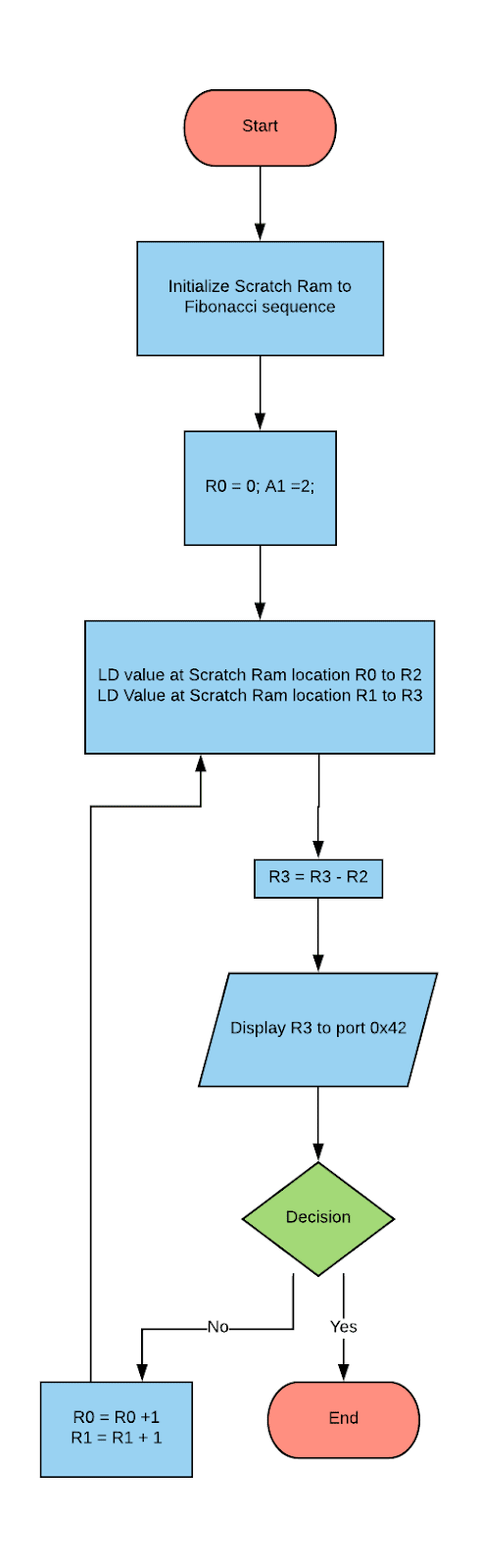
Behavior Description

The following program stores the first 14 values of the Fibonacci sequence inside the first 14 Scratch Ram memory location. It then computes the difference between values that are 3 memory locations apart. These values are then displayed to the user in port 0x42.

Flow Chart



Verification

To verify the validly of the program every single possibility was tested and tabulated in table 1 below. The values start with the first combinations of {0,2} and end with the last combination of { 55, 233}.

|  |  |  |  |
| --- | --- | --- | --- |
| Test # | Lower Number | Upper Number | Output Port 0x42 |
| 1 | 0 | 2 | 0x02 |
| 2 | 1 | 3 | 0x2 |
| 3 | 1 | 5 | 0x04 |
| 4 | 2 | 8 | 0x06 |
| 5 | 3 | 13 | 0x0A |
| 6 | 5 | 21 | 0x10 |
| 7 | 8 | 34 | 0x1A |
| 8 | 13 | 55 | 0x2A |
| 9 | 21 | 89 | 0x44 |
| 10 | 34 | 144 | 0x6E |
| 11 | 55 | 233 | 0xB2 |

**Table 1:** Test Cases for Software Assignment 5

Assembly Source Code

; -------------------------------------------------------

; Software Assignment 5 - Arrays

; Engineers: Stan Carpenco & Luis Gomez

; Date; 7 February 2019

; Description:

; the following program stores the first 14 values

; of the Fibonacci sequence into the ScratchRam, then,

; it computes the difference between all the numbers that

; are 3 places away. These numbers are then

; outputted to port 0x42.

; REGISTERS

; R0= Location of lower number being compared

; R1 = Location of upper number being compared

; R2 = Stores value at Scratch Ram location in adress R0

; R3 = Stores value at Scratch Ram location at adress R1

.EQU OUT\_PORT = 0x42

.DSEG

.ORG 0x00

.DB 0 , 1 , 1 , 2 , 3 , 5, 8 , 13 , 21 , 34 , 55 , 89 , 144 , 233

.CSEG

.ORG 0x1D

**MOV** R0, 0 ; Moves zero into R0

**MOV** R1, 2 ; Moves two into R1

LOOP: LD R2, (R0) ; Loads value into R2 stored in ScratchRam location R0

LD R3, (R1) ; Loads value into R3 stored in ScrarchRam location R1

**SUB** R3, R2 ; R3= R2 - R2

ufc **OUT** R3, OUT\_PORT ; Outputs result to User in port 0X42

**ADD** R0, 1 ; Incriments bottom count by 1

**ADD** R1, 1 ; Incriments upper count by 1

**CMP** R1, 14 ; Compare R1 with upper limmit 14

BRNE **LOOP** ; Branch to loop if != 0