Chapter-4

4.1

RegWrite	Mem Read	ALUMux	Mem Write	ALUop	RegMux	Branch
0	0	1 (Imm)	1	ADD	X	0

ALUMux is the control signal that controls 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.

RegMUX 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 uses instruction memory, both register read ports, the ALU to add Rd and Rs together, data memory and write port in Registers.

None, This instruction can be implemented using existing blocks.

4.3

1) The latency of this path is 400ps + 200ps + 30ps + 120ps + 350ps + 30ps

= 1130 ps

1430 ps (1130ps + 300ps, 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 for this instruction is through the instruction memory, Sign-extend and Shift left-2. to get the offset, Add unit to compute 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 instructions so the answer is: (25% + 10%)

= 35%

The sign-extend circuit is actually computing a result 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 tes to 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 write to any registers, so the implementation may or may not allow us to set RegDst to 0. So we can test for this fault. As a result, we can not reliably test for this faut.

4.7

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

Wr Reg Mux	ALU MUX	MEM/ ALU MUX	Branch MUX	Jump MUX
2 or 0 (RegDst)	2 0	x	PC+4	PC+4

4.8

We already computed clock cycle times for pipelined and single cycle organizations. and the multi-cycle organization has the same clock rate time as the pipelined organization. We will compute execution time relative to the pipelined organization.

In single-cycle every instruction take one (long) clock cycle.

Multi-cycle execution time is x times pipelined execution time, where x is:	Single-cycle execution time is x times pipe-lined execution thime, wher
0·20×5 + 0·80× 4 = 4·20	1250 PS / 350 PS = 3·57

4.9

Instruction sequence	Dependences
I1: OR R1, R2, R3	RAW on R1 from I1 to I2
I2: OR R2, R1, R4	RAW on R2 from I2 to I3
I3: OR R1, R1, R2	WAR on R2 from I1 to I2
	WAR on R1 from I2 to I3
	WAR on R1 from I1 to I3

4.9.2

OR R1, R2, R3 NOP NOP	Delay I2 to avoid RAW hazard on R1 from I1
OR R2, R1, R4 NOP NOP OR R1, R1, R2	Delay I3 to avoid RAW hazard on R2 from I2

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 time with 5 stages	Cycle time with 4 stages	Speedup
200 ps (IF)	210 ps (MEM + 20ps)	(9 × 200) / (8 × 210) = 1.07

4.11

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

4.12

Dependences to the 1st next instruction result in 2 stall cycles and the stall is also 2 cycles if the dependence is to both 1st and 2nd next instruction. Dependences to the 2nd next instruction result in one stall cycle.

We have:

CPI	Stall cycles
1+0.35×2 +0.15×1 =1.85	46% (0.85/1.85)