CS320 Fall 2014
Homework#5
Quiz in Lecture Thursday October 30, 2014
There are no make-up quizzes!

Problem 1: **Datapath timing**

Assuming the following timings for the components of the datapath. All other components have no delay.

- Data/Instruction Memory Read: 200ps
- Data Memory Write: 100ps
- Register File Read: 100ps
- Register File Write: 50ps
- Adders: 80ps
- ALUs: 200ps
- Logical Shifter/Sign Extension: 30ps
- Multiplexors and other gates: 10ps

a. Calculate the delay for each single cycle instruction type (R-type, lw, sw, beq, j).
b. Calculate the delay for each clock cycle of the finite state machine (each stage of each instruction, i.e. each bubble).
c. What is the minimum clock cycle period at which the multi-cycle datapath can operate properly? What is the length of each instruction type in ps (R-type, lw, sw, beq, j)?

Problem 2: **MIPS Execution**

Based on the timings from Question 1, examine the following MIPS program P.

```
        add  $t0, $s4, $s3
        and  $t2, $t0, $s1
        add  $t1, $t1, $t1
        add  $t2, $t2, $t2
        lw   $t3, OPTION
        beq  $s0, $t3, PARSE_WRITE_SP_LE
        and  $t0, $t1, 0xff
        and  $t1, $t1, 0xff00
        or   $t1, $t1, $t0
        and  $t0, $t2, 0xff
        and  $t2, $t2, 0xff00
        or   $t2, $t2, $t0
PARSE_WRITE_SP_LE:
        sw $t1, 0($a1)
        lw $v0, WRITE_FILE
        sw $t2, 0($a1)
```

a. In the multicycle datapath, in which clock cycle is the `lw $t3, OPTION` instruction fetched?
b. In the multicycle datapath, how many cycles does it take Program P to execute, assuming the branch is not taken?
c. Which values (eg. MEM[$s0], branch address, Instruction[15:0], etc) are stored in the A, B, ALUOut, and MDR registers during the execution cycle of the following instruction: `sw $t1, 0($a1)`

Problem 3: **Multi-cycle datapath**

In class we covered the MIPS multi-cycle implementation also which operates for the same basic subset of MIPS instructions as in the single-cycle basic implementation. In this problem you will introduce functionality for additional MIPS instructions. Use this datapath and finite state machine to specify your changes. In all cases, try to find a solution that minimizes the number of clock cycles required for the new instruction. Explicitly state how many cycles it takes to execute the new instruction in your modified finite state machine. Use the Multi-cycle handout posted on PIAZZA.

Modify the datapath and the finite state machine to:

a. Implement 'addi' instruction
   Reg[rt] ← Reg[rs] + immed[15:0];

register rs is added to immediate value in the instruction and stored in rt

b. Implement 'jal' instruction.
   Reg[31] ← PC+4 # $ra register stores the return address
   PC ← PC+4[31:28], (Instruction[25:0] << 2)
   # PC is the top 4 bits of the PC+4 value concatenated with the lowest 26 bits of the jump address
   # shifted the left by 2 bits

c. Implement a new 'wai' instruction. The instruction, `wai rt`, stands for "where am i" and puts the instruction address into a register specified by the rt field. (Immediate format)
   Reg[rt] ← PC # $ra register stores the return address

d. Implement 'bne'
   if(Reg[rs] != Reg[rt]) PC ← PC+4 + Instruction[15:0] << 2

e. Implement 'beqi' (Assume the rs field is a 5-bit 2's complement value)
   if(Reg[rt] == 2's complement value in rs field)
      PC ← PC+4 + Instruction[15:0] << 2

f. Implement 'bgtz' (Assume the rt register is set to $0)
   if(Reg[rs] >= 0) PC ← PC+4 + Instruction[15:0] << 2

g. Implement a new 'incbeq' instruction. The instruction, `incbeq rs, rt, label`, stands for "increment and branch on equal". Increment always occurs!
   if(Reg[rs] == Reg[rt]) PC ← PC+4 + Instruction[15:0] << 2
   Reg[rs] ← Reg[rs] +4 #the increment always occurs

h. Implement a new 'sneg' instruction. The instruction, `sneg rs`, stands for "set on negative". (Immediate format)
   if (Reg[rs] < 0) Reg[rt] = 1 , else Reg[rt] ← 0

i. Implement a new 'lwdec' instructon. The instruction `lwdec rs, offset(rt)`, loads a value and decrements the rt value by one memory word.
   Reg[rt] ← Mem[Reg[rs] + sign-extended offset]
   Reg[rs] ← Reg[rs] – 4

j. Implement a new 'swr' instruction. The instruction `swr rd, rt(rs)`, uses a register value for the offset when calculating the memory address to store data to.
   Mem[Reg[rs] + Reg[rt]] = Reg[rd]

k. Implement a new 'slti' instruction. The instruction sets `slti rs, rt, immediate` sets the rs register to the immediate value if the value in register rs is less than register rt.
   if(Reg[rs] < Reg[rt])
      Reg[rs] ← 2's complement immediate value

Problem 4: Show how the jump register instruction 'jr' can be implemented to the multi-cycle datapath by simply making changes to the finite state machine. (Hint: $0 = $zero = 0)

Problem 5: Calculate the delay in the datapath after the addition of ALL instructions in Problem 1. Assume the same delays. What is the minimum clock cycle period at which this design can operate properly?

Problem 6: Consider changes to the original multi-cycle datapath that alters the register file so that it has only one read port. Describe any changes that will need to be made to the datapath in order to support this modification. Use the datapath diagram to illustrate the changes. Modify the finite state machine to indicate how the instructions will work given your new datapath.

Problem 7: Consider eliminating the two shift left by 2 units in the multicycle datapath. Instead of these units, the ALU will be used for shifting the values instead. What impact does this have on the original 5 instructions (lw, sw, R-type, beq, j)?

Problem 8: In the single cycle datapath control unit for the original 5 instructions (lw, sw, R-type, beq, j), the MemtoReg control signal can be eliminated and the MemRead or ALUSrc control signals can be used to control the multiplexor instead. This reduces the number of control signals required (less logic gates to implement).

a. Which other signals in the single cycle datapath can be eliminated and replaced by another existing control signal, or the inverse (NOT) of the signal.
b. Are there any signals in the multi-cycle datapath (basic 5 instructions only) which can be eliminated and replaced?