1 Week 1/2

1.1 Introduction to RISC-V

1.1.1 Binary Representation

All information passed through a processor can be expressed as a sequence of binary digits.

Conversion between a base 2 number and a base 10 number can be done rather efficiently:

$$1001\,0111_2 = 151_{10} \tag{1}$$

	1	$\times 1$	1
	1	$\times 2$	2
	1	$\times 4$	4
	0	$\times 8$	0
	1	$\times 16$	16
	0	$\times 32$	0
	0	$\times 64$	0
+	1	$\times 128$	128
			151

1.1.2 Hexadecimal Representation

We must also recall the hexadecimal representation of numbers that makes binary just a bit easier to look at.

1.1.3 Two's Complement

This is a convention for the expression of signed numbers, such that the most significant bit is negative instead of positive.

$$1001\,0111 = -105_{10} \tag{2}$$

	1	$\times 1$	1
	1	$\times 2$	2
	1	$\times 4$	4
	0	$\times 8$	0
	1	$\times 16$	16
	0	$\times 32$	0
	0	$\times 64$	0
+	1	$\times$**-128**	**-128**
			-105

1.2 Data in Memory

All elements in a computer's memory is labeled with a memory address and the units of data in a computer's memory is a byte, which is 8-bits and is represented (usually) as two hex characters $\texttt{0xFF}_{16} = 1111\,1111_2 = 255_{10}$.

Each memory address is represented by a 4-byte word (32-bits) which are arranged sequentially for the given process

Note 1 (Word) *It should be remembered that the 32-bit piece of memory that we have is called a 'word' in RISC-V 32 IF (which is the complete name of the instruction set we are dealing with). This is easy to remember since the processor acts on 32-bit instructions. More on this later.*

Note 2 (Memory Allocation) *Remember that the allocation of this memory is handled by the operating system, this is outside the scope of this class.*

1.3 Registers

Registers are given a name in assembly, we will come back to this later and by now you already know about them.

They are just a whole bunch of faster-to-access storage locations for data and hold 1 word worth of data.

1.3.1 Endian-ness

RISC-V is little-endian, which means that we read the contents of memory from right to left. This is nice for the processor but not nice for us.

What little endian means is like this:

$$\begin{array}{ccccccc} 0x & 12 & 34 & 56 & 78 \\ & & & & \uparrow \end{array}$$

The byte with the arrow pointing to it is the first byte in the sequence, we then read to the left by **byte** (not bit). So, for example, if we stored the word 'BEAN' into our 32-bit word, we would see this:

$$\begin{array}{ccccccc} 0x & N & A & E & B \\ & & & & \uparrow \end{array}$$

If we were to represent the word in ASCII.

Note 3 (ASCII Characters) *The ASCII alphabet contains 256 characters, thus each ASCII character is represented in 1 byte (8 bits).* ***Excersize****, what is the binary representation for the 256th ASCII letter.*

If we take a look at the RISC-V Emulator, RARS, we can get a feel for a larger example of the little endian nature of riscv.

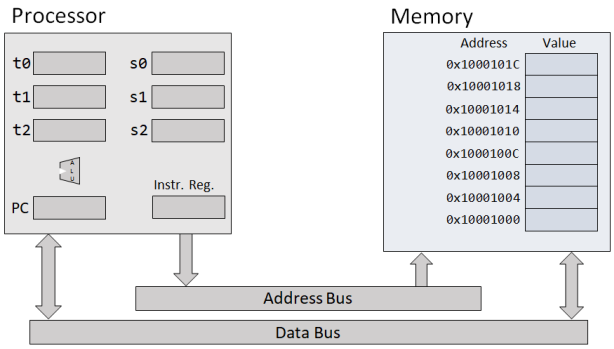

Figure 1: The organization of the processory and memory in a conventional computer detailing the contents of the processor registers and memory address-value map

1.4 Computer Organization

1.4.1 The Processor

The processor is the brain of the computer and is used to perform the computations that we give it, we will learn more about the performance of the processor in the **??** section. Some important things to note about the processor[1]:

- **Registers**: The fastest possible data access location.
- **I/D Cache**: The instruction and Data cache, they store information that is essentially "currently in use" (NEED VERIFICATION).
- **L0 cache**: Note that some computers do not have an L0 cache. This would be a cache to which the operational units have direct access. On the order of 128 bits wide.
- **L1 cache**: Typically the "on-chip split instruction and data caches" or "unified on-chip cache". These are fast caches that run at the chip clock speed and can be accessed within one cycle. Between 8kb and 32kb (lower case b is 'bits').
- **L2 cache**: An external cache that is much larger (256kb to 2Mb) that can be accessed within some multiple of the CPU clock speed.
- **L3+ cache**: There can be many levels of cache, they generally get progressively slower and larger.

There are also some special registers in the processor that do important things:

[1]These are mostly supplementary and begin diving into the nitty-gritty of how a processor is architected. I grabbed most of my information from here.

- **Instruction Register**: Keeps track of where we are in the program, more about this later. In RISC-V this is the program counter `pc` register.
- **Global Pointer**:
- **Return Address**: The address that stores the address to return to after an internal function call. More about this later.
- **Stack Pointer**: The pointer to the base of the stack at the current moment. More about this later too. `sp`
- **Frame Pointer**: The pointer that points to the base of the current frame `s0, fp`. This can double as another saved register in optimized code and is only ever used for debugging purposes.
- **Thread Pointer**: `tp`
- **Temporary Registers**: `t1 - t6`, for storing temporary data that can be erased at any point. Volatile registers.
- **Saved Registers**: `s1 - s11`, for data that need to be preserved across function calls and exceptions.

1.4.2 Memory

This is essentially just a giant map of data with key-value pairs that represent certain data.

1.4.3 Data Bus and Control Bus

1.4.4 Storing and Loading from memory

The two basic loading and storing operations in RISC-V are the `lw, sw` instructions which mean load-word and store-word respectively.

6 RISC-V Green Sheet (but better)

MNEMONIC	FMT	NAME	DESCRIPTION	NOTE	TYPE
add, addi					
and, andi					