118 Guessing Game Assembly Code

JTAG UART

//define the data section to store strings for game start

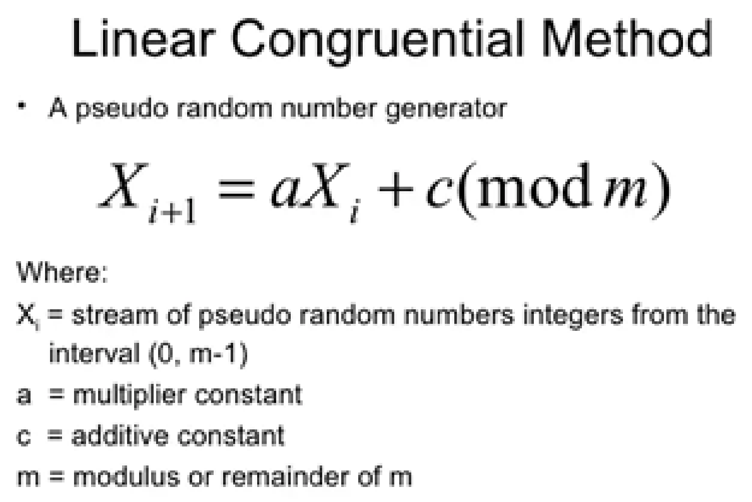
Code for Random # Generator

```
// Generate random # between seed and m
.data

randn: .word 0,0,0,0,0

.align

Seed: .word 0x0//x
Mul_Constant: .word 0x1//a
Mod: .word 0x10//m
Add_Constant: .word 0x7//c

.align

.text

.global _start
_start:

LDR r1, = Seed
LDR r1,[r1]
LDR r2, = Mul_Constant
LDR r2,[r2]
LDR r3, = Mod
LDR r3,[r3]
LDR r4, = Add_Constant
LDR r4,[r4]
MOV r0, #0 //iterator
LDR r5, = randn // load in address of the array
SUB r3, #1 // m-1
BL rand_num
```

```
rand_num:
   PUSH {LR}
   loop:
         CMP r0, #20
         BEQ DONE
         MUL r1, r2 //aXi
         ADD r6, r1,r4 // (aXi + c)
         AND r7, r6, r3 // (aXi + c) %m
         STR r7,[r5,r0]
         MOV r1, r7
         ADD r0, #4
         B loop
   endloop:
   BX LR

DONE: B DONE
```

Code for array looping

```
MOV r0, #0 ;initialise loop index to 0
MOV r1, #100 ;number of iterations

Loop:
ADD r0, r0, #1 ;increment loop index
CMP r0, r1
BLE Loop
```



Linear Congruential Method

- A pseudo random number generator

$$X_{i+1} = aX_i + c(\text{mod}\, m)$$

Where:

X_i = stream of pseudo random numbers integers from the interval (0, m-1)

a = multiplier constant

c = additive constant

m = modulus or remainder of m

Code for Modulo Operator

```
.global _start
_start:

    MOV r0, #5
    MOV r1, #2

    // perform 5 mod 2
    AND r2, r0, #1

    // perform 5 mod 1
    AND r3, r0, #0
```

Code to Write strings in JTAG UART

```
.data

string1: .asciz "Welcome to the guessing game! Each player has three guesses.\n"// first
message
.align //align memory address
string2: .asciz "Two Players guess numbers. Correct one wins!\n" //second message
```

```
.align //align memory address
.text

.global _start
_start:
            // Start first message
           LDR r0, = 0xff201000
           LDR r1,= string1
           LDR r2,[r1]
           MOV r4, #0 // iterator
            BL message
            //Start second message

            NEXT: LDR r1, = string2
            LDR r2,[r1]
           MOV r4, #0 // iterator
            BL message2

message: //first procedure
PUSH {LR}
loop:
           CMP r2,#0
           BEQ NEXT
           LDRB r2,[r1,r4]
           STRB r2,[r0]
           ADD r4,r4,#1
           B loop
endloop:

BX LR

message2:
PUSH {LR}

loop2:
           CMP r2,#0
           BEQ DONE
           LDRB r2,[r1,r4]
           STRB r2,[r0]
           ADD r4,r4,#1
```

```
        B loop2
endloop2:

BX LR

DONE: B DONE
```

Code for User Input:

```
.data

PoneGuess: .asciz "P1 Guess 1: "

.align

.text


.global _start
_start:
        // Start first message
        LDR r0, = 0xff201000
        LDR r1, = PoneGuess
        LDR r2,[r1]
        MOV r5,#0
        MOV R4, #0
        BL message

        // Start second message
        NEXT: LDR r0, = 0xff201000
        LDR r1,= 0xff200050
        LDR r2,[r1]
        MOV r5,#0
        BL message2

        message: //first procedure
        PUSH {LR}
        loop:
        CMP r2,#0
        BEQ NEXT
        LDRB r2,[r1,r4]
        STRB r2,[r0]
        ADD r4,r4,#1
        B loop
        endloop:

        BX LR

        message2: //first procedure
        PUSH {LR}
         MOV r3, #7
        loop2:
        CMP r2,#0
        BEQ DONE
         LDR r2,[r1,r5]
         CMP r2, #0xA
         BLT skip
        ADD r2,r3,r2
         skip: ADD r2, #0x30
        STR r2,[r0]
        ADD r5,r5,#1
        B loop2
        endloop2:


        BX LR


DONE: B DONE
```

OLD CODE FOR JTAG UART

```
Help_msg: .word 84, 119, 111, 32, 112, 108, 97, 121, 101, 114, 115, 32, 103, 117, 101, 115, 115,
32, 110, 117, 109, 98, 101, 114, 115, 44, 32, 99, 111, 114, 114, 101, 99, 116, 32, 111, 110, 101,
32, 119, 105, 110, 115, 33

.data
Welcome_msg: .word 87, 101, 108, 99, 111, 109, 101, 32,116, 111, 32, 116, 104, 101, 32, 71,
117, 101, 115, 115, 105, 110, 103, 32, 103, 97, 109, 101, 46, 32, 69, 97, 99, 104, 32, 112, 108,
97, 121, 101, 114, 32, 104, 97, 115, 32, 116, 104, 114, 101, 101, 32, 103, 117, 101, 115, 115,
101, 115, 46
Help_msg: .word 84, 119, 111, 32, 112, 108, 97, 121, 101, 114, 115, 32, 103, 117, 101, 115, 115,
32, 110, 117, 109, 98, 101, 114, 115, 44, 32, 99, 111, 114, 114, 101, 99, 116, 32, 111, 110, 101,
32, 119, 105, 110, 115, 33
.text
```

```
.global _start
_start:
LDR r0, = 0xff201000
LDR r1,= Welcome_msg
LDR r2,[r1]
MOV r3, #236//len of arrayx4
MOV r4, #0 // iterator

loop:
CMP r4,r3
BGT NEXT
LDR r2,[r1,r4]
STR r2,[r0]
ADD r4,r4,#4
B loop

NEXT:

LDR r0, = 0xff201000
LDR r1,= Help_msg
LDR r2,[r1]
MOV r3, #172//len of arrayx4
MOV r4, #0 // iterator

loop1:
CMP r4,r3
BGT DONE
LDR r2,[r1,r4]
STR r2,[r0]
ADD r4,r4,#4
B loop1

DONE: B DONE




USER INPUT:

ldr r5, 0xff200040
str [r5], r5
ldr r6, 0xff200050
str [r6], r6
```

```
IF:
cmp r6, #0
BGT USER_INPUT

USER_INPUT:
LDR r0, = 0xff201





NEXT1:
LDR r0, = 0xff201000
LDR r1,= Player_one
MOV r3, #108//len of arrayx4
BL message3

message3:
MOV r2,#0
LDR r2,[r1]
MOV r4, #0 // iterator

loop2:
CMP r4,r3
BGT DONE
LDR r2,[r1,r4]
STR r2,[r0]
ADD r4,r4,#4
B loop2
BX LR

DONE:  //user input

ldr r5, 0xff200040
str [r5], r5 //switch value
ldr r6, 0xff200050
str [r6], r6 //push button

IF:
cmp r6, #0
BGT USER_INPUT

USER_INPUT:
LDR r0, = r5  //r0 is user input’s variable to be printed out on JTAG
```

Str JTAG, r0

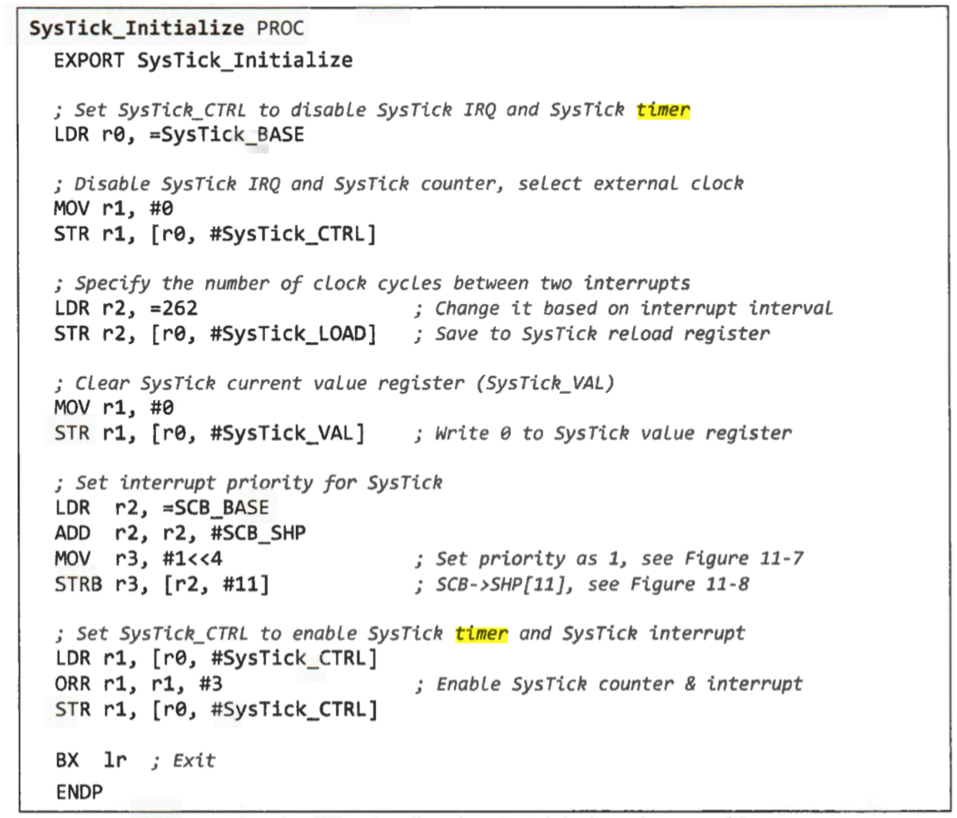
```
HEX_DISPLAYS:
cmp r0, r10 //check if the player guessed the correct number r0: generated number, r10:
guessed number
ldr r1, 0xff200020
ldr r2, 0x7d
str [r2], r1
```

FULL CODE FOR GAME

```
.data
Welcome_msg: .word 87, 101, 108, 99, 111, 109, 101, 32,116, 111, 32, 116, 104, 101, 32, 71,
117, 101, 115, 115, 105, 110, 103, 32, 103, 97, 109, 101, 46, 32, 69, 97, 99, 104, 32, 112, 108,
97, 121, 101, 114, 32, 104, 97, 115, 32, 116, 104, 114, 101, 101, 32, 103, 117, 101, 115, 115,
101, 115, 46
Help_msg: .word 10, 84, 119, 111,32, 32, 112, 108, 97, 121, 101, 114, 115, 32, 103, 117, 101,
115, 115, 32, 110, 117, 109, 98, 101, 114, 115, 44, 32, 99, 111, 114, 114, 101, 99, 116, 32, 111,
110, 101, 32, 119, 105, 110, 115, 33, 46
Player_one: .word 80, 108, 97, 121, 101, 114, 32, 110, 117, 109, 98, 101, 114, 32, 111, 110, 101,
39, 115, 32, 103, 117, 101, 115, 115, 115, 58, 32
.align
.text
.global _start
_start:

// Start first message
LDR r0, = 0xff201000
LDR r1,= Welcome_msg
MOV r3, #236//len of arrayx4
BL message1

// Start second message
NEXT:
LDR r0, = 0xff201000
LDR r1,= Help_msg
MOV r3, #180//len of arrayx4
BL message2
```

```
NEXT1:
LDR r0, = 0xff201000
LDR r1,= Player_one
MOV r3, #108//len of arrayx4
BL message3

message1:
MOV r2,#0
LDR r2,[r1]
MOV r4, #0 // iterator

loop:
CMP r4,r3
BGT NEXT
LDR r2,[r1,r4]
STR r2,[r0]
ADD r4,r4,#4
B loop
BX LR

message2:
MOV r2,#0
LDR r2,[r1]
MOV r4, #0 // iterator

loop1:
CMP r4,r3
BGT NEXT1
LDR r2,[r1,r4]
STR r2,[r0]
ADD r4,r4,#4
B loop1
BX LR

message3:
MOV r2,#0
LDR r2,[r1]
MOV r4, #0 // iterator

loop2:
CMP r4,r3
BGT DONE
LDR r2,[r1,r4]
```

```
STR r2,[r0]
ADD r4,r4,#4
B loop2
BX LR

DONE:

ldr r5, 0xff200040
str [r5], r5
ldr r6, 0xff200050
str [r6], r6

IF:
cmp r6, #0
BGT USER_INPUT

USER_INPUT:
LDR r0, = 0xff201

HEX_DISPLAYS:
cmp r0, r10 //check if the player guessed the correct number r0: generated number, r10:
guessed number
ldr r1, 0xff200020
ldr r2, 0x7d
str [r2], r1
```



```
SysTick_Initialize PROC
  EXPORT SysTick_Initialize

  ; Set SysTick_CTRL to disable SysTick IRQ and SysTick timer
  LDR r0, =SysTick_BASE

  ; Disable SysTick IRQ and SysTick counter, select external clock
  MOV r1, #0
  STR r1, [r0, #SysTick_CTRL]

  ; Specify the number of clock cycles between two interrupts
  LDR r2, =262                  ; Change it based on interrupt interval
  STR r2, [r0, #SysTick_LOAD]   ; Save to SysTick reload register

  ; Clear SysTick current value register (SysTick_VAL)
  MOV r1, #0
  STR r1, [r0, #SysTick_VAL]    ; Write 0 to SysTick value register

  ; Set interrupt priority for SysTick
  LDR  r2, =SCB_BASE
  ADD  r2, r2, #SCB_SHP
  MOV  r3, #1<<4                ; Set priority as 1, see Figure 11-7
  STRB r3, [r2, #11]            ; SCB->SHP[11], see Figure 11-8

  ; Set SysTick_CTRL to enable SysTick timer and SysTick interrupt
  LDR r1, [r0, #SysTick_CTRL]
  ORR r1, r1, #3                ; Enable SysTick counter & interrupt
  STR r1, [r0, #SysTick_CTRL]

  BX  lr  ; Exit
  ENDP
```

Example 11-10. Configuring SysTick timer in assembly

```
LDR R0, =0xFF200000
LDR R1, =0x0
LDR R2, =0xFFFEC600//PRIVATE TIMER BASE ADDRESS
LDR R3, = 0x5FFFFF0
STR R3, [R2]
LDR R3, =0x0103 //ENABLE AND SET INTERUPT & AUTO ON
STR R3, [R2, #8] //AND PRESCALAR SET TO 0
LOOP:
STR R1, [R0]
LDR R4, [R2, #12]
CMP R4, #1
BNE LOOP
ADD R1, R1, #1
CMP R1, #16
BNE SKIP
LDR R1, =0x0
SKIP:
LDR R3, =0x1
STR R3, [R2, #12]
B LOOP
```

```
P1GG:

ldr r1, =0xff200020
ldr r2, =0x73067d7d
str r2, [r1]

ldr r3, =0xFFFFF
Delay:
cmp r3, #0
beq out
sub r3, #1
b Delay

out:
ldr r1, =0xff200020
ldr r2, =0x00000000
str r2, [r1]

ldr r3, =0x5FFFF
Delay1:
cmp r3, #0
beq out1
sub r3, #1
b Delay1
out1:
b P1GG
```

```
P2GG:
ldr r1, =0xff200020
ldr r2, =0x735b7d7d
str r2, [r1]


ldr r3, =0xFFFFF
Delay:
cmp r3, #0
beq out
sub r3, #1
b Delay

out:
ldr r1, =0xff200020
ldr r2, =0x00000000
str r2, [r1]

ldr r3, =0x5FFFF
Delay1:
cmp r3, #0
beq out1
sub r3, #1
b Delay1
out1:
b P2GG
```

```
LOSE:

ldr r1, =0xff200020
ldr r2, =0x383f6d79
str r2, [r1]


ldr r3, =0xFFFFF
Delay:
cmp r3, #0
beq out
sub r3, #1
b Delay

out:
ldr r1, =0xff200020
ldr r2, =0x00000000
str r2, [r1]

ldr r3, =0x5FFFF
Delay1:
cmp r3, #0
beq out1
sub r3, #1
b Delay1
out1:
b LOSE
```

```
NEXT:

ldr r1, =0xff200020
ldr r2, =0x54797678
str r2, [r1]


ldr r3, =0xFFFFF
Delay:
cmp r3, #0
beq out
sub r3, #1
b Delay

out:
ldr r1, =0xff200020
ldr r2, =0x00000000
str r2, [r1]

ldr r3, =0x5FFFF
Delay1:
cmp r3, #0
beq out1
sub r3, #1
b Delay1
out1:
b NEXT
```

Guess inside JTAG UART

```
.data

string1: .asciz "Welcome to the guessing game! Each player has three guesses.\n"// first
message
.align //align memory address
.text

.global _start
_start:
        // Start first message
        LDR r0, = 0xff201000
        LDR r1,= string1
        LDR r2,[r1]
        MOV r4, #0 // iterator
        BL message
        //Start second message

message: //first procedure
PUSH {LR}
loop:
        CMP r2,#0
        BEQ pushbutton
        LDRB r2,[r1,r4]
        STRB r2,[r0]
        ADD r4,r4,#1
        B loop
endloop:

BX LR

pushbutton:

LDR r9,  = 0xff201000
LDR r6, =  0xff200050
LDR r8, [r6]

loop3:
   LDR r5, = 0xff200040
   LDR r7,
   LDR r6, =  0xff200050
   LDR r8, [r6]
   CMP r9, #8
   BGE display
   B loop3
endloop3:

display: ADD r8, #0x30
          STR r8,[r9]
            B DONE


DONE: B DONE




.data

string1: .asciz "Welcome to the guessing game! Each player has three guesses.\n"// first
message
.align //align memory address
.text

.global _start
_start:

          // Start first message
          LDR r0, = 0xff201000
          LDR r1,= string1
          LDR r2,[r1]
          MOV r4, #0 // iterator
          BL message
          //Start second message

message: //first procedure
PUSH {LR}
loop:
          CMP r2,#0
          BEQ pushbutton
          LDRB r2,[r1,r4]
          STRB r2,[r0]
          ADD r4,r4,#1
          B loop
endloop:
```

```
BX LR

pushbutton:
   LDR r9, = 0x0
loop3:
         LDR r5, = 0xff200040
         LDR r6, = 0xff200050
         LDR r8, [r5]
         CMP r8, #512
         BGE display
   B loop3
endloop3:

   loop4:
   display:
   CMP r7,#0
   BEQ DONE
   LDR r7,[r6,r9]
   ADD r7, #0x30
         STR r7,[r9]
   ADD r9, #1
   B loop4
endloop4:

DONE: B DONE
```

GAME SO FAR without r generator:

```
.data

string1: .asciz "Welcome to the guessing game! Each player has two guesses.\n"// first message
.align //align memory address
PoneGuess: .asciz "P1 Guess 1: "
.align
DataTable: .Byte 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x77, 0x7c, 0x39,
0x5e, 0x79, 0x71
Random: .Byte 0x77
.text
.global _start
_start:
//Load the Random into r11
ldr r12, =Random
ldr r12, [r12]
```

```
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
//Just a preview PLAY display untill user turns the far left switch on.
Play:
        ldr r6, =0xff200040
        ldr r7, [r6]
        cmp r7, #512
beq Game
        ldr r4, =0xff200020
        ldr r5, =0x7338776e
        str r5, [r4]
bl Delaylong

out:

        ldr r4, =0xff200020
        ldr r5, =0x00000000
        str r5, [r4]
bl Delayshort


b Play

Game: // it will display strings on Jtag at this point.

ldr r4, =0xff200020
ldr r5, =0x00000000
str r5, [r4]
bl Delaylong

         // Start first message
        LDR r0, = 0xff201000
        LDR r1,= string1
        LDR r2,[r1]
        MOV r4, #0 // iterator
         BL message
         //Start second message


message: //first procedure
```

```
PUSH {LR}
loop:
        CMP r2,#0
        BEQ DONE
        LDRB r2,[r1,r4]
        STRB r2,[r0]
        ADD r4,r4,#1
        B loop
endloop:

BX LR

DONE: // PLAYERS STARTS GUESSING

enter:
ldr r1, =0xff200030
ldr r4, =0xff200020
ldr r5, =0x0079
ldr r2, =0x54787950
str r5, [r1]
str r2,[r4]
bl Delaylong
ldr r8, =0xff200050
ldr r9,[r8]
cmp r9, #0
bge input
b enter

pushbuttons:
ldr r8, =0xff200050
ldr r9, [r8]
cmp r9, #1
bge input
b enter

input:
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]

//indicating p1 is entering
bl Delayshort
ldr r5, =0x7306
```

```
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP: //initial guess for p1
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #1
beq p1
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP

p1: //player 1 first guess display
ldr r1, =0xff200030
ldr r5, =0x7306
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP1


//indicating p2 is entering
bl Delayshort
ldr r5, =0x735b
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP1:
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #3
```

```
beq p2
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP1


p2: //player 2 first guess
ldr r1, =0xff200030
ldr r5, =0x735b
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP2

//indicating p1 is entering
bl Delayshort
ldr r5, =0x7306
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP2:
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #7
beq p1_2
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP2

p1_2: //player 1 second guess display
ldr r1, =0xff200030
ldr r5, =0x0000
str r5, [r1]
ldr r5, =0x7306
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP1


//indicating p2 is entering
bl Delayshort
ldr r5, =0x735b
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP3:
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #15
beq p2_2
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP3

p2_2: //player 2 second guess display
ldr r1, =0xff200030
ldr r5, =0x0000
str r5, [r1]
ldr r5, =0x735b
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
```

```
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP2


bl Delaylong

LOSE:
ldr r6, =0xff200040 // restarts the program
ldr r7, [r6]
cmp r7, #256
beq  _start

ldr r5, =0x383f6d79
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delaylong

ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delayshort
b LOSE

WIN_forP1:
ldr r6, =0xff200040 // restarts the program
ldr r7, [r6]
cmp r7, #256
beq _start

ldr r5, =0x73067d7d
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delaylong

ldr r5, =0x00000000
ldr r2, =0x0000
```

```
str r2, [r1]
str r5, [r4]
bl Delayshort

b WIN_forP1

WIN_forP2:
ldr r6, =0xff200040 // restarts the program
ldr r7, [r6]
cmp r7, #256
beq _start

ldr r5, =0x735b7d7d
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delaylong

ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delayshort

b WIN_forP2


// Long and short delays. Could be called with bl function.
Delayshort:
ldr r3, =0x7FFFF
D1:
cmp r3, #0
bxeq lr
sub r3, #1
b D1

Delaylong:
ldr r3, =0x1AFFFF
D:
cmp r3, #0
bxeq lr
sub r3, #1
b D
```

Random # Gen

```
.data

.align
DataTable: .Byte 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x77, 0x7c, 0x39,
0x5e, 0x79, 0x71
.align
randn: .word 0

.align

Seed: .word 0x0//x
Mul_Constant: .word 0x03//a
Mod: .word 0xE//m
Add_Constant: .word 0x01//c

.align
.text
.global _start
_start:
//Load the Random into r12

LDR r1, = Seed
LDR r1,[r1]
LDR r2, = Mul_Constant
LDR r2,[r2]
LDR r3, = Mod
LDR r3,[r3]
LDR r4, = Add_Constant
LDR r4,[r4]
MOV r0, #0 //iterator
LDR r5, = randn // load in address of the array
SUB r3, #1 // m-1
BL rand_num
rand_num:
        loopp:
        CMP r0, #4
        BEQ check
        MUL r1, r2 //aXi
        ADD r6, r1,r4 // (aXi + c)
        AND r12, r6, r3 // (aXi + c) %m
        STR r12,[r5,r0]
        ADD r0, #4
        B loopp
        endloopp:

check:
LDRB r7, = DataTable
MOV r9, r12
LDRB r12,[r7,r9]
B DONE


DONE: B DONE
```

Game with Random # Generator

```
.data
string1: .asciz "Welcome to the guessing game! Each player has two guesses.\n" //firstmessage
.balign 8 // divisible by 8
DataTable: .Byte 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x77, 0x7c, 0x39,
0x5e, 0x79, 0x71
.balign 8 // divisible by 8
randn: .word 0,0,0,0,0
.align 4 // divisible by 4

.text
.global _start
_start:
//Load the Random into r12

LDR r1, = 0x0 // load in seed
LDR r2, = 0x02 // load in a
LDR r3, = 0x10 // load in modulus
LDR r4, = 0x02 // load in constant
LDR r0, = 0x0 //iterator
LDR r5, = randn // load in address of the array
SUB r3, #1 // m-1(modulus -1)
BL rand_num // branch to procedure
rand_num:
        PUSH {LR}
        loopp:
        CMP r0, #20 // compare with the total # of bits in array
        BEQ check
        MUL r1, r2 //aXi
        ADD r6, r1,r4 // (aXi + c)
```

```
          AND r12, r6, r3 // (aXi + c) %m
          STR r12,[r5,r0] // stores the value of rand # in rand with preindex
          MOV r1, r12 // recusri, Xi+1 = Xi
          ADD r0, #4 // address is separated by 4 bits
          B loopp
          endloopp:
          BX LR

check:
LDR r12,[r5,#4] // loads in the 2nd number from rand # array
LDR r0, = DataTable // address of data table stored in r0
MOV r9, r12 // saves original value of rand #
LDRB r12,[r0,r9] // converts rand # to hex for 7-digit display
B Begin


Begin:

//clear registers
LDR r0, = 0x00000000
LDR r1, = 0x00000000
LDR r2, = 0x00000000
LDR r3, = 0x00000000
LDR r4, = 0x00000000
LDR r5, = 0x00000000
LDR r6, = 0x00000000
LDR r9, = 0x00000000
LDR r10,= 0x00000000

//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
//Just a preview PLAY display untill user turns the far left switch on.
Play:
          ldr r6, =0xff200040
          ldr r7, [r6]
          cmp r7, #512
beq Game
          ldr r4, =0xff200020
          ldr r5, =0x7338776e
          str r5, [r4]
bl Delaylong
```

```
out:

        ldr r4, =0xff200020
        ldr r5, =0x00000000
        str r5, [r4]
bl Delayshort


b Play

Game: // it will display strings on Jtag at this point.

ldr r4, =0xff200020
ldr r5, =0x00000000
str r5, [r4]
bl Delaylong

        // Start first message
        LDR r0, = 0xff201000 // loads in address of JTAG into r0
        LDR r1,= string1 // address of first string loaded into r1
        LDR r2,[r1] // value of r1 loaded into r2
        MOV r4, #0 // iterator
        BL message
        //Start second message


message: //first procedure
PUSH {LR}
loop:
        CMP r2,#0 // compares r2 to 0 to see if message is empty
        BEQ DONE
        LDRB r2,[r1,r4] // loads in one byte at a time from string1
        STRB r2,[r0] // stores byte into JTAG to display in terminal
        ADD r4,r4,#1 // iterates until end of string
        B loop
endloop:

BX LR

DONE: // PLAYERS STARTS GUESSING

enter:
ldr r1, =0xff200030
ldr r4, =0xff200020
ldr r5, =0x0079
ldr r2, =0x54787950
str r5, [r1]
str r2,[r4]
bl Delaylong
ldr r8, =0xff200050
ldr r9,[r8]
cmp r9, #0
bge input
b enter

pushbuttons:
ldr r8, =0xff200050
ldr r9, [r8]
cmp r9, #1
bge input
b enter

input:
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]


//indicating p1 is entering
bl Delayshort
ldr r5, =0x7306
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP: //initial guess for p1
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #1
beq p1
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP
```

```
p1: //player 1 first guess display
ldr r1, =0xff200030
ldr r5, =0x7306
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP1


//indicating p2 is entering
bl Delayshort
ldr r5, =0x735b
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP1:
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #3
beq p2
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP1


p2: //player 2 first guess
ldr r1, =0xff200030
ldr r5, =0x735b
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
```

```
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP2

//indicating p1 is entering
bl Delayshort
ldr r5, =0x7306
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP2:
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #7
beq p1_2
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP2

p1_2: //player 1 second guess display
ldr r1, =0xff200030
ldr r5, =0x0000
str r5, [r1]
ldr r5, =0x7306
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP1
```

```
//indicating p2 is entering
bl Delayshort
ldr r5, =0x735b
str r5, [r1]
bl Delayshort
ldr r2, =0x0000
str r2, [r1]

LOOP3:
ldr r6, =0xff200040
ldr r7, [r6]
cmp r7, #15
beq p2_2
LDR r9, [r8]
LDR r10, =DataTable
LDRB r2, [r10, r9]
STR r2, [r1]
B LOOP3

p2_2: //player 2 second guess display
ldr r1, =0xff200030
ldr r5, =0x0000
str r5, [r1]
ldr r5, =0x735b
mov r11, r2
str r5, [r1]
str r11,[r4]
bl Delaylong
//clear Hex
ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
cmp r11, r12
beq WIN_forP2


bl Delaylong

LOSE:
ldr r6, =0xff200040 // restarts the program
ldr r7, [r6]
cmp r7, #256
```

```
beq  _start

ldr r5, =0x383f6d79
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delaylong

ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delayshort
b LOSE

WIN_forP1:
ldr r6, =0xff200040 // restarts the program
ldr r7, [r6]
cmp r7, #256
beq _start

ldr r5, =0x73067d7d
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delaylong

ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delayshort

b WIN_forP1

WIN_forP2:
ldr r6, =0xff200040 // restarts the program
ldr r7, [r6]
cmp r7, #256
beq _start

ldr r5, =0x735b7d7d
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delaylong

ldr r5, =0x00000000
ldr r2, =0x0000
str r2, [r1]
str r5, [r4]
bl Delayshort

b WIN_forP2


// Long and short delays. Could be called with bl function.
Delayshort:
ldr r3, =0x7FFFF
D1:
cmp r3, #0
bxeq lr
sub r3, #1
b D1

Delaylong:
ldr r3, =0x1AFFFF
D:
cmp r3, #0
bxeq lr
sub r3, #1
b D
```