JOINT INSTITUTE
交大密西根学院

Ve370 Introduction to Computer Organization

Homework 5

Assigned: November 4, 2021

Due: 2:00pm on November 11, 2021

Submit a PDF file on Canvas

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)

   Answer: the clock cycle time won't change since the ALU operations and data memory operations wouldn't be in parallel. This won't change the critical stage.

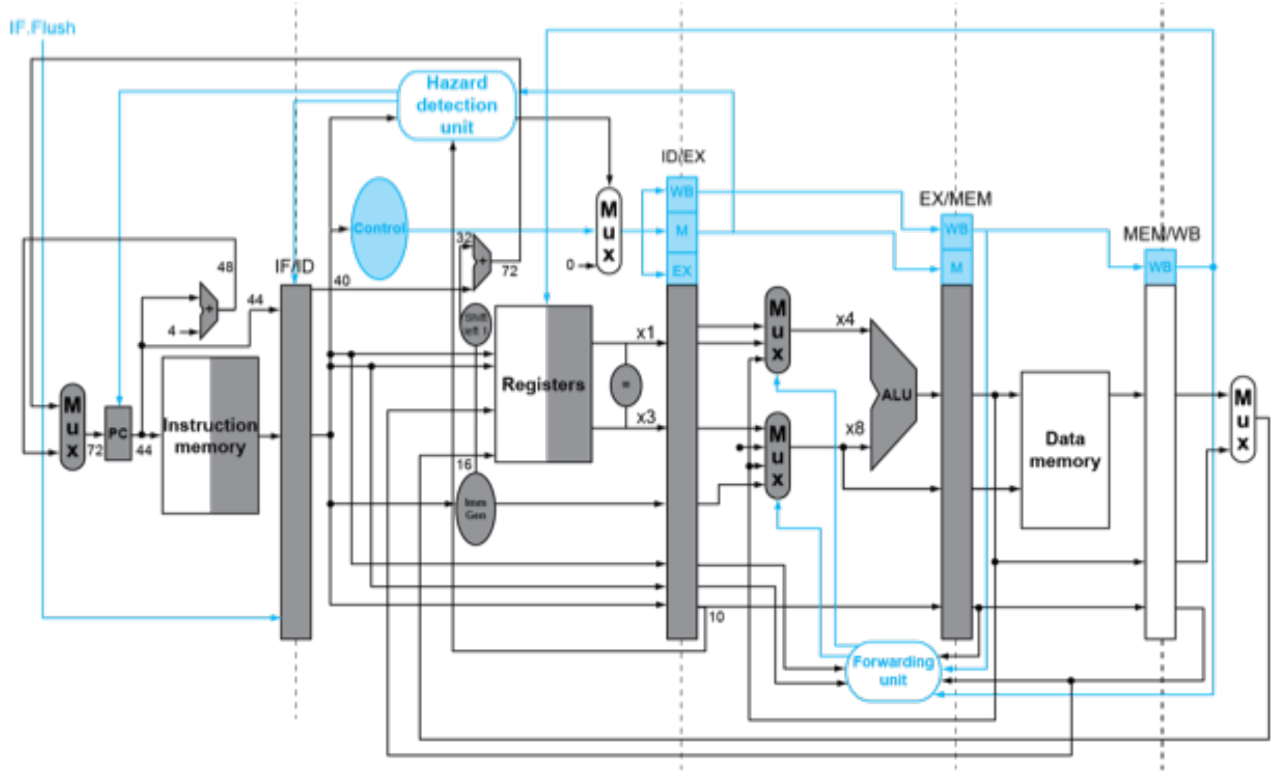
   (2) How might this change improve the performance of the pipeline? (5 points)

   Answer: Moving the MEM stage in parallel with the EX stage will eliminate the need for a cycle between loads and operations that use the result of the loads. This can potentially reduce the number of stalls in a program.

   (3) How might this change degrade the performance of the pipeline? (5 points)

   Answer: Removing the offset from load and store may increase the total number of instructions because some load and store instructions will need to be replaced with addi + load or addi + store pairs.

2. (25 points) One of the solutions to control hazard is to always stall the instruction following the branch or jump instruction by inserting nop instructions. Using the following diagram as a reference:



(1) How many nop should be inserted after each beq instruction? (5 points)

Answer: 1

(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)

Answer: (give points as long as students tried on this question)

Method 1: put nop (addi) instruction in a special address of the instruction memory, choose that address for PC input when beq is detected in the IF stage. This is simple but would involve extra time in the IF stage.

Method 2: When beq is detected in the ID stage, provide a signal to affect the control of PCSrc in the IF stage, and make PC (in addition to PC+4 and branch target) an option of the mux for PC source. This is to make sure at the next clock cycle, PC can be loaded with either PC (address of the instruction immediately following beq) or branch target. At the same time, provide a signal to clear IF/ID pipeline register so that a nop is inserted.

(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)

Answer: (go easy on this question)
Student answer could be any stage:
IF stage: in that case, no stall need be inserted. But IF stage may be prolonged due to extra work for the jal instruction.
ID stage: in that case, 1 stall need be inserted. In the ID stage, write data need to be selected between the regular data to be written into register file and PC+4, which might take a bit longer time.
EX stage: 2 stalls need be inserted. ….
MEM stage: 3 stalls needed. ….

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.

Answer :

lw x10, 0(x13)	IF	ID	EX	MEM	WB											
lw x11, 8(x13)		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				
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. (20 points) The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent flushing due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-type	branch	jal	lw	sw
40%	25%	5%	25%	5%

Also, assume the following branch predictor accuracies:

Always-Taken	Always-Not-Taken	2-Bit
45%	55%	85%

(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)

Answer :

Extra CPI by jump :

assume the number of instructions is X. The number of jump instructions is X * 5%.

Assume jump instruction is determined in the ID stage, then each jump instruction adds 1 clock cycle.

Total CPI = (X+4+X*5%)/X

Ideal CPI without stalls = (X+4)/X

Extra CPI =Total CPI – Ideal CPI = 5%

Extra CPI by mispredicted branch :

Total CPI = (X+4+X*55%*25%)/X

Ideal CPI without stalls = (X+4)/X

Extra CPI = Total CPI – Ideal CPI = 55%*25% = 0.1375

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

Answer：Extra CPI = 15% * 25% = 0.0375

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)

Answer :

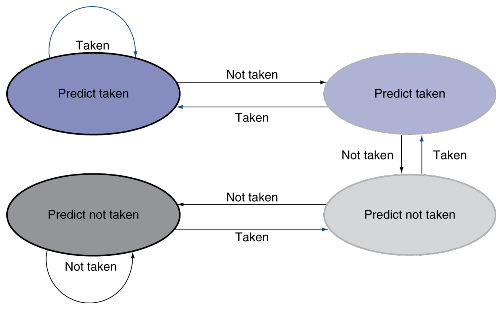
Always Taken : 3/5 = 60%

Always Not Taken : 2/5 = 40%

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

Answer :

The first few recurrences of this pattern do not have the same accuracy as the later ones because the predictor is still warming up. To determine the accuracy in the “steady state,” we must work through the branch predictions until the predictor values start repeating (i.e., until the predictor has the same value at the start of the current and the next recurrence of the pattern).



Refer to the state diagram we discussed during the lecture, and assume we start off from 00 (strong not taken), assume the other states are 01 (weak not taken), 10 (weak taken), 11 (strong taken). Following will be the prediction results :

1st round:

Branch outcome	T	NT	T	T	NT
State	00	01	00	01	10
Prediction	NT	NT	NT	NT	T
Correct?	N	Y	N	N	N

2nd round:

Branch outcome	T	NT	T	T	NT
State	01	10	01	10	11
Prediction	NT	T	NT	T	T
Correct?	N	N	N	Y	N

3rd round:

Branch outcome	T	NT	T	T	NT
State	10	11	10	11	11
Prediction	T	T	T	T	T
Correct?	Y	N	Y	Y	N

4th round:

Branch outcome	T	NT	T	T	NT
State	10	11	10	11	11
Prediction	T	T	T	T	T
Correct?	Y	N	Y	Y	N

So the predictor reaches steady state starting from the 3rd round. In steady state, the accuracy is 60%.

(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)

Answer : The predictor may be an N-bit rotate register, where N is the number of branch outcomes in the target pattern. The register should be initialized with the pattern itself (0 for NT, 1 for T, 10110 for this problem), and the prediction is always the value in the leftmost bit of the register. The register should be shifted after each predicted branch.