Concepts Introduced

- introduction to memory hierarchies and the principle of locality
- basic cache organizations
- virtual memory

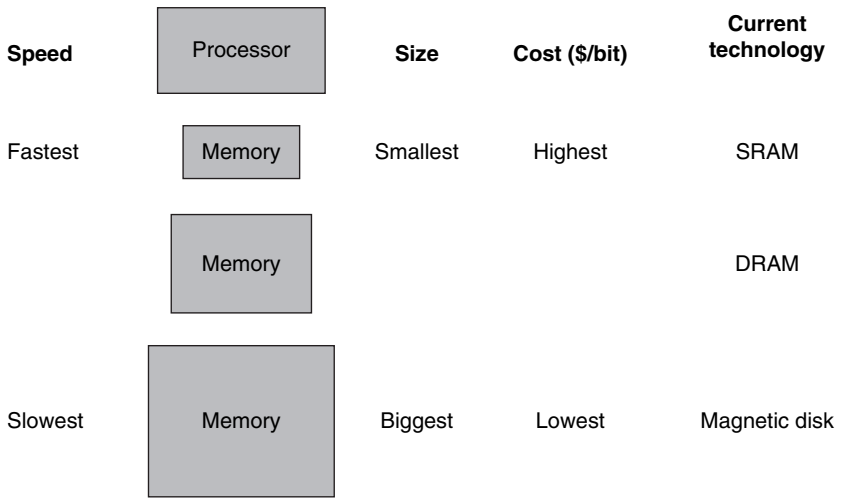
Principle of Locality

- temporal locality - If an item is referenced, then it will tend to be referenced again soon.
  - Instructions in loops are repeatedly referenced.
  - Instructions in recursive or utility functions tend to be repeatedly referenced.
  - Scalar variables are often repeatedly referenced.
- spatial locality - If an item is referenced, then items whose addresses are close by will tend to be referenced soon.
  - Instructions are accessed sequentially unless there is a transfer of control.
  - Data elements of an array tend to be accessed sequentially.
  - Fields of a structure are often accessed close in time.

Memory Hierarchy

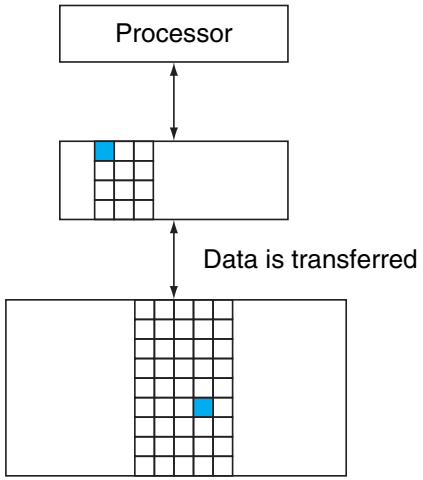
- Exploits the principle of locality.
- Use a smaller, faster, and more expensive (per byte) memory.
- Use a larger, slower, and cheaper (per byte) memory.
- All data found in one level is typically also found in the next larger level.
- Goals:
  - Catch most of the references in the smaller, faster, and more expensive memory.
  - Have most of the cost per byte be at the larger, slower, and cheaper memory.
- The memory hierarchy is used to implement protection schemes as well.

General Memory Hierarchy



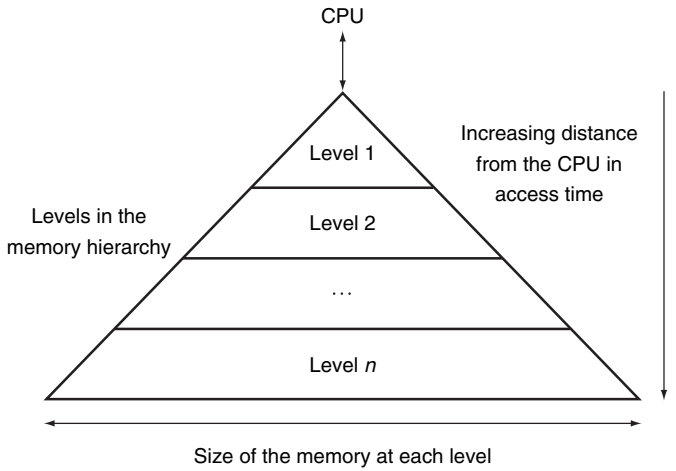
Pair of Memory Hierarchy Levels

- The unit of data that is transferred between two levels is fixed in size and is called a *block*.



Structure of the Memory Hierarchy

- The size of each memory hierarchy level increases as the distance from the processor increases.



Components of a Memory Hierarchy

- registers (flip flops) - measured in words, managed by the compiler
- cache (SRAMs) - measured in Kbytes (L1) to Mbytes (L3), managed by hardware
- main memory (DRAMs) - measured in Gbytes, managed by hardware
- disk - measured in Tbytes, managed by the operating system

Memory Hierarchy Terms

- hit - item found in a specified level of the hierarchy
- miss - item not found in a specified level of the hierarchy
- hit time - time required to access the desired item in a specified level of the hierarchy (includes the time to determine if the access is a hit or a miss)
- miss penalty - the additional time required to service the miss
- hit rate - fraction of accesses that are in a specified level of the hierarchy
- miss rate - fraction of accesses that are not in a specified level of the hierarchy
- block - unit of information that is checked to reside in a specified level of the hierarchy and is retrieved from the next lower level on a miss

Memory Hierarchy Technologies

- The access time and price per bit vary widely among memory hierarchy technologies.

memory technology	typical access time	$ per GiB in 2012
SRAM	0.5ns - 2.5ns	$500 - $1000
DRAM	50ns - 70ns	$10 - $20
flash memory	5,000ns - 50,000ns	$0.75 - $1.00
magnetic disk	5,000,000ns - 20,000,000ns	$0.05 - $0.10

SRAM

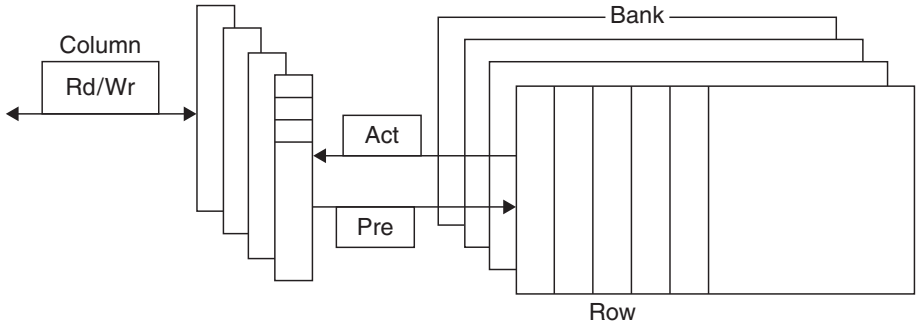
- SRAM - Static Random Access Memory
- Used in caches.
- Usually has a single access port that can provide either a read or write access.
- Requires 6 transistors per bit to prevent data from being corrupted when read and requires 4 times the amount of space as compared to DRAM for each stored bit.
- Each bit value is stored in a cell by using a pair of inverting gates and the value can be kept indefinitely as long as power is applied, which is why SRAM is called static.
- SRAM assess time is about 5 to 10 times faster than DRAM.
- SRAM is perhaps 20 times more expensive than DRAM.
- Synchronous SRAM (SSRAM) has a synchronous interface to allow burst transfers, where a clock is used to transfer successive words given only a starting address and length.

DRAM

- DRAM - Dynamic Random Access Memory
- Used for main memory.
- Requires a single transistor per bit, which is lost after being read, so each read requires that the data be written back.
- The value representing a bit is kept in a cell that is stored as a charge in a capacitor that is accessed by the single transistor.
- DRAM requires that the data be refreshed periodically, about 1% to 2% of the cycles, which is why DRAM is called dynamic and is accomplished by reading the data and writing it back.

Internal Organization of a DRAM

- Modern DRAMs are organized in banks, which can be simultaneously accessed.
- Each bank consists of a series of rows, where a row is transferred to a buffer, which can be transferred at successive addresses.



DRAM Accesses

- Access time - time between when a read is requested and the desired word arrives.
- Cycle time - minimum time between requests to memory.
- The address pins to a DRAM chip are typically decreased by two by multiplexing the address lines.
- Row Access Strobe (RAS) - first half of the address is sent for the row
- Column Access Strobe (CAS) - second half of the address is sent for the column

DRAM Optimizations

- DRAMs allow repeated accesses to the same row without another RAS, which is called fast page mode.
- Synchronous DRAM (SDRAM) adds a clock signal to avoid overhead of synchronizing with the memory controller and allows a variable number of bytes to be sent over multiple cycles per memory request.
- Double data rate (DDR) transfers data on both the rising and falling edges of the DRAM clock signal.
- SDRAMs introduced 2 to 8 banks that can operate independently and simultaneously service independent requests, which also reduces power.
- SDRAMs have a low power mode, which disables the SDRAM except for the internal refresh. Returning to an active power mode requires about 200 cycles.

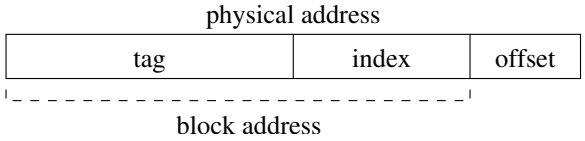
Memory Hierarchy Questions

- Where can a block be placed in the current level? (Block Placement)
- How is a block found if it is in the current level? (Block Identification)
- Which block should be replaced on a miss? (Block Replacement)
- What happens on a write? (Write strategy)

Cache Terms

- line - a block of information residing in the cache
- set - set of cache lines that can be accessed with the same index
- valid bit - indicates if a cache line contains a valid tag
- dirty bit - indicates if the cache line has been updated (used for writes)

Physical Address Used to Access the Cache

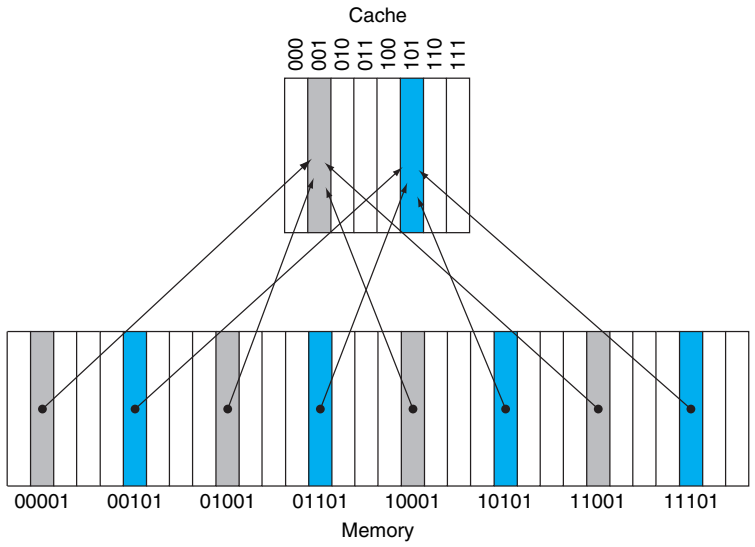


- The *offset* is used to indicate the first byte that is accessed within the block and its size in bits is log_2(number of bytes in a block).
- The *index* is used to access a specific set within the cache and its size in bits is log_2(number of sets in the cache).
- The *tag* is used to verify that the desired block has been found in the cache and its size in bits is the address size minus the number of bits for the index and offset.
- The *tag* and *index* together comprise the block address.

Cache Organizations

- A *direct-mapped* cache has only one line per set, so a memory block can be placed in the single cache line accessed by the index field.
- A *fully-associative* cache has only one set, so a memory block can be placed in any cache line.
- A *set-associative* cache has a specified number of lines per set, so a memory block can be placed in any cache line within the set accessed by the index field.

A Direct-Mapped Cache with Eight Entries



Example Accesses to a Direct-Mapped Cache

- The addresses shown in this example are block addresses.

decimal block address of reference	binary block address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_2	miss	$(10110_2 \bmod 8) = 110_2$
26	11010_2	miss	$(11010_2 \bmod 8) = 010_2$
22	10110_2	hit	$(10110_2 \bmod 8) = 110_2$
26	11010_2	hit	$(11010_2 \bmod 8) = 010_2$
16	10000_2	miss	$(10000_2 \bmod 8) = 000_2$
3	00011_2	miss	$(00011_2 \bmod 8) = 011_2$
16	10000_2	hit	$(10000_2 \bmod 8) = 000_2$
18	10010_2	miss	$(10010_2 \bmod 8) = 010_2$

State of Cache after Example Accesses That Miss

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

a. The initial state of the cache after power-on

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

b. After handling a miss of address (10110_{two})

State of Cache after Example Accesses That Miss (cont.)

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

c. After handling a miss of address (11010_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

d. After handling a miss of address (10000_{two})

State of Cache after Example Accesses That Miss (cont.)

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

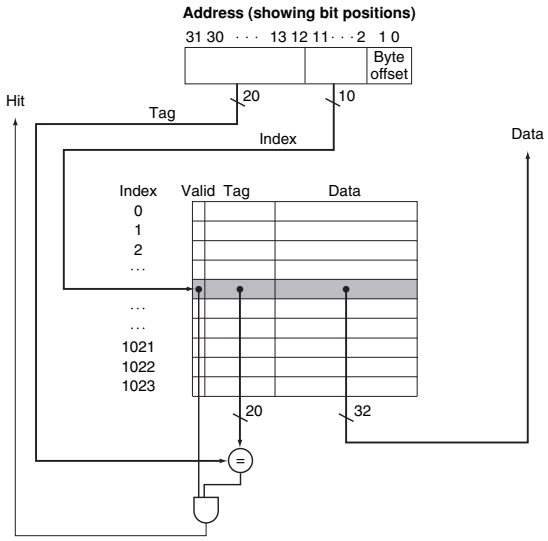
e. After handling a miss of address (00011_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

f. After handling a miss of address (10010_{two})

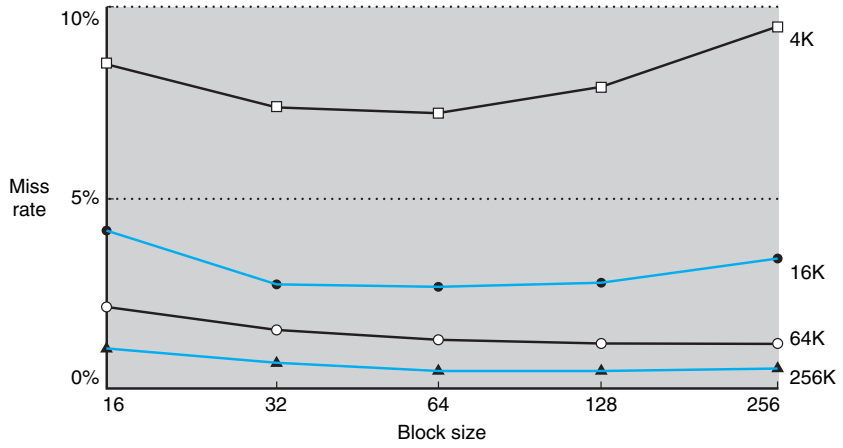
Example Direct-Mapped Cache Organization

- Each line in this cache contains only a single word of data.



Miss Rate versus Block Size

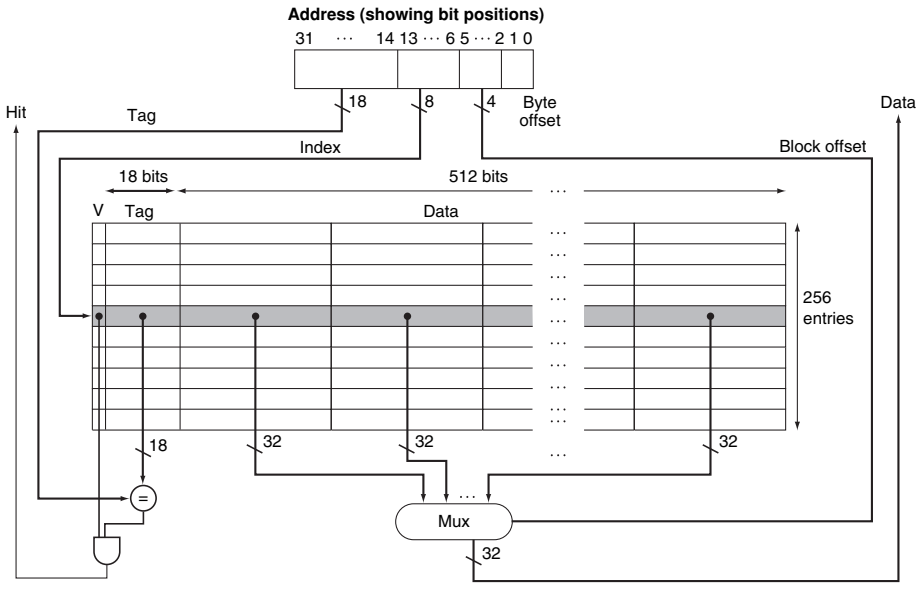
- Spatial locality can be exploited by increasing the block size.



Number of Bits Stored in a Cache

- How many bits are required to be stored in a direct-mapped cache that contains 64K bytes of data, assuming each block is 4 bytes in length and a 32-bit address?
- How many bits are required to be stored in a direct-mapped cache that contains 64K bytes of data, assuming each block is 8 bytes in length and a 32-bit address?

A Cache with 64-Byte Blocks



Larger Block Example

- Assume a direct-mapped cache with 4 blocks and 8 bytes per block.
- How is the physical address partitioned?

tag	index	offset

- Fill in the appropriate information for the following memory references.

Address	Tag	Index	Offset	Result
0x4				
0x8				
0xc				
0x0				
0x20				
0x4				

Steps to Be Taken on an Instruction Cache Miss

- Send the block address of the original PC value (current PC-4) to the next level of the memory hierarchy.
- Instruct the next level of the memory hierarchy to perform a read and stall until the read has completed.
- Update the block in cache. Assign the upper bits of the address to the tag. Set the valid bit.
- Refetch the instruction.

Cache Impact on Performance

- Consider only the impact of an instruction cache. Assume the following:
  - miss penalty is 10 cycles
  - miss ratio is 0.10
- The average access time is:
  - hit_time + miss_rate * miss_penalty
- The number of cycles disregarding pipeline and data cache stalls would be:
  - instruction_count*average_access_time
    instruction_count*(hit_time+miss_rate*miss_penalty)
    instruction_count*(1+0.10*10)
    instruction_count*2.0
- With a miss ratio of 0.05, the number of cycles would be:
  - instruction_count*(1+0.05*10)
    instruction_count*1.5

Pipelined Processors Use Separate Caches

- The access to memory in a pipelined machine is first an access to a cache.
- Separate instruction and data caches are used to prevent a structural hazard.
- This allows the architect to tune the attributes of each cache.

cycle	1	2	3	4	5	6	7	8
inst 1	IC	ID	EX	**DC**	WB			
inst 2		IC	ID	EX	DC	WB		
inst 3			IC	ID	EX	DC	WB	
inst 4				**IC**	ID	EX	DC	WB

Cache Misses Can Be Overlapped with Other Pipeline Stalls

- Assume the following two instructions were in cache.

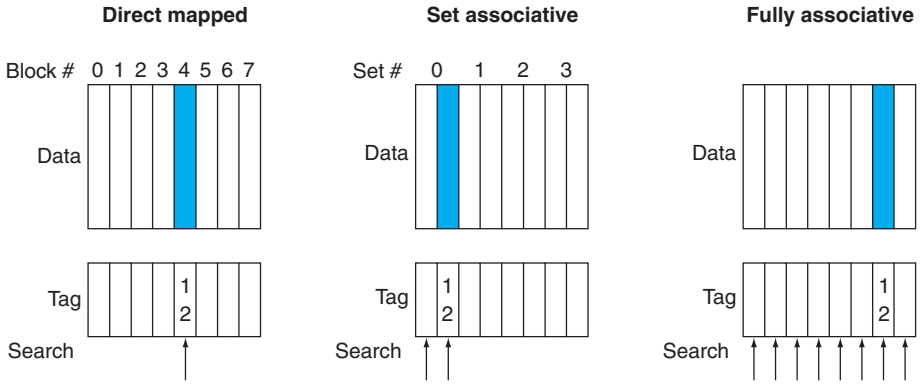
cycle	1	2	3	4	5	6	7
lw $2,0($3)	IF	ID	EX	MEM	WB		
add $4,$2,$1		IF	ID	stall	EX	MEM	WB

- Assume the second instruction is not in cache and there is a 2 cycle miss penalty. The load hazard stall is overlapped with the instruction cache miss stall.

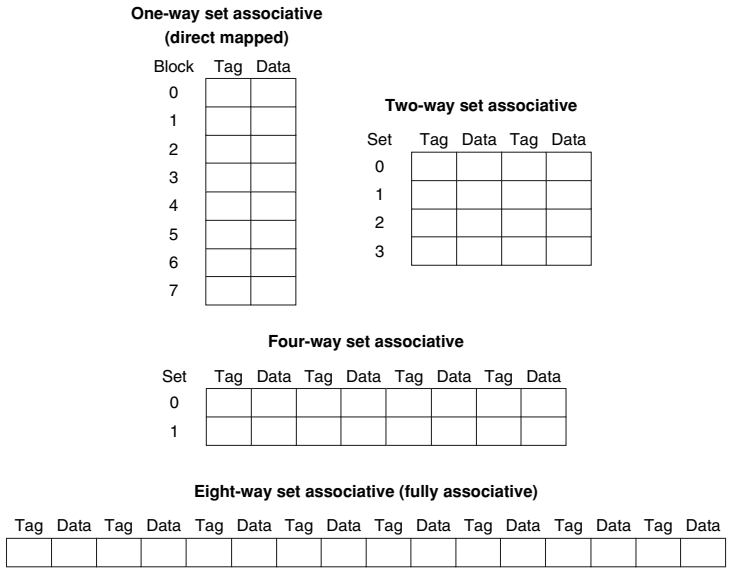
cycle	1	2	3	4	5	6	7	8
lw $2,0($3)	IF	ID	EX	MEM	WB			
add $4,$2,$1		IF	stall	stall	ID	EX	MEM	WB

Location of a Block with Different Cache Organizations

- The example below has a reference with a block address of 12 and three cache organizations with 8 blocks.

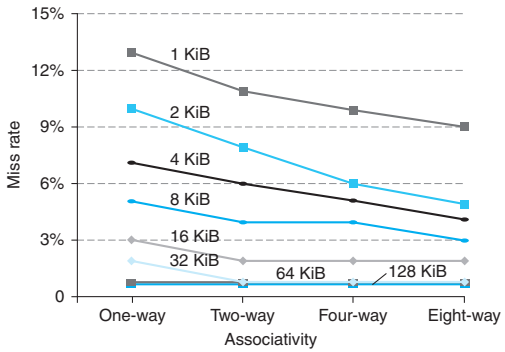


Example Cache Organizations



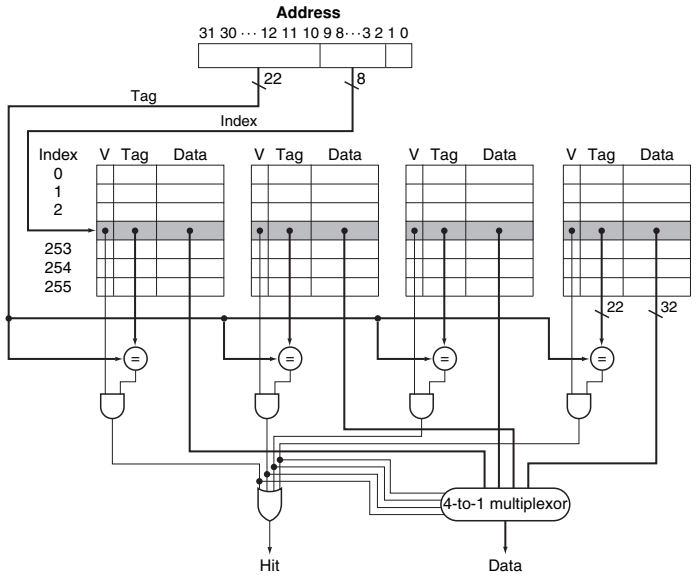
Set Associative Caches

- Increasing associativity can decrease miss rate.



- disadvantages
  - Requires a comparator for each way.
  - Requires more tag bits per cache block.
  - Requires logic to determine which line to replace.
  - May increase the hit time.

A Four-Way Set Associative Cache

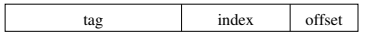


Block Replacement

- If there is more than one block in a cache set, then a block must be selected to be replaced on a cache miss.
- random
  - The block replaced is randomly chosen.
  - Random replacement easy to implement in hardware.
- LRU
  - The block replaced is the least-recently accessed block.
  - LRU typically reduces the miss rate better than random.
  - LRU can be expensive to implement for high levels of associativity.
  - The space required is $\lceil log2(n!) \rceil$.

Set Associativity Example

- Assume a 2-way set-associative cache with 64 cache sets, 4 words per block, and an LRU replacement policy.
- How is the physical address partitioned?



- Fill in the information for the following memory references.

Address	Tag	Index	Offset	Result
0x12c				
0x130				
0x4c0				
0x1134				
0x1138				
0x24c8				
0x128				
0x130				
0x4c4				
0x8c8				

Write Policy

- write through
  - The data is written to both the current level and next level of the memory hierarchy.
  - Simpler to implement.
  - Can use write buffers to reduce stalls.
  - The current and next levels of the memory hierarchy are consistent.
- write back
  - The data is written to only the cache. The modified cache block (dirty bit set) is only written to the next level of the memory hierarchy when it is replaced.
  - Reduces the number of accesses to the next larger level of the memory hierarchy.

Write Miss Policy

- write allocate
  - The block is loaded on a write miss.
  - Typically used with write back.
- no-write allocate
  - The block is not loaded on a write miss, but is updated in the next lower level of the memory hierarchy.
  - Typically used with write through.

Write-Through, No-Write Allocate Example

- Assume a 2-way set-associative cache with 64 cache sets, 4 words per block, and an LRU replacement policy. Fill in the appropriate information for the following memory references.

R/W	Addr	Tag	Index	Offset	Result	Memref	Update Cache?
W	0x12c						
R	0x130						
R	0x1134						
W	0x1138						
W	0x2130						
R	0x2134						
R	0x130						

Write-Back, Write-Allocate Example

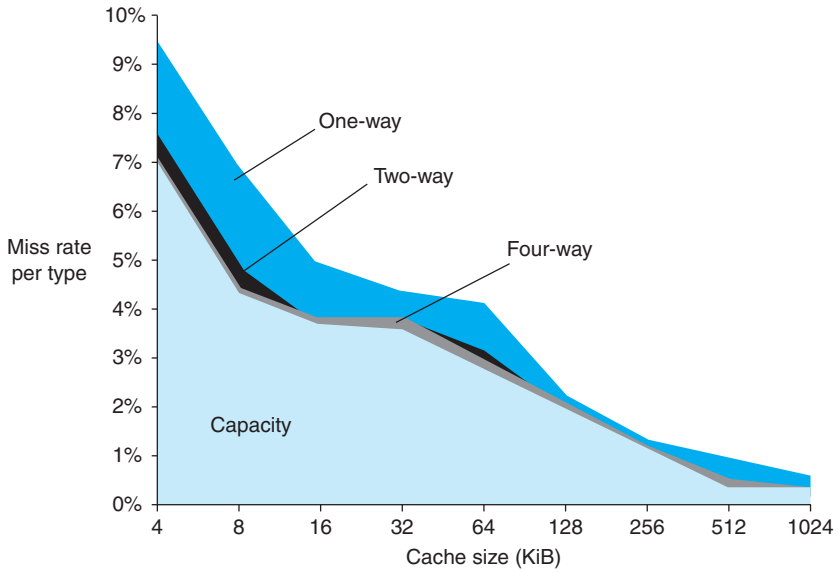
- Assume a 2-way set-associative cache with 64 cache sets, 4 words per block, and an LRU replacement policy. Fill in the appropriate information for the following memory references.

R/W	Address	Tag	Index	Offset	Result	Memref	Update Cache?
W	0x12c						
R	0x130						
R	0x1134						
W	0x1138						
W	0x2130						
R	0x2134						
R	0x130						

Categorizing Memory Hierarchy Misses

- compulsory misses
  - Caused by the first access to a block.
  - Can be decreased by increasing the block size.
- capacity misses
  - Caused when the memory hierarchy level cannot contain all the blocks needed during the execution of a process.
  - Can be decreased by increasing the cache size.
- conflict misses
  - Occur in direct-mapped and set-associative organizations when too many blocks compete for the same set.
  - Can be decreased by increasing associativity.

Categorizing Miss Results



Critical Word First and Early Restart

- Critical word first means to request the missed word first from the next memory hierarchy level to allow the processor to continue while filling in the remaining words in the block, usually in a wrap-around fill manner.
- Early restart means to fetch the words in the normal order, but allow the processor to continue once the requested word arrives.
- Both approaches effectively reduce the miss penalty.
- Status bits must be used to indicate how much of the block has arrived.
- More beneficial for caches with large block sizes.

Multilevel Caches

- As the clock rate of processors has increased faster than the time to access main memory (DRAM), multiple levels of caches have been introduced.
- Three levels of cache all on the same chip are now common, where there are separate L1 instruction and data caches and unified L2 and L3 caches.
- The L1 (L2) caches are typically much smaller than L2 (L3) caches with lower associativity to provide faster access times.
- The L1 (L2) caches typically have smaller block sizes than L2 (L3) caches to have a shorter miss penalty.
- The L2 (L3) caches are typically much larger and have higher associativity than L1 (L2) caches to decrease the miss rate due to the higher L2 (L3) miss penalties.

Performance of Multilevel Caches

- The miss penalty of an upper level cache is the average access time of the next lower level cache.

$$avg\_access\_time = hit\_time\_L1 + miss\_rate\_L1 * miss\_penalty\_L1$$
$$miss\_penalty\_L1 = hit\_time\_L2 + miss\_rate\_L2 * miss\_penalty\_L2$$

- What is the average access time give that the L1 hit time is 1 cycle, the L1 miss rate is 0.05, the L2 hit time is 4 cycles, the L2 miss rate is 0.25, and the L2 miss penalty is 50 cycles?
- avg_access_time = 1 + 0.05 * (4 + 0.25 * 50) = 1.85
- The miss rate can be calculated locally or globally.

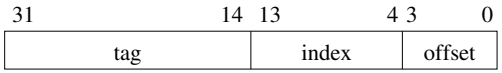
$$local\_miss\_rate = \frac{misses\_in\_cache}{accesses\_to\_cache}$$
$$global\_miss\_rate = \frac{misses\_in\_cache}{accesses\_to\_L1\_cache}$$

Techniques for Improving Cache Performance

- techniques for reducing the miss rate
  - Increase the associativity to exploit temporal locality.
  - Increase the block size to exploit spatial locality.
- techniques for reducing the miss penalty
  - Use wrap-around filling of a line (early restart and critical word first).
  - Use multilevel caches.
- techniques for reducing the hit time
  - Use small and simple L1 caches.

Designing a Cache Controller

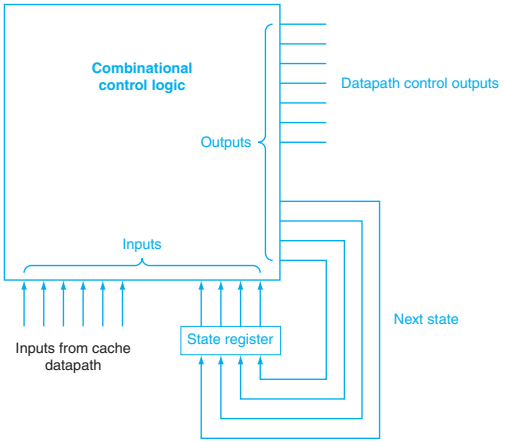
- cache characteristics
  - direct-mapped, write-back with write allocate
  - block size is 16 bytes
  - cache size is 16KiB
  - 32-byte addresses



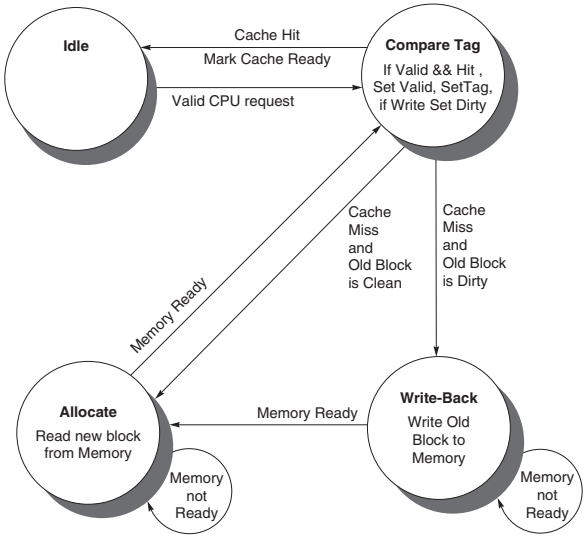
- signals between the processor and cache and between cache and memory
  - 1-bit read or write signal
  - 1-bit valid signal indicating if a cache or memory operation
  - 32-bit address
  - data 32-bit processor => cache or 128-bit cache => memory
  - data 32-bit cache => processor or 128-bit memory => cache
  - 1-bit ready signal that cache or memory operation is complete

General Cache Controller

- A cache controller can be implemented as a finite state machine using a combinational logic block and a register to hold the current state.



Finite State Machine for a Cache Controller



Overlays

- An application programmer used to have to manage main memory when the application was larger than the available memory.
  - A programmer divides their programs into pieces.
  - The programmer determines which pieces never needed to be used at the same time.
  - A portion of the program loads from disk or stores to disk these pieces during execution.
  - The programmer ensures that the maximum number of program pieces used at the same time fits into physical memory.

Virtual Memory

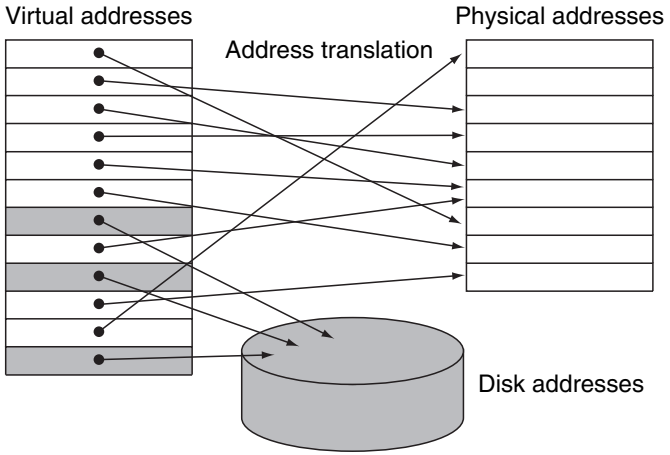
- Divides physical memory into fixed-size blocks and allocates these blocks to different processes.
- Provides a mapping between blocks in physical memory and blocks on disk.
- Allows a process to execute with only portions of the process being in main memory.
- Also reduces program startup time.
- Provides protection to prevent processes from accessing blocks inappropriately.

Virtual Memory Terms

- *Page* is the name used for a block.
- *Page fault* is the name for a miss.
- *Virtual address* is the address produced by a CPU.
- *Physical address* is the address used to access main memory and typically cache as well.
- *Page table* is the data structure containing the mappings between virtual and physical addresses.
- *Translation Lookaside Buffer* is a cache that contains a portion of the page table.

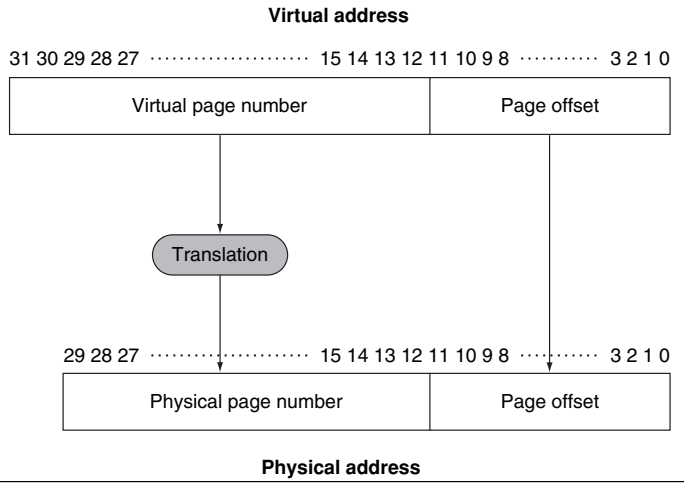
Virtual to Physical Page Mapping

- Virtual memory gives the illusion of a contiguous area of memory for each process.



Mapping from a Virtual to Physical Address

- The virtual page number in the virtual address is translated to a physical page number in the physical address and the page offset remains the same.

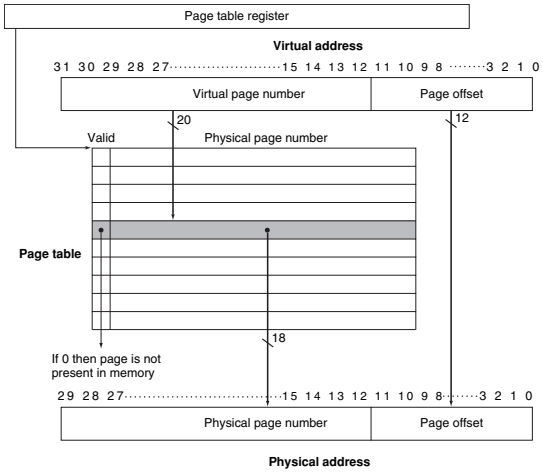


Page Tables

- Number of entries in a page table is equal to the number of virtual pages.
- Number of virtual pages is the size of the virtual address space divided by the page size.
- Each process has its own page table and the page table register points to the beginning of the page table in memory.
- A page table entry will typically contain a physical page number, resident bit, dirty bit, use bit, protection field (e.g. read only), disk address.

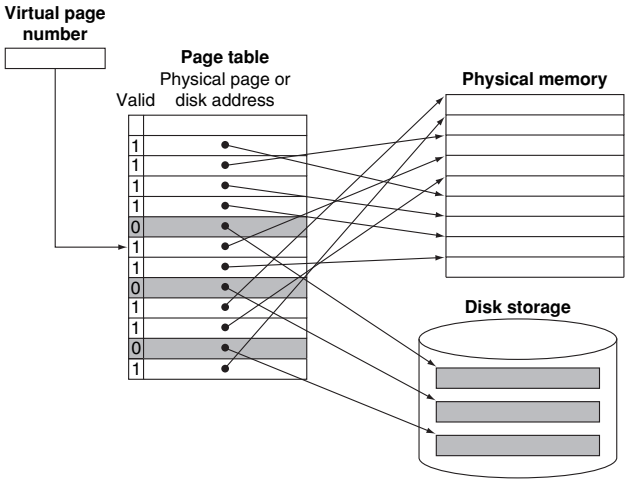
Page Table Used to Obtain Physical Page Number

- The page table is used to translate the virtual page number to a physical page number.



Page Table Also Used to Determine Disk Address

- The page table is also used to keep track of where pages are located on disk.



Virtual Memory Questions

- *Where can a block be placed in main memory?* It can be placed anywhere (fully associative) to reduce the miss rate.
- *How is a block found if it is in main memory?* The page table is indexed by the virtual page number and contains the physical page number.
- *Which block should be replaced on a virtual memory miss?* Use bits are used to approximate LRU.
- *What happens on a write?* A write-back, write-allocate policy is always used.

Designing Virtual Memory Systems

- The page size should be large enough to avoid the high latency of disk access (32KB to 64KB are common for newly designed systems).
- Fully associative placement of pages is used to reduce the number of page faults.
- Page faults are handled by software (operating system) to use better algorithms for minimizing the number of page faults.
- A write-back policy is used (instead of write-through), so the number of disk accesses can be decreased.

Reducing the Storage for Page Tables

- Can keep limit registers to restrict the size of the page table. If the limit is exceeded, then the page table data is reallocated to a larger table, which allows the page table to grow.
- Often the page table is split into two parts, where one part contains the stack and the other contains the heap since both parts can grow. A high-order bit of the address can be inspected to determine which table to access.
- Another approach is to use an *inverted page table*, which has the number of entries of the pages in physical memory.
- Multiple levels of pages tables are sometimes used. The first level maps large blocks of pages sometimes called segments and indicates if any pages are resident in the accessed segment.
- The page table itself can be paged.

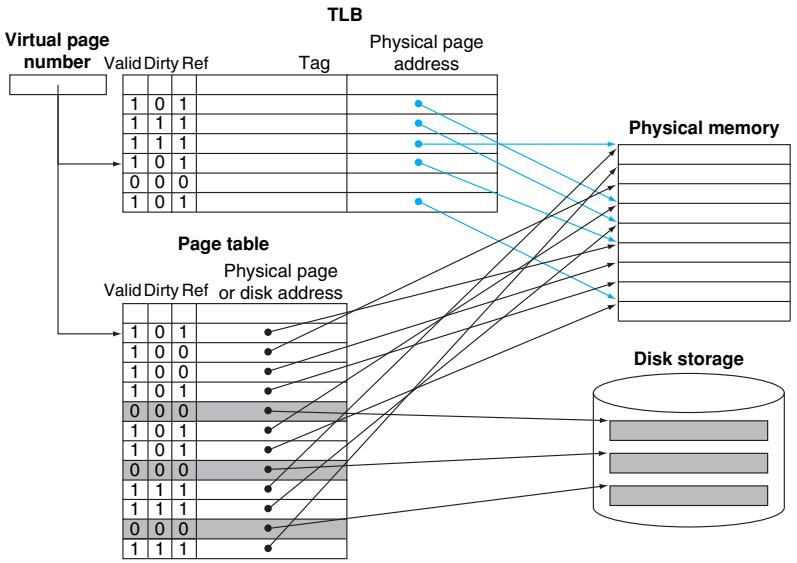
Handling Page Faults

- Obtain the location of the referenced page on disk in the page table entry.
- Choose a physical page to replace.
- If the page is dirty, write it back to disk.
- Read the referenced page from disk into the chosen physical page.
- Usually a context switch occurs on a page fault so that useful work can occur during the disk access to service the page fault.

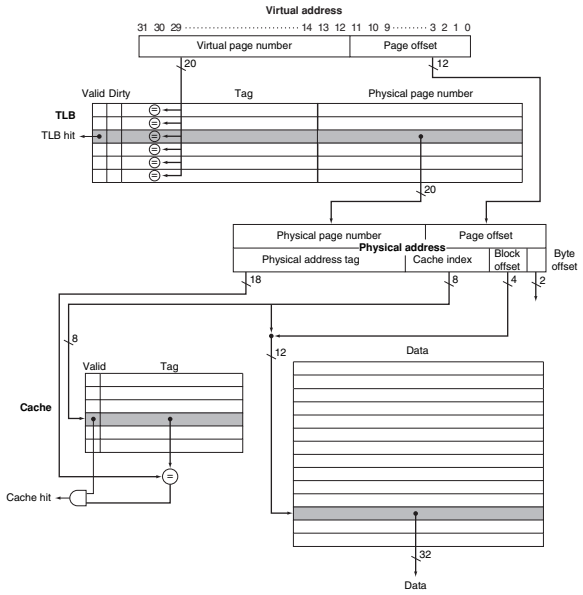
Translation Lookaside Buffers

- Most machines use special caches called TLBs that contain a portion of the entries in a page table.
- Each entry in a TLB contains a tag (portion of the virtual page number) and most of the information in a page table entry.
- The page table is accessed only when there is a TLB miss.
- TLBs are typically invalidated when a context switch is performed.
- TLBs are typically quite small to provide a very fast translation.
- There are typically separate TLBs for instructions (ITLB) and data (DTLB) to support simultaneous access due to pipelining.

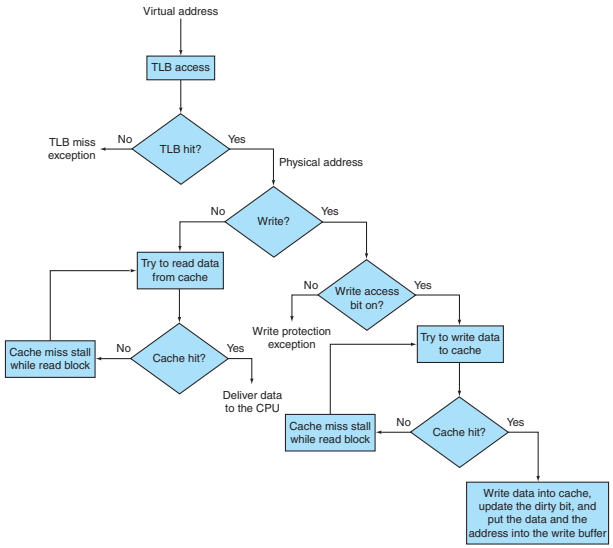
TLB Is Effectively a Cache for the Page Table



TLB and Cache in the Intrinsity FastMATH Processor



A Memory Access in the Intrinsity FastMATH Processor



Combinations of Events in the TLB, Page Table, and Cache

- The following tables depicts which combinations of TLB, page table, and cache events can occur.

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

Virtual Memory Supports Multiprogramming

- *Multiprogramming* means that several processes can concurrently share a computer.
- A *process* is the code, data, and any state information used in the execution of a program.
- A *context switch* means transfering control of the machine from one process to another.
- On a context switch either:
  - the TLB must be cleared
  - or tags in the TLB must be extended to include a PID (Process IDentifier).

TLB and Page Table Example

- Page size is 512 bytes. TLB is direct-mapped and has 64 sets.

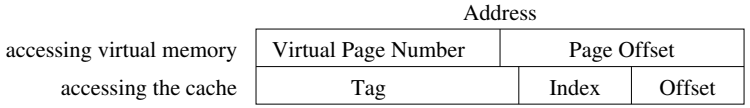
TLB			
Index	Page #	Tag	V
0	?	?	?
...	...	...	...
7	?	?	0
8	2	0	1
...	...	...	...
63	?	?	?

Page Table					
Index	Page #	Resident	Dirty	Disk Addr	...
0	?	?	?	?	...
...	...	...	...	...	...
71	9	Yes	?	?	...
...	...	...	...	...	...

- Given virtual address 0x8FDF, what is the physical address?
- Given virtual address 0x10DF, what is the physical address?

Virtually Indexed, Physically Tagged Caches

- Index into the cache using bits from the page offset.
- Do the tag comparison after obtaining the physical page number.
- Advantage is that the access to the data in the cache can start sooner.
- Limitation is that one way of a VIPT cache can be no larger than the page size.



Process Protection

- Proper protection must be provided to prevent a process from inappropriately affecting another process or itself.
- Page tables are kept in the address space of the operating system.
- Virtual memory systems keep bits in the page table and TLB entries to indicate the type and level of access that the process has to each of the pages.
- Two modes of execution are supported:
  - User mode is used by the process.
  - Kernel mode is used by the operating system. This mode can be used for updating page tables and accomplishing I/O.
- A switch from user to kernel mode is accomplished by a system call.
- A switch from kernel to user mode is accomplished by returning from an exception (system call).

Typical Design Parameters for Memory Hierarchy Levels

- Below are typical design parameters for different memory hierarchy levels in 2012.
- Server processors also have L3 caches, which are not shown.

Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	2500–25,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	125–2000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%

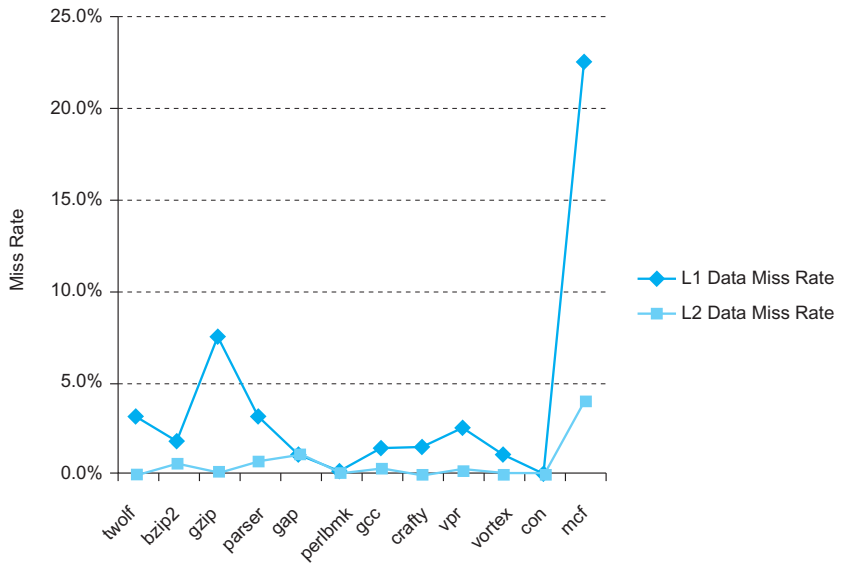
Example Address Translation Hardware

Characteristic	ARM Cortex-A8	Intel Core i7
Virtual address	32 bits	48 bits
Physical address	32 bits	44 bits
Page size	Variable: 4, 16, 64 KiB, 1, 16 MiB	Variable: 4 KiB, 2/4 MiB
TLB organization	1 TLB for instructions and 1 TLB for data Both TLBs are fully associative, with 32 entries, round robin replacement TLB misses handled in hardware	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, 7 per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB is four-way set associative, LRU replacement The L2 TLB has 512 entries TLB misses handled in hardware

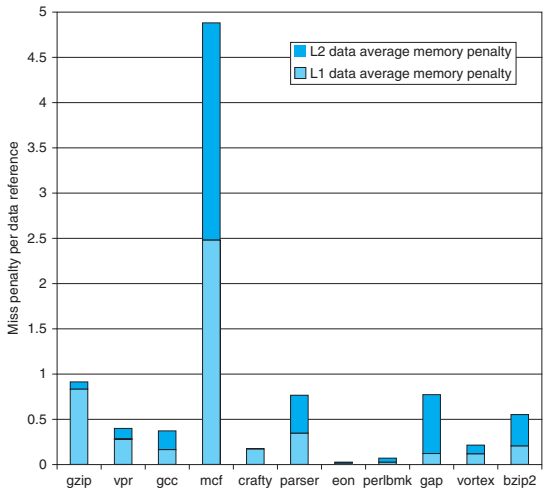
Example Caches

Characteristic	ARM Cortex-A8	Intel Nehalem
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	4-way (I), 4-way (D) set associative	4-way (I), 8-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate(?)	Write-back, No-write-allocate
L1 hit time (load-use)	1 clock cycle	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 1 MiB	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative	8-way set associative
L2 replacement	Random(?)	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate (?)	Write-back, Write-allocate
L2 hit time	11 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles

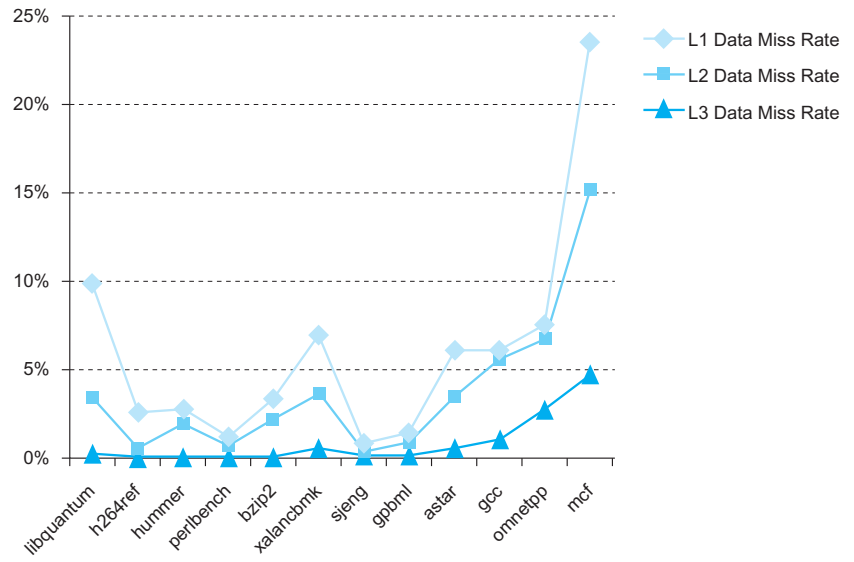
Data Cache Miss Rates for the ARM Cortex-A8



Average Memory Access Penalty per Data Ref on the ARM



Cache Miss Rates for the Intel i7 920



Fallacies and Pitfalls

- Pitfall: Ignoring memory system behavior when writing programs or when generating code in a compiler.
- Pitfall: Having less set associativity for a shared cache than the number of cores or threads sharing that cache.
- Pitfall: Using average memory access time to evaluate the memory hierarchy of an out-of-order processor.