# Computer Systems 414

# Practical 4A

14 April 2021

Charl du Toit: 21708886

George Soule: 21602204

Walt Deyzel: 21750793

# Results

## Raspberry Pi

| **source file** | **primes.c** | | | **sprimes.c** | | | **card.cpp** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **real** | **user** | **sys** | **real** | **user** | **sys** | **real** | **user** | **sys** |
| **gcc -o primes <source file>** | 12.827s | 12.8s | 0.0s | 0.112s | 0.1s | 0.01s | - | - | - |
| **gcc -o primes <source file> -01** | 10.883s | 10.860s | 0.0s | 0.092s | 0.07s | 0.01s | - | - | - |
| **gcc -o primes <source file> -02** | 12.662s | 12.610s | 0.01s | 0.089s | 0.07s | 0.01s | - | - | - |
| **gcc -o card <source file> -03** | - | - | - | - | - | - | 2m10.174s | 2m6.69s | 3.25s |

# Explanation

Using optimization -O1, -O2 with -O1 the user time (time code to execute) decreased by 2 seconds form the real time. But when optimization -O2 is used it is as if no optimization was used. My deduction is because over optimization is not necessarily a good thing thus resulting is a longer user and real time. But you cannot have a real time less than non-optimization that is why the times are basically the same. The same can be observed when sprimes.c runs. But this time when -O2 optimization is used it stagnates at the efficiency of -O1. Why this is the case I am not sure. My assumption would be that you can only optimize to a point before it makes no more difference. Where as before we optimized too much and it had a negative impact on the time.  Sys time took 0.1 seconds with sprimes.c

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

**Q4-2 What role does the HAL play in the platform?**

HAL stands for hardware abstract layer. It provides an interface to the system peripherals by concealing a different architecture than the OS the system is using. (https://www.techopedia.com/definition/4288/hardware-abstraction-layer-hal#:~:text=The%20main%20purpose%20of%20a,interface%20to%20the%20system%20peripherals.)

**Q4-4 Describe the role of these signals in a bus:**

- **R/W:**
  Read/Write means the CPU is writing or reading from the device or memory by setting the signal line R or W to a certain value.
- **Data ready:**
  When signal is high the CPU knows the data is ready to be accessed.
- **Clock**
  Internal clock of the CPU that times all the fetch instructions for example.

**Q4-14 Draw timing diagrams for:**

- **A device becoming bus master.**
  There is a handshake the bus does with a slave before the slave becomes a master. The CPU will send a signal over the bus with a specific code to talk to the slave. The slave will send back a acknowledge bit then the slave and the cpu is connected and the slave can become the master.
- **The device returning control of the bus to the CPU**

**Q4-28 Assume an A/D converter is supplying samples at 44.1 kHz:**

- **How much time is available per sample for CPU operations?**
  The period between samples is 1/44.1k = 22.68µs. The CPU only has so much time to do calculations
- **If the interrupt handler executes 100 instructions obtaining the sample and passing it onto the application routine, how many instructions can be executed on a 20-MHz RISC processor that executes 1 instruction per cycle?**
  20M/100 = 200k

**Q8-4 You are designing an embedded system using an Intel Xeon as a host. Does it make sense to add an accelerator to implement the function z = ax + by + c? Explain.**

No. The equation is linear and would not benefit from a accelerator.