



ARCHITETTURE DEI CALCOLATORI

cache - esercitazione

Mariagiovanna Sami



Esercizio 1

- Quanti bit in totale sono necessari per una cache istruzioni di tipo direct mapped con 64 KB di dati e blocchi di una parola, supponendo che gli indirizzi siano lunghi 32-bit?

1 Word=4 byte

Numero di blocchi = $64\text{KB}/4\text{B}=2^{14}$ blocchi

Bit di Tag = $32-14[\text{indice}]-2[\text{spiazzamento}]=16$

dimensione = $[16(\text{tag})+1(\text{validbit})+4(\text{blocksize}) * 8] * 2^{14} = 802816$



Esercizio 2: cache DM di 64blocchi x 32 byte

- Supponendo indirizzamento al **byte** e indirizzi di 32 bit, quanti bit ci sono in ognuno dei campi tag, Index, e Spiazzamento?

Index=6 bit, spiazzamento=5 bit; tag=21bit

- Quanti **bytes** di dati, in totale, si possono memorizzare nella cache?

2KB

- Quanti **bytes** di memoria usa la cache (inclusendo tag, valid bit, e dati)?

$(21+1[\text{valid}]) * 64 / 8 + 32 * 64 = 2224 \text{ Bytes}$

- Quanti sono i blocchi che fanno riferimento allo stesso blocco in cache?

2^{21}

- Se nella cache si caricano blocchi a caso, quale è la probabilità che, dato un indirizzo, si abbia riscontro nel campo tag?

$1 / (2^{21})$



Esercizio 3

- Sia data una cache con:
 - Dimensione della Cache = 128 byte in totale.
 - Blocchi di 2 parole.
 - set associativa a 2-vie.
- Quanti blocchi ha la cache? $[128/(8)]=16$
- Di quanti bit è l'indice? $[3=\log(16/2)]$
- Di quanti bit è il tag? $[32-3-3(\text{offset})=26]$

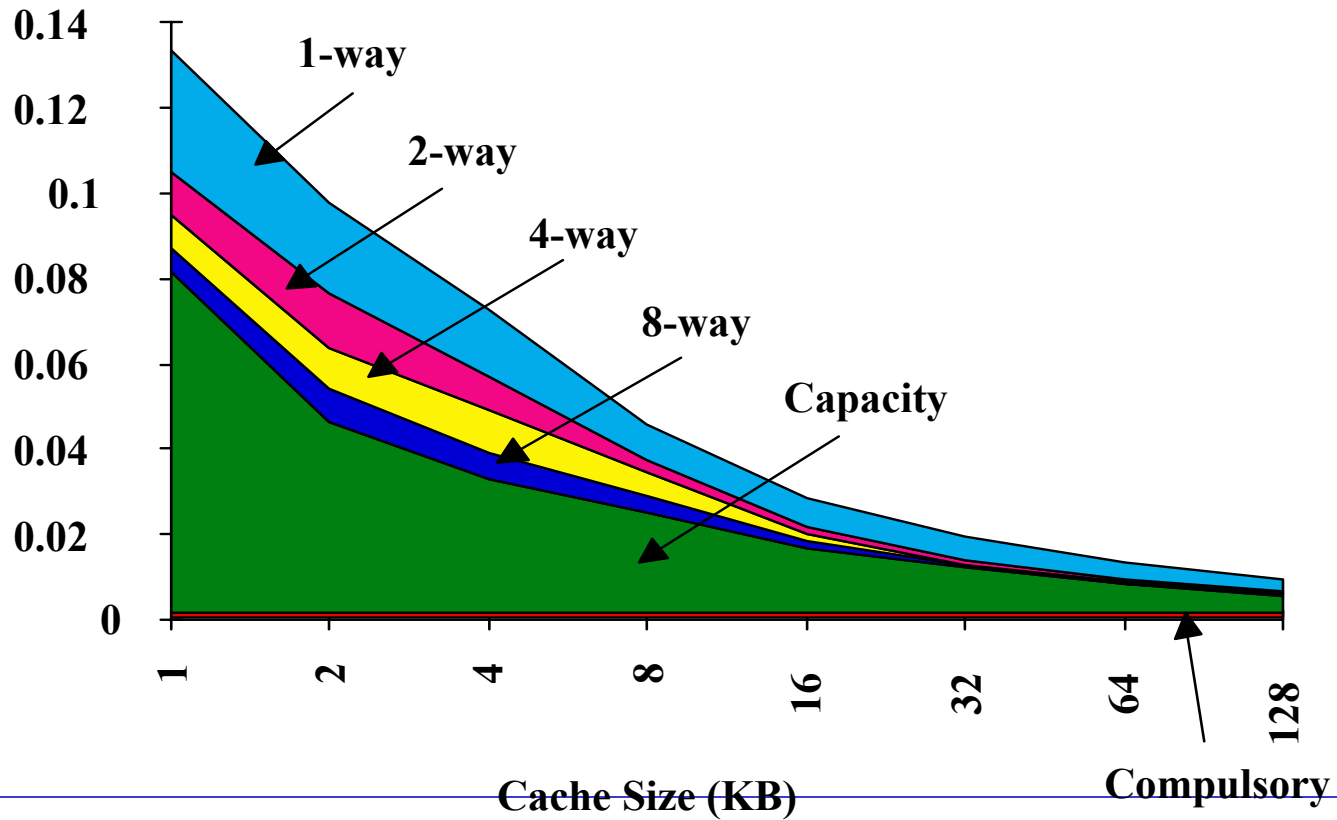


Prestazioni della cache

- $\text{CPUtime} = \text{Instruction Count} \times (\text{CPI}_{\text{execution}} + \text{Mem accesses per instruction} \times \text{Miss rate} \times \text{Miss penalty in cycles}) \times \text{Clock cycle time}$
 - Misses per instruction = #Memory accesses per instruction x Miss rate
 - $\text{CPI} = \text{CPI}_{\text{execution}} + \text{Misses per instruction} \times \text{Miss penalty cycles}$
- $T_A = \text{HitTime} + \text{MissRate} * \text{MissPenalty}$



Miss Rate Assoluti (SPEC92)





Esercizio 4

- Calcolatore di riferimento: VAX-11/780
 - RAM = 6 cicli
 - CPI esecuzione = 8.5
 - MissRate = 0.11
 - #acc_mem/istruzione = 3
- Si calcoli il CPI dell'architettura con la cache

$$\begin{aligned} \text{CPI}_{\text{realCache}} &= \text{CPI}_{\text{exec}} + \# \text{accmem} / \text{instr} * \text{MR} * \text{MP} = \\ &= 8.5 + 3 * 0.11 * 6 = 10.48 \end{aligned}$$



Esercizio 5

- Confrontare la precedente architettura nel caso di miss rate al 100% con la stessa nel caso di hit rate al 100%. Si confronti lo speedup della cache reale con quello della cache ideale.

- 100%miss

$$\text{CPI}_{\text{noCache}} = 8.5 + 3 * 6 = 26.5$$

- 100%Hit:

$$\text{CPI}_{\text{idealCache}} = \text{CPI}_{\text{esec}} = 8.5$$

$$\text{Speedup}(\text{idealCache}, \text{realCache}) = 10.48 / 8.5 = 1.23$$



Esercizio 6

- Si calcoli il CPI di un'architettura dotata di cache con:
- $CPI_{ideal}=1.5$
- $MP=10$
- $MR=0.11$
- $\#acc_mem / instr=1.4$

$$CPI_{realCache} = 1.5 + 1.4 * 0.11 * 10 = 3.04$$



Exercise (6 cont.)

- Compare the case of 100% hit rate with the case of 100% miss rate.

$$CPI_{noCache} = 1.5 + 1.4 * 10 = 15.5$$

- Speedup in ideal cache:
 $CPI_{idealCache} = 1.5$

$$Speedup = 3.04 / 1.5 = 2$$



Exercise 7

- Consider two architectures: A and B
 - $T_{clk}(A)=20\text{ns}$, 8.5% faster than $T_{clk}(B)$
 - Both A and B have $\#mem_acc/instr=1.3$
 - $MP(A)=MP(B)=200\text{ ns}$
 - $MR(A)=3.9\%$, $MR(B)=3.0\%$
- Compute $AMAT(A)$ and $AMAT(B)$
- Compute $CPI(A)$ and $CPI(B)$



Solution 7

□ CPI(A)=

$$1.5+1.3*10*3.9\%=2.07$$

□ CPI(B)=

$$1.5+1.3*[200\text{ns}*3\%/(20\text{ns}+8.5\%*20\text{ns})] =1.85$$

□ AMAT(A)=

□ AMAT(B)=

$$20\text{ns}+200\text{ns}*3.9\%=27.8\text{ns}$$

$$20\text{ns}(1+8.5\%)+200\text{ns}*3.0\%=27.7\text{ns}$$



Exercise 8

- Architecture A[I\$,D\$]:
 - 1 instr. on 85% of the cycles; other cycles NOP.
- Architecture B[I\$,D\$]:
 - 2 instr. on 65% of cycles; 1 instr. on 30% of the time; other cycles NOP.
- Assume hit time= 1 cycle, miss time = 50 cycles.
- I\$ hit rate = 100%
- D\$ hit rate= 98%
- L/S instr = 33% of all instr.



Exercise 8 (cont.)

□ CPI(A) and CPI(B) with a perfect memory system?

- AMAT in cycles relative to D\$:
- $CPI(A) = 100 \text{ cycles} / 85 \text{ instr} = 1.17$
 - $CPI(B) = 100 / (65 * 2 + 30) = 0.62$

$$1 + 0.02 * 49 = 1.98 \text{ cycles}$$



Exercise 8 (cont.)

- CPI(A) and CPI(B) with actual cache?

- $\text{CPI}(A) = 1.17 + 0.33 * 0.02 * 49 = 1.49$

Speedup(B, A) = 1.58:

- $\text{CPI}(B) = 0.62 + 0.33 * 0.02 * 49 = 0.94$



Exercise 9

- ❑ 300 MHz CPU, 50 MHz bus speed
- ❑ Cache has 2 64-bit words per block
- ❑ Buses:
 - 2 bytes wide
 - burst transfer mode:
 - each block read is: 4-1-1-1-1-1-1-1 (bus clocks)
- ❑ Hit time= 1 cycle
- ❑ 6% miss rate.



Exercise 9 (cont.)

- Consider only read data accesses. What is the effective AMAT in *ns*?

- How would you speedup?
 $(1 + 0.06 * ((4 + 7) * 300 / 50))$ CPU clocks
 $= 4.96$ CPU clocks, 16.5 ns
 - Doubling bus width?
 - Doubling bus speed?
 - Compute first AMAT and then speedup



Exercise 9

- Doubling bus width?

First datum in 4 bus clocks, then 1-1-1

AMAT=

$$(1+0.06*(4+3)*6)\text{CPUclocks}$$

$$=3.52\text{CPUclocks}$$

$$=11.7 \text{ ns}$$



Exercise 9 (cont.)

- Doubling bus speed?

1 bus clock = 3 CPU cycles

AMAT =

$(1 + 0.06 * (4 + 7) * 3) \text{CPU clocks}$

$= 2.98 \text{CPU clocks}$

$= 9 \text{ ns}$

$\text{Speedup}(2 \times \text{freq}, 2 \times \text{width}) = 1.18$



Multi-Level Caches



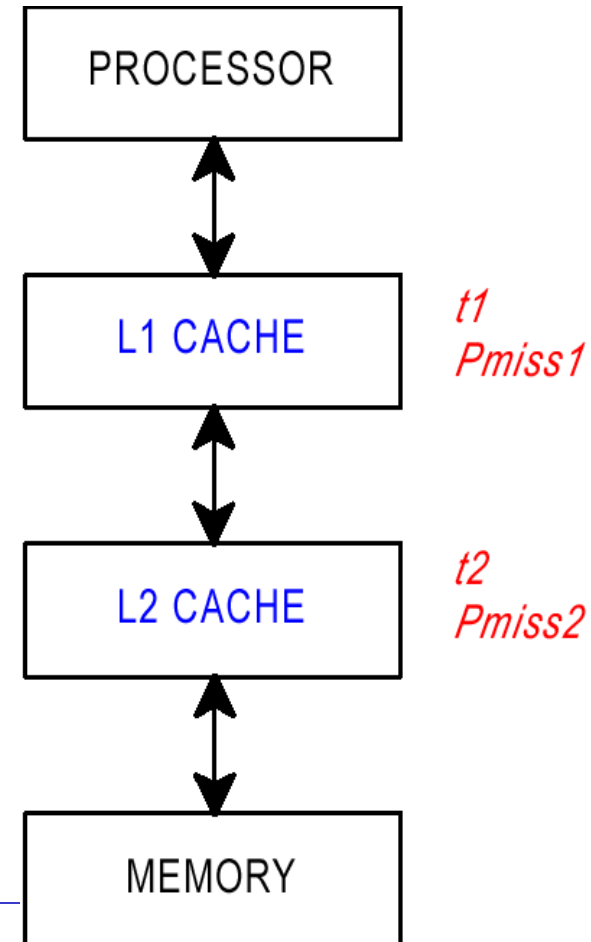
Multilevel Caches

Small, fast Level 1 (L1) cache

- Often on-chip for speed and bandwidth

Larger, slower Level 2 (L2) cache

- Closely coupled to CPU;
- may be on-chip, or “nearby” on module





Multilevel cache sizes examples

Intel:	L1	L2
80386	sometimes off-chip	none
80486:	8K	none; or 64K+ off-chip
Pentium:	(split) 8K each	256K - 512K off-chip
Pentium Pro:	(split) 8K each	256K - 512K on-module
Pentium II:	(split) 16K each	512K on-module
Pentium III:	(split) 16K each	256K - 512K on-module
Pentium III- Celeron:	(split) 16K each	128KB on-chip
Pentium IV:	12K/I\$, 4K/D\$	256K on chip



Why an L2 is necessary slower

- Longer **critical path**
- Off-chip access is **slower** than on-chip access
- Off-chip access is **narrower** than on-chip access (less bandwidth)



Two Level Miss Rates

□ Two-Level Miss Rates

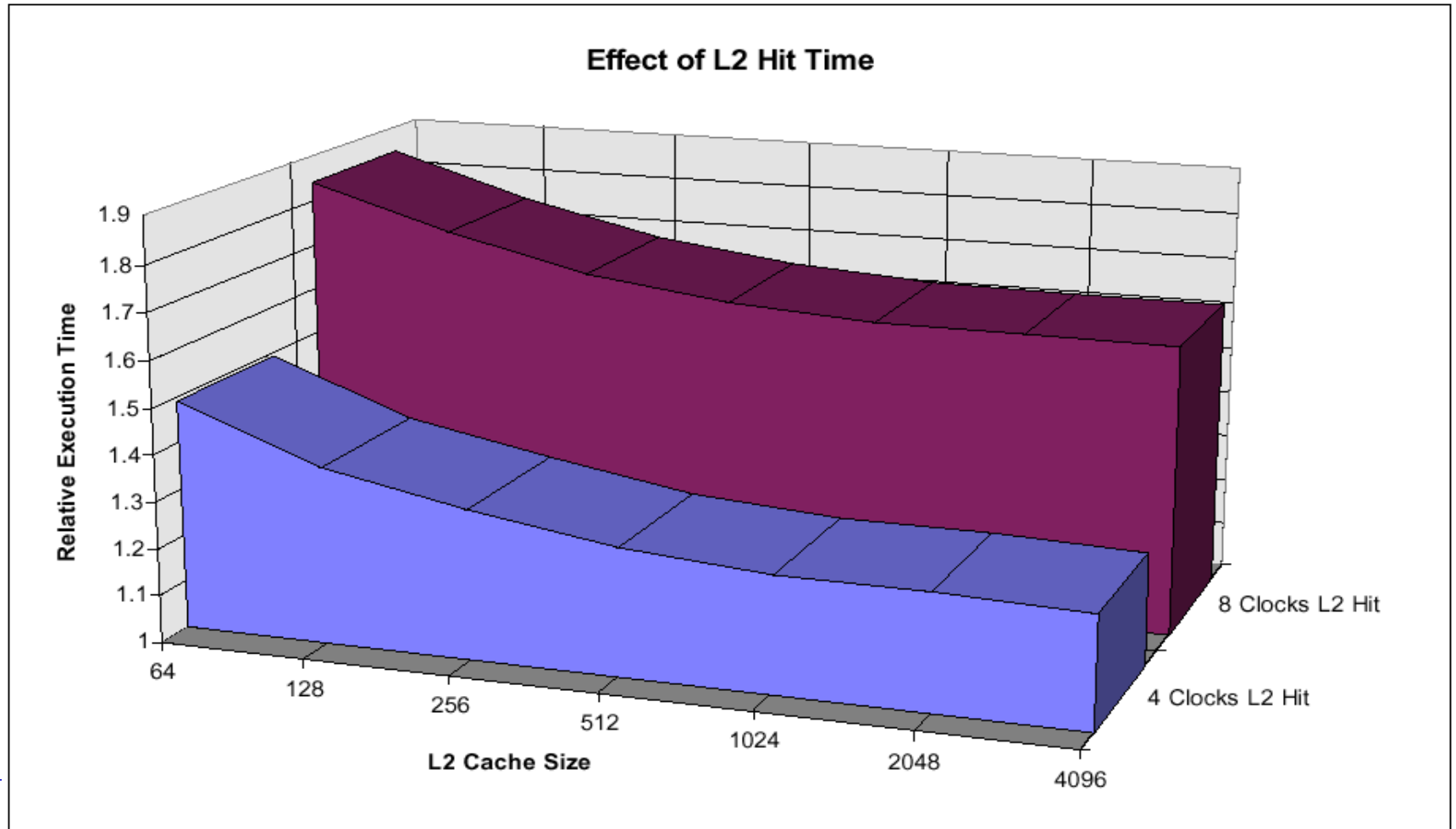
- **Local** miss rate: misses in cache / accesses to cache
 - L1 cache => P miss1
 - L2 cache => P miss2

- **Global** miss rate: misses in cache / accesses from CPU
 - L1 cache => P miss1
 - L2 cache => P miss1 * P miss2



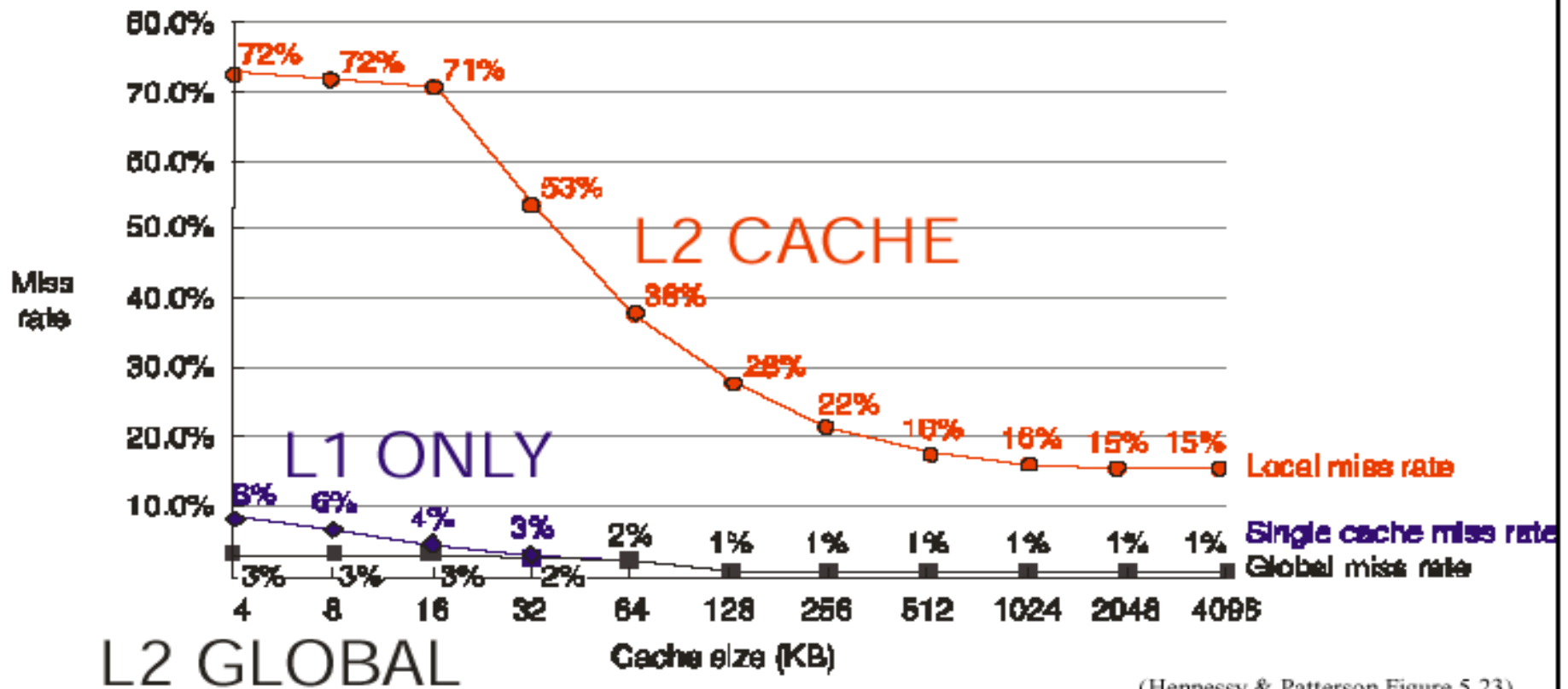
Effect of L2 cache size

1 cache fixed 32KB





Example Performance



(Hennessy & Patterson Figure 5.23)



Evaluating Multi-Level Miss Rates

- Use **Global** Miss rates when evaluating traffic filtering of 2-level caches
- **Sequential forward** model (local miss rates):

$$t_{ea} = t_{hitL1} + (P_{miss1} * t_{hitL2}) + (P_{miss1} * P_{miss2} * t_{transport})$$



Diversity Motivation

- L1 and L2 should have **differences** to improve overall performance
- **Issues:**
 - Split vs. Unified & bandwidth vs. flexibility
 - Write through vs. write back & write allocation
 - Block size & latency vs. bandwidth
 - Associativity vs. cycle time



Split vs. Unified

- **Split caches give bandwidth; unified caches give flexibility**
 - split L1 combined with unified L2
- **Split L1 cache features**
 - Good data & instruction access
 - Bad hit rate
- **Unified L2 cache advantages**
 - Chip costs reduced
 - No assumption on how memory is used
 - Varies with workloads



Write Policies

- Write through? Write allocation?
- **L1 cache: advantages of write through + no-write-allocate**
 - Control
 - No stalls
 - Avoids L1 cache pollution
 - Avoids problems with coherence
- **L2 cache: advantages of write back + write-allocate**
 - Reduces overall bus traffic
 - Captures temporal locality
 - **Safety net** for programs where write-allocate helps a lot



Block Size

- **Balances:**
 - miss rate vs. traffic ratio
 - Or, latency vs. bandwidth

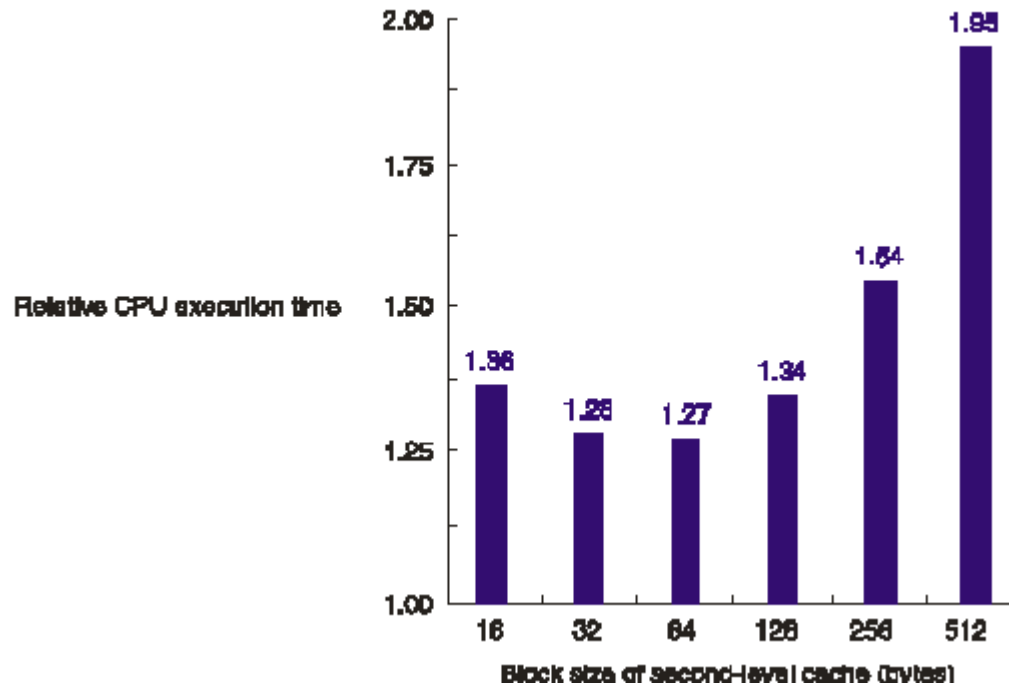
- **Smaller L1 cache sectors & blocks**
 - reduces conflict/capacity misses
 - reduces refill time
 - Limit ≥ 32 bits due to long floats

- **Larger L2 cache blocks**
 - Reduces misses
 - Typically, main memory has large latency on L2 miss
 - L1/L2 refill overlapped



Larger Block Sizes for L2

- Conflict misses relatively less important with larger cache





Associativity

□ **Balance** complexity, speed, efficiency

□ **L1 -- no clear winner**

- DM: faster cycle time, lower hit rate
- Set associative: slower cycle time, better hit rate

□ **L2 -- no clear winner**

- DM: minimizes pin & package count for cache
- Set associativity less advantageous for really large caches
- Set associative L2 gives flexibility
 - Handles degenerate cases
 - Associative time penalty is a smaller percentage of total miss delay w.r.t. L1



Multi-Level Inclusion

□ Complete **inclusion** means all elements in highest level of memory hierarchy are present in lower levels (also called “subset property”)

- Useful for multiprocessor coherence;

□ **Inclusion requires**

- Number of L2 sets \geq number of L1 sets
- L2 associativity \geq L1 associativity
- L1 shares LRU data with L2 to coordinate replacements



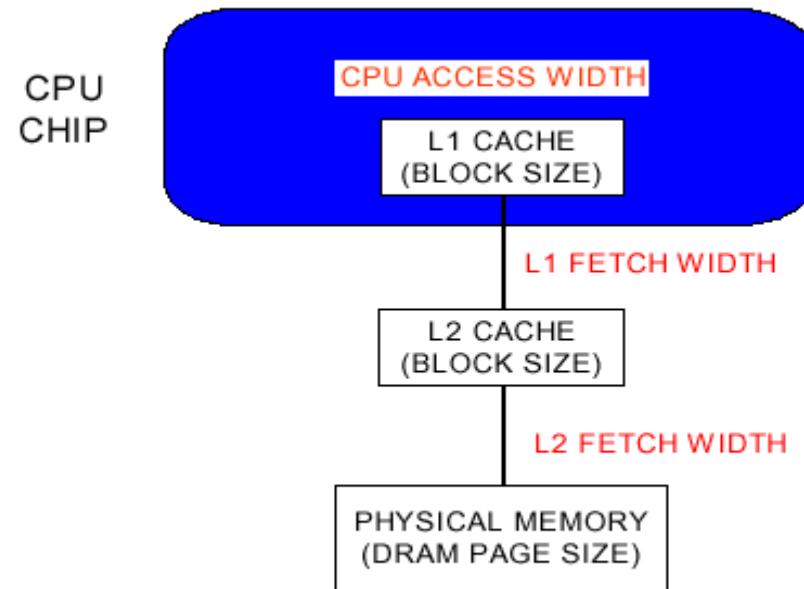
L1 vs. L2 Tradeoff Examples

□ Pentium Pro	L1	L2
□ Size	16KB	none - 256KB - 512KB
□ Organization	Split (8KB + 8KB)	Unified
□ Write Policies	programmable; programmable	
□ Block size	32 bytes	32 bytes
□ Associativity	D: 2-way; I: 4-way	4-way
□ MIPS R10000 L1 L2		
□ Size	64KB	512KB - 16 MB
□ Organization	Split (32KB + 32KB)	Unified
□ Write Policies	write back	write back
□ Block size	D: 32 bytes I: 64 bytes	64 or 128 bytes
□ Associativity	2-way	2-way



Multi-level Block Sizes

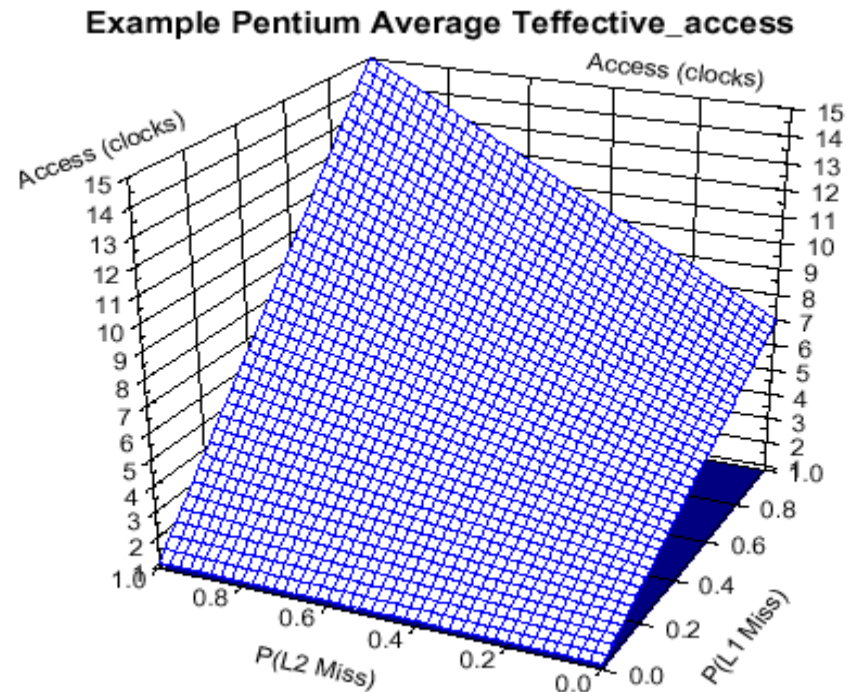
- Tradeoff for large block sizes vs. available access width
- Example: Pentium+430HX set
 - CPU Access Width
 - 8 bytes instruction / clock
 - 8 bytes data / clock
 - L1 Block Size = 32 bytes
 - L1 Fetch Width = 8 bytes
 - Example L2 Access: $3-1-1-1=6$
- L2 Block Size = 32 bytes
 - L2 Fetch Width = 8 bytes
 - Example L2 Miss: $8-2-2-2=14$
 - DRAM Page size is proportional to $\sqrt{\text{chip size}}$
 - (e.g., 16K bits for 16Mx4 chip)





Example of Multi-Level Access Time Equation

- $t_{ea} = t_{L1hit} + P_{L1miss} * t_{L2hit} + P_{L1miss} * P_{L2miss} * t_{L2miss}$
Pentium example (using marginal L2 miss penalties, not absolute)
- $t_{L1hit} = 1$ clock
- $t_{L1miss} = 6$ clocks
- $t_{L2miss} = 14$ clocks





Exercise 1

- CPU at 500MHz;
- L1 unified data cache
 - HtimeL1=2ns
 - MRL1=5%
- L2 unified data cache
 - HtimeL2=20ns
 - MRL2=25% (local miss rate)
 - MPL2=100ns



Exercise 1 (cont.)

- Consider the architecture w/o L2. It is worth to double L1 size (miss rate becomes 4%) while increasing hit time=2.4ns (consider clock period=hit time)?

Compute the speedup of this solution w.r.t. the original one.

$$T_{\text{cpu clock}}(\text{undoubled}) = 2\text{ns}$$

$$\text{CPI}(\text{undoubled}) = 1 \text{ clock} + 0.05 * 50 = 3.5 \text{ clocks} = 7\text{ns}$$

$$T_{\text{cpu clock}}(\text{doubled}) = 2.4\text{ns}$$

$$\text{CPI}(\text{doubled}) = 1 \text{ clock} + 0.04 * (100/2.4) = 2.68 \text{ clocks} = 6.4\text{ns}$$

$$\text{Speedup}(\text{doubled}/\text{undoubled}) = 1.088 \Rightarrow 8.8\% \text{ faster}$$



Exercise 1 (cont.)

- When the L2 cache is added, it is a good idea to double the L1 cache?

Compute the speedup(wL2,w/oL2).

$T_{cpu\ clock}(undoubled) = 2ns$

$HTL1 = 1 \text{ clock}$

$HTL2 = 10 \text{ clocks } (20ns/2ns)$

$MRL1 = 5\%$

$MRL2 = 25\%$

$MPL2 = 50 \text{ clocks } (100ns/2ns)$

$MPL1 = 10 + 0.25 * 50 = 22.5 \text{ clocks}$

$$\begin{aligned} \mathbf{AMAT} &= \mathbf{1 + 0.05 * 22.5 = 2.12\text{clk}} \\ &= \mathbf{4.25ns} \end{aligned}$$



Exercise 1 (cont.)

$T_{\text{cpu clock(doubled)}} = 2.4\text{ns}$

$\text{HTL1} = 1 \text{ clock}$

$\text{HTL2} = 9 \text{ clocks (20ns/2.4ns)}$

$$\text{AMAT} = 1 + 0.04 * 19.5 = 1.78\text{clk} \\ = 4.27\text{ns}$$

$\text{MRL1} = 4\%$

$\text{MRL2} = 25\%$

$\text{MPL2} = 42 \text{ clocks (100ns/2.4ns)}$

$\text{MPL1} = 9 + 0.25 * 42 = 19.5 \text{ clocks}$

$\text{Speedup(doubled/undoubled)} = 0.995 \Rightarrow -0.5\%!!!$



Exercise 2

- Consider the following system
 - L1
 - Split cache (4K data, 4K instr).
 - Both Direct mapped
 - 8 bytes/block
 - write through
 - hit time is 1 clock
 - local miss rate = 0.15



Exercise 2 (cont.)

- L2
 - Unified cache
 - 160 KB
 - 5 way set associative
 - 8 bytes block
 - L2 hit=5 clocks
 - local miss rate = 0.05
 - l2 miss = 50 clock cycles
 - Write back
 - Write allocate



Exercise 2 (cont.)

- Total n° bits for each L1 block?

$$\text{Indexbits} = \log(2^{12}/2^3) = 9$$

- Total n° bits for each L2 block? Dont consider LRU timers
 $64\text{bit}[\text{block}] + [32 - 3[\text{offset}] - 9[\text{index}]] + 1\text{valid} = 85\text{bit}$

$$\text{Indexbits} = \log(160\text{KB}/5/2^3) = 12\text{bits}$$

$$64\text{bit}[\text{block}] + [32 - 3[\text{offset}] - 12[\text{index}]] + 1\text{valid} + 1\text{dirty} = 83\text{bit}$$



Exercise 2

□ AMAT?

□ If a program is interrupted and flushed and then restarted, how much longer will it take to run?

$1 + 0.15 * 5 + 0.15 * 0.05 * 50 = 2.125$ clocks