Engineering 100 *Midterm Exam* Technical Part

Fall 2010

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	/10
3	/15
4	/15
5-7	/30
Total	**/70**

NOTES:

- This part of the exam is worth 70% of your grade.
- 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 or circle the correct answer **[10, 2 points each]**

a) Write the value ***-6*** as a 5-bit 2's complement number. _________________

b) Convert the 6-bit 2's complement number 001001 into decimal. _____________

c) The largest (closest to positive infinity) that can be written as an 8-bit 2's complement

number is: _______________

d) "Output solely determined by the current input" is a characteristic of which of the following?

i) Combinational logic
ii) Sequential logic
iii) Memory
iv) Anything in an always @(posedge clock) block.

e) Draw an AND gate where A and B are the inputs and C is the output.

2) Write an e100 assembly *function* called “odd” It is to take one arguments (odd_a) and it should return a 1 if that argument is odd and a -1 if it is even. The return value should be named odd_rv, and the return address should be named odd_ra. You should declare space for all needed variables here. **[15]**

3) Consider the following code:

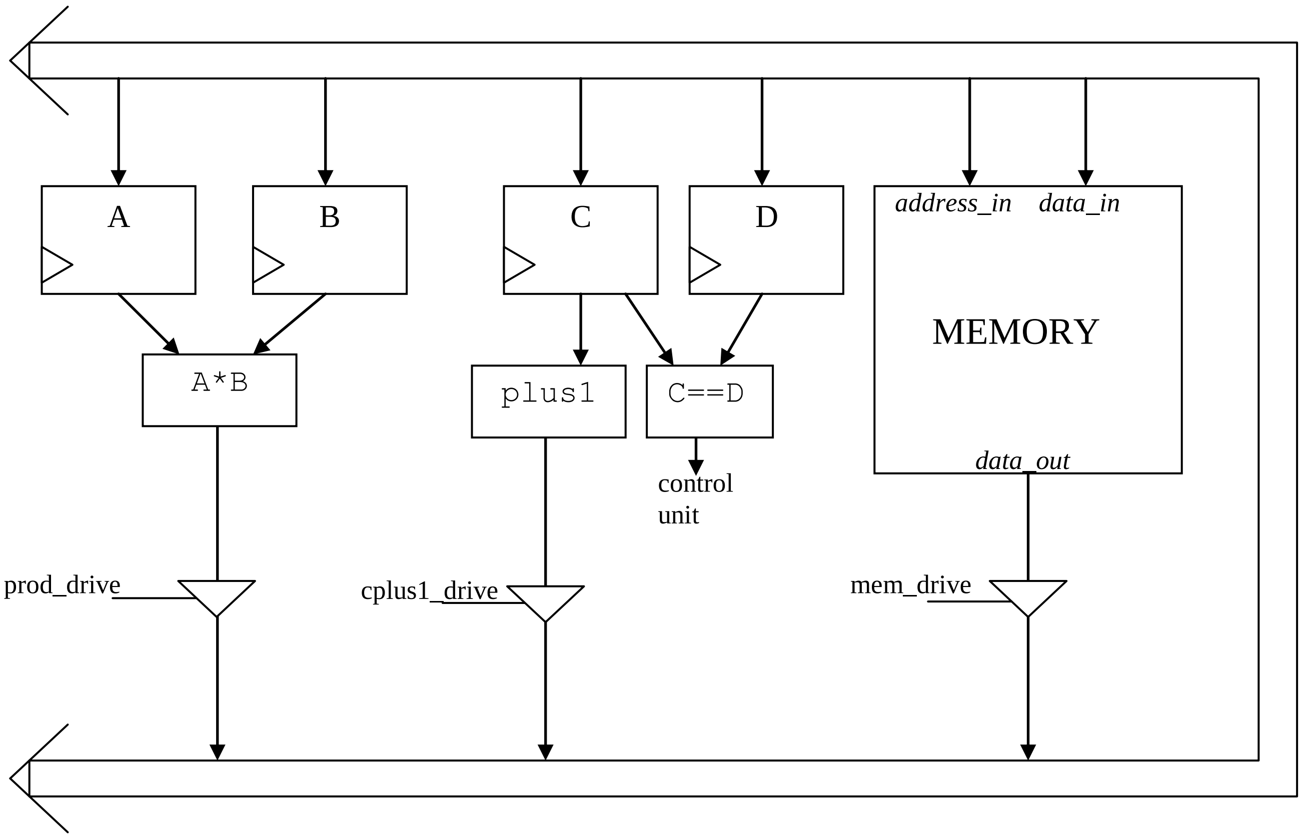
```
start     cpfa bob tom 4
          add  done one 1
done      halt
one       .data 1
bob       .data 20
tom       .data done
          .data -42
```

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 the following datapath diagram.



- Memory is the same as the memory we've used in class and in lab. Address is initially zero.
- A, B, C and D are registers. They are all initially zero.
- `Plus1` is a combinational block which outputs a value one greater than the input value.
- `A*B` is a combinational block which computes the product of A and B.
- `C==D` outputs a 1 if the value in register C is equal to the value in register D, otherwise it outputs 0.

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

Let X=memory[0]. For each memory location 1 to X you are to change that memory location to be its initial value times X. All other memory locations should be unchanged. You can assume that memory[0] is greater than or equal to 1.

So if you start with:	You should end with:
memory[0]=2	memory[0]=2
memory[1]=3	memory[1]=6
memory[2]=5	memory[2]=10
memory[3]=6	memory[3]=6

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. **[30]**

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).

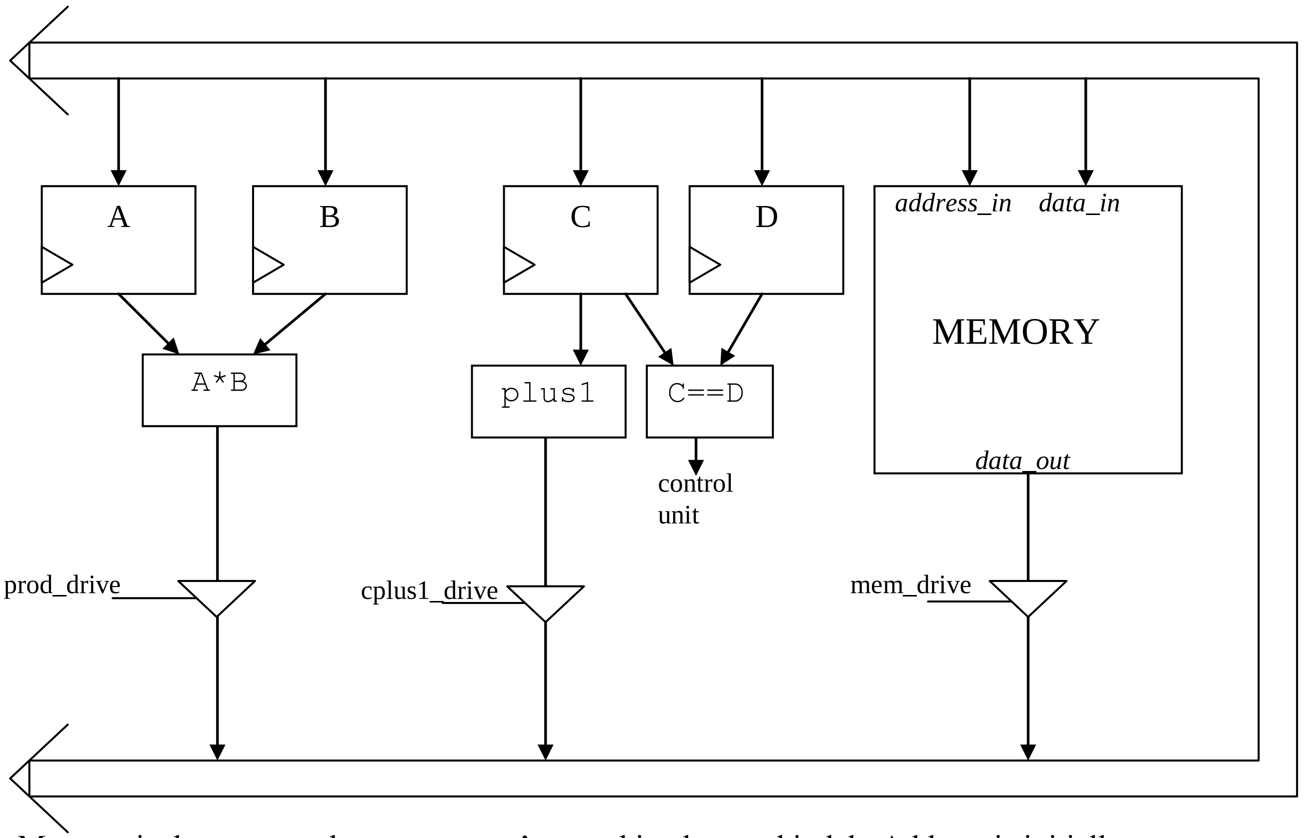
call	16	`memory[addr1] = PC+4`
		`PC = addr0`

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 4. 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.
- A, B, C and D are registers. They are all initially zero.
- `Plus1` is a combinational block which outputs a value one greater than the input value.
- `A*B` is a combinational block which computes the product of A and B.
- `C==D` outputs a 1 if the value in register C is equal to the value in register D, otherwise it outputs 0.

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

> Let X=memory[0]. For each memory location 1 to X you are to change that memory location to be its initial value times X. All other memory locations should be unchanged. You can assume that memory[0] is greater than or equal to 1.

So if you start with:	You should end with:
memory[0]=2	memory[0]=2
memory[1]=3	memory[1]=6
memory[2]=5	memory[2]=10
memory[3]=6	memory[3]=6

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. **[30]**