?
- Stockage : registres
  - Mémo en fin de cycle + conservation cycle suivant
  - Circuit :
    - Entrées branchées sur sorties de registres
    - Sorties branchées sur entrées de registres — Évt le même



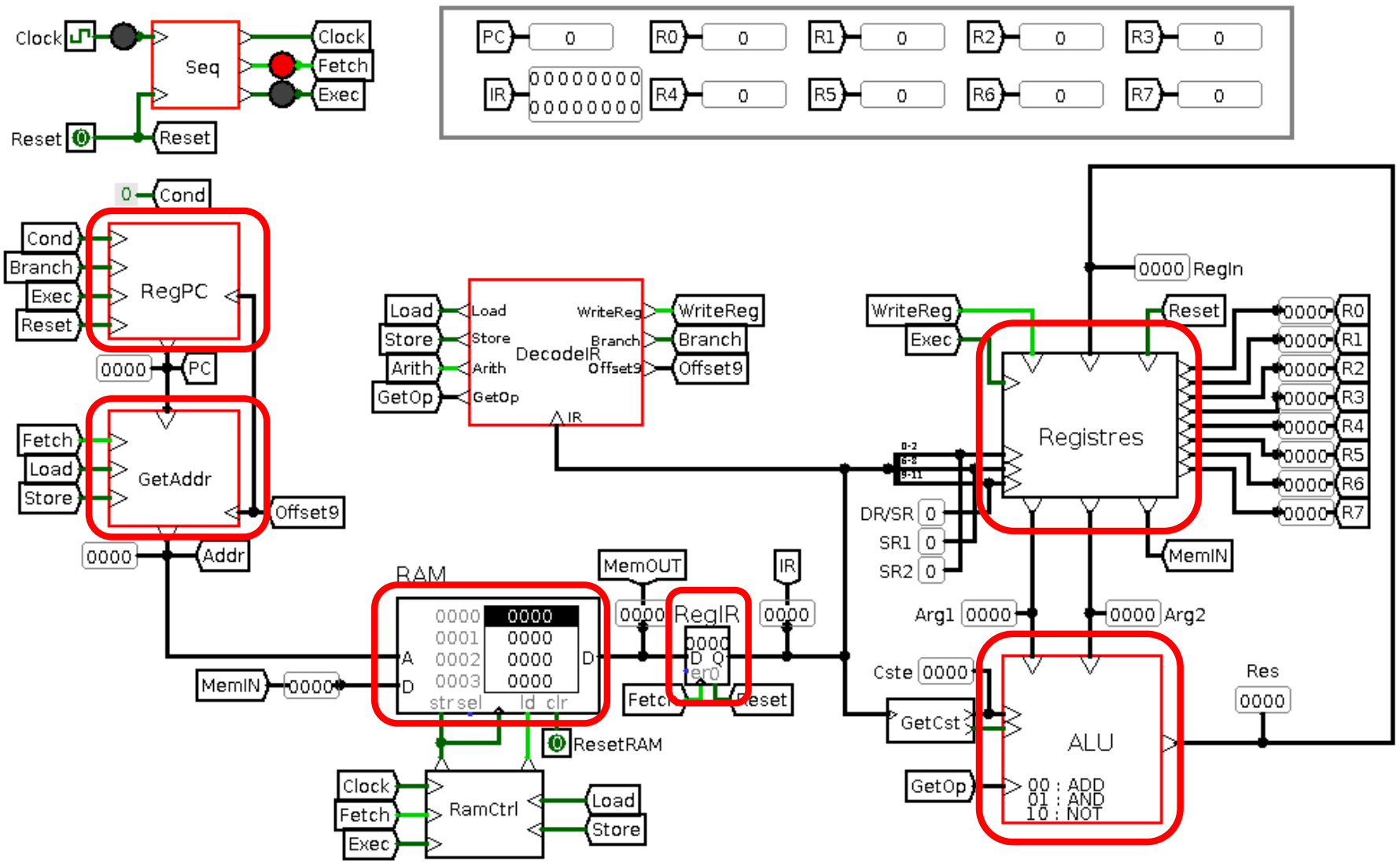




Unité centrale
Circuits

- De quoi à-t-on besoin ?
  - RAM : mémo + adressage déjà vu
  - UAL : exemple sélection par multiplexeur déjà vu
  - Registres généraux
  - Programme → registres spéciaux IR, PC

Unité centrale



Unité centrale

- Exemple

```
R0 <- 5
R1 <- 6
R2 <- R0 + R1
```

	Syntaxe	action	NZP	codage															
				opcode				arguments											
				F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Arith - logique	NOT DR, SR	DR <- not SR	*	1	0	0	1	DR			SR			1	1	1	1	1	1
	ADD DR, SR1, SR2	DR <- SR1 + SR2	*	0	0	0	1	DR			SR1			0	0	0	SR2		
	ADD DR, SR1, Imm5	DR <- SR1 + SEXT(Imm5)	*	0	0	0	1	DR			SR1			1	Imm5				
	AND DR, SR1, SR2	DR <- SR1 and SR2	*	0	1	0	1	DR			SR1			0	0	0	SR2		
	AND DR, SR1, Imm5	DR <- SR1 and SEXT(Imm5)	*	0	1	0	1	DR			SR1			1	Imm5				
charg. rang.	LEA DR,label	DR <- PC + SEXT(PCoffset9)	*	1	1	1	0	DR			PCoffset9								
	LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*	0	0	1	0	DR			PCoffset9								
	ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR		0	0	1	1	SR			PCoffset9								
	LDR DR,BaseR,Offset6	DR <- mem[BaseR + SEXT(Offset6)]	*	0	1	1	0	DR			BaseR			Offset6					
	STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] <- SR		0	1	1	1	SR			BaseR			Offset6					
branchement	BR[n][z][p] label Si (cond)	PC <- PC + SEXT(PCoffset9)		0	0	0	0	n	z	p	PCoffset9								
	~~NOP~~	~~No Operation~~		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	RET	PC <- R7		1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
	JSR label	R7 <- PC ; PC <- PC + SEXT(PCoffset11)		0	1	0	0	1	PCoffset11										

Unité centrale
Cycles

- Cycle instruction ≠ cycle horloge
  - Comp : opérandes lues dans registres, rangement résultat à la fin
  - Mem : liaison registres / mémoire, écriture à la fin
  - BR : calcul adresse et écriture dans PC à la fin

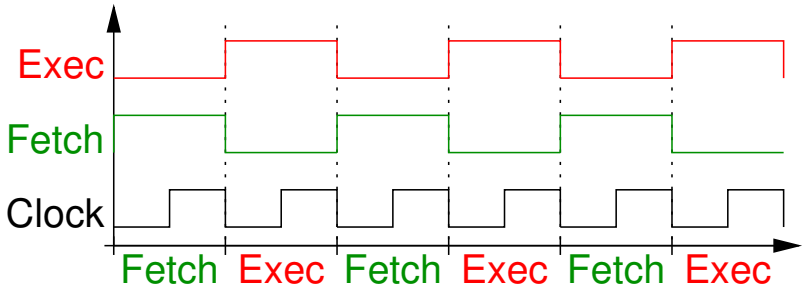
- MAIS instruction qq. part, ici dans RI bien câblé

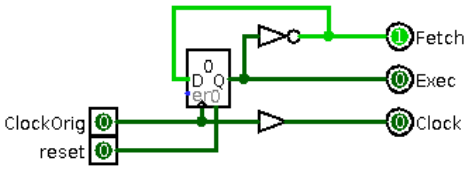
- Ici : Mémoire → IR puis exécution

- Cycle d'instruction :
  1. LI (Fetch)
  2. EX (Exec)

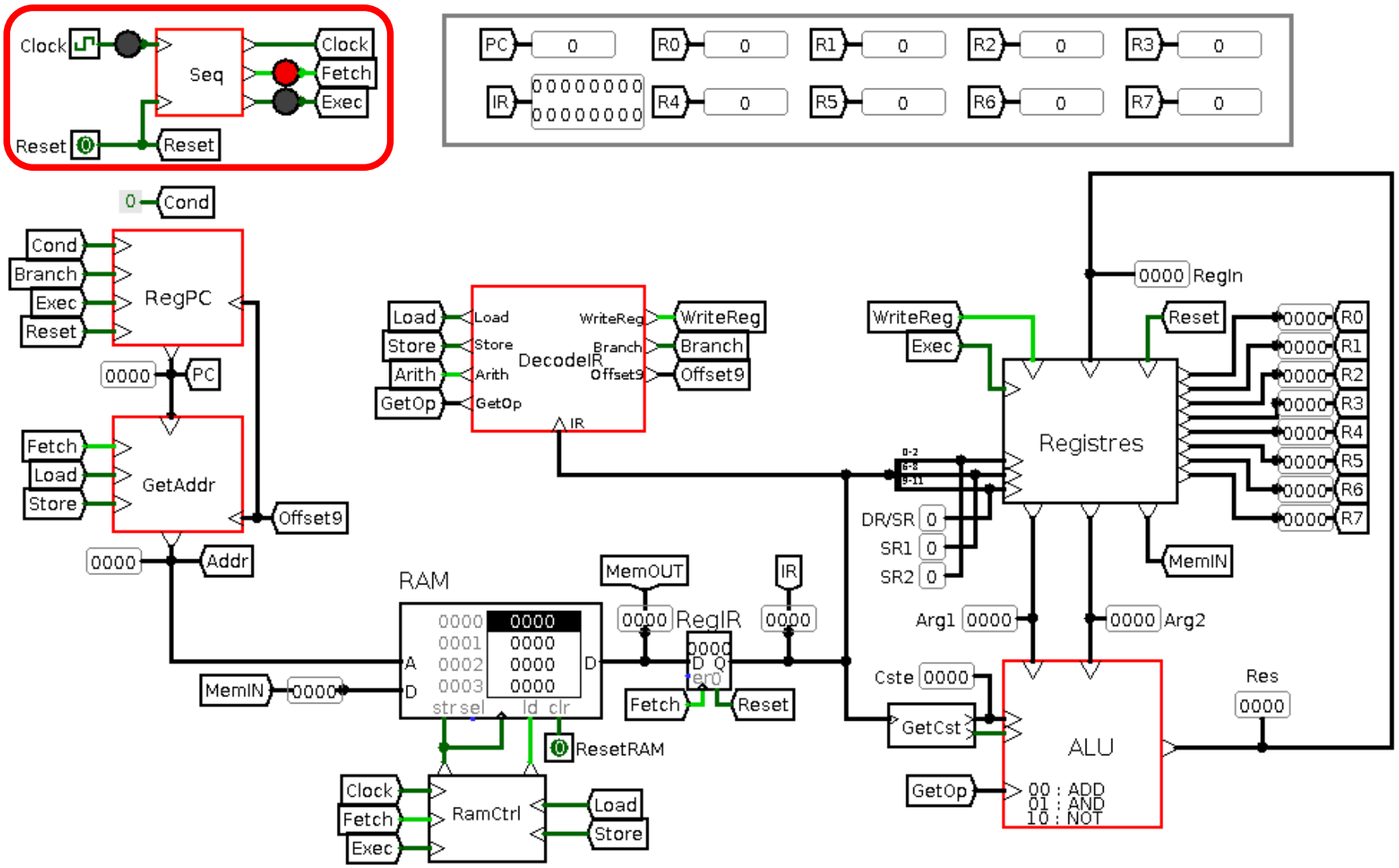
Unité centrale
Synchro

- Différentes activités suivant LI ou EX ⇝ séquenceur
  - in : Clk, (RAZ)
  - out : Clk, LI, EX





Unité centrale



Unité centrale

Synchro

	Syntaxe	action	NZP	codage															
				opcode				arguments											
				F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Arith - logique	NOT DR, SR	DR <- not SR	*	1	0	0	1	DR			SR			1	1	1	1	1	1
	ADD DR, SR1, SR2	DR <- SR1 + SR2	*	0	0	0	1	DR			SR1			0	0	0	SR2		
	ADD DR, SR1, Imm5	DR <- SR1 + SEXT(Imm5)	*	0	0	0	1	DR			SR1			1	Imm5				
	AND DR, SR1, SR2	DR <- SR1 and SR2	*	0	1	0	1	DR			SR1			0	0	0	SR2		
	AND DR, SR1, Imm5	DR <- SR1 and SEXT(Imm5)	*	0	1	0	1	DR			SR1			1	Imm5				
charg. rang.	LEA DR,label	DR <- PC + SEXT(PCoffset9)	*	1	1	1	0	DR			PCoffset9								
	LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*	0	0	1	0	DR			PCoffset9								
	ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR		0	0	1	1	SR			PCoffset9								
	LDR DR,BaseR,Offset6	DR <- mem[BaseR + SEXT(Offset6)]	*	0	1	1	0	DR			BaseR			Offset6					
	STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] <- SR		0	1	1	1	SR			BaseR			Offset6					
branchement	BR[n][z][p] label Si (cond)	PC <- PC + SEXT(PCoffset9)		0	0	0	0	n	z	p	PCoffset9								
	~~NOP~~	~~No Operation~~		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	RET	PC <- R7		1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
	JSR label	R7 <- PC ; PC <- PC + SEXT(PCoffset11)		0	1	0	0	1	PCoffset11										

- Qui touche aux registres ?
- Comment ?

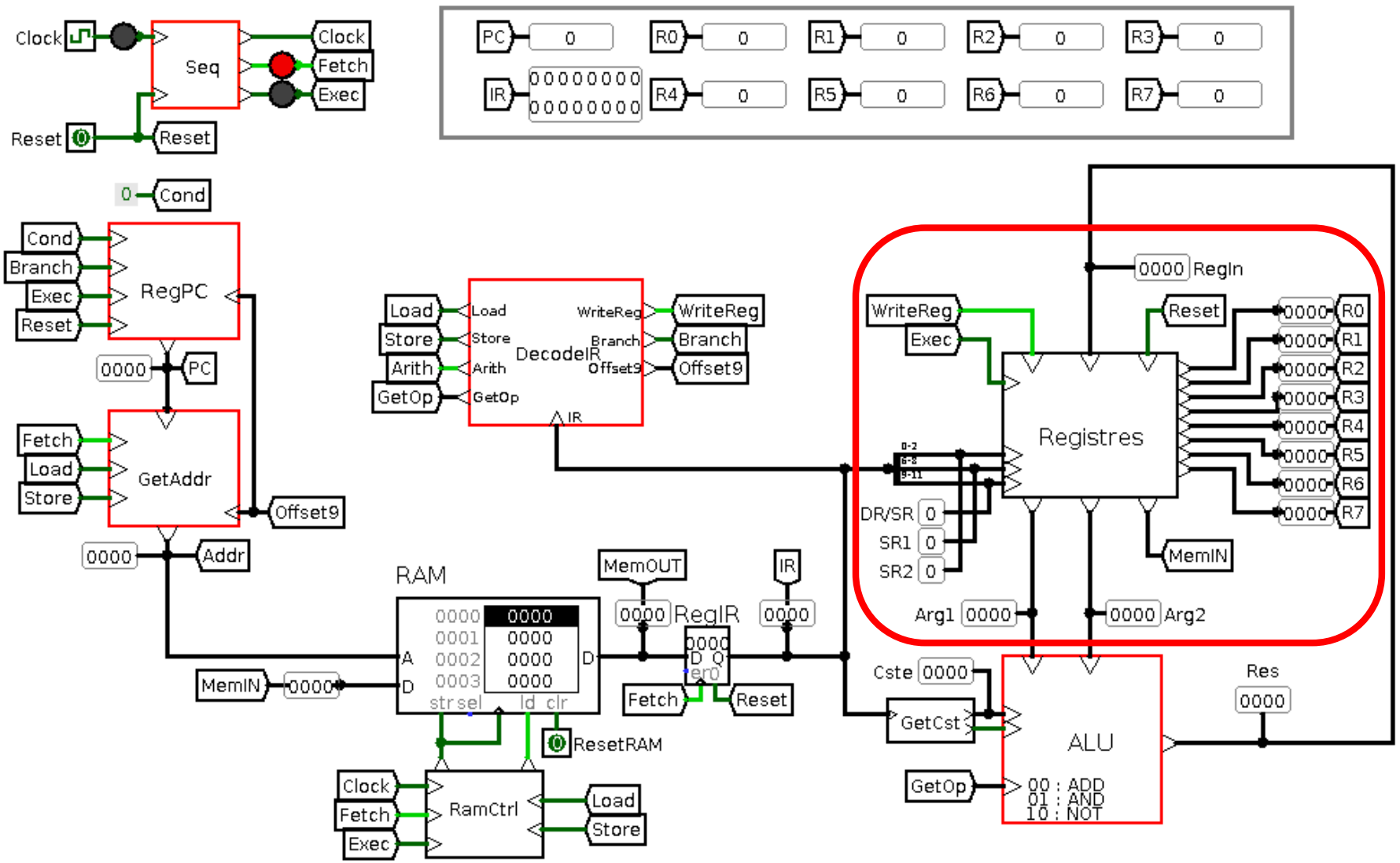
Unité centrale

Dialogue avec les registres

- Adressage pertinent
- Sorties pertinentes
- Entrées pertinentes
- Et signaux. . .

Unité centrale

Dialogue avec les registres



Unité centrale
Dialogue avec l'UAL

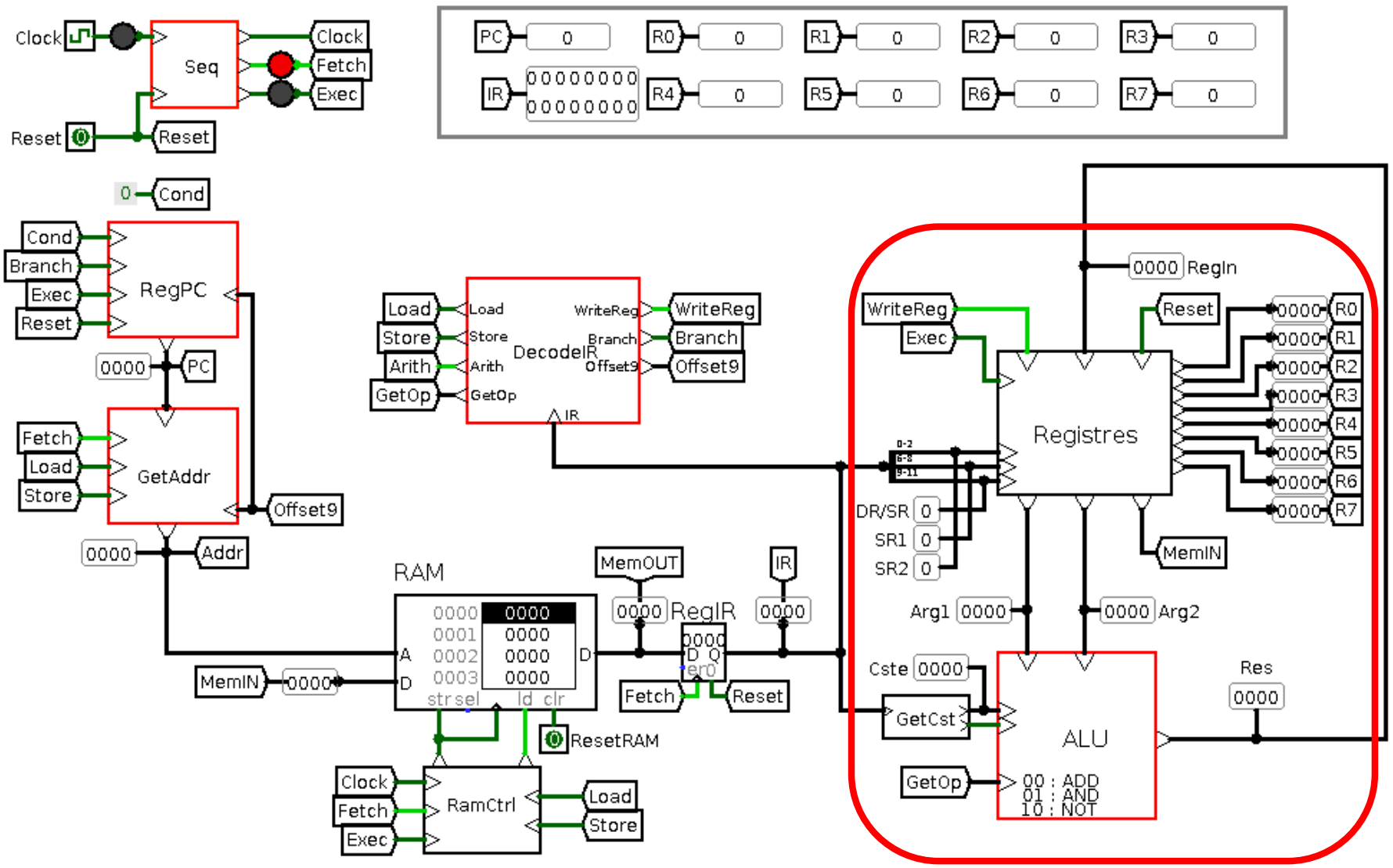
Arith - logique	NOT DR, SR	DR <- not SR	*	1	0	0	1	DR	SR	1	1	1	1	1	1
	ADD DR, SR1, SR2	DR <- SR1 + SR2	*	0	0	0	1	DR	SR1	0	0	0	SR2		
	ADD DR, SR1, Imm5	DR <- SR1 + SEXT(Imm5)	*	0	0	0	1	DR	SR1	1	Imm5				
	AND DR, SR1, SR2	DR <- SR1 and SR2	*	0	1	0	1	DR	SR1	0	0	0	SR2		
	AND DR, SR1, Imm5	DR <- SR1 and SEXT(Imm5)	*	0	1	0	1	DR	SR1	1	Imm5				

- Avec qui ?

- Quels argument ? Entrées / Sorties / Signaux de l'UAL ?
  - Dest, Source1, Source2
  - Dest, Source1, Immédiat
  - Dest, Source1

Unité centrale

Dialogue avec l'UAL



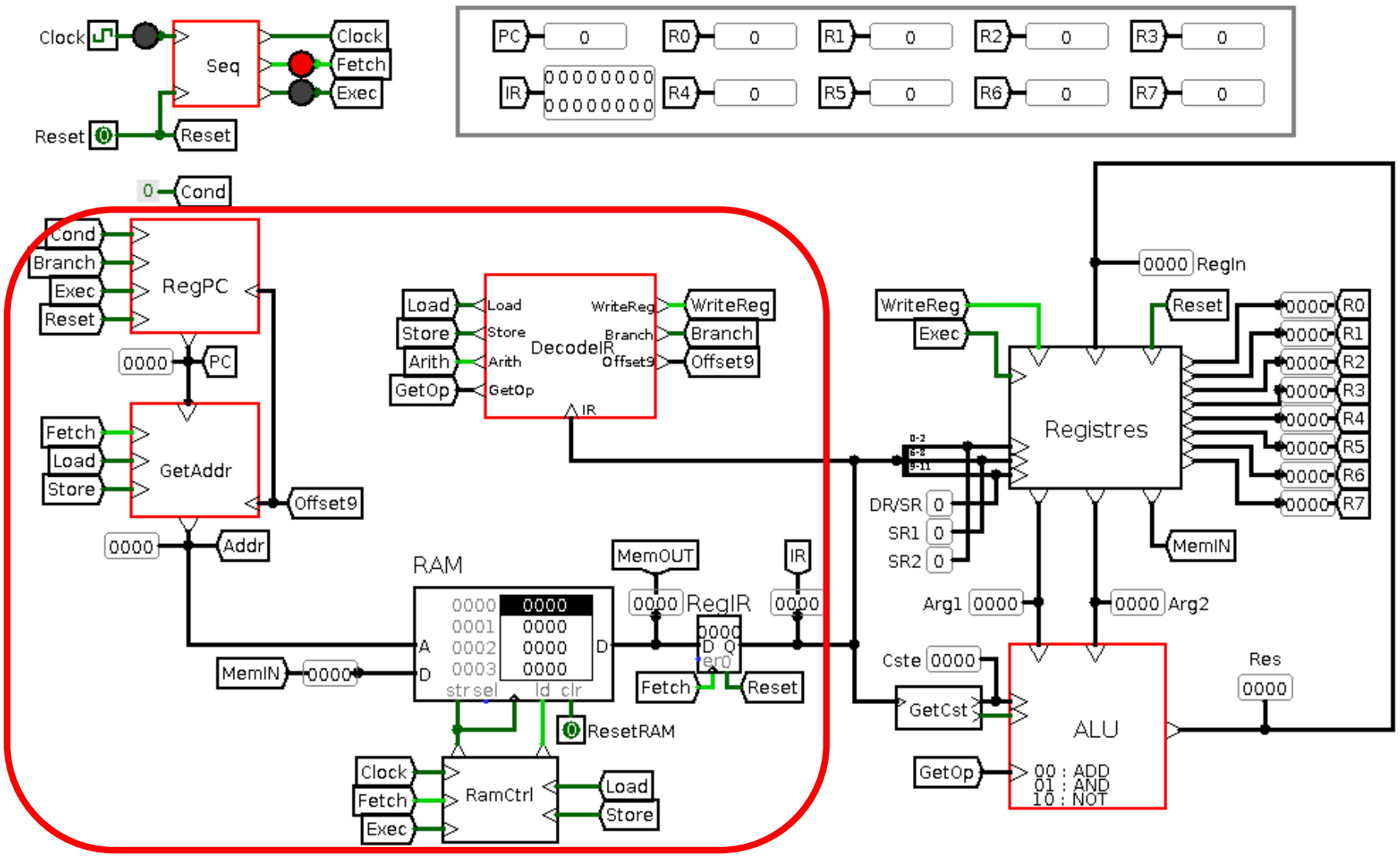
Unité centrale
Dialogue avec mémoire

LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*	0	0	1	0	DR	PCoffset9
ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR		0	0	1	1	SR	PCoffset9

- Avec qui ?
  - LI et EX
- Quand ?
  - PC + registres généraux. . .
- Signaux : Load/Store, Fetch, Exec, Clk
- Calculs ⇝ GetAddr PC ou PC+. . .

Unité centrale

Dialogue avec mémoire

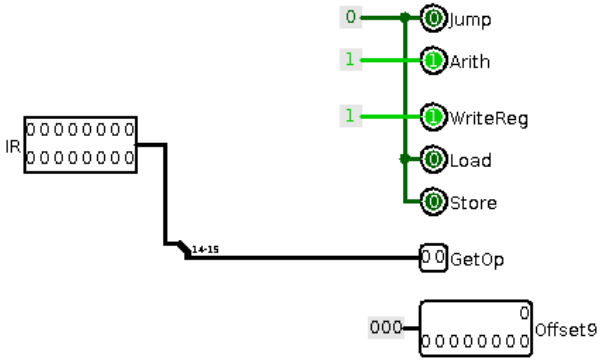


Unité centrale

Dialogue avec IR (décodage)

Syntaxe	action	NZP	codage															
			opcode				arguments											
			F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
NOT DR, SR	DR <- not SR	*	1	0	0	1	DR			SR			1	1	1	1	1	1
ADD DR, SR1, SR2	DR <- SR1 + SR2	*	0	0	0	1	DR			SR1			0	0	0	SR2		
ADD DR, SR1, Imm5	DR <- SR1 + SEXT(Imm5)	*	0	0	0	1	DR			SR1			1	Imm5				
AND DR, SR1, SR2	DR <- SR1 and SR2	*	0	1	0	1	DR			SR1			0	0	0	SR2		
AND DR, SR1, Imm5	DR <- SR1 and SEXT(Imm5)	*	0	1	0	1	DR			SR1			1	Imm5				
LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*	0	0	1	0	DR			PCoffset9								
ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR		0	0	1	1	SR			PCoffset9								
BR[n][z][p] label Si (cond)	PC <- PC + SEXT(PCoffset9)		0	0	0	0	n	z	p	PCoffset9								

- Instruction dans IR ⇝ Quels signaux pour quelle instruction ?
  - GetOp ?
  - Load ?
  - Store ?
  - WriteReg ?



Unité centrale
Organisation

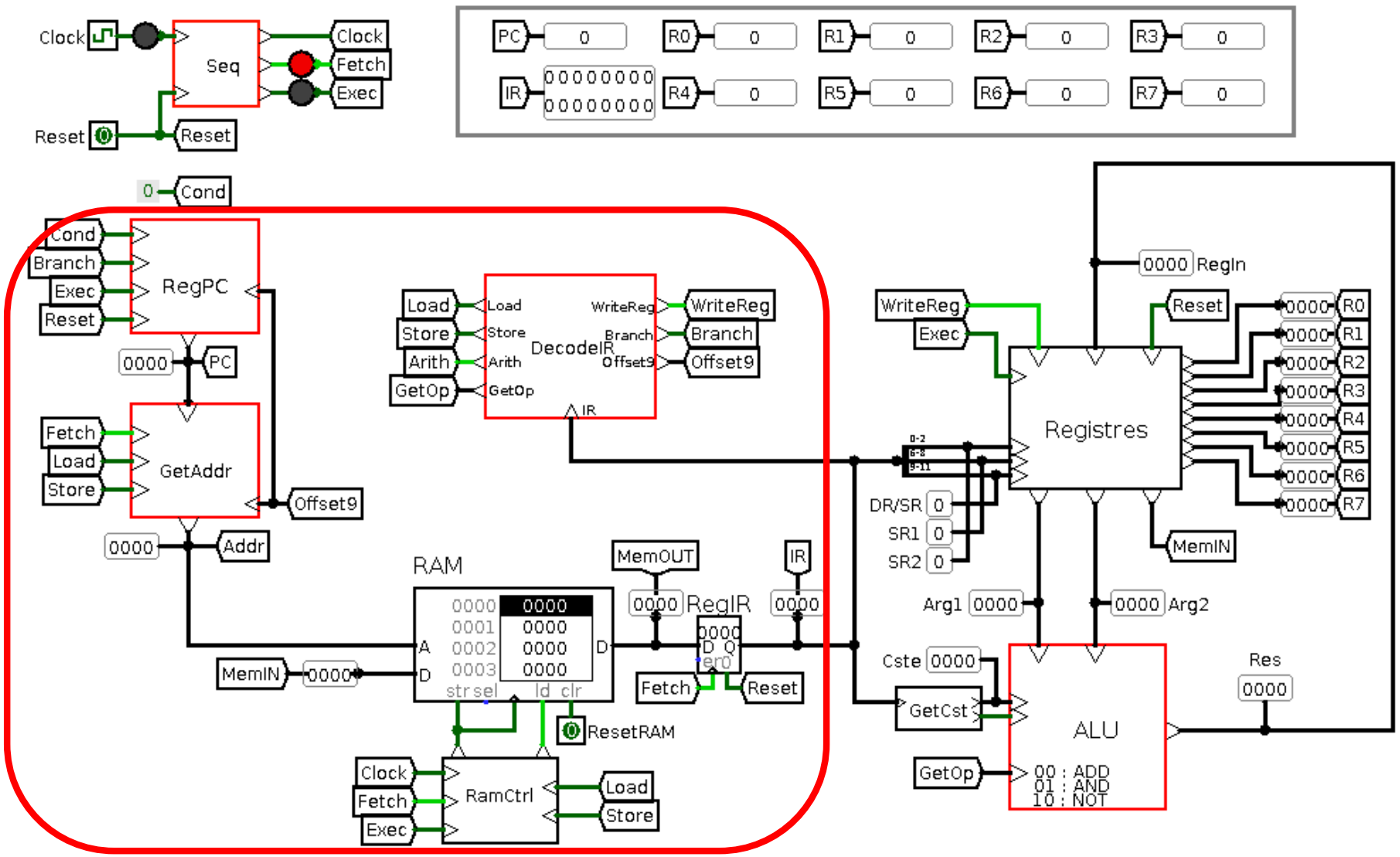
Phase LI

- Charger Mem[PC] dans IR
- Signal Fetch = 1 ⇝ mémoire mode « lecture »
- PC directement accessible
- IR reçoit MemOUT ⇝ écriture et verrou quand Fetch → 0

Unité centrale

Organisation

LI

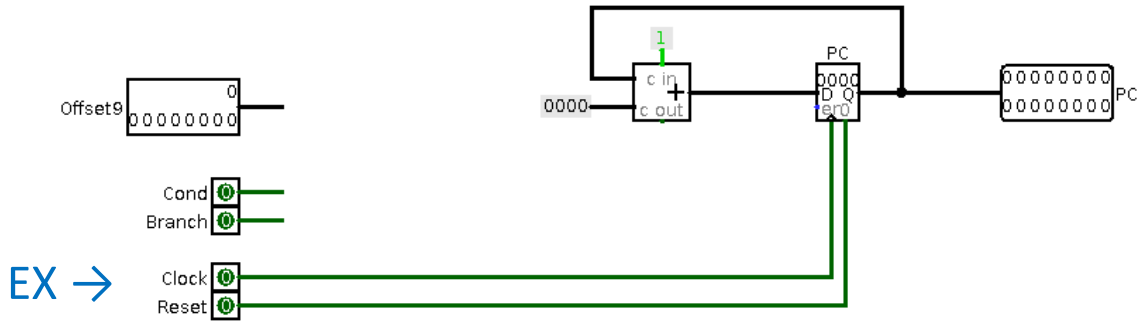


Unité centrale
Organisation

Phase EX

- Mise à jour PC

on sait faire



- Opération. . .
- Rangement résultat (à la fin)

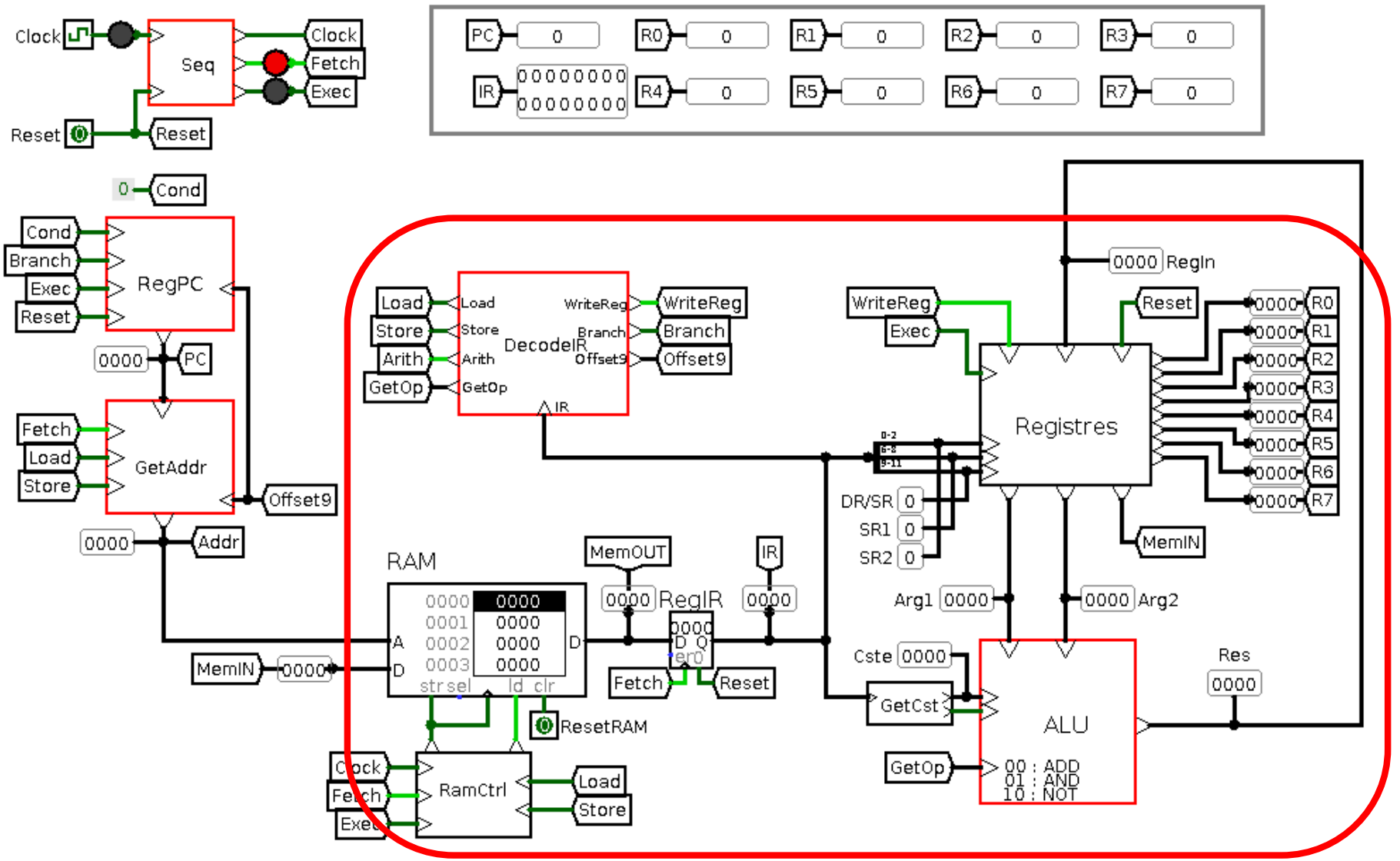
- Opération : calcul ou mémoire

Unité centrale

Organisation

EX calcul

Exemple : ADD DR, SR1, SR2



Unité centrale

Organisation

EX mémoire

Exemple : ST SR, label