Name:_____

Problem 1: (15 points)

Question A: (5 points)
Briefly (in 4-5 sentences or a bulleted list) explain why many of the transistors on a modern microprocessor chip are devoted to Level 1, Level 2, and sometimes Level 3 cache. Your answer must fit in the box below!

Students' answers should say something to the effect of:

- Processing logic is faster than off-chip, 1-transistor DRAM – and this performance gap has been continually growing
- Idea: bring in subsets of (off-chip) main memory into faster on-chip memory (SRAM) that can operate at the speed of the processor
- Caches help to ensure faster "data supply times" to ensure that logic is not idle for larger number of CCs (e.g. the time to access off-chip memory)
- If instruction encodings and data for load instructions could not be accessed in 1-2 CCs, CPU performance would be significantly (and negatively) impacted.
- *A discussion of spatial vs. temporal locality should receive little to no credit – speed differentials are the most important consideration in this answer.*

In HW 8, you saw that some versions of the Pentium 4 microprocessor have two 8 Kbyte, Level 1 caches – one for data and one for instructions. However, a design team is considering another option – a single, 16 Kbyte cache that holds both instructions and data.

Additional specs for the 16 Kbyte cache include:

- Each block will hold 32 bytes of data (not including tag, valid bit, etc.)
- The cache would be 2-way set associative
- Physical addresses are 32 bits
- Data is addressed to the word and words are 32 bits

Question B: (3 points)
How many blocks would be in this cache?

Answer

- The cache holds 2^{14} bytes of data
- Each block holds 2^5 bytes
- Thus, there are $2^{14} / 2^5 = 2^9 = 512$ blocks

Name:_____

Question C: (3 points)
How many bits of tag are stored with each block entry?

Answer
We need to figure out how many bits are dedicated to the offset, index and tag. (Basically, this question asks how many bits of tag are needed.)

- Index:
  - # of sets: 1024 / 2 = 256 = 2^8
  - Therefore 8 bits of index are needed
- Offset:
  - # of words per block = 32 / 4 = 8
  - $2^3 = 8$
  - Therefore 3 bits of offset
- Tag
  - 32 – 3 – 8 = 21 bits of tag

Therefore, 21 bits of tag need to be stored in each block.

Question D: (4 points)
Each instruction fetch means a reference to the instruction cache and 35% of all instructions reference data memory. With the first implementation:

- The average miss rate in the L1 instruction cache was 2%
- The average miss rate in the L1 data cache was 10%
- In both cases, the miss penalty is 9 CCs

For the new design, the average miss rate is 3% for the cache as a whole, and the miss penalty is again 9 CCs.

Which design is better and by how much?

Answer

$\text{Miss penalty}_{v1} = (1)(.02)(9) + (0.35)(.1)(9) = .18 + .063 = 0.495$

$\text{Miss penalty}_{v2} = (.03)(9) = 0.270$

V2 is the right design choice

Name:_____

Problem 2: (10 points)

Question A: (4 points)
Explain the advantages and disadvantages (in 4-5 sentences or a bulleted list) of using a direct mapped cache instead of an 8-way set associative cache. Your answer must fit in the box below!

Answer

- A direct mapped cache should have a faster hit time; there is only one block that data for a physical address can be mapped to
- The above "pro" can also be a "con"; if there are successive reads to 2 separate addresses that map to the same cache block, then there may never be a cache hit. This will significantly degrade performance.
- In contrast, with a set associative cache, a block can map to one of 8 blocks within a set. Thus, if the situation described above were to occur, both references would be hits and there would be no conflict misses.
- However, a set associative cache will take a bit longer to search – could decrease clock rate.

Question B: (2 points)
Assume you have a 2-way set associative cache.

- Words are 4 bytes
- ***Addresses are to the byte***
- Each block holds 512 bytes
- There are 1024 blocks in the cache

If you reference a 32-bit physical address – and the cache is initially empty – how many data words are brought into the cache with this reference?

Answer

- The entire block will be filled
- If words are 4 bytes long and each block holds 512 bytes, there are 2^9 / 2^2 words in the block
- i.e. there are 2^7 or 128 words in each block

Question C: (4 points)
Which set does the data that is brought in go to if the physical address F A B 1 2 3 8 9 (in hex) is supplied to the cache?

Answer
We need to determine what the index bits are. From above, we know the offset is 9 bits (remember, data is byte addressable) – so we will need to break up the hex address into binary:

1111 1010 1011 0001 0010 0011 1000 1001

Our offset for this address is: 1 1000 1001

1024 / 2 = 2^{10} / 2^1 = 512 = 2^9 – therefore 9 bits of index are required.

These are: 01 ■ 0010 ■ 001 which implies the address maps to the 145th set.

Name:_____

Problem 6: (10 points)

Consider an Intel P4 microprocessor with a 16 Kbyte unified L1 cache. The miss rate for this cache is 3% and the hit time is 2 CCs. The processor also has an 8 Mbyte, on-chip L2 cache. 95% of the time, data requests to the L2 cache are found. If data is not found in the L2 cache, a request is made to a 4 Gbyte main memory. The time to service a memory request is 100,000 CCs. On average, it takes 3.5 CCs to process a memory request. How often is data found in main memory?

Average memory access time = Hit Time + (Miss Rate x Miss Penalty)

$\text{Average memory access time} = \text{Hit Time}_{L1} + (\text{Miss Rate}_{L1} \times \text{Miss Penalty}_{L1})$

$\text{Miss Penalty}_{L1} = \text{Hit Time}_{L2} + (\text{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2})$

$\text{Miss Penalty}_{L2} = \text{Hit Time}_{Main} + (\text{Miss Rate}_{Main} \times \text{Miss Penalty}_{Main})$

$$3.5 = 2 + 0.03\,(15 + 0.05\,(200 + X\,(100{,}000)))$$
$$3.5 = 2 + 0.03\,(15 + 10 + 5000X)$$
$$3.5 = 2 + 0.03\,(25 + 5000X)$$
$$3.5 = 2 + 0.75 + 150X$$
$$3.5 = 2.75 + 150X$$
$$0.75 = 150X$$
$$X = .005$$

Thus, 99.5% of the time, we find the data we are looking for in main memory.