RISC / Super Scalaire

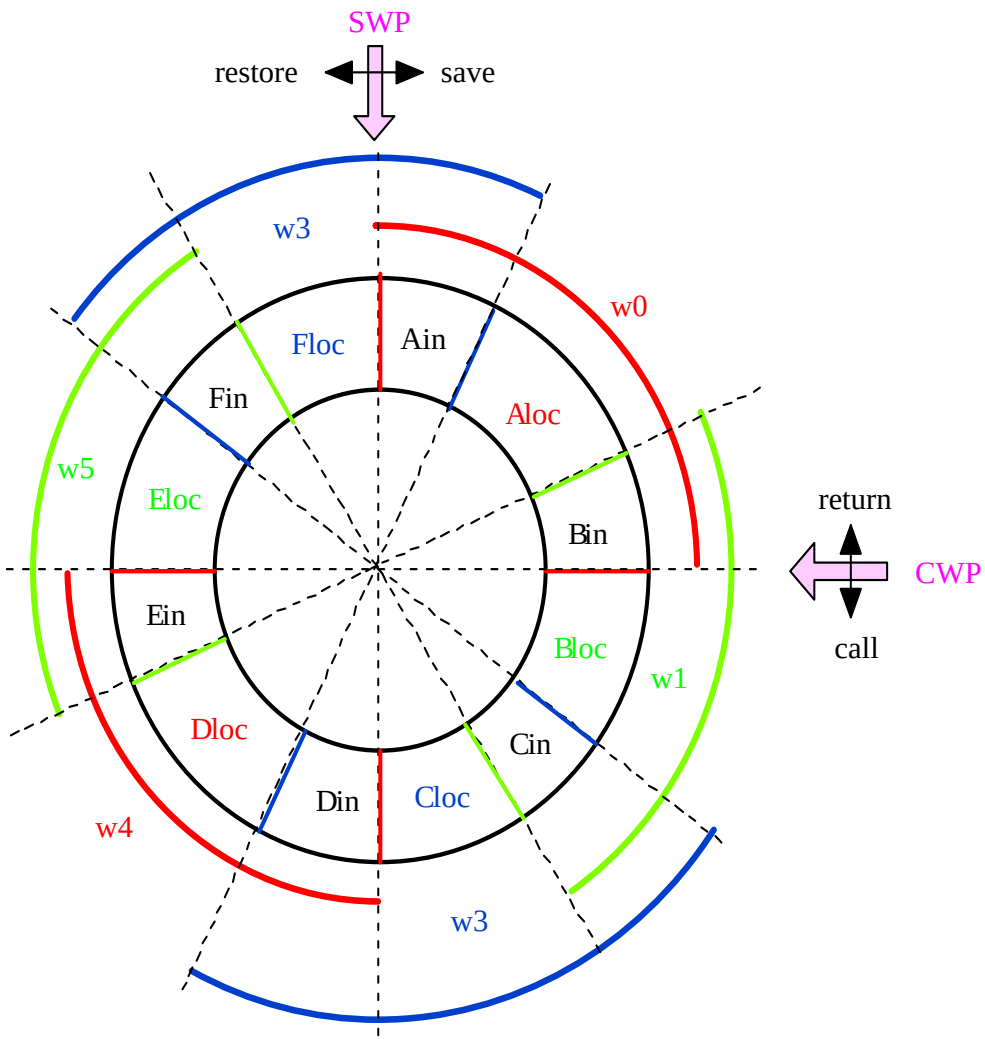
RISC

- RISC= Reduced Instruction Set Computer
- Historique
- Caractéristiques
  - Jeu d'instructions réduit
  - Une instruction par cycle
  - Opérations registre/registre, chargement, rangement
  - Modes d'adressage simples
  - Formats d'instructions simples
  - Nombre de registres élevé

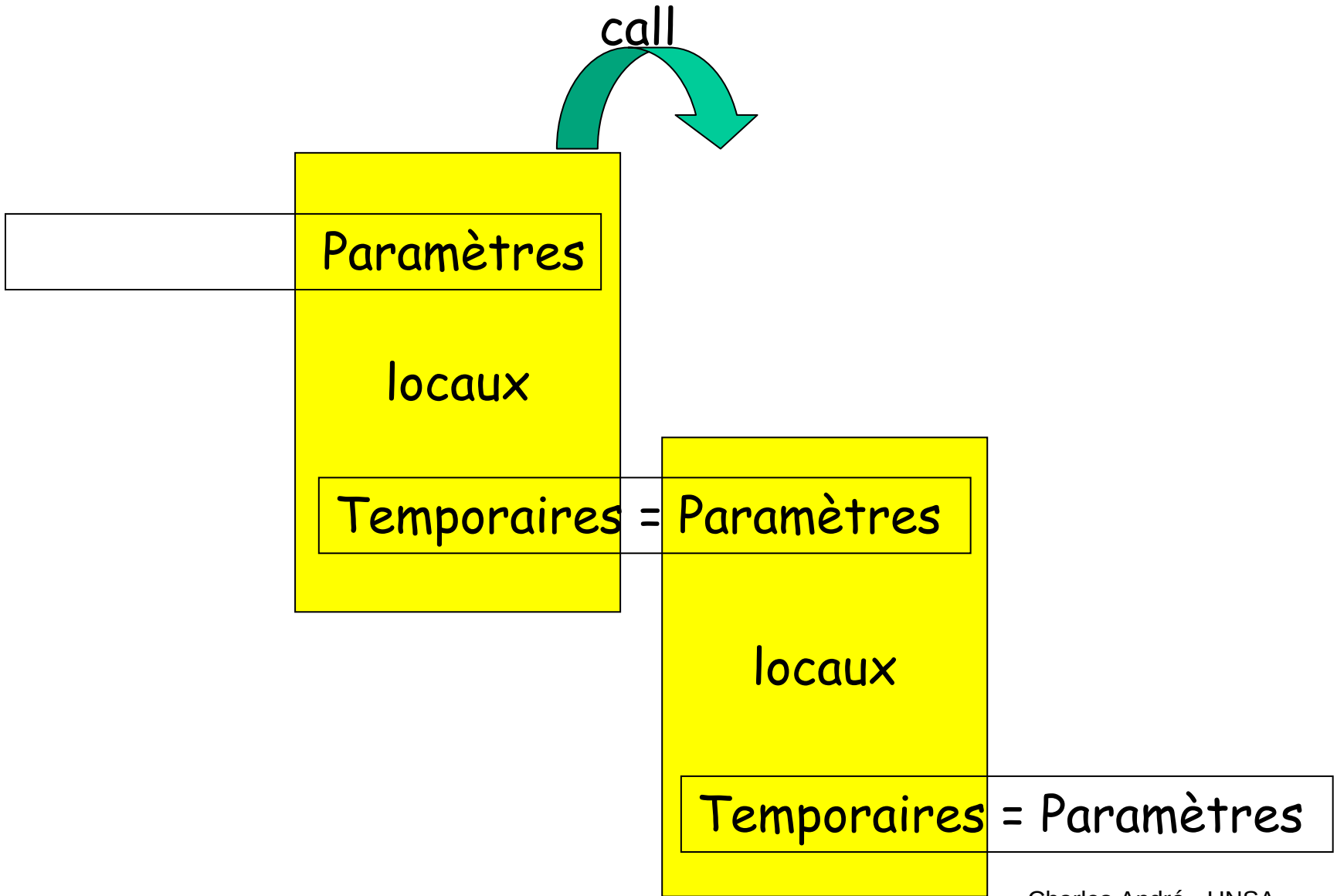
File de registres

- Registres = mémoires rapides
- Fenêtre de registres
  - Registres de paramètres
  - Registres locaux
  - Registres temporaires
- Appels de sous-programmes
  - CWP = Current Window Pointer
  - SWP = Saved Window Pointer

File de registres avec fenêtre



Changement de contexte



File de registres / Cache

	File de registres	cache
Variables locales	Toutes (ou presque)	Les plus récemment référencées
Accès aux variables globales	individuels	Par blocs
Sauvegarde restauration	Appels imbriqués, décalage de fenêtre	Basé sur l'algorithme de remplacement
Mode adressage	registre	mémoire

Risc et pipeline

- Instructions reg/reg:
  - I (Instruction Fetch)
  - E (Execute)
- Instructions chargement/rangement
  - I (Instruction Fetch)
  - E (Execute)
  - D (Memory)

Pipe-line simple (sol 1)

(a)load A, M1
(b)load B, M2
(c)add C, A, B
(d)store M3, C
(e)branch X
(f)no-op

1	2	3	4	5	6	7	8	9	10
Ia	Ea	Da							
	Ib		Eb	Db					
			Ic		Ec				
					Id	Ed	Dd		
						Ie		Ee	
								If	Ef

Pipe-line simple (sol 2)

(a) load A, M1
(b) load B, M2
(c) no-op
(d) add C, A, B
(e) store M3, C
(f) branch X
(g) no-op

1	2	3	4	5	6	7	8	9	10
Ia	Ea	Da							
	Ib	Eb	Db						
		Ic	Ec						
			Id	Ed					
				Ie	Ee	De			
					If	Ef			

Pipe-line (sol 3)

(a) load A, M1
(b) load B, M2
(c) no-op
(d) add C, A, B
(e) store M3, C
(f) branch X
(g) no-op

E1 = lecture de la file de registres

E2 = opération ALU et écriture registres

1	2	3	4	5	6	7	8	9	10
Ia	E1a	E2a	Da						
	Ib	E1b	E2b	Db					
		Ic	E1c	E2c					
			Id	E1d	E2d				
				Ie	E1e	E2e	De		
					If	E1f	E2f		

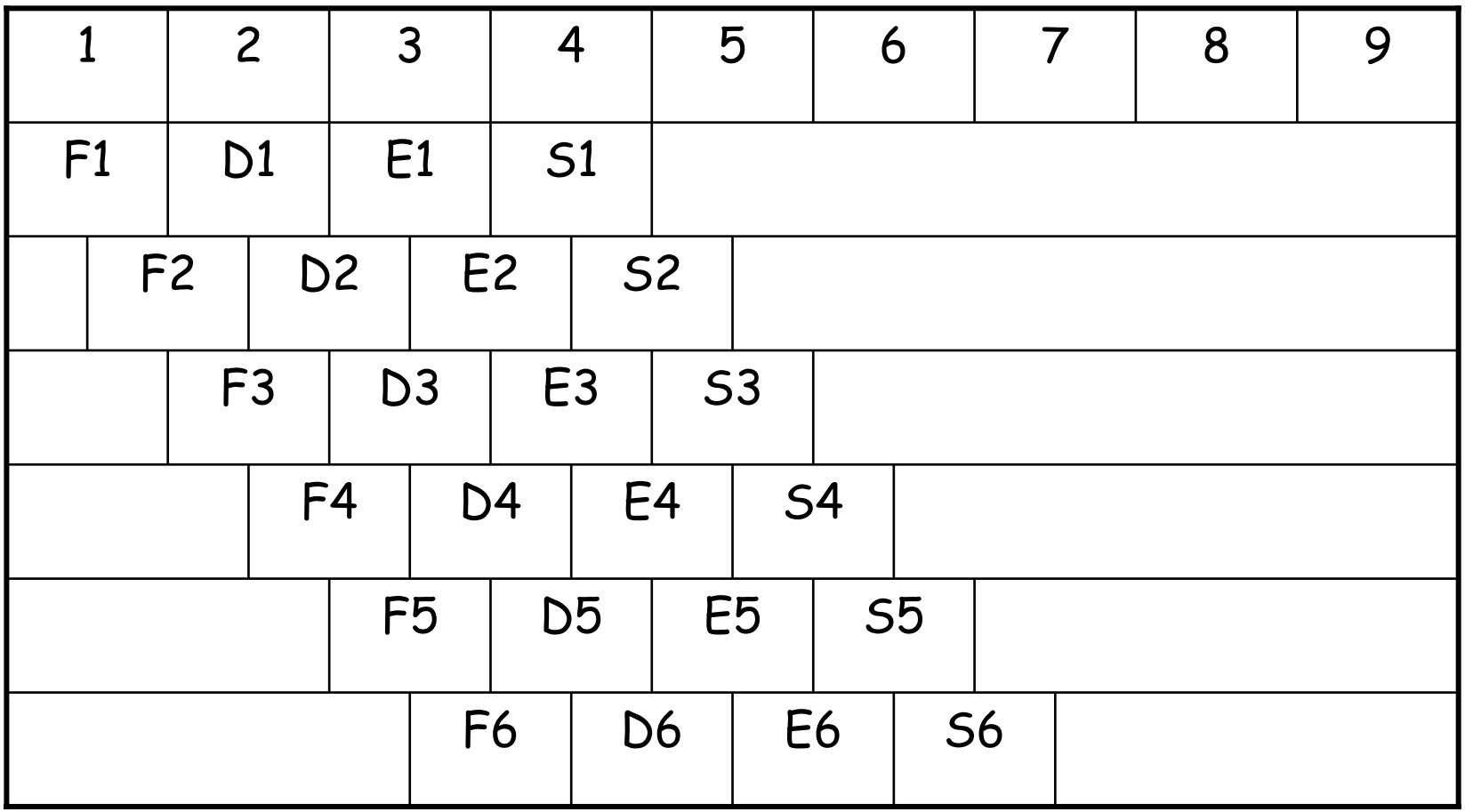
Super Scalaires

Pipe-line simple

1	2	3	4	5	6	7	8	9
F1	D1	E1	S1					
	F2	D2	E2	S2				
		F3	D3	E3	S3			
			F4	D4	E4	S4		
				F5	D5	E5	S5	
					F6	D6	E6	S6

Super Scalaires

Super Pipe-line



Super Scalaires

Super scalaire

1	2	3	4	5	6	7	8	9
F1	D1	E1	S1					
F2	D2	E2	S2					
	F3	D3	E3	S3				
	F4	D4	E4	S4				
		F5	D5	E5	S5			
		F6	D6	E6	S6			

Problèmes de conception

- Parallélisme instruction / parallélisme machine

Supposons parallélisme machine = 3

```
load R1, 23(R2)
addi R3, R3, 1
add  R4, R4, R2
```

```
addi R3, R3, 1
add  R4, R3, R2
store  (R4), R0
```

Parallélisme instruction = 3

Parallélisme instruction = 1

Un exemple

- 6 instructions I1, …, I6 à exécuter « au mieux »
- I1 a besoin de 2 cycles
- I4 produit pour I5
- I3 et I4 sont en conflit
- I5 et I6 sont en conflit

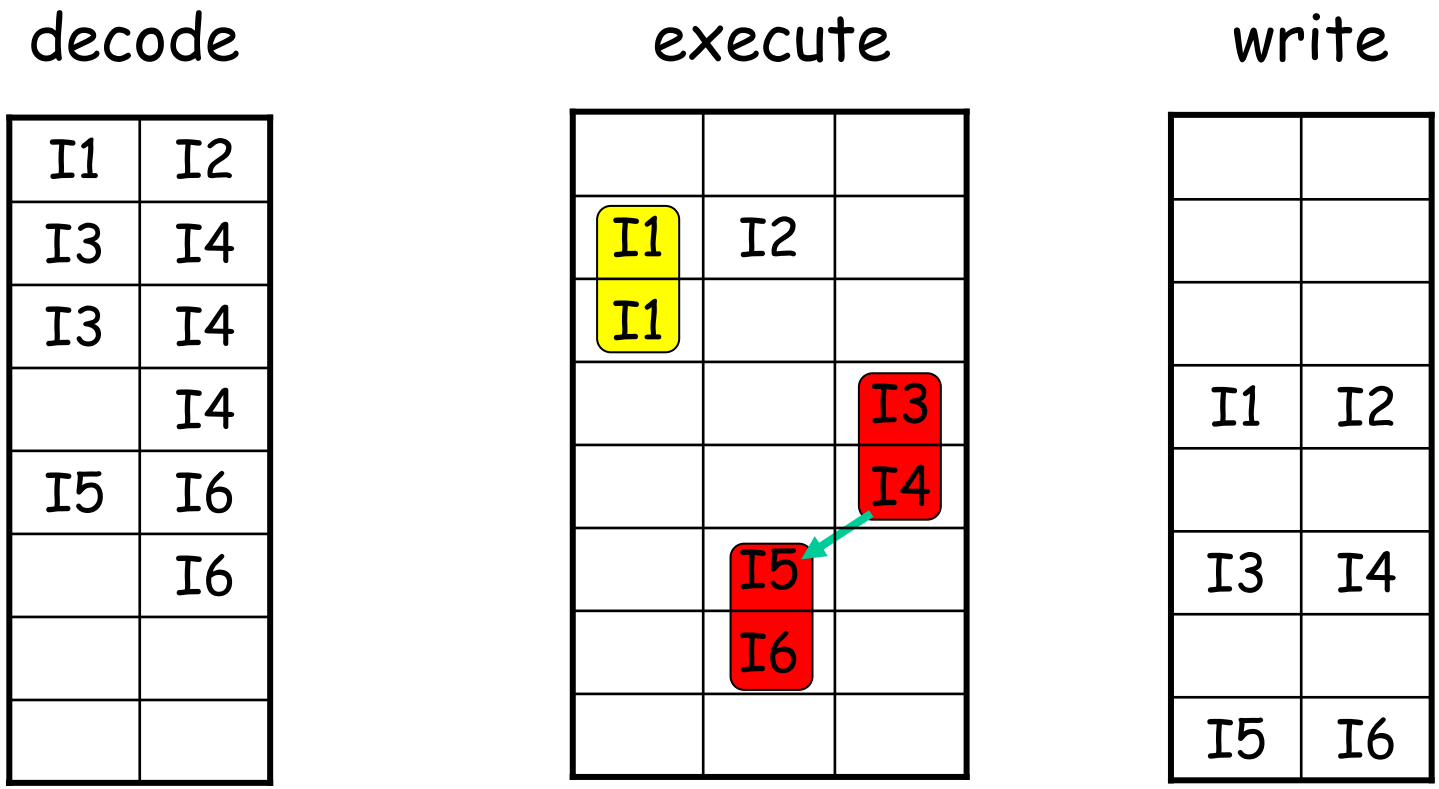
durée

précédence

Utilisent la même unité fonctionnelle

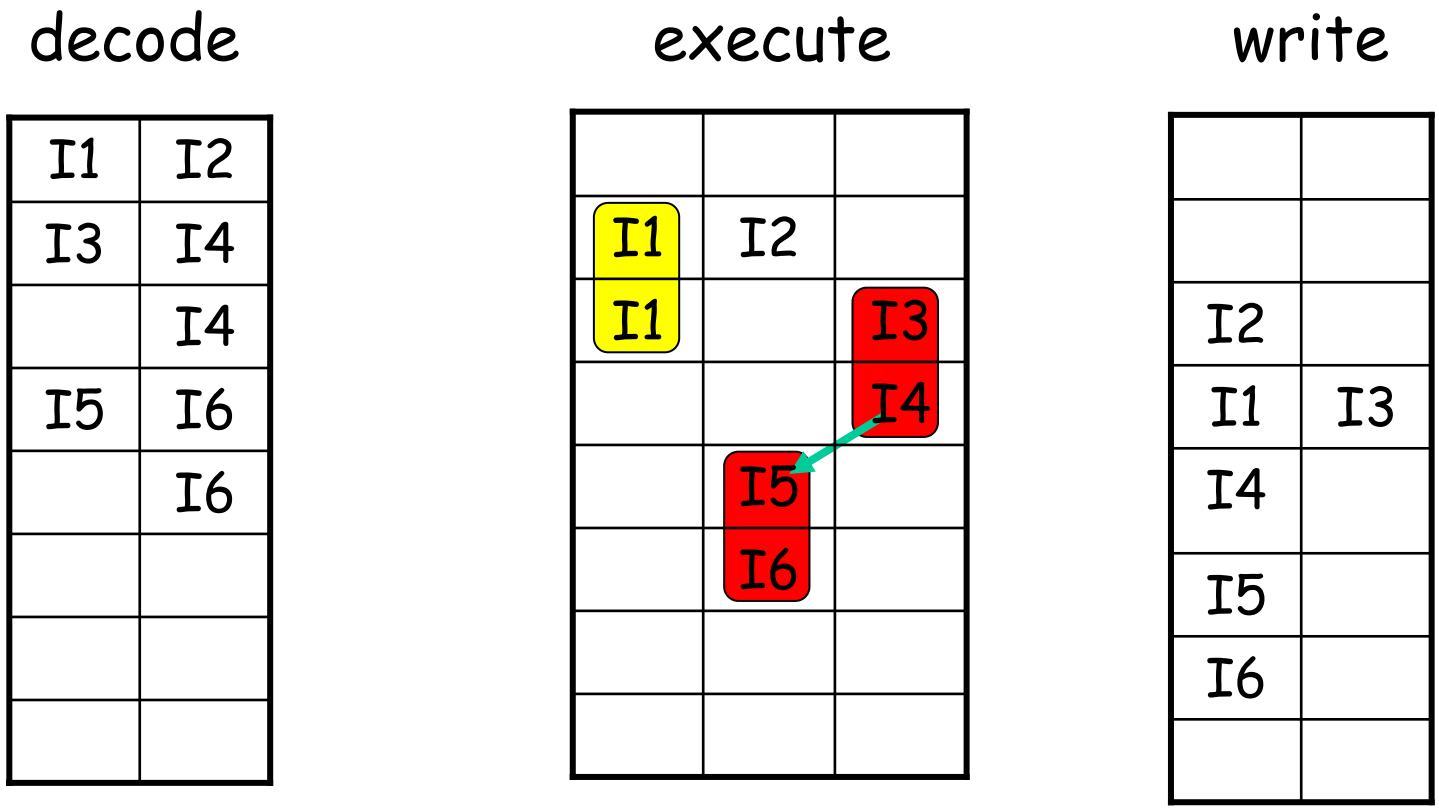
Ordre de traitement

- Lancement dans l'ordre, terminaison dans l'ordre



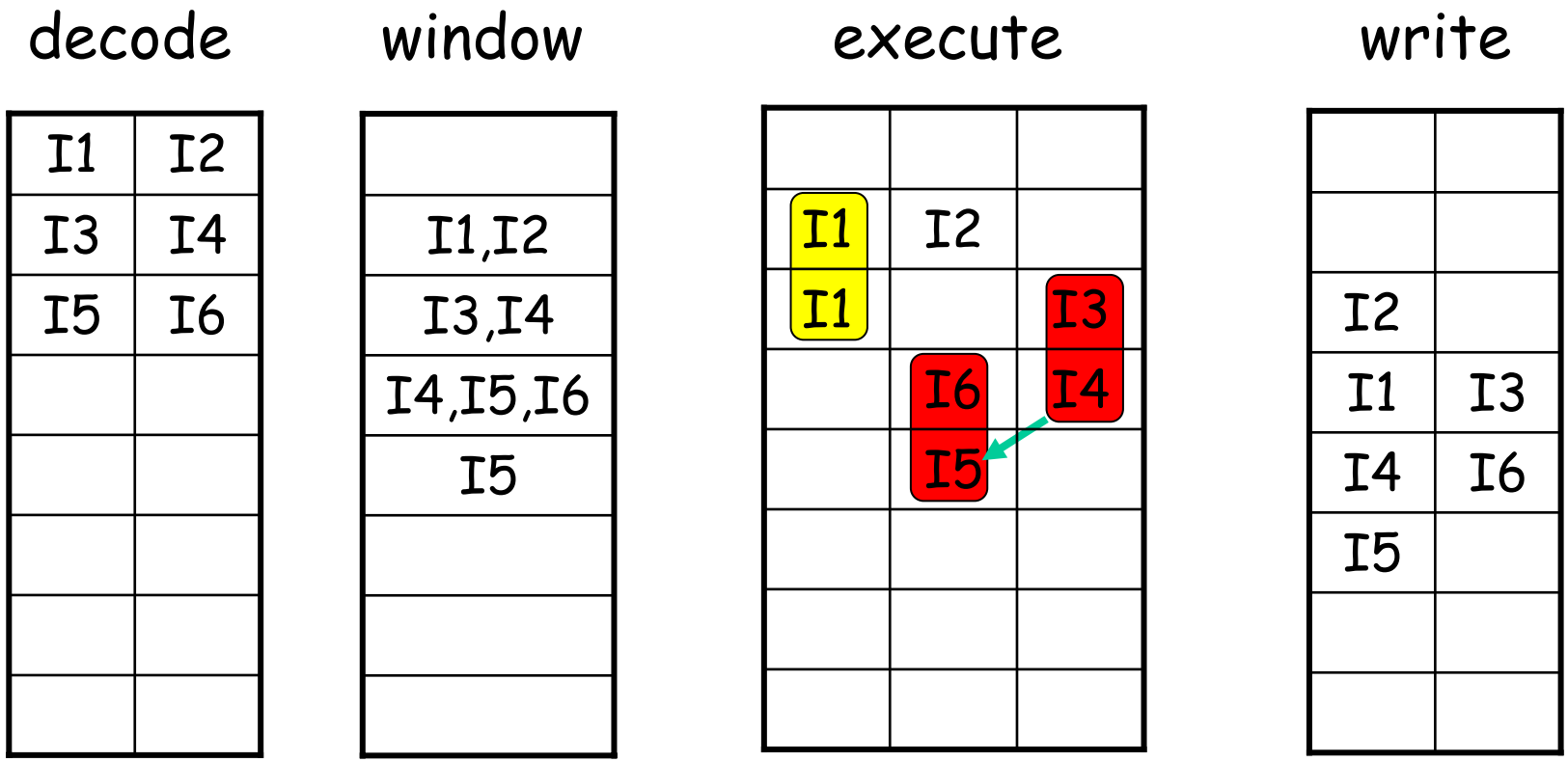
Ordre de traitement

- Lancement dans l'ordre, terminaison dans le désordre



Ordre de traitement

- Lancement dans le désordre, terminaison dans le désordre



Renommage de registres

```
add  R3, R3, R5
addi R4, R3, 1
addi R3, R5, 1
add  R7, R3, R4
```

```
add  R3b, R3a, R5a
addi R4b, R3b, 1
addi R3c, R5a, 1
add  R7b, R3c, R4b
```

Cas du pentium