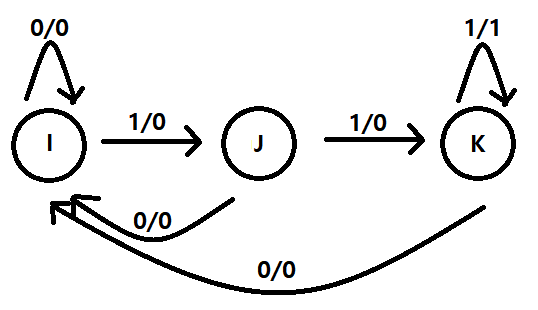
CS147 HW2

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**  **J**  **K** | I  I  I | J  K  K | 0  0  0 | 0  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**  **01**  **11** | 00  00  00 | 01  11  11 | 0  0  0 | 0  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** |
|  | **Ut** | **Vt** | **X** | **Ut+1** | **Vt+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

**X 0 1**

UtVt

|  |  |
| --- | --- |
| 0 | 1 |
| 2 | 3  1 |
| 6 | 7  1 |
| 4  X | 5  X |

**00**

**01**

**11**

**10**

Ut+1 = DU(Ut,Vt,X)

= ∑m(3,7)

= XVt

**X 0 1**

UtVt

|  |  |
| --- | --- |
| 0 | 1  1 |
| 2 | 3  1 |
| 6 | 7  1 |
| 4  X | 5  X |

**00**

**01**

**11**

**10**

Vt+1 = DV(Ut,Vt,X)

= ∑m(1,3,7)

= X

**X 0 1**

UtVt

|  |  |
| --- | --- |
| 0 | 1 |
| 2 | 3 |
| 6 | 7  1 |
| 4  X | 5  X |

**00**

**01**

**11**

**10**

Y = ∑m(7)

= XUt

State Equation

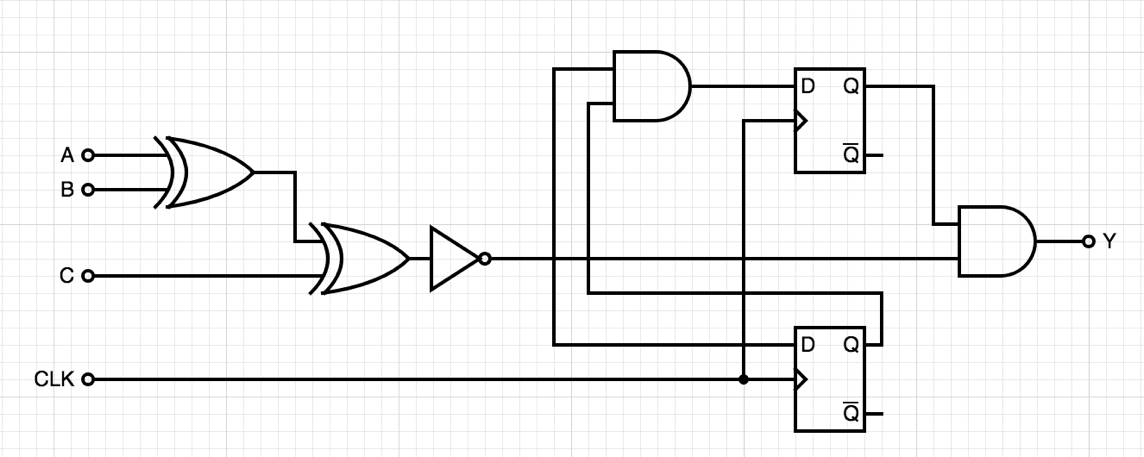
Ut+1 = XVt

Vt+1 = X

Output Equation

Y = XUt

f) Technology mapping



Ut+1

Vt+1

Vt

V

Ut

U

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.

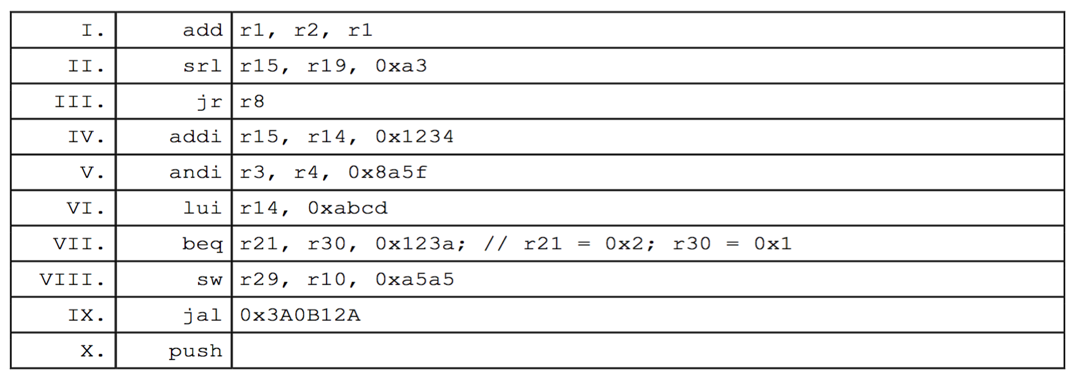
Such that, 32bit ripple carry adder-subtractor circuit has 576 gates.

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.



1. 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(hex)** | **0x1(hex)** | **0x1(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’h006000 00 | 32’h006096 0B |

1. 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(hex)** | **0xA(hex)** | **0xA(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’h029000 00 | 32’h029092 0B |

1. 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’h000000 00 | 32’h000000 09 |

1. 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(hex)** | **0x1(hex)** | **0x1(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’h004800 00 | 32’h004896 4B |

1. 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(hex)** | **0x5(hex)** | **0x5(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’h01400000 | 32’h01400000 | 32’h0140960B |

1. 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’h08000020 | 32’h00000010 | 32’h00000000 | 32’h000000 00 | 32’h0000D60B |

1. beq r21, r30, 0x123a // r21 = 0x2; r30 = 0x1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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(hex)** | **0x7(hex)** | **0x7(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’h01E00000 | 32’h01E00000 | 32’h01E0000D |

(Operation: PC = PC + 1 + SignExtImm; if R[rs] == R[rt] or R[rs] != R[rt] )

1. 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(hex)** | **0x1(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’h08000020 | 32’h00000110 | 32’h00480000 | 32’h004800 40 | 32’h000000 0B |

1. jal 0x3a0b12a

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

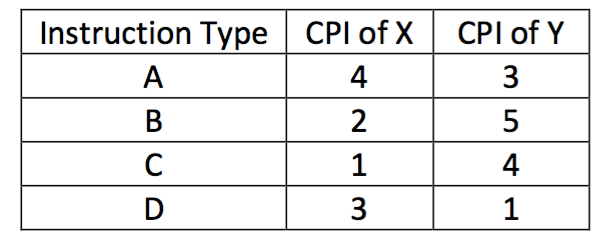
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Control Signal** | **Stage** | | | | | |
| **CTRL** | **IF** | **ID/RF** | **EXE** | **MEM** | **WB** |
| CTRL[0] | pc\_load | **0** | **0** | **0** | **0** | **1** |
| CTRL[1] | pc\_sel\_1 | 0 | 0 | 0 | 0 | 0 |
| CTRL[2] | pc\_sel\_2 | 0 | 0 | 0 | 0 | 0 |
| CTRL[3] | pc\_sel\_3 | 0 | 0 | 0 | 0 | **0** |
| CTRL[4] | ir\_load | **0** | **0** | **0** | **0** | **0** |
| CTRL[5] | mem\_r | **0** | **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 | 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 | 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 | 0 | 0 | 0 | 0 | 0 |
| CTRL[28] | md\_sel\_1 | 0 | 0 | 0 | 0 | 0 |
| CTRL Signal Value in Hex | | 32’h00000000 | 32’h00000000 | 32’h00000000 | 32’h00000000 | 32’h00000A01 |

1. 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(hex)** | **0x2(hex)** | **0x2(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’h00000190 | 32’h00920000 | 32’h009220 40 | 32’h009308 0B |

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 (PY/PX) with respect to this benchmark program? [**20pts**]



ANS)

i) Since P = 1/E and E = N \* T = N/F

ii) Ex = (1/1.6GHz)( (4\*0.25)+(2\*0.25)+(1\*0.3)+(3\*0.2) ) = 2.4/1.6GHz

EY = (1/2.5GHz)( (3\*0.25)+(5\*0.25)+(4\*0.3)+(1\*0.2) ) = 3.4/2.5GHz

iii) PY/PX = EX/EY = (NX/NY)(FY/FX) = (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).

1. 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**]
2. Fill out data hazard and resolution table after the stall is inserted. [**5pts**]
3. 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 |