le cnam

Limitations du Pipeline

NSY 104

Dépendances

- Détection des dépendances
  - 2 instructions parallèles
    - peuvent s'exécuter dans le pipeline
    - sans créer de suspension
  - 2 instructions dépendantes ne sont pas parallèles
    - Elles doivent s'exécuter dans l'ordre
    - Avec un recouvrement partiel possible
- 3 types de dépendances
  - de nom
  - de donnée
  - de contrôle

Dépendances

- Les dépendances peuvent causer des aléas
- Les aléas peuvent causer des suspensions

- Propriétés de l'organisation du pipeline
  - Découpage des étages par exemple

- Une dépendance
  - Indique la possibilité d'un aléa
  - Détermine l'ordre d'exécution des opérations
  - Limite la quantité de parallèlisme

- Objectifs:
  - Identifier la dépendance et éviter l'aléa
  - Eliminer la dépendance (réorganiser le code)

Les dépendances de données

- Interviennent quand
  - Une instruction *a* utilise le résultat de l'instruction *b*
  - Une instr. *b* a une dep. de donnée avec l'instr. *c* et que *c* a une dep. de donnée avec *a*.
    - Transitivité
    - Chaines de dépendances
- Peuvent être difficiles à identifier
  - Circulation via des cases mémoires
    - 10 (R1) = 20 (R2)

Les dépendances de nom

- Interviennent quand des instructions indépendantes utilisent
  - le même nom de registre
  - la même case mémoire
- Anti-dépendance entre 2 instructions *a* et *b*
  - Quand *b* écrit dans un emplacement lu par *a*
- Dépendance de sortie
  - Quand *a* et *b écrivent dans le même emplacement*
- Le renommage permet de contourner ces obstacles
  - Changer les contenants
  - A la volée ou statiquement, à la compilation

Les aléas de données

- LAE (lecture après écriture)
  - Le plus fréquent
- EAL (écriture après lecture)
- EAE (écriture après écriture)

- LAL (lecture après lecture) n'est pas un aléa
  - Pas de modification

Les aléas de données

- Code
  - ADD R1,R2, R3
  - SUB R4, R1, R5
  - …
- Sans précaution, une architecture pipelinée permettrait à SUB de lire R1 alors que ADD n'a pas rangé son résultat.
  - Valeur erronée
- Solution (matérielle): l'envoi
  - « dérivation »

Les aléas

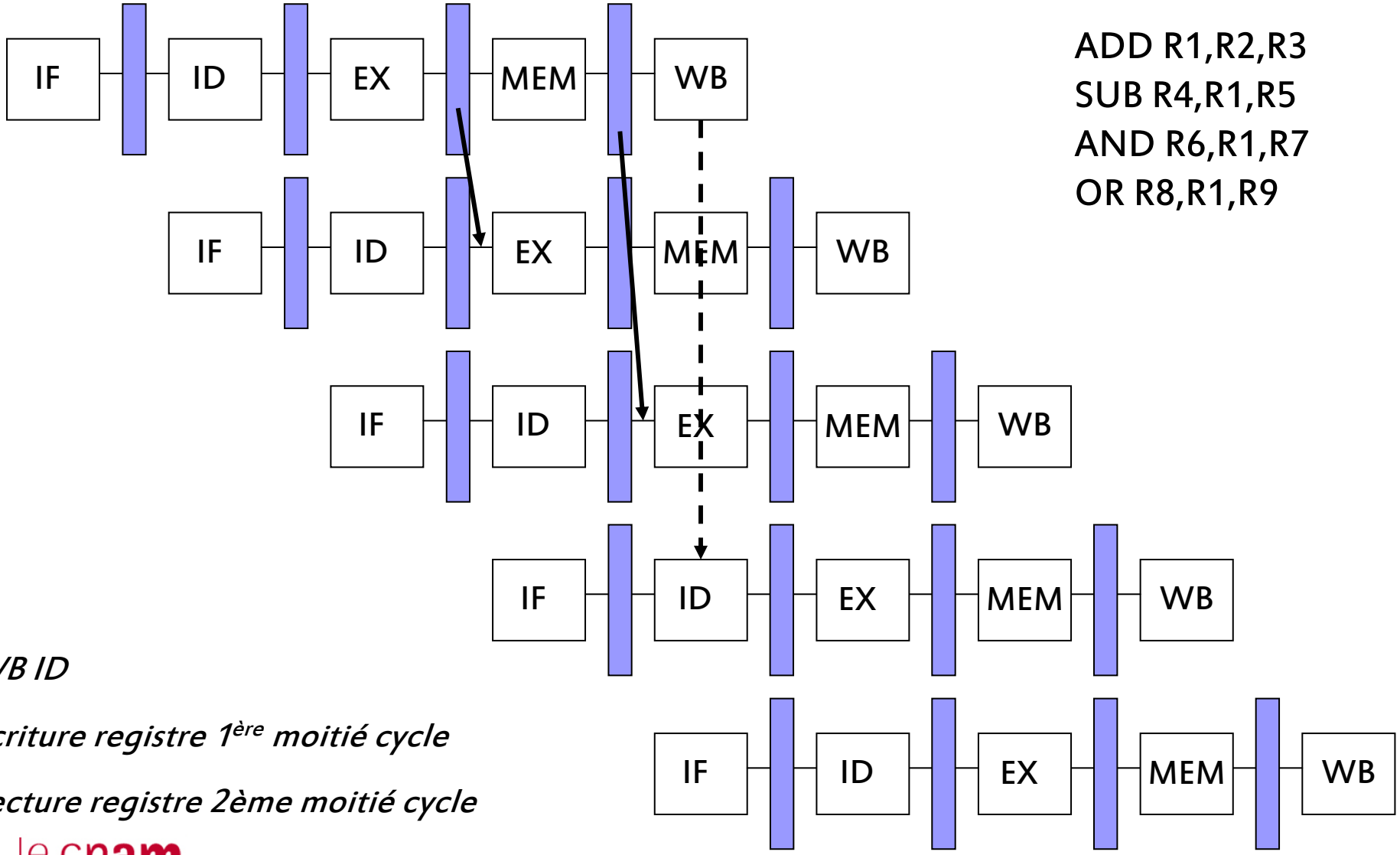
- Rappel
  - ADD R1,R2, R3
  - SUB R4, R1, R5
  - ADD R12, R5, R4
  - ...

- Les problèmes viennent
  - des dépendances entre instructions
  - qui empêchent leur parallèlisation
- Identification des aléas lors de DI

Les aléas de données

- La valeur est disponible à la fin de EX (étage 3)
- On créé un moyen pour communiquer ce résultat en avance
  - chemin matériel
- Fonctionnement
  - Le résultat UAL est toujours renvoyé aux entrées UAL
    - Depuis le registre EX/MEM
    - Depuis le registre MEM/WB
  - La logique de contrôle détecte ce mécanisme
    - et sélectionne la valeur envoyée
    - plutôt que celle lue dans le registre source

Les aléas de données



ADD R1,R2,R3
SUB R4,R1,R5
AND R6,R1,R7
OR R8,R1,R9

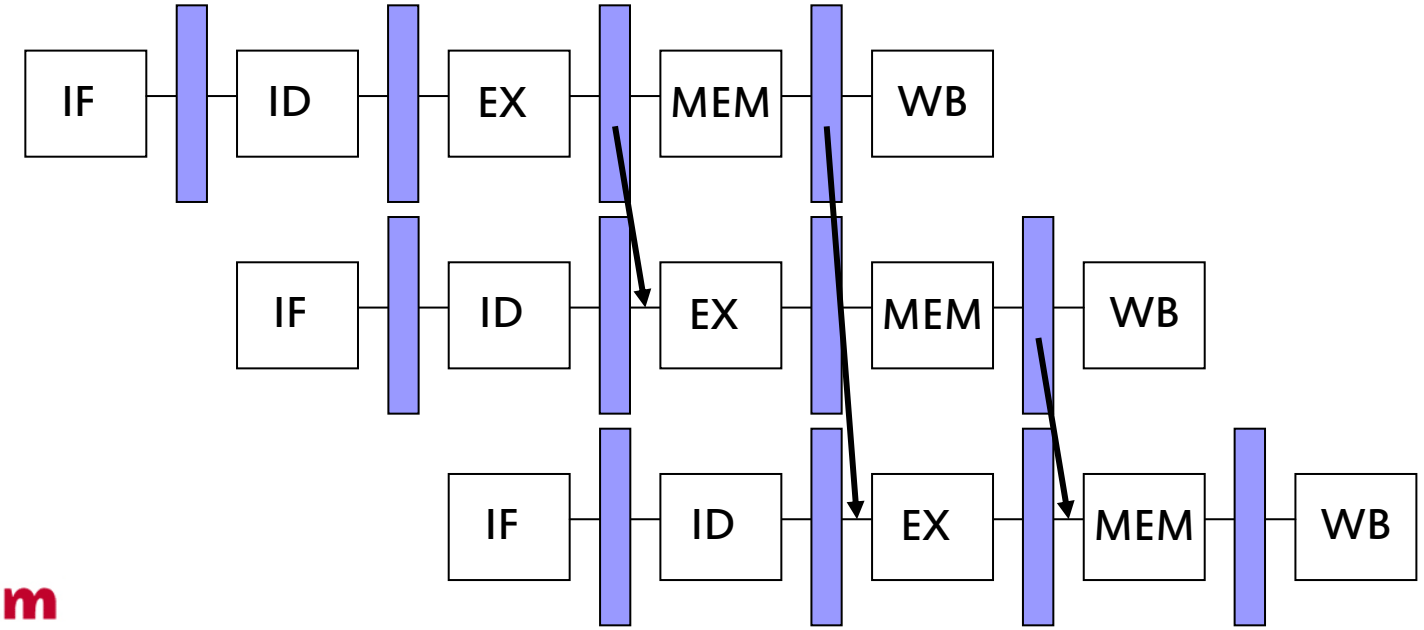
WB ID

Ecriture registre 1ère moitié cycle

Lecture registre 2ème moitié cycle

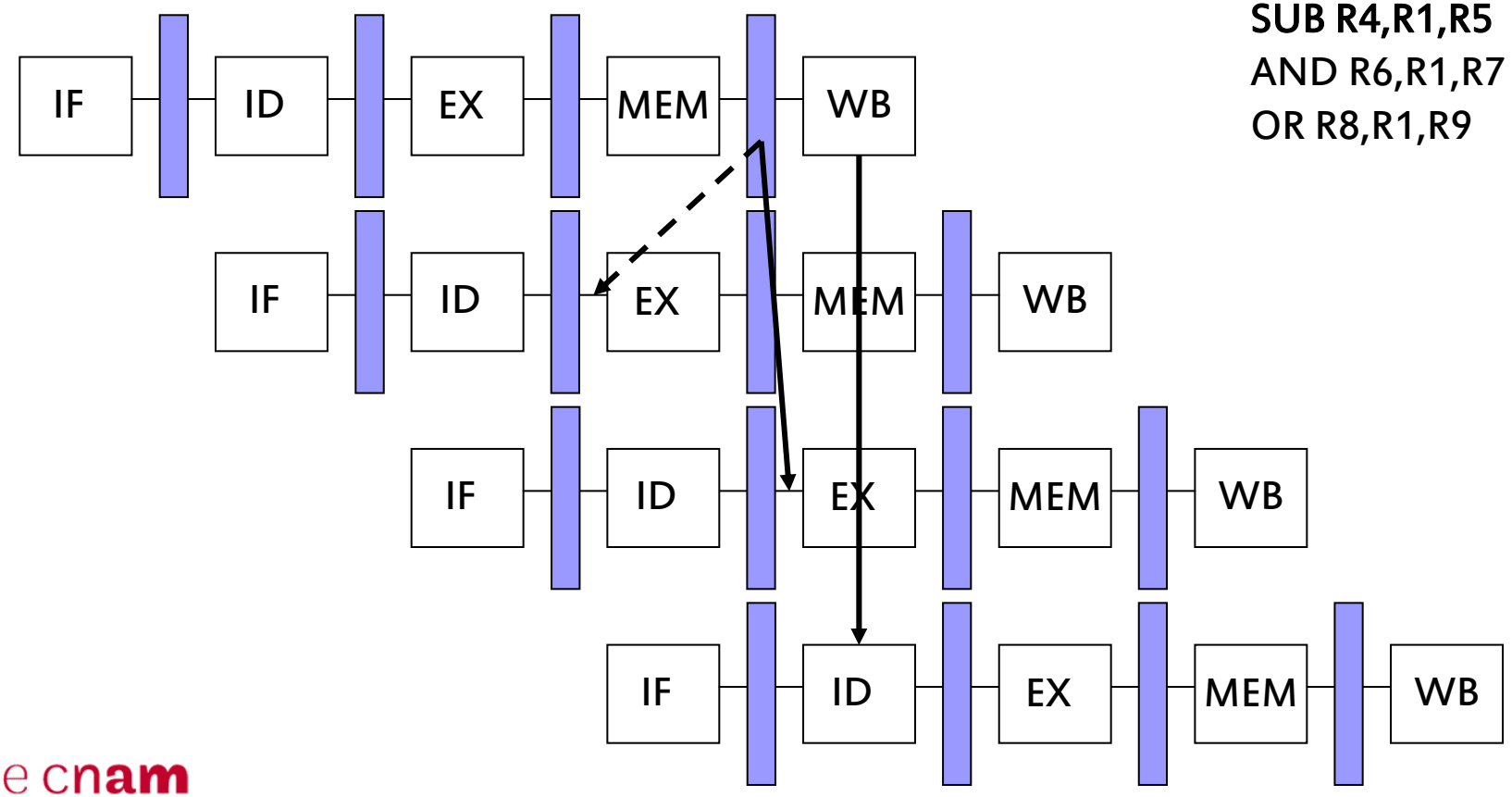
Les aléas de données

- Cela permet d'éviter des suspensions du pipeline
- On peut généraliser ces chemins
  - vers d'autres étages du pipeline
    - ADD R1,R2,R3
    - LD R4,0(R1)
    - ST R4,10(R1)



Les aléas de données

- Certains aléas ne peuvent être résolus
  - Suspension du pipeline incontournable
  - Exemple



Les aléas de données

- Insertion d'un cycle de suspension (No-Op)

LD R1, 0(R2)
SUB R4,R1,R5
AND R6,R1,R7
OR R8,R1,R9

IF	ID	EX	MEM	WB				
	IF	ID	*Noop*	EX	MEM	WB		
		IF	*Noop*	ID	EX	MEM	WB	
			Noop	IF	ID	EX	MEM	WB

Les aléas de contrôle

- Peuvent être plus pénalisants que les aléas de données
- De 10 à 30% de réduction de performance

- Le branchement
  - Modifie le CP avec autre chose que CP+4 (longueur de l'instruction)
    - BRANCHEMENT PRIS
    - Répétition de l'étape IF
      - Rupture de la séquence
  - Laisse l'exécution se dérouler en séquence (CP+4)
    - BRANCHEMENT NON PRIS
    - Pas de suspension
      - L'instruction qui a été chargée ensuite est la bonne.

Les aléas de contrôle

- Minimiser les pénalités des branchements
  - Par le vidage du pipeline
    - ou « gel » du pipeline
    - Consiste à supprimer (suspendre) toutes les instructions après le branchement
    - jusqu'à la connaissance de la destination

IF	ID	EX	MEM	WB				
	Noop	IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB

Les aléas de contrôle

- Par le schéma de prédiction non pris
  - Traite tous les branchements comme non pris
    - Chargement en séquence
  - Si un branchement pris apparaît
    - Annulation de l'instruction chargée
    - Chargement de la cible

IF	ID	EX	MEM	WB				
	IF	*Noop*	*Noop*	*Noop*	*Noop*			
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	

Les aléas de contrôle

- Par le schéma de prédiction pris
  - Traite tous les branchements comme pris
  - Dès que le branchement est décodé
    - On lit l'instruction destination
    - On commence l'exécution
  - Dans notre exemple à 5 étage
    - Pas d'intérêt
    - On ne connait pas l'adresse de destination avant de savoir s'il pris ou non
  - Dans d'autres schémas de pipeline

Parallélisme des instructions

- Techniques matérielles ou logicielles pour paralléliser
  - Techniques matérielles, dynamiques
    - majoritaires
  - Techniques logicielles, statiques
    - compilateur
    - informatique enfouie, IA64, Itanium
- Parallélisme dans un bloc d'instructions sans branchement est faible
  - Dépendances forte à l'intérieur du bloc
- Pour un MIPS, la fréquence moyenne des branchements est de 15 à 25%
  - Entre 2 branchements, 4 à 7 instructions en séquence

Parallélisme des instructions

- On peut exploiter le parallélisme de boucle
  - Parallélisme entre les itérations d'une boucle
- Pour en dégager un parallélisme d'instructions

- Méthodes
  - Dérouler la boucle
    - À la compilation
    - Dynamiquement (matériel)
  - Utilisation d'instructions vectorielles
    - Opérant sur une suite d'éléments de données

L'ordonnancement dynamique

- Limitation du pipeline
  - Lancement et d'exécution ordonnés des instructions
    - Si une instruction est suspendue, tout le pipeline est affecté
    - Pas de lancement possible d'une instruction suivante

- Principe de l'ordonnancement dynamique
  - Casser ce blocage en permettant le lancement d'instructions sans dépendance
  - *« out of order » (OOO)*

- Exemple
  - DIV R0, R2, R4
  - ADD R10, R0, R8
  - SUB R12, R8, R14

L'ordonnancement dynamique

- Nouvel aspect à gérer
  - Le lancement non ordonné entraîne une terminaison non ordonnée
- Apparition des aléas qui n'existent pas dans le pipeline ordonné
  - EAL
  - EAE
- Division de l'étage DI en 2 parties
  - Lancement
    - Décodage de l'instruction, test des aléas structurels
  - Lecture opérandes
    - Attente de la fin des aléas de données, lecture des opérandes

L'ordonnancement dynamique

- Par tableau
  - Toutes les instructions traversent l'étage de lancement dans l'ordre.
  - Elles peuvent passer dans l'étage suivant
    - dans l'ordre,
    - être suspendue
    - passer l'une devant l'autre
      - ordonnancement dynamique.
  - La technique du tableau est connue sous le nom de tableau de marques.
    - Le but du tableau est d'exécuter les instructions dès que possible.
    - Si une instruction est suspendue, les suivantes peuvent être exécutées à condition
      - qu'il n'y ait pas de dépendance
      - que les ressources nécessaires soient disponibles

L'ordonnancement dynamique

- Lancement
  - si une unité fonctionnelle est libre pour l'instruction
  - et qu'aucune autre instruction active n'a le même registre destination
  - le tableau lance l'instruction vers l'unité fonctionnelle

- Lecture des opérandes
  - Le tableau de marques contrôle la disponibilité des opérandes sources.
  - Un opérande source est disponible si aucune instruction active lancée auparavant ne va la modifier en écriture
  - ou si aucune unité fonctionnelle active n'est en train d'écrire l'opérande dans le registre.
  - Si les opérandes sources respectent cette condition, le tableau indique à l'unité fonctionnelle qu'elle peut lire les opérandes et commencer l'exécution.
  - Cette étape termine le décodage de l'instruction.

L'ordonnancement dynamique

- Exécution
  - l'unité fonctionnelle commence l'exécution à la réception des opérandes.
  - Quand le résultat est prêt, elle avertit que l'exécution est terminée.
  - Cette étape utilise plusieurs cycles du pipeline.
- Ecriture du résultat
  - une fois que le tableau sait que l'unité fonctionnelle à terminé son exécution.

L'ordonnancement dynamique

- Méthode de Tomasolu
  - Identifier les dépendances d'instructions
    - Permettre l'exécution dès que les opérandes sont disponibles
      - Evite les aléas LAE
  - Renommage des registres
    - Evite les aléas EAL et EAE
  - Bus Commun de Données (BCD)
- Stations de réservation
  - Fournissent le service de renomage de registre
  - Mémorisent les opérandes des instr. en attente de démarrage
    - Dès qu'il est disponible
    - Ne passe plus par les registres originaux

L'ordonnancement dynamique

- Méthode de Tomasolu
  - 3 étapes pour chaque instruction
    - Lancement (Renome les registres – élimine EAL et EAE)
      - Obtenir l'instruction suivante (FIFO)
      - Lancer l'instr. à une station libre
      - Sinon, attendre la libération d'une station (aléa structurel)
      - Si les opérandes ne sont pas dans les registres, observer les unités fonctionnelles qui vont les produire
    - Exécution
      - Tous les opérandes sont disponibles: lancer l'exécution
      - Sinon, continuer de les attendre (élimine LAE)
    - Ecriture du résultat
      - Résultat disponible: écriture sur le BCD
        - sur le banc de registres
        - dans les stations qui l'attendent

L'ordonnancement dynamique

Instruction	Lancement	Exécution	Ecriture
LD F6, 34(R2)	X	X	X
LD F2, 45(R3)	X	X	
MUL F0,F2,F4	X		
SUB F8,F2,F6	X		
DIV F10,F0,F6	X		
ADD F6,F8,F2	X		

Etat quand le premier chargement est terminé et écrit

Champ	F0	F2	F4	F6	F8	F10
Qi	Mult1	Load2		Add2	Add1	Mult2

Nom	Occupé	Op	Vj	Vk	Qj	Qk	A
Load1	N						
Load2	O	Chargement					45+reg[R3]
Add1	O	SUB		Mem[34+reg[R2]]	Load2		
Add2	O	ADD			Add1	Load2	
Add3	N						
Mul1	O	MUL		Reg[F4]	Load2		
Mul2	O	DIV		Mem[34+reg[R2]]	Mult1		

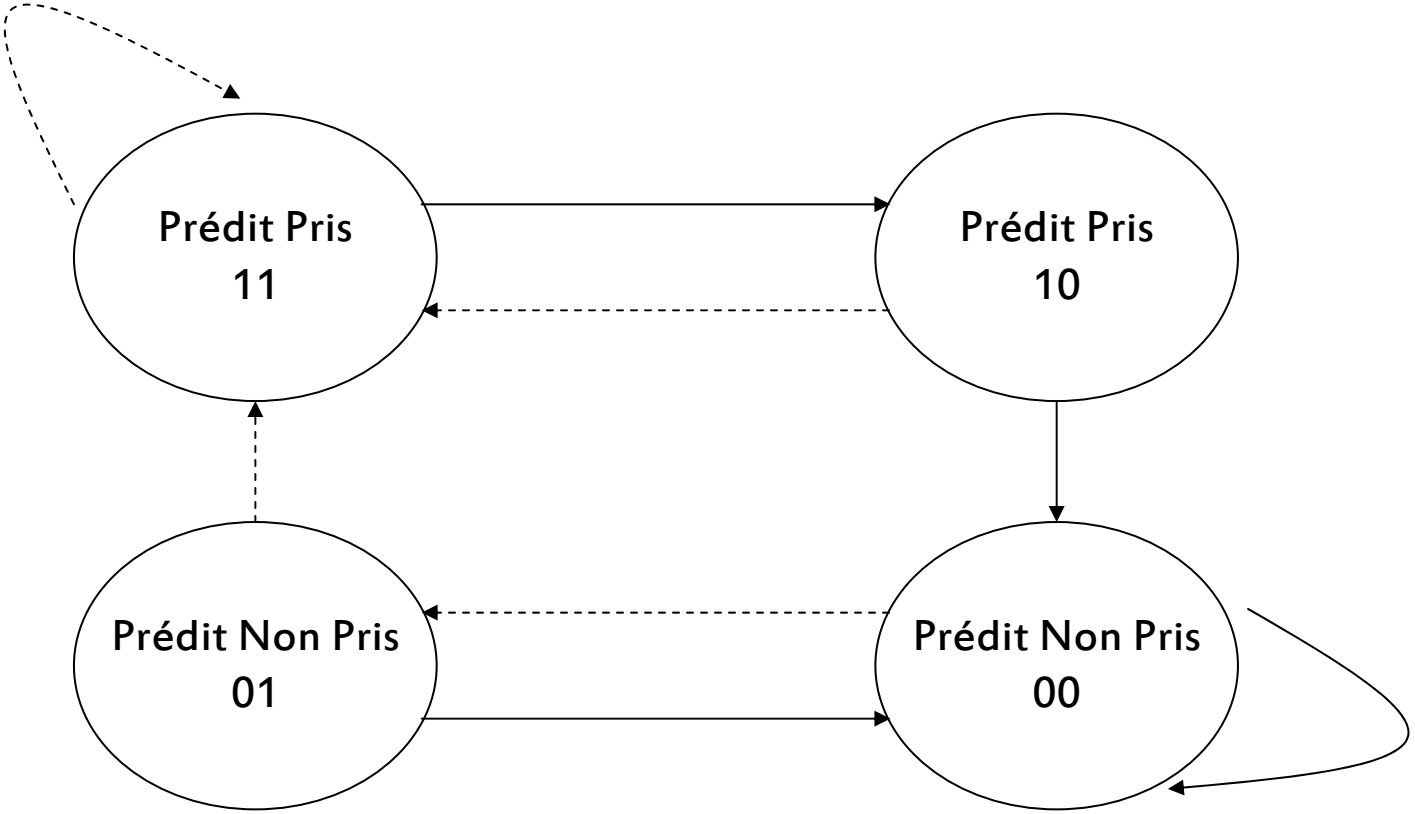
La prédiction de branchement

- Réduire les suspensions dues aux dépendances de contrôle

- Solution la plus simple:
  - Tampon de prédiction de branchement
    - Table des historiques des branchements
  - Petite mémoire indexée par la partie basse de l'adresse de l'instruction de branchement
    - Contient un bit pour indiquer la prise récente de ce branchement

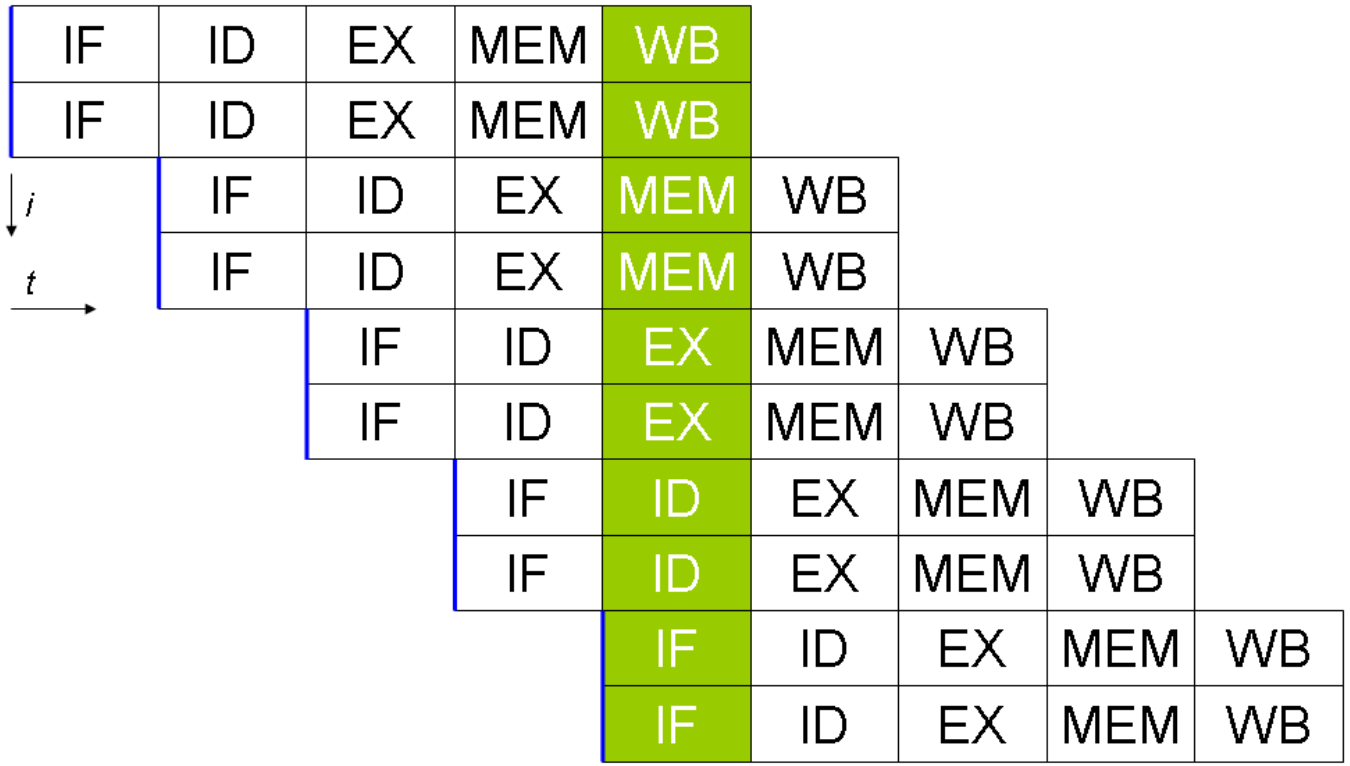
La prédiction de branchement

- Version améliorée avec un prédicteur à 2 bits
  - Nécessite 2 erreurs consécutives pour être permuté



Les processeurs superscalaires

- Permettent l'exécution de plusieurs instructions en même temps
  - On duplique le matériel pour avoir plusieurs chemins de traitement
  - Généralement combiné au pipeline



Les processeurs superscalaires

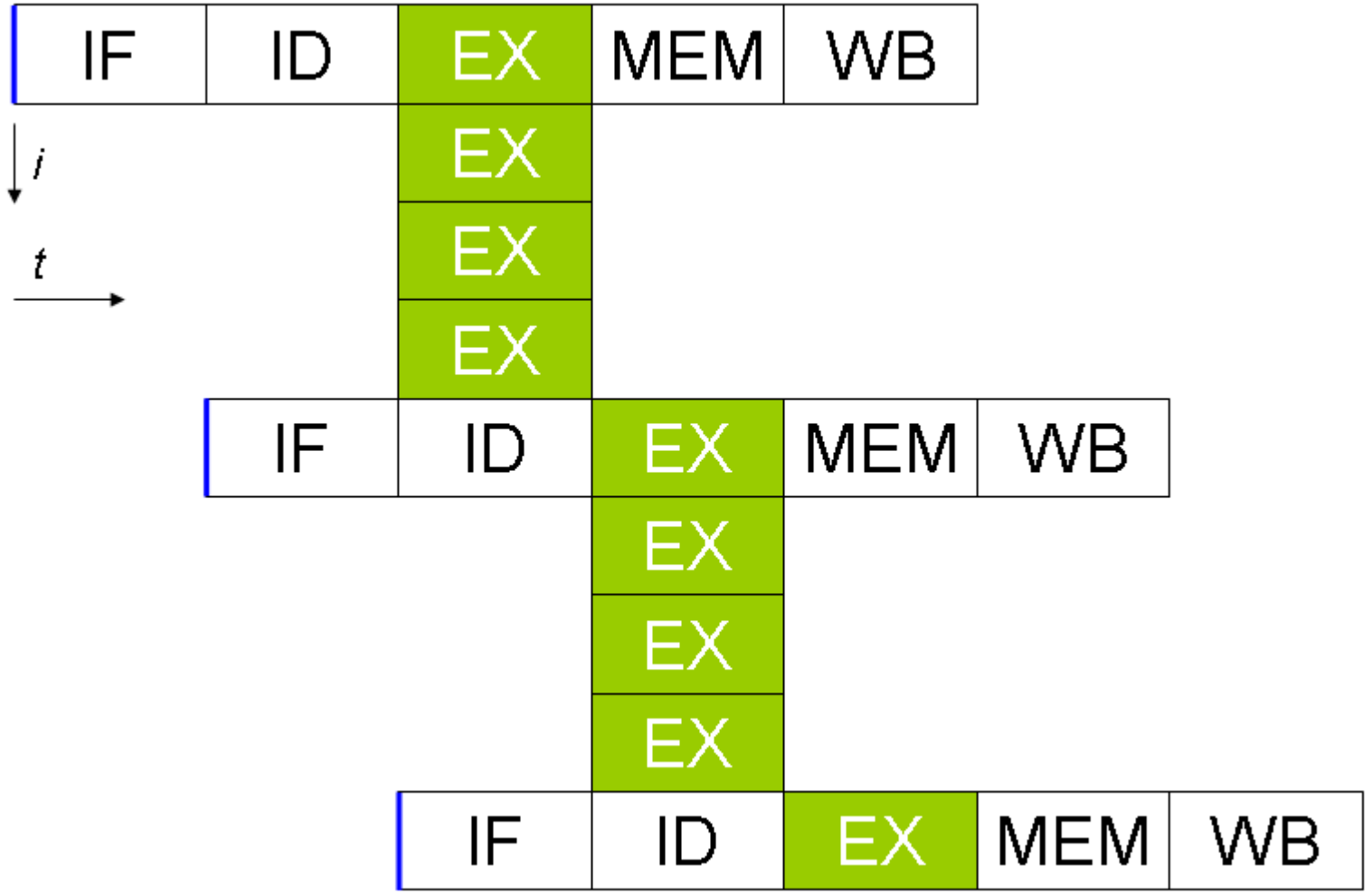
- La nature des limitations est comparable aux architectures précédentes
  - Degré de paralèllisme intrinsèque du code
  - Le coût de la logique de contrôle
  - Le traitement des instructions de branchement

- Optimisation par ordonnancement
  - Statique (compilateur)
  - Dynamique (matériel)

L'approche VLIW

- Very Long Instruction Word
  - Alternative au superscalaire pur
  - S'appuie sur les compilateurs pour réorganiser le code
    - En groupant des instructions parallèles
    - En taggant le dépendances possibles
    - ...
  - Longueur de mot de 64 bits et plus
  - Un mot regroupe plusieurs instructions parallèles
  - Le matériel dispose de plusieurs unités fonctionnelles indépendantes

L'approche VLIW



L'approche VLIW

- Compatibilité du code binaire
  - Compilation très proche
    - Du jeu d'instruction
    - De la structure pipeline
  - Différents matériel = compilations différentes