Architecture des ordinateurs
Pipeline

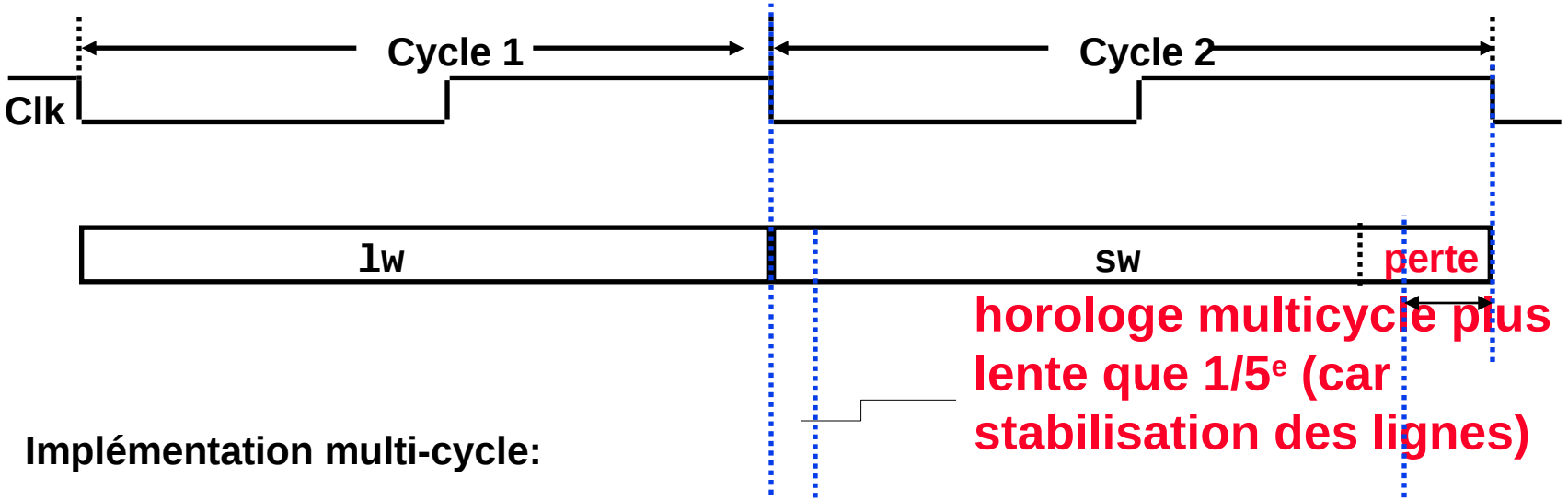
Rappel : motivation multicycle

Les instructions n'utilisent pas toutes les mêmes parties du chemin de données :

- format R / branch : pas d'accès à la mémoire de données
- store : pas d'écriture de registre

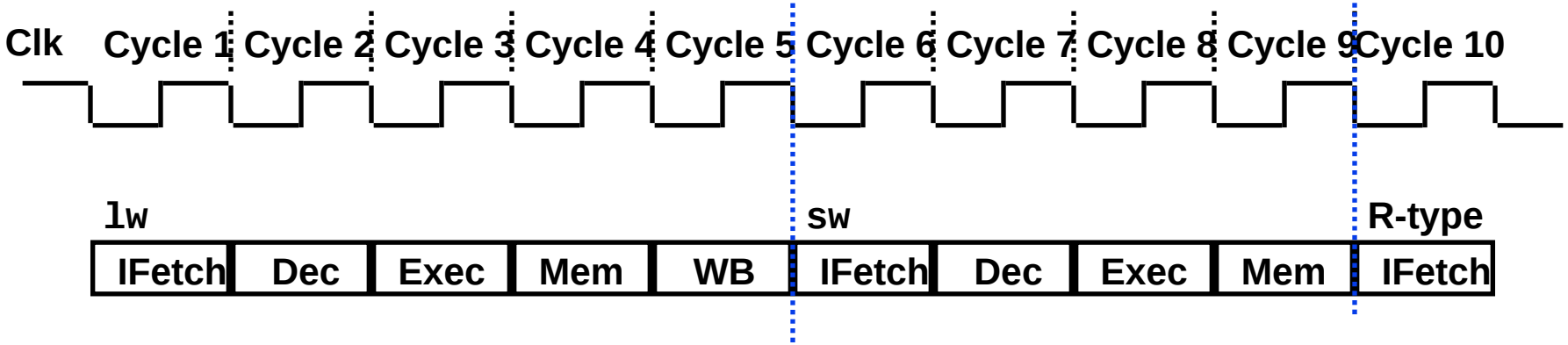
Mono-Cycle vs. Multi-Cycle

Implémentation mono-cycle:



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

Implémentation multi-cycle:



Pipeline

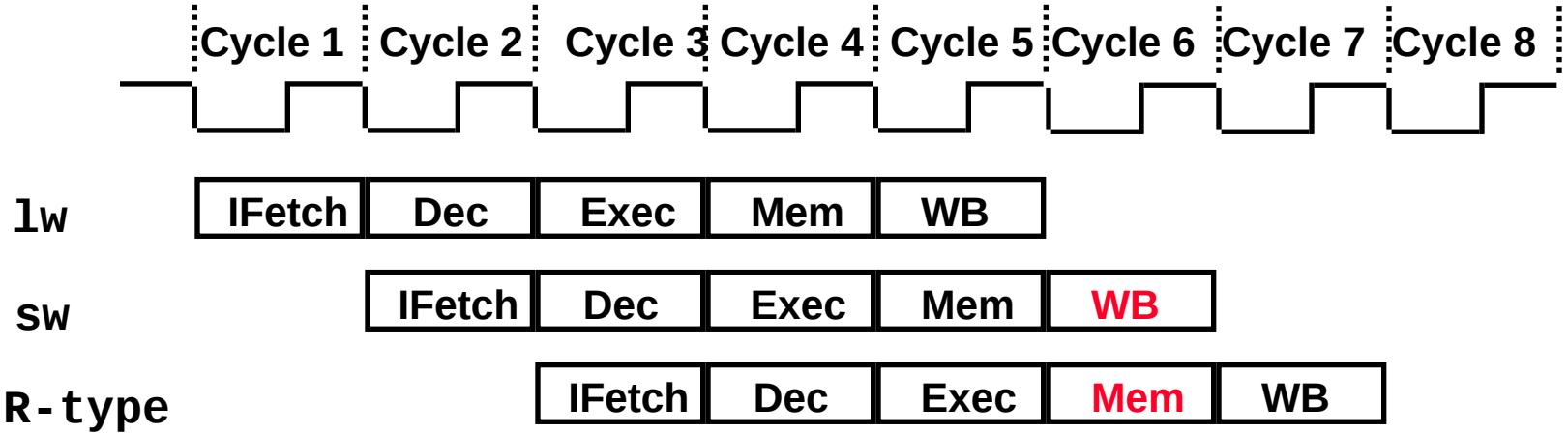
Pipeline

- Idée : faire travailler les différents circuits en parallèle :
  - instruction suivante cherchée pendant que la précédente est décodée, etc.
  - reprend l'idée du travail à la chaine

- Les CPUs modernes sont généralement pipelinés voire superpipelinés (plusieurs pipelines en parallèle), on parle d'architectures super-scalaires

Pipeline MIPS

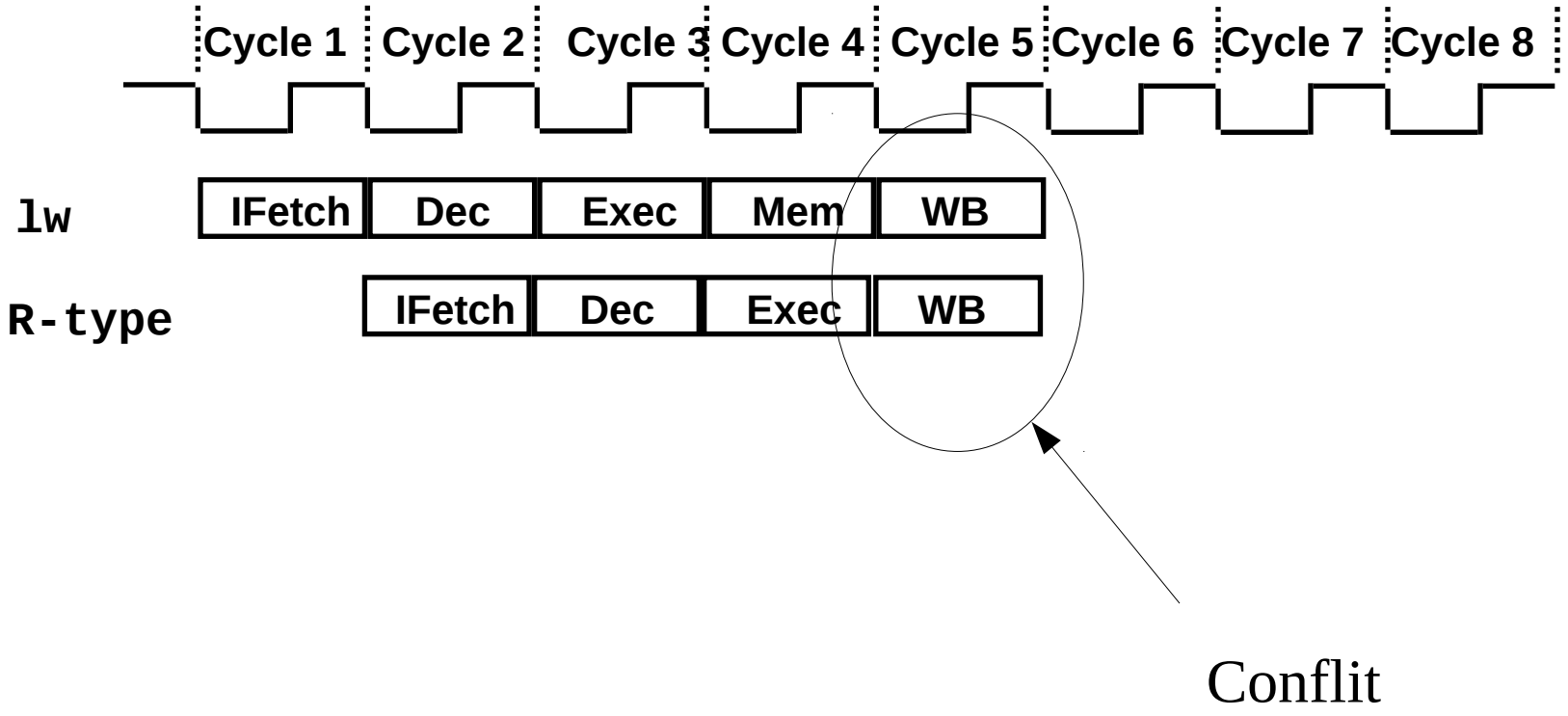
- Le pipeline :
  - augmente le débit : nombre d'instructions exécutés par seconde
  - mais pas la latence : temps d'exécution d'une seule instruction



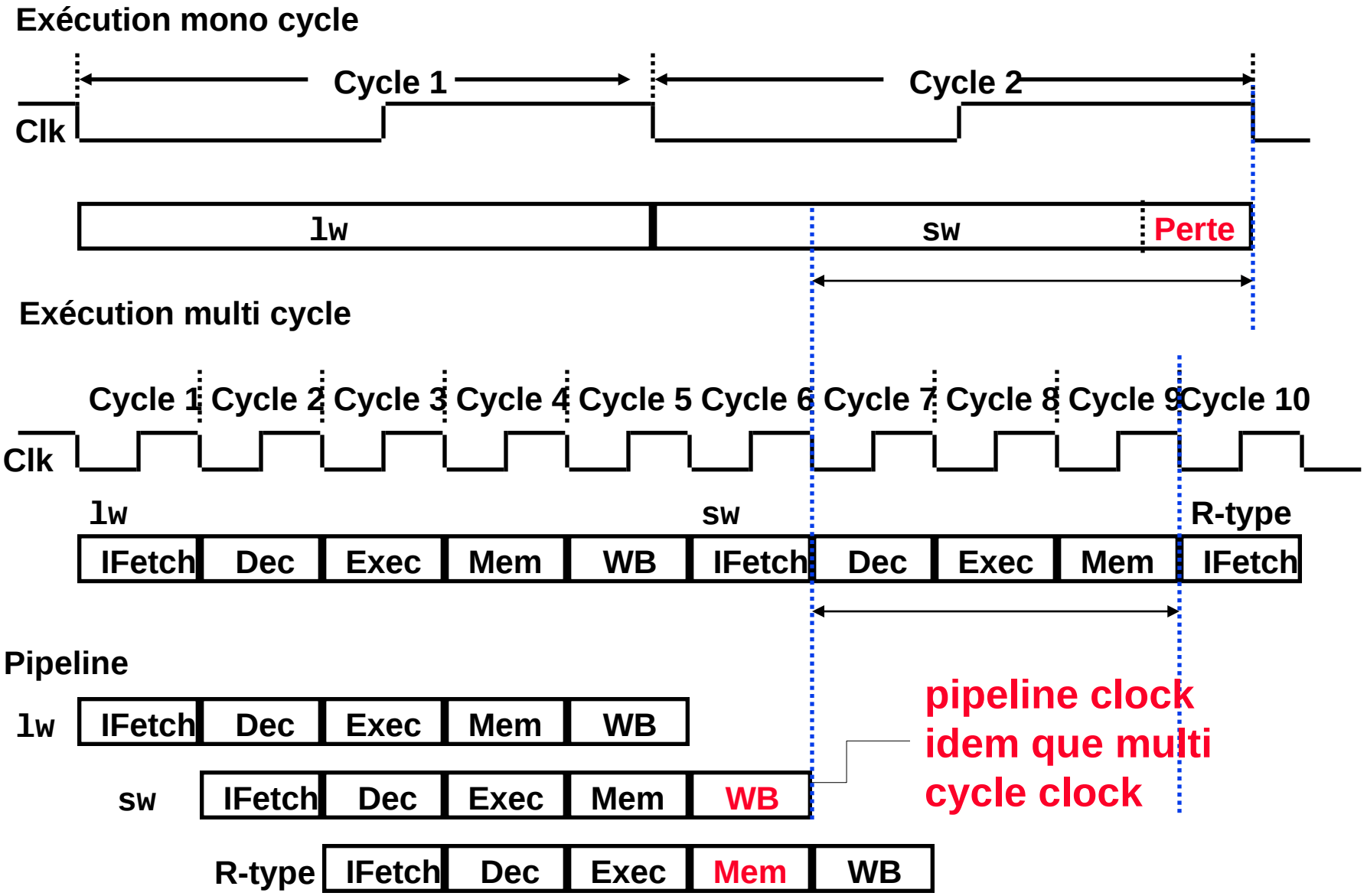
- temps de cycle donné par l'unité la plus lente
- CPI identique pour toutes les instructions

Pipeline MIPS

- CPI identique pour toutes les instructions sinon conflit

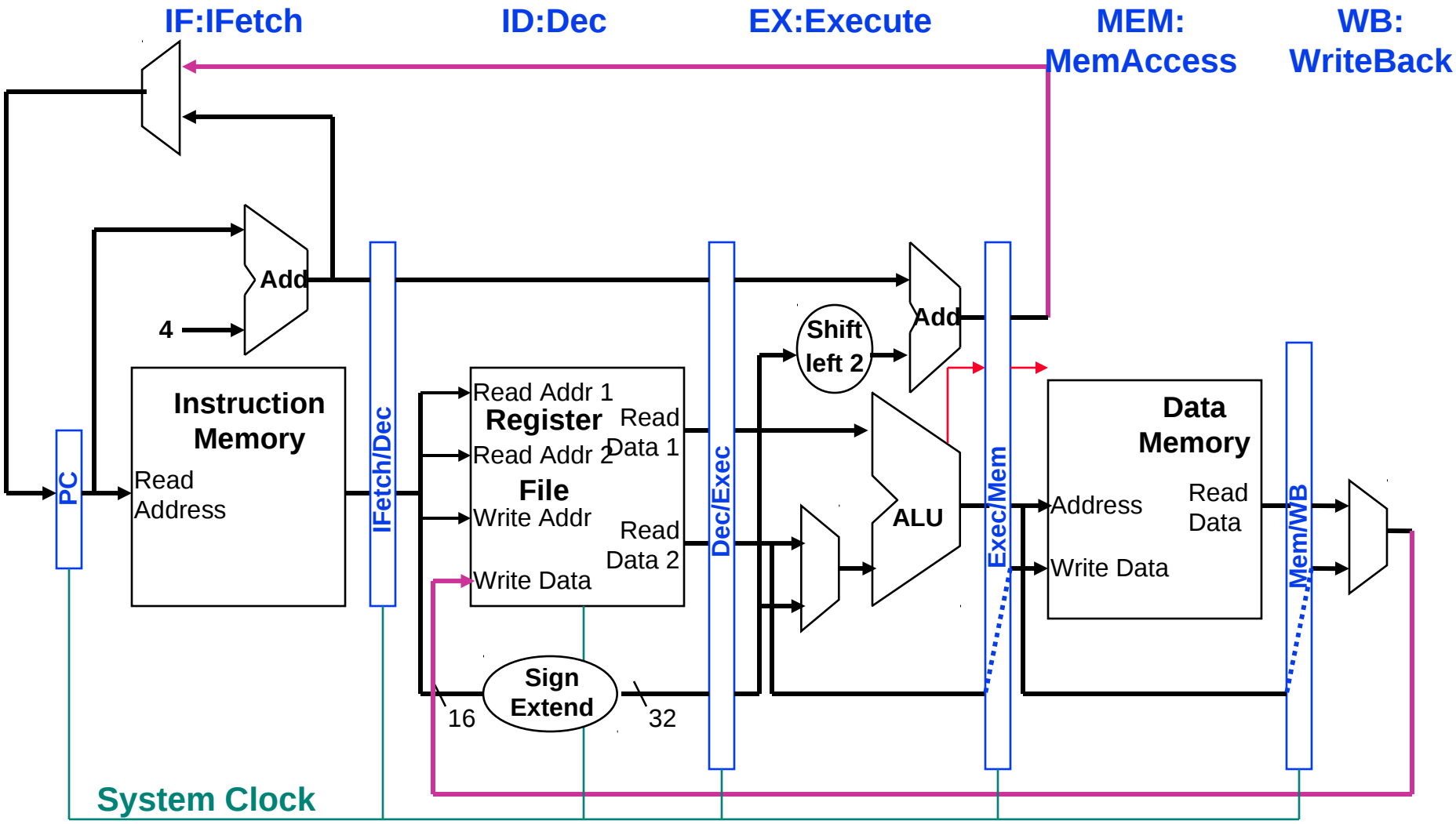


Single Cycle, Multiple Cycle, vs. Pipeline



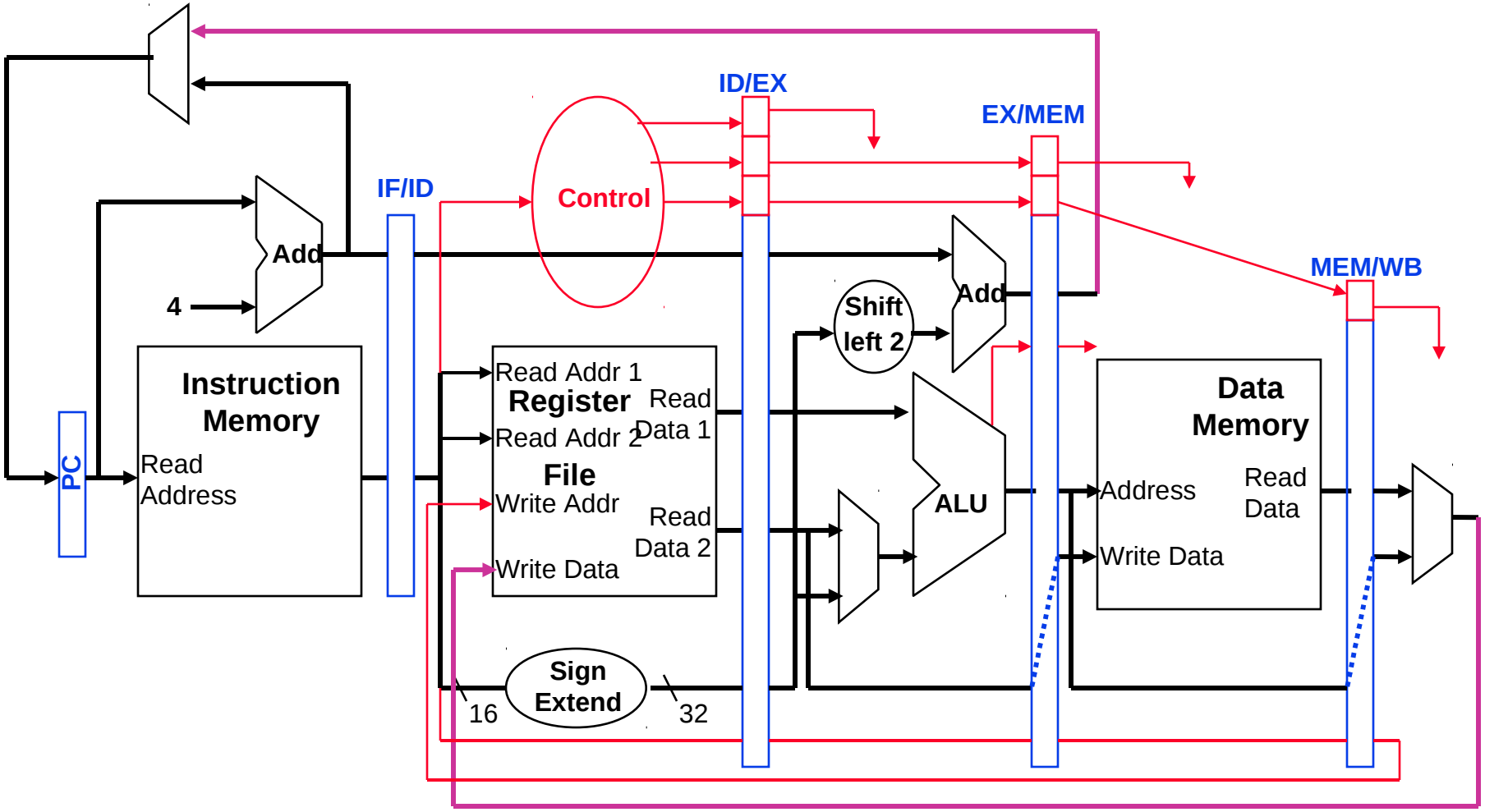
Modification du chemin de données

- Séparation des étages du pipeline => introduction registres de pipeline



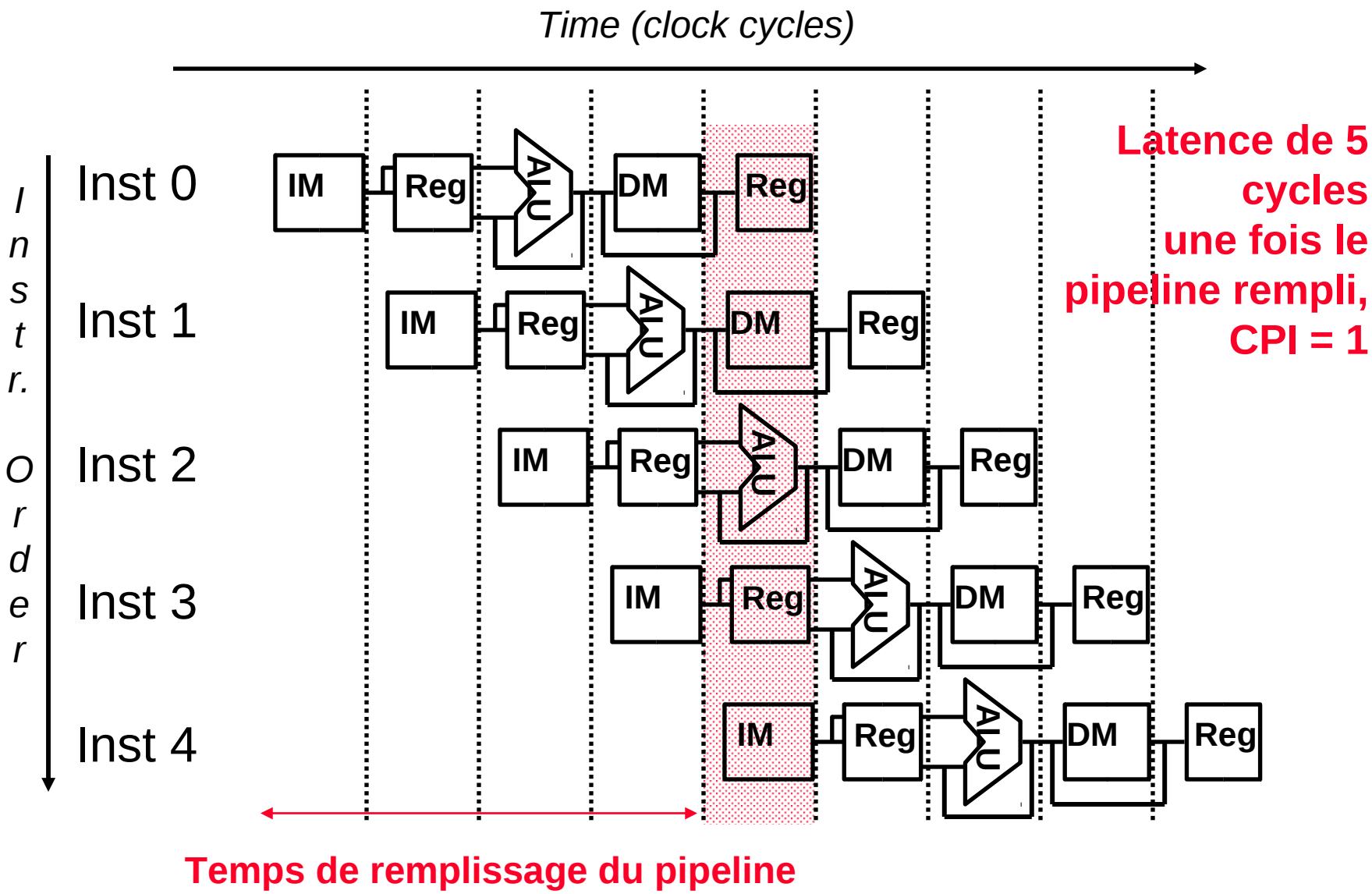
Modification du chemin de données

- Signaux de contrôle déterminés lors du décodage
  - et conservés dans les registres de pipeline



Performance

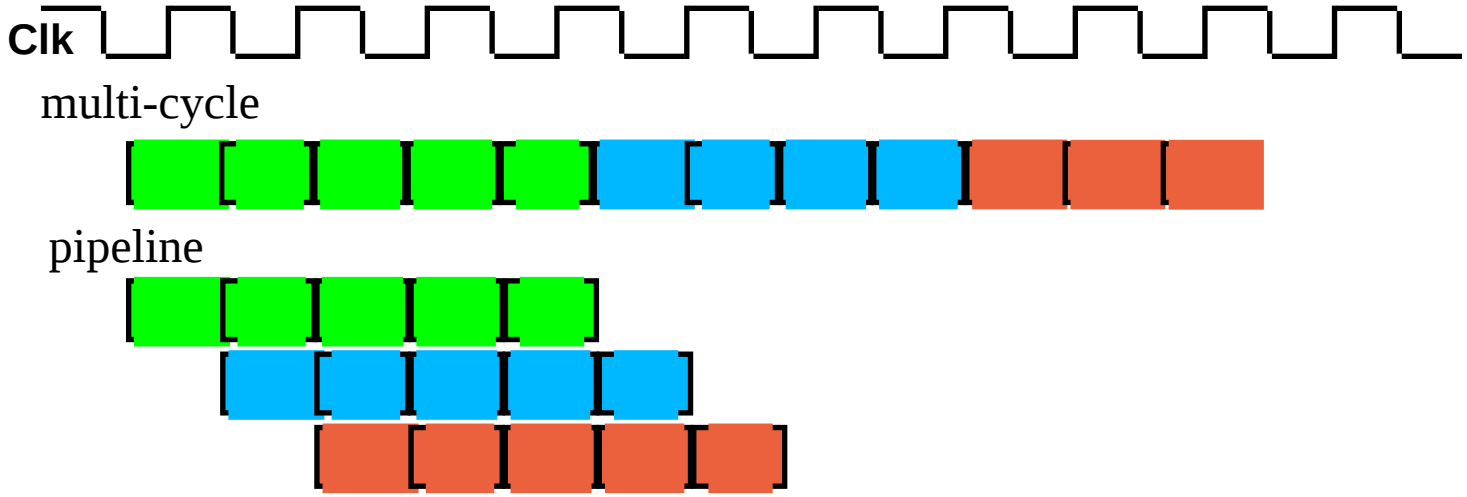
Performance



Performance

- speedup *n* instructions =

  (CPI moyen multi-cycle * n) / (latence + (n-1))



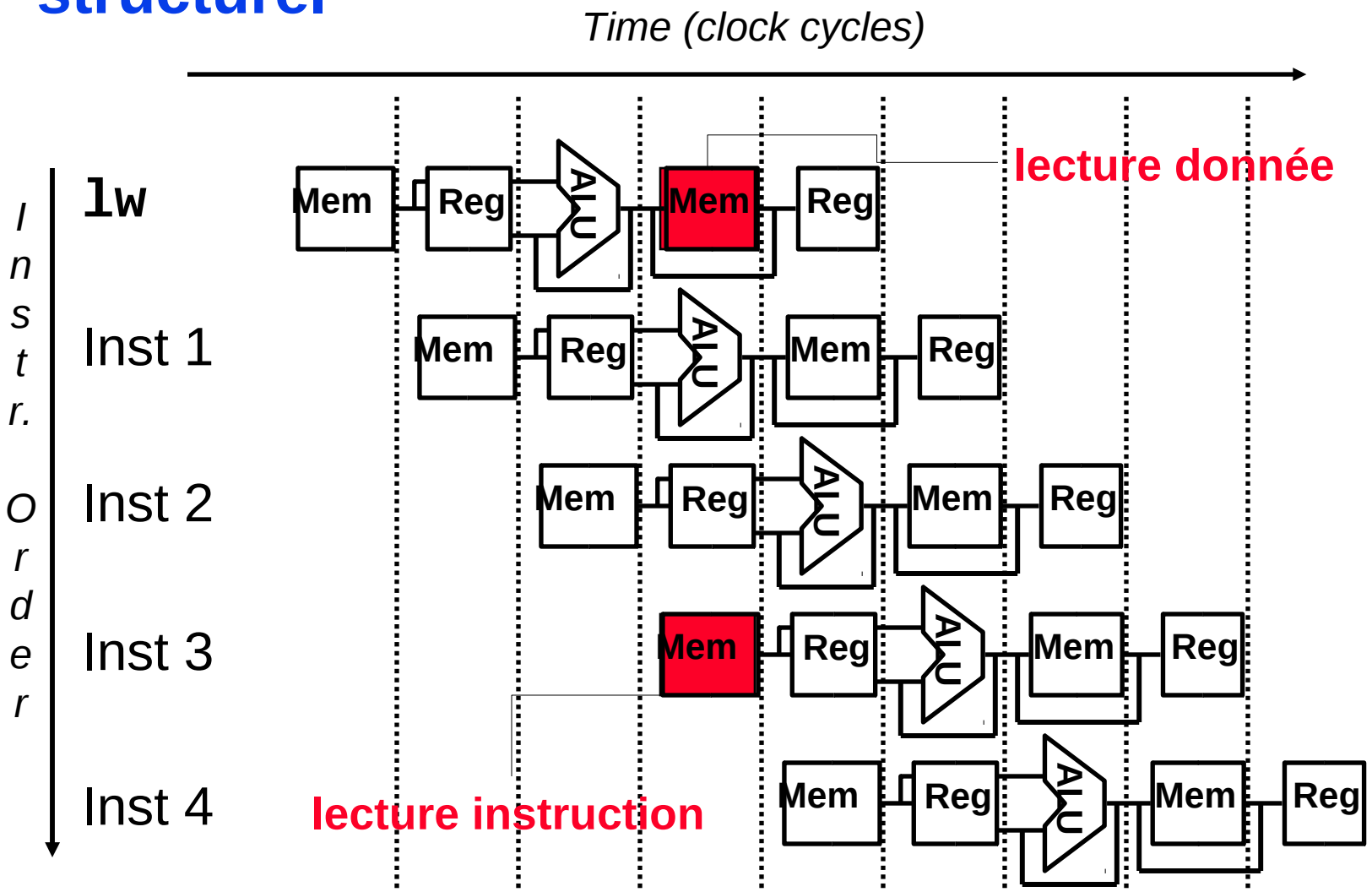
- *rem : le temps horloge est le même, donc n'intervient pas*
- ex : pour 1 instruction, speedup = 4.04/5 = 0.8
- pour 1000 instr., speedup = 4040/1004 ~ 4

Problèmes soulevés par le pipeline

Différents problèmes

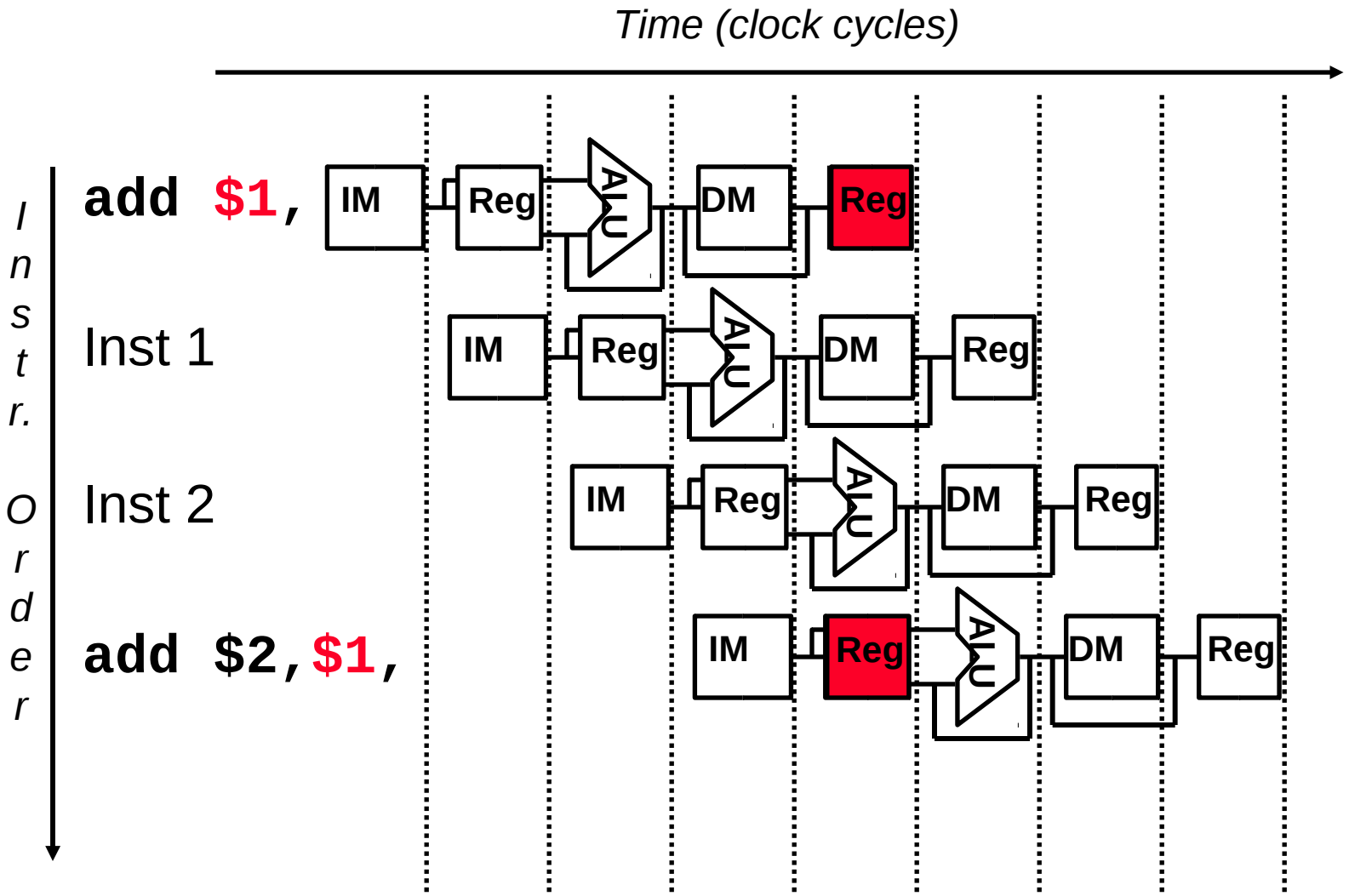
- Conflits structurels : utilisation de la même ressource au même moment

L'accès mémoire peut provoquer un conflit structurel

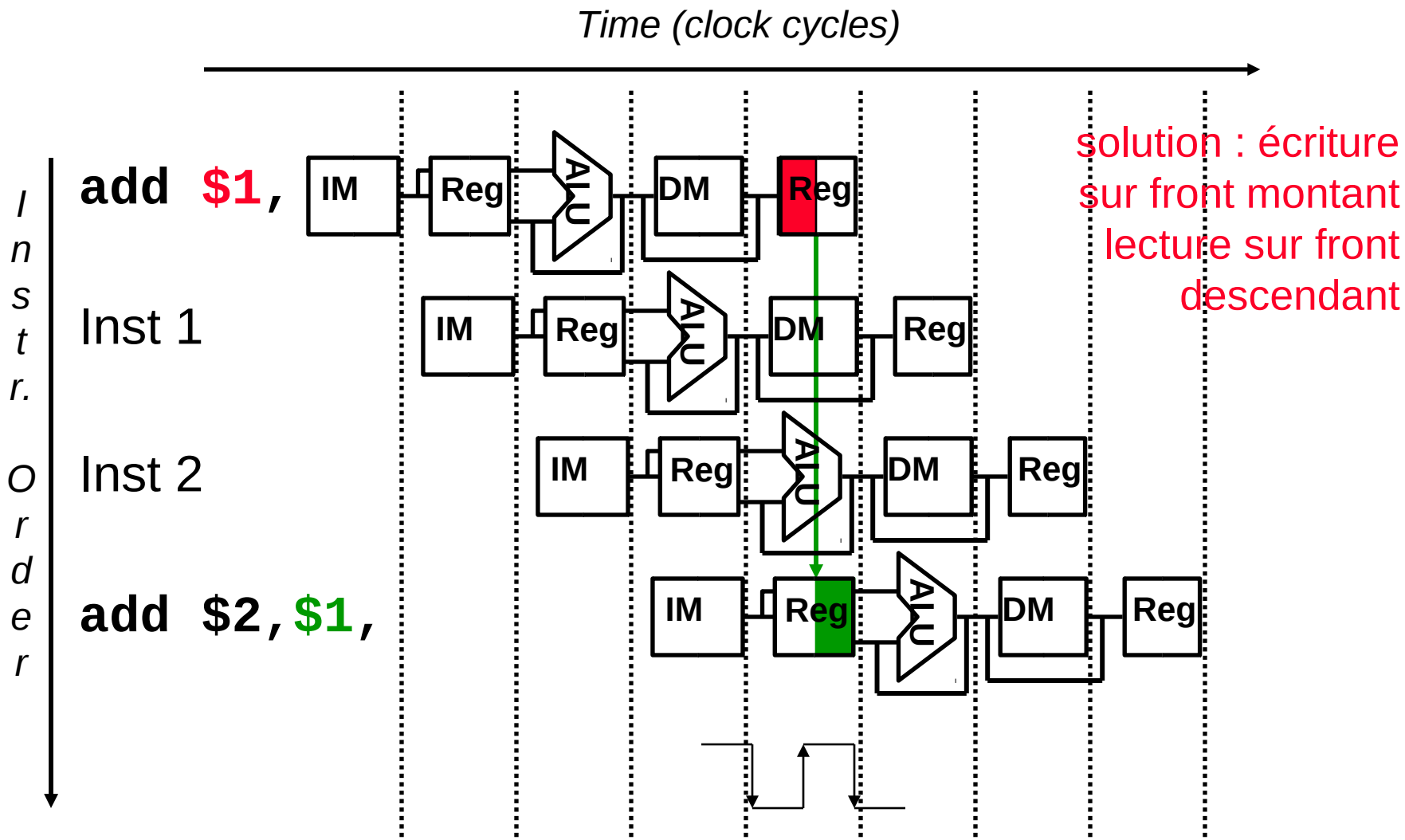


- solution : séparer les mémoires

Accès banc de registres



Accès banc de registres



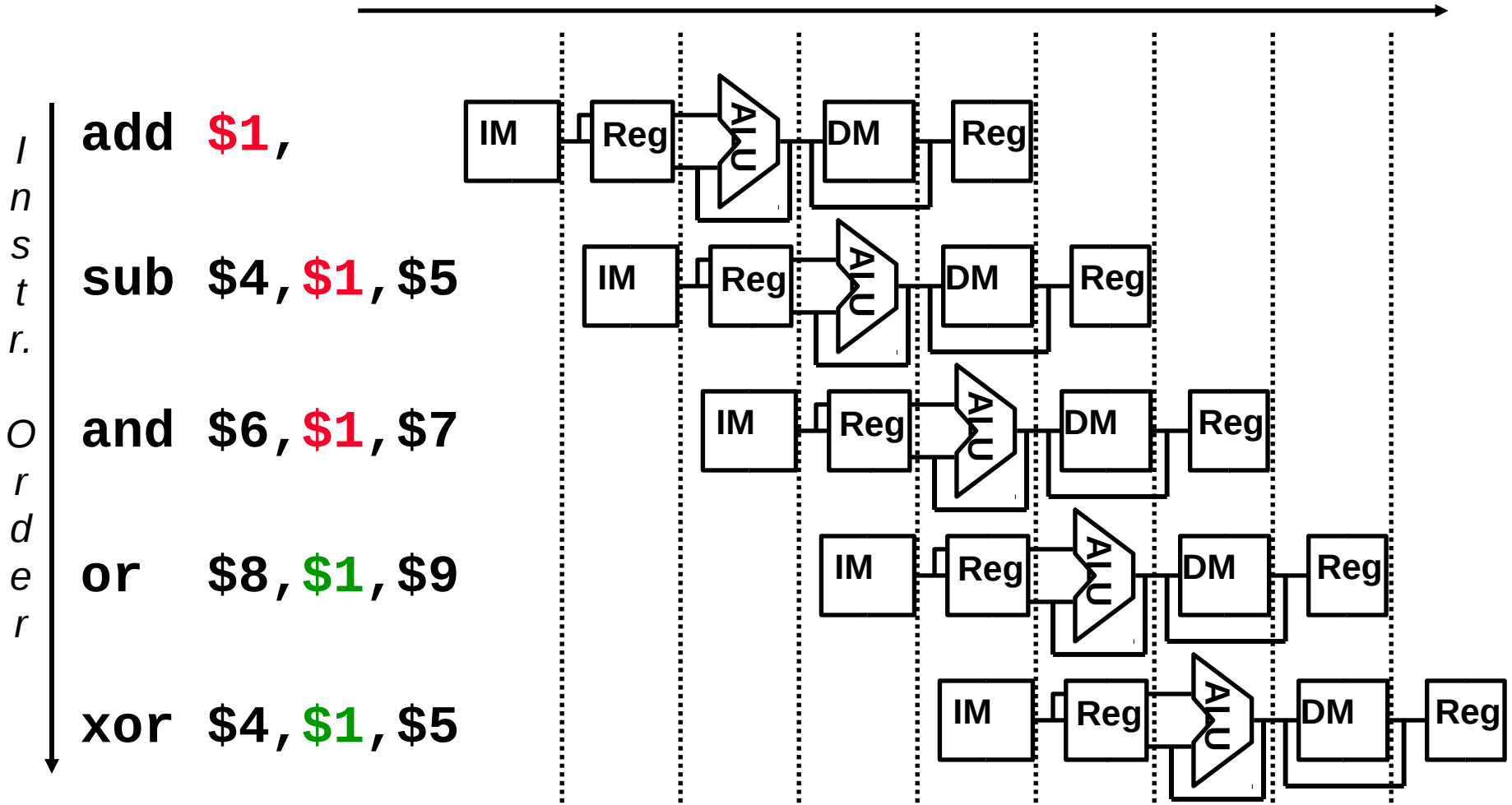
Différents problèmes

- Conflits structurels : utilisation de la même ressource au même moment

- Conflits de données : accès à une donnée en cours de modification

Conflit de données

- dépedance de registres entre instructions

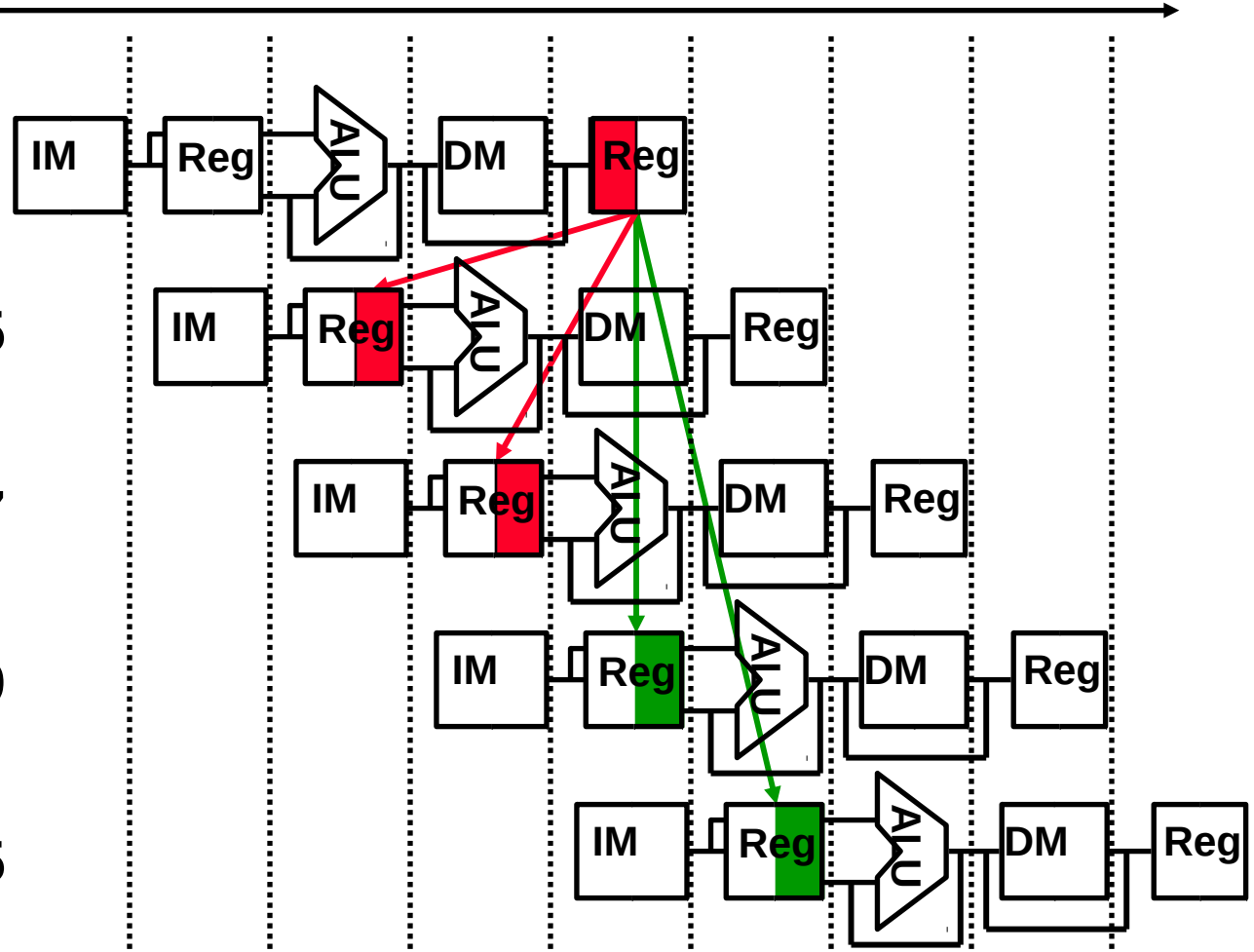


- Conflit « lecture après écriture »

Conflit de données

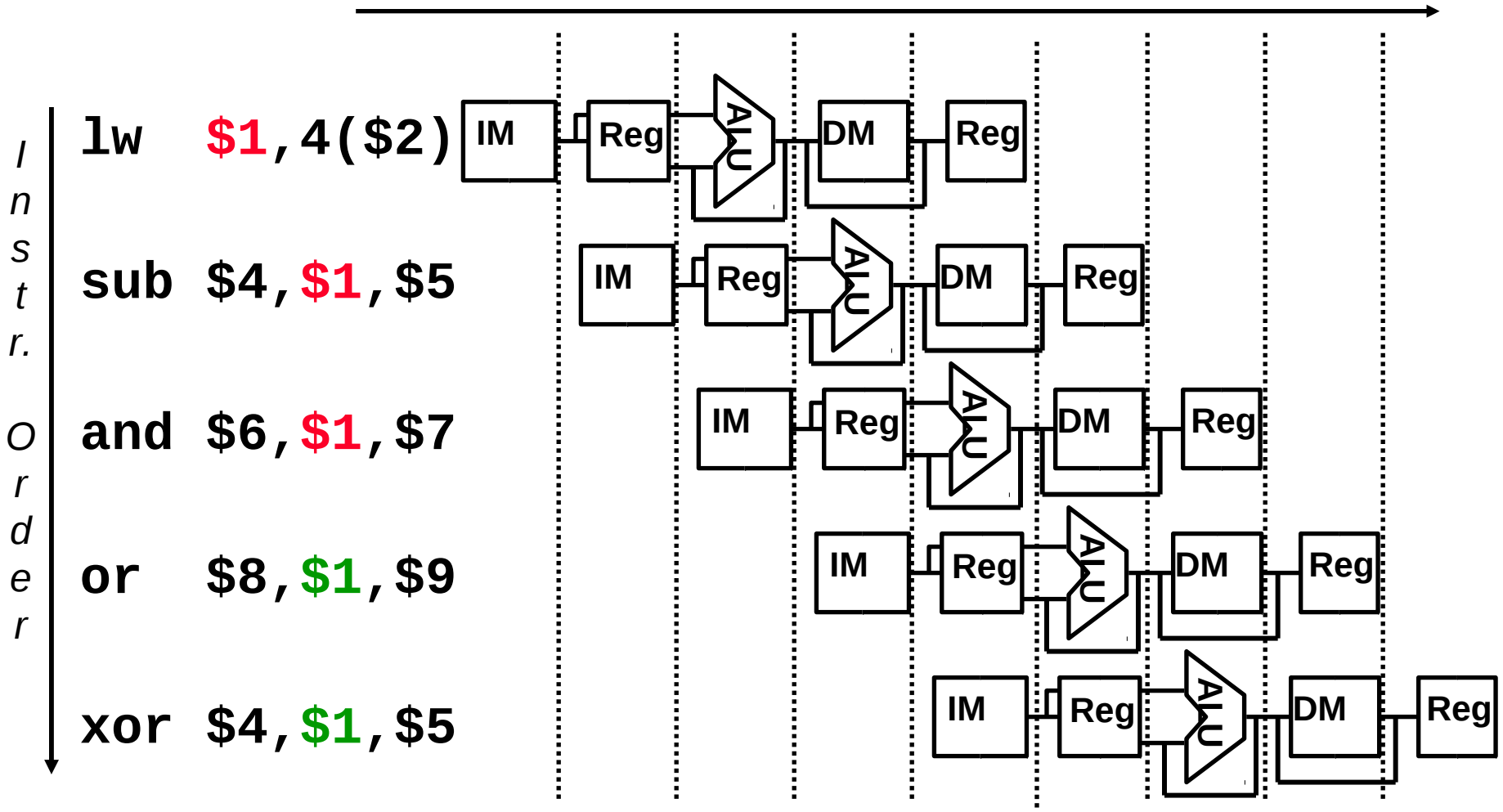
- dépedance de registres entre instructions

```
add  $1,
sub  $4,$1,$5
and  $6,$1,$7
or   $8,$1,$9
xor  $4,$1,$5
```

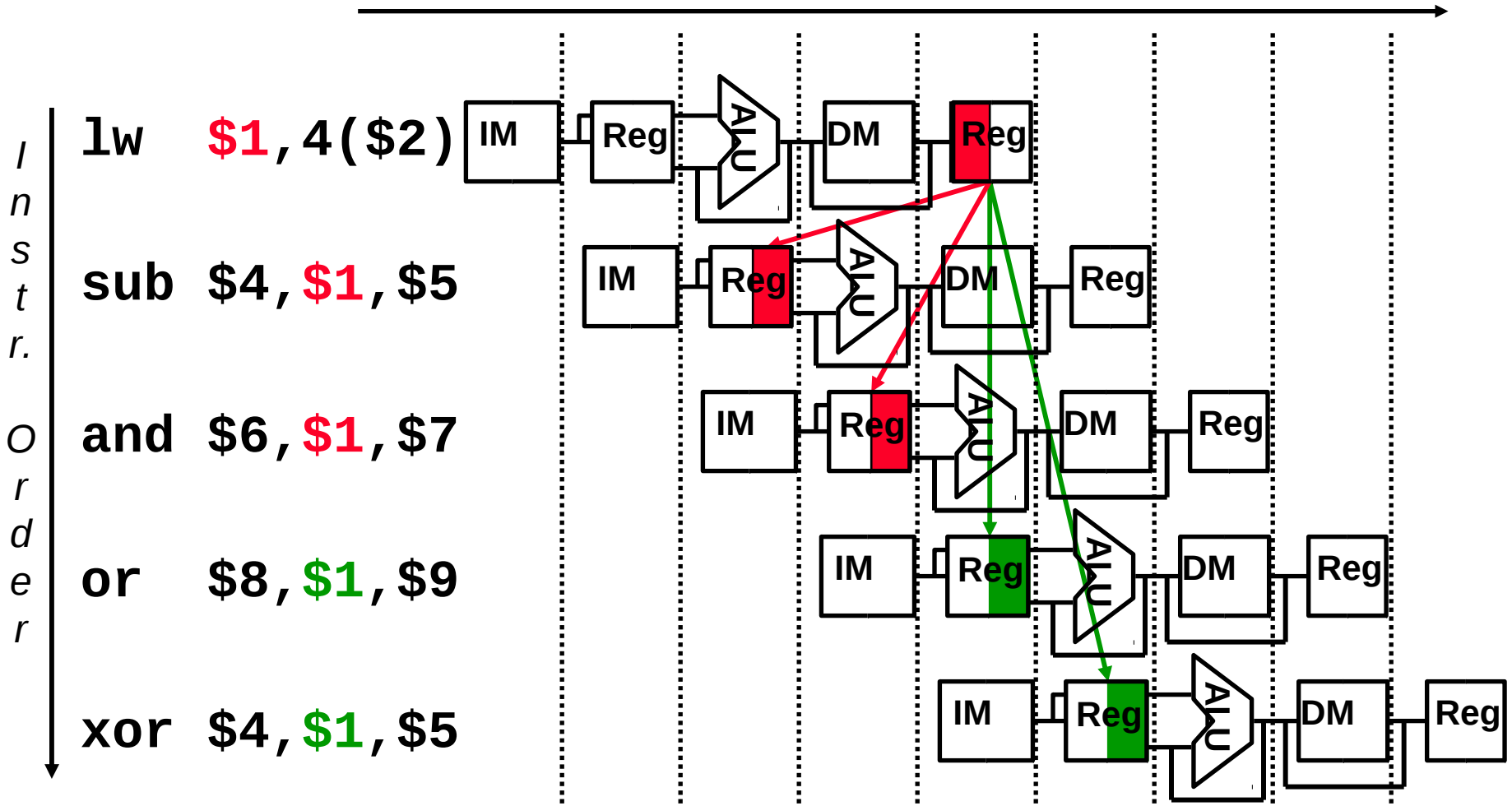


- Conflit « lecture après écriture » (RAW : Read After Write)

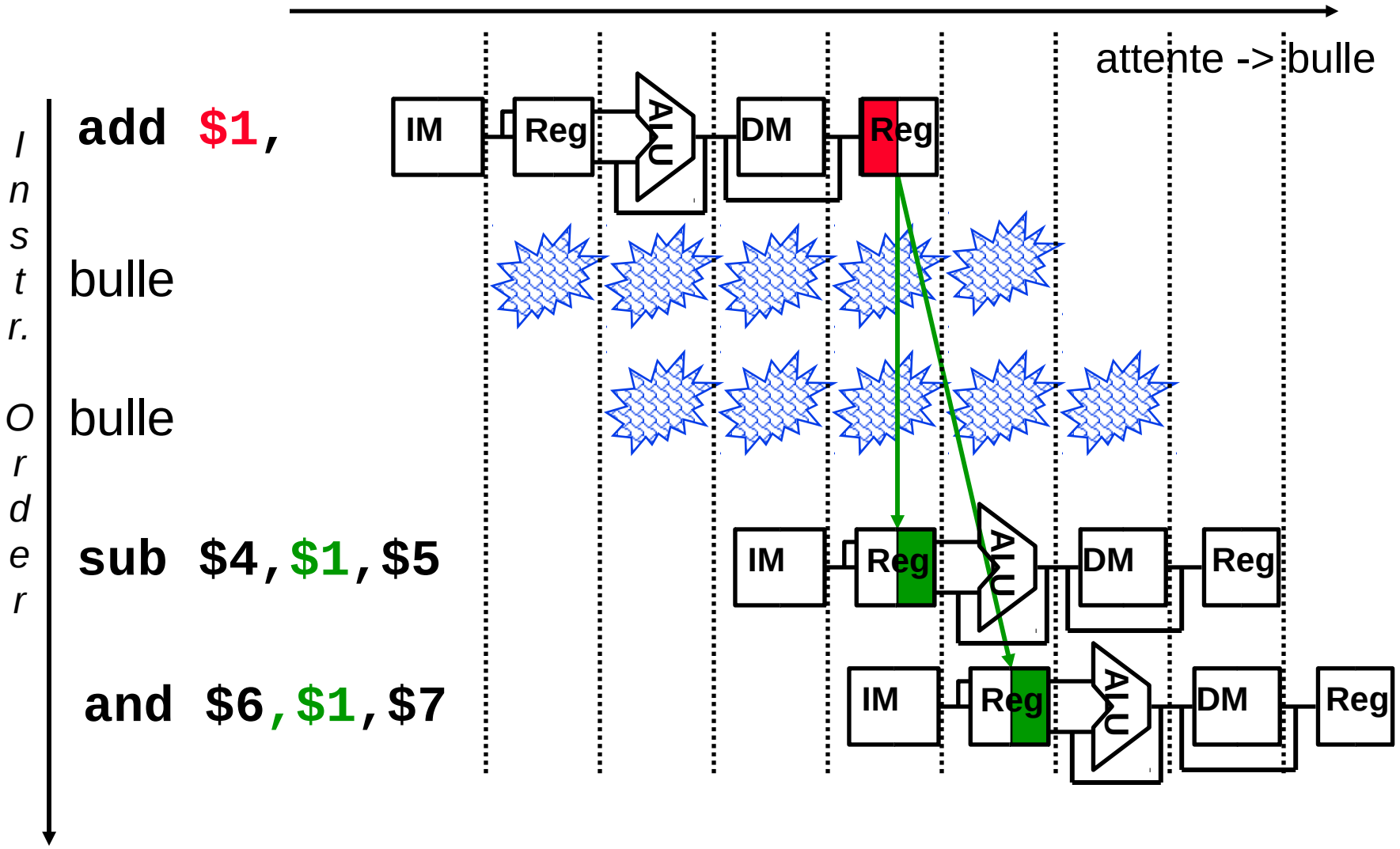
Conflit de données : load



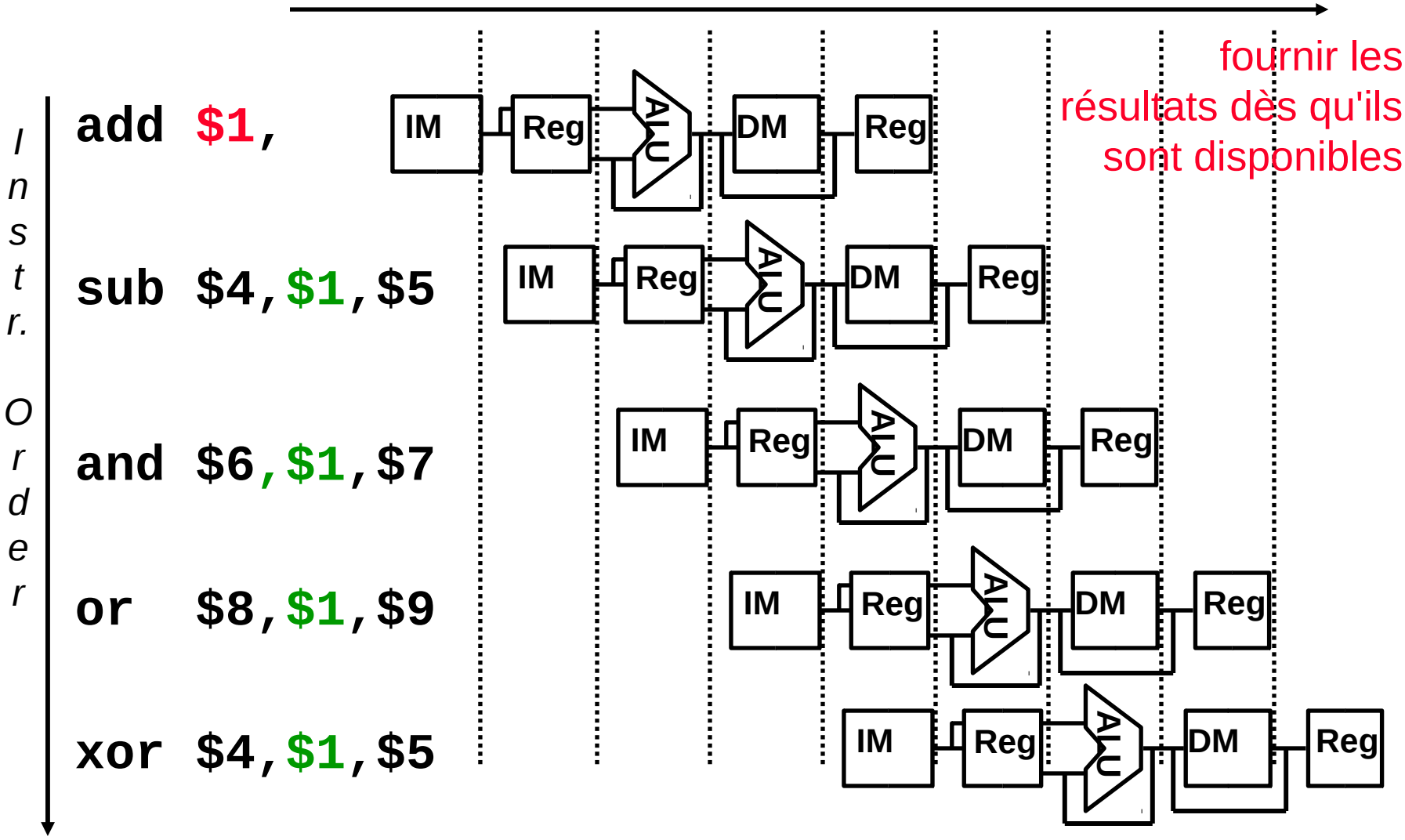
Conflit de données : load



Solution 1 : bulle



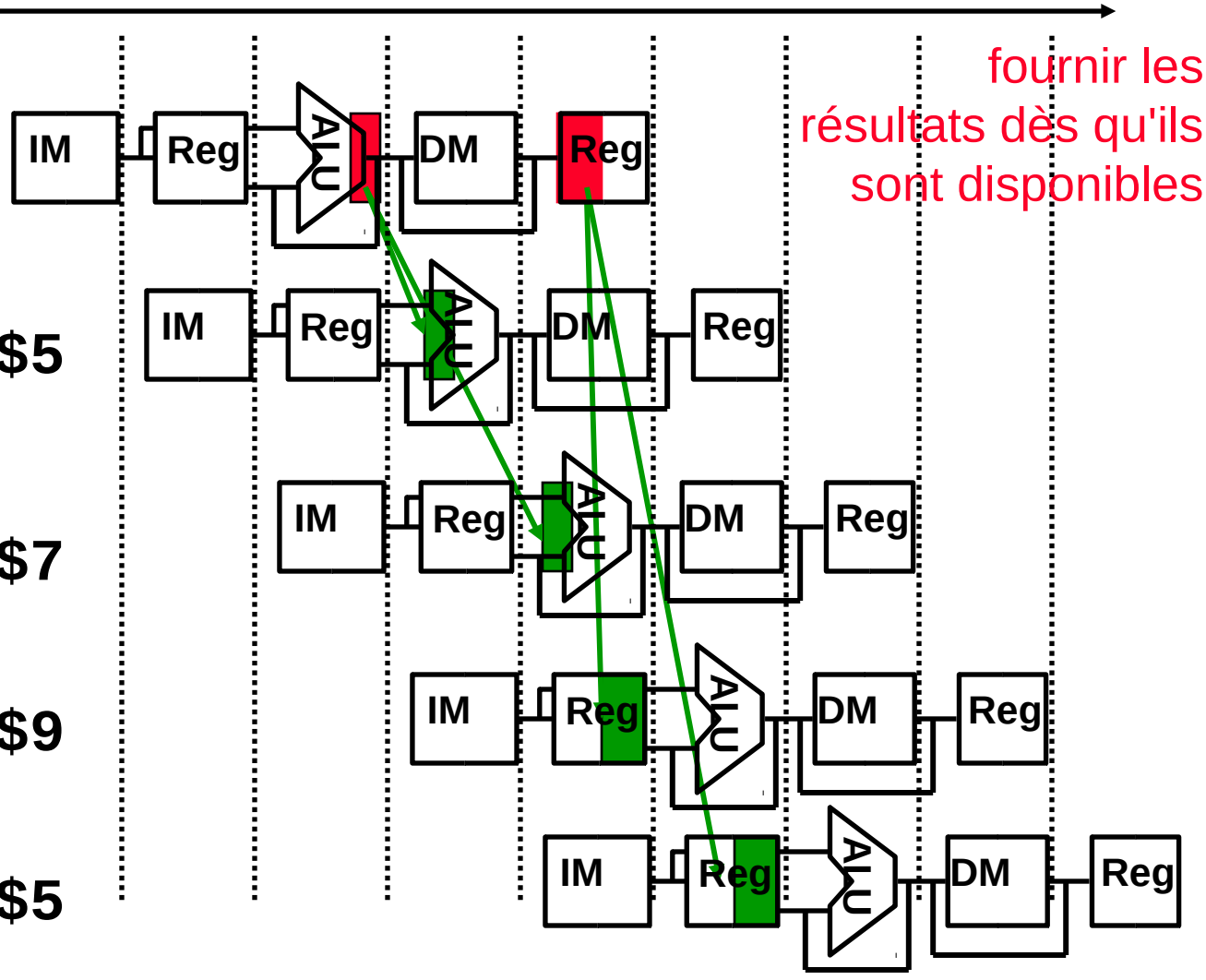
Solution 2 : forwarding de résultat



Solution 2 : forwarding de résultat

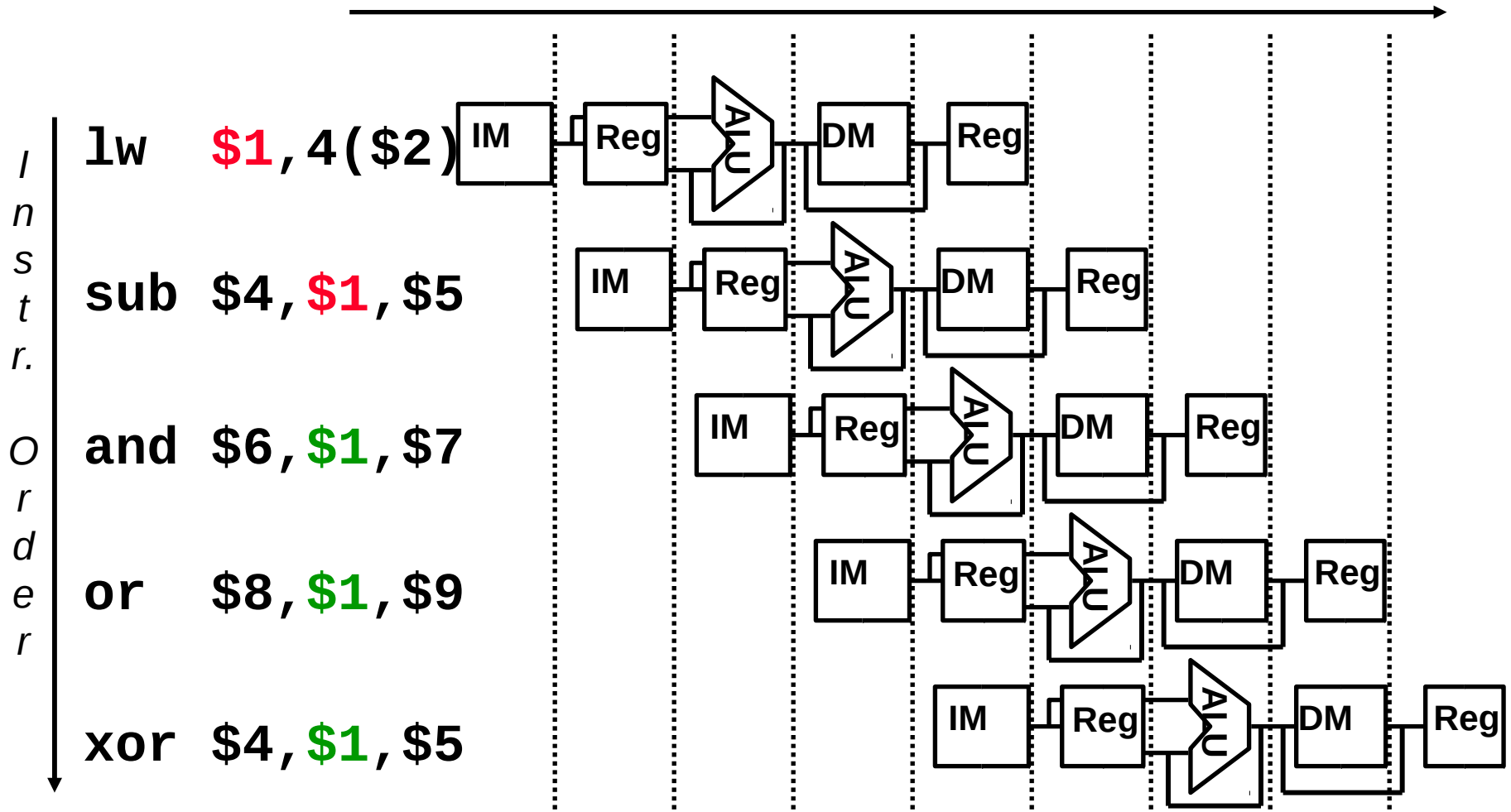
Instr. Order

```
add $1,
sub $4,$1,$5
and $6,$1,$7
or  $8,$1,$9
xor $4,$1,$5
```

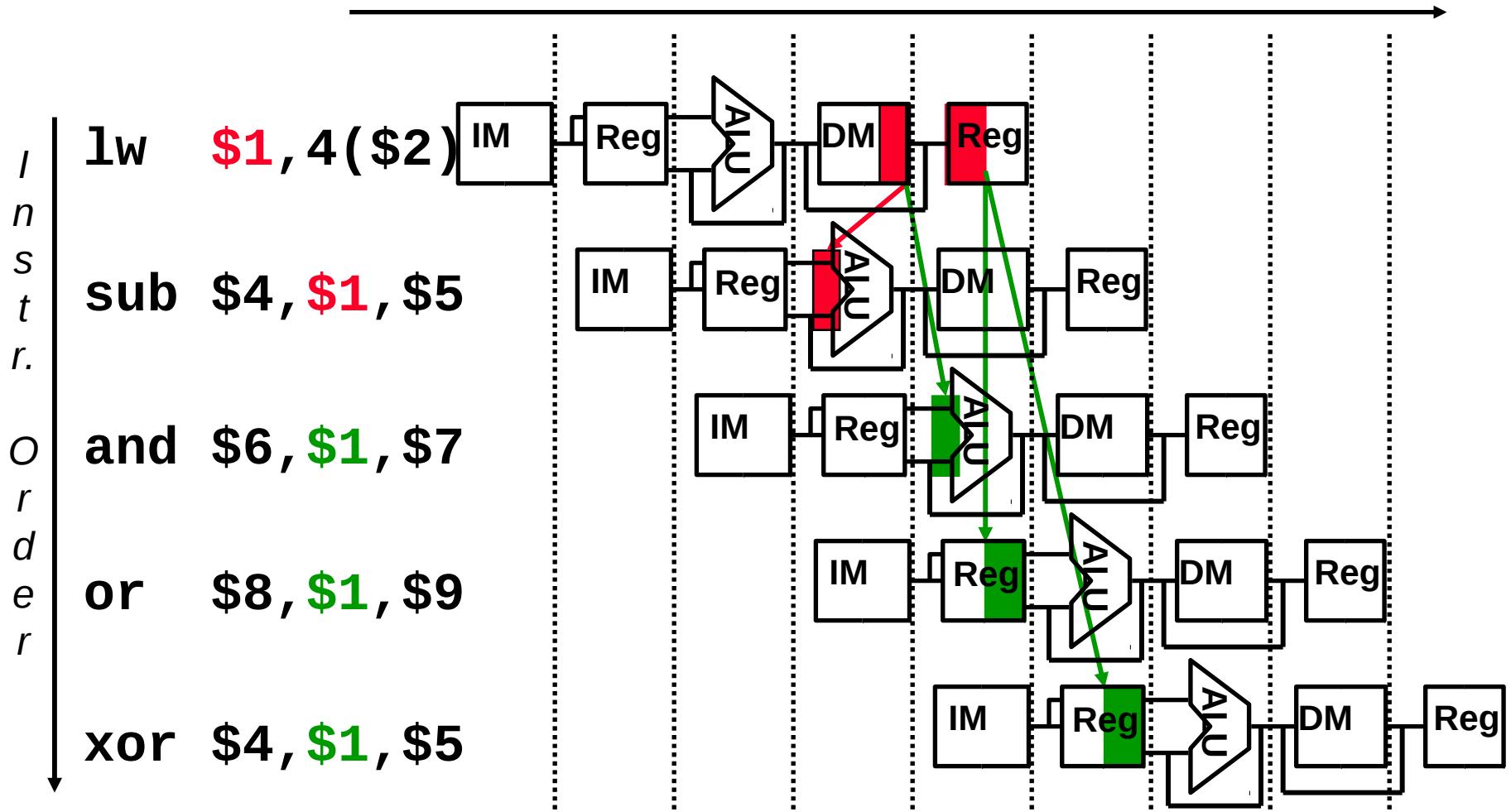


fournir les résultats dès qu'ils sont disponibles

Forwarding sur chargement



Forwarding sur chargement



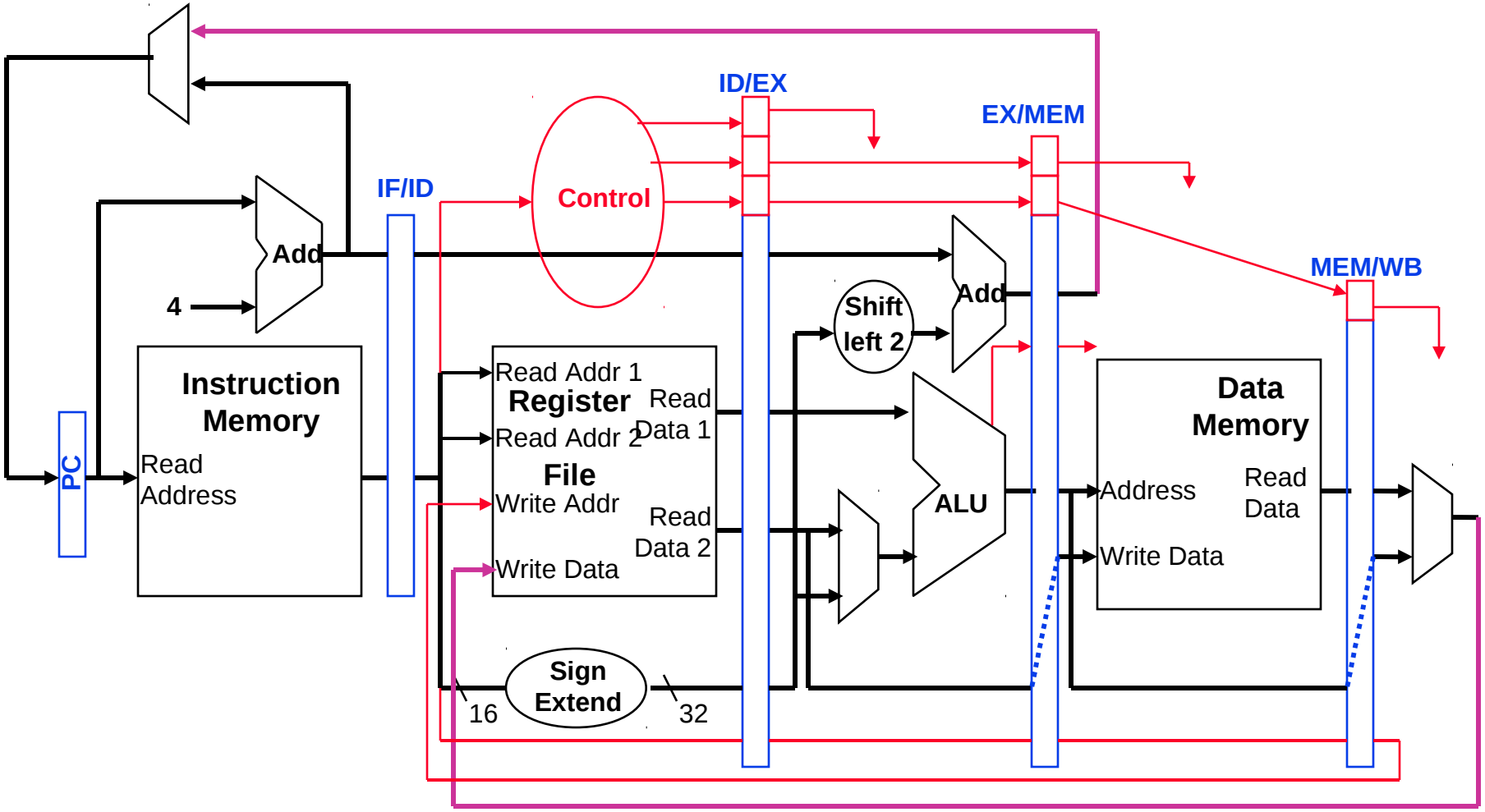
Différents problèmes

- Conflits structurels : utilisation de la même ressource au même moment
- Conflits de données : accès à une donnée en cours de modification
- Aléas de contrôle : l'instruction suivante n'est pas à la suite dans la mémoire

Aléas de contrôle

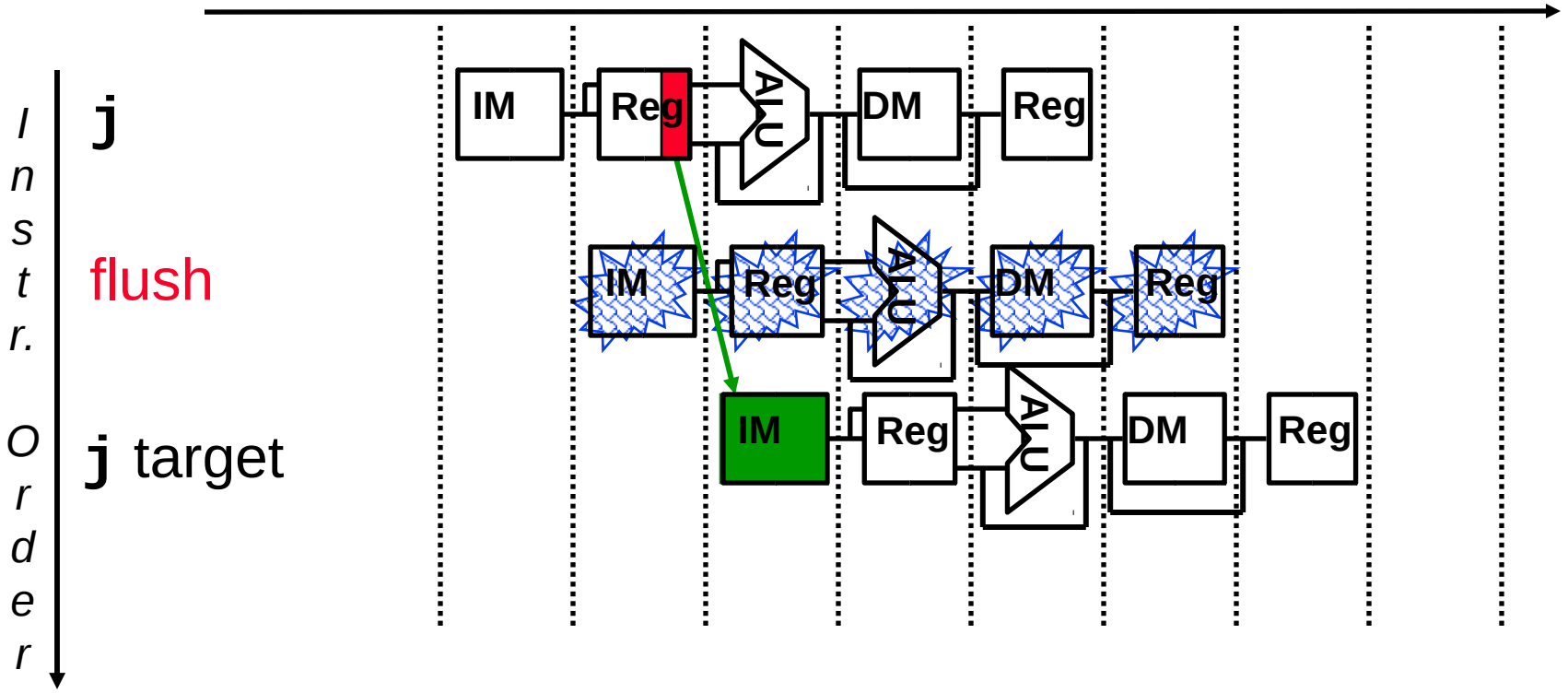
- Lorsque le flot d'instruction n'est pas séquentiel (si PC ≠ PC + 4)
  - sauts conditionnels (`beq`, `bne`)
  - sauts inconditionnels (`j, jal, jr`)
  - Exceptions
- Solutions
  - bulles (diminue la performance)
  - forwarding de résultat (ne supprime pas toutes les bulles)
  - prédictions
- Moins fréquent que les conflits de données mais aucune solution aussi efficace

Rappel : chemin de données



Sauts inconditionnels (jump)

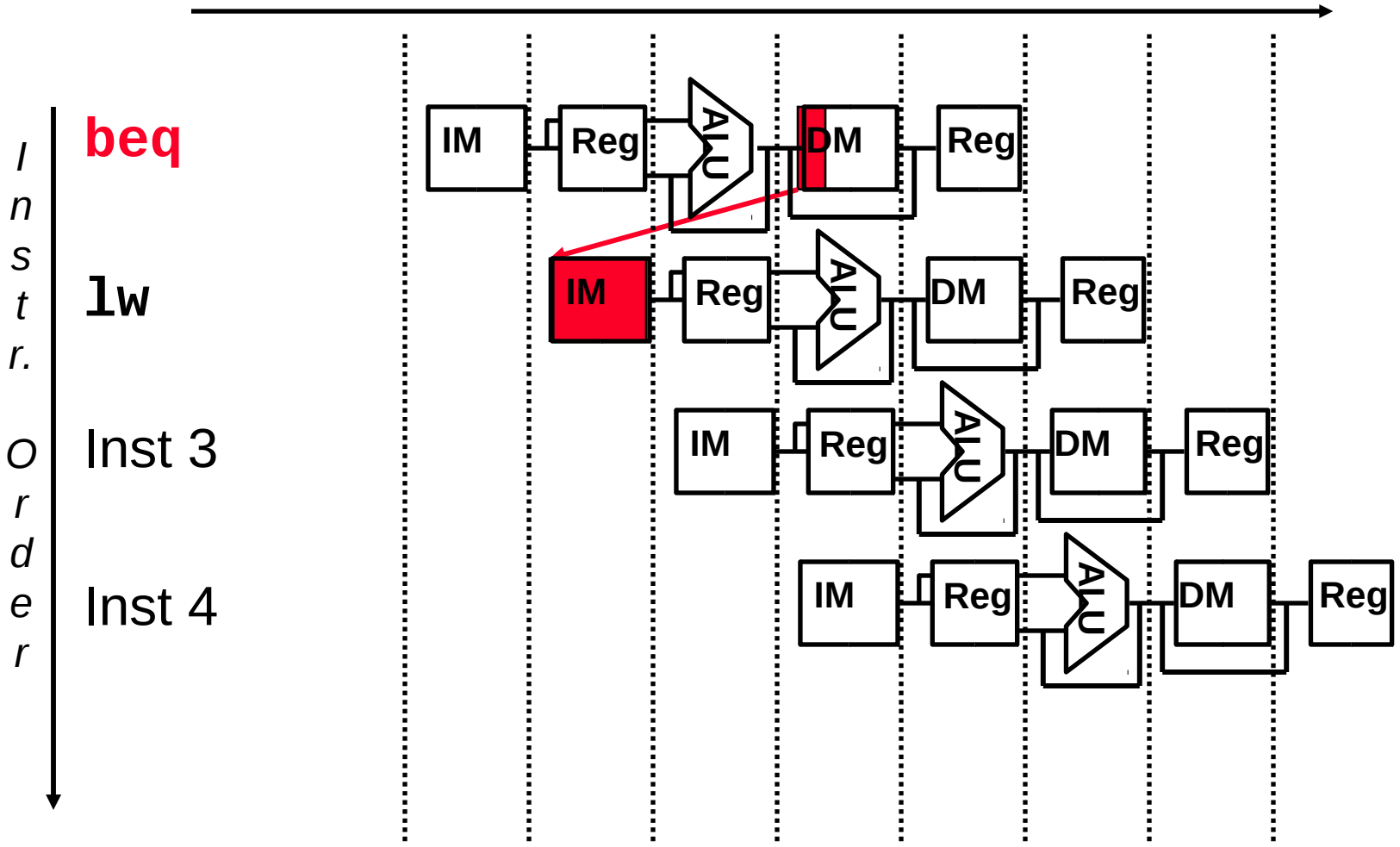
- calcul ramené dans l'unité de décodage => vidange



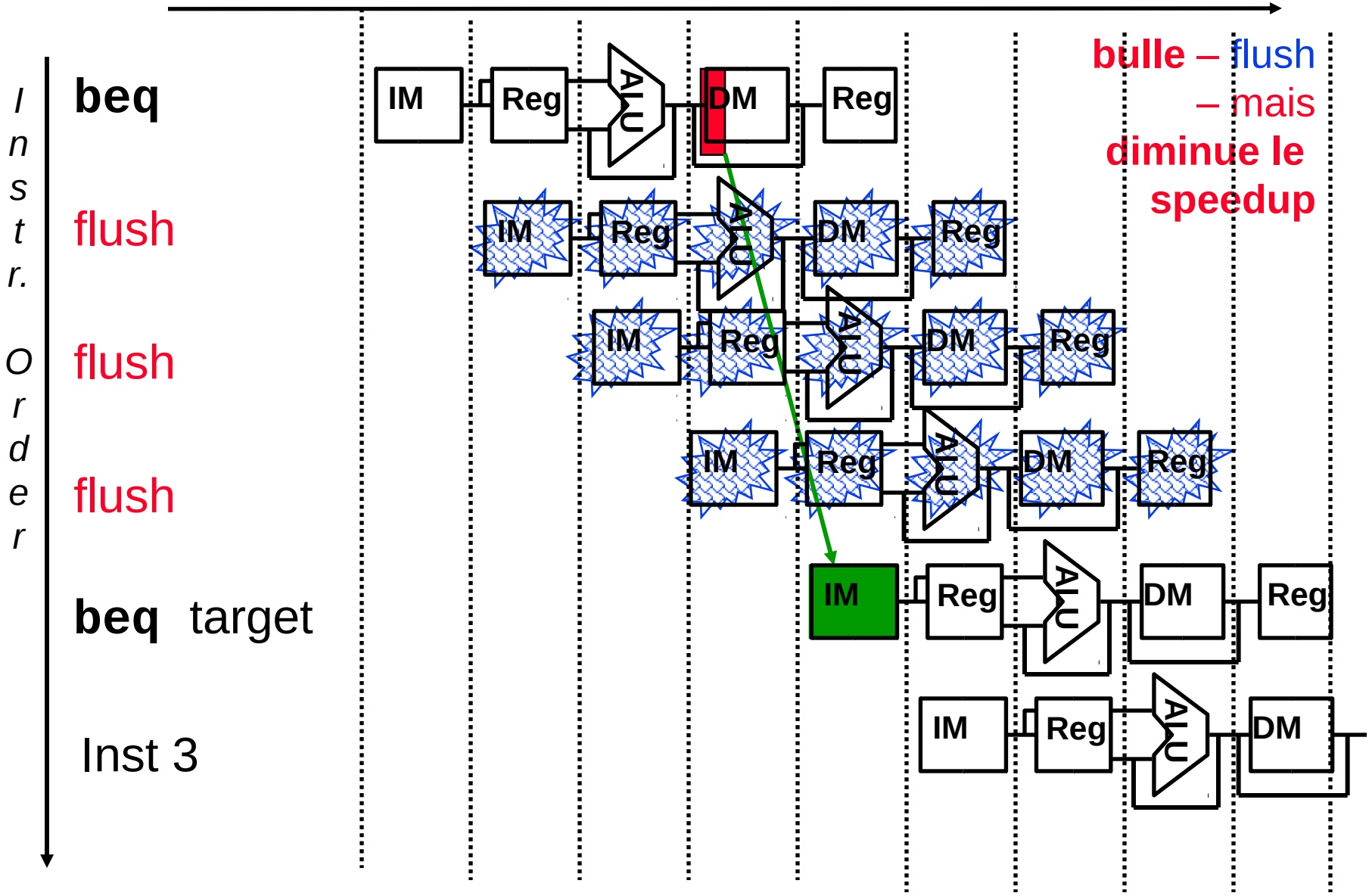
- Heureusement moins de 3% des instructions

Sauts conditionels (branch)

- connus après étage d'exécution

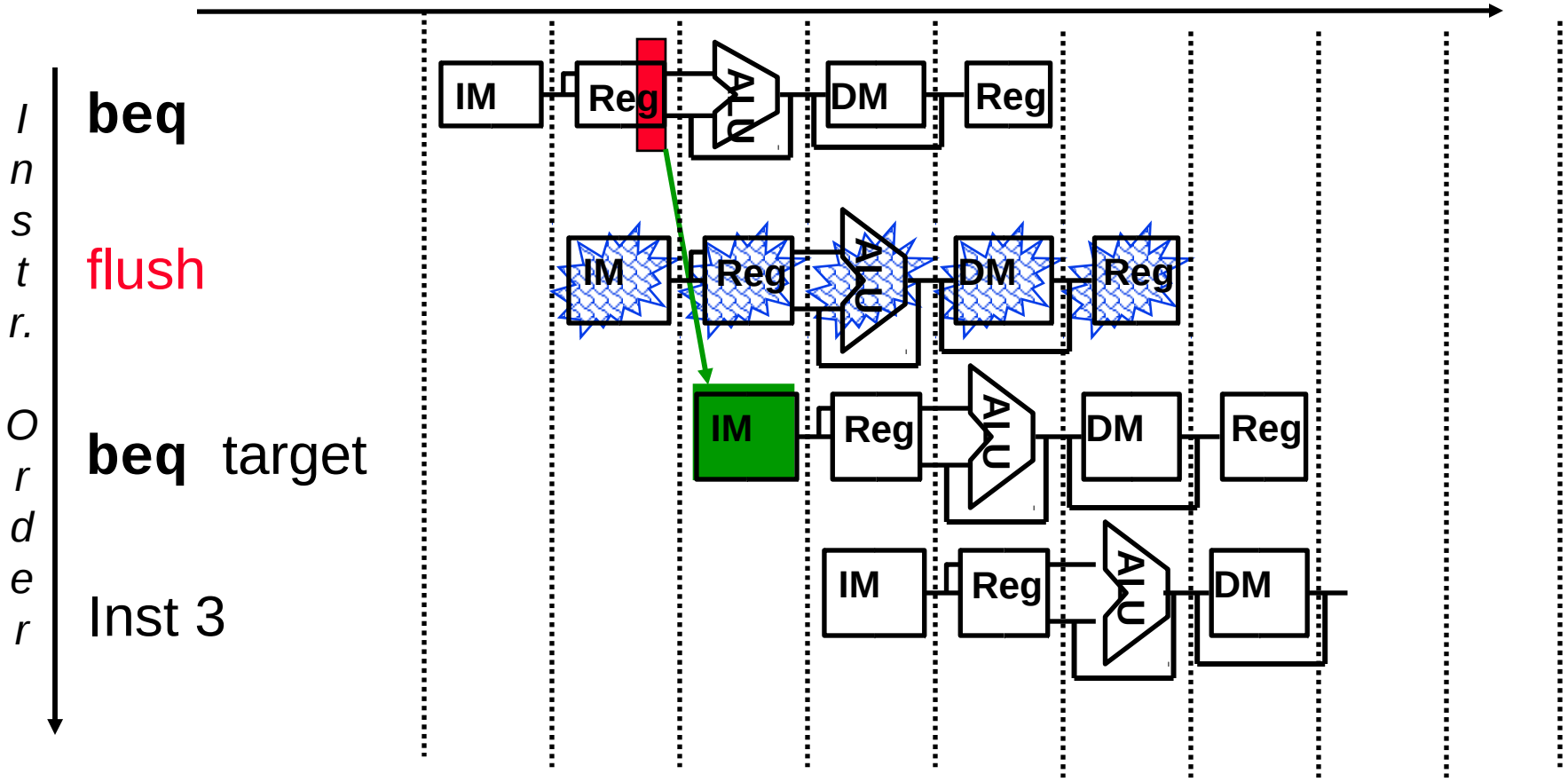


Solution 1 : flush (vidange)

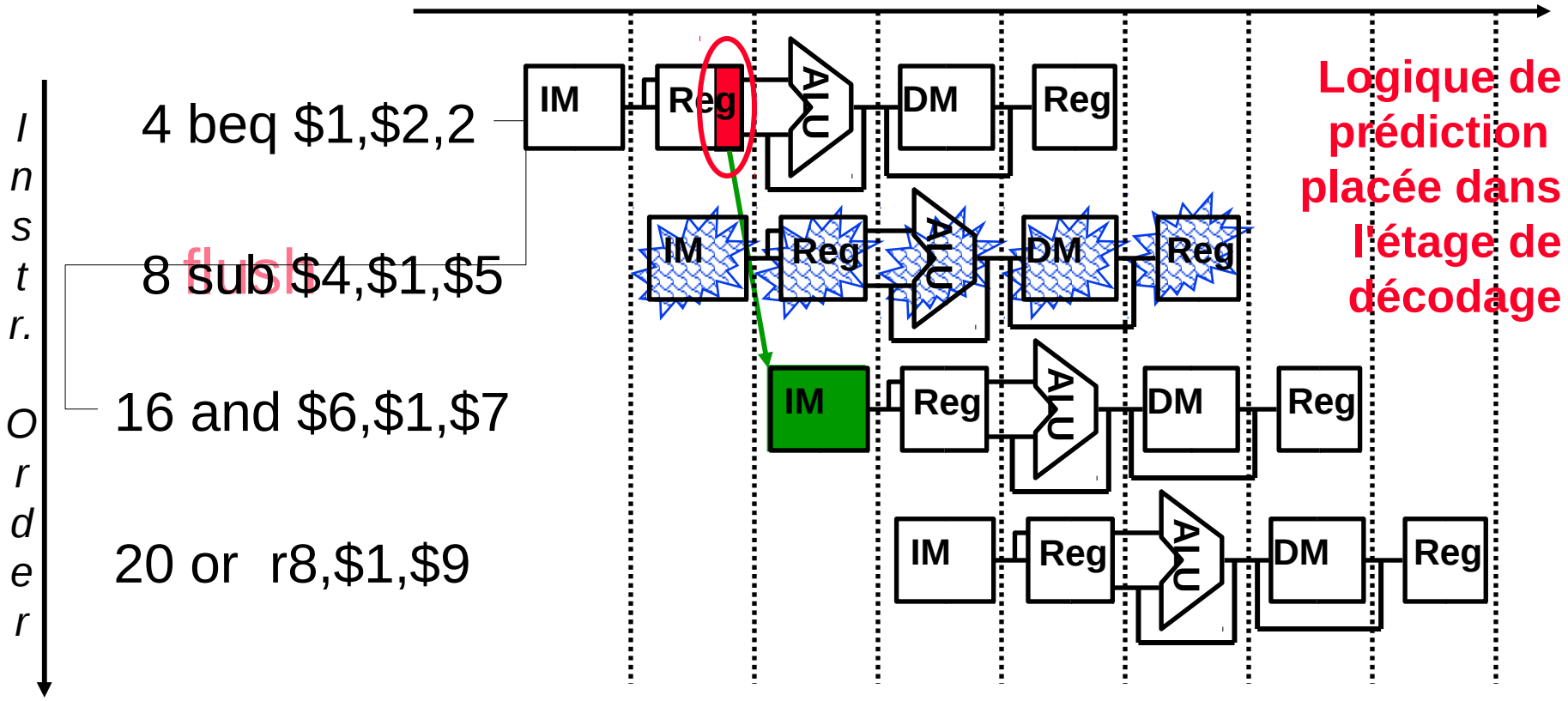


Solution 2 : forwarding

- réalisation du calcul dans l'unité de décodage



Solution 3 : prédiction de branchement



Deux types de mise en attente

- Bulle (instruction nop)

- Flushes (vidange de pipeline) : remplacement d'une instruction par une autre (sauts)

Résumé

- Les processeurs modernes utilisent le pipeline
- Le pipeline ne change pas la latence mais améliore le débit
- CPI potentiel : une instruction à chaque cycle

- Le pipeline est limité par l'étage le plus lent, ainsi :
  - Des étages différemment cadencés rendent le pipeline inefficace
  - Les temps de remplissage et de vidange influe sur les performances
- Des conflits (données, ressources) sont à résoudre
  - Mise en attente augmente le CPI moyen