Lab 6

Madiba Hudson-Quansah

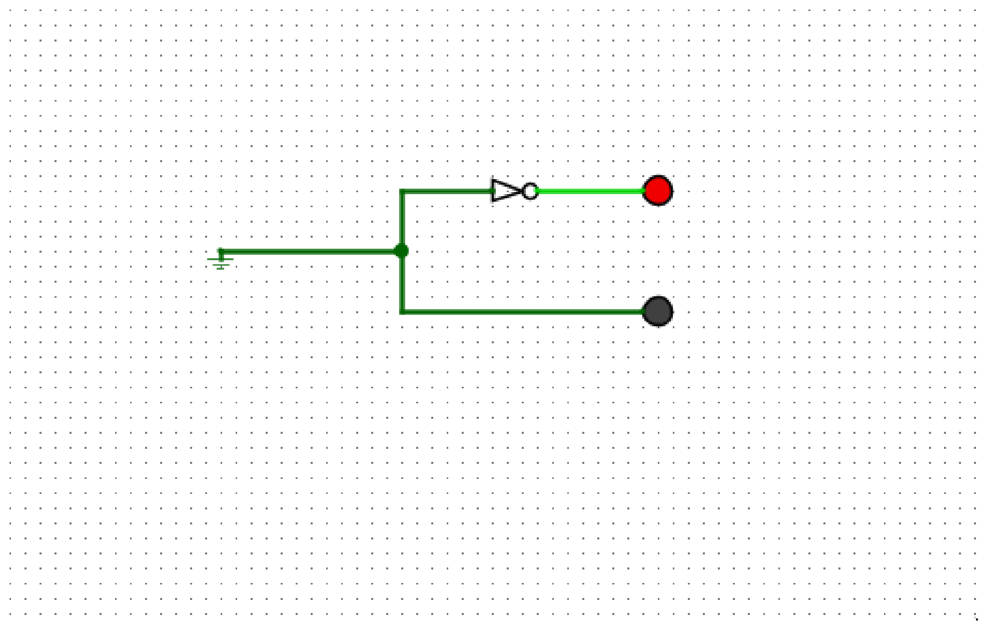
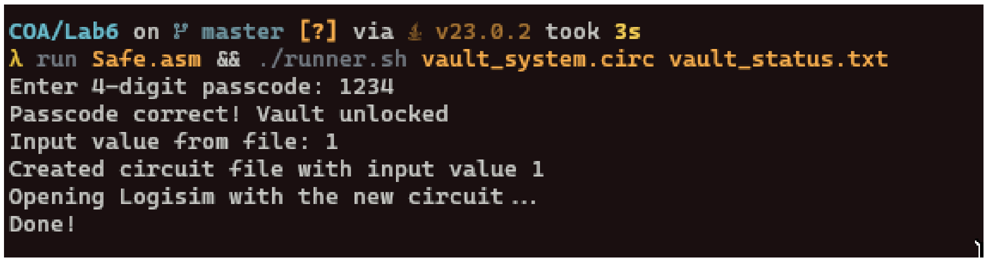
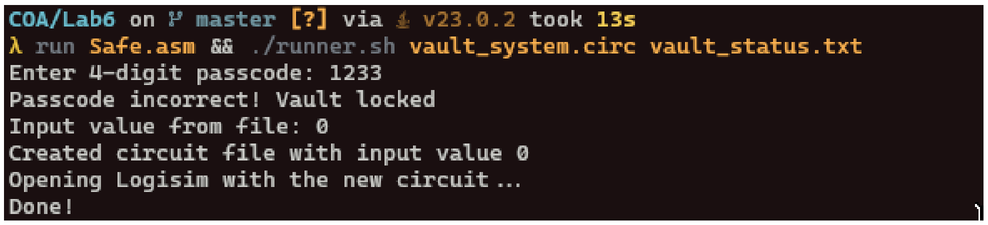


Figure 1: Modified Vault



```
COA/Lab6 on  master [?] via  v23.0.2 took 3s
λ run Safe.asm && ./runner.sh vault_system.circ vault_status.txt
Enter 4-digit passcode: 1234
Passcode correct! Vault unlocked
Input value from file: 1
Created circuit file with input value 1
Opening Logisim with the new circuit...
Done!
```

Figure 2: Correct Code



```
COA/Lab6 on  master [?] via  v23.0.2 took 13s
λ run Safe.asm && ./runner.sh vault_system.circ vault_status.txt
Enter 4-digit passcode: 1233
Passcode incorrect! Vault locked
Input value from file: 0
Created circuit file with input value 0
Opening Logisim with the new circuit...
Done!
```

Figure 3: Incorrect Code

Question 1

What was the most challenging part about writing the conditional check in MIPS compared to languages like Python or C?

Solution: The most challenging thing about branches in MIPS is manually setting the branch target and handling returning back to the main execution flow. In Python and C branches are contained syntactically within a code block making it easy to follow the program flow and logic, but with MIPS the branch target is separated from the branch instruction making it harder to follow.

Question 2

How do you handle the concept of else in MIPS, where there are no direct ”else” keywords?

Solution: Else logic can be handled in MIPS in two ways. In the case of an else if another branch instruction with a corresponding branch target can be used but in the case of a simple else the condition logic can be made so that when the if condition fires the branch target skips over the else block logic.

Question 3

What would happen if you accidentally reversed the operands in the bgt instruction?

Solution: It would change the logic of that instruction. For example if the original instruction was `bgt $t0, $t1, label` meaning if the value in register $t0 is greater than the value in register $t1 then branch to label. If the operands were reversed to `bgt $t1, $t0, label` then the instruction would mean if the value in register $t1 is greater than the value in register $t0 then branch to label.

Question 4

If you wanted to compare three numbers instead of two, how would you extend your logic?

Solution: I would use the principle of associativity to compare the numbers. Given numbers a, b and c, I would first compare a and b and then compare the result with c. Therefore in the case where $a > b$ and $b < c$ it can be inferred that $a > c > b$ and so on.

Question 5

How did you go about your space-separated input in Case challenge 1? What was your logic?

Solution: My logic involved first skipping leading spaces by checking for the ascii value of the space character (32) while iterating through the input string, and storing the index of the first non-space character in a register. Then from that point I would iterate through the string converting the ascii number to a integer by subtracting its value from the ascii value of the character ’0’ (48) and multiplying the result by 10 and adding it to the sum, until a space or a newline was found. I then placed these values into an integer array repeating the process until the end of the string was reached.