Name:_____

Problem 1: (17 points)

Question A: (5 points)

A cache may be organized such that:

- In one case, there are more data elements per block and fewer blocks
- In another case, there are fewer elements per block but more blocks

However, in both cases – i.e. larger blocks but fewer of them OR shorter blocks, but more of them – the cache's total capacity (amount of data storage) remains the same.

What are the pros and cons of each organization? Support your answer with a short example assuming that the cache is direct mapped. Your answer must fit in the box below.

Solution:

If block size is larger:

- Con: There will be fewer blocks and hence a higher potential for conflict misses
- Pro: You may achieve better performance from spatial locality due to the larger block size
- Example: If you have a high degree of sequential data accesses, this makes more sense

If there are fewer elements per block and more blocks:

- Con: You may be more subject to compulsory misses due to the smaller block size
- Pro: You may see fewer conflict misses due to more unique mappings
- Example: If you have more random memory accesses, this makes more sense

Question B: (5 points)

Assume:

- A processor has a direct mapped cache
- Data words are *8 bits* long (i.e. 1 byte)
- Data addresses are *to the word*
- A physical address is *20 bits* long
- The tag is *11 bits*
- Each block holds 16 *bytes* of data

How many blocks are in this cache?

Solution:

Given that the physical address is 20 bits long, and the tag is 11 bits, there are 9 bits left over for the index and offset

We can determine the number of bits of offset as the problem states that:

- Data is word addressable and words are 8 bits long
- Each block holds 16 bytes

As there are 8 bits / byte, each block holds 16 words, thus **4** bits of **offset** are needed.

This means that there are 5 bits left for the index. **Thus, there are 2^5 or 32 blocks in the cache.**

Name:_____

Question C: (7 points)

Consider a *16-way set-associative* cache

- Data words are *64 bits* long
- Words are addressed to the *half-word*
- The cache holds 2 Mbytes of data
- Each block holds 16 data words
- Physical addresses are 64 bits long

How many bits of tag, index, and offset are needed to support references to this cache?

Solution:

We can calculate the number of bits for the **offset** first:

- There are 16 data words per block which implies that at least 4 bits are needed
- Because data is addressable to the ½ word, an additional bit of offset is needed
- Thus, the **offset** is **5 bits**

To calculate the **index**, we need to use the information given regarding the total capacity of the cache:

- 2 MB is equal to 2^{21} total bytes.
- We can use this information to determine the total number of *blocks* in the cache…
  - 2^{21} bytes **x** (1 block / 16 words) **x** (1 word / 64 bits) **x** (8 bits / 1 byte) = 2^{14} blocks
- Now, there are 16 (or 2^4) blocks / set
  - Therefore there are 2^{14} blocks **x** (1 set / 2^4 blocks) = 2^{10} or 1024 sets
- Thus, **10 bits of index** are needed

Finally, the remaining bits form the tag:

- 64 – 5 – 10 = 49
- Thus, there are **49 bits of tag**

To summarize: **Tag**: 49 bits; **Index**: 10 bits; **Offset**: 5 bits