CSEE 3827: Fundamentals of Computer Systems, Spring 2011

11. Caches

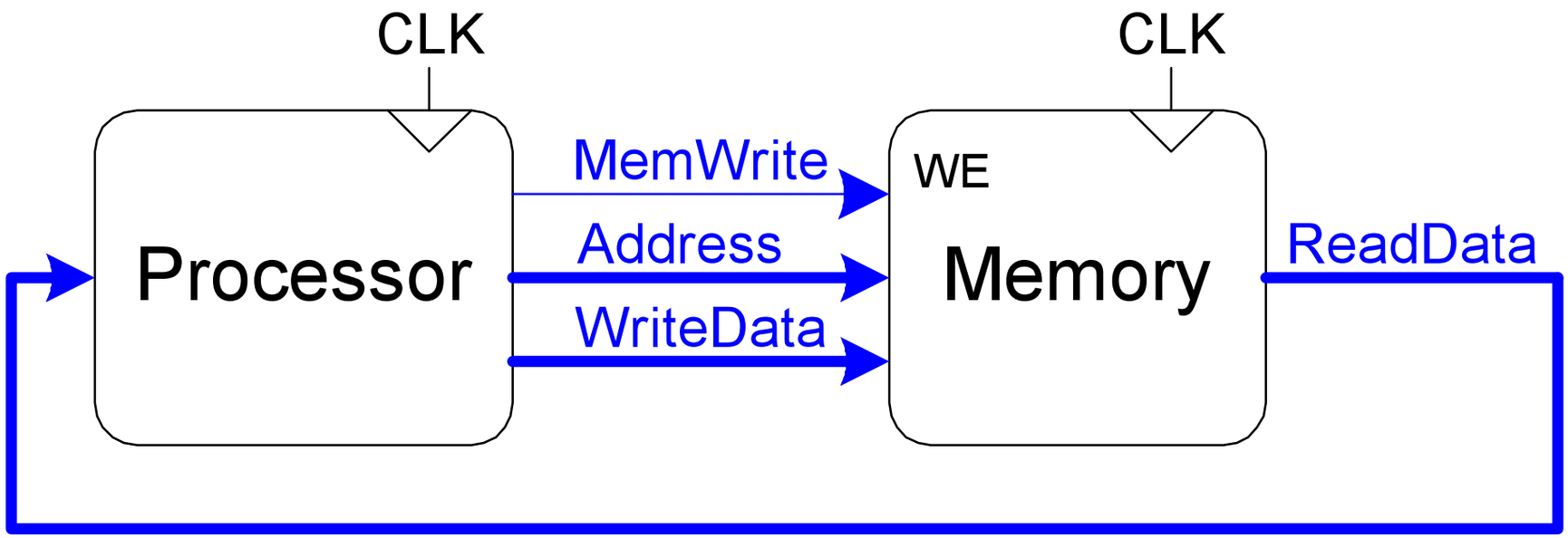
Prof. Martha Kim (martha@cs.columbia.edu)
Web: http://www.cs.columbia.edu/~martha/courses/3827/sp11/

Outline (H&H 8.2-8.3)

- Memory System Performance Analysis
- Caches

Introduction

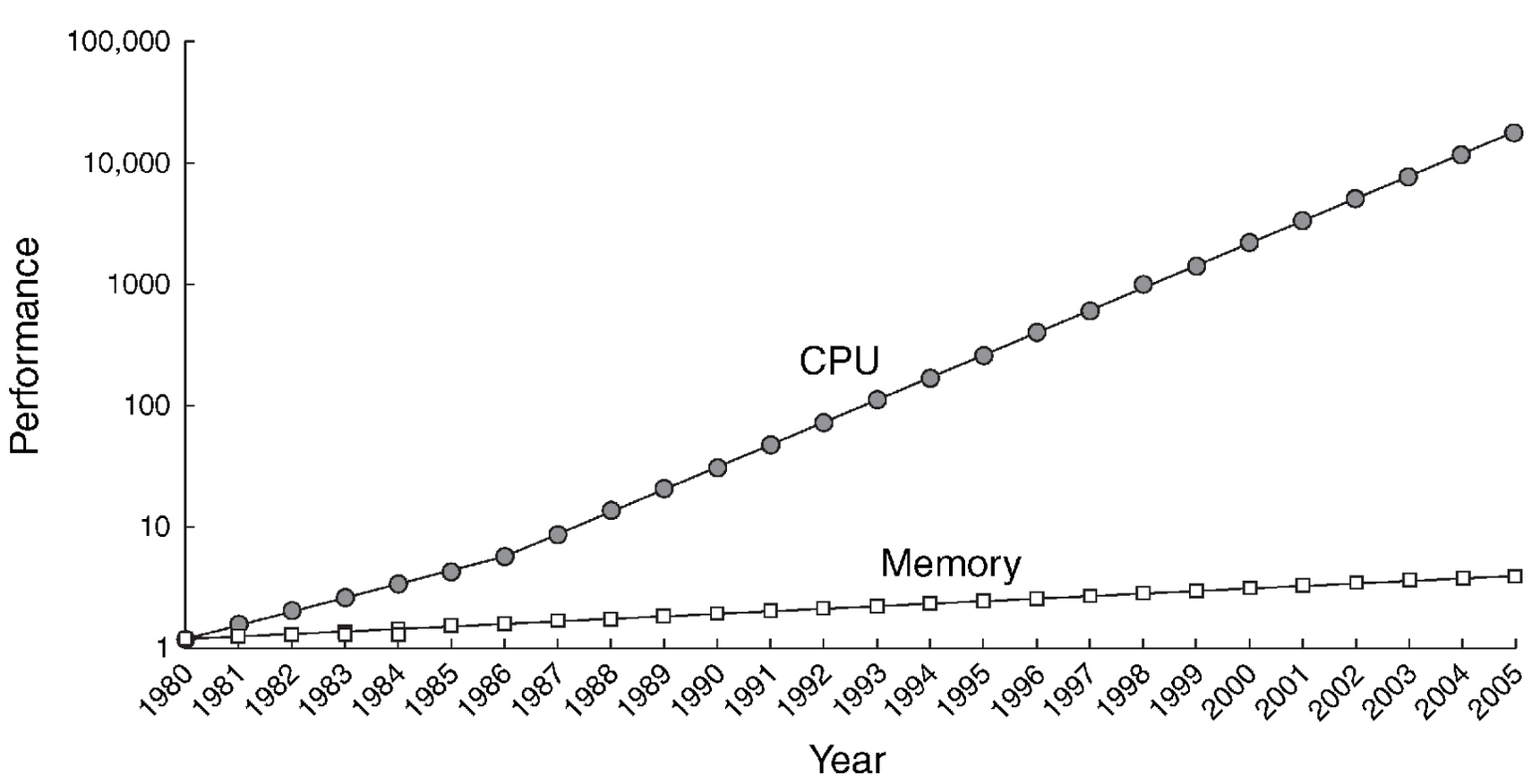
- Computer performance depends on:
  - Processor performance
  - Memory system performance



CPU time = (CPU clock cycles + Memory-stall clock cycles) * Cycle time

Memory Speed History

- So far, assumed memory could be accessed in 1 clock cycle
- That hasn’t been true since the 1980’s

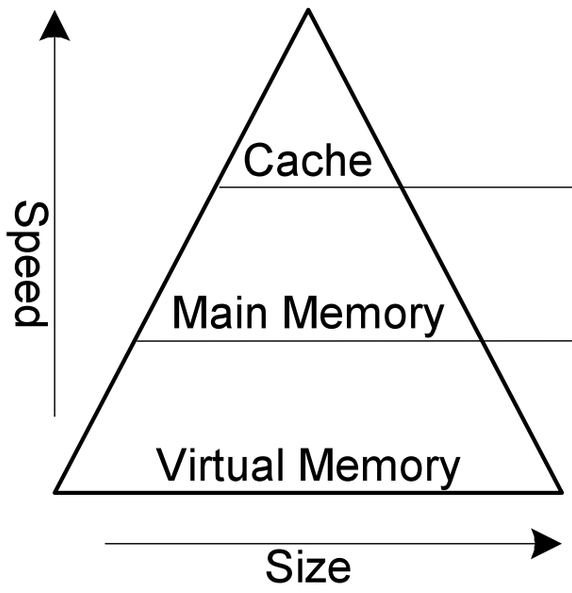


Memory Hierarchy

- Make memory system appear as fast as processor
- Ideal memory
  - Fast
  - Cheap (inexpensive)
  - Large (capacity)

} choose two!

- Solution: Use a hierarchy of memories



Technology	cost / GB	Access time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

Locality

- Exploit locality to make memory accesses fast
- **Temporal Locality**
  - Locality in time (e.g., if looked at a Web page recently, likely to look at it again soon)
  - If data used recently, likely to use it again soon
  - **How to exploit**: keep recently accessed data in higher levels of memory hierarchy
- **Spatial Locality**
  - Locality in space (e.g., if read one page of book recently, likely to read nearby pages soon)
  - If data used recently, likely to use nearby data soon
  - **How to exploit**: when access data, bring nearby data into higher levels of memory hierarchy too

Memory Performance

- **Hit:** is found in that level of memory hierarchy
- **Miss:** is not found (must go to next level)
- **Hit Rate** = # hits / # memory accesses = 1- Miss Rate
- **Miss Rate** = # misses / #memory accesses = 1 - Hit Rate
- **Expected Access Time:** average time to access data from level L of the hierarchy

$$EAT_L = AT_L + (MR_L \times EAT_{L+1})$$

Memory Performance Example

- A program has 2,000 load and store instructions
- 1,250 of these data values found in cache
- The rest are supplied by other levels of memory hierarchy
- What are the hit and miss rates for the cache?

Hit Rate = 1250/2000 = 0.625
Miss Rate = 750/2000 = 0.375 = 1 – Hit Rate

- Suppose hierarchy has two levels:
  - cache (1 cycle AT)
  - main memory (100 cycle AT)
- What is the EAT for this program?

*EAT(cache) = AT(cache) + MR(cache) * EAT(memory)*
*EAT(cache) = 1 + .375*100 = 38.5 cycles*

Cache

- Highest level in memory hierarchy
- Fast (typically ~ 1 cycle access time)
- Ideally supplies most of the data to the processor
- Usually holds most recently accessed data
- Cache design questions
  - *What data is held in the cache?*
  - *How is data found?*
  - *What data is replaced?*
- We'll focus on data loads, but stores follow same principles

What data is held in the cache?

- Ideally, cache anticipates data needed by processor and holds it in cache
- But impossible to predict future
- So, use past to predict future – temporal and spatial locality:
  - Temporal locality: copy newly accessed data into cache. Next time it's accessed, it's available in cache.
  - Spatial locality: copy neighboring data into cache too. Block size = number of bytes copied into cache at once.

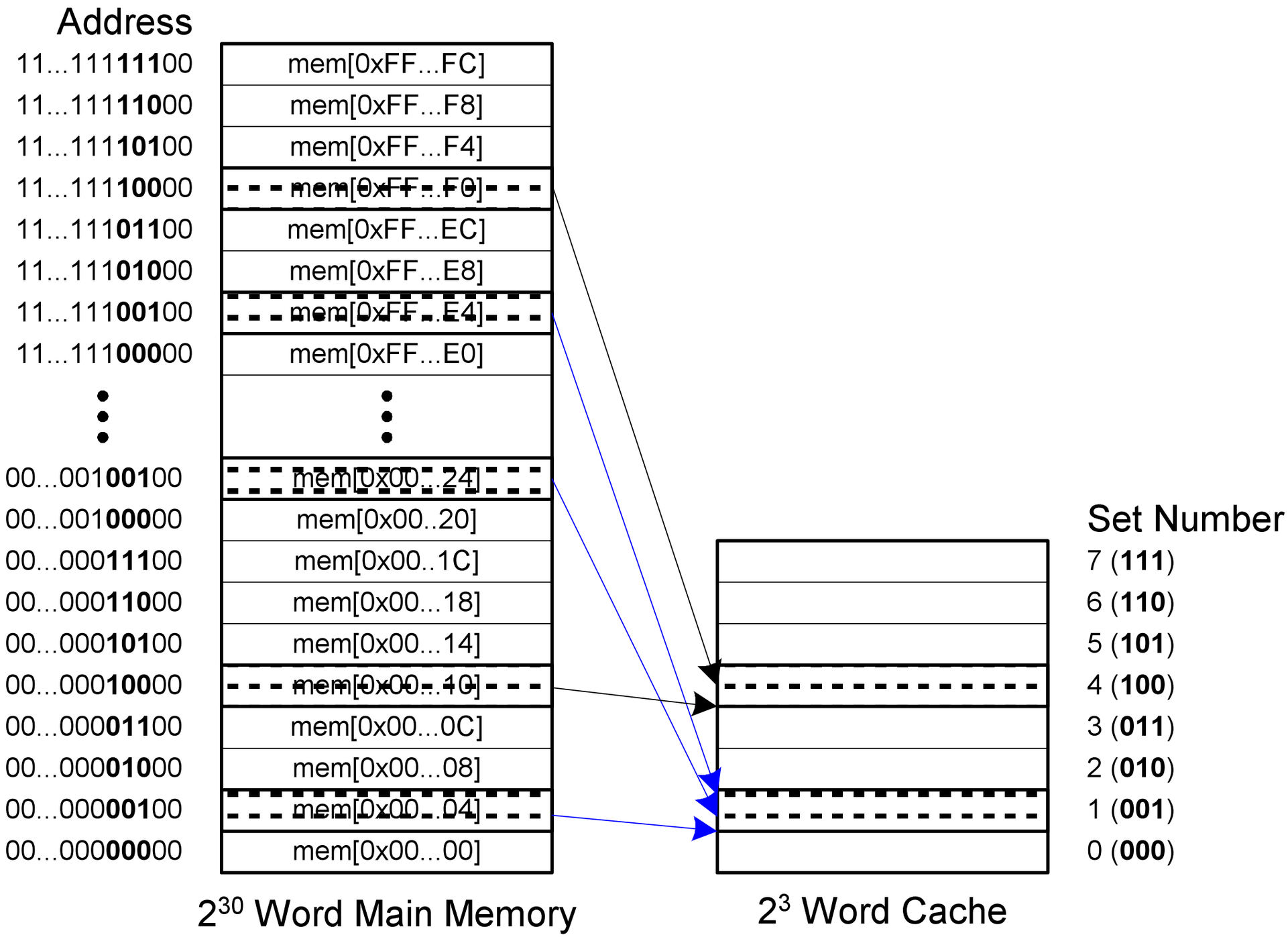
Cache Terminology

- **Capacity (C):** the number of data bytes a cache stores
- **Block size (b):** bytes of data brought into cache at once
- **Number of blocks (B = C/b):** number of blocks in cache: B = C/b
- **Degree of associativity (N):** number of blocks in a set
- **Number of sets (S = B/N):** each memory address maps to exactly one cache set

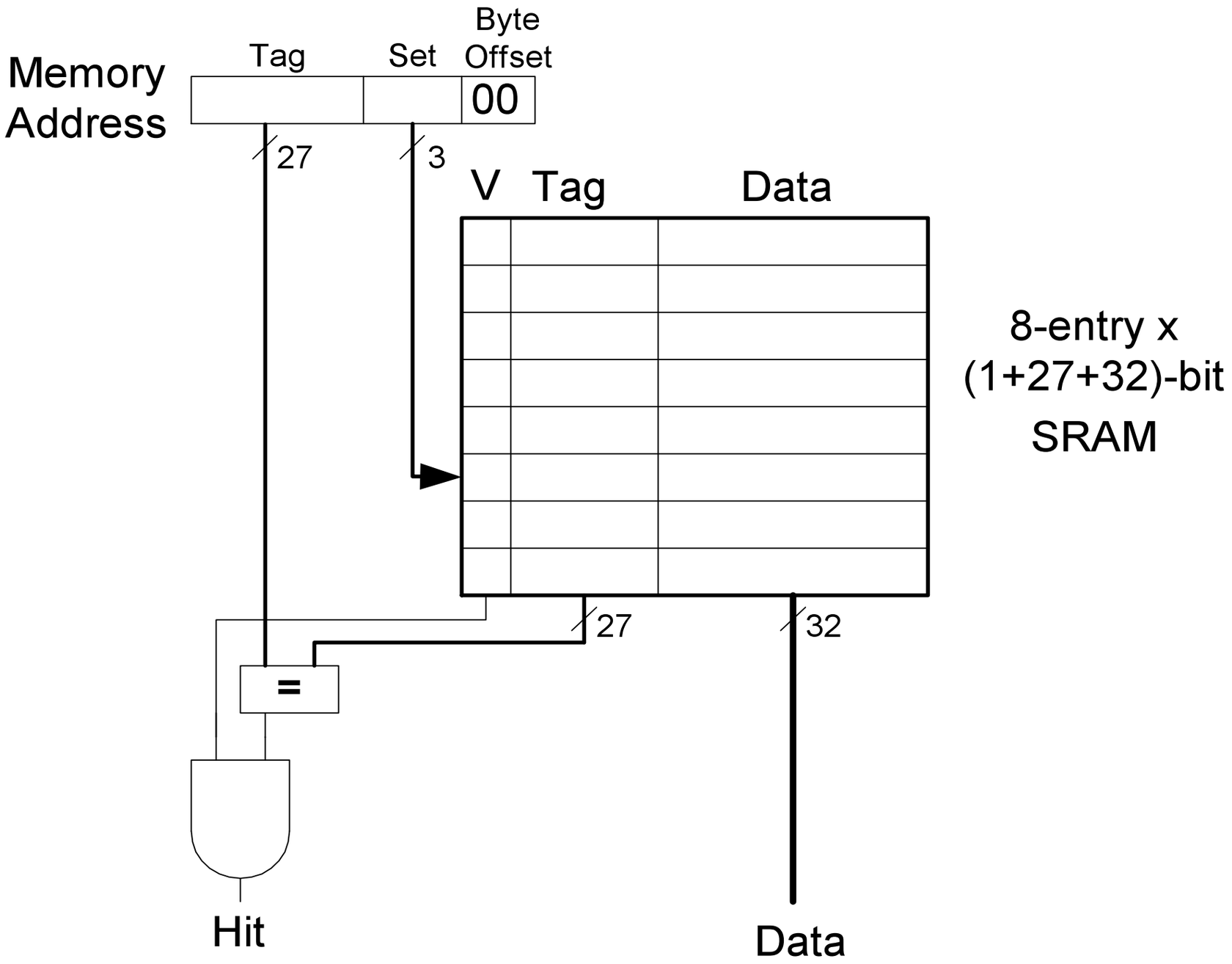
How is data found?

- Cache organized into S sets
- Each memory address maps to exactly one set
- Caches categorized by number of blocks in a set:
  - Direct mapped: 1 block per set
  - N-way set associative: N blocks per set
  - Fully associative: all cache blocks are in a single set
- Examine each organization for a cache with:
  - Capacity (C = 8 words)
  - Block size (b = 1 word)
  - So, number of blocks (B = 8)

Direct Mapped Cache (Concept)

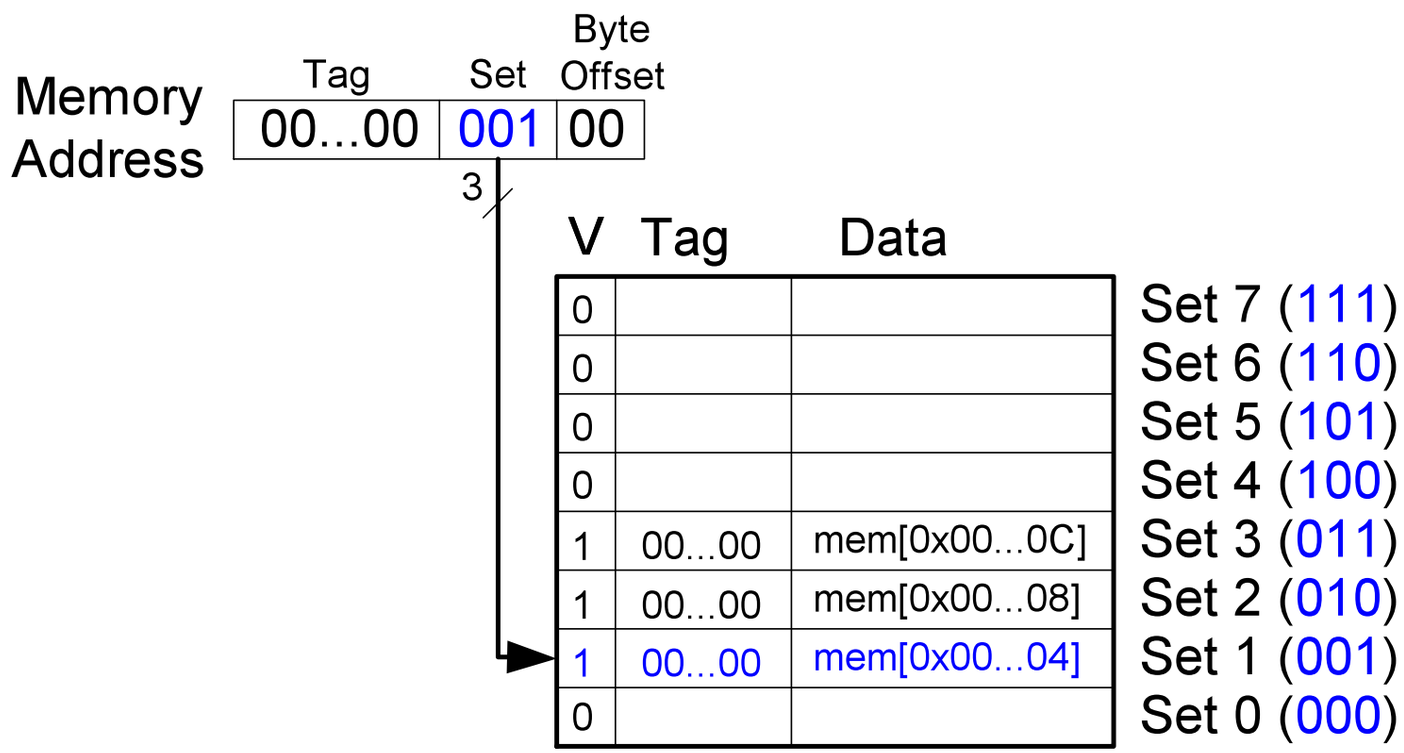


Direct Mapped Cache (Hardware)



Direct Mapped Cache Performance

```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

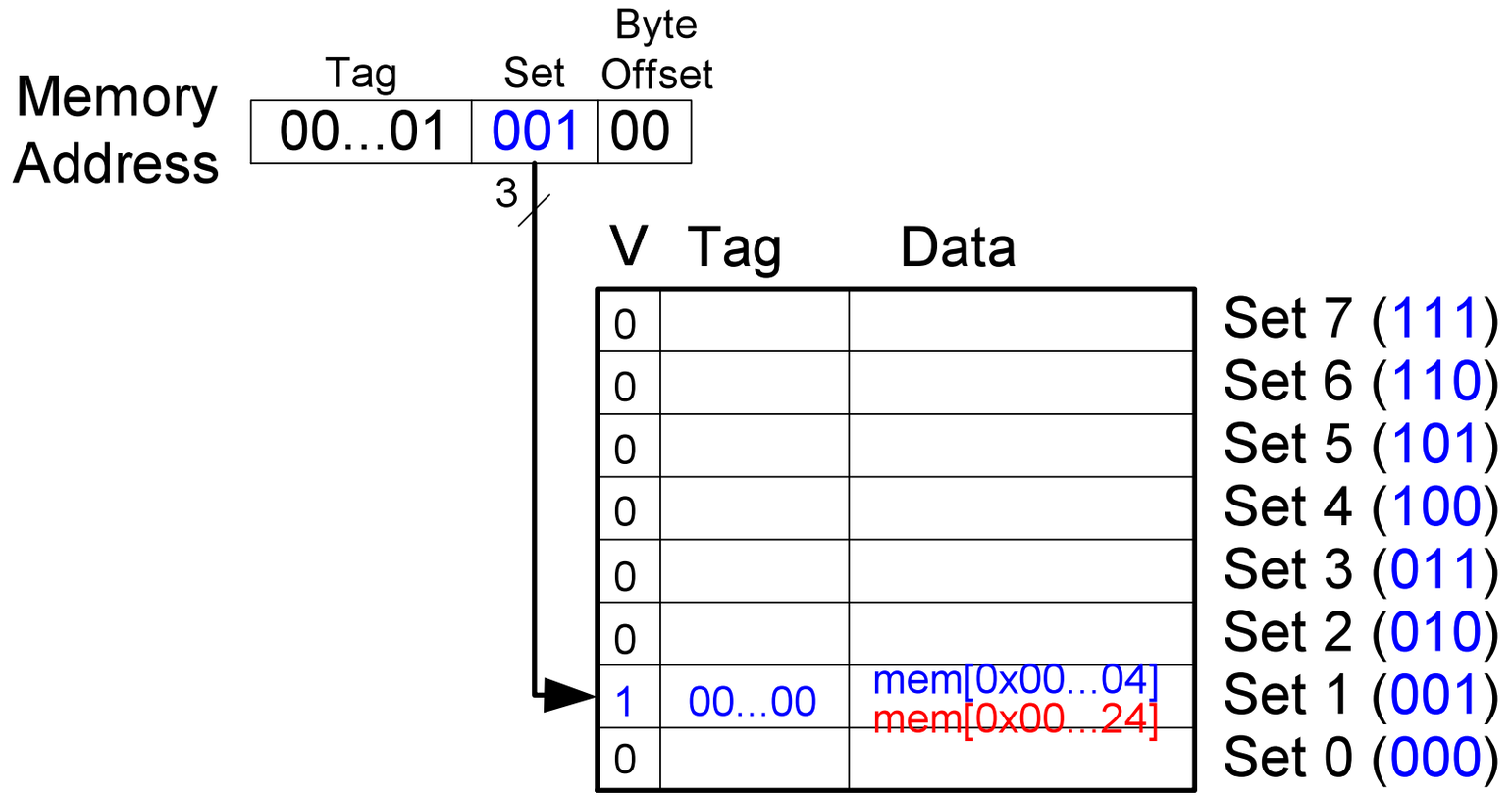


Miss Rate = 3/15
= 20%

Temporal Locality
Compulsory Misses

Direct Mapped Cache: Conflict

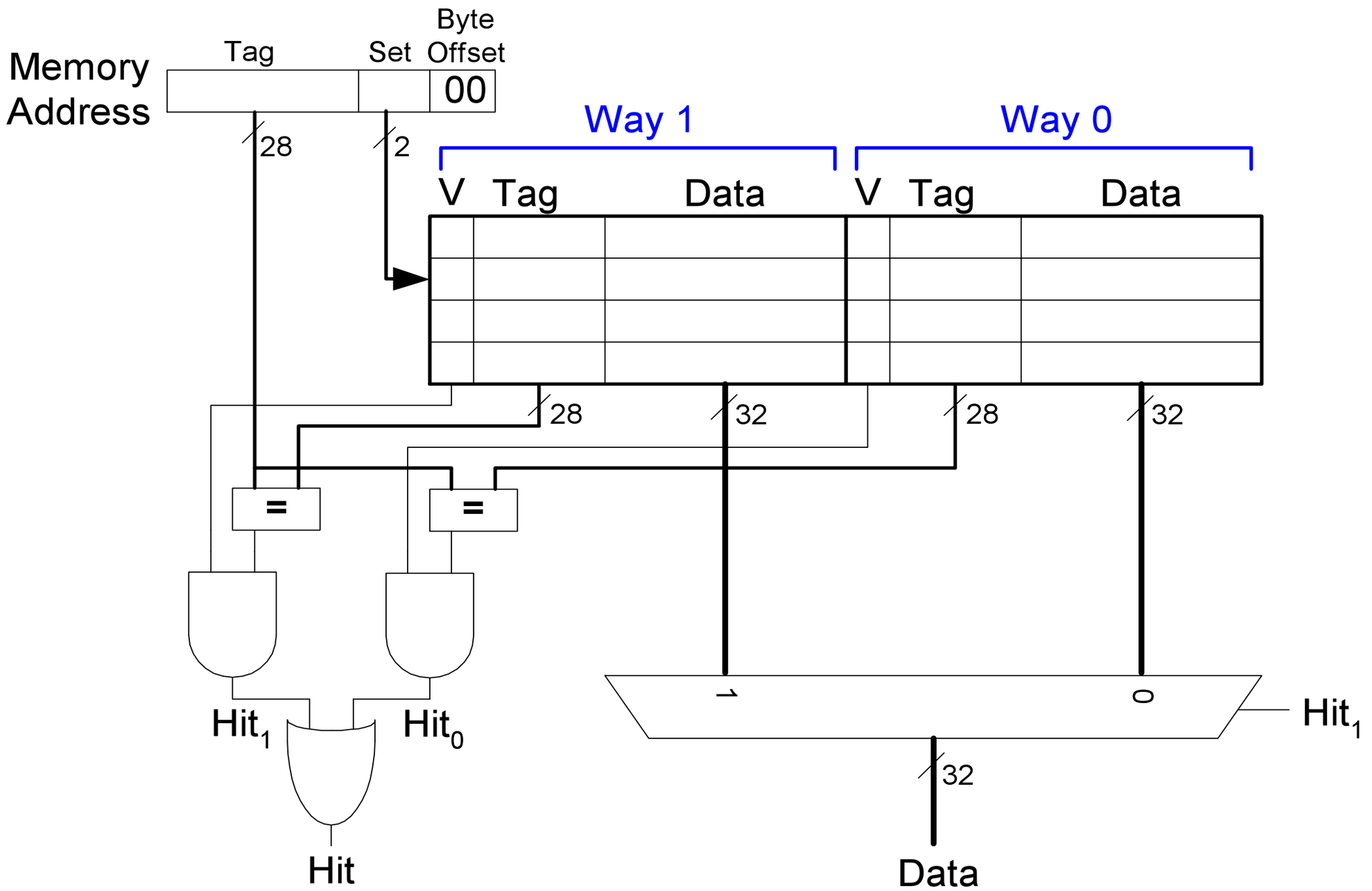
```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Miss Rate = 10/10
= 100%

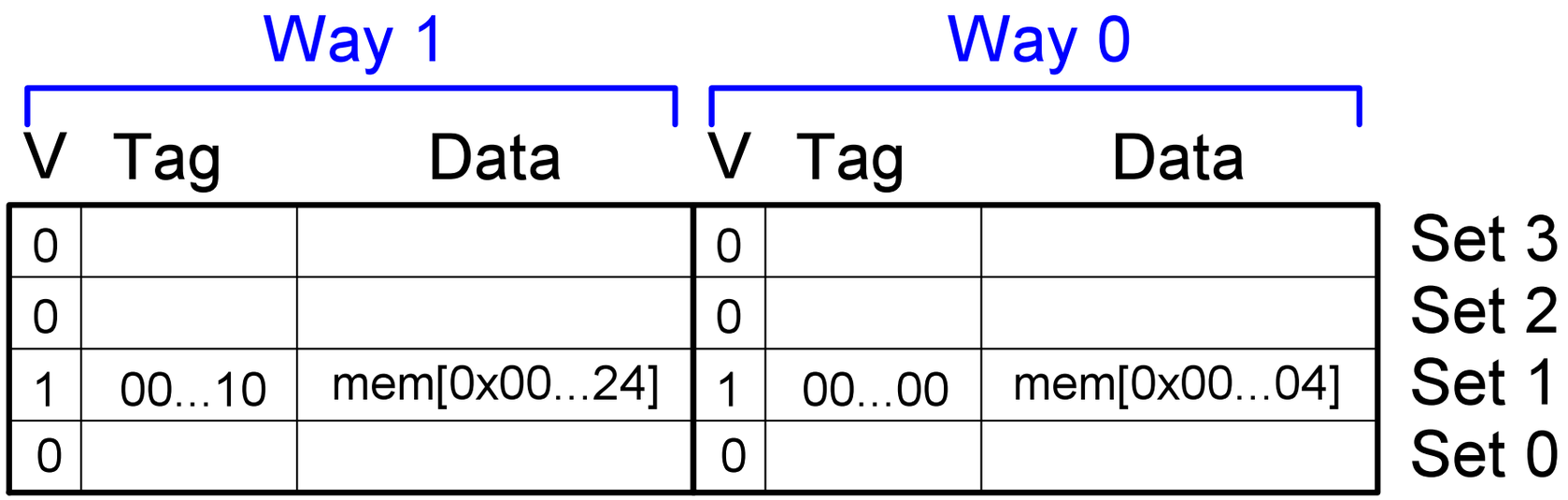
Conflict Misses

N-Way Set Associative Cache



N-Way Set Associative Performance

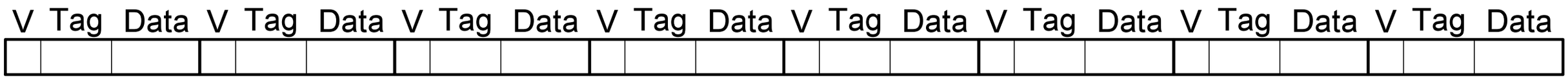
```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Miss Rate = 2/10
= 20%

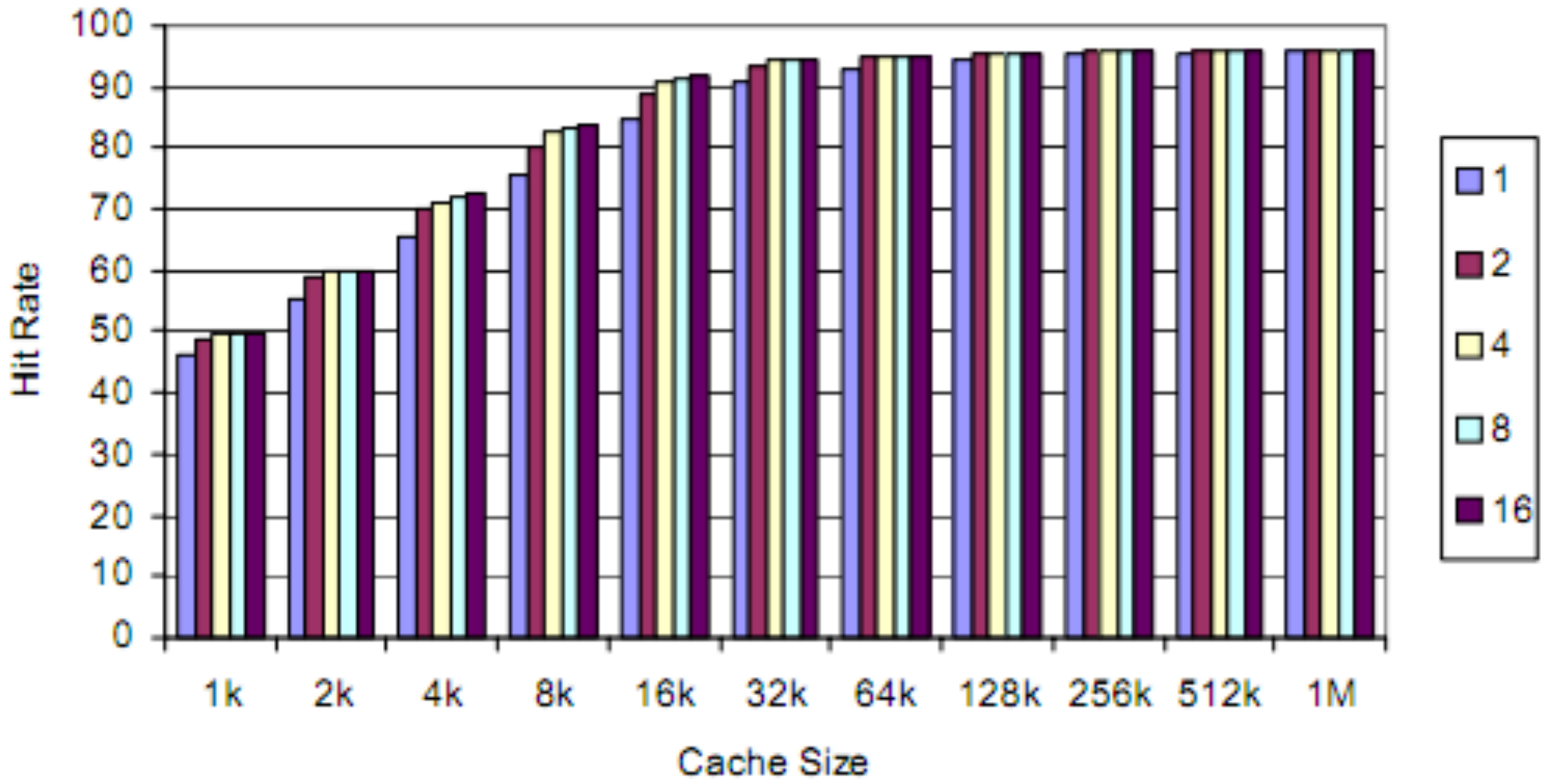
Associativity reduces conflict misses

Fully Associative Cache



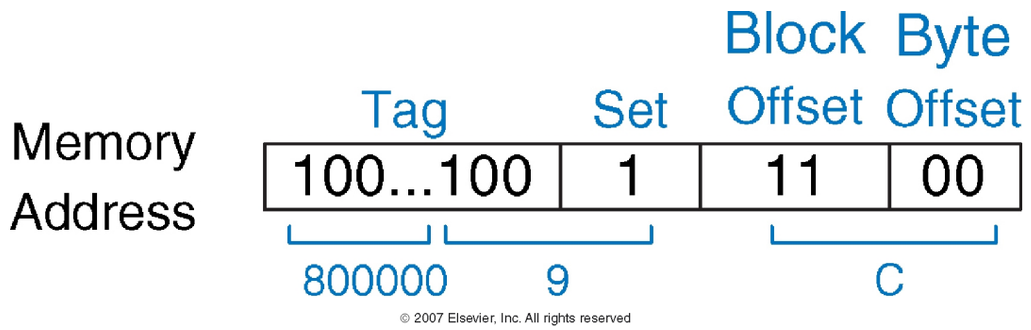
No conflict misses (all misses either compulsory or capacity)
Very expensive to build due to associative lookup

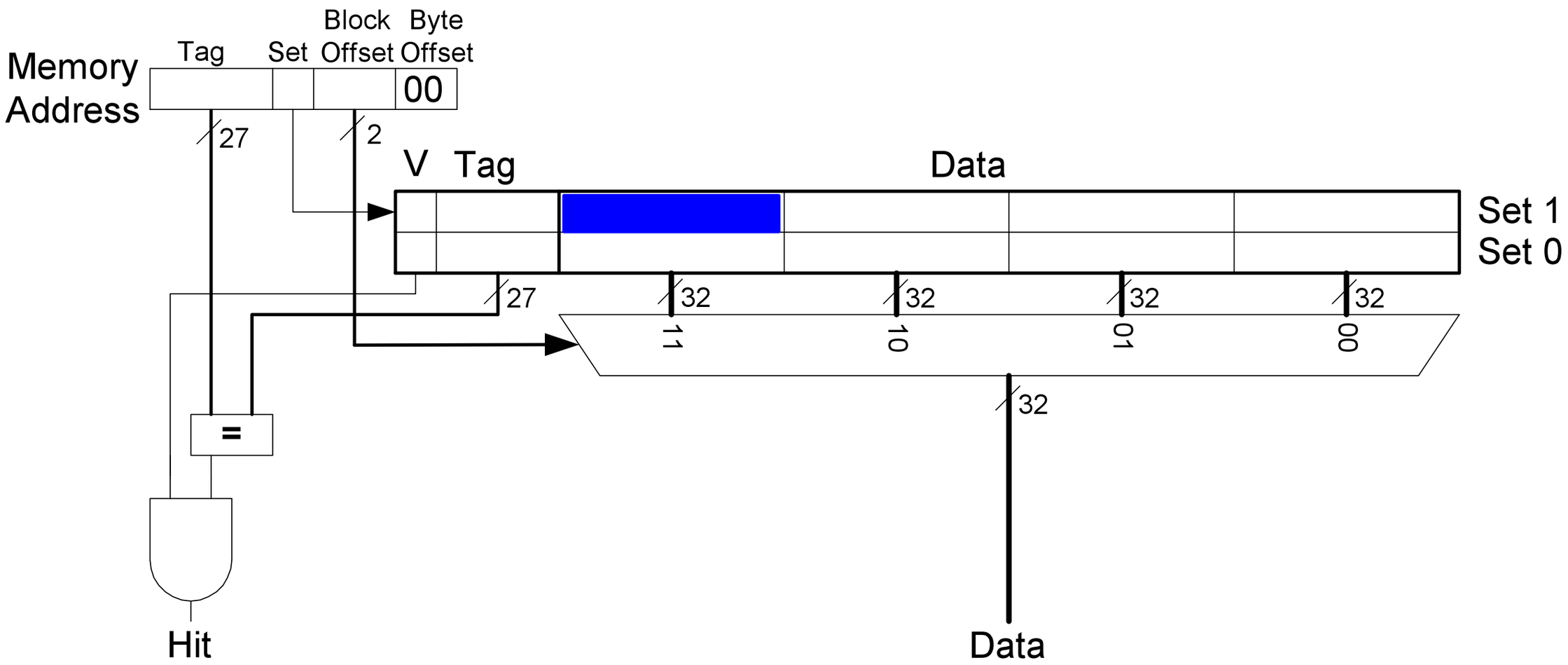
Hit Rate v. Associativity & Cache Size



(L1 cache, Running GCC)

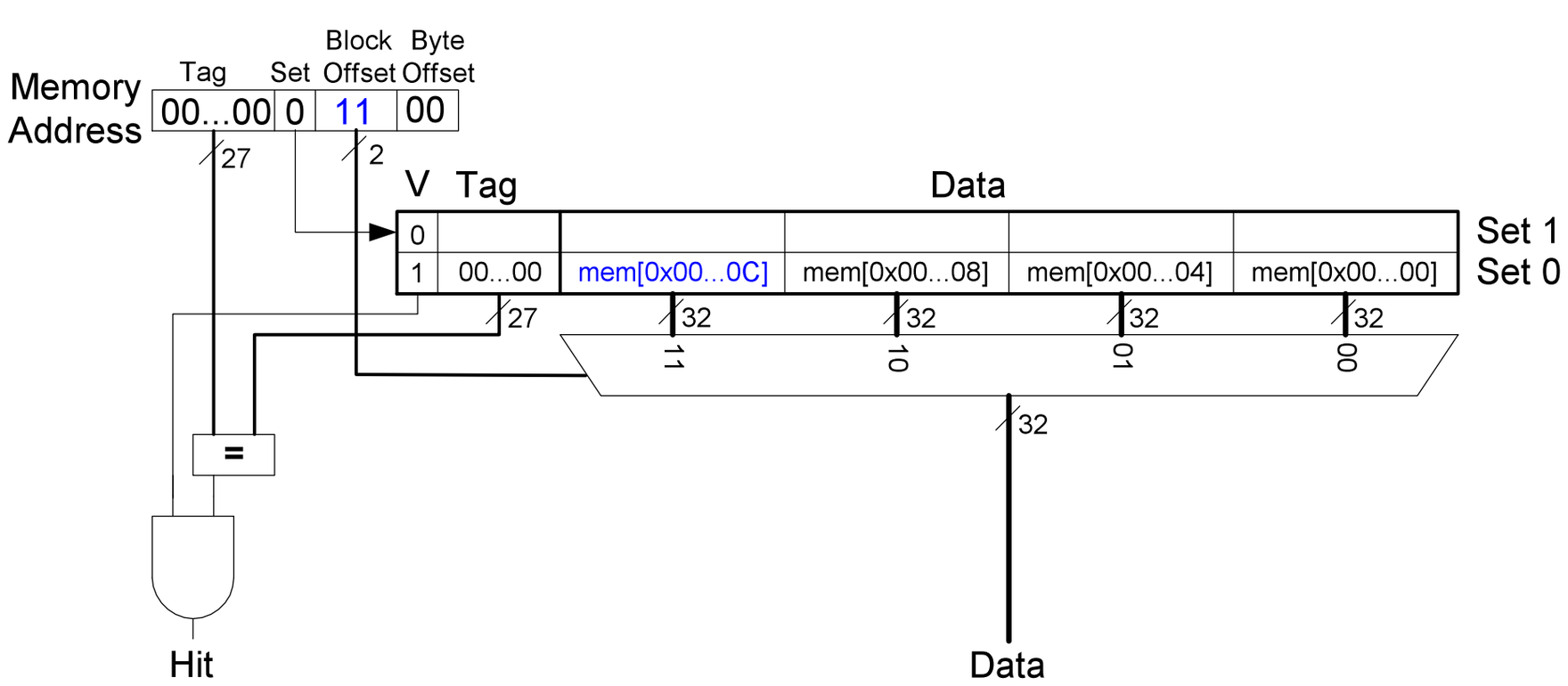
Cache with Larger Block Size





Direct Mapped Cache Performance

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Miss Rate = 1/15
= 6.67%

Larger blocks reduce compulsory misses through spatial locality

Cache Organization Recap

- Capacity: C
- Block size: b
- Number of blocks in cache: B = C/b
- Number of blocks in a set: N
- Number of Sets: S = B/N

Organization	Number of Ways (N)	Number of Sets (S = B/N)
Direct Mapped	1	B
N-Way Set Associative	1 < N < B	B / N
Fully Associative	B	1

Capacity Misses

- Cache is too small to hold all data of interest at one time
- If the cache is full and program tries to access data X that is not in cache, cache must evict data Y to make room for X
- **Capacity miss** occurs if program then tries to access Y again
- X will be placed in a particular set based on its address
  - *In a direct mapped cache, there is only one place to put X*
  - *In an associative cache, there are multiple ways where X could go in the set.*
- How to choose Y to minimize chance of needing it again?
- **Least recently used (LRU) replacement**: the least recently used block in a set is evicted when the cache is full.

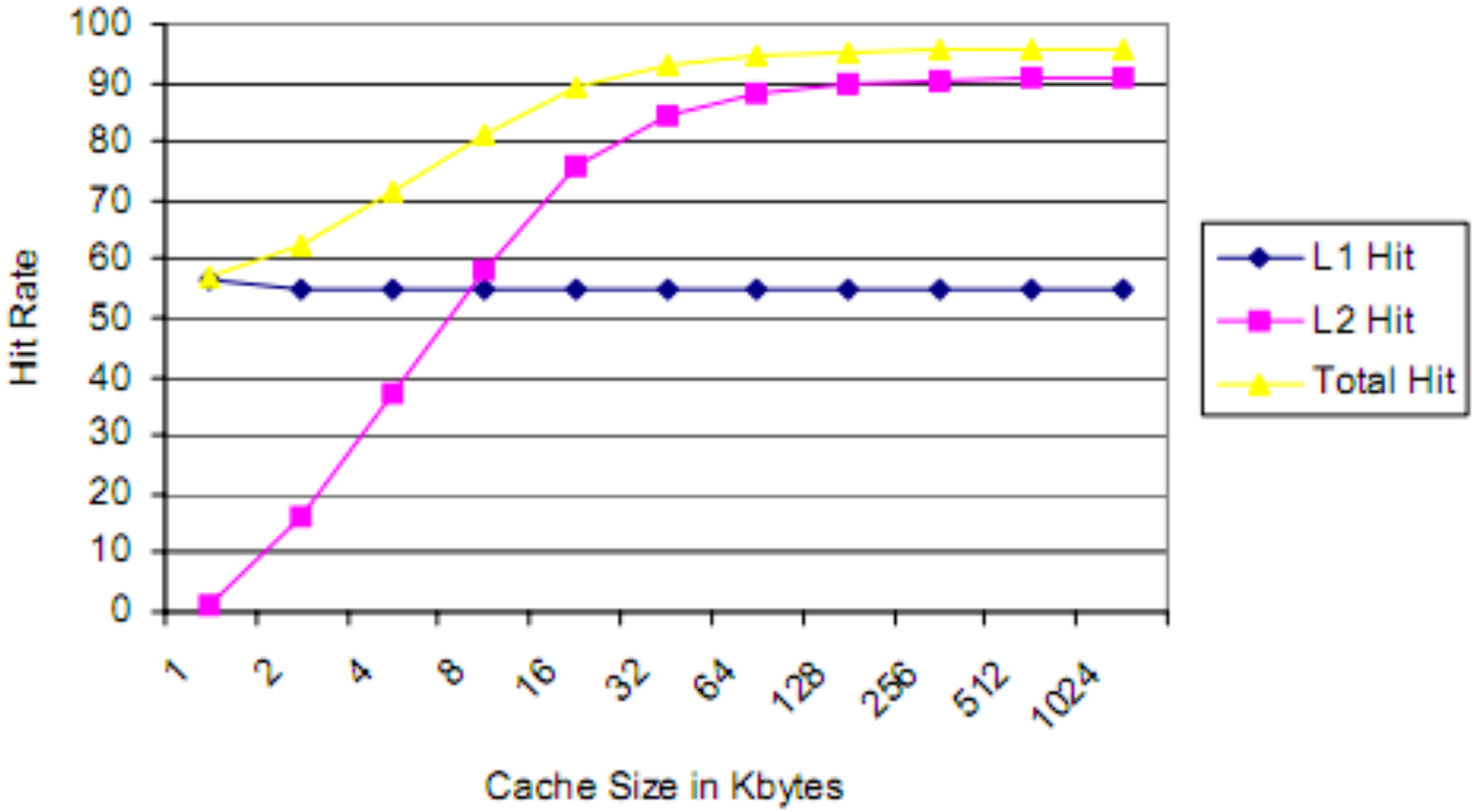
Caching Summary

- What data is held in the cache?
  - Recently used data (temporal locality)
  - Nearby data (spatial locality, with larger block sizes)
- How is data found?
  - Set is determined by address of data
  - Word within block also determined by address of data
  - In associative caches, data could be in one of several ways
- What data is replaced?
  - Least-recently used way in the set

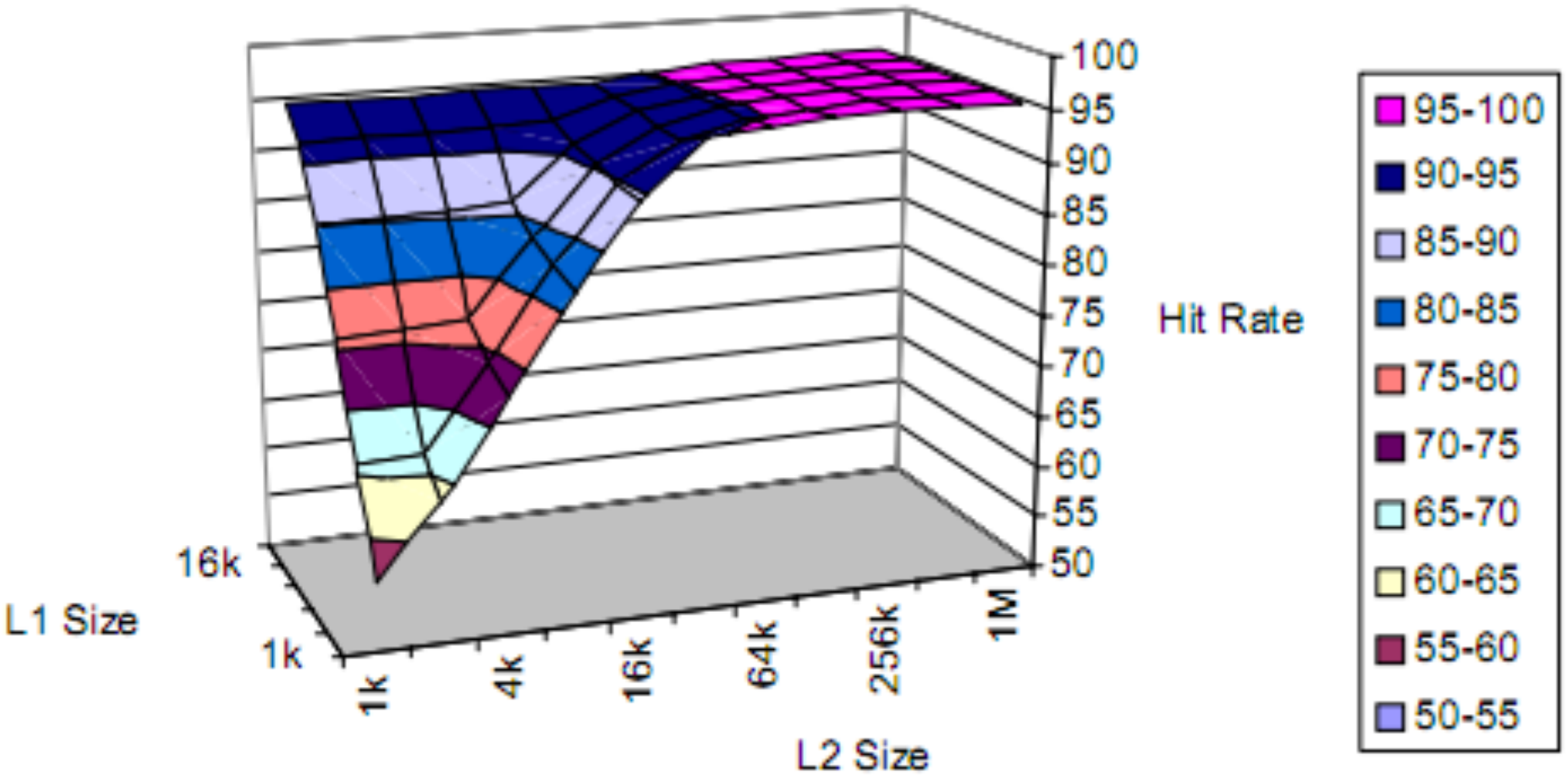
Multilevel Caches

- Larger caches have lower miss rates, longer access times
- Expand the memory hierarchy to multiple levels of caches
- Level 1: small and fast (e.g. 16 KB, 1 cycle)
- Level 2: larger and slower (e.g. 256 KB, 2-6 cycles)
- Even more levels are possible

Hit Rates for Constant L1, Increasing L2



Hit Rate v. L1 and L2 Cache Size

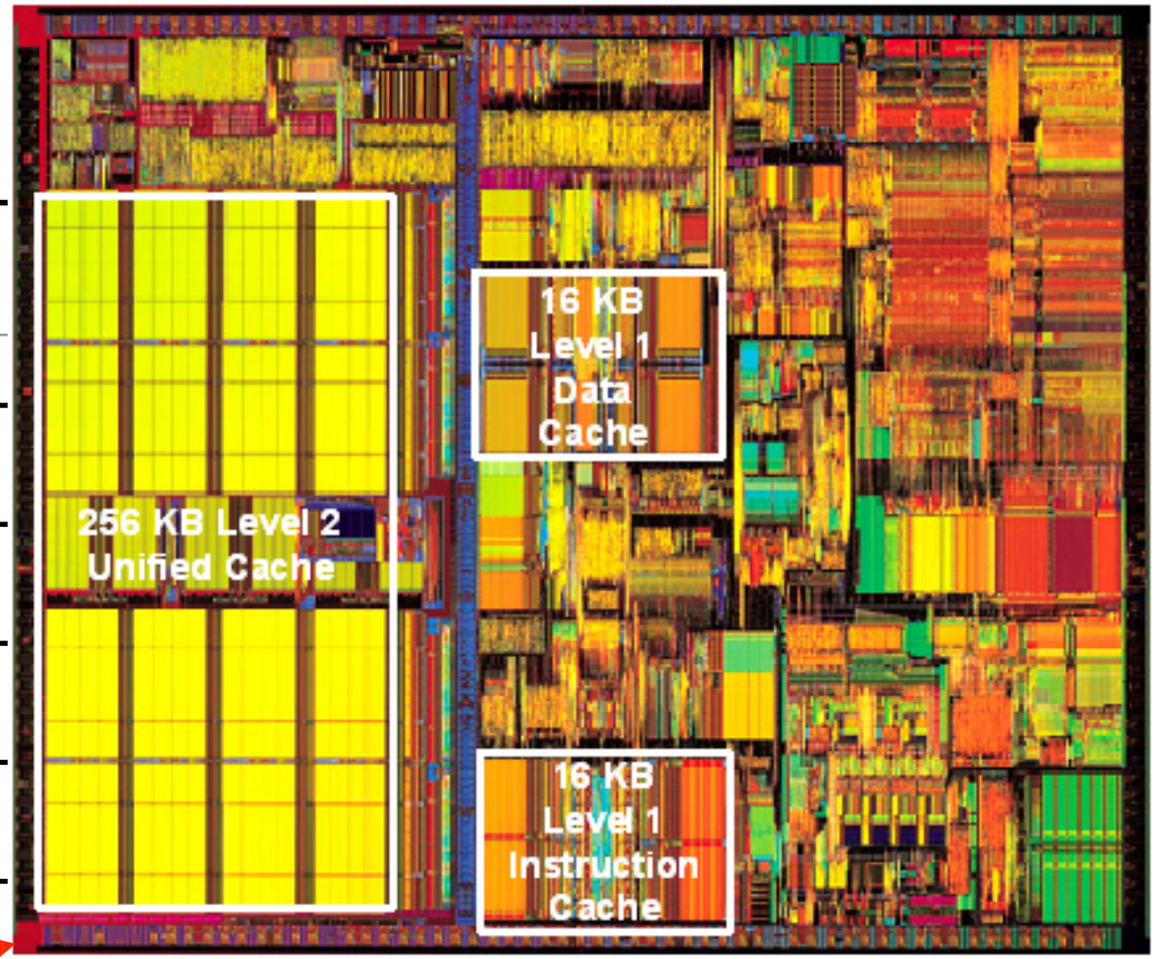


Evolution of Cache Architectures

Processor	Year	Freq. (MHz)	L1 Data	L1 Instr.	L2 Cache
80386	1985	16-25	none	none	none
80486	1989	25-100	8KB unified		none on chip
Pentium	1993	60-300	8KB	8KB	none on chip
Pentium Pro	1995	150-200	8KB	8KB	256KB-1MB in MCM
Pentium II	1997	233-450	16KB	16KB	256-512KB on cartridge
Pentium III	1999	450-1400	16KB	16KB	256-512KB on chip
Pentium 4	2001	1400-3730	8-16KB	12k op trace cache	256KB-2MB on chip
Pentium M	2003	900-2130	32KB	32KB	1-2MB on chip
Core Duo	2005	1500-2160	32KB/core	32KB/core	2MB shared on chip

Evolution of Cache Architec

Processor	Year	Freq. (MHz)			
80386	1985	16-25			
80486	1989	25-100			
Pentium	1993	60-300			
Pentium Pro	1995	150-200			
Pentium II	1997	233-450	16KB	16KB	256-512KB on cartridge
Pentium III	1999	450-1400	16KB	16KB	256-512KB on chip
Pentium 4	2001	1400-3730	8-16KB	12k op trace cache	256KB-2MB on chip
Pentium M	2003	900-2130	32KB	32KB	1-2MB on chip
Core Duo	2005	1500-2160	32KB/core	32KB/core	2MB shared on chip

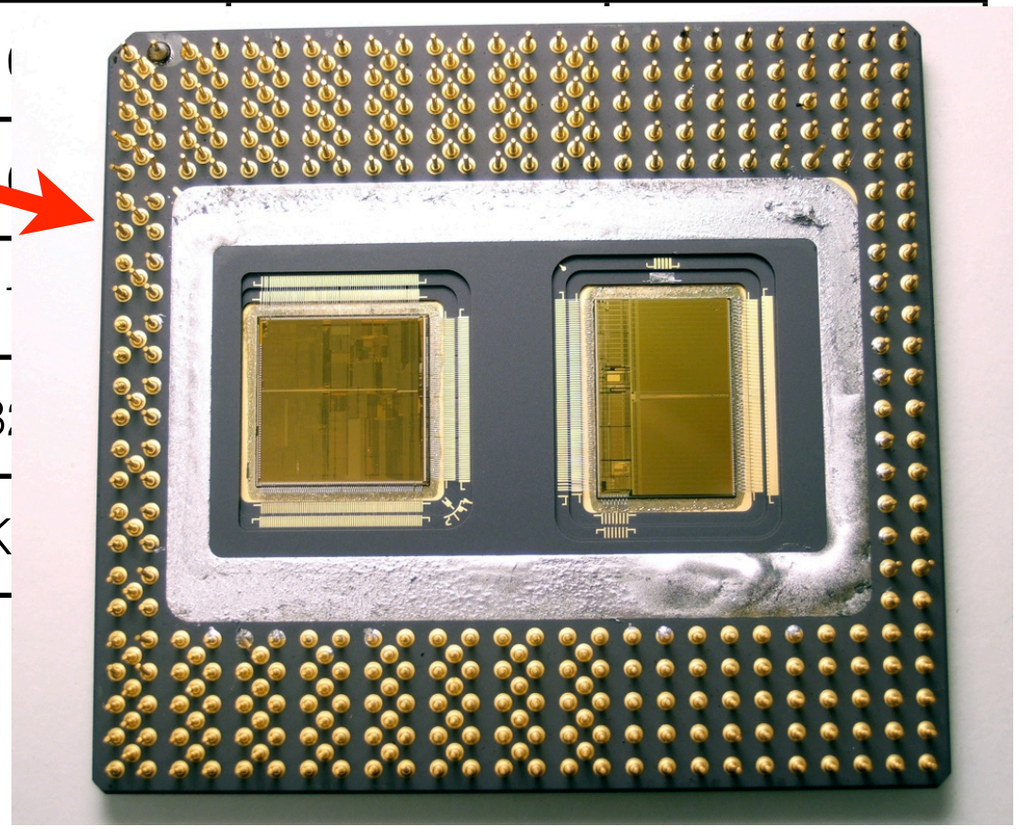


Evolution of Cache Architectures

Processor	Year	Freq. (MHz)	L1 Data	L1 Instr.	L2 Cache
80386	1985	16-25	none	none	none
80486	1989	25-100	8KB unified		none on chip
Pentium	1993	60-300	8KB	8KB	none on chip
Pentium Pro	1995	150-200	8KB	8KB	256KB-1MB in MCM
Pentium II	1997	233-450	16KB	16KB	256-512KB on cartridge
Pentium III	1999	450-1400	16KB	16KB	256-512KB on chip
Pentium 4	2001	1400-3730	8-16KB	12k op trace cache	256KB-2MB on chip
Pentium M	2003	900-2130	32KB	32KB	1-2MB on chip
Core Duo	2005	1500-2160	32KB/core	32KB/core	2MB shared on chip

Evolution of Cache Architectures

Processor	Year	Freq. (MHz)	L1 Data	L1 Instr.	L2 Cache
80386	1985	16-25	none	none	none
80486	1989	25-100	8KB unified		none on chip
Pentium	1993	60-300	8KB	8KB	none on chip
Pentium Pro	1995	150-200	8KB	8KB	256KB-1MB in MCM
Pentium II	1997	233-450	1(		
Pentium III	1999	450-1400	1(		
Pentium 4	2001	1400-3730	8-1		
Pentium M	2003	900-2130	3		
Core Duo	2005	1500-2160	32K		

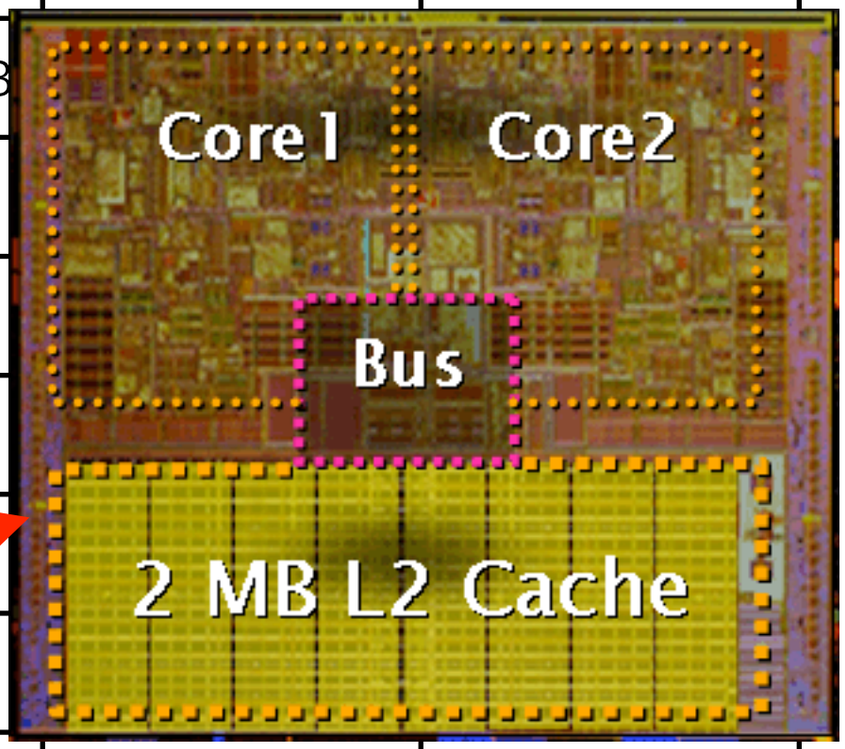


Evolution of Cache Architectures

Processor	Year	Freq. (MHz)	L1 Data	L1 Instr.	L2 Cache
80386	1985	16-25	none	none	none
80486	1989	25-100	8KB unified		none on chip
Pentium	1993	60-300	8KB	8KB	none on chip
Pentium Pro	1995	150-200	8KB	8KB	256KB-1MB in MCM
Pentium II	1997	233-450	16KB	16KB	256-512KB on cartridge
Pentium III	1999	450-1400	16KB	16KB	256-512KB on chip
Pentium 4	2001	1400-3730	8-16KB	12k op trace cache	256KB-2MB on chip
Pentium M	2003	900-2130	32KB	32KB	1-2MB on chip
Core Duo	2005	1500-2160	32KB/core	32KB/core	2MB shared on chip

Evolution of Cache Architectures

Processor	Year	Freq. (MHz)	L1 Data	L1 Instr.	L2 Cache
80386	1985	16-25	none	none	none
80486	1989	25-100	8KB		
Pentium	1993	60-300	8KB		
Pentium Pro	1995	150-200	8KB		
Pentium II	1997	233-450	16KB		
Pentium III	1999	450-1400	16KB		
Pentium 4	2001	1400-3730	8-16KB		
Pentium M	2003	900-2130	32KB	32KB	1-2MB on chip
Core Duo	2005	1500-2160	32KB/core	32KB/core	2MB shared on chip



Evolution of Cache Architectures

Processor	Year	Freq. (MHz)	L1 Data	L1 Instr.	L2 Cache
80386	1985	16-25	none	none	none
80486	1989	25-100	8KB unified		none on chip
Pentium	1993	60-300	8KB	8KB	none on chip
Pentium Pro	1995	150-200	8KB	8KB	256KB-1MB in MCM
Pentium II	1997	233-450	16KB	16KB	256-512KB on cartridge
Pentium III	1999	450-1400	16KB	16KB	256-512KB on chip
Pentium 4	2001	1400-3730	8-16KB	12k op trace cache	256KB-2MB on chip
Pentium M	2003	900-2130	32KB	32KB	1-2MB on chip
Core Duo	2005	1500-2160	32KB/core	32KB/core	2MB shared on chip