Cormac Passion Unit

By: Cormac Taylor

I pledge my honor that I have abided by the Stevens Honor System.

How to use assembler:

The assembler is written in Python. It translates standard ARM ADD, MUL, LDR, and STR instructions into machine language for the Cormac Passion Unit to understand. The resulting file will be named “out.txt” and is created or updated in the same file as the assembler program is located. Make sure the “assembler.py” file and your program file are in the same folder. In order to run it, open a environment that runs python, run the file, then input the name of your program file of type .txt. The “out.txt” file should be created or updated according to the “demo_program.txt” file in the same folder as the “assembler.py” file.

CPU’s Architecture:

The CPU has four 1-byte general purpose registers referred to in instructions as X0, X1, X2, and X3 respectively. Instruction memory has 2^8 2-byte addresses which contain exactly a single instruction as detailed in the “Instructions” section of this document. Additionally, data memory is byte-addressed consisting of 2^8 unique addresses. The unit is able to perform lightning fast addition and multiplication, as well as data access and storage.

Instructions:

16 bit binary encoding

Opcode:

First 5 bits of the binary encoding. From left to right the bits are defined as follows:

Immediate numbers? (1 for true; 0 for false)
ADD? (1 for true; 0 for false)
MUL? (1 for true; 0 for false)
LDR? (1 for true; 0 for false)
STR? (1 for true; 0 for false)

Rm, Rn, and Rt:

2 bits long as that is the fewest number of bits to encode each of the 4 unique registers in the register file.

imm:

7 bits long as that is the reminder of the bits not otherwise used in the binary encoding.

16 - 5 (opcode) - 2 (Rm) - 2 (Rn) = 7 bits

Rm = destination register; bits represented by m
Rn = register 1; bits represented by n
Rt = register 2; bits represented by t
imm = immediate number; bits represented by i
Unused bits represented by x

General Format	Binary Encoding
ADD Rm, Rn, Rt	0b01000mmnnxxxxxtt
ADD Rm, Rn, imm	0b11000mmnniiiiii
MUL Rm, Rn, Rt	0b00100mmnnxxxxxtt
MUL Rm, Rn, imm	0b10100mmnniiiiii
LDR Rm, [Rn, Rt]	0b00010mmnnxxxxxtt
LDR Rm, [Rn, imm]	0b10010mmnniiiiii
STR Rm, [Rn, Rt]	0b00001mmnnxxxxxtt
STR Rm, [Rn, imm]	0b10001mmnniiiiii