Architecture des systèmes à processeur

Prof. Géraldine Conti

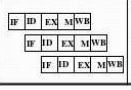
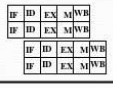
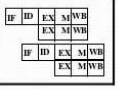
Basé sur les cours des Profs. Sanchez, Starkier, Mosqueron et Dassatti

1

Types de parallélisme

- **Bit-level** parallelism
  - Augmenter la taille des mots
  - 64 bits (mais DDR2 de 256 bits)
- **Instruction level** parallelism
  - Pipeline
  - Superscalar
  - VLIW
- **Data** parallelism
  - Traitement vectoriel, images, matrices
- **Task (thread)** parallelism
  - Simultaneous Multithreading

CISC vs RISC vs SS vs VLIW

	CISC	RISC	Superscalar	VLIW
Instruction size	variable size	fixed size	fixed size	fixed size (but large)
Instruction format	variable format	fixed format	fixed format	fixed format
Registers	few, some special	many GP	GP and rename (RUU)	many, many GP
Memory reference	embedded in many instr's	load/store	load/store	load/store
Key Issues	decode complexity	data forwarding, hazards	hardware dependency resolution	code scheduling, (compiler)
Instruction flow				

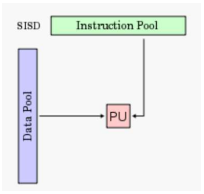
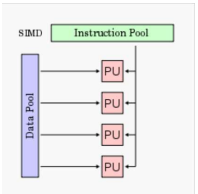
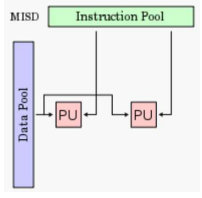
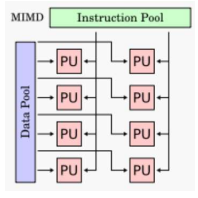
2

Taxonomie de Flynn (1966)

	Single Data	**Multiple Data**
Single Instruction	SISD	SIMD
Multiple Instruction	MISD	MIMD

3

Taxonomie de Flynn (1966)

	Single Data	**Multiple Data**
Single Instruction		
Multiple Instruction		

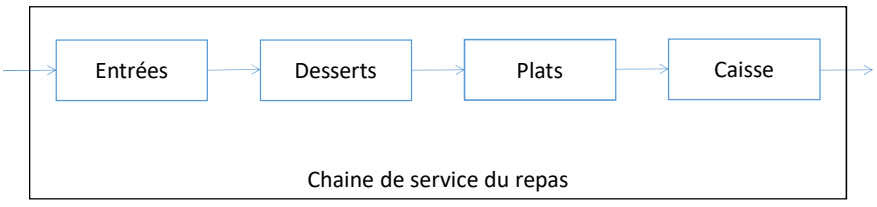
4

Taxonomie de Flynn (1966)

	Single Data	**Multiple Data**
Single Instruction		
Multiple Instruction	Rarement utilisé (traitement du signal)	

5

Exemple : restaurant universitaire

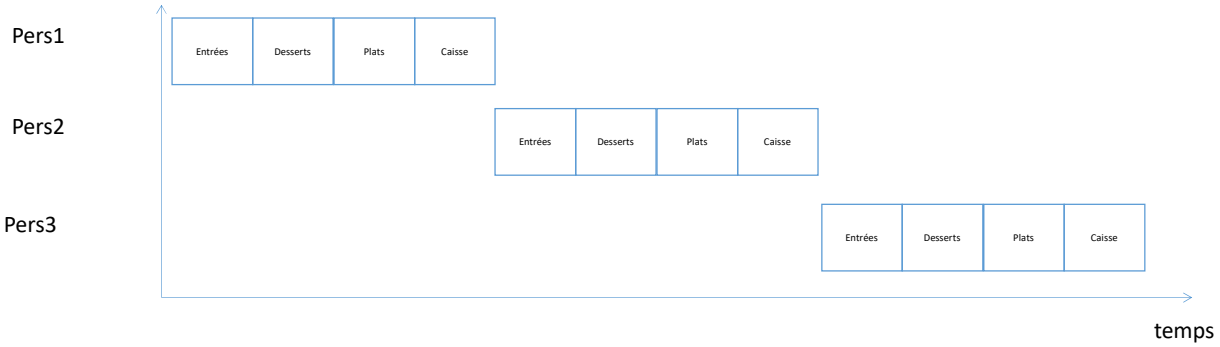


Dans l'ordre, nous devons passer devant ces 4 éléments

7

Exemple : restaurant universitaire

1er mode



T_e = temps de passage à chaque étape

$T_p = n * T_e$ = temps de passage pour n étapes

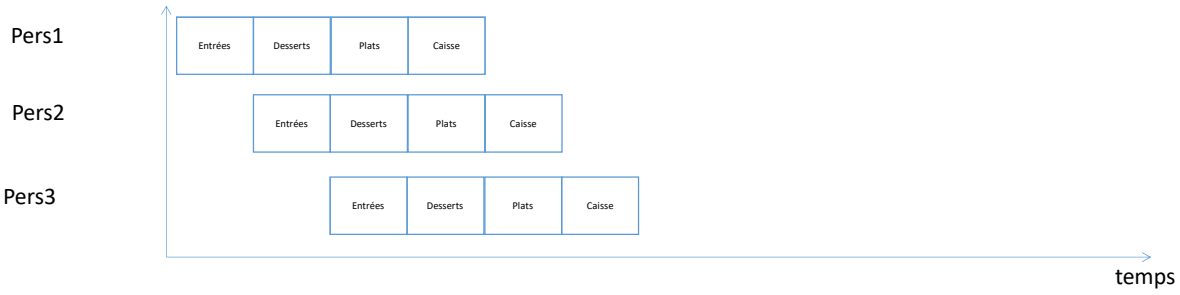
$T_t = m * T_P$ = temps total pour m personnes

$$\Rightarrow \boldsymbol{T_t = n * m * T_e}$$

8

Exemple : restaurant universitaire

2e mode



T_e = temps de passage à chaque étape

$T_p = n * T_e$ = temps de passage pour n étapes = temps de passage de la 1ere personne

$T_t =$

$$\Rightarrow \boldsymbol{T_t = f(n, m, T_e)} =$$

9

Exemple : restaurant universitaire

- 3 personnes, T_e=10s, temps total =

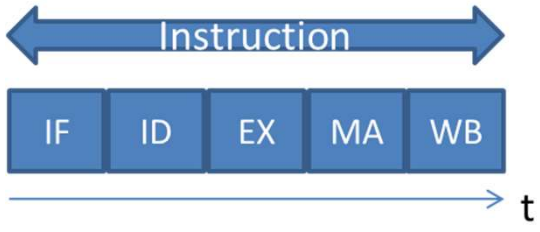
Passage séquentiel	Pipeline

- Intérêts du 2e mode :
  - Plusieurs personnes se servent en même temps.
  - Gain de temps : plus de personnes passent pendant une même durée.
  - Meilleure gestion des éléments : toujours utilisés.
- Inconvénients du 2e mode :

11

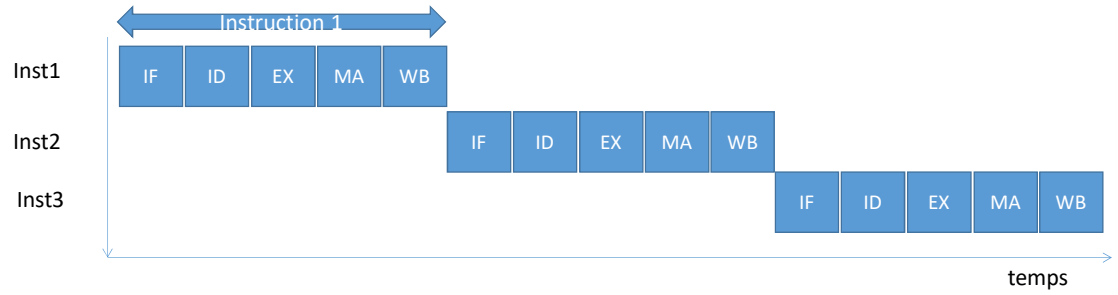
Jeu d'instructions

- Découpage d'une instruction dans le cas d'un processeur MIPS.
  - En sous parties élémentaires.
  - En relation avec les étapes de traitement de l'instruction.



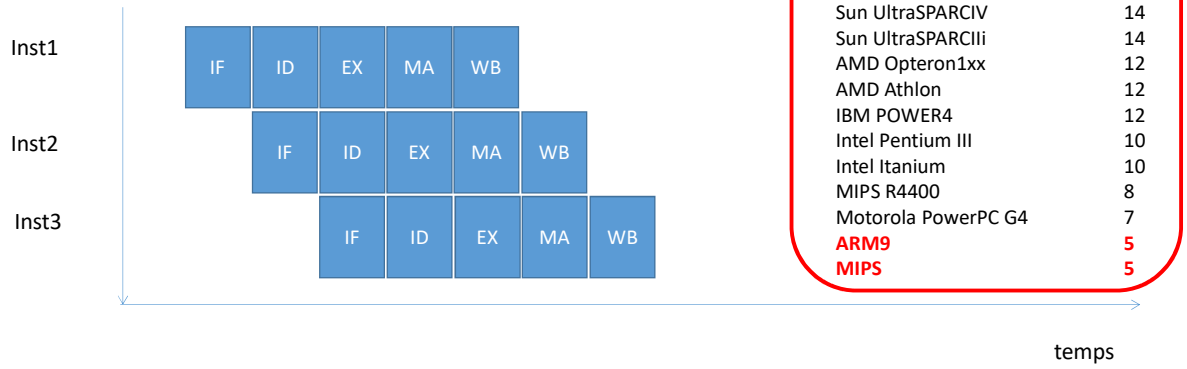
Fetch	Recherche d'instruction
Decode	décodage de l'instruction et lecture des registres opérandes
Execute	exécution de l'opération ou calcul de l'adresse de mémoire
Memory	accès de la mémoire ou écriture dans le PC de l'adresse de saut
Write Back	écriture dans un registre du résultat de l'opération

Traitement non-pipeliné (séquentiel)



14

Pipeline

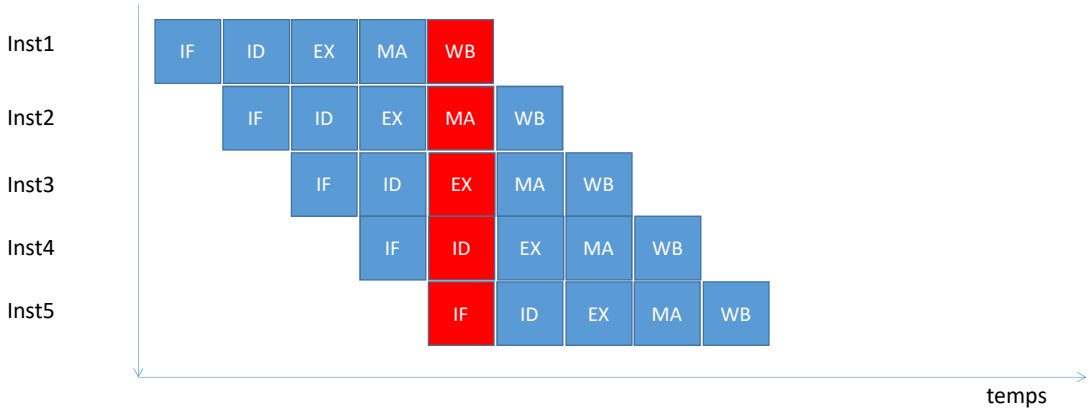


Intel Pentium 4 Prescott	31
Intel Pentium 4	20
AMD K10	16
IBM POWER5	16
IBM PowerPC 970	16
Intel Core2 Duo	14
Intel Pentium II	14
Sun UltraSPARCIV	14
Sun UltraSPARCIIi	14
AMD Opteron1xx	12
AMD Athlon	12
IBM POWER4	12
Intel Pentium III	10
Intel Itanium	10
MIPS R4400	8
Motorola PowerPC G4	7
ARM9	**5**
MIPS	**5**

Exemple de pipeline à 3 niveaux : Fetch, Decode, Execute

15

Parallélisme d'instructions : 5 étages

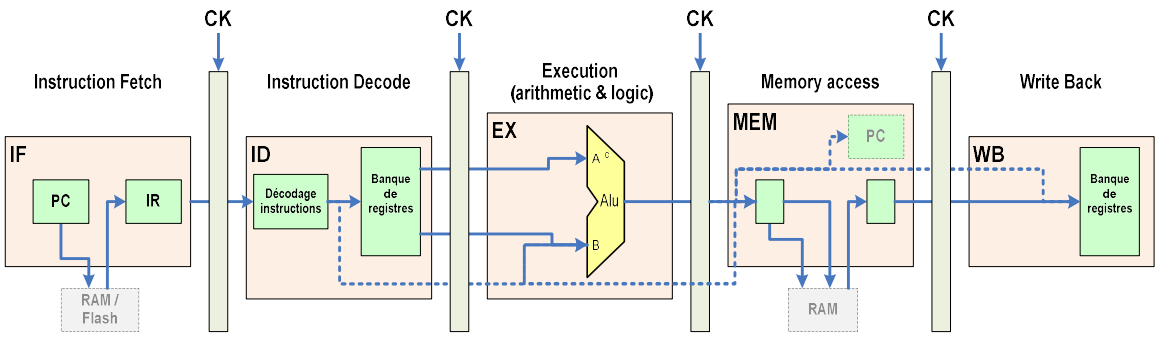


Comment peut-on l'implémenter ?

16

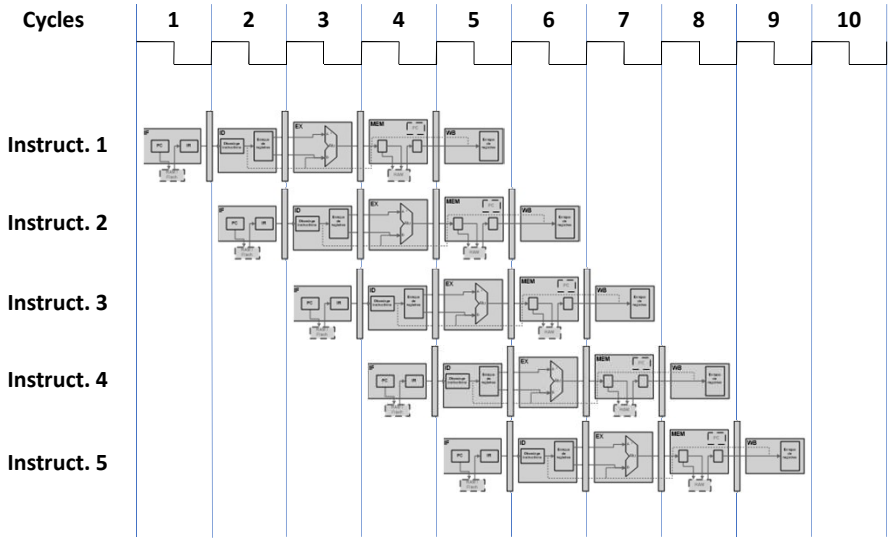
Pipeline à 5 niveaux

- Chaque étage est **isolé par un registre**
- Temps de cycle identique pour chaque étape (ou phase)
- Les traitements 1, 2, 3, 4 et 5 s'effectuent en parallèle



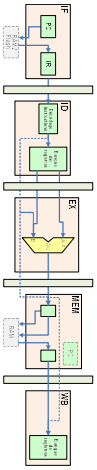
17

Pipeline à 5 niveaux



18

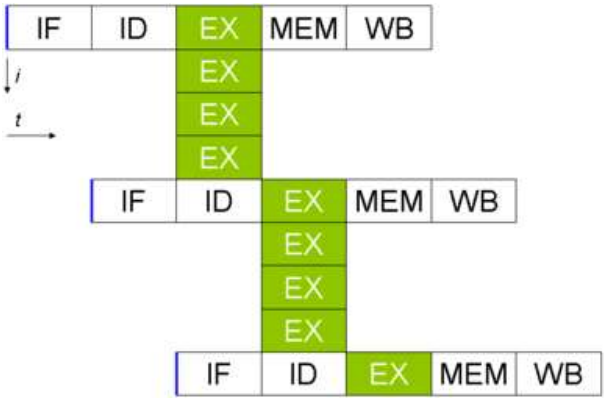
Pipeline à 5 niveaux

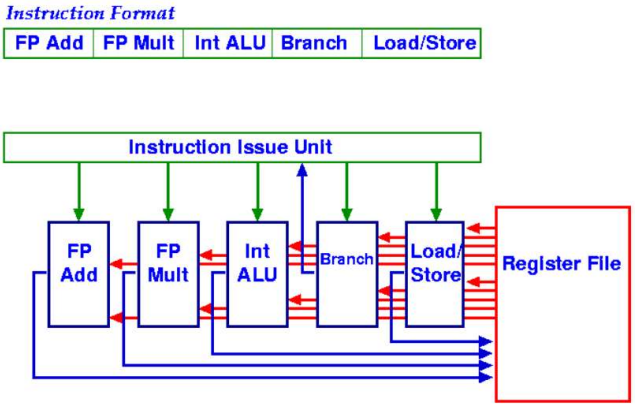


Cycles	1	2	3	4	5	6	7	8	9	10
	Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6				
		Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6			
			Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6		
				Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6	
					Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6

19

Architecture VLIW

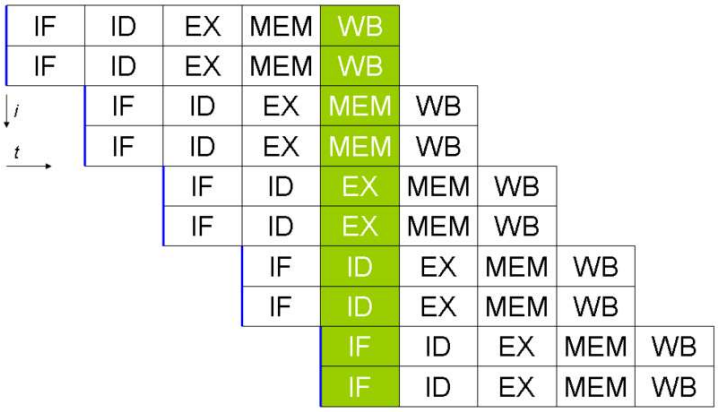


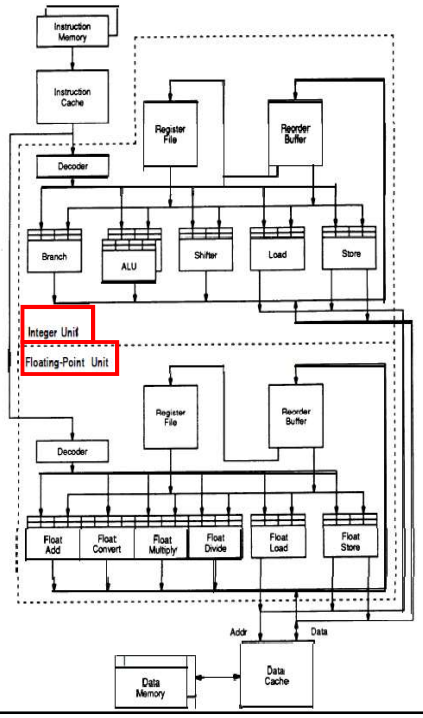


20

Architecture Superscalaire

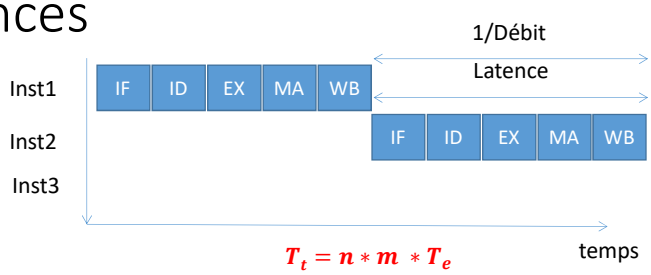
- Traitement simultané de plusieurs instructions





Calcul de performances

- **Latence** : temps écoulé entre le début et la fin d'exécution de la tâche (instruction).



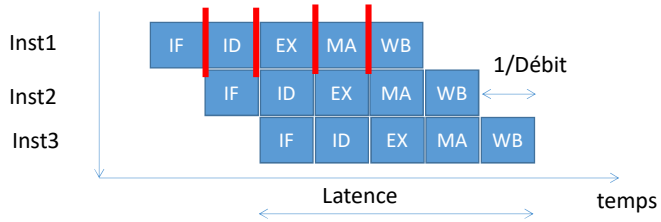
$T_t = n * m * T_e$

- **Débit**: nombre d'opérations (tâches, instructions) exécutées par unités de temps.

$D = \frac{1}{T_{exi}}$, avec $T_{exi} = \frac{T_t}{m}$, temps exécution total ramené à 1 instruction:

$$D = \frac{m}{(n+m-1)*T_e} \approx \boxed{\frac{1}{T_e}}$$

Cycle machine



$T_t = (n + m - 1) * T_e$

22

Calcul de performances

- **Accélération**: nombre de fois plus vite qu'en séquentiel.

$$A = \frac{T_{seq}}{T_{pip}} = \frac{m*n*T_e}{(n+m-1)*T_e} = \frac{m*n}{n+m-1} \sim n \text{ (pour m très grand)}$$

23

Pipeline : Aléas

Aléas
- Types d'aléas
- Résolution d'aléas
  - Forwarding/bypassing
  - Ordonnancement dynamique
    - Scoreboarding
    - Méthode de Tomasulo
  - Arrêt de pipeline
  - Prédiction dynamique de branchement

24

Problèmes des architectures pipelinées (aléas)

- Les aléas sont inhérents au *parallélisme*

- **Aléas structurels**
  - conflits d'accès aux ressources

- **Aléas de données**
  - modification de l'ordre d'accès aux opérandes

- **Aléas de contrôle**
  - décision de branchement

25

Dépendance de données

- **RAW (read after write):**
  L'instruction n+x lit une source modifiée par l'instruction n

```
ADD R1, R2, R3      R1 = R2 + R3
SUB R5, R1, #2      R5 = R1 - 2
```

- **WAR (write after read):**
  L'instruction n+x écrit dans une destination que l'instruction n utilise comme source

```
ADD R1, R2, R3      R1 = R2 + R3
SUB R2, R4, #2      R2 = R4 – 2
```

- ***WAW (write after write):***
  L'instruction n+x écrit dans une destination et l'instruction n écrit dans cette même destination.

```
ADD R1, R2, R3      R1 = R2 + R3
AND R5, R1, R2
SUB R1, R4, #2      R1 = R4 – 2
```

26

Exercice : indiquer les dépendances de type :

RAW

```
1 LDR  R1, [R0]
2 LDR  R2, [R1]
3 ADD  R6, R5, R4
4 ADD  R3, R1, R2
5 LDR  R4, [R6]
6 SUB  R2, R0, R4
7 ADD  R7, R1, 4
8 ADD  R4, R1, R3
9 SUB  R6, R7, R4
```

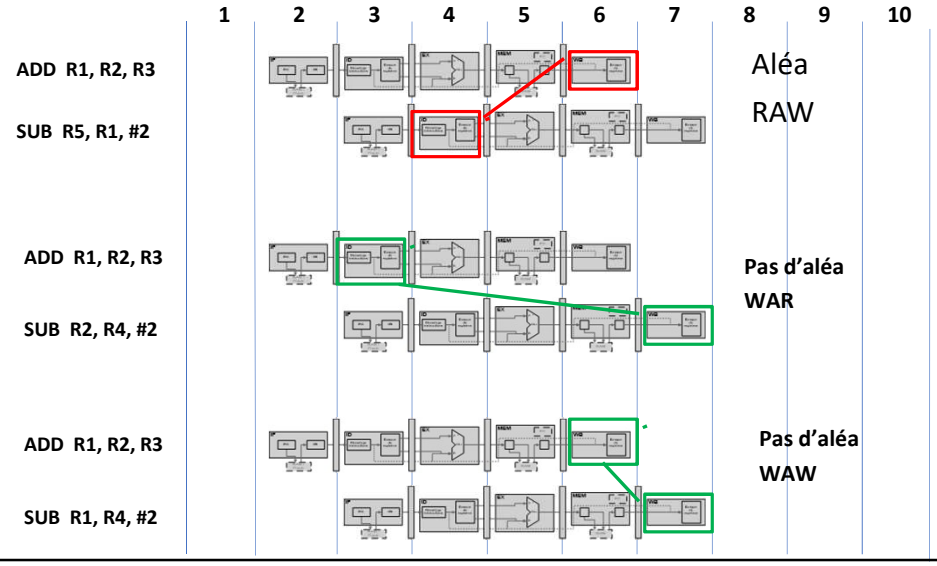
WAR

```
1 LDR  R1, [R0]
2 LDR  R2, [R1]
3 ADD  R6, R5, R4
4 ADD  R3, R1, R2
5 LDR  R4, [R6]
6 SUB  R2, R0, R4
7 ADD  R7, R1, 4
8 ADD  R4, R1, R3
9 SUB  R6, R7, R4
```

WAW

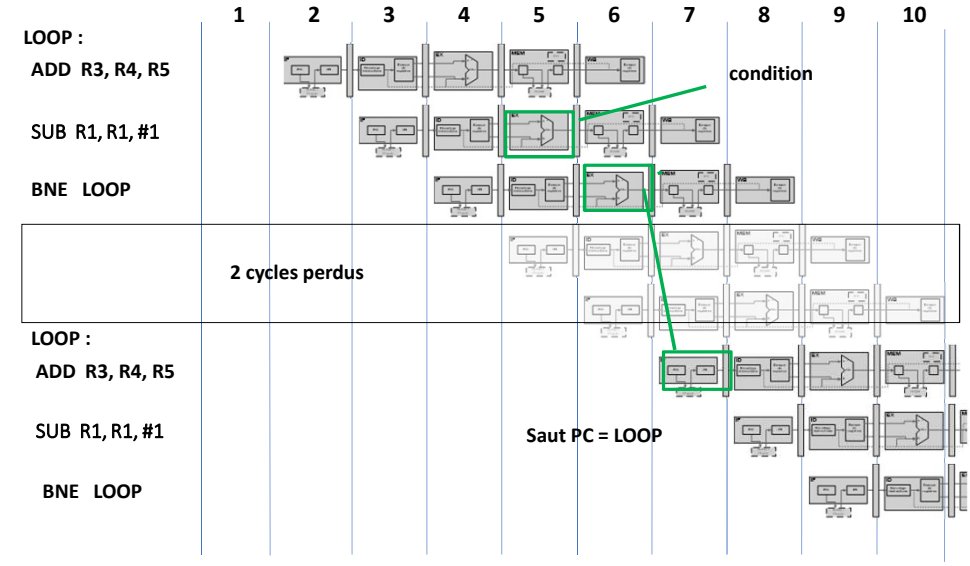
```
1 LDR  R1, [R0]
2 LDR  R2, [R1]
3 ADD  R6, R5, R4
4 ADD  R3, R1, R2
5 LDR  R4, [R6]
6 SUB  R2, R0, R4
7 ADD  R7, R1, 4
8 ADD  R4, R1, R3
9 SUB  R6, R7, R4
```

Aléas de données (ARM9)



30

Aléas de contrôle (ARM9)



31

Résolution d'aléas

Aléas
- Aléas de données
- Aléas de contrôle
- Résolution d'aléas
  - Arrêt de pipeline
  - Forwarding/bypassing
  - Ordonnancement dynamique
    - Scoreboarding
    - Méthode de Tomasulo
  - Prédiction dynamique de branchement

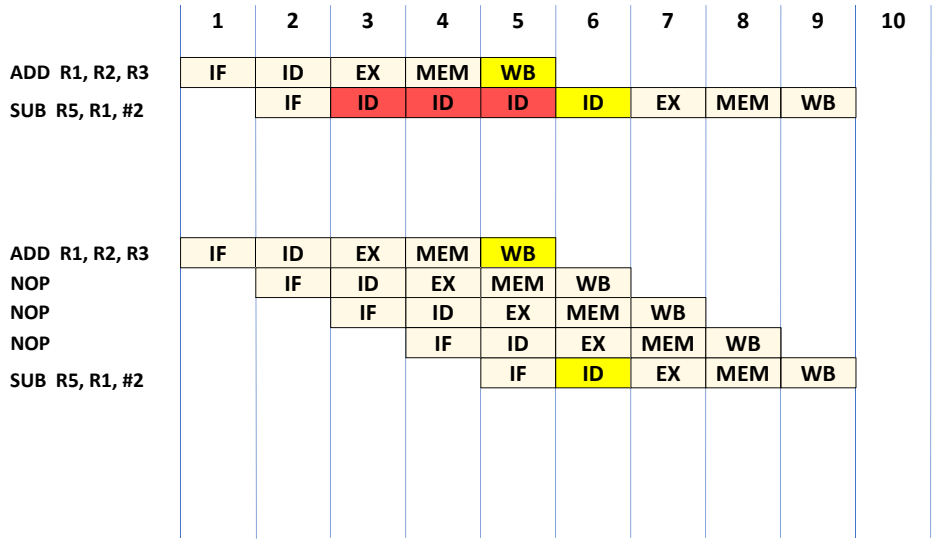
32

1. Arrêt de pipeline (hardware/software)

- Méthodes simples (et pénalisantes) :
  - **Hardware**: arrêter le pipeline (stall / break)
  - **Software** : insérer des NOPs (no opération)
- Calcul de la pénalité et de l'IPC (nombre d'instruction par cycle)
- **Exemple :**
  - **Arrêt de 3 cycles pour 20% des instructions**
  - **IPC = 1/ (0.8 x 1 + 0.2 x 4) = 0.625 instructions par cycle**

33

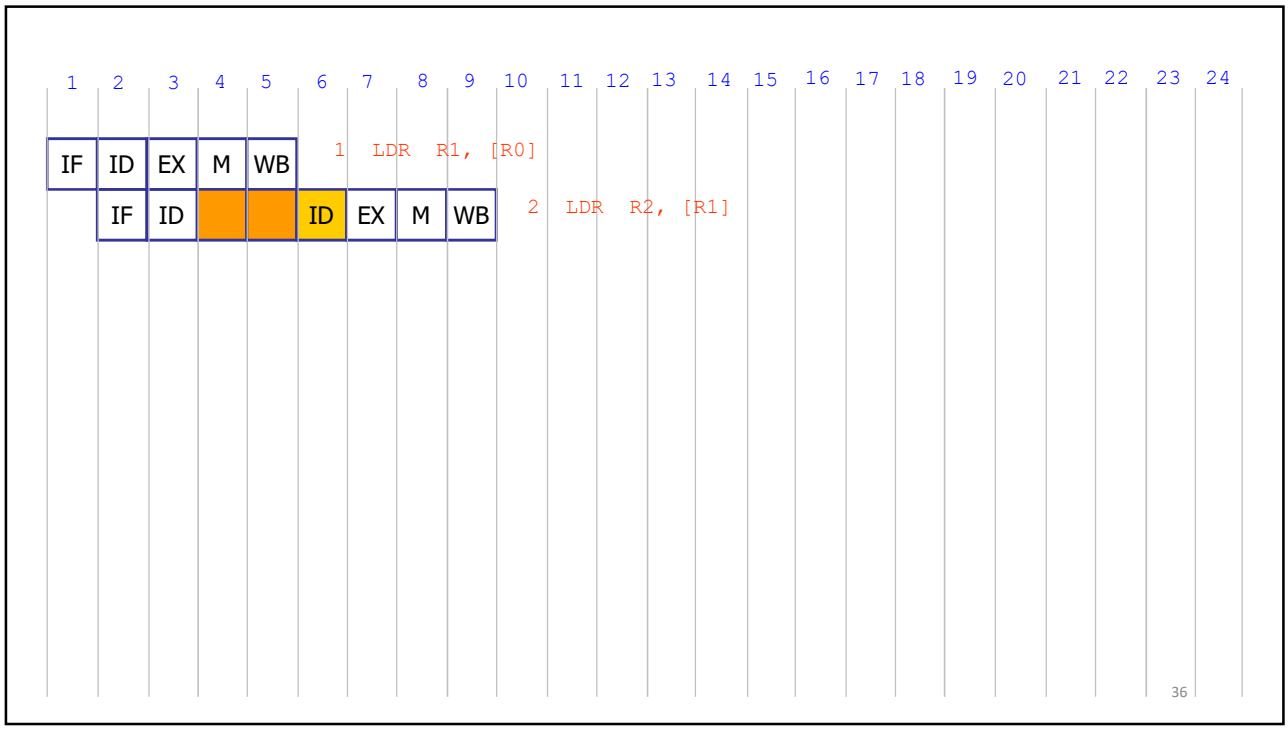
1. Arrêt de pipeline (hardware/software)

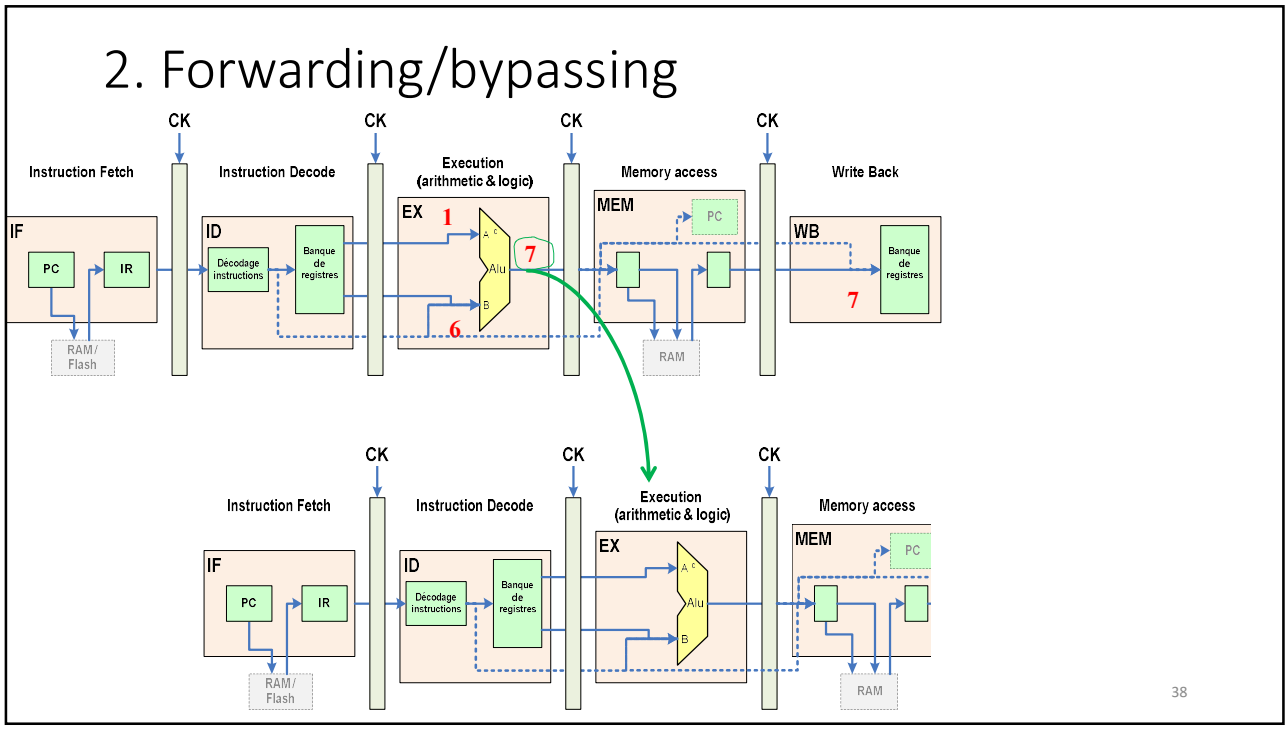


Exercice : Arrêt de pipeline

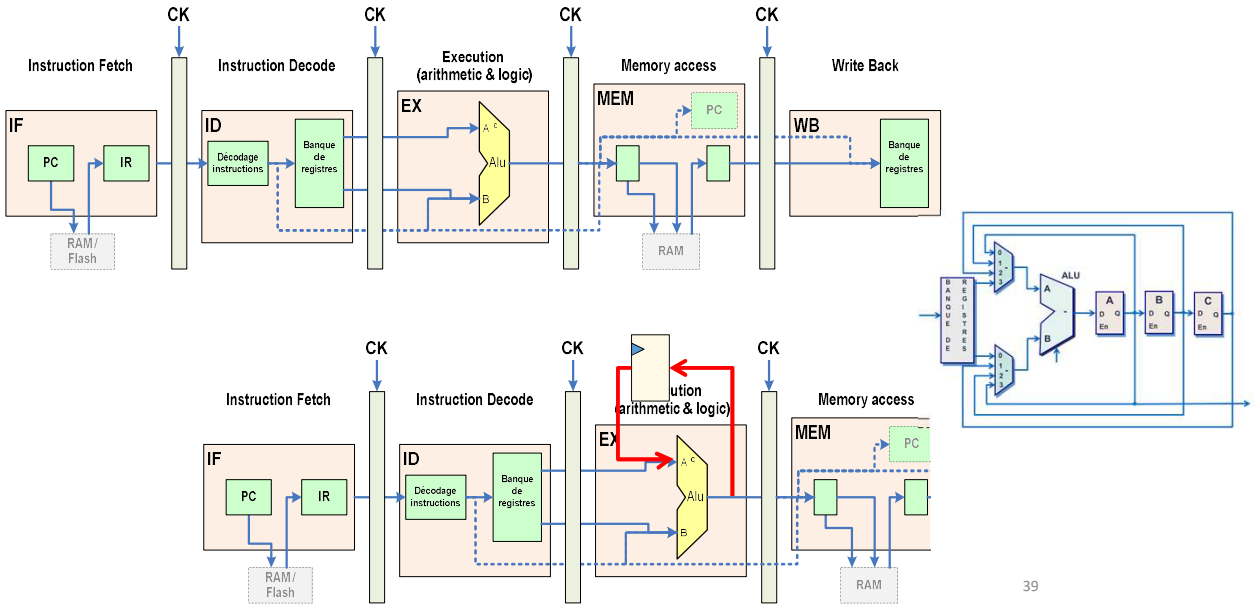
- Dessinez le chronogramme de l'exécution du code avec résolution des aléas par arrêt (hardware) du pipeline. Quel est l'IPC pour ces 9 instructions ?

```
1  LDR   R1, [R0]
2  LDR   R2, [R1]
3  ADD   R6, R5, R4
4  ADD   R3, R1, R2
5  LDR   R4, [R6]
6  SUB   R2, R0, R4
7  ADD   R7, R1, #4
8  ADD   R4, R1, R3
9  SUB   R6, R7, R4
```


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
IF ID EX M WB
IF ID ID EX M WB
1 LDR R1, [R0]
2 LDR R2, [R1]
36


2. Forwarding/bypassing
CK
Instruction Fetch
Instruction Decode
Execution (arithmetic & logic)
Memory access
Write Back
IF
PC
IR
RAM / Flash
ID
Décodage instructions
Banque de registres
EX
1
A
B
Alu
7
6
MEM
PC
RAM
WB
7
Banque de registres
38

2. Forwarding/bypassing



39

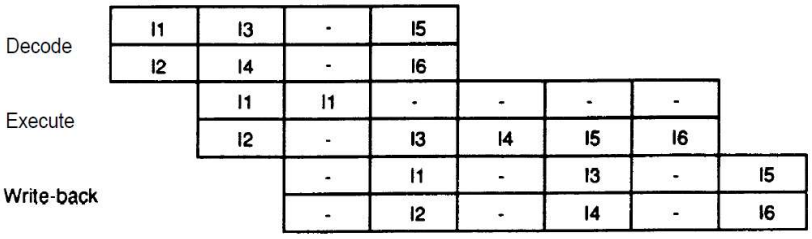
Autres méthodes de résolution d'aléas

- Aléas insolubles par les méthodes vues précédemment si :
  - Les pipelines sont **très longs** >5
  - Le temps de traitement et/ou le nombre de niveau de pipeline **varie** selon les instructions
- Deux approches dans ces cas :
  - **Ordonnancement dynamique**
    - **Scoreboarding**
    - **méthode de Tomasulo**
  - **Prédiction dynamique**

40

Instructions «In-order»

- Instructions lues dans l'ordre où elles sont lues en mémoire
- Processeur ralenti par des dépendances et des conflits



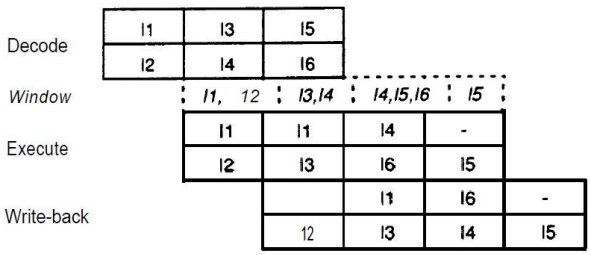
Total number of cycles = 8

Notes:
I1 requires two cycles to execute
I3 and I4 conflict for functional unit
I5 depends on I4
I5 and I6 conflict for functional unit

41

Ordonnancement dynamique («out-of-order»)

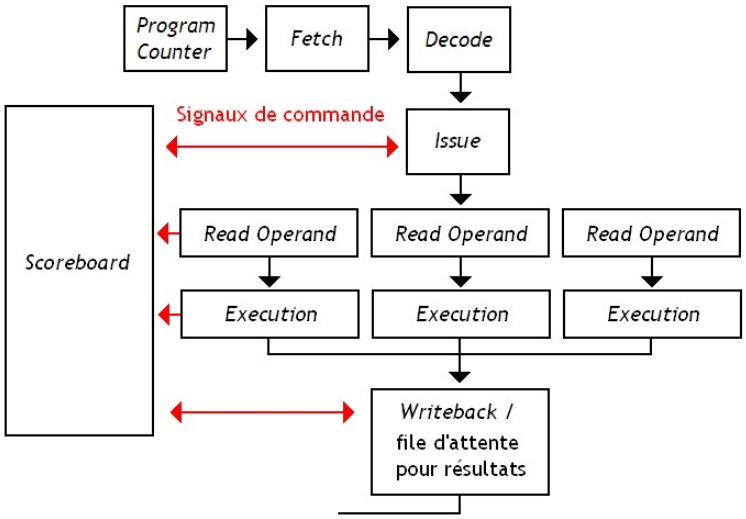
- Principe : **l'ordre d'exécution** des instructions est modifié dynamiquement pour éliminer les dépendances (et les aléas)
  - **Scoreboarding**
  - **Méthode de Tomasulo**



Total number of cycles = 6
I1 and I6 complete out-of-order
I6 issues out-of-order

Notes:
I1 requires two cycles to execute
I3 and I4 conflict for functional unit
I5 depends on I4
I5 and I6 conflict for functional unit

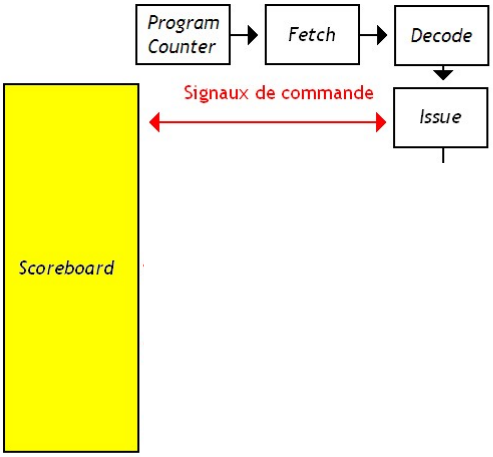
3. Scoreboarding



- Instructions décodées **dans l'ordre**
- **Scoreboard** utilisé pour gérer et détecter les dépendances
- **4 étapes**

43

3. Scoreboarding



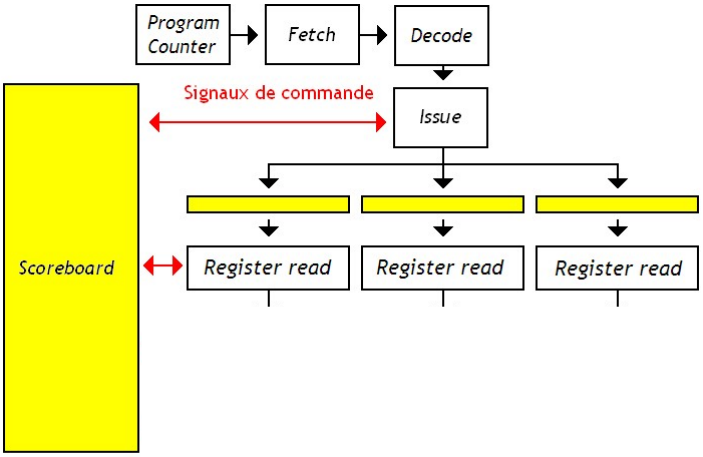
- *Emission* : le système vérifie quels registres vont être lus et écrits par l'instruction

WAW
Emission (issue)
L'instruction est bloquée jusqu'à ce que les instructions qui veulent écrire dans le même registre sont terminées.

Inconvénient du scoreboarding (WAW) :

44

3. Scoreboarding

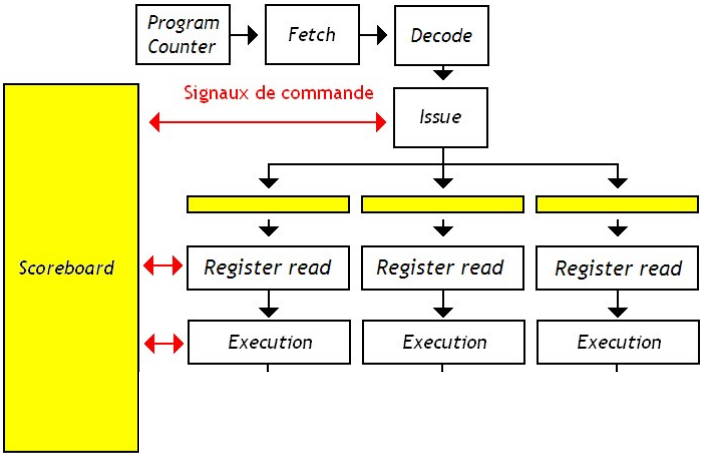


- *Read Operand :* les instructions dont les opérandes ne sont pas disponibles sont mise en attente dans une mémoire tampon

RAW
Read Operand
Les registres qui doivent être écrits par une autre instruction ne sont pas considérés disponibles jusqu'à ce qu'ils soient écrits.

46

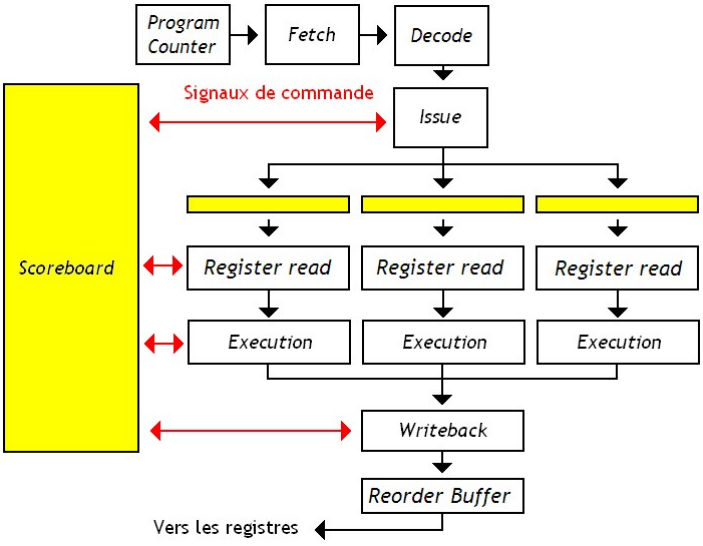
3. Scoreboarding



- *Execution :* Quand toutes les opérandes sont disponibles, l'instruction est exécutée. Lorsque le résultat est prêt, le scoreboard est notifié, ce qui lui permet de détecter la disponibilité des opérandes.

47

3. Scoreboarding



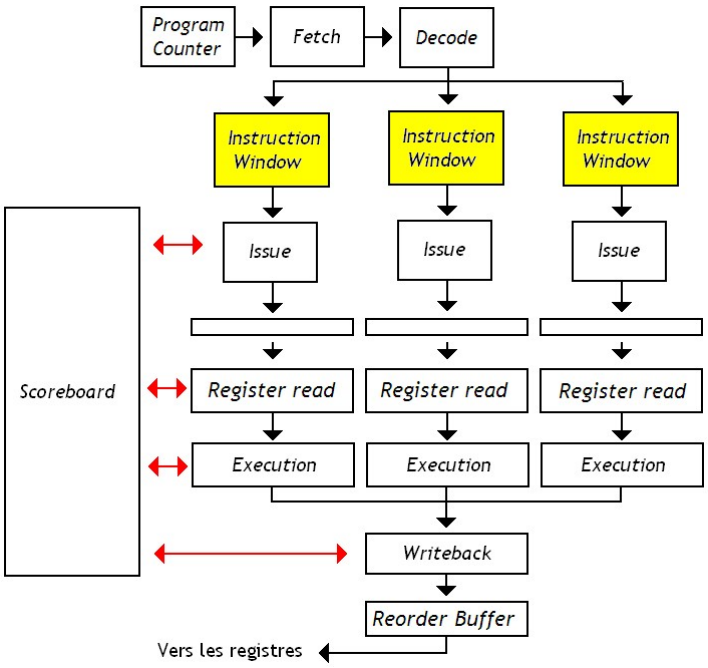
- *Writeback :* les résultats sont placés dans une file d'attente et s'enregistrent dans le banc de registres quand c'est possible. C'est le scoreboard qui donne l'autorisation d'écriture.

WAR
Writeback
Cette opération est retardée jusqu'à ce que les instructions précédentes (qui veulent lire le registre dans lequel cette instruction veut écrire) ont fini leur étape de «Read Operand».

48

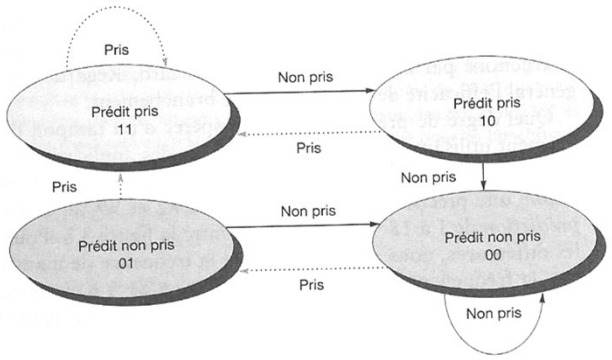
4. Méthode de Tomasulo

- L'étage d'émission (Issue) met en attente les instructions sans bloquer le pipeline.
- Utilisation de mémoires tampons (**fenêtre d'instructions** ou **stations de réservation**)



5. Prédiction dynamique de branchement

- Mécanisme dit du compteur 2 bits
- Machine d'état à 4 états
- Détecte si le branchement est habituellement pris ou non pris
- Accepte une exception



50

Exercice : forwarding

Un processeur ARM avec pipeline à 5 niveaux exécute le programme assembleur suivant :

- Indiquez les dépendances de type RAW présentes dans le programme.
- Représentez schématiquement, cycle d'horloge par cycle d'horloge, l'exécution du programme dans le temps si la seule solution aux aléas du pipeline est l'arrêt de ce dernier (sans forwarding).
- Représentez de la même façon l'exécution du programme si le pipeline est doté de forwarding.

1	**LDR**	**r1,[r5]**
2	**ADD**	**r5,r1,1**
3	**LDR**	**r1,[r6]**
4	**ADD**	**r3,r5,r6**
5	**SUB**	**r2,r6,#1**
6	**SUB**	**r4,r3,#5**
7	**ADD**	**r3,r2,r4**
8	**LDR**	**r2,[r7]**
9	**ORR**	**r4,r2,r1**
10	**SUB**	**r7,r3,#9**

51

			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	LDR	r1,[r5]		F	D	E	M	W																						
2	ADD	r5,r1,1	1																											
3	LDR	r1,[r6]																												
4	ADD	r3,r5,r6	2																											
5	SUB	r2,r6,#1																												
6	SUB	r4,r3,#5	4																											
7	ADD	r3,r2,r4	5,6																											
8	LDR	r2,[r7]																												
9	ORR	r4,r2,r1	3,8																											
10	SUB	r7,r3,#9	4,7																											

			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	LDR	r1,[r5]		F	D	E	M	W																						
2	ADD	r5,r1,1	1																											
3	LDR	r1,[r6]																												
4	ADD	r3,r5,r6	2																											
5	SUB	r2,r6,#1																												
6	SUB	r4,r3,#5	4																											
7	ADD	r3,r2,r4	5,6																											
8	LDR	r2,[r7]																												
9	ORR	r4,r2,r1	3,8																											
10	SUB	r7,r3,#9	4,7																											

Exercice : arrêt de pipeline et forwarding

Un processeur ARM avec pipeline à 5 niveaux exécute le programme assembleur suivant :

1	LDRH	r1,[r0]
2	LDRH	r2,[r0,#1*2]
3	ADD	r3,r1,r2
4	STRH	r3,[r0,#6*2]
5	LDRH	r4,[r0,#4*2]
6	ADD	r5,r1,r4
7	STRH	r5,[r0,#8*2]

a) Indiquez les dépendances de type RAW présentes dans le programme.

			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	LDRH	r1,[r0]		F	D	E	M	W																					
2	LDRH	r2,[r0,#1*2]																											
3	ADD	r3,r1,r2	1,2																										
4	STRH	r3,[r0,#6*2]	3																										
5	LDRH	r4,[r0,#4*2]																											
6	ADD	r5,r1,r4	1,5																										
7	STRH	r5,[r0,#8*2]	6																										

Exercice : arrêt de pipeline et forwarding

b) Représentez schématiquement, cycle d'horloge par cycle d'horloge, l'exécution du programme dans le temps si la seule solution aux aléas du pipeline est l'arrêt de ce dernier (sans forwarding).

			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	LDRH	r1,[r0]		F	D	E	M	W																					
2	LDRH	r2,[r0,#1*2]																											
3	ADD	r3,r1,r2	1,2																										
4	STRH	r3,[r0,#6*2]	3																										
5	LDRH	r4,[r0,#4*2]																											
6	ADD	r5,r1,r4	1,5																										
7	STRH	r5,[r0,#8*2]	6																										

56

Exercice : arrêt de pipeline et forwarding

c) Représentez de la même façon l'exécution du programme si le pipeline est doté de forwarding.

			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	LDRH	r1,[r0]		F	D	E	M	W																					
2	LDRH	r2,[r0,#1*2]																											
3	ADD	r3,r1,r2	1,2																										
4	STRH	r3,[r0,#6*2]	3																										
5	LDRH	r4,[r0,#4*2]																											
6	ADD	r5,r1,r4	1,5																										
7	STRH	r5,[r0,#8*2]	6																										

58

Exercice : arrêt de pipeline et forwarding

d) Réordonnez les instructions du programme de façon à supprimer tous les arrêts du pipeline avec le forwarding.

			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	LDRH	r1,[r0]		F	D	E	M	W																					
2	LDRH	r2,[r0,#1*2]																											
5	LDRH	r4,[r0,#4*2]																											
3	ADD	r3,r1,r2	1,2																										
4	STRH	r3,[r0,#6*2]	3																										
6	ADD	r5,r1,r4	1,5																										
7	STRH	r5,[r0,#8*2]	6																										

Exercice : arrêt de pipeline et forwarding

e) Supposons maintenant que le processeur ne puisse pas arrêter son pipeline et ne comporte pas un mécanisme de forwarding. Le compilateur résout les aléas en réordonnant les instructions et en introduisant des NOP. Donnez le code optimisé (minimum d'instructions) généré par le compilateur.

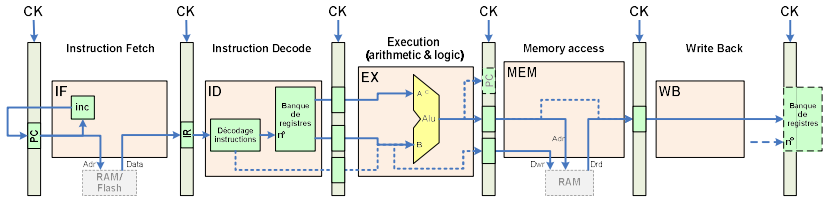
			Dep.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	LDRH	r1,[r0]		F	D	E	M	W																					
2	LDRH	r2,[r0,#1*2]																											
5	LDRH	r4,[r0,#4*2]																											
	nop																												
	nop																												
3	ADD	r3,r1,r2	1,2																										
6	ADD	r5,r1,r4	1,5																										
	nop																												
	nop																												
4	STRH	r3,[r0,#6*2]	3																										
7	STRH	r5,[r0,#8*2]	6																										

Exercice : arrêt de pipeline

Soit un processeur ARM avec pipeline à 5 niveaux ayant l'architecture suivante et exécutant le programme suivant :

a) Indiquez les dépendances de type RAW présentes dans le programme.

b) Indiquez les aléas de contrôle.




```
0x00    MOV     r0,#0x05
0x02    MOV     r1,#0x13
0x04    ADD     r3,r0,r1
0x06    MOV     r4,#0x20
0x08    STRH    r3,[r4]
0x0A    ADD     r3,r3,r0
0x0C    MOV     r0,#0x01
0x0E    B       0x50
0x10    MOV     r3,#00
0x12    ADD     r1,r2,r3
0x14    LDRH    r1,[r4]
.....

0x50    LDRH    r3,[r4]
0x52    ADD     r1,r3,r0
```

64

Exercice : arrêt de pipeline

c) Représentez schématiquement, cycle d'horloge par cycle d'horloge, l'exécution du programme dans le temps si la seule solution aux aléas du pipeline (de données ou de contrôle) est l'arrêt de ce dernier.

Adr	Instr.	Dépendance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0x00	MOV r0,#0x05		F	D	E	M	W																				
0x02	MOV r1,#0x13																										
0x04	ADD r3,r0,r1	r0, r1																									
0x06	MOV r4,#0x20																										
0x08	STRH r3,[r4]	r3, r4																									
0x0A	ADD r3,r3,r0	r0, r3																									
0x0C	MOV r0,#0x01																										
0x0E	B 0x50																										
0x10	MOV r3,#00																										
0x12	ADD r1,r2,r3	r3																									
0x14	LDRH r1,[r4]	r1																									
0x16																											
0x50	LDRH r3,[r4]																										
0x52	ADD r1,r3,r0	r0, r3																									

66

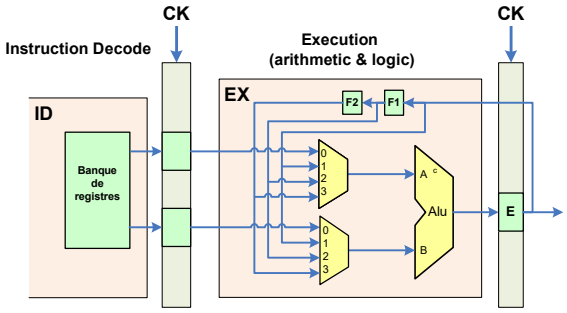
Exercice : état des registres avec forwarding

Un processeur ARM avec pipeline à 5 niveaux exécute le programme assembleur suivant.

- Représentez schématiquement, cycle d'horloge par cycle d'horloge, l'exécution du programme dans le temps dans le cas d'un pipeline avec *forwarding*.
- Vous devez indiquer l'état des registres E, F1 et F2 du forwarding en indiquant les situations où l'un de ces registres est utilisé.

```
0x00    MOV    r0,#0x05
0x02    MOV    r1,#0x13
0x04    MOV    r2,#0xA0
0x06    ADD    r3,r2,r1
0x08    ADD    r4,r1,r3
```

Structure du *forwarding* dans l'étage *Execute* :



Exercice : état des registres avec forwarding

Adr	Instr.	Dépendance		1	2	3	4	5	6	7	8	9	10	11	12	Utilisation registres forwarding
0x00	MOV r0,#0x05			F	D	E	M	W								
0x02	MOV r1,#0x13															
0x04	MOV r2,#0xA0															
0x06	ADD r3,r2,r1															
0x08	ADD r4,r1,r3															
0x0A																
0x0C																

Etat des registres du forwarding

	registre E					r0	r1	r2	r3	r4				
	registre F1													
	registre F2													

69