Ellie Makin (erm67)

Computer Design Supervision 2

2017 Paper 5 Question 2

a. For one thing, a benchmark is not a typical program - they are specifically designed to test one particular aspect of the situations a processor may need to handle. Secondly, many processor manufacturers have been known to tune their architectures in order to maximise performance on the most common benchmarks for marketing purposes, which means that the results of a benchmark test may not be indicative of how well the given processor performs on a more real-world task.

b. The amount performed by one instruction varies between ISAs: something that may take only one instruction on x86 may consist of two or three on RISC-V, so RISC-V would require a higher IPS in order to match the performance of x86.

c. It may be the case that a CPU has been optimised to perform certain tasks over others, so that at peak performance, it may be performing several of these optimised tasks, but in average operation, it is much slower. This would cause temporary spikes in the performance, without significantly raising the observed performance.

d. If the range of tasks that a computer needs to perform is reduced, it permits optimisations in order to perform the remaining tasks faster. To take a simple example, a computer that only ever needs to perform arithmetic operations can be optimised to do that much faster than one which needs to handle other instruction types too. For this reason, it is impossible to make a general purpose computer which will perform ideally on all tasks. This is sometimes called the “Turing Tax”.

e. Adding more pipeline stages may increase the number of instructions that can be performed at once , but it also increases the total execution time for each instruction. This means that if a program contains many branch instructions, and the pipeline has to be emptied frequently, the next instruction will take many clock cycles to reach the end of the pipeline again each time. The time taken to recover from a branch is proportional to the length of the pipeline.

2016 Paper 5 Question 2

a. If one instruction tries to fetch from a register that has not yet been written back to by an earlier instruction in the pipeline, this causes a data hazard. If the processor fetches instructions while there is a branch instruction in the pipeline, and then continues to execute these before the branch has been completed, this causes a control hazard. These cases must be handled carefully so that they do not occur.

b. The programming model for ISAs doesn’t include aspects of the processor that may be implementation dependent. In the case of RISC-V, the way control hazards are handled is not

included in the specification, and may be handled differently by different processor architectures, so this is not included in the programming model.

c. For pipeline A, all of the reading and writing of registers is handled in the same stage, so there cannot be any data hazards, since the register accesses cannot be interleaved (there may still be control hazards).

   Pipeline B has a read-after-write hazard. To solve this, we can forward the result from writeback into the execute stage, in the case that a register is under hazard.

   Instead of forwarding from execute, pipeline C can instead forward the result of the memory access into the execute stage. However, in this case, if a hazard is detected in the decode stage, a bubble must be introduced to the execute stage, to wait for the result of the memory access to be available.

d. An exception may introduce a control hazard if an exception from hardware occurs before a branch instruction has completed, since then the PC may not have been updated yet. This can be quite complicated to handle properly, if we want to resume the program after the exception.

e. An interrupt is less likely to cause a control hazard, since in this case we can finish processing the instructions in the pipeline before handling the interrupt.

2009 Paper 5 Question 3

a. In a machine that guarantees sequential execution, memory access latency can delay the execution of the entire program, if the result of the memory access is needed soon afterward. If the requested memory is not in the caches, it can take hundreds of clock cycles for the result to be returned, which is a significant delay.

b. Caches assume that memory accesses by the same program will tend to be of a relatively contiguous section of memory, and that subsequent memory accesses will likely be to sequential memory addresses. Caches take advantage of this by storing a range of memory around the most recent memory access, to increase the chance of a cache hit on the next access.

c. In a direct-mapped cache, there is no choice of replacement policy, since each set corresponds to only one cache line. For a set-associative cache, there are many different options for replacement policy. For low associativities, least recently used may be a good option, otherwise random is not a bad choice.

d. One option is to keep a *write buffer* which contains the data to be written back to the cache and memory, which lets the processor continue while the write happens. Another way is to not write the data back to memory immediately, and instead only write to the cache, and update the memory later by keeping track of which cache lines we have modified, and writing them back to memory when they are replaced.

Questions from the recommended text

4.17

The time taken to execute a single instruction on a k-stage pipeline is k cycles. Increasing the number of instructions by 1 only adds a single clock cycle, since the next instruction can start processing only one clock cycle after the previous one. This means that the total time for n instructions to execute on a k-stage pipeline is $k + n - 1$

4.20

```
addi x11, x12, 5  # takes 3 clock for result of write to be available in execute
nop
nop
add  x13, x11, x12
addi x14, x11, 15
nop
add  x15, x13, x12
```

5.5

1. 8 words
2. 32 blocks
3. The total bits required for one cache line is 64 label bits, and $8 \times 64 = 512$ data bits. This gives a ratio of $(512 + 64)/512 = 9/8$.