University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Spring 2016 Instructors: Vladimir Stojanovic, Nicholas Weaver 2016-04-04

After the exam, indicate on the line above where you fall in the emotion spectrum between "sad" & "smiley"...

Last Name	
First Name	
Student ID Number	
CS61C Login	**cs61c-**
*The name of your **SECTION** TA (please circle)*	Alex | Chris | Howard | Jack | Jason | Rebecca | Stephan | William
Name of the person to your LEFT	
Name of the person to your RIGHT	
*All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. **(please sign)***	

Instructions (Read Me!)

- This booklet contains 10 numbered pages including the cover page.
- Please turn off all cell phones, smartwatches, and other mobile devices. Remove all hats & headphones. Place your backpacks, laptops and jackets under your seat.
- You have 110 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use one handwritten 8.5"x11" page (front and back) of notes in addition to the provided green sheet.
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided.

	Q1	Q2	Q3	Q4	Q5	Q6	Total
Points Possible	16	6	12	20	20	6	80

SID: _____

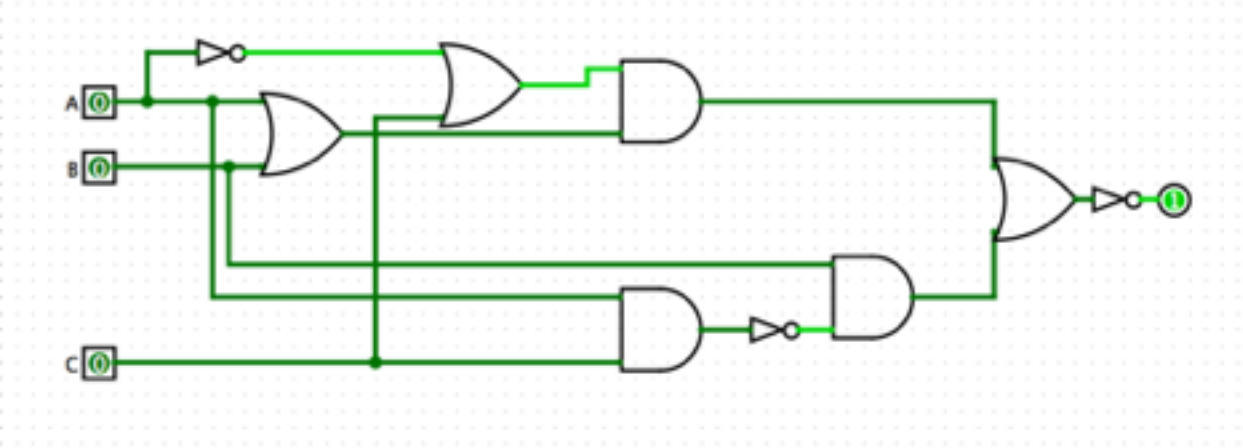
Corrections:

Q1c: Please only use 2-input AND and OR gates and 1-input NOT gates
Q1d: For the example, if the input is 111001, you would output 1011**1**0, not 101100.
Q4a: The comparator logic happens WITHIN the decode stage (just at the end)
Q4b: The End label is not part of the loop
Q5: Loop 1 should use N_ROWS and N_COLS instead of NUM_ROWS/NUM_COLS.
Cache is write back with write allocate.
Q6: S is a 2-bit unsigned number

SID: _____

Q1: Ben Bitdiddle's Plight (16 points)

Ben Bitdiddle was asked to design a new circuit to be used in an awesome top secret project. Unfortunately, Ben Bitdiddle isn't the best when it comes to circuit design, and this one has a lot of excess gates that aren't needed, so it's up to us to create a simpler one for him.



a) Fill out the truth table for this circuit. One row of the table is already filled out for you.

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

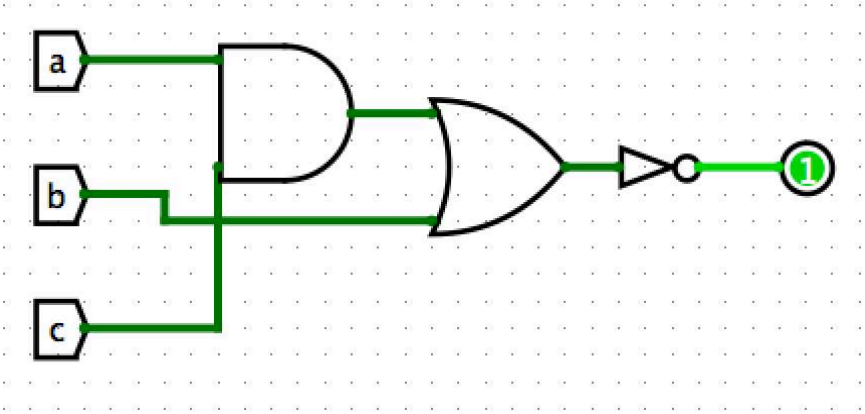
b) Given your truth table from part a, use sum of products to write the unsimplified, boolean expression for this circuit. This question will be graded independently of the correctness of part a. **DO NOT** simplify the expression.

(A ~B ~C) + (~A ~B C) + (~A ~B ~C)

c) To decouple this question from the previous parts, lets say after you derived the boolean expression and did some simplifying, you got:

$$(\bar{B}\bar{C})(A+\bar{A})+\bar{A}\bar{B}C$$

Using boolean logic, simplify the expression further so that it uses as few gates as possible. Draw the new circuit that corresponds the simplified expression. You may **only** use **binary (2 input) NOT**, **AND**, and **OR** gates

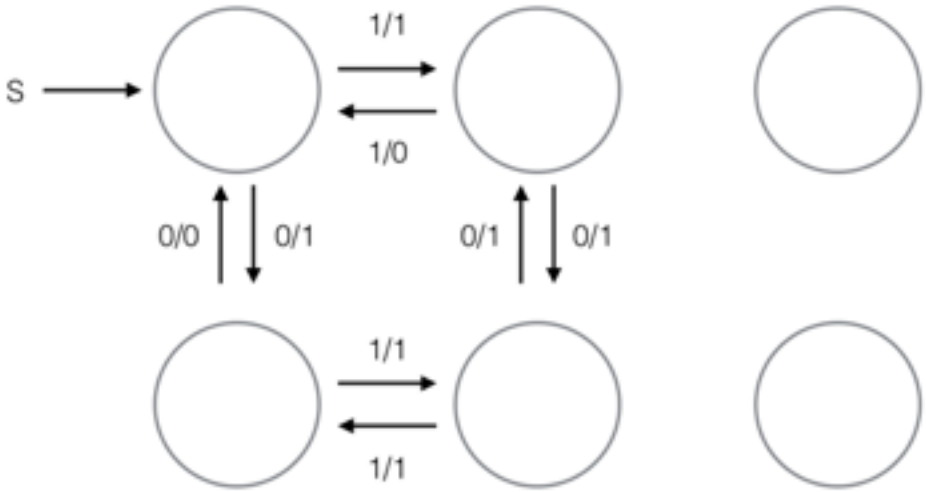


~(B + AC)

Ben's boss is very pleased with his work, and he gets another task for this super secret project!

d) Design a FSM that takes in either 0 or 1 as input, and outputs 1 if the total number of 1s seen is odd or the total number of 0s seen is odd, and 0 otherwise. For example, if the input is 111001, you would output 101110.

You may not need all of the provided states. One transition is given to you.



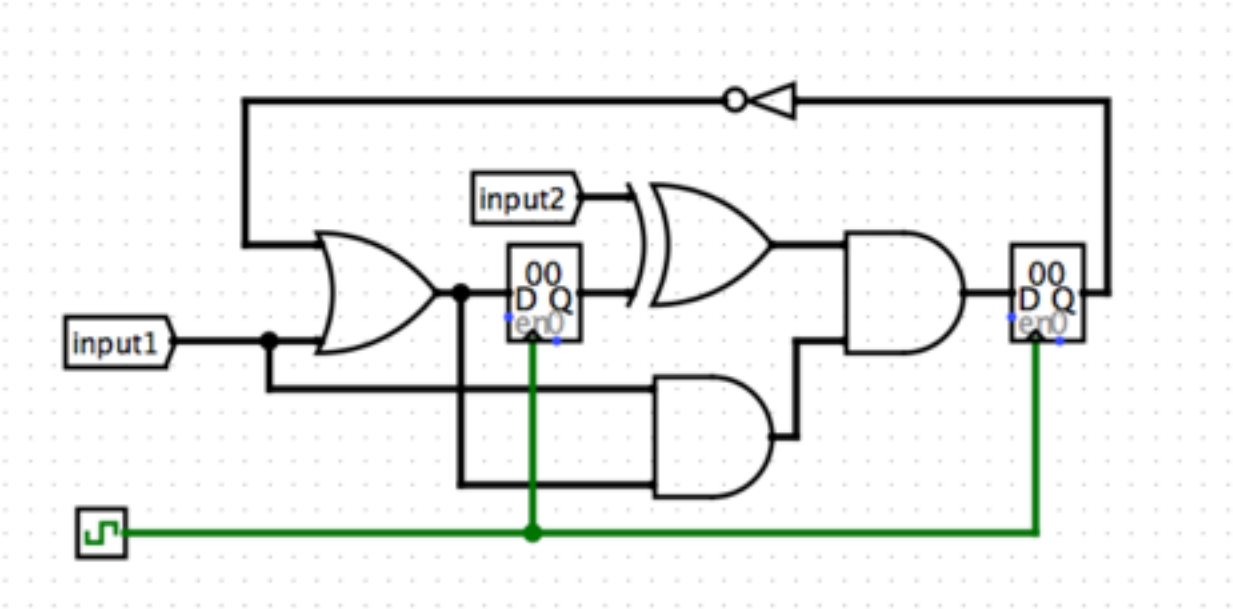
SID: _____

Q2: Timing question (6 points)

For the following circuit below, assume that circuit has the following timing criteria:

Inverter delay (NOT):	2	ns
OR block delay:	8	ns
XOR block delay:	16	ns
AND gate delay:	6	ns
Register setup time:	3	ns

The inputs come every 2 ns after every clock edge.



a) Assuming the clk-to-q delay is 1 ns, what is the shortest possible clock period at which we can run this circuit at?

_____27____ ns

right register:
a = input + XOR + AND + setup time = 27s (use the input over the clk-to-q)
b = clk-to-q + inverter + OR + AND + AND + setup = 26 ns

left register:
path will be shorter than either a or b from above

Taking the max = 27ns.

b) Assuming a clock period of 30 ns, what's the maximum possible clk-to-q of the circuit?

_____5____ ns

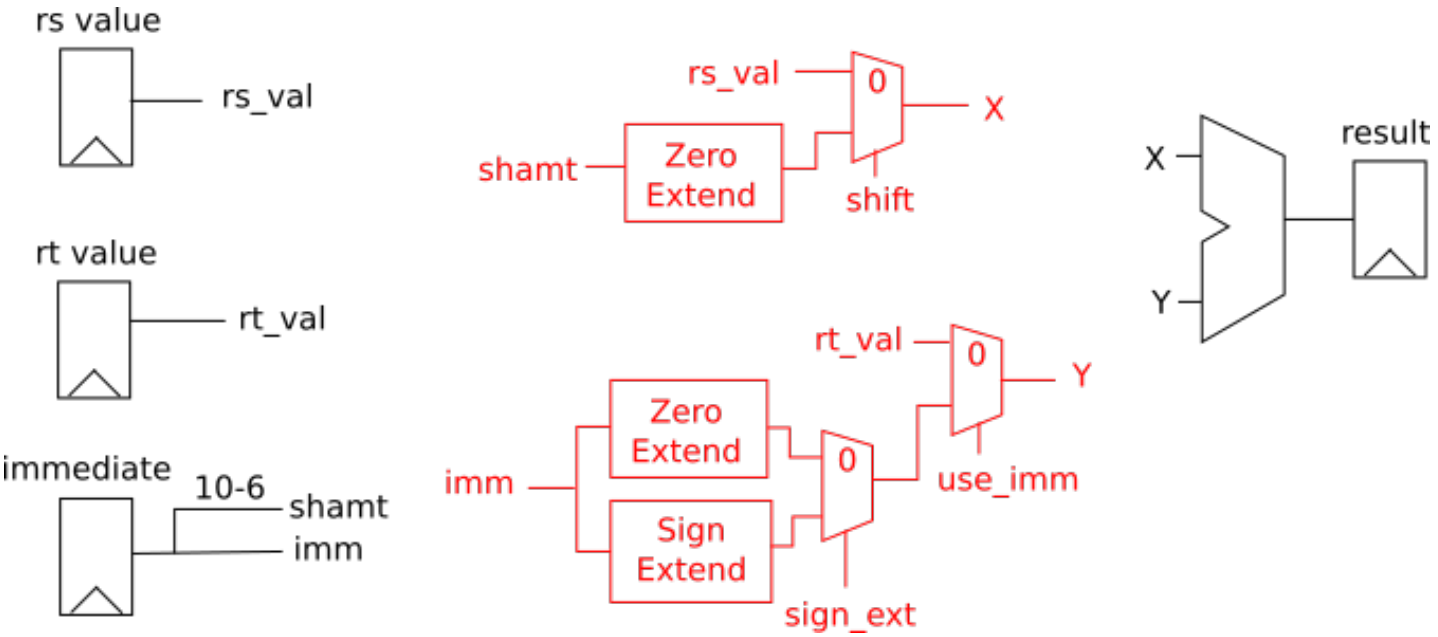
Take path b from above, which depends on clk-to-q. Instead of it being 1 ns, it can go up to 5 ns and sum up to 30 ns

SID: _____

Q3: A *DataPath*-finding Problem (12 points)

a) Below is an incomplete datapath for the third stage of a five-stage MIPS pipelined processor. Fill in the datapath so that it can execute any of the instructions given in the table below (part b).

- You may use 2-1 multiplexers, sign extenders, zero extenders, and any of the basic logic gates to build your datapath. Use tunnels whenever necessary to keep things clean.
- Do not worry about data hazards or forwarding logic. You also do not have to worry about the switch input on the ALU.
- You should use only the control signals given in the table.
- Assume that, for shifts, the ALU shifts Y by X.



b) Fill in the values the control signals should take for each instruction. If the value of a signal does not matter for a particular instruction, fill in an X for don't care.

Instruction	shift	sign_ext	use_imm
`add`	**0**	**x**	**0**
`addi`	**0**	**1**	**1**
`ori`	**0**	**0**	**1**
`sll`	**1**	**x**	**0**
`lw`	**0**	**1**	**1**

SID: _____

Q4: Pipelining (20 points)

Consider the following code segment:

```
Loop:   lw   $t1 0($t2)
        srl  $t1 $t1 16
        sw   $t1 0($t2)
        addi $t2 $t2 -4
        sub  $t4 $t3 $t2
        bne  $t4 $0  loop
End:    sll  $0  $0  $0
```

Assume that originally, `$t3 = $t2 - 196`

a) Assume a standard 5 stage pipeline with no forwarding. Register file writes happen before reads, in the same clock cycle. Comparator logic begins at the end of the **decode** stage. We do not have a branch delay slot. Fill in the corresponding pipeline stages (F, D, E, M, W) at the appropriate times in the table below.

If the instruction requires a stall, write the stage again in the table. For example, if an instruction starts at cycle 2 but needs two stalls for the execute stage, then you would write "F D E E E M W" for cycles 2-8.

Instr/Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
`lw $t1 0($t2)`	**F**	**D**	**E**	**M**	**W**														
`srl $t1 $t1 16`		**F**	**D**	**D**	**D**	**E**	**M**	**W**											
`sw $t1 0($t2)`			**F**	**F**	**F**	**D**	**D**	**D**	**E**	**M**	**W**								
`addi $t2 $t2 -4`						**F**	**F**	**F**	**D**	**E**	**M**	**W**							
`sub $t4 $t3 $t2`									**F**	**D**	**D**	**D**	**E**	**M**	**W**				
`bne $t4 $0 loop`										**F**	**F**	**F**	**D**	**D**	**D**	**E**	**M**	**W**	

b) How many cycles does this loop take to fully execute (from the first `lw` to the `End` label)?

___**196/4 * 15**__ cycles

c) Now assume the pipeline has 1 delayed branch slot and standard forwarding hardware. Also, reordering of instructions is allowed to minimize stalls. Write out the reordered sequence of instructions that achieves a minimal number of stalls needed. If you would like, you may also fill in the rest of the table below, but it **will not be graded**.

```
lw $t1 0($t2)
srl $t1 $t1 16
sw $t1 0($t2)
addi $t2 $t2 -4
sub $t4 $t3 $t2
bne $t4 $0 loop

The branch delay slot cannot be used.
```

d) How many cycles does this loop take to fully execute (from the first `lw` to the `End` label)?

The branch delay slot stalls for one cycle. The lw-srl also stalls for one cycle.

____**49*8**__ cycles

SID: _____

Q5: Time to *Cache* in on all of that studying (20 points)

Assume we have a direct mapped cache with 16 blocks. Each block holds 8 bytes. Assume also that both memory addresses and integer variables are 32 bits in size. You can also assume that the code starts executing with a cold cache.

a) What is the number of tag, index, and offset bits used by the cache?

Tag: ____25___
Index: _____4____
Offset: _____3____

Consider the following C code:

```
#define N_COLS 8
#define N_ROWS 8

int *a = malloc(sizeof(int) * N_ROWS * NCOLS);
int i, j;
int total = 0;

/* Loop 1 */
for (i = 0; i < NUM_ROWS; i++) {
    for (j = 0; j < NUM_COLS; j++) {
        a[i * N_COLS + j] = i + j;
    }
}

/* Loop 2 */
for (j = 0; j < N_COLS; j++) {
    for (i = 0; i < N_ROWS; i++) {
        total += a[i * N_COLS + j];
    }
}
```

Assume that array **a** is cache-aligned. Assume also that the compiler places `total`, `i`, and `j` in registers rather than in memory.

b) What is the cache hit rate (in percentage) when `Loop 1` is executed? If any cache misses occur, what kind are they (which of the 3 C's)?

The hit rate is 50%. We step through the array with stride 1, and each cache block holds two array elements. Accessing elements at even indexes induces cache misses, while accessing elements at odd elements induces cache hits.

The cache misses are compulsory misses.

c) What is the cache hit rate (in percentage) when `Loop 2` is executed? Note that this immediately follows the execution of Loop 1. If any cache misses occur, what kind are they (which of the 3 C's)?

The hit rate is 0%. Data from the end of the array is left over in the cache from Loop 1 but is all evicted before it would ever be needed. We step through the array with stride 8, so data loaded in as part of a block is not reused. During each iteration of the outer loop, data from the end of the array evicts data from the beginning of the array, so blocks are evicted before they would be used again, the next time through the loop.

The cache misses are capacity misses. If the cache were twice as large, we could fit all of array a in the cache, and there would be no cache misses at all in the second loop.

Now, say we double the number of blocks in the cache, so it now has 32 blocks each storing 8 bytes. Remember, we start with a cold cache.

d) What is the number of tag, index, and offset bits used by the cache?

Tag: ____**24**___
Index: _____**5**____
Offset: _____**3**____

e) What is the cache hit rate (in percentage) when `Loop 1` is executed? If any cache misses occur, what kind are they (which of the 3 C's)?

The hit rate is 50%. The same sequence of events occurs as before, the only difference being that all of the array a is in the cache by the end of Loop 1.

The cache misses are compulsory misses.

f) What is the cache hit rate (in percentage) when `Loop 2` is executed? Note that this immediately follows the execution of `Loop 1`. If any cache misses occur, what kind are they (which of the 3 C's)?

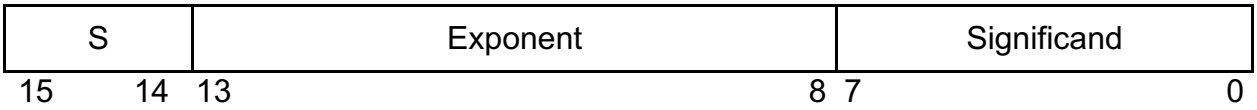
The hit rate is 100%. All of the array's elements are now in the cache, and no evictions occur as we step through the array, so every access results in a cache hit.

There are no cache misses to categorize.

SID: _____

Q6: This is a *Complex Floating Point* (6 points)

Consider the following 16-bit representation for complex, floating point numbers:



Bits per field:

- Sign: 2 (Now all numbers are in the form i^{Sign})
  Example - If you wanted to represent a number with -1 sign, your sign bits would be `10`
- Exponent: 6
- Significand: 8
- Everything else follows the IEEE standard 754 for floating point, except in 16 bits

Bias: 31

a) Convert 11.3125i into floating point using our new representation. Write your answer in **hexadecimal**.

`0x626A`

b) We want to be able to represent positive integers of the form $2^n - 8$. What is the largest such number that can be represented by the above complex floating point representation?

n = **12**

c) Are there more imaginary numbers, more real numbers, or an equal number of both with this representation? You can ignore infinities and NaNs for this question.

More imaginary **More real** **Equal**

0, 0i, -0, and -0i are all real.

Congratulations on finishing! Remember to fill in the emotional spectrum on the title page.