Lecture 18: Caches

I. Computer Memory System Overview
   A. Terms
      1. Word = typically size of int, 32 bits now.
      2. Addressable units: usually bytes, but can be words. 2^A = number of addressable units, where A is bits in an address.
      3. Unit of transfer = number of bits written out or read in to memory at a time. On Pentium, 64-bits.
   B. Characteristics of Memory Systems
      1. Location: internal vs. external
      2. Capacity: Number of bytes, number of words
      3. Units of transfer
      4. Access Methods
         a) Sequential: Use linear search to pass through intermediate records, O(n), e.g. tape drives.
         b) Direct access: Reach general vicinity, then sequential to exact location, O(n/tracks + sectors) e.g. hard disks.
         c) Random access: Each addressable location has unique, physically wired-in addressing mechanism, O(1), e.g. RAM.
         d) Associative (Content Addressable Memory or CAM): Random access that searches all memory in parallel for values of the data, O(1). Used for applications that need very high speed searches, e.g. computer networking switch MAC tables and router routing tables.
      5. Performance
         a) Access time (latency): in RAM the time to perform a read or write operation, for other devices it is the time to get to the desired location.
         b) Memory cycle time: in RAM, access time plus time until another access can be made because the circuit/memory must reset.
         c) Transfer rate = rate at which data can be transferred into or out of a memory unit. For non-RAM, timing starts after the access time.
      6. Physical Type: semiconductor, magnetic, optical, and magneto-optical (laser heats to Curie point so a magnet causes changes that are permanent when the medium cools)
      7. Physical Characteristics: Volatile vs nonvolatile, and eraseable vs. non-eraseable.
   C. Memory Hierarchy based on the trade-offs of quantity, speed, and cost.
      1. Forms a pyramid with small (in bytes), costly, fast registers at top, and huge (in bytes), cheap, slow tape drives at bottom.
      2. Key to success is the decreasing frequency of access as we move down the pyramid.
         a) Dependent on the principle of locality of reference. During the execution of a program, its memory references (both instruction and data) tend to cluster temporally. Caches rely on this for their success.

II. Cache Memory Principles
   A. Terms
      1. Memory *block* = unit of main memory stored in a cache line.
      2. Cache *line* = basic unit of a cache that contains a block of memory, a tag, and the control bits used to determine when the line should be replaced.
      3. Cache *tag* = a number stored in a line that is combined with the line's position in the cache to determine the address of the line's block in main memory.
      4. Cache *hit* = when the data for a CPU specified address is in a cache.
      5. Cache *miss* = when the data for a CPU specified address is not in a cache. This will force some line in the cache to be replaced with the desired data.
      6. *Dirty* = when a line contains updated data that differs from the corresponding main memory. Before overwriting, this data must be copied back to the main memory.
   B. Cache size is much, much smaller than main memory.
   C. Multi-level cache organizations work as long each succeeding cache is substantially larger, e.g. 32 kB, 256 kB, 6 MB.

III. Elements of Cache Design
   A. Cache Addresses: *Physical* (using actual main memory addresses), or *logical* that uses the virtual addresses that are used by the program and converted by the memory management unit (MMU) to actual physical address.
      1. Logical is faster because it doesn't have to wait for the MMU to make the conversion.

2. Logical must deal with many programs using the same virtual addresses.
   a) Either the entire cache must be flushed with each application context switch, or extra control bits must be added to each line to differentiate between the different applications.

B. Cache Size. The larger the cache, the more gates involved in addressing them, and slower. No optimum size.

C. Mapping Function

1. Given: number of lines in cache = $m = 2^r$, block size = 2^w bytes, then s = address length (in bits) – w = number of bits to specify the block numbers.
   a) For example, with a cache of 16 lines of 8 byte blocks, and 1MB = 2^{20} bytes of memory, then
      (1) $w = \log_2 8 = 3$ = LSB bits of address used to specify the location of each byte in an 8-byte block.
      (2) $s = 20 - 3 = 17$ = number of bits needed to specify an individual 8-byte block.
2. *Direct* = map each block of main memory to one possible cache line. Cache line number = block number mod cache size.
   a) Advantage of being simple to implement.
      (1) The line index will be the LSBs of the block number, and the tag will be MSBs of the block number.
      (2) For our example, tag length = 17 – 4 = 13 bits = number of bits remaining to be specified when we utilize the 4-bit line index as the LSBs of the block number.
   b) The disadvantage is that if during a period of time two frequently used blocks map to the same line, then that line must be written often to main memory instead of using other dormant lines. One solution is a victim cache that stores the most recently evicted lines. The victim cache is between the direct mapped cache and the next larger level of memory.
3. *Associative* = blocks can be placed in any line, with the tags containing the entire block number. To determine if a given address is in the cache, the cache control logic tests all tags simultaneously. This requires complex circuitry, but minimizes cache misses caused by mapping functions.
4. *Set Associative* = the cache is divided into *v* sets of lines with *k* lines in each set. This is known as a *k*-way set-associative mapping.
   a) The use of 2 lines per set is the most common set associative organization, though 4-ways make a modest additional improvement.
   b) For our example, we could divide the 16 lines into 8 sets for a 2-way set-associative mapping.
   c) With set-associative mapping, each block is mapped to a set using block number mod v. Similar to direct mapping, the set number bits contribute the LSBs of the block number, and the tag is the MSBs.
      (1) With our 8 sets, the last 3 bits could be determined from the set number, and the tag length would be 17 – 3 = 14 bits.
   d) Alternatively, this can be implemented as k direct-mapped caches with each cache containing one line from each of the v sets. Each block is mapped to the same line in each set using block number mod k to determine the line number. This method is typically used when the number lines per set, k, is small.
      (1) With our 8 sets, that last 3 bits of the block's number could be determined from the line number, and the tag length would be 17 -3 = 14 bits for the MSBs.

D. Replacement Algorithms

1. Least recently used (LRU). In 2-way set associative use a USE bit that is set when a line is referenced, and cleared in the other line in the set.
2. First in first out (FIFO) implements each set as a circular queue with the oldest at the front.
3. Least frequently used (LFU) by adding a counter to each line.
4. Random just picks a line at random. Works almost as well as any of the others!

E. Write Policy

1. Write through = all write operations are made to main memory as well as cache. The main disadvantageis that it generates a lot of memory traffic across the FSB.
2. Write back = when a line is replaced that line is copied back to main memory if its *dirty bit* is set. The *dirty bit* is set whenever the CPU updates a line. This has the disadvantage that main memory may not match the cache, and mislead DMA devices . So accesses by such devices must be made through the cache.
3. 15% of memory references are writes for normal programs. For high performance computers this may be 33% (vector-vector multiplication), or even 50% for matrix transposition.

F. Line Size
G. Number of Caches
    1. Single or two level
    2. Unified or split

IV. Pentium 4 Cache Organization

A. All of the Pentium processors include two on-chip L1 caches, one for data and one for instructions.
    1. The data cache is 16 Kbytes, with a line size of 64 bytes, and a four-way set associative organization. Uses write-back policy. Has an instruction that writes back and invalidates the caches for context switches.
    2. The instruction cache between instruction decode logic and the execution core because the microcode is more uniform, and isolates the cache from the slow decoding of the CISC to RISC.

B. L2 Cache feeds both L1 caches is eight-way set associative with a size of 512 kB and a line size of 128 bytes.

C. L3 cache is on-chip and is eight-way set associative with a line size of 128 bytes.

V. ARM Cache Organization

A. Use split set-associate cache. Line size, associativity, and addressing vary.

B. Has a 4-word FIFO write buffer between the L2 cache and main memory. Data written to the write buffer cannot be read until it has been stored in the main memory. This is the reason that it is small.