Exercise sheet 6 – Processor architecture

Goals:

- Registers
- Addressing modes

Exercise 6.1: Addressing modes (theoretical)

(a) Which addressing modes can be used for direct realisation of stack-operations on a CISC architecture (Freescale ColdFire, 32 bit architecture)? Are there any alternatives if those addressing modes are not available? Explain this by pushing the content of the D0 register to the stack; after that, pop the stack content to the D1 register. *Hint: You may use some pseudo-code (assembler) to express your idea.*

(b) How can a CISC architecture (Freescale ColdFire) support array-accesses? Find and use an appropriate addressing mode. Consider a 32 bit architecture. Use given values to describe your idea. *Hint: You may write some pseudo-code (assembler) and draw a sketch.*

- access to x[i] (element i of array x, x contains integer data)
- x starts at memory address 0x10000
- i = 20
- use the registers A2 and D3

Exercise 6.2: Understanding a concrete Intel x86/64 instruction (theoretical)

(a) Try to understand the `XCHG` (Exchange Register/Memory with Register) assembler instruction. Here are useful links:

- `https://www.felixcloutier.com/x86/xchg`
- `https://www.amd.com/system/files/TechDocs/24594.pdf` (page 360)

(b) Which addressing modes does the `XCHG` instruction support?

Exercise 6.3: Use a concrete Intel x86/64 instruction (coding)

Given is the same endianness example („Endianness with integer (coding)“) as from the last exercise: A *big-endian* system program—the Java runtime environment—that transfers data via a file to a little-endian system C program.
Now, we want to use a single assembler instruction to perform the swap operation.

(a) Update the `CA_exercises` repository with `git pull`.

(b) Change into the directory `CA_exercises/sheet_06_registers/Endianness/C_LE_asm_swap`

(c) Inspect, build, and run the given C program.

(d) Analyse the output of the C program. What has happened? What could be the cause of this?

(e) Fix the problem in the C program, following the *TODOs*. *Hint: use the* `XCHG reg/mem8, reg8` *variant of the* `XCHG` *instruction to perform the swap.*

(f) Build and run the C program again. Is the problem now solved?