Name: _____

CIS 351 Practice Final

1. Complete Homework 5 and be comfortable answering similar questions.
2. What are the four elements that hold the Single Cycle CPU state?
3. Why are the inputs (B, C) to the register file 5-bits while the outputs (I, G) are 32-bits? (Figure 1)
4. Is line F active for all instructions? If so, why does it have data on it for R-type or other instructions where it isn't used? If not, why is it not used and how is it kept empty?
5. Explain the function of the MemWrite flag going into the Data memory component.

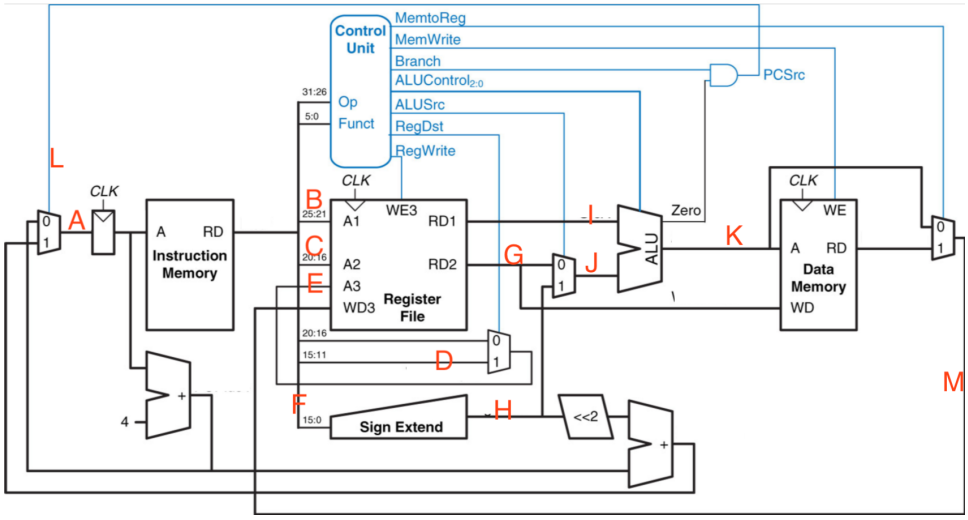

Figure 1: Single Cycle CPU with labeled points

Name: _____

Problems 6-8 are similar to a problem on the homework. However, in this case, you are not given which instruction to use.

6. Write a single assembly language statement that will set bits 1, 2, and 6 of register `$t0` to 1 and leave all other bits unchanged.

7. Write two or three assembly language statements that will set bits 4, 5, 7, and 8 of register `$t0` to 0 and leave all other bits unchanged.

8. Write a single assembly language statement that will flip bits 0 and 7 of register `$t0` and leave all other bits unchanged.

9. Write one or two assembly language statements that will move bits 2 through 5 of register `$t1` into bits 0 through 3 of register `$t0`. Register `$t1` should remain unchanged. Bits 4 - 31 of register `$t0` should be 0.

Name: _____

Any coding questions on the exam will require strict adherence to assembly conventions, including preserved regsisters. They are intentionally designed to give few or no points to an almost-correct solution that does not correctly preserve registers or makes incorrect assumptions about which registers will be preserved by called functions.

10. Convert the following Java code to assembly. Use standard procedure-calling conventions — including preserving registers where appropriate.

```
int p1(int[] a0) {
  int x = 0;
  while (check(array[x], array[x + 2]) != 0) {
    array[x] = array[x + 2];
    x++;
  }
  return x;
}
```

11. Implement the procedure `countX(int val, int[] array, int size)` recursively. You may find the C implementation below helpful. Note: I will not ask you to write a recursive function on the exam, but I may ask you to read and understand recursive code. Also, recursive functions are good practice for calling functions correctly.

```
int countX(int val, int array[], int size)
{
  int count = 0;
  if (size == 0) { return 0;}

  if (array[0] == val) { count = 1; }

  /* The expression "array + 1" is legal in C because,
     just as it is in assembly, an array variable is simply
     the memory address of the first element in the array.
     In C, however, the "+1" means to add the *size* of one
     element, which, in this case, is 4 bytes.
   */
  return count + countX(val, array + 1, size -1);
}
```

12. Convert the following Java code to assembly. Use standard calling and register usage conventions. You may assume that `method1`, `method2`, and `method3` all exist. (In other words, call them but don't write them.)

```
int composition(int a, int b) {
   return method1(a, b, method2(method3(a) + method3(b)));
}
```