**Lab 3 Body Fat Monitor**

**Joseph Tomalewski**

**Date: 9/27/2022**

**Purpose of Lab**

I must utilize AVR assembly conditionals to create a body fat monitor with an Arduino board kit while also using a breadboard with LED’s and programming the AVR assembly in the Arduino IDE. I must organize and program for the AVR Atmega328 to light up a certain color of lights depending on the gender, age, and body fat of an individual.

**Description of Solution(s)**

To program the solution for the body fat monitor, I would have to make my code very organized and readable for myself to see exactly where I was and what else I would need to work on in the following problem I would be stuck on. For example, first I made a registers block to first import and initialize the gender, age, and fat to register 18, 19, and 20. After that block I have my first comparison branch to check whether the input will compare to a male or female input, depending on if male (0A) or female (0F) is inputted, it will branch to its corresponding block.

The C program that is working with my S file is slightly modified to start running my registers block and then my branch block. After that my assembly file will run on its own to take input through the serial monitor and produce the output of the correct LED light for the body fat monitor.

After my registers and branch block is executed, they will then branch off to do checks for the gender, age, and body fat to then produce the blinking LED light corresponding to the gender, age, and body fat. I would occasionally have bridges for my branches due to them having far instructions.

**Test Results**

The most common test I performed for this lab was to see if my program was branching properly by making the LED’s light up whenever they would branch properly with the corresponding input. If they didn’t light up then I knew there was a bug I would have to fix, then when I know it branches properly I would then code and implement more branches to make comparisons on the inputs the program would be given. One bug I had that had me stuck for a while was incorrect registers for the blink program. I accidentally somehow deleted some of the numbers for the blink block and the blink block would make my LED blink only once. To fix this I had to carefully examine the block of code and the TA also helped me find the issue. This would be slightly hard to find as the compiler would not say anything about the issue, even when I was incorrectly using register 1.

**Answers to Questions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Gender** | **Hex** | **Age** | **Hex** | **Body Fat** | **Hex** | **Light** |
| M | 0A | 24 | 18 | 33 | 21 | ALL |
| F | 0F | 52 | 34 | 50 | 32 | ALL |
| M | 0A | 30 | 1E | 22 | 16 | RED |
| F | 0F | 33 | 21 | 60 | 3C | ALL |
| M | 0A | 65 | 41 | 15 | 0F | GREEN |
| F | 0F | 60 | 3C | 45 | 2D | ALL |

**Discussion**

I believe I learned to be very familiar with using comparisons specifically coding using conditional branches and unconditional branches. I had to really understand how to branch if my comparisons were higher or lower, and if they would equal to 0 or not. I believe the lab was relatively easy if you can easily organize your code because it can get easy to be lost on where you are or what you were trying to do since assembly can be hard to read sometimes, especially if you have many branches. I liked doing this lab since it had to use LED’s and to make them blink which was fun. If I had a suggestion, it would be making this lab easier to understand by maybe listing certain pages in the book or online videos to explain some of the instructions used just in case someone would forget exactly what an instruction means. However, it referenced the previous labs which would help tremendously in completely this lab.

**Contribution to Teamwork**

This was a solo lab.

**References**

For the blink algorithm I used from lab 1. This algorithm was very useful as it seems slightly tedious and complicated to make my own. I will also include the C file I used to start my program.

