CS 382 Project 2

Alim & Aidan

December 2022

1 Our CPU

We named our CPU Logan8r because Logan is the best. Logan8r currently has the following features:

- 32 working registers*;
- full support of **ADD**, **SUB**, and **LDR** instructions;
- Python image script for instruction memory (probably the best one you'll see);
- (EXTRA) Python image script for data memory (probably the best one too);
- (EXTRA) LED lights that show you the instruction running!

2 Compiling

In the folder with the circuit, you will find two Python scripts: one is for instruction memory and the other one is for data memory.

2.1 Instruction Memory

To run the instruction memory script, open the terminal in the folder and type:

```
python imgcreator.py
```

Then you just have to do what the prompt's asking you! It's that simple. Some things to consider:

- for testing, we suggest putting all three instructions. You could use any registers, but I promise you, they're all the same;
- **ADD** and **SUB** use Rd, R1, R2 format, while **LDR** uses Rd, [R1, R2] just to be consistent with an actual ARM processor. The square brackets don't add any difference, they're here for descriptive purposes;
- it starts to count instructions from 1 and not 0 just for simplicity purposes;
- if you make a mistake, the input line will recognize it and let you retry.

2.2 Data Memory

To test **LDR**, you need data in your memory. This is why we created a Data Memory script! To run the data memory script, again, open the terminal in the folder and type:

```
python datacreator.py
```

It takes hex data as bytes, one by one.

- it does allow you to put in 256 bytes, but typing it manually will take you longer than it took us to build the CPU;
- it starts to count bytes from 1 and not 0 just for simplicity purposes;
- it can read both single-digit and double-digit hex, so putting ”0f” is the same as typing just ”f”.

You’re done! Now you can upload the *.txt* files into Logisim by pressing right-click onto the both components. Don’t forget, *image.txt* is for instruction memory and *data.txt* is for data memory.

2.3 Using the CPU

To use any operations, you have to have numbers to work with. You’ll have to manually fill the registers you want to work with hex integer values.

As you might have noticed, the image for the instruction memory has a *0x00000000* command in the beginning by default. You could say it exists to ”turn on” the CPU. Basically, on launch, all the wiring has to be updated once to start working properly.

Controls:
Press *CTRL + R (CMD + R)* to ”turn on” Logan8r.
Press *CTRL + T (CMD + T)* to go forward half-cycle.
Press *F9 (FN + F9)* to go forward one full cycle.

3 Structure

3.1 Registers

Because our CPU is limited to only 3 instructions and we wanted to preserve the textbook opcodes, you can see a lot of missing wires that are used for the remaining operations. And again, because of this very weird rule we made up, we have 32 registers instead of the required 4 or optimal 8. This is why I put a * at the beginning - there are 32 registers, but most of them are completely useless! And because we’re very talented programmers, the script limits you to use only the first 8. Though you could just change the number of registers to use on the top of the *imgcreator.py* and it would work just fine.

3.2 LEDs

Logan8r's LEDs show the current instruction running. It uses the opcode signal to determine which instruction it is running and a bit of clever math to count the hex to choose which lights to turn on.

4 Other Stuff

Although we mostly worked on everything together, most of the hardware was done by Aidan. The Python scripts and this *.pdf* were done by Alim.

I also provided some pre-made images in the *.zip*, so you immediately have something to work with.