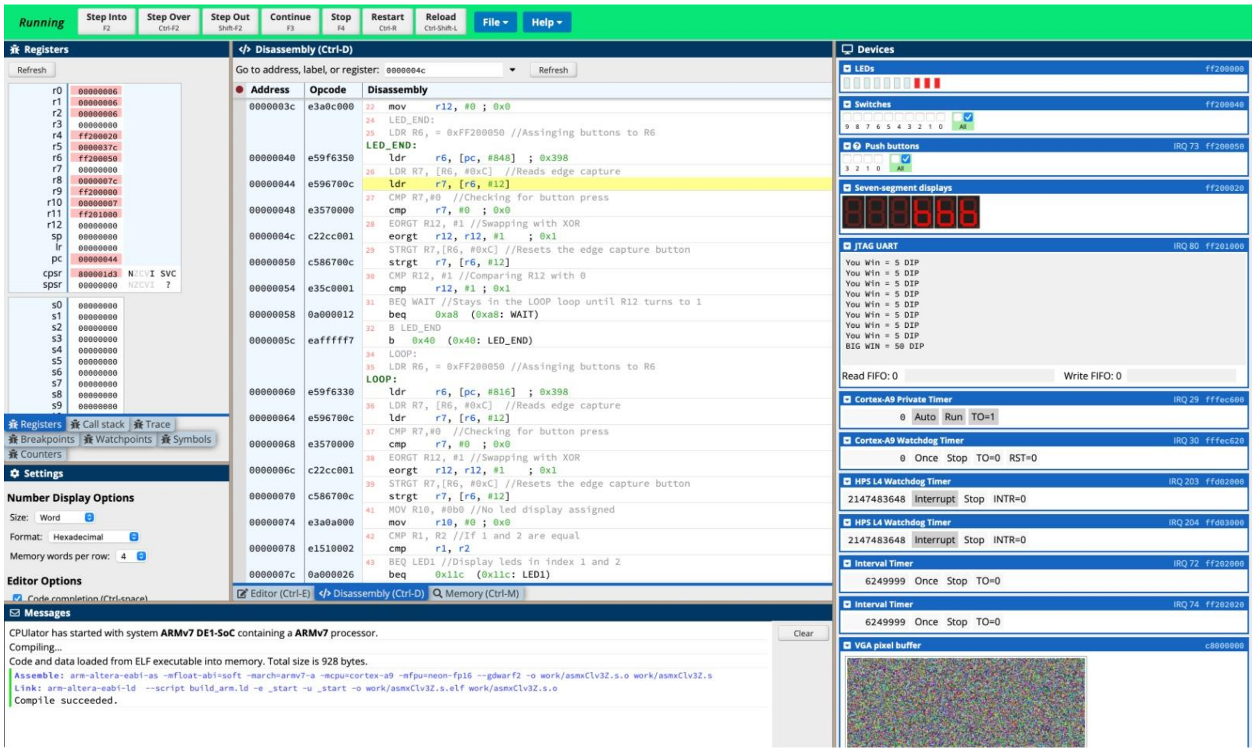
Explanation of the project

A slot machine is implemented in this code. Three rows of LEDs on the slot machine serve as the spinning reels. By pushing a button, the user can start the slot machine, leading the 7-segment display to begin spinning. A private timer has been programmed with the motion of spinning. The numbers on each row of the 7-segment display successively increase from 0 to 9 while the timer runs, replicating the spinning motion.
The spinning animation ends when the user pushes the button once more, and the final results are shown on the 7-segment display. The combination of values determines which LEDs are turned on. A particular LED pattern is shown and a message is produced from the JTAG UART if the values on the first and second rows are equal, if the values on the first and third rows are equal, if the values on the second and third rows are equal and if the values on all rows are equal.


Running
Step Into F2
Step Over Ctrl-F2
Step Out Shift-F2
Continue F3
Stop F4
Restart Ctrl-R
Reload Ctrl-Shift-L
File
Help
Registers
Refresh
Disassembly (Ctrl-D)
Devices
LEDs ff200000
Switches ff200040
Push buttons IRQ 73 ff200050
Seven-segment displays ff200020
JTAG UART IRQ 80 ff201000
Cortex-A9 Private Timer IRQ 29 fffec600
Cortex-A9 Watchdog Timer IRQ 30 fffec620
HPS L4 Watchdog Timer IRQ 203 ffd02000
HPS L4 Watchdog Timer IRQ 204 ffd03000
Interval Timer IRQ 72 ff202000
Interval Timer IRQ 74 ff202020
VGA pixel buffer c8000000
Registers
Call stack
Trace
Breakpoints
Watchpoints
Symbols
Counters
Settings
Editor (Ctrl-E)
Disassembly (Ctrl-D)
Memory (Ctrl-M)
Messages
Clear

```
r0    00000006
r1    00000006
r2    00000006
r3    00000000
r4    ff200020
r5    0000037c
r6    ff200050
r7    00000000
r8    0000007c
r9    ff200000
r10   00000007
r11   ff201000
r12   00000000
sp    00000000
lr    00000000
pc    00000044
cpsr  800001d3  NZCVI SVC
spsr  00000000  NZCVI  ?

s0    00000000
s1    00000000
s2    00000000
s3    00000000
s4    00000000
s5    00000000
s6    00000000
s7    00000000
s8    00000000
s9    00000000
```

Go to address, label, or register: 0000004c

```
Address   Opcode    Disassembly
0000003c  e3a0c000     mov     r12, #0 ; 0x0
                       LED_END:
                       LDR R6, = 0xFF200050 //Assinging buttons to R6
                    LED_END:
00000040  e59f6350     ldr     r6, [pc, #848]  ; 0x398
                       LDR R7, [R6, #0xC]  //Reads edge capture
00000044  e596700c     ldr     r7, [r6, #12]
                       CMP R7,#0  //Checking for button press
00000048  e3570000     cmp     r7, #0  ; 0x0
                       EORGT R12, #1 //Swapping with XOR
0000004c  c22cc001     eorgt   r12, r12, #1     ; 0x1
                       STRGT R7,[R6, #0xC] //Resets the edge capture button
00000050  c586700c     strgt   r7, [r6, #12]
                       CMP R12, #1 //Comparing R12 with 0
00000054  e35c0001     cmp     r12, #1 ; 0x1
                       BEQ WAIT //Stays in the LOOP loop until R12 turns to 1
00000058  0a000012     beq     0xa8  (0xa8: WAIT)
                       B LED_END
0000005c  eafffff7     b   0x40  (0x40: LED_END)
                       LOOP:
                       LDR R6, = 0xFF200050 //Assinging buttons to R6
                    LOOP:
00000060  e59f6330     ldr     r6, [pc, #816]  ; 0x398
                       LDR R7, [R6, #0xC]  //Reads edge capture
00000064  e596700c     ldr     r7, [r6, #12]
                       CMP R7,#0  //Checking for button press
00000068  e3570000     cmp     r7, #0  ; 0x0
                       EORGT R12, #1 //Swapping with XOR
0000006c  c22cc001     eorgt   r12, r12, #1     ; 0x1
                       STRGT R7,[R6, #0xC] //Resets the edge capture button
00000070  c586700c     strgt   r7, [r6, #12]
                       MOV R10, #0b0 //No led display assigned
00000074  e3a0a000     mov     r10, #0 ; 0x0
                       CMP R1, R2 //If 1 and 2 are equal
00000078  e1510002     cmp     r1, r2
                       BEQ LED1 //Display leds in index 1 and 2
0000007c  0a000026     beq     0x11c  (0x11c: LED1)
```

```
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
BIG WIN = 50 DIP
```

Read FIFO: 0 Write FIFO: 0

Cortex-A9 Private Timer: 0 Auto Run TO=1
Cortex-A9 Watchdog Timer: 0 Once Stop TO=0 RST=0
HPS L4 Watchdog Timer: 2147483648 Interrupt Stop INTR=0
HPS L4 Watchdog Timer: 2147483648 Interrupt Stop INTR=0
Interval Timer: 6249999 Once Stop TO=0
Interval Timer: 6249999 Once Stop TO=0

Number Display Options
Size: Word
Format: Hexadecimal
Memory words per row: 4

Editor Options
[✓] Code completion (Ctrl-space)

```
CPUlator has started with system ARMv7 DE1-SoC containing a ARMv7 processor.
Compiling...
Code and data loaded from ELF executable into memory. Total size is 928 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmxClv3Z.s.o work/asmxClv3Z.s
Link: arm-altera-eabi-ld  --script build_arm.ld -e _start -u _start -o work/asmxClv3Z.s.elf work/asmxClv3Z.s.o
Compile succeeded.
```

Figure 1. All Screen

BUTTON_STATUS:

The BUTTON_STATUS subroutine is watching over the push buttons' status. It starts by reading the edge capture register to check if any button press event has occurred. A bitwise XOR operation with 1 is used to modify R12's value if a button press is detected, thereby switching it between 0 and 1. Each time a button push is detected, the BUTTON_STATUS subroutine uses the XOR technique to toggle the value of R12 between 0 and 1.
The subroutine clears the edge capture register to reset the button press event. To establish if a button push has been detected, it compares the value of R12 with 1. It branches to the SPINNING subroutine if R12 is 1. If not, it restarts the BUTTON_STATUS subroutine and loops back to the beginning to keep track of the buttons.

ACTIVATE_LEDS:

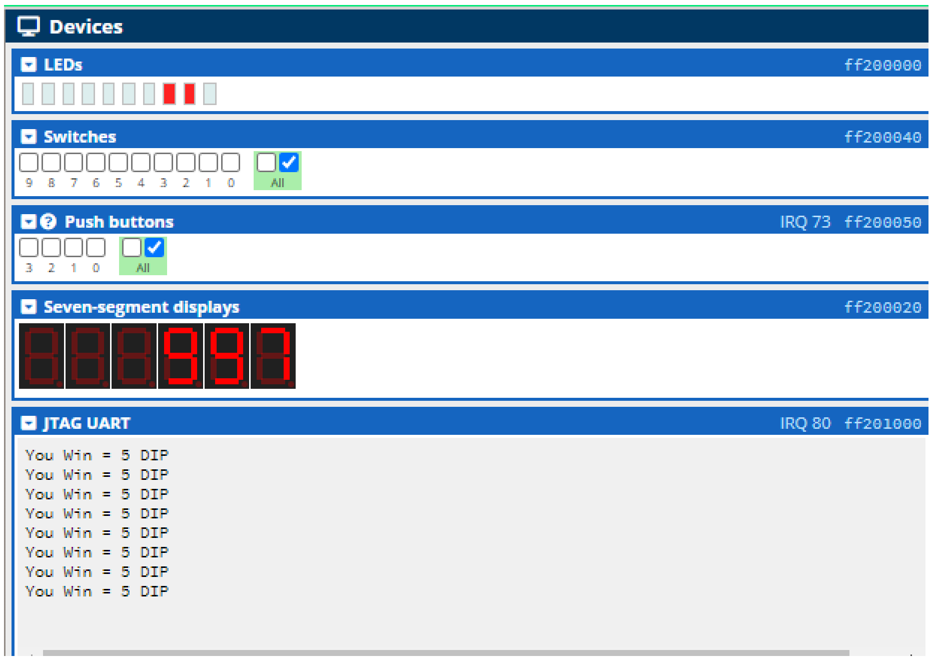
Based on button presses, the ACTIVATE_LEDS subroutine is responsible for turning on the LEDs. To detect any button presses, it first reads the edge capture register. When a button press is detected, the edge capture register is reset and the value of R12 is XORed with 1.
The value of R12 is checked. If it's zero, the ACTIVATE_LEDS subroutine begins over and the wait for a button push continues. If R12 is 1, the LEDs are turned on.
The subroutine sets R10 to 0b0, so that no LED will be seen. Then, it compares R0, R1, and R2 values to determine which LEDs need to be turned on. It branches to certain labels like LED1, LED2, or LED3 to carry out the appropriate operations if certain conditions are satisfied.

SPINNING:

The SPINNING subroutine is responsible for the spinning effect of the 7-segment display.R10 is first set to 0b0, indicating that no LEDs should be shown. Then, to turn off all LEDs, it puts the value of R10 in the LED register.
To determine if the timer flag is set, it examines the timer status. If the flag is not set -R11 equals 0-, it branches back to the beginning of the "SPINNING" subroutine to keep waiting. When the timer flag is set -R11 equals 1-, it resets the timer flag.
The subroutine simulates the spinning effect by increasing the values of R0, R1, and R2. Each value is compared to 10 and resets to 0 if they are equal. Then, it loads the corresponding values into R6, R7, and R8 from the HEXTABLE.
The 7-segment display is represented by R10, which combines the values in R6, R7, and R8. The 7-segment display register stores this value, which activates the spinning 7-segment display.
It checks for button presses and XORs R12's value. When the button is pressed, the ACTIVATE_LEDS subroutine is called; otherwise, it returns to the SPINNING subroutine.

LED 1:

If R1 and R2 are equal to each other, it is checked whether R0 and R1 can be equal. If R0 and R1 are equal, it goes to LED4, if not equal, LED1 lights up and a JTAG UART interface, LED display, and the inscription "You Win = 5 DIP" are displayed. Then, it returns to the BUTTON_STATUS subroutine.

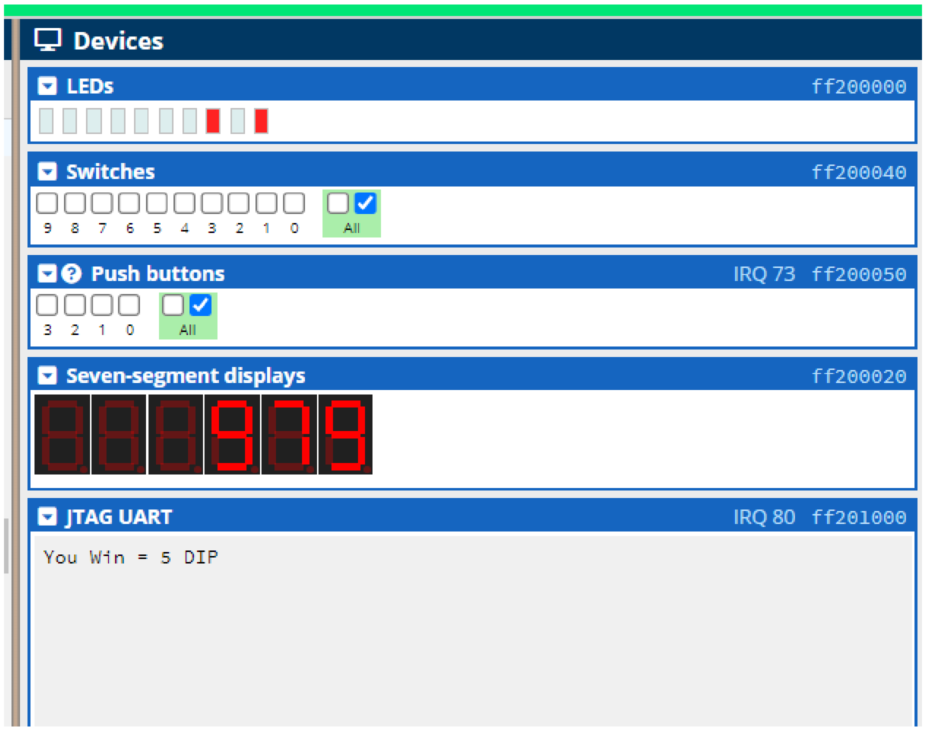


```
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
```

Figure 2. Output for LED1 status

LED 2:

If R0 and R2 are equal to each other, it is checked whether R0 and R1 can be equal. If R0 and R1 are equal, it goes to LED4, if not equal it goes to LED2, and a JTAG UART interface, LED display, and the inscription "You Win = 5 DIP" are displayed. Then, it returns to the BUTTON_STATUS subroutine.

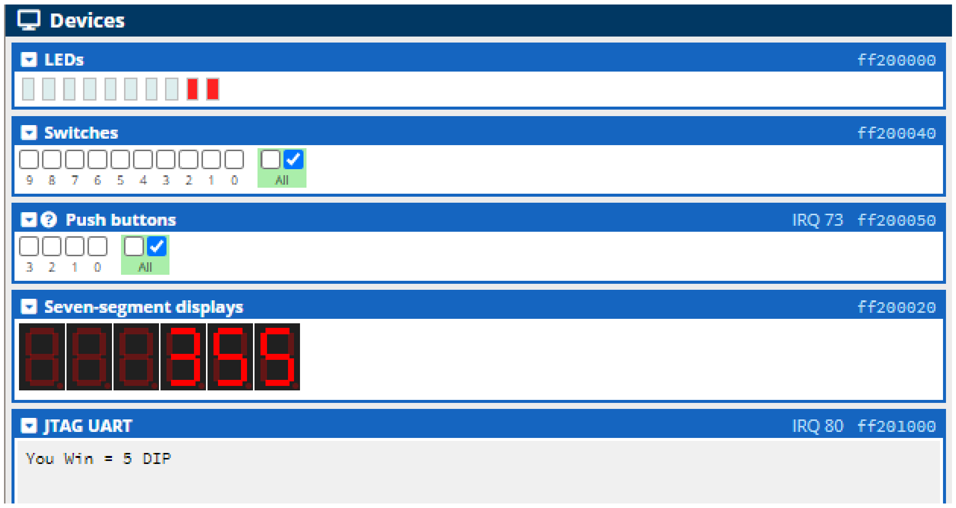

Devices
LEDs ff200000
Switches ff200040
9 8 7 6 5 4 3 2 1 0 All
Push buttons IRQ 73 ff200050
3 2 1 0 All
Seven-segment displays ff200020
JTAG UART IRQ 80 ff201000

```
You Win = 5 DIP
```

Figure 3. Output for LED2 status

LED 3:

If R0 and R1 are equal to each other, it is checked whether R0 and R2 can be equal. If R0 and R2 are equal, it goes to LED4, if not equal, it goes to LED3, and a JTAG UART interface, LED display, and the inscription "You Win = 5 DIP" are displayed. Then, it returns to the BUTTON_STATUS subroutine.

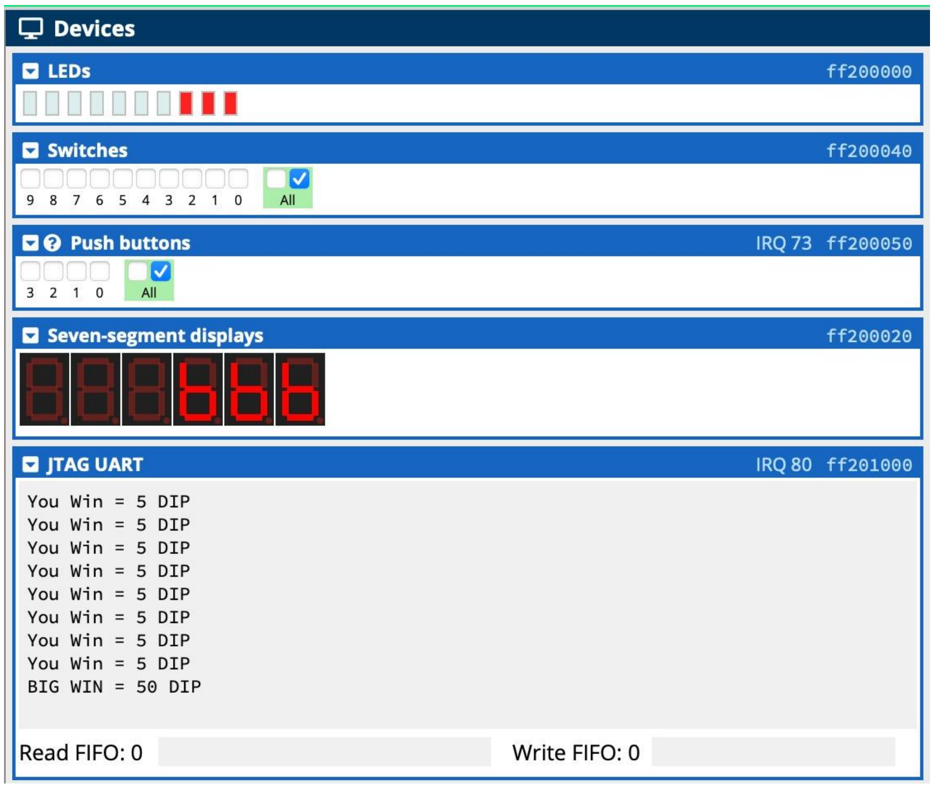

Devices
LEDs ff200000
Switches ff200040
9 8 7 6 5 4 3 2 1 0 All
Push buttons IRQ 73 ff200050
3 2 1 0 All
Seven-segment displays ff200020
JTAG UART IRQ 80 ff201000

```
You Win = 5 DIP
```

Figure 4. Output for LED3 status

LED4:

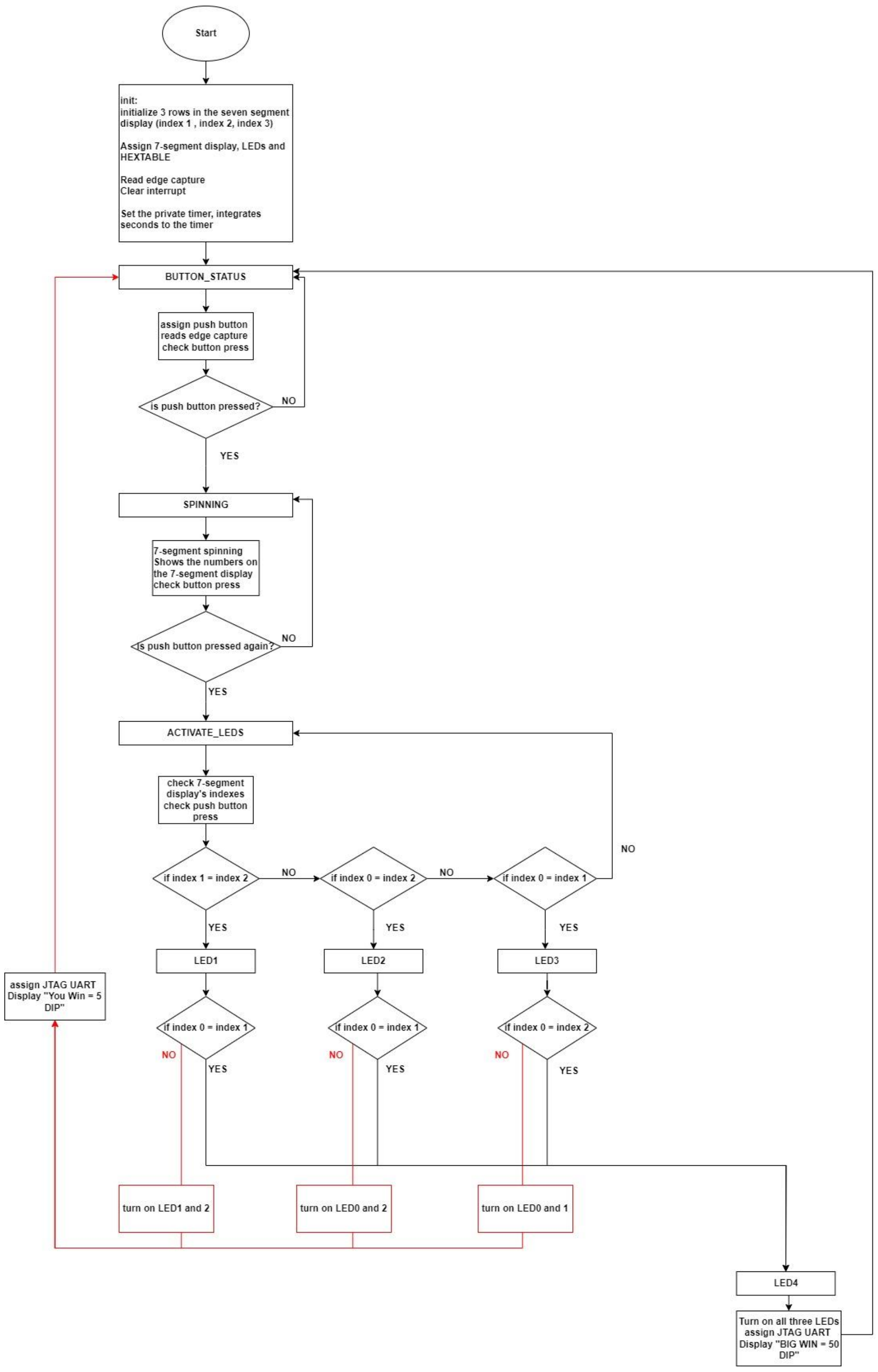
This subroutine is called when one of the LED1, LED2, or LED3 conditions are met. In other words, it is called when R0, R1, and R2 are equal. And JTAG displays the message "BIG WIN = 50 DIP" on the UART interface. Then, it returns to the BUTTON_STATUS subroutine.



```
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
You Win = 5 DIP
BIG WIN = 50 DIP
```

Figure 5. Output for LED4 status

Flowchart of the project


Start
init:
initialize 3 rows in the seven segment display (index 1 , index 2, index 3)
Assign 7-segment display, LEDs and HEXTABLE
Read edge capture
Clear interrupt
Set the private timer, integrates seconds to the timer
BUTTON_STATUS
assign push button reads edge capture check button press
is push button pressed?
NO
YES
SPINNING
7-segment spinning Shows the numbers on the 7-segment display check button press
is push button pressed again?
NO
YES
ACTIVATE_LEDS
check 7-segment display's indexes check push button press
if index 1 = index 2
NO
if index 0 = index 2
NO
if index 0 = index 1
NO
YES
YES
YES
LED1
LED2
LED3
assign JTAG UART Display "You Win = 5 DIP"
if index 0 = index 1
if index 0 = index 1
if index 0 = index 2
NO
YES
NO
YES
NO
YES
turn on LED1 and 2
turn on LED0 and 2
turn on LED0 and 1
LED4
Turn on all three LEDs assign JTAG UART Display "BIG WIN = 50 DIP"