Homework 3

Memory and Cache

1. Reorder the fields in this structure so that the structure will (a) consume the most space and (b) consume the least space on an IA32 machine on Linux.

```
struct foo {
  char a;
  short b;
  int c;
  char d;
  double e;
}
```

2. The IA32 architecture includes the scaled index addressing mode, which is useful for accessing arrays, especially in loops The instruction

```
movl (%ecx, %ebx, 4), %eax
```

   loads the word starting at address %ecx+4*%ebx, Mem[Reg[%ecx]+4*Reg[%ebx]] in the notation of your textbook. Notice how the parentheses denote one level of indirection. Some architectures include an indexed addressing mode that is doubly indirect, meaning that you compute the address, then you fetch a word, then use the word as an address to fetch the final value. An example instruction, in IA32 GAS style, might look something like

```
movl ((%ecx, %ebx, 4)), %eax
```

   which would mean that we would fetch the word Mem[Mem[Reg[%ecx]+4*Reg[%ebx]]]. (a) Give an example (in C) of a loop that could make use of such an addressing mode.

   Suppose that it was possible do double index as well, so that we could write

```
movl ((%ecx, %ebx, 4), %edx, 4), %eax
```

   which would mean that we would fetch the word at Mem[Mem[Reg[%ecx]+4*Reg[%ebx]] + 4*Reg[%edx]]. (b) Give an example (in C) of a loop nest that could make use of such an addressing mode. If you think these kinds of indexing modes are kind of cool, look at info on the VAX addressing modes to see just how far this can go.

3. Consider a processor which uses 16 bit addresses and can address 2^16=64K bytes of memory. Suppose that it has one level of cache. As in Figure 6.25 of your textbook,

the address is split into a t bit tag, an s bit set index, and a b bit block offset. The cache consists of 1024 bytes, with a block size of 32 bytes. Answer each of the following for direct-mapped, 4-way set associative, and fully associative versions of the cache.

   a. How many cache lines are there?
   b. What is b?
   c. What is s?
   d. What is t?

4. For the cache in problem 3, draw the cache given it is structured as follows. You can elide replicated components, but annotate your drawing with how many components there are.
   a. Direct-mapped
   b. 4-way set associative
   c. Fully associative

5. Our company wants to optimize the performance of the following code

```
void vector_add(int n, int *a, int *b, int *c) {
   int i;
   for (i=0;i<n;i++) {
     c[i]=a[i]+b[i];
   }
}
```

   run on the same processor and cache as described in problem 3. The cache is write-back, write-allocate, and has an LRU replacement policy. Integers are 32 bits.
   a. Suppose the cache is direct mapped. Let n=2048, a=0x4000, b=0x8000, c=0xc000. On average, how many times per loop iteration will you load a cache block from main memory? How many times per loop iteration will you flush a cache block back to main memory?
   b. What is the minimum degree of associativity (i.e., the n in n-way) that the cache needs to reduce the answers in (a) to 0.25 cache blocks read per iteration and 0.125 cache blocks written per iteration?
   c. While we're all fired up to buy ultra-cool mega-associative cache hardware (which comes only in black), a smart alec programmer claims that we can get the same effect by having a=0x4000, b=0x8020, and c=0xc040. Is he right? Why or why not?