GoGoS CPU Manual

Name: Tri Luu

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

CPU Description:

In this CPU, we are able to use LDR to assign the value to register, ADD, and MUL for arithmetic computation into the 4 general-purpose registers using the instruction memory and data memory.

My CPU uses a language similar to assembly, which converts the assembly code into compilable and executable language to the register and returns the result to the pin inside the CPU. Using the multiplexors, we can determine which command we are using depending on the opcode.

Translator Description:

The translator was created using Python and its array library, by reading each line from toTranslate.txt, I converted them to hex code and write it back to the image file under the RAM format available in logisim.

How to generate an image file:

Type the Commands into toTranslate.txt and run the translator.py file to obtain the image file.

How to load:

Open the GogoS.circ in Logisim and right click to the Instruction Memory Ram, and choose load image, after that, click on the clock to process the commands

Available commands:

LDR immediate, register, target: locate the value's address using the immediate offset from the register, and save it to the target register

ADD save location, value 1, value 2: add value 1 and value 2 and save the result into the save location

MUL save location, value 1, value 2: multiply value 1 and value 2 and save the result into the save location

Translation rules:

For instance, the first two bits(7-6) would be translated to command, which is 00: nothing (nop), 01: ADD, 10 is MUL, and 11 is LDR

Similarly, (5-4) is an immediate number or register 2: (00: register 0; 01: register 1; 10: register 2; 11: register 3)

(4-3) is to read register 1 ((00: register 0; 01: register 1; 10: register 2; 11: register 3)

and (2-1) is for the target (00: register 0; 01: register 1; 10: register 2; 11: register 3)

Therefore, we can convert them into binary code using these rules and write them under 8bits instruction XXXXXXXX (X can be 0 or 1) and convert them into hex digits so that the RAM can read them from there.
For example:
LDR 0, x1, X2 could be translated to 11000110 in binary and c6 in hex
ADD X1, X1, X2 could be translated to 01010110 in binary and 56 in hex.
MUL X1, X1, X2 could be translated to 10010110 in binary and 96 in hex
00 is just an empty value that will not do any instruction and result

Limitations:
Can not operate other commands other than LDR, ADD, and MUL, can not check if the command is wrong (no compiler available). The user have to check the binary code and enter manually the corresponding values for translated code to data memory.