RHODES UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

EXAMINATION: JUNE 2018

COMPUTER SCIENCE HONOURS
PAPER 5 – ARCHITECTURE

Internal Examiners: Prof P Machanick

MARKS: 100
DURATION: 4 hours

External Examiner: Prof Michelle Kuttel

GENERAL INSTRUCTIONS TO CANDIDATES

1. This paper consists of 5 pages, 4 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. Answer all questions in the answer book provided OR in your preferred word processor and save regularly to your Exam Folder.
4. State all assumptions and show working for all numeric examples.

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

Question 1 – Instruction Set Design **[4 + 4 + 6 = 16 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) The RISC-V architecture always uses the R_d field as a destination for all instruction types that change a register value. Compare with the MIPS instruction set design and discuss whether the RISC-V approach is better.
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 any strengths or weaknesses of this design feature.
c) Did Intel win the instruction set war? Consider the breadth of computer applications and where Intel is most and least competitive. Compare with any designs that may have proved less or more competitive than IA32 and its current 64-bit incarnation.

Question 2 – Memory and Quantitative Design **[2 + 3 + 2 + 7 = 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
- 2 instructions 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 50 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) in the absence of any stalls not related to memory?
c) With the given DRAM parameters, for a cache block size of 32 bytes, how wide must the bus be to fit miss to a DRAM into 50 cycles?
d) A design is proposed where a large hardware write buffer eliminates stalls for write misses. This design results in a smaller L2 cache, causing misses to DRAM to increase to 11% of L2 accesses. Calculate CPI in this new scenario and contrast with your answer to part (b).

Question 3 – Pipelines and ILP **[(4+6) + 3 + (5+3+1) = 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] * b[i]
        mult R5, R6, R6
        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 60 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:
      - 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) Would a 2-level branch predictor make a difference in this case, versus the simple branch predictor of part (a)(ii)? Explain.

c) We now explore the benefit of loop unrolling. Assume we always do an even number of iterations.
   i. Unroll the loop once (write out two iterations of the loop body) and ***only as necessary*** rename registers and reorder code to minimize dependences. Mark name dependences using the arrow notation used in the course.
   ii. How many stalls are removed in the unrolled version of the loop compared with that of part (a)(ii) – aggressive forwarding, no rewriting?
   iii. How would Tomasulo's algorithm help in generalising the example?

Question 4 – Multiprocessors **[(3 + 6) + 6 + 4 + 5 = 24 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 splits it into two threads. The first thread executes:

```
for (int i = 0; i < N; i+=2)
  a[i] = init (i); // init even entries
```

and the second executes:

```
for (int i = 1; i < N; i+=2)
  a[i] = init (i); // init odd entries
```

i. Timing runs of the code shows that the 2-thread code runs twice as long as running the initialization without an extra thread. Explain how the memory access pattern could explain this.

ii. Rewrite the two loops, changing the split of data between the two loops to fix the problem you identify in 3(a)(i) above and explain why the memory access pattern will likely result in faster execution with your revised code.

b) Explain how a ticket lock can reduce memory delays compared with a spinlock. In your answer consider the following scenarios:

A very short critical section that only has one instruction that updates a shared data structure.

A critical section in which numerous shared data structures are updated.

c) Two cores attempt to modify the same location in memory at the same time, with a store word (sw) instruction. Outline the steps taken including bus transactions from the start of the first sw instruction to the completion of the second sw instruction. ***Assume both cores at the start have the block in shared (S) state.***

Question 5 – Practical Question **[24 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 not a 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 maintaininclusion.
   i. Explain what multilevel inclusion is and why it is done.
   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
    - miss time 1 cycle (most of the miss time is from the levels below);
  - second level
    - 1MiB, 32B blocks
    - 2-way associative
    - hit time: 10 cycles
    - miss time: 10 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