University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Summer 2015 Instructor: Sagar Karandikar 2015-07-28

CS61C MIDTERM 2

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	
Login	`cs61c-`
The name of your ***SECTION*** *TA (please circle)*	Derek | Harrison | Jeffrey | Nathaniel | Rebecca
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 exam contains 7 numbered pages including the cover page. **The back of each page is blank and can be used for scratch-work but will not be looked at for grading.** (i.e. the sides of pages without the printed "SID: ________" header will not even be scanned into Gradescope).
- 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 80 minutes to complete this exam. The exam is closed book; you may not use any computers, phones, wearable devices, or calculators. You may use one page (US Letter, front and back) of handwritten 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. "IEC format" refers to the mebi, tebi, etc prefixes.

	Q1	Q2	Q3	Q4	Q5	Total
Points Possible	12	17	11	20	17	77

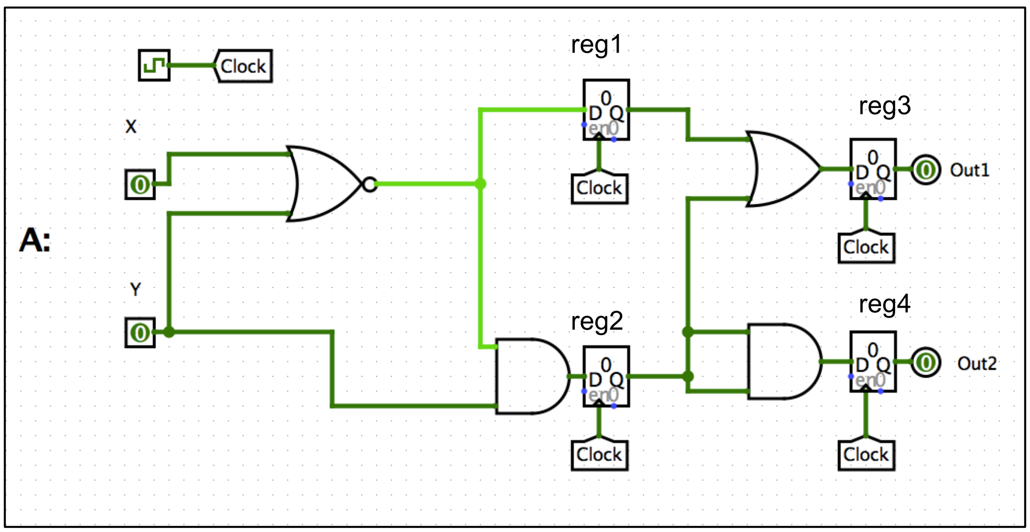
SID: _______________

Q1) Potpourri (12 pts)

a) Convert the following from single-precision IEEE 754 representation to decimal: **`0xFF800000`**

b) Convert the following from decimal to single-precision IEEE 754 representation. Report your answer as hex: **`-10.5625`**

The following sub-questions will use the circuit below.



Circuit specs:

- 15 ns clk-to-Q time
- Negligible hold time
- 20 ns logical gate propagation delay
- Gates with bubbles on the output have the same delay as "normal" gates
- Assume that X and Y arrive at the positive edge of the clock

Given Circuit A, with a clock period of 100ns, determine the maximum theoretical setup time we can have for this circuit to satisfy register-timing constraints (and function correctly). With this setup time, if signals X and Y are undefined until t = 0, when would Out1 be stable with the correct computed value?

c) Max setup time: _______________

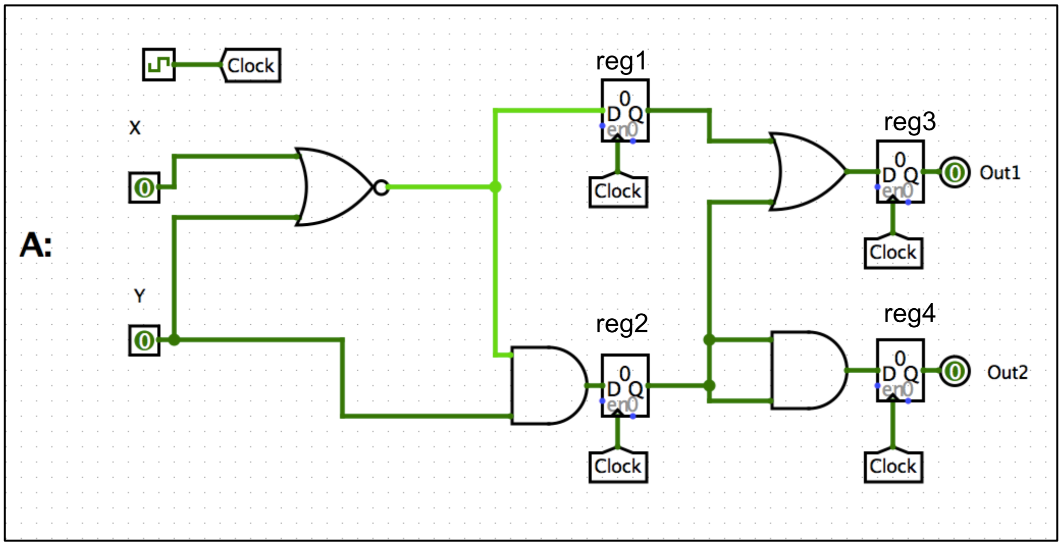
d) Out1 stable at t = _______________

e) We discover that our registers are faulty and have a 30ns hold time requirement. Assuming a setup time of 15ns and other timing parameters from the parts (a) and (b), is this a problem? If it is, suggest the simplest change that could be made so the circuit behaves as intended.

SID: ____________

Q2) Pipelining (17 pts)

a) Below is a copy of the diagram from the previous question. Now, you need to pipeline the circuit to improve the clock period. **Draw a star on any wire where you would place a pipelining register**. You may place up to 3 registers (but you may not need all 3). Your solution may introduce a cycle delay, but should not change the sequence of outputs after that initial delay (assuming the circuit gets a single uninterrupted stream of incoming X and Y values).



Circuit specs:
- **15 ns setup time**
- 15 ns clk-to-Q time
- Negligible hold time
- 20 ns logical gate propagation delay
- Gates with bubbles on the output have the same delay as "normal" gates
- Assume that X and Y arrive at the positive edge of the clock

b) Now, assume we have a Pipelined 5-stage MIPS CPU with the following specs:
- The CPU stalls on hazards, there is no forwarding
- Branch comparison happens in stage 2 and we **DO NOT** have a branch delay slot. (i.e. the branch "decision" is clocked into the PC at the end of stage 2).
- Both memory and registers **CAN** be written and read in the same clock cycle
- All Loads and Stores hit in the cache (ie. loads/stores take one cycle in the Mem stage)

Fill in the corresponding pipeline stages (F, D, E, M, W) at the appropriate times in the table below for the following 8 MIPS instructions assuming the above properties of your CPU. Suppose that any branches in the code are not taken for this specific instance. (You should use the back of the page for scratch work)

🡣 Instr / Cycle # 🡢	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`sll $v1 $v1 8`	F	D	E	M	W													
`xor $v0 $a1 $a2`																		
`addu $a1 $s3 $t1`																		
`andi $v0 $v0 1`																		
`addu $t0 $a0 $t1`																		
`lw $s0 0($t0)`																		
`bne $s0 $0 End`																		
`j Taketwo`																		

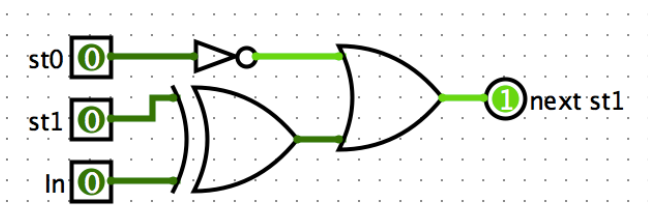
Q3) Synchronous Digital Systems (11 pts)

Use the truth table below for the following questions:

st_0	st_1	In	$next\_st_0$	$next\_st_1$	Out_0	Out_1
0	0	0	1		1	
0	0	1	0		0	
0	1	0	0		1	
0	1	1	1		0	
1	0	0	0		1	
1	0	1	0		0	
1	1	0	0		0	
1	1	1	0		0	

a) Write a Boolean formula for **$next\_st_0$** using sum of products. You should not simplify the expression you write. (Also, do not write a solution that you get "intuitively").

b) Use the circuit below to fill in $next\_st_1$ in the truth table.



c) Simplify the following expression so it can be implemented with 3 basic gates (AND, OR, or NOT). Your solution should require as few gates as possible. You do not need to fill in the table above.

$$Out1 = \left((\overline{st0} * \overline{st1}) + (\overline{st0} * st1) + (st0 * \overline{st1})\right) * \overline{In}$$

SID: ________________

Q4) Datapath (20 pts)

Given the new instruction **`beqalr`** represented by the RTL below, answer the following questions.

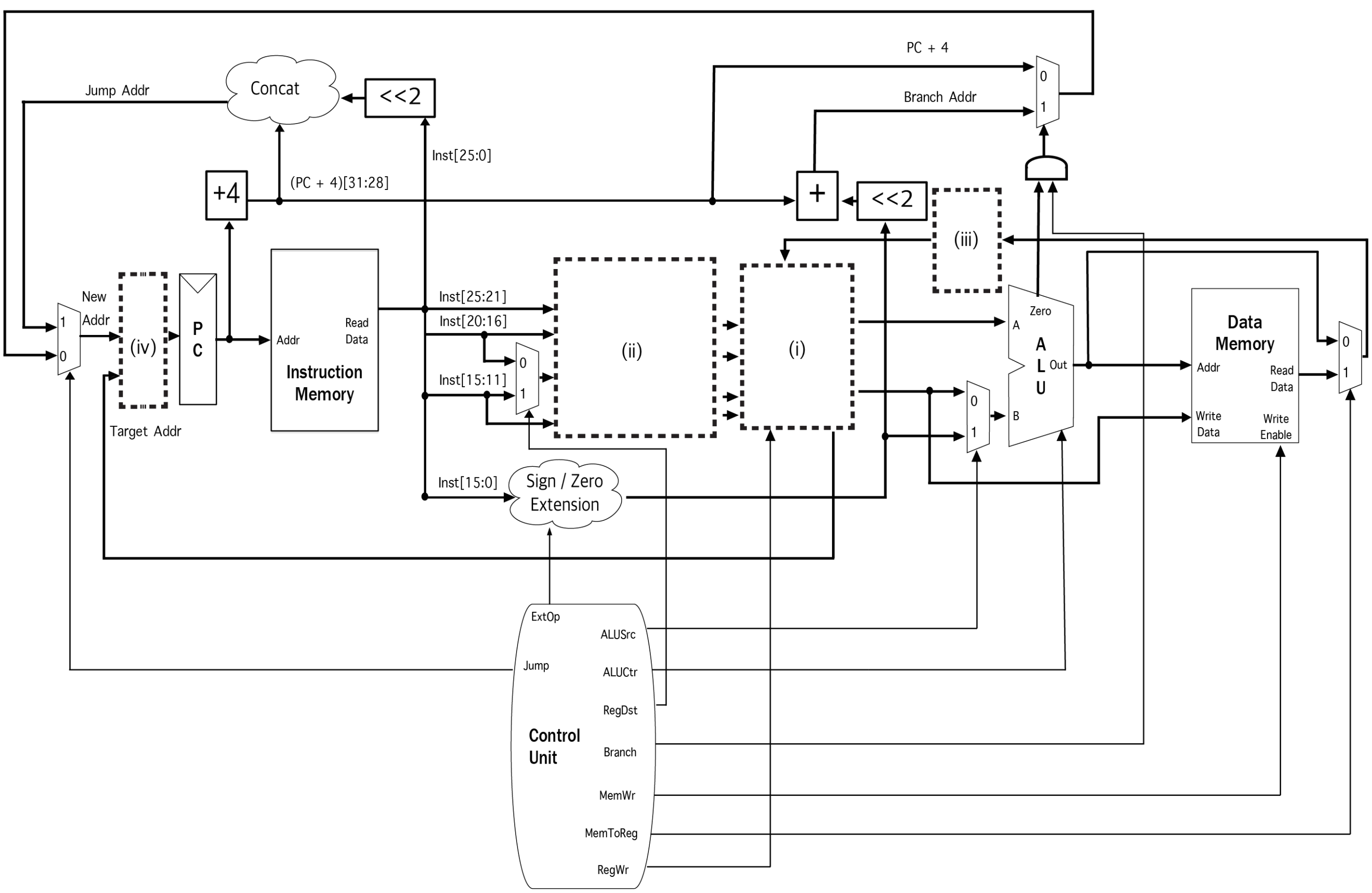
Instruction	RTL
`beqalr rd, rs, rt`	**`if (R[rs] == R[rt]) {`** **`R[31] = PC + 4;`** **`PC = R[rd];`** **`}`**

a) Briefly describe what we would use this instruction for in a program. (Hint: think about what C code that gets compiled into this instruction would look like.)

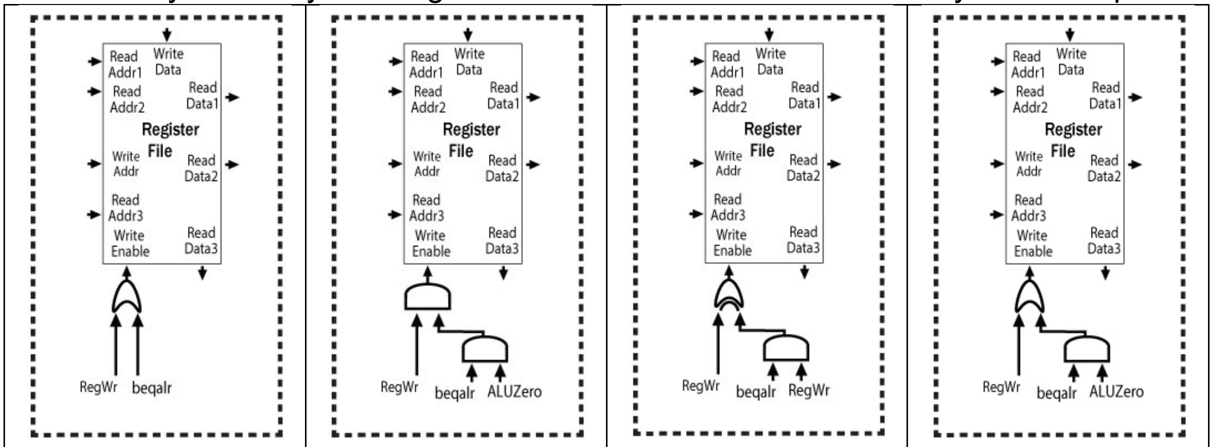
b) Write equivalent TAL MIPS for the **`beqalr`** instruction. **`$ra`** should be set to the instruction after the expanded **`beqalr`** (labeled "**`resume`**"). **Our system does not have delay slots.**

`beqalr $s0 $t0 $t1`	**`beqalr: bne`** ____________________ ____________________ **`resume:`**

c) Now we want to modify our pipeline so that it is able to execute the **`beqalr`** instruction. Below (and attached to the green sheet), you will find a base CPU diagram with empty boxes. For each missing piece in the datapath, circle which options on the next page are the best in implementing **`beqalr`**. If there are multiple working solutions, **choose the one with the least hardware**. The control signal for **`beqalr`** is 1 if and only if the instruction is **`beqalr`**. **`RegWr`** is **`0`**.

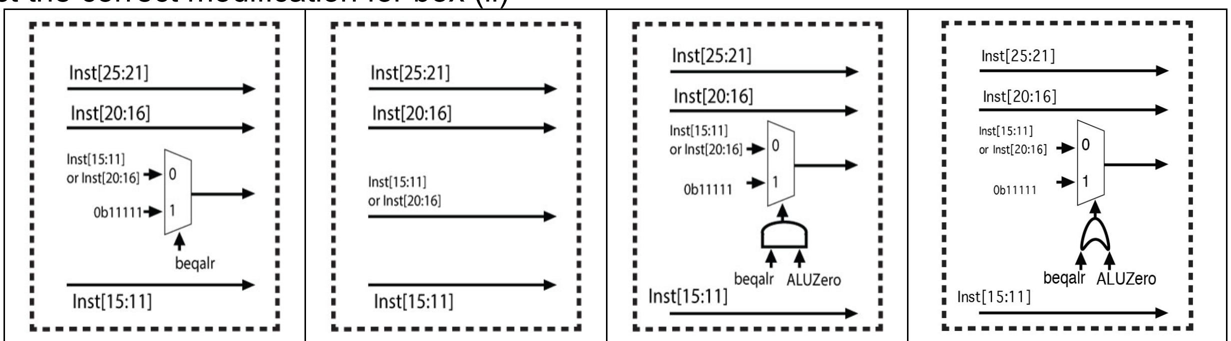


i) We have the ability to modify our RegFile. Which modification is necessary for our implementation?



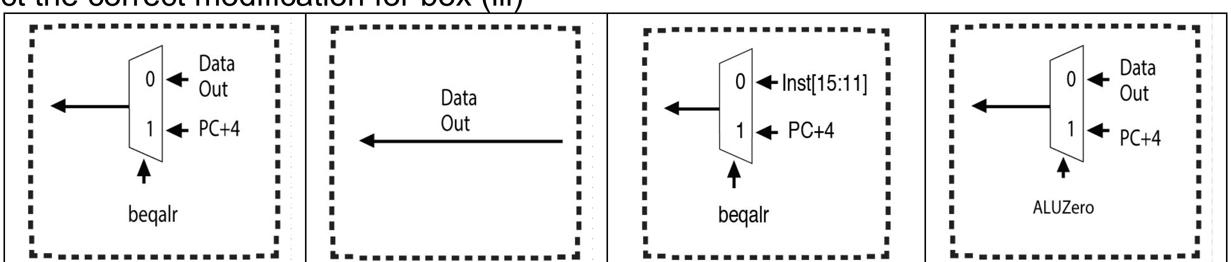
1	2	3	4

ii) Select the correct modification for box (ii)



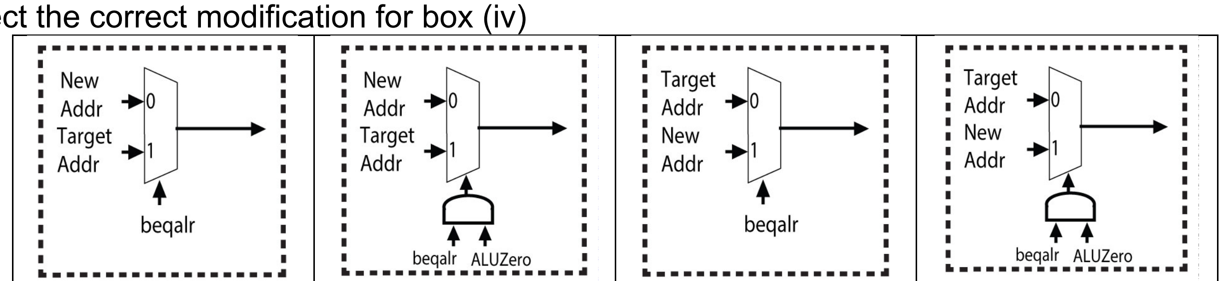
1	2	3	4

iii) Select the correct modification for box (iii)



1	2	3	4

iv) Select the correct modification for box (iv)



1	2	3	4

d) Generate the control signals for **`beqal`**. The values should be 0, 1, or X (don't care) terms. You must use don't care terms where possible.

beqalr	RegDst	ExtOp	RegWr	ALUSrc	ALUCtr	MEMWr	MemToReg	Jump	Branch
1			0		XXXX				

SID: ________________

Q5) Caches (17 pts)

Assume we are working in a 32-bit physical address space. We have two possible data caches: cache X is a direct-mapped cache, while cache Y is 2-way associative with LRU replacement policy. Both are 4 KiB caches with 512 B blocks and use write-back and write-allocate policies.

a) Calculate the number of bits used for Tag, Index and Offset:

Cache	Tag bits	Index bits	Offset bits
X			
Y			

Use the code below to answer the following parts. Assume that ints are 4 B and doubles are 8 B.

```
int DOUBLE_ARRAY_SIZE = 2 * 1024;
double double_arr[DOUBLE_ARRAY_SIZE];

for (int i = 0; i < DOUBLE_ARRAY_SIZE; i++) /* loop 1 */
      double_arr[i] = i;
for (int i = 0; i < DOUBLE_ARRAY_SIZE;  i += 8) /* loop 2 */
     double_arr[i] *= double_arr[0];
```

b) What is the hit rate for each cache if we run only loop 1? (hint: they're both the same). What types of misses do we get?

c) What is the hit rate of each cache when you execute loop 2? Assume that you have executed loop 1. Assume the worst case ordering of accesses within a single iteration of the loop if multiple orders are possible. You may leave your answer as an expression involving products and sums of fractions.

X: ________________________________ Y: ________________________________

d) Compute the AMAT for the following system with 3 levels of caches. (You should not need any information from the previous parts of this problem.) <u>Give your answer as a decimal value.</u>

L1$	L2$	L3$	Main Memory
Global miss rate: 50% Hit time: 1ns	Local miss rate: 20% Hit time: 5ns	Local miss rate: 1% Hit time: 15ns	Hit time: 500ns