**Fall 2013: COMP 7300 Advanced Computer Architecture**

**Test 2 (100 pts)**

Grading policy:

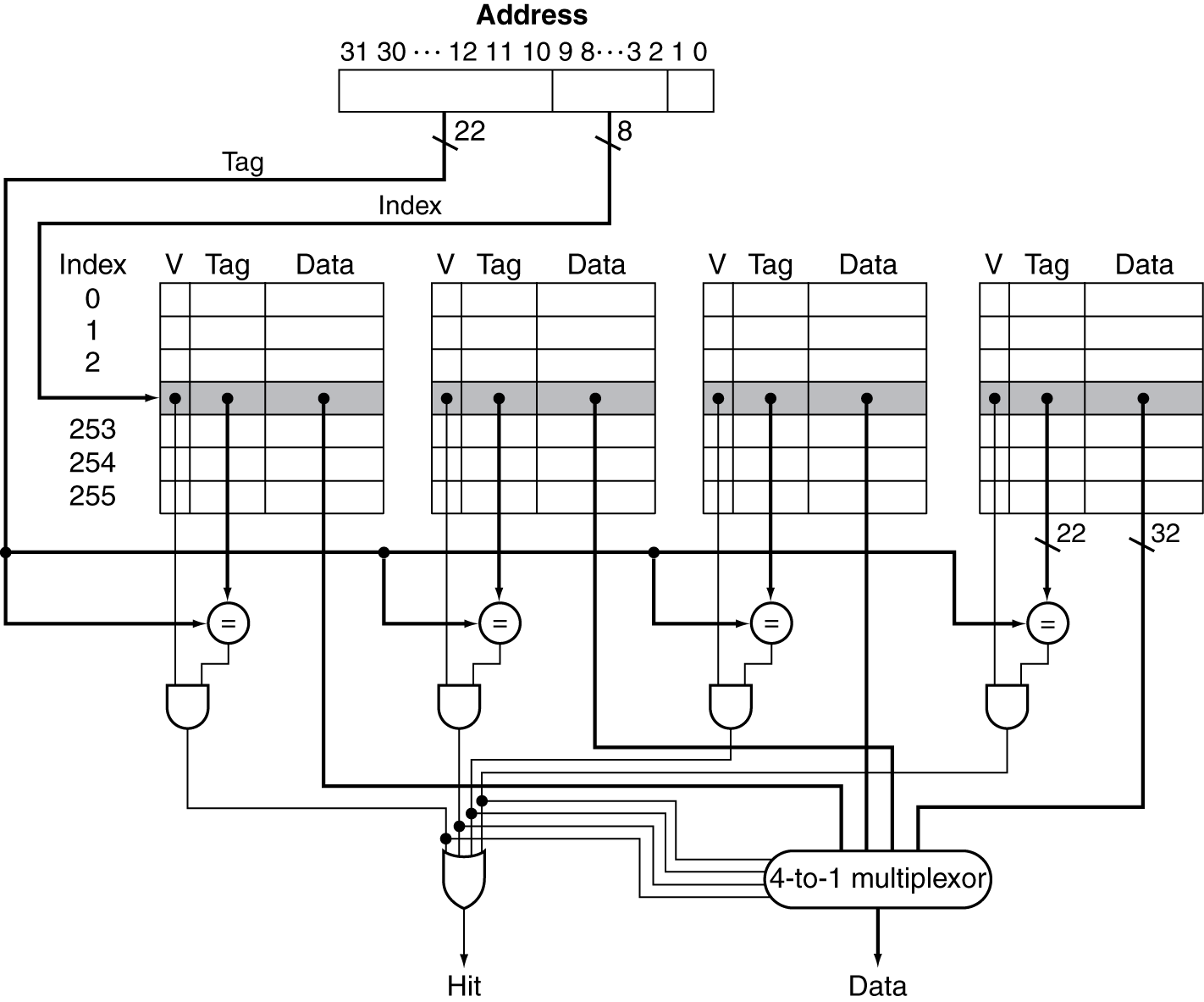
¼ Credit for correct answer

¾ Credit for wellwritten and solid justification/facts/arguments. **Show your work.**

**For Test 2, we assume a CPU with a 32-bit address bus.**

**>>>>>> Last Question (Last Page) is easy <<<<<<<<<<<**

1. **Cache (Questions) Justify your answers**
2. We consider a direct mapped 16 KB cache with 512 byte blocks. The CPU generates address 25,000.
   1. **(4 points)** What is the block address of Block ***B*** containing address 25,000?
   2. **(4 points)** What is Block B’s tag?
   3. **(4 points)** Where will Block  ***B*** be stored in the cache?
3. We consider a 4-way associative 16 KB cache with 512 byte blocks. The CPU generates address 25,000.
   1. **(1 point)** What is the block address of Block ***B*** containing address 25,000?
   2. **(4 points)** Where will Block  ***B*** be stored in the cache? 1
4. **(9 points)** Compare a direct mapped cache versus a fully associative cache in terms of hit time, energy cost, and miss rate. For each metric (hit time, energy, and miss rate), briefly justify why one outperforms (if applicable) the other.
5. Consider the cache on the figure below. Answer the following questions based on the figure below



* 1. **(2 points)** Is this a direct mapped or an associative cache? Explain
  2. **(6 points)** If associative, is it 2 way, 4 way.... or x-way associative? Explain
  3. **(4 points)** If associative, how many sets does this cache have? Explain

1. **Exercise 1:** We assume that the data cache size is 16 KB (data) and the block size is 4 KB. We assume that M[i][j] is adjacent to M[i][j+1] in the memory and that the cache is fully associative using LRU replacing. Consider the following code:

int A[32][256];

int i,j;

for (j = 0; j < 256; j++){

for (i = 0; i < 32; i++)

A[i][j] = 2 \* A[i]j];

}

1. **(12 points)** We assume that the code is in a separate instruction cache and the variables i and j are in registers. How many compulsory, capacity, and conflict misses occur for the above code?
2. **(16 points)** Would a loop interchange decrease the number of misses? If yes, How many compulsory, capacity, and conflict misses occur for the above code with loop interchange?
3. The block size is 128 bytes. The cache size is 32 KB. Assume a fully associative cache with LRU replacement. We consider the following algorithm to achieve an in-place transposition of a matrix:

int i, j; // in registers

char A[256][256];

for (j = 0; j < 256; j++){

for (i = j+1; i < 256; i++) {

// swap A[i][j] and A[j][i]

A[i][j] = A[i][j] ^ A[j][i];

A[j][i] = A[i][j] ^ A[j][i];

A[i][j] = A[i][j] ^ A[j][i];

}

}

1. **(8 points)** We assume that the code is in a separate instruction cache and the variables i and j are in registers. How many compulsory misses occur for the above code?
2. **(4 points)** Would a loop interchange decrease the number of misses? If yes, How many compulsory misses occur for the above code with loop interchange?
3. **(10 points)** What should be the minimal size of the cache to take advantage of a blocked execution?

1. **Virtual Memory**

**(12 points)** We consider a 32 bit address bus and a 512 KB physical memory. Page size is 8 KB. Assuming a validity bit and a dirty bit, what is the size of the page table? **Show the different steps to determine the page table size.**