Computer Science M151B, Homework 9

Michael Wu
UID: 404751542

June 6th, 2018

Problem 1

If we choose a write-around policy, we have the following operation times.

1. Read hits will take 6ns, because they need only one cache access.
2. Read misses that replace a dirty block will need one cache access for the initial miss. Then they need to read from the cache four times, write to main memory four times, read from main memory four times, and write to the cache four times. Overall this takes $6 + 8 \times (6 + 65) = 574$ns.
3. Read misses that replace a clean block will need one cache access for the initial miss. Then they need to read from main memory four times and write to the cache four times. Overall this takes $6+4\times(6+65) = 290$ns.
4. Writes only need to write to main memory once, then access the cache to unset the valid bit. Overall this takes $6 + 65 = 71$ns.

Assuming that dirty blocks are uniformly distributed throughout the cache, the average access time over all memory access is the following.

$$0.65(0.9 \times 6 + 0.1(0.7 \times 574 + 0.3 \times 290)) + 0.35 \times 71 = 60.132\text{ns}$$

If we choose a write-allocate policy, we have the same operation times except in the case of a write. Then we have the following write types.

1. Write hits will take 6ns, because they need only one cache access.

2. Write misses that replace a dirty block will need one cache access for the initial miss. Then they need to read from the cache four times, write to main memory four times, read from main memory three times, and write to the cache four times. Overall this takes $6 + 7 \times (6 + 65) + 6 = 509$ns.

3. Write misses that replace a clean block will need one cache access for the initial miss. Then they need to read from main memory three times and write to the cache four times. Overall this takes $6+3\times(6+65)+6 = 225$ns.

Assuming that dirty blocks are uniformly distributed throughout the cache, the average access time over all memory access is the following.

$$0.65(0.95 \times 6 + 0.05(0.75 \times 574 + 0.25 \times 290)) \\ + 0.35(0.95 \times 6 + 0.05(0.75 \times 509 + 0.25 \times 225)) = 29.7125\text{ns}$$

Therefore we should go with the second option, as it will lead to better performance.

Problem 2

If the memory is byte addressable, then there are 13 bits of byte offset. Then there are 19 bits for the virtual page number. This means that for each process, there is a maximum of 2^{19} page table entries. Thus the maximum size of the page table entries for each process is 2^{21} bytes. Even though each process only uses half of the total possible memory available, a single level page table system must contain space for every possible address. Since there will be five page tables, the total page table size will be 10 MiB.

Problem 3

The upper level page table will use the upper 8 bits of the virtual address, and it takes up 1536 bytes. Then the next 11 bits are used for the lower level page table, since there are 13 bits of byte offset. Each lower lever page table takes up $4 \times 2^{11} = 8192$ bytes. At a minimum there are 128 valid entries in the upper level page table that are used, since each process only uses half of

the available memory. Then this is $1536 + 128 \times 8192 = 1050112$ bytes per process, and 5250560 bytes overall. This is about 5 MiB. At a maximum, all lower level page tables have some valid entries in them, meaning that there are 256 valid entries in the upper level page table that are used. Then we have $1537 + 256 \times 8192 = 2098689$ bytes per process, and 10493445 bytes overall. This is about 10 MiB.

Problem 4

a) We have the following address accesses in binary.

101111000000010010
001101000100100000
000000010101010000
001101010110001011
000111000110010100
110000000000000000
000111000001010010
000101010101010000
101111101000000010
000000011101000100
000111001001110110

The upper 6 bits will be the virtual page number. Then the underlined instructions will cause a page fault.

2f012, d120, 550, d58b, 7194, 30000, 7052, 5550, 2fa02, 744, 7276

b) The underlined instructions will cause a page fault.

2f012, d120, 550, d58b, 7194, 30000, 7052, 5550, 2fa02, 744, 7276

Problem 5

a) You would also need the percent of cache accesses that are to dirty blocks, as well as the implementation of the page table. Multi-level page

table implementations would require more memory accesses in the event of a TLB miss.

b) Assume that every read miss is on a dirty block and we use a single level page table. Then we have the following operation times.

1. Read hits with a TLB hit will need one TLB access and one cache access, which will take 14ns.

2. Read hits with a TLB miss will need two TLB accesses, one memory access, and one cache access, which will take 129ns.

3. Read misses with a TLB hit will replace a dirty block. Thus they will need one TLB access and one cache access for the initial miss. Then they need to read from the cache eight times, write to main memory eight times, read from main memory eight times, and write to the cache eight times. Overall this takes $14 + 16 \times (9 + 110) = 1918$ns.

4. Read misses with a TLB miss will replace a dirty block. Thus they will need two TLB access, one memory access, and one cache access for the initial miss. Then they need to read from the cache eight times, write to main memory eight times, read from main memory eight times, and write to the cache eight times. Overall this takes $129 + 8 \times (9 + 110) = 2033$ns.

5. Writes with a TLB hit will need one TLB access, one memory access, and one cache access to unset the valid bit, which will take 124ns.

6. Writes with a TLB miss will need two TLB accesses, two memory accesses, and one cache access to unset the valid bit, which will take 239ns.

Then we get the following effective access time.

$$0.6 \times (0.98(0.94 \times 14 + 0.06 \times 1918) + 0.02(0.94 \times 129 + 0.06 \times 2033)) \\ + 0.4 \times (0.98 \times 124 + 0.02 \times 239) = 128.844\text{ns}$$

Problem 6

There are 13 bits of page offset. So the virtual page number is a 51 bit number. The physical page number is a 19 bit number. There are 64 sets in the TLB, so 6 bits will be used as the index in the TLB. This leaves 45 bits as the tag in the TLB. Assume there is a valid bit, a dirty bit, and a reference bit in the TLB. Then each TLB entry takes $3 + 45 + 19 = 67$bits. Then the total size of memory needed for the TLB is 67 KiB.

Problem 7

a) Let the following variables hold.

$$A = \texttt{X[0]++;}$$
$$B = \texttt{print X[1];}$$
$$C = \texttt{X[1]+=2;}$$
$$D = \texttt{print X[0];}$$

Because cache coherence is maintained, there are five possible execution sequences assuming no reordering of statements by the compiler or processor.

$$ABCD$$
$$ACBD$$
$$CABD$$
$$CADB$$
$$CDAB$$

This leads to the possible values $(0, 1), (2, 1), (2, 0)$.

b) One possible output value that may output if cache coherence is not maintained is $(0, 0)$. This occurs because each process operates using a separate cache, so they do not see the changes that happen in the other process. So they each print out zero.

c) Assume that one process executes completely before the other one. Then there will be two compulsory misses that occur, one on the first statement of each process. No additional misses will occur since the processes do not

invalidate each other's caches. Thus the minimum number of misses will be 2.

d) Assume that both processes initially load the block into memory. This has two compulsory misses. Then one process writes. This invalidates one cache, adding a miss. Then the next process writes. This invalidates the other cache, adding a fourth miss. Then both processes print. These do not cause additional misses since they are reads. So the maximum number of misses will be 4.