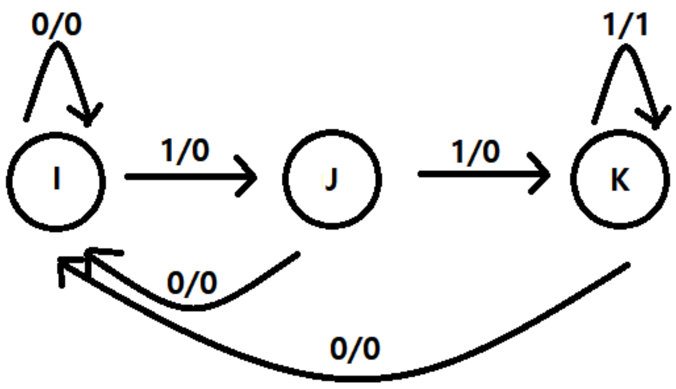
CS147 HW2
Keonwoong Min

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.

ANS)

a) State Diagram



State I Nothing matching
State J 1 bit sequence in 3 bit streams matching
State K 2 bit sequence in 3 bit streams matching

b) State Table

Present State	**Next States**		**Output Y**	
	(A XOR B XOR C)` = 0	(A XOR B XOR C)` = 1	(A XOR B XOR C)` = 0	(A XOR B XOR C)` = 1
I	I	J	0	0
J	I	K	0	0
K	I	K	0	1

c) State Assignment

Present State	**Next States**		**Output Y**	
	(A XOR B XOR C)` = 0	(A XOR B XOR C)` = 1	(A XOR B XOR C)` = 0	(A XOR B XOR C)` = 1
00	00	01	0	0
01	00	11	0	0
11	00	11	0	1

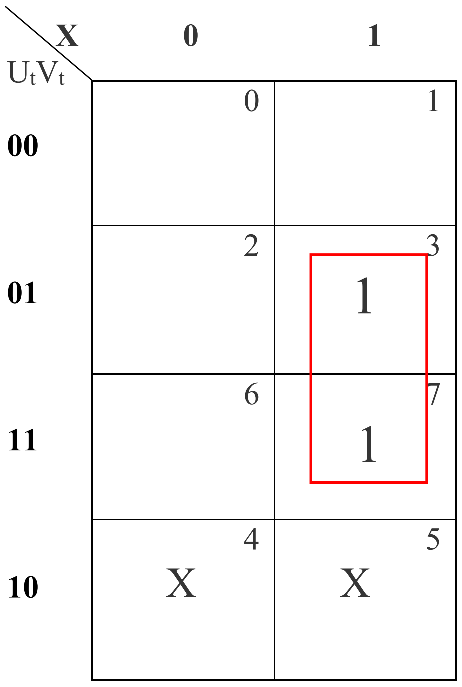
- States are in gray code
  I = 00
  J = 01
  K = 11

d) Truth Table

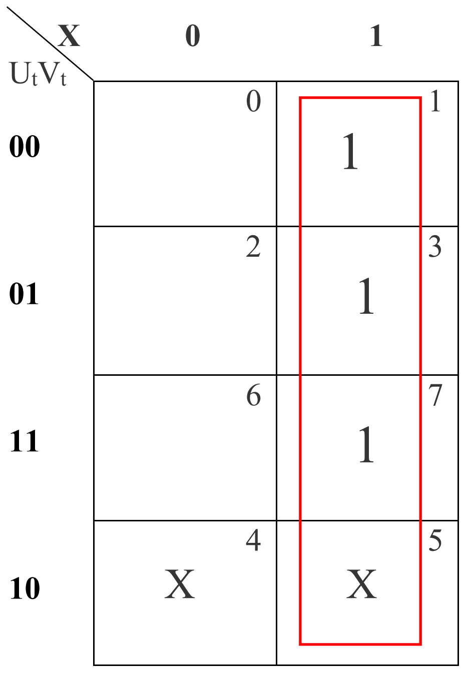
Set X = (A XOR B XOR C)`,

	State Variables at present state		**Primary Input**	**State Variable at next state**		**Primary Output**
	U_t	**V_t**	**X**	**U_{t+1}**	**V_{t+1}**	**Y**
m0	0	0	0	0	0	0
m1	0	0	1	0	**1**	0
m2	0	1	0	0	0	0
m3	0	1	1	**1**	**1**	0
m6	1	1	0	0	0	0
m7	1	1	1	**1**	**1**	**1**
m4	x	x	x	x	x	x
m5	x	x	x	x	x	x

e) Optimization with K-map



$U_{t+1} = D_U(U_t, V_t, X)$
$= \sum m(3,7)$
$= XV_t$



$$V_{t+1} = D_V(U_t, V_t, X)$$
$$= \sum m(1,3,7)$$
$$= X$$

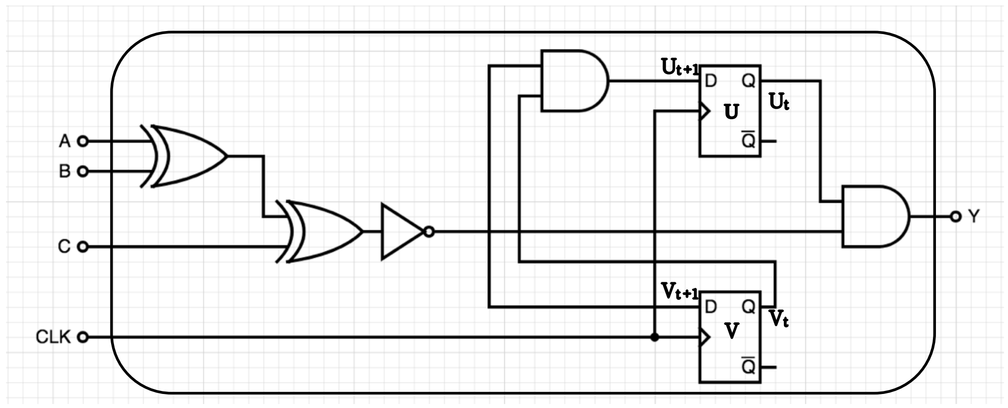
U_tV_t \ X	0	1
00	0	1
01	2	3
11	6	7 1
10	4 X	5 X

$Y = \sum m(7)$
$\quad = XU_t$

<u>State Equation</u>
$U_{t+1} = XV_t$
$V_{t+1} = X$

<u>Output Equation</u>
$Y = XU_t$

f) Technology mapping



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**]

ANS)
32-bit ripple carry adder-subtractor circuit has 32 full adders and one Overflow register.
One full adder contains 2 XOR gates, 2 AND gates and 1 OR gate.
XOR can be replaced with 4 NAND gates, AND gate can be replaced with 2 NAND, and OR gate can be replaced with 1 NOR and 1 NOT gates. It is 14 gates in total per full adder.
Since we have 32 adders, it is 32 * 14 = 448 gates for full adders.
And there are 32 XOR gates between SnA and one 32bit register,
Such that there are 32*4 NAND gates which is 128 NAND gates.
Such that, 32bit ripple carry adder-subtractor circuit has 576 gates.

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	

I. add r1, r2, r1

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**0**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**1**
CTRL[10]	wa_sel_1	0	0	0	0	**1**
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	**1**
CTRL[13]	wd_sel_1	0	0	0	0	**0**
CTRL[14]	wd_sel_2	0	0	0	0	**0**
CTRL[15]	wd_sel_3	0	0	0	0	**1**
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	**0**	**0**	**0**
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	0	0	0
CTRL[21]	op2_sel_4	0	0	**1**	**1**	**1**
CTRL[22:25]	Alu_oprn	0	0	**0x1$_{\text{(hex)}}$**	**0x1$_{\text{(hex)}}$**	**0x1$_{\text{(hex)}}$**
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’h08000020	32’h00000110	32’h00600000	32’h00600000	32’h0060960B

II. srl r15, r19, 0xa3 (0xA Shift Right Logical RESULT = OPRND1 >> OPRND2)

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**0**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**1**
CTRL[10]	wa_sel_1	0	0	0	0	**0**
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	**1**
CTRL[13]	wd_sel_1	0	0	0	0	**0**
CTRL[14]	wd_sel_2	0	0	0	0	**0**
CTRL[15]	wd_sel_3	0	0	0	0	**1**
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	**0**	**0**	**0**
CTRL[18]	op2_sel_1	0	0	**1**	**0**	**0**
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	**1**	**1**	**1**
CTRL[21]	op2_sel_4	0	0	**0**	**0**	**0**
CTRL[22:25]	Alu_oprn	0	0	**0xA$_{\text{(hex)}}$**	**0xA$_{\text{(hex)}}$**	**0xA$_{\text{(hex)}}$**
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’h08000020	32’h00000110	32’h02940000	32’h02900000	32’h0290920B

III. jr r8 (Operation: PC = {6`b0, address})

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**0**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**0**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**0**
CTRL[10]	wa_sel_1	0	0	0	0	0
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	0
CTRL[13]	wd_sel_1	0	0	0	0	0
CTRL[14]	wd_sel_2	0	0	0	0	0
CTRL[15]	wd_sel_3	0	0	0	0	0
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	0	0	0
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	0	0	0
CTRL[21]	op2_sel_4	0	0	0	0	0
CTRL[22:25]	Alu_oprn	0	0	0	0	0
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’h08000020	32’h00000110	32’h00000000	32’h00000000	32’h00000009

IV. addi r15, r14, 0x1234d (Operation: R[rt] = R[rs] (op) SignExtImm)

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**1**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**1**
CTRL[10]	wa_sel_1	0	0	0	0	**1**
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	**1**
CTRL[13]	wd_sel_1	0	0	0	0	**0**
CTRL[14]	wd_sel_2	0	0	0	0	**0**
CTRL[15]	wd_sel_3	0	0	0	0	**1**
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	**0**	**0**	**0**
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	**1**	**1**	**1**
CTRL[20]	op2_sel_3	0	0	**0**	**0**	**0**
CTRL[21]	op2_sel_4	0	0	**0**	**0**	**0**
CTRL[22:25]	Alu_oprn	0	0	**0x1**$_{\text{(hex)}}$	**0x1**$_{\text{(hex)}}$	**0x1**$_{\text{(hex)}}$
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32'h08000020	32'h00000110	32'h00480000	32'h00480000	32'h0048964B

V. andi r3, r4, 0x8a5f (Operation: R[rt] = R[rs] (op) ZeroExtImm)

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**0**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**1**
CTRL[10]	wa_sel_1	0	0	0	0	**1**
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	**1**
CTRL[13]	wd_sel_1	0	0	0	0	**0**
CTRL[14]	wd_sel_2	0	0	0	0	**0**
CTRL[15]	wd_sel_3	0	0	0	0	**1**
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	**0**	**0**	**0**
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	**0**	**0**	**0**
CTRL[20]	op2_sel_3	0	0	**0**	**0**	**0**
CTRL[21]	op2_sel_4	0	0	**0**	**0**	**0**
CTRL[22:25]	Alu_oprn	0	0	**0x5**$_{\text{(hex)}}$	**0x5**$_{\text{(hex)}}$	**0x5**$_{\text{(hex)}}$
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’hx08000020	32’h00000110	32’h01400000	32’h01400000	32’h0140960B

VI. lui r14, 0xabcd   Operation: R[rt] = {imm, 16'b0}

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**0**
CTRL[7]	r1_sel_1	0	0	0	0	0
CTRL[8]	reg_r	**0**	**0**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**1**
CTRL[10]	wa_sel_1	0	0	0	0	**1**
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	**1**
CTRL[13]	wd_sel_1	0	0	0	0	0
CTRL[14]	wd_sel_2	0	0	0	0	**1**
CTRL[15]	wd_sel_3	0	0	0	0	**1**
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	0	0	0
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	0	0	0
CTRL[21]	op2_sel_4	0	0	0	0	0
CTRL[22:25]	Alu_oprn	0	0	0	0	0
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’hx08000020	32’h00000010	32’h00000000	32’h00000000	32’h0000D60B

VII. beq r21, r30, 0x123a // r21 = 0x2; r30 = 0x1
(Operation: PC = PC + 1 + SignExtImm; if R[rs] == R[rt] or R[rs] != R[rt])

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	0
CTRL[2]	pc_sel_2	0	0	0	0	**1**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	**0**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**0**
CTRL[10]	wa_sel_1	0	0	0	0	0
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	0
CTRL[13]	wd_sel_1	0	0	0	0	0
CTRL[14]	wd_sel_2	0	0	0	0	0
CTRL[15]	wd_sel_3	0	0	0	0	0
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	**0**	**0**	**0**
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	0	0	0
CTRL[21]	op2_sel_4	0	0	**1**	**1**	**1**
CTRL[22:25]	Alu_oprn	0	0	**0x7**$_{\text{(hex)}}$	**0x7**$_{\text{(hex)}}$	**0x7**$_{\text{(hex)}}$
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’hx08000020	32’h00000110	32’h01E00000	32’h01E00000	32’h01E0000D

VIII. sw r29, r10, 0xa5a5

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**1**	**0**
CTRL[7]	r1_sel_1	0	**0**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**0**
CTRL[10]	wa_sel_1	0	0	0	0	0
CTRL[11]	wa_sel_2	0	0	0	0	0
CTRL[12]	wa_sel_3	0	0	0	0	0
CTRL[13]	wd_sel_1	0	0	0	0	0
CTRL[14]	wd_sel_2	0	0	0	0	0
CTRL[15]	wd_sel_3	0	0	0	0	0
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	**0**	**0**	0
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	**1**	**1**	0
CTRL[20]	op2_sel_3	0	0	**0**	**0**	0
CTRL[21]	op2_sel_4	0	0	**0**	**0**	0
CTRL[22:25]	Alu_oprn	0	0	**0x1**$_{\textbf{(hex)}}$	**0x1**$_{\textbf{(hex)}}$	0
CTRL[26]	ma_sel_1	0	0	0	**0**	0
CTRL[27]	ma_sel_2	**1**	0	0	**0**	0
CTRL[28]	md_sel_1	0	0	0	**0**	0
CTRL Signal Value in Hex		32’hx08000020	32’h00000110	32’h00480000	32’h00480040	32’h0000000B

IX. jal 0x3a0b12a
(Operation: R[31] = PC + 1; PC = {6'b0, address})

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	**1**	0	0	0	0
CTRL[2]	pc_sel_2	**0**	0	0	0	0
CTRL[3]	pc_sel_3	**1**	0	0	0	**0**
CTRL[4]	ir_load	**0**	**1**	**1**	**1**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**0**	0
CTRL[7]	r1_sel_1	0	0	0	0	**1**
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	0
CTRL[10]	wa_sel_1	0	0	0	0	**1**
CTRL[11]	wa_sel_2	0	**1**	0	0	**0**
CTRL[12]	wa_sel_3	0	**0**	0	0	**1**
CTRL[13]	wd_sel_1	0	**1**	0	0	**0**
CTRL[14]	wd_sel_2	0	**0**	0	0	**1**
CTRL[15]	wd_sel_3	0	**1**	0	0	**0**
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**0**
CTRL[17]	op1_sel_1	0	0	0	0	0
CTRL[18]	op2_sel_1	0	0	0	0	0
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	0	0	0
CTRL[21]	op2_sel_4	0	0	0	0	0
CTRL[22:25]	Alu_oprn	0	0	0	0	0
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’hx0800002A	32’h0000A910	32’h00000010	32’h00000010	32’h00005481

X. push

(M[$sp] = R[0]; $sp = $sp – 1)

Control Signal	**Stage**					
	CTRL	**IF**	**ID/RF**	**EXE**	**MEM**	**WB**
CTRL[0]	pc_load	**0**	**0**	**0**	**0**	**1**
CTRL[1]	pc_sel_1	0	0	0	0	**1**
CTRL[2]	pc_sel_2	0	0	0	0	**0**
CTRL[3]	pc_sel_3	0	0	0	0	**1**
CTRL[4]	ir_load	**0**	**1**	**0**	**0**	**0**
CTRL[5]	mem_r	**1**	**0**	**0**	**0**	**0**
CTRL[6]	mem_w	**0**	**0**	**0**	**1**	**0**
CTRL[7]	r1_sel_1	0	**1**	0	0	0
CTRL[8]	reg_r	**0**	**1**	**0**	**0**	**0**
CTRL[9]	reg_w	**0**	**0**	**0**	**0**	**0**
CTRL[10]	wa_sel_1	0	0	0	0	0
CTRL[11]	wa_sel_2	0	0	0	0	**1**
CTRL[12]	wa_sel_3	0	0	0	0	**0**
CTRL[13]	wd_sel_1	0	0	0	**1**	0
CTRL[14]	wd_sel_2	0	0	0	0	0
CTRL[15]	wd_sel_3	0	0	0	0	0
CTRL[16]	sp_load	**0**	**0**	**0**	**0**	**1**
CTRL[17]	op1_sel_1	0	0	**1**	**1**	**1**
CTRL[18]	op2_sel_1	0	0	**0**	**0**	**0**
CTRL[19]	op2_sel_2	0	0	0	0	0
CTRL[20]	op2_sel_3	0	0	**1**	**1**	**1**
CTRL[21]	op2_sel_4	0	0	**0**	**0**	**0**
CTRL[22:25]	Alu_oprn	0	0	**0x2**$_{\text{(hex)}}$	**0x2**$_{\text{(hex)}}$	**0x2**$_{\text{(hex)}}$
CTRL[26]	ma_sel_1	0	0	0	0	0
CTRL[27]	ma_sel_2	**1**	0	0	0	0
CTRL[28]	md_sel_1	0	0	0	0	0
CTRL Signal Value in Hex		32’hx08000020	32’h00000190	32’h00920000	32’h00922040	32’h0093080B

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

ANS)
i) Since P = 1/E and E = N * T = N/F
ii) E_x = (1/1.6GHz)((4*0.25)+(2*0.25)+(1*0.3)+(3*0.2)) = 2.4/1.6GHz
    E_Y = (1/2.5GHz)((3*0.25)+(5*0.25)+(4*0.3)+(1*0.2)) = 3.4/2.5GHz
iii) $P_Y/P_X = E_X/E_Y = (N_X/N_Y)(F_Y/F_X)$ = (2.4/3.4)(2.5GHz/1.6GHz) = 1.10
So, The system Y is about 1.10 times faster than system X

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**]

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

Instruction		Pipeline Stages											
ID	Statement	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
1	lw r1, r20, 0x2056	IF	ID/RF	EXE	MEM	WB							
2	lw r2, r21, 0xF5C4		IF	ID/RF	EXE	MEM	WB						
3	add r3, r1, r2			IF	ID/RF	EXE	MEM	WB					
4	lw r4, r22, 0x0014				IF	ID/RF	EXE	MEM	WB				
5	addi r8, r4, 0x1A					IF	ID/RF	EXE	MEM	WB			
6	add r5, r8, r4						IF	ID/RF	EXE	MEM	WB		

Data Hazard (Original)			
STAGE	**DEPENDENCY**	**RESOLUTION**	**FWD-FORM**
3-EXE	1-WB	FWD	1-MEM
3-EXE	2-WB	STALL	
5-EXE	4-WB	STALL	
6-EXE	4-WB	FWD	4-MEM
6-EXE	5-WB	FWD	5-EXE

b) Data hazard table after the STALL.
After stalling, it would be like this table

Instruction		**Pipeline Stages**											
ID	**Statement**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**
1	lw r1, r20, 0x2056	IF	ID/RF	EXE	MEM	WB							
2	lw r2, r21, 0xF5C4		IF	ID/RF	EXE	MEM	WB						
3	add r3, r1, r2			IF	ID/RF	**bubble**	EXE	MEM	WB				
4	lw r4, r22, 0x0014				IF	**bubble**	ID/RF	EXE	MEM	WB			
5	addi r8, r4, 0x1A					**bubble**	IF	ID/RF	**bubble**	EXE	MEM	WB	
6	add r5, r8, r4							IF	**bubble**	ID/RF	EXE	MEM	WB

Data Hazard (After Stall)			
STAGE	**DEPENDENCY**	**RESOLUTION**	**FWD-FORM**
3-EXE	1-WB	FWD	1-MEM
3-EXE	2-WB	FWD	2-MEM
5-EXE	4-WB	FWD	4-MEM
6-EXE	4-WB	FWD	4-MEM
6-EXE	5-WB	FWD	5-EXE

c) Minimal re-ordered code to avoid STALL would be as following.
We change the Stage 3, add r3, r1, r2 and Stage 4, lw r4, r22, 0x0014 and we get

Instruction		**Pipeline Stages**											
ID	**Statement**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**
1	lw r1, r20, 0x2056	IF	ID/RF	EXE	MEM	WB							
2	lw r2, r21, 0xF5C4		IF	ID/RF	EXE	MEM	WB						
4	lw r4, r22, 0x0014			IF	ID/RF	EXE	MEM	WB					
3	add r3, r1, r2				IF	ID/RF	EXE	MEM	WB				
5	addi r8, r4, 0x1A					IF	ID/RF	EXE	MEM	WB			
6	add r5, r8, r4						IF	ID/RF	EXE	MEM	WB		

Data Hazard (After Reorder)			
STAGE	**DEPENDENCY**	**RESOLUTION**	**FWD-FORM**
3-EXE	1-WB	FWD	1-MEM

3-EXE	2-WB	FWD	2-MEM
5-EXE	4-WB	FWD	4-MEM
6-EXE	4-WB	FWD	4-MEM
6-EXE	5-WB	FWD	5-EXE