Manchester Baby Simulator

Today's exercise involves downloading a simulator for the Manchester "Baby" and writing, loading, and executing some code on it. By the end of the exercise, you should have an appreciation for how wonderful we have it today along with an idea of how a simple architecture might be implemented.

Downloading the Simulator

In 2001, as part of a body of detailed research into the historical operation of the Manchester Baby, David Sharp of the University of Warwick created as accurate a simulator of the Baby as I have ever seen. Not only does it simulate the operation of the computer, it also simulates the peculiar I/O associated with the machine. He has offered his work to the word in the form of a Java-based simulator complete with a graphical interface of the control panels and output screen.

You can download this simulator from the URL http://www.davidsharp.com/baby/index.html. At the bottom of the page, you should see a number of links, one of which is titled, "Photo-realistic emulator - to run as an application." Clicking on this link allows you to download the archive of the simulator.

Running the Simulator

The simulator is downloaded as a compressed zip file. By right-clicking on this file, you should see the option to "Extract all…" Doing this will extract all of the compressed files into a folder titled, "Baby."

The Java simulator must be run from the command window with the source code contained in the current directory/folder. To do this, navigate to the "Baby" folder inside of Windows Explorer. Inside the "Baby" folder, you should see a folder titled, "simulator."

While holding both the Ctrl and Shift keys down, right-click on the simulator folder. From the list of options that appear, select, "Open command window here."

At the command prompt, type "java Baby" without the quotes and with the correct capitalization.

After you do this, two windows will appear, one on top of the other. The one with the array of red buttons is the input panel. The one with the black circle with the matrix of green dots and lines is the CRT that functioned as memory.

Loading a Program

Loading a program into the Baby's memory is complicated for us "modern" programmers for a couple of reasons.

- The least significant bit is the leftmost bit, i.e., the binary values read right to left.
- Each bit is cleared (erased) or set (written) on an individual basis.

To help, it is possible to clear the whole memory so that programming involves simply setting the bits that need to be one. To do this, there is a set of eight buttons in a black rectangle at the bottom of the input window. Clearing memory can be done by clicking on the button labeled, "KSC."

The memory locations are viewed from the top down on the round display with the top row representing address 0 and the bottom row representing address 31.

To select an address, use the five switches in the middle of the input panel screen to select a row. A switch in the up position represents 0 while down represents 1.

To program a memory location, put the erase/write switch in the appropriate position depending on whether you want to place a zero (erase) or a one (write). Next, use the red button array at the top of the panel to select which bits you want to erase or write. Usually, I clear the store, then set the bits to one that I want.

To run the program, you can simply tap the button labeled, "Run," on the simulator. Sometimes, however, the simulator is in a halted state. If this is the case, go to the switch panel view, verify that the "Stop/CS" switch is pointing to CS. Now try to run it.

Important Programming Notes

There are a number of important things to remember when programming the Manchester Baby. I have listed the ones I can remember here, but I'm sure to have forgotten at least one or two.

- Before an instruction is executed, the program counter (called the CI or control register in the Baby) is incremented by one immediately before executing an instruction. Therefore, upon start up when the CI is reset to zero, the first instruction to be executed is actually the second memory location, address 1.
- Another problem with this "pre-incrementing" of CI is that a JMP needs to be to the address immediately BEFORE the actual destination address.
- The arithmetic unit can only subtract and negate. Therefore, loads, additions, and such may need to use a temporary memory location in order to work with positive values.
- Remember that all address and data values stored to the Baby's memory are backwards, e.g., bit 0 is the left-most bit. This forces the address $20_{10} = 10100_2$ to be entered as 0-0-1-0-1 from left to right.

Assignment

Although you may feel as if you've been transported back to CSCI 1260, our first assignment is to create three very simple programs. These programs will be very simple because you've actually been transported back to 1948 where programs couldn't be that complicated.

The first program you will write is simply an infinite loop that will increment a counter.

Your second assignment is to make the first program that you wrote stop when it reaches 65,535, i.e., $1111\ 1111\ 1111\ 1111_2$. For this, you will need to use the CMP and the STP functions.

The third assignment you need to write is to create a program that will add 9 numbers together to get a result. Those numbers along with the reverse binary representation are shown below.

Decimal	Reverse Order Binary
54	01101100000000000000000000000000
190	01111101000000000000000000000000
30	01111000000000000000000000000000
27	11011000000000000000000000000000
19	11001000000000000000000000000000
21	10101000000000000000000000000000
250	01011111000000000000000000000000
47	11110100000000000000000000000000
44	00110100000000000000000000000000