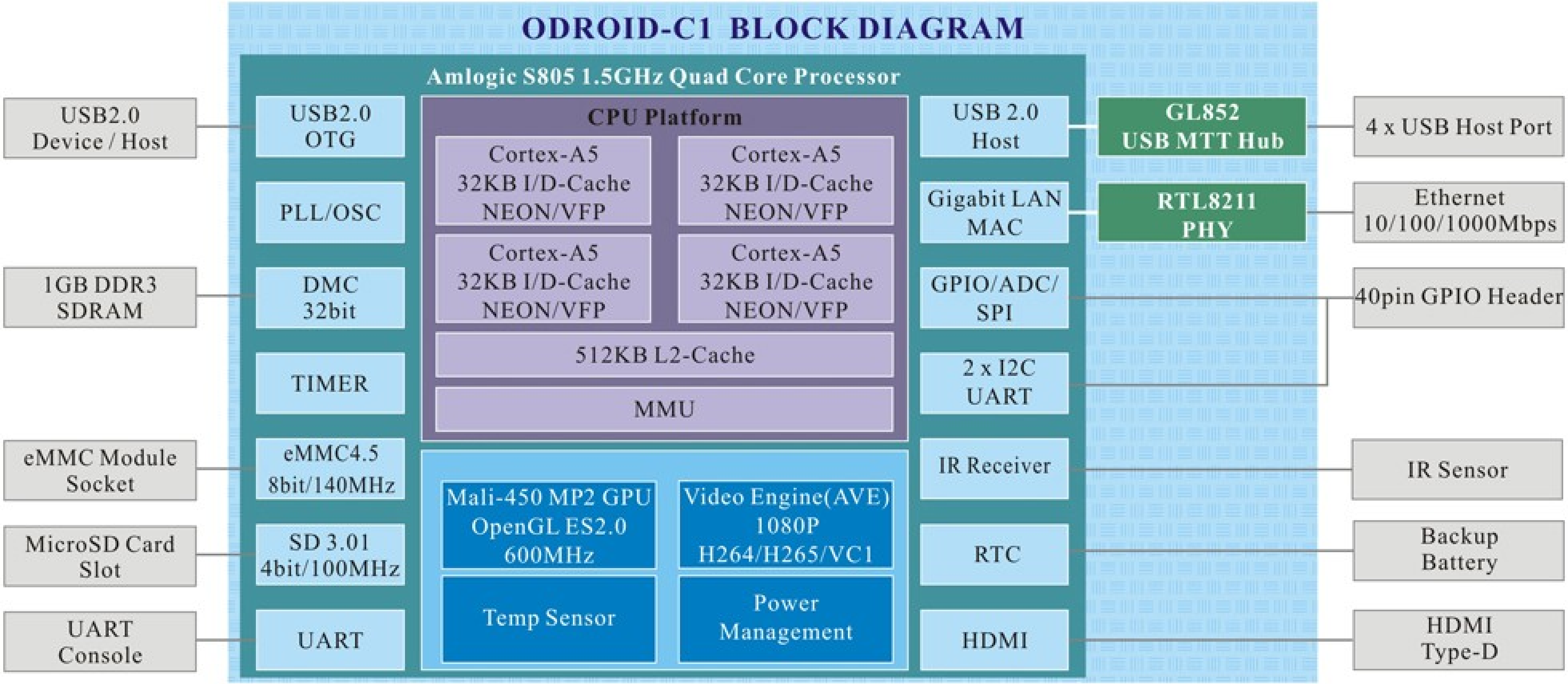
Exercise 4

Cordoví, Hörmandinger, Nieleck, Strelchenko

Used Platform

- **Odroid C1**
  - Cortex-A5 (ARMv7)
  - 4 Kernel @ 1.5 GHz
  - NEON SIMD
    - 128 bit vector (dual 64 bit)
    - 16 x 8 bit integer operations in parallel

Odroid-C1

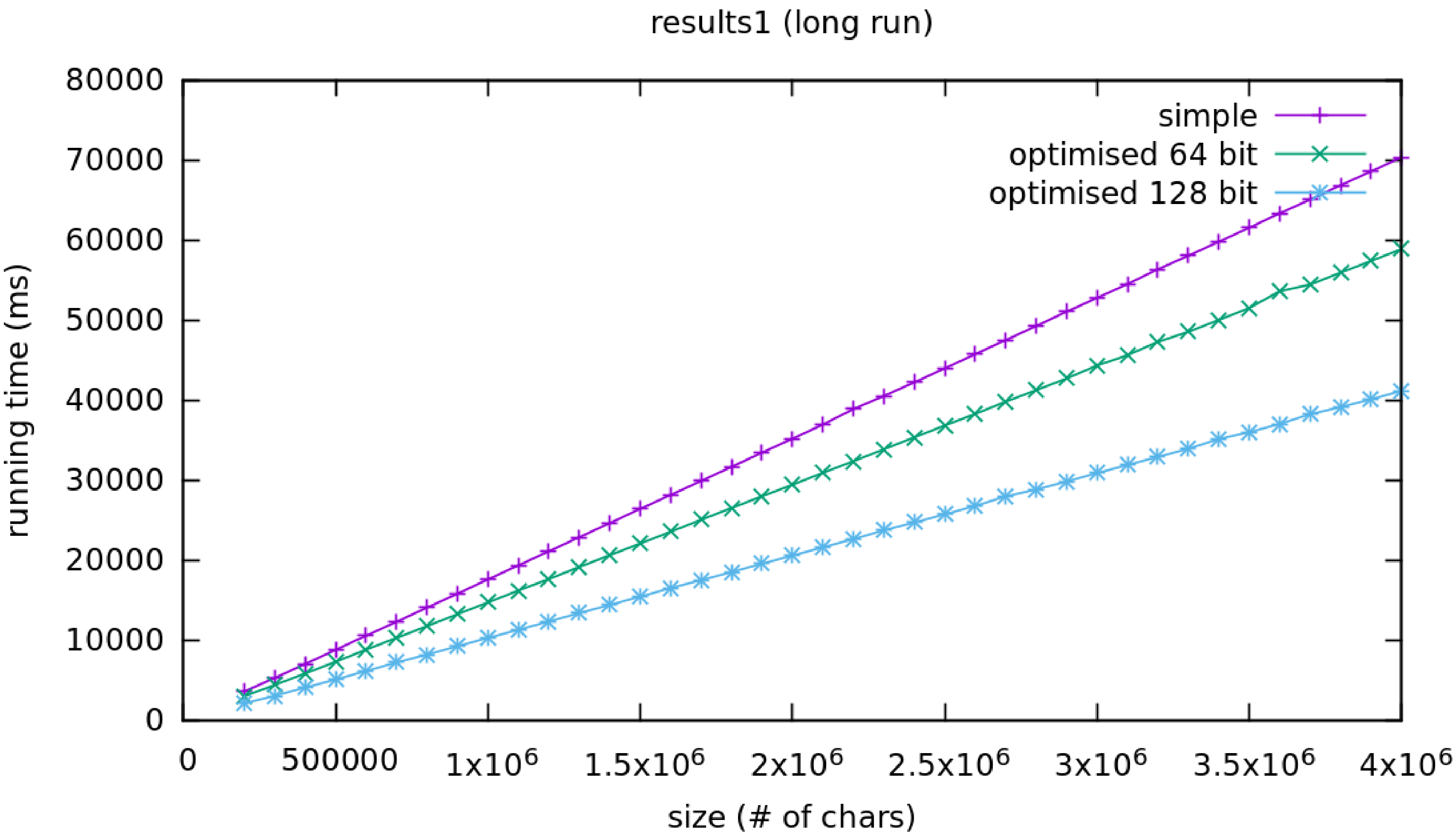


Quelle: Hardkernel co., Ltd, http://dn.odroid.com/homebackup/201411241452444193.jpg

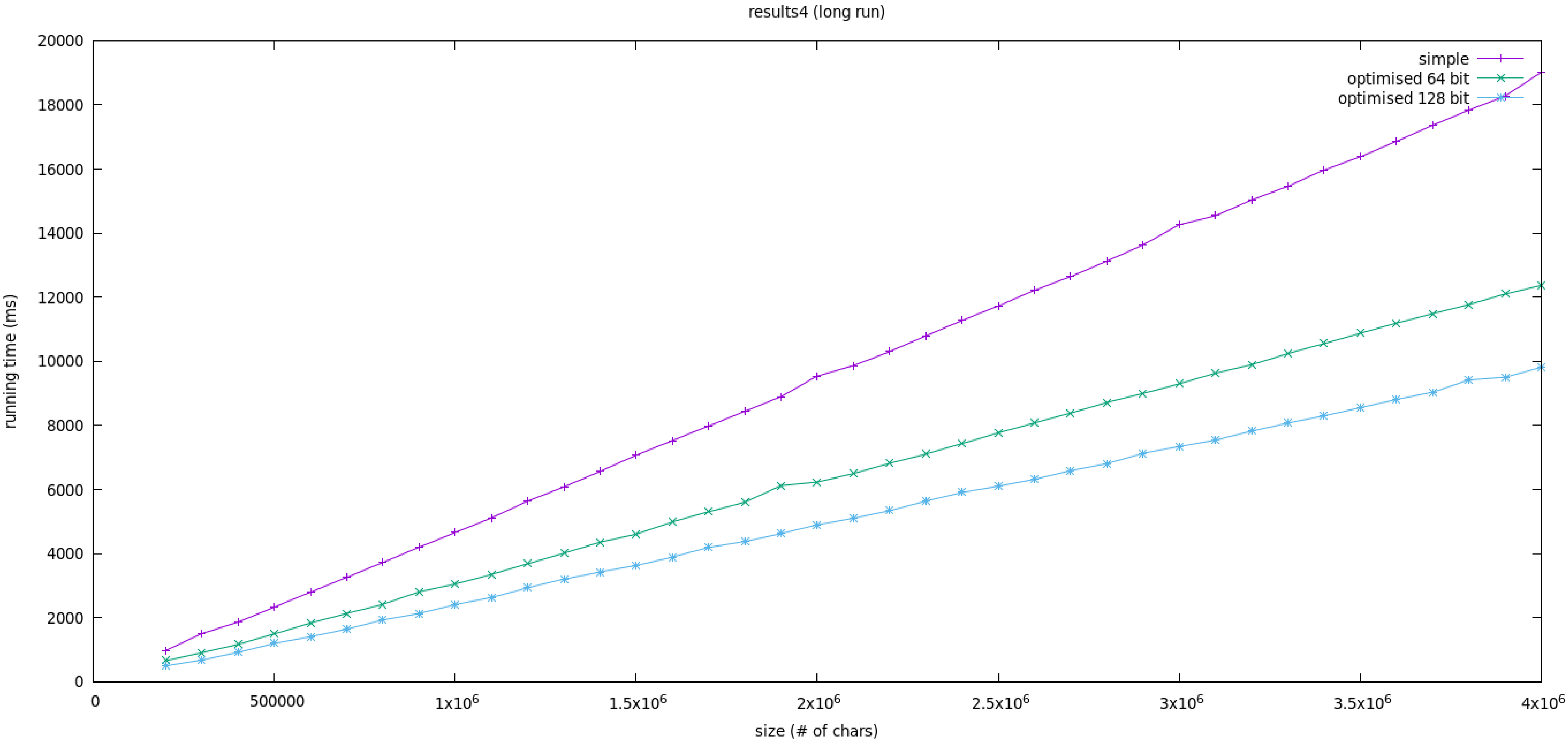
The Code

```
static void toupper_optimised_neon128(char *text) {

    uint8x16_t cmp_v = vdupq_n_u8(0x60);      // Create 128bit vector (16 x int8), fill with scalar
    uint8x16_t and_v = vdupq_n_u8(0x20);
    uint8x16_t str_v, tmp_v;
    int length = strlen(text); int modulus = length % 16;

    int i, j;
    for (i = 0; i < length - modulus; i += 16) {
      str_v = vld1q_u8(&text[i]); // load chunks of 16 characters of text into the vector register

      tmp_v = vcgtq_u8(str_v, cmp_v);
      tmp_v = vandq_u8(tmp_v, and_v);
      str_v = vsubq_u8(str_v, tmp_v);

      vst1q_u8(&text[i], str_v);  // store chunks of 16 characters back to the text array
    }
...
}
```

Results – No optimization (-O0)



Results – Optimization (-O3)


results4 (long run)
simple
optimised 64 bit
optimised 128 bit
running time (ms)
1x10^6
1.5x10^6
2x10^6
2.5x10^6
3x10^6
3.5x10^6
4x10^6
size (# of chars)