1 Memory Hierarchy

Processor (registers)	Fast, small, expensive
Cache	$\downarrow$
Main Memory	$\downarrow$
Disk	Slow, large, cheap

- **Hit**: data appears in some block in the upper levels of memory (closer to the processor)
  - **Hit Rate**: the fraction of memory access found in the upper level
  - **Hit Time**: the time to access the upper level = memory access time + time to determine hit/miss
- **Miss**: data needs to be retrieved from a block in the lower level of memory.
  - **Miss Rate**: 1 - (hit rate)
  - **Miss Penalty** = time to replace block in upper level + time to deliver block to processor
- **average access time** = hit time × hit rate + miss penalty × miss rate
- **average memory access time (AMAT)** = hit time + miss penalty × miss rate

2 Cache

2.1 Direct-Mapped Cache

- Each memory address is associated with one possible *block* within the cache
- **Block**: the unit of transfer between cache and memory
- Address in Direct-Mapped Cache

t a g	idx	o

  - **tag**: to check if have correct block
  - **index**: to select block (which "row")
  - **byte offset** within block
- larger cache blocks take better advantage of spacial locality
- every block has a *valid bit* determines whether anything stored in that row

2.1.1 Example

16KB of data in direct-mapped cache with 4 word blocks. Determine size of tag, index and offset fields (given 32-bit architecture)

- **index**: # blocks = 16 KB / 2^4 = 2^{10} B $\rightarrow$ 10 bits
- **byte offset**: 4 words = 16 bytes = 2^4 bytes $\rightarrow$ 4 bits
- **tag**: 32 - 4 - 10 = 18 $\rightarrow$ 18 bits

2.1.2 Impact of Large Block Size

- **benefit**: reduce miss rate (takes advantage of spacial locality)
- **drawbacks**: larger miss penalty (takes longer to load new block from next level) and possibly a higher miss rate

2.2 Fully Associative Cache

- no rows, basically a hash map
- **benefits**: no conflicts among different memory addresses
- **drawbacks**: need hardware comparator for every single entry: this is infeasible

2.3 N-Way Set Associative Cache

- memory address fields: tag (same as before), offset (same as before), index (points us to correct row (called a set in this case))
- cache is direct-mapped with respect to sets, each set is set associative
- has best of direct-mapped and fully associative

2.4 Examples: Number of Bits Needed

64 KB of data and one word blocks
64 KB = 16 K words = 2^{14} words = 2^{14} blocks

- **direct-mapped cache**
  block size = 4 bytes $\rightarrow$ offset size = 2 bits
  # sets = # blocks = 2^{14} $\rightarrow$ index size = 14 bits
  tag size = 32 - 14 - 2 = 16 bits
  bits/block = data + tag + valid = 32 + 16 + 1 = 49
  bits/cache = # blocks × bit/block = $2^{14} \times 49$ = 98 KB
- **4-way set associative**
  block size = 4 bytes $\rightarrow$ offset size = 2 bits
  # sets = # blocks/4 = $2^{14}/4 = 2^{12}$ $\rightarrow$ index size = 12 bits
  tag size = 32 - 12 - 2 = 18 bits
  bits/block = data + tag + valid = 32 + 18 + 1 = 51
  bits/cache = # blocks × bit/block = $2^{14} \times 51$ = 102 KB
- **increase associativity $\rightarrow$ increase bits in cache**
- **8 word blocks**
  64 KB = 2^{14} words = $2^{14}/8$ blocks = 2^{11} blocks
  block size = 4 words = 32 bytes $\rightarrow$ offset size = 5 bits
  # sets = # blocks = 2^{11} $\rightarrow$ index size = 11 bits
  tag size = address - index - offset = 32 - 11 - 5 = 18 bits
  bits/block = data + tag + valid = $8 \times 32 + 16 + 1$ = 273bits
  bits/cache = # blocks × bit/block = $2^{11} \times 273$ = 68.25 KB
- **increase block size $\rightarrow$ decrease bits in cache**

2.5 Accessing a Cache

- **cache hit**: cache block is valid and contains proper address, read the desired word
- **cache miss**: nothing in cache in appropriate block, fetch from memory
- **cache miss, block replacement**: wrong data is in cache at appropriate block, so discard it and fetch desired data from memory
- **Types of Cache Misses**
  - Compulsory Miss: occur when program first starts
  - Conflict Misses: occurs when two addresses map to the same cache location (problem in direct-mapped cache)
  - Capacity Misses: miss that occurs because the cache has limited size (problem in fully associative caches)

2.6 Block Replacement

- if there are any locations with valid bit of 0, usually write over that
- if all possible locations already have valid block, choose replacement strategy
  - random (simple to implement)
  - least-recently used (LRU) (expensive)

2.7 Mulit-Level Cache Hierarchy

Given 2 levels of cache:
AMAT = L1 hit time + L1 miss rate × **L1 miss penalty**
L1 miss penalty = L2 hit time + L2 miss rate × L2 miss penalty

3 Examples

3.1 Average Memory Address Time (AMAT)

Assume: hit time = 1 cycle, miss rate = 5%, miss penalty = 20 cycles
AMAT $= 1 + 0.05 \times 20 = 1 + 1 cycles = 2 cycles$