Méthodes de transfert d'informations

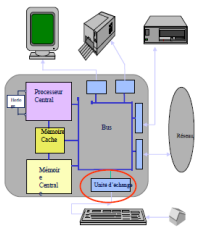
Les périphériques

Dispositifs matériels permettant d'assurer les échanges d'informations en entrée et en sortie entre l'ordinateur et l'extérieur ou de stocker de manière permanente des informations

- ✓ Clavier
- ✓ Souris
- ✓ Imprimantes
- ✓ Écrans, …

L'unité d'échange

- ➢ Rôle de l'unité d 'échange
  - ✓ (adaptation) connexion des unités périphériques au bus
  - ✓ gestion des échanges entre le processeur et les périphériques
- ➢ Particularités des unités d 'échanges
  - ✓ Constitution et adressage
  - ✓ Gestion des transferts



Interfacage Entrées-sorties

Le processeur se connecte au bus et va directement envoyer sur le bus : l'adresse, les données, et autres commandes à envoyer à l'entrée-sortie ou au périphérique.

⇨ Intercaler des registres entre le processeur et les entrées-sorties. Ces registres servent à faciliter la communication avec le processeur.

Il suffit au processeur de lire ou écrire dans ces registres pour communiquer avec le périphérique .

⇩

Registre d'interface

↙ Contenir ↘

données ordres

Contrôleur de périphérique

- ➢ Le contrôleur de périphérique transmet au périphérique les commandes plus détaillées nécessaires à la réalisation de l'opération requise.
- ➢ En déchargeant cette responsabilité sur le contrôleur, l'UC est libre d'accomplir simultanément d'autres tâches.
- ➢ Chaque dispositif d'E/S possède un contrôleur spécifique.

N.B. La plupart des contrôleurs peuvent servir plusieurs périphériques à la fois.

Pilotes de périphériques

Lorsqu'un ordinateur utilise un système d'exploitation, celui-ci ne connait pas toujours le fonctionnement d'un périphérique et/ou de son contrôleur

⇨ Il faut donc installer un petit programme qui va s'exécuter quand on souhaite communiquer avec le périphérique

S'occupera de tout ce qui est nécessaire pour le transfert des données, l'adressage du périphérique

Ce petit programme est appelé **driver ou pilote de périphérique.**

Gestion des entrées-sorties

Trois méthodes de gestion des entrées/sorties

- ✓ La scrutation ou spolling
- ✓ Les entrées-sorties pilotées par les interruptions
- ✓ L'utilisation d'un dispositif permettant des accès directs à la mémoire, DMA

La scrutation ou spolling

Périodiquement, le processeur va interroger chaque dispositif pour savoir si il a besoin d'un service. Si un périphérique requiert un service, la routine va être enclenchée. Le processus qui consiste à interroger un dispositif et à recevoir une réponse en retour s'appelle le « handshaking »

Cette technique est avantageuse car elle est facile à mettre en place. Mais le système perd du temps à scruter tout les périphériques.

7

Les interruptions

8

Interruptions

➢ Les interruptions sont des fonctionnalités du processeur qui vont interrompre temporairement l'exécution d'un programme afin de réagir à un événement extérieur (matériel, erreur fatale d'exécution d'un programme...) et de le traiter en temps voulu, avant de rendre la main au programme interrompu.

➢ Ces interruptions sont les sonnettes que tirent les périphériques pour dire au processeur que quelque chose se passe. Mais si chaque périphérique pouvait envoyer directement un signal au processeur, il faudrait sur celui-ci autant de broches (jouant le rôle de cordons de sonnette) que de périphériques.

⇨ Les périphériques envoient leur requête d'interruption à une puce à laquelle ils sont connectés, le contrôleur d'interruption. C'est ce contrôleur qui va envoyer à leur place une interruption au processeur.

9

Interruptions logicielles

➢ Les interruptions logicielles, sont aussi appelées trappes ou déroutements. Elles incluent aussi les fautes et les arrêts.

- ✓ Une faute se produit quand le processeur détecte une erreur durant le traitement d'une instruction. Par exemple, division par 0, opcode invalide, etc.
- ✓ Quand une erreur est grave au point qu'une partie du contexte d'exécution est perdue, le résultat est un arrêt.

➢ Peut être un arrêt de l'exécution d'un programme pour exécuter une routine d'interruption du **DOS ou BIOS.**

10

Interruptions matérielles

➢ Les interruptions matérielles sont générées par les périphériques : souris, clavier, disque, horloge temps réel, etc…

➢ Ces composants électroniques agissent directement sur une ligne du bus de contrôle : la ligne IRQ « Interrupt Request ».

➢ Cette ligne provient en fait d'un composant programmable : le **contrôleur d'interruptions** qui rassemble les signaux d'interruptions de divers éléments périphériques.

➢ Le contrôleur d'interruption après avoir répercuté sur la ligne IRQ le signal provenant de l'une de ses entrées doit donner ensuite au processeur un numéro qui identifie l'origine du signal.

➢ C'est ce numéro qui servira à retrouver l'adresse de la routine adéquate dans le vecteur d'interruption.

11

Contrôleur programmable d'interruptions (1)

Circuit intégré se trouvant entre le CPU et les périphériques, permettant de séparer davantage le CPU et l'électronique.

➢ Le Contrôleur Programmable d'Interruptions (PIC) a deux fonctions essentielles

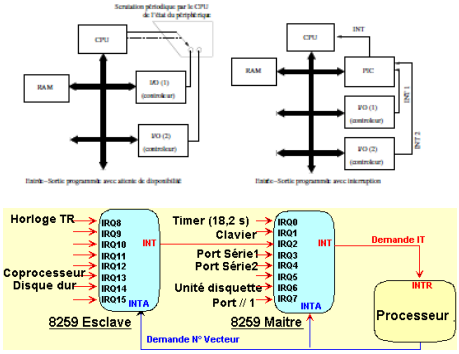
- ✓ établir des priorités entre les différentes sources
- ✓ décharger le CPU de la détermination de la source d'une interruption, quand celle-ci se présente

➢ Utilise une ou des "lignes d'interruptions" (IRQ :Interrupt ReQuest), chacune étant associée avec un niveau de priorité

➢ Connaît l'adresse des instructions à charger (ISR : Interrupt Service Routine) quand l'interruption est détectée (vecteur d'interruption)

12

Contrôleur programmable d'interruptions (2)



13

Bornes pour les interruptions

- Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.
- NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
- INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
- INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

14

Prise en charge d'une interruption par le microprocesseur (1)

- Une source d'interruption est connectée à chacune des 8 entrées IR0 à IR7. Selon sa priorité, et s'il n'y a pas d'autre interruption en cours, le PIC décide s'il peut transmettre l'interruption au CPU.
- Il émet un accusé de réception de demande d'interruption (Interrupt Acknowledge) indiquant au circuit d'E/S que la demande d'interruption est acceptée

!!!Remarque : le microprocesseur peut refuser la demande d'interruption

- Le PIC répond à son tour en envoyant le numéro d'interruption sur les lignes D0 à D7. Ce numéro est un index dans la table des vecteurs d'interruption.
- Le CPU est maintenant prêt à appeler le sous-programme de traitement d'interruption approprié.

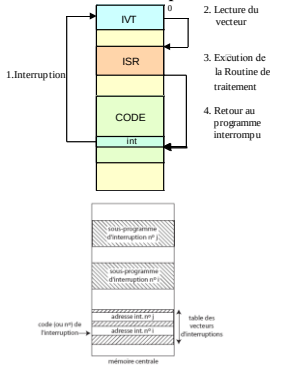
15

Prise en charge d'une interruption par le microprocesseur (2)

- Le microprocesseur termine l'exécution de l'instruction en cours ;
  - ✓ *il range le contenu des principaux registres sur la pile de sauvegarde: pointeur* d'instruction, flags, ...
- Il abandonne l'exécution du programme en cours et va exécuter un sous-programme de service de l'interruption (ISR : Interrupt Service Routine) ;
  - ✓ *après l'exécution de l'ISR, les registres sont restaurés à partir de la pile et le microprocesseur* reprend l'exécution du programme qu'il avait abandonné

16

Gestion d'interruptions



Ligne IRQ	Origine	
IRQ0	Timer (18.2 s)	Plus prioritaire
IRQ1	Clavier	
IRQ2	Contrôleur IT (8059)	
IRQ8	Horloge temps réel	
IRQ9		
IRQ10		
IRQ11		
IRQ12		
IRQ13	Coprocesseur	
IRQ14	Disque dur	
IRQ15		
IRQ3	Port série 1	
IRQ4	Port série 2	
IRQ5	Port // 2	
IRQ6	Unité disquette	
IRQ7	Port // 1	Moins prioritaire

Hiérarchie des priorités des demandes d'IT matérielles dans un PC standard

17

Le vecteur d'interruption

- Le vecteur d'interruption est une table qui contient les adresses des routines d'interruption.
- Les interruptions sont numérotées et ces numéros (allant de 0 à 255 dans un PC) servent d'index pour rechercher dans le vecteur d'interruption l'adresse de la routine à exécuter.

N° V	Vecteur	Cause	Origine	N° V	Vecteur	Cause	Origine
0	0	CPU	Division par 0	---	-----	-----	----------
---	-----	-------	-----	70	1C0	IRQ8	Horloge temps réel
8	020	IRQ0	Timer (18.2 s)	71	1C4	IRQ9	
9	024	IRQ1	Clavier	72	1C8	IRQ10	
10	028	IRQ2	Contrôleur IT (8059)	73	1CC	IRQ11	
11	02C	IRQ3	Port série 1	74	1D0	IRQ12	
12	030	IRQ4	Port série 2	75	1D4	IRQ13	Coprocesseur
13	034	IRQ5	Port // 2	76	1D8	IRQ14	Disque dur
14	038	IRQ6	Unité disquette	77	1DC	IRQ15	
15	03C	IRQ7	Port // 1	---	--------	-----	----------

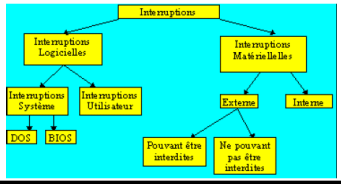
Standard de la table des vecteurs d'interruptions matérielles dans un PC

18

Conclusion

Ce qui est important à retenir :

- C'est quoi une interruption
- Types d'interruptions
  - ✓ Matérielle
  - ✓ Logicielle
- Gestion des interruptions
- Table des vecteurs d'interruptions



19

Accès direct à la mémoire

20

Notions de DMA

➢Il s'agit d'un transfert direct et bidirectionnel de données via un contrôleur adapté entre un périphérique et la mémoire principale de la machine.

➢Ce procédé ne fait pas intervenir le processeur que pour initier et conclure le transfert à travers des interruptions.

➢Ainsi le transfert DMA offre un moyen plus rapide pour l'échange de blocs de données, entre la mémoire et le périphérique en question ou entre deux blocs de mémoire, que la méthode classique.

✓En effet, cette dernière fait intervenir le processeur dans le transfert de chaque octet ; c'est-à-dire par la lecture, l'écriture, la vérification de la fin de l'échange et l'incrémentation d'adresses en plus du traitement des données transférées.

21

Le contrôleur DMA (1)

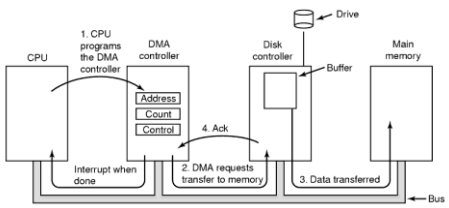
➢ En ce qui concerne la technique DMA, un autre périphérique spécialisé se charge du transfert, à savoir le contrôleur DMA ou le DMAC, et le processeur n'a qu'à s'occuper du traitement de données.

➢ Le processeur donne au DMAC les informations concernant l'échange, c'est à dire l'adresse de début du bloc en mémoire, la taille du bloc et le sens de l'échange et autorise l'échange.

➢ Il y a également une introduction de la notion de priorité vu que la mémoire n'autorise qu'un seul accès par cycle. En fait, n'ayant pas le pouvoir d'attendre aussi longtemps que le processeur, le DMAC est plus prioritaire en accédant à la mémoire que le processeur. Cette technique s'appelle le vol de cycle.

22

Le contrôleur DMA (2)



23

Les Modes DMA (1)

Il existe trois façon de transférer des données entre le périphérique et la mémoire.

➢Mode block, le contrôleur mémoire se réserve le bus mémoire, et effectue le transfert en une seule fois, sans interruptions.

✓ Avantage: Le plus rapide. Il est très utilisé pour charger un programme du disque dur dans la mémoire, par exemple.

✓ Désavantage : le processeur ne peut pas accéder à la mémoire durant toute la durée du transfert entre le périphérique et la mémoire.

➢Mode cycle stealing , on est un peu moins strict : cette fois-ci, le contrôleur ne bloque pas le processeur durant toute la durée du transfert.

✓ En cycle stealing , le contrôleur va simplement transférer un byte (un octet) à la fois, avant de rendre la main au processeur. Puis, le contrôleur récupérera l'accès au bus après un certain temps.

24

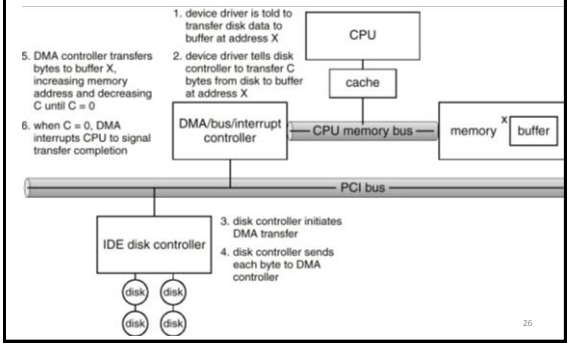
Les Modes DMA (2)

➢Mode transparent, dans lequel le contrôleur DMA accède au bus mémoire uniquement quand le processeur ne l'utilise pas.

25

DMA: six étapes

Exemple: entrée disq→mém

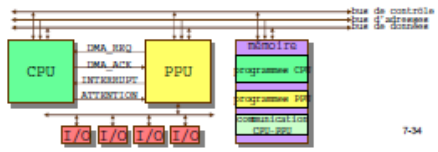


26

Processeurs d'entrées/sorties

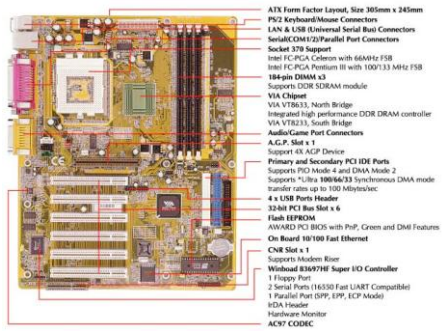
Les processeurs d'entrée/sortie (**PPU ou *peripheral processing units)*** sont l'extension du concept de DMA.

Un processeur prend la place du contrôleur DMA et gère les transferts de données (généralement un seul processeur s'occupe de plusieurs périphériques).



27

Conclusion



28

Optimisation et augmentation des performances des processeurs

29

Augmentation des performances

- ➢ Recherche permanente de l'augmentation des performances des CPU
  - ✓ Évolution des architectures
- ➢ Points principaux de cette évolution
  - ✓ Fréquence de fonctionnement
  - ✓ Mémoire cache
  - ✓ Parallélisassions et optimisation des séquences d'instructions
    - ❖ Pipeline
    - ❖Architectures parallèles
    - ❖Architectures superscalaires
  - ✓ Jeu d'instructions
- ➢ Chaque point influence en bien ou en mal un autre
  - ✓ Recherche du meilleur compromis

Technologiques et physique
- **Temps de propagation des signaux électriques**
- **Dégagement de chaleur**
- **Mémoire : latence moyenne très rapide entre la mémoire centrale et le processeur**

30

Traitement en pipeline des instructions

Principe du pipeline par l'exemple: le lavage du linge

- Soient 4 étudiants qui désirent faire leur lavage
- Il y a une laveuse, une sécheuse, et une table pour plier le linge
- Le lavage prend 30 minutes
- Le séchage prend 40 minutes
- Le pliage prend 20 minutes
- Si l'opération commence à 6h00 du soir, à quelle heure termineront-ils?

31

Lavage du linge: Solution Séquentielle

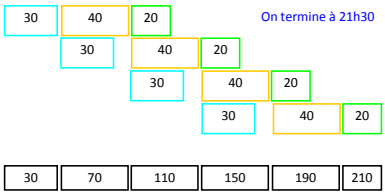
- L: 30
- S: 40
- P: 20
- 4 * (30 + 40 + 20) = 360 minutes
- On termine à minuit

- ✓ Plusieurs personnes se servent en même temps
- ✓ Gain de temps : plus de personnes passent pendant une même durée
- ✓ Meilleure gestion des éléments : toujours utilisés

32

Lavage du linge: 2e Solutions: pipeline

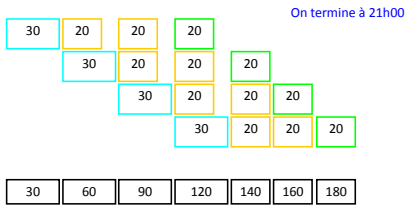
- ✓ L:30, S: 40, P: 20
- ✓ On superpose le lavage, séchage et pliage:



33

Lavage du linge: 3e Solution: pipeline

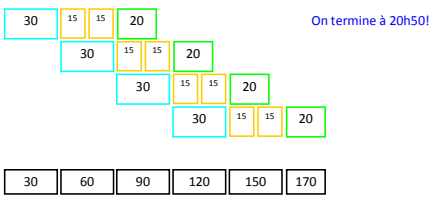
- ✓ L:30, S1: 20, S2: 20, P: 20



34

Lavage du linge: 6e itération: 2 sécheuses « haute température »

- ✓ L:30, S1: 15, S2: 15, P: 20



35

Leçons du pipeline de lavage

- ✓ Le pipeline n'améliore pas la latence. Il améliore le temps moyen par tâche (throughput)
- ✓ La vitesse du pipeline est limitée par l'étage le plus lent
- ✓ Plusieurs tâches se font en parallèle
- ✓ Accélération possible: nombre d'étages
- ✓ Les durées d'étage inégales limitent l'accélération
- ✓ Le temps de remplir et de vider le pipeline réduit l'accélération

36

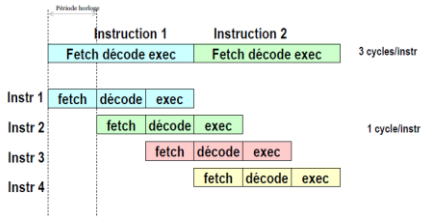
Principes

Dans un processeur, utilisation d'un pipeline pour exécution d'une opération

- Une opération comporte plusieurs sous-opérations
  - ✓ Pipeline pour exécution de ces sous-opérations
  - ✓ Une sous-opération utilise une sous-unité du processeur qui n'est
  - ✓ pas utilisée par d'autres sous-opérations (si possible...)
- Suivant les types des processeurs:
  - ✓ Pipeline à 3 niveaux: fetch, Decode, Execute
  - ✓ Pipeline à 5 niveaux (Processeur ARM9, MIPS): IF, ID, EX, MEM, WB
  - ✓ Jusqu'à 25 niveaux : 20 niveaux pour le Pentium 4.

37

Pipeline à 3 niveaux:



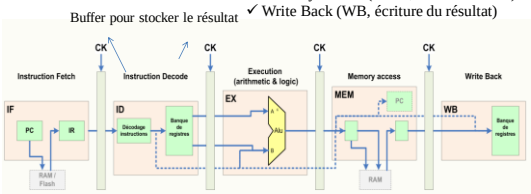
38

Pipeline à 5 niveaux: Architecture

Exemple de pipeline simple (fictif)

- ✓ IF: Recherche d'instruction
- ✓ ID: Instruction Decode
- ✓ EX: Instruction Execution
- ✓ Memory Access (MEM, accès mémoire)
- ✓ Write Back (WB, écriture du résultat)

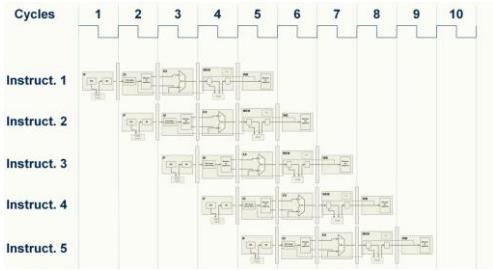
Buffer pour stocker le résultat



Insertion de registres d'étages (les niveaux du pipeline)
Les données et tous les signaux de contrôle sont mémorisés et propagés à chaque niveau
L'écriture dans les buffers est faite à chaque fin de cycle par les signaux : FI/IDWrite, ID/ExWrite, EX/MemWrite, Mem/RbWrite

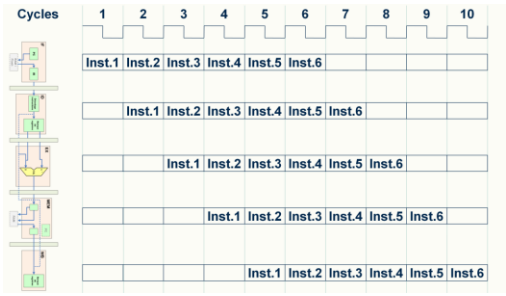
39

Pipeline à 5 niveaux:



40

Pipeline à 5 niveaux:



41

Exécution

- Avantage du pipeline:
  - ✓ Meilleure utilisation des différentes unités
  - ✓ Utilise du parallélisme entre instructions (5 instructions simultanément dans le pipeline MIPS)
  - ✓ Fait gagner en performances, en augmentant la fréquence
- Inconvénient du pipeline
  - ✓ Nécessite de raccourcir le temps d'un cycle → augmente la fréquence donc la dissipation d'énergie
  - ✓ Les instructions qui sont simultanément dans le pipeline ne doivent pas dépendre les unes des autres → difficile à imposer a priori !
    - ❖ Solution 1: le matériel gère ce problème.
    - ❖ Solution 2: le compilateur gère.

42

Aléas dans les Pipeline

➢ Le bon fonctionnement du pipeline peut être perturbé par plusieurs événements appelés aléas

✓ Aléas de structure:

• Matériel ne supporte pas certaines combinaisons d'instructions (manque de ressources)

✓Aléas de données:

• Instruction dépend de résultats d'instructions précédentes pas encore complétées

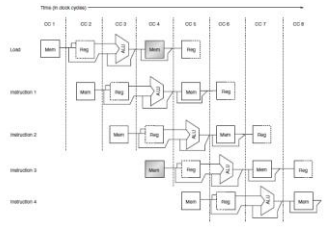
✓Aléas de contrôle:

• Délais entre la lecture des instructions et les décisions de changement de cours du programme (branchement conditionnel)

Aléas de structure (1):

La réalisation du chemin de données interdit certaines combinaisons d'opérations

➢ Exemple : IF et MEM accèdent tous les deux à la mémoire en cas de l'exécution d'une instruction de lecture écriture.



Aléas de structure (2):

➢Solutions :

✓attendre que l'unité soit disponible en retardant l'exécution : peu efficace, le pipeline doit être suspendu durant un cycle pour lever l'aléa (on parle de bulle de pipeline)

✓dupliquer les différentes unités du chemin de données : ici, accès mémoire : découpage en 2 parties du cache L1

• IF accède à la partie "instruction"

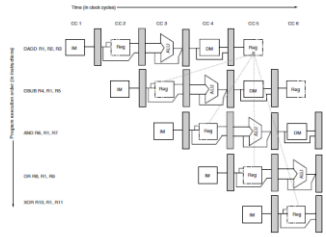
• MEM accède à la partie "données"

Aléas de données (1):

Il apparait lorsqu'il y a des dépendances de donnée entre les instructions.

➢Exemple: Instruction essaie de lire un registre avant qu'il ne soit écrit

ADD R1,R2,R3
SUB R4,R1,R3



Aléas de données (2):

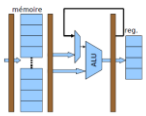
➢ Solutions :

✓Arrêter le calcul de R4 tant que R1 n'est pas connu
✓Changer l'ordre d'exécution des instructions : **réordonnancement** (réalisé soit à la compilation, soit par le processeur à la volée)

EX:

addi $1, $2, 12	addi $1, $2, 12
multi $3, $1, 2	li $5, 4
li $5, 4	add $4, $5, $6
add $4, $5, $6	multi $3, $1, 2

✓On court-circuite le pipeline (et l'exécution normale des instructions) en plaçant le contenu du registre de sortie de UAL directement dans un des ses registres d'entrée.



Aléas de contrôle (1) :

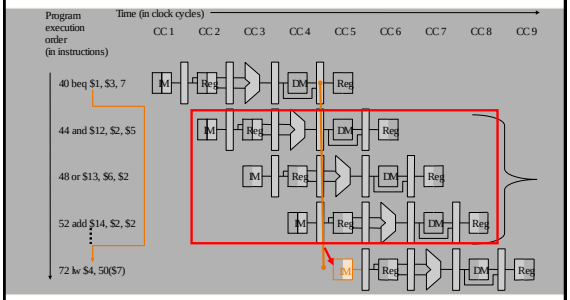
Résulte de l'exécution en pipeline des branchement et des instructions dont l'issue est inconnue.

Exemple: L'exécution d'un saut conditionnel ne permet pas de savoir quelle instruction il faut charger dans le pipeline puisque deux choix sont possibles.

N.B. Cet aléa est le principal facteur de dégradation de performance dans une architecture pipeline.

Aléas de contrôle (2)

Le résultat de la condition du branchement n'est connu qu'à la fin de la phase 4.
Pendant la phase 2, et 3 et 4, les trois instructions suivantes ont été chargées
Problème si la condition est vérifiée , On écrase des registres !!!!!!



Aléas de contrôle : Solutions

- Dupliquer l'architecture de pipeline pour traiter les deux cas du branchement (pris ou pas) ;
- Précharger l'instruction (ou la suite d'instruction) correspondant à l'adresse de branchement (quitte à ne pas l'utiliser) ;
- Se baser sur une prédiction des branchements
  - ✓supposer qu'un branchement ne sera jamais/toujours pris ;
  - ✓supposer que certains opcodes favorisent le branchement ;
  - ✓se baser sur un historique des branchements ;
- Générer des instruction NOP (No OPeration) après l'instructions de branchement le temps que le branchement puisse se conclure.

Pipeline : Résumé

- Le pipeline améliore le débit mais pas le temps par instruction : il faut toujours cinq cycles à une instruction d'un pipeline à cinq étages pour s'exécuter.
- Les dépendances de données et de contrôle dans les programmes imposent une limite supérieure au gain que peut générer le pipeline car le processeur doit parfois attendre la fin d'une instruction pour que les dépendances soit résolues.
- On peut élever cette limite, mais pas l'éliminer, en réduisant les aléas de contrôle par des optimisations, et les aléas de données par un ordonnancement des instructions par le compilateur.

Limites du pipelining, et comment les dépasser (2)

- Une question fondamentale se pose:
  1. Est-il préférable de tenter de simplifier le circuit pour augmenter la fréquence de l'horloge?
  2. Ou bien, doit-on rendre le circuit plus complexe pour tenter de faire plus durant la même période d'horloge?
- L'architecture RISC, avec le pipeline, procède de la première approche. Les résultats sont probants, mais il semble qu'on ne puisse réduire la période de l'horloge en dessous d'un certain seuil.
- D'où: il faut aussi considérer la deuxième approche (ce que l'on fait avec le VLIW et le superscalaire).

Deux façons de faire cela:
- Décider au moment de la compilation des instructions qui peuvent s'exécuter en parallèle (approche VLIW, EPIC)
- Décider au moment de l'exécution des séquences d'instructions qui peuvent se faire en parallèle (approche superscalaire)

Architecture Superscalaire

But

Augmenter le nombre d'instructions exécutées par cycle

Solution

Plusieurs pipelines (spécialisés)

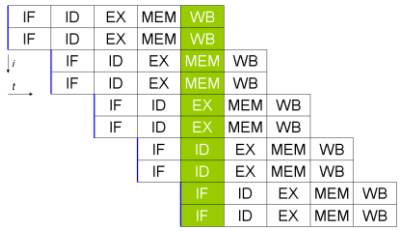
Processeur superscalaire

- Duplication du matériel du chemin de données.
- Exécution de deux instructions simultanées, si elles n'ont pas de conflit de données.
- Double le débit d'instructions avec le même cycle d'horloge.
- Parfois duplication seulement des éléments les plus lents (les plus rapide traitent plus vite leur tâche).
- Aussi duplication du chemin de données pour traiter calculs flottants et entiers indépendamment.
- Jusqu'à huit unités juxtaposées.

Superscalaire
– Le matériel est responsable à l'exécution du lancement parallèle des instructions
• Contrôle des dépendances de données et de contrôle par matériel

55

Architecture superscalaire



- Le matériel est responsable à l'exécution du lancement parallèle des instructions
  - ✓ Contrôle des dépendances de données et de contrôle par matériel

56

Very Large Instruction Word

- Le compilateur est responsable de présenter au matériel des instructions exécutables en parallèle
- Contrôle des dépendances de données et de contrôle par le compilateur
- Évolution du superscalaire.
- Une instruction donne du travail à toutes les unités d'exécution.
- Solution pratique aux problèmes de dépendances dans les superscalaires.

57

VLIW & Superscalaire

VLIW

- VLIW: technique pour exécuter plusieurs opérations en parallèle
- Méthode: on définit des groupes de plusieurs instructions, qui sont lues, décodées et exécutées en parallèle. Chaque groupe devient une « super-instruction » destinée à être lue d'un bloc par le processeur VLIW
- Avantage: le compilateur décide des instructions à grouper ensemble. Ceci élimine pour le processeur VLIW le fardeau du scheduling des instructions (détermination des instructions qui peuvent être exécutées en parallèle). Le processeur est donc plus simple à concevoir, et par conséquent peut être plus rapide.

Superscalaire

- Superscalaire: technique pour exécuter plusieurs opérations en parallèle
- Méthode: lit en même temps des blocs d'instructions séquentielles (compilées de façon traditionnelle), et le processeur décide dynamiquement quelles instructions du bloc peuvent s'exécuter en parallèle
- Avantage:
  - On utilise un compilateur traditionnel pour produire le code exécutable
  - Le parallélisme est extrait de façon dynamique par le processeur, donnant une plus grande marge de manœuvre pour le scheduling des instructions

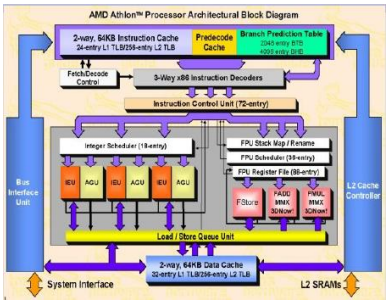
58

Exemple 1 : Athlon (AMD)

- Superscalaire out-of-order.
- Format d'instructions variable.
- 6 unités de décodage parallèles.
- 9 pipelines : 3 IEU, 3 FPU (add, mul, MMX), 3AGU.
- Pipeline entier : 10 étages
- Pipeline flottant : 15 étages.
- Data cache + Instruction cache L1 64Ko : caches associatifs par ensemble de 2 blocs
- Cache L2 512 Ko.

59

Exemple 1 : Athlon (AMD)



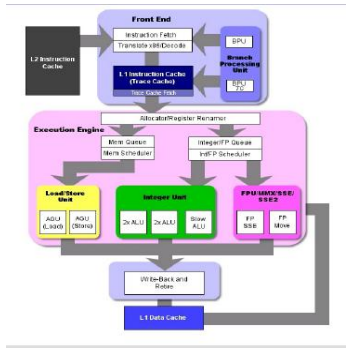
60

Exemple 2 : Pentium 4

- chemin entre cache L2 et L1 : 256 bits contre 64 pour le P3 (mais toujours 64 bits entre L2 et prefetch BTB).
- instruction cache APRES le décodage => trace cache (mémoires de micro instructions), 12 000 micro-codes, 100 bits/micro-codes => 150 Ko
- 128 registres internes pour le renommage des 8 registres accessibles en programmation.
- 4 IEU, 2 FPU, 2 AGU
- cache L2 : 256, 512, 1024 ou 2048 Ko selon les modèles, L1 : 8 Ko ou 16Ko
- données 2008 : cache L2 : 1Mo à 8Mo, cache L1 : -> 256 Ko

61

Exemple 2 : Pentium 4



62

Mémoires caches

63

Hiérarchie mémoire

Contraintes de conception d'une mémoire:
- ✓ son coût,
- ✓ sa vitesse,
- ✓ et son prix.

Il faut faire un compromis entre ces trois caractéristiques:
- ✓ coût,
- ✓ capacité
- ✓ et temps d'accès.

Tenir compte des relations suivantes :
- ✓ Plus le temps d'accès est court, plus le prix par bit est élevé.
- ✓ Plus la capacité est importante, plus le prix par bit est faible.
- ✓ Plus la capacité est importante, plus le temps d'accès est long.

64

Introduction:

Le principe de la ***mémoire cache*** *ou* simplement ***cache*** permet d'obtenir la plupart des avantages des deux technologies : rapidité et faible coût.

- La mémoire cache est appelée souvent L1 ou L2.
- C'est une mémoire statique, plus rapide et plus chère que la mémoire dynamique utilisée dans les barrettes RAM.
- Elle offre aux CPU une meilleure performance en échange de données.

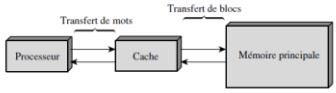
65

Principales caractéristiques des systèmes mémoire.

Emplacement	**Performances**
Processeur	Temps d'accès
Interne (principale)	Temps de cycle
Externe (auxiliaire)	Débit de transfert
Capacité	**Type physique**
Taille du mot	Semi-conducteur
Nombre de mots	Magnétique
Unité de transfert	Optique
Mot	Magnéto-optique
Bloc	**Caractéristiques physiques**
Méthode d'accès	Volatile/non volatile
Séquentielle	Effaçable/non effaçable
Directe	**Organisation**
Aléatoire	
Associative	

66

Principes de la mémoire cache



Un accès à la mémoire centrale prend entre 5 et 10 cycles d'horloge. Afin d'être rentable, le cache doit fournir une donnée en un cycle d'horloge.

La donnée recherchée —se trouve→ La mémoire rapide

Lorsqu'une donnée doit être recherchée dans la mémoire lente —écrite→ La mémoire rapide pour des références ultérieures

On obtient alors une rapidité proche de celle de la mémoire rapide pour un coût généralement associé à la mémoire lente.

L'accès aux données ne doit pas être aléatoire.

Principe de localité:

Les données les plus souvent utilisées sont co-localisées, spatialement ou temporellement.

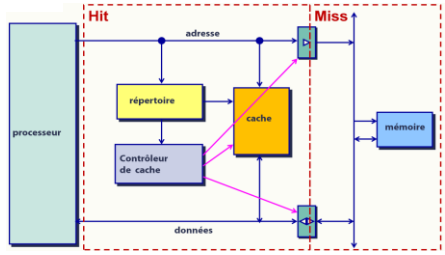
Les programmes possèdent deux caractéristiques intéressantes :

✓ localité spatiale: un programme qui s'exécute sur un processeur à tendance à utiliser des instructions et des données qui ont des adresses mémoires très proches;

✓ localité temporelle: Un programme a tendance à réutiliser les instructions et données qui ont étés accédées dans le passé;

Un programme accède à ses données (et à son code) d'une manière relativement prévisible. La probabilité qu'un accès à une adresse a soit suivi d'un accès à une adresse proche ou égale à a est très élevée.

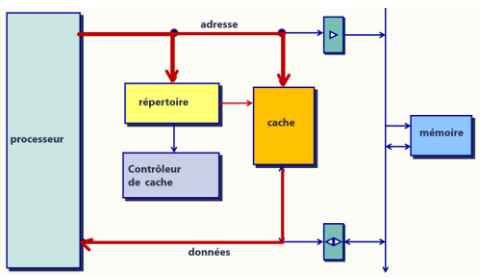
Organisation mémoire cache



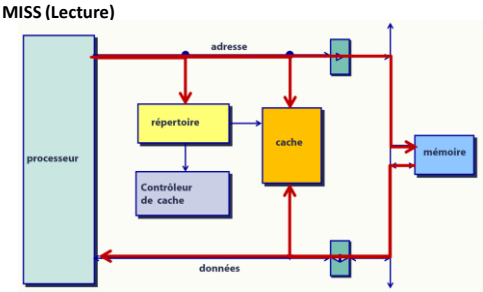
Hit (réussite) quand la donnée se trouve dans le cache. ⇨ Inutile d'accéder à la mémoire principale.

Miss (échec) quand la donnée ne se trouve pas dans le cache ⇨ Il est nécessaire d'accéder à la mémoire principale.

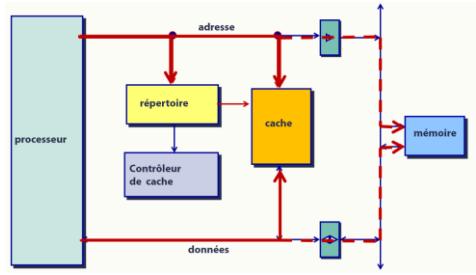
Hit (Lecture)



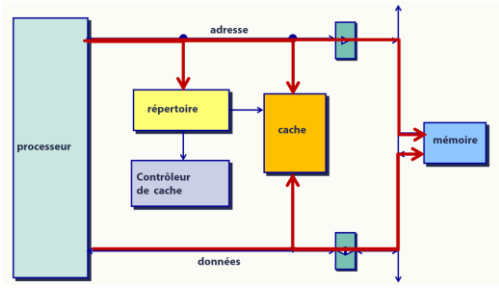
MISS (Lecture)



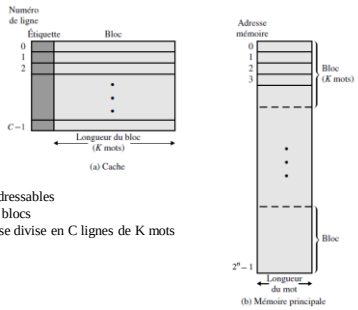
HIT Ecriture



MISS (Ecriture)



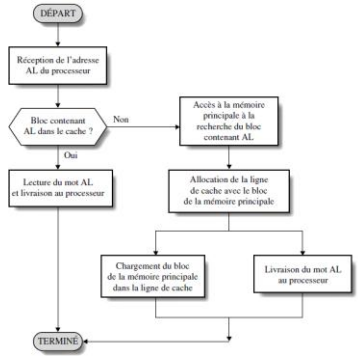
Structure cache/mémoire principale



- 2^n mots adressables
- $M = 2^n/K$ blocs
- Le cache se divise en C lignes de K mots
- C << M

Ce schéma est tiré du livre:

Opération de lecture du cache.



Ce schéma est tiré du livre:

Éléments de la conception du cache

Taille du cache	**Stratégies d'écriture**
Fonction de correspondance	Écriture immédiate (*Write through*)
Directe	Écriture différée (*Write back*)
Associative	Écriture unique (*Write once*)
Associative par ensemble	**Taille des lignes**
Algorithme de remplacement	**Nombre de caches**
LRU (*Least recently used*, moins récemment utilisé)	Un ou deux niveaux
FIFO (*First in first out*, premier entré premier sorti)	Unifié ou séparé
LFU (*Least frequently used*, moins fréquemment utilisé)	
Aléatoire	

Fonction de correspondance

Dans la mesure où les lignes de cache sont moins nombreuses que les blocs de mémoire principal

⇓

Algorithme pour la correspondance entre les blocs et les lignes de cache

⇒ Trouver un moyen pour déterminer quel bloc de la mémoire principale occupe la ligne de cache.

NB Le choix de la fonction de correspondance détermine l'organisation du cache

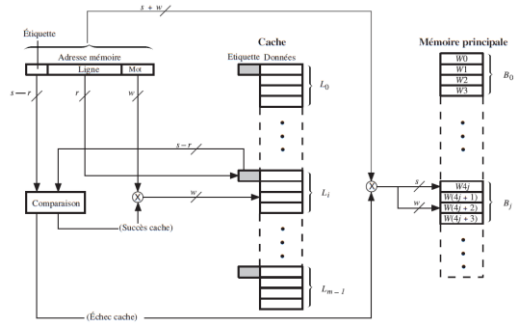
Trois techniques de correspondance:
- Directe
- Associative
- Associative par ensemble

Organisation d'un cache avec correspondance directe

Associe chaque bloc de la mémoire principale à une seule ligne de cache possible.

- longueur de l'adresse = $(s + w)$ bits
- nombre d'unités adressables = 2^{s+w} mots ou octets
- taille du bloc = taille de la ligne = 2^w mots ou octets
- nombre de blocs dans la mémoire principale = $\frac{2^{s+w}}{2^w} = 2^s$
- nombre de lignes dans le cache = $m = 2^r$
- taille de l'étiquette = $(s - r)$ bits

Organisation d'un cache avec correspondance directe



Ce schéma est tiré du livre:

Le résultat de cette correspondance:

Ligne du cache	Blocs de la mémoire principale attribués
0	$0, m, 2m, \ldots, 2^s - m$
1	$1, m+1, 2m+1, \ldots, 2^s - m + 1$
…	…
$m-1$	$m-1, 2m-1, 3m-1, \ldots, 2^s - 1$

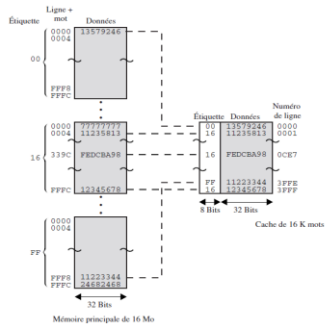
Ainsi, en utilisant une partie de l'adresse comme numéro de ligne, on obtient une correspondance unique de chaque bloc de mémoire principale dans le cache.
Lorsqu'on lit ce bloc dans la ligne qui lui est attribuée, on doit marquer les données pour le distinguer des autres blocs qui peuvent être dans cette ligne. Pour ce faire, on fait appel aux s – r bits les plus significatifs.

Exemple

Le cache peut contenir 64 Ko.
• Les données sont transférées entre la mémoire principale et le cache sous forme de blocs de 4 octets. Autrement dit, le cache est organisé en 16 K = 2^{14} lignes de 4 octets.
• La mémoire principale fait 16 Mo, chaque octet étant directement adressable par une adresse 24 bits (2^{24} = 16 M). Ainsi, pour la correspondance, nous pouvons considérer que la mémoire principale se compose de 4 M de blocs de 4 octets chacun.

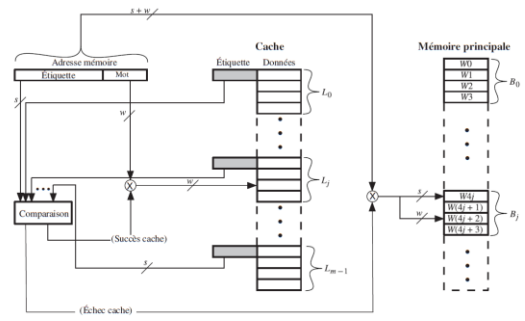
Ligne de cache	Adresse mémoire de départ des blocs
0	000000, 010000, …, FF0000
1	000004, 010004, …, FF0004
…	…
$2^{14} - 1$	00FFFC, 01FFFC, …, FFFFFC

Exemple de correspondance directe.



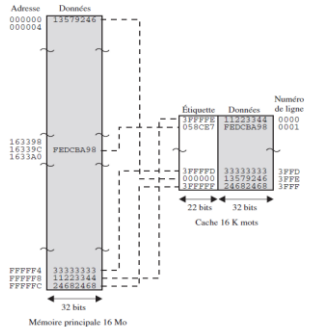
Ce schéma est tiré du livre:

Organisation d'un cache totalement associatif



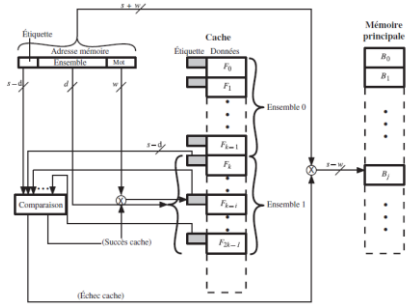
Ce schéma est tiré du livre:

Exemple de correspondance associative



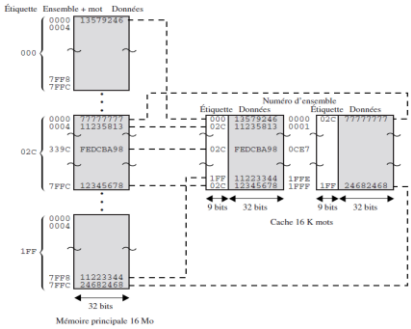
Ce schéma est tiré du livre:

Organisation du cache associatif k voies.



Ce schéma est tiré du livre:

Exemple de correspondance associative 2 voies



Ce schéma est tiré du livre:

Cache associatif par ensemble de blocs

- Le cache associatif par ensemble de blocs est un compromis entre le cache purement associatif et le cache à accès direct. On a 2 ou 4 blocs de données.
- D'un ensemble à l'autre, le cache est associatif. Cependant, les blocs (rangées) sont gérés comme dans le cache à correspondance direct.
- La performance de ce type de cache pratiquement équivalente à celle du cache purement associatif, à un prix beaucoup moindre.
- En pratique, on n'a aucun avantage à dépasser 4 ensembles.

Correspondance directe: pour ou contre?

- **Avantages:**
  - ✓ Utilise de la mémoire rapide mais « standard » (sans comparateur intégré)
  - ✓ Moins coûteux que la mémoire associative
- **Désavantages:**
  - ✓ Requiert un peu de circuiterie additionnelle
  - ✓ Si 2 adresses utilisées dans la même période de temps ont le même index, il y aura continuellement des collisions…

Algorithmes de remplacement

- Lorsque la mémoire cache est remplie et une nouvelle valeur doit y être écrite, on doit faire de la place…
- Les méthodes utilisées sont variables:
  - ✓ Remplacement aléatoire
  - ✓ Remplacement « Least Frequently Used », LFU
  - ✓ Un compteur comptabilise le nombre de fois qu'une valeur est utilisée
  - ✓ Remplacement « Least Recently Used », LRU
- Avec une mémoire cache assez grande, les méthodes LRU et aléatoires donnent des résultats comparables (sachant que la méthode aléatoire est beaucoup plus simple à mettre en place, que choisiriez-vous?)

Performance des mémoires caches

- La performance des mémoires cache dépend évidemment de la quantité de « misses ».

On peut séparer ceux-ci en trois catégories:

- ✓ Obligatoires – lorsqu'on accède à une adresse pour la première fois
- ✓ De capacité – lorsque la mémoire cache était pleine et qu'on a dû laisser aller certaines des valeurs lues précédemment
- ✓ De conflit – lorsque deux adresses distinctes correspondent à la même entrée dans la mémoire cache

- La performance du système dépend aussi du délai encouru lors d'un « miss ».
- Pour optimiser la performance d'un système, il faut donc à la fois réduire le nombre de misses, et réduire les délais encourus lors de « misses »

91

Types de caches

1. Caches transparents: (par rapport à l'applicatif)

- ✓ Heuristiques internes sur ce qu'il faut maintenir dans le cache.
- ✓ Version originale rémanente sur un support hiérarchiquement inférieur
- ✓ Cache de GPP, file cache (sys. fichiers distribués), cache proxy, etc.

2. Scratch-pads:

- ✓ Géré explicitement par « l'applicatif client »
- ✓ Unique version, cache non autonome
- ✓ Fichier de registres, DSP (SRAM non *taggé)*

3. Caches hybrides:

- ✓ Autonomie + gestion explicite
- ✓ Ex: buffer cache (linux), stocke des portions de fichiers
- ✓ applicatif + peut servir au noyau (gestion explicite)

92

Von Neumann vs. Harvard

93

Architecture Classique "Von Neumann"

- ➢ Une mémoire contenant les adresses et les données.
- ➢ Une "Central processing unit" (CPU) qui lit les instructions de la mémoire.
- ➢ La CPU contient des registres :
  - ✓ program counter (PC),
  - ✓ instruction register (IR),
  - ✓ general-purpose registers, …
  - ✓ Etc….

94

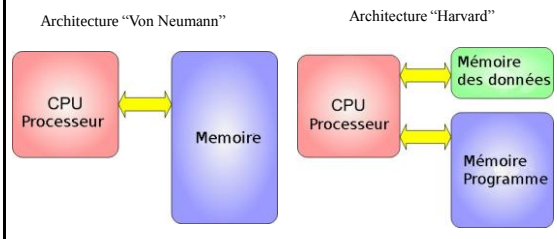
Architecture "Harvard"

Howard Aiken, université de Harvard, 1946

- ➢Utilise un programme stocké en mémoire
- ➢Programme et données stockés dans des mémoires séparées
- ➢Dispose de 2 ensembles de bus, soit :
  - ✓ un ensemble : CPU-Mémoire programme
  - ✓ un ensemble : CPU-Mémoire donnée-E/S

95

von Neumann vs. Harvard



96

Von Neumann vs. Harvard

- L'architecture Harvard permet deux lectures simultanées .
- La plupart des DSP (digital signal processor) utilisent l'architecture Harvard
  - Note : DSP processeur à archi dédiée pour traitement de signal (opération sur les matrices)
- Intéressante dans le cas des applications traitant des données à la volée « streaming data »:
  - Permet une plus grande bande passante pour les données (nombre de données lues ou écrites par cycles importantes)
  - Comme la lecture des données et la lectures des instructions se font sur des bus différents, il y a moins d'interférence, une plus grande prédictibilité au niveau de la largeur de bande.

97

Les architectures RISC et CISC (1)

Actuellement l'architecture des microprocesseurs se composent de deux grandes familles :

- L' architecture CISC
  (Complex Instruction Set Computer)
- L'architecture RISC
  (Reduced Instruction Set Computer)

98

Les architectures RISC et CISC (2)

Architecture RISC	Architecture CISC
instructions simples ne prenant qu'un seul cycle	instructions complexes prenant plusieurs cycles
instructions au format fixe	instructions au format variable
décodeur simple (câblé)	décodeur complexe (microcode)
beaucoup de registres	peu de registres
peu de modes d'adressage	beaucoup de modes d'adressage
compilateur complexe	compilateur simple

99