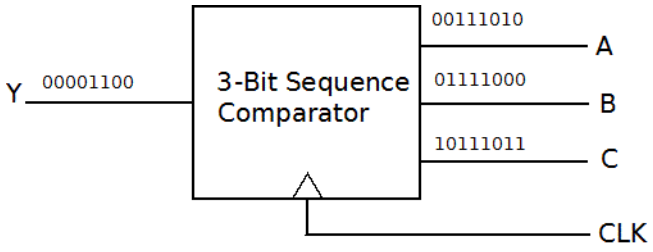
Class	CS147, Sec 01
Homework	II
Due Date	Nov 05, 2018 11:59 PM PST
Instructions	1. There are 5 questions with total 100 points. 2. Please create electronic document with your answer. 3. There is no need to include the question itself. However, you **MUST** include question number and sub-part index if any. Example: 5(b) 4. Please create a PDF document **hw2.pdf** and ***upload that in Canvas*** assignment page by the due date. 5. **Please re-check you submission for any logistic errors (empty file, corrupted PDF, and many more) and re-submit if needed. Once grading is started, any file with logistics errors will be given 0 point.** 6. **NO** handwritten (and scanned) document is accepted. You answer must be typed in (and computer drawn if diagram is asked) using word processing software. 7. **NO LATE SUBMISSION.** 8. **Please explain your answer clearly – just writing the final answer in a word or two is not sufficient in most of the cases.**

1. Design a 3-bit sequence comparator circuit which has 3 bits data inputs (A, B, C), 1 clock signal (CLK) and 1 bit output (Y). This circuit takes stream of bits per clock cycle through these 3 data input pins. Output turns 1 in a clock cycle if latest 3 bit sequence in 3 bit streams matches. Show all the necessary steps to implement this logic circuit and draw final schematic diagram. Use D-F/F as storage if needed. Sample input / output is given in following block diagram of this circuit. [**20pts**]



2. How many basic logic gates a 32-bit ripple carry adder-subtractor circuit will have (as in lecture note 9, page 14). Assume this digital implementation has basic logic gate list as 2-input NAND, 2-input NOR, and NOT. [**10pts**]

3. For a non-pipeline implementation of data and control path in following diagram for a processor implementing CS147DV show the control signal logic values (in compact Hex format) at different phase of the processor executing the following instructions. You need to construct 10 tables similar to Table shown (may use hexadecimal for multi-bus signal, put 0 if don't care). Assumptions on this design are as following. [**30pts**]

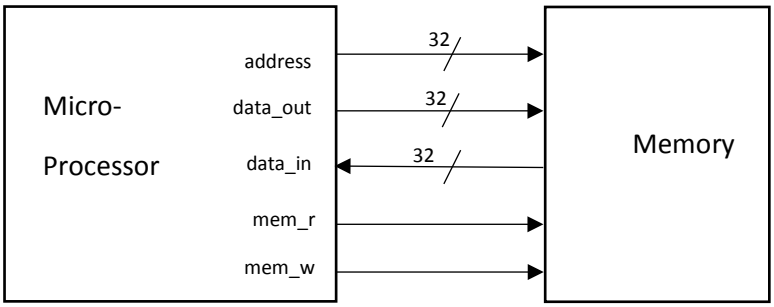
a) Assume that the memory / register file with read=0, write=0 is hold configuration (hold the previous read data) and read=1, write=1 causes electrical isolation of the memory (HiZ). Both of memory and register file reads with read=1, write=1 and writes with read=0, write=1. If memory or register needs to hold previous value, keep it at hold configuration (not in 'read' configuration).
b) ALU assumes operation code as in Lecture 11 notes. For 'alu_oprn' CTRL[25] is MSB and CTRL[22] is LSB.
c) Instruction register is implemented with a transparent latch. This means, as soon as 'ir_load' is turned to 1, input is transferred to output without any waiting for successive clock cycle.

I.	add	r1, r2, r1
II.	srl	r15, r19, 0xa3
III.	jr	r8
IV.	addi	r15, r14, 0x1234
V.	andi	r3, r4, 0x8a5f
VI.	lui	r14, 0xabcd
VII.	beq	r21, r30, 0x123a; // r21 = 0x2; r30 = 0x1
VIII.	sw	r29, r10, 0xa5a5
IX.	jal	0x3A0B12A
X.	push	

Control Signal	CTRL	IF	ID/RF	EXE	MEM	WB
CTRL[0]	pc_load					
CTRL[1]	pc_sel_1					
CTRL[2]	pc_sel_2					
CTRL[3]	pc_sel_3					
CTRL[4]	ir_load					
CTRL[5]	mem_r					
CTRL[6]	mem_w					
CTRL[7]	r1_sel_1					
CTRL[8]	reg_r					
CTRL[9]	reg_w					
CTRL[10]	wa_sel_1					
CTRL[11]	wa_sel_2					
CTRL[12]	wa_sel_3					

CTRL[13]	wd_sel_1					
CTRL[14]	wd_sel_2					
CTRL[15]	wd_sel_3					
CTRL[16]	sp_load					
CTRL[17]	op1_sel_1					
CTRL[18]	op2_sel_1					
CTRL[19]	op2_sel_2					
CTRL[20]	op2_sel_3					
CTRL[21]	op2_sel_4					
CTRL[22:25]	alu_oprn					
CTRL[26]	ma_sel_1					
CTRL[27]	ma_sel_2					
CTRL[28]	md_sel_1					
CTRL Signal Value in Hex		32'hxxxxxxxx	32'hxxxxxxxx	32'hxxxxxxxx	32'hxxxxxxxx	32'hxxxxxxxx

The system looks like following with data path of the microprocessor as in https://sjsu.instructure.com/courses/1265088/modules/items/9704715



4. A computing system X is running on 1.6GHz clock. Another system Y running on 2.5GHz clock. Both of these systems support 4 types of instruction A, B, C, D. The following is the CPI table per instruction type in both the system. To compare performance between these two system one benchmark program has been used which has mix of 25% type A, 25% type B, 30% type C and 20% type D. Compare performance between these two systems (P_Y/P_X) with respect to this benchmark program? [**20pts**]

Instruction Type	CPI of X	CPI of Y
A	4	3
B	2	5
C	1	4
D	3	1

5. Consider the following piece of code in a 5-stage pipeline processor (as discussed in class).
    a. Fill out the data hazard and resolution table. Use <inst#>-<stage> to denote instruction-stage in pipe line. For example, 1-MEM means 'MEM stage for instruction 1'. Mark the resolution methods as FWD (data forward) and STALL (stall). Consider no reordering in this case. [**5pts**]
    b. Fill out data hazard and resolution table after the stall is inserted. [**5pts**]
    c. Write down minimally re-ordered code to avoid stall. Fill out data hazard and resolution after re-ordering. Do not alter the instruction ID numbers in left most columns [**10pts**]

ID	Instruction	Pipeline Stages											
1	lw r1, r20, 0x2056												
2	lw r2, r21, 0xF5C4												
3	add r3, r1, r2												
4	lw r4, r22, 0x0014												
5	addi r8, r4, 0x1A												
6	add r5, r8, r4												

Ans:

a) The data hazard table for original code as following.

Data Hazard (Original)			
STAGE	**DEPENDENCY**	**RESOLUTION**	**FWD-FROM**

b) Data hazard table after the STALL.

Data Hazard (After Stall)			
STAGE	**DEPENDENCY**	**RESOLUTION**	**FWD-FROM**

c) Minimal re-ordered code to avoid STALL would be as following.

ID	Instruction	Pipeline Stages											

The data hazard table would be as following.

Data Hazard (After Reorder)			
STAGE	**DEPENDENCY**	**RESOLUTION**	**FWD-FROM**