Computer Architecture
Homework 4
Theme: Memory to Cache mapping

All questions are weighted equally. You must show work to receive credit.

1. What are the differences among sequential, direct and random accesses? Give examples where each one of these mechanisms is used. Define memory access time, cycle time and transfer rate.

   In **sequential access**, the memory is organized into units called records, which must be accessed in a specific linear sequence.

   For **direct access**, however, the blocks or records are given an address based on their location, and they are reached by directly accessing a specific region and then sequentially searching an area, counting or waiting.

   For **random access**, each location had an address which is connected by a wiring mechanism, which gives access a constant time independent of the sequence of access which came before.

   Access time, also known as latency, is different for random and non-random-access memory. For random-access memory, it is the time that it takes to perform a read or write operation, which is the time from when the address is presented to the memory to when the data is stored or available for use. For non-random-access memory, it is the time it takes to position the read-write mechanism at the desired location.

   Memory cycle time is the access time for non-random access memory plus any other time that it takes before another access can begin. This is typically the time for transient signals on lines to die or to regenerate data if it is read constructively.

   Transfer rate is the rate at which data can be transferred into or out of memory. For random access memory, it is the inverse of the cycle time.

2. Briefly define direct, associative and set associative mapping. Compare and contrast their relative advantages and disadvantages. Why systems with extremely large cache sizes could actually become slower?

   Direct Mapping: This is the simplest cache mapping technique, where each block of main memory is mapped into exactly 1 cache line. Each block in the main memory maps into a single unique line in the cache. The following B lines in the main memory map into the cache similarly. The block B of main memory maps into line L of the cache; block B+1 maps into line L+1; so on and so forth. This mapping can be implemented using the main memory's address. In this address, there are three parts – Tag, Line, and Word. The Tag portion indicates whether the correct line of main memory data is currently mapped to a specific line in the Cache. The Line portion indicates which line in the Cache's tag should be checked for the data. The Word portion specifies the exact portion of the line in which the data that is requested is stored (a line may store more than 1 byte of data).

   Associative Mapping: Associative mapping avoids the issues of Direct Mapping by allowing a main memory block to be loaded into any of the lines of the cache. This way, the problem in

Direct Mapping of repeatedly having to swap in lines to the cache from main memory because they both map to the same line in the cache will be alleviated. This mapping is implemented using the main memory's address. In this address, there are two parts – Tag and Word. The Tag portion indicates whether the correct line of main memory data is currently mapped to a specific line in the cache. When an address is being checked against the Cache's data, every tag in the Cache is queried to check if it contains the data for the requested address. The Word portion specifies the exact part of the line in which the data that is requested is stored (a line may store more than 1 byte of data).

Set Associative Mapping: This mapping combines the ideas of the 2 above approaches to improve performance and decrease disadvantages. In this mapping, the cache is split into some amount of sets. Each set is then split into some number of lines. These lines contain some number of words of data. The main memory's address is split into three parts – Tag, Set, and Word. The Set portion identifies which set in the cache should be checked for the data. Within that set, the Tag portion of the address will be checked against each of the lines in the chosen set for matches. In the case of a matching tag, the Word portion will identify exactly where the requested data is in the chosen line (a line may store more than 1 byte of data).

3. A 2-way set associative cache has lines of 16 bytes and a total size of 8KB. The 64 MB main memory is byte addressable. Show the format of main memory addresses.

Given, main memory size is 64 MB and memory is byte addressable
=> memory word size = 1 Byte
=> number of words in main memory = 64 MB / 1 Byte = 64 M = 2^{26}
=> main memory address length is 26 bits (4 points)

Given, cache line size = memory block size = 16 Byte
=> number of words in cache = 2^{4}
=> cache address length is 4 bits
=> number of blocks in memory = $2^{26} / 2^{4} = 2^{22}$
=> block address length is 22 bits; it needs to be split into tag and set address. (4 points)

Given, cache size is 8 KB
=> number of cache line = 8 KB / 16 Byte = 2^{9}
Given, cache is 2-way set-associative
=> number of cache line per set = 2
=> number of sets in cache = $2^{9} / 2 = 2^{8}$
Hence, cache set address length is 8 bits (4 points)
Hence, cache tag address length is 22 bits – 8 bits = 14 bits (4 points)

Memory address of 26 bits is organized as 14 bits tag + 8 bits set + 4 bits word address.

4. Given the following specifications for an external cache memory: 4-way set associative; line size of two 16-bit words; able to accommodate a total of 4K 32-bit words from main memory; used with a 16-bit processor that issues 24-bit addresses, design the cache structure with all pertinent information and show how it interprets the processor's addresses.

   Given, 24-bit addresses
   => main memory size = 2^{24} words

   Given, line size is two 16-bit words
   => main memory block size = 32 bits = 2 words = 4 bytes
   => word address length is 2 bits

   Given, cache is 4-way set associative
   => cache set size is 4 lines
   => number of sets in cache = number of lines in cache / cache set size
   = 4K / 4 = 1 K = 2^{10}
   => set address length is 10 bits
   => tag address length is 24 – 10 – 2 = 12 bits

5. A set associative cache has a block size of four 16-bit words and a set size of 2. The cache can accommodate a total of 4096 words. The main memory size that is cacheable is 64K x 32 bits. Design the cache structure and show how the processor's addresses are interpreted.

   Given, main memory size is 64K×32 bits
   => main memory = 64K × 32 bits = 64K × 4 Bytes = 256KB = 2^{18}
   Thus, the total address length is 18 bits (3 points)

   Given, cache size is 4096 words;
   => Total number of lines = cache size / words per line = 4096 / 4 = 1024
   => Total number of sets = number of lines / lines per set = 1024 / 2 = 512 = 2^9
   Thus, set length is 9 bits (3 points)

   Given, block size is four 16-bit words
   => block size = 4 × 16 bits = 8 Bytes = 2^3
   Thus, the word offset length is 3 (3 points)
   Thus, tag length = total address length – set length – word offset length
   = 18 – 9 – 3 = 6 bits (3 points)
   Thus, the address format is 6-bit tag + 9-bit line + 3-bit word offset

6. Consider a memory system that uses a 32-bit address to address at the byte level, plus a cache that uses a 64-byte line size.
   a) Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

      Given, line size is 64 bytes
      => line size = 64 bytes = 2^6
      Thus, line length is 6 bits (1 points)
      Given, total address length is 32 bits; tag length is 20 bits
      => offset = total address length – line length – tag length = 32 – 20 – 6 = 6 bits
      Thus, the address format is 20-bit tag + 6-bit line + 6-bit word offset (1 points)

      Number of addressable units = 2^{32} bytes = 4GB (1 points)
      Number of blocks in main memory = main memory size / block size
      = $2^{32} / 2^6 = 2^{26}$ (1 points)
      Number of lines in cache = $2^6 = 64$ (1 points)
      Size of tag = 20 bit (1 points)

   b) Assume an associative cache. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

      Number of addressable units = 2^{32} bytes = 4GB (1 points)
      Number of blocks in main memory = main memory size / block size
      = $2^{32} / 2^6 = 2^{26}$ (1 points)
      For an associative cache, the number of lines in cache is undetermined (1 points)
      Size of tag = total address length – word offset length = 32 – 6 = 26 (1 points)

   c) Assume a four-way set-associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in set, number of sets in cache, number of lines in cache, size of tag.

      Given, total address length is 32 bits; tag length is 9 bits
      => set length = total address length – tag length – word offset
      = 32 – 9 – 6 = 17 (1 points)
      Thus, the address format is 9-bit tag + 17-bit set + 6-bit word offset. (1 points)

      Number of addressable units = 2^{32} bytes = 4GB (1 points)
      Number of blocks in main memory = main memory size / block size
      = $2^{32} / 2^6 = 2^{26}$ (1 points)
      Number of lines in set = 4 (4-way set-associative)
      Number of sets in cache = 2^{17}
      Number of lines in cache = number of lines in set × number of sets
      = $4 \times 2^{17} = 2^{19}$ (1 points)
      Size of tag = 9 bits (1 points)