

SIMON

A “MEMORY GAME”

When you're learning coding and
you finally understand why the
website is called stack overflow

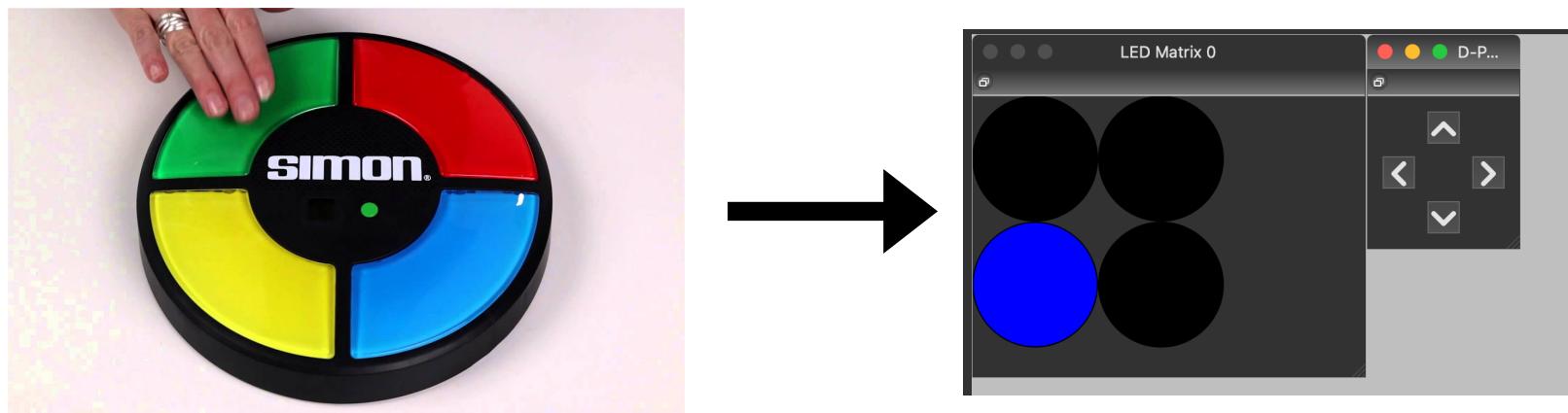


By: Muntasir Munem

HOW TO PLAY SIMON

SIMON IS A MEMORY GAME THAT REQUIRES YOU, THE PLAYER, TO REMEMBER THE ORDER IN WHICH LED'S WERE FLASHED.

YOU MUST MEMORIZE THE ORDER OF THE COLOURS THAT WERE FLASHED AND INPUT THEM USING THE D-PAD (MORE ON THIS LATER).



DOWNLOADING RIPES

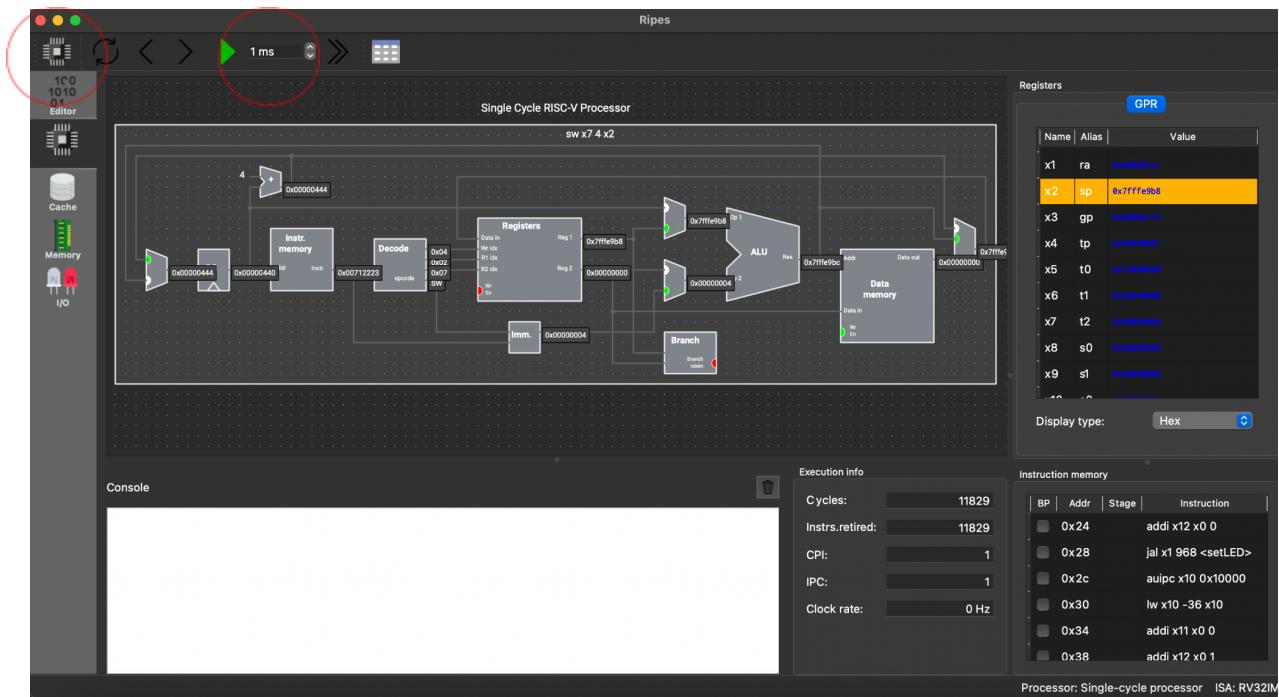
IN ORDER TO PLAY THIS GAME, YOU
MUST DOWNLOAD RIPES, WHICH IS
USED TO VIEW COMPUTER
ARCHITECTURE AND EDIT ASSEMBLY
CODE.

PLEASE OPEN THIS [LINK](#) AND
DOWNLOAD RIPES FOR YOUR
RESPECTIVE OS:

<https://github.com/mortbopet/Ripes/releases/tag/v2.2.3>

SETTING UP RIPES

ONCE DOWNLOADED, OPEN RIPES AND
CLICK ON SELECT PROCESSOR:



THEN CHOOSE SINGLE CYCLE
PROCESSOR, CLICK OK.
CHANGE YOUR AUTO CLOCK
INTERVAL TO 1MS.

OPENING THE GAME FILE

TO OPEN THE GAME FILE:
FILE → LOAD PROGRAM → SELECT
SOURCE FILE → CLICK OPEN →
NAVIGATE TO THE .s GAME FILE →
CLICK OPEN

YOUR SCREEN SHOULD LOOK LIKE
THIS

The screenshot shows a debugger interface with the following details:

- Processor:** Single-cycle processor, ISA: RV32IM.
- Registers:** A table showing register values from x0 to x18.
- Memory:** A visualization of memory with sections for Processor, Cache, and Memory.
- Source code:** The assembly code for the game file.
- Registers:** A table showing register values from x0 to x18.
- Display type:** Hex.

Registers Table:

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7FFFFFFF
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000000
x11	a1	0x00000000
x12	a2	0x00000000
x13	a3	0x00000000
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Assembly Code:

```
1 .data
2 sequence: .byte 1,1,1
3 black: .word 0x00000000
4 green: .word 0x00000000
5 red: .word 0x00000000
6 blue: .word 0x00000000
7 purple: .word 0x00000000
8 newline: .string "\n"
9 here: .string "here\n"
10 nowhere: .string "nowhere\n"
11 question: .string "Would you like to play again? Enter 1 for YES\\Enter anything else for NO\\n"
12 sequencestr: .string "sequence is "
13 question: .string "Would you want to play again? Enter 1 for YES\\Enter anything else for NO\\n"
14
15
16 .globl main
17 .text
18
19 addi $t0, $x0, 1           # Used for Enhancement B Option 1
20
21 main:
22     # TODO: Before we deal with the LEDs, we need to generate a random
23     # sequence of numbers that we will use to indicate the button/LED
24     # to light up. For example, we can have 0 for UP, 1 for DOWN, 2 for
25     # LEFT, and 3 for RIGHT. Store the sequence in memory. We provided
26     # a declaration above that you can use if you want.
27     # Hint: Use the rand function provided to generate each number
28
29     # Set everything to black first
30     la $t1, black
31     la $t2, black
32     la $t3, black
33     jal $t0, setLED
34     la $t1, black
35     la $t2, black
36     la $t3, black
37     jal $t0, setLED
38     la $t1, black
39     la $t2, black
40     la $t3, black
41     jal $t0, setLED
42     la $t1, black
43     la $t2, black
44     la $t3, black
45     jal $t0, setLED
46
47
48     li $t0, sequence
49     li $t1, 1
50     li $t2, 4
51     li $t3, 1
52
53     li $t4, sequencestr
54
```

RUNNING THE GAME

Source code

```

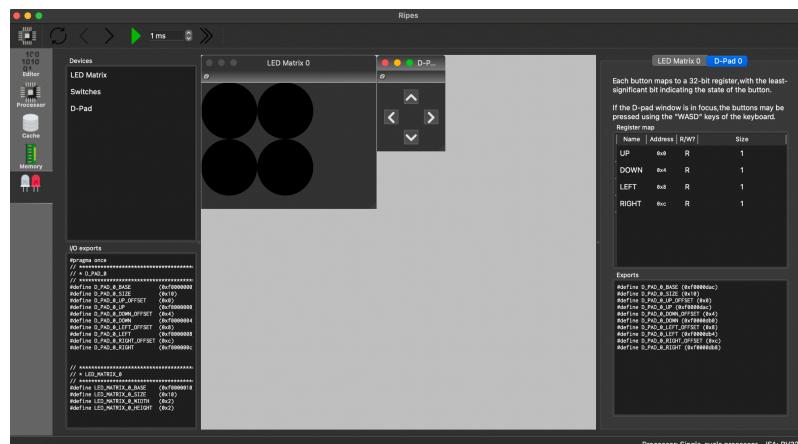
1 .data
2 sequence: byte $A,B,C
3 count: word 0
4 black: .word 0x000000
5 green: .word 0x00ff00
6 red: .word 0x0000ff
7 blue: .word 0x000000ff
8 purple: .word 0x0000ff00
9 newline: .string "\n"
10 space: .string " "
11 nothere: .string "nothere\n"
12 sequencestr: .string "sequence is "
13 question: .string "Do you want to play again? Enter 1 for YES/Enter anything else for NO\n"
14 
15 .globl main
16 .text
17 
18 addi x4, sp, -4           # Used for Enhancement B Option 1
19 
20 main:
21     # TODO: Before we deal with the LEDs, we need to generate a random
22     # sequence of numbers that we will use to indicate the button/LED
23     # to light up. For example, we can have 0 for UP, 1 for DOWN, 2 for
24     # LEFT, and 3 for RIGHT. We leave the sequence in memory. We provided
25     # a declaration so that you know what you want.
26     # Hint: Use the rand function provided to generate each number
27 
28     # Set everything to black first
29     ldi x0, black
30     ldi x1, black
31     ldi x2, black
32     ldi x3, black
33     setLED
34     ldi x0, black
35     ldi x1, black
36     ldi x2, black
37     ldi x3, black
38     setLED
39     ldi x0, black
40     ldi x1, black
41     ldi x2, black
42     ldi x3, black
43     ldi x0, black
44     ldi x1, black
45     ldi x2, black
46     ldi x3, black
47     setLED
48 
49     ldi x0, sequence
50     ldi x1, 0             # t0 is the base address of sequence array
51     ldi x2, 4             # t1 = 0
52     ldi x3, 4             # t2 = 4
53     ldi x4, sequencestr
54 
```

Input type: Assembly C Executable code View mode: Binary Disassembled GPR

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7fffffe0
x3	gp	0x00000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000000
x11	a1	0x00000000
x12	a2	0x00000000
x13	a3	0x00000000
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Processor: Single-cycle processor ISA: RV32IM

SIMPLY CLICK THE GREEN RUN
BUTTON
THEN CLICK ON I/O ON THE LEFT BAR
YOUR SCREEN SHOULD LOOK LIKE
THIS:



PLAYING THE GAME

SIMPLY WATCH AS THE COLOURS
CHANGE ON THE LED MATRIX AND
ONCE THE PATTERN IS FINISHED,
INPUT THE PATTERN USING THE DPAD
ON THE RIGHT OF THE MATRIX

IF YOU MESS UP, THE LED MATRIX
WILL FLASH RED
IF YOU WIN, THE LED MATRIX WILL
FLASH GREEN

CONTROLS

UP REFERS TO: **RED**

DOWN REFERS TO: **GREEN**

LEFT REFERS TO: **BLUE**

RIGHT REFERS TO: **PURPLE**

ONCE YOU FINISH ONE ROUND OF THE GAME, CLICK ON THE PROCESSOR ON THE LEFT PANEL TO SEE THE CONSOLE

IF YOU'D LIKE TO PLAY AGAIN,
SIMPLY INPUT 1, IF NOT, ENTER
ANYTHING ELSE.

**THANK YOU FOR
PLAYING THE
GAME!
ENJOY!**