Lab 4: How colorful is Your Name

Victor Yuan

Microcontrollers and Embedded Systems

March 1, 2017

Introduction:

The Goal of this lab was to write a program in assembly read in a line of ASCII characters add each asciz character into a array, then convert every 3 asciz characters into the grey scale

Procedure:

I: create rgb array

First create an array with the length of however long your name is rounded up to the nearest multiple of three, with the width of ten rows. Multiple length and width and store this value in r6 then left shifting by 1 and adding the value back into itself, this will have the same result as multiplying r6 by three. Branch Link to malloc to allocate the array in memory. At this point the array pointer is stored in r0 mov this pointer to register r8.

II: initialize array

Right now r8 contains an array of zeros. Initialize the array with the asciz numbers of your name. Load your name to r7. Use ldrb to load a single byte to a temporary register r0 for example, load from r7, the register holding the pointer to your name variable. Offset r7 so that the next character can be accessed later. Use strb to store the single asciz number stored in r0 into the first value of [r8] offset this by 1 so that the next empty slot can be accessed later. Using r2 as a counter for how many characters of your name you have entered add 1 onto r2 every time you have initialized a portion of the array. Depending on how long your name is u will need to calculate how many times you will need your loop to loop for. For my name there were 11 characters so I compared r2 to #11 if the condition were set to equal then the program would load my name back into r7 resetting the pointer to the first position it would add a extra offset to the pointer in my array so that it would skip a 0 therefore effectively making

my name in the array 12 characters, 11 with a value and the last one 0. And lastly reset the counter back to 0.

I recorded every time the first mini-loop which was used to load a single character in r1 when r1 reached 110, it signified that I had performed the loop 10 times and that 10 rows of my array had been initialized. Move the program counter back to the link register to finish part1 of the lab.

III: Grey scale

Part2 of the lab was converting to grey scale. For this section set r1, 2, and 3 to the values 54, 184, 18. Move the pointer in r8 back 120 spaces so that it begins reading at the start of the array. Use r11 as a tracker. Call your subroutine that converts your array to grey scale. Load 1 byte into r4 the next into r5 and the next in r6 offset for the first 2 and leave the 3rd where it is for now. Multiply r4 with r1(54), r5 with r2(184), r6 with r3(18). Store these values in r7, 8, and 9. Add r7 and r8 and store in r7. Add r7 and r9 and store in r7. Right shift this number by 8 which is the same as diving by 2^8 , 256, this will result in r7 to be the grey scale value. Store this value back into r8 and shift it by 1. Repeat this 40 times for my case. It will be the size of your name rounded up to the nearest multiple of 3 times 10 divided by 3. Spot check this in the memory address of r8.

Code:

```
1         .data
2         .equ    i_red,   0
3         .equ    i_green, 1
4         .equ    i_blue,  2
5         .equ    i_size,  4
6         .equ    NULL,    0
7     name:   .asciz  "victor yuan"
8     width:  .word   12
9     height:  .word   10
10
11        .text
12
13    .global main
14
15    main:
16
17        ldr r4, =width
18        ldr r4, [r4]
19        ldr r5, =height
20        ldr r5, [r5]
21
22        mul r6, r4,r5
23        add r0,r6,r6, lsl #1
24        bl malloc
25        cmp r0, #NULL
26        beq else
27        mov r8, r0          @r8 now holds my array
28        mov r1, #0         @height
29        mov r2, #0        @width
30        mov r0, #0
31
32        ldr r7, =name
33        bl setrow
34
35
```

Setting up the array putting name into r7 and calling subroutine setrow, part1 of the lab

```
setrow:
        ldrb r0, [r7], #1       @loads 1 byte-character of my name into array and increases pointer count
        strb r0, [r8], #1       @r0 is the temp place holder
        add r2, r2, #1          @width counter increased by 1 for every letter stored

        cmp r2, #11             @compare r2 counter with total width,12
        ldreq   r7, =name       @if r2 = 12 then these 2 instructions exacute
        addeq   r8,r8,#1        @extra filler to make name 12 increments atm, its 11
        moveq r2, #0          @resets name setter portion

        add r1, r1, #1          @needs to add up to 110 to end
        cmp r1, #110
        blt setrow
        movge pc, lr
```

Set row subroutine

```
@now part2 of the lab converting to grey scale
@using r1,2,3 for values for scalar multiple of rgb
mov r1, #54
mov r2, #184
mov r3, #18
sub r8, r8, #120
mov r11, #0

bl rgb2grey


mov r0,#0
mov pc,lr
```

Sets up r1, 2, and 3 as constant values, calls grey scale conversion function

```
rgb2grey:
        ldrb r4, [r8], #1
        ldrb r5, [r8], #1
        ldrb r6, [r8]   @moved the shift to strb

        mul r7,r1,r4
        mul r9,r2,r5
        mul r10, r3,r6

        add r7, r7,r9       @r7 = r7 + r9
        add r7, r7,r10      @r7= r7+ r9+r10

        lsr r7, #8          @divide by 256
        strb r7,[r8], #1    @store back into array at the last spot
        add r11,r11, #1
        cmp r11, #40
        blt rgb2grey
        movge   pc,lr
```

Rgb2grey subroutine

Results:

Regis	Hex	Integer
r0	0x0	0
r1	0x6e	110
r2	0x0	0
r3	0x76e1f4d4	1994519764
r4	0xc	12
r5	0xa	10
r6	0x78	120
r7	0x206f4	132852
r8	0x21080	135296
r9	0x0	0
r10	0x76fff000	1996484608
r11	0x0	0
r12	0x21008	135176
sp	0x7efff550	2130703696
lr	0x10500	66816
pc	0x10504	66820
cpsr	0x60000010	1610612752

counter r1 is full r2 just finished its cycle

Current state of the array

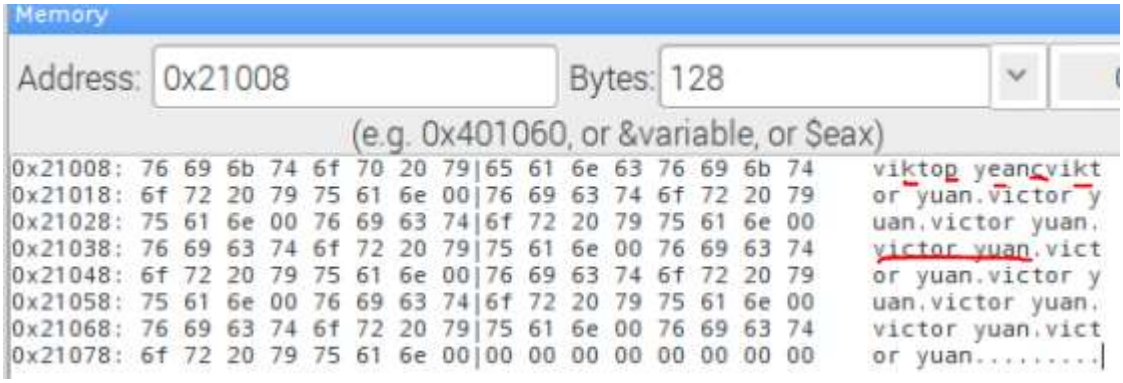
```
0x21008: 76 69 63 74 6f 72 20 79|75 61 6e 00 76 69 63 74    victor yuan.vict
0x21018: 6f 72 20 79 75 61 6e 00|76 69 63 74 6f 72 20 79    or yuan.victor y
0x21028: 75 61 6e 00 76 69 63 74|6f 72 20 79 75 61 6e 00    uan.victor yuan.
0x21038: 76 69 63 74 6f 72 20 79|75 61 6e 00 76 69 63 74    victor yuan.vict
0x21048: 6f 72 20 79 75 61 6e 00|76 69 63 74 6f 72 20 79    or yuan.victor y
0x21058: 75 61 6e 00 76 69 63 74|6f 72 20 79 75 61 6e 00    uan.victor yuan.
0x21068: 76 69 63 74 6f 72 20 79|75 61 6e 00 76 69 63 74    victor yuan.vict
0x21078: 6f 72 20 79 75 61 6e 00|00 00 00 00 00 00 00 00    or yuan.........
```

“victor_yuan “ was repeated 10 times.

Part 1 was successful

Regis	Hex	Integer
r0	0x6e	110
r1	0x36	54
r2	0xb8	184
r3	0x12	18
r4	0x74	116
r5	0x69	105
r6	0x63	99
r7	0x6b	107
r8	0x21018	135192
r9	0x4b78	19320
r10	0x6f6	1782
r11	0x5	5
r12	0x21008	135176
sp	0x7efff550	2130703696
lr	0x10518	66840
pc	0x10550	66896
cpsr	0x80000010	2147483664

The counter r11 shows that it's in this 5th loop therefore there should be 5 changes



5 letters where changed c-> k , r->p, u-> e, " " -> c, and repeats at c->k again.

Part 2 was successful.

Conclusion:

To make a complete rbg2grey conversion I only have to replace the values two spaces before the every change to the same number in every 3rd spot. One thing I learned in this lab is conditional operators can help manipulate a loop very easily