MI01
Performances

Optimisation et parallélisme

- Serveurs WEB
  - Plusieurs millions de sessions simultanées
- Routeurs Large Bande Passante
  - Points d'intersection importants
- Applications à calculs intenses
  - Météo, modélisation nucléaire, etc.

obsession des performances

- Texe = N * CPI * 1/F

- Texe : Temps d'exécution d'un programme donné sur un processeur donné
- N : nombre d'instructions dans ce programme
- CPI : (cycles per instruction) nombre de cycles horloge nécessaire par instruction
- F : fréquence d'horloge

Performances, comment les calculer ?

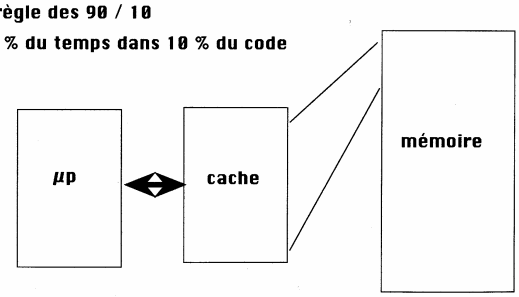
- Texe = N * CPI * 1/F
- Objectif : exécuter plus vite, donc minimiser Texe
- Donc minimiser chaque composante du produit
- Minimiser N => optimiser les instructions (le noyau du processeur)
- Minimiser CPI => utiliser un cache
- Minimiser 1/F => augmenter F (fréquence d'horloge)

Optimisations diverses

mémoire Cache

Le µp est beaucoup plus rapide que la mémoire
"wait state" : cycles d'attente

La règle des 90 / 10
90 % du temps dans 10 % du code



- Texe = N * CPI * 1/F
- Objectif : exécuter plus vite, donc minimiser Texe
- Donc minimiser chaque composante du produit
- *Minimiser N => optimiser les instructions (le noyau du processeur)*
- Minimiser CPI => utiliser un cache
- *Minimiser 1/F => augmenter F (fréquence d'horloge)*

Optimisations diverses

Cache Memory

Adapté des “lectures notes” de Rabi Mahapatra & Hank Walker, Dr. Patterson and Dr. Kubiatowicz of UC Berkeley

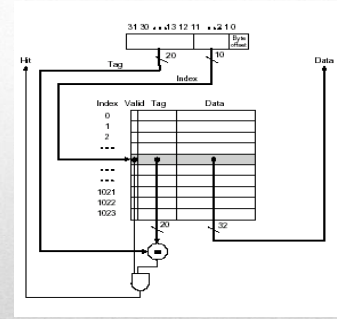
Revisiting Memory Hierarchy

- Facts
  - Big is slow
  - Fast is small
- Increase performance by having “hierarchy” of memory subsystems
- “Temporal Locality” and “Spatial Locality” are big ideas

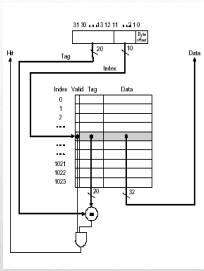
Revisiting Memory Hierarchy

- Terms
  - Cache Miss
  - Cache Hit
  - Hit Rate
  - Miss Rate
  - Index, Offset and Tag

Direct Mapped Cache



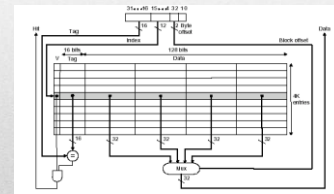
Direct Mapped Cache [contd...]



- What is the size of cache ?
  4K
- If I read
  0000 0000 0000 0000 0000 0000 1000 0001
- What is the index number checked ?
  64
- If the number was found, what are the inputs to comparator ?

Direct Mapped Cache [contd...]

Taking advantage of spatial locality, we read 4 bytes at a time

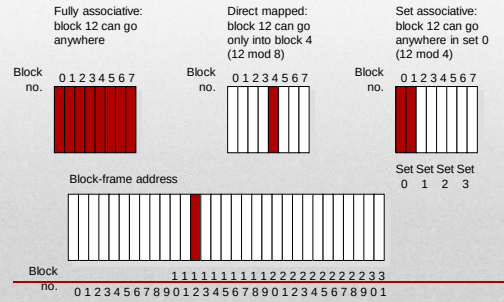


- Advantage
  - Simple
  - Fast
- Disadvantage
  - Mapping is fixed !!!

Direct Mapped Cache [contd...]

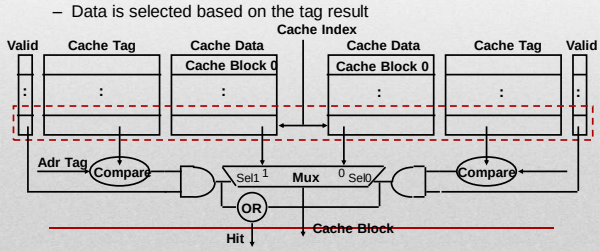
Associative Caches

- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Modulo Number Sets

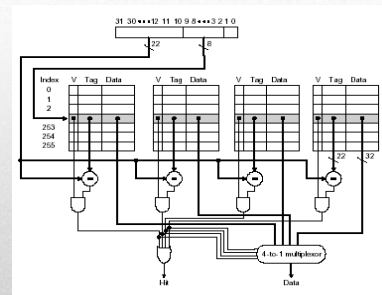


Set Associative Cache

- N-way set associative: N entries for each Cache Index
  - N direct mapped caches operates in parallel
- Example: Two-way set associative cache
  - Cache Index selects a "set" from the cache
  - The two tags in the set are compared to the input in parallel
  - Data is selected based on the tag result



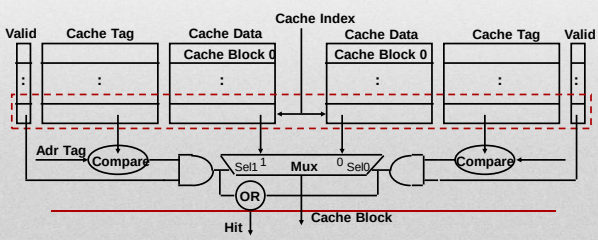
Example: 4-way set associative Cache



What is the cache size in this case ?

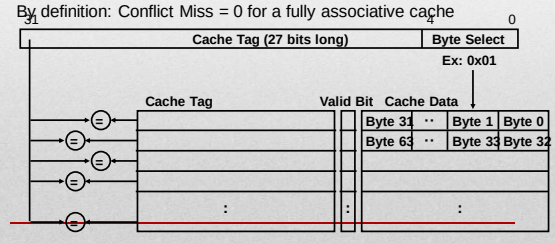
Disadvantages of Set Associative Cache

- N-way Set Associative Cache versus Direct Mapped Cache:
  - N comparators vs. 1
  - Extra MUX delay for the data
  - Data comes AFTER Hit/Miss decision and set selection
- In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:



Fully Associative Cache

- Fully Associative Cache
  - Forget about the Cache Index
  - Compare the Cache Tags of all cache entries in parallel
  - Example: Block Size = 32 B blocks, we need N 27-bit comparators
- By definition: Conflict Miss = 0 for a fully associative cache



Cache Misses

- Compulsory (cold start or process migration, first reference): first access to a block
  - "Cold" fact of life: not a whole lot you can do about it
  - Note: If you are going to run "billions" of instruction, Compulsory Misses are insignificant
- Capacity:
  - Cache cannot contain all blocks access by the program
  - Solution: increase cache size
- Conflict (collision):
  - Multiple memory locations mapped to the same cache location
  - Solution 1: increase cache size
  - Solution 2: increase associativity
- Coherence (Invalidation): other process (e.g., I/O) updates memory

Design Options at Constant Cost

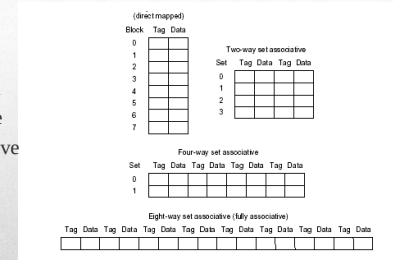
	Direct Mapped	N-way Set Associative	Fully Associative
Cache Size	Big	Medium	Small
Compulsory Miss	Same	Same	Same
Conflict Miss	High	Medium	Zero
Capacity Miss	Low	Medium	High
Coherence Miss	Same	Same	Same

Four Questions for Cache Design

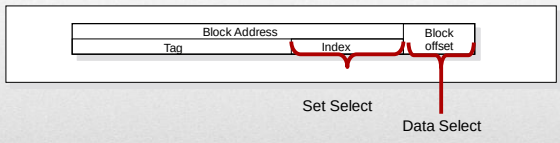
- Q1: Where can a block be placed in the upper level? *(Block placement)*
- Q2: How is a block found if it is in the upper level? *(Block identification)*
- Q3: Which block should be replaced on a miss? *(Block replacement)*
- Q4: What happens on a write? *(Write strategy)*

Where can a block be placed in the upper level?

- Direct Mapped
- Set Associative
- Fully Associative



How is a block found if it is in the upper level?



- Direct indexing (using index and block offset), tag compares, or combination
- Increasing associativity shrinks index, expands tag

Which block should be replaced on a miss?

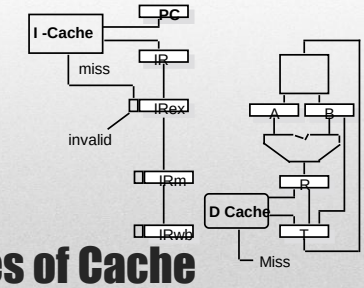
- Easy for Direct Mapped
- Set Associative or Fully Associative:
  - Random
  - LRU (Least Recently Used)

Associativity:	2-way		4-way		8-way	
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

- *Write through*—The information is written to both the block in the cache and to the block in the lower-level memory.
- *Write back*—The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.
  - is block clean or dirty?
- Pros and Cons of each?
  - WT: read misses cannot result in writes
  - WB: no writes of repeated writes
- WT always combined with write buffers so that don't wait for lower level memory

What happens on a write?

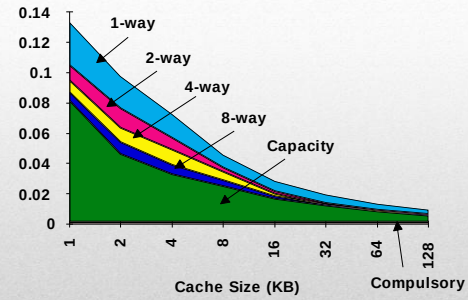
- Instruction Cache
- Data Cache



Two types of Cache

- Reduce Miss Rate
  - Associativity
  - Victim Cache
  - Compiler Optimizations
- Reduce Miss Penalty
  - Faster DRAM
  - Write Buffers
- Reduce Hit Time

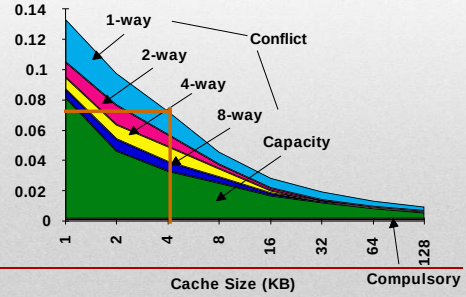
Improving Cache Performance



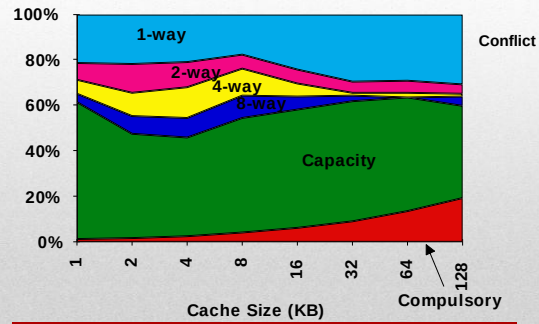
Reduce Miss Rate

Reduce Miss Rate [contd...]

miss rate 1-way associative cache size X
= miss rate 2-way associative cache size X/2



3Cs Comparision



- Compiler Optimizations to reduce miss rate
  - Loop Fusion
  - Loop Interchange
  - Merge Arrays

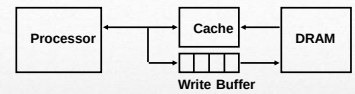
Compiler Optimizations

- Faster RAM memories
  - Driven by technology and cost !!!
  - Eg: CRAY uses only SRAM

Reducing Miss Penalty

- Lower Associativity
  - Add L1 and L2 caches
  - L1 cache is small => Hit Time is critical
  - L2 cache has large => Miss Penalty is critical

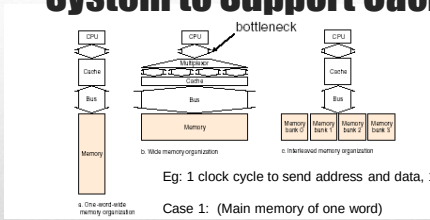
Reduce Hit Time



- A Write Buffer is needed between the Cache and Memory
  - Processor: writes data into the cache and the write buffer
  - Memory controller: write contents of the buffer to memory
- Write buffer is just a FIFO:

Reducing Miss Penalty [contd…]

Designing the Memory System to Support Caches



Eg: 1 clock cycle to send address and data, 15 for each DRAM access

Case 1: (Main memory of one word)
1 + 4 x 15 + 4 x 1 = 65 clk cycles, bytes/clock = 16/65 = 0.25

Case 2: (Main memory of two words)
1 + 2 x 15 + 2 x 1 = 33 clk cycles, bytes/clock = 16/33 = 0.48

Case 3: (Interleaved memory)
1 + 1 x 15 + 4 x 1 = 20 clk cycles, bytes/clock = 16/20 = 0.80

- Read Hit
  - Good for CPU !!!
- Read Miss
  - Stall CPU, fetch, Resume
- Write Hit
  - Write Through
  - Write Back
- Write Miss
  - Write entire block into memory, Read into cache

The possible R/W cases

Performance

- CPU time = (CPU execution clock cycles + Memory stall clock cycles) x clock cycle time
- Memory stall clock cycles = (Reads x Read miss rate x Read miss penalty + Writes x Write miss rate x Write miss penalty)
- Memory stall clock cycles = Memory accesses x Miss rate x Miss penalty
- Different measure: AMAT

  Average Memory Access time (AMAT) = Hit Time + (Miss Rate x Miss Penalty)
- Note: *memory hit time is included in execution cycles.*

Performance [contd...]

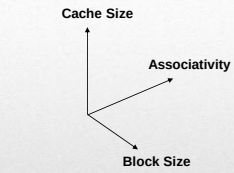
- Suppose a processor executes at
  - Clock Rate = 200 MHz (5 ns per cycle)
  - Base CPI = 1.1
  - 50% arith/logic, 30% ld/st, 20% control
- Suppose that 10% of memory operations get 50 cycle miss penalty
- Suppose that 1% of instructions get same miss penalty
- CPI = Base CPI + average stalls per instruction
  1.1(cycles/ins) +
  [0.30 (DataMops/ins) x 0.10 (miss/DataMop) x 50 (cycle/miss)] +
  [1 (InstMop/ins) x 0.01 (miss/InstMop) x 50 (cycle/miss)]
  = (1.1 + 1.5 + .5) cycle/ins = 3.1
- 58% of the time the proc is stalled waiting for memory!
- AMAT=(1/1.3)x[1+0.01x50]+(0.3/1.3)x[1+0.1x50]=2.54

Summary

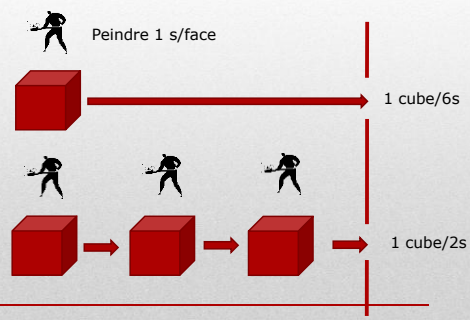
- The Principle of Locality:
  - Program likely to access a relatively small portion of the address space at any instant of time.
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space
- Three (+1) Major Categories of Cache Misses:
  - Compulsory Misses: sad facts of life. Example: cold start misses.
  - Conflict Misses: increase cache size and/or associativity. Nightmare Scenario: ping pong effect!
  - Capacity Misses: increase cache size
  - Coherence Misses: Caused by external processors or I/O devices
- Cache Design Space
  - total size, block size, associativity
  - replacement policy
  - write-hit policy (write-through, write-back)
  - write-miss policy

Summary [contd...]

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
  - write allocation
- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins



Principe du pipeline



	T1	T2	T3	T4	T5	T6
Recherche	I1	I2	I3	I4	I5	I6
Décodage		I1	I2	I3	I4	I5
Exécution			I1	I2	I3	I4

Pipeline processeur

Example: 6 tâches, divisées en 4 segments

1	2	3	4	5	6	7	8	9
T1	T2	T3	T4	T5	T6			
	T1	T2	T3	T4	T5	T6		
		T1	T2	T3	T4	T5	T6	
			T1	T2	T3	T4	T5	T6

Pipeline Performance

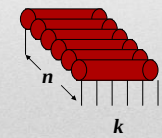
- n: instructions
- k: stages in pipeline
- τ: clockcycle
- T_k: total time

n est équivalent au nombre de cubes
K est le nombre d'étages (=n ici)

Cycle horloge pour la tâche la plus lente

$$T_k = (k + (n-1))\tau$$

$$Speedup = \frac{T_1}{T_k} = \frac{nk}{k + (n-1)}$$



- Considérer un pipeline avec k-segment travaillant sur n data sets.
- > cela prend k cycles d'horloge pour remplir le pipeline et obtenir la première sortie du pipeline
- Ensuite, les n-1 résultats restant arriveront après chaque cycle d'horloge
- Par conséquent, cela prend (k + n - 1) cycles d'horloge clock pour compléter la tâche

SPEEDUP

- Si on exécute la même tâche séquentiellement sur un simple processeur, cela prendra (k * n) cycles d'horloge
- Le speedup obtenu par le pipeline est :
- **S = k * n / (k + n - 1)**

SPEEDUP

- **S = k * n / (k + n - 1)**

Pour n >> k (ex : 1 million data sets sur un pipeline de 3 étages),

- **S ~ k**
- Ainsi, on obtient un speedup équivalent au nombre d'unités fonctionnelles pour un ensemble de données très large, car ces unités sont considérés comme travaillant en parallèle sauf pendant l'amorçage du pipeline ou suite à une rupture

SPEEDUP

Famille x86

- Une dizaine de types de parallélismes existe jusqu'à maitenant
- Exemples : Pentium\parallélismes pour pentium mi01.ppt

Autre approche : Parallélisme !

Craig Barret, (ex PDG d'Intel), s'excusant publiquement de ne pas respecter la promesse de cette compagnie de mettre sur le marché un Pentium 4 fonctionnant à 4Ghz!
La course à la vitesse est arrêtée...

19 octobre 2004 - Orlando

- 2 processeurs côte à côte
  - Adapter le processeur pour une telle connexion
  - Bi processeurs, Quad processeurs
- Hyperthreading
- Double core à l'intérieur d'un même processeur
  - Deux noyaux très liés

Intel : Plusieurs vues du parallélisme

- Hyperthreading
- (http://or1cedar.intel.com/media/training/intro_ht_dt_v1/tutorial/index.htm)

Hyperthreading

- Noyau Prescott
  - Profondeur pipeline = 32 niveaux
  - Couplé à la station de réservation
  - Appellation « Threading »
    - Empruntée des systèmes d'exploitation multi-tâches
      - Ou multi « thread »

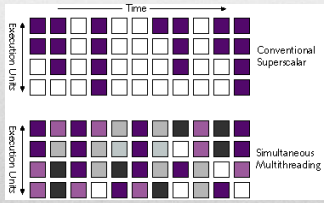
Hyperpipeline et threading

- Pentium version Itanimum (1 & 2)

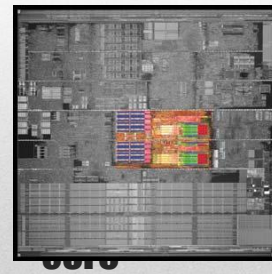
2 processeurs côte à côte

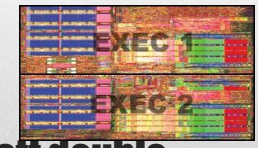
Simultaneous Multithreading (SMT)

- Instruction Level Parallelism
  - Ordonnancement dynamique de “morceaux” d’instructions (multithreading)
  - Jeux de Registres temporaires pour le changement rapide de contexte (changement de tâches) renommables pour chaque thread

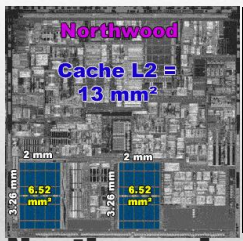


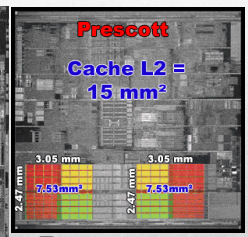
Source: Micrprocessor Report, December 6, 1999 “Compaq Chooses SMT for Alpha”



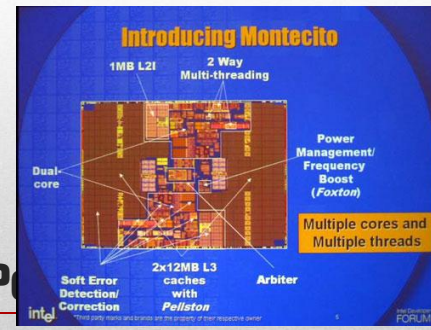


cott double

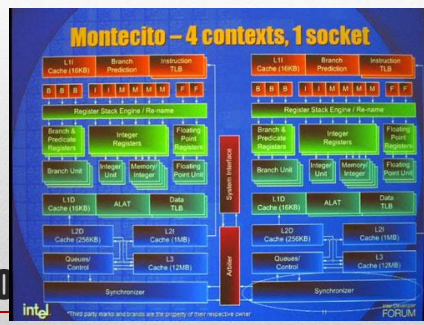




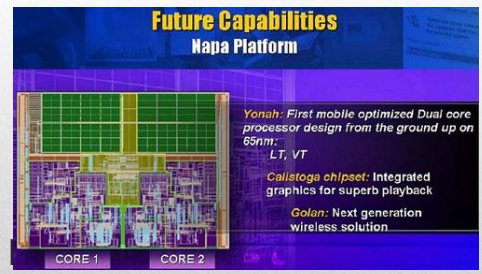
Northwood vs Prescott



P



Mo



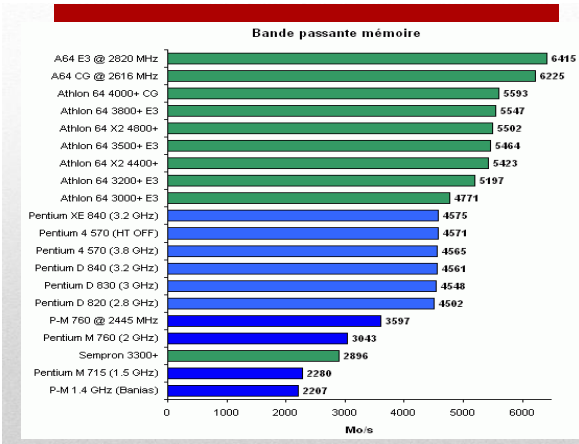
+ GP et Wireless

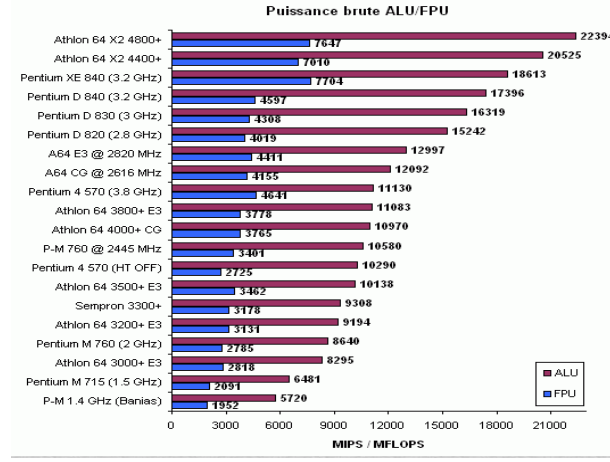
Spécifications des processeurs dual-core Intel

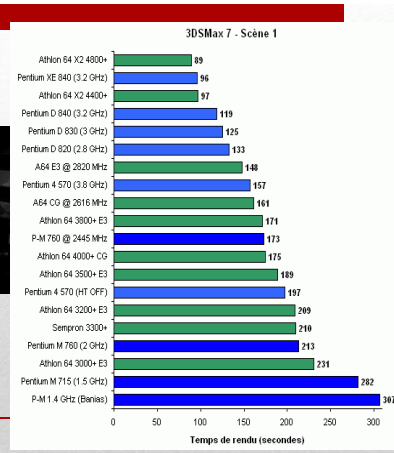
CPU	Pentium D 820	Pentium D 830	Pentium D 840	Pentium XE 840
Fréquence	2800 MHz	3000 MHz	3200 MHz	3200 MHz
Cache L1	2 x 28 Ko	2 x 28 Ko	2 x 28 Ko	2 x 28 Ko
Cache L2	2 x 1024 Ko	2 x 1024 Ko	2 x 1024 Ko	2 x 1024 Ko
FSB	800 MHz	800 MHz	800 MHz	800 MHz
Socket	LGA 775	LGA 775	LGA 775	LGA 775
Voltage	1,25 - 1,39 V	1,25 - 1,39 V	1,25 - 1,39 V	1,25 - 1,39 V
TDP	95 W	130 W	130 W	130 W
Nombre de transistors	230 millions	230 millions	230 millions	230 millions
Process	.09µ strained silicon	.09µ strained silicon	.09µ strained silicon	0.09µ strained silicon
Surface	206 mm²	206 mm²	206 mm²	206 mm²
Support du 64 bits	Oui	Oui	Oui	Oui
Support de l'EIST	Non : TM1 + TM2+ C0	Oui, TM1, TM2, C0	Oui, TM1, TM2, C0	Oui, TM1, TM2, C0
Support de l'HyperThreading	Non	Non	Non	Oui
Prix officiel	241 $	316 $	530 $	999 $

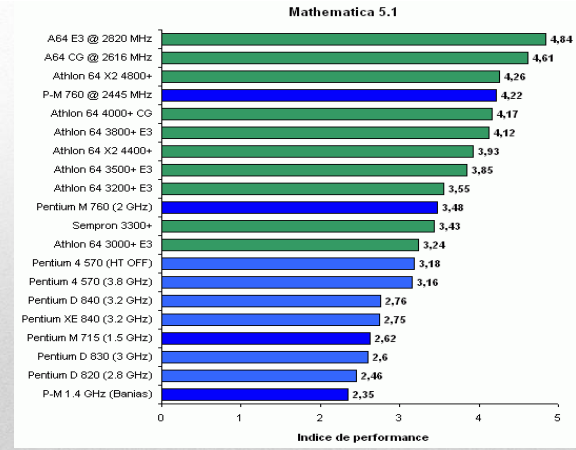
Spécifications des processeurs dual-core AMD

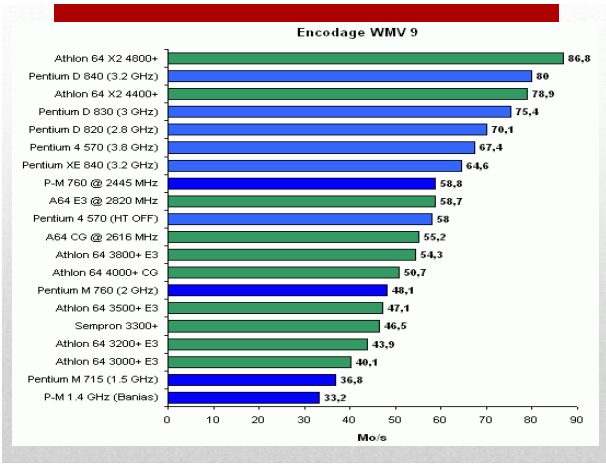
CPU	Athlon 64 X2 4200+	Athlon 64 X2 4400+	Athlon 64 X2 4600+	Athlon 64 X2 4800+
Fréquence	2200 MHz	2200 MHz	2400 MHz	2400 MHz
Cache L1	2 x 128 Ko	2 x 128 Ko	2 x 128 Ko	2 x 128 Ko
Cache L2	2 x 512 Ko	2 x 1024 Ko	2 x 512 Ko	2 x 1024 Ko
FSB	200 MHz	200 MHz	200 MHz	200 MHz
Socket	939	939	939	939
Voltage	1,35 -1,4 V	1,35 -1,4 V	1,35 -1,4 V	1,35 -1,4 V
TDP	110 W	110 W	110 W	110 W
Nombre de transistors	233 millions	233 millions	233 millions	233 millions
Process	.09µ SOI + DSL	.09µ SOI + DSL	.09µ SOI + DSL	.09µ SOI + DSL
Surface	199 mm²	199 mm²	199 mm²	199 mm²
Support du 64 bits	Oui	Oui	Oui	Oui
Support du Cool & Quiet	Oui	Oui	Oui	Oui
Support de l'HyperThreading	Non	Non	Non	Non
Prix officiel	537 $	581 $	803 $	1001 $









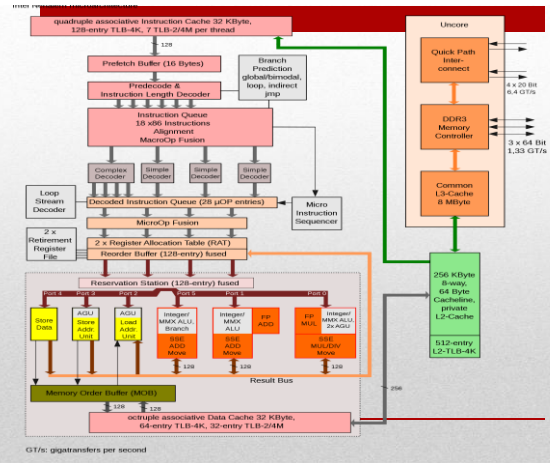


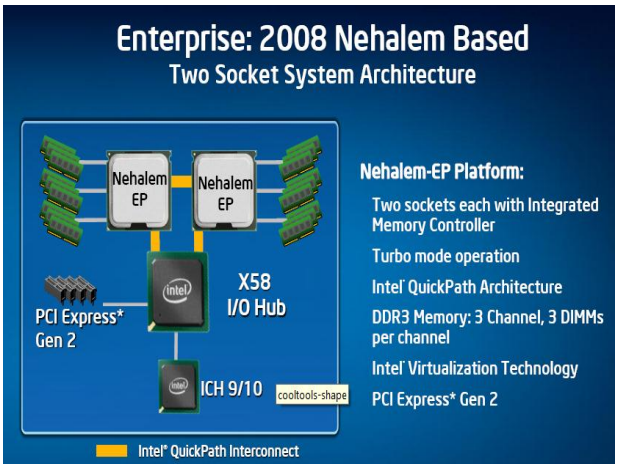
- Cohérence des caches et mémoires
  - Si 2 processeurs (ou plus) ont dans leur cache la copie de la même zone mémoire
  - si un processeur modifie son cache, refléter cette modification dans les caches de tous les autres processeurs
  - Et refléter ce changement dans la mémoire extérieure

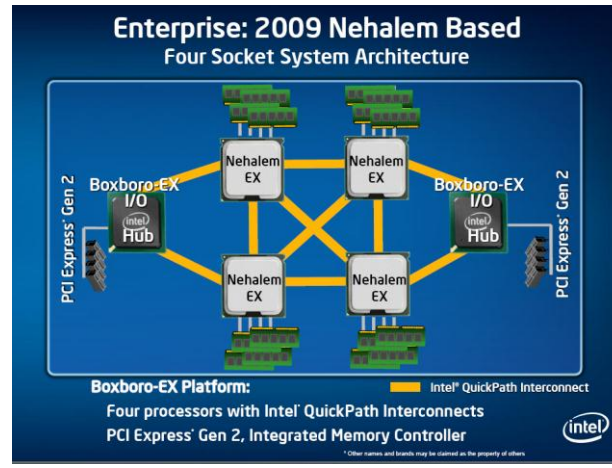
Inconvénients du multi core

Numérotation Δ	Mémoire cache	Fréquence d'horloge	Bus principal	Nb. de cœurs	Technologie Intel® VT*	Intel® 64*	Technologie Intel® TXT	Bit de verrouillage*
45 nm								
i7-950	8 MB	3,066 GHz	QPI 4,8 GT/s	4	✓	✓		✓
i7-940	8 MB	2,933 GHz	QPI 4,8 GT/s	4	✓	✓		✓
i7-920	8 MB	2,666 GHz	QPI 4,8 GT/s	4	✓	✓		✓
i7-870	8 MB	2,93 GHz	DMI 2,5 GT/s	4	✓	✓	✓	✓
i7-860	8 MB	2,8 GHz	DMI 2,5 GT/s	4	✓	✓	✓	✓

Pentium 4 (hyper Threading)

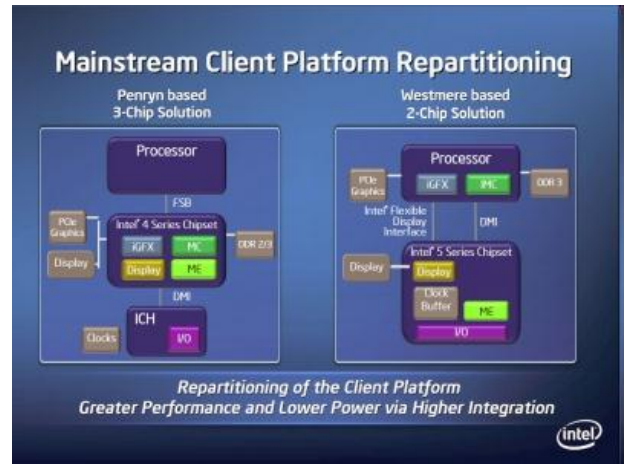


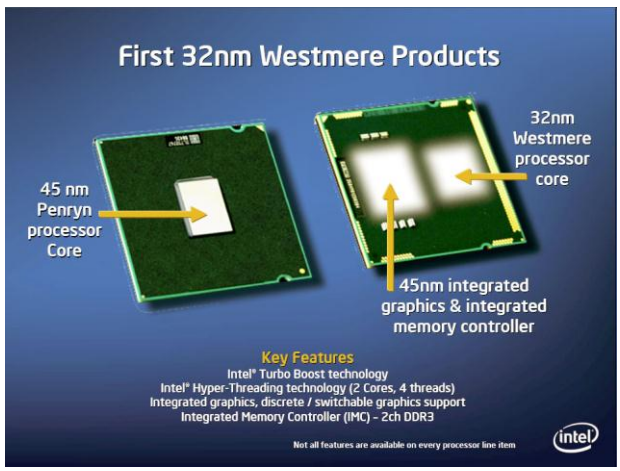




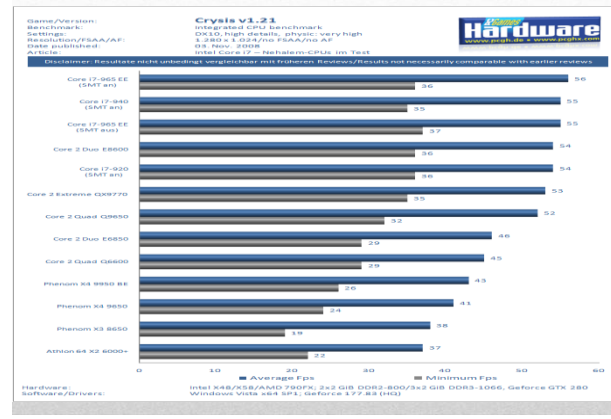
- Nommé Westmere
- Fonctionalité GPU déplacée du pont nord vers CPU

Core : 3 à 2 chip



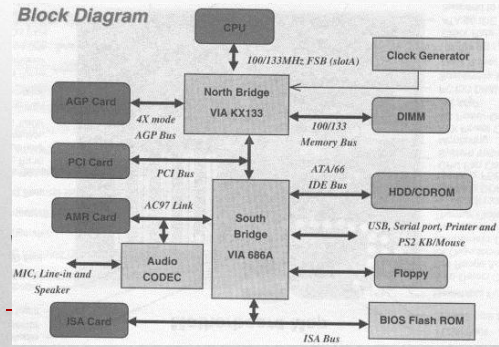


Other Fun Info



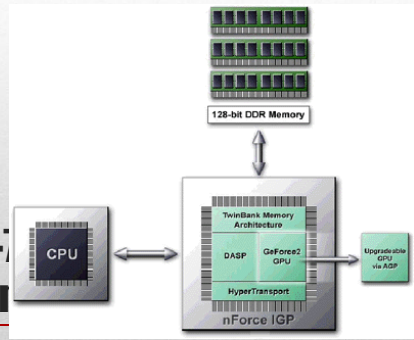
Pentium

Interface Graphique avancée



IGP : Integrated Graphics Processor

AMD-
Duron



Le MCP