ICS Lab Report - lab4

SCGY Cao Gaoxiang PB20000061

December 28, 2021

Lab Name

Riddle

Lab Purpose

Functionally reproduce a C++ program.

Lab Content

For this experiment you need to functionally reproduce a C++ program. The program is as follows:

```
int judge(int r0) {
    int i = 2;
    r1 = 1;
    while (i * i <= r0) {
        if (r0 % i == 0) {
            r1 = 0;
            break;
        }
        i++;
    }
    return r1;
}
```

Your program should follow a specific framework:

```
.ORIG x3000
... ; TO BE DONE
HALT
JUDGE ... ; TO BE DONE
... ; TO BE DONE
RET
... ; TO BE DONE
.END
```

r0 (an integer, $0 \leq r0 \leq 100000$) is given before program executes (just like lab1), and store the final result in r1. (No need to print out with TRAP)

Lab Environment

Windows 11 Home Edition version 21H2, Visual Studio Code, LC3Tools v2.0.2.

Lab Procedure

Since LC-3 has no multiply and modular instructions, first we need to designed them. For multiply, we simply add i for i times to get i^2. For modular instruction, we continuously subtract R0 with i, until R0 is less than i.
For each subroutine, the state of the registers will be restored after the subroutine is over. The code is as follows.

```
SQUARE ;Calculate i^2, value stored in R5
    ST R3, REG3
    AND R5, R5, #0
    ADD R3, R2, #0
    LOOP_SQUARE BRnz BREAK_SQUARE
    ADD R5, R5, R2
    ADD R3, R3, #-1
    BRnzp LOOP_SQUARE
    BREAK_SQUARE LD R3, REG3
```

```
RET

MOD ;Calculate R0%i, value stored in R3
    ST R0, REG0
    ST R2, REG2
    ST R4, REG4
    AND R3, R3, #0
    NOT R2, R2
    ADD R2, R2, #1
    LOOP_MOD ADD R4, R0, R2
    BRnz BREAK_MOD
    ADD R0, R4, #0
    BRnzp LOOP_MOD
    BREAK_MOD ADD R4, R0, R2
    BRz RETURN_MOD
    ADD R3, R0, #0
    RETURN_MOD LD R0, REG0
    LD R2, REG2
    LD R4, REG4
RET
```

The iterator i will be stored in R2, and the main routine of the program is as follows.

```
JUDGE AND R2, R2, #0
    ST R7, REG7
    ADD R2, R2, #2 ;i
    AND R1, R1, #0
    ADD R1, R1, #1
    NOT R4, R0
    ADD R4, R4, #1
    LOOP_JUDGE JSR SQUARE
    ADD R5, R5, R4
    BRp BREAK
    JSR MOD
    ADD R3, R3, #0
    BRz NO
    ADD R2, R2, #1
    BRnzp LOOP_JUDGE
    NO AND R1, R1, #0
    BREAK LD R7, REG7
RET
```

Correctness Verification

We used LC3Tools for correctness verification. The correct answer comes from The C program. The following cases of R0 have been tested: 7, 31, 97, 137, 142, 10000