6.1 If the time for an ALU operation is 4 ns, the cycle time would be 10 ns in the single-cycle implementation. The cycle time for the pipelined implementation would be 4 ns and the speed-up obtained from pipelining the single cycle implementation would only be $10/4 = 2.5$.

6.2 The diagram should show three instructions in the pipeline. A forwarding path is needed from the second instruction to the third instruction because of the dependency involving $s3$.

6.3 Obviously either the load or the addi must occupy the branch delay slot. We can't just put the addi into the slot because the branch instruction needs to compare $s3$ with register $s4$ and the addi instruction changes $s3$. In order to move the load into the branch delay slot, we must realize that $s3$ will have changed. If you like, you can think of this as a two-step transformation. First, we rewrite the code as follows:

```assembly
Loop: addi $s3, $s3, 4
lw $s2, 96($s3)
beq $s3, $s4, Loop
```

Then we can move the load into the branch delay slot:

```assembly
Loop: addi $s3, $s3, 4
beq $s3, $s4, Loop
lw $s2, 96($s3)   # branch delay slot
```

6.4 The second instruction is dependent upon the first ($s2$). The third instruction is dependent upon the first ($s3$). The fourth instruction is dependent upon the first ($s2$) and second ($s4$). All of these dependencies will be resolved via forwarding.

6.11 In the fifth cycle of execution, register $s4$ will be written and registers $s11$ and $s12$ will be read.

6.12 The forwarding unit is seeing if it needs to forward. It is looking at the instructions in the fourth and fifth stages and checking to see whether they intend to write to the register file and whether the register written is being used as an ALU input. Thus, it is comparing $8 = 47.8 = 179 = 179 = 17$.

6.13 The hazard detection unit is checking to see whether the instruction in the ALU stage is a lw instruction and whether the instruction in the ID stage is reading the register that the lw will be writing. If it is, it needs to stall. If there is a lw instruction, it checks to see whether the destination is register 11 or 12 (or registers being read).

6.14 $5 - 99 \times 2 = 203$ cycles to execute the instructions, CPI = 2.03.

6.15 It will take 8 cycles to execute the code, one of which is a bubble needed because of the dependency involving the subtract instruction and the load.