Processor Architecture

Bonus Assignment in IS2202 Computer Systems Architecture, 2024

Instructions

- The assignment must be solved *individually*.
- Solutions should be properly *motivated*, with a few short sentences and/or formulas.
- Answer the question in your own words. Copying complete answers from other sources is not acceptable. You may, however, quote other sources as part of an explanation or discussion. In that case, include a proper reference to the source and describe your interpretation. *Specifically, the description of your interpretation must be longer than the quote.*
- For any material that is not your own: if you quote it or use it in any way, you must include a proper reference. This includes books, research papers, Internet sources, other students, and other material that is not your own.
- A proper reference describes the source clearly and concisely, so that any reader instantly can find the relevant page of the source. Author, title, date, and URL must be present, if available.
- All submitted reports will be automatically checked for plagiarism.
- Solutions may be given in English or Swedish, but not a mix of the languages.
- Only hand in PDF files from computer generated sources. *All text must be machine-readable.* Scanned text is not allowed.
- Solutions must be handed in through Canvas. *Submissions by e-mail are not accepted.*
- You need to score at least 15 points to qualify for a bonus point at the exam.
- The deadline is given in Canvas. After the deadline, submission is closed and no further submissions are possible.

Parameters for problems

020717-5563

- You have a Swedish civic registration number (personnummer) of the form YyMmDd-XXXX. Use the following formulas to calculate the parameters in the problems on the following pages.
- Register $t**D** = 1
- Branch-history entry-size **H** = M + 1 = 1 (bits)
- Number of branch-history entries **E** = 16×2^d = 16 × 2^7=2^11
- Degree of superscalarity **P** = M + 2 = 2 (number of instructions)

Problem 1. Dependences and Hazards, 5 points

Parameters Y, y, M, m, D, d, refer to the formulas on page 1.

Consider the following snippet of Mips assembly code. All operands are 32 bits wide (4 bytes). A comment after each instruction contains pseudo-code, that describes the effect of the instruction.

```
i1:     lw    $t1,0($s0)      # $t1 <- memory($s0)
i2:     lw    $t2,4($s0)      # $t2 <- memory(4 + $s0)
i3:     add   $t3,$t1,$t2     # $t3 <- $t1 + $t2
i4:     addi  $t4,$t3,-7      # $t4 <- $t3 - 7
i5:     sw    $t4,8($s0)      # memory(8 + $s0) <- $t4
i6:     lw    $t5,12($s0)     # $t5 <- memory(12 + $s0)
i7:     addi  $t6,$t5,212     # $t6 <- $t5 + 212
i8:     beq   $t5,$t4,Li11    # if $t5 == $t4 goto Li11
i9:     nop                   # unused branch delay slot
i10:    addi  $t0,$t3,-7      # $t0 <- $t3 - 7
Li11:   sw    $t0,20($s0)     # memory(20 + $s0) <- $t0
```

There are three different classes of hazards: *structural hazards*, *data hazards*, and *control hazards*.

Data hazards are caused by dependeces, and dependences can be divided into three different types:

- RAW – Read After Write dependence (also known as a data dependence)
- WAR – Write After Read dependence (also known as an anti-dependence)
- WAW – Write After Write dependence (also known as an output dependence)

The following questions assume a processor with a simple, 5-stage in-order pipeline, with forwarding paths (bypassing paths), as usual.

a) Write down your personal values of Y, y, M, m, D, d, and the resulting register $tD.

b) For each and every dependence involving register $tD, give the two instructions that are dependent, and also state the kind of dependence for that particular instruction pair.

c) For each and every dependence involving register $t5, give the two instructions that are dependent, and also state the kind of dependence for that particular instruction pair.

d) Does any dependence involving register $t5 in the code snippet above cause a hazard that requires stalling? Why, or why not?

e) Explain why WAR and WAW hazards cannot occur in a simple in-order 5-stage pipeline. Use up to 3 sentences.

f) Explain what a *structural hazard* is. Use up to 3 sentences.

Problem 2. Static and Dynamic Scheduling, 5 points

Parameter P refers to the formula on page 1.

The following questions assume a superscalar processor (called ”Bruno”). The processor can execute P instructions in parallel, if the P instructions are independent. Dependent instructions can never execute in the same cycle, but always in two consecutive cycles.

There are enough functional units to execute P instructions simultaneously, as long as at most one of the P instructions is a branch or jump instruction.

In this problem, assume the following:

- There is **no pipelining,** and therefore no pipelining-related hazards.
- All instructions have already been fetched by the processor, and are waiting for execution.
- Each instruction is *either* executed completely in **one** clock-cycle, *or* must wait because of dependences and/or hazards.

a) Consider the (unmodified) code snippet from Problem 1. Please specify in which cycle each instruction of the code snippet will execute on a superscalar ”Bruno” processor, with in-order execution. Assume that instruction `i1` executes in cycle number 1, and that the the branch (`i8`) is not taken. (1 point)

Instruction scheduling is reordering the instructions of a program, to improve performance on a specific type of computer.

b) The instructions in the code snippet can be scheduled for faster execution on the ”Bruno” processor. Show the order of instructions that gives the fastest possible execution, while still guaranteeing identical results to the unmodified code. *Use only instruction numbers for your answer,* such as: i1, i7, i3, … or similar. Only the order of instructions may be changed, nothing else. For this question, you may assume that no exceptions occur. Please note, that you **cannot** assume that branches are correctly predicted. (2 points)

Out-of-order execution means that the processor reorders instructions to improve performance. Out-of-order execution is often used together with *renaming*.

c) Consider the (unmodified) code snippet from Problem 1. Assume a version of the superscalar ”Bruno” processor, with out-of-order execution and renaming. Please specify in which cycle each instruction of the code snippet will execute on this processor. Assume that instruction `i1` executes in cycle number 1, and that the the branch (`i8`) is not taken. (2 points)

Problem 3. Branch prediction, 5 points

Parameters H and E refers to the formulas on page 1.

One type of branch predictor is the H-bit branch-prediction table. The table uses some bits of the program-counter as the input, and produces a prediction as the output.

a) Explain the main advantage of a 2-bit predictor over a 1-bit predictor. Use up to 3 sentences.

b) How many bits are required to implement a H-bit branch-prediction table with E entries?

c) Assume a processor, similar to Mips or RISC-V, with 32-bit addresses and a byte-addressed memory. All instructions have a size of 32 bits (4 bytes). The processor has an H-bit branch-prediction table with E entries. Specify exactly which bits of the program-counter are used as the input to the prediction table. Use bit-numbers in your answer. The program counter has 32 bits, numbered from bit 31 (most significant bit) through bit 0 (least significant bit).

d) Describe how an E-entry branch target buffer (BTB) would work. Considering a correctly predicted branch, please specify the number of input bits, the number of output bits, and the status bits (if any). Use up to 3 sentences.

e) An extension of the branch target buffer (BTB) is *branch folding*. Describe how branch folding can be implemented in the BTB, and how performance can be improved by branch folding. Use up to 3 sentences.

Problem 4. Architecture and Microarchitecture, 5 points

A superscalar processor typically has a deep pipeline, with more than 5 pipeline-stages. Shown below is a simplified version of the integer pipeline of a superscalar processor. Every clock cycle, the processor fetches two instructions in its first pipeline stage.

stage 1	stage 2	stage 3	stage 4	stage 5	stage 6	stage 7
Instruction Fetch	Branch Prediction	Instruction Decode	Register Read	Execute 1	Execute 2	Register Write

a) If a branch is mispredicted, which stage in this pipeline would be the earliest possible stage where the misprediction could be detected? Why?

b) *Please note, that the numbers and circumstances in this particular question are unlikely in real programs.* Assume that every 10th instruction is a branch, and that 90% of branches are correctly predicted. Also assume that non-branch instructions and correctly predicted branches execute without stalling. What percentage of fetched instructions will have to be discarded (flushed) because of mispredictions?

c) If a Load instruction causes a page-fault, which stage in this pipeline would be the earliest possible stage where the page-fault could be detected? Explain why, in up to 3 sentences.

A highly desirable feature of a processor is *precise exceptions*. Precise exceptions is a guarantee that all side effects of instructions happening before the exception are visible, and that no side effects from later instructions are visible.

d) How can *precise exceptions* be implemented in a processor with out-of-order execution? Explain in up to 3 sentences.

Three common types of Instruction-Set Architectures are: stack-based, accumulator-based, and register-based. All of them have been implemented in hardware at some point in time.

e) Most operations require 3 different operands: 2 input operands and 1 destination operand. In accumulator-based machines, one of the input operands is always in the *accumulator*. What is normally the location of the 2nd input operand?