EAST WEST UNIVERSITY

Department of Computer Science and Engineering

Fall 2024

CSE360 Computer Architecture

Assignment

Section: 03

Submitted By,

Name: Khalid Mahmud Joy

ID: 2022-3-60-159

Chapter - 4

4.1/

Reg write	Mem Read	ALU MUX	Mem write	ALU OP	Reg MUX	Branch
0	0	1 (Imm)	1	Add	X	0

ALU MUX is the control-signal that control's MUX at the ALU input, 0(reg) selects the output of the register file, and 1 (Imm) select the immediate from the instruction word as the second input to the ALU. Reg MUX is the control signal that controls the MUX at the Data input to the register file. 0 (ALU) selects the output of the ALU and 1 (Mem) selects the output of memory.

4.2/

This instruction was instruction memory, both register read points, the ALU to add Rd and Rs together, data memory and write port in register.
None, this instruction can be implemented using existing blocks.

4.3/ (a) The latency of this path is 400 ps + 200 ps + 30 ps + 120 ps + 350 ps + 30 ps = 1130 ps.

1430 ps (1130 ps + 300 ps, ALU is on the critical path)

(ii)

we need 5% fewer cycles for a program but cycle time is 1430 instead of 1130.

So, we have a speed up of $(1/0.95) \times (1130/1430)$

$= 0.83$

which means we actually have a slow down.

4.4 The critical path of this instruction is through the instruction memory, sign-extend and shift left-2 to get offset, Add unit to compile the new pc and mux to select that value instead of pc+4. we have -

200 ps + 15 ps + 10 ps + 70 ps + 20 ps = 315 ps.

4.4.3 The path through pcsrc, is longer for these latencies,

200 ps + 90 ps + 20 ps + 90 ps + 20 ps = 420 ps

4.5 The data memory is used by LW and SW instruction so the answer is: (25% + 10%) = 35%

The sign extend circuit is actually compiling a result ~~is~~ in every cycle, but its output is ignored for ADD and NOT instructions, so the answer is,

20% + 25% + 25% + 10% = 80%

4.6

The test for this fault, we need an instruction whose jump is 1. so it has to be the jump instruction. however for the jump instruction the RegDst signal is "don't care". because it doesn't writte to any registers. so the implemenation may or may not allow us to set RegDst to 0. so, we can not meliably test for this fault.

4.7

ALU OP[1-0]	Instruction [5-0]
00	010100

Nr Reg MUX	ALU MUX	MEM ALU MUX	Branch MUX	Jump MUX
2 or 0 (RegDst)	20	X	PC+4	PC+4

4.8 we already computed clock cycle times for piplined and single cycle organizations and the multi-cycle organization has the the same clock rate time as the pipelined organization, we will compute execution time relative to the pipelined org. In the single cycle every instruction take one clock cycle,

Multicycle execution time is x times piplined execution time, where	Single-cycle execution time is x times pipelined execution time when,
0.20×5 + 0.80×4 = 4.20	1250 ps / 350 ps = 3.87

4.9

Instruction sequences	Dependencies
I_1 : OR R_1, R_2, R_3 I_2 : OR R_2, R_1, R_4 I_3 : OR, R_1, R_1, R_2	RAW on R_1 from I_1 to I_2 RAW on R_2 from I_2 to I_3 WAR on R_2 from I_1 to I_2 WAR on R_1 from I_2 to I_3 WAR on R_1 from I_1 to I_3

4.9.2

OR R_1, R_2, R_3 NOP NOP OR R_2, R_1, R_4 NOP NOP OR R_1, R_1, R_2	Delay I_2 to avoid RAW hazard on R_1 from I_1 Delay I_3 to avoid RAW hazard on R_2 from I_2

4.10

Instruction executed	cycle with 5 stages	cycle with 4 stages	speedup
5	4+5 = 9	3+5 = 8	9/8 = 1.13

cycle with 5 stages	cycle with 4 stages	speedup
200 ps (IF)	210 ps (MEM + 200ps)	(9×200)/(8×210) = 1.07

4.11

LW R1,0(R1)
LW R1, 0(R1)
BEQ · R1,R0 , LOOP
LW R1, 0(R1)
AND R1,R1, R2
LW R1,0(R1)
LW R1,0(R1)
BEQ R1,R0,LOOP
	WB
	EX MEM WB
	ID *** EX · MEM WB
	IF *** ID EX MEM WB
	IF ID *** EX MEM

4.12 Dependecies to the 1st next instruction result in 2 stall cycles, and the stall is also 2 cycles if the depedencies is to both 1st and 2nd next instruction result in one stall cycle. we have -

$$CPI = 1 + 0.35 \times 2 + 0.15 \times 1 = 1.85$$

Stall cycles = 46% (0.85/1.85)

4.18.1 To translate the given c code loop directly into MIPS assembly code, we will the provided register assignment.

i is in R5
j is in R8
a is in R1
b is in R2

```
Loop inc
for (i=0; i!=j; i+=2)
  b[i] = a[i] - a[i+1].
```

```
Loop start:
beq R5, R6, loop-end  # if i==j, exit loop
sll R10, R5, 2        # R10 = i*4 (shift left logical by 2)
add R10, R1, R10      # R10 = a+i*4
lw  R11, 0(R10)       # R11 = a[i]
lw  R12, 4(R10)       # R12 = a[i+1]
sub R11, R11, R12
add R10, R2, R10      # R10 = b+i*4
sw  R11, 0(R10)       # b[i] = a[i] - a[i+1]
addi R5, R5, 2        # i = i+2
j loop-start
loop-end;
```