Solution to HW 3 – CpE 313 – Fall 2004

Solution to Question 1

There are two acceptable solutions. The first one transfers the minimal amount of information from one stage to the next, and is given below.

Pipeline register	***Contents***
IF/ID	IF/ID.NPC, IF/ID.IR
ID/EX	ID/EX.NPC, ID/EX.IR , ID/EX.A, ID/EX.B, ID/EX.Imm
EX/MEM	EX/MEM.Cond, EX/MEM.ALUOutput, EX/MEM.B, EX/MEM.IR
MEM/WB	MEM/WB.LMD, MEM/WB.ALUOutput, MEM/WB.IR

The second acceptable solution transfers the entire context of the instruction from one stage to the next, and is given below. This solution is acceptable because that is what we did in the lectures. The previous solution is leaner and is preferable in reality.

Pipeline register	***Contents***
IF/ID	IF/ID.NPC, IF/ID.IR
ID/EX	ID/EX.NPC, ID/EX.IR , ID/EX.A, ID/EX.B, ID/EX.Imm
EX/MEM	EX/MEM.NPC, EX/MEM.IR , EX/MEM.A, EX/MEM.B, EX/MEM.Imm, EX/MEM.Cond, EX/MEM.ALUOutput
MEM/WB	MEM/WB.NPC, MEM/WB.IR , MEM/WB.A, MEM/WB.B, MEM/WB.Imm, MEM/WB.Cond, MEM/WB.ALUOutput, MEM/WB.LMD

Solution to Question 2

a. For unpipelined case,
   Total number of cycles = cycles per instruction * number of instructions
   = 5 * 12 = 60

b. Because there is no forwarding between the pipeline stages, there are two cycles of stall between the data dependant instructions. In the table below, a "+2" appears for each instruction that is data dependent on the previous instruction. The bold font italicized "***+2***" appears for each instruction that is data dependent on a load instruction.

Instructions	Cycle
add r1, r2, r3	5
sub r2, r3, r5	6
lw r3, 20(r4)	7
add r4, r3, r5	***8+2***

lw r5, 20(r6)	11
add r6, r5, r6	***12+2***
sub r5, r6, r2	15+2
sub r2, r1, r3	18
and r12, r2, r5	19+2
or r13, r6, r12	22+2
add r14, r2, r12	25
sw 100(r2), r15	26

The total number of clock cycles for these instructions is 26. Please resist the temptation to add one cycle of stall for “add r14, r2, r12” instruction. Although it is true that a source register, r12, of this instruction is the destination register of the previous-to-previous instruction (“and r12, r2, r5”), but the previous instruction (“or” … …) has already waited two cycles for r12. There is no need to add any further stall cycles for “add r14, r2, r12.”

c. Because there is forwarding between the pipeline stages, all ALU-to-ALU data hazards will disappear. However, for load-to-ALU data hazards, there is still one cycle of stall left.

Instructions	Cycle
add r1, r2, r3	5
sub r2, r3, r5	6
lw r3, 20(r4)	7
add r4, r3, r5	8+1
lw r5, 20(r6)	10
add r6, r5, r6	11+1
sub r5, r6, r2	13
sub r2, r1, r3	14
and r12, r2, r5	15
or r13, r6, r12	16
add r14, r2, r12	17
sw 100(r2), r15	18

The total number of clock cycles for these instructions is 18.

d. There is no room for any further optimization. Do not fall prey to swapping the instruction "add r4, r3, r5" and "lw r5, 20(r6)." If you swap, the machine will stall after issuing the lw instruction because, as far as the machine is concerned, the add instruction needs “the r5 that load instruction yet has to produce.”