Stallings 10th ed.

CS 320
Ch 4 – Cache Memory

1. The author list 8 classifications for memory systems;
   - Location
   - Capacity
   - Unit of transfer
   - Access method (there are four:Sequential, Direct, Random, and Associative)
   - Performance (there are three measures: access time, cycle time, and transfer rate.)
   - Physical type (there are four:Semiconductor, Magnetic, Optical, and Magneto-optical)
   - Physical characteristics (such as packaging, volatility, and erasability
   - Organization.

Memory systems get progressively faster each year. The memory speed needs to be matched to the CPU speed.   If the memory is too slow it becomes a bottleneck, If it is too fast (which is unlikely) you are wasting your money)

In general the faster the memory access time the higher the cost for a given technology.

In general the larger the memory the greater the cost for a given technology.

In general the larger the memory the slower the access time – this is due to address decoding.

Stallings 10th ed.

The memory hierarchy shows that faster, smaller, more expensive memory with more volatility with higher usage is at the top.

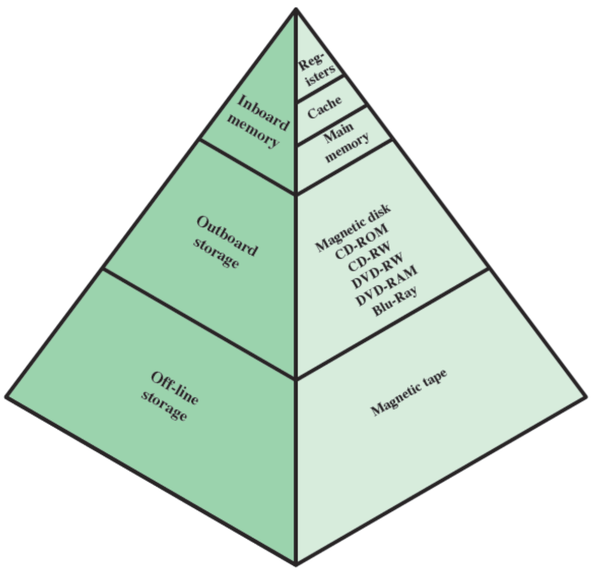


Figure 4.1 The Memory Hierarchy

The principle of *Locality of Reference* means that the next location you need will be very close to the location that you have.

This comes from the von Neumann architecture which stores the program sequentially.

Cache memory added to most desktop computer systems to p**rovide high speed at low cost? Not the highest speed and not the lowest cost but a reasonable tradeoff.**

Data is transferred between the CPU and a cache memory in words.
Data is transferred between the cache and main memory in b**locks.**

Mapping is a process that is used to change a main memory address to a cache address.

Mapping is necessary because the cache is smaller than the main memory and has a different address space.

Main memory is organized into blocks each of which contains a number of words. Cache is organized into *cache lines* or *lines*. A line is an area of cache which stores a block of memory copied from the main memory. A line also stores a **tag** which allows us to identify the block.

Stallings 10th ed.

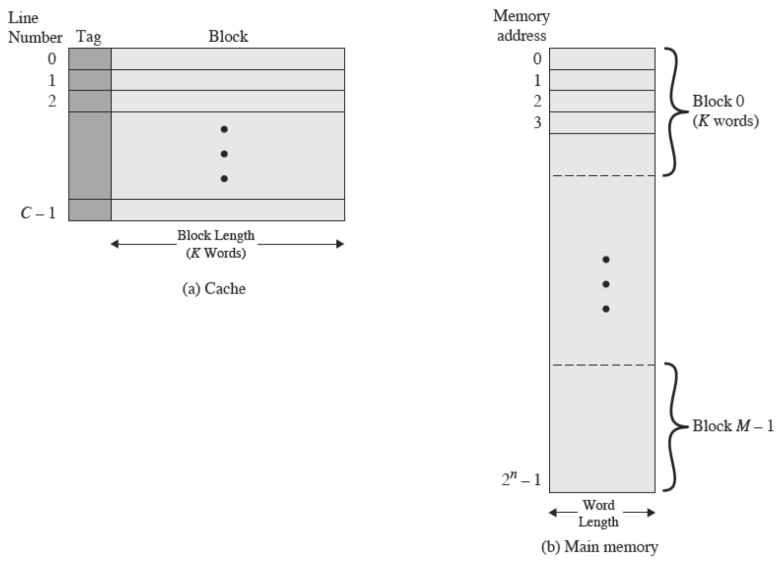


Figure 4.4 Cache/Main-Memory Structure

The author lists six design parameters for cache systems;

Cache size – cache is fast and expensive. Making it larger does not significantly improve the efficiency.

mapping function – The mapping function tells us how a block of memory gets mapped into the cache address space.

replacement algorithm – The replacement algorithm tells what we are going to throw away (replace) when the cache line gets full.

write policy – Data can be in both the cache and the main memory so if you write to cache the write policy tells us how main memory gets updated.

line size – The line size tells us how many words of memory go into a line. Since the cache size is limited a larger line size means fewer lines and vice-versa.

number of caches – It is common now to place a cache on the CPU and a second or third cache nearby. Three caches are not unusual.

Stallings 10th ed.

Figure 4.5 shows a flow chart for accessing a cache memory.

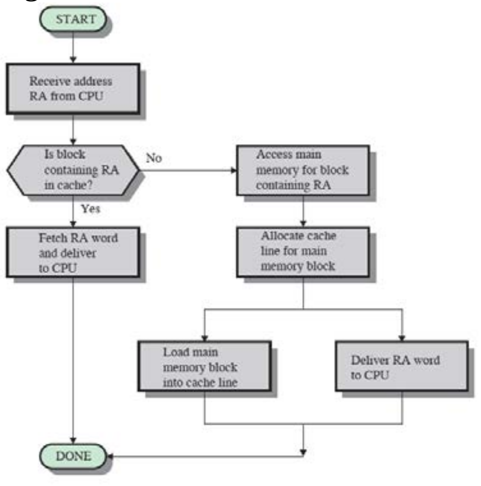

START
Receive address RA from CPU
Is block containing RA in cache?
No
Yes
Access main memory for block containing RA
Fetch RA word and deliver to CPU
Allocate cache line for main memory block
Load main memory block into cache line
Deliver RA word to CPU
DONE

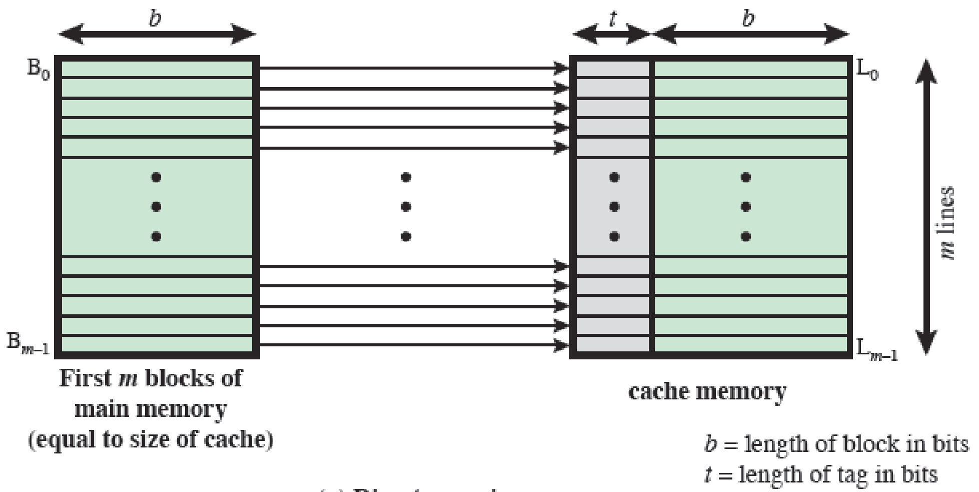
Figure 4.5 Cache Read Operation

There are three mapping functions currently in use **Direct, Associative, and Set Associative**

Direct Mapping
In direct mapping blocks of main memory mapped into the cache such that e**ach block has a unique cache line to which it maps.**

In direct mapping the main memory address is divided into three fields which are called the *Tag, Line,* and *Word.*

The Word **Specifies the word (or byte) within the block or line.**
The Line **Specifies the line number in the cache where the block is to be stored.**
The Tag is th**e most significant address bits. For a given address the tag is checked against the tag at the line number to verify that the data is in the cache.**


b
t
b
B_0
B_{m-1}
L_0
L_{m-1}
m lines
First m blocks of main memory (equal to size of cache)
cache memory
b = length of block in bits
t = length of tag in bits

(a) Direct mapping

Stallings 10th ed.

How direct mapping works for a cache memory.

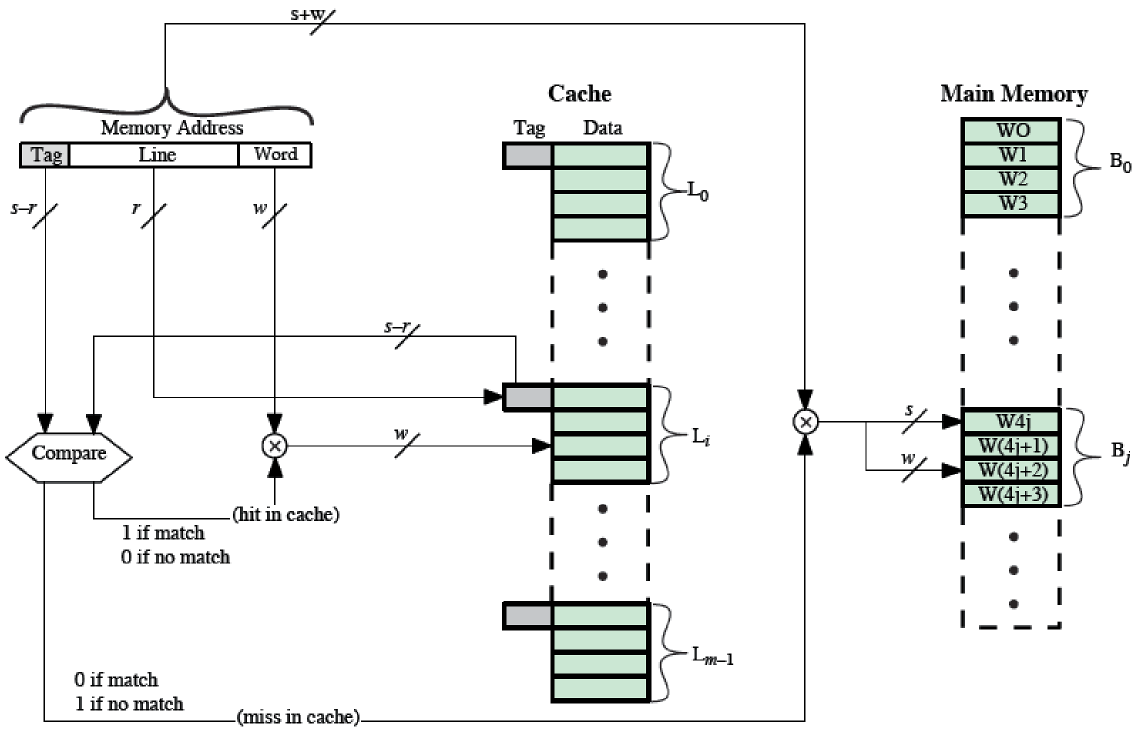


Figure 4.9 Direct-Mapping Cache Organization

For a system with a 64K cache, 16M main memory, and a block size of 4 bytes, how is the 24 bit address divided into its component parts?(In this case a byte is a word)
16 M => 24 bit address
64K cache/4 byte per block = 16K blocks = 14 bit line number.
4 bytes means 2 bits for the word.
The remaining 8 bits becomes the tag.

8 bit Tag, 14 bit Line, 2 bit Word.

Since multiple blocks in main memory map to the same cache line, when the CPU accesses data that is not in the cache That line gets replaced.

The advantages of direct mapping are **Simplicity and low cost.**
The disadvantages of direct mapping are **Potentially high miss ratio in some cases where successive data that maps to the same line is accessed.**

Stallings 10th ed.

Associative mapping
For associative mapping a memory block can go into and line in the cache.

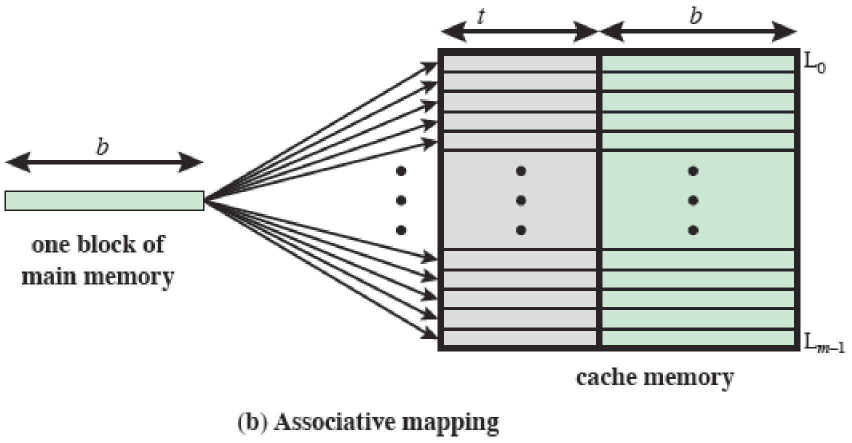


Figure 4.8 Mapping From Main Memory to Cache: Direct and Associative

In associative mapping the address is divided into only two parts which are the tag and the word. **Any memory block can go to any line so it's not necessary to specify the line.**

When a main memory address is generated, the tag is compared to every tag in the cache to determine if the block is in the cache. If the tag is present then the word bits is used to locate the particular word needed.

The following figure explains how associative mapping works.

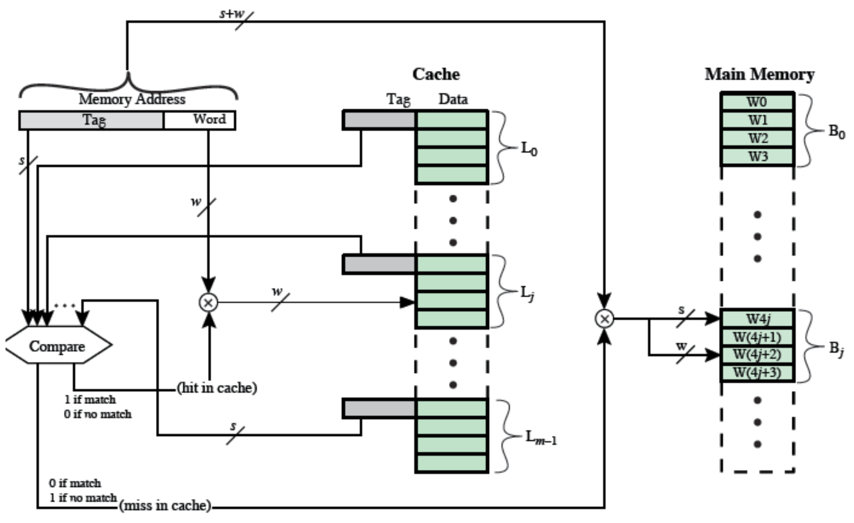


Figure 4.11 Fully Associative Cache Organization

Stallings 10th ed.

For a system with a 64K cache, 16M main memory, and a block size of 4 bytes, how is the 24 bit address divided into its component parts for an associative mapped cache? Since there are 4 bytes per line we need two bits for the word. The remaining 22 bits are the tag.

The advantage of associative mapping is that **Any word in memory can go to any line in the cache so it is potentially faster with a higher hit ratio.**

The disadvantage of associative mapping is that it is e**xpensive and complex circuitry needed to compare tag to all tags in cache in parallel. (Associative memory)**

For associative memory you have the data and you want to know if the data is in the memory. To make this fast enough the associative look up must be done in hardware.

Set Associative Memory
In set associative mapping the cache is further divided into sets each of which contain several lines. Each memory address specifies a fixed set but the word can be anywhere in the set. **This is a compromise between the direct mapping and the fully associative mapping.**

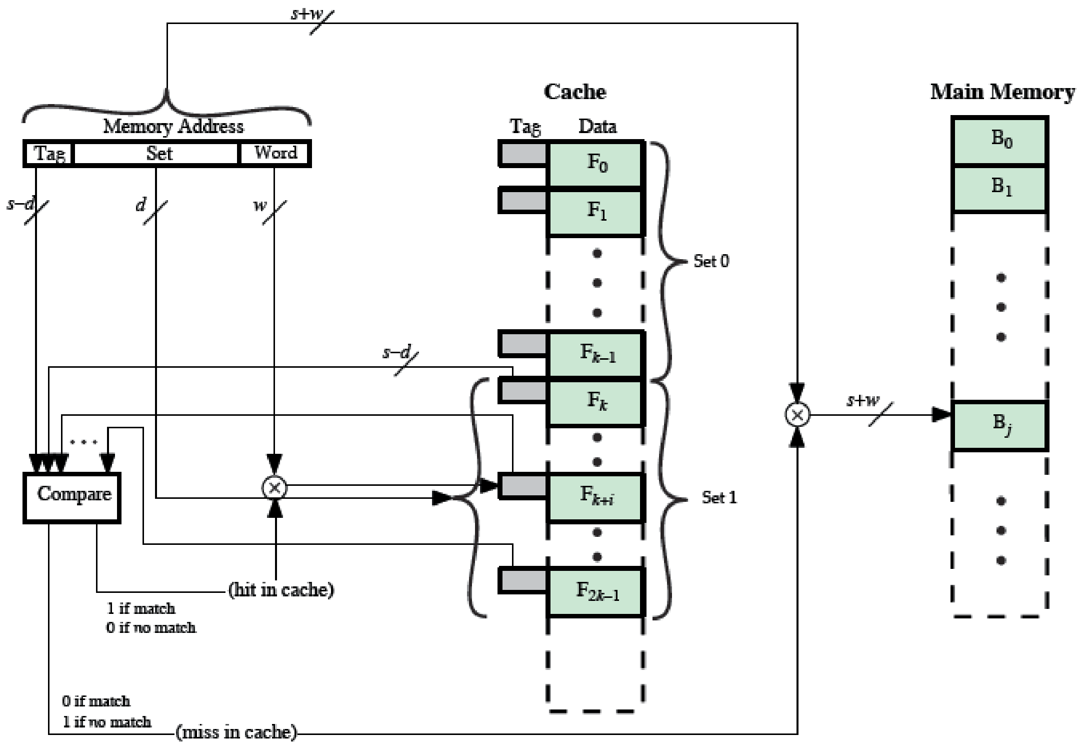


Figure 4.14 *k*-Way Set Associative Cache Organization

In set associative mapping the main memory address is divided into three fields which are called the *Tag, Set,* and *Word.*

The word **Specifies the word (or byte) within the block.**

Stallings 10th ed.

The Set **Specifies the set of blocks in the cache where the particular memory block can be stored.**
The Tag is t**he most significant address bits. For a given address the tag is checked against all of the tags in the set to verify that the data is in the cache.**

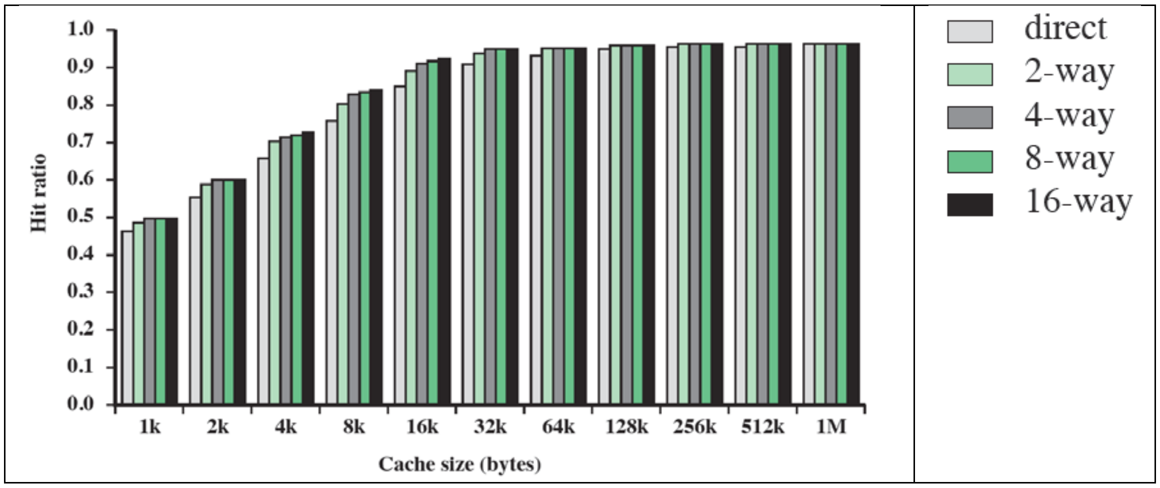
For a system with a 64K cache, 16M main memory, and a block size of 4 bytes, how is the 24 bit address divided into its component parts if the cache is two-way set associative?
64K cache/4 byte per block = 16K blocks.
With two blocks per set there are 8 K sets = 13 bit set number.
4 bytes in a block means 2 bits for the word.
The remaining 9 bits becomes the tag.
9 bit Tag, 13 bit Line, 2 bit Word.

If the memory/cache system in the above problem is 4-way set associative instead of 2-way set associative, how does the address get divided up into its component parts?
64K cache/4 byte per block = 16K blocks.
With four blocks per set there are 4 K sets = 12 bit set number.
4 bytes in a block means 2 bits for the word.
The remaining 10 bits becomes the tag.
10 bit Tag, 12 bit Line, 2 bit Word.

The advantages of set associative mapping is that it **Retains the low cost of a direct mapped cache but has the high hit ratio similar to the fully associative cache.**

The disadvantages of a set associative cache is **Slightly higher in cost and more complex than the direct mapped cache.**

The figure below shows the relationship between associativity, cache size, and hit ratio.



Stallings 10th ed.

Replacement Algorithms

A replacement algorithm is used to determine what needs to be replaced in the cache when it is full. For a direct mapped cache **there is no choice as to what is to be replaced. Every block has a fixed location where it goes in cache.**

The replacement algorithms currently in use are:

- **LRU – Least Recently Used**
- **Random – Pick an arbitrary line to discard**
- **FIFO – First in First Out**
- **LFU – Least Frequently Used**

LRU means least recently used. Replace the block that has been in the cache the longest *without being referenced*.

LFU means least frequently used, i.e. replace the block that has the least number of references.

In terms of additional hardware: f**or LRU some bits are needed to determine use. For LFU a counter is needed to determine frequency of use.**

FIFO or First In First Out means Replace the block that has been in the cache the longest.

Surprise! Random replacement is nearly as efficient as algorithms based on usage

Write Policy

The write policy governs how memory is written to in the presence of a cache. The author discusses two commonly used write policies; write through and write back. For write through **Every write to cache also writes simultaneously to memory.**

The advantage of Write Through is **Main memory is always up to date and the policy is easy to implement.**

The disadvantage is that this technique may generate lots of bus traffic. About 15% of instructions involve write operations but this may go as high as 50% in some applications.

Write back **Writes to cache do not cause an automatic write to main memory. Memory is only written to when the cache block is changed out.**

Write back r**educes bus traffic but memory may be out of date if it is used by multiple sources. This technique also requires some additional hardware in the way of an update bit in the cache.**

Stallings 10th ed.

Cache coherency: **In systems with multiple memories this term refers to the validity of a location. Cache may be out of date if the memory is changed by an outside source by way of DMA for example.**

Non-cacheable memory: **Some portion of main memory is not allowed to go into the cache. In this way the data can be shared and changed without fear of cache coherency problems.**

There is no optimal value for the line size. Line sizes of 8 to 32 bytes are most frequently used as empirically optimal. Larger line reduce the number of lines that can go into the cache meaning that blocks will have to be replaced more often.

Cache systems may be multilevel: This t**ypically involves two caches one of which is on the cpu chip and the second is off chip. These are referred to as the L1 and L2 caches.**

Cache may also be split. This means there is **a separate area for instructions and data? This is typical since data and instructions don't have much location of reference.**
The disadvantage of a split cache is that there is **less room for instructions and is slightly more complicated to implement.**

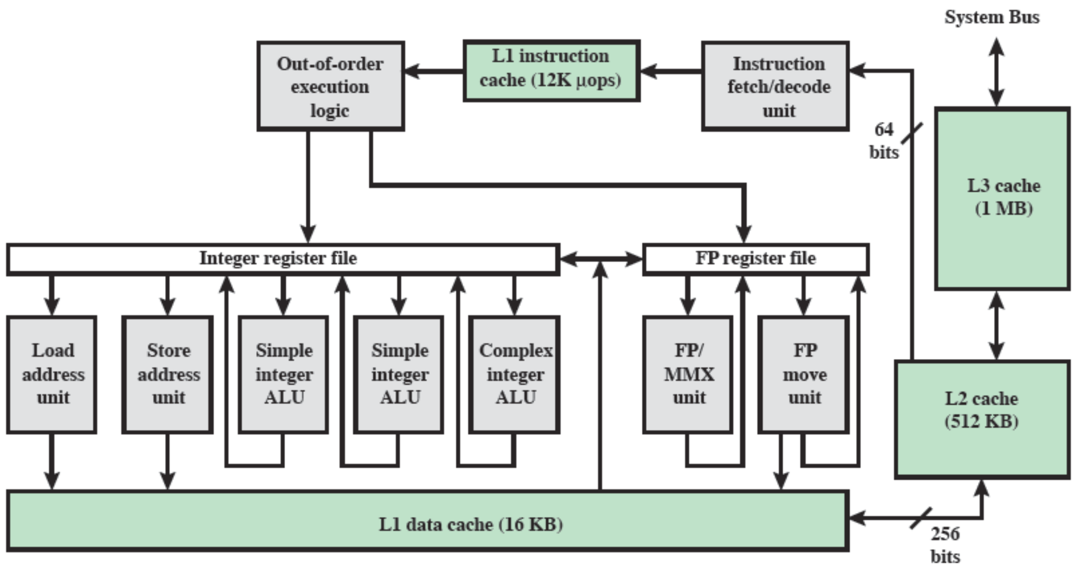


Figure 4.18 Pentium 4 Block Diagram

The Pentium 4 has three on board caches. There are **Two L1 caches for instructions and data and one unified L2 cache. The L3 cache is available only on high end versions.**

Stallings 10th ed.

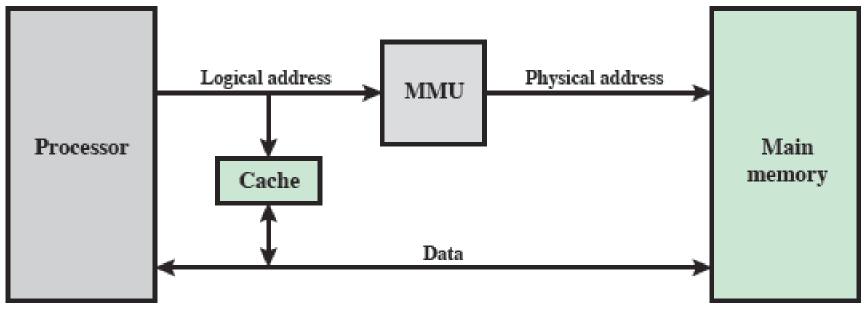
The P4's L2 cache organized as **8-way set associative with a line size of 128 bytes and an overall size of 256K.**

The two L1 caches on the P4 are organized as **4-way set associative, with a line size of 64 bytes and a total of 8K. From Ch 14, the L1 instruction cache is a trace cache and includes a different organization that features out of order instruction execution.**

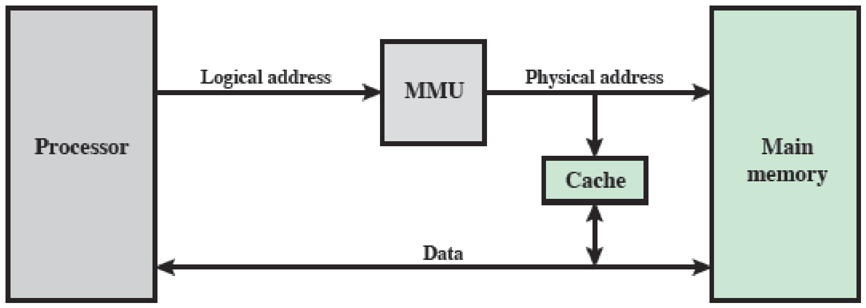
Note that the **P4 L1 instruction cache comes after the instruction decoder and thus holds microinstructions. These are more RISC like and support the superscalar architecture in the P4.**

The P4 data cache uses *Write Back*. The instruction cache needs no write policy.

The ARM processors mostly use a *Logical Cache* with the exception of the ARM 11 which uses a *Physical Cache*. The difference between these two cache is that the **Logical cache caches virtual addresses and physical cache caches physical addresses.**



(a) Logical Cache



(b) Physical Cache