Master 1 - SEM

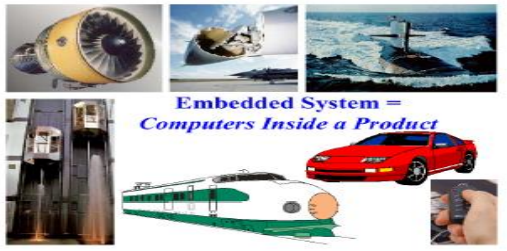
Processeurs embarqués

Cours 5.2

Pipeline

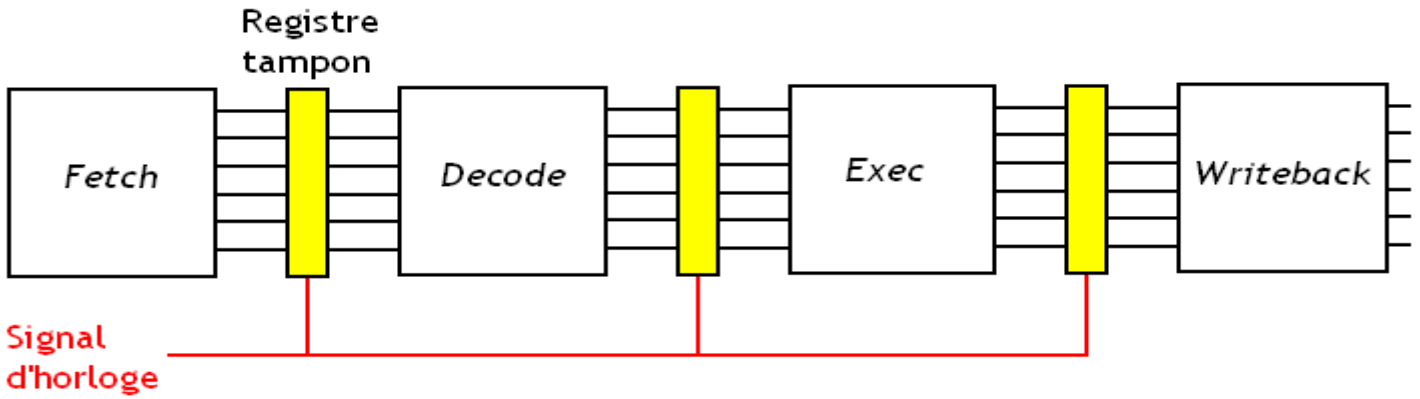
Année 2020-2021

Pr. R. BOUDOUR



Mise en oeuvre de pipeline

- Le pipeline traitant les instructions d'un processeur est plus complexe qu'un pipeline où les instructions avanceraient d'étage en étage : il comporte des étapes d'interaction avec les registres, la mémoire, ual, et souvent la possibilité de bloquer la progression des données pour attendre la résolution d'une instruction.
- Il reste que les données doivent passer d'un étage à un autre. Pour ce faire, il existe trois grands types d'implémentations, parmi lesquelles les pipelines synchrones ;



Mise en oeuvre de pipeline

- Les différents étages sont séparés par des registres tampons, qui sont tous reliés au signal d'horloge du processeur.
- A chaque cycle d'horloge, les registres deviennent accessibles en écriture, ce qui fait que les données passent à l'étage suivant.
- C'est de loin la technique la plus utilisée.

Aléas du pipeline

- **Pourquoi ne peut-on pas obtenir un CPI = 1 ?**
- Plus le pipeline est long, plus le nombre de cas où il n'est pas possible d'atteindre la performance maximale est élevé
- Il existe 3 principaux cas où la performance d'un processeur pipeliné peut être dégradé. Ces cas de dégradation de performance sont appelés aléas.

Aléas du pipeline

- les aléas empêchent l'instruction suivante de s'exécuter au cycle prochain comme prévu :
  - **Les aléas structurels :**
    - Ils interviennent lors des conflits de ressource.
    - Deux instructions différentes utilisent le même matériel dans le même cycle
  - **Les aléas de données :** Dépendance de données
    Ils interviennent lorsqu'une instruction dépend du résultat d'une instruction précédente.
  - **Les aléas de contrôle :** Ils interviennent lors de l'exécution des instructions de branchement et des instructions modifiant le PC

Aléas du pipeline

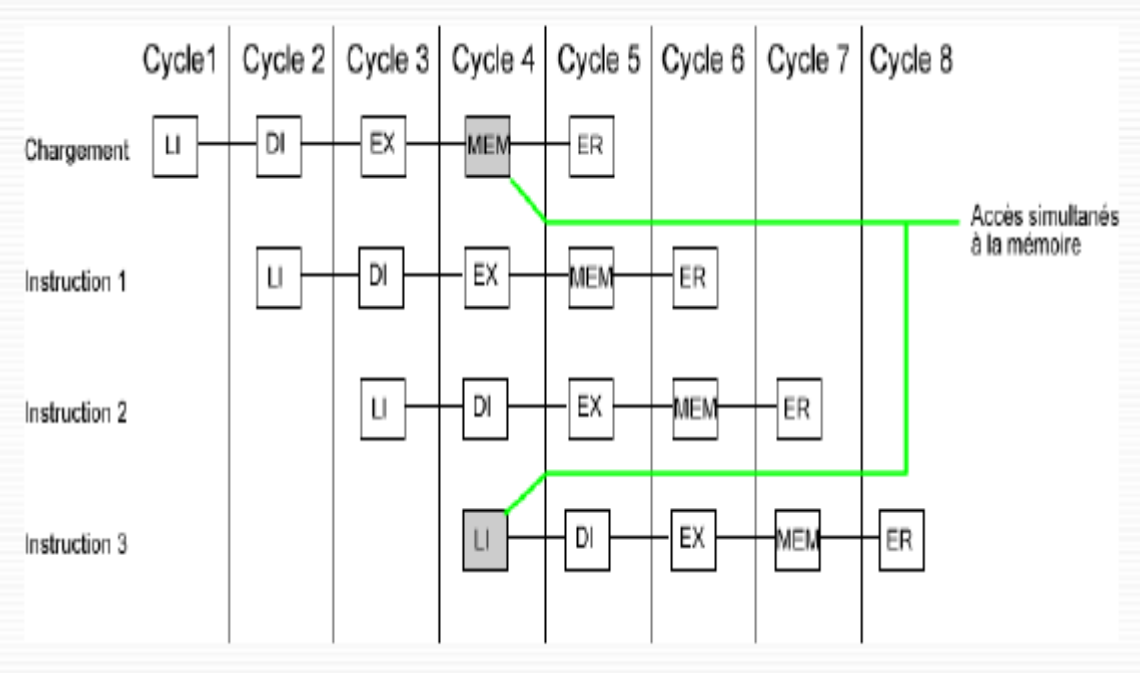
- **Conséquences possibles :**
  - Blocage du pipeline
  - Augmentation du CPI
- **Solution générale :**
  - Bloquer l'instruction qui pose problème et toutes celles qui suivent dans le pipeline jusqu'à ce que le problème se résolve.
  - Insertion de bulles ou des instructions NOP (No operation)

Aléas structurels

- Cet aléa correspond au cas où deux instructions ont besoin d'utiliser la même ressource du processeur (conflit de ressources matérielles).
  - Il s'explique par le fait qu'une unité fonctionnelle n'est pas complètement pipelinée.
  - Une ressource n'a pas été suffisamment dupliquée
  - Exemple d'un bus d'écriture unique dans les registres et un besoin d'écriture simultané provoqué par 2 instructions durant le même cycle d'horloge.
  - Ce problème est généralement résolu en séparant la mémoire en mémoire d'instructions et mémoire de données.

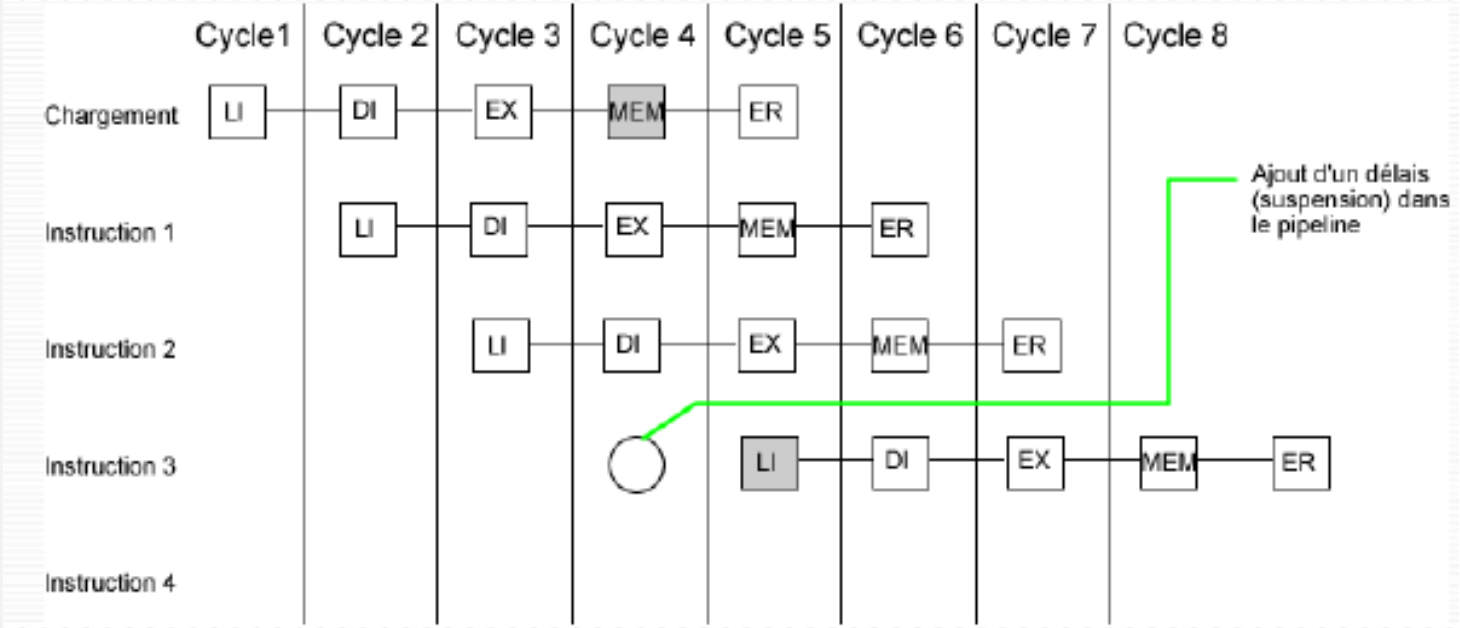
Aléas structurels

- Illustration d'un aléa structurel dans un pipeline



Aléas structurels

- Solution d'un aléa structurel



Aléas structurels

- Autres **solutions**
  - On peut régler un aléa structurel en fournissant un accès mémoire séparé pour les instructions ou diviser le cache en caches séparés pour instructions et données
  - Utilisation des registres pour diminuer le nombre d'accès mémoire
  - Aussi changer l'ordre des instructions (compilateur) en exploitant les instructions sans variables ou constantes

Aléas de contrôle

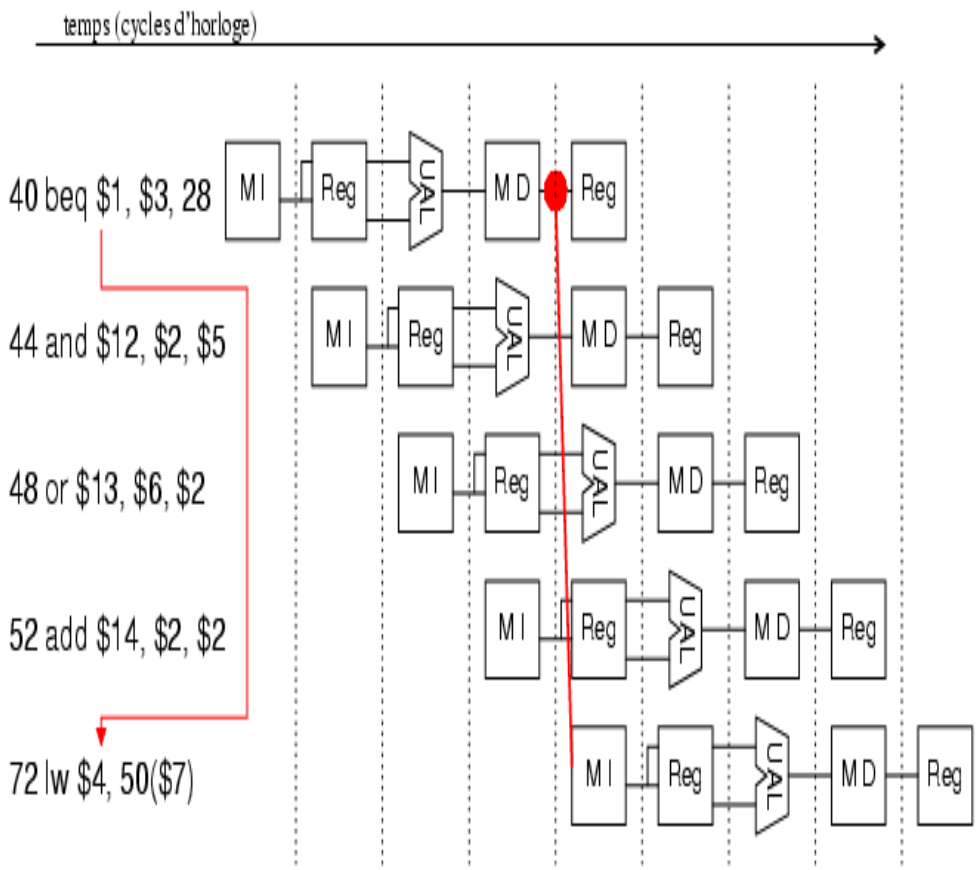
- Ils se produisent chaque fois qu'une instruction de branchement est exécutée.
- Lorsqu'une instruction de branchement est chargée, il faut normalement attendre de connaitre l'adresse de destination du branchement pour pouvoir charger l'instruction suivante.
- Les instructions qui suivent le saut et qui sont en train d'être traitées dans les étages inférieurs le sont en général pour rien. Il faudra alors vider le pipeline.
- Ces aléas peuvent être plus pénalisants que les aléas de données.

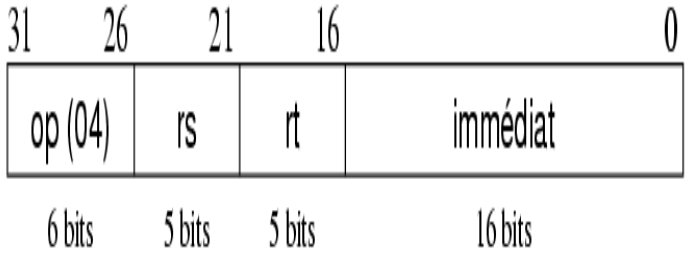
  De 10% à 30% de réduction de performance

Aléas de contrôle

- Lorsqu'un aléa se produit, cela signifie qu'une instruction ne peut continuer de progresser dans le pipeline.
- Pendant un ou plusieurs cycles, l'instruction va rester bloquée dans un étage du pipeline, mais les instructions situées plus en avant pourront continuer à s'exécuter jusqu'à ce que l'aléa ait disparu.
- Plus le pipeline possède d'étages, plus la pénalité est grande. Les compilateurs s'efforcent d'engendrer des séquences d'instructions permettant de maximiser le remplissage du pipeline.

Aléas de contrôle





- **beq rs, rt, imm16**
  - mem[PC] Extraire l'instruction
  - Cond ← R[rs] - R[rt] Calcul de la condition
  - if (Cond == 0) adresse suivante
    - PC ← PC + 4 + (SignExt(imm16) × 4)

    else
    - PC ← PC + 4

Aléas de données

Dépendances entre instructions = mêmes opérandes

Les dépendances peuvent entrainer des aléas

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

- *WAR (write after read):* EAL (Ecriture après lecture)
  L'instruction n+x écrit dans un destination que l'instruction n utilise comme source

- *WAW (write after write):* EAE (Ecriture après écriture)
  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

Solutions des aléas de données

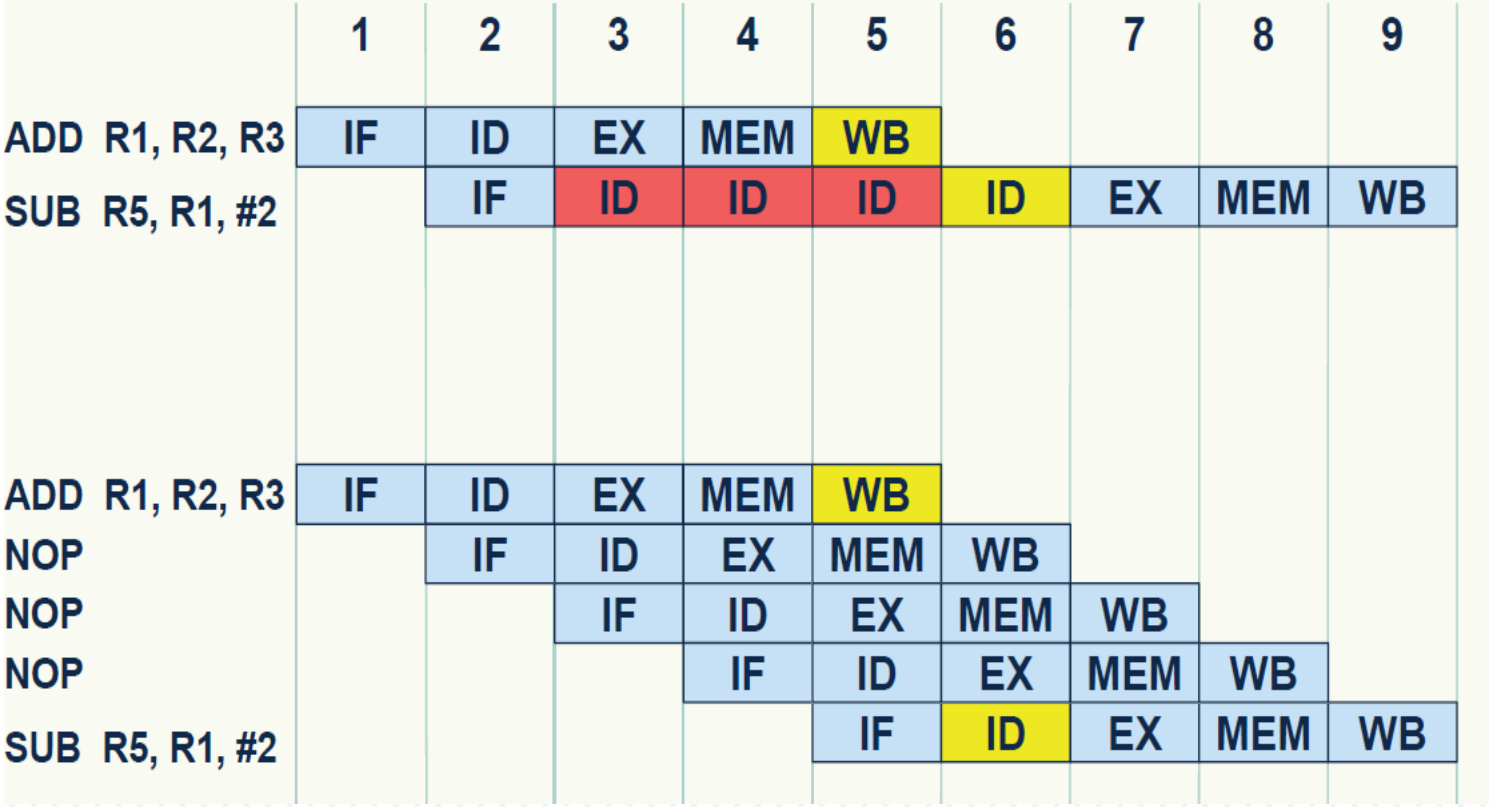
- Méthodes simples (et pénalisantes) :
  - Hardware: arrêter le pipeline (stall / break)
  - Software : insérer des NOPs (no opération)
- Calcul de la pénalité et de l'IPC (nombre d'instruction par cycle)
- Exemple :

IPC = 1 / CPI

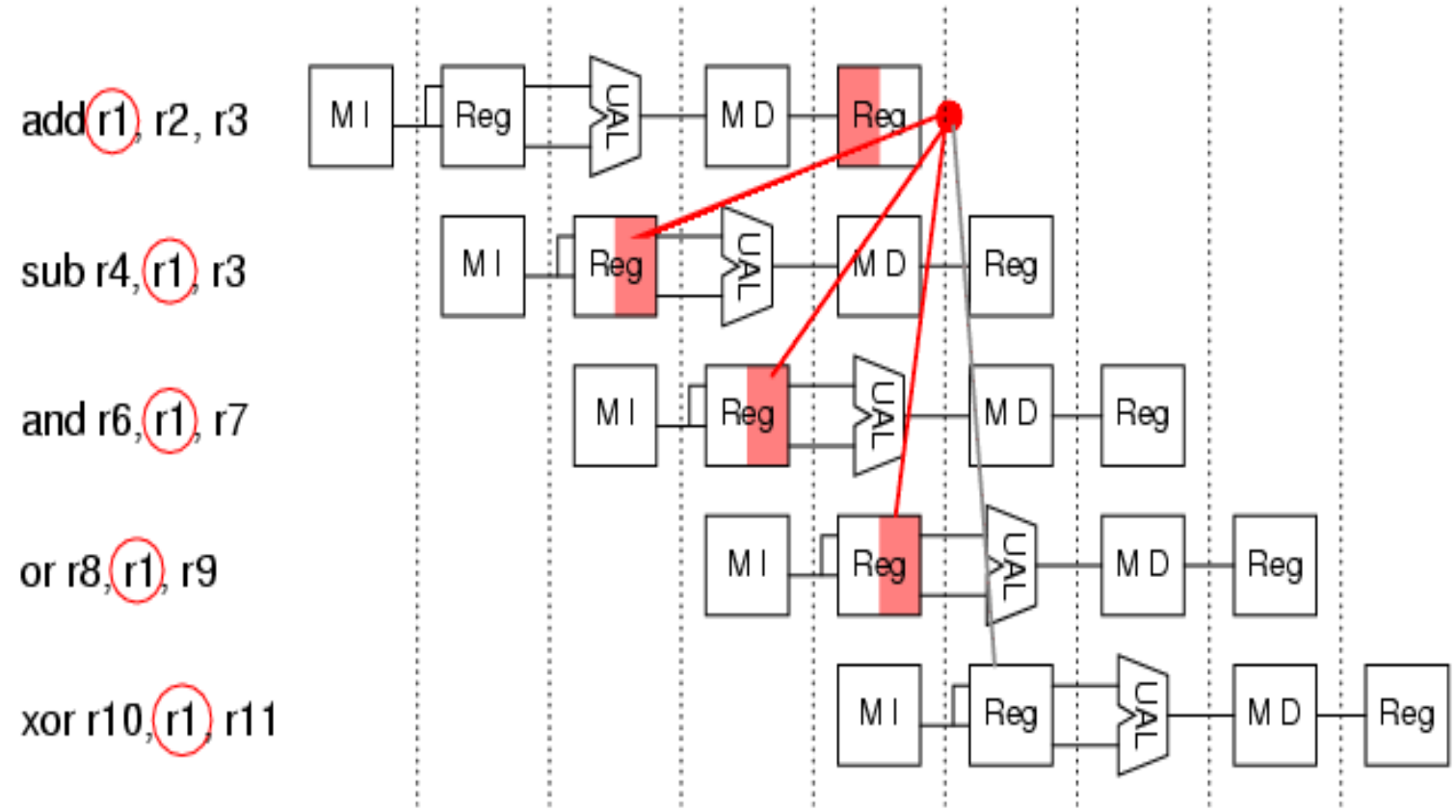
  - Arrêt de 3 cycles pour 20% des instructions
  - IPC = 1/ (0.8 x 1 + 0.2 x 4) = 0.625 instructions par cycle

Solutions des aléas de données

- **Arrêt de pipeline ou insertion de NOP**

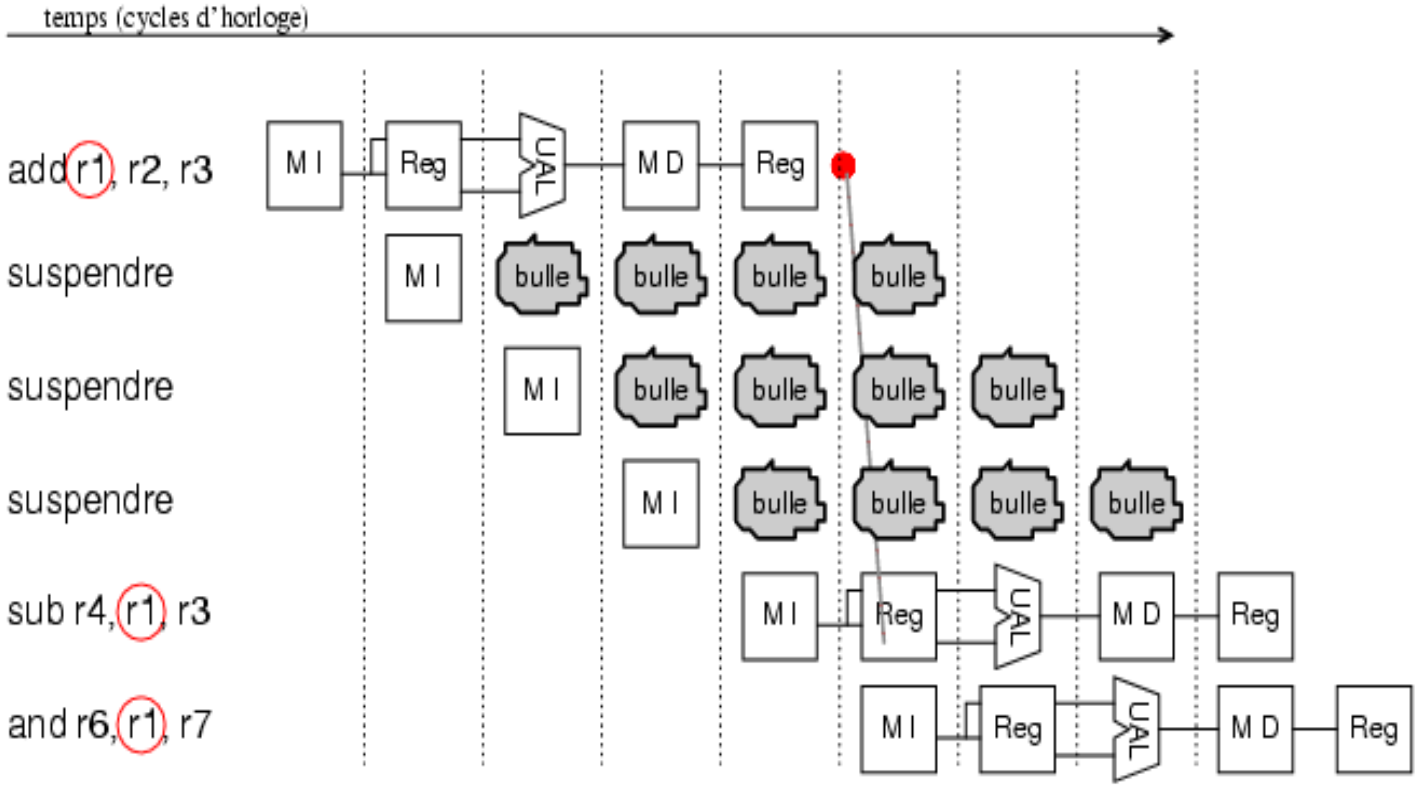


Aléas de données



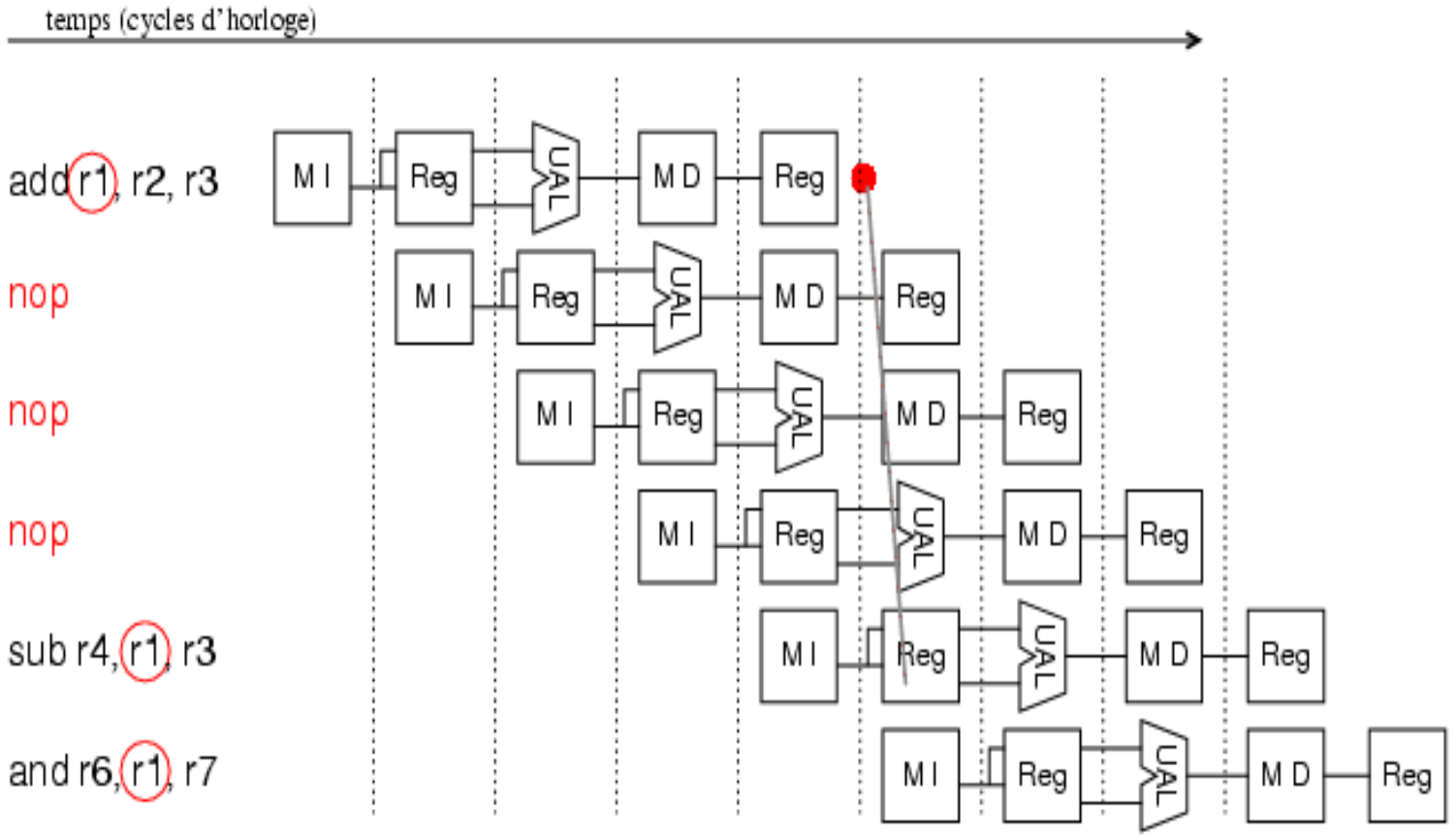
Solution hardware (suspension)

PC inchangé, signaux de contrôle à des valeurs bénignes

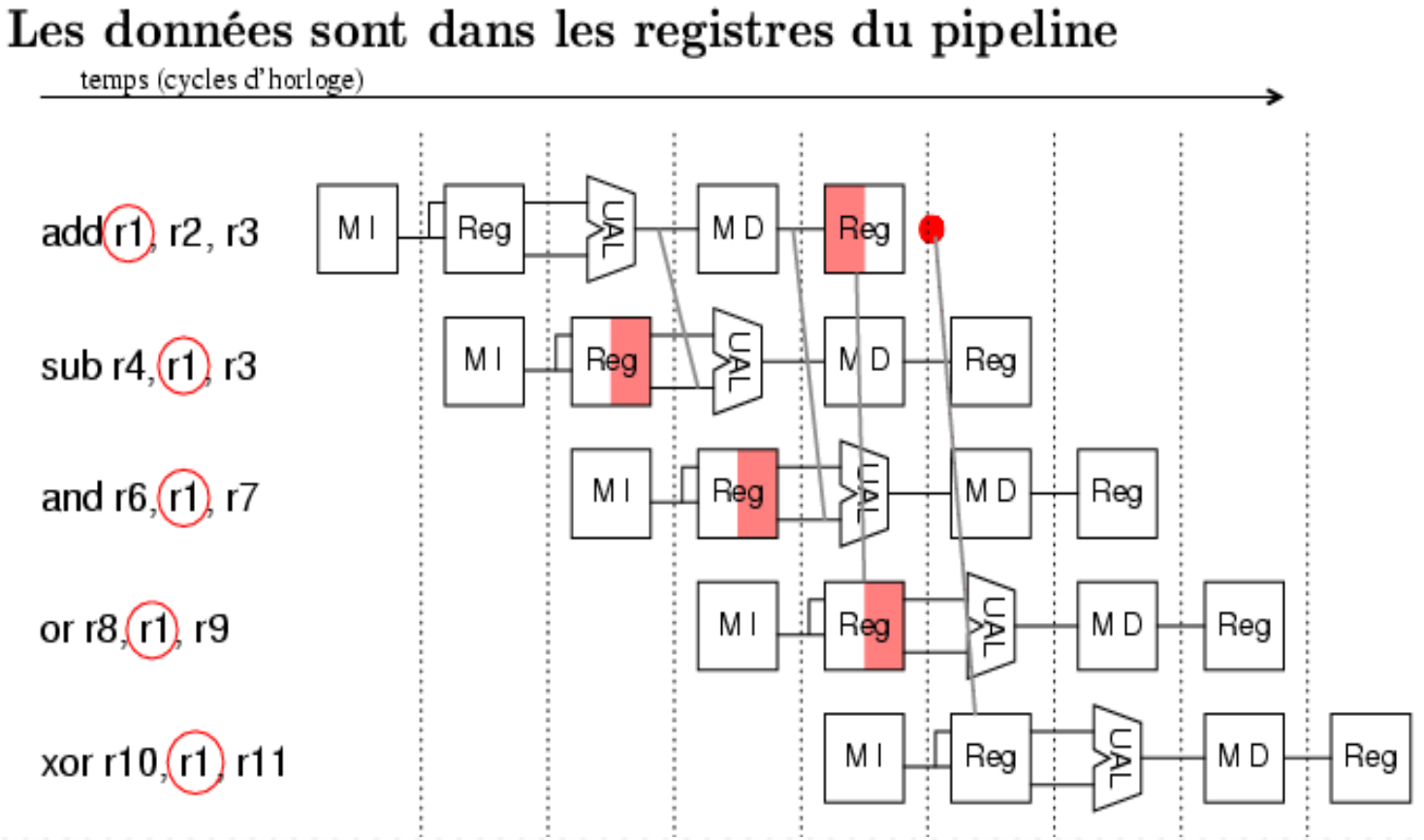


Solution software

Dans le pire des cas on insère des nop :

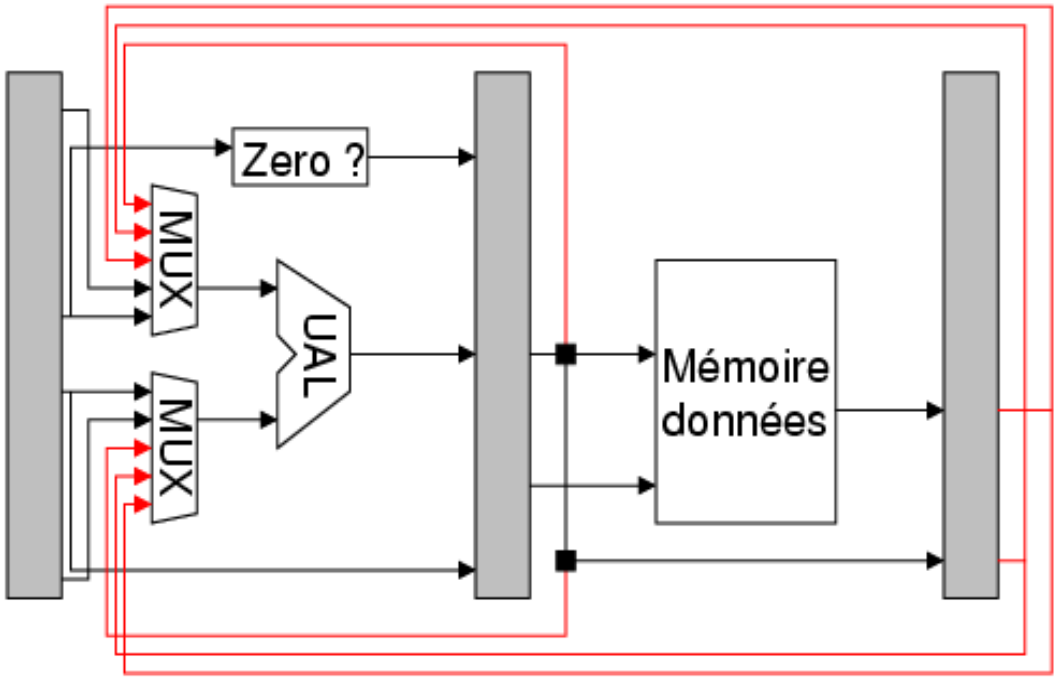


Solution interne : Technique de l'envoi



Les modification du matériel pour l'envoi

- augmenter les multiplexeurs
- supposer : lire un registre pendant l'écriture donne la nouvelle valeur.



Autres méthodes

- 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).

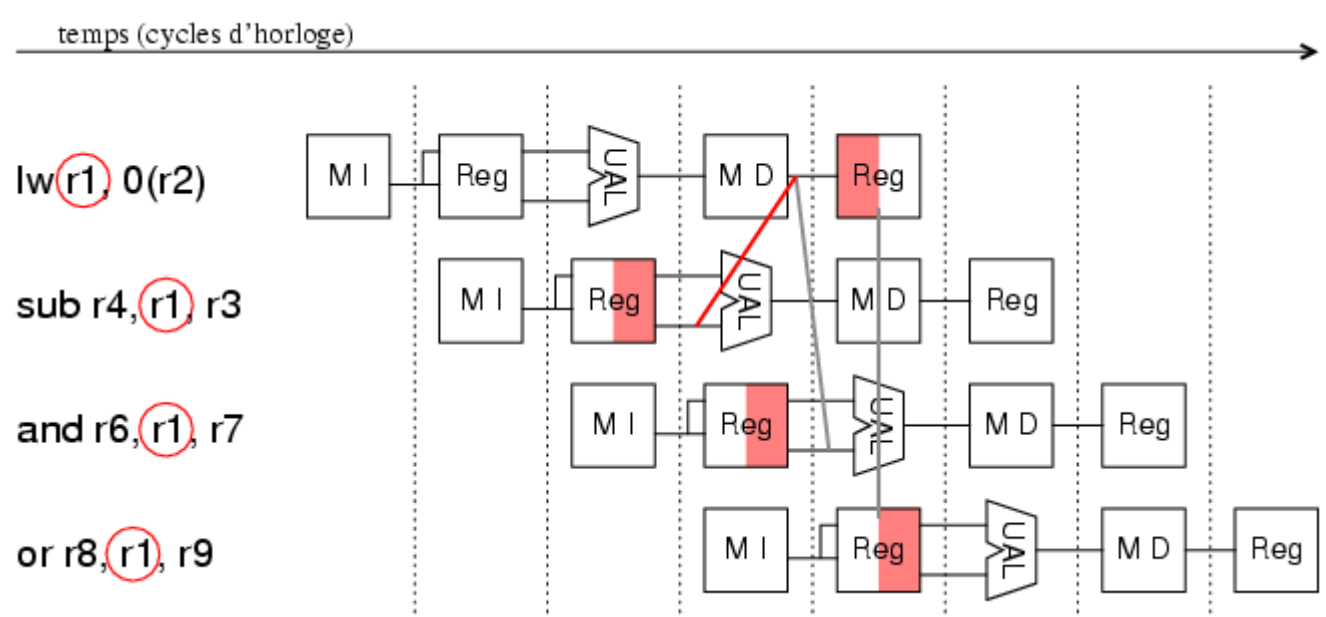
Méthode de Tomasulo

- Différence essentielle avec le score boarding: 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)

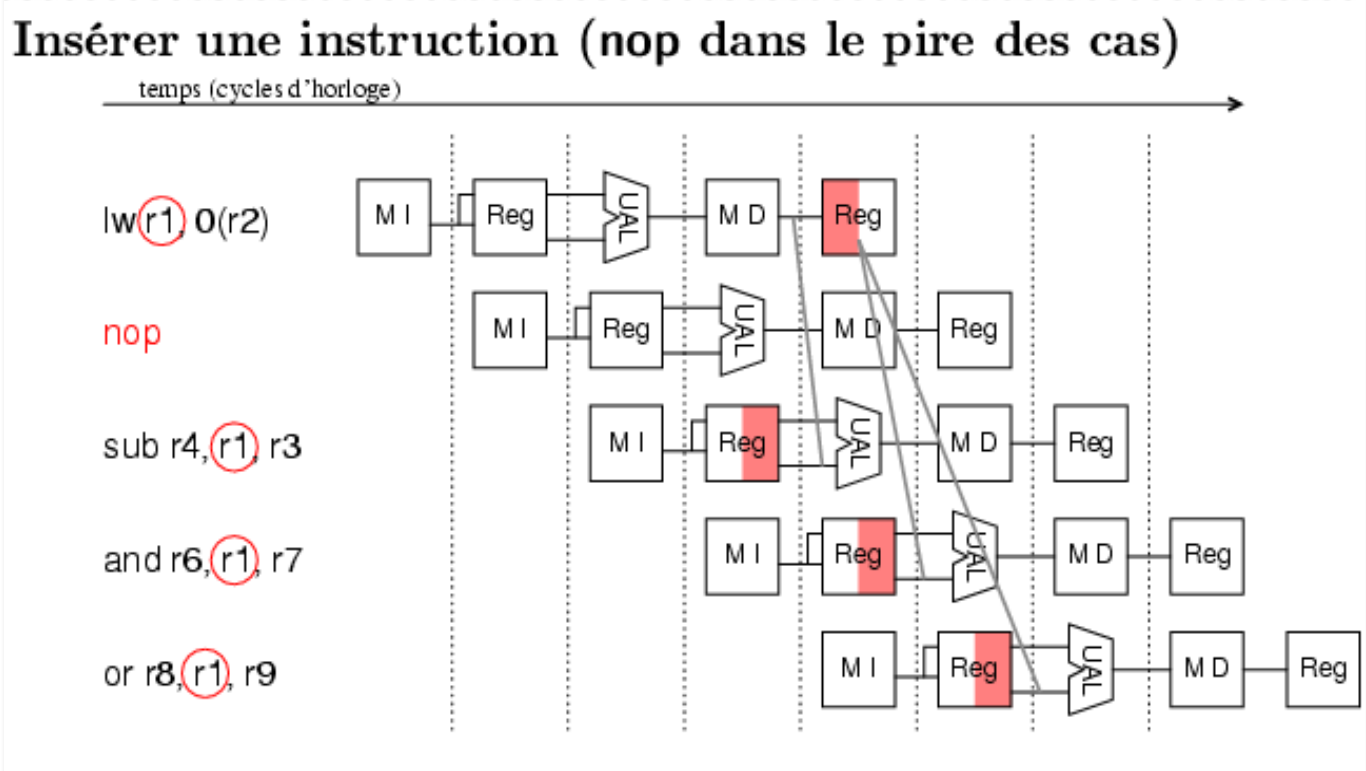
Prédiction dynamique

- 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.

Aléas de données avec un load



Solution logicielle du MIPS



Pipeline : Influence de la profondeur

- **Avantage d'un pipeline long**
  - Plus d'instructions en exécution parallèle
  - Montée en fréquence du processeur facilitée
  - Gain en nombre d'instructions exécutées en un temps donné
- **Inconvénient d'un pipeline long**
  - Une erreur de prédiction est plus coûteuse
  - Plus d'instructions en cours d'exécution à annuler
- **Solution globale**
  - Trouver le bon compromis entre gain d'un côté et perte de l'autre
  - Améliorer les algorithmes et unités de prédiction de branchement

Conclusion

- Pipelining attempts to maximize instruction throughput by overlapping the execution of multiple instructions (but not latence of instruction)
- Pipelining offers amazing speedup.
  - In the best case, one instruction finishes on every cycle, and the speedup is equal to the pipeline depth.
- Classic five-stage pipeline for RISC

  IF – ID – EX – MEM - WB
- Pipeline hazards limit ideal pipelining structural/data/control hazard

Questions

Q1. Pourquoi le pipelining améliore-t-il la performance ?

Q2. Quelles sont les limites de l'amélioration de performance offerte par le pipelining ?

Q3. Donner deux techniques proposant des solutions aux aléas : l'une logicielle et l'autre matérielle


THANK YOU