Homework #2
Assigned: Wednesday, January 22. Due: Wednesday, January 29.

IMPORTANT POLICY CHANGE:
All homeworks are due at the beginning of class. Please bring your homework to class and turn it into the TA before the class starts.

Assignment

(1) We want to add the instruction \textit{jal} (see pages 132–133 of H&P) to the single-cycle datapath described in chapter 5. Add any necessary datapaths and control signals to the single-cycle datapath of Figure 5.19 (page 360) and show the necessary additions to Figure 5.20 (page 361).

(2) In estimating the performance of the single-cycle implementation, we assumed that only the major functional units had any delay (i.e., the delay of the multiplexors, control unit, PC access, sign extension unit, and wires was considered to be negligible). Assume that we change the delays specified on page 373 such that we use a different type of adder for simple addition:

- ALU: 3ns
- adder for PC + 4: X ns
- adder for branch address computation: Y ns

(a) What is the cycle time if X = 1 and Y = 2?

(b) What is the cycle time if X = 2 and Y = 2?

(c) What is the cycle time if X = 0.5 and Y = 1.5?

(3) If the time for an ALU operation were actually 2.5 ns instead of 2 ns (as described in Figure 6.2 on page 439), how would this affect the speedup obtained from pipelining the single-cycle implementation?

(4) Using the pipeline from Figure 6.30 (page 470), write down \textit{all control signals} for every clock cycle for the instructions:

\begin{verbatim}
add $1,$2,$3
sw $1,0($4)
addi $4,$4,-4
beq $1,$5,80
\end{verbatim}

In this code, assume that the branch is taken (i.e., control transfers to the indicated PC-relative target address). Make a table like we did in class: Columns are the control signals and rows are the cycles. Your table should include the control signals: RegDst, RegWrite, ALUSrc, PCSrc, Mem-
Read, MemWrite, MemtoReg, Branch, and ALUOp. See pages 355 and 468 for a description of each control signal. For a cycle where a particular pipeline stage is empty, treat it like a nop instruction is in that stage.

(5) Consider the following idea: Let's modify the instruction set architecture and remove the ability to specify an offset for memory access instructions. Specifically, all load-store instructions with nonzero offsets would become pseudoinstructions and would be implemented using two instructions. For example:

```
addi $at,$t1,104
sw $t0,$at
```

What changes would you make to the single-cycle datapath and control if this simplified architecture were to be used?