MF-CPU User Manual

By: Vedhesh Anandhan Kamaraj and Dhruv Mittal

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

Welcome to the Millenium Falcon CPU

Who did what:

- Dhruv and Vedhesh:
  - Worked on CPU Architecture
- Dhruv:
  - Worked on the demo program
  - Designed CPU in Logism Evolution
- Vedhesh:
  - Worked on User Manual
  - Created Assembler for the CPU

How to use our CPU:

1) Download the “demo.mf”, “assembler.py”, “CPU.circ”.
2) Open “assembler.py” and “demo.mf” in your IDE of choice and “CPU.circ” in Logism Evolution.
    a) The demo.mf file consists of two parts – the data and text section.
    b) The data section contains the hexadecimal values that will be stored in the main memory.
    c) The text section contains the instructions for the CPU to execute. The Python assembler will turn the instructions into binary, convert the binary into hexadecimal and create an image file of those values to be loaded into the Instruction Memory of the CPU.
3) Navigate to assembler.py and run the file (Either by hitting the play button or running “py assembler.py” in the terminal). This will generate a “data.txt” file and an “instruction.txt” file in that same directory.
    a) Note: Make sure that the terminal is in the **same directory** as “assembler.py” and “demo.mf” or else it will not work.
4) Navigate to “CPU.circ” in Logism Evolution
5) Right click on the “Instruction Memory” RAM component, click on “Load Image”, navigate to the directory where the files were generated and open the “instruction.txt” file.
6) Next, right click on Main Memory, click on “Load Image”, navigate to the directory where the files were generated and open the “data.txt” file.
7) Run the simulation and watch the CPU execute the instructions!

Architecture Description Design

Our CPU architecture has 4 registers that are aptly named R0, R1, R2, R3. Each register has the ability to store an 8 bit number ranging from 0-255. Our CPU has instructions to Add, Subtract, Load data from Memory to a Register, and Store data from a Register to Memory.

General Instruction Format

Instruction	Format	Opcode	Destination Register (Rd)	Target Register (Rt)	First Register (Rn)	Second Register (Rm)	Memory Address (Mem)
Addition	Ad Rd Rn Rm	00	2 bit	-	2 bit	2 bit	-
Subtraction	Su Rd Rn Rm	01	2 bit	-	2 bit	2 bit	-
Load	Lo Rt Mem	10	-	2 bit	-	-	4 Bit
Store	St Rt Mem	11	-	2 bit	-	-	4 Bit