
Architecture des ordinateurs Introduction

Daniel Etiemble
de@lri.fr

Les grandes classes de système

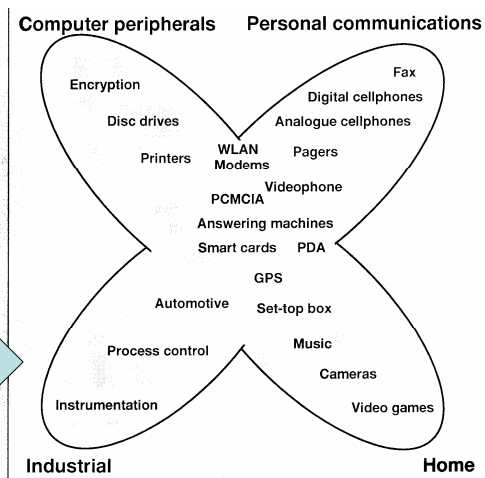
Caractéristique	Ordinateur de bureau	Serveur	Enfoui/embarqué
Prix du microprocesseur	100 à 1000 €	200 à 2000 € par processeur	0,20 à 200 € par processeur
Microprocesseurs vendus en 2000	150 millions	4 millions	300 millions (en ne comptant que les 32 et 64 bits)
Critères	Prix-performance Performance graphique	Débit, disponibilité, extensibilité	Prix, puissance dissipée, performance pour l'application

Ventes des microprocesseurs (fin du siècle dernier ☺)

- Processeurs enfouis/embarqués
 - 4 bits : 2 milliards
 - 8 bits : 4,7 milliards
 - 16 bits : 700 millions
 - 32 bits : 400 millions
- DSP (traitement du signal)
 - 600 millions
- Généralistes classiques
 - 150 millions

Les applications

- Usage général
- Calcul Scientifique
- **GRAPHIQUE**
- Traitement du signal
- **JAVA**
- BD
- WEB
- Enfoui et embarqué



PERFORMANCE

$$T_{\text{exécution}} = \text{NI} * \text{CPI} * T_c = \frac{\text{NI}}{\text{IPC} * F}$$

IPC * F Temps de cycle

Nombre de cycles/Instruction

- Nombre d'instructions
 - Jeu d'instructions et compilateur
- CPI
 - Microarchitecture
- T_c
 - Technologie CMOS et Microarchitecture

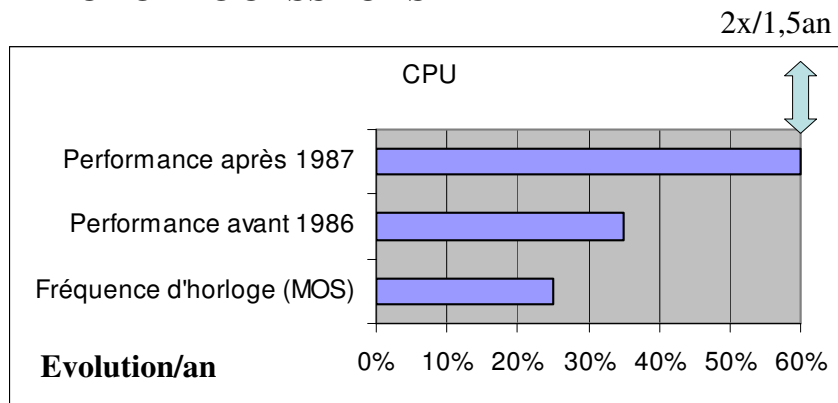
IFIPS2
2007-08

Architecture des ordinateurs
D. Etiemble

5

DES EXPONENTIELLES

MICROPROCESSEURS



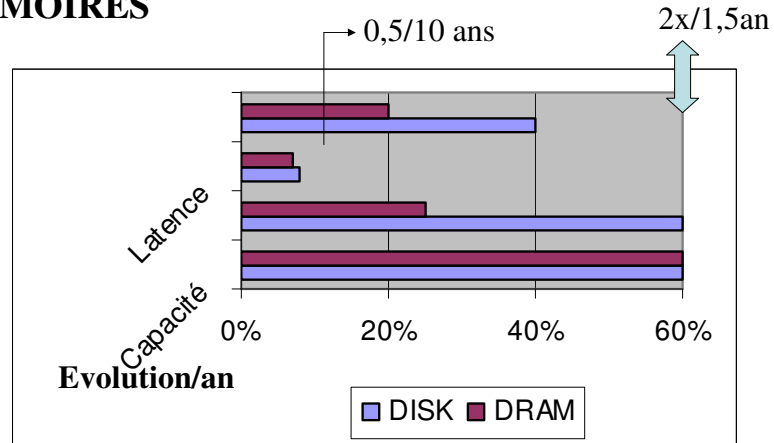
IFIPS2
2007-08

Architecture des ordinateurs
D. Etiemble

6

DES EXPONENTIELLES

MEMOIRES



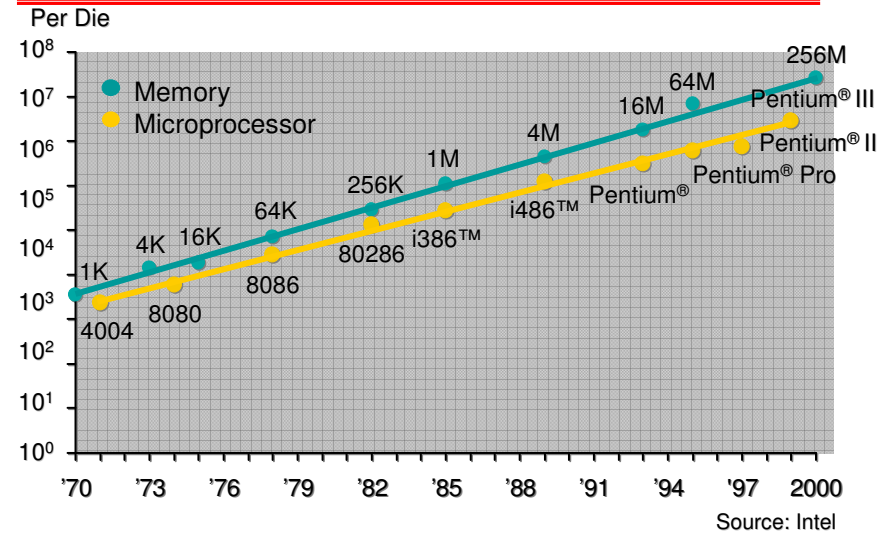
IFIPS2
2007-08

Architecture des ordinateurs
D. Etiemble

7

Moore's Law

Transistors

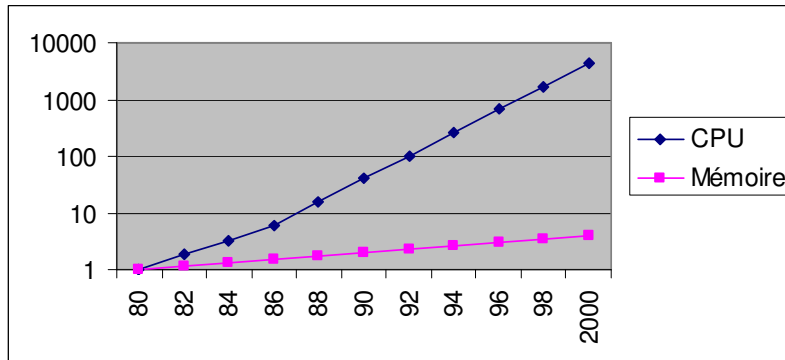


IFIPS2
2007-08

Architecture des ordinateurs
D. Etiemble

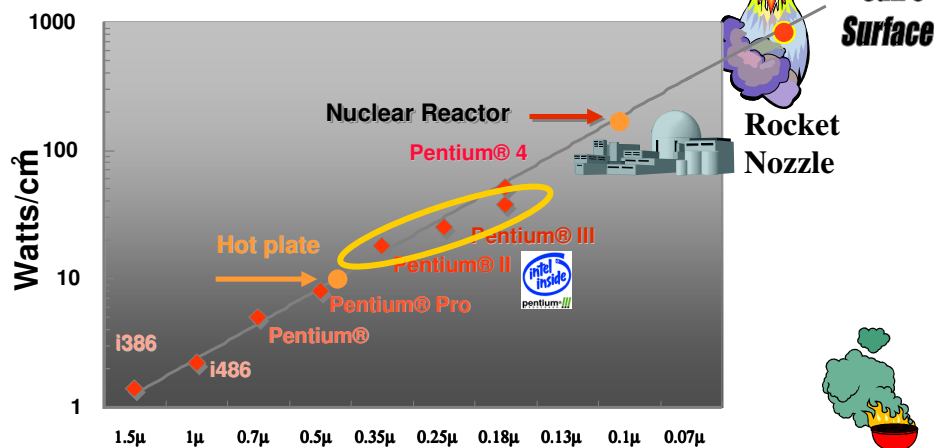
8

LES DIFFERENTIELS



Complexité croissante de la hiérarchie mémoire

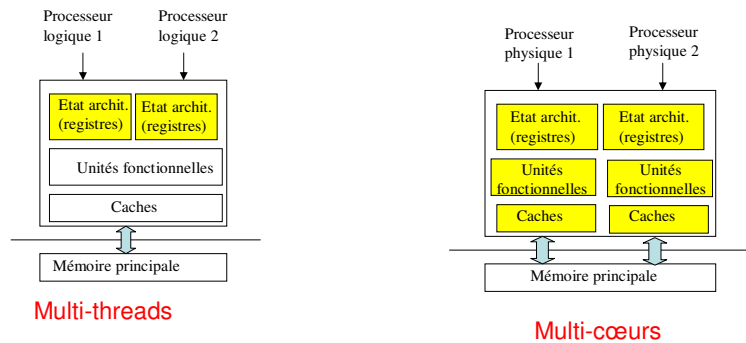
Densité de puissance



* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

Le grand virage...

- Evolution des processeurs pour PC (Intel, AMD)
 - De l'augmentation de la fréquence d'horloge...
 - Au parallélisme



IFIPS2
2007-08

Architecture des ordinateurs
D. Etiemble

11

Jeux d'instructions

- Des objectifs différents selon les classes d'applications
 - Vitesse maximale (PC, serveurs)
 - Taille de code minimale (embarqué)
 - Consommation
 - essentiel pour embarqué
 - important pour tous
- Taille des instructions
 - Fixe
 - Variable
- Modèles d'exécution

IFIPS2
2007-08

Architecture des ordinateurs
D. Etiemble

12

Les objectifs

- Performance
 - Pipeline efficace
 - Instructions de longueur fixe
 - Décodage simple
 - Modes d'adressage simples
- Taille du code
 - Minimiser la taille des instructions
 - Instructions de longueur variable (ou fixe)
 - Accès aux données efficace
 - Modes d'adressage complexes et efficaces pour applications visées
- Compatibilité binaire avec les générations précédentes
 - Exemple IA-32 (x86)

Modèles d'exécution

- Modèles d'exécution (n,m)
 - n : nombre d'opérandes par instruction
 - m : nombre d'opérandes mémoire par instruction
- Les différents modes
 - RISC : (3,0)
 - Instructions de longueur fixe
 - Load et Store : seules instructions mémoire
 - IA-32 : (2,1)
 - Pile (0,0)
 - Tous les opérandes sont accédés via la pile

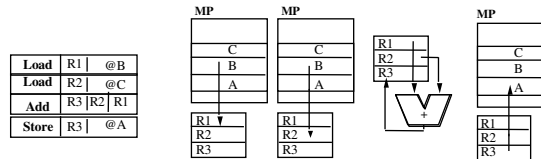
Modèle d'exécution RISC

(n,m)

n : nombre d'opérandes par instruction
 m : nombre d'opérandes mémoire par instruction

Ex : A := B + C

LOAD-STORE (3,0)



Instructions de longueur fixe
 Seules les instructions Load et Store accèdent à la mémoire

Registres: organisation RISC

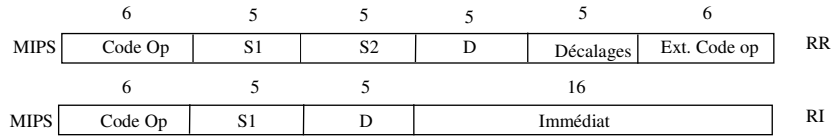
- 32 registres généraux (entiers) R0 à R31
- 32 registres flottants
- Instructions UAL et mémoire
 - Registre – registre
 - $R_d \leftarrow R_{s1} \text{ op } R_{s2}$
 - Registre – immédiat
 - $R_d \leftarrow R_{s1} \text{ op } \text{immédiat}$
 - $R_d \leftrightarrow \text{Mémoire } (R_{s1} + \text{dépl.})$

General Purpose Registers		
R0	alwayszero	local var A
R1		local var B
R2		local var C
R3		local var D
R4		local var E
R5		
R6		
R7		
R8	compiler temp 1	
R9	compiler temp 2	
R10	compiler temp 3	
R11	compiler temp 4	
R12		
R13		stack pointer
R14		
R15		

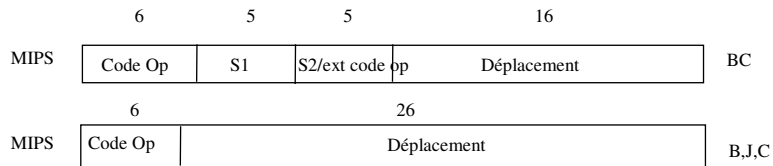
Typical RISC Processor

Formats d'instructions RISC

- Instructions UAL et Mémoire



- Sauts et branchements



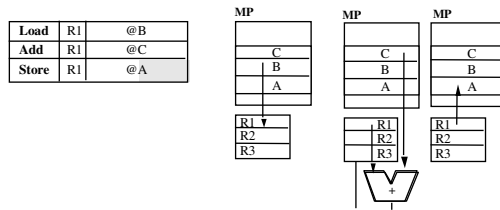
Modèle (2,1)

(n,m)

n : nombre d'opérandes par instruction
m : nombre d'opérandes mémoire par instruction

Ex : A := B + C

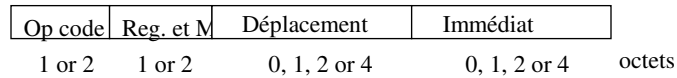
REGISTRE-MEMOIRE (2,1)



CISC compatible avec la technologie MOS des années 75-80

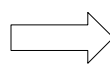
Caractéristiques IA-32

- Instructions de longueur variable



- Inst dest, source

REG	REG
REG	MEM
REG	IMM
MEM	REG
MEM	IMM



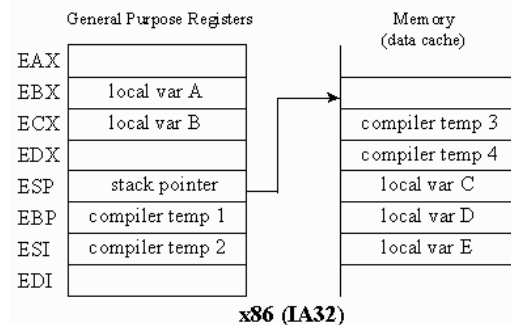
Lecture mémoire,
 Exécution,
 Ecriture mémoire

- Instructions complexes
 - Rep
- Modes d'adressage complexes

$$\text{Adresse mémoire} = \text{Rb} + \text{RI} \times \text{f} + \text{déplacement}$$

Registres : organisation IA-32

- Organisation non homogène
 - 8 registres «généraux» avec rôle spécifique
 - Registres flottants fonctionnant en pile (x87)
 - Registres «SIMD» MMX, SSE-SSE2-SSE3)



Le débat RISC-CISC pour les PC

- Définition
 - RISC : modèle (3,0)
 - CISC : tous les autres
- RISC et pipeline
 - Les jeux d'instructions RISC facilitent la réalisation de pipelines performants
- « Solution » Intel et AMD pour IA-32
 - Convertir les instructions CISC en instructions RISC lors du décodage des instructions (conversion dynamique)
 - On conserve la compatibilité binaire
 - On a l'efficacité des pipelines « RISC »

Traduction des instructions x86

Pentium Pro, PII, PIII, P4

Instructions x86	Opérations RISC
add EAX, [EBP +d8]	load temp, [EBP + d8] add EAX, temp
add [EBP +d8], EAX	load temp, [EBP + d8] add EAX, temp store EAX, [EBP+8]
cmp EAX, imm32	cmp EAX, imm32
push ECX	sub ESP, 4 store [ESP], ECX

Jeux d'instructions et applications

- Instructions SIMD
- Instructions pour traitement du signal (processeur DSP)
 - Multiplication - accumulation
 - Produit scalaire
 - Modes d'adressage complexes pour simplifier l'accès aux données
 - Modes d'adressage spécifiques au traitement du signal
 - Mode d'adressage circulaire
 - Mode d'adressage « bits inversées » pour la FFT