The Donger User Manual

Created By: Christopher Arias and Alexander Ha

Job Description

Alexander Ha:

- Designed and tested the CPU architecture in Logisim
- Developed and tested the assembly compiler
- Composed the User Manual

Christopher Arias:

- Developed and tested the assembly compiler
- Streamlined CPU architecture in Logisim
- Composed the User Manual
- Wrote demo program (assembly.txt)

How to Use

Step 1: Create a text file with the name "assembly.txt" and keep it in the same folder as the assembly compiler

Step 2: Input Instructions into text file

Format (MUST BE THIS FORMAT; see demo program):

```
INSTRUCTION1 Rw, R1, R2
      or
INSTRUCTION2 Rw, R1, imm
```

Examples:

```
ADD X2, X0, X1
LDR X0, [X1, 16]
SUB X1, X0, 4
```

Notes:

Possible Instructions: LDR, ADD, SUB

Only Registers X0, X1, X2, X3 (must be capital X)

CPU can only handle immediates up to 6 bits large

CPU can only handle values up to 8 bits large

CPU can only handle offsets greater than or equal to 0

If you are loading at an offset of zero or without an offset, you must specify that it is at offset 0

```
      LDR X0, [X1, 0]
```

Keep only 1 instruction per line in the text file (needs a new line at end of assembly.txt)

Everytime you want to make a new image file (running the compiler) you can not have an existing file in the same folder or it will fail (delete image.txt or move it out of the folder)

Put immediates in their respective registers to test load

Step 3: Run the Compiler and an "image.txt" file will generate with the proper hex output

Step 4: Load image file in Logisim in InstructionRAM

Architecture Description

Consists of:

- 1 program counter
- 1 CPU Clock
- 1 Data Clock
- Instruction Ram
- Decoder
- Data Ram
- 4 registers
  - X0, X1, X2, X3
- Multiplexors:
  - RegDataW
  - AddOrSub
  - RegisterOrImmediate
- Signals:
  - RegData1
  - RegData2
  - ValueData2
  - Add
  - RegDataW
  - LDR
- 4 LED outputs

Binary Encoding

ADD Rw, R1, R2

15	14 <-> 9	8 <-> 7	6 <-> 5	4 <-> 3	2	1	0
0	000000	Rw	R2	R1	1	0	0
bufferbit	6 bit immediate	write Register	Register2	Register 1	addSignal	ldrSignal	immSignal

ADD Rw, R1, imm

15	14 <-> 9	8 <-> 7	6 <-> 5	4 <-> 3	2	1	0
0	immediate	Rw	00	R1	1	0	1
bufferbit	6 bit immediate	write Register	Register2	Register 1	addSignal	ldrSignal	immSignal

SUB Rw, R1, R2

15	14 <-> 9	8 <-> 7	6 <-> 5	4 <-> 3	2	1	0
0	000000	Rw	R2	R1	0	0	0
bufferbit	6 bit immediate	write Register	Register2	Register 1	addSignal	ldrSignal	immSignal

SUB Rw, R1, imm

15	14 <-> 9	8 <-> 7	6 <-> 5	4 <-> 3	2	1	0
0	immediate	Rw	00	R1	0	0	1
bufferbit	6 bit immediate	write Register	Register2	Register 1	addSignal	ldrSignal	immSignal

LDR Rw, [R1, imm]

15	14 <-> 9	8 <-> 7	6 <-> 5	4 <-> 3	2	1	0
0	immediate	Rw	00	R1	1	1	1
bufferbit	6 bit immediate	write Register	Register2	Register 1	addSignal	ldrSignal	immSignal