Multicycle

Multi cycle

- exécution mono-cycle : cycle calculé sur l'instruction la plus longue
- exécution multi-cycle : les instructions ont des temps d'exécution différents
- dans la pratique les CPU ne sont pas monocycles
- Cherchons l'instruction la plus rapide à exécuter ainsi que l'instruction la plus longue à exécuter

Chemin critique

- Calculons le temps de cycle en supposant négligeable certains délais (mux, unité de contrôle, extension de signe, accès à PC, décalage de 2) :

  accès mémoire données et instruction (4ns)

  UAL et additionneurs (2 ns)

  R/W banc de registres (1 ns)

Instr.	I Mem	Reg Rd	ALU Op	D Mem	Reg Wr	Total
R-type						
load						
store						
beq						
jump						

Chemin critique

- Calculons le temps de cycle en supposant négligeable certains délais (mux, unité de contrôle, extension de signe, accès à PC, décalage de 2) :

  accès mémoire données et instruction (4ns)

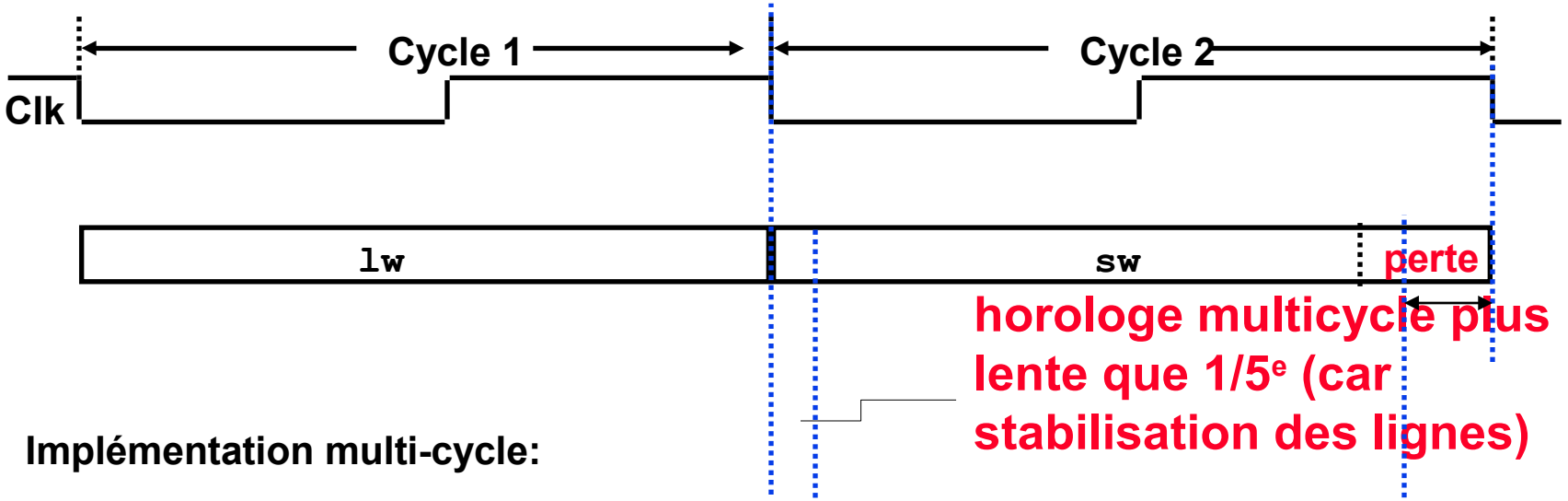
  UAL et additionneurs (2 ns)

  R/W banc de registres (1 ns)

Instr.	I Mem	Reg Rd	ALU Op	D Mem	Reg Wr	Total
R-type	4	1	2		1	8
load	4	1	2	4	1	12
store	4	1	2	4		11
beq	4	1	2			7
jump	4					4

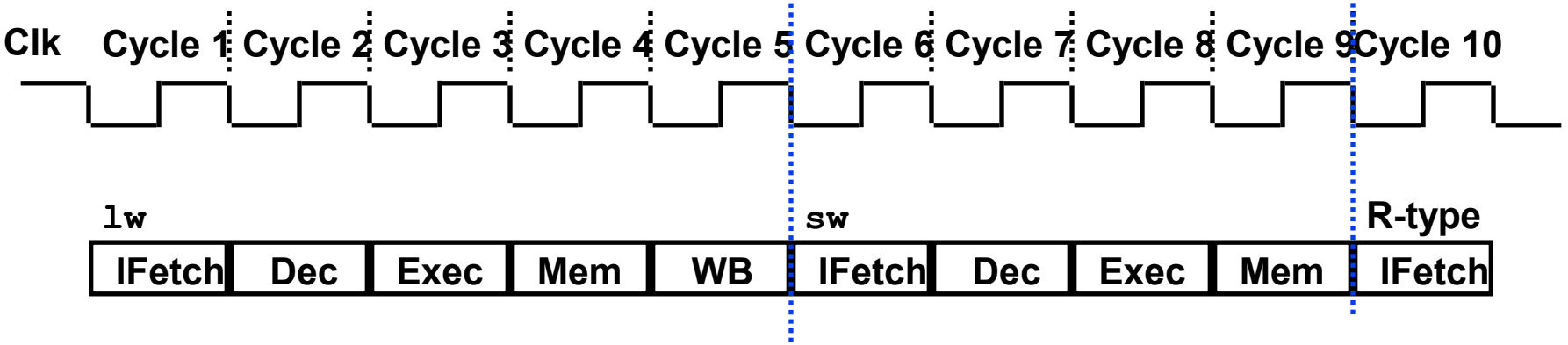
Mono-Cycle vs. Multi-Cycle

Implémentation mono-cycle:



horologe multicycle plus lente que 1/5e (car stabilisation des lignes)

Implémentation multi-cycle:



Multicycle

- Toutes les instructions ne prennent pas le même nombre de cycles

- Cycle plus rapide (fréquence plus élevée)

  découpage en travaux de temps équivalents

- Les unités (UAL, mux, extension de signe) peuvent être utilisées plusieurs fois par une même instruction

  ex : +4 pour PC peut être réalisé par l'UAL

Performance

- CPI : cycle par instruction
- $$speedup = \frac{CPI\ mono\ cycle}{CPI\ moyen\ multi\ cycle} \times \frac{tps\ clock\ mono\ cycle}{tps\ clock\ multi\ cycle}$$
- ex :

Instruction type	CPI	Freq	Freq * CPI
Load	5	0.25	1.25
Store	4	0.08	0.32
ALU	4	0.43	1.72
Branch taken	4	0.6 * 0.17	0.41
Branch not taken	3	0.4 * 0.17	0.20
Jump	2	0.07	0.14
Total			**4.04**

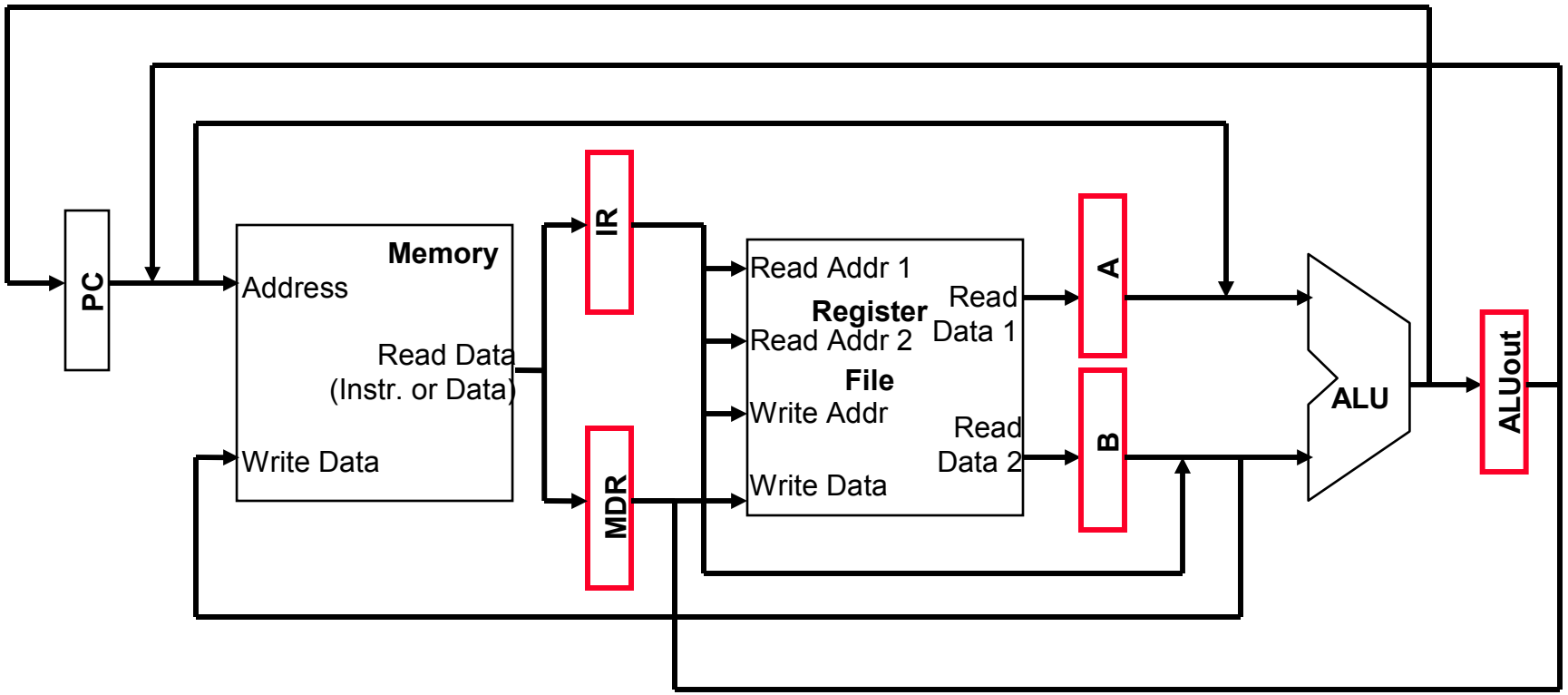
tps mono = 1 ns tps multi = 1.1 ns

speedup = (5/4.04) * (1/1.1) = 1.125

=> gain : 12,5%

Chemin de données multicycle (vue simplifiée)

- Des registres doivent séparer les différentes unités

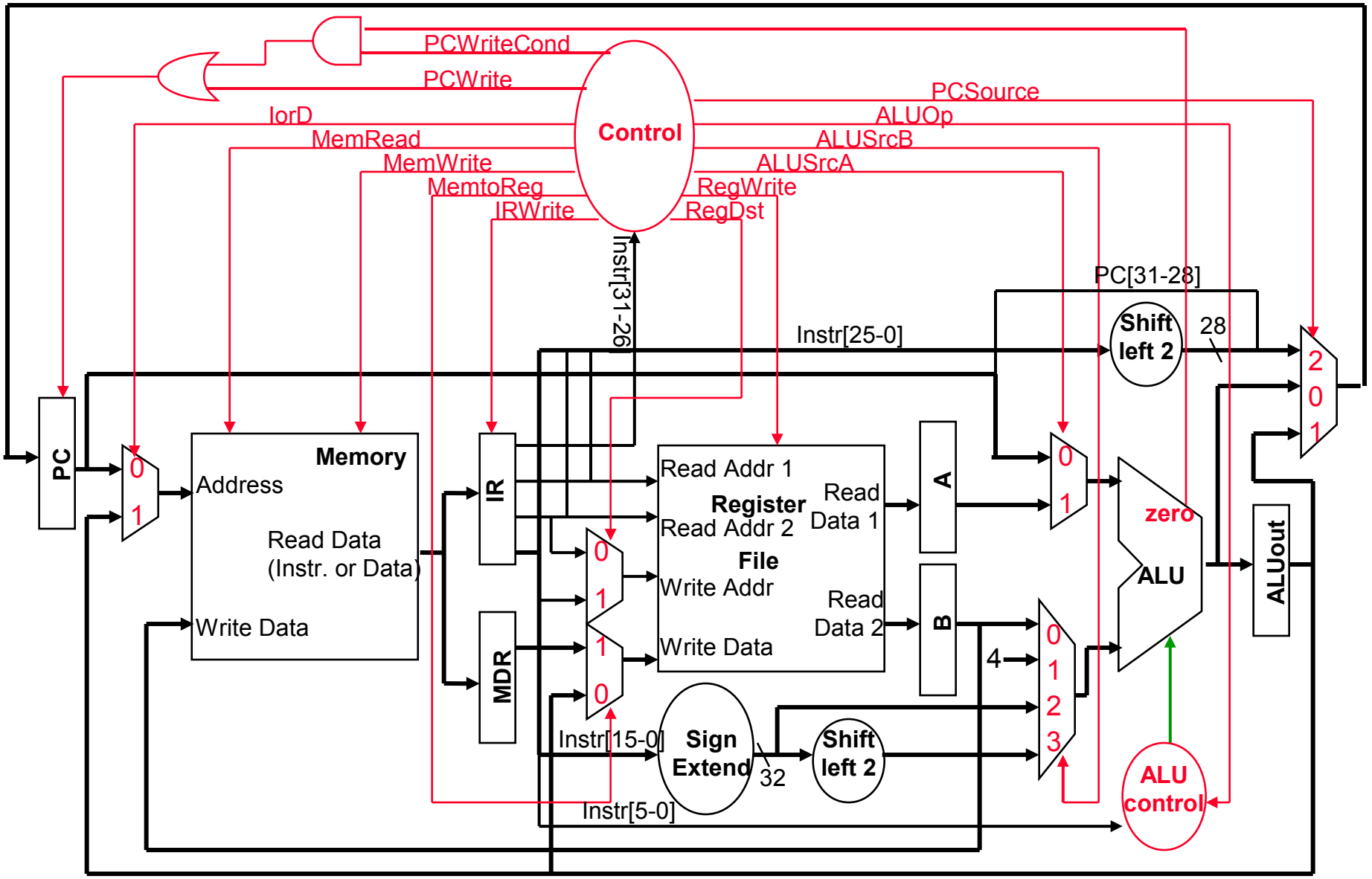


5 étapes

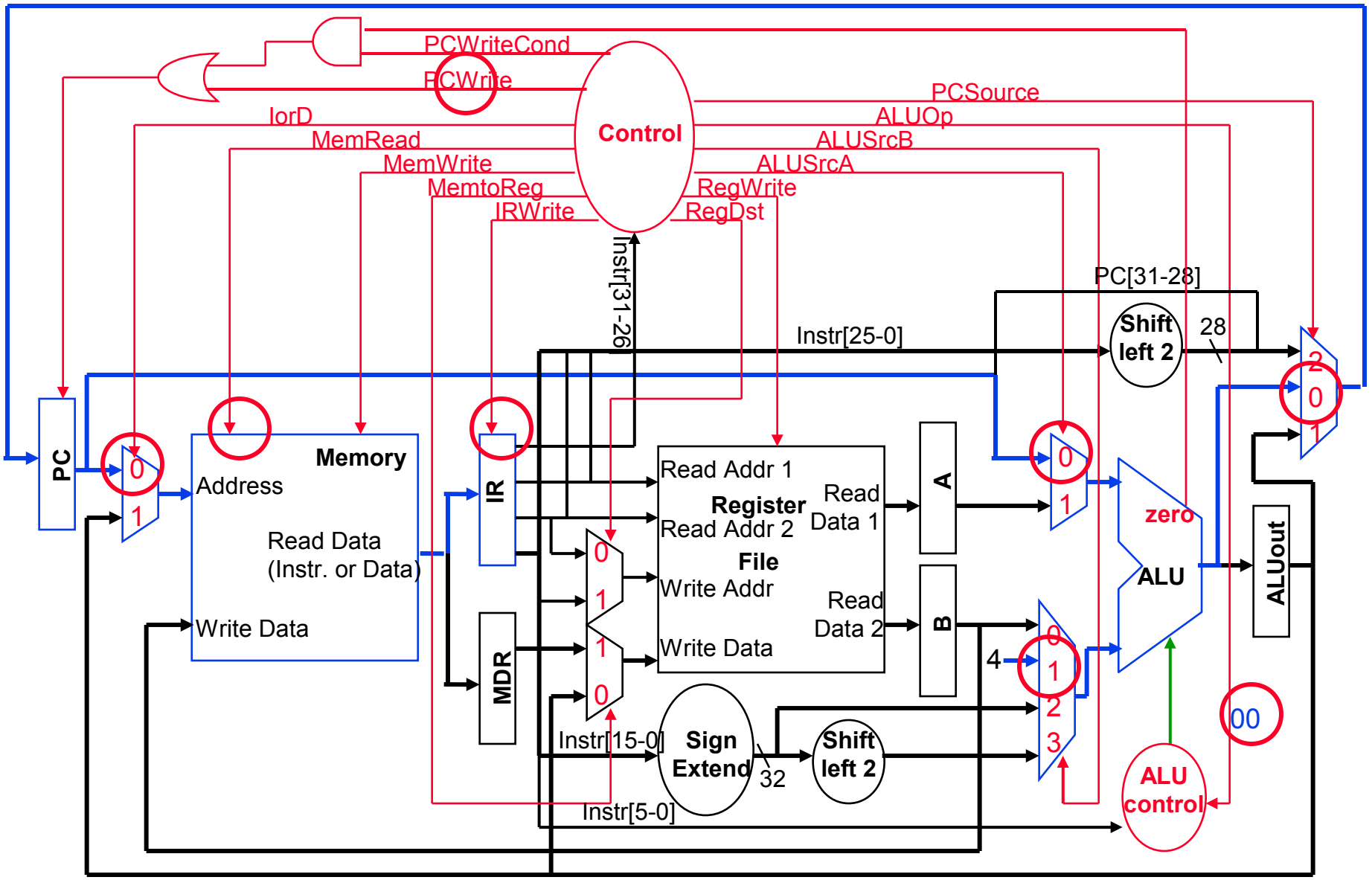
1. Fetch : recherche de l'instruction
2. Decode et lecture de registre
3. Exécution instruction de type R, calcul @ mémoire, branch ou jump complétion
4. Lecture ou écriture mémoire
5. Écriture banc de registres (Write Back)

Les instructions prennent entre 3 et 5 cycles

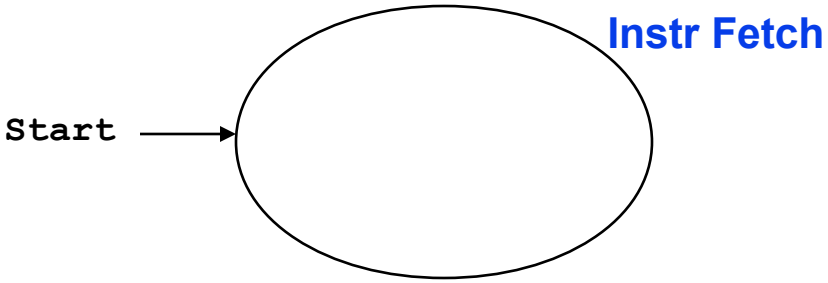
Chemin de données multicycle



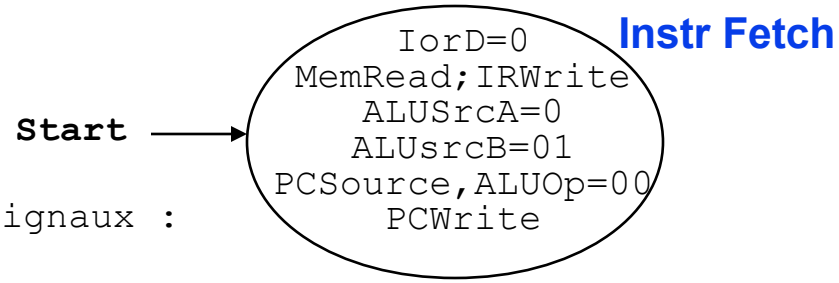
Signaux pour le cycle Fetch



Construction d'un automate à états



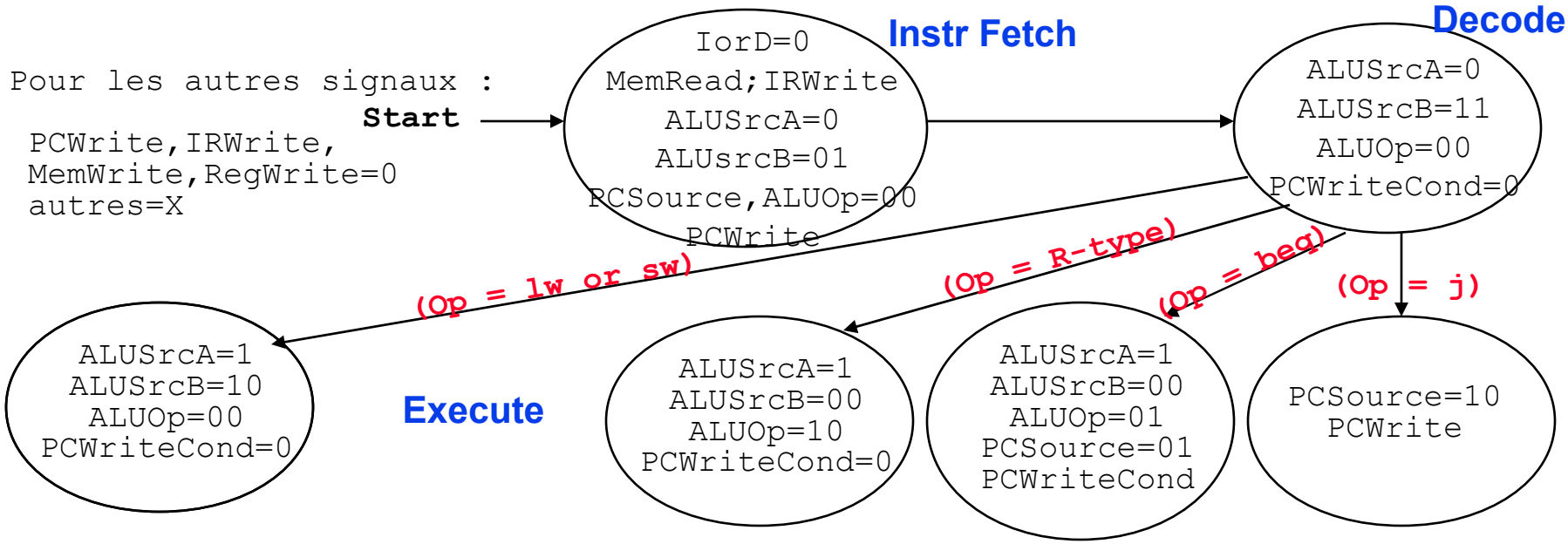
Construction d'un automate à états



Pour les autres signaux :

PCWrite,IRWrite,
MemWrite,RegWrite=0
autres=X

Construction d'un automate à états (suite)



Etc. (étape 4 et 5)

Mise en oeuvre de l'automate

- circuit combinatoire (avec signaux "état" en entrée et sortie)

- Mémoire de micro-programme (=> extension du jeu d'instruction plus facile)

- RISC (Reduced Instruction Set Computer) : jeu d'instruction réduit => circuit
- CISC (Complex Instruction Set Computer) : jeu d'instruction important => mémoire micro-programme
- actuellement : mélange des deux techniques