COMPUTER ARCHITECTURE MIDTERM EXAM

Fall 2022

Instructions:

1. Sign your name, student ID and solutions on your answer sheet.
2. Show all work. If you cannot finish a problem, your written work will help us to give you partial credit.
3. Write legibly and clearly.
4. Read each question carefully before answering it. Try to solve the problems that seem easy before attacking the harder ones.

1. For each statement below, indicate whether the statement is true (T) or false (F). (10%)

1) (F) Load-store architecture means that an ALU instruction can load data from or store data to memory directly.

2) (F) CPU time of a program = Instruction Count * Clock Cycle Time.

3) (T) The PC counter is used to point to an instruction which is ready to execute.

4) (F) In the following sequence of instructions:

LW R1, 0(R2)

ADD R1, R2, R3

there is a WAR dependence.

5) (F) Anti-dependences (WAR) and output dependences (WAW) are considered true dependences because they may not be removed either by the compiler or by the hardware.

2. What are the differences of structural hazards, data hazards and control hazards? (10%)

结构冲突是因为硬件资源不够，流水线中多条指令想同时使用某一硬件

数据冲突是后一条指令想使用之前某指令得到的数据

控制冲突是流水线中跳转指令无法立即得到跳转方向和地址，后一条指令无法判断是否执行

3. When the pipeline depth decreases, i.e., the pipeline is divided into less pipeline stages, will the latency to process a single instruction be increased, decreased or unchanged in general? Explain your reason briefly. (10%)

时延减少 ∵可以减少流水线寄存器的开销

但如果划分不当，也会增大时延

4. A new floating-point unit speeds up floating-point operations by two times. In an application, 1/5 of the instructions are floating-point operations. (10%)

1) What is the overall speedup? (Ignore the penalty to other instructions).

2) Assume that the speeding up of the floating-point unit mentioned above slowed down load and store accesses resulting in a 1.2x (1.2 倍) slowdown. Assume the load instructions constitute 15% and store instructions constitute 10% of the total instruction. What is the effective overall speedup now?

1) $speedup = \frac{1}{1-\frac{1}{5}+\frac{1}{5}\div 2} = \frac{10}{9} \approx 1.11$

2) speed up = $\frac{1}{1-45\% + \frac{20\%}{2} + 25\% \times 1.2} = 1.05$

5. We begin with a computer implemented in single-cycle implementation, and convert it to a pipelined implementation. The original machine had a clock cycle time of 7ns. After the stages were split, the measured times of each stage are: IF (1ns), ID (2ns), EX (1ns), MEM (3ns) and WB (2ns). The pipeline register delay is 0.1ns. Ignore all the other overhead. (15%)

1) What is the clock cycle time of the 5-stage pipelined machine?

2) If there exists a stall every 4 instructions, what is the CPI of the new machine?

3) Based on the assumption of question 2), what is the speedup of the pipelined machine over the single-cycle machine?

1) 流水线 clk 取最长时钟，因此为 3ns + 0.1ns = 3.1 ns

2) CPI = 5 cycles / 4 inst = 1.25

3) speedup = $\frac{IC \times CPI \times CCT}{IC' \times CPI' \times CCT'} = \frac{IC \times 1 \times 9.1}{IC \times 1.25 \times 3.1} = 2.348$

6. Suppose a machine adopts always-untaken as branch prediction technique, and the branch frequencies (as percentages of all instructions) are as follows:

- Conditional branches 20%
- Jumps and Calls 5%

Among all conditional branches, 50% are taken, 50% not taken.

We are examining a 4-stage pipeline where the branch is resolved at the end of the 2nd stage for Jumps and Calls and at the end of the 3rd stage for conditional branches. Ignoring other pipeline stalls, what is the CPI of this machine? (15%)

采用 untaken 预测，对于 Jumps and Calls，预测永远为错，每次加入 1 个 stall

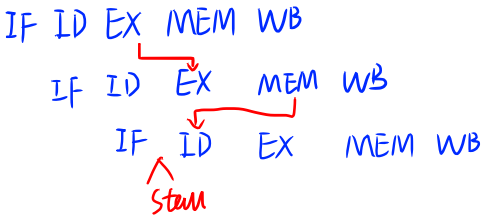
对于 conditional branch. 50% 正确，无开销

50% 错误，因为 3rd 才出结果，所以加入 2 个 stall

∴ CPI = 1 + 20% × 1 + 5% × 50% × 2 = 1.25

7. Assume all branch instructions finish at ID stage in the 5-stage classic MIPS pipeline. Write a small piece of code in which the branch instruction in the ID stage causes a RAW hazard, even with data forwarding. (10%)

```
Loop: DADDI R1, R2, R3
      LW    R4, 20(R1)
      BEQ   R4, Loop
```



8. Pipeline Timing Chart (20%)

instr1 Loop: LD R1, 0(R2) ;load R1 from address 0+R2
2 LD R5, 2(R2) ;load R5 from address 2+R2
3 DADDI R5, R1, R5 ;R5=R1+R5
4 SD R5, 0 (R2) ;store R5 at address 0+R2
5 DADDI R2, R2, #4 ;R2=R2+4
6 DSUB R4, R3, R2 ;R4=R3-R2
7 BNEZ R4, Loop ;branch to Loop if R4!=0

Assume branch instruction finishes at ID stage, and new PC available at the next cycle.

(1) Data hazards are caused by data dependences in the code. Whether a dependency causes a hazard depends on the machine implementation (i.e., number of pipeline stages). List all of the data dependences in the code above. Record the register, source instruction, and destination instruction;

(2) Use a pipeline timing chart (no more than 20 cycles) to show the timing of this instruction sequence for the 5-stage MIPS pipeline without any forwarding or bypassing hardware but **assuming that a register read and a write in the same clock cycle "forwards" through the register file**. Assume that the branch is handled by flushing the pipeline;

(3) Use a pipeline timing chart (no more than 15 cycles) to show the timing of this instruction sequence for the 5-stage MIPS pipeline with full forwarding and bypassing hardware. Assume that the branch is handled by predicting it as not taken.

(1) 标号如上

R1: instr1 → 3. RAW

R2: instr1 → 5. 2 → 5, 4 → 5. WAR
instr 5 → 6. RAW

R3 无依赖

R4: instr 6 → 7, RAW

R5: instr 2 → 3, WAW
instr 2 → 3, 3 → 4, 2 → 4, RAW

(2)

Loop: LD R1, 0(R2) IF ID EX MEM WB
LD R5, 2(R2) IF ID EX MEM WB
DADDI R5, R1, R5 IF stall stall ID stall EX MEM WB
SD R5, 0 (R2) IF stall stall ID stall EX MEM WB
DADDI R2, R2, #4 IF ID EX MEM WB
DSUB R4, R3, R2 IF stall stall ID stall EX MEM WB
BNEZ R4, Loop IF stall stall ID stall EX MEM WB

(3)

```
Loop:  LD R1, 0(R2)          IF ID EX MEM WB
       LD R5, 2(R2)             IF ID EX MEM WB
       DADDI R5, R1, R5            IF ID stall EX MEM WB
       SD R5, 0 (R2)                  IF stall ID EX MEM WB
       DADDI R2, R2, #4                  IF ID EX MEM WB
       DSUB R4, R3, R2                      IF ID EX MEM WB
       BNEZ R4, Loop                           IF stall ID stall EX MEM WB
```