Homework 2

Computer 5: Fall 2022

1. In this question, we are going to investigate the difference between single-cycle, multi-cycle and pipelined implementation of the same MIPS architecture in Figure C.21 (5th edition of the text book, page C-34). The latencies associated with various computations in Figure C.21 are given in the following table. Note that for MUX, you have to consider two types delay: data input to output as well as selection input to output.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction memory | Add | Mux | Registers | ALU | Zero? | Data memory | Sign-extend | Other |
| 200ps | 100ps | 20ps | 110ps | 120ps | 10ps | 200ps | 20ps | 0 |

Suppose the processor only executes BEQZ.

1. **[10 point]** Suppose FigureC.21 is a single-cycle non-pipelined implementation. What is the minimum cycle length? In your computation show all possible activated data paths to perform BEQZ and select the path with the longest delay.
2. **[10 point]** What is the minimum cycle length if FigureC.21 is a multi-cycle non-pipelined implementation?
3. **[10 point]** What is the minimum cycle length if FigureC.21 is pipelined? Does it outperform (a)? Assume that the processor only executes BEQZ and there is no branch predication.

图示, 示意图

描述已自动生成答：a)

因此，如上图所示，最小周期长度为：200+110+140+20 = 470ps

b) 如果是多周期非流水线的情况下，最小周期长度为：200ps

c) 如果是流水线，最小周期长度为：200ps，要优于a

1. **[40 points]** For the following instructions:

Loop: L.D F1, 2048(R1)

MUL.D F1, F3, F1

L.D F0, 1024(R1)

SUBIU R1, R1, #8

DIV.D F0, F3, F0

ADD.D F5, F1, F0

MUL.D F5, F5, F6

S.D F5, 1024(R1)

BNEZ R1, Loop

The third column in the following table shows the number of cycles of latency between a source instruction (first column) and any subsequent instruction (of any type) consuming the result of the source instructions. The third column indicates the number of functional units available for executing the respective type of source instruction.

|  |  |  |  |
| --- | --- | --- | --- |
| Function Unit | Related Instruction | Latency Cycles | Number of Units |
| Integer ALU | L.D S.D SUBIU BNEZ | 1 | 2 |
| Memory Unit | L.D S.D | 4 | 2 |
| FP Adder | ADD.D | 2 | 1 |
| FP Multiplier | MUL.D | 6 | 1 |
| FP Divider | DIV.D | 8 | 1 |

Assume that the reservation station and the reorder buffer both have infinite size. The integer ALUs are used for effective address calculation for load/store instructions, execution of SUBIU and BNEZ instructions. Assume that you can make at most two writes to CDB in one clock cycle. Create two tables showing when each instruction issues, begins execution, accesses memory and writes its result to the CDB for the first two iterations for the following two scenarios using **Tomasulo’s** **algorithm**.

a) Use a MIPS pipeline with **two-issue** and **without speculation**. Assume that branches are issued alone (single-issue for that time step) and branch prediction is perfect.

b) Use a MIPS pipeline with **two-issue** and **with speculation**. You also need to specify when each instruction commits. Assume that up to two instructions of any type can commit per cycle. Branches are issued alone (single-issue for that time step) and branch prediction is perfect. Note that stores will spend 4 cycles in the commit stage, because its memory access occurs during commit.

(First three rows in each table are given as examples.)

Without speculation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iter. | Instruction | | Issue | Execute | Memory | Write-CDB |
| 1 | L.D | F1, 2048(R1) | 1 | 2 | 3-6 | 7 |
| 1 | MUL.D | F1, F3, F1 | 1 | 8-13 |  | 14 |
| 1 | L.D | F0, 1024(R1) | 2 | 3 | 4-7 | 8 |
| 1 | SUBIU | R1, R1, #8 | 2 | 3 |  | 4 |
| 1 | DIV.D | F0, F3, F0 | 3 | 9-16 |  | 17 |
| 1 | ADD.D | F5, F1, F0 | 3 | 18-19 |  | 20 |
| 1 | MUL.D | F5, F5, F6 | 4 | 21-26 |  | 27 |
| 1 | S.D | F5, 1024(R1) | 4 | 5 | 28-31 |  |
| 1 | BNEZ | R1, Loop | 5 | 6 |  |  |
| 2 | L.D | F1, 2048(R1) | 6 | 7 | 8-11 | 20 |
| 2 | MUL.D | F1, F3, F1 | 6 | 27-32 |  | 33 |
| 2 | L.D | F0, 1024(R1) | 7 | 8 | 9-12 | 21 |
| 2 | SUBIU | R1, R1, #8 | 7 | 8 |  | 9 |
| 2 | DIV.D | F0, F3, F0 | 8 | 22-29 |  | 30 |
| 2 | ADD.D | F5, F1, F0 | 8 | 34-35 |  | 36 |
| 2 | MUL.D | F5, F5, F6 | 9 | 37-42 |  | 43 |
| 2 | S.D | F5, 1024(R1) | 9 | 10 | 44-47 |  |
| 2 | BNEZ | R1, Loop | 10 | 11 |  |  |

With speculation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Iter. | Instruction | | Issue | Execute | Memory | Write-CDB | Commit |
| 1 | L.D | F1, 2048(R1) | 1 | 2 | 3-6 | 7 | 8 |
| 1 | MUL.D | F1, F3, F1 | 1 | 8-13 |  | 14 | 15 |
| 1 | L.D | F0, 1024(R1) | 2 | 3 | 4-7 | 8 | 15 |
| 1 | SUBIU | R1, R1, #8 | 2 | 3 |  | 4 | 16 |
| 1 | DIV.D | F0, F3, F0 | 3 | 9-16 |  | 17 | 18 |
| 1 | ADD.D | F5, F1, F0 | 3 | 18-19 |  | 20 | 21 |
| 1 | MUL.D | F5, F5, F6 | 4 | 21-26 |  | 27 | 28 |
| 1 | S.D | F5, 1024(R1) | 4 | 5 |  |  | 28-31 |
| 1 | BNEZ | R1, Loop | 5 | 6 |  |  | 29 |
| 2 | L.D | F1, 2048(R1) | 6 | 7 | 8-11 | 20 | 30 |
| 2 | MUL.D | F1, F3, F1 | 6 | 27-32 |  | 33 | 34 |
| 2 | L.D | F0, 1024(R1) | 7 | 8 | 9-12 | 21 | 34 |
| 2 | SUBIU | R1, R1, #8 | 7 | 8 |  | 9 | 35 |
| 2 | DIV.D | F0, F3, F0 | 8 | 22-29 |  | 30 | 35 |
| 2 | ADD.D | F5, F1, F0 | 8 | 34-35 |  | 36 | 37 |
| 2 | MUL.D | F5, F5, F6 | 9 | 37-42 |  | 43 | 44 |
| 2 | S.D | F5, 1024(R1) | 9 | 10 |  |  | 44-47 |
| 2 | BNEZ | R1, Loop | 10 | 11 |  |  | 45 |

1. Accurate branch prediction is crucial to modern processors. Some early studies discovered that random predicators have quite good performance on some benchmarks. A random predictor is a predictor, which randomly gives results with a fixed probability: p for TAKEN and 1-p for NOT TAKEN. Due to the technology limit, we can only physically fabricate one type of random predictor, RP5, which has p=0.5.
2. **[10 points]** Is it possible to make a random predictor with p=0.4 based on RP5? If the latency of RP5 is t, what is the expected latency of your predictor?
3. **[10 points]** Suppose 40% of the branches are taken in a benchmark program. What is the miss prediction ratio of your random predictor with p=0.4 on the benchmark?
4. **[10 points]** Can you design a deterministic predictor that always has a lower mis-prediction rate than your random predictor with p=0.4 on the benchmark?

答：a) 我们从0.5预测器的物理构造出发，设计0.4预测器。0.5预测器是由1个二进制位表示的，0表示预测失败，1表示预测成功，利用这个思想，我们可以使用3个二进制位表示0.4预测器，也就是说，我们使用3个0.5预测器产生3个二进制。3个二进制位可以表示8种情况，我们选择5种情况为有效情况，分别是000、001、010、011、111，其余三种为无效情况，将有效情况中的000、001、010代表为预测成功，011、111代表为预测失败，就实现了0.4预测器。此时0.4预测器的期望延迟是5\*t

b) 0.4

c) 如果在基准程序中条件之间存在关系的话，使用确定性预测器可以有更好的预测效果，比如我们可以记录上一次实际的跳转情况，并直接用它当作下一次的预测。