

7 Introduction to Computer Architecture (swm11)

(a) What is Amdahl's law? [2 marks]

(b) For a 64-bit RISC-V pipelined processor, how could the following code be optimised to reduce execution time? Justify your answer. [4 marks]

```
lw t0, 0(a0)    # load 32-bits from address a0 into register t0
sw t0, 0(a1)    # store 32-bits to address a0 from register t0
lw t1, 4(a0)
sw t1, 4(a1)
lw t2, 8(a0)
sw t2, 8(a1)
lw t3, 12(a0)
sw t3, 12(a1)
```

(c) A new RISC-V R-type instruction `swap` is proposed to swap two registers. For example:

```
swap t0, t1
```

would swap the contents of registers `t0` and `t1`. What would be the challenges in implementing this proposed `swap` instruction? [3 marks]

(d) A counterproposal is to introduce `swap` as a pseudo instruction that unpacks to a short sequence of real instructions. What sequence of instructions could be generated that swaps two registers without using an additional register? [*Hint*: it involves using `xor`] [3 marks]

(e) RISC-V provides an atomic swap in memory operation `amoswapd` that takes an address held in a register (e.g. `a0`), a source register (e.g. `t0`), a destination register (e.g. `t1`). For these example register allocations, the operation performs an atomic read-modify-write where the value at address `a0` is stored in register `t1` and the value in register `t0` is written to address `a0`. How could the same operation be achieved using load-reserved (`lr`) and store-conditional (`sc`) instructions instead of using `amoswapd`? Explain your code by commenting it. [4 marks]

(f) If a computer system uses the MSI cache coherence protocol, and if multiple processor cores are reading shared data at address *a* at the same time, what happens when one processor core writes a word to address *a*? [4 marks]