RHODES UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

EXAMINATION: NOVEMBER 2020

COMPUTER SCIENCE HONOURS
PAPER 3 – ARCHITECTURE

Internal Examiners: Prof P Machanick

External Examiner: Prof Ian Sanders

MARKS: 100
DURATION: 4 hours

GENERAL INSTRUCTIONS TO CANDIDATES

1. This paper consists of 5 pages, 5 questions and a sheet with login details. **PLEASE MAKE SURE THAT YOU HAVE A COMPLETE PAPER.**
2. Run under Practical Exam conditions, open book, open access to Internet
3. State all assumptions and show working for all numeric examples.
4. Credit is given for insight – pay attention to where ***explain*** or ***discuss*** is requested.

PLEASE DO NOT TURN OVER THIS PAGE UNTIL TOLD TO DO SO.

Question 1 – Instruction Set Design [4 + 4 + 6 = 14 Marks]

In each of the following your own views, as justified by facts you use, are important – I am not looking for a repeat of what you were told.

a) In the MIPS instruction set, a register field can in some cases be a source of a value and in others a target. Discuss how this design contrasts with the RISC-V approach.

b) The RISC-V design has some immediate formats where the data bits are split into different parts of the word. Comment on why this was done and explain any strengths or weaknesses of this design feature.

c) Intel is weak at instruction set architecture but strong on microarchitecture and fabrication technology. Discuss.

Question 2 – Memory and Quantitative Design [2 + 6 + 6 = 14 Marks]

In each of the following take care to quantify your answer in the terms requested. Assume the following information as a baseline for each part of this question:

- 2GHz clock
- 1 instruction per clock in the absence of stalls
- L1 cache access takes 2 cycles but is fully pipelined
- L2 cache access takes 10 cycles but results in a stall
- DRAM access takes 100 cycles
- Instruction mix: 20% branches, 20% loads, 10% stores, remainder ALU

State any assumptions needed to answer (e.g. filling in missing detail).

a) What is the time per instruction in *ns* in the absence of any stalls?

b) If 1% of instructions result in a miss to L2 and 10% of L2 references result in a miss to DRAM, what is the average number of clock cycles per instruction (CPI) assuming all stalls arise from memory accesses? Explain your answer.

c) Assume now that 20% of all misses are data misses and that they split between load and store misses in the same proportion as load and store instructions. Explain how you could use this information to evaluate a write buffer that eliminates stalls for write misses. Discuss factors and data missing from this example that your design should take into account.

Question 3 – Pipelines and ILP **[(6+8) + 4 + 4 = 22 Marks]**

For this question, use the following code in the machine code notation used in the course. The instruction set is described in the notes (RISC-V with simplified register naming), as is the 5-stage pipeline to be used in this question.

```
# registers:
# R1: base address of a; a copy we can change
# R2: base address of b; a copy we can change
# R3: N; constant in this code
# R4: i; loop counter
# R5: holds value of a[i]
# R6: holds value of b[i]

        li R4, 0          # for (int i = 0; i < N; i++) {
        j fortest         # test loop at end
forbody: lw R6, 0(R2)     #   a[i] = b[i] * 2
        li R5, 2          #   placeholder for a[i]
        mult R5, R6, R5
        sw R5, 0(R1)
fornext: addi R4, R4, 1 # increment loop counter
        addi R1, R1, 4 # increment base addresses by 4:
        addi R2, R2, 4 #   (word size in bytes)
fortest: blt R4, R3, forbody
# }
```

a) Draw a timing diagram (in the style of Figure 4.4, p 62 of the notes) illustrating the time it takes to do two (2) iterations of the loop under the following assumptions:

   i. No stalls.

   ii. Redraw the timing diagram for maximum possible forwarding and explain how stalls can be eliminated, given these design parameters and discuss your answer:
   - pipeline registers update in the first half of a cycle and can be read in the second half
   - backward branches predicted as taken; forward branches predicted as not taken.

b) Discuss how a more sophisticated branch predictor would or would not make a difference in this case, versus the simple branch predictor of part (a)(ii).

c) Discuss how would Tomasulo's algorithm could apply to this example and what could change that would make it more useful.

Question 4 – Multiprocessors **[6 + 6 + 8 = 20 Marks]**

In this question, use the latencies in Table 5.1, p 87 of the notes. A multicore design shares data through the L3 cache; updates have to go to L3 before they are seen elsewhere. A MESI protocol is used.

a) A programmer has a loop initializing an array and wants to splits it into two threads. The original code is as follows:

```
for (int i = 0; i < N; i++)
  a[i] = calc (i);
```

Discuss how to split this code between two threads and the performance problems you need to avoid, particularly in terms of memory accesses.

b) Explain how the M-lock introduced in the course scales up well as the number of threads trying to acquire the lock increases.

c) One core attempts to modify the same location in memory (sw instruction) and another core tries to read it (lw instruction). Outline the steps taken including bus transactions from the start of the sw or lw instruction to the completion of the other instruction, taking into account differences in timing between the two cores. ***Assume both cores at the start have the block in shared (S) state***.

Question 5 – Practical Question **[30 Marks]**

You are supplied with code that simulates a multilevel cache system. You may run the code, compile it and alter it if you need to. You are supplied with a Linux account for this question; use the given login details (good only for the exam). In this question I am looking for insights so there is no fixed breakdown between questions: you may answer the parts where you can give the best answers in more detail than the rest.

a) Look for a function in the given source files called handleMiss.
   i. Explain why all levels of the cache above the one where the miss occurred need to be examined to handle a miss.
   ii. Now examine the given code and explain why it is implemented the way it is.

b) Run the given code set up as follows (you will need to study the README file for how to do this):

- 2 levels of cache
  - first level
    - split I and D, each 16KiB, 32B blocks
    - I and D both direct-mapped
    - hit time
      - zero for D cache (absorbed by the pipeline)
      - 1 cycle for I cache
    - lookup time (to check tags) 1 cycle;
  - second level
    - 256MiB, 32B blocks
    - 2-way associative
    - hit time: 10 cycles
    - lookup time: 2 cycles
  - third level
    - 4MiB, 64B blocks
    - 4-way associative
    - hit time: 20 cycles
    - lookup time: 4 cycles
  - DRAM
    - Unlimited, no misses
    - Hit time: 100 cycles

   i. Report on statistics output by the code
   ii. Vary parameters to see what difference it makes if you use a bigger or smaller cache, vary the block size and change associativity.
   iii. Now discuss what effect it would have on the numbers if you realistically adjust access times for higher associativity and larger block sizes.

c) Comment on any aspects of the simulation you choose in terms of whether it is correct, could be done better or does or does not accurately capture simulated time.

END OF EXAMINATION