CSC 220 Homework #2 – Prof. Bill's Answers

Due Monday April 10, 2006
My answers are in small.

1. Our author notes that improvements in RAM or main memory speed have helped RISC architecture performance as compared to CISC. Why?

> "Complex" CISC instructions will reside primarily in hardware or micro-code ROM. The equivalent instructions in a "simpler" RISC architecture will require more steps and longer programs. These programs will be stored in RAM, so any speed improvement in RAM will benefit a RISC architecture over its CISC competition.

2. Page 132, problem 8… Instead of a 6-bit number, can you tell me how many "trits" are needed to hold an 8-bit number, or a byte.

> The largest number that can be represented in 8 bits (`1111 1111`) is 255. How do we get 255 in base 3? Well, 3^5 equals 243, that's a pretty good start. Add in 12 as 3^2 plus 3^1, and you're set. The answer: $(10\ 0110)_3$ … so you'll need 6 "trits."

3. Page 133, problem 12… also, how many address bits are needed to access this memory given that the smallest thing I can access is a byte.

> A memory with n address bits can hold 2^n words. To hold fewer words than this would be wasteful.
> That number looks like 256 Mbytes to me. 256 is 2^8 and a mega is 2^{20} bytes. It requires 28 bits to address 2^{28} bytes of information, one byte at a time.

4. If I have 16 bits (that's 2^4 bits for those of you keeping score at home) of data that I want to encode using the Hamming code algorithm for single bit error correction (ala the example on pages 76-77), then what is the overhead of this approach, or the percentage of my word that must be dedicated to check bits? What is the overhead if my data has 256 bits? And in the general case, n bits?

> Well, addressing the general case first, we need `(log n) + 1` check bits for an n-bit data word. Two specific cases:
>
> - For 16 bits, we need `(log 16) + 1` or 5 check bits. The overhead is `5/21 = 24%`
> - For 256 bits, we need `(log 256) + 1` or 9 check bits. The overhead is `9/265= 3.4%`

5. Page 133, problem 23… heck this is a math/physics problem, but it is fun. Speaking of fun, if you get an answer for this problem, then tell me something… using this level of compression, how much storage would you need to video the next 10 years (including the remaining exciting 8 weeks of 220!) of your life at this resolution?

OK, let's record our life, and I'll do some rounding because it doesn't matter:

- 1 second: 720 * 480 pixels/frame * 3 bytes/pixel * 30 frames/sec = 31.104 Mbytes
- 1 minute: 31.104 Mbytes/sec * 60 sec/minute = ~2 Gbytes
- 1 hour: 2 Gbytes * 60 minutes/hour = ~120 Gbytes
- 1 day: 120 Gbytes * 24 hours/day = ~3 Tbytes
- 1 year: 3 Tbytes * 365 days/year = ~1,000 Tbytes = 1 Pbyte (peta-byte!)
- 10 years: 1 Pbyte * 10 years = 10 Pbytes

So, recording 10 years of life, uncompressed is 10 Pbytes.
The compression rate? Well, 133 minutes would require 133 minutes * 2 Gbytes/minute = 266 Gbytes. If that is compressed onto a 3.5 Gbyte DVD, then your compression rate is 266 / 3.5 = 76x. Phew.

6. Page 134, problem 26… this used to be a nice engineering (and marketing!) workaround when memory was expensive, and hint: the answer's right in the chapter, eh.

Using a color palette, you can access 256 3-byte color choices in your image using a single byte at each pixel location.

7. Page 134, problems 32 & 33… a couple of digital camera questions that are just poser math/physics questions

OK, how many bytes are we talking here?
3000 * 2000 pixels * 3 bytes/pixel = 18 Mbytes
We get a 5x compression using Jpeg, that's 18/5 = 3.6 Mbytes.
Doing that in 2 seconds requires a transfer of 1.8 Mbytes/second

(16 million pixels * 3 bytes/pixel) / 5 = 9.6 Mbytes/picture using Jpeg
Can we round that to 10 MB/picture? If so, then 1 GB is 1,000 MB. So, we can store 1,000 / 10 = 100 pictures.

8. Finally, some trivia… the section is in brackets. Keep it short and sweet:

a) If I need to speed up my compute environment by 100x, would you suggest I try instruction-level parallelism (pipelining) or processor-level processing? [2.1]

A 100x speed-up using pipelining implies 100 steps in your pipe. This is not reasonable, so processor-level processing is your best bet.

b) If an address has n bits, then how many words can be addressed? [2.2]

2^n words

c) What is the relationship between the locality principle and caching? [2.2]

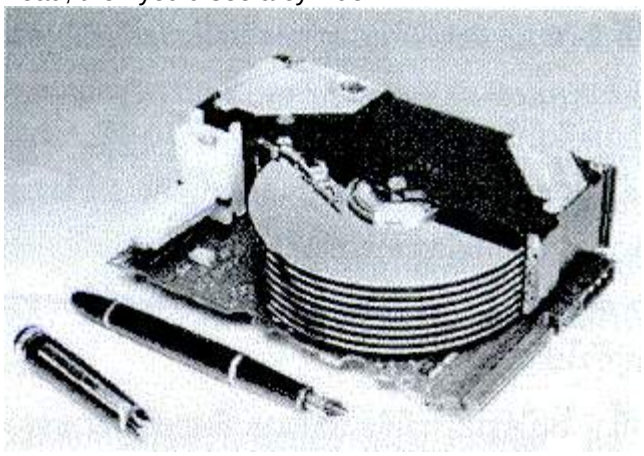
Caching brings in neighboring section of data into the faster memory for faster processing. The locality principle says that these neighbors are more likely to be accessed in the short-term than other random memory locations. Without the locality principle the advantage of and reason for caching would disappear.

d) What is the difference between a unified cache and a split cache? [2.2]

A unified cache holds instructions and data. In a split caches, these elements are stored in separate cache memories.

e) Draw the cylinder, sector and track a disk/hard drive. [2.3]

Here's a cool, old-school disk drive picture that I found… see the platters. Anyway, tracks are the circles around each platter. A sector is a section of the track. And if you pile all the tracks from these platters on top of each other in your head, then you'd see a cylinder.



f) When you're not listening to your MP3 player, does your CD spin at a constant velocity? Explain why. [2.3]

Velocity increases as your CD is read from the inside out. This maintains a steady streaming rate of 120 cm/sec.

g) What kind of mouse are you using right now: mechanical, optical, or opto-mechanical? Explain. [2.4]

I don't know about you, but mine is mechanical. It has a little ball in the bottom that tracks all my mickeys. Ha!

h) I told a buddy that my mouse's mickey was 2-3 feet. He didn't believe me. Do you? Explain. [2.4]

No way. A mickey is the minimum distance a mouse travels before he reports movement to the CPU. So, 2-3 feet would be silly and very irritating.

i) What is halftoning? [2.4]

Halftoning is a method of producing gray-appearing tones in a black and white printer.

j) Draw the three modem communication modes: amplitude modulation, frequency modulation, and phase modulation. [2.4]

Each describes how 0's and 1's are communicated across the waves of electrical signals carried by your modem. There's a nicer picture than I can draw on page 118.

k) Pictures stored in JPEG format are compressed. That's cool, but what is the downside of this? [2.4]

JPEG compression is called "lossy compression", in that some detail in the image is lost during the compression.

l) Does your internet browser support ASCII, Unicode or both? [2.4]

Well, Firefox and IE both support Unicode. Unicode is a superset of ASCII, therefore ASCII is supported as well, by default.