**EXERCISE 1**

Let Iu = Number of instructions in unoptimized version

Let Io = Number of instructions in optimized version

Let Ru = Clock rate of unoptimized

Let Ro = Clock rate of optimized

CPI = 1 for all instructions

Only loads and stores are affected therefore; Io = Iu – (1/3) × (0.3 × Iu) = Iu – 0.1Iu = 0.9Iu

Execution time =

Ro = 0.95% of Ru since its clock rate is 5% lower than Ru = 0.95Ru

= = = = 0.947

The result above states that optimized version takes 94.7% of unoptimized version’s execution time

Speedup percentage improvement = { – 1 } × 100% = 5.3%

Hence the optimized version is faster by 5.3%

**EXERCISE 2 PART 1**

Since CPI is constant and there is increase in clock period by 5%, we reduce the instruction count;

Let X = Fraction of loads eliminated

New instruction count(Ī) = I – X(0.228I) = I(1-0.228X)

To maintain its performance:

Therefore to maintain the same performance, at least 21.9% of the load must be eliminated.

**EXERCISE 2 PART 2**

If there are dependencies on Rx in later instructions, it cannot be replaced directly;

LOAD Rx, 0(Ry)

ADD Rz, Rx, Rw

An example showing a MUL instruction which depends on Rx meaning LOAD cannot be replaced;

LOAD Rx, 0(Ry)

MUL Rp, Rx, Rq

ADD Rz, Rx, Rw

**DISCUSSION 1**

RISC architecture has evolved overtime in where the modern RISC ISA can contain almost as many instructions as the older CISC. Because the point of RISC is to have fewer instructions, people may wonder whether modern RISC processors are no longer the normal kind of RISC. I think modern RISC processors are still like the normal RISC. This is because of the underlying design principles that persists even if they have grown in terms of size and complexity incorporating features like a larger number of instructions. The definition of simplicity has shifted since it is not only about the number of instructions anymore but also how those instructions are handled.

For example features like these:

1. Fixed and simple instruction encoding – This is where each instruction has this predictable format making it easier to decode simplifying all the processes that needs to happen making it more streamlined.
2. Register-oriented instructions – This is where mostly operations happen on data that is stored on registers for faster access.

**DISCUSSION 2**

Processors like Intel x86 ISA that have been traditionally considered CISC are using RISC principles in their implementation. It is like speaking CISC on the outside but thinking RISC on the inside. In short, they are taking advantage of RISC efficiencies while maintaining their CISC compatibility. This brings about the dilemma of evaluating ISA complexity where it is not sure on what to talk about if it is the microarchitecture like the very intricate hardware or the software interface where the compiler interacts with the processors.

The implications are if we prioritize the software interface, then compatibility with existing software becomes very important. When we think about all the programs written by Intel x86 architecture over the decades, it is a massive ecosystem so if we were to switch to a completely different ISA we would have to re-write or re-compile the software which will be a huge undertaking. From this point of view, there is need to maintain the CISC architecture even if the underlying hardware is taking up more RISC-like approach.

If we prioritize the processor’s microarchitecture, the use of microinstructions in modern processors becomes very important. The microprocessors can be executed way more quickly by the processor’s hardware. Even if the software sees this like a complex CISC instruction, the processor is handling it as a series of more efficient RISC-like operations. This is a type of hybrid approach which is used by both RISC and CISC processors. This approach allows processors to leverage the compatibility and flexibility of CISC with the speed and efficiency of RISC.

One may wonder if this makes Intel x86 processors RISC, CISC or a hybrid. I think it really depends on how you define the terms and concepts because the rigid categories of RISC and CISC may not be relevant as they once were. Just like the field of technology, we are ever evolving especially when driven by the relentless pursuit of performance.