Honours Architecture 2022 Exercises

19 June 2022

Note that cross-references are to pages and sections in the notes.

1 Introduction

1. Does high-level language-oriented design seem like a good idea to you? Consider historical examples and compare with current possibilities.

2. If Intel has features in their current instruction set architecture that made sense in the 1980s and less so now, why is Intel still successful?

3. Contrast the RISC strategy with any design that differs from the core concepts: how much do these differences make it hard to achieve:

   (a) Speed

   (b) Low-energy design?

4. Parallel architectures are becoming mainstream after being confined to narrow niches like supercomputers. Explain why.

5. How does warehouse-scale computing change design options for computer architects?

6. If you had to design a new instruction set from scratch, what factors would you take into account and how would the likely area of application influence design choices?

2 Principles of Instruction Set Design

1. Look for instruction set manuals for MIPS, RISC-V and Intel-64 (the current architecture).

   (a) How much similarity do you find between them?

   (b) Could you learn one of the others easily if you knew just one other of these instruction sets? If so, which would you start from?

   (c) Compare RISC-V and MIPS: to what extent is it true that the RISC-V designers learnt from past errors, as claimed?

2. RISC-V always uses the R_d field as a destination for instructions that require one register to be updated. Compare with MIPS and comment on why RISC-V is designed this way.

3. RISC-V in some instruction formats has immediate operands that are split into different parts of the instruction word. Explain why the designers made this choice and discuss whether this has positive or negative consequences for performance.

4. Explain why condition codes are a problem for scaling up to more aggressive pipelines.

5. ARM is branching out from the mobile and embedded markets to high-performance computation. Discuss problems they may run into with this move and any advantages they may have over established competitors.

6. The VAX had a single instruction to set up the call stack yet a sequence of simpler instructions ran 20% faster [Patterson 1985]. Explain what we can learn from this example.

3 Memory and Quantitative Design

1. You are investigating a new page table organization that reduces page faults by 1% of total memory references; to implement this you will lose hardware support for TLB misses. Assume:

   - a page fault takes 1-million cycles to handle

- cycles for handling a TLB miss are:
  - 50 with hardware support
  - 100 without hardware support
- the only change in number of misses is the number of page faults

(a) What is the net speed gain (or loss) of the new page table design if 0.1% of TLB references are misses?

(b) Is this a useful calculation for a real system? Consider what a real system does on a page fault.

2. Assume we have a machine that in the absence of misses executes on average 2 instructions per cycle. Such a machine would have a higher peak throughput but would be limited by other limits on ILP such as branches.

   (a) Redo the calculation of the case study (3.3.1) under this assumption.

   (b) Now allow for a non-blocking cache that can avoid a stall on average for 5 instructions before having to stall.

   (c) Is a non-blocking cache a useful improvement given the miss cost of this example? When might change your answer?

3. Look up memory specifications on a current design. Have issues moved on much from the references cited in this chapter?

4 Pipelines and ILP

In all examples where code is required, use the following:

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

Translated into:

```
            addi R1,R0,0   # i = 0
            addi R4,R2,0   # copy base address of a
            addi R6,R3,0   # copy base address of b
            j forTest001
forBody001: lw R5,0(R4)    # get value of a[i]
```

```
            lw R7,0(R6)    # get value of b[i]
            add R7,R7,R5   # calculate a[i]+b[i]
            sw R7,0(R6)    # update b[i]
forNext001: addi R4,R4,4   # increment address of element of a
            addi R6,R6,4   # increment address of element of b
            addi R1,R1,1   # increment loop count
forTest001: blt R1, R8, forBody001
```

1. Do a pipeline timing diagram of the given example under the following assumptions:
   (a) No stalls.
   (b) Maximum stalls, based on when values are available (as in Section 4.1) and without any forwarding.
   (c) Stalls without maximum forwarding.
   (d) Unroll the loop once (two copies) and show how to reduce stalls by reordering and (if necessary) register renaming.
   (e) Can this example benefit from Tomasulo's algorithm? Explain.
2. Rework the pipeline timing diagram of the given example under the following assumptions:
   (a) We can issue any pair of instructions, subject only to dependences; assume the most aggressive achievable model of forwarding.
   (b) Add now the ability to do register renaming; assume the hardware is smart enough to rename any register after a write to it and no limit to the number of virtual registers (do you run out of paper?).
3. Now go back to the simpler dual-issue pipeline without register renaming and evaluate the effect of a simple 2-bit branch predictor.
   (a) Will a more sophisticated scheme like a 2-level adaptive scheme make a difference here? Explain.
   (b) Now assume we have an `if` statement in the loop that only does the assignment on odd values of `i`. Write out the assembly language for this case (you may fudge some details as long as the branches are plausible). Will a more sophisticated branch predictor help in this case? Explain.

4. Assume we have a floating-point pipeline in which a multiply takes 2 cycles and a divide 4 cycles (both in the execution stage; other stages are the same as for integer instructions). Explain how these instructions introduce new types of hazard not present in the integer pipeline and why they present problems for interrupts.

5. Look up how precise exceptions or precise interrupts are handled in multiple-issue implementations.

6. VLIW was based on the premise that a compiler technique, *trace scheduling*, could expose significant ILP. Look up trace scheduling and analyse its strengths and weaknesses.

7. The Pentium Pro introduced a concept of cracking complex instructions into simpler micro-operations, or μ-ops[1]. These μ-ops could be pipelined much more easily than the native instruction set.

   (a) Digital Equipment Corporation tried a similar idea with the VAX [Clark 1987]. See if you can find out what happened to that attempt. Can you still buy a VAX machine today?

   (b) Explain how this feature helped to bridge the gap between RISC and CISC and why Intel could not have done this with earlier designs.

5 Multiprocessors

1. Use the latencies in Table 5.1, and the timing illustrated in Figure 5.4. You can assume every instruction can complete in one clock if fully pipelined, a store modifies memory in the MEM stage (4th stage of the pipeline), and an invalidate requires the latency of a snoop for the invalidating core if the bus is not occupied.

   (a) Calculate the total time it takes before core 0 manages to complete the illustrated store instruction.

   (b) Now assume that core 1 acquires the block as soon as possible after core 0's store completes. Calculate the total time from the start of Figure 5.4 until core 1 completes its first store, assuming that core 0 continues with the loop with minimal stalls.

[1] Pronounced "mu-ops" in deference to the pronunciation of the Greek letter μ.

(c) Calculate the total time it takes for 10 iterations of the loop for both cores, stating assumptions about timing of competing events.

(d) Why is it a reasonable assumption that an invalidate requires a relatively short stall (here, 2 cycles), not a longer delay, e.g., the 27-cycle delay required for a snoop?

2. Will the M-lock as described on page 98 work as you expect if N tasks enter it and leave it, then try to re-enter at a later stage? Hint: what will the global pointer have as its value, and what will be stored at that location?

3. Spinlocks are often used as a core primitive to implement more complex, scalable synchronization techniques like semaphores that put a process or thread to sleep and wake it when it reaches the head of a queue. Are spinlocks a reasonable choice in that scenario, or would you still look for a more scalable option like a ticket lock?

4. Is a test and set instruction superior or inferior to an atomic swap? Explain, considering the design philosophy of a RISC ISA.

6 GPUs

1. You can find some specifications of the Cray T-90 here: http://www.netlib.org/utk/papers/advanced-computers.0/crayv.html. Based on numbers you find:

   (a) What is the maximum number of loads and stores possible in one clock cycle?

   (b) In an 8-processor configuration, with the maximum possible numbers of loads and stores, how many banks of 15ns RAM are required to keep up with demand? Assume each load or store can be divided into as many banks as are needed.

   (c) The top model of this range, the T932, had up to 32 processors and a slightly faster clock speed than that in the above reference (2.167ns). It had 1024 banks of RAM, and the RAM was upgraded from a 15ns cycle time to twice as fast. Was this upgrade necessary?

2. Look up details of the AltiVec instruction set. How does it compare with the other architecture styles we've examined? Is it a reasonable fit to the RISC philosophy?

3. Find a detailed example of NVIDIA PTX code. Explain how parallelism is achieved in the example.

4. If you were designing the Intel AVX instructions from scratch, rather than as an extension of previous designs, how different would your approach be?

7 Warehouse-Scale Computing

Note that the standard abbreviation for byte is "B" and for bit is "b". Recall that binary prefixes have an "i" added to differentiate them from decimal multiples (e.g., Ki means 2^{10}, whereas G means 10^9).

1. With an average year of 8766 hours, how many hours of downtime does four nines of availability represent?

2. You would like to offer four nines of availability on a 2,500 server configuration. Which of the following gets you closest to this goal (starting from the base of the figures in Table 7.3, which gave us 2 nines of availability):

   (a) replacing the hard drives by solid-state drives (SSDs), reducing the expected number of failures to 10 a year

   (b) replacing the RAM with DRAMs with error checking and correction (ECC), reducing the number of uncorrectable failures to 20 per year

   (c) running a more robust operating system that reduces failure to 250 per year

   Given the above, comment on Google's actual approach, which is to tolerate failures.

3. Google is a significant investor in clean energy technologies, and Apple has reportedly commissioned one of the largest solar energy facilities not owned by a power utility. Discuss why this may be the case.

4. Use the Intel Nehalem latencies from Table 5.1, with the network latency in this chapter (Table 7.4):

(a) Assume a uniprocessor task running on a local CPU, and redo the calculation for the fraction of remote accesses that double the average memory access time, assuming global miss rates from L1 or 10%, from L2 of 1% and from L3 of 0.1%.

(b) Now redo the calculation assuming 10% of L2 misses incur snoop latency (implying a multiprocessor task). How does this change your answer?

(c) Adjust your answers by doubling network latency to allow for the round trip, and adding 10% to allow for network congestion. How much of a difference does this adjustment make?

(d) Assume the network latency adds for each switch. How much difference will it make if you have to go through 3 switches to obtain a data item?

(e) In general terms, discuss the performance hit going to a remote machine rather than local accesses, even with multiprocessor overheads.

5. Gbit ethernet switches are commodity technology. Let's consider whether 10Gbit switches are worth considering. Assume switching latency is the same, and the only change is the transfer rate.

(a) Ignoring switching latency and packet overheads, how long does it take to move a packet of 4KiB at 1Gb/s?

(b) Ignoring switching latency and packet overheads, how long does it take to move a packet of 4KiB at 10Gb/s?

(c) How big a difference is there in these two numbers if we add $10\mu s$ switching latency?

(d) MapReduce operates on relatively large chunks of data. Relate switching latency to transfer time in this example, and explain why MapReduce is generally used that way.

(e) If you were designing a new WSC facility would you consider 10Gb ethernet switches? Explain.

6. It's been a while since I updated references. Choose a part of this chapter that you find interesting and look up more recent material.

(a) Have the fundamentals shifted or are the issues the same, just on a bigger scale?

(b) How much different is Cloud Dataflow from MapReduce? Do these differences have much of an effect on the ideal hardware architecture?

8 New Developments

1. It is becoming increasingly useful to package dies (chips) in 3 dimensions as well as to build 3-dimensional logic structures within a die.

   (a) Discuss in general terms, making clear that you understand the difference between the two concepts.

   (b) Compare 3D Xpoint RAM and HMC RAM in terms of the way they are constructed.

   (c) Which category does picoserver fit into? Discuss whether this is a good idea.

2. If nonvolatile RAM (NVRAM) could be made almost as fast as DRAM

   (a) Explain how this could alter the memory hierarchy.

   (b) Explain potential benfits for long-running computations.

   (c) Would this be a win for mobile devices? Explain.

3. Are deep learning architectures likely to win wide acceptance? Consider the list of points that have worked against specialist processors on page 136.

4. Are FPGAs likely to win wide acceptance? Consider the list of points that have worked against specialist processors on page 137.

5. Examples in this chapter mainly focus on Google. Look up what other big players in WSC are up to, such as Amazon. Are the only working on software, or do they have their own hardware projects?

References

Clark, D. W. (1987). Pipelining and performance in the VAX 8800 processor. *ACM SIGARCH Computer Architecture News*, 15(5):173–177.

Patterson, D. A. (1985). Reduced instruction set computers. *Communications of the ACM*, 28(1):8–21.