Engineering 100 *Midterm Exam*

Fall 2007

Name: _____ unique name: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Page #	Points
2	/20
3	/15
4	/20
5-7	/45
Total	**/100**

NOTES:

- Closed book, notes, and calculator. Basically just you, the test and a writing utensil.
- There are **9** pages including this one – ***The last two are for you to rip out.***
  - Don't put anything on those pages you want graded!
- Don't spend too much time on any one problem.
- You have about 80 minutes for the exam.

1) Fill in the blank **[8, 2 points each]**

   a) The number 34 is represented in an 8-bit binary number as _____.

   b) The 4-bit 2's complement value 1001 is _____ as a decimal number.

   c) The bitwise XOR of the binary numbers 100010 and 100101 is: _____.

   d) In Verilog you use an "@*" block when representing _____ logic.

2) Write an e100 assembly *function* called "max". It is to take two arguments and return the larger of the two values. The arguments should be named max_a1 and max_a2. The return value should be named max_rv, and the return address should be named max_ra. You should declare space for all needed variables here. **[12]**

3) Consider the following code:

```
start     add tom 4 mary
          sub mary tom 1
          halt
tom       .data 1
mary      .data 5
```

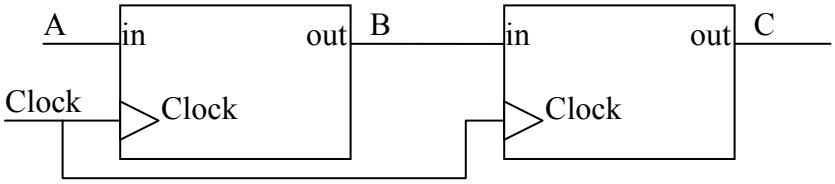
a) In the column labeled "Initial value" write the value in each memory location after the code is assembled, but before it is run. If a given memory location's value isn't known, put a zero. Write all numbers in decimal. **[7]**

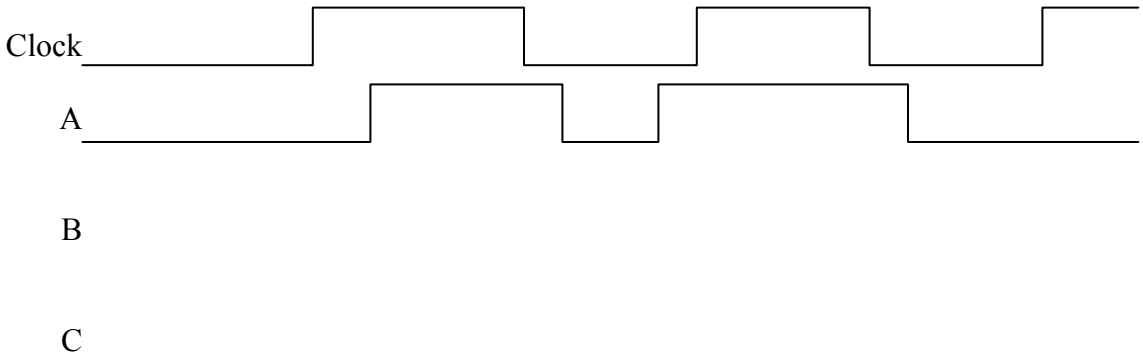
b) In the column labeled "Final value" write the value in each memory location after the code is run. If a given memory location's value isn't known, put a zero. *If the value hasn't changed since from the initial value, you are to leave it blank*. Write all numbers in decimal. **[8]**

Memory location	Initial Value	Final Value
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

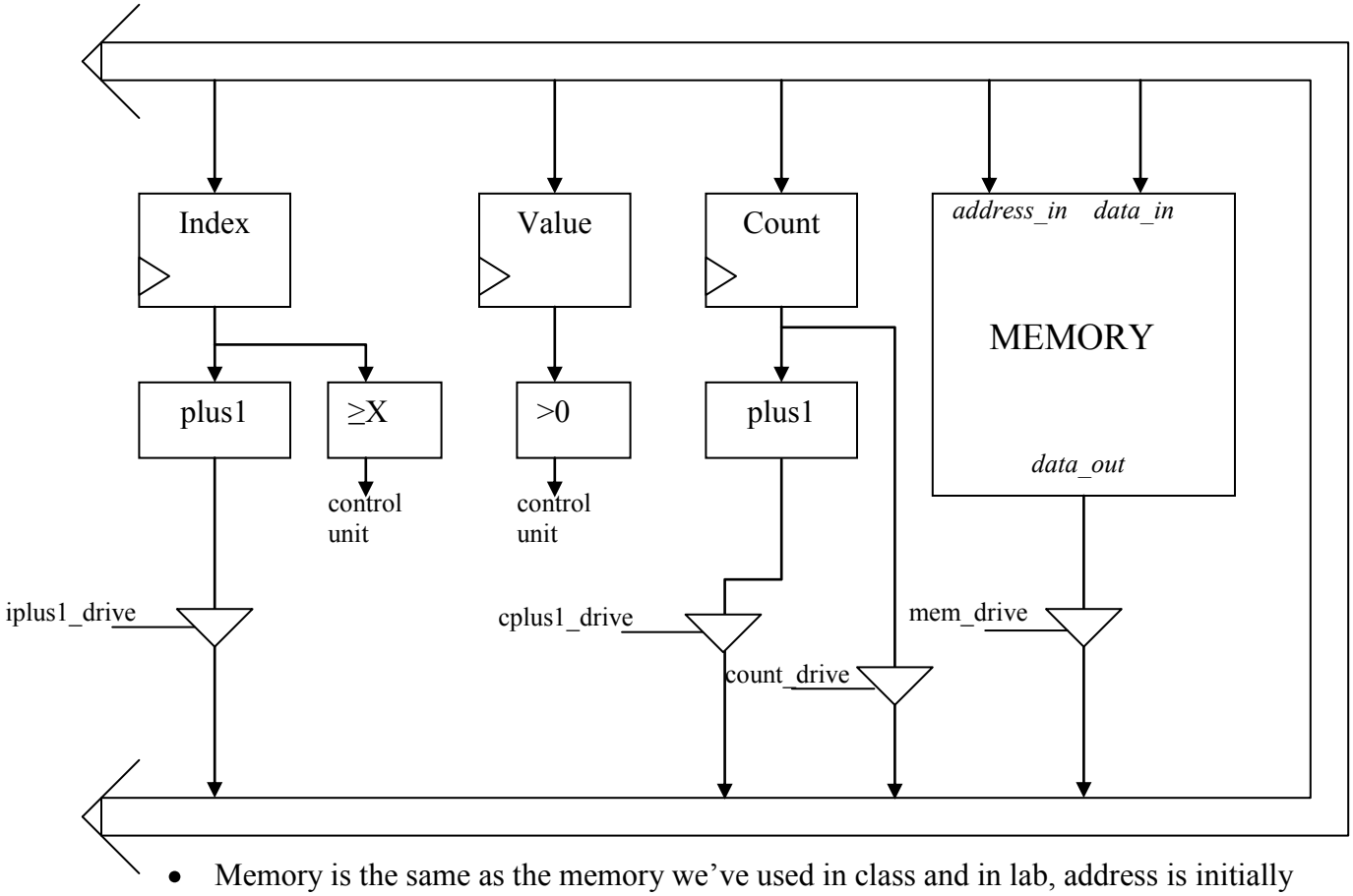
4) Consider an array, "BOB", which contains "X" elements. Write an e100-assembly program that finds the number of values in BOB that are less than or equal to zero. It should place the result in a memory location called "total". You may assume that BOB, X, and total have all been declared elsewhere. **[15]**

5) Complete the timing diagram for the following circuit. **[5]**





6) Consider the following datapath diagram.



- Memory is the same as the memory we've used in class and in lab, address is initially zero.
- Value, Index and Count are registers. They are all initially zero.
- Plus1 is a combinational block which outputs a value one greater than the input value.
- ≥X outputs a 1 if the input value is greater than or equal to X, otherwise it outputs a 0.
- >0 outputs a 1 if the input value is greater than 0, otherwise it outputs a 0.

You are to use this datapath to solve the following problem:

> You are to consider the first "X" memory locations and count the number of values in those memory locations that have a value less than or equal to zero. The final number should be stored in memory location X.

So if "X" were 10, memory locations 0 to 9 would be searched, and the number of them that have a value of less than or equal to zero would be placed into MEM[10].

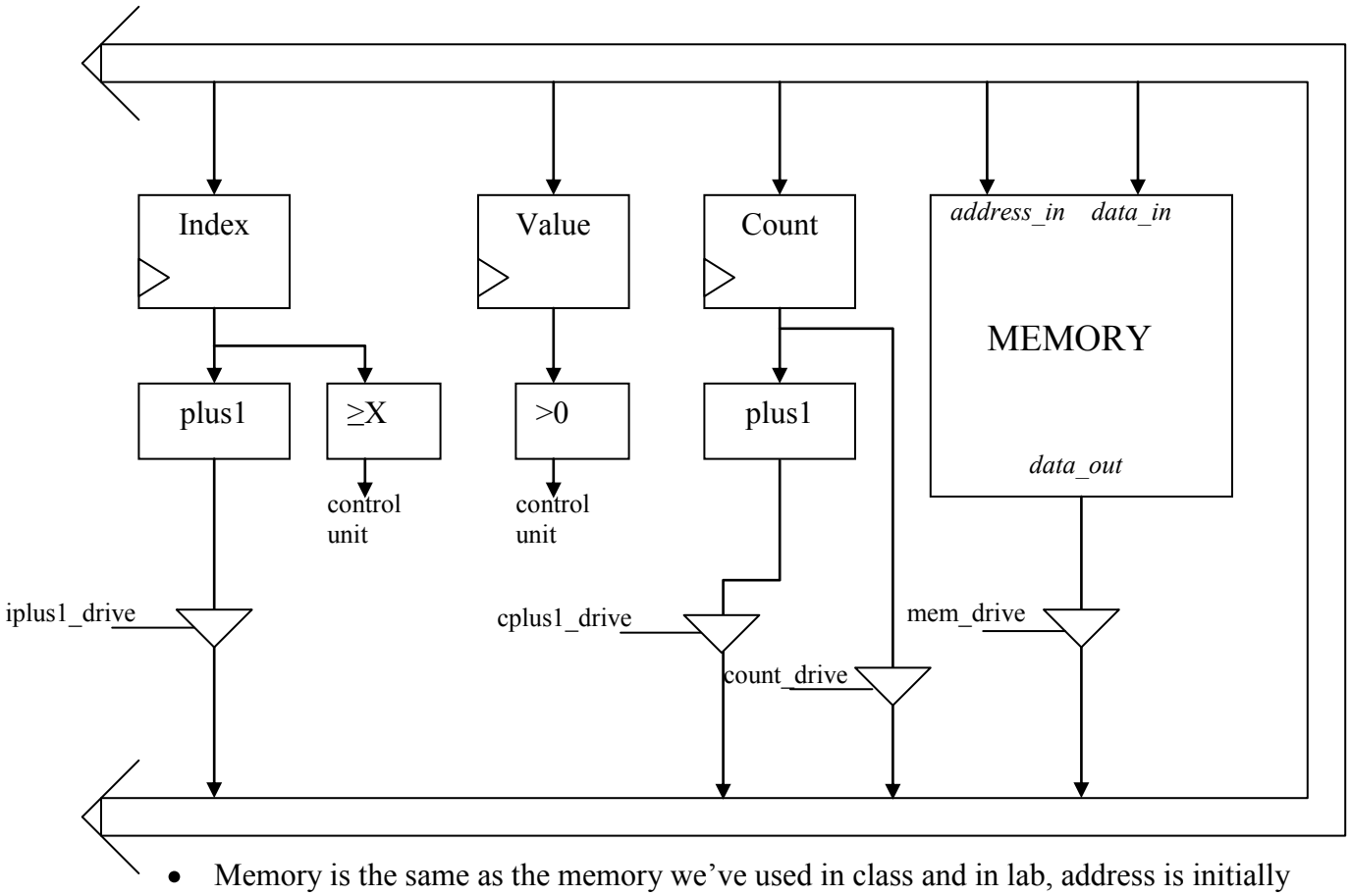
Write the control unit for this circuit by filling in the truth table on the following pages (you need not use all rows or columns shown). Use blanks in the cells for control signals to indicate a value of zero. Use blanks in the cells for input signals to indicate their value is ignored. Describe the actions of each row in the Comment column (use pseudo-code). There are likely more rows and columns than needed. Just leave the ones you don't use blank. **[45]**

CURRENT STATE					NEXT STATE												COMMENT

CURRENT STATE					NEXT STATE												COMMENT

Instruction name	Opcode	Effect
halt	0	PC = PC+4 stop executing instructions
add	1	PC = PC+4 memory[addr0] = memory[addr1] + memory[addr2]
sub	2	PC = PC+4 memory[addr0] = memory[addr1] - memory[addr2]
mult	3	PC = PC+4 memory[addr0] = memory[addr1] * memory[addr2]
div	4	PC = PC+4 memory[addr0] = memory[addr1] / memory[addr2]
cp	5	PC = PC+4 memory[addr0] = memory[addr1]
and	6	PC = PC+4 memory[addr0] = memory[addr1] & memory[addr2]
or	7	PC = PC+4 memory[addr0] = memory[addr1] | memory[addr2]
not	8	PC = PC+4 memory[addr0] = ~memory[addr1]
sl	9	PC = PC+4 memory[addr0] = memory[addr1] << memory[addr2]
sr	10	PC = PC+4 memory[addr0] = memory[addr1] >> memory[addr2]
cpfa	11	PC = PC+4 memory[addr0] = memory[addr1 + memory[addr2]]
cpta	12	PC = PC+4 memory[addr1 + memory[addr2]] = memory[addr0]
be	13	if (memory[addr1] == memory[addr2]) { PC = addr0 } else { PC = PC+4 }
bne	14	if (memory[addr1] != memory[addr2]) { PC = addr0 } else { PC = PC+4 }
blt	15	if (memory[addr1] < memory[addr2]) { PC = addr0 } else {PC = PC+4} Comparisons take into account the sign of the number. E.g., 16'hffff (-1) is less than 16'h0000 (0).
call	16	memory[addr1] = PC+4 PC = addr0
ret	17	PC = memory[addr0]
in	18	PC = PC + 4 memory[addr1] = data from I/O port addr0
out	19	PC = PC + 4 I/O port addr0 = memory[addr1]

THIS IS A COPY OF PROBLEM 6. YOU SHOULD RIP THIS OUT AND USE IT AS NEEDED



- Memory is the same as the memory we've used in class and in lab, address is initially zero.
- Value, Index and Count are registers. They are all initially zero.
- Plus1 is a combinational block which outputs a value one greater than the input value.
- ≥X outputs a 1 if the input value is greater than or equal to X, otherwise it outputs a 0.
- >0 outputs a 1 if the input value is greater than 0, otherwise it outputs a 0.

You are to use this datapath to solve the following problem:

> You are to consider the first "X" memory locations and count the number of values in those memory locations that have a value less than or equal to zero. The final number should be stored in memory location X.

So if "X" were 10, memory locations 0 to 9 would be searched, and the number of them that have a value of less than or equal to zero would be placed into MEM[10].

Write the control unit for this circuit by filling in the truth table on the following pages (you need not use all rows or columns shown). Use blanks in the cells for control signals to indicate a value of zero. Use blanks in the cells for input signals to indicate their value is ignored. Describe the actions of each row in the Comment column (use pseudo-code). There are likely more rows and columns than needed. Just leave the ones you don't use blank. **[45]**