Virtual Machines

Jochem Arends

Introduction

Virtual Machine are programs to emulate hardware.

Little Computer 3

Little Computer 3 (LC-3) is an educational computer architecuture developed at the University of Texas at Austin. The LC-3 has a simple yet versatile instruction set.

Registers

Registers are a type of computer memory that are used by instructions. The LC-3 has eight 16-bit general purpose registers, these are called `R0-R7`. There's also a 16-bit program counter which is ofter referred to as `PC`.

Instruction Encoding

The four most significant bits of an intruction form the opcode. The LC-3 architecture knows 15 distinct operations, which are listed below in the form of an `enum class`, but can also be found in the ISA.

opcodes.h

```
#ifndef OPCODES_H
#define OPCODES_H

namespace lc3 {
    enum class opcode {
        ADD  = 0b0001,
        AND  = 0b0101,
        BR   = 0b0000,
        JMP  = 0b1100,
        JSR  = 0b0100,
        LD   = 0b0010,
        LDI  = 0b1010,
        LDR  = 0b0110,
        LEA  = 0b1110,
        NOT  = 0b1001,
        RTI  = 0b1000,
        ST   = 0b0011,
        STI  = 0b1011,
        STR  = 0b0111,
        TRAP = 0b1111,
    };
}

#endif
```

Instruction Decoding

Instructions consist of named bit sequences that occupy fixed sequence of bits.

Term	**Bits**
Destination Register (`DR`)	11..9
Source Register (`SR`)	11..9
Source Register 1 (`SR1`)	8..6
Source Register 2 (`SR2`)	2..0
Base Register (`BaseR`)	8..6
5-bit Immediate Value (`imm5`)	4..0
6-bit Offset Value (`offset6`)	5..0
9-bit PC Offset Value (`PCoffset9`)	8..0
11-bit PC Offset Value (`PCoffset11`)	10..0
8-bit Trap Vector Value (`trapvect8`)	7..0

types.h

```cpp
#ifndef TYPES_H
#define TYPES_H

#include <cstdint>

namespace lc3 {
    using word = std::uint16_t;
    using sword = std::int16_t;

    template<typename T>
    concept type = requires {
        T::width;
        T::index;
    };

    struct DR {
        static constexpr word width{3};
        static constexpr word index{9};
    };

    struct SR {
        static constexpr word width{3};
        static constexpr word index{9};
    };

    /* etc.. */
}

#endif
```

encoding.h

```
#ifndef ENCODING_H
#define ENCODING_H

#include <lc3/types.h>
#include <array>
#include <concepts>

namespace lc3 {
    template<std::size_t N>
    word bit_at(std::integral auto bin) {
        return (bin >> N) & 1;
    }

    template<type T>
    word extract(word bin) {
        auto a = static_cast<word>(bin >> T::index);
        return zero_extend<T>(a);
    }

    template<type... Ts>
    auto decode(word bin) {
        return std::array<word, sizeof... (Ts)>{(extract<Ts>(bin))...};
    }
}

#endif
```

Memory

The LC-3 architecture uses 16-bit wide addresses. For now we just use an `std::array` for memory, but a type alias is used so it's easier to change in the future, such as adding memmory mapped IO.

memory.h

```
#ifndef MEMORY_H
#define MEMORY_H

#include <lc3/types.h>
#include <array>
#include <limits>

namespace lc3 {
    using memory = std::array<sword, 0x10000>;
}

#endif
```

The CPU

cpu.h

```
#ifndef CPU_H
#define CPU_H

#include <lc3/memory.h>
#include <lc3/opcodes.h>
#include <lc3/types.h>
#include <algorithm>
#include <ranges>

namespace lc3 {
    class cpu {
    public:
        void load(const std::ranges::input_range auto& bin, word offset = 0x0000) {
            std::ranges::copy(bin, std::next(m_memory.begin(), offset));
        }

        void execute(word bin);

        void run();
    private:
        template<opcode Opcode>
        void perform(word bin);

        void setcc(sword value);

        word m_pc{0x3000};

        sword m_regs[8]{};

        memory m_memory{};

        bool m_halted{};

        struct {
            unsigned int n : 1;
            unsigned int z : 1;
            unsigned int p : 1;
        } m_condition{0, 1, 0};
    };
}

#endif
```