COMP7300 Fall 2021 HW04

Libo Sun lzs0101@auburn.edu
Compute Science and Software Engineering
Auburn University, Auburn, AL, U.S.

1. (a) **What are the differences among sequential, direct and random accesses? Give examples where each one of these mechanisms is used.**
   - **Sequential access:** Memory is organized into units of data called records, Access must be made in a specific linear sequence and the access time is highly variable. **Tape units**, e.g., magnetic tape, are sequential access.
   - **Direct access:** As sequential access, it involves a shared read-write mechanism.Individual blocks or records have a unique address based on physical location. Access time is variable. **Disk units**, e.g., magnetic disk, CD-ROM, are direct access.
   - **Random access:**Each addressable location in memory has a unique, physically wired-in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant. Thus, any location can be selected at random and directly addressed and accessed. **Main memory and some cache systems**, e.g., DRAM, are random access.

   (b) **Define memory access time, cycle time and transfer rate.**
   - **Memory access time(latency):** For random-access memory, it is defined as the time to perform a read or write operation. The time starts from the instant that an address is presented to the memory to the instant tat data have been stored or made available for use. For non-random-access memory, access time is the time it takes to position the read-write mechanism at the desired location.
   - **Memory cycle time:** The time includes the access time plus any additional time required before the next access an commence. Additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively. Concerned with the system bus,not the processor.
   - **Transfer Rate:** This is the rate at which data can be transferred into or out of a memory unit. For random-access memory it is equal to **1/(cycle time)**. For non-random-access memory:
   $$T_n = T_A + \frac{n}{R}$$
     - T_n = Average time to read or write n bits
     - T_A = Average access time
     - n = Number of bits
     - R = Transfer rate, in bits per second(bps)

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?**

(a) **Direct mapping:**

i. The simplest technique, it maps each block of main memory into only one possible cache line. The mapping is expressed as:

$$i = j \mod m$$

where

- i = cache line number
- j = main memory block number
- m = number of lines in the cache

ii. **Advantage:** Simple and inexpensive to implement.

iii. **Disadvantage:** There is a fixed cache location for any given block. Thus,if a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low (a phenomenon known as thrashing).

(b) **Associative mapping:**

i. The cache control logic interprets a memory address simply as a Tag and a Word field. The Tag field uniquely identifies a block of main memory. To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's tag for a match.

ii. **Advantage:** With associative mapping, there is flexibility as to which block to replace when a new block is read into the cache.

iii. **Disadvantage:** The complex circuitry required to examine the tags of all cache lines in parallel.

(c) **Set associative mapping:**

i. Set associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages. In this case, the cache consists of number sets, each of which consists of a number of lines. The relationships are:

$$m = v \times k$$

$$i = j \mod v$$

Where,

- i = cache set number
- j = main memory block number
- m = number of lines in the cache
- v = number of sets
- k = number of lines in each set

This is referred to as k-way set associative mapping. With set associative mapping, block B_j can be mapped into any of the lines of set j.

ii. **Advantage:** Significantly improves the hit ratio over direct mapping. Four-way set associative (v = $m/4$, k = 4) makes a modest additional improvement for a relatively small additional cost. Further increases in the number of lines per set have little effect.

iii. **Disadvantage:** The complexity of the cache increases in proportion to the associativity, and in this case would not be justifiable against increasing cache size to 8 or even 16 kB. A final point to note is that beyond about 32 kB, increase in cache size brings no significant increase in performance.

(d) Increasing the cache size doesn't equate to increase in overall performance. For cache to be fast, it has to be very close to the CPU core. If it is big, it won't fit that close. The latency of looking up a higher size cache for a hit is slower than looking up a smaller size cache.

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.**

   - Number of lines in set: $k = 2$
   - Given, Block size = line size = $2^w = 16$ (bytes), hence, $w = 4$
   - Given, Memory size 64MB addressable, $64MB = 2^{26}$, hence, address is 26 bits
   - Compute set size d
     - Cache size $k \times 2^{d+w} = 8KB = 2^{13}$
     - Since we have $k = 2$ and $w = 4$
     - Hence, we can get $2^{d+5} = 2^{13}$
     - In end, $d = 8$
   - Compute size of tag
     - Having: $s + w = 26$ and $w = 4$
     - $s = 22$
     - With $d = 8$
     - Conclude Tag size: $s - d = 14$
   - Memory address format 26 bits:

Tag ($s-d$)	Set (d)	Word (w)
14	8	4

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, 4-way set ==> $k = 4$
   - Given 24-bit addresses ==> Memory is 24 bits address
   - Given line size two 16-bit words
     ==> $linesize = 2^w \, bytes = 2 \times 16 \, bits = 32 \, bites$
     ==> $2^w \, bytes = 4 \, bytes$
     ==> Word size: $w = 2$ bits
   - Given cache size is 4k 32-bits
     ==> the number of lines: $4k/4 = 2^{12}/4 = 2^{10}$, hence, $d = 10$
   - Tag size = $s - d = (24 - 2) - 10 = 12$ bits
   - Memory address interpretation

Tag ($s-d$)	Set (d)	Word (w)
12	10	2

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 cache-able is 64K x 32 bits. Design the cache structure and show how the processor's addresses are interpreted.**

   - Given, $k = 2$ (set size 2), $w = 4$ (four 16-bit words) and cache 4096 ($4k$) words
     ==> Number of lines: $2^d = 4k \div 4 \div 2 = 2^9$, hence, $d = 9$
   - Given,block size four 16 bits
     ==> $2^w = 4 \times 16bits = 2^3 bytes$, hence, $w = 3$
   - Given cache-able memory $64k \times 32bits$
     ==> $64k \times 32bits = 2^{18} bytes$
     ==> Size of tag: $18 - 9 - 3 = 6$
   - Interpretation:

Tag ($s-d$)	Set (d)	Word (w)
6	9	3

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.*

   - **number of addressable units**: 2^{32} bytes = 4GB
   - **number of blocks in main memory**:
     Given, line size = 64
     ==> 2^w = 64 ==> $w = 6$
     ==> blocks: 2^{32-6} = 2^{26}
   - **number of lines**: $2^{32-20-6} = 2^6 = 64$
   - **size of tag**: 20

   (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
   - **number of blocks in main memory**:
     Given, line size = 64
     ==> 2^w = 64 ==> $w = 6$
     ==> blocks: 2^{32-6} = 2^{26}
   - **number of lines**: not determined
   - **size of tag**: $32 - 6 = 26$

   (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.*

   - **number of addressable units**: 2^{32} bytes = 4GB
   - **number of blocks in main memory**:
     Given, line size = 64
     ==> 2^w = 64 ==> $w = 6$
     ==> blocks: 2^{32-6} = 2^{26}

- **number of lines in set**: ==> $k = 4$
- **number of sets in cache**:
  ==> $d = Memory\, address - w - Tag\, size = 32 - 6 - 9 = 17$
  ==> Number of sets in cache: $2^d = 2^{17}$
- **Number of lines in cache**: $k \times sets = 4 \times 2^{17} = 2^{19}$
- **size of tag**: 9