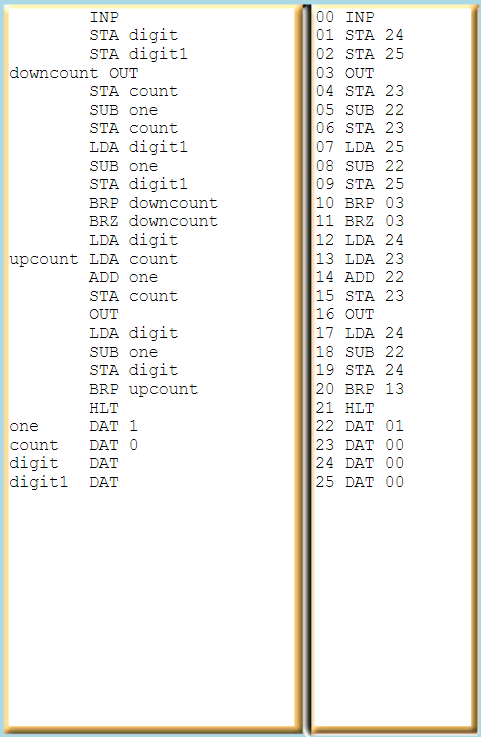
Name: REIMARC G. CORPUZ

Course/Sec: BSCPE IV G-MIX

**UP AND DOWN**

**PROGRAM:**

 INP

STA digit

STA digit1

downcount OUT

STA count

SUB one

STA count

LDA digit1

SUB one

STA digit1

BRP downcount

BRZ downcount

LDA digit

upcount LDA count

ADD one

STA count

OUT

LDA digit

SUB one

STA digit

BRP upcount

HLT

one DAT 1

count DAT 0

digit DAT

digit1 DAT

**PSEUDOCODE:**

1. Input a value and store it in 'digit'.

2. Store the value in 'digit1'.

3. Repeat the following steps until 'digit1' is non-positive:

a. Output 'digit1'.

b. Subtract 1 from 'count' and update it.

c. Subtract 1 from 'digit1' and update it.

4. Repeat the following steps until the 'digit' is non-positive:

a. Add 1 to 'count' and update it.

b. Output 'digit'.

c. Subtract 1 from 'digit' and update it.

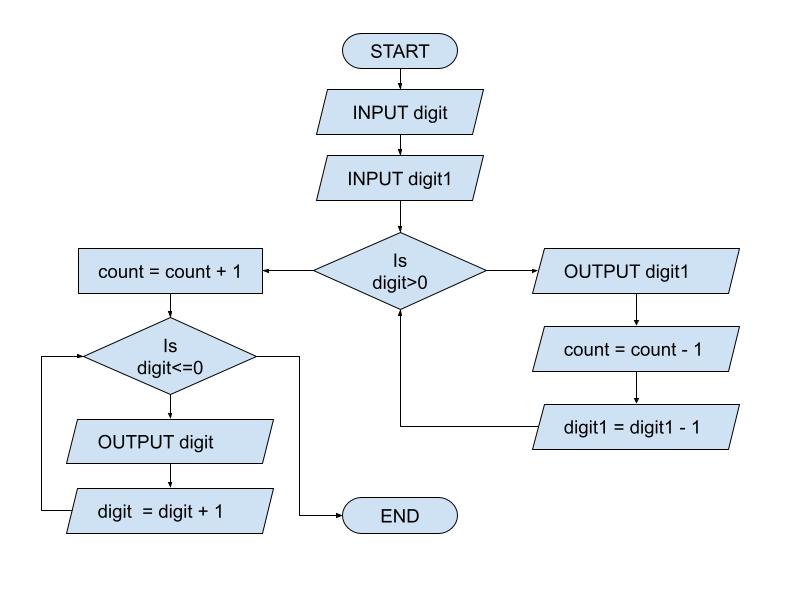
5. Halt the program.

Data Definitions:

- 'one' is initialized to 1.

- 'count' is initialized to 0.

- 'digit' and 'digit1' are uninitialized.

**FLOWCHART:**

**MNEMONIC ASSEMBLER:**

|  |  |
| --- | --- |
| **MAILBOX** | **MNEMONIC** |
| 00 | INP |
| 01 | STA 24 |
| 02 | STA 25 |
| 03 | OUT |
| 04 | STA 23 |
| 05 | SUB 22 |
| 06 | STA 23 |
| 07 | LDA 25 |
| 08 | SUB 22 |
| 09 | STA 25 |
| 10 | BRP 03 |
| 11 | BRZ 03 |
| 12 | LDA 24 |
| 13 | LDA 23 |
| 14 | ADD 22 |
| 15 | STA 23 |
| 16 | OUT |
| 17 | LDA 24 |
| 18 | SUB 22 |
| 19 | STA 24 |
| 20 | BRP 13 |
| 21 | HLT |
| 22 | DAT 01 |
| 23 | DAT 00 |
| 24 | DAT 00 |
| 25 | DAT 00 |

**LMC:**

INP *// Input a value from the user*

STA 24 *// Store the input value in memory location 'digit'*

STA 25 *// Store the input value in memory location 'digit1'*

downcount  *// Label for the loop to decrement 'digit1' and 'count'*

STA 23 *// Store the current value of 'count' in memory location 'count'*

SUB 22 *// Subtract 1 from 'count'*

STA 23 *// Store the updated value of 'count' back in memory location 'count'*

LDA 25 *// Load the current value of 'digit1' into the accumulator*

SUB 22 *// Subtract 1 from 'digit1'*

STA 25 *// Store the updated value of 'digit1' back in memory location 'digit1'*

BRP downcount *// Branch if the value of the accumulator is positive (continue loop)*

BRZ downcount *// Branch if the value of the accumulator is zero (continue loop)*

*// At this point, 'digit1' and 'count' have been decremented to zero*

upcount *// Label for the loop to increment 'count' and 'digit'*

LDA 24 *// Load the current value of 'count' into the accumulator*

ADD 22 *// Add 1 to 'count'*

STA 24 *// Store the updated value of 'count' back in memory location 'count'*

OUT *// Output the value of 'count'*

LDA 24 *// Load the current value of 'count' into the accumulator for comparison*

SUB 22 *// Subtract 1 from 'digit'*

STA 24 *// Store the updated value of 'digit' back in memory location 'digit'*

BRP upcount *// Branch if the value of the accumulator is positive (continue loop)*

HLT *// Halt the program*

one *// Data section, 'one' is initialized with value 1*

DAT 01

count  *// Data section, 'count' is initialized with value 0*

DAT 00

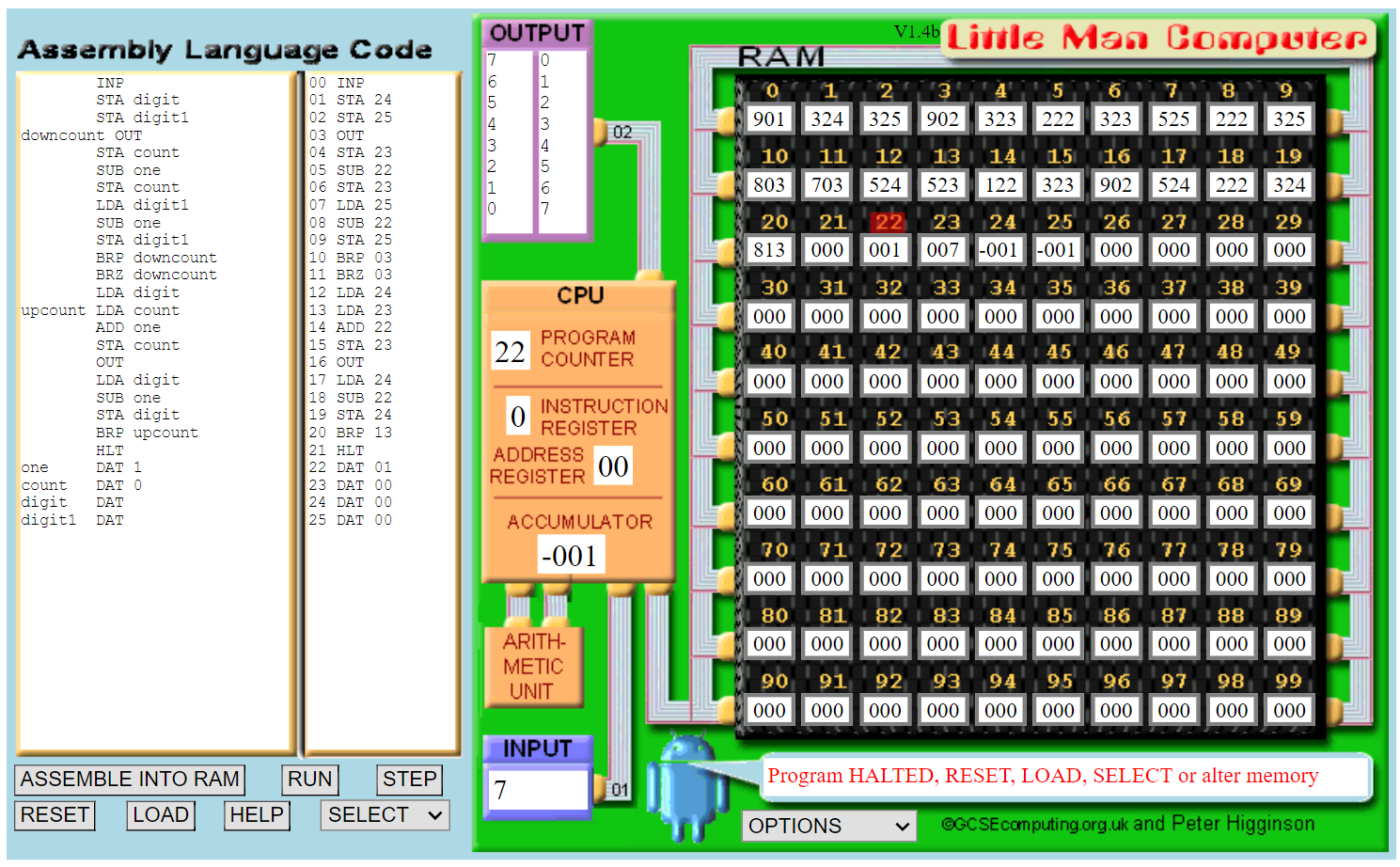
digit *// Data section, 'digit' is initialized with value 0*

digit1 *// Data section, 'digit1' is initialized with value 0*

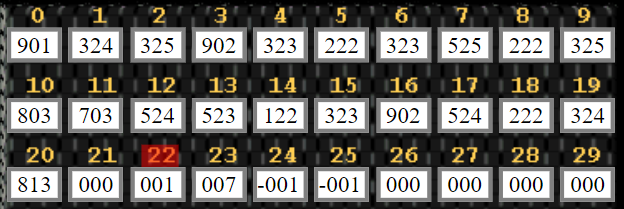
**ADDRESS:**

|  |  |
| --- | --- |
| **OPCODE** | **ADDRESS** |
| 901 | - |
| 302 | 24 |
| 312 | 25 |
| 902 | - |
| 301 | 23 |
| 801 | 22 |
| 301 | 23 |
| 501 | 25 |
| 801 | 22 |
| 603 | 03 |
| 603 | 03 |
| 201 | 24 |
| 101 | 23 |
| 201 | 22 |
| 301 | 23 |
| 902 | - |
| 201 | 24 |
| 801 | 22 |
| 302 | 24 |
| 613 | 13 |
| 000 | - |

**SCREENSHOT OF LMC SIMULATOR ASSEMBLER:**



**SCREENSHOT OF LMC MAILBOXES:**



**CONSOLE MESSAGE:**

1. RUN/STEP your program, SELECT, LOAD or edit program
2. FETCH CYCLE - get current instruction and add 1 to PC
3. INPUT required
4. INPUT value loaded into accumulator
5. STORE value in accumulator at RAM address 24
6. STORE value in accumulator at RAM address 25
7. OUTPUT value stored in accumulator
8. STORE value in accumulator at RAM address 23
9. SUBTRACT from accumulator the contents of RAM address 22
10. STORE value in accumulator at RAM address 23
11. LOAD into accumulator the contents of RAM address 25
12. SUBTRACT from accumulator the contents of RAM address 22
13. STORE value in accumulator at RAM address 25
14. BRANCH (if zero or positive) to memory address 3
15. OUTPUT value stored in accumulator
16. (Same process until 0 is displayed in OUTPUT, same as in going upward) then;
17. Program HALTED, RESET, LOAD, SELECT or alter memory

**VERILOG PROGRAM FOR ELEVATOR:**

module elevator(CLOCK\_50, KEY, SW, HEX0, HEX1, HEX2, HEX3, LEDG);

input CLOCK\_50;

input [3:0] KEY;

input [1:0] SW;

output [6:0] HEX0, HEX1, HEX2, HEX3;

output [7:0] LEDG;

reg [30:0] counter;

reg [6:0] number0;

reg [7:0] state;

reg [2:0] CURRENTFLOOR = 3'b001;

reg [2:0] INPUTFLOOR = 3'b001;

assign HEX0 = {1'b1, number0};

assign HEX1 = {1'b1, number0};

assign HEX2 = {1'b1, number0};

assign HEX3 = {1'b1, number0};

assign LEDG = state;

parameter n1 = 7'b1111001; // DISPLAY 1

parameter n2 = 7'b0100100; // DISPLAY 2

parameter n3 = 7'b0110000; // DISPLAY 3

parameter n4 = 7'b0011001; // DISPLAY 4

always @(posedge CLOCK\_50)

begin

if (SW[0] == 1 || SW[1] == 1)

begin

if (KEY[3] == 0)

INPUTFLOOR = 3'b100; // 4th floor

else if (KEY[2] == 0)

INPUTFLOOR = 3'b011; // 3rd floor

else if (KEY[1] == 0)

INPUTFLOOR = 3'b010; // 2nd floor

else if (KEY[0] == 0)

INPUTFLOOR = 3'b001; // 1st floor

end

if(counter == 50000000)

begin

if (SW[0] == 1 && SW[1] == 1)

begin

end

else if (CURRENTFLOOR < INPUTFLOOR && SW[0] == 1)

CURRENTFLOOR = CURRENTFLOOR + 1;

else if (CURRENTFLOOR > INPUTFLOOR && SW[1] == 1)

CURRENTFLOOR = CURRENTFLOOR - 1;

end

else

counter <= counter + 1;

case (CURRENTFLOOR)

3'b100: number0 <= n4; // 4th floor

3'b011: number0 <= n3; // 3rd floor

3'b010: number0 <= n2; // 2nd floor

3'b001: number0 <= n1; // 1st floor

default: ;

endcase

if (CURRENTFLOOR == INPUTFLOOR)

begin

if (counter == 0)

state <= 8'b00000000;

else if (counter == 150000000)

state <= 8'b11111111;

else if (counter > 150000000)

state <= 8'b00000000;

end

else

state <= 8'b00000000;

end

endmodule