ALU, Memory, and the Bus

Andre Mello

1 Abstract

My project was to build an 8 bit programmable computer on a breadboard and modify it so that it can be programmed using the MSP Microcontroller. Due to several bugs and issues, I decided to downscale the project and finished the bus, two registers, and the arithmetic logic unit, effectively building a calculator capable of adding and subtracting from basic electronic components. The project is an elucidating educational piece of computer architecture and how electronics make up computers. The rest of the device is still a work in progress. As for the performance, the circuit functioned correctly, but had issues sometimes flipping bits, since no error correcting is built in. It would be hard to correct this without adding complexity to the machine that takes away from its original pedagogical purpose, therefore, I believe it performs optimally for its purpose.

2 Introduction

This project began with a strict pedagogical purpose. I have an interest in software, but had no prior knowledge of computer architecture. Therefore, I believed it would be a good idea to build a computer and code it using the MSP. With no knowledge of computer architecture, and still being a beginner at electronics, I found that the best way to achieve this was to use the resources made available by Ben Eater (1) and make slight modifications to work with the MSP Microcontroller. The architecture used was created by Albert Malvino (2).

3 Apparatus

The circuitry was a combination of two identically built registers and the arithmetic logic unit. Each register was placed to one side of the ALU and were made inputs for the calculations of addition and subtraction performed by the ALU. Figure 1 shows the schematic for a register and Figure 2 shows the schematic for the ALU and the Flags Register, but the Flags Register is still currently being built. Figure 1 shows the connections for register A, but the schematic for register B simply has the connections named A__ be named B__ instead (where _ stands for a wildcard character).

3.1 Registers

$\overline{AI}$ would make the register take an input and write it to memory, replacing the old value, if it were low, and do nothing if it were high. $\overline{AO}$ would output the contents of register A to the bus if it were low and do nothing if it were high. CLR would clear the content of the register if set high, and do nothing otherwise. Finally, CLK was the clock input. The whole of the register only functions when a signal fed through CLK transitions from low to high.

3.2 ALU

Just as the Registers, the ALU was fed a CLK signal without which, it would not function. If the signal did not transition to high, the ALU did nothing. $\overline{EO}$ was the output signal. If it were low, the ALU would output to the bus, if it were high, it would do nothing. SU governed whether the ALU performed an addition or subtraction operation. If it were high, it would subtract, otherwise it would do nothing.

3.3 MSP

The MSP was connected to the bus through pins zero through seven of port 1, and connected to at most 5 other pins described above through port 2. A timer module was built, but I just had the MSP connect to the CLK, the $\overline{AI}$, $\overline{BI}$, and SU pins through port 2. LEDs were connected to the registers and ALU in a manner that let us see the outputs of each, and the MSP outputted to the bus and set $\overline{AI}$ low to write to register A. It did similarly for register B, and it set the SU signal high when it would like to subtract. All numbers displayed were in binary.

4 Results

The device performed according to expectations. Every so often, it would, however flip a bit, resulting in an operation off by the addition of some power of 2. This was not especially surprising to me, since breadboards do not seem to me like a terribly reliable transmission channels. Since the purpose of the circuit is purely educational, adding error correction would have detracted from its appeal by adding unwanted complexity. Making it more reliable without adding complexity would be the next major improvement, but I am currently unaware of how to do so without adding more circuitry that would detract from its simplicity. This bit flipping behavior was rare enough that it still functioned well, but made it so I would not be confident deploying such a machine. Overall, I am quite happy with the results that the circuit was able to achieve, given its educational goals.

5 Discussion

The only problem encountered with the circuit is related to the occasional bit flipping, but transitioning from breadboards to a more reliable information channel should fix the issue, I believe. An issue related to the construction of said circuit, however, is the difficulty in keeping all the wires clean and organized, with enough space to wire all the connections. It took a lot of time to wire the ALU for this reason, and experience with organization is an asset when building such a circuit.

Register

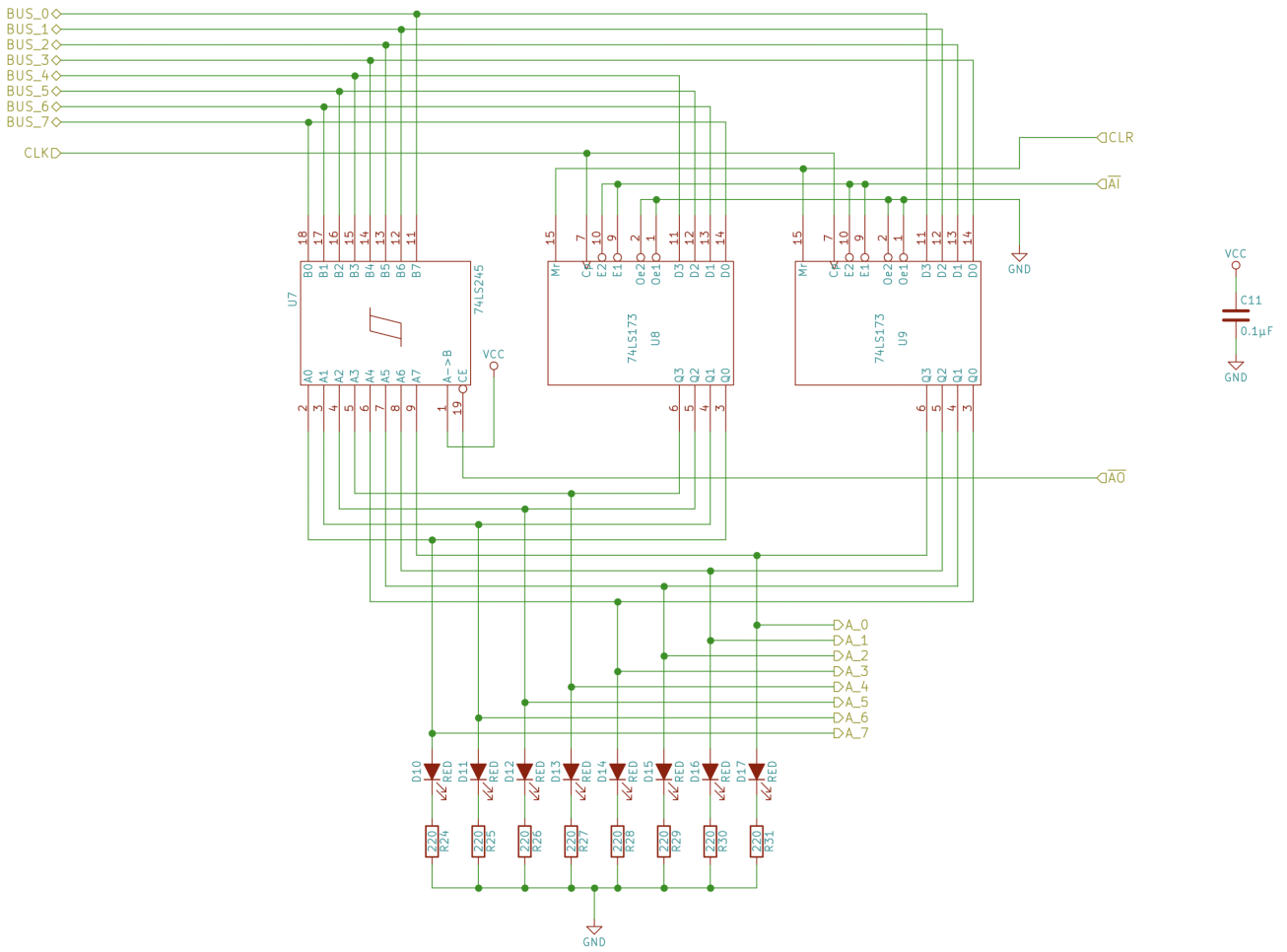

Figure 1: Schematic of the Registers used by the computer made by Ben Eater (1).

6 Conclusion

The device was absolutely worth constructing. It proved a phenomenal educational resource and an elucidating first step into computer architecture. I plan to continue building the rest of the computer.

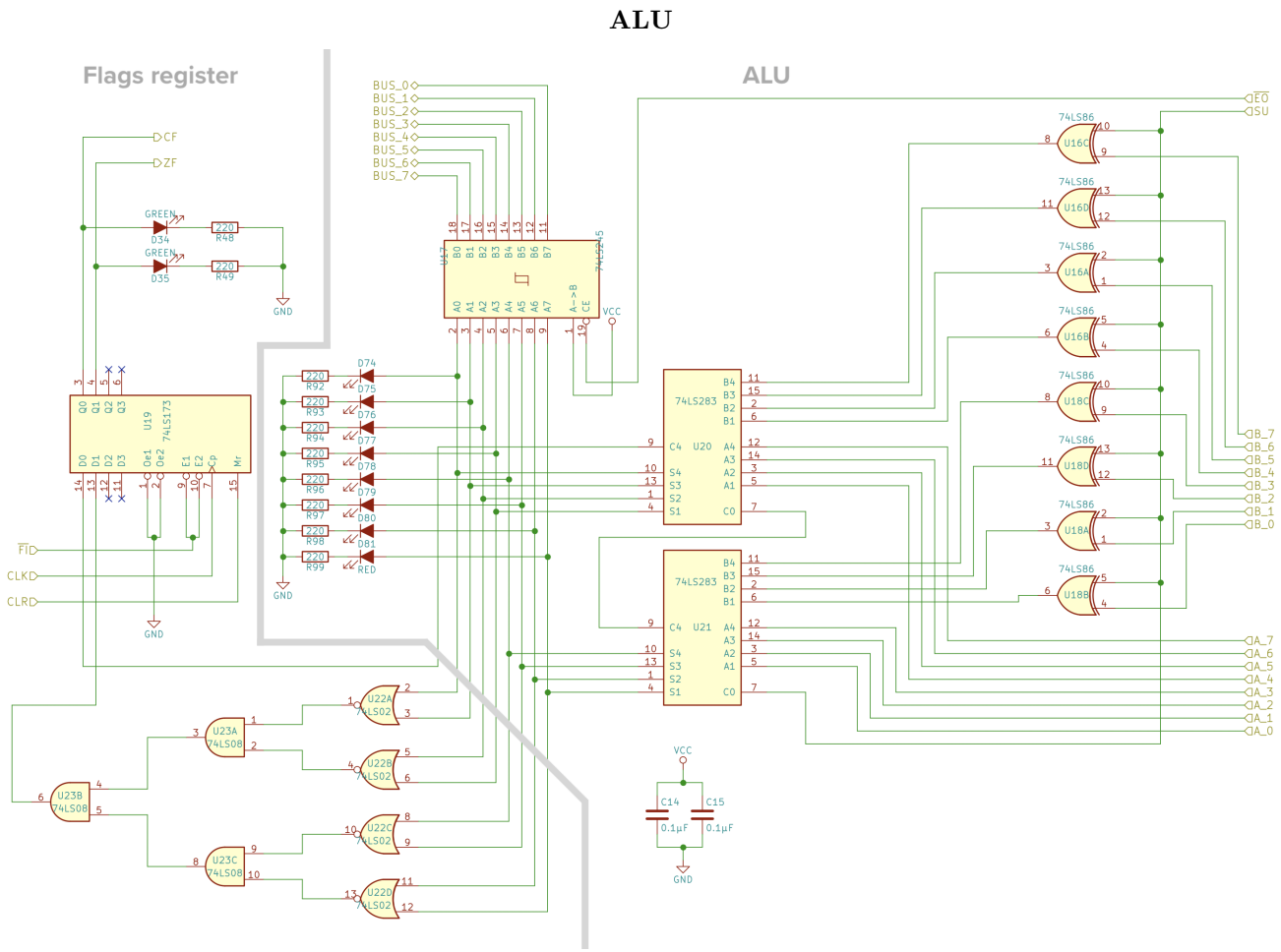

Figure 2: Schematic of the ALU used by the computer made by Ben Eater (1).

References

[1] Eater, Ben. "Build an 8-Bit Computer From Scratch." *Ben Eater*, 2016, https://eater.net/8bit.

[2] Malvino, Albert. *Digital Computer Electronics*. McGraw-Hill, 1977