heig-vd
Haute Ecole d'Ingénierie
du Canton de Vaud

RéDS
heig-vd

ARO2

Traitement en pipeline des instructions

Basé sur le cours du prof. E. Sanchez
et le cours ASP du prof. M.Starkier

Romuald Mosqueron

Cours ARO2

PIPELINE

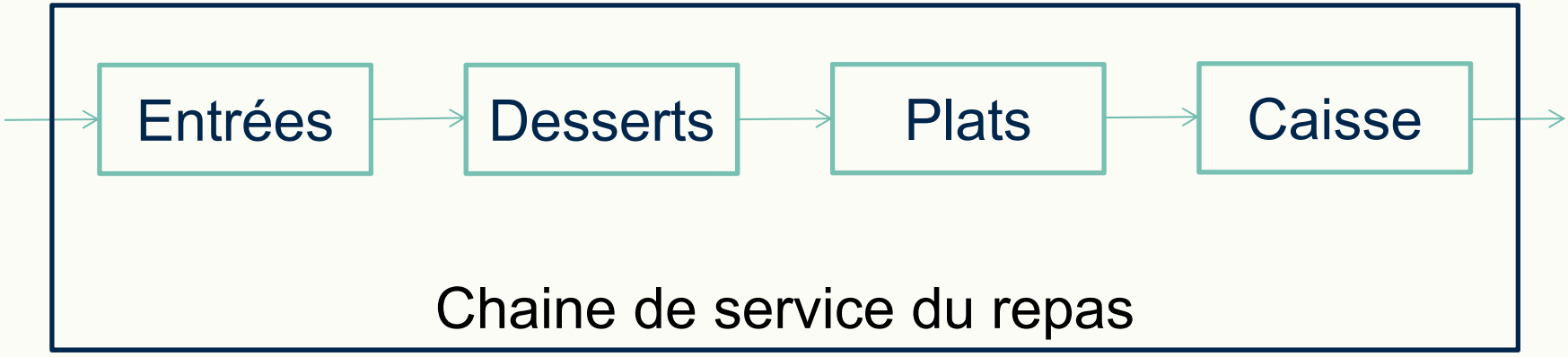
Explication par l'exemple

Exemple du restaurant universitaire

- Opération: se servir un repas

Architecture de la chaîne de service décomposée en sous-éléments indépendants:

- Un présentoir pour les entrées
- Un présentoir pour les desserts
- Un présentoir pour les plats de résistance
- Une caisse

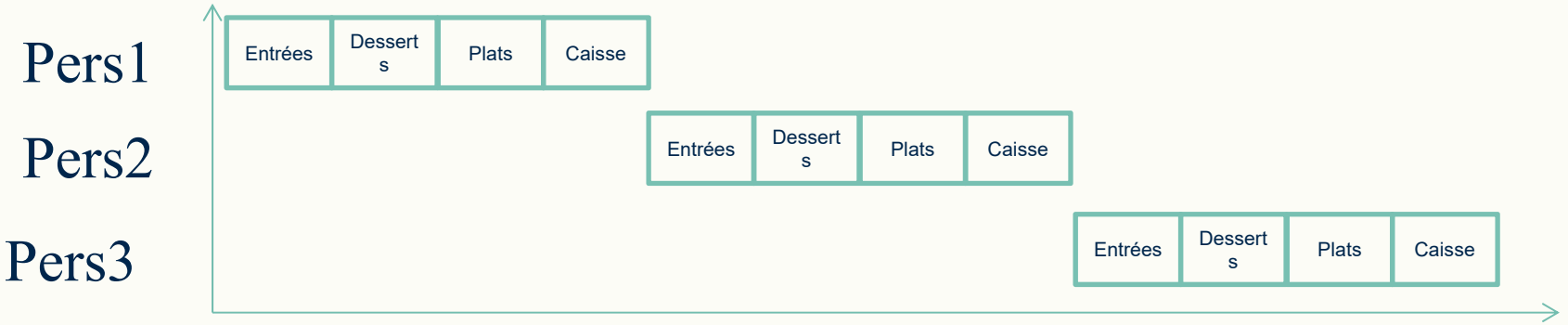


dans l'ordre, nous devons passer devant ces 4 éléments

Exemple: restaurant universitaire

2 modes d'utilisation pour se servir un repas.

- **1ère méthode :**
  - Une seule personne à la fois dans toute la chaîne de repas.
  - Quand elle a passé toute la chaîne et est sortie, une autre personne entre pour se servir.



Soit T_e, le temps de passage à chaque étape.

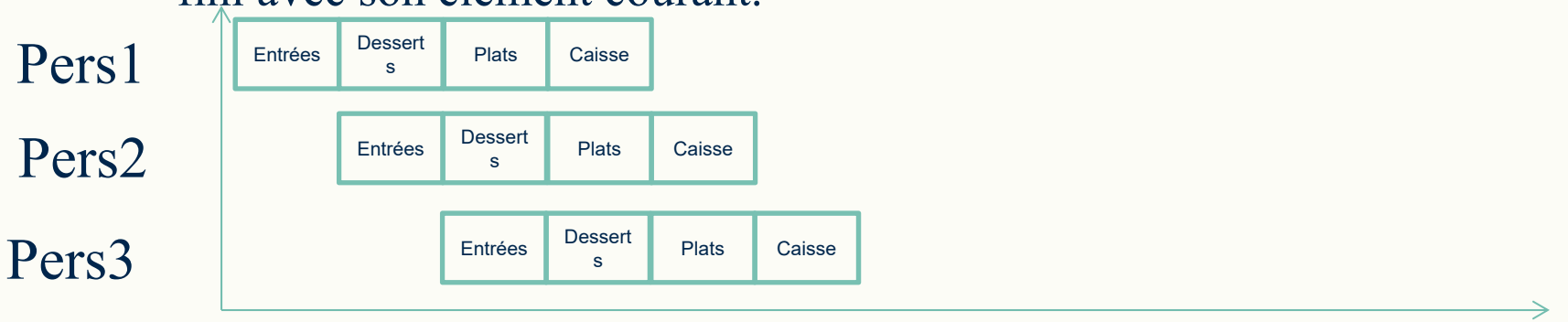
Alors $T_p = n * T_e$, T_p le temps de passage et n le nombre d'étapes dans un passage.

Donc, le temps total T_t pour m personnes est de $T_t = m * T_P$.

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

Exemple: restaurant universitaire

- **2ème méthode**
  - Plusieurs personnes à la fois, en décalé.
  - Une personne à chaque présentoir (élément).
  - Une personne passe à l'élément suivant quand il est libre et qu'elle en a fini avec son élément courant.



Soit T_e, le temps de passage à chaque étape.

Alors $T_p = n * T_e$, T_p le temps de passage et n le nombre d'étapes dans un passage.

$\Rightarrow$ temps de passage de la 1ère personne.

Il faut 1 temps d'étape par personne supplémentaire, i.e. $1 * T_e$, donc pour m personnes $T_t = T_p + (m-1) * T_e$, $m-1$ car m personne moins la première

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

$\Rightarrow \boldsymbol{T_t = (n + m - 1) * T_e}$

Exemple: restaurant universitaire

- Dans l'exemple pour 3 personnes si $T_e = 10s$ alors
  - Passage séquentiel = 120s.
  - Pipeline = 60s.

$\Rightarrow$ Intérêts du deuxième 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.

Pipeline: définition

- Définition

Technique utilisée pour optimiser le temps d'exécution d'une opération répétitive.

  - Principes
    - Lancer le traitement d'une opération avant que la précédente ne soit terminée ⇒ recouvrement des opérations.
    - A chaque étape de l'opération est affectée une ressource indépendante, de façon à pouvoir exécuter plusieurs opérations en parallèle, chacune à une étape différente

      ⇒ exploite le parallélisme entre les opérations d'un flot d'opérations séquentielles.
    - Optimiser le temps d'utilisation des différents éléments.
- Découpage des opérations en sous-parties élémentaires.
  - En relation avec les étapes de traitement de l'opération
  - Définition des étages du pipeline.
  - «travail à la chaine».

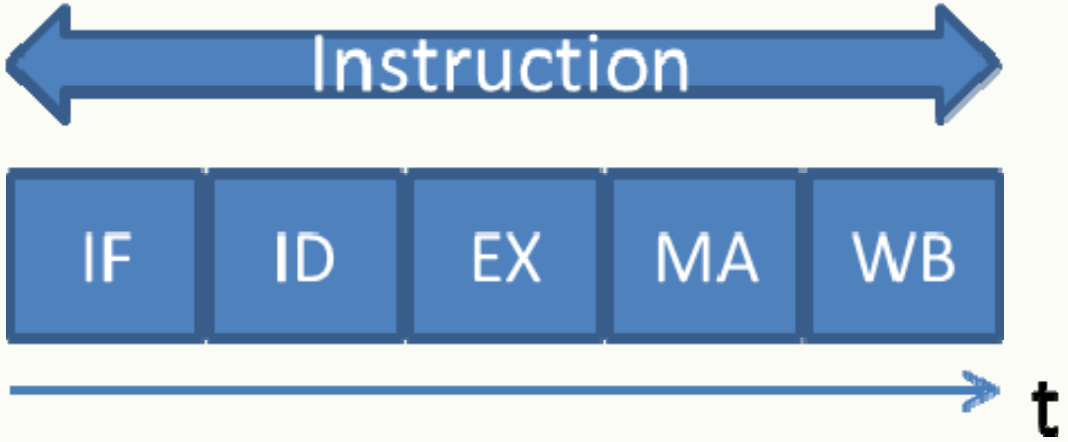
⇒ Exécution des sous-parties élémentaires dans les étages correspondants du pipeline.

Le pipeline

- Technique utilisée pour optimiser le temps d'exécution d'un processus
- Méthode : ne pas attendre d'avoir finir toutes les étapes du traitement du 1er processus pour commencer le 2ème ....
- Le traitement des étapes s'effectue en parallèle, c'est à dire simultanément pour plusieurs processus

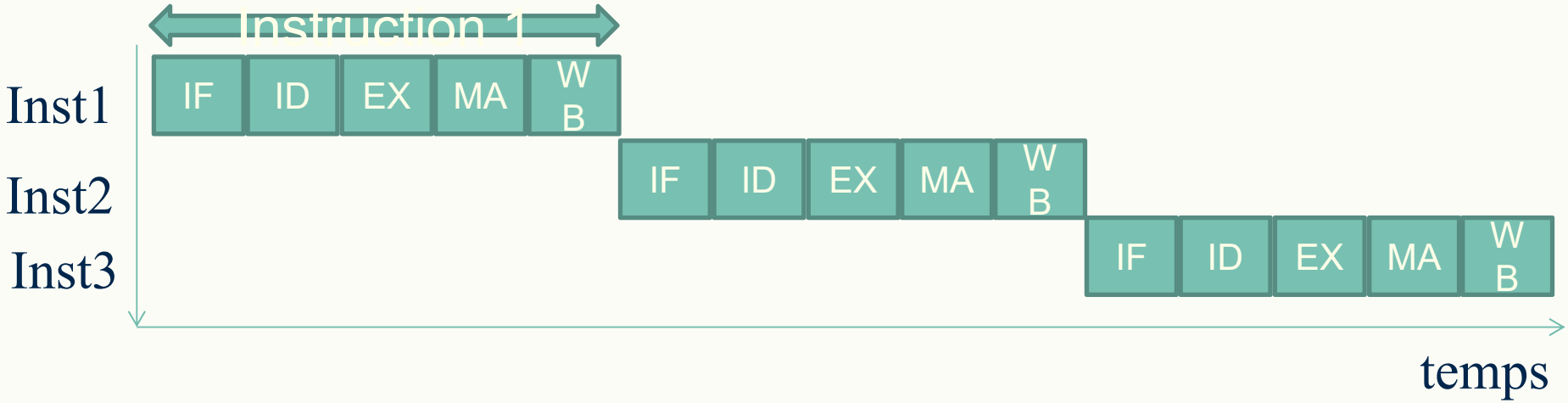
Pipeline: découpage d'une instruction

- 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.



Cas du processeur: sans pipeline

- Traitement normal non-pipeliné (séquentiel) du flot d'instructions.
  - Les instructions sont traitées les unes à la suite des autres.
  - Lorsque le traitement se situe à une étape, les autres éléments sont au repos i.e. inutilisés.
  - Pas de recouvrement.



Dans un micro-processeur sans pipeline, les instructions sont exécutées les unes après les autres. En supposant que chaque étape met 1 cycle d'horloge pour s'exécuter

⇒ 5 cycles pour exécuter une instruction

⇒ 15 cycles pour 3 instructions.

Cas du processeur: avec pipeline

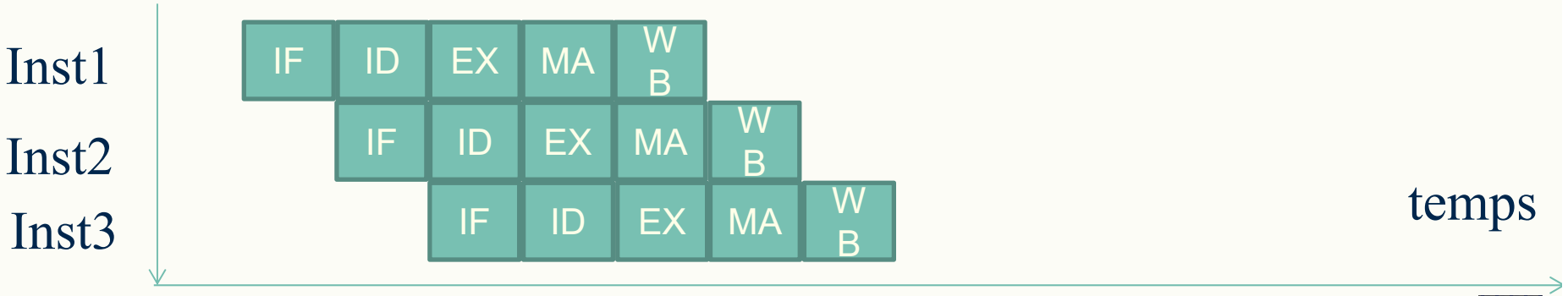
- Solution avec pipeline:
  - A chaque étape du processus est affectée une ressource indépendante, de façon à pouvoir exécuter plusieurs processus en parallèle, chacun à une étape différente.

⇒ Exécuter simultanément les différentes étapes sur des données distinctes (parallélisme d'instructions).

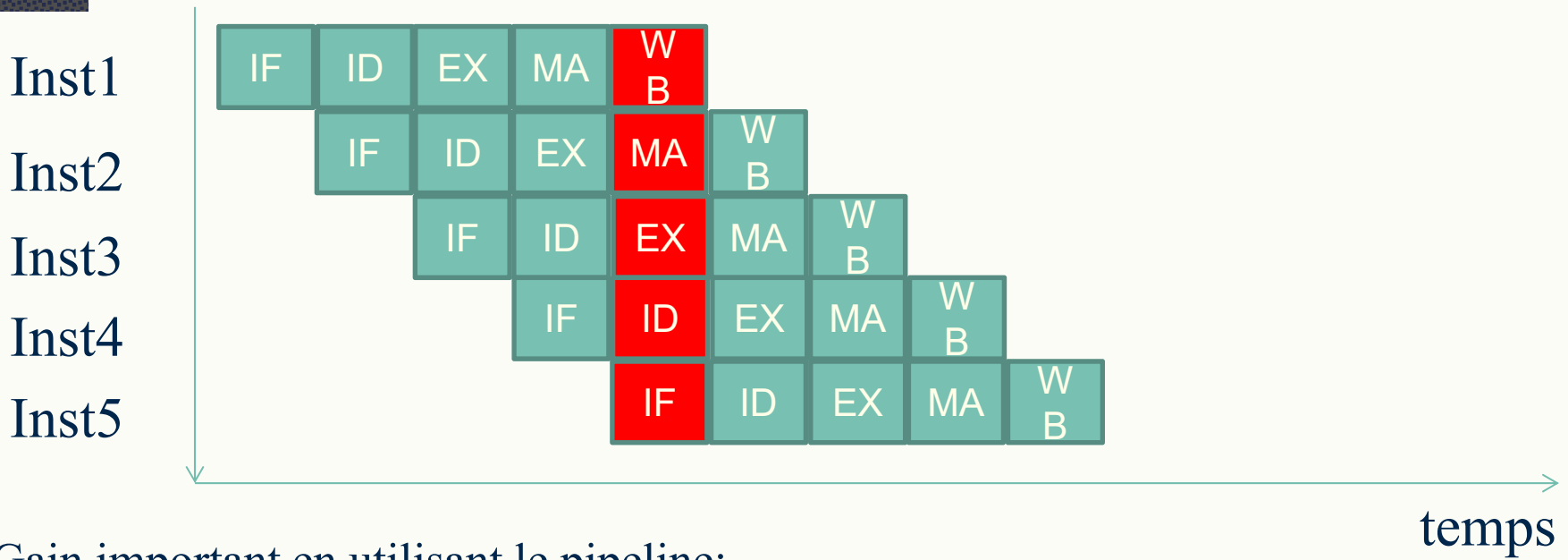
- 3 instructions sont traitées en parallèle en 7 cycles d'horloge:

⇒ Permet de multiplier le débit avec lequel les instructions sont exécutées par le processeur.

⇒ Le gain se situe au niveau du débit, le temps de traitement d'une instruction reste le même.



Exemple pipeline à 5 étages



Gain important en utilisant le pipeline:

- Sans : exécution séquentielle de 2 instructions en 10 cycles.
- Avec : exécution parallèle de 5 instructions en 9 cycles.

L'idéal est d'équilibrer la longueur des étages du pipeline, sinon les étages les plus rapides attendent les plus lents.

⇒ Temps de passage dans chaque étape identique (ou très proche).

Ici, nous pouvons voir que chaque élément du processeur est en utilisation.

⇒ aucun élément au repos.

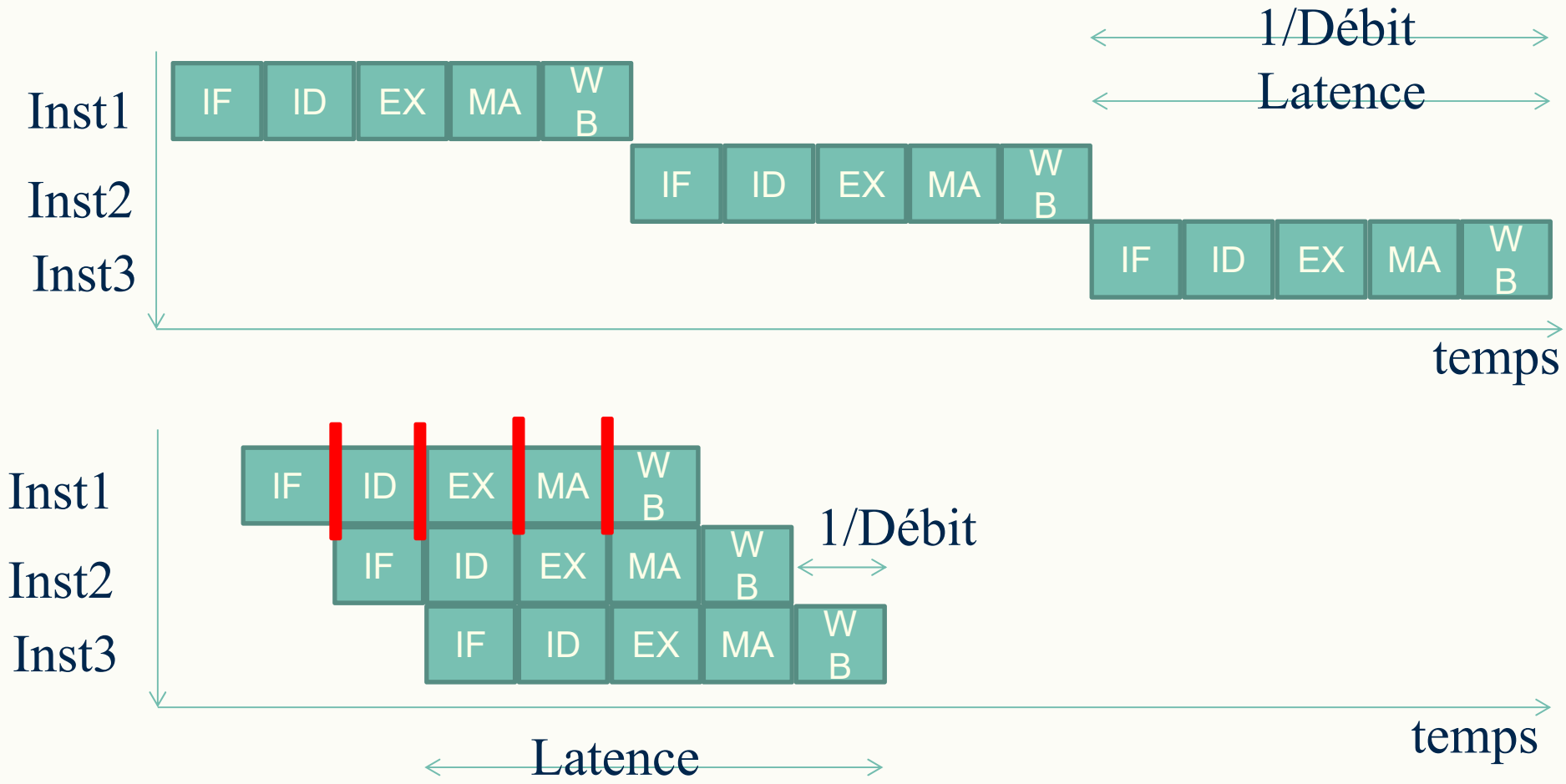
Terminologie

- Cycle time (temps de cycle)
  - Temps de traitement de chaque étape
  - = une ou plusieurs période d'horloge
- Flowthrough time (latence)
  - temps de latence, d'attente du premier résultat
  - (n cycles)
- Pipeline throughput (débit)
  - nombre d'instructions par seconde traitées dans le pipeline
  - typiquement égal à 1/cycle time

Calcul de performances

- Deux paramètres sont souvent utilisé pour mesurer la performance d'un processeur:
  - La **latence**: Le temps de réponse ou temps d'exécution d'une certaine tâche:
    - Temps écoulé entre le début et la fin d'exécution de la tâche (instruction).
  - Le **débit**: quantité totale de travail réalisé dans un certain emps.
    - Nombre d'opérations (tâches, instructions) exécutées par unités de temps.
- L'amélioration du temps de réponse implique toujours une amélioration du débit. Le contraire n'est pas toujours vrai.
  - Par exemple, augmenter le nombre de processeurs augmentera le débit mais pas le temps d'exécution.
- **L'accélération**: nombre de fois plus vite qu'en séquentiel.

Calcul de performances



Cas réel = insertion de registres intermédiaires entre les étages

Calcul de performances

- Calcul du **temps total d'exécution des instructions**:
  - Si T_e , le temps d'exécution d'un élément,
  - n le nombre d'éléments que compose une instruction
  - T_p , le temps d'exécution d'une instruction
  - m le nombre d'instruction à effectuer
  - Et T_t , le temps total

- **Séquentiel**

  $\Rightarrow T_t = m * T_p$, avec $T_p = n * T_e$

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

- **Pipeline**

  $\Rightarrow T_t = T_p + (m-1) * T_e$ Fin de la première instruction à $T_p = n * T_e$

  $\Rightarrow T_t = n * T_e + (m-1) * T_e$ Toutes les unités de temps suivantes

  $\Rightarrow \boldsymbol{T_t = T_e * (n + m - 1)}$ => fin d'une nouvelle instruction.

Si m est beaucoup plus grand que n, alors on peut considérer $\boldsymbol{T_t \approx m * T_e}$

Calcul de performances

- Calcul de **l'accélération**:

$$A = T_{seq}/T_{pip}$$

$$A = \frac{m*n*T_e}{(n+m-1)*T_e}$$

$$\boldsymbol{A = \frac{m*n}{n+m-1}}$$

Si m très grand alors $\boldsymbol{A = n}$

$\Rightarrow$ **cas idéal,** $\boldsymbol{T_{pip} = \frac{T_{seq}}{profondeur\ du\ pipeline}}$

- Calcul du **débit** du pipeline
  - $D = \frac{1}{T_{exi}}$, avec $T_{exi} = \frac{T_t}{m}$, temps exécution total ramené à 1 instruction
  - $D = \frac{m}{(n+m-1)*T_e}$
  - $\boldsymbol{D \approx \frac{1}{T_e}}$

$\Rightarrow \boldsymbol{\frac{1}{T_e}}$ **est souvent appelé le cycle machine**

Exemple de profondeur

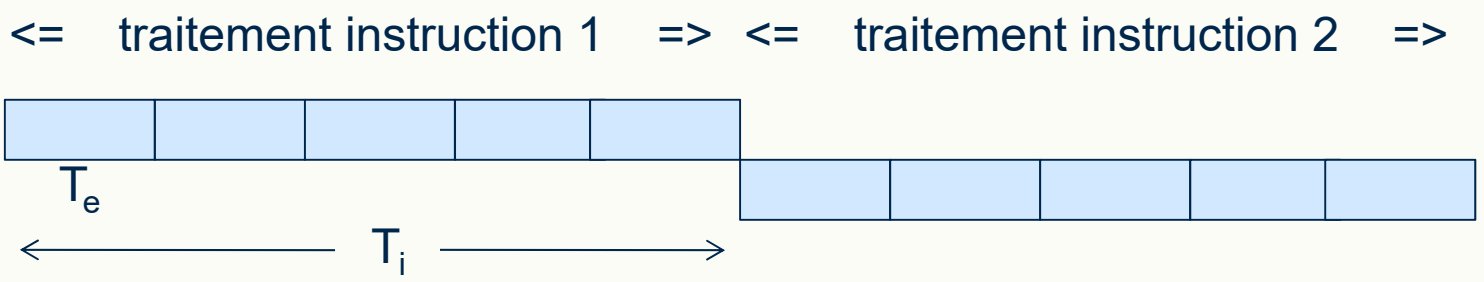
- Exemples de profondeur de pipeline (nombre d'étages d'un pipeline)

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

Traitement d'une instruction

- **Le traitement d'une instruction s'effectue généralement en plusieurs étapes**

- **Exemple sans pipeline**



- **soit n le nombre d'étapes par instruction**
- **et Te le temps de traitement d'un étape** $T_i = nT_e$
- **pour m instructions** $T_t = m\,T_i = mnT_e$

Traitement avec pipeline (rappel)

- **Exemple avec pipeline**

  <= traitement instruction 1 =>

  <= traitement instruction 2 =>

- **pour m instructions**

  $T_t = T_i + (m-1)T_e = nT_e + (m-1)T_e = (n+m-1)T_e$

- **pour un grand nombre d'instructions**

  $T_t = m\ T_e$ **soit n fois moins que le traitement sans pipeline**

Image Wikipédia

Traitement des instructions

- **Suivant les types de processeurs :**
  - **Pipeline à 3 niveaux :**
    - **Fetch, decode , execute**
  - **Pipeline à 5 niveaux (processeurs ARM 9, MIPS) :**
    - **IF, ID, EX, MEM, WB**
  - **Jusqu'à 25 niveaux ! (20 niveaux pour le Pentium 4)**

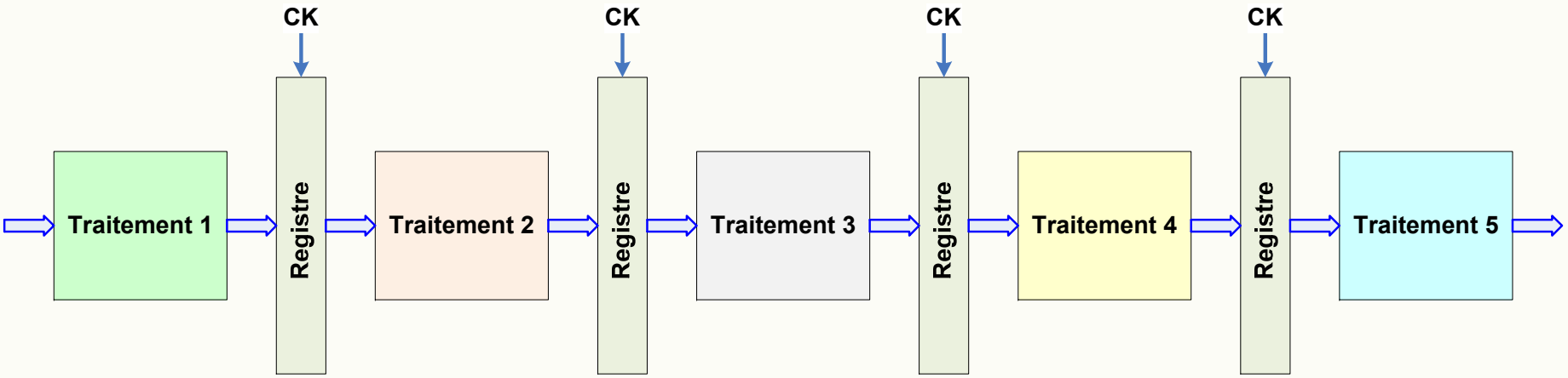
Pipeline à 5 niveaux : phases

- *Phase IF (Instruction Fetch) :* **recherche de l'instruction**
- *Phase ID (Instruction Decode):* **décodage de l'instruction et lecture des registres opérandes**
- *Phase EX (Execution):* **exécution de l'opération ou calcul de l'adresse de mémoire**
- *Phase MEM (Memory):* **accès de la mémoire ou écriture dans le PC de l'adresse de saut**
- *Phase WB (Write Back):* **écriture dans un registre du résultat de l'opération**

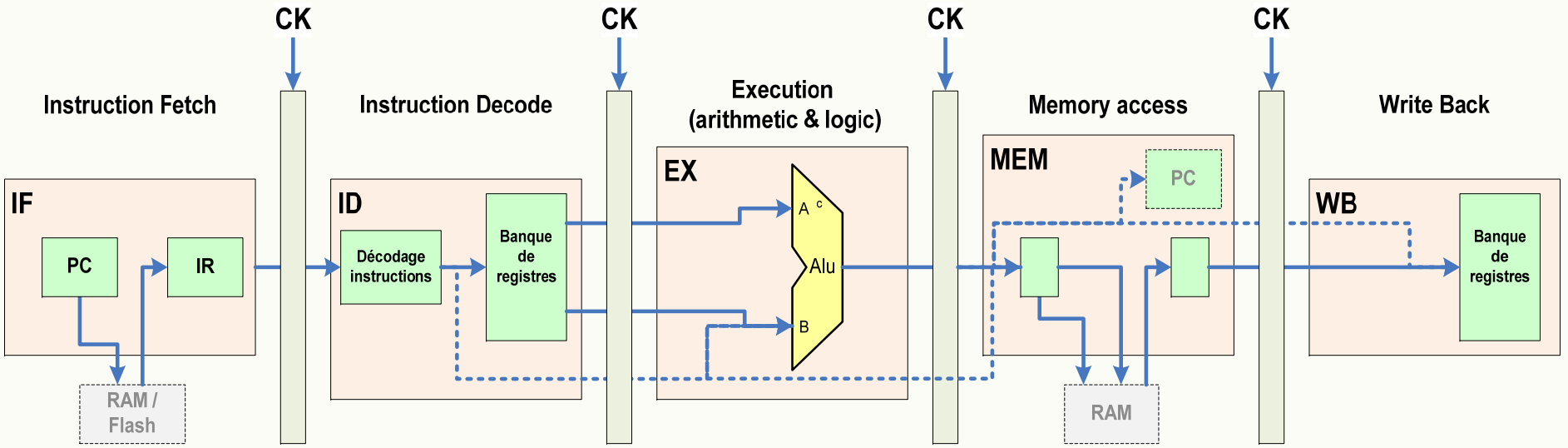
Architecture de pipeline

- Exemple de pipeline à 5 étages ou niveaux

- Le traitement s'effectue en 5 étapes (ou 5 cycles)
- 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

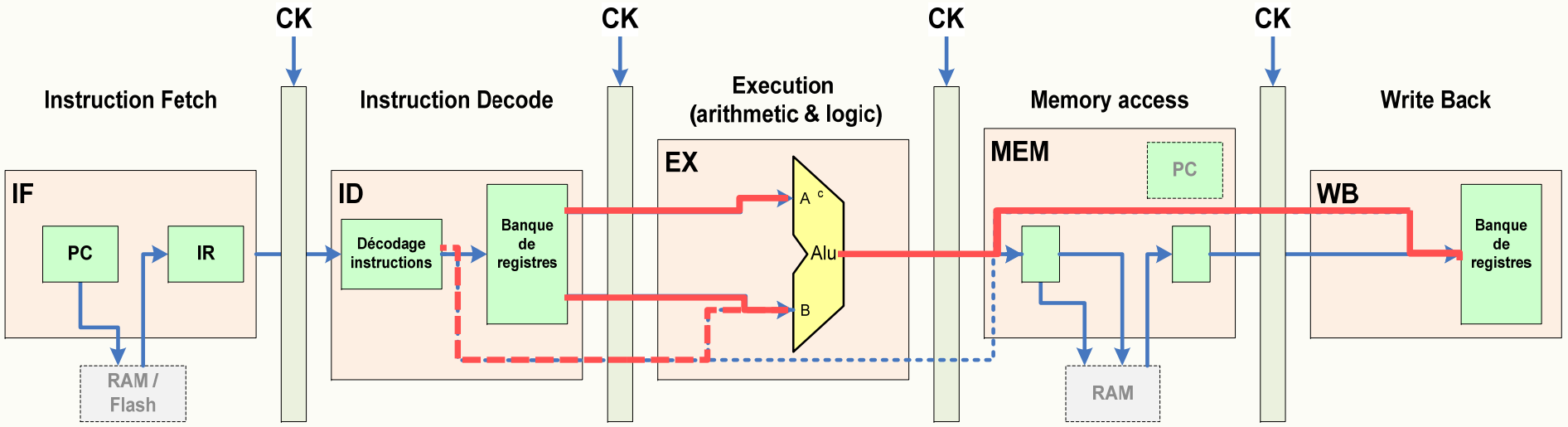


Pipeline à 5 niveaux : architecture



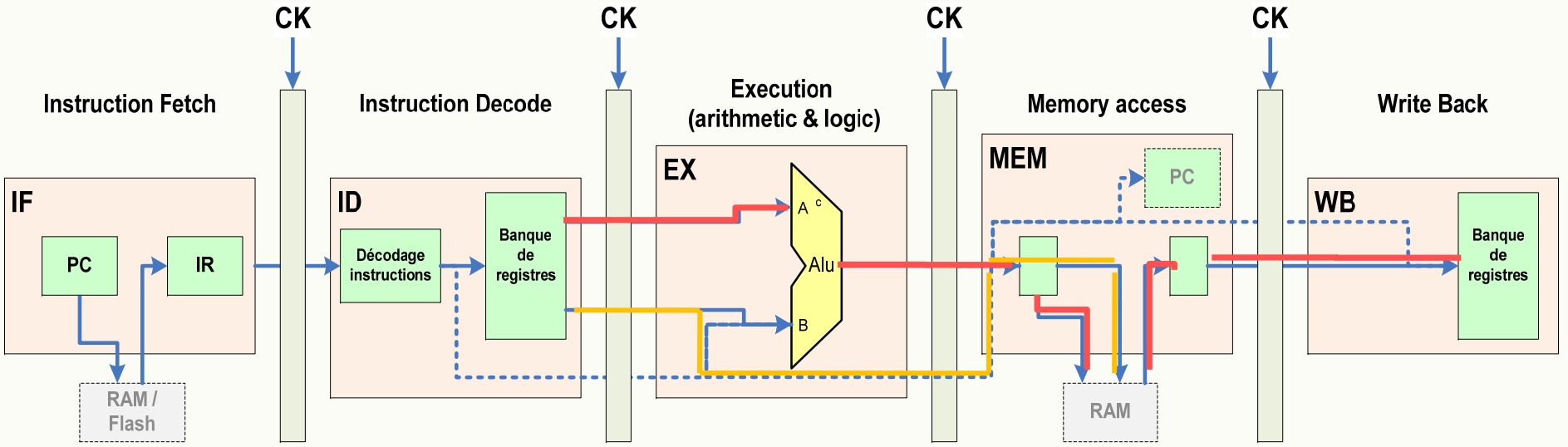
Traitement des instructions (1/3)

- **Adressage direct (par registre) :**
  - ADD Rn, Rm, Rp => Rn = Rm + Rp
  - SUB Rn, Rm, Rp => Rn = Rm – Rp
- **Adressage direct & immédiat :**
  - ADD Rn, Rm, #P => Rn = Rm + P



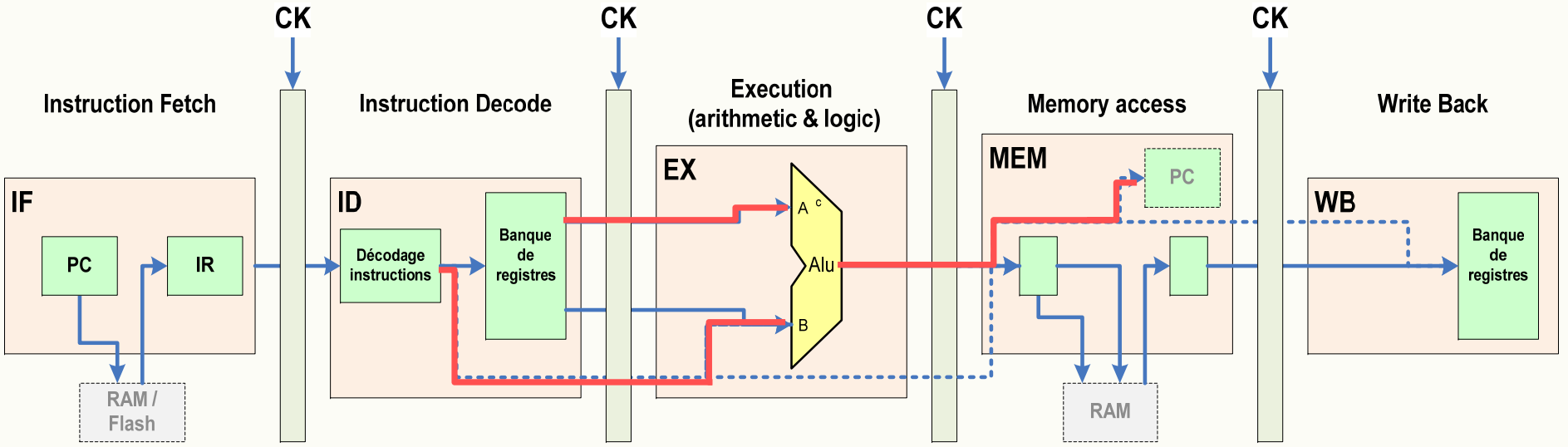
Traitement des instructions (2/3)

- **Transferts mémoire / registre:**
  - **LDR Rn, [Rp] Rn ← mem [Rp]**
  - **STR Rn, [Rp] mem [Rp] ← Rn**



Traitement des instructions (3/3)

- **Instructions de contrôle:**
  - **BEQ LABEL** **if résultat = 0 jump to LABEL**

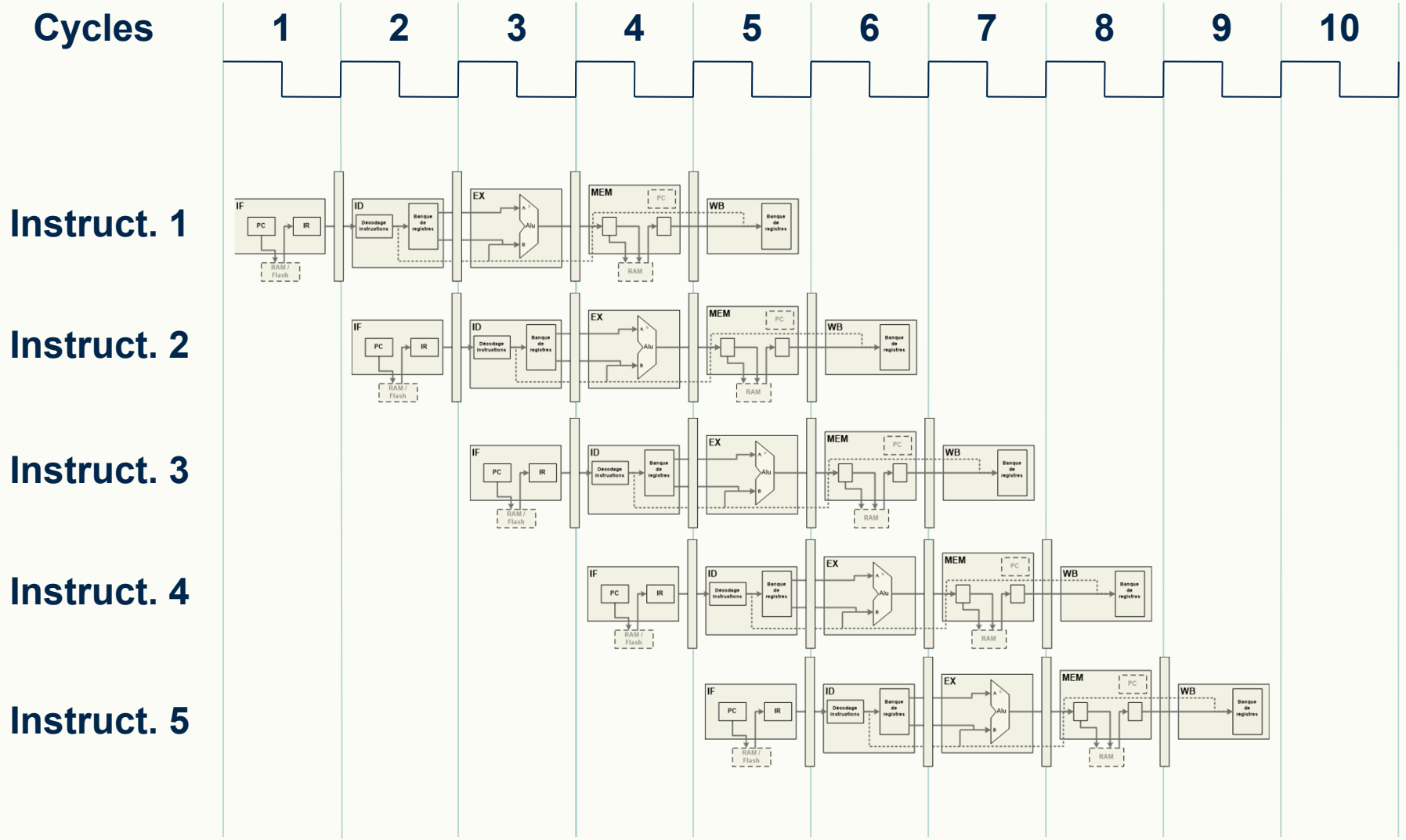


Pipeline à 5 niveaux

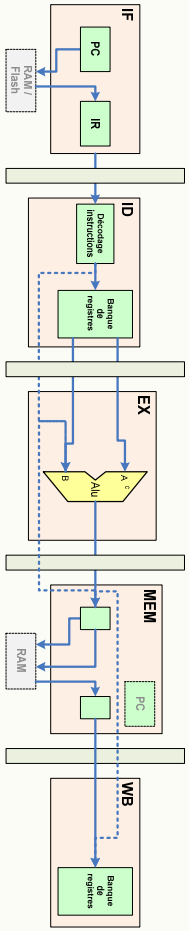
Cycles	1	2	3	4	5	6	7	8	9	10
Instruct. 1	IF	ID	EX	MEM	WB					
Instruct. 2		IF	ID	EX	MEM	WB				
Instruct. 3			IF	ID	EX	MEM	WB			
Instruct. 4				IF	ID	EX	MEM	WB		
Instruct. 5					IF	ID	EX	MEM	WB	
Instruct. 6						IF	ID	EX	MEM	WB

IF	Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6				
ID		Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6			
EX			Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6		
MEM				Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6	
WB					Inst.1	Inst.2	Inst.3	Inst.4	Inst.5	Inst.6

Pipeline à 5 niveaux



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

Cours ASP

ALEAS DE PIPELINE

Problèmes des architectures pipelinées: les aléas (hazard)

- 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

Dépendances de données

- Dépendances entre instructions = même opérandes
- Les dépendances peuvent entrainer des aléas

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

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

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

Exemples de dépendances

- ***Ces dépendances n'entrainent pas forcément des aléas***

  - ***RAW (read after write):***

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

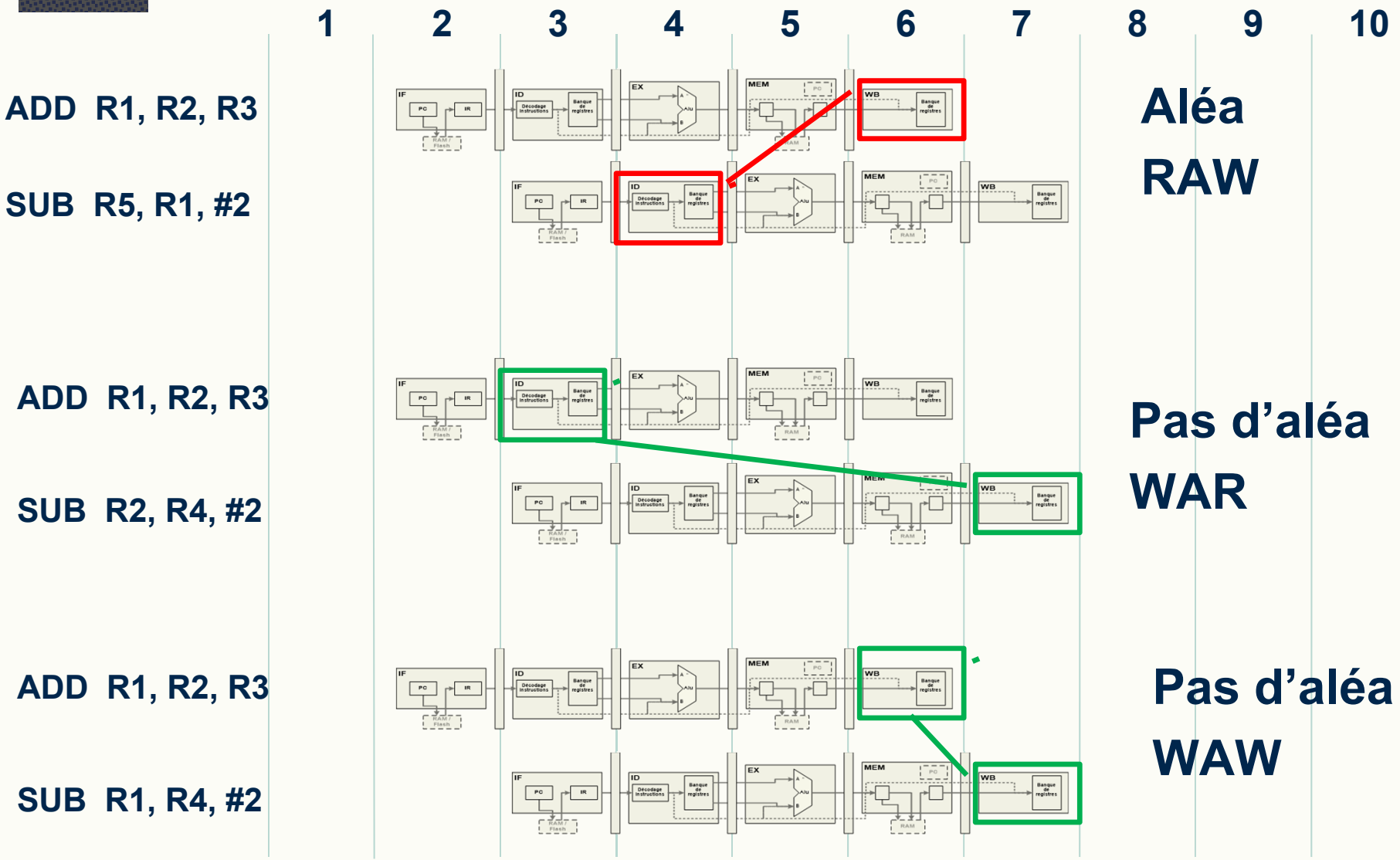
  - ***WAR (write after read):***

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

  - ***WAW (write after write):***

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

Aléas de données : exemples

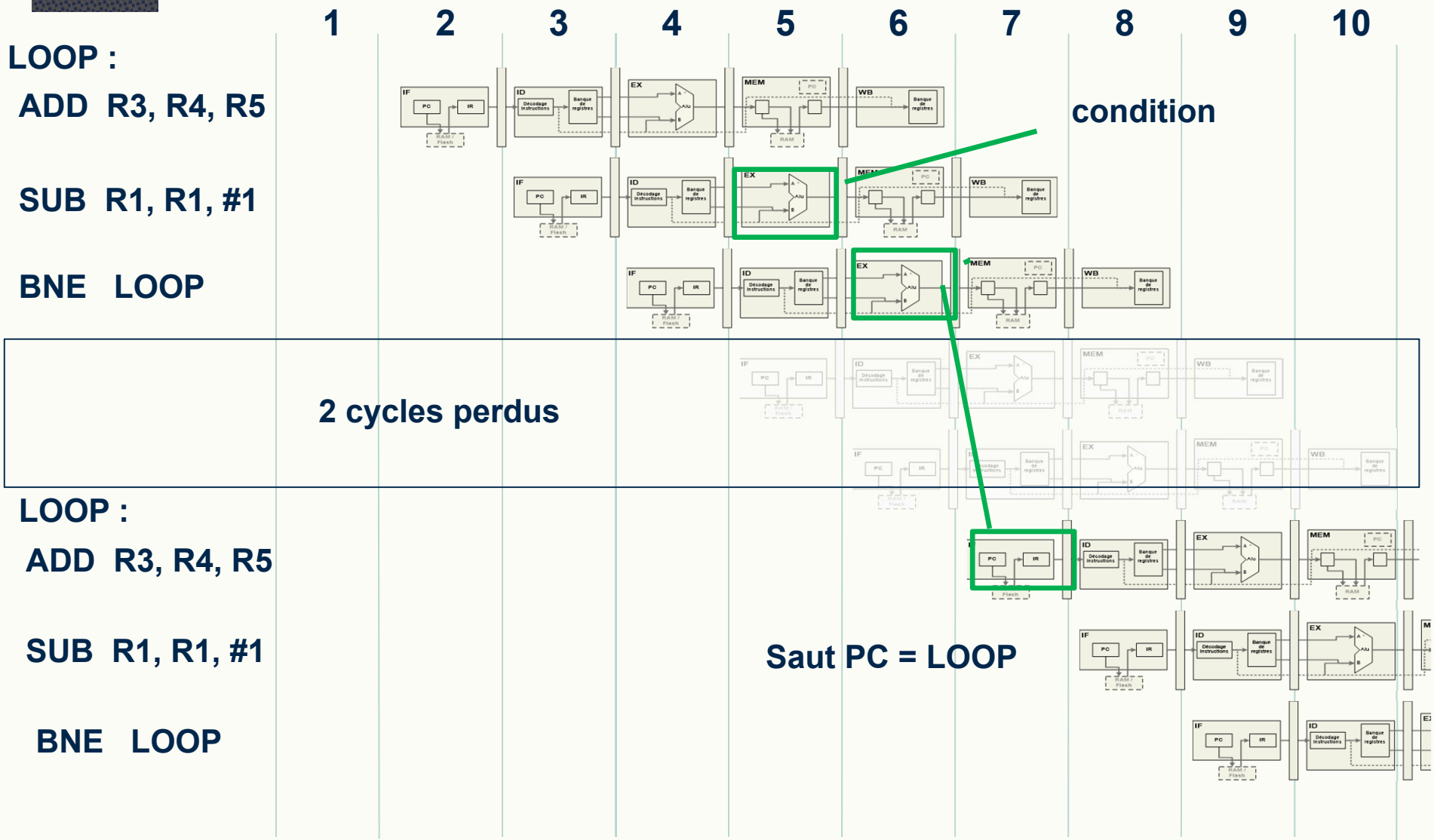


Exercice : dépendances et aléas de pipeline

- Indiquez les dépendances de types RAW, WAR et 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 contrôle : exemple ARM 9



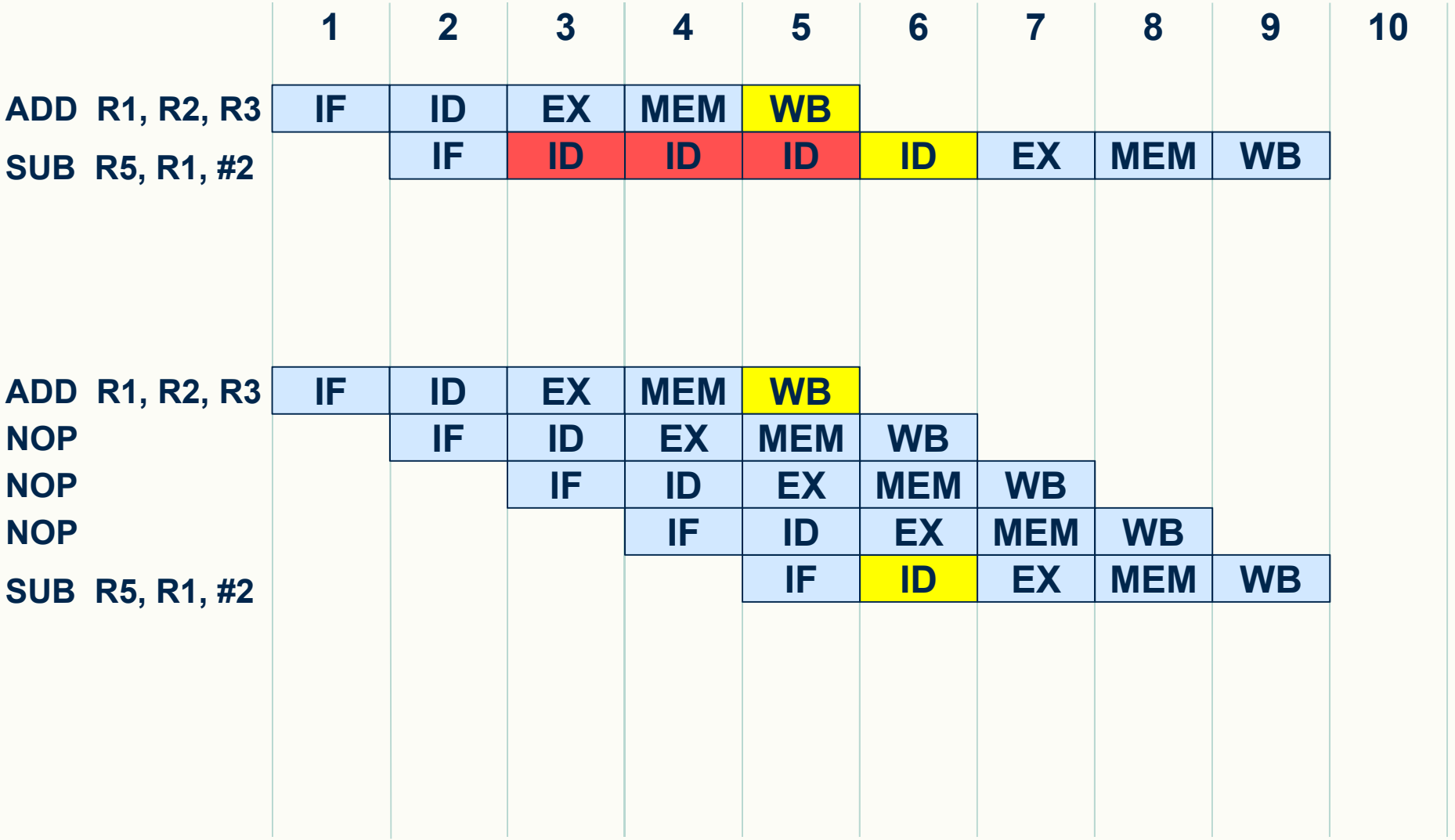
Cours ASP

RESOLUTION DES ALEAS

Résolution des problèmes d'aléas

- **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**

Arrèt de pipeline ou insertion de NOPs



Exercice : dépendances et aléas de pipeline

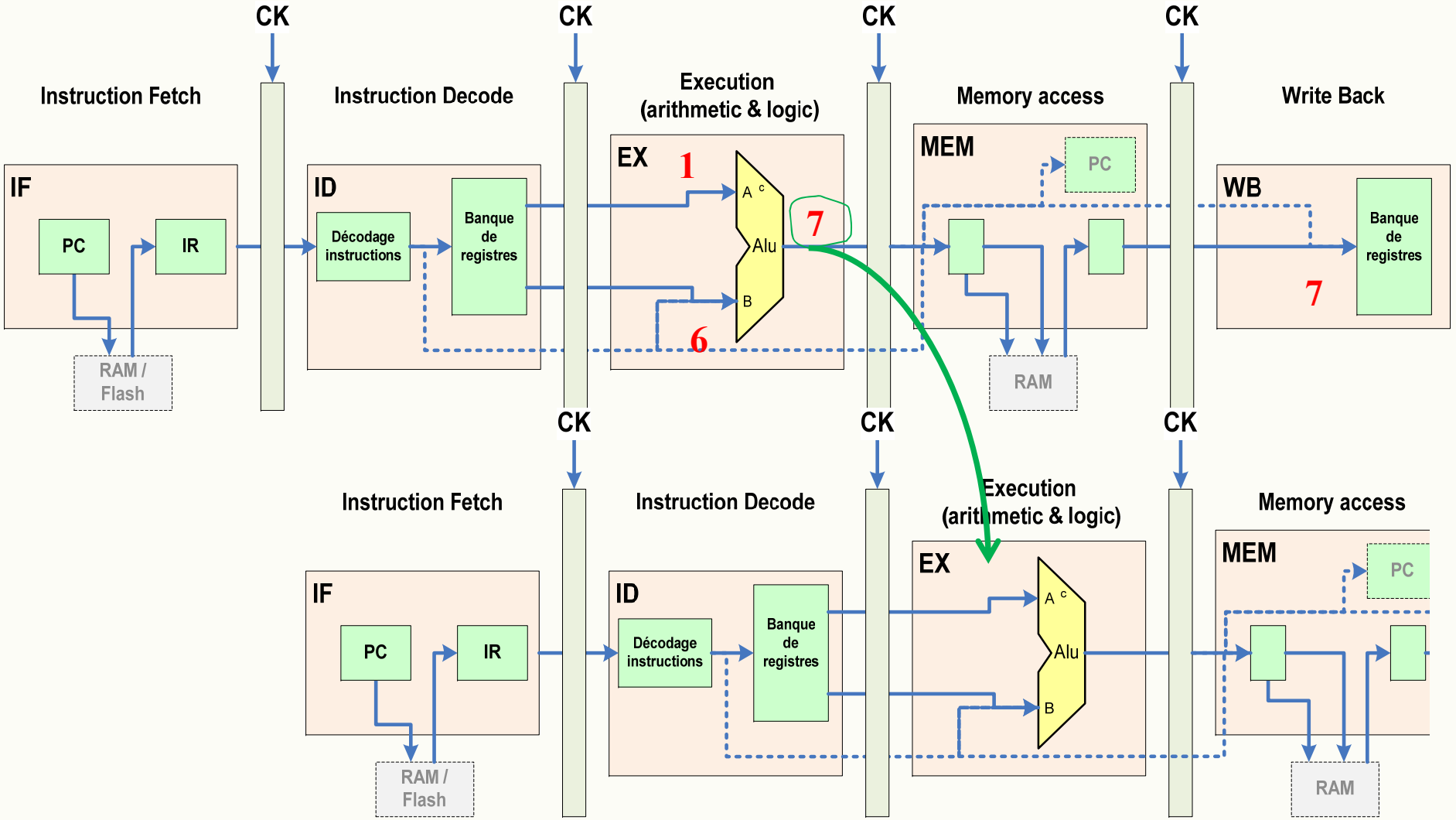
- Dessinez le chronogramme de l'exécution du code avec résolution des aléas par arrêt (hardware) du pipeline. Quel est l'IPC pour ces 9 instructions ?

```
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
```

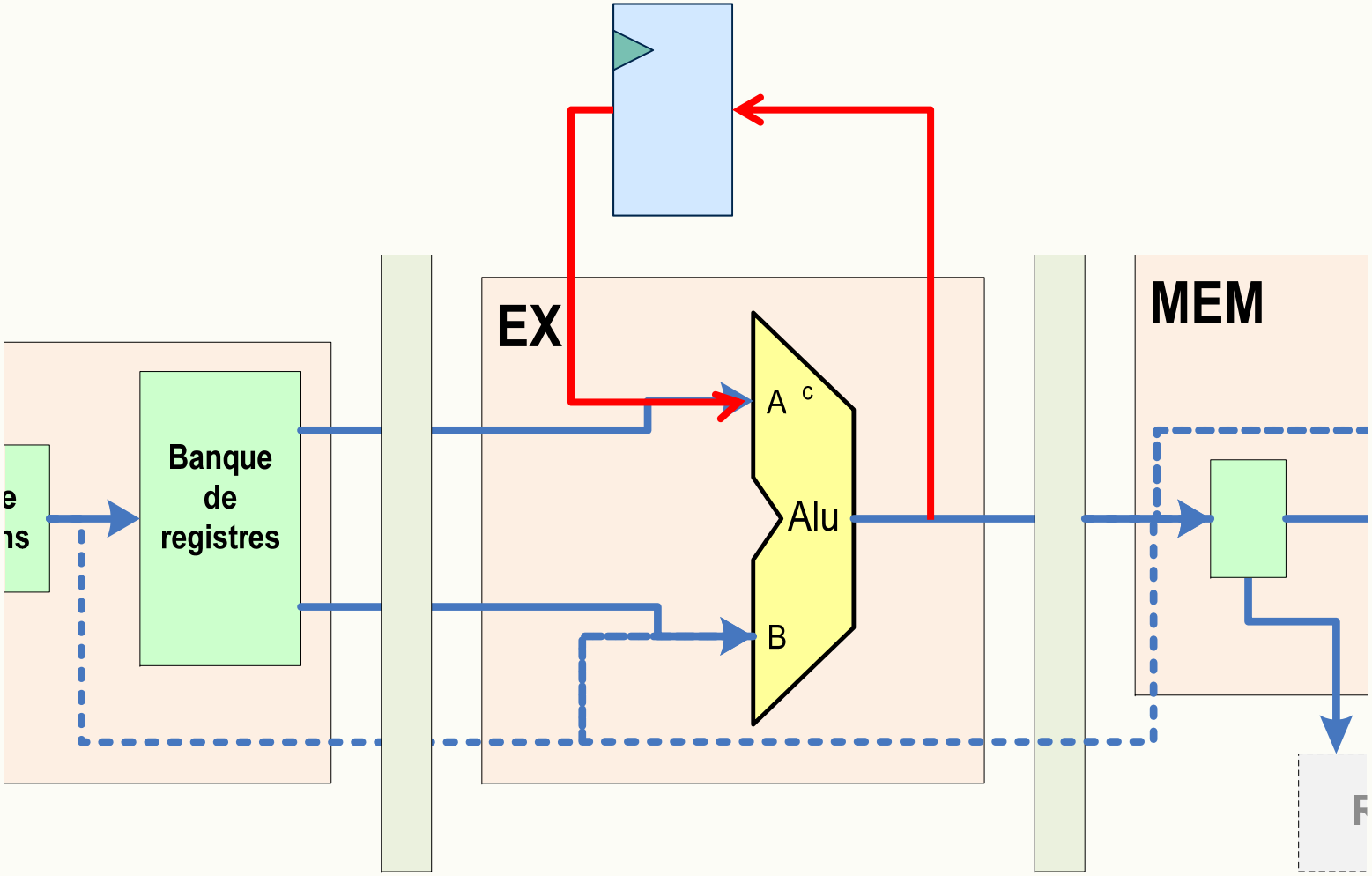
Méthode de résolution des aléas de données

- **Forwarding ou bypassing**
- **Utilisation des résultats à la sortie de l'ALU sans attendre le cycle WB**
- **Les opérandes sont stockés dans des registres**
- **Il faut stocker 3 résultats pour un pipeline à 5 étages**

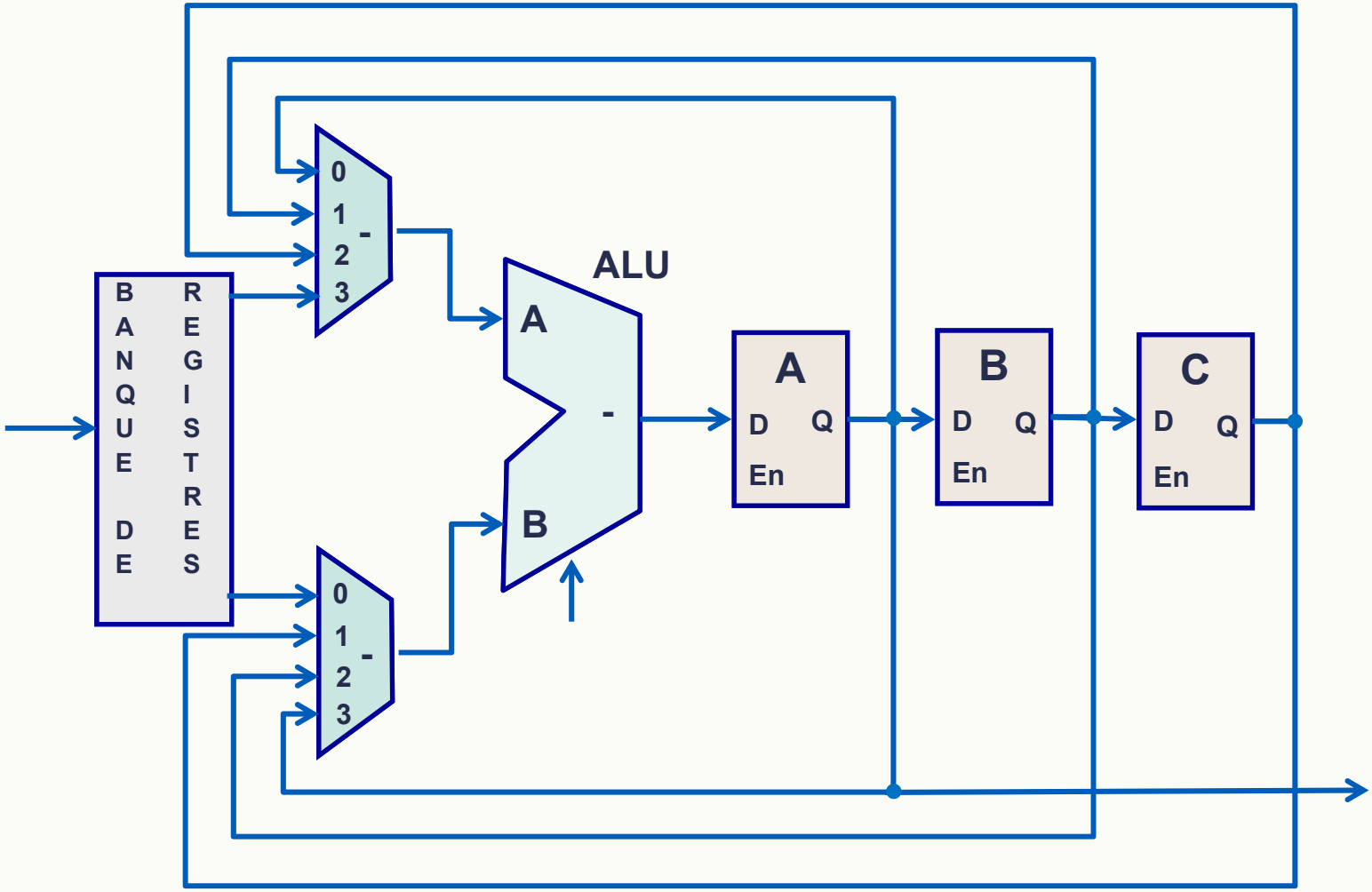
Forwarding (ou bypassing) (1/3)



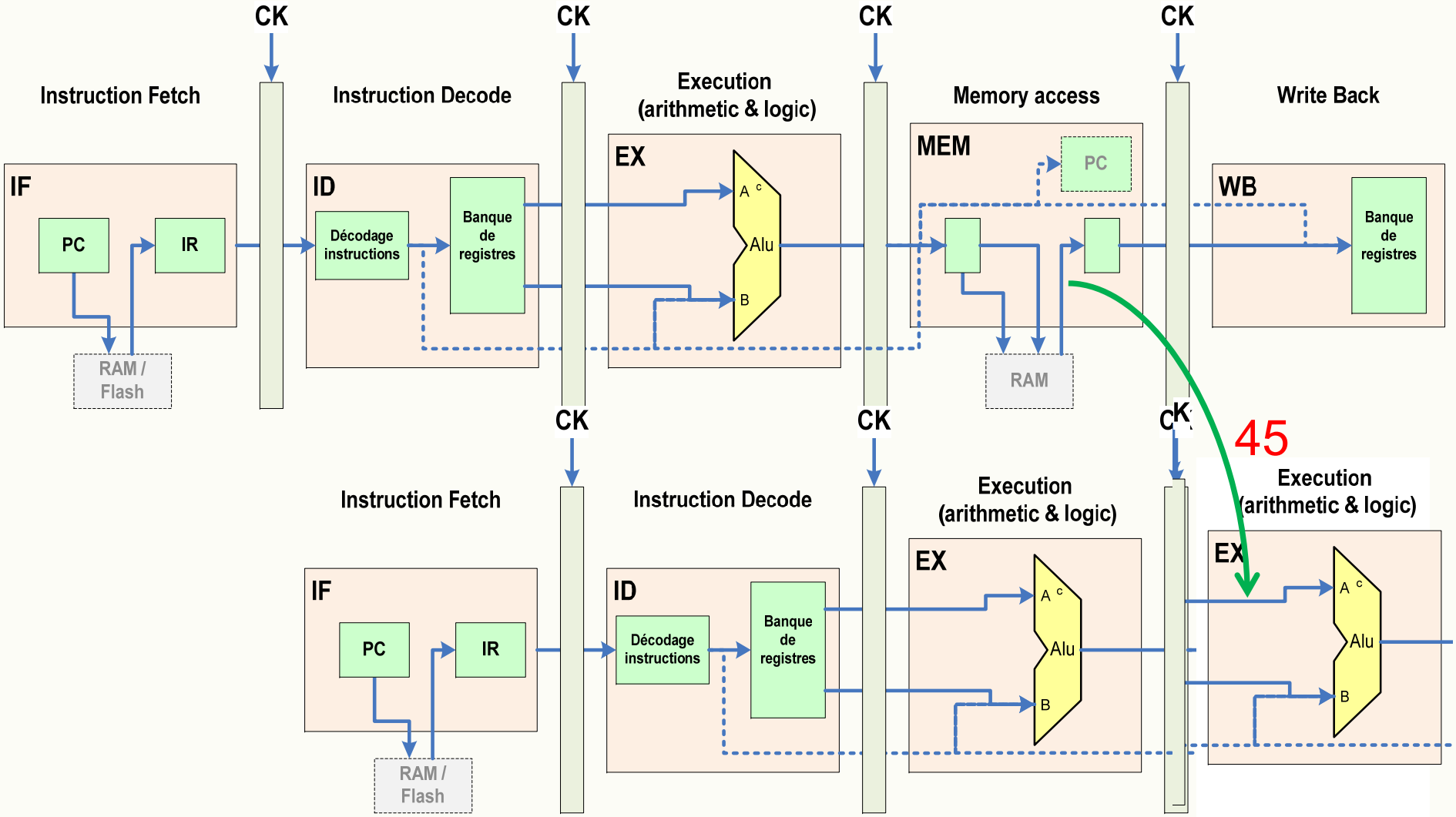
Forwarding (ou bypassing) (2/3)



Forwarding (ou bypassing) (3/3)



Forwarding : Cas particulier de Load (lecture en mémoire)



Autres méthodes de résolution des aléas de pipeline

- Aléas insolubles par les méthodes vues précédemment (excepté l'arrêt du pipeline) 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
  - Prédiction dynamique

Ordonnancement dynamique

- Technique utilisée pour éliminer tous les aléas de données (long pipeline ou pipeline hétérogène)

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

Scoreboarding

- Les instructions sont exécutées dans le désordre (pas dans l'ordre du programme)
- Les instructions sont lues dans la mémoire dans l'ordre du programme, les dépendances sont enregistrées, et les instructions sont stockées dans l'attente de la disponibilité des opérandes
- Le Scoreboarding est efficace pour les aléas RAW et moins pour les aléas WAR et WAW (résolus par arrét du pipeline).

Méthode de Tomasulo

- Différence essentielle avec le scoreboarding: les registres sont renommés (utilisation de registres temporaires).
- Ainsi le pipeline n'est pas arrêté par les aléas WAR ou WAW
- Les opérandes sont stockés dans des "reservation stations" qui contiennent également de nombreuses autres informations (tags)

Méthodes de résolution des aléas de contrôle

- Arrêt du pipeline pendant N instructions (cas de l'ARM 9) jusqu'à ce que l'adresse de l'instruction suivante soit connue.
- Exécution des instructions de branchement non pris avec annulation si le branchement est pris (ou l'inverse)
- Exécution des tests de branchement dans la section ID
- Changement d'ordre des instructions par le compilateur (combiné avec la méthode précédente)
- Prédiction dynamique

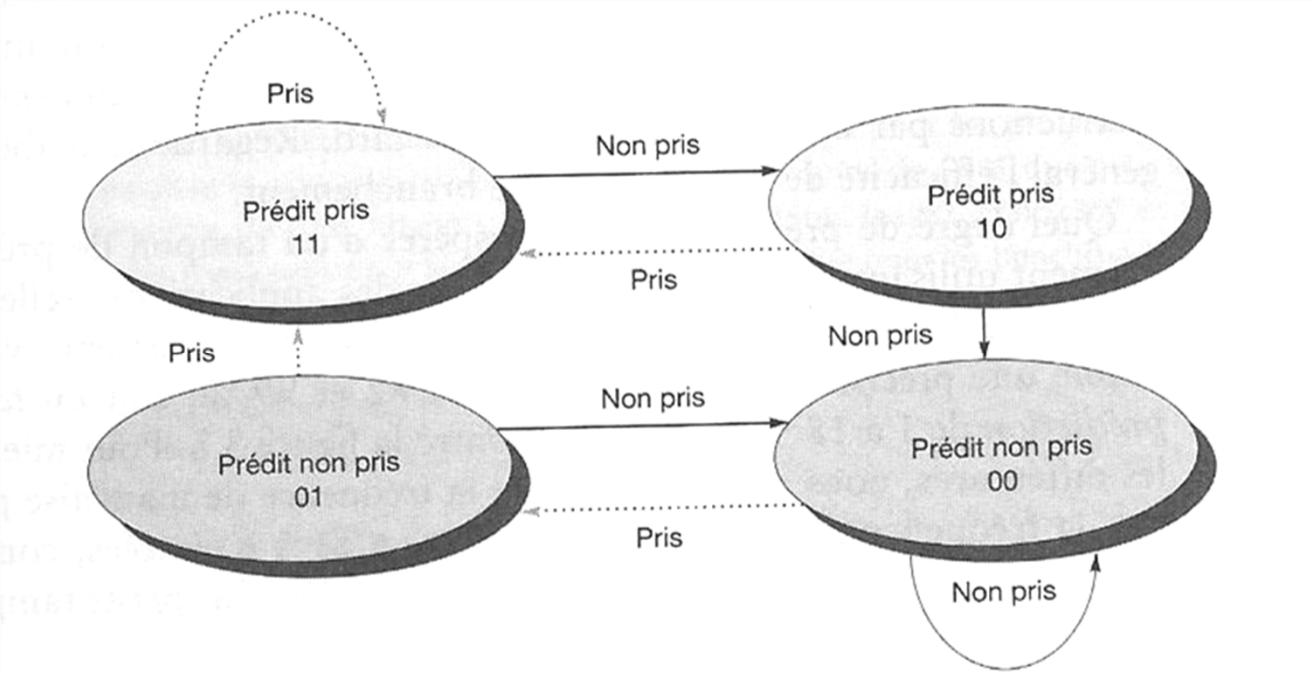
Prédiction dynamique (1/2)

- Technique utilisée pour éliminer les aléas de contrôle
- Permet d'anticiper les branchements en choisissant (entre pris et non pris) la probabilité la plus élevée
- Pour chaque instruction Branch, l'adresse de branchement la plus probable est stockée dans une table
- Le saut est effectué immédiatement, après l'instruction Branch, sans attendre le calcul de l'adresse de saut. En cas de prédiction incorrecte, plusieurs cycles sont perdus.

Prédiction dynamique (2/2)

- **Prédiction à partir d'une table d'historique des branchements**

- **Méthode simples : Prédiction à 1 bit (pris, non pris) ou à 2 bits(plus adapté aux boucles)**



Exercice : pipeline ARM à 5 étages

ARM9E-S

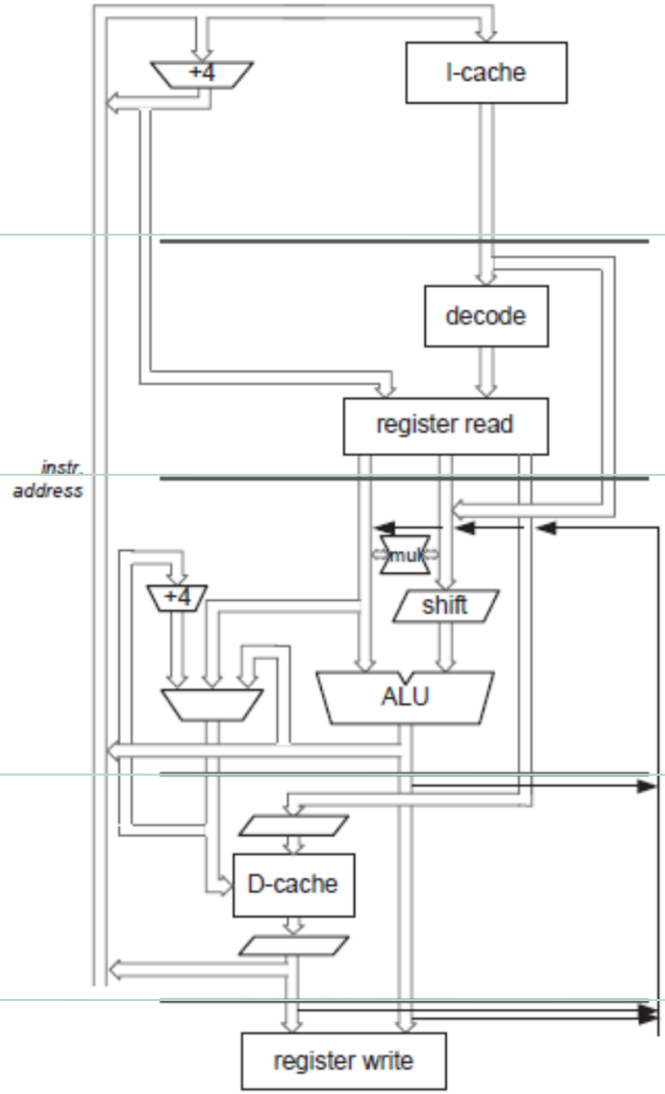
IF

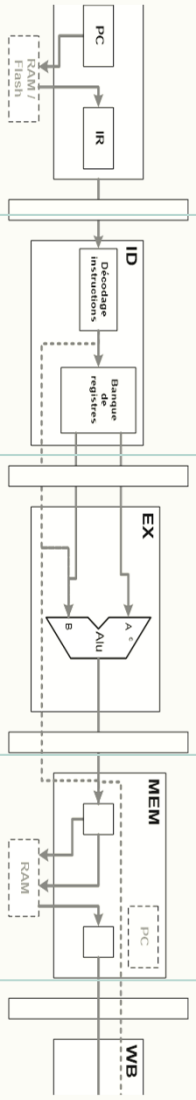
ID

EX

MEM

WB





Cours ASP

CORRIGÉ DES EXERCICES

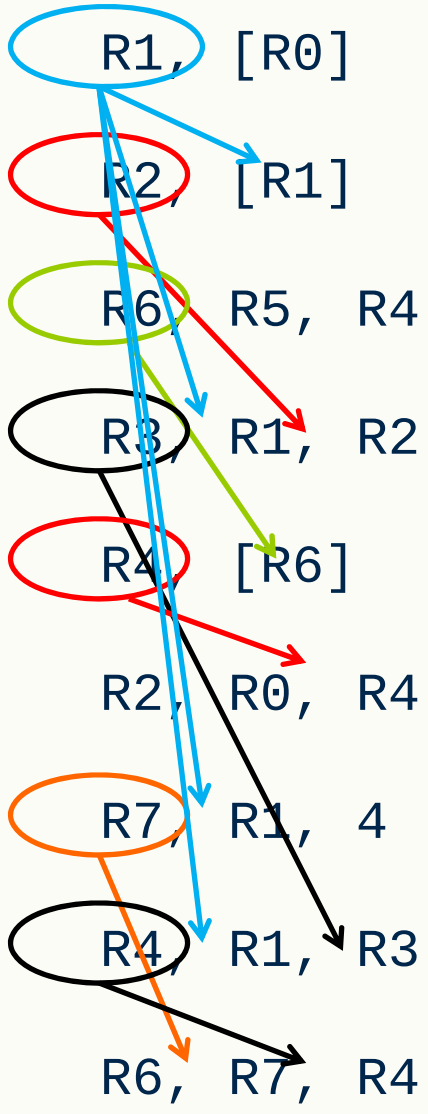
Corrigé exercice slide 5

- L'horloge du processeur est 400MHz
- Le pipeline est à 6 niveaux.
- Quel est le temps nécessaire pour traiter 1000 instructions, a) si le traitement n'est pas pipeliné, b) si le traitement est pipeliné ?

- P = 1/400 x 10-6 = 2.5 ns
- T = 1000 x 6 x 2.5 ns = 15 us
- T = (6+999) x 2.5 ns = 1005 x 2.5 ns = 2512.5 ns

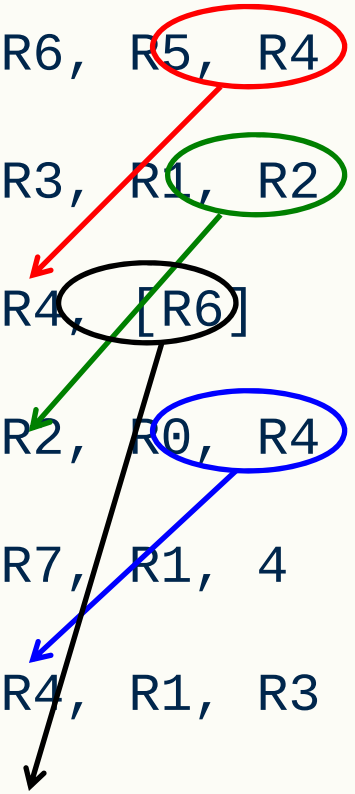
- *RAW (read after write):*

```
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
```



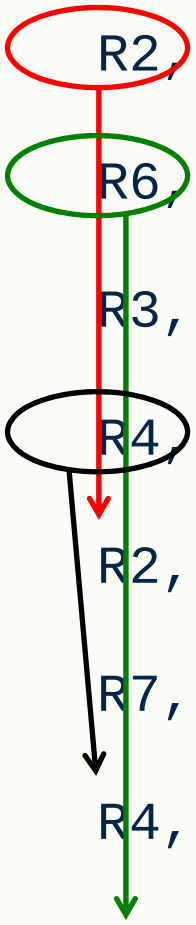
- *WAR(write after read):*

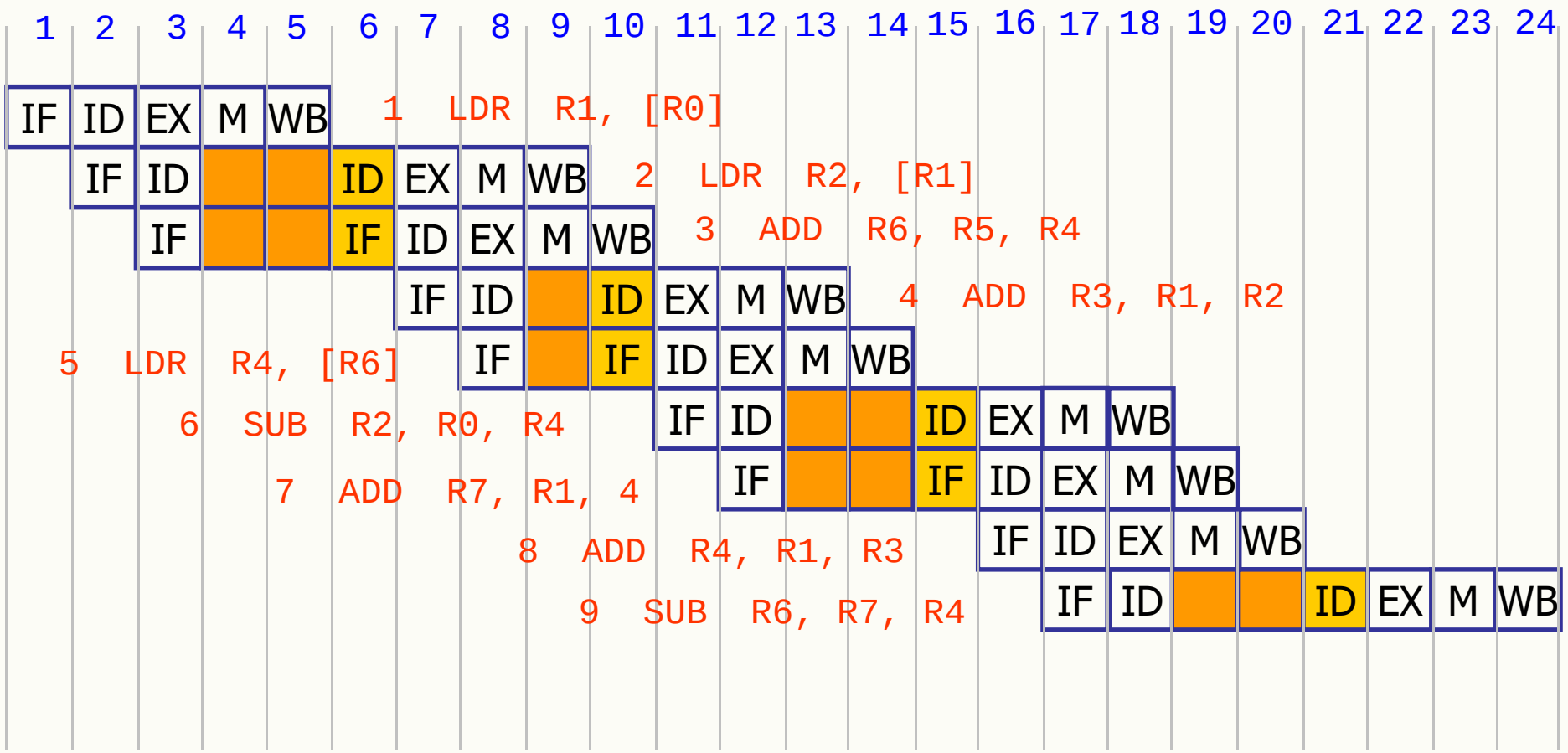
```
1  LDR     R1, [R0]
2  LDR     R2, [R1]
3  ADD     R6, R5, R4
4  ADD     R3, R1, R2
5  LDR     R4, [R6]
6  SUB     R2, R0, R4
7  ADD     R7, R1, 4
8  ADD     R4, R1, R3
9  SUB     R6, R7, R4
```



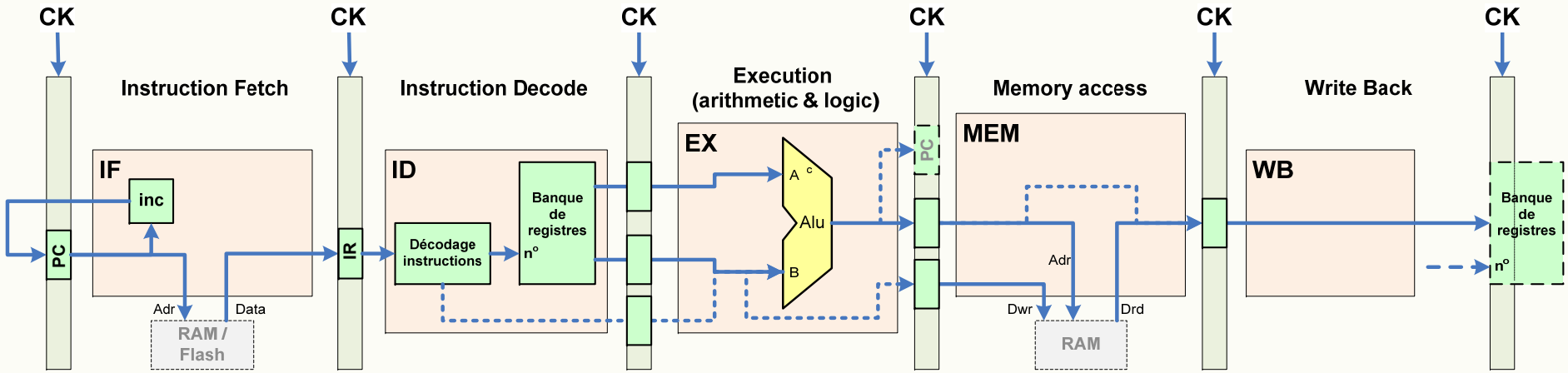
- *WAW(write after write):*

```
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
```

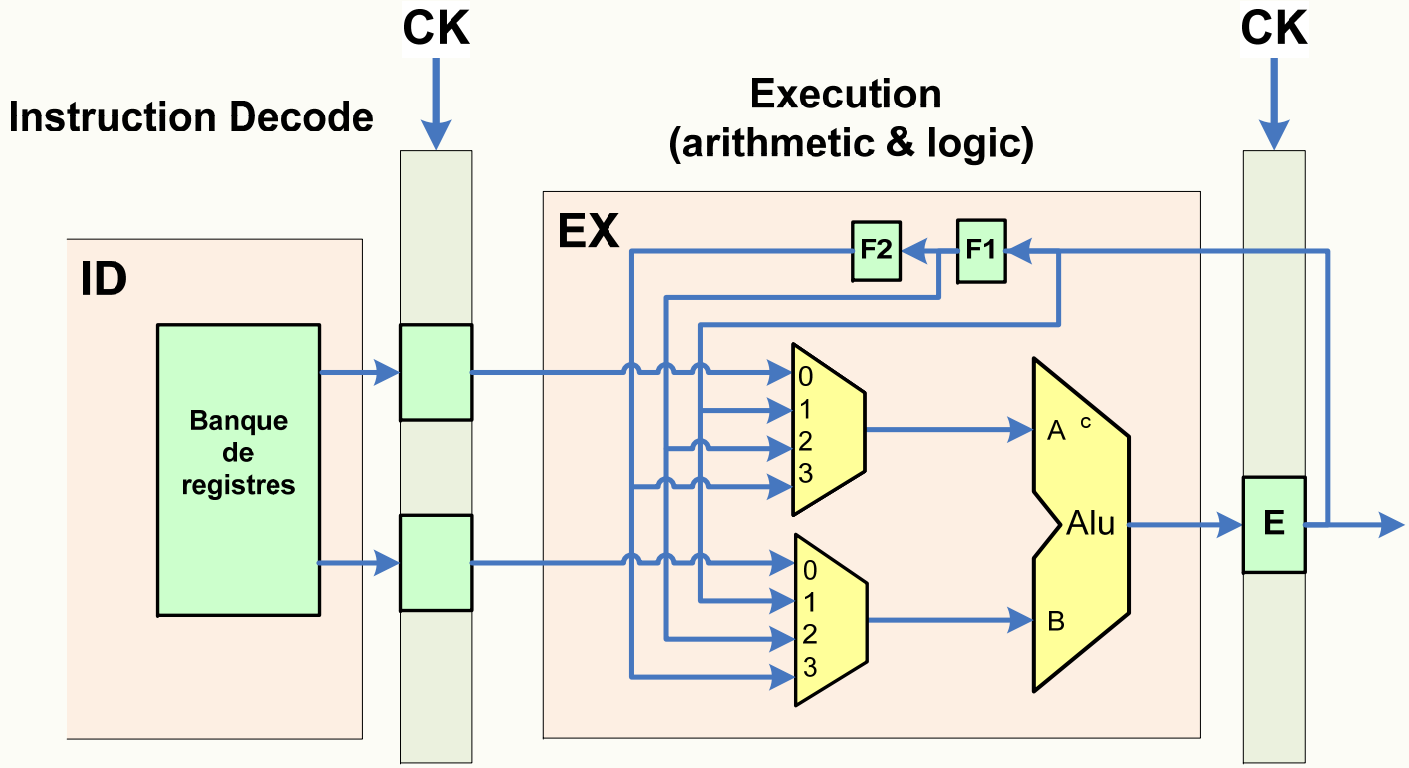



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
IF ID EX M WB
1 LDR R1, [R0]
IF ID EX M WB
2 LDR R2, [R1]
IF ID EX M WB
3 ADD R6, R5, R4
IF ID EX M WB
4 ADD R3, R1, R2
IF ID EX M WB
5 LDR R4, [R6]
IF ID EX M WB
6 SUB R2, R0, R4
IF ID EX M WB
7 ADD R7, R1, 4
IF ID EX M WB
8 ADD R4, R1, R3
IF ID EX M WB
9 SUB R6, R7, R4

Annexe 1 :Bloc diagramme détaillé (avec registres)



Annexe 2 :) Détails du mécanisme de forwarding dans un pipeline

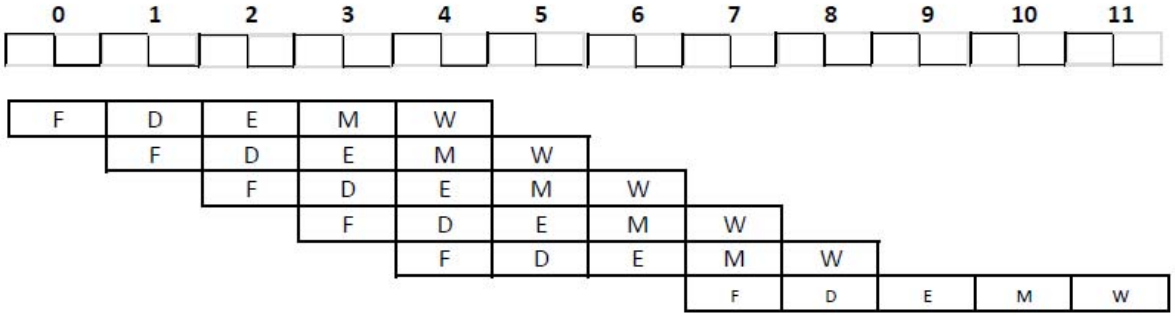


Annexe 3 : Chronogramme 1

Cours ARO2 2015
Exemple de pipeline Version 2.1

r1 = 0x28B4, r2 = 8, r3 = 5,
M[0x28BA] = 0xABCD
LOOP:

16:	1c9c	adds	r4, r3, #2
18:	189d	adds	r5, r3, r2
1a:	88c8	ldrh	r0, [r1, #3*2]
1c:	810A	strh	r2, [r1, #4*2]
1e:	e7fa	b.n	16 <LOOP>
16:	1c9c	adds	r4, r3, #2



		16	18	1A	1C	1E	20	22	16	18	1A	1C	1E
	PC		1C9C	189D	88C8	810A	E7FA			1C9C	189D	88C8	810A
	FETCH			5	5	28B4	28B4	22			5	5	28B4
DECODE	OP1			2	8	6	8	"-6*2"			3	8	6
DECODE	OP2						8(r2)						
DECODE	D(OP3)												
	EXECUTE				7	13	28BA	28BC	16			7	13
	MEMORY					7	13	ABCD(rd)	write				7
	WRITE BACK						r4 = 7	r5=13	r0=ABCD				

Annexe 3 : Chronogramme 2 (bypass)

Cours ARO2 2015 **Version 2.1**

Exemple de pipeline avec bypass

r4 = 0x12, r3 = 0x28B2,
M[0x28BA] = 0xABCD

				0	1	2	3	4	5	6	7	8	9	10	11
16:	1c9a	adds	r2, r3, #2	F	D	E	M	W							
18:	18A5	adds	r5, r4, r2		F	D	E	M	W						
1a:	88d0	ldrh	r0, [r2, #6]			F	D	E	M	W					
1c:	1d01	adds	r1, r0, #4				F	D	E	M	W				
			PC	16	18	1A	1C	1E	20	22	24	26	28	2A	2B
			FETCH		1C9A	18A5	88D0	1D81							
		DECODE	OP1			28B2	12	rB=28B4	r0 =?	rA = ABCD					
			OP2			2	rA=28B4	6	4	4					
			D(OP3)												
			EXECUTE				28B4	28C6	28BA		ABD1				
			MEMORY					28B4	28C6	ABCD(rd)		ABD1			
			WRITE BACK						r2=28B4	r5=28C6	r0=ABCD		r1=ABD1		
			REG bypass A			xxxx	28B4	28C6	28BA	ABCD	ABD1	xxxx	xxxx	xxxx	
			REG bypass B			xxxx	xxxx	28B4	28C6	28BA	ABCD	ABD1	xxxx	xxxx	
			REG bypass C			xxxx	xxxx	xxxx	28B4	28C6	28BA	ABCD	ABD1	xxxx	