ECE 485/585 – Computer Organization and Design

HOMEWORK #4 - SOLUTION

Solve the following exercises. <u>You must show your work to get full points.</u>

Question 1.

a. What would be the total execution latency for an instruction *sw $t1, 20($t0)*? (5 Points)

- any instruction would execute in order of IF→ID→EX→MEM→WB.
- IF stage: Uses Add for PC+4, Instruction Memory to fetch instruction. Runs in parallel.
  - Thus, total latency in IF stage: **300 ps** (Add will be done before Instruction Memory)
- ID stage: Uses Register Files, and Sign-extension. Runs in parallel.
  - Thus, total latency in ID stage: **85 ps** (Sign-ext will be done before Registers)
- EX stage: Uses ALU, MUX (to select output from Sign-ext). Runs in series.
  - Thus, total latency in EX stage: 15 + 100 = **115 ps**.
- MEM stage: Uses Data Memory
  - Thus, total latency in MEM stage: **400 ps**.
- WB stage: Uses MUX, Registers. Runs in series
  - Thus, total latency in WB stage: 15 + 85 = **100 ps**.
- Thus, total execution latency is 300 + 85 + 115 + 400 + 100 = **1000 ps**.

b. What would be the total execution latency for an instruction *beq $t1, $t0, LABEL*? (5 Points)

- IF stage: Uses Add for PC+4, Instruction Memory to fetch instruction. Runs in parallel.
  - Thus, total latency in IF stage: **300 ps** (Add will be done before Instruction Memory)
- ID stage: Uses Register Files, and Sign-extension. Runs in parell.
  - Thus, total latency in ID stage: **85 ps** (Sign-ext will be done before Registers)
- EX stage: Uses ALU, MUX (to select output from Sign-ext) => Runs in series. Also uses Add and Shifter => Runs in series. These two run in parallel, then uses PCSrc MUX in series.
  - ALU, MUX latency ➔ 115 ps.
  - Shifter, Add latency ➔ 70 ps.
  - Thus, slowest is ALU, MUX latency. Then need to add PCSrc MUX as PCSrc will be based on ALU output.
  - Thus, total latency in EX stage: 115 + 15 = **130 ps**.
- Thus, total execution latency is 300 + 85 + 130 = **515 ps**.

c. Show the critical path for all MIPS instruction set types. (5 Points)

- R-type: PC → Instruction Memory → Registers → ALU → Registers
- I-type: PC → Instruction Memory → Registers → ALU → Data Memory → Registers
- J-type: PC → Instruction Memory → Shifter → Add
  - Note: J-type cannot be fully determined as given figure is incompatible with J-type. Assuming Shifter and Adder are added to the figure

d. Consider pipeline features to be added to the above figure. What would be the desired clock rate of the system to execute in pipeline mode. Prove your answer by showing your work leading to the desired clock rate. (You may discard latencies in the temporary registers in between stages) (10 Points)

- Clock rate needs to be consistent throughout each pipeline stages. We will consider the standard 5-stage pipeline MIPS architecture. Thus, we need to identify the longest delay (total latency) per stage.
    - IF stage: 300 ps
    - ID stage: 85 ps
    - EX stage: 115 ps.
    - **MEM stage: 400 ps. Longest Delay (latency)**
    - WB stage: 100 ps.
- Thus, the desired clock rate should have 400 ps per stage. **Thus, clock rate should be 1/(400ps) = 2.5 GHz**

ECE 485/585 – Computer Organization and Design

Question 2.

a) List any HW modification required for **sw** instruction along with datapath and control signal (5 Points):

⇨ **This architecture will support sw instruction without any H/W modification.**

b) Explain the operation at each component (marked red above) in RTL representation and all control signal values (indicated in blue above) for the **sw** instruction (10 Points):

Instruction Memory:

IR <- MEM[PC]

PC = PC + 4

Registers:

Read data 1 <- Reg[IR[25:21]]
Read data 2 <- Reg[IR[20:16]

ALU:

ALU result <- Read data 1 + sign_ext[IR[15:0]] (ALUsrc = 1, ALUop for add operation)

Data Memory:

Write data <- Read data 2 (MemWrite = 1, MemRead = 0, MemtoReg = X)
Address <- ALU result

Adders (both):

1st left adder's output selected.

PC <- PC + 4 (PCSrc = 0)

ECE 485/585 – Computer Organization and Design

Question 3. Assume there are no pipeline stalls and the breakdown of executed instructions is:

add	addi	not	beq	lw	sw
5%	15%	15%	35%	15%	15%

a. In what fraction of all cycles is the data memory used? (5 Points)
   - Memory is used for the lw and sw instructions.
   - Thus, **15% + 15% = 30%**

b. In what fraction of all cycles is the input of the sign-extend circuit needed? What is this circuit doing in cycles in which its input is not needed? (5 Points)
   - Sign-extend circuit needed for addi, beq, lw and sw instructions
   - Thus, 15% + 35% + 15% + 15% = 80%
   - This circuit will still operate at anytime whenever the instruction is being executed including add, not instructions. The system will simply not select the output of the sign-extend circuit as an input to other functional units.

Question 4. Exercise 4.8 from Textbook (10 Points)

a.
   - Pipelined: Clock cycle must be equal to the longest latency => 350 ps
   - Single-Cycle: Sum of all stages
     1. 250 + 350 + 150 + 300 + 200 = 1250 ps

b.
   - Pipelined: Sum of number of stages with defined clock cycle
     1. 350 * 5 = 1750 ps
   - Single-Step: Same as clock-cycle = 1250 ps

c. We'd prefer to split the longest latency, so we will split the ID stage
   - After this has been done, the new clock cycle is the new longest latency = 300 ps

d. Usage of data memory: LW + SW = 20% + 15% = 35%

e. Usage of "registers": ALU and BEQ: 45% + 20% = 65%

f.
   - Multi-Cycle: For our multi-cycle we need to set our clock cycle to be that of the longest stage (350 ps), but in some instructions we save stages.
     1. LW – uses all 5 stages
     2. SW – uses only 4 stages (no WB)
     3. ALU – uses only 4 stages (no MEM)
     4. BEQ – uses only 4 stages (no WB)
     5. Average Execution: (0.20*5 + 0.80*4)*350 = 1470 ps
   - Pipelined: On average, we see an instruction completed every cycle. In this way, our execution time will be 350 ps per instruction.

ECE 485/585 – Computer Organization and Design

Question 5. Exercise 4.7 from Textbook (10 Points)

a.

- Sign-Extend:
    1. 0000 0000 0000 0000 0000 0000 0001 0100
- Shift-left-2 (take 25-0 of Instr. And shift left by 2)
    1. 00 0110 0010 0000 0000 0000 0101 0000

b. Instruction is a SW instruction

- ALUOp – 00
- Instruction [5-0] = 010100

c.

- New PC
    1. PC + 4 (unknown)
- Path
    1. PC through Add (PC and 4), to branch MUX to jump Mux to update PC

d.

- WrRegMux
    1. 2 or 0 (RegDst is X)
- ALU Mux
    1. 20
- Mem/ALU MUX
    1. X
- Branch Mux
    1. PC + 4
- Jump MUX
    1. PC + 4

e.

- ALU: -3 and 20
- Add (PC+4): PC and 4
- Add (Branch): PC+4 and 20*4

f.

- Read Reg 1: -3
- Read Reg 2: -1
- Write Register: X
- Write Data: X
- RegWrite: 0