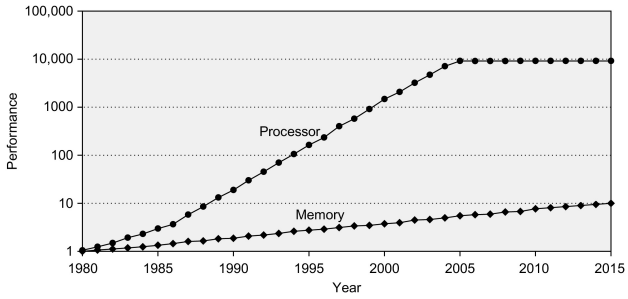

## Advanced Computer Architecture

- Memory Hierarchy
  - Why memory is relevant for Performance
  - Basics of Cache
  - Cache Performance Optimizations
    - Basic Optimizations
    - Advanced Optimizations

## ACA: Memory Hierarchy

### Relevance

- Memory Access Requests per Sec Vs Accesses in DRAM per Sec
- Peak Bandwidth grows as the number of cores grow
- Memory Bandwidth is the bottleneck




- Example: Intel Core i7 can generate 2 memory ref per cycle and it has 4 cores @3.2GHz
  Peak mem. Ref. per second: 25.6 Billion (64-bit) + 12.8 Billion instructions per sec (128-bit)
  Total Peak Bandwidth required: 409.6 GB/Sec!
  In contrast, the peak bandwidth to DRAM main memory is only 6% of this (25 GB/sec).



## ACA: Memory Hierarchy

### Solution

- Infeasible to satisfy required bandwidth with main Memory
- Bandwidth is achieved by
  - The use of multiple levels of caches
    - Using separate first & sometimes second-level caches per core
  - Using a separate instruction and data cache at the first level

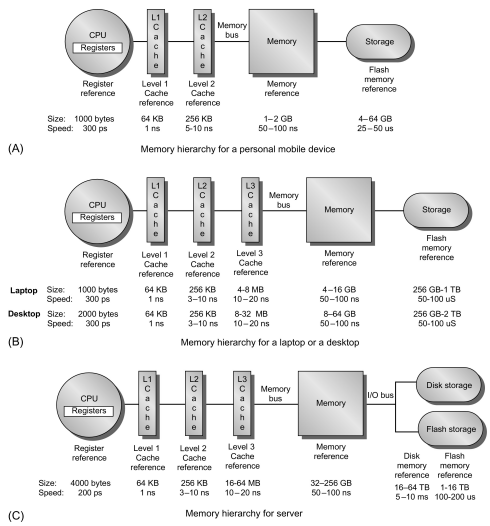
## ACA: Memory Hierarchy

### Solution –Memory Hierarchy

- Economical solution to the desire of unlimited and fast memory
- Takes advantage of locality and trade-offs in the cost-performance of memory technologies
  - The principle of locality says that most programs do not access all code or data uniformly.
  - Locality occurs in time (temporal locality) and in space (spatial locality)



## ACA: Memory Hierarchy




**The levels in a typical memory hierarchy in a personal mobile device (PMD), such as a cell phone or tablet (A), in a laptop or desktop computer (B), and in a server (C).**

## ACA: Memory Hierarchy

### Solution –Multilevel Memory Hierarchy

- Fast memory is expensive!
- Memory hierarchy is organized into several levels—each smaller, faster, and more expensive per byte than the next lower level, which is farther from the processor
- Designer's Goal: Provide a memory system with cost per byte almost as low as the cheapest level of memory and speed almost as fast as the fastest level.
- Inclusion Property: In most cases, the data contained in a lower level are a superset of the next higher level.
- Inclusion is always required for the lowest level of the hierarchy

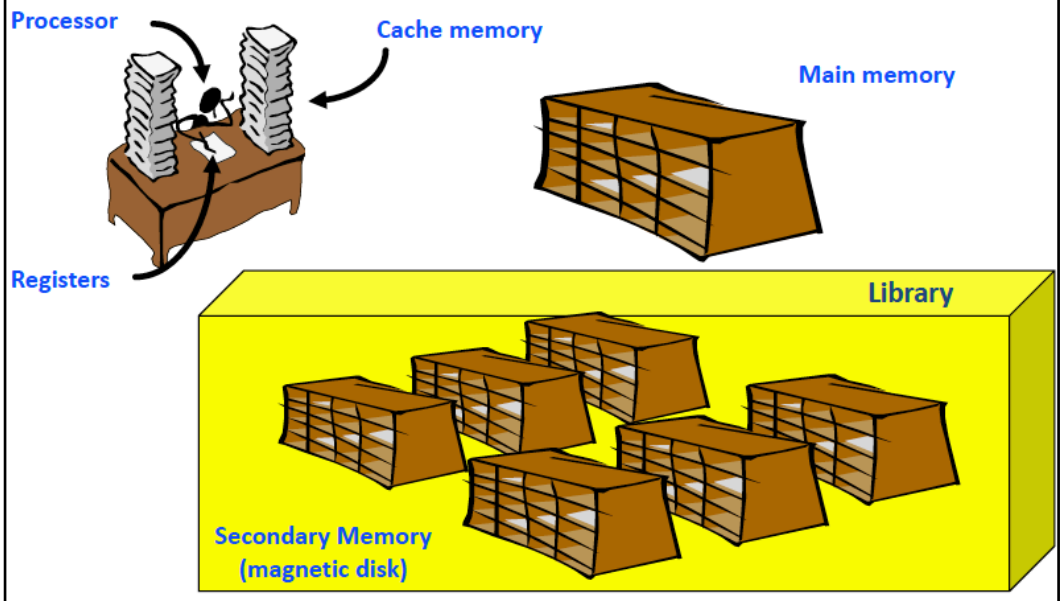
7

## ACA: Memory Hierarchy

- Two important requirements
  - Memory should be large
  - Memory should be fast
- Both requirements are contradictory to each other
  - Flat large and fast memory is not possible
- The library analogy

8

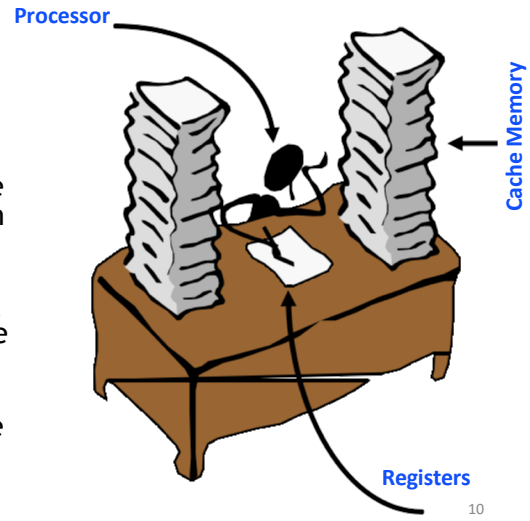
## ACA: Memory Hierarchy






## ACA: Memory Hierarchy

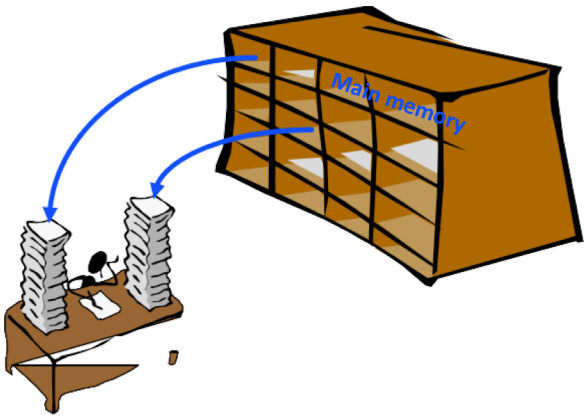
- Worker
  - works on available registers (pages)
  - Can rapidly access to data which is available on its desk in the form of books
  - The size of desk is limited and only small amount of data can be placed on it
  - Data from books can be accessed in a page by page manner




10

## ACA: Memory Hierarchy

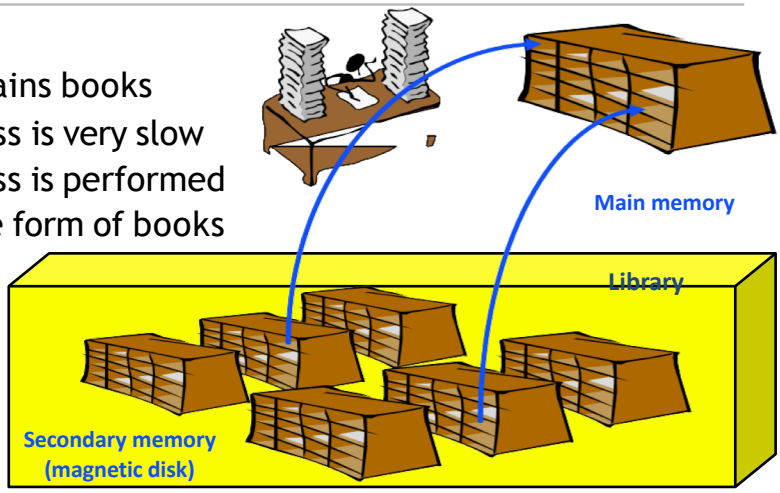
- Book shelf
  - Contains more books than on the desk
  - Access is slower
  - Access is performed on book titles not on pages
  - Books are brought to desk and then data is accessed in page by page manner






## ACA: Memory Hierarchy

- Library
  - Contains books
  - Access is very slow
  - Access is performed in the form of books



## ACA: Memory Hierarchy

- Similar to the library analogy
  - Memory is organized in hierarchy
  - Illusion of large and fast memory is created through hierarchy
- In library analogy, usually
  - If you reference a book, you are more likely to reference it again very soon (temporal locality)
  - If you reference a book in a particular section, you are more likely to reference other books lying in that section (spatial locality)



## ACA: Memory Hierarchy

- Similar to library analogy, multiple levels of memory hierarchy are created
- We need large and fast memory
- Fast memory is expensive
- We can not afford to have large blocks of fast memory
- Slow memory is cheap
- Large amount of slow memory is possible
- Faster but smaller memories are physically closer to the processor
- Larger but slower memories are farther from processor

## ACA: Memory Hierarchy

- Hierarchy levels

| Level | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Name | Registers | Cache | Main memory | Disk storage |
| Typical size | <1 KB | 32 KB–8 MB | <512 GB | >1 TB |
| Implementation technology | Custom memory with multiple ports, CMOS | On-chip CMOS SRAM | CMOS DRAM | Magnetic disk |
| Access time (ns) | 0.15–0.30 | 0.5–15 | 30–200 | 5,000,000 |
| Bandwidth (MB/sec) | 100,000–1,000,000 | 10,000–40,000 | 5000–20,000 | 50–500 |
| Managed by | Compiler | Hardware | Operating system | Operating system/ operator |
| Backed by | Cache | Main memory | Disk | Other disks and DVD |



## ACA: Memory Hierarchy

### Important Terminology for Memory Hierarchy

- Block: the minimum unit of information that can be either present or not present in the memory hierarchy
- Hit rate: the fraction of memory accesses found in cache
- Miss rate: the fraction of memory accesses not found in a certain level of memory (1 – hit rate)
- Hit time: time required to access a level of memory hierarchy when it is a hit
- Miss penalty: time required to replace a block of memory in upper level with a block from lower level plus the time to deliver the block to processor

## ACA: Memory Hierarchy

### Basics of Cache Memory

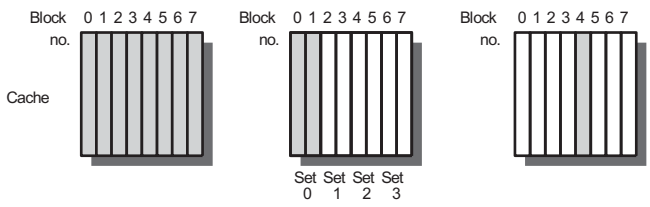
- Higher Level(s) in the hierarchy outside Processor
- Smaller in size, faster is access time
- Organized in different ways
  - Directly-mapped: Structure in which each memory location is mapped to exactly one location in the cache
  - Fully Associative: Structure in which each memory location can be mapped to any location in the cache
  - Set-Associative: Structure in which each memory location is mapped to any location with in the specific $set$ in the cache



## ACA: Memory Hierarchy

### Basics of Cache Memory

- Organized in different ways



## ACA: Memory Hierarchy

### Basics of Cache Memory

- FOUR Questions
  1. Where can a block be placed in the Cache? (Block Placement)
  2. How is a block found if it is in the Cache? (Block Identification)
  3. Which block should be replaced on a miss? (Block Replacement)
  4. What happens on a write? (Write Strategy)

19



## ACA: Memory Hierarchy

### Basics of Cache Memory

#### 1 Block Placement

- Block placement creates 03 categories
- If each block has only one place it can appear in the cache, the cache is said to be direct mapped.
- The mapping is usually

(Block address) MODULO (Number of blocks in cache)

20

## ACA: Memory Hierarchy

### Basics of Cache Memory

① Block Placement

- If a block can be placed anywhere in the cache, the cache is said to be fully associative.
- NO formulation is required to determine block placement



## ACA: Memory Hierarchy

### Basics of Cache Memory

① Block Placement

- If a block can be placed in a restricted set of places in the cache, the cache is set associative.
- A set is a group of blocks in the cache.
- A block is first mapped onto a set, and then the block can be placed anywhere within that set.
- The mapping is usually

  (Block address) MODULO (Number of sets in cache)

If there are *n* blocks in a set, the cache placement is called *n-way set associative*.

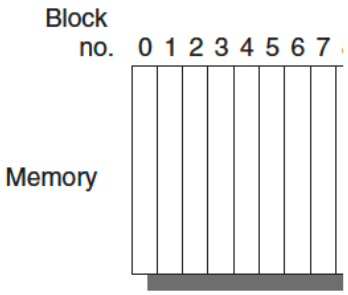
Dr. Khurram Bhatti, Associate Professor | Information Technology University (ITU) | Fall-2023

## ACA: Memory Hierarchy

### Basics of Cache Memory

#### ① Block Placement

- Example: Block address 12 has to be placed in cache of 8 blocks and 4 sets using all three categories:




1. Directly Mapped location?
2. Fully associative location?
3. Set Associative location?

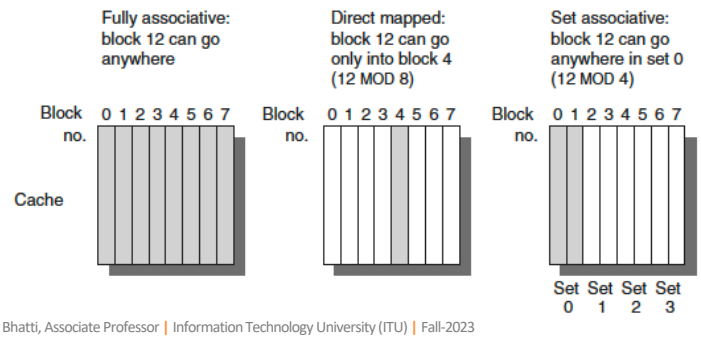


## ACA: Memory Hierarchy

### Basics of Cache Memory

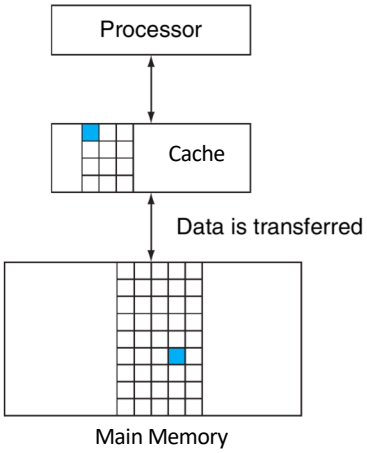
#### ① Block Placement –Example

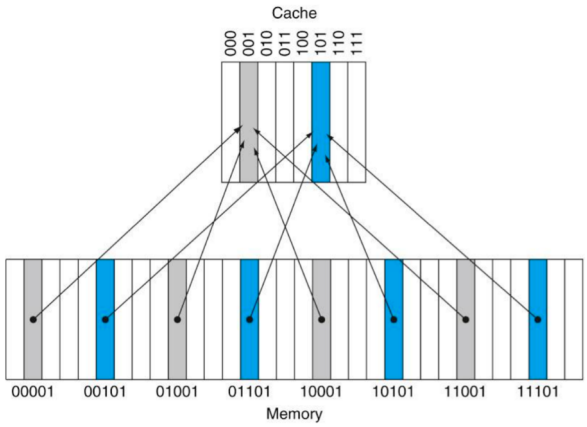
- Block address 12 has to be placed in cache of 8 blocks using all three categories:



# ACA: Memory Hierarchy

## Basics of Cache Memory

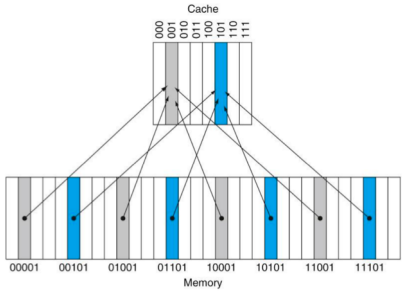


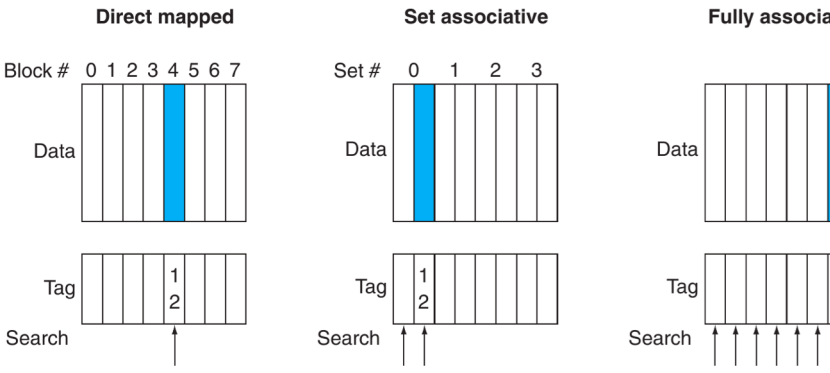






# ACA: Memory Hierarchy

## Basics of Cache Memory



ACA: Memory Hierarchy

## Basics of Cache Memory

### ①Block Placement -Generalized Associativity

- In general, *m* blocks in cache, *n* blocks in a set, *s* sets in cache
- In total : m = s * n
- n-way set associative => n> 1 and s> 1
  - Fully associative => n = m and s = 1 (m-way s.a.)
  - Direct mapped => n = 1 and s = m (1-way s.a.)
- Mapping: *set* number is called "index" also
  - Index (sets) = (block #) modulo (s)

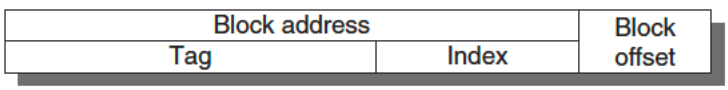


ACA: Memory Hierarchy

## Basics of Cache Memory

### ②Block Identification

- Divide the memory address into fields
  - Block address (higher-order bits)
    - Indicates the block number in memory we will access
  - Block offset (lower-order bits)
    - Indicates the data within a block we want to retrieve

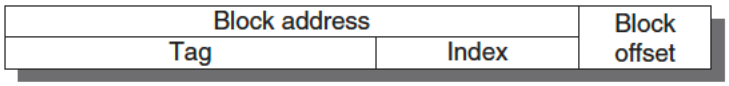


## ACA: Memory Hierarchy

### Basics of Cache Memory

#### ②Block Identification

- Divide the memory address into fields
  - Index field: Select the set (or entry in the cache)
    - Cache with $2^n$ sets requires an n-bit index
  - Tag field: compared for a hit
    - Many blocks map to the same entries
    - If the tag matches, the entry contains the right data





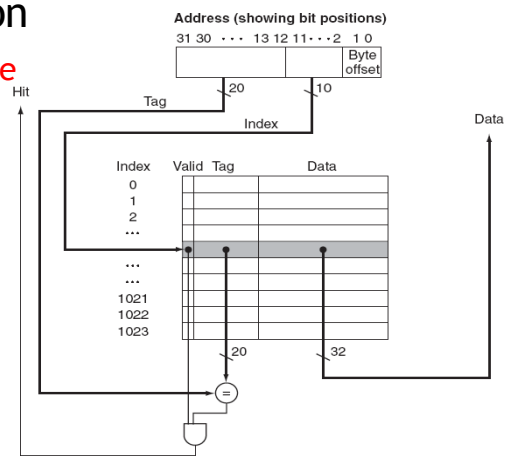



## ACA: Memory Hierarchy

### Basics of Cache Memory

#### ②Block Identification

- Directly mapped cache

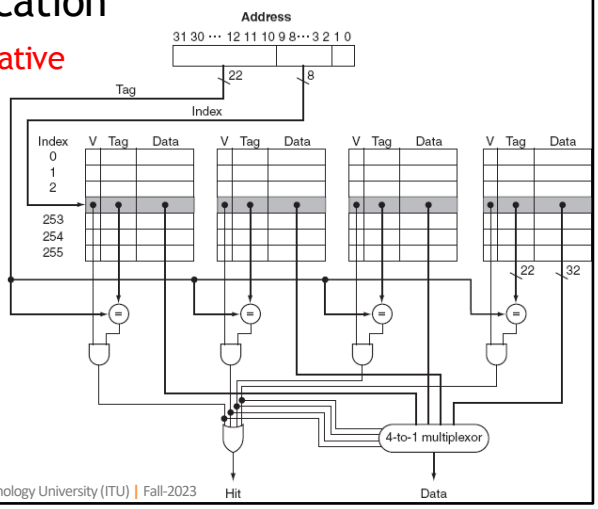


## ACA: Memory Hierarchy

## Basics of Cache Memory

### ②Block Identification

- 4-way set associative cache






## ACA: Memory Hierarchy

## Basics of Cache Memory

### ②Block Identification

- If the total cache size is kept the same, how will increasing associativity affect the following?
  - Number of Blocks per Set?
  - Index?
  - Tag?

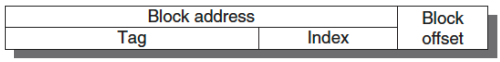
| Block address | | Block offset |
|---|---|---|
| Tag | Index | |

ACA: Memory Hierarchy

## Basics of Cache Memory

### ②Block Identification

- If the total cache size is kept the same, how will increasing associativity affect the following?
  - The number of blocks per set will increase (due to associativity)
  - Index size will decrease (because it tells # of Sets)
  - The tag size will increase
  - The tag-index boundary moves to the right with increasing associativity, with the end point of fully associative caches having no index field at all!






ACA: Memory Hierarchy

## Basics of Cache Memory

### ③Block Replacement

- When a miss occurs, the cache controller must select a block to be replaced with the desired data
- With Direct-mapped caches: No choice as such
  - Only one block frame is checked for a hit, and only that block can be replaced
- For Set-associative and Fully associative: 03 Strategies
  - Random
  - Least Recently Used (LRU)
  - First In, First Out (FIFO)

ACA: Memory Hierarchy

## Basics of Cache Memory

### ③Block Replacement

- For Set-associative and Fully associative: 03 Strategies
  - Random:
    - To spread allocation uniformly, candidate blocks are randomly selected
  - Least Recently Used (LRU) :
    - To reduce the chance of throwing out information that will be needed soon.
    - If recently used blocks are likely to be used again, then a good candidate for disposal is the least recently used block.
  - First In, First Out (FIFO):
    - Because LRU can be complicated to calculate, this approximates LRU by determining the oldest block rather than the LRU.



ACA: Memory Hierarchy

## Basics of Cache Memory

### ④Write Strategy

- Reads dominate processor's cache accesses
  - All 'instruction-accesses' are reads, and most instructions don't write to memory
- Write policies often distinguish cache designs
- There are two basic policies:
  - Write-Through (WT)
  - Write-Back (WB)

## ACA: Memory Hierarchy

### Basics of Cache Memory

#### ④Write Strategy

- There are two basic policies:
  - Write-through: The information is written (after computation) to both the block in the cache and to the block in the lower-level memory
    - Easier to implement than write-back.
    - Next lower level has the most current copy of the data, which simplifies data coherency
    - Cache is always clean!
    - Unlike write-back, read misses never result in writes to the lower level.



## ACA: Memory Hierarchy

### Basics of Cache Memory

#### ④Write Strategy

- There are two basic policies:
  - Write-back: The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced
    - Writes occur at the speed of the cache memory
    - Multiple writes within a block require only one write to the lower-level memory
    - Write-back is attractive in multiprocessors.
    - Write-back uses the rest of the memory hierarchy and memory interconnect less than write-through, it also saves power, making it attractive for embedded applications
    - Dirty bit feature used to reduce the frequency of writing back

## ACA: Memory Hierarchy

## Cache Performance Optimizations

- Cache Performance
  - A measure of memory hierarchy performance is the average memory access time:

$$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

  - Hit Time is the time to hit in the cache
  - Miss Rate is the percentage of cache miss
  - Miss Penalty is the number of cycles consumed per miss

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Cache Performance
  - A measure of processor performance in the presence of memory stalls would be:

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory stall clock cycles}) \times \text{Clock cycle time}$$

  - Hit time is supposedly included in the CPU execution clock cycles



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations
  - SIX basic optimizations based on the following Equation

$$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

  - THREE Categories
    - Reducing the Miss Rate: larger block size, larger cache size, and higher associativity
    - Reducing the Miss Penalty: multilevel caches and giving reads priority over writes
    - Reducing the hit time in the cache

ACA: Memory Hierarchy

## Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Causes of Miss Rate: 3C's Model
  - 03 Categories of Cache Misses
    - Compulsory Miss:
      - The very first access to a block cannot be in the cache
      - The block must be brought into the cache
      - Also called cold-start misses or first-reference misses
    - Capacity Miss:
      - If the cache cannot contain all the blocks needed during execution of a program, capacity misses (in addition to compulsory misses) will occur because of blocks being discarded and later retrieved
      - *Thrashing* Occurs!



ACA: Memory Hierarchy

Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Causes of Miss Rate: 3C's Model
  - 03 Categories of Cache Misses
    - Conflict Miss:
      - If the block placement strategy is set-associative or direct mapped, conflict misses (in addition to compulsory and capacity misses) will occur
      - A block may be discarded and later retrieved if too many blocks map to its set
      - These misses are also called collision misses

## ACA: Memory Hierarchy

## Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Frequency of Miss Rates under 3Cs:
    - Analyze the following
      - Impact of Cache size on 3Cs
      - Impact of Associativity on 3Cs



| Cache size (KB) | Degree associative | Total miss rate | Miss rate components (relative percent) (sum = 100% of total miss rate) Compulsory | | Capacity | | Conflict | |
|---|---|---|---|---|---|---|---|---|
| 4 | 1-way | 0.098 | 0.0001 | 0.1% | 0.070 | 72% | 0.027 | 28% |
| 4 | 2-way | 0.076 | 0.0001 | 0.1% | 0.070 | 93% | 0.005 | 7% |
| 4 | 4-way | 0.071 | 0.0001 | 0.1% | 0.070 | 99% | 0.001 | 1% |
| 4 | 8-way | 0.071 | 0.0001 | 0.1% | 0.070 | 100% | 0.000 | 0% |
| 8 | 1-way | 0.068 | 0.0001 | 0.1% | 0.044 | 65% | 0.024 | 35% |
| 8 | 2-way | 0.049 | 0.0001 | 0.1% | 0.044 | 90% | 0.005 | 10% |
| 8 | 4-way | 0.044 | 0.0001 | 0.1% | 0.044 | 99% | 0.000 | 1% |
| 8 | 8-way | 0.044 | 0.0001 | 0.1% | 0.044 | 100% | 0.000 | 0% |
| 16 | 1-way | 0.049 | 0.0001 | 0.1% | 0.040 | 82% | 0.009 | 17% |
| 16 | 2-way | 0.041 | 0.0001 | 0.2% | 0.040 | 98% | 0.001 | 2% |
| 16 | 4-way | 0.041 | 0.0001 | 0.2% | 0.040 | 99% | 0.000 | 0% |
| 16 | 8-way | 0.041 | 0.0001 | 0.2% | 0.040 | 100% | 0.000 | 0% |
| 32 | 1-way | 0.042 | 0.0001 | 0.2% | 0.037 | 89% | 0.005 | 11% |
| 32 | 2-way | 0.038 | 0.0001 | 0.2% | 0.037 | 99% | 0.000 | 0% |
| 32 | 4-way | 0.037 | 0.0001 | 0.2% | 0.037 | 100% | 0.000 | 0% |
| 32 | 8-way | 0.037 | 0.0001 | 0.2% | 0.037 | 100% | 0.000 | 0% |
| 64 | 1-way | 0.037 | 0.0001 | 0.2% | 0.028 | 77% | 0.008 | 23% |
| 64 | 2-way | 0.031 | 0.0001 | 0.2% | 0.028 | 91% | 0.003 | 9% |
| 64 | 4-way | 0.030 | 0.0001 | 0.2% | 0.028 | 95% | 0.001 | 4% |
| 64 | 8-way | 0.029 | 0.0001 | 0.2% | 0.028 | 97% | 0.001 | 2% |
| 128 | 1-way | 0.021 | 0.0001 | 0.3% | 0.019 | 91% | 0.002 | 8% |
| 128 | 2-way | 0.019 | 0.0001 | 0.3% | 0.019 | 100% | 0.000 | 0% |
| 128 | 4-way | 0.019 | 0.0001 | 0.3% | 0.019 | 100% | 0.000 | 0% |
| 128 | 8-way | 0.019 | 0.0001 | 0.3% | 0.019 | 100% | 0.000 | 0% |
| 256 | 1-way | 0.013 | 0.0001 | 0.5% | 0.012 | 94% | 0.001 | 6% |
| 256 | 2-way | 0.012 | 0.0001 | 0.5% | 0.012 | 99% | 0.000 | 0% |
| 256 | 4-way | 0.012 | 0.0001 | 0.5% | 0.012 | 99% | 0.000 | 0% |
| 256 | 8-way | 0.012 | 0.0001 | 0.5% | 0.012 | 99% | 0.000 | 0% |
| 512 | 1-way | 0.008 | 0.0001 | 0.8% | 0.005 | 66% | 0.003 | 33% |
| 512 | 2-way | 0.007 | 0.0001 | 0.9% | 0.005 | 71% | 0.002 | 28% |
| 512 | 4-way | 0.006 | 0.0001 | 1.1% | 0.005 | 91% | 0.000 | 8% |
| 512 | 8-way | 0.006 | 0.0001 | 1.1% | 0.005 | 95% | 0.000 | 4% |

## ACA: Memory Hierarchy

### Cache Performance Optimizations

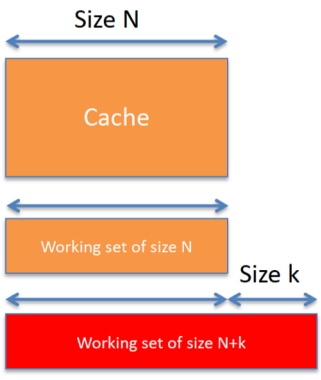
- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs: CAPCITY
    - Working Set (of Data): Data that is currently important for the program execution
    - If the working set is bigger than cache, no matter what the associativity, program will generate cache misses frequently!
    - Associativity is just the logical organization of how data is put into caches!



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs: CAPACITY



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs:
    - Effects of increasing the given cache parameters on each type of miss

| | Cache Parameters | | |
|---|---|---|---|
| Miss Type | Cache Size | Block Size | Associativity |
| Compulsory | | | |
| Capacity | | | |
| Conflict | | | |



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs:
    - Effects of increasing the given cache parameters on each type of miss

| | Cache Parameters | | |
|---|---|---|---|
| Miss Type | Cache Size | Block Size | Associativity |
| Compulsory | Unchanged / up | Down | Unchanged |
| Capacity | Down | Unchanged | Unchanged |
| Conflict | Unchanged | Unchanged / up | Down |

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs:
    - Cache Size Increased:
      - Compulsory miss will go up as there are more blocks to miss on in a cold cache (with a single block - you will only get one compulsory miss!)
      - Capacity miss will go down as there is more space to hold the working set of data for any given program! (regardless of associativity)
      - Conflict miss will remain unchanged as conflict comes from eviction, which can happen with small or large cache equally!



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs:
    - Block Size Increased:
      - Compulsory miss will go down as increasing the block size means more adjacent words will be fetched on each miss, so references to these words will not cause compulsory misses
      - Capacity miss remain unchanged because the increased block size will be compensated with decreased number of blocks
      - Conflict miss may increase as there is a greater chance to displace a useful block from the cache.

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - Miss Rates in terms of 3Cs:
    - Associativity Increased:
      - Compulsory miss will be unchanged as Associativity only affects how cache blocks are arranged, not how they are fetched from main memory!
      - Capacity miss remain unchanged because the total number of blocks remains the same no matter what the associativity (larger working set will always create capacity miss!)
      - Conflict miss may go down since conflict misses arise from blocks from main memory mapping to the same position in the cache



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations
  - SIX basic optimizations based on the following Equation

Average memory access time = Hit time + Miss rate × Miss penalty

  - THREE Categories
    - Reducing the Miss Rate: larger block size, larger cache size, and higher associativity
    - Reducing the Miss Penalty: multilevel caches and giving *read* priority over *write*
    - Reducing the hit time in the cache

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ① Larger Block Size to Reduce Miss Rate (weak)
    - Simplest way to reduce miss rate is to increase the block size
    - Larger block sizes will reduce also compulsory misses
    - Larger blocks take advantage of spatial locality
    - But larger blocks increase the miss penalty
    - Number of blocks in the cache are reduced
    - Larger blocks may increase conflict misses and even capacity misses if the cache is small
    - There is little reason to increase the block size to such a size that it increases the miss rate.
    - Increase in Miss Penalty can outweigh decrease in Miss Rate



## ACA: Memory Hierarchy

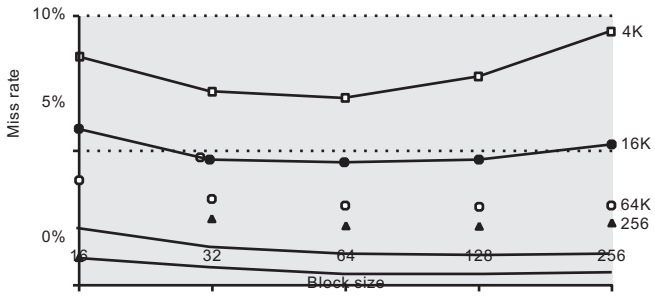
### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ① Larger Block Size to Reduce Miss Rate
    - The selection of block size depends on both the latency and bandwidth of the lower-level memory
    - High bandwidth & High Latency encourage large block size since the cache gets many more bytes per miss for a small increase in miss penalty (time).
    - Low bandwidth & Low Latency encourage smaller block sizes since there is little time saved from a larger block
    - Example: twice the miss penalty of a small block may be close to the penalty of a block twice the size

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ① Larger Block Size to Reduce Miss Rate




Each line represents a cache of different size.

Miss rate actually goes up if the block size is too large relative to the cache size.





## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ① Larger Block Size to Reduce Miss Rate

| Block size | Cache size 4K | 16K | 64K | 256K |
|---|---|---|---|---|
| 16 | 8.57% | 3.94% | 2.04% | 1.09% |
| 32 | 7.24% | 2.87% | 1.35% | 0.70% |
| 64 | 7.00% | 2.64% | 1.06% | 0.51% |
| 128 | 7.78% | 2.77% | 1.02% | 0.49% |
| 256 | 9.51% | 3.29% | 1.15% | 0.49% |

Actual miss rate versus block size for the five different-sized caches

Example: For a 4 KB cache, 256-byte blocks have a higher miss rate than 32-byte blocks

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ① Larger Block Size to Reduce Miss Rate

| | Cache size | | | |
|---|---|---|---|---|
| Block size | 4K | 16K | 64K | 256K |
| 16 | 8.027 | 4.231 | 2.673 | 1.894 |
| 32 | **7.082** | 3.411 | 2.134 | 1.588 |
| 64 | 7.160 | **3.323** | **1.933** | **1.449** |
| 128 | 8.469 | 3.659 | 1.979 | 1.470 |
| 256 | 11.651 | 4.685 | 2.288 | 1.549 |

Average memory access time versus block size for five different-sized caches
Example: Block sizes of 32 and 64 bytes dominate for smallest average time per cache size

56



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ② Large Caches to Reduce Miss Rate
    - Large caches reduce the capacity conflicts
    - Potential drawbacks
      - Longer hit time
      - Higher power consumption

57

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ③ Higher Associativity to Reduce Miss Rate
    - Increased associativity reduces conflict misses
    - Higher associativity is as good as full associativity in reducing miss rates
    - Direct mapped cache of size N has about same miss rate as 2-way set associative cache of size N/2
    - Drawback: higher associativity increases the hit time of the cache (just like larger block size increases miss penalty)



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - Cache performance formula assures that improvements in miss penalty can be just as beneficial as improvements in miss rate
    - Performance gap between processors and memory
      - Should the cache be faster to keep pace with the speed of processors?
      - Should the cache be larger to overcome the widening gap between the processor and main memory?

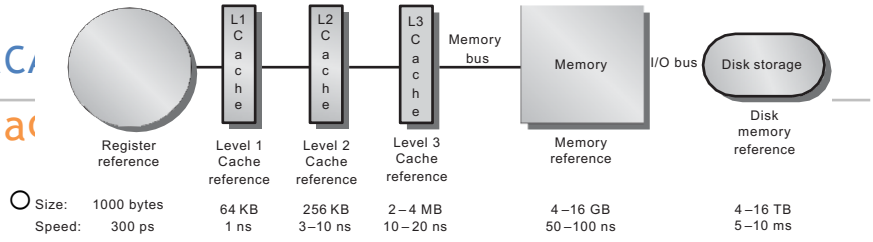
## ACA: Memory Hierarchy

## Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - Solution: Do Both!
    - Adding another level of cache between the original cache and memory simplifies the decision
      - First-level cache can be small enough to match the clock cycle time of the fast processor
      - Second-level cache can be large enough to capture many accesses that would go to main memory

60






- ④ Multi-level Caches to Reduce Miss Penalty
  - Average Memory Access Time (AMAT) for 2-level Caches

$$\text{Average memory access time} = \text{Hit time}_{L1} + \text{Miss rate}_{L1} \times \text{Miss penalty}_{L1}$$

$$\text{Miss penalty}_{L1} = \text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2}$$

$$\text{Average memory access time} = \text{Hit time}_{L1} + \text{Miss rate}_{L1} \times (\text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2})$$

**The second-level miss rate is measured on the leftovers from the first-level cache.**

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - To avoid ambiguity, these terms are adopted here for a two-level cache system:
      - Local Miss Rate (ref point is the cache in question)
        - The number of misses in a cache divided by the total number of memory accesses that reach to this cache
        - For the first-level cache it is Miss $rate_{L1}$, and for the second-level cache it is Miss $rate_{L2}$



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - To avoid ambiguity, these terms are adopted here for a two-level cache system:
      - Global Miss Rate
        - The number of misses in a cache divided by the total number of memory accesses generated by the processor
        - The global miss rate for the first-level cache is still the same Miss $rate_{L1}$, but for the second-level cache it is multiplicative Miss $rate_{L1}$ × Miss $rate_{L2}$.

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - For a particular application on 2-level cache hierarchy:
      - 1000 memory references
      - 40 misses in L1, 20 misses in L2
      - Hit time for L1 is 1, Hit time for L2 is 10
      - Miss Penalty for L2 is 100
      - Calculate AMAT?



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - For a particular application on 2-level cache hierarchy:
      - Calculating AMAT

- Local Miss rates
  - L1 => 40/1000 = 4%
  - L2 => 20/40 = 50%
- Global miss rates
  - L1 => 40/1000 = 4%
  - L2 => 20/1000 = 2%
- Avg. Memory Access Time (AMAT) = 1 + 4% x (10 + 50% x 100) = 3.4 cycles

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ④ Multi-level Caches to Reduce Miss Penalty
    - Global vs Local Miss Rate
      - Global miss rate is more useful measure:
      - It indicates what fraction of the memory accesses that leave the processor go all the way to memory
      - Local cache miss rate is not a good measure for secondary caches
        - It is a function of the miss rate of the first-level cache
        - Can vary by changing the first-level cache
        - Global cache miss rate should be used when evaluating second-level caches



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ⑤ Prioritizing Read Misses over Writes to Reduce Miss Penalty
    - Idea is to, somehow, serve reads BEFORE the writes could complete
    - In write through policy: updated information is first written in a *write buffer*, which then places the information in memory
      - On a read-miss , if the required information is in buffer, either wait for the buffer to be empty and then access main memory
      - Or read the data from buffer to reduce miss penalty
    - Similar technique can be used for write back policy

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ⑥ Avoiding Address Translation during Indexing to Reduce Hit Time
    - First we need to understand Virtual Memory!
      - Each process has its own *dedicated contiguous* address space
      - Physical memory is limited -sharing mechanisms among processes are required
      - V. Mem. automatically manages various levels of memory hierarchy and gives processes a transparent/contiguous view





## ACA: Memory Hierarchy

### Cache Performance Optimizations

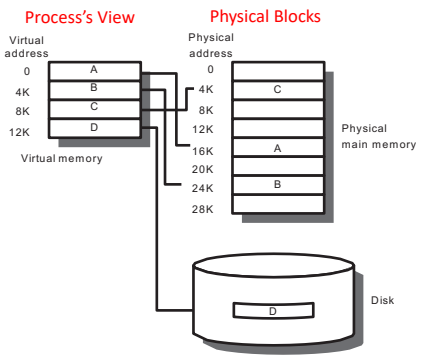
- Basic Cache Optimizations (06)
  - ⑥ Avoiding Address Translation during Indexing to Reduce Hit Time

Virtual Address in Pages (A, B, C, D)
Physical Address in Blocks

Page size = Block Size

Virtual address contains page number and an offset

Offset is common part in Virtual & Physical address



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ⑥ Avoiding Address Translation during Indexing to Reduce Hit Time
    - Address Translation: Processor produces virtual addresses that are translated by a combination of hardware and software to physical addresses to eventually access memory.
    - Caches must cope with address translation from virtual to physical addresses in order to access memory



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations (06)
  - ⑥ Address Translation during Indexing to Reduce Hit Time
    - In case of a CACHE HIT, two tasks are performed
      - Cache Indexing: selecting the set in which the address may be cached
      - Address comparison: tag matching
    - OFFSET Part that is common in physical and virtual address is used to index the cache
    - While cache is being indexed, the virtual part is translated for physical tag matching and thus a hit can be made slightly faster
    - This improves hit time (very slightly)

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations – Summary

| Technique | Hit time | Miss penalty | Miss rate | Hardware complexity | Comment |
|---|---|---|---|---|---|
| Larger block size | | – | + | 0 | Trivial; Pentium 4 L2 uses 128 bytes |
| Larger cache size | – | | + | 1 | Widely used, especially for L2 caches |
| Higher associativity | – | | + | 1 | Widely used |
| Multilevel caches | | + | | 2 | Costly hardware; harder if L1 block size ≠ L2 block size; widely used |
| Read priority over writes | | + | | 1 | Widely used |
| Avoiding address translation during cache indexing | + | | | 1 | Widely used |

## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Basic Cache Optimizations
  - Six basic optimizations cover hit time, miss rate, and miss penalty
  - Advanced optimizations add cache bandwidth and power consumption to the list of factors to be optimized



## ACA: Memory Hierarchy

### Cache Performance Optimizations

- Advance Cache Optimizations
  - 10 optimizations are divided into 05 categories
    - Reducing the hit time: small and simple first level caches, way prediction.
      - Both decrease power consumption too
    - Increasing cache bandwidth: pipelined caches, multi-banked caches, non blocking caches
      - Varying impact on power consumption

ACA: Memory Hierarchy

## Cache Performance Optimizations

- Advance Cache Optimizations
  - 10 optimizations are divided into 05 categories
    - Reducing the miss penalty: critical word first, merging write buffers.
      - Little impact on power consumption.
    - Reducing the miss rate: compiler optimizations. Improved power consumption
    - Reducing the miss penalty or miss rate via parallelism: Hardware prefetching, compiler prefetching.
      - Increase in power consumption due to unused prefetched data

ACA: Memory Hierarchy

## Assignment: Reading Assignment

- Read 10 advance cache optimizations
  - H&P Book, Chapter 2, section 2.2 in order to understand how these optimizations work

77