VE370 Homework 5 Wang Lan 5193709084

1. (15 points) If we change load/store instructions to use a register (without an offset) as the base address, these instructions no longer need to use the ALU. As a result, the MEM and EX stages can be overlapped and the pipeline has only four stages.
   (1) How will the reduction in pipeline depth affect the clock cycle time? (5 points)
   (2) How might this change improve the performance of the pipeline? (5 points)
   (3) How might this change degrade the performance of the pipeline? (5 points)

(1) No change in clock cycle time, because the execution time of the slowest stage doesn't change.

(2) This can reduce the nop needed in load-use data hazard. The number of total cycles is also reduced.

(3) More instructions are needed if we want to use offset, because we need to calculate it with addi first.

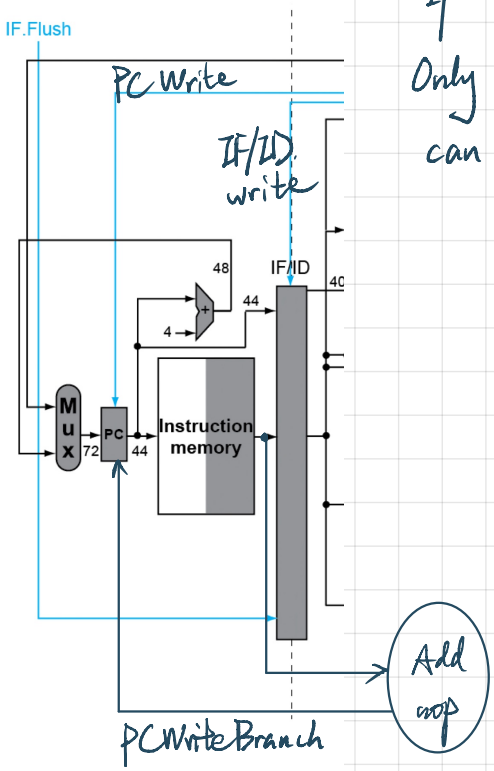
2. (1) How many nop should be inserted after each beq instruction? (5 points)
   (2) How can this stall be implemented in hardware rather than in software? Hint: nop instruction is realized as addi x0, x0, 0. (10 points)
   (3) If the above pipeline is modified to support jal instruction, which would be the earliest stage the jump instruction is identified and jump target is calculated? In that case, how many stalls would have to be inserted? How would the clock cycle time be affected? (10 points)

(1) One nop

(2) In IF stage, add a unit to stall.

If Instruction will branch, then PCWriteBranch=0

Only when PCWrite==1 and PCWriteBranch == 1 can PC update its value.



(3) In ID stage.

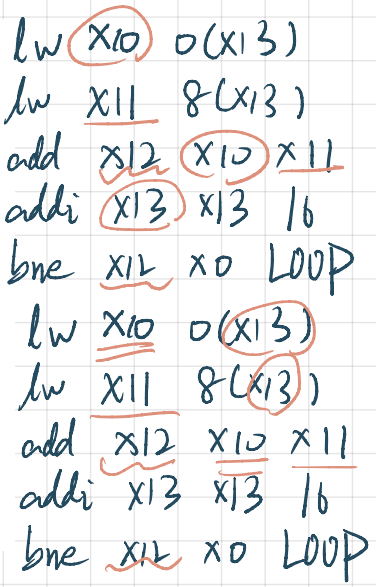
One stall.

Clock cycle time will not change unless stall is implemented by hardware and the longest clock cycle time changes.

3. (20 points) Consider the following loop.

```
LOOP: lw x10, 0(x13)
      lw x11, 8(x13)
      add x12, x10, x11
      addi x13, x13, 16
      bne x12, x0, LOOP
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, that the pipeline has full forwarding support, and that branches are resolved in the EX (as opposed to the ID) stage. Show a pipeline execution (multicycle) diagram for the first two iterations of this loop. Hint: unfold the loop first. Hint : you may use Excel to show the execution diagram.



	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16
lw x10 0(x13)	IF	ID	EX	MEM	WB											
lw x11 8(x13)		IF	ID	EX	MEM	WB										
nop			IF	ID	EX	MEM	WB									
add x12 x10 x11				IF	ID	EX	MEM	WB								
addi x13 x13 16					IF	ID	EX	MEM	WB							
bne x12 x0 LOOP						IF	ID	EX	MEM	WB						
lw x10 0(x13)							IF	ID	EX	MEM	WB					
lw x11 8(x13)								IF	ID	EX	MEM	WB				
nop									IF	ID	EX	MEM	WB			
add x12 x10 x11										IF	ID	EX	MEM	WB		
addi x13 x13 16											IF	ID	EX	MEM	WB	
bne x12 x0 LOOP												IF	ID	EX	MEM	WB

4. (1) Stall cycles due to mispredicted branches and jumps increase the CPI. What is the extra CPI due to jumps? What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the ID stage and that there are no data hazards, and that no delay slots are used. (10 points)

(2) Repeat (1) for the 2-bit predictor. (10 points)

(1) Jump will cause one stall per instruction, so CPI will increase 5%.

Branch : $1 \times (1-0.45) \times 0.25 = 13.75\%$

(2) Jump still causes CPI to increase 5%.

Branch : $1 \times (1-0.85) \times 0.25 = 3.75\%$

5. (20 points) This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, NT, T, T, NT. (T: taken, NT: not taken)

(1) What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes? (5 points)

(2) What is the accuracy of the 2-bit predictor if this pattern is repeated forever? (5 points)

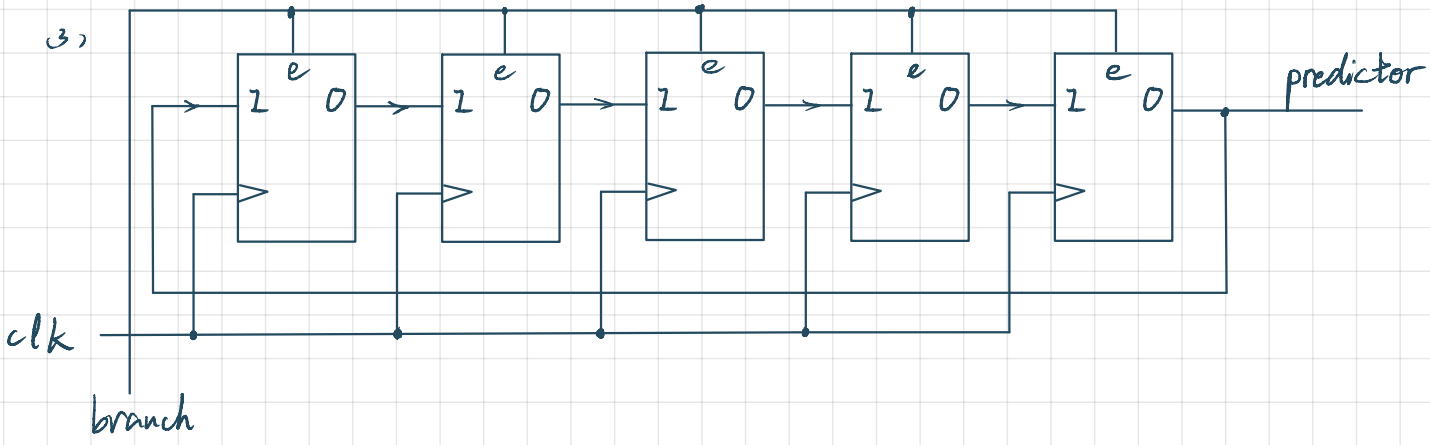
(3) Design a predictor that would achieve a perfect accuracy if this pattern is repeated forever. You predictor should be a sequential circuit with one output that provides a prediction (1 for taken, 0 for not taken) and no inputs other than the clock and the control signal that indicates that the instruction is a conditional branch. (10 points)

(1) always taken : $\frac{3}{5} \times 100\% = 60\%$

always not taken : $\frac{2}{5} \times 100\% = 40\%$

(2) Start from strong taken state, then the predictor can be correct for each "Taken" : $\frac{3}{5} \times 100\% = 60\%$

(3)



initially stored : NT T T NT T

so after each branch, the sequence will shift right, so that the prediction is always correct.