x80Hao User Manual

Andrew Schomber & Matthew Bernardon

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

Job Descriptions:

- Both
  - Worked on the CPU together
- Drew:
  - Created the assembler
  - Added fun CPU components
- Matt:
  - Designed instruction architecture
  - Created the user manual
  - Created the demo program

How to Use:

Step 1) Create the assembly file.
The first line of this file is the data segment consisting of 16 8-bit decimal numbers (0-255) with spaces between them. These 16 values will be the initial DRAM.
All following lines are the text segment and contain the instructions. The first character of each instruction is the first letter of that operation (‘A’ for add, ‘S’ for store, etc). For add and multiply, there are three more arguments which are 2-bit register numbers in decimal (0-3). For store and load, there are two more arguments which are a 2-bit register number in decimal (0-3) and a 4-bit memory address in decimal (0-15).
Comments can be added to any instruction without the need of a comment symbol. Simply type out the comment on any instruction line after the instruction itself to comment that line.
Refer to ‘Instruction General Format’ and ‘demo.hao’ for more information on how to code the assembly file.

Step 2) Move the assembly file to the same directory as the assembler, ‘as.py’. Open the terminal and navigate to this directory.

Step 3) Run the command ‘python as.py <filename.extension>’. This will generate two .o files, ‘instructions.o’ and ‘ram.o’.

Step 4) Open the CPU in Logisim. Right click on the instruction memory and load instructions.o. Do the same with the DRAM, loading ram.o.

Step 5) Run the simulation. Note that when all instructions are complete, the CPU will continue to run the operation ‘A 0 0 0’, adding the value of R0 to itself.

Architecture Description:

Our CPU has 4 general purpose registers that are referred to simply by number (0, 1, 2, 3). Each register can store an 8 bit number (0-255). The functions of our CPU are add, multiply, load, and store.

A: Adds the values of two registers and stores the result in a register.

M: Multiples the values of two registers and stores the result in a register.

L: Takes a value from a memory address and stores it in a register.

S: Takes a value from a register and stores it in a memory address.

Instruction General Format:

Instruction	General Format	Opcode	Rd (Destination Register)	Rn (1st Register)	Rm (2nd Register)
Addition	A Rd Rn Rm	00	2 bits	2 bits	2 bits
Multiply	M Rd Rn Rm	01	2 bits	2 bits	2 bits

Instruction	General Format	Opcode	Rt (Target Register)	Mem (Memory Address)
Load	L Rt Mem	10	2 bits	4 bits
Store	S Rt Mem	11	2 bits	4 bits

The opcode needs 2 bits, as there are 4 instructions to choose from.
All register fields (Rd, Rn, Rm, Rt) also need 2 bits, as there are 4 registers to choose from.
Mem needs 4 bits, as our memory has 16 unique addresses to load from and store to.