Name _____ ID# _____

Sample Final

Chapter 7 Only

Name _____ ID# _____

PART A

1. For each of the MIPS code sequences below, state if the memory accesses show: temporal locality, spatial locality, both or neither.

(a)
```
        li $t0, 100
LOOP:   lw $t1, 19196($s0)
        lw $t2, 0($s0)
        addi $t0, $t0, -1
        bne $t0, zero, LOOP
```

(b)
```
        li $t0, 100
LOOP1:
        li $t1, 4
LOOP2:
        add $t2, $s0, $t1
        lb $s1, 0($t2)
        addi $t1, $t1, -1
        bne $t1, zero, LOOP2
        addi $t0, $t0, -1
        bne $t0, zero, LOOP1
```

(c)
```
        li $t0, 100
LOOP:
        lw $s1, 0($t0)
        addi $t0, $t0, -1
        bne $t0, zero, LOOP
```

2. How many bits are needed to implement a direct mapped cache with i index bits, t tag bits and block size of B bits?

3. If a cache requires N bits to implement and can store n bits of data, the space overhead is defined as $N - n$. Assuming a 32-bit address, consider a direct-mapped cache capable of holding B blocks of data, where each block contains b bytes (both B and b are powers of two). What is the space overhead for this cache?

Name _____ ID# _____

4. Suppose the cache in the previous problem is modified as follows: it is now a *2k*-way associative cache, capable of holding the same number of bits of data as the cache in the previous problem. What is the space overhead now?

5. For what value of *k* would the space overhead be maximized? When would you want to choose such a cache?

PART B

1. Multiple-byte blocks

A new processor has a direct-mapped cache capable of holding *B* bytes of data in total, and each block of the cache contains *b* bytes of data (both *B* and *b* are powers of two). How many bits does it take to implement the cache?

2. The Mystery of the big B and the little b

We want to figure out what the values of *B* and *b* are for the new processor using the following code excerpt

```
// We will choose values for LENGTH and STEP
char a[LENGTH];
int i, j, temp;

for(i = 0; i < 10000; i++)
        for(j = 0; j < LENGTH; j = j + STEP)
                temp = temp + a[j];
```

Name _____ ID# _____

Part 1: Suppose STEP is initially 1.

(a) If *LENGTH* ≤ *B*, how many cache misses would we expect each time the outer *for* loop is executed?

(b) How would things change if we increased *LENGTH* until it became larger than *2B*?

(c) What would happen if we kept increasing *LENGTH*?

Part 2: Suppose we fix *LENGTH* to some value larger than *2B*, and increase *STEP*:

(a) If *STEP* is less than *b*, how many cache misses would we expect each time the outer *for* loop is executed?

(b) After *STEP* equals *b*, what would happen if we kept increasing *STEP*?

Part 3: Experimentation time

We ran some experiments, varied the values of *LENGTH* and *STEP*, and computed the average memory-access time (*AMAT*) in the statement

temp = temp + a[j];

The table below lists the *AMAT* (in nanoseconds) for different values of *LENGTH* and *STEP*. Note that the time measurements are not perfectly accurate, so there may be some fluctuations in the data:

Name _____ ID# _____

	STEP							
LENGTH	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**
8		53					50	
16		47		51				
32	54	49						50
64		168	275			502	52	
128	108			506			497	
256	101		272		507	498		

Drawing inferences from the data:

(a) What is the value of *B*? Explain your answer.

(b) What is the value of *b*? Explain your answer.

(c) What is the approximate time for a cache hit?

(d) What is the approximate time for a cache miss?

(e) Fill in the missing table entries with reasonable values.