

Architecture Matérielle des Ordinateurs

**Quatrième Partie :
Architectures évoluées**

©Theoris 2007

Plan

- 1 **Le calculateur numérique**
- 1 **Les Entrées - Sorties**
- 1 **Notions d'assembleur**
- 1 **Architectures évoluées**

Plan quatrième partie

- u **Introduction:**
 - u Philosophie globale
 - u Rappel historique
 - u Le marché des processeurs
- u **Évolution Unité Centrale:**
 - u Pipeline,
 - u Pipeline performant:
 - u Super-scalaire,
 - u Exécution spéculative,
 - u Renommage.
 - u TLP
 - u Hiérarchie mémoire (DRAM, Cache, ...),

Introduction: Philo globale

- 1 ***Augmenter la performance du CPU:***
 - v Approche dirigée par la contrainte de réalisation jusqu'en 1980 (CISC),
 - v Approche dirigée par l'étude de la programmation après 1980, en se basant sur des jeux de tests (architecture RISC).
- 1 **Langage machine est compatible (x86)**
Les processeurs d'aujourd'hui sont des « CRISC » où l'on prend le meilleur des deux mondes.

Introduction: RISC vs CISC

Reduce Instruction Set Computer	Complex Instruction Set Computer
Instructions simples ne prenant qu'un seul cycle	Instructions complexes prenant plusieurs cycles
Format fixe	Format variable
Décodeur câblé	Décodeur micro-codé
Beaucoup de registres	Peu de registres
Seules les instructions LOAD et STORE ont accès à la mémoire	Toutes les instructions sont susceptibles d'accéder à la mémoire
Peu de mode d'adressage	Beaucoup de mode d'adressage
Compilateur complexe	Compilateur simple

© Theoris 2004

5

Introduction: historique (1)

1 Évolution du CPU pour favoriser l'exécution d'instructions (famille x86 / Motorola uniquement):

- ∨ **Augmentation de la largeur de bits traités:**
 - › 4bits en 71 (8086), 32bits en 79 (68000), 64bits en 92 (AMD Athlon).
- ∨ **Cache: 1975 (68010) quelques octets,**
- ∨ **Integration des MMU/FPU: 1989 (80486 /68040),**
- ∨ **Pipeline: 1989 (68040 / 80486), heritage du RISC**
- ∨ **Super-scalaire: 1993 (Pentium / 68060)**

© Theoris 2004

6

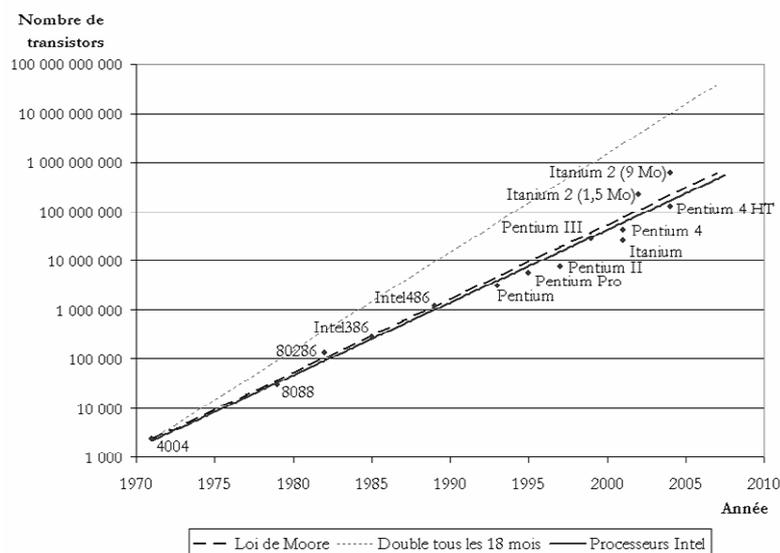
Introduction: historique (2)

- v Exécution dé-séquentée: 1995 (Pentium Pro),
 - v Instructions SIMD: 1993 (Pentium avec le MMX),
 - v Contrôleur mémoire intégré: 2003 (AMD Athlon 64)
- 1 Toutes ces améliorations sont possibles grâce à l'intégration de plus en plus de transistors sur le CPU (baisse taille gravure)
 - 1 Augmentation de la fréquence permet d'augmenter la performance mécaniquement
 - 1 Pb récent (2003): le nombre de transistors et l'augmentation de la fréquence entraînent une dissipation énergétique qui bruite le signal, et crée des problèmes de dissipation.

© Theoris 2004

7

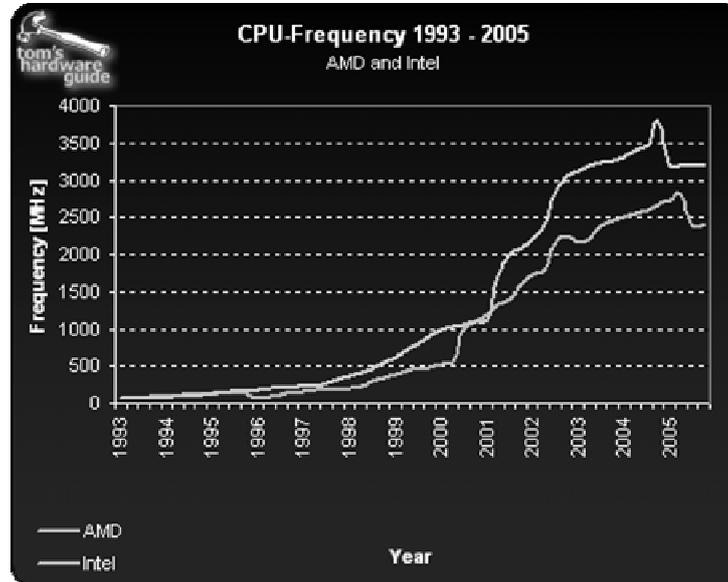
Introduction: historique (3)



© Theoris 2004

8

Introduction: historique (4)



© Theoris 2004

9

Introduction: historique (5)

- 1 **Pour pallier au problème de consommation, on introduit l'exécution simultanée de programmes**
- 1 **Augmenter le TLP: Parallélisation**
 - ∨ **SMT (Hyper-threading): 2002 ? (Pentium 4)**
 - ∨ **CMP (Dual-Core): 2005/2006 (Athlon X2, Pentium D, Core Duo)**
- 1 **Une des voies d'avenir des processeurs est de faire monter le nombre de cœurs du processeur**

© Theoris 2004

10

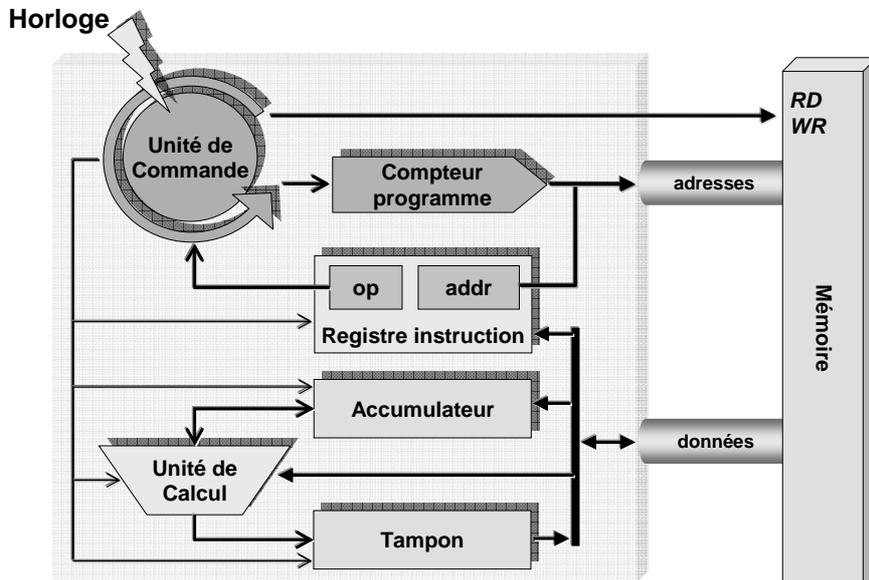
Introduction: marché

- 1 **Les CPUs sont le résultat d'un compromis suivant les segments (conso, perf, prix):**
 - v **Portable: Faible consommation, exigence performance « faible », prix élevé.**
 - v **Bureau: Consommation importe peu, exigence performance moyenne, prix bas.**
 - v **Serveur: Consommation importe peu, exigence performance élevée, prix très élevé.**
- 1 **Concentration des acteurs depuis dix ans, acteurs étudiés: AMD, Intel**
- 1 **Évolution récente d'Intel: une seule architecture pour tous les segments (architecture « Core 2 Duo »)**

Plan quatrième partie

- u **Introduction:**
 - u **Philosophie globale**
 - u **Rappel historique**
 - u **Le marché des processeurs**
- u **Évolution Unité Centrale:**
 - u **Pipeline,**
 - u **Pipeline performant:**
 - u **Super-scalaire,**
 - u **Exécution spéculative,**
 - u **Renommage.**
 - u **TLP**
 - u **Hierarchie mémoire (DRAM, Cache, ...),**

Organisation de l'unité centrale



© Theoris 2004

13

RISC vs CISC

Reduce Instruction Set Computer	Complex Instruction Set Computer
Intructions simples ne prenant qu'un seul cycle	Instructions complexes prenant plusieurs cycles
Format fixe	Format variable
Décodeur câblé	Décodeur micro-codé
Beaucoup de registres	Peu de registres
Seules les instructions LOAD et STORE ont accès à la mémoire	Toutes les instructions sont susceptibles d'accéder à la mémoire
Peu de mode d'adressage	Beaucoup de mode d'adressage
Compilateur complexe	Compilateur simple

© Theoris 2004

14

Pipeline: introduction

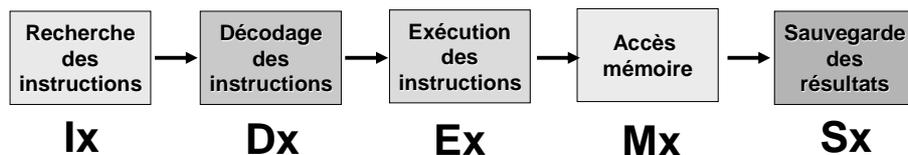
- 1 L'objectif du pipeline est de permettre l'exécution efficace d'instructions
- 1 La mise à disposition d'un nombre de transistors élevé est une opportunité
- 1 Une solution: en décomposant en exécutions élémentaires de durées équivalentes, on obtient un seul schéma d'exécution pour toutes les instructions.
- 1 Objectif de performance: 1 cycle par instruction élémentaire (CPI = 1).

Pipeline: traduction d'instruction

- 1 Les instructions de haut de niveau (asm x86) sont traduites en instructions élémentaires:
 - v Load Reg, Adresse Mem
 - v Store Adresse Mem, Reg
 - v Add / Sub / Mul Reg1, Reg2, Reg 3 (UAL)
 - v Addf / Subf / Mulf FReg1, FReg2, Freg3 (F-UAL)
 - v Branch Reg, imm (conditionnel)
 - v Jump imm (in-conditionnel)

Pipeline: principe

1 Exemple de pipeline à 5 unités fonctionnelles ou étages



© Theoris 2004

17

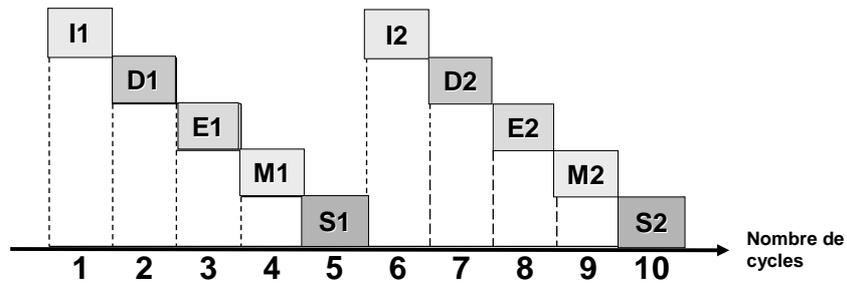
Pipeline: instruction élémentaire

1. **Charger l'instruction à exécuter depuis la mémoire dans le registre d'instruction ET modifier le compteur ordinal pour qu'il pointe sur l'instruction suivante**
2. **Décoder l'instruction que l'on vient de charger**
3. **Exécuter l'instruction, soit:**
 1. **Calculer les adresses en mémoire d'éventuelles données nécessaires (instructions de Load / Store) OU**
 2. **Exécuter l'instruction (calculs)**
4. **Si l'instruction est de type Load/Store, charger les données dans les registres généraux de l'U.C.**
5. **Revenir à l'étape 1 pour entreprendre l'exécution de l'instruction suivante**

© Theoris 2004

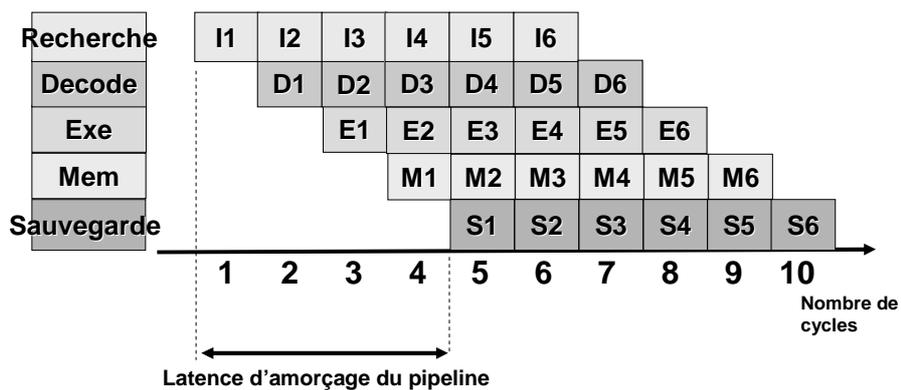
18

Pipeline: modèle classique



Perf: Chaque instruction a une latence de 5c,
Le CPU un CPI = 5c

Pipeline: dynamique

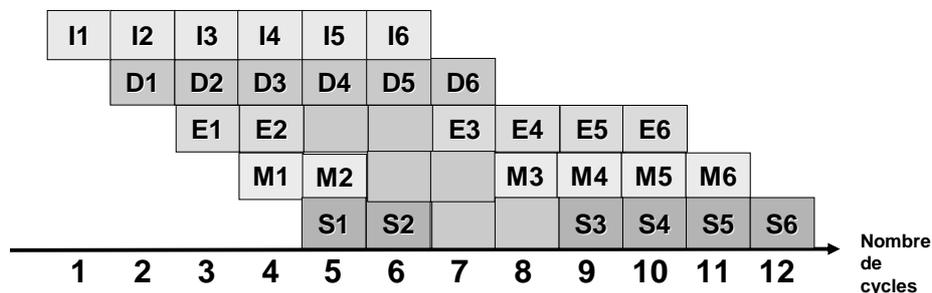


Perf: Chaque instruction a une latence de 5c,
Le CPU a par contre un CPI = 1c

Pipeline: les aléas

- 1 La performance du pipeline est dégradée dans les cas suivants:
 - ∨ aléa structurel: deux instructions ont besoin de la même ressource du processeur (accès mem) Bulle
 - ∨ aléa de donnée: une instruction dépend du résultat de la précédente Bulle
 - ∨ aléa de contrôle: branchement attente de l'étage « Mem ». L'impact performance dépend de la longueur du pipeline.
 - ∨ Accès mémoire: la donnée ne se trouve pas dans les registres Bulle.

Pipeline: exemple d'aléa



Perf: Le CPI du processeur est > 1

Pipeline: Le bilan

- 1 **Intel: Un pipeline qui varie bcp**
 - ∨ P-IV Core « northwood »: 20.
 - ∨ P-IV Core « Prescott / Presler »: 31.
 - ∨ P-IV Core « Tejas »: 45 (jamais sorti)
 - ∨ Pentium-M Core « Yonah »: 12
 - ∨ Core-Duo 2 Core « Conroe »: 14
- 1 **AMD: Un pipeline court**
 - ∨ Athlon XP: 10
 - ∨ Athlon 64: 12
 - ∨ Athlon X2 Core « Windsor »: 12

Pipeline: conclusion

- 1 **Principe fondateur de tous les processeurs actuels.**
- 1 **Les aléas peuvent impacter fortement la performance (en fonction de la profondeur du pipeline, latence mémoire).**
- 1 **La décomposition en éléments simples accélère la montée en fréquence.**
- 1 **Le CPI effectif d'un pipeline idéal sur un processeur idéal en régime permanent est de 1 (pas d'accès mémoire, ...).**

Plan quatrième partie

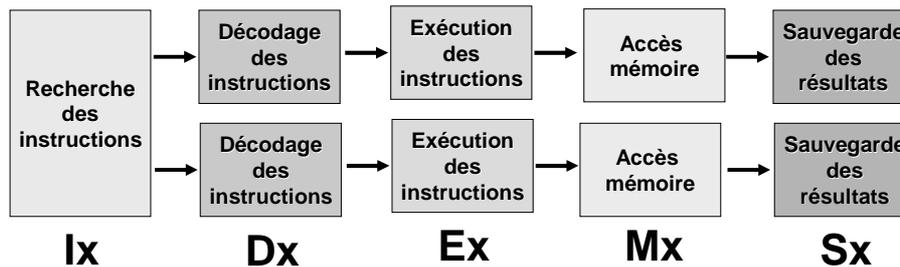
- u **Introduction:**
 - u **Philosophie globale**
 - u **Rappel historique**
 - u **Le marché des processeurs**
- u **Évolution Unité Centrale:**
 - u **Pipeline,**
 - u **Pipeline performant:**
 - u **Super-scalaire,**
 - u **Exécution spéculative,**
 - u **Renommage.**
 - u **TLP**
 - u **Hiérarchie mémoire (DRAM, Cache, ...),**

Pipeline performant: Introduction

- 1 **Le but est double:**
 - v **augmenter encore le CPI et**
 - v **réduire l'impact des aléas.**
- 1 **Les solutions consistent en un nombre de techniques indépendantes spécifiques à chaque problème, motivées par approche quantitative.**
- 1 **L'objectif de performance sera un CPI < 1 en augmentant l'ILP.**
- 1 **Ce chapitre est en constante évolution.**

Superscalaire: introduction

- 1 Pour augmenter le CPI on recherche les instructions pouvant s'exécuter en parallèles,
- 1 $CPI\ idéal = 1 / \text{nbr de pipelines (3-4)}$



© Theoris 2004

27

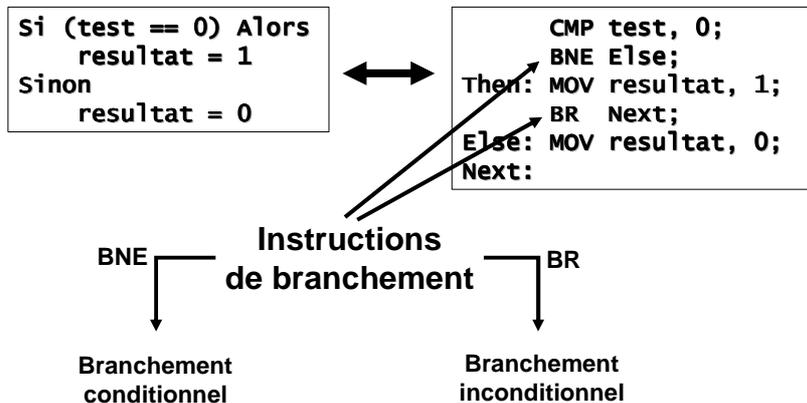
Les aléas: branchement

- 1 Le résultat d'un branchement oblige à attendre le résultat de l'étage MEM avant de poursuivre l'exécution.
- 1 La bulle créée dans le pipeline est importante: impact en performance.
- 1 Prédire les branchements, et exécuter spéculativement le flot d'instructions prédit.
- 1 Si la prédiction est exacte, on continue, sinon on revient à l'état précédent le branchement.

© Theoris 2004

28

Les aléas: type branchement



© Theoris 2004

29

Les aléas: type branchement

1 Branchement inconditionnel

- ∨ Retard d'analyse dans le pipeline \pm l'instruction suivante le branchement est amorcée (créneau temporel)
- ∨ On se repose sur le compilateur pour avoir une instruction « utile »
- ∨ Souvent il s'agira d'un NOP

1 Branchement conditionnel

- ∨ créneau temporel
- ∨ retard dans la localisation du saut: l'information arrive tardivement dans le pipeline
- ∨ mécanisme de prédiction de branchement basé sur une statistique de réalisation des conditions de branchement

© Theoris 2004

30

Les aléas: Hypothèses sur les branchements

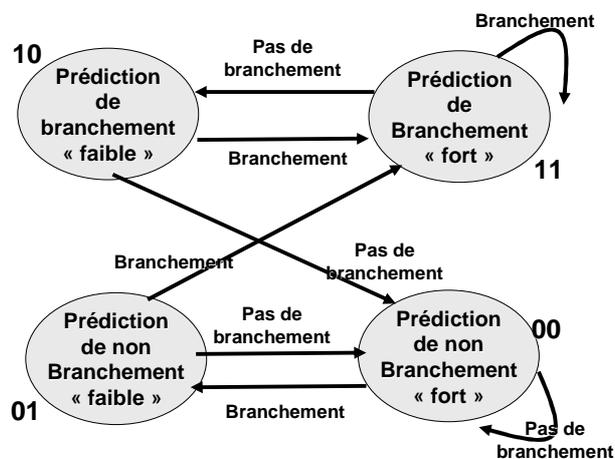
- 1 Les branchements conditionnels arrières se réalisent presque toujours
 - ∨ Principe de la boucle
 - ∨ Bonne hypothèse
- 1 Les branchements conditionnels avants ne se réalisent quasiment jamais
 - ∨ Principe du test d'erreur
 - ∨ Hypothèse plus douteuse (un if dans une boucle)

© Theoris 2004

31

Les aléas: Prédiction dynamique de branchement

⌘ Mise en place d'une table d'histoire associative (BHT) (idéalement une par adresse)



© Theoris 2004

32

Les aléas: Prédiction dynamique de branchement

- 1 **En cas d'erreur de branchement, l'instruction en cours n'est pas la bonne**
- 1 **Dans tous les cas la cohérence de l'état machine doit être assurée**
 - v **Soit on choisit de modifier des registres fantômes inaccessibles à l'utilisateur pendant la prédiction**
 - v **Soit on mémorise l'état machine avant la prédiction pour pouvoir le restaurer ensuite**
- 1 **Branchements imbriqués: prédicteur à corrélation**

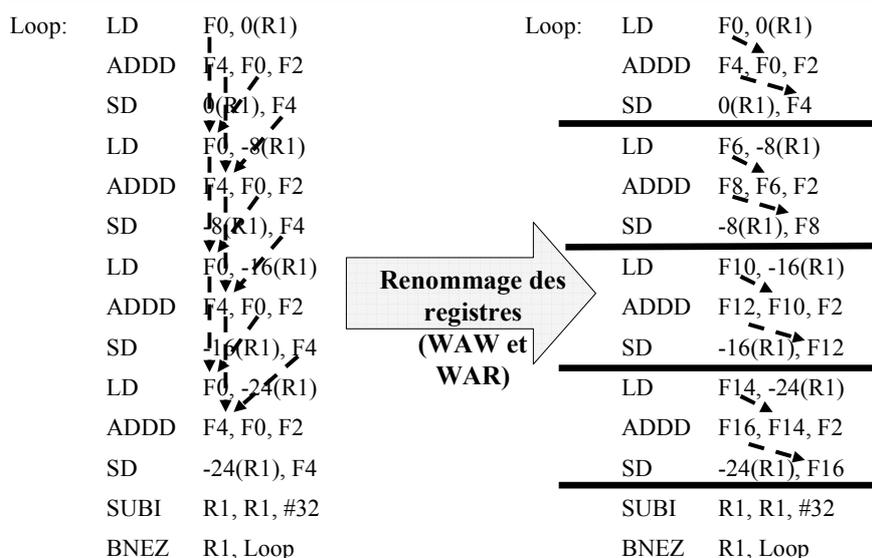
Exécution spéculative: bilan

- 1 **Avec la technique de prédiction de branchement, on peut alors exécuter les instructions de la branche prédite, c'est un gain de temps appréciable.**
- 1 **Si par contre la prédiction est fautive on est obligé de revenir à l'état d'avant le branchement, ce qui coûte beaucoup (vidage pipe, remise des registres d'état)**
- 1 **Cependant les prédicteurs marchent bien (taux > 95%) donc gain perf.**

Retour sur les aléas: Dépendances des données

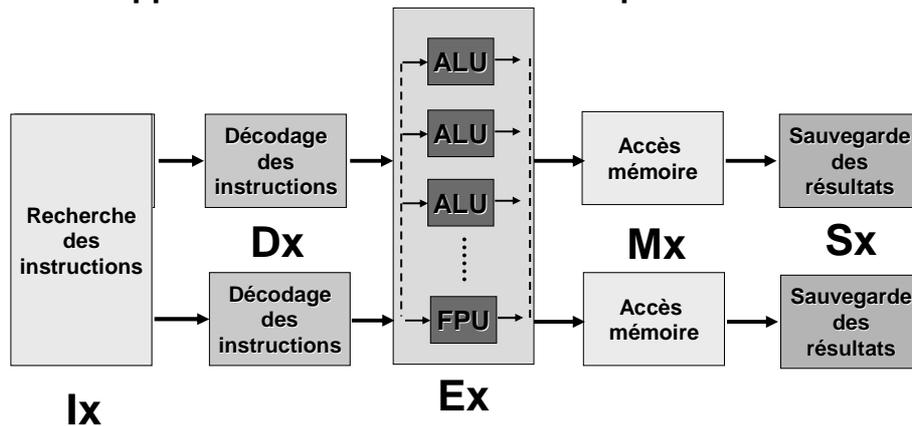
- 1 Enregistrement des états lecture/écriture des registres dans une unité de marquage
- 1 Différents types de dépendances liées aux accès registres:
 - ∨ RAW: Read After Write
 - ∨ WAR: Write After Read
 - ∨ WAW: Write After Write
- 1 WAR et WAW peuvent être résolus par l'utilisation de registres fantômes
 - ⌘ renommage de registre (étage supplémentaire)
- 1 RAW implique un blocage, on essaie de poursuivre avec d'autres instructions indépendantes
 - ⌘ exécution déséquentée

Renommage des registres



Exécution déséquenté

- 1 Pour augmenter la performance, on multiplie les unités d'exécution: ALU, FLU, ... Le but est de supporter un flot d'instructions important.



© Theoris 2004

37

Exécution déséquenté

- 1 Les instructions sont lues dans l'ordre du programme et exécutées dans l'ordre des dépendances de données.
- 1 On peut ainsi lancer les instructions « lentes » au plus tôt, en donc pas dans l'ordre original.
- 1 Reordonnement à la fin:
 - v ROB
 - v Table de registre
- 1 Complexe à mettre en œuvre (Tomasulo).

© Theoris 2004

38

VLIW (Very Long Instruction Word)

- 1 **Processeur à mot d'instruction très long: 128, 256 bits ...**
- 1 **A l'intérieur de l'instruction sont codées les instructions élémentaires à effectuer par chaque unité de calcul**
- 1 **Dépend totalement du compilateur**
- 1 **Concept hérité des mainframes IBM des années 60 (360, 370)**
- 1 **Première machine produite par Multiflow en 1984-85: 256 bits (8 groupes de 32 bits)**
- 1 **L'arrivée du RISC a stoppé l'essor de ces processeurs généralistes, et le réserve au marché de l'embarqué**

VLIW (Very Long Instruction Word)

- 1 **Ressort actuellement avec:**
 - v **Transmedia de Philips**
 - v **Crusoë de Transmeta**
- 1 **Héritage important dans le domaine de la compilation**
- 1 **EPIC (Explicitly Parallel Instruction Computing) d'Intel dans les processeurs *Itanium* (2001) s'en inspire: disparition du réordonnement des instructions pour se reposer entièrement sur l'ordre des instructions fournit par le compilateur**

Pipeline performant: bilan

- 1 **Pipeline performant du PIV:**
 - v **Superscalaire (3 instr)**
 - v **Prédiction dynamique de branchement et tampons d'adresses de branchements (BTB, 4K entrées)**
 - v **Pipeline: Fetch, Decode, Alloc (renaming), Reorder, Execute, WB, Retirement**
 - v **Chaque étage est effectué en plusieurs cycle d'horloge pour permettre de mettre moins de portes par sous-étage.**
- 1 **Permet la montée en fréquence.**

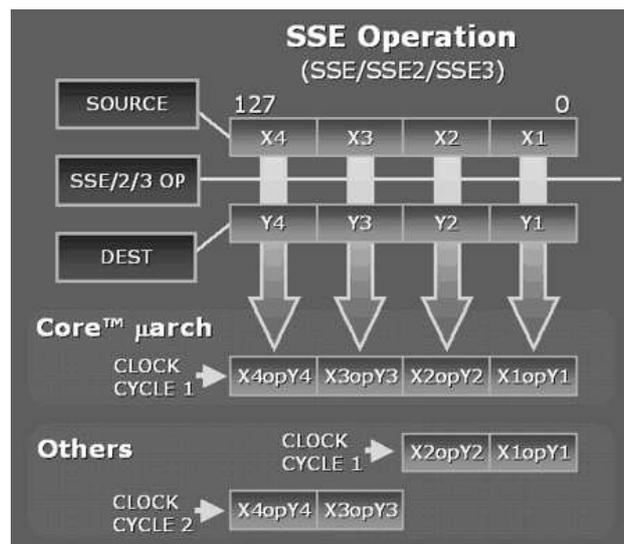
Pipeline performant: bilan

- 1 **Un ensemble de techniques quasi-indépendantes.**
- 1 **La performance résultante est une balance entre ces techniques.**
- 1 **Les processeurs actuels implémentent la plupart de ces techniques (Superscalaire, pred branchement, OOO, renommage registres).**
- 1 **On a effectivement fait baisser le CPI**

SIMD: Instructions multimédia

- 1 **Constat: les images et autres données multimédia sont de longues fils de données consécutifs en mémoire.**
- 1 **La modification des données sont réguliers (filtrage sur TOUTE l'image)**
- 1 **Une solution pour accélérer ces données sont de traiter plusieurs par bloc de données.**
- 1 **Nouvelles instructions: MMX, SSE, 3DNow.**

SIMD: schéma



SIMD: Support matériel

- 1 **On définit des registres spécifiques larges (128-bits par exemple dans SSE)**
- 1 **Arithmétique spécifique (ex: saturation 8bit: Si $(res \geq 256) \rightarrow res = 256$).**
- 1 **Les opérations dépendent de la largeur de la sous donnée a traiter (8, 16, 32 ou 64 bits).**
- 1 **On ajoute des unités d'exécutions spécifiques qui utilisent ces registres, en modifiant Ex.**

SIMD: conclusion

- 1 **Fort facteur de gain sur des applications spécifiques, en fonction de la taille des données à traiter (jusqu'à x16).**
- 1 **Une technique qui se repose cependant beaucoup sur le programmeur, et peu sur le compilateur (vectorisation automatique).**
- 1 **Cependant les modifications sont locales à un module software.**
- 1 **Augmente fortement la pression sur la hiérarchie mémoire.**

Plan quatrième partie

- u **Introduction:**
 - u **Philosophie globale**
 - u **Rappel historique**
 - u **Le marché des processeurs**
- u **Évolution Unité Centrale:**
 - u **Pipeline,**
 - u **Pipeline performant:**
 - u **Super-scalaire,**
 - u **Exécution spéculative,**
 - u **Renommage.**
 - u **TLP**
 - u **Hiérarchie mémoire (DRAM, Cache, ...),**

TLP: technique

- 1 **Augmentation du ILP n'est pas la seule voie, le TLP en est une autre.**
- 1 **Plusieurs solutions sont envisagées:**
 - v **SMT et CMP.**
- 1 **Cependant ces méthodes s'appuient sur la coopération du programmeur (multi-thread).**
- 1 **Du point de vue de l'OS ou des utilisateurs le système est vu comme plusieurs processeurs physiques**

TLP: les threads

- 1 **Une thread est une unité d'exécution élémentaire partageant des ressources avec les autres threads au sein d'un processus.**
- 1 **Les processus ont des espaces mémoires séparés, pas les threads.**
- 1 **Pb de synchronisation limite le nombre de thread d'une application (problème d'architecture du software).**
- 1 **Du point de vue de l'OS ou des utilisateurs le système est vu comme plusieurs processeurs physiques**

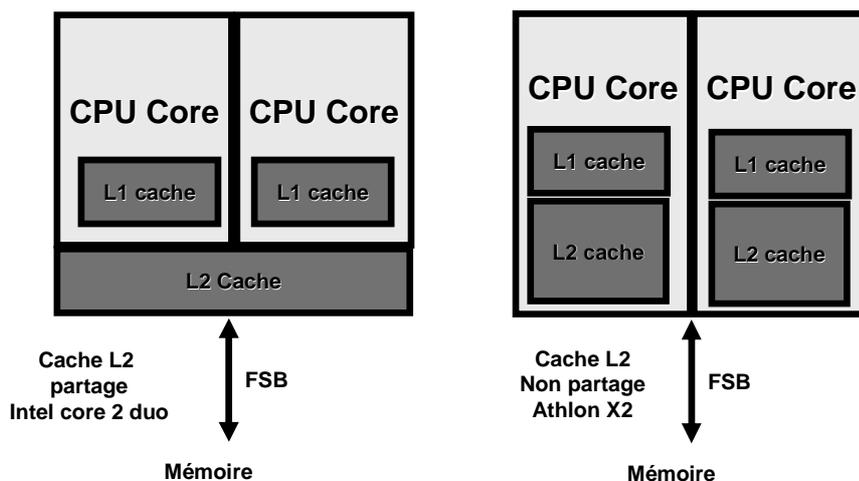
OS et threading: SMT

- 1 **Chaque processeur logique possède:**
 - ∨ des registres généraux,
 - ∨ des registres de contrôle,
 - ∨ quelques registres d'état,
 - ∨ son contrôleur d'interruption (APIC)
- 1 **Les processeurs logiques partagent:**
 - ∨ les caches,
 - ∨ les unités d'exécution,
 - ∨ les prédicateurs de branches,
 - ∨ les bus

OS et threading: CMP

- 1 Chaque processeur physique possède:
 - ∨ Des registres,
 - ∨ des caches,
 - ∨ les unités d'exécution,
 - ∨ les prédicateurs de branches,
 - ∨ les bus
- 1 Certains caches peuvent être partagés
- 1 Problématique de cohérence des caches partagés: MOESI.

CMP: diagramme



TLP: Conclusion

- 1 **Une voie d'avenir avec l'augmentation du nombre de cœurs (arrivée de processeurs quadri-core).**
- 1 **Une problématique mixte soft / hard avec le besoin de thread pour tirer parti de l'architecture matérielle.**
- 1 **La problématique software ne doit pas être négligée.**
- 1 **Augmentation de la pression sur le système mémoire.**

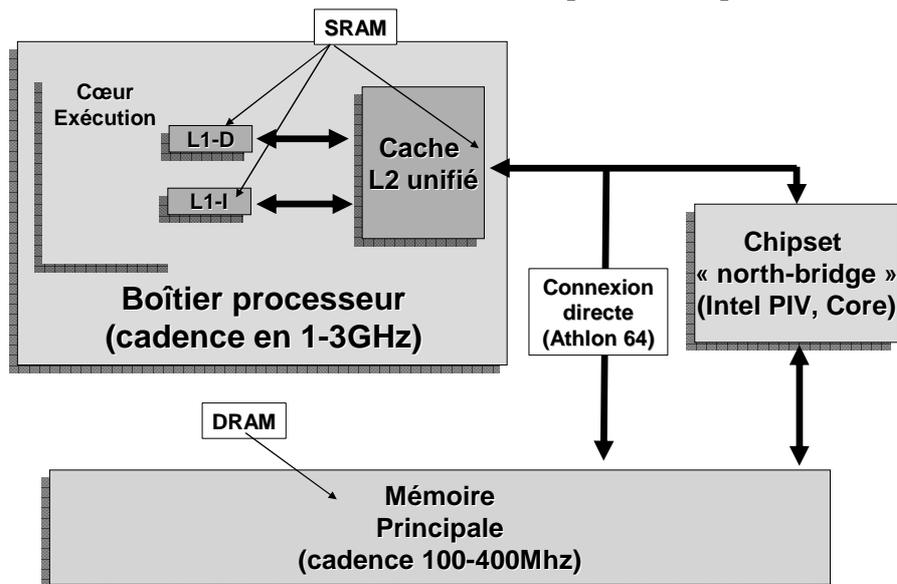
Plan quatrième partie

- u **Introduction:**
 - u **Philosophie globale**
 - u **Rappel historique**
 - u **Le marché des processeurs**
- u **Évolution Unité Centrale:**
 - u **Pipeline,**
 - u **Pipeline performant:**
 - u **Super-scalaire,**
 - u **Exécution spéculative,**
 - u **Renommage.**
 - u **TLP**
 - u **Hierarchie mémoire (DRAM, Cache, ...).**

Hiérarchie mémoire

- 1 L'objectif de la hiérarchie mémoire est de ne pas faire attendre le CPU pour les instructions et les données à exécuter
- 1 Contrainte de coût: la mémoire rapide est très chère (SRAM / DRAM).
- 1 Solution: organisation en niveaux, chaque niveau étant plus petit et plus rapide que le précédent.
- 1 Critère performance: débit et latence

Hiérarchie mémoire: principe



Techno des mémoires vives (RAM = Random Access Memory)

- 1 Mémoires statiques
 - Faciles à mettre en œuvre
 - Rapides
 - β Volumineuses
- 1 Mémoires dynamiques
 - Bon marché
 - Grande capacité
 - β Nécessitent un rafraîchissement
 - β Lentes (cycles de sélection)

© Theoris 2004

57

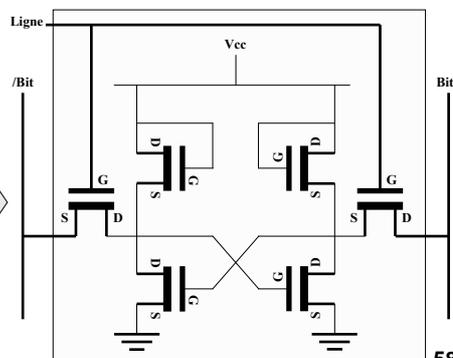
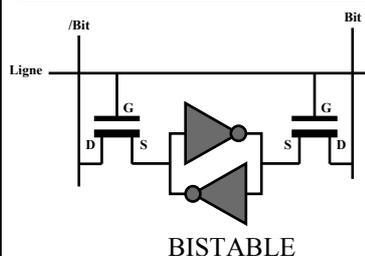
S-RAM : cellule statique

Écriture :
Polarisation du signal Ligne
Polarisation des lignes Bit et /Bit
Stockage dans la paire d'inverseurs

Lecture :
Polarisation du signal Ligne
Récupération des sorties Bit et /Bit

Coût : 6 transistors / bit

Implantation sur le CPU



© Theoris 2004

58

D-RAM : cellule dynamique

Écriture :

Polarisation de Ligne
Stockage dans la capacité

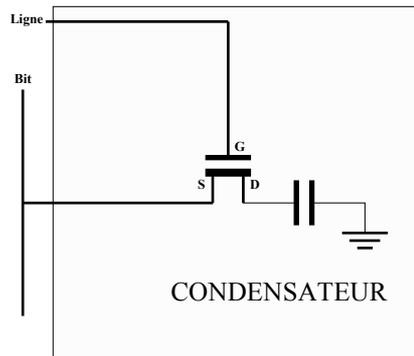
Nécessité d'augmenter la charge de la capacité avant lecture

La capacité est déchargée par la lecture
Une **réécriture** est nécessaire après lecture

Le déchargement progressif de la capacité nécessite des opérations de **rafraîchissement** périodiques

Coût : 1 transistor + 1 capacité / bit

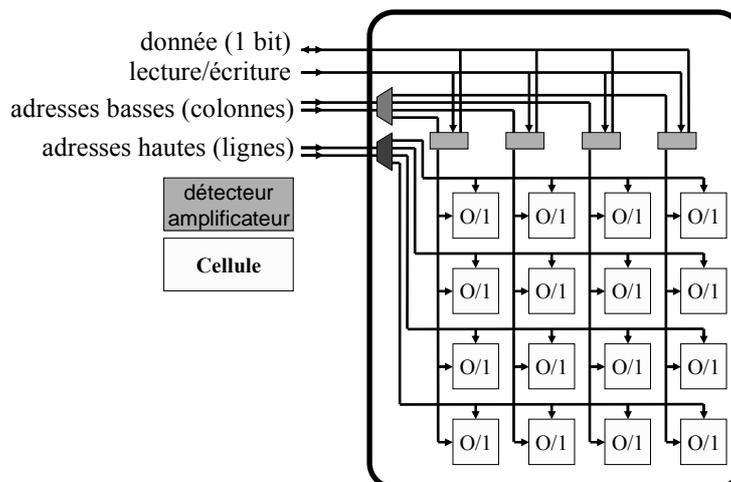
Implantation en dehors du CPU



© Theoris 2004

59

Mémoire : principe général

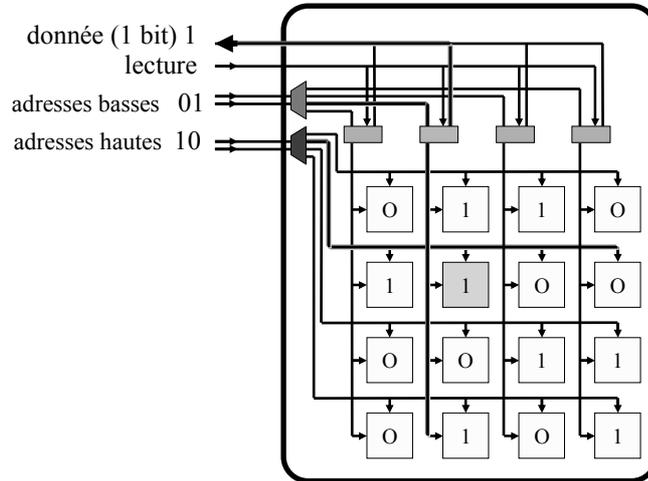


© Theoris 2004

60

Mémoire : exemple de lecture

1 Bit à l'adresse 9 du boîtier :



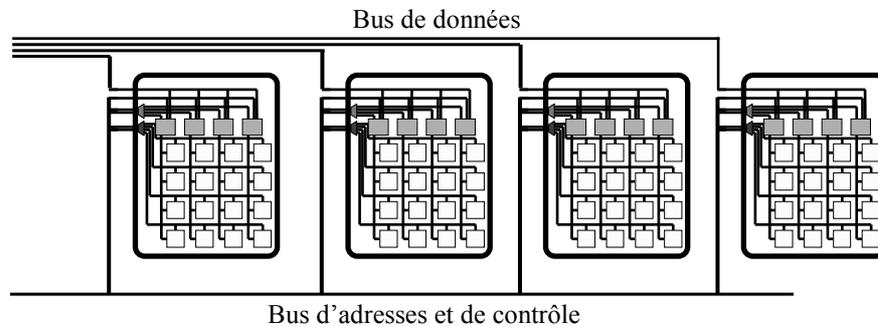
© Theoris 2004

61

Mémoire : bus de données

1 Création d'une mémoire de mots par assemblage de boîtiers à un bit

Mots de un octet = 8 boîtiers



© Theoris 2004

62

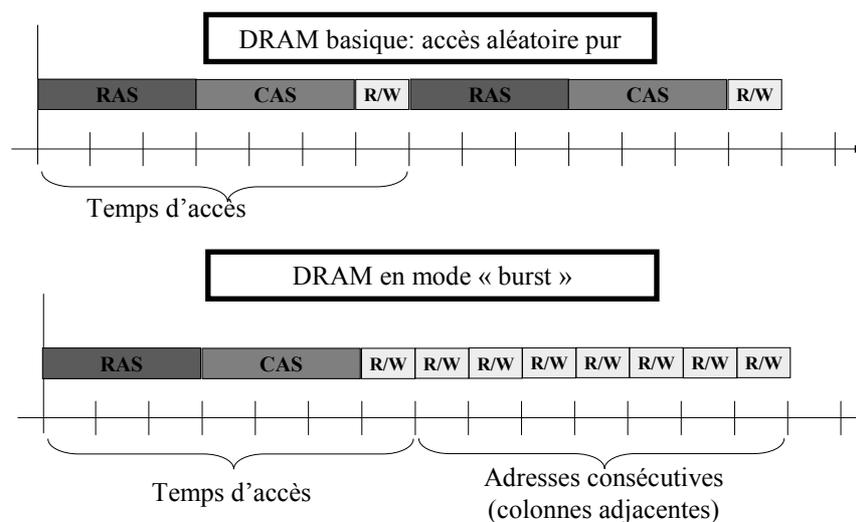
D-RAM: cycles d'accès

- 1 **Signaux d'accès RAS et /CAS**
 - Raw Acces Strobe / Column Acces Strobe
 - Chaque cycle dure environ 2 à 3 coups d'horloge
 - Coût d'accès aléatoire : environ 7 coups d'horloge
- 1 **Accès en rafale (burst)**
 - Chargement rapide des données situées à des adresses consécutives
 - Un seul cycle RAS/CAS pour 8 ou 16 mots
 - Coût d'accès réduit à environ 2 coups d'horloge

© Theoris 2004

63

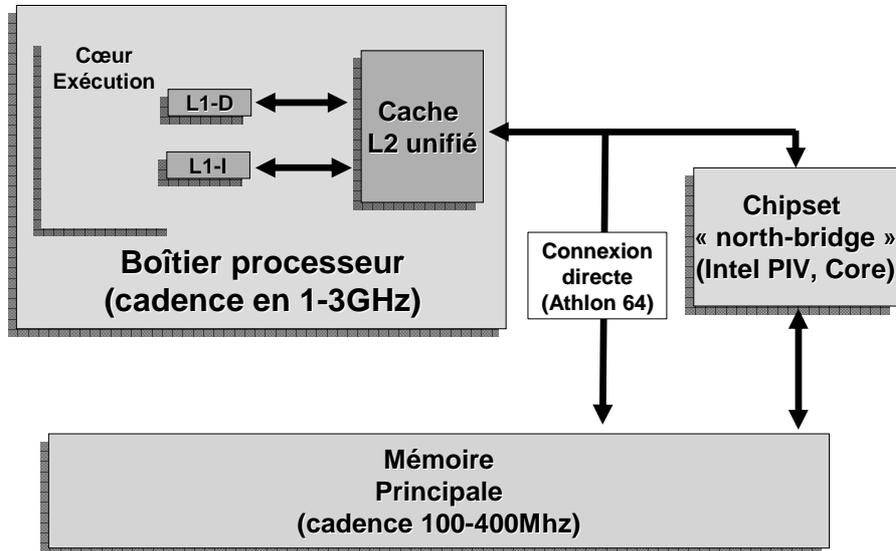
Cycles d'accès RAM dynamique



© Theoris 2004

64

Hiérarchie mémoire: CACHE

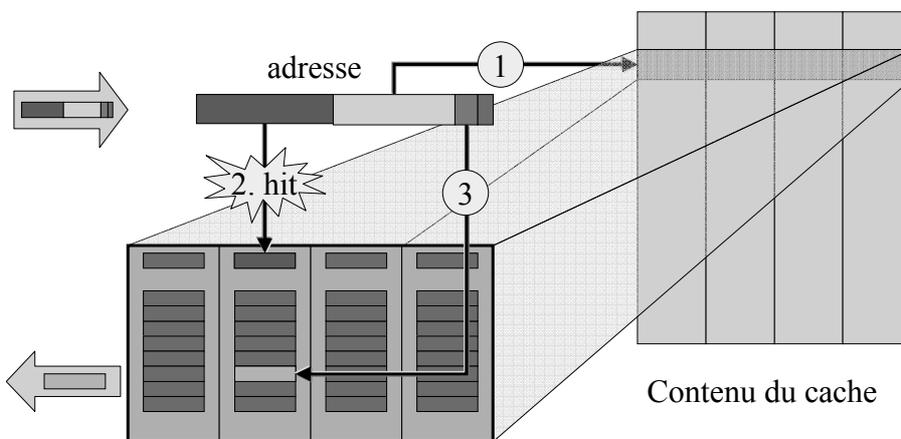


© Theoris 2004

65

Fonctionnement du cache

1 Principe : une mémoire associative

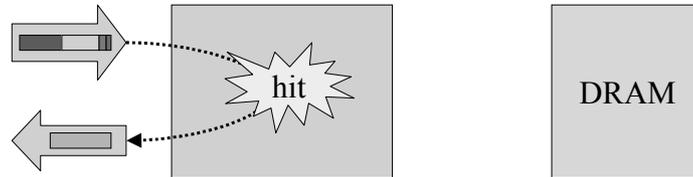


© Theoris 2004

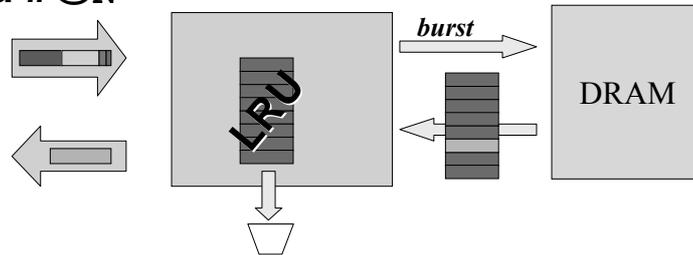
66

Fonctionnement du cache

Gagné !



Perdu #@N



© Theoris 2004

67

Mémoires: dernières évolutions

1 Dual-channel (DDR):

- ∨ Parallélisation des accès mémoires par l'ajout « dual-channel », où l'on accède à plusieurs barrettes en parallèle.
- ∨ Double effectivement la bande passante MAX de la mémoire. Peu de gain en pratique (~10%), mais coût très faible.

1 Prefetcher:

- ∨ Ajout d'un prefetcher hardware, qui anticipe les accès réguliers à la mémoire.
- ∨ Efficace en accès « séquentiel » (applications de traitement d'image, SIMD), masquant la latence mémoire.

© Theoris 2004

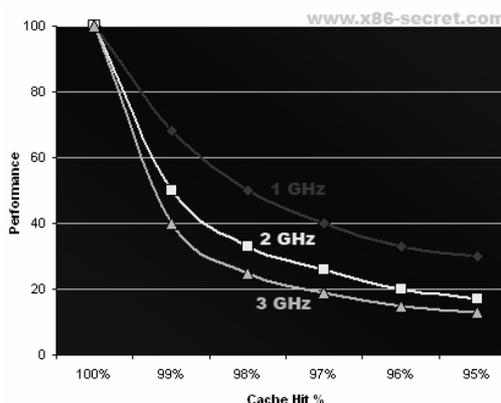
68

D-RAM: Le bilan

- 1 **DRAM DRAM-FP DRAM-EDO**
SDRAM (5-1-1-1)
 - ∨ **R-DRAM (Rambus) 16bits/800Mhz, 3.2Go/s (PS2) XDR 16Bits/3.2Ghz Nx25Go/s.**
 - ∨ **DDR-SDRAM (200-400Mhz) 64bits / 3.2Go/s**
DDR2-SDRAM (600-800), 64bits 6.4Go/s
 - › **FB-Dimm (serveur) avec contrôleur (40Go/s)**
 - › **G-DDRII et III spécifique pour les GPUs, à 1GHz délivre 16Go/s.**

Cache: bilan

- 1 **Plus la fréquence augmente plus de taux de succès du cache est important**



Performance mémoire: Bilan

CPU	L1	L2	Mémoire (Aléatoire)
Pentium IV 3.2GHz	16Ko I/D 4cyc, 25G/s	1Mo I+D 31c,15G/s	DDR2-533 (800) 140cyc, 4.2Go
Intel Core 2 Duo 3.0GHz	2x32Ko I/D 3c, 2x48Go/s 16o/c	4Mo I+D 14c 48Go/s 16o/2c	DDR2-1066 90c, 10.6Go/s
AMD Athlon X2 4800+ 2.4Ghz	2x64Ko I+D 3c, 2x20Go/s 2x8o	2x1Mo I+D 12c, 2x9Go/s	DDR2-800 108c, 6Go/s
IBM Cell PS3	Spécial: 16Ko / SPU 64Go/s (continu)		2xXDR – 256 Mo 25Go/s x 2

© Theoris 2004

71

Hiérarchie mémoire: conclusion

- 1 **Système efficace des caches (perf, coût et consommation), mais la latence reste problématique.**
- 1 **Domaine critique: les CPUs ne font qu'attendre des instructions et des données à traiter,**
- 1 **Avec l'augmentation du nombre de coeurs, la demande mémoire va augmenter.**

© Theoris 2004

72

Christophe BIQUARD
cbiquard@theoris.fr

Yannick BALERE
yannick.balere@theoris.fr

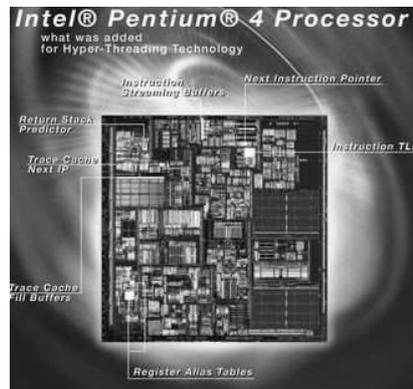
NetBurst



- 1 **Architecture Intel sortie en 2000**
- 1 **Pipeline long (20 étages)**
- 1 **Introduction du « Trace Cache »: le cache L1 d'instruction cède sa place à un cache optimisé contenant les instructions déjà décodées**
- 1 **Cette architecture lance la course au MHz pour contrer le handicap de la longueur de pipeline**
- 1 **Le *Northwood* (0.13µm, 512Ko de cache) cadencé à 3.2GHz a permis à cette architecture de faire ses preuves**
- 1 **La dissipation thermique due au cadencement toujours plus haut marque un nouveau coup d'arrêt**
- 1 **La fin de l'architecture NetBurst est programmée**

Pentium 4

- 1 Clock Speed 3.06GHz
- 1 **Hyperthreading Technology for increased performance in Multi-tasking and Multi-threaded applications**
- 1 .13 micron manufacturing process
- 1 512K on chip, Full Speed L2 Cache
- 1 128bit Floating Point/Multimedia unit
- 1 "Hyper Pipelined" Technology for extremely high clock speeds
- 1 Intel "NetBurst" micro-architecture
- 1 Intel MMX media enhancement technology
- 1 Memory cacheability up to 4 GB of addressable memory space and system memory scalability up to 64 GB of physical memory



Hyperthreading

- 1 **Simulation de deux processeurs sur une seule et même puce**
 - ∨ Duplication de l'état de l'architecture sur chaque processeur
 - ∨ 1 seul ensemble de ressources partagé pour l'exécution
- 1 **Premier processeur, le Pentium 4 d'Intel**



Hyperthreading

- 1 Du point de vue de l'OS ou des utilisateurs le système est vu comme plusieurs processeurs physiques
- 1 Chaque processeur logique possède:
 - ∨ des registres généraux,
 - ∨ des registres de contrôle,
 - ∨ quelques registres d'état,
 - ∨ son contrôleur d'interruption (APIC)
- 1 Les processeurs logiques partagent:
 - ∨ les caches,
 - ∨ les unités d'exécution,
 - ∨ les prédicteurs de branches,
 - ∨ les bus

© Theoris 2004

77

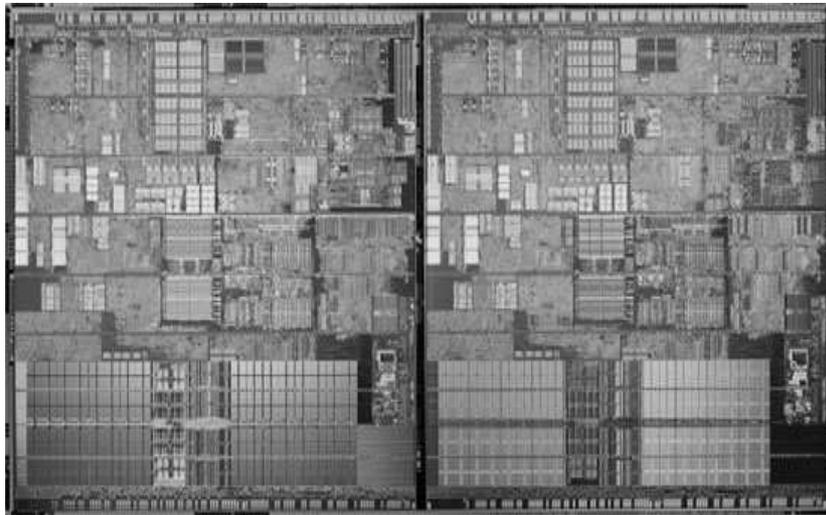
Dual Core

- 1 Solution « bâtarde » d'Intel pour sortir de l'impasse des problèmes de dissipation thermique
- 1 Couplage de deux cores *Prescott*:
 - ∨ 1 Mo de cache
 - ∨ 2.8 à 3.2 GHz
 - ∨ 130 Watts
- 1 Version Extrême Edition avec Hyperthreading
- 1 Contrairement à AMD, Intel à une solution purement « copier/coller » qui fait du Dual Core un système multi-processeur standard
- 1 AMD a prévu un bus de communication entre les deux cores *Opteron*: le SRI (System Request Interface)

© Theoris 2004

78

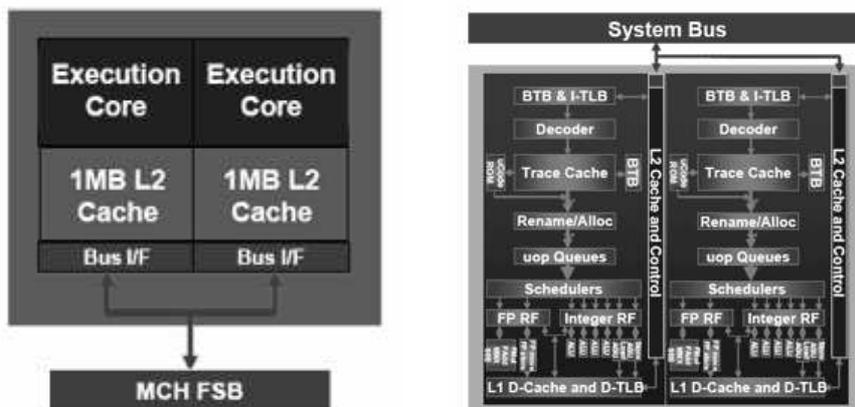
Dual Core: *Smithfield*



© Theoris 2004

79

Dual Core: *Smithfield*

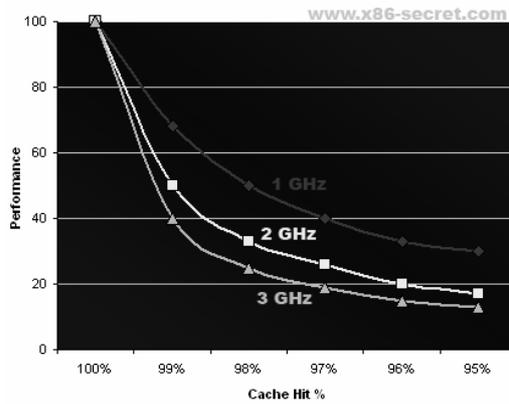


© Theoris 2004

80

Evolution

- 1 Plus la fréquence augmente plus de taux de succès du cache est important
- 1 Les mécanismes de prédiction de branches sont à leur limite (98-99%)

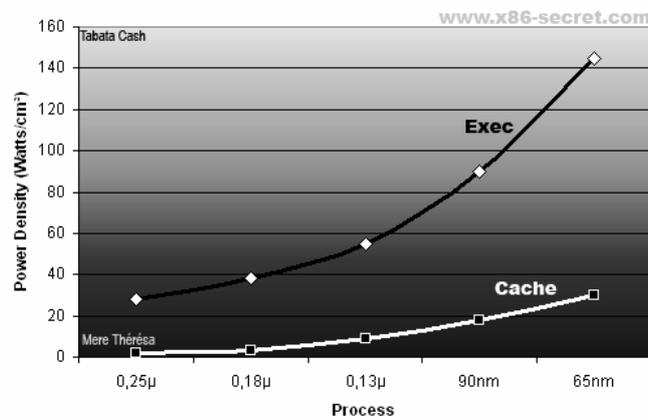


© Theoris 2004

81

Evolution

- 1 Augmentation du cache n'a pas d'incidence majeure sur la consommation

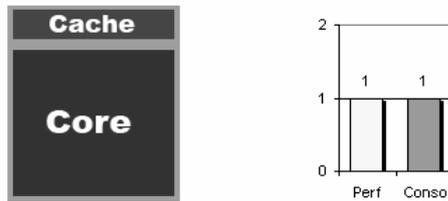


© Theoris 2004

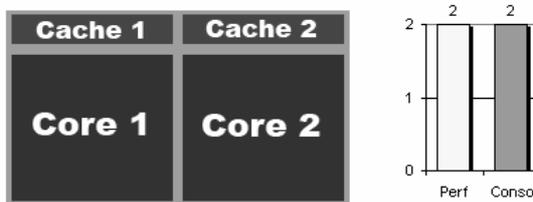
82

Evolution

Pentium 4 (Single Core)



Pentium D (Dual Core)

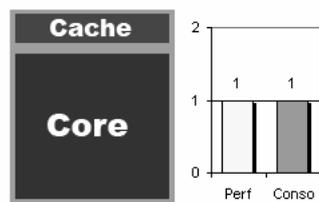


© Theoris 2004

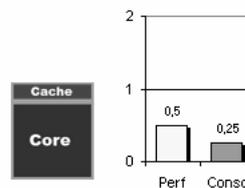
83

Evolution

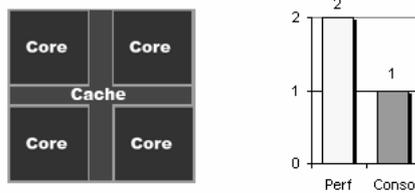
CPU Pentium X



1/4 CPU Pentium X



(1/4 CPU Pentium X) * 4



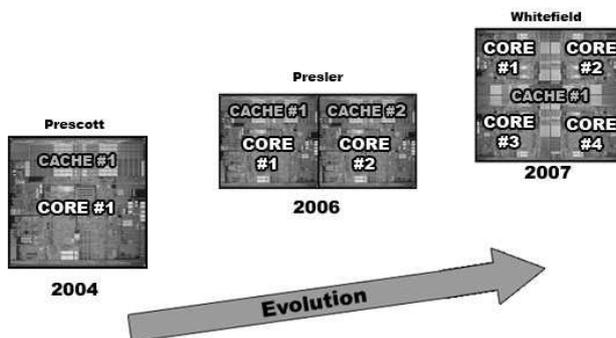
© Theoris 2004

84

Evolution



- 1 Unification du cache
- 1 Evolution MultiCore limitée: une application ne peut se scinder en un nombre infini de threads
- 1 Dernières architectures symétriques ?

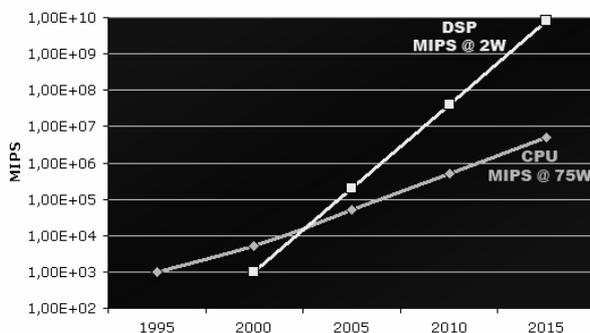


© Theoris 2004

85

Evolution

- 1 DSP = Digital Signal Processor
- 1 Un DSP est optimisé pour effectuer du traitement numérique de signal (FFT, convolution, filtrage, ...)
- 1 Evolution des DSP est très prometteuse

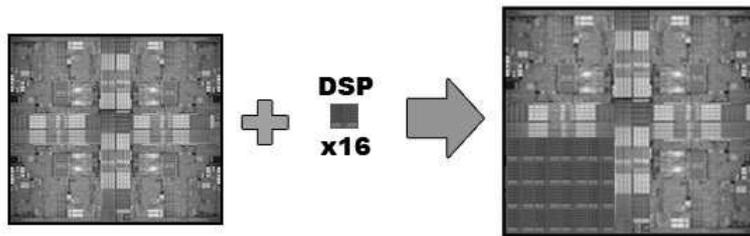


© Theoris 2004

86

Evolution

- 1 DSP peuvent traiter les tâches nécessitant des longs pipelines ou des fréquences élevées
- 1 Taille et consommation faible



© Theoris 2004

87

Processeur matriciel/vectorel

- 1 **ILLIAC IV: université de l'Illinois dans les années 70**
 - v 4 quadrants de 8x8 processeurs/memoires
 - v 1 UC / quadrant pour le scheduling
 - v 1 seul quadrant fut construit
 - v 50 MFLOPS
- 1 **Effectuent des intructions simples sur des tableaux de données**
- 1 **Architectures:**
 - v MIMD: Multiple Instruction Multiple Data
 - v SIMD: Single Instruction Multiple Data
- 1 **Destinés à des applications spécifiques**
- 1 **Souvent associés comme co-processeur à un processeur conventionnel**

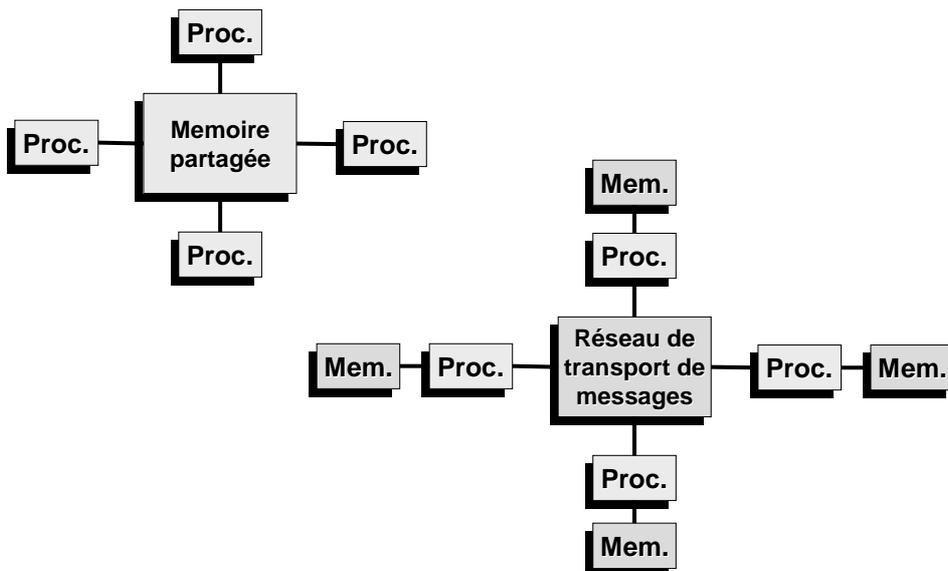
© Theoris 2004

88

Processeur matriciel/vectorel

- 1 **MMX (Multi Media eXtension):**
 - v intégrée au Pentium MMX
 - v registres 64 bits
 - v entiers
- 1 **SSE (Streaming Simd Extension):**
 - v Pentium III
 - v 128 bits
 - v entiers ou flottants 32 bits
- 1 **SSE2:**
 - v évolution SSE sur Pentium 4
 - v flottants 64 bits
- 1 **3D Now !**
 - v version MMX / SSE des processeurs AMD
 - v A partir du K6-II
 - v 64 bits

Multiprocesseur Vs Multiordinateur



Multiprocesseur

- 1 **Système à mémoire partagée**
- 1 **Facilité de communication**
- 1 **Exemple:**
 - ∨ **Enterprise de SUN**
 - ∨ **NUMA-Q de Sequent (IBM)**
 - ∨ **Origin 3000 de SGI**
 - ∨ **Convex Exemplar d'HP**

Multiordinateur

- 1 **Système à mémoire distribué**
- 1 **Chaque UC dispose de sa propre mémoire**
- 1 **Nécessité d'un réseau d'interconnexion pour la communication**
- 1 **Exemple:**
 - ∨ **SP/2 d'IBM**
 - ∨ **COW Wisconsin à base de SPARCstations**

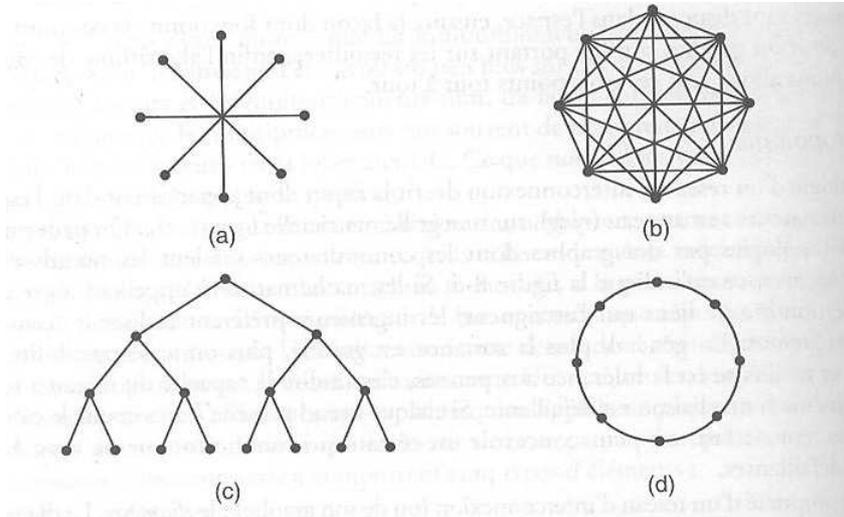
Réseaux d'interconnexion

- 1 **Topologie**
- 1 **Commutation**
- 1 **Gestion des conflits sur les ressources**
- 1 **Algorithme de routage**

Topologie

- 1 **Disposition dans l'espace des commutateurs**
- 1 **Modélisation par graphe**
 - ∨ **Degré d'un nœud ou sortance = nombre de liens qui atteignent un nœud**
 - ∨ **Diamètre = distance entre les deux nœuds les plus éloignés**
 - ∨ **Bandwith = la pire capacité de transmission du réseau**
 - ∨ **Dimension = nombre de choix possible pour un chemin**

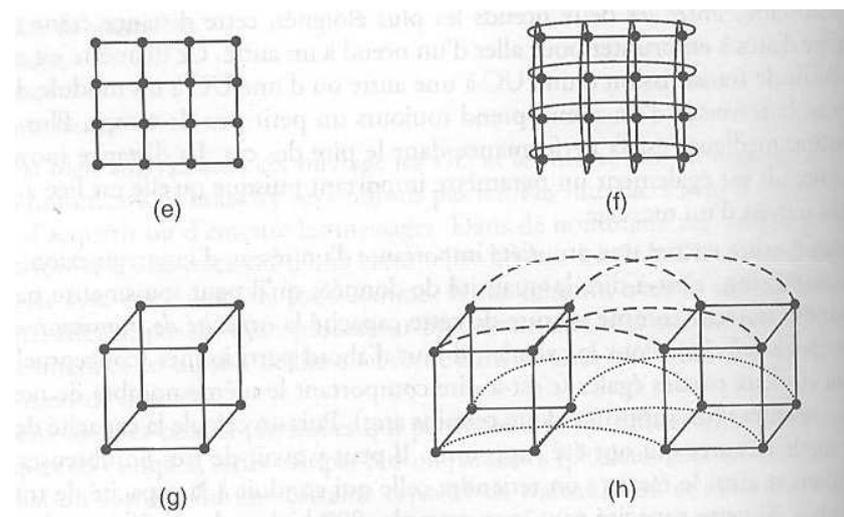
Topologie



© Theoris 2004

95

Topologie



© Theoris 2004

96

Commutation

- 1 **Commutation de circuits**
 - ∨ **Le chemin source-destination est entièrement réservé avant émission**
 - ∨ **Déclaration préalable**
 - ∨ **Aucun trafic étranger**
 - ∨ **Pleine vitesse de transfert**
- 1 **Commutation par paquets**
 - ∨ **Souple et efficace**
 - ∨ **Augmente le temps de latence**

Routage

- 1 **Risque d'étreinte fatale ou dead-lock**
- 1 **Routage par la source: le chemin est déterminé initialement par la source**
- 1 **Routage distribué: chaque commutateur décide**
 - ∨ **Routage statisque**
 - ∨ **Routage adaptatif**
- 1 **Routage dimensionnel**
 - ∨ **Adapté aux grilles matricielles**
 - ∨ **X puis Y par exemple**

Multiprocesseur à mémoire partagée Accès à la mémoire

- 1 **Cohérence stricte**
 - v Simple: FIFO sur la mémoire
 - v Utopiste: Goulot d'étranglement
- 1 **Cohérence séquentielle**
 - v Toute les UC voit les actions sur la mémoire dans le même ordre
- 1 **Cohérence du processeur**
 - v Toutes les UC voit les actions des autres UC dans le même ordre
- 1 **Cohérence faible**
 - v Notion de point de synchronisation
 - v Vidage du pipeline des écritures mémoires pour garantir un état stable
- 1 **Cohérence par libération**
 - v Amélioration du modèle de cohérence faible
 - v Basé sur le principe de section critique

Multiprocesseur à mémoire partagée Architectures

- 1 **SMP sur bus UMA**
- 1 **Multiprocesseur UMA avec commutateur crossbar**
- 1 **Multiprocesseur UMA avec réseaux de commutation multi-étages**
- 1 **Multiprocesseur NUMA**
- 1 **Multiprocesseur NUMA à cohérence de caches**
- 1 **Multiprocesseur COMA**

Multiordinateurs à transfert de messages MPP (Processeurs Massivement Parallèles)

- 1 « Descendants » des mainframes des années 60
- 1 **UC standard:**
 - ∨ Pentium d'Intel
 - ∨ UltraSPARC de Sun
 - ∨ Alpha de DEC
- 1 **Caractérisés par leur réseau de communication dédié qui garanti une faible latence pour un haut débit**
- 1 **Exemple:**
 - ∨ T3E de Cray (Alpha 21164 de DEC, 1 To de mémoire adressable, 2048 nœuds, Tore 3D bidirectionnel + 1 GigaAnneau)
 - ∨ Option Red d'Intel/Sandia (4608 nœuds, Pentium Pro 200Mhz, sur commande)
- 1 **Tolérance aux pannes nécessaire**

Multiordinateurs à transfert de messages COW (grappes de stations de travail)

- 1 **Différence avec MPP: le réseau de communication**
- 1 **La disponibilité de tous les composants sur le marché:**
 - ∨ Grande production
 - ∨ Economie d'échelle
- 1 **Les MPP sont relégués dans les marchés de niche**
- 1 **Ordonnancement**
 - ∨ FIFO
 - ∨ Sans blocage en tête de file
 - ∨ Carrelage: avec profil UC-Temps d'utilisation

Multiordinateurs à transfert de messages Mémoire partagée au niveau applicatif

- ┆ **Mémoire partagée répartie**
 - ┆ DSM (Distributed Shared Memory)
 - ┆ Basé sur la pagination, les pages sont réparties sur les différentes UC
 - ┆ Problème de défaut de page
- ┆ **LINDA**
 - ┆ Mémoire répartie structurée (tuplets)
 - ┆ Enrichissement des langages par des primitives dédiées: C-Linda, FORTRAN-Linda
 - ┆ Les mécanismes de synchro se font naturellement par échange de tuplets
- ┆ **ORCA**
 - ┆ Travaille sur des objets partagés qui se présente comme des régions critiques qui garantissent l'intégrité de l'objet
 - ┆ Chaque UC agit sur un objet via ses méthodes
- ┆ **GLOBE: extension aux réseaux multiordinateurs planétaires**

Yannick BALERE

yannick.balere@theoris.fr