Mezemir Gebre

Arm Assembly Programming Report

**Part 1**

**A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

The above two screenshots are the source code and an output from the gdb for the “fourth” program. This program is basically implementing if statements. It loads 0 to variable x and store that in register r1 then it compares that value with 0, if they are equal which they are equal in this case(The zero flag is set) so it jumps and execute the code under the “thenpart” and finally the value of y which is zero loaded in the r2 register. As you can see on the second screenshot, the value of r2 register is 0 and I confirmed that by accessing its memory address as “x/1wd 0x200a5” as you can see, I got 0 here as well. By doing p/t $cpsr, I could see the zero flag is set as you can see in the above image (11100000….1000).

**Part 2**

**A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

The above two screenshots shows the source code, registers value, memory access and the Z flag value for the fourth program after replacing beq with bne and the back to back branch instruction. The only difference with the first program and this one is that this program is more efficient and jumps when the condition is false that means in this case we used branch on not equal operation (z==0) instead of branch on equal(beq) operation, however the value at r1 is zero and zero = zero so the condition is contradicting and the program jumps to the “thenpart “ like the first program and execute the code under that label. The z flag is zero is clear in this time as you can see in the above image at “cpsr” register and the value at r2 is zero as well like the first program.

**Part 3A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

The first screenshot is a source code for my controlStructure1 program. The second screenshot shows the X value in hex, the result in registers and in the memory for the destination operand and the status of the z flag as you can see. It first loads the value of x which is zero the r1 register and compares that with 3 and I used “ble” operation to compare them since 0 is less than 3, the program jumps to the label “tp” and executes the code below it. It subtracts 1 from the r1 register and store the result in it, as you can see the result in r1 register. I displayed the value in the z flag by using the command “p/t $cpsr”.