Spring 2000 CDA 4101 Homework 3 Solutions

6.2

6.3
The original code:
loop:  lw $2, 100($3)
      addi $3,$3, 4
      beq $3,$4, loop

Neither the lw or the addi can move into the delay slot as given due to a chain of dependences. First we must modify some of the addressing.

The intermediate code:

loop:  addi $3,$3, 4
       lw $2, 96($3)
       beq $3,$4, loop

We swap the lw and addi. However, the value of $3 is now 4 more than it was when the lw used it before. We can compensate by changing the constant in the instruction to 96. At this point both the beq and the lw read the same value of $3 so the lw can be moved into the delay slot.

The final code:

loop:  addi $3,$3, 4
       beq $3,$4, loop
       lw $2, 96($3)

6.4

statement 1  add $2, $5, $4
statement 2  add $4, $2, $5
statement 3  sw $5, 100($2)
statement 4  add $3, $2, $4

1. Statement 2 consumes the value of $2 produced in statement 1.
2. Statement 3 consumes the value of $2 produced in statement 1.
5. Forwarding handles all of these dependences without stalling.
6.5

Note that figure 6.25 does not contain the control lines.

1. IF/ID pipeline register
   - $PC + 4$, 32 bits
   - instruction, 32 bits
   - total 64 bits

2. ID/EX pipeline register
   - $PC + 4$, 32 bits
   - read data 1, 32 bits
   - read data 2, 32 bits
   - sign-extended offset, 32 bits
   - possible destination register number, 5 bits
   - a second possible destination register number, 5 bits
   - total 138 bits

3. EX/MEM pipeline register
   - branch target address, 32 bits
   - Zero signal, 1 bit
   - ALU Result, 32 bits
   - Read data 2, 32 bits
   - destination register number, 5 bits
   - total 102 bits

4. MEM/WB pipeline register
   - data from memory, 32 bits
   - ALU Result, 32 bits
   - destination register number, 5 bits
   - total 69 bits
6.12
The forwarding unit is combinational logic that is active during all cycles, i.e., it is always doing something. Specifically, at any cycle it is considering the information available on the instructions currently in the MEM and WB stages of the pipeline. If the instructions write registers then the destinations are compared against the source registers of the instruction currently in the EX stage of the pipeline. So for this particular instruction sequence, the forwarding using is comparing $8 = 4?$, $8 = 1?$, $9 = 4?$, $9 = 1?$. No need for forwarding is detected and the forwarding unit sets the control lines accordingly.

6.13
Similar to the previous answer the hazard unit is combinational logic and therefore is active during all cycles. Specifically, it checks the instruction in the EX stage. If it is a lw then a stall may be required. If the instruction in the ID phase during the same cycle reads the register that the lw writes a stall must be generated. In this case during the fifth cycle we have the the add $10,11,12$ in the ID stage so the hazard unit would be comparing $11$ and $12$ to the potential destination register of a lw in the EX phase. (For this instruction sequence a lw is not in the EX phase during the fifth cycle.)
6.15

add $5, $6, $7

lw $6, 100($7)

sub $7, $6, $8

add $5, $6, $7

lw $6, 100($7)

sub $7, $6, $8

8 cycles required due to stall from load-use