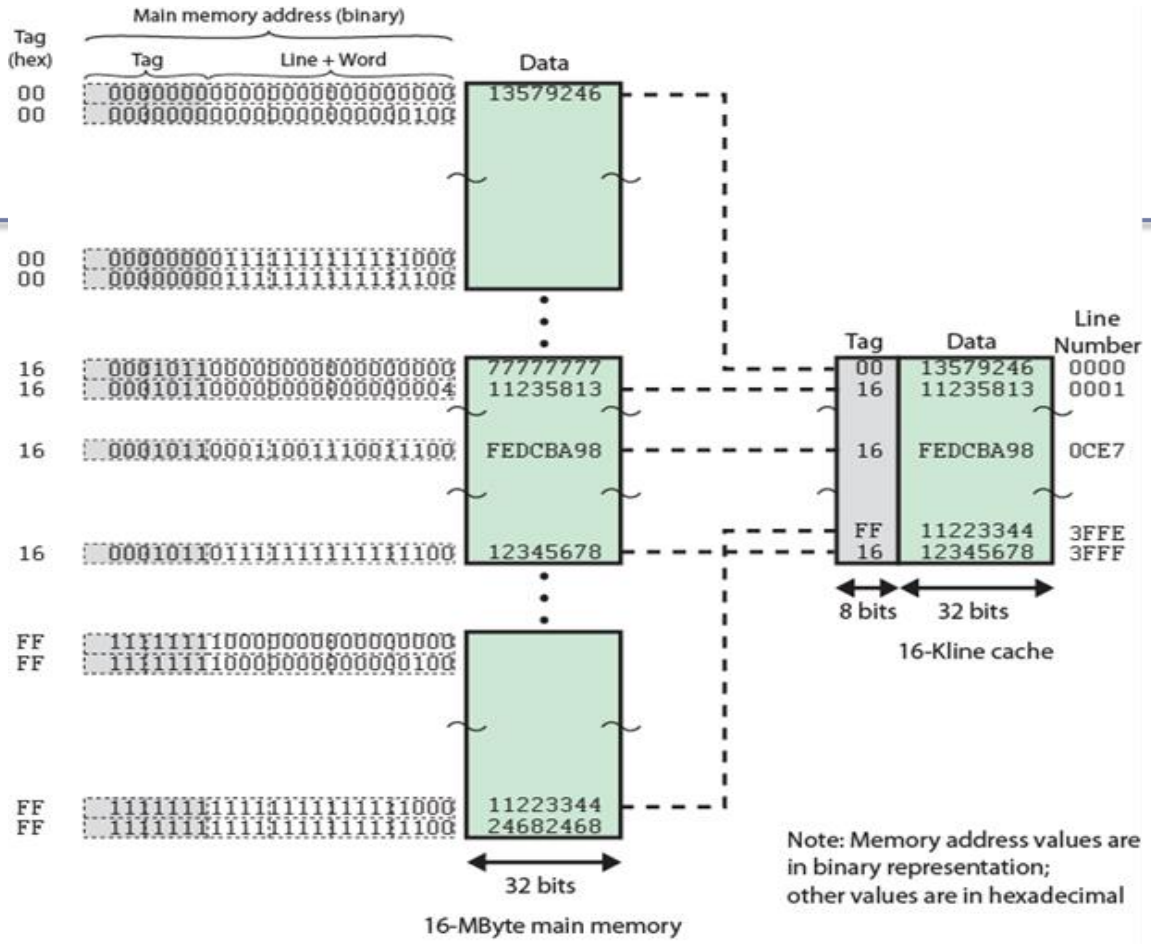
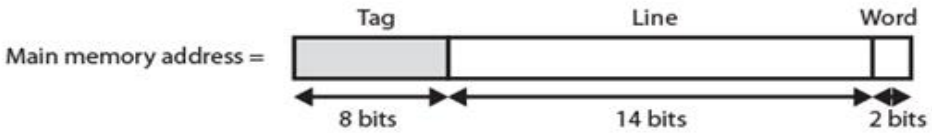
CS430 Computer Architecture

Spring 2016

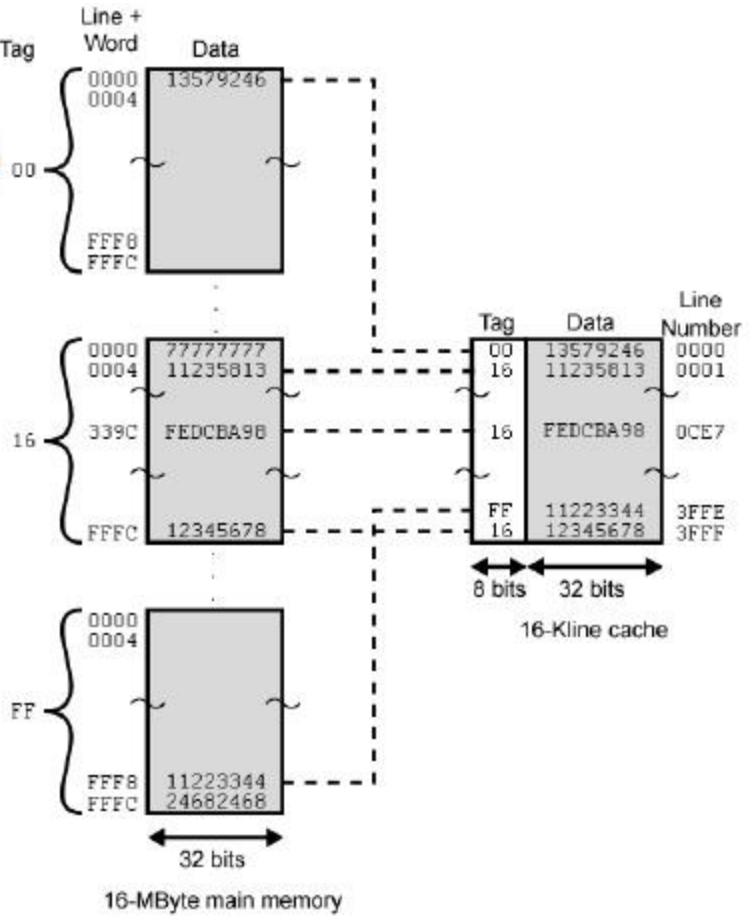
Direct Mapping

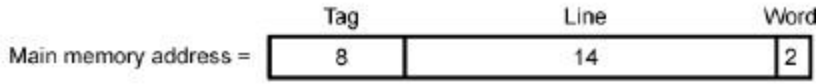




Direct Mapping

Direct Mapping Example





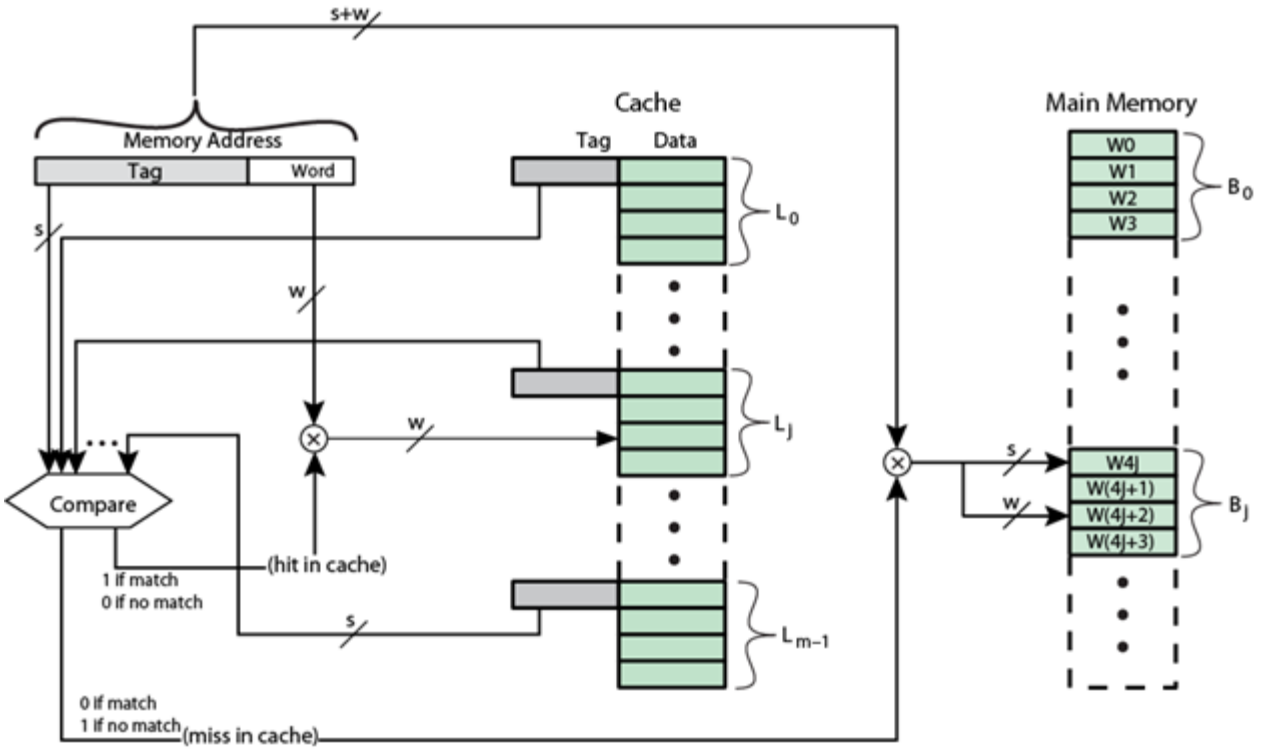
Direct Mapping Summary

- Advantages of direct mapping:
  - The technique is simple
  - The mapping scheme is easy to implement
- Disadvantage of direct mapping:
  - Each block of main memory maps to a fixed location in the cache which could lead to thrashing

Associative Mapping

- With associative mapping, any block of memory can be loaded into any line of the cache.
- A memory address is simply a tag and a word (note: there is no field for line #).
- To determine if a memory block is in the cache, each of the tags are simultaneously checked for a match.

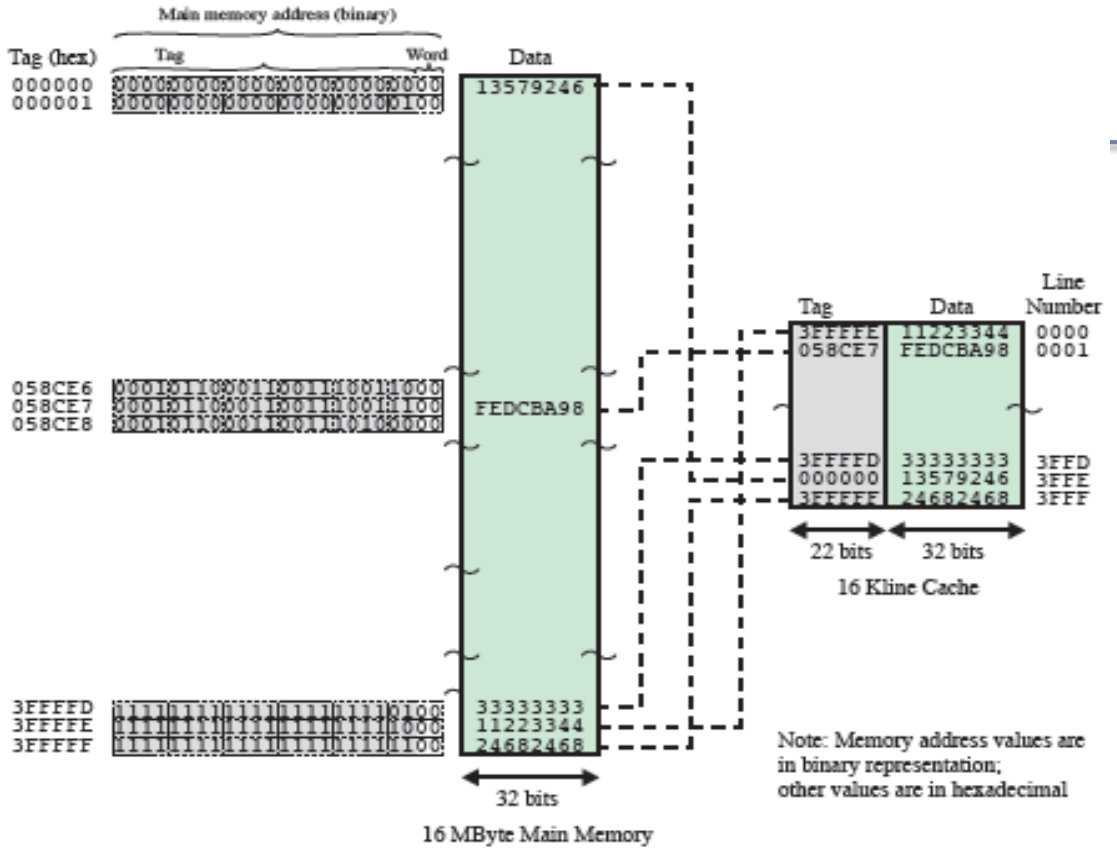
Associative Mapping



Associative Mapping

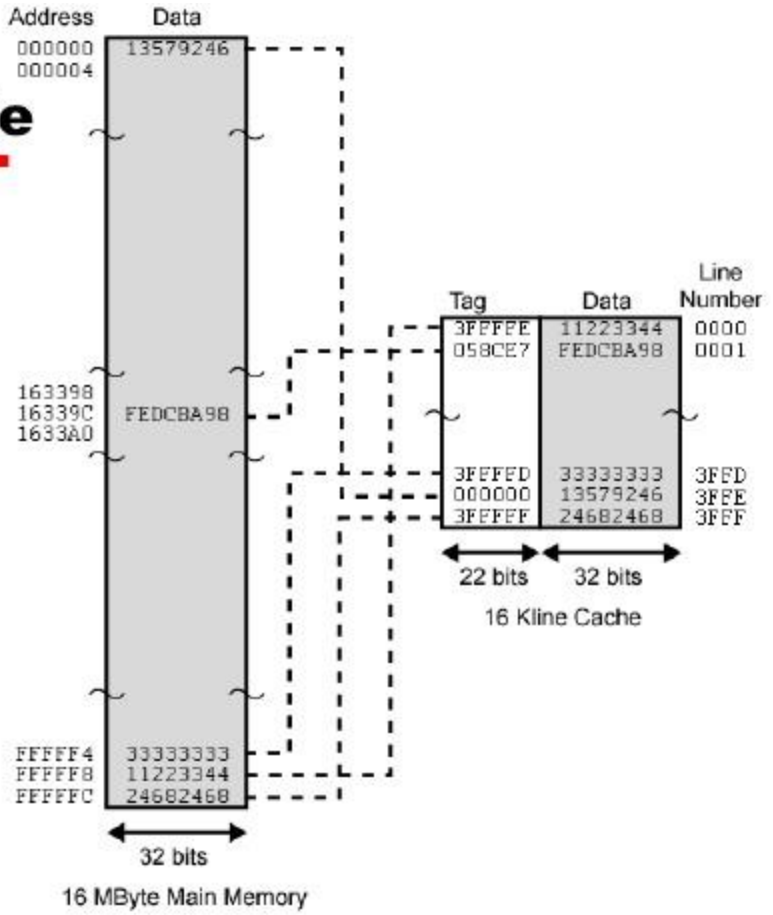
- Address Length is (s + w) bits
- Number of addressable units is 2^{s+w} bytes
- Block size = line size = 2^{w} bytes
- Number of blocks in main memory is $\frac{2^{s+w}}{2^{w}} = 2^{s}$
- Number of cache lines is undetermined
- Tag size is (s) bits

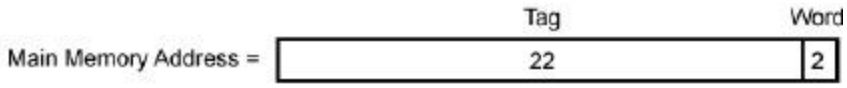
Associative Mapping Example



Associative Mapping Example

Associative Mapping Example





Associative Mapping

Advantage of associative mapping:

1. There is flexibility when mapping a block to any line of the cache

Disadvantages of associative mapping:

1. A replacement algorithm must be used to determine which line of cache to swap out
2. More space is needed for the tag field
3. The most important disadvantage is the complex circuitry needed to examine all of the tags in parallel in the cache

Set Associative Mapping

- Utilizes the strengths of direct and associative mapping while trying to reduce their disadvantges

- The cache is divided into v sets of k lines per set

k-way set associative

- Calculate each

1. Block size
2. # blocks in main MEM
3. # lines in set
4. # of sets
5. # lines in cache
6. size of the cache
7. tag size

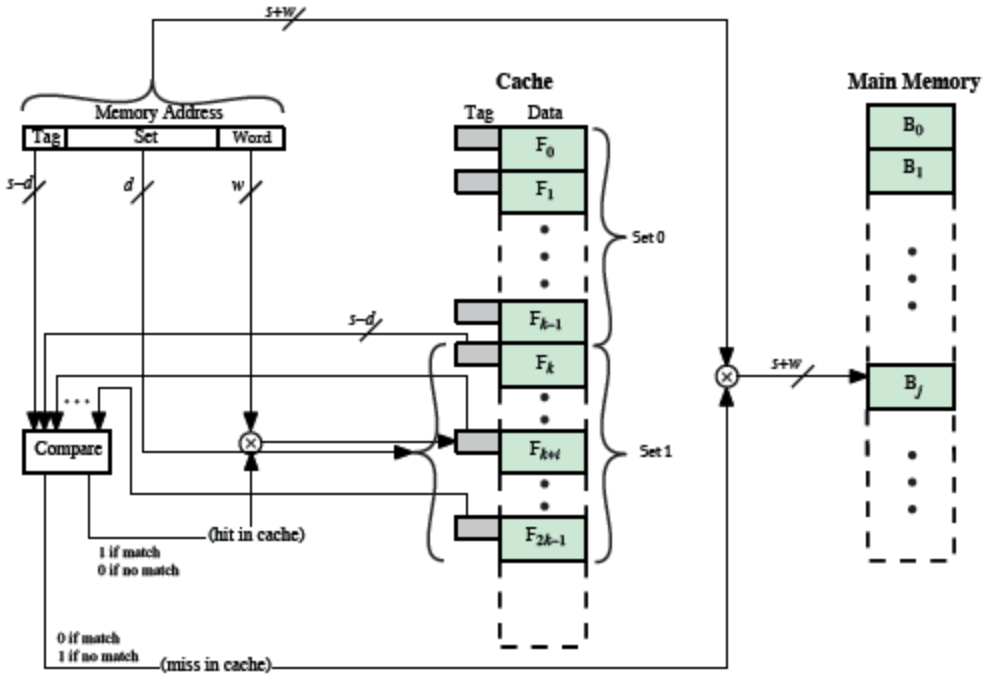


Figure 4.14 *k*-Way Set Associative Cache Organization

Two-way set associative

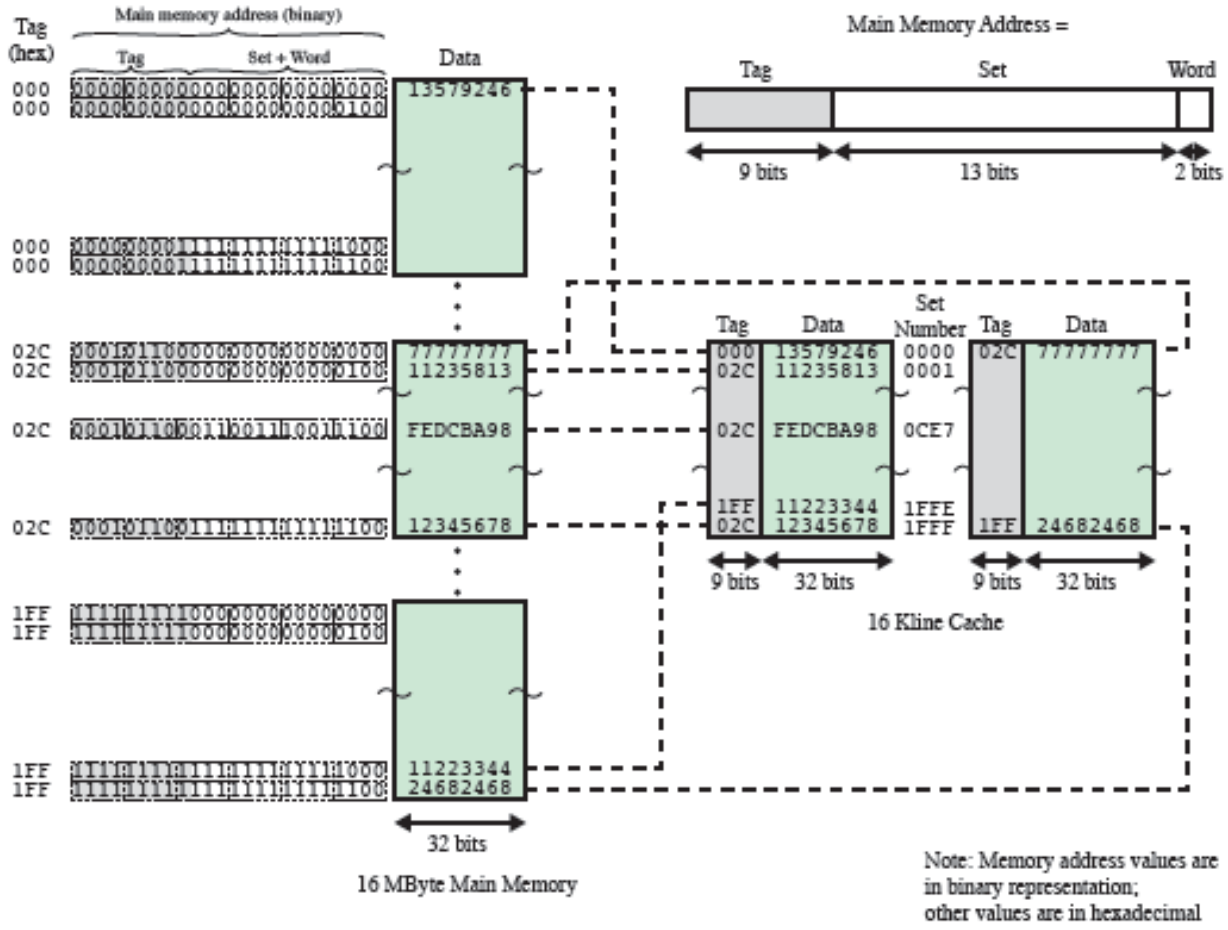
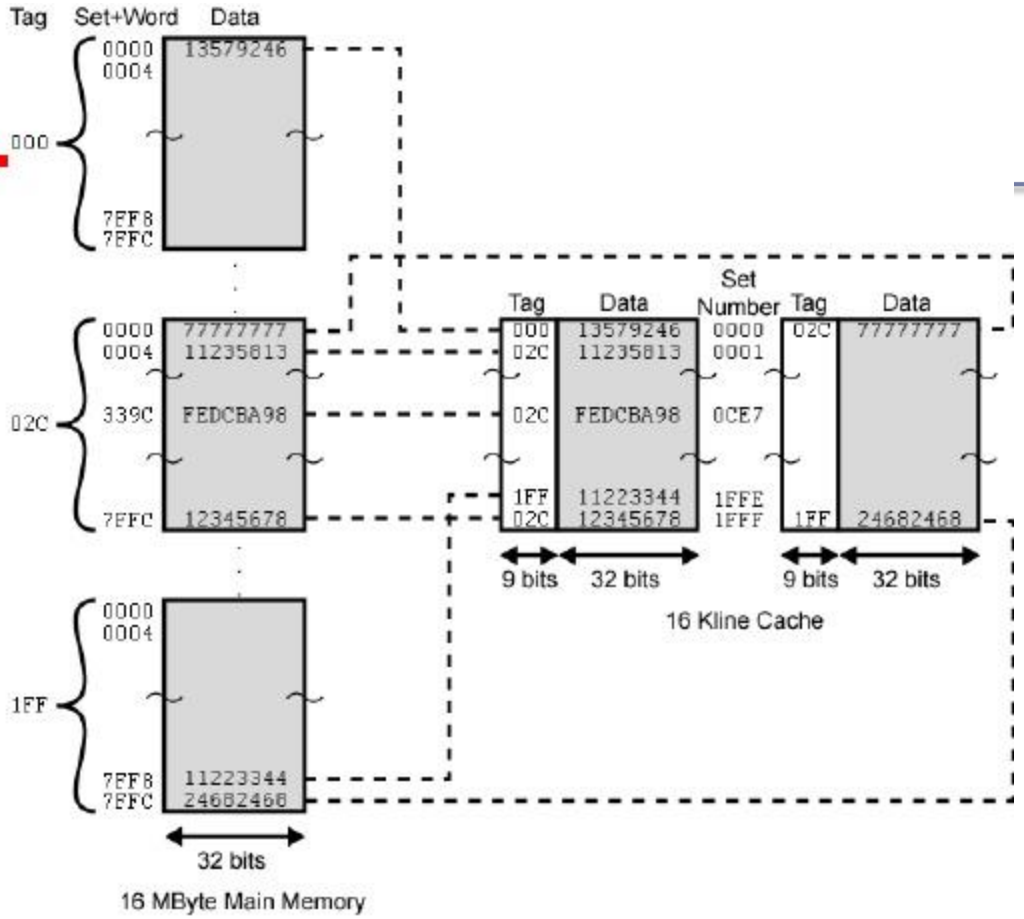
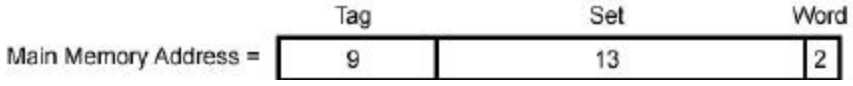


Figure 4.15 Two-Way Set Associative Mapping Example

Two-way set associative

Two Way Set Associative Mapping Example





Cache Replacement Algorithms

- Replacement algorithms are only needed for associative and set associative techniques. To achieve high speed, these algorithms must be implemented in hardware.

1. Least Recently Used (LRU) – replace the cache line that has been in the cache the longest with no references to it (most effective)
2. First-in First-out (FIFO) – replace the cache line that has been in the cache the longest
3. Least Frequently Used (LFU) – replace the cache line that has experienced the fewest references
4. Random – pick a line at random from the candidate lines (simulations have shown this to be slightly inferior to 1. to 3.)

Cache Write Policies

- If a cache line has not been modified, then it can be overwritten immediately; however, if one or more words have been written to a cache line, then main memory must be updated before replacing the cache line.

- There are two main potential write problems:

1. If an I/O module is able to read/write to memory directly, then if the cache has been modified a memory read cannot happen right away. If memory is written to, then the cache line becomes invalid.

2. If multiple processors each have their own cache, if one processor modifies its cache, then the cache lines of the other processors could be invalid.

Cache Write Policies

- write through – this is the simplest technique where all write operations are made to memory as well as cache ensuring main memory is always valid. This generates a lot of main memory traffic and creates a potential bottleneck
- write back – updates are made only to the cache and not to main memory until the line is replaced
- Note: Certain studies have shown that about 15% of memory references are writes except for HPC may approach 33% (vector-vector multiplication) and 50% (matrix transposition)
- cache coherency – keeps the same word in other caches up to date using some technique. This is an active field of research.

Cache Coherency

- cache coherency - keeps the same word in other caches up to date. This is an active field of research.
  1. Bus watching with write through - each cache controller monitors bus lines to detect write operations by other bus masters. If so, the cache entry is marked invalid
  2. Hardware transparency - additional hardware is used to write through changes to memory AND update all caches
  3. Noncacheable memory - a portion of main memory is shared by more than one processor and all accesses are cache misses (.i.e. shared memory is never copied into the cache)

Unified vs Split Caches

- Recent cache designs have gone from a unified cache to a split cache design (one for instructions and one for data).

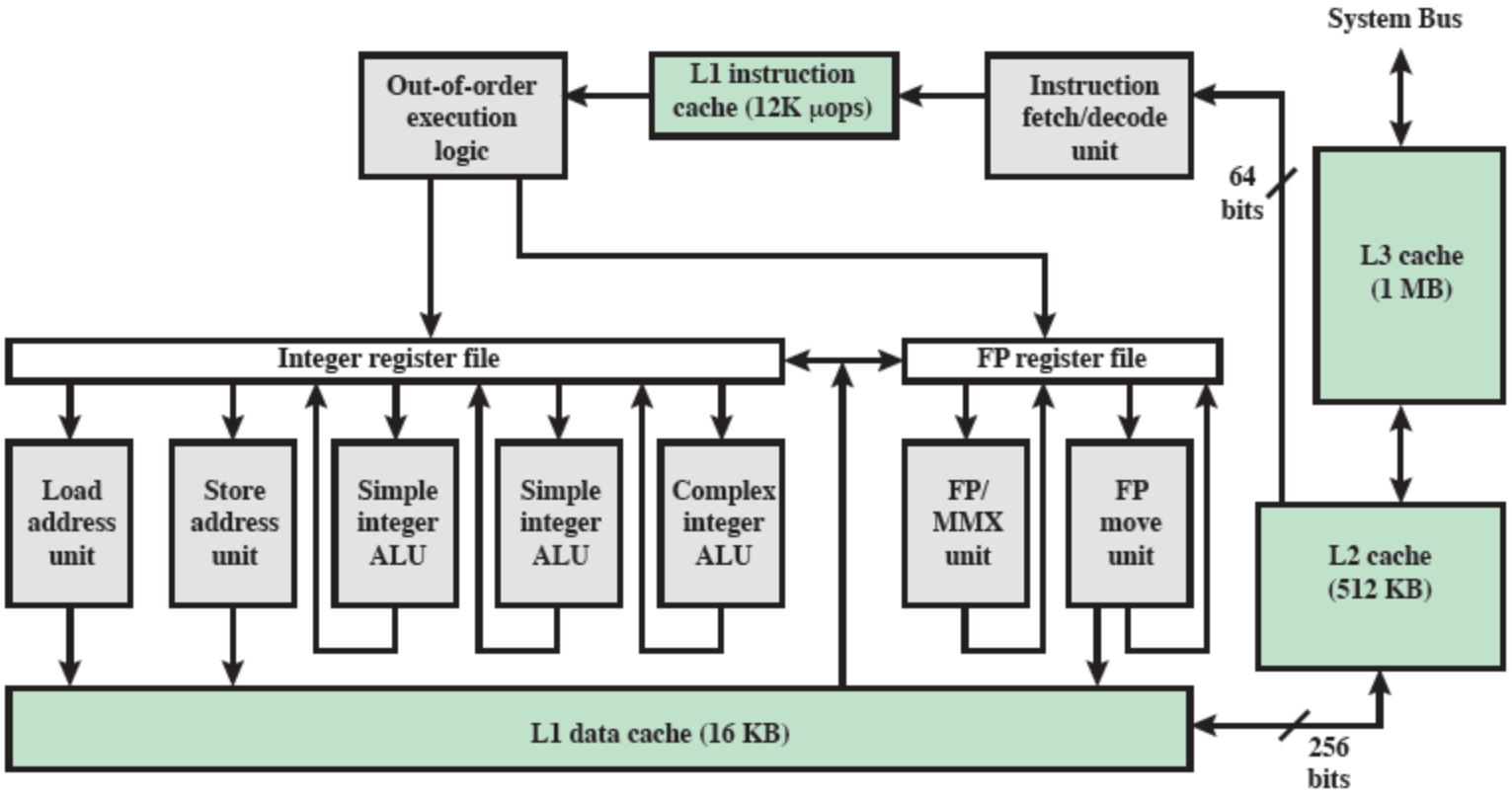
- Unified caches have the following advantages:
  1. unified caches typically have a higher hit rate
  2. only one cache is designed and implemented

- Split caches have the following advantages:
  1. parallel instruction execution and prefetching is better handled because of the elimination of contention between the instruction fetch/decode unit and execution unit.

Intel Cache Evolution

Problem	Solution	Processor on which Feature First Appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Pentium 4 Cache Organization



Pentium 4 Core Processor

- Fetch/Decode Unit
  1. Fetches instructions from L2 cache
  2. Decode into micro-ops
  3. Store micro-ops in L1 cache

- Out of order execution logic
  1. Schedules micro-ops
  2. Based on data dependence and resources
  3. May speculatively execute

Pentium 4 Core Processor

- Execution units
  1. Execute micro-ops
  2. Data from L1 cache
  3. Temporarily stores results in registers
- Memory subsystem
  1. Accesses main memory on a cache miss

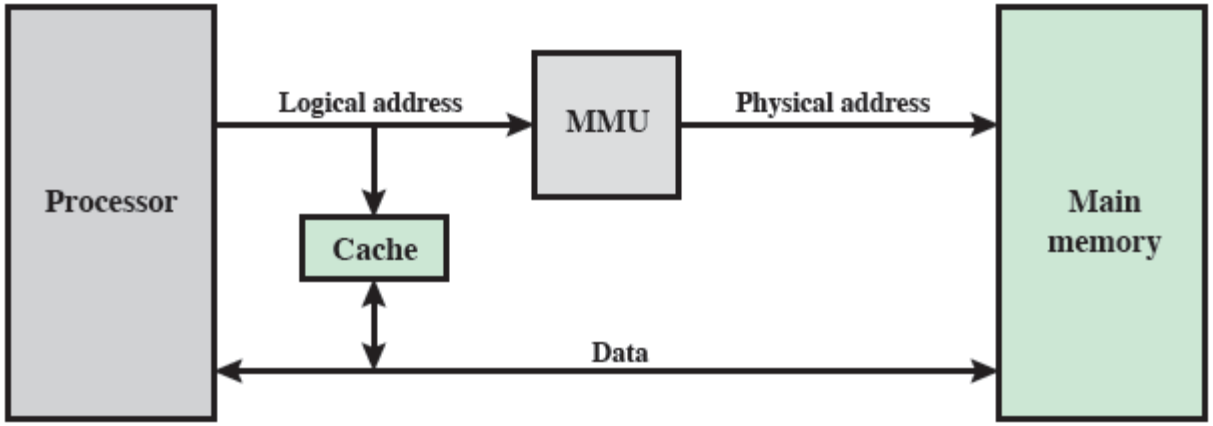
Virtual Memory

- Almost all non-embedded processors and many embedded processors support virtual memory
- virtual memory - allows programs to address memory from a logical point of view (i.e. without regard to the amount of physical main memory)
- When virtual memory is used, the address fields of machine instructions contain virtual addresses
- MMU (Memory Management Unit) - translates virtual addresses into physical addresses

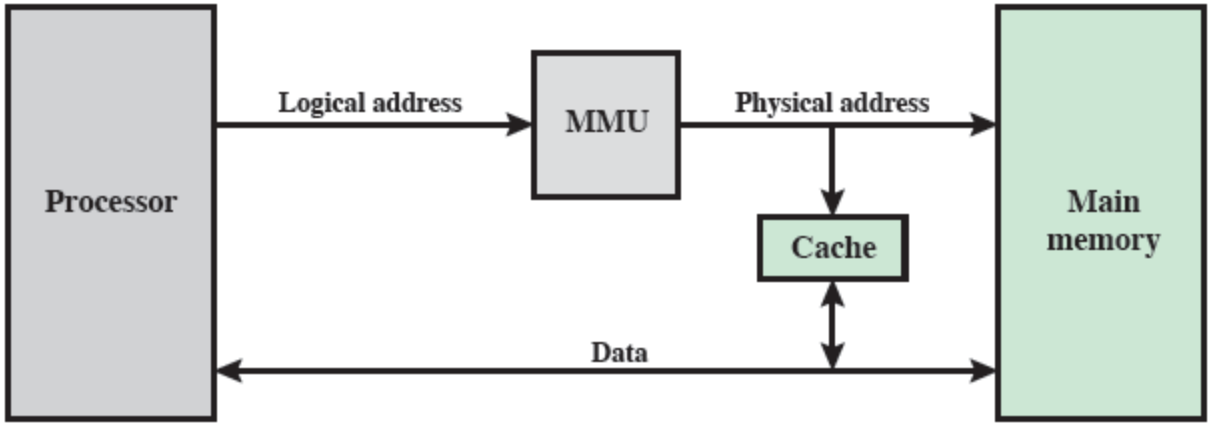
Virtual Memory

- When virtual memory addresses are used, the system designer can either:
  1. place the cache between the processor and MMU (logical cache uses virtual addresses)
  2. place the cache between the MMU and main memory (physical cache uses physical addresses)

Logical Cache



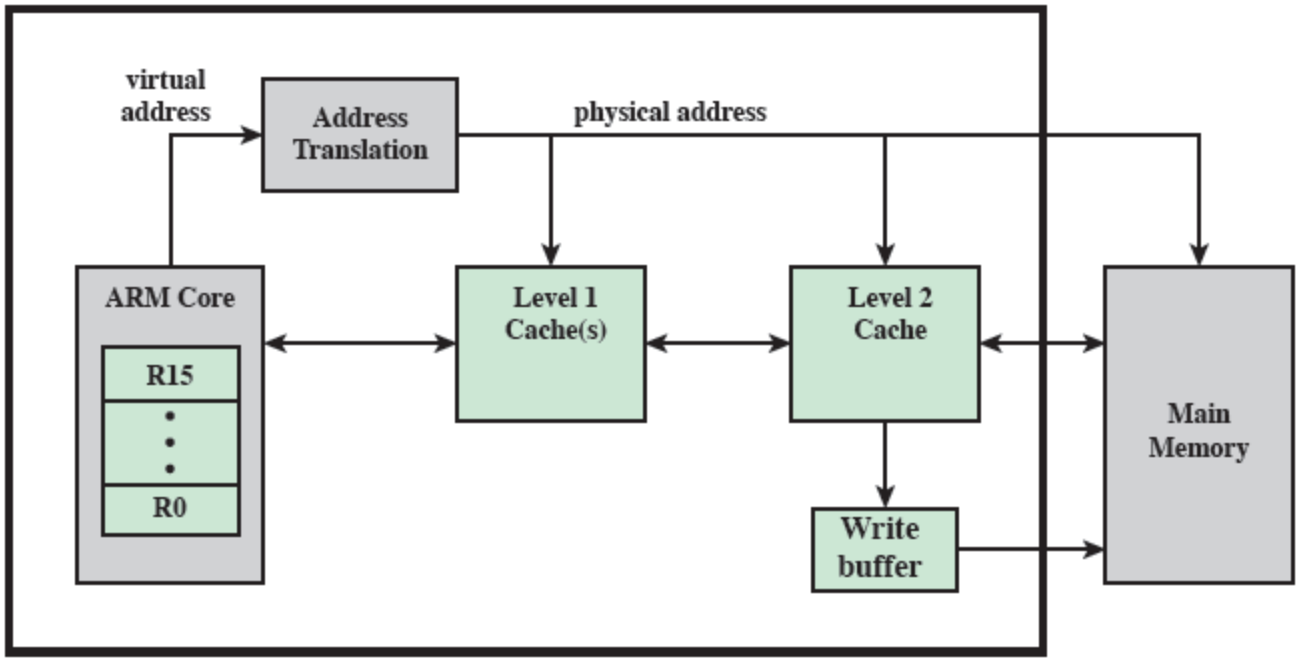
Physical Cache



ARM Cache Organization

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32

ARM Cache


virtual address
Address Translation
physical address
ARM Core
R15
R0
Level 1 Cache(s)
Level 2 Cache
Write buffer
Main Memory

Small FIFO write buffer

1. Enhances memory write performance
2. Between cache and main memory
3. Data put in write buffer at processor clock speed
4. Processor continues execution
5. External write in parallel until empty
    a. If buffer full, processor stalls
6. Data in write buffer not available until written
    a. Keep buffer small (e.g. 4 words of data)