University of California Irvine

EECS 112: Organization of Digital Computers (Spring 2022)

Final Exercises

1. Consider the following program and cache behaviors.

Data Reads per 1000 instructions	Data Wirtes per 1000 instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (bytes)
250	100	0.30%	2%	64

   (a) Suppose a CPU with a write-through, write-allocate cache achieves a CPI of 2. What are the read and write bandwidths (measured by bytes per cycle) between RAM and the cache? (Consider both instruction cache and data cache. Assume each miss generates a request for one block.)

   (b) For a write-back, write-allocate cache, assuming 30% of replaced data cache blocks are dirty, what are the read and write bandwidths needed for a CPI of 2?

Solution:

   (a) When the CPI is 2, there are, on average, 0.5 instruction cache accesses per cycle. 0.3% of these instruction cache accesses cause a cache miss (and subsequent memory request). Assuming each miss requests one block, instruction accesses generate an average of $0.5 \times 0.003 \times 64 = 0.096$ bytes/cycle of read traffic.

   25% of instructions generate a data read request. 2% of these generate a cache miss; thus, read misses generate an average of $0.5 \times 0.25 \times 0.02 \times 64 = 0.16$ bytes/cycle of read traffic.

   10% of instructions generate a write request. 2% of these generate a cache miss. Because the cache is a write-through cache, only one word (4 bytes) must be written back to memory; but, every write is written through to memory (not just the cache misses). Thus, write misses generate an average of $0.5 \times 0.1 \times 4 = 0.2$ bytes/cycle of write traffic. Because the cache is a write-allocate cache, a write miss also makes a read request to RAM. Thus, write misses require an average of $0.5 \times 0.1 \times 0.02 \times 64 = 0.064$ bytes/cycle of read traffic.

   Hence: The total read bandwidth = 0.096 + 0.16 + 0.064 = 0.32 bytes/cycle, and the total write bandwidth is 0.2 bytes/cycle.

   (b) The instruction and data read bandwidth requirement is the same as in part (a). With a write-back cache, data are only written to memory on a cache miss. But, it is written on every cache miss (both read and write), because any line could have dirty data when evicted, even if the eviction is caused by a read request. Thus, the data write bandwidth requirement becomes $0.5 \times (0.25 + 0.1) \times 0.02 \times 0.3 \times 64 = 0.0672$ bytes/cycle.

2. Cache block size (B) can affect both miss rate and miss latency. Assuming a machine with a base CPI of 1, and an average of 1.35 references (both instruction and data) per instruction, find the block size that minimizes the total miss latency given the following miss rates for various block sizes.

8: 4%	16: 3%	32: 2%	64: 1.5%	128: 1%

   (a) What is the optimal block size for a miss latency of $20 \times B$ cycles?

   (b) What is the optimal block size for a miss latency of $24 + B$ cycles?

   (c) For constant miss latency, what is the optimal block size?

   Solution:

   (a)
   - AMAT for B = 8: $0.040 \times (20 \times 8) = 6.40$
   - AMAT for B = 16: $0.030 \times (20 \times 16) = 9.60$
   - AMAT for B = 32: $0.020 \times (20 \times 32) = 12.80$
   - AMAT for B = 64: $0.015 \times (20 \times 64) = 19.20$
   - AMAT for B = 128: $0.010 \times (20 \times 128) = 25.60$

   B = 8 is optimal.

   (b)
   - AMAT for B = 8: $0.040 \times (24 + 8) = 1.28$
   - AMAT for B = 16: $0.030 \times (24 + 16) = 1.20$
   - AMAT for B = 32: $0.020 \times (24 + 32) = 1.12$
   - AMAT for B = 64: $0.015 \times (24 + 64) = 1.32$
   - AMAT for B = 128: $0.010 \times (24 + 128) = 1.52$

   B = 32 is optimal.

   (c) B = 128 is optimal: Minimizing the miss rate minimizes the total miss latency.

3. Compare and contrast the ideas of virtual memory and virtual machines. How do the goals of each compare?

   Solution:

   Virtual memory aims to provide each application with the illusion of the entire address space of the machine. Virtual machines aim to provide each operating system with the illusion of having the entire machine at its disposal. Thus they both serve very similar goals, and offer benefits such as increased security. Virtual memory can allow for many applications running in the same memory space to not have to manage keeping their memory separate.

4. There are several parameters that affect the overall size of the page table. Listed below are key page table parameters.

Virtual Address Size	Page Size	Page Table Entry Size
32 bits	8 KiB	4 bytes

   (a) Given the parameters shown above, calculate the maximum possible page table size for a system running five processes.

(b) Given the parameters shown above, calculate the total page table size for a system running five applications that each utilize half of the virtual memory available, given a two-level page table approach with up to 256 entries at the 1st level. Assume each entry of the main page table is 6 bytes. Calculate the amount of memory required for this page table.

(c) A cache designer wants to increase the size of a 4 KiB virtually indexed, physically tagged cache. Given the page size shown above, is it possible to make a 16 KiB direct-mapped cache, assuming four 32-bit words per block? How would the designer increase the data size of the cache?

Solution:

(a) The tag size is $32 - log_2(8192) = 32 - 13 = 19$ bits. All five page tables would require $5 \times (2^{19} \times 4)$ bytes = 10 MB.

(b) In the two-level approach, the 2^{19} page table entries are divided into 256 segments that are allocated on demand. Each of the second-level tables contains $2^{(19-8)} = 2048$ entries, requiring $2048 \times 4 = 8$ KB each and covering 2048×8 KB = 16 MB (2^{24}) of the virtual address space.

"half the memory" means 2^{31} bytes, the amount of memory required for the second-level tables would be $5 \times (2^{31}/2^{24}) \times 8$ KB = 5 MB. The first-level tables would require an additional $5 \times 128 \times 6$ bytes = 3840 bytes.

(c) The page index is 13 bits (address bits 12 down to 0).

A 16 KB direct-mapped cache with two 64-bit words per block would have 16-byte blocks and thus 16 KB/16 bytes = 1024 blocks. Thus, it would have 10 index bits and 4 offset bits and the index would extend outside of the page index.

The designer could increase the cache's associativity. This would reduce the number of index bits so that the cache's index fits completely inside the page index.

5. Consider the following instruction mix:

R-type	I-type (non-lw)	Load	Store	Branch	Jump
24%	28%	25%	10%	11%	2%

(a) What fraction of all instructions use data memory?

(b) What fraction of all instructions use instruction memory?

(c) What fraction of all instructions use the sign extend?

(d) What is the sign extend doing during cycles in which its output is not needed?

Solution:

(a) 25 + 10 = 35%. Only Load and Store use Data memory.

(b) 100%. Every instruction must be fetched from instruction memory before it can be executed.

(c) 28 + 25 + 10 + 11 + 2 = 76%. Only R-type instructions do not use the Sign extender.

(d) The sign extend produces an output during every cycle. If its output is not needed, it is simply ignored.

6. The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-type	beqz/bnez	jal	lw	sw
40%	25%	5%	25%	5%

Consider an always-taken predictor whose accuracy is 45%. Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the EX stage. Assume there are no data hazards.

Solution:

The CPI increases from 1 to 1.4125. An incorrectly predicted branch will cause three instructions to be flushed: the instructions currently in the IF, ID, and EX stages. (At this point, the branch instruction reaches the MEM stage and updates the PC with the correct next instruction.) In other words, 55% of the branches will result in the flushing of three instructions, giving us a CPI of $1 + (1 - 0.45)(0.25)3 = 1.4125$. (Just to be clear: the always-taken predictor is correct 45% of the time, which means, of course, that it is incorrect $1 - 0.45 = 55\%$ of the time.)

7. Using the IEEE 754 single-precision floating-point format, write down the bit pattern that would represent -0.25. Can you represent -0.25 exactly?

Solution:

`1 01111101 00000000000000000000000` (Note: exponent is $125 - 127 = -2$.)

Yes, we can represent -0.25 exactly?

8. Assume that for a given program 70% of the executed instructions are arithmetic, 10% are load/store, and 20% are branch.

   (a) Given this instruction mix and the assumption that an arithmetic instruction requires two cycles, a load/store instruction takes six cycles, and a branch instruction takes three cycles, find the average CPI.

   (b) For a 25% improvement in performance, how many cycles, on average, may an arithmetic instruction take if load/store and branch instructions are not improved at all?

   (c) For a 50% improvement in performance, how many cycles, on average, may an arithmetic instruction take if load/store and branch instructions are not improved at all?

Solution:

   (a) Take the weighted average: 0.7*2 + 0.1*6 + 0.2*3 = 2.6.

   (b) For a 25% improvement, we must reduce the CPU to $2.6 \times 0.75 = 1.95$. Thus, we want $0.7 \times x + 0.1 \times 6 + 0.2 \times 3 <= 1.95$. Solving for x shows that the arithmetic instructions must have a CPI of at most 1.07.

   (c) For a 50% improvement, we must reduce the CPU to $2.6 \times 0.5 = 1.3$. Thus, we want $0.7x + 0.1 \times 6 + 0.2 \times 3 <= 1.3$. Solving for x shows that the arithmetic instructions must have a CPI of at most 0.14.