Quantum CPU

Sean Maida

CPU Architecture

Registers:

- There are 4 general purpose registers which each store 8 bits.
    - X0, X1, X2, X3
- To refer to a register in the assembly program use X0, X1, X2, or X3.

- RAM: The CPU takes an image file as input which is loaded into RAM.
- Multiplexores
    - 3 multiplexores
    - There are 2 multiplexores after the registers which both take the data inside each registers as input where ReadReg1 and ReadReg2 determine which register to pass along to the adder/subtractor.
    - The third multiplexor takes in the results from the adder and subtractor and decides which one to pass through to the output.
- PC: stores the address of the next instruction to be executed.
- Clocks:
    - There are 2 clocks.
    - One clock is connected to the RAM and the other clock is connected to each register.

Function:

- The operations the CPU performs are addition and subtraction.
    - The input values will both go through the adder and subtractor, but there's a multiplexor which takes in those results and decides on which one to pass through to the output which is decided by the switch.

Instruction Library

`LOAD Xt, num`

- Loads a number into Xt
- Binary Encoding: 11000000

`PLUS Xd, Xn, Xm`

- Adds Xn and Xm and stores the result in Xd
  - Xd = Xn + Xm
- Binary Encoding: 10000000

`MINUS Xd, Xn, Xm`

- Subtracts Xn by Xm and stores the result in Xd
  - Xd = Xn – Xm
- Binary Encoding: 11000000

- All binary encodings for each instruction are 8 bits. 8 bits are enough to distinguish between the operation/instruction while also indicating the destination and the two registers used in the operation/instruction.
  - For arithmetic instructions, the MSB is 1 while the seventh bit is used to indicate which operation we want to use: 0 for addition (PLUS), while 1 for subtraction (MINUS). Bits 2:1 are used to indicate the destination register, bits 4:3 are used to indicate register Xn, while bits 6:5 are used to indicate register Xm.
    - 00 → Register 0
      01→ Register 1
      10 → Register 2
      11 → Register 3
    - Example: 11100111 → MINUS X3, X1, X2
  - For memory addressed instruction(LOAD), the MSB is zero while the seventh bit is 1 to indicate LOAD. Bits 2:1 are used to indicate the destination register you want to move the number into.
    - 00 → Register 0
      01→ Register 1
      10 → Register 2
      11 → Register 3
    - Example: 01000010 → LOAD X2, 0

Assembler

- The assembler is made in python.
- The program to be executed is to be written in a .txt file.
- The .txt file and assembler must be in the same folder.
- The assembler takes the program as input, so when executing the assembler you would write *assembler("filename.txt")*.
- The assembler goes through each line, identifies what instruction is being performed, what registers are being used and converts it to machine code. Then it converts it to hexadecimal which is put into another .txt file which will be the image file.
    - The image file is created by the assembler which takes the assembly program, turns each line of code into machine code and breaks it up into a certain amount of bits which indicate the operation to perform, the register, or an input value, then it converts those sequences into hexadecimal.
- When loading the image file into ram, a menu will pop up called "Hex file format" and you want to select "v3.0 hex"; don't change anything under "v3.0 hex".