1. How many cycles does it take to fill the pipeline? (Notice that DLXview labels the cycles beginning with zero. To avoid ambiguity, fill in the blank: "In DLXview, the first cycle during which all the pipeline stages are busy is labeled **4**.")
2. How many cycles does it take for the computer to execute the first instruction completely?  **It is out of the pipeline in stage 5**
3. What pipeline stage is the instruction and r6, r7, r8 in during cycle number 5 (i.e., the cycle labeled 5)? **It is in the MEM stage**
4. During which cycle does the the processor begin computing the instruction and r12, r13, r14? (Make sure your answer clearly states "the xth cycle" or "the cycle labeled x").

**The 4th cycle**

1. During which cycle does the the processor finish computing the instruction and r12, r13, r14? (Make sure your answer clearly states "the xth cycle" or "the cycle labeled x"). **The processor finishes computing the instruction at cycle 9 when it leavesthe pipeline**
2. Which registers are being read during cycle 5 (the cycle labeled 5)?

**Registers R13 and R14 are being read**

1. Which registers are being written during cycle 5 (the cycle labeled 5)?

**Register R3 is being written**

1. Where does input to the main ALU come from during cycle 2 (the cycle labeled 2)?

**r1,0x0(r2) is coming from pipeline labeled ID/EX**

1. Where does input to the main ALU come from during cycle 3 (the cycle labeled 3)?

**R4,r5 is coming from pipeline ID/EX through the mux into the ALU**

1. What is the purpose of all of the **nops** at the end of the sample program?

**It is a null instruction pushed through the processor to prevent hazards.**

1. How does the DLX pipeline architecture resolve this conflict?

**The DLX solves this conflict by holding the values in registers while the other instructions are using hardware. The clock prevents values from moving forward until the hardware is complete.**

1. What purpose does it serve?

**Used for branch instructions**

1. Identify all pairs of instructions that have a data dependency. In particular identify all pairs of instructions (not necessarily adjacent) where the result of the second instruction depends directly on the result of the first.

**and r6,r3,r7 depends on sub r3,r4,r5**

**or r8,r9,r3 depends on sub r3,r4,r5**

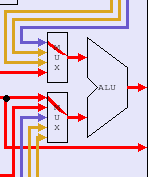
1. Describe in your own words how the DLX hardware addresses this dependency problem. Your answer should be precise enough to convince me that you understand the mechanism used.

**DLX stores values in pipeline registers. Then the instruction will look in the pipeline register file rather than the basic register file.**

1. Trace through the progress of the fourth instruction in a manner similar to that used in the section above ("**A simple pipelined program**"). There is a "visual bug" or discrepancy that you will notice as you trace through the execution of this program. Identify it.

**The bottom mux directly before the ALU is not showing a connection wire.**

1. Step to cycle 4. Use KSnapshot (or another tool of your choice) to capture the main DLXview window. Print the snapshot and highlight the set of wires that shows how the result of sub r3, r4, r5 is routed directly to the main ALU



1. read, describe how you would properly coordinate the reading and writing of the registers.

1. Why must the **add** operation be delayed one cycle? Your answer should consider timing issues and functional units. Be sure to explain why forwarding cannot solve the problem.
2. Describe how hardware can *detect* a load data hazard.
3. Why is the branch delay slot necessary? In other words, what would go wrong if we removed the nop?
4. Suppose you had a smarter compiler. Explain what it could put in the branch delay slot instead of a nop.
5. Write an optimized assembly program incorporating your code adjustments. Your solution must still contain a loop.
6. How many cycles does the original program take to assign the final result to register R6? (In other words, during which cycle is the instruction add r6, r4, r5 in the "write-back" phase?)