Wawrzynek & Weaver CS 61C
Sp 2018 Great Ideas in Computer Architecture

MT 2

PRINT your name: ______________________, ______________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will be reported to the Center for Student Conduct, and may result in partial or complete loss of credit.

SIGN your name: ____

PRINT your class account login: `cs61c-`______ and SID: ______________________

Your TA's name: ____

Your section time: ____

Exam # for person sitting to your left: ____________

Exam # for person sitting to your right: ____________

You may consult two sheets of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

You have 110 minutes. There are 5 questions, of varying credit (90 points total). The questions are of varying difficulty, so avoid spending too long on any one question. Parts of the exam will be graded automatically by scanning the **bubbles you fill in**, so please do your best to fill them in somewhat completely. Don't worry—if something goes wrong with the scanning, you'll have a chance to correct it during the regrade period.

If you have a question, raise your hand, and when an instructor motions to you, come to them to ask the question.

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	Total
Points:	17	17	19	20	17	90

Figure 1: This Space Deliberately Left Blank

Problem 1 *RISCy Business* (17 points)

Bubble in one answer per question:

(a) Select the stage that computes the offset for a `beq` instruction.

- ○ Compiler
- ○ Linker
- ○ Assembler
- ○ Loader

(b) Select the stage that computes the offset for a `jal` instruction to a function from a different object file.

- ○ Compiler
- ○ Linker
- ○ Assembler
- ○ Loader

(c) Select the stage that creates a symbol table and a relocation table.

- ○ Compiler
- ○ Linker
- ○ Assembler
- ○ Loader

(d) Select the stage that emits `add s0, s1, s2`.

- ○ Compiler
- ○ Linker
- ○ Assembler
- ○ Loader

(e) Select the stage that combines several object files into a single executable.

- ○ Compiler
- ○ Linker
- ○ Assembler
- ○ Loader

(f) Complete the following RISC-V procedure `jal_address_fixing` that handles address relocation for all `jal` instructions. It first calls `find_next_jal` to find a `jal` instruction that does not yet have its offset filled in (the immediate bits are all zeroes), calculates the jump offset, and fills the immediate field of the `jal` instruction.

- Fill in **one** instruction for each of the 5 blanks.
- You can assume `jal_address_fixing` has the ability to modify text segment instruction memory.
- The function `find_next_jal` returns two values: the first is the address of a `jal` instruction stored in `a0`; the second is the address of the target instruction this `jal` instruction is jumping to stored in `a1`. If there are no more `jal` instructions to fill in offsets for, it returns 0 and 0.

```
jal_address_fixing:
    jal ra, find_next_jal
    beq a0, x0, DONE

    _______________________     # set a1 as the jump offset
IMM_20:         # sets imm[20]

    _______________________
    slli a5, a5, 31     # a5 has imm[20]
IMM_19_12:     # sets imm[19:12]
    li a3, 0xFF000
    and a3, a1, a3
    or a5, a5, a3     # now a5 has imm[20] and imm[19:12]
IMM_10_1:     # sets imm[10:1]
    li a3, 0x7FE
    and a3, a1, a3
    slli a3, a3, 20
    or a5, a5, a3     # now a5 has imm[20], imm[10:1], and imm[19:12]
IMM_11:     # sets imm[11]
    li a3, 0x800
    and a3, a1, a3
    slli a3, a3, 9
    or a5, a5, a3     # now a5 has imm[20], imm[10:1], imm[11], and imm[19:12])
UPDATE:     # inserts immediate into jal instruction

    _______________________          # load the current jal instruction

    _______________________          # insert the immediate

    _______________________          # save the updated instruction
    j jal_address_fixing          # jump back to fix the next one
DONE:
    ...
```

(g) The above code works for a `jal` target address that is 2^{16} bytes smaller than the `jal` instruction address.

○ True ○ False

(h) The above code works for a `jal` target address that is 2^{24} bytes larger than the `jal` instruction address.

○ True ○ False

Problem 2 ***Tell Us The Truth*** **(17 points)**

X	Y	Z	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

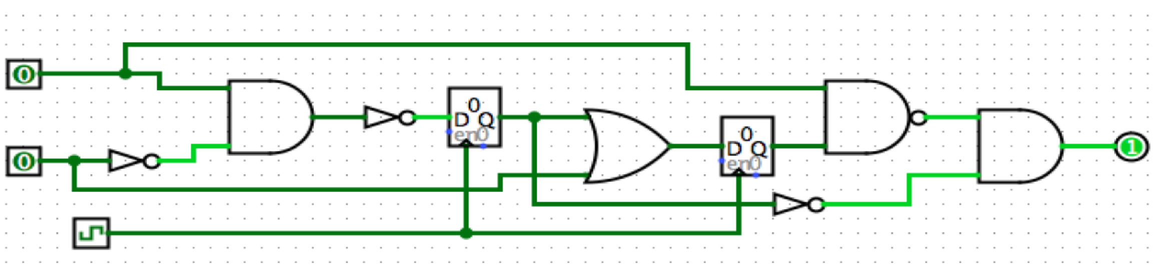
(a) Select all of the following expressions that are equivalent to the truth table above.

- ○ $(X + \bar{Y} + Z)(\bar{X} + \bar{Y} + Z)$
- ○ $X\bar{Y}Z + \bar{X}\bar{Y}Z$
- ○ $\bar{Z} + Y$
- ○ $X\bar{Y} + \bar{X}\bar{Y} + Z\bar{X} + ZX$
- ○ $\bar{Y} + Z$
- ○ $\bar{Y} + \bar{Y}Z + Z$

(b) Suppose you wanted to implement $\bar{A} + B$, but the only available gates are NAND gates. What is the minimum number of NAND gates you need to implement the above truth table correctly?

____

Now, consider the following circuit:



You are given the following timing parameters: Register Clk-To-Q: 2ps, Register Setup: 2ps, NOT Gate: 1ps, AND Gate: 4ps, OR Gate: 3ps, NAND Gate: 4ps. **Assume the 2 inputs comes from registers and the output is connected to a register as well.**

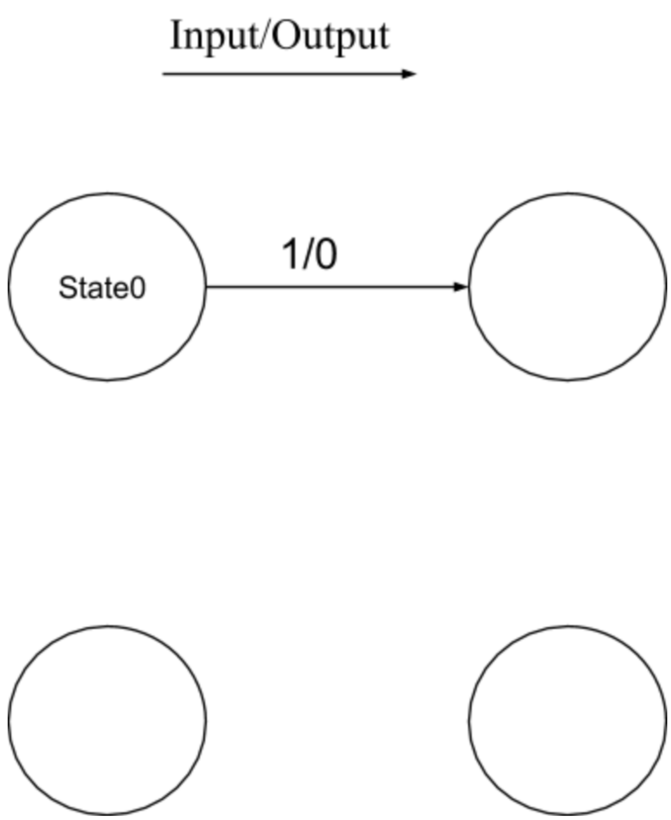
(c) What is the minimum clock period at which this circuit can be run?

____

(d) What is the maximum hold time that would allow for this circuit to run correctly?

____

(e) Draw the State Transition Diagram for a Finite State Machine that, given a sequence of binary digits, outputs 1 if the second most recently seen digit is 1 and outputs 0 otherwise. For example, an input of 01100111 has the output 00110011. Label all states and transition inputs/outputs you draw. You may or may not need all 6 states.



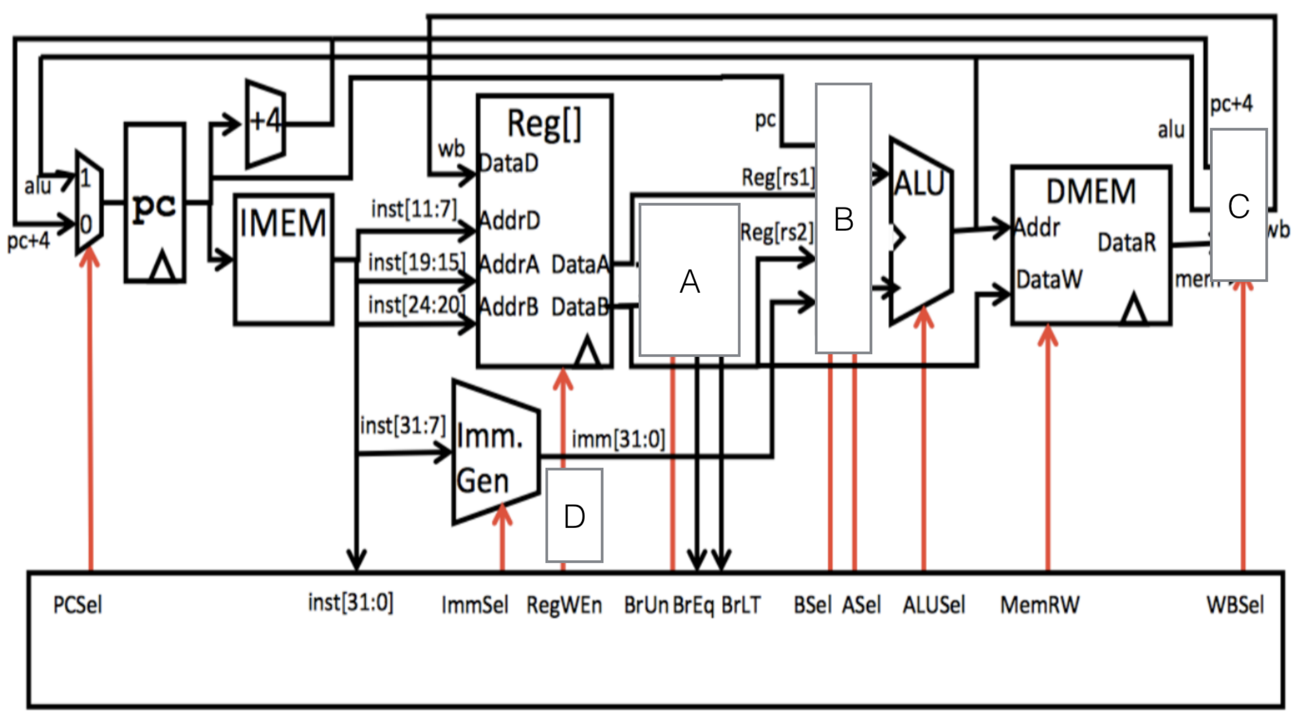
Problem 3 ***Set ... If Zero*** **(19 points)**

We wish to introduce a new instruction into our single-cycle datapath. The instruction **SIZ** (set if zero) works as follows:

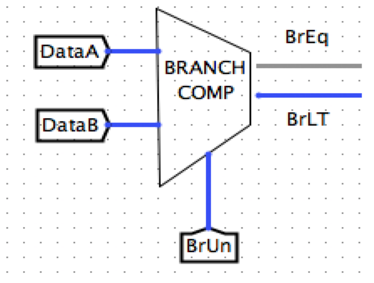
```
if (R[rs2] == 0):
    R[rd] = R[rs1]
```

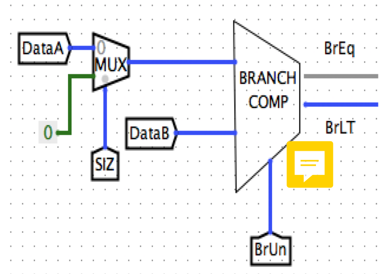
Given the single cycle datapath below, select the correct modifications in parts (a) - (d) such that the datapath executes correctly for this new instruction (and all core instructions!). You can make the following assumptions:

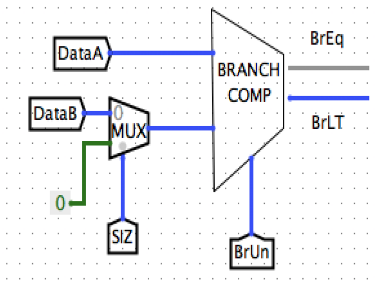
- the `SIZ` signal is 1 if and only if the instruction is `SIZ`
- `ALUSel` is **ADD** when when we a have `SIZ` instruction.
- **the immediate generator outputs ZERO** when we have a `SIZ` instruction.

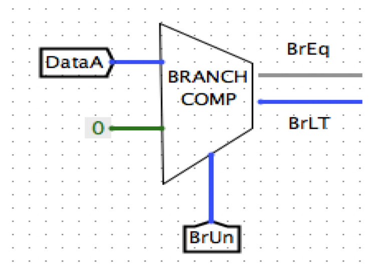


(a) Consider the following modifications to the branch comparator inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set?

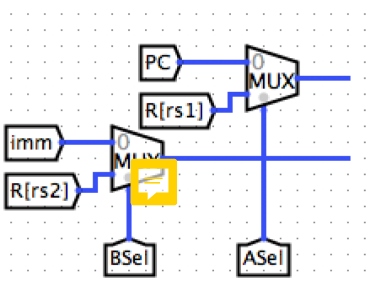
○ 

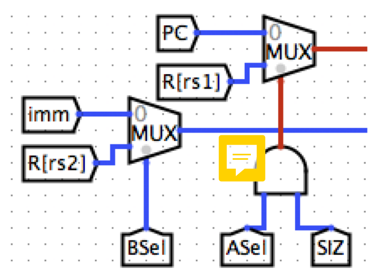
○ 

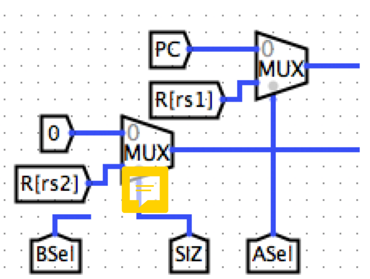
○ 

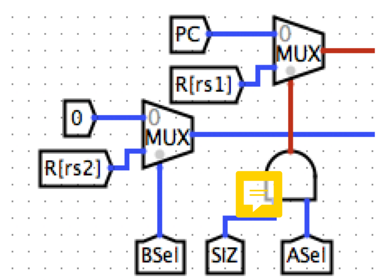
○ 

(b) Consider the following modifications to the ALU inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set? Select the configuration that requires **minimum** modifications to the original datapath. Notice in the bottom left choice `BSel` is unused.

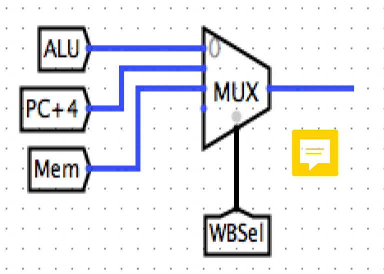
○ 

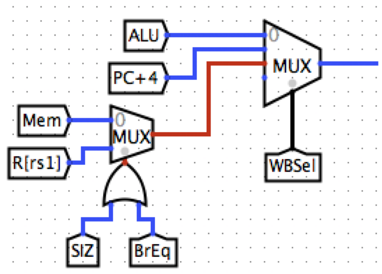
○ 

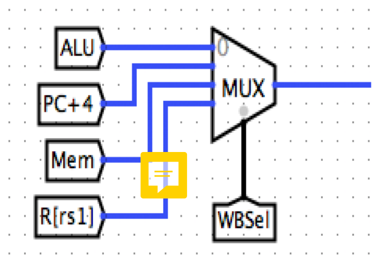
○ 

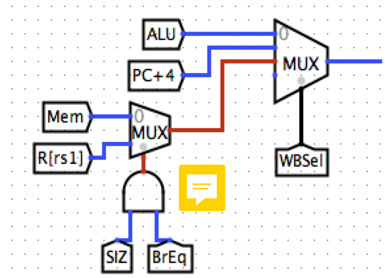
○ 

(c) Consider the following modifications to the WB mux inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set? Select the configuration that requires **minimum** modifications to the original datapath.

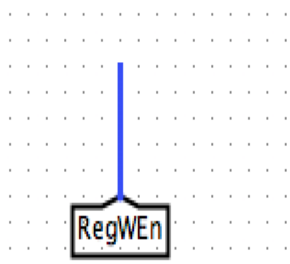


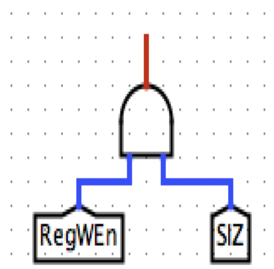


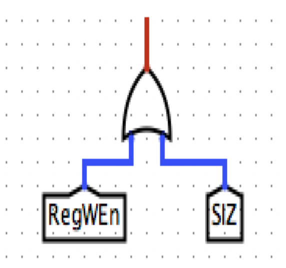


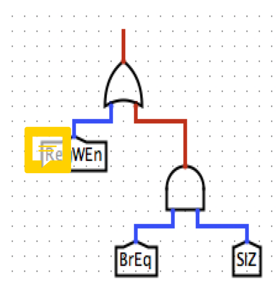


(d) Consider the following modifications to the RegWEn inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set?









(e) Given your selections above, decide the rest of the control signals for this instruction based on the diagram given at the beginning of the problem. Select `X` when a signal's value doesn't matter. You can assume:

- the `SIZ` signal is 1 if and only if the instruction is `SIZ`
- `ALUSel` is **ADD** when when we a have `SIZ` instruction.
- the immediate generator outputs **ZERO** when we have a `SIZ` instruction.

1. `PCSel`:

   ○ 1   ○ 0   ○ X

2. `RegWEn`:

   ○ 1 (Enable)   ○ 0 (Disable)   ○ X

3. `BrUn`:

   ○ 1 (Signed)   ○ 0 (Unsigned)   ○ X

4. `BSel`:

   ○ 1   ○ 0   ○ X

5. `ASel`:

   ○ 1   ○ 0   ○ X

6. `MemRW`:

   ○ 1 (Enable)   ○ 0 (Disable)   ○ X

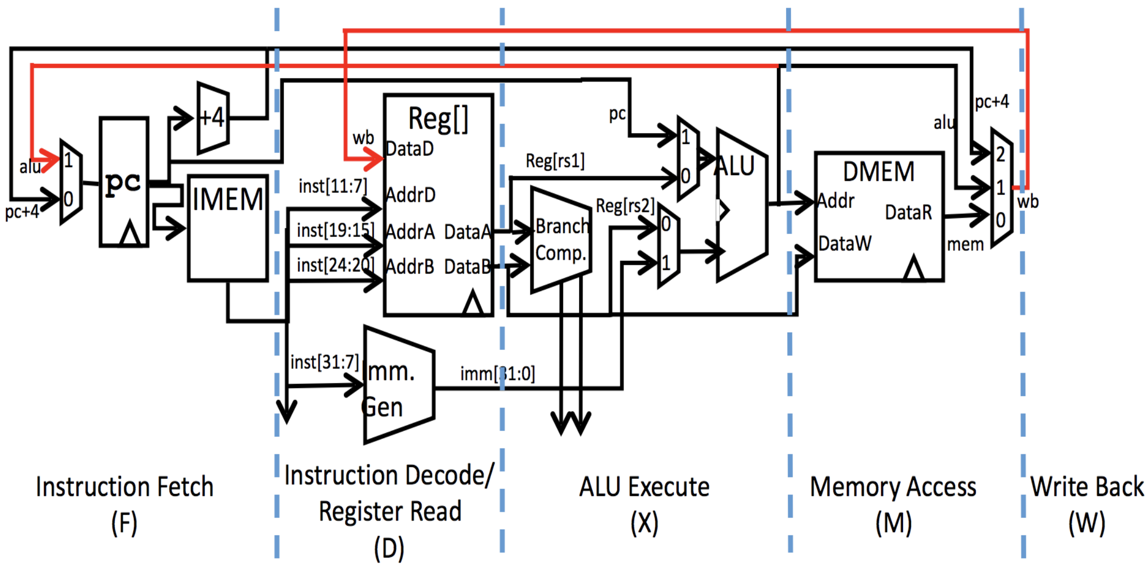
7. `WBSel`

   ○ ALUOut   ○ MemOut

   ○ PC + 4   ○ Other: Please specify:_______________

Problem 4 ***More or Less*** **(20 points)**

Consider a typical 5-stage (**F**etch, **D**ecode, **EX**ecute, **M**emory, **W**rite**B**ack) pipeline. Assume pipeline registers exist where the dotted lines are.



For this question, consider the following parameters:

- Forwarding is not implemented
- The branch predictor always predicts the branch is not taken. Flush the pipeline if prediction is wrong.
- We cannot read and write from the same registers or memory address in the same clock cycle.
- No other optimizations are implemented in this datapath.

(a) Fill in the corresponding pipeline stages for the code sequence below for the 5-stage pipeline. The first instruction is done for you. If you need to stall a cycle, write "*" in that cycle.

```
begin:
        ori s1, x0, 0xF
        andi s2, x0, 0
        beq s1, s2, exit
        lw s1, 0xc(s0)
        xor s1, s1, s2
exit:
        lw s1, 0xc(s0)
```

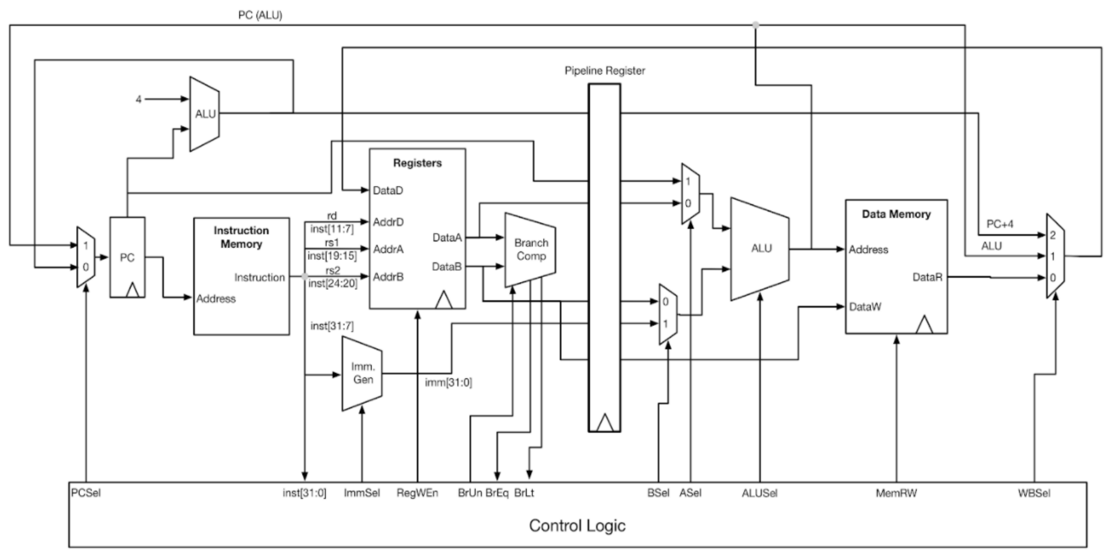
Instructions	Cycles																			
	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	C19	C20
ori s1 x0 0xf	F	D	E	M	W															
andi s2 x0 0																				
beq s1 s2 exit																				
lw s1 0xc(s0)																				
xor s1 s1 s2																				
lw s1 0xc(s0)																				

(b) Assume the maximum delays through each stage are: `F: 200ns, D: 150ns, EX: 100ns, MEM: 300ns, WB: 250ns`.

Assume the delays for the pipeline registers are factored into the pipeline stage delays. What is the latency and best case throughput of this 5-stage pipelined CPU? You may leave your answers as fractions. Don't forget the units!

Latency:______________ Throughput: ______________

You decide to combine certain stages to make a two-stage pipeline by removing certain registers. The two-stage pipelined datapath looks like the following:



Again, consider the following parameters:

- Forwarding is not implemented
- The branch predictor always predicts the branch is not taken. Flush the pipeline if prediction is wrong.
- We cannot read and write from the same registers or memory address in the same clock cycle.
- No other optimization is implemented on this datapath.

(c) Fill in the corresponding pipeline stages for the same code sequence for the 2-stage pipelined CPU. The code sequence is reproduced below. Use “A” to denote stage 1, “B” for stage 2. The first instruction is done for you. If you need to stall a cycle, write “*” in that cycle.

```
begin:
        ori s1, x0, 0xF
        andi s2, x0, 0
        beq s1, s2, exit
        lw s1, 0xc(s0)
        xor s1, s1, s2
exit:
        lw s1, 0xc(s0)
```

Instructions	Cycles													
	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
ori s1 x0 0xf	A	B												
andi s2 x0 0														
beq s1 s2 exit														
lw s1 0xc(s0)														
xor s1 s1 s2														
lw s1 0xc(s0)														

(d) Suppose we want to execute three instructions on this two-stage pipelined CPU. The first instruction begins executing at the start of Cycle C0, the second begins at the start of Cycle C1, and the third, C2.
Fill in the correct control signals on each clock cycle in order to execute these instructions correctly:

- Any signals set by earlier instructions (before the first) should be set to "E".
- Any signals set by later instructions (after the third) should be set to "L".
- Indicate `Enable` or `Disable` for write enable signals.
- ImmSel should be set as `I, S, SB, U` or `UJ`.
- All other signals should be set as 0, 1, an ALU operation, or X (doesn't matter).
- The list of available ALU operations are `ADD, AND, OR, SRL, SRL, SLT`.

You may assume that there are no structural or data hazards.

```
Program:
    srl t1, t2, t3
    sw t0, 4(a0)
    bltu s0, t2, 44
```

Cycle	Signals							
	PCSel	ImmSel	RegWEn	BrUn	BSel	ASel	ALUSel	MemRW	WBSel
C0									
C1									
C2									
C3									

Problem 5 ***$$$*** **(17 points)**

You are given following RISC-V Code:

```
// a0:  Integer array location
// a1:  End bound of the array
// a2:  Offset to new location in words
// Assume these registers hold the following values
// at the start of the program:
// a0 -> 0x1000, a1 -> 2048, a2 -> 2048

     add t0, x0, x0
     slli t3, a2, 2
loop:
     beq t0, a1, done
     slli t1, t0, 2
     add t1, t1, a0
     lw t2, 0(t1)
     add t1, t1, t3
     sw t2, 0(t1)
     addi t0, t0, 4
     jal x0, loop
done:
     ...
```

Assume there is enough memory allocated for the array such that there are no memory out of bounds issues. Also assume the code is run on a machine with a 32-bit address space. Questions will only involve the code starting from `loop` and only refer to one data cache. This cache uses a LRU replacement policy and is write allocate unless otherwise stated.

(a) Consider an 8 words/block, 512B **direct-mapped** data cache. The cache starts empty and we run the program above to completion.

(i) Calculate the number of tag, index, and offset bits for this cache.

Tag:__________ Index: __________ Offset:__________

(ii) What is the hit rate of this direct-mapped cache?

____

(iii) What types of of cache misses occur? Mark all that apply.

○ Capacity ○ Conflict

○ Compulsory

(iv) Assume the cache is emptied and we re-run the program above to completion, but this time with a cache **block size of 4 words**. What is the hit rate of this new cache?

____

(b) Now consider an 8 words/block, 512B **Two-Way Set Associative** data cache. The cache starts empty and we run the program above to completion.

(i) What is the hit rate of this Two-Way Set Associative cache?

____

(ii) What types of of cache misses occur? Mark all that apply.

- ○ Capacity
- ○ Conflict
- ○ Compulsory

(iii) Assume the cache is emptied and we re-run the program above to completion, but this time with a cache **block size of 4 words**. What is the hit rate of this new cache?

____

(c) Now consider a 8 words/block, 512B **Four-Way Set Associative** data cache. The cache starts empty and we run the program above to completion.

(i) What is the hit rate of this Four-Way Set Associative cache?

____

(ii) Assume the Four-Way Set Associative cache is emptied and we re-run the program above to completion, but this time with a **random replacement policy**. How would the hit rate most likely change compared to part c.i?

- ○ Increase
- ○ Stay the Same
- ○ Decrease

(d) Consider the 8 words/block, 512B **direct-mapped** data cache again. The cache starts empty and we run the program above to completion, **except this time with a2 initialized to 2056**. What is the hit rate of this direct-mapped cache?

____