Jean-luc.dekeyser@lifl.fr
Version 2013

FONCTIONNEMENT PIPELINE

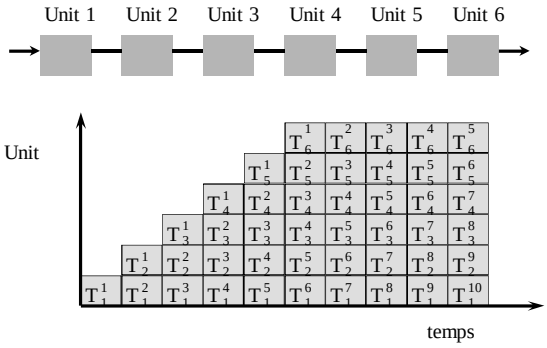
Système pipeline

- Superposition dans le temps
  - Augmentation des performances
  - Parallélisme temporel

Décomposition

- Division d'une tâche en sous tâches
- 1 unité logique / sous tâche
- Les sorties d'une unité sont connectées aux entrées de la suivante
- Les données entrent par le premier étage et sortent par le dernier.

Principe du pipeline



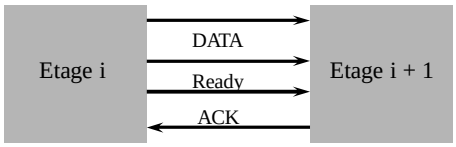
Définitions

- Latence du pipeline : temps (en cycles) entre deux instructions consécutives
- Débit du pipeline : Nombre d'instructions exécutées par cycle, on appelle aussi le degré d'un processeur superscalaire
- Conflit sur ressource : Deux ou plus instructions demandent l'utilisation de la même unité au même instant

Transfert des données pipeline

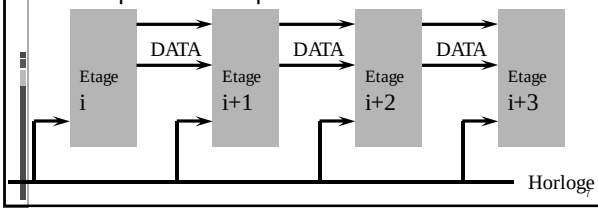
Modèle Asynchrone:

- Mécanisme de "handshacking" entre chaque couple d'unités



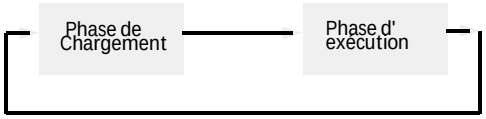
Modèle Synchrone

- Horloge contrôle le transfert des données entre 2 unités
- Pipe d'instructions
- Pipe arithmétique



Pipeline d'instructions:

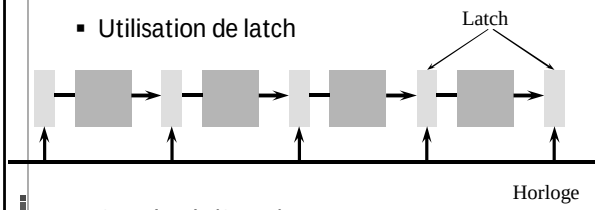
- Fetch / Execute 2 unités séparées



- On réalise les deux phases en même temps sur des instructions différentes.
- Il suffit de connaître la prochaine instruction: instruction prefetch

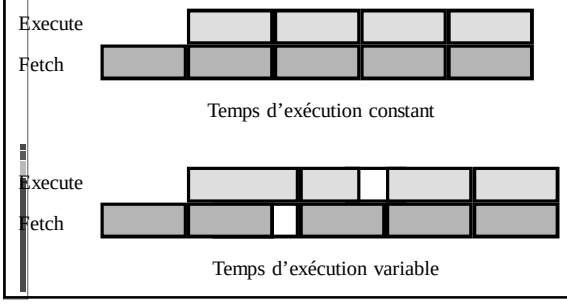
Construction

- Utilisation de latch



- Sans latch (CRAY)
- On parle de Maximum Rate pipe-line

Pipeline 2 étages



Speed-up

- Sur un pipe n-étages, il faut:
  - n cycles pour obtenir le 1er résultat
  - pour s-1 résultats, il faut ensuite s-1 cycles.
  - Soit n+s-1 cycles pour s résultats (sur un processeur scalaire, il faut ns cycles)
- Le Speed-up est égal à

$$S = T_{scal} / T_{pipe} = \frac{ns}{n+s-1}$$

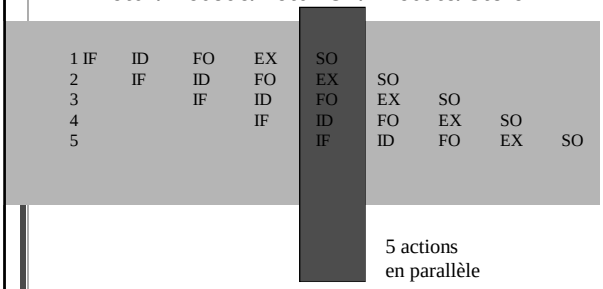
- $S \to n$ quand $s \to +\infty$

Augmentation du parallélisme

- L'instruction prefetch permet au plus de doubler la vitesse du processeur.
- Exécution souvent plus long que le chargement.
- La recherche de parallélisme nécessite un découpage plus fin
- Souvent en 4 ou 5 étapes successives

5 étages

- Fetch/ Decode/ Fetch OP/ Execute/ Store

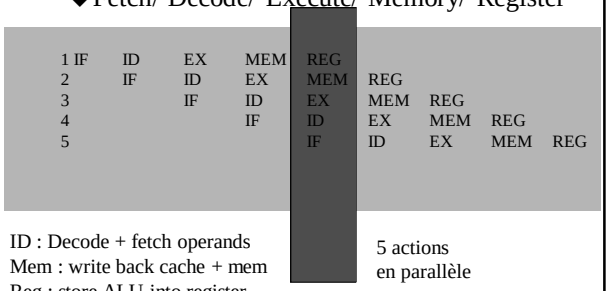


5 actions
en parallèle

13

Exemple le MIPS

◆ Fetch/ Decode/ Execute/ Memory/ Register



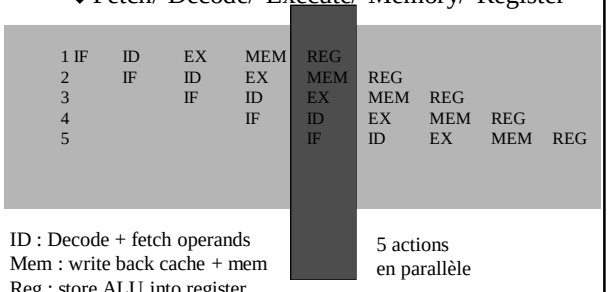
ID : Decode + fetch operands
Mem : write back cache + mem
Reg : store ALU into register

5 actions
en parallèle

14

Exemple le MIPS

◆ Fetch/ Decode/ Execute/ Memory/ Register



ID : Decode + fetch operands
Mem : write back cache + mem
Reg : store ALU into register

5 actions
en parallèle

15

Le pipe-line du 80486

- Fetch: Instructions depuis le cache ou la mémoire vers 2 buffers de prefetch. Chaque buffer contient en moyenne 5 instructions. Remplissage dès que possible des buffers.
- Decode stage 1: Décode le code opération et les modes d'adressage
- Decode stage 2: Génère les signaux pour l'ALU. Réalise les adressages plus complexes
- Execute: Opérations ALU, Accès cache, registres
- Write Back: Maj des flags, écriture des résultats sur le cache et le buffer de l'interface du Bus

16

Efficacité

- Soit t_i le temps de traversée de l'étage i.
- Le temps d'exécution d'une instruction est: $t_{inst} = \sum t_i$
- Le délai entre 2 instructions successives est $t_D = Max_i\, t_i$
  t_D est appelé cycle du pipeline

17

Rupture de pipeline

- Instructions de branchement
- Data dépendance
- Défauts de cache
- Conflits hardware (mémoire)

18

Instructions de Branchement

- Lors de l'exécution d'un JUMP
- La prochaine instruction est la suivante
  - OK
- Ce n'est pas la suivante
  - Vidage du pipe-line
  - Remplissage du pipe avec les bonnes instructions

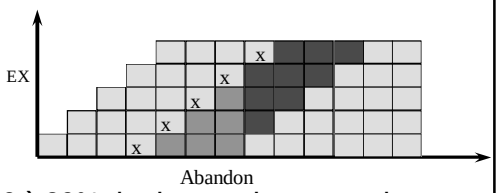
19

Buffer d'instructions

- FIFO d'instructions : assure un flux constant
- On utilise 2 FIFOs
  - Une pour les instructions suivant le JUMP
  - Une pour les instructions à l'adresse du JUMP

22

Réduction de la vitesse



- 10 à 20% des instructions sont des JUMP
  - réduction de la vitesse globale
    - Boucle TantQue, Exit

20

Multiple pre-fetch

- On charge les 2 FIFOs après un JUMP
- Après le JUMP, on choisit l'instruction dans l'une des 2 FIFOs
- Si plusieurs JUMP pris en compte, il faut plusieurs FIFO (2^n = n JUMP)
- Augmente les conflits d'accès mémoire

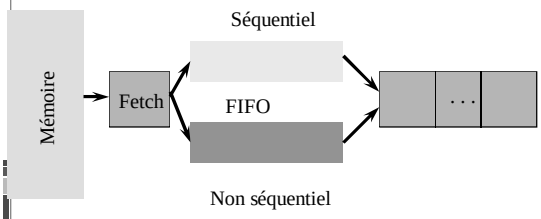
23

Réduction des ruptures

- 4 techniques pour réduire les ruptures de pipeline
  - Buffer d'instructions
  - Loop buffer
  - Table de branchement
  - Branchement retardé

21

Double FIFO



24

Loop Buffer

- Petite mémoire rapide contrôlée par l'unité de chargement des instructions
- Contient les N instructions les plus récemment chargées.
- Fonctionnement comme un cache d'instructions mais en séquence
- A chaque fetch on recherche d'abord l'instruction dans le loop buffer

25

Fonctionnement

- Avec le prefetch, quelques instructions suivantes sont présentes.
- Si le JUMP saute quelques instructions en avant, on peut la trouver dans le buffer ex: if then else
- Si le JUMP saute vers le début d'une boucle, la boucle complète peut se trouver dans le buffer
- Cray 1, motorola 68010

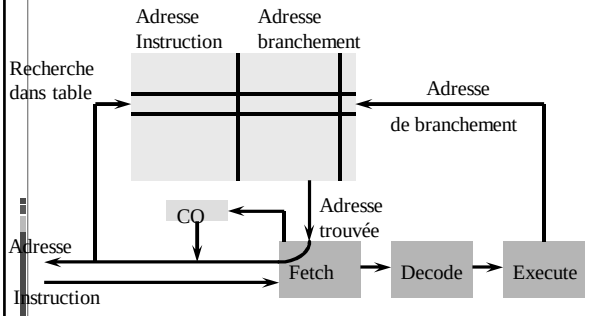
26

Table de Branchement

- Prédire dynamiquement l'adresse suivante : utilisation répétée de la même adresse
- La 1ère exécution range l'adresse de branchement
- Lorsque la même instruction JUMP est exécutée (Fetch), on utilise l'adresse mémorisée dans une table (mémoire rapide)

27

Decode history table



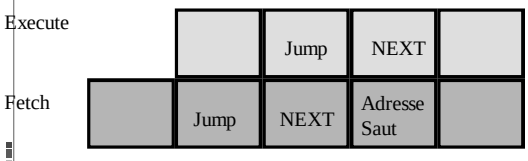
28

Branchement retardé

- Les instructions de JUMP sont placées dans le code avant qu'elles ne prennent effet
- Pour un pipeline à 2 étages , le JUMP prend effet après la prochaine instruction
- n étages : le JUMP prend effet après les n-1 suivantes

29

Pour un 2 étages



30

Efficacité

- Pas de vidage de pipe
- Optimisation du compilateur
- 70% des JUMP peuvent être suivis d'une instruction RISC

31

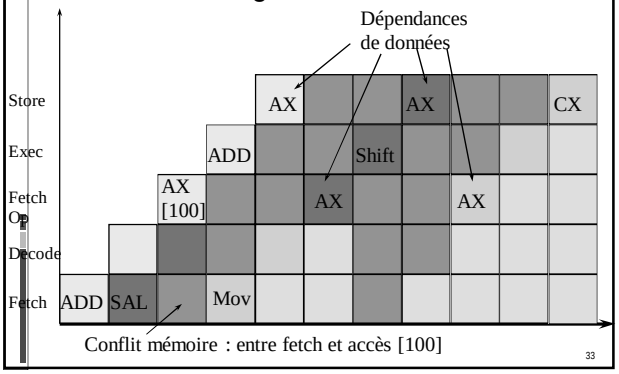
Data dépendances

- Exemple:
  C = 2 * (A + [@100])

```
ADD AX, [100]% A+100%
SAL AX, 1          % * 2 %
MOV CX, AX         % C %
```

32

Sur 5 étages



33

Fonctionnement

- Détection de la dépendance et blocage de pipeline jusqu'à résolution.
- Autoriser le chargement des instructions qui sont indépendantes (pas de data dépendance)
- On ne retarde que les instructions dépendantes.

34

SUPERSCALAIRE / SUPERPIPELINE

35

Superscalaire (1987)

- CISC et RISC produisent une instruction par cycle.
- Sur un superscalaire on produit plusieurs instructions par cycle.
- On utilise plusieurs pipelines d'instructions.

36

Taux de parallélisme

- Les superscalaires permettent d'extraire encore plus de parallélisme.
- Toutes les instructions indépendantes peuvent être exécutées en même temps (limité par le nombre de pipeline)

Fonctionnement

1	IF	ID	FO	EX	SO				
2	IF	ID	FO	EX	SO				
1		IF	ID	FO	EX	SO			
2		IF	ID	FO	EX	SO			
1			IF	ID	FO	EX	SO		
2			IF	ID	FO	EX	SO		
1				IF	ID	FO	EX	SO	
2				IF	ID	FO	EX	SO	
1					IF	ID	FO	EX	SO
2					IF	ID	FO	EX	SO

10 actions
en parallèle

Efficacité

- Par observation, en moyenne on peut obtenir deux instructions à exécuter en parallèle.
- On limite le degré du processeur à 3 ou 5. (projet de degré à 16 sur Alpha - utilisation multi-user)

Superpipeline (1988)

- Utilisation d'une horloge multiphase
- Redécomposition pipeline de chaque étage du pipeline d'instruction
- Demande une fréquence d'horloge élevée
- Exemple R4000

Fonctionnement

1	IF1	IF2	ID1	ID2	FO1	FO2	E X1	EX2	SO1	SO2										
2		IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2									
1			IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2								
2				IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2							
1					IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2						
2						IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2					
1							IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2				
2								IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2			
1									IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2		
2										IF1	IF2	ID1	ID2	FO1	FO2	EX1	EX2	SO1	SO2	

10 actions
en parallèle

Limitations

- Augmenter le parallélisme au niveau de l'instruction
  - Compilateur ou Hardware
- 5 limitations
  - Dépendance de flot
  - Dépendance procédurale
  - Conflits sur ressources
  - Dépendance de sortie
  - Antidépendance

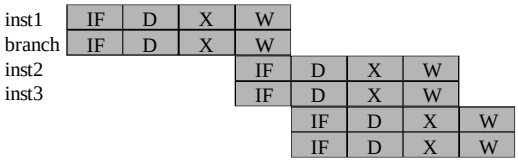
Dépendance de flot

- Exemple:
  - add R1, R2
    move R3, R1
  - La deuxième instruction ne peut être exécutée tant que la première n'est pas exécutée

43

Dépendance procédurale

- Présence de Branch impose le vidage de tous les pipes



44

Conflit sur ressource

- Utilisation de la même unité fonctionnelle, mémoire, cache, bus…
- On peut supprimer ce conflit en dupliquant les ressources ou en pipelinant les unités fonctionnelles

45

Instruction-issue

- Ordre dans lequel le processeur exécute les différentes étapes des instructions
  - Ordre de fetch
  - Ordre d'exécution
  - Ordre de modification des registres ou mémoires
- Modifier l'ordre en assurant une exécution correcte

46

Les stratégies

- Accès et exécution ordonnés
- Accès ordonné et exécution désordonnée
- Accès et exécution désordonnés

47

Accès et exécution ordonnés

- Même ordre que sur une machine séquentielle.
- Exemple:
  - superscalaire avec Fetch et Decode deux instructions en parallèle
  - 3 unités fonctionnelles
  - 2 unités d'écriture des résultats

48

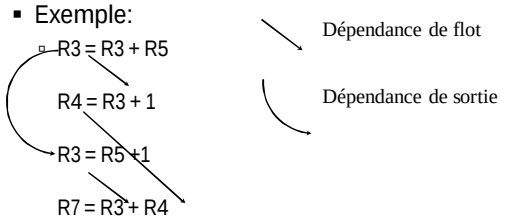
Exemple

- I1 demande 2 cycles pour exécution
- I3 et I4 sont en conflit sur une unité fonctionnelle
- I5 dépend du résultat produit par I4
- I5 et I6 sont en conflit sur une unité fonctionnelle

49

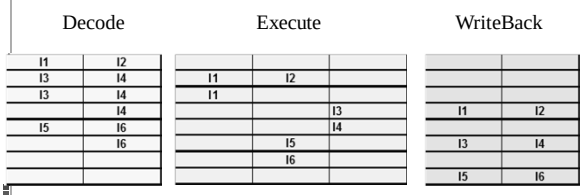
Dépendance de sortie

- Exemple:
  - R3 = R3 + R5
  - R4 = R3 + 1
  - R3 = R5 +1
  - R7 = R3 + R4



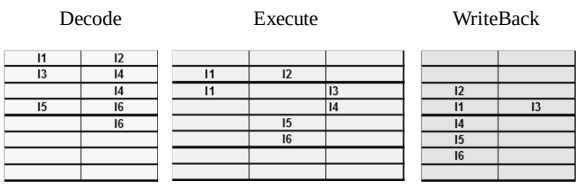
52

Accès et exécution ordonnés



50

Exécution désordonnée



53

Exécution désordonnée

- Utile pour les instructions demandant plusieurs cycles
  - Exemple: floating point sur Motorola 88000
- Autant d'instructions en exécution qu'il y a d'unités
- Respect des dépendances de flot et procédurale
- Dépendance de sortie

51

Accès et exécution désordonnés

- Découplage des étages decode et execute
- Un buffer d'instructions: Instruction Window
- Decode place les instructions dans la fenêtre
- Quand une UF devient libre elle obtient une instruction exécutable dans la fenêtre: ressource et dépendance

54

Fonctionnement

- Le choix parmi les instructions à exécuter est plus large
- Respect des dépendances
- Respect de l'antidépendance

55

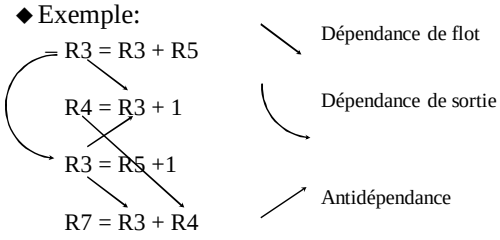
Renommage des registres

- Eliminer les dépendances de sorties et antidépendances
- Plusieurs instructions utilisent le même registre
- Allocation dynamique des registres par le processeur (ce n'est plus le compilateur)
- Un même registre du code peut référencer plusieurs registres lors de l'exécution

58

Antidépendance

◆ Exemple:



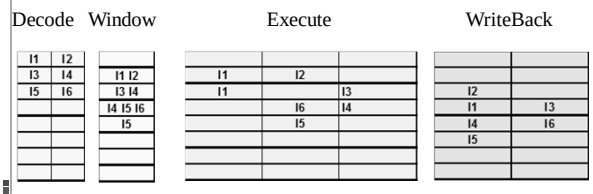
56

Pour notre exemple

- Exemple:
  - R3b = R3a + R5a
  - R4b = R3b + 1
  - R3c = R5a +1
  - R7b = R3c + R4b

59

Accès et exécution désordonnés



57

Compilation

- Lorsque l'on ne peut exécuter autant d'instructions que le degré du processeur, certains pipelines restent en attente.
- Le nombre d'instructions en cours d'exécution dépend des dépendances de données entre les instructions, et des conflits d'accès aux ressources
- Le travail du compilateur devient capital: réorganisation du code...

60