Name _____

Do **not** write your id number or any other confidential information on this page.

There are 11 questions worth a total of 100 points. Please budget your time so you get to all of the questions.

You will want to use a copy of the "green card" from the textbook. We have additional copies if you do not have one. Other than that, the exam is closed book, closed notes, etc.

Keep your answers brief and to the point. Your graders thank you for your assistance.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 8

2. _____ / 10

3. _____ / 6

4. _____ / 6

5. _____ / 10

6. _____ / 6

7. _____ / 10

8. _____ / 8

9. _____ / 12

10. _____ / 12

11. _____ / 12

Question 1. (8 points) Consider the 32-bit word 0x00000000.

(a) If we interpret this as a MIPS machine instruction, what is it if we write it in MIPS assembly language?

(b) What happens if this instruction is executed in a MIPS program?

Question 2. (10 points) (a) Convert the hexadecimal integer value 0x010A to a decimal (base-10) integer.

(b) Convert the decimal number -127 to a 16-bit 2's complement hexadecimal integer.

Question 3. (6 points) Suppose we have a 2-dimensional *n* by *n* matrix A stored in main memory in a C/C++ program. That means that the individual elements of A are stored in *row-major* order: first row 0, then row 1, and so forth; or, element-by-element, first `A[0][0]`, then `A[0][1]`, `A[0][2]`, …, `A[0][n-1]`, `A[1][0]`, `A[1][1]`, …, `A[1][n-1]`, `A[2][0]`, …, `A[n-1][n-1]`. We want to write a nested loop to initialize all of the elements of the array to 0. Here are two proposed ways to do it:

```
I.   for (row = 0; row < n; row++)
       for (col = 0; col < n; col++)
         A[row][col] = 0;

II.  for (col = 0; col < n; col++)
       for (row = 0; row < n; row++)
         A[row][col] = 0;
```

Both of these correctly initialize the array. The code is to be run on a conventional x86 machine with the usual memory hierarchy. The machine has enough storage to hold all of array A in main memory at the same time.

Which of the two nested loops is preferable for initializing the array, or does it matter which one we use? Give a brief justification for your answer.

Question 4. (6 points) Suppose we have a processor that uses a **2-bit** dynamic branch prediction scheme to try to guess whether conditional branch instructions will be taken or not (i.e., whether they will jump or will fall through and execute the next sequential instruction without jumping).

Now suppose that we are executing a loop that contains a conditional jump instruction at the bottom. The loop has executed 17 times and the jump has been taken on each of these 17 iterations. On the 18th iteration, the jump is **not** taken and the loop terminates.

The question is: the next time that this particular jump instruction is encountered during execution, will the 2-bit dynamic branch prediction hardware predict that the jump will be taken or will it predict that it will not? Give a brief but precise justification of your answer. You may find it helpful (but it is not required) to explain the states of a 2-bit branch prediction scheme and how they apply to this situation.

Question 5. (10 points) Barney Bitblitter has been asked to implement a high-tech self-service photo copying kiosk at the newly remodeled Northgate mall. The copy stand contains three workstations where customers can set up their jobs and enter their credit-card information. The copy stand also contains a single scanner where the users place the photos they want copied and a single printer where the copies are produced. Only one customer can use both the scanner and printer at the same time, and both are needed at the same time to copy photos.

The control program in each workstation verifies all of the customer's information, then obtains exclusive (locked) access to both the scanner and the printer. Once all of this has been done, the photos can be scanned and copies printed.

Unfortunately, Barney isn't all that good at concurrent programming, and sometimes the program locks up and cannot proceed without shutting down and starting over again. All Barney has been able to figure out so far is that the deadlock is somehow related to multiple workstations trying to obtain exclusive access to both the scanner and the printer at the same time.

(a) Describe how a deadlock could occur in this situation.

(b) Describe a way in which the software could be modified to prevent or avoid this deadlock situation.

Question 6. (6 points) Another one of your colleagues is trying to understand how locks are implemented. She proposes that a lock could be implemented as follows, using an ordinary integer variable.

```
int lock;   /* 1 if lock is set, 0 if it is free */

/* wait for the lock to be free, then set it */
void getlock() {
   while (lock == 1) { }    /* do nothing – spin lock */
   lock = 1;                /* set lock */
}

/* release the lock */
void unlock() {
   lock = 0;                /* release lock */
}
```

While the spin lock (busy-wait) in `getlock` ties up the processor, this is ok because this implementation is only designed to run on a multiprocessor machine.

Is this implementation sufficient to ensure that the lock behaves correctly when multiple concurrent processes are trying to use it to ensure exclusive access to some resource? If so, give a brief explanation of why it works; if not, give an example of how it can fail.

Question 7. (10 points) (a) In a virtual memory system, what is the *working set* of a process?

(b) What causes *thrashing* in a virtual memory system? Explain your answer in terms of your definition of working set from part (a).

Question 8. (8 points) Suppose we have a memory system that includes a 2-level cache. The caches and main memory have the following access times:

L1 cache	10 nsec
L2 cache	20 nsec
Main memory	120 nsec

Now suppose that 55% of the total memory accesses are satisfied by the L1 cache, 35% are satisfied by the L2 cache, and the remaining 10% require access to main memory. What is the effective access time of this memory system?

Question 9. (12 points) (a) Suppose we have a virtual memory system with 32-bit virtual addresses and 8192 byte (8K) pages. Of the 32-bits in the virtual address, how many are used to specify the page number and how many specify the offset in the page?

(b) Complete the following MIPS assembly language function `getreal` to translate the virtual address in register `$a0` to a real (physical) address. The resulting real address should be returned in `$v0`. Register `$a1` contains the address of a single-level page table. The page table is an array of 32-bit words containing the addresses of the page frames in main memory (i.e., the first word in the array is the real memory address of virtual page 0, the next word is the address of page 1, etc.). You should assume that the page table array is as long as needed and that all entries contain a valid address (i.e., you do not need to simulate page faults or worry about whether the page table entries contain valid data). Each page contains 8192 bytes.

```
# Given a virtual address in $a0 and a page table address
# in $a1, store the corresponding real address in $v0

getreal:

        jr   $ra            # return with result in $v0
```

Question 10. (12 points) A few questions about scheduling.

(a) Which of the following scheduling policies minimize the overall average wait time of a collection of jobs: first come first served, round robin, or shortest job first? Give a brief justification for your answer.

(b) What is the difference between preemptive and non-preemptive scheduling?

Question 11. (12 points) And a couple of questions about caches.

(a) Suppose that we have a 4KB direct-mapped cache with 32-byte cache blocks. If addresses are 32 bits each, how many bits are used for the cache index, offset, and tag fields?

(b) What is the difference between write-back and write-through caches? Give one advantage of each.