CPE 233: Software assignment 6

Prof. Bridget Benson

Luis Gomez, Stan Carpenco

## Behavior

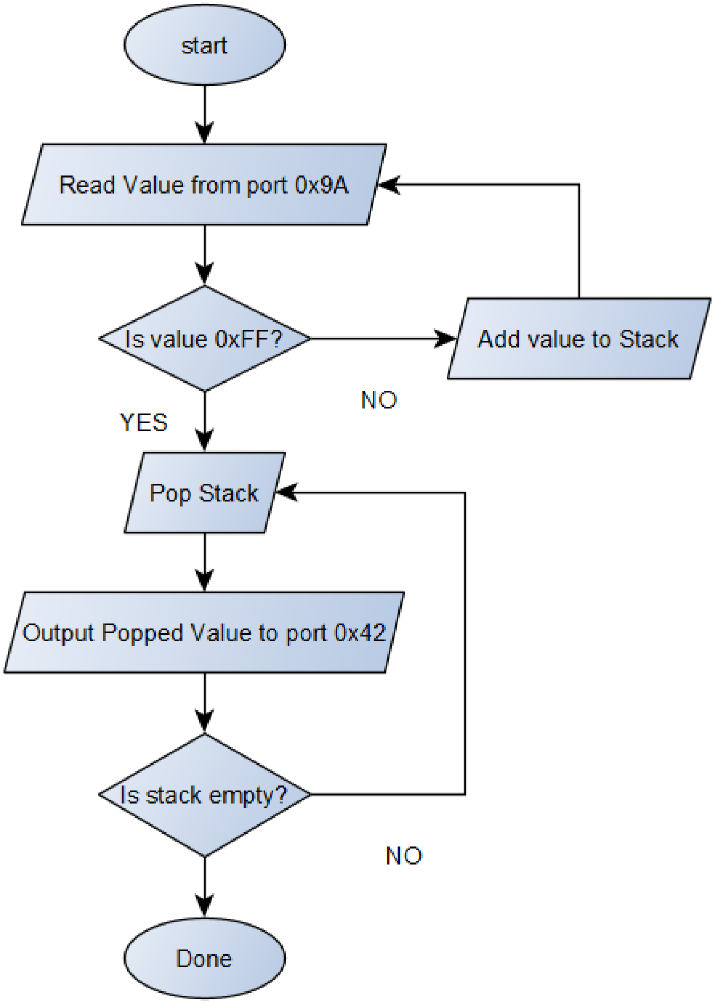
In this assignment, we wrote two Assembly programs using the RAT simulator.

**Program 1**: A Rat Assembly implementation of a Stack Data Structure (Last In First Out behavior). This program continuously reads 8-bit inputs from port 0x9A, adding each value to the 'Top' of the stack (Memory location) until 0xFF is read. When 0xFF is input, the program begins to 'Pop' and output values (port 0x42) opposite in order to which they were input.

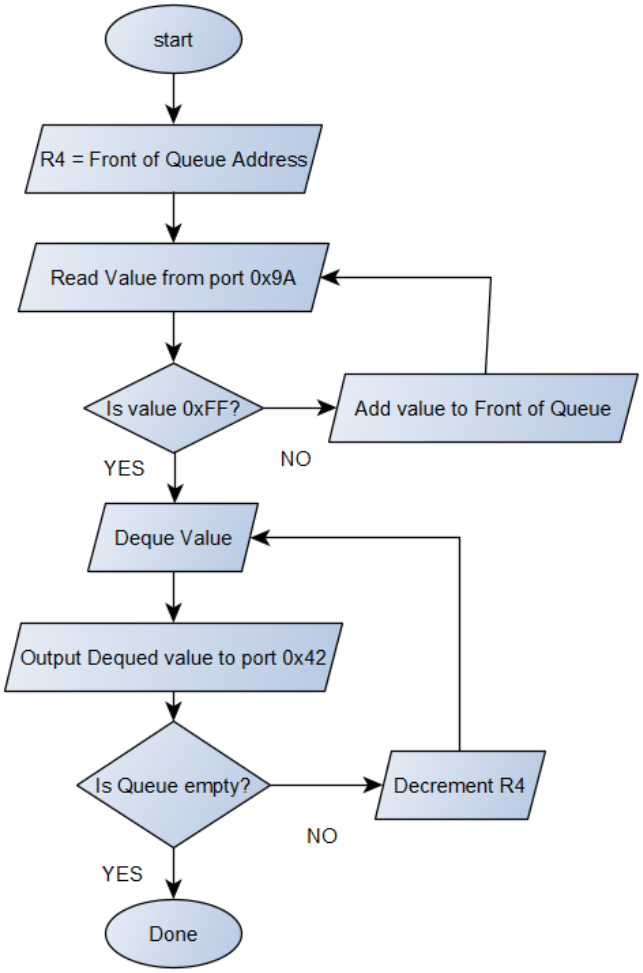
**Program 2:** A Rat Assembly implementation of a Queue Data Structure (First In First Out behavior). This program continuously reads 8-bit inputs from port 0x9A, adding each value to the 'Front' of the Queue (Memory location) until 0xFF is read. When 0xFF is input, the program begins to 'Deque' and output values (port 0x42) in order by which they were input.

## Flowchart

**Program 1**



## Program 2


start
R4 = Front of Queue Address
Read Value from port 0x9A
Is value 0xFF?
Add value to Front of Queue
NO
YES
Deque Value
Output Dequed value to port 0x42
Is Queue empty?
Decrement R4
NO
YES
Done

## Verification

**Program 1 Verification**

| | | | Memory | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Input (hex) | Output | R3 (stack size) | 0xF8 | 0xF9 | 0xFA | 0xFB | 0xFC | 0xFD | 0xFE | 0xFF |
| 01 | -- | 01 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 001 |
| 02 | -- | 02 | 000 | 000 | 000 | 000 | 000 | 000 | 002 | 001 |
| 03 | -- | 03 | 000 | 000 | 000 | 000 | 000 | 003 | 002 | 001 |
| 04 | -- | 04 | 000 | 000 | 000 | 000 | 004 | 003 | 002 | 001 |
| 05 | -- | 05 | 000 | 000 | 000 | 005 | 004 | 003 | 002 | 001 |
| 06 | -- | 06 | 000 | 000 | 006 | 005 | 004 | 003 | 002 | 001 |
| 07 | -- | 07 | 000 | 007 | 006 | 005 | 004 | 003 | 002 | 001 |
| 08 | -- | 08 | 008 | 007 | 006 | 005 | 004 | 003 | 002 | 001 |
| FF | 08 | 07 | 000 | 007 | 006 | 005 | 004 | 003 | 002 | 001 |
| -- | 07 | 06 | 000 | 000 | 006 | 005 | 004 | 003 | 002 | 001 |
| -- | 06 | 05 | 000 | 000 | 000 | 005 | 004 | 003 | 002 | 001 |
| -- | 05 | 04 | 000 | 000 | 000 | 000 | 004 | 003 | 002 | 001 |
| -- | 04 | 03 | 000 | 000 | 000 | 000 | 000 | 003 | 002 | 001 |
| -- | 03 | 02 | 000 | 000 | 000 | 000 | 000 | 000 | 002 | 001 |
| -- | 02 | 01 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 001 |
| -- | 01 | 00 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 |

**Program 2 Verification**

| | | | | Memory | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Input (hex) | Output | R3 (Queue size) | R4 (Front of Queue) | 0xF8 | 0xF9 | 0xFA | 0xFB | 0xFC | 0xFD | 0xFE | 0xFF |
| 01 | -- | 01 | 0xFF | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 001 |
| 02 | -- | 02 | 0xFF | 000 | 000 | 000 | 000 | 000 | 000 | 002 | 001 |
| 03 | -- | 03 | 0xFF | 000 | 000 | 000 | 000 | 000 | 003 | 002 | 001 |
| 04 | -- | 04 | 0xFF | 000 | 000 | 000 | 000 | 004 | 003 | 002 | 001 |
| 05 | -- | 05 | 0xFF | 000 | 000 | 000 | 005 | 004 | 003 | 002 | 001 |
| 06 | -- | 06 | 0xFF | 000 | 000 | 006 | 005 | 004 | 003 | 002 | 001 |
| 07 | -- | 07 | 0xFF | 000 | 007 | 006 | 005 | 004 | 003 | 002 | 001 |
| 08 | -- | 08 | 0xFF | 008 | 007 | 006 | 005 | 004 | 003 | 002 | 001 |
| FF | 01 | 07 | 0xFE | 008 | 007 | 006 | 005 | 004 | 003 | 002 | 000 |
| -- | 02 | 06 | 0xFD | 008 | 007 | 006 | 006 | 004 | 003 | 000 | 000 |
| -- | 03 | 05 | 0xFC | 008 | 007 | 006 | 006 | 004 | 000 | 000 | 000 |
| -- | 04 | 04 | 0xFB | 008 | 007 | 006 | 006 | 000 | 000 | 000 | 000 |
| -- | 05 | 03 | 0xFA | 008 | 007 | 006 | 000 | 000 | 000 | 000 | 000 |
| -- | 06 | 02 | 0xF9 | 008 | 007 | 000 | 000 | 000 | 000 | 000 | 000 |
| -- | 07 | 01 | 0xF8 | 008 | 000 | 000 | 000 | 000 | 000 | 000 | 000 |
| -- | 08 | 00 | 0xF7 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 |

## Source Code

PROGRAM 1: Stack Implementation

```
.EQU IN_PORT = 0x9A
.EQU OUT_PORT = 0x42
.CSEG
.ORG 0x01
;-------------------
; registers used:
;   R1- Input
;   R2- Ouput
;   R3- stack size counter
;-------------------
start:   IN R1,IN_PORT;READ: read value from port 0x9A
         CMP R1,0xFF  ;Is value 0xFF?
         BREQ popped
         PUSH R1          ;If not, push value to stack, return to
READ
         ADD R3,0x01      ; increment stack size
         BRN start
popped: POP R2            ;Popped: pop stack
         OUT R2,OUT_PORT   ;output popped value
         SUB R3, 0x01 ;decrement stack size
         CMP R3, 0x00
         BRNE popped  ;is stack empty? return to Popped
```

## PROGRAM 2: Queue Implementation

```
.EQU IN_PORT = 0x9A
.EQU OUT_PORT = 0x42
.CSEG
.ORG 0x01
;-------------------------
; Registers Used:
;    R1- input
;    R2- output
;    R3- Queue size counter
;    R4- Front of Queue address
;-------------------------
         MOV R4, 0xFF  ; Front of Queue address
queue:   IN R1, IN_PORT
         CMP R1, 0xFF  ; Check if value read is 0xFF
         BREQ deque
         PUSH R1           ; Push to stack
         ADD R3,0x01       ; increment stack size
         BRN queue
deque:   LD R2,(R4)        ; deque from front
         OUT R2,OUT_PORT
         SUB R4,0x01       ; shift queue address
         SUB R3,0x01       ; decrement stack size
         CMP R3,0x00       ; is stack empty?
         BRNE deque
```