

**Computer Microprocessor
Architecture & Programming
HCA1109**

Cache Memory

Characteristics

- Location
- Unit of transfer
- Access method
- Performance
- Physical type
- Physical Characteristics

Location

- CPU
- Internal
- External

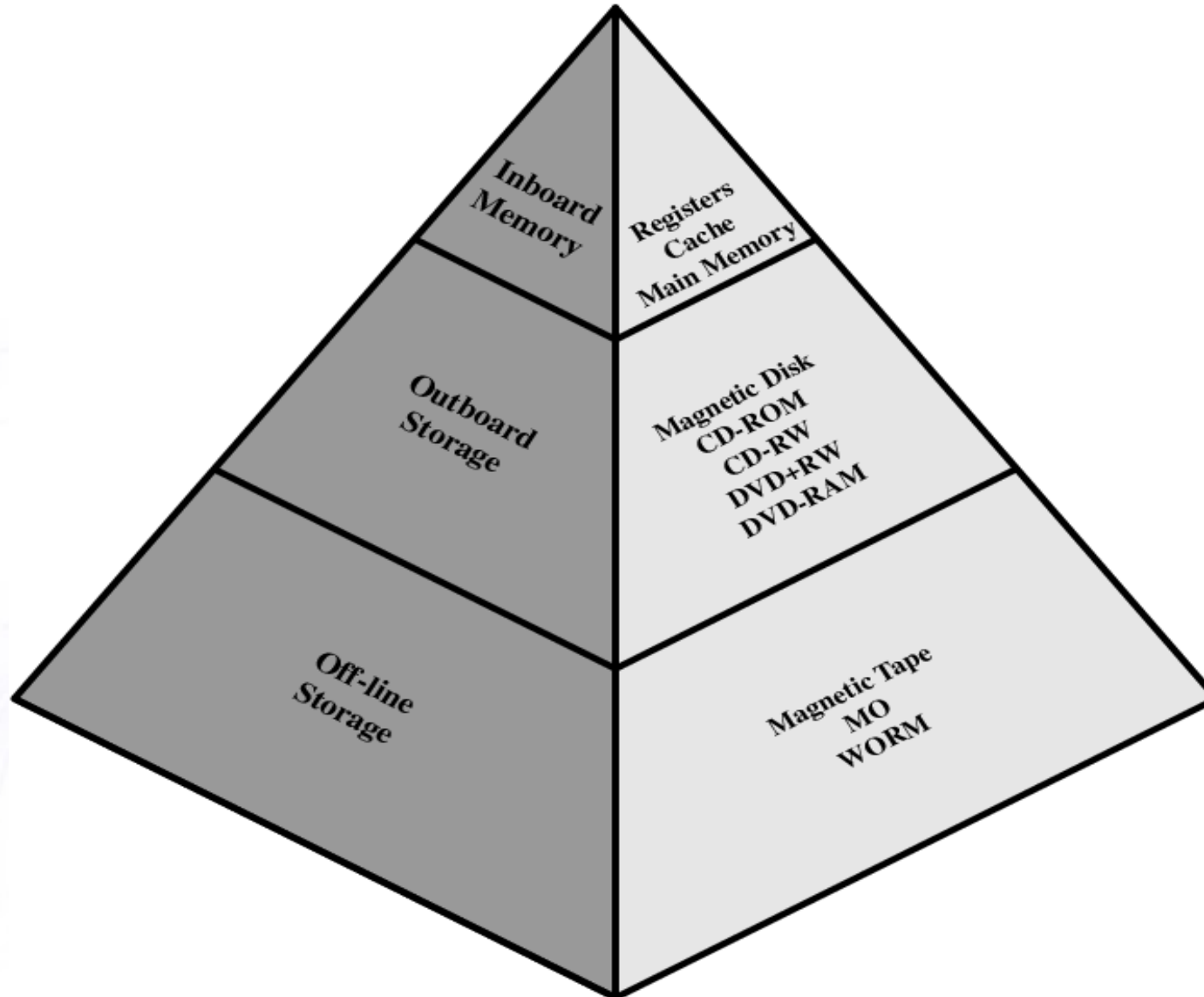
Unit of Transfer

- Internal
 - Usually governed by data bus width
- External
 - Usually a block which is much larger than a word
- Addressable unit
 - Smallest location which can be uniquely addressed
 - Word internally
 - Cluster on disks

Access Methods

- **Sequential**
 - Start at the beginning and read through in order
 - Access time depends on location of data and previous location e.g. Tape
- **Random**
 - Individual addresses identify locations exactly
 - Access time is independent of location or previous access e.g. RAM
- **Direct**
 - Individual blocks have unique address
 - Access is by jumping to vicinity plus sequential search
 - Access time depends on location and previous location e.g. Hard Disk
- **Associative**
 - Data is located by a comparison with contents of a portion of the store
 - Access time is independent of location or previous access e.g. Cache

Memory Hierarchy - Diagram



Performance

- Access time
 - Time between presenting the address and getting the valid data
- Memory Cycle time
 - Time may be required for the memory to “recover” before next access
 - Cycle time is access + recovery
- Transfer Rate
 - Rate at which data can be moved

Physical Types

- Semiconductor
 - RAM
- Magnetic
 - Disk & Tape
- Optical
 - CD & DVD
- Others
 - Bubble
 - Hologram

Physical Characteristics

- Decay
- Volatility
- Erasable
- Power consumption

Hierarchy List

- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

So you want fast?

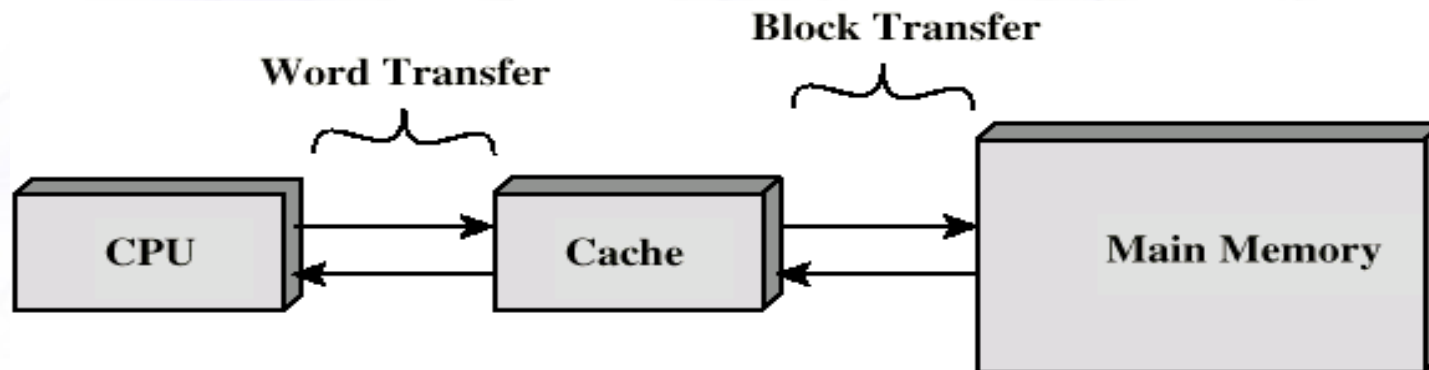
- It is possible to build a computer which uses only static RAM (see later)
- This would be very fast
- This would need no cache
 - How can you cache cache?
- This would cost a very large amount

Locality of Reference

- During the course of the execution of a program, memory references tend to cluster
- e.g. loops, arrays,...

Cache

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module



Cache operation - Overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

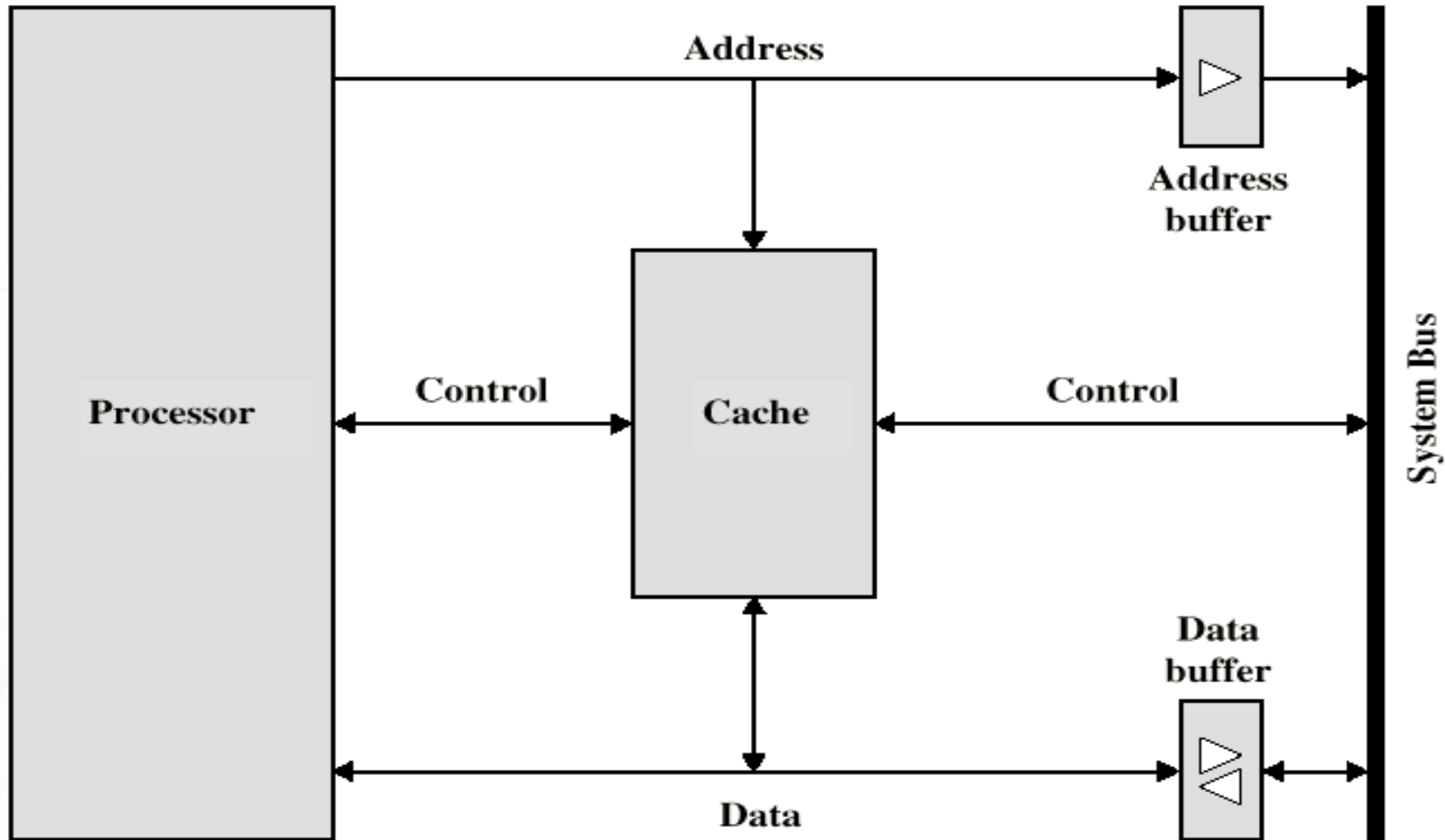
Cache Design

- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Size does matter

- Cost
 - More cache is expensive
- Speed
 - More cache is faster (up to a point)
 - Checking cache for data takes time

Typical Cache Organization



Mapping Function

- Cache of 64 kilobyte
- Cache block of 4 bytes
 - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16 Megabytes main memory
- 24 bit address
 - ($2^{24} = 16\text{M}$)

Direct Mapping

- Each block of main memory maps to only one cache line
 - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of $s-r$ (most significant)

Direct Mapping -Address Structure

| Tag s-r | Line or Slot r | Word w |
|---------|----------------|--------|
| 8 | 14 | 2 |

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
 - 8 bit tag (=22-14)
 - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

Direct Mapping - Cache Line Table

- Cache line held Main Memory blocks

-

0

0, m, 2m, 3m...2 -m

-

1

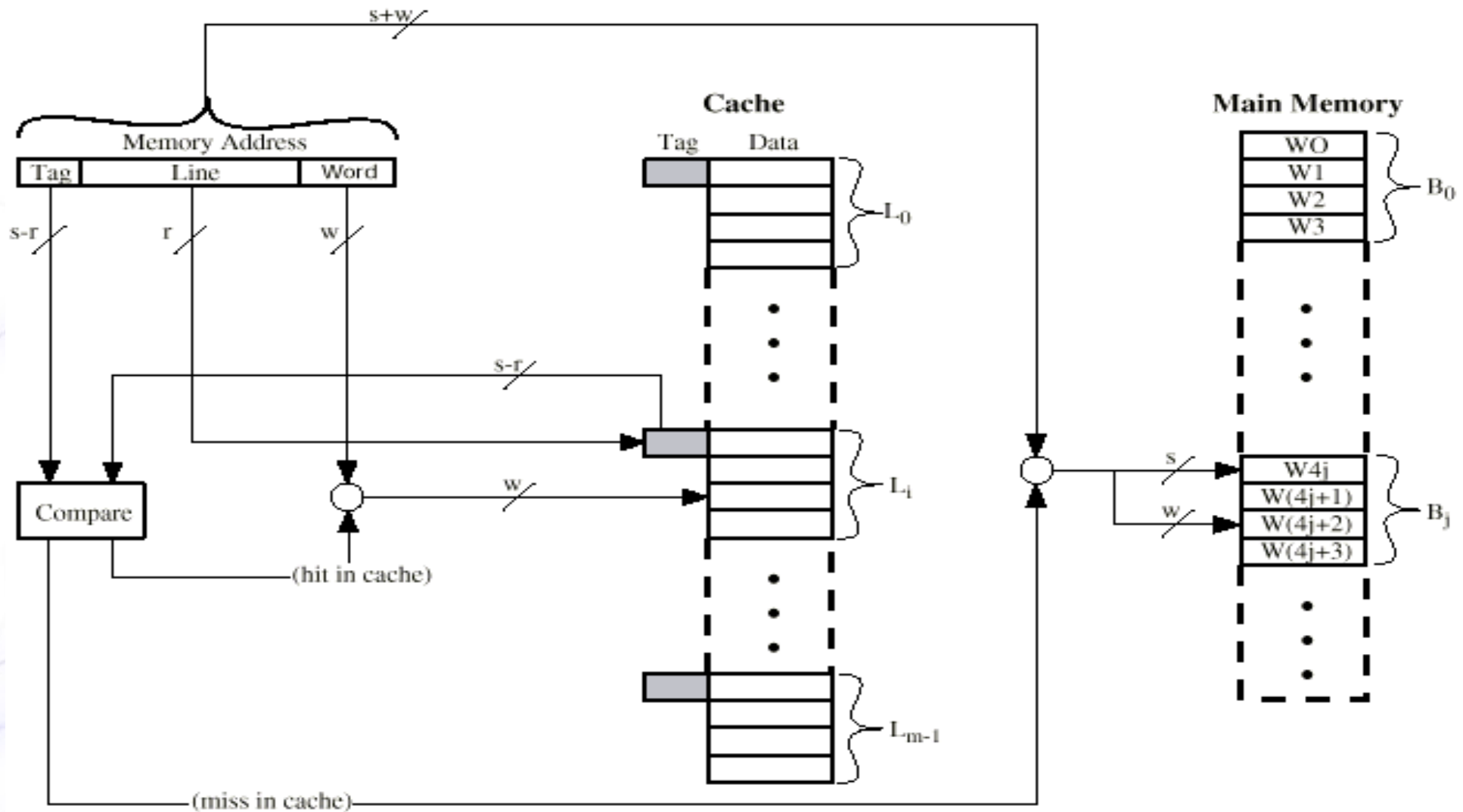
1, m+1, 2m+1...2 -m+1

-

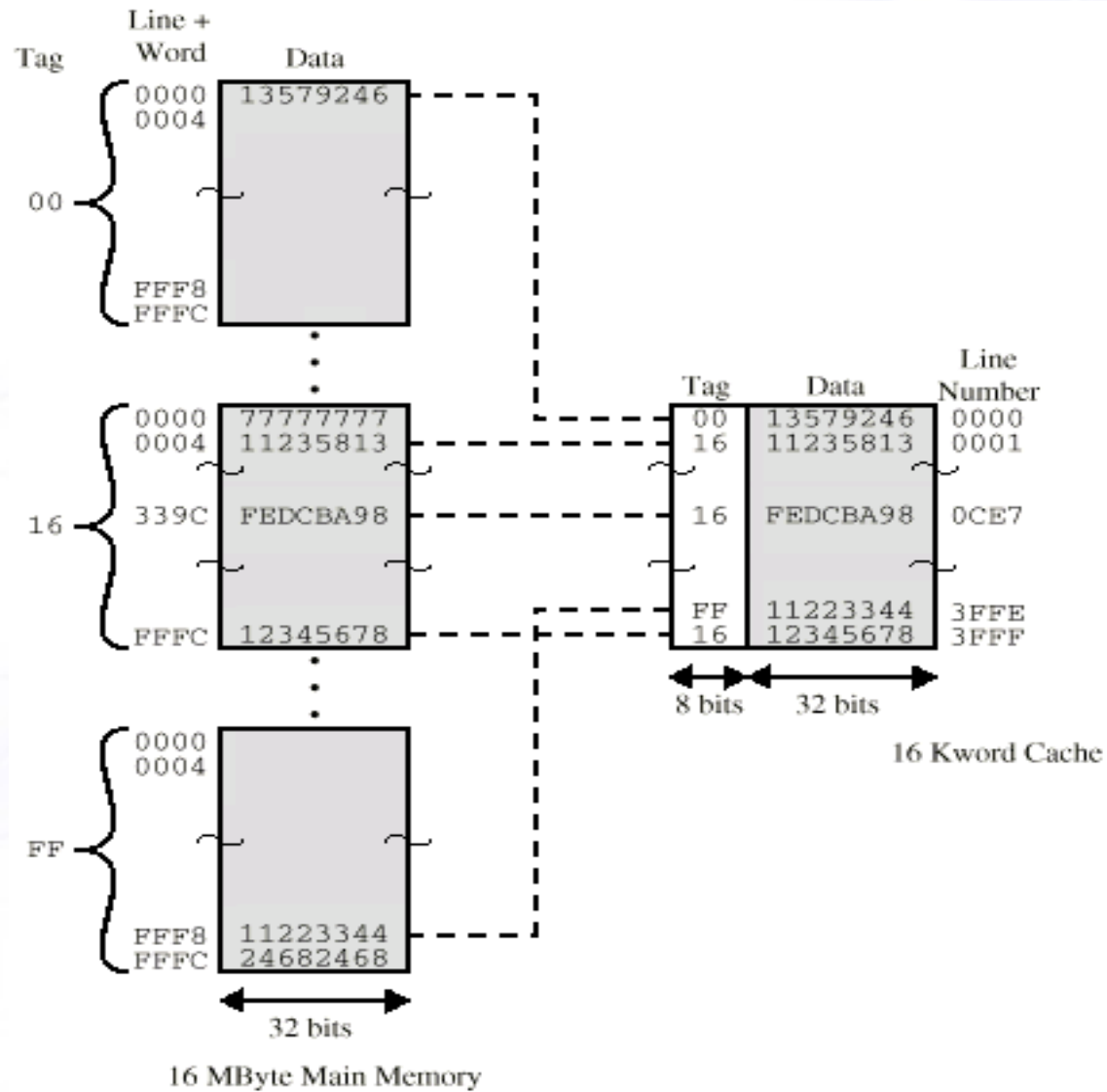
m-1

m-1, 2m-1, 3m-1...2 -1

Direct Mapping Cache Organization



Direct Mapping Example



Direct Mapping Summary

- Address length = $(s + w)$ bits
Number of addressable units = 2 words or bytes

-

Block size = line size = 2 words or bytes

-

Number of blocks in main memory = $2/2 = 2$

-

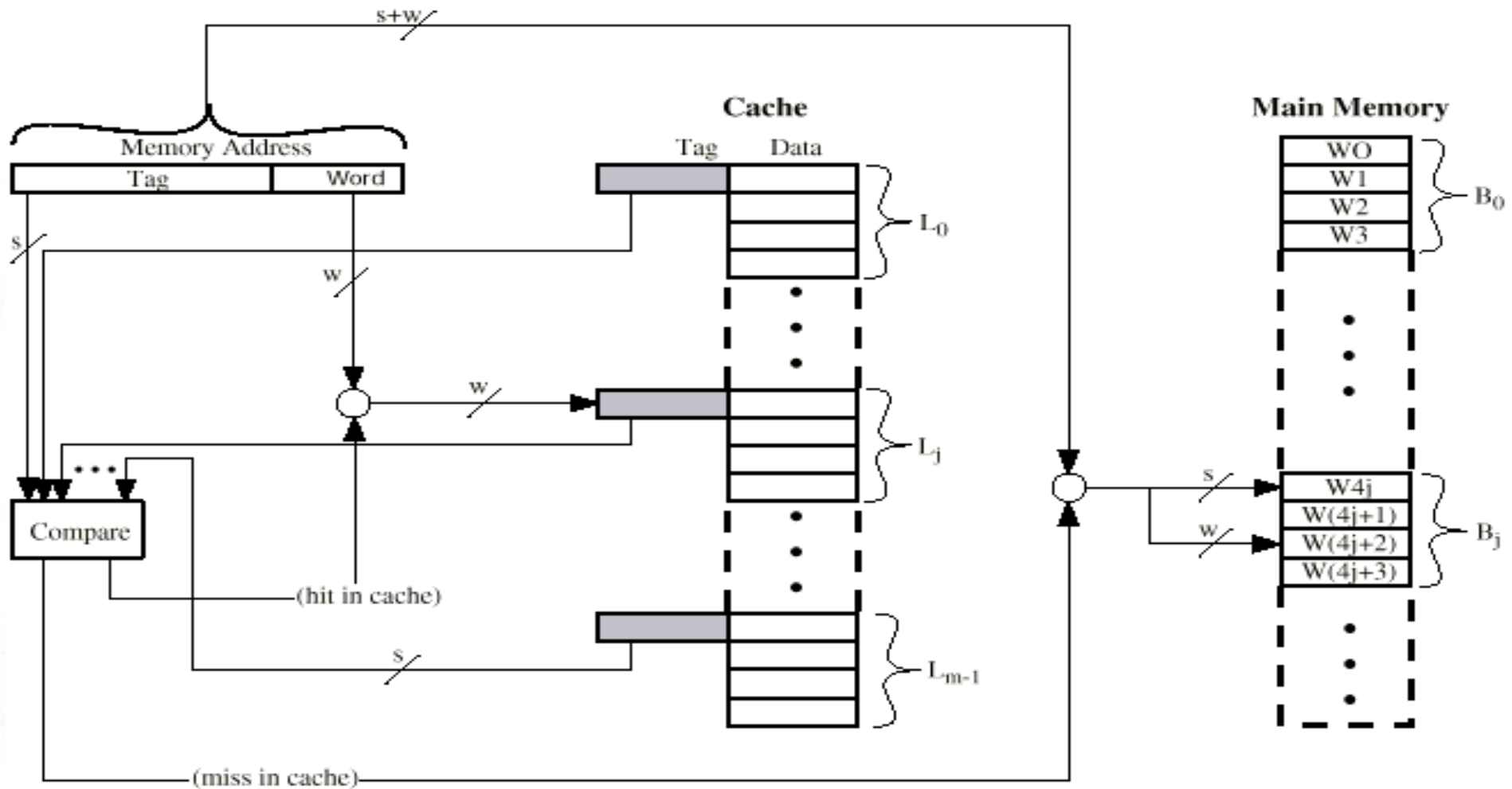
Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
 - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

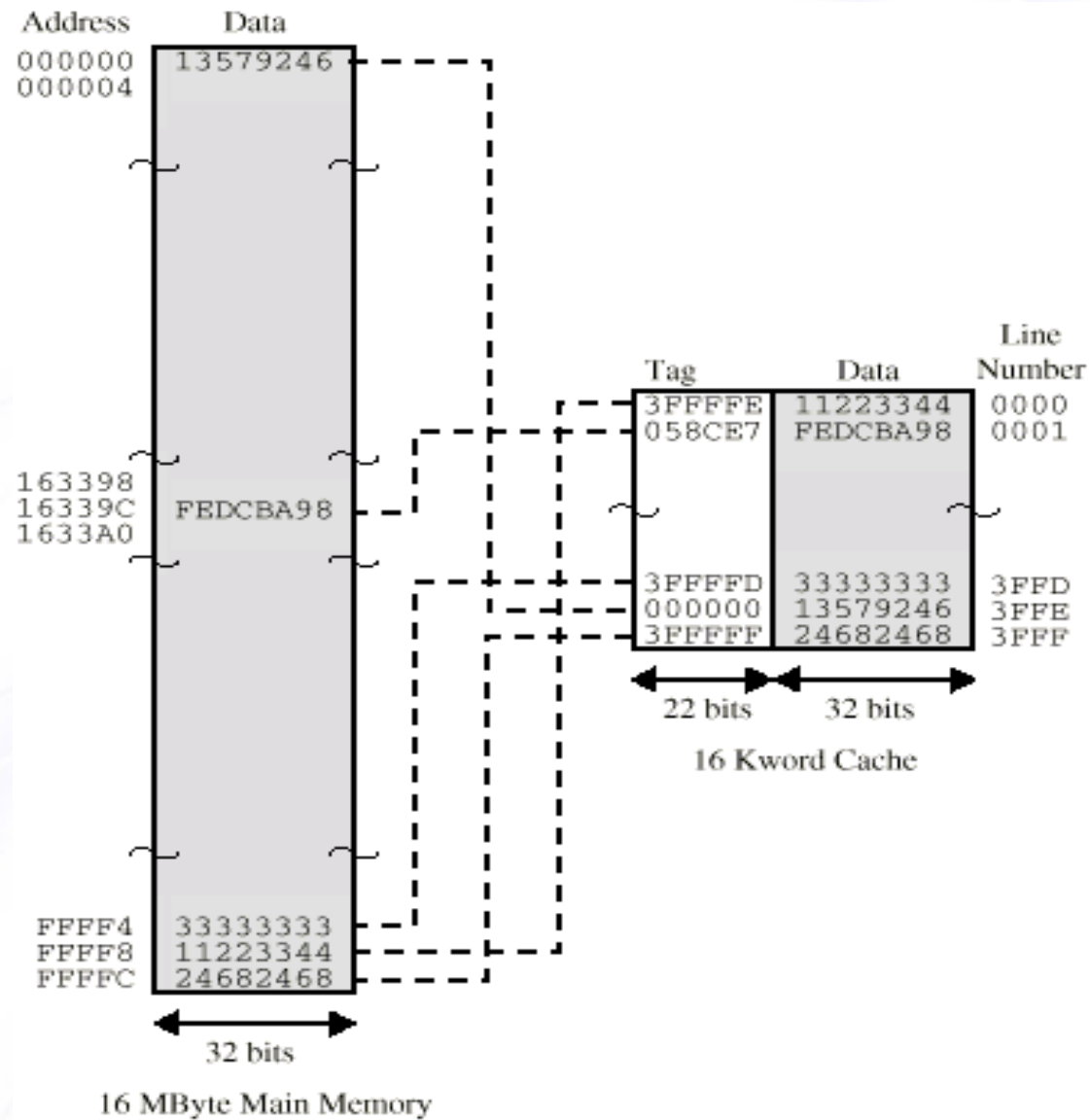
Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

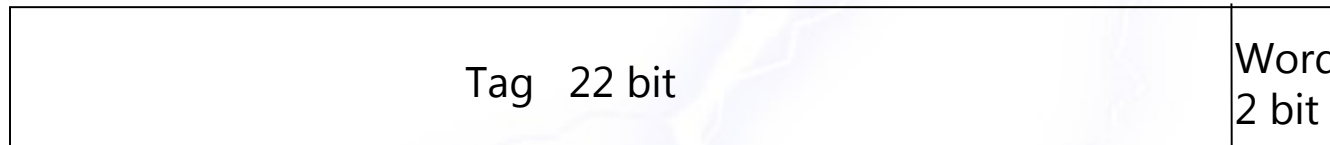
Fully Associative Cache Organization



Associative Mapping Example



Associative Mapping - Address Structure



- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block
- e.g.

| – Address | Tag | Data | Cache line |
|-----------|--------|----------|------------|
| FFFFFC | FFFFFC | 24682468 | 3FFF |

Associative Mapping Summary

- Address length = $(s + w)$ bits
- Number of addressable units = 2 words or bytes
- Block size = line size = 2 words or bytes
- Number of blocks in main memory = $2/2 = 2$
-
- Number of lines in cache = undetermined
- Size of tag = s bits

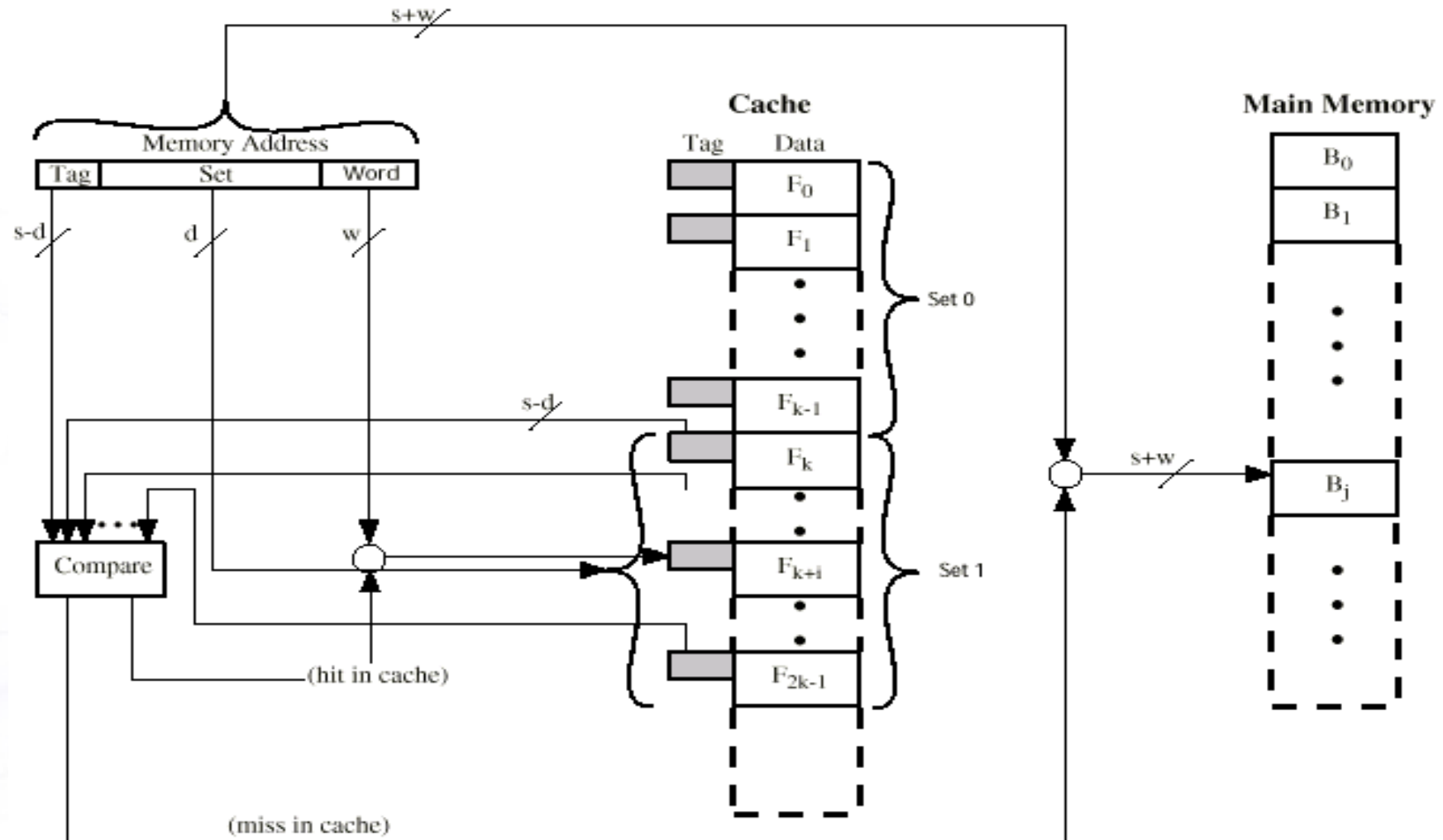
Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
 - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
 - 2 way associative mapping
 - A given block can be in one of 2 lines in only one set

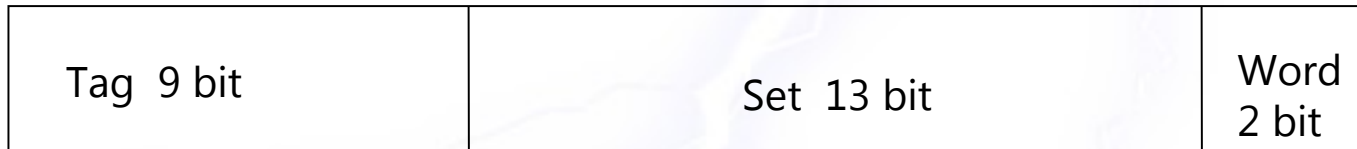
Set Associative Mapping - Example

- 13 bit set number
- Block number in main memory is modulo 2
- 000000, 00A000, 00B000, 00C000 ...
map to same set

Two Way Set Associative Cache Organization



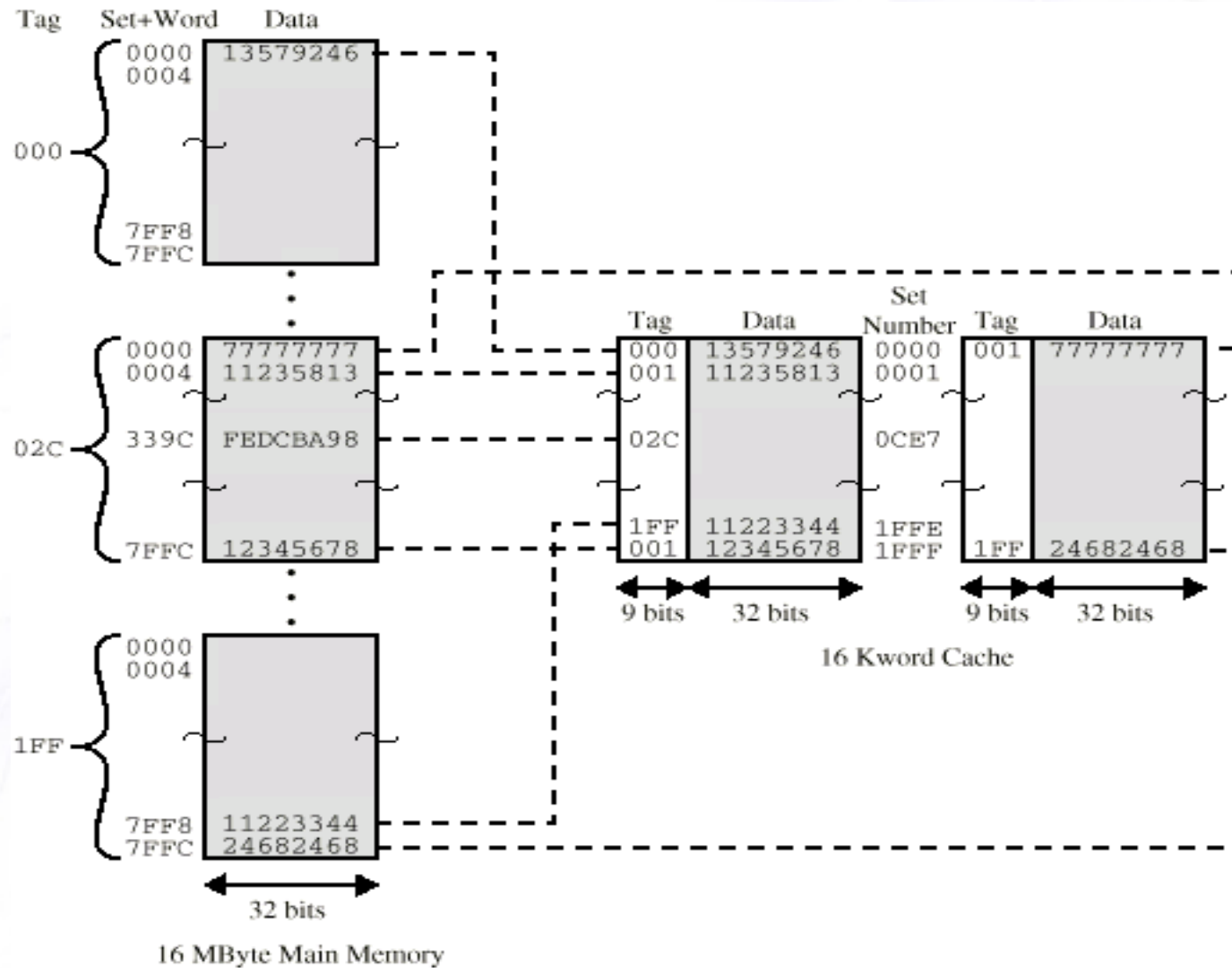
Set Associative Mapping Address Structure



- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit
- e.g

| Address | Tag | Data | Set number |
|----------|-----|----------|------------|
| 1FF 7FFC | 1FF | 12345678 | 1FFF |
| 001 7FFC | 001 | 11223344 | 1FFF |

Two Way Set Associative Mapping Example



Set Associative Mapping Summary

- Address length = $(s + w)$ bits
- Number of addressable units = 2 words or bytes
- Block size = line size = 2 words or bytes
Number of blocks in main memory = 2
-
- Number of lines in set = k
Number of sets = $v = 2$
-

Replacement Algorithms - Direct mapping

- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms - Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
 - e.g. in 2 way set associative
 - Which of the 2 block is LRU?
- First in first out (FIFO)
 - replace block that has been in cache longest
- Least frequently used
 - replace block which has had fewest hits
- Random

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes
- Remember bogus write through caches!

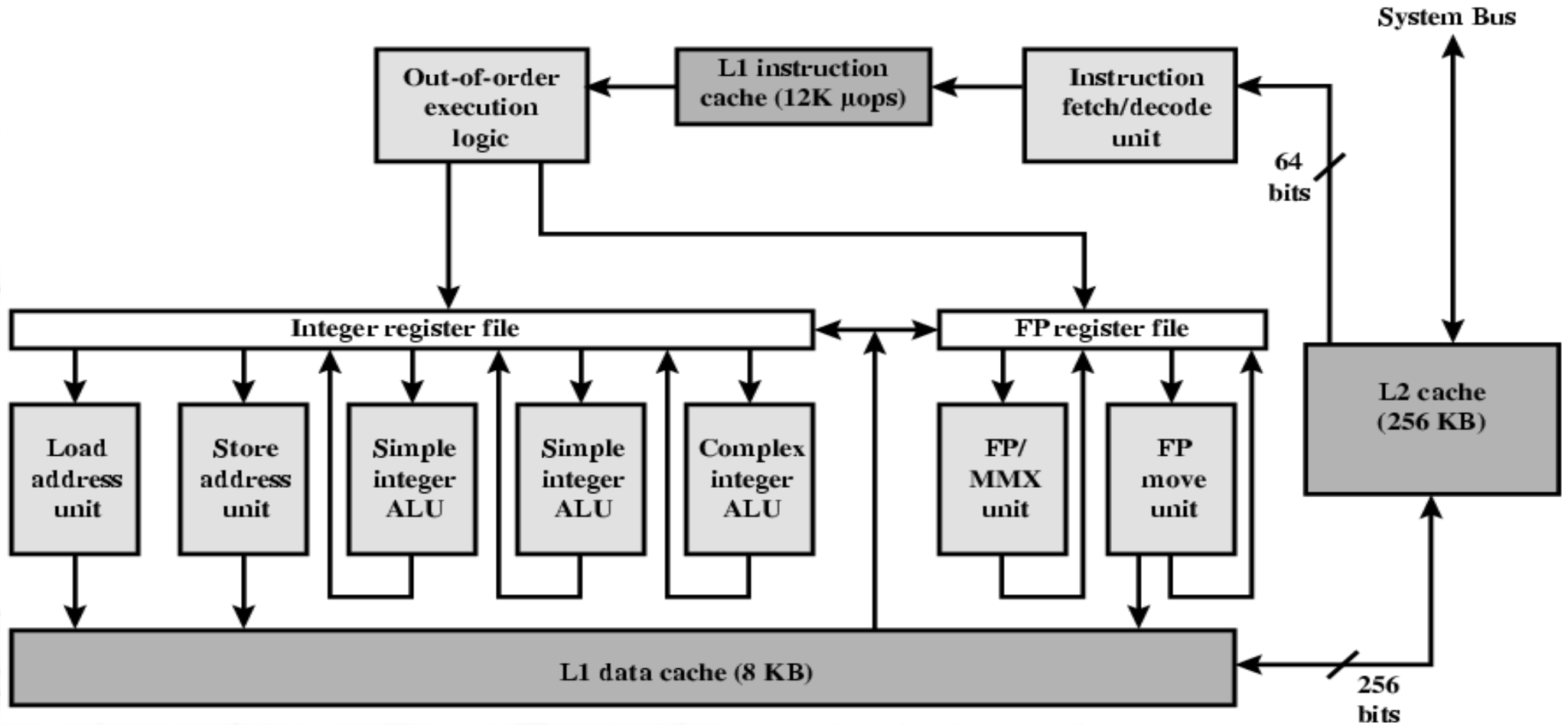
Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

Pentium 4 Cache

- 80386 – no on chip cache
- 80486 – 8k using 16 byte lines and four way set associative organization
- Pentium (all versions) – two on chip L1 caches
 - Data & instructions
- Pentium 4 – L1 caches
 - 8k bytes
 - 64 byte lines
 - four way set associative
- L2 cache
 - Feeding both L1 caches
 - 256k
 - 128 byte lines
 - 8 way set associative

Pentium 4 Diagram (Simplified)



Pentium 4 Core Processor

- Fetch/Decode Unit
 - Fetches instructions from L2 cache
 - Decode into micro-ops
 - Store micro-ops in L1 cache
- Out of order execution logic
 - Schedules micro-ops
 - Based on data dependence and resources
 - May speculatively execute
- Execution units
 - Execute micro-ops
 - Data from L1 cache
 - Results in registers
- Memory subsystem
 - L2 cache and systems bus

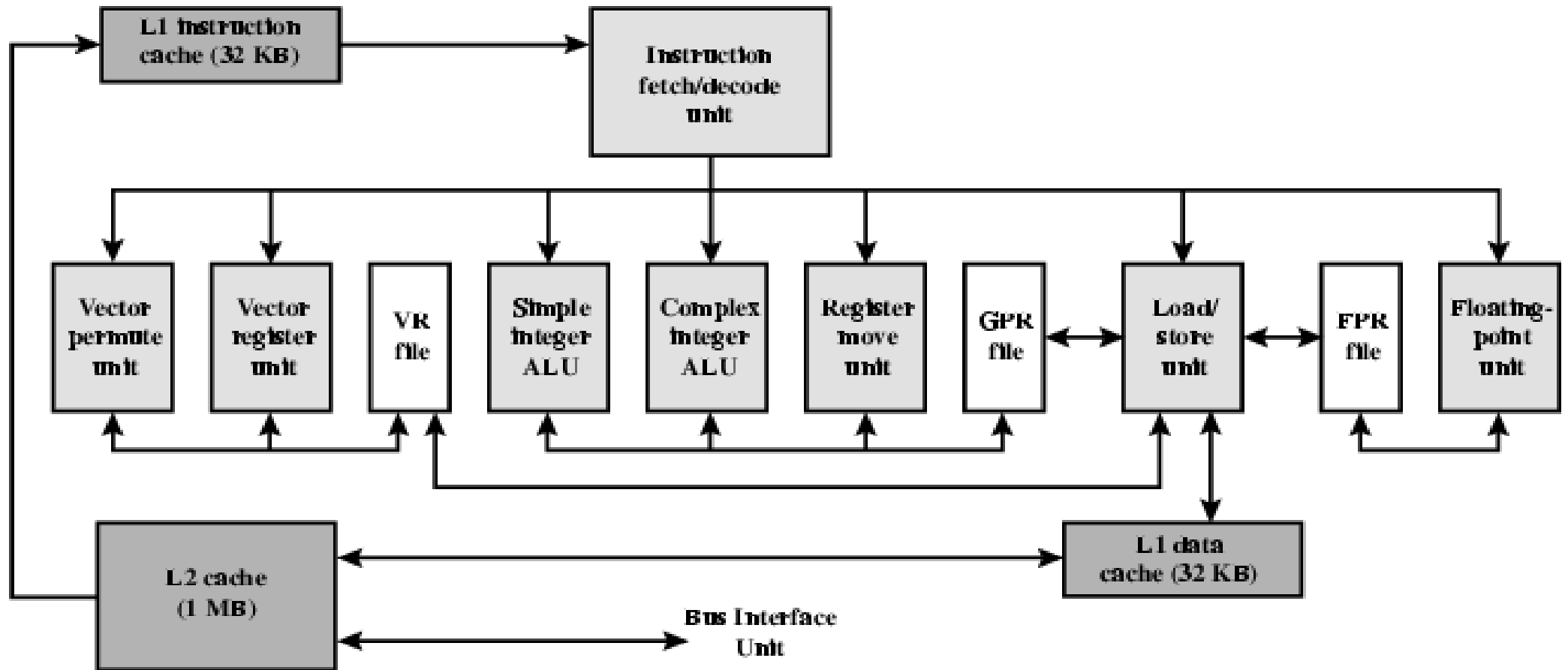
Pentium 4 Design Reasoning

- Decodes instructions into RISC like micro-ops before L1 cache
- Micro-ops fixed length
 - Superscalar pipelining and scheduling
- Pentium instructions long & complex
- Performance improved by separating decoding from scheduling & pipelining
- Data cache is write back
 - Can be configured to write through
- L1 cache controlled by 2 bits in register
 - CD = cache disable
 - NW = not write through
 - 2 instructions to invalidate (flush) cache and write back then invalidate

Power PC Cache Organization

- 601 – single 32 kB 8-way set associative
- 603 – 16 kB (2 x 8 kB) 2-way set associative
- 604 – 32 kB
- 610 – 64 kB
- G3 & G4
 - 64 kB L1 cache
 - 8-way set associative
 - 256 kB, 512 kB or 1MB L2 cache
 - two way set associative

PowerPC G4



Comparison of Cache Sizes

| Processor | Type | Year of Introduction | L1 cache ^a | L2 cache | L3 cache |
|-----------------------|-----------------------------------|----------------------|-----------------------|----------------|----------|
| IBM 360/85 | Mainframe | 1968 | 16 to 32 KB | — | — |
| PDP-11/70 | Minicomputer | 1975 | 1 KB | — | — |
| VAX 11/780 | Minicomputer | 1978 | 16 KB | — | — |
| IBM 3033 | Mainframe | 1978 | 64 KB | — | — |
| IBM 3090 | Mainframe | 1985 | 128 to 256 KB | — | — |
| Intel 80486 | PC | 1989 | 8 KB | — | — |
| Pentium | PC | 1993 | 8 KB/8 KB | 256 to 512 KB | — |
| PowerPC 601 | PC | 1993 | 32 KB | — | — |
| PowerPC 620 | PC | 1996 | 32 KB/32 KB | — | — |
| PowerPC G4 | PC/server | 1999 | 32 KB/32 KB | 256 KB to 1 MB | 2 MB |
| IBM S/390 G4 | Mainframe | 1997 | 32 KB | 256 KB | 2 MB |
| IBM S/390 G6 | Mainframe | 1999 | 256 KB | 8 MB | — |
| Pentium 4 | PC/server | 2000 | 8 KB/8 KB | 256 KB | — |
| IBM SP | High-end server/ supercomputer | 2000 | 64 KB/32 KB | 8 MB | — |
| CRAY MTA ^b | Supercomputer | 2000 | 8 KB | 2 MB | — |
| Itanium | PC/server | 2001 | 16 KB/16 KB | 96 KB | 4 MB |
| SGI Origin 2001 | High-end server | 2001 | 32 KB/32 KB | 4 MB | — |

^a Two values separated by a slash refer to instruction and data caches

^b Both caches are instruction only; no data caches