CS211 Computer Architecture II

Pipeline and CPU Cache

Assigned 28/09/2020 Homework #2 *Due 23:59:59 19/10/2020*

https://toast-lab.gitee.io/courses/CS211@ShanghaiTech/Fall-2020/

The homework is intended to help you learn the material, and we encourage you to collaborate with other students and to ask questions in discussion sections and office hours to understand the problems. However, each student must turn in their own solution to the problems.

The homework also provides essential background material for the mid-term and final exams. It will be graded primarily on an effort basis, but if you do not work through it, you are unlikely to succeed on the mid-term and final exams! We will distribute solutions to homework assignment. Note that each homework assignment is due at its respective due time, and all assignments are to be submitted **in English**. However, late submissions will **NOT** be accepted, except for extreme circumstances and/or with prior arrangement.

Name: _____

ID: _____

Problem 1: Pipeline and Hazards

In this problem, your task is to go through following problems to understand pipeline and hazards.

Problem 1.1 Pipeline and Hazards (3.5 points)

Let us begin by considering the following assembly code:

```
Loop: ld x11,0(x12)    ; load x11 from address 0+x12
      addi x11,x11,64  ; x11=x11+64
      sd x11,0,(x12)   ; store x11 at address 0+x12
      addi x12,x12,4   ; x12=x12+4
      sub x14,x13,x12  ; x14=x13-x12
      bnez x14, Loop   ; branch to Loop if x14 != 0
```

Assume that the initial value of `x13` is `x12+16`.

1.1.a. (0.5 point) Data hazards are caused by data dependencies 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; for example, "there is a data dependency for register `x11` from the `ld` to the `addi`".

1.1.b. (1 point) Show the timing of this instruction sequence for the 5-stage RISC pipeline without any forwarding or bypassing hardware but assuming that a register read and write in the same clock cycle "forwards" through the register file, as between the `add` and `sll` shown on the 42nd slide of L01 (https://toast-lab.gitee.io/courses/CS211@ShanghaiTech/Fall-2020/lecture_notes/L01-Intro_p1.pdf). Fill the following pipeline chart (F: instruction fetch; D: instruction decode; X: execution; M: memory access; W: writeback) and add more rows and/or columns if necessary. Assume that the branch is handled by *flushing the pipeline*. If all memory references take 1 cycle, how many cycles does this loop take to execute?

`ld x11,0(x12)`	F	D	X	M	W							
`addi x11,x11,1`		F										
`sd x11,0,(x12)`												
`addi x12,x12,4`												
`sub x14,x13,x12`												
`Bnez x14, Loop`												

1.1.c. (1 point) Show the timing of this instruction sequence for the 5-stage RISC pipeline with full forwarding and bypassing hardware. Fill the pipeline timing chart below. Assume that the branch is handled by predicting it *as not taken*. If all memory references take 1 cycle, how many cycles does this loop take to execute?

`ld x11,0(x12)`	F	D	X	M	W							
`addi x11,x11,1`		F										
`sd x11,0,(x12)`												
`addi x12,x12,4`												
`sub x14,x13,x12`												
`Bnez x14, Loop`												

1.1.d. (1 point) High-performance processors have very deep pipelines—more than 15 stages. Imagine that you have a 10-stage pipeline in which every stage of the 5-stage pipeline has been split in two. The only catch is that, for data forwarding, data are forwarded from the end of a pair of stages to the beginning of the two stages where they are needed. For example, data are forwarded from the output of the second execute stage to the input of the first execute stage, still causing a 1-cycle delay. Show the timing of this instruction sequence for the 10-stage RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in 1.1.b and 1.1.c (but with stages labelled IF1, IF2, ID1, etc.). Assume that the branch is handled by predicting it *as taken*. If all memory references take 1 cycle, how many cycles does this loop take to execute?

Problem 2: CPU Cache

In this problem, your task is to go through following problems to consider the access to a two-level cache.

Problem 2.1 Critical Word First and Early Restart (1.5 points)

From time to time, the processor normally needs just one 4-byte word of a cache block at a time. As a result, there are two strategies that have been studied.

Critical word first—Request the missed word first from memory and send it to the processor as soon as it arrives; let the processor continue execution while filling the rest of the words in the block.
Early restart—Fetch the words in normal order, but as soon as the requested word of the block arrives, send it to the processor and let the processor continue execution.

Consider that we have an inclusive two-level cache with critical word first and early restart on L2 cache misses. Assume a 1MB L2 cache with 64-byte blocks and the bus between it and main memory is 128-bit wide. Assume that the L2 can be written with 16 bytes every 4 processor cycles, the time to receive the first 16-byte block from the memory controller is 120 cycles, each additional 16-byte block from main memory requires 16 cycles, and data can be bypassed directly into the read port of the L2 cache. Ignore any cycles to transfer the miss request to the L2 cache and the requested data to the L1 cache.

2.1.a. (0.5 point) How many cycles would it take to service an L2 cache miss with and without critical word first? How about with and without early restart?

2.1.b. (0.5 point) Do you think critical word first and early restart would be more important for L1 caches, or L2 caches, and what factors would contribute to their relative importance?

2.1.c. (0.5 point) Do you think critical word first and early restart would be more useful for instructions, or data? Why?