Lab 1: Intro to ARM Assembler on R Pi 3
Report

Tyler Nardecchia
Yerania Hernandez
CSCE 462 - 502
September 11, 2017

Thinking and Exploring

1) *Do you have an alternative way to blink this external LED? If so, write down your answer.*
   Besides using assembly to blink the LED, we could have also used another programming language, such as Python in order to accomplish the same result. When using Python, the functions we would have used instead of pinMode, digtialWrite, and delay would have been GPIO.setup(), GPIO.output(), and time.sleep(). This would have allowed us to blink the LED as well by assigning the pin, and setting the GPIO high to turn on, low to turn off, and sleep for a number of seconds similar to the delay function.

2) *What can you add/change to the code in section 3.2-3.3 if you want to control the LED on/off time to different numbers? (For example LED on for 3 seconds and off for 4 seconds, repeat.) Can we replace function Delay in our code?*
   The function pinMode sets the mode of a pin to output in this case considering we are trying to control the LED. The function digitalWrite allows us to turn on or off the specific pin number we have assigned. As a result, we could change the delay function in the for loop to last three seconds on (3000 ms), then use digitalWrite to turn off the LED and use the delay function to make it last four seconds off (4000 ms), and repeat within the loop. It is also possible to replace the Delay function with a function such as clock_gettime (in the time.h library). This would involve storing the current time in a variable for each blink, and then doing nothing in a loop until a certain amount of time has passed. This is less efficient than using the Delay function, but it would still work.

3) *Can we still produce music by this buzzer using the same way we do to the LED? If yes, provide your solution. If no, what needs to be changed in order to produce music?*
   There are two type of buzzers that can be used: active and passive. Depending on which type is implemented, the code will be either similar to the LED (active) or much different due to having to set the frequency and time for beat in a song (passive). If we implemented an active buzzer, than the code and setup are similar to the blinking LED we implemented. Unlike the LED, the buzzer will not need a resistor, simply connecting it to the ground and a GPIO pin will work. As for the code in assembly, nothing will change considering we need pinMode to assign the pin and digitalWrite to turn the pin on and off. This will allow us to hear the buzzer beeping. If we wanted to hear music instead, the same setup would apply. However, the code would be different where we would be using PWM (pulse width modulation) functions. We would use pinMode to assign the pin and set it to PWM_OUTPUT, pwmWrite in order to assign the initial frequency, a for loop that would change the frequency of a song using pwmWrite, and the delay function that would provide the beat based on the time delayed. As a result, the buzzer will be able to produce music.

```
            .data
            .balign 4
pin:        .int 7
delay_time: .int 200
i:          .int 0
error:      .asciz "Error in intitializing\n"
OUTPUT      = 1

            .text
            .global main
            .extern printf
            .extern wiringPiSetup
            .extern pinMode
            .extern digitalWrite
            .extern delay

main:
    push    {r12, lr}
    bl  wiringPiSetup
    mov r1, #-1
    cmp r0, r1
    bne init
    ldr r0, =error
    bl  printf
    b   exit

@PinMode Setup
init:
    ldr r0, =pin
    ldr r0, [r0]
    mov r1, #OUTPUT
    bl  pinMode

@ForLoop Setup
    ldr r4, =i
    ldr r4, [r4]
    mov r5, #10
    mov r6, #1
loop:
    cmp r4, r5
    beq exit

@digitalWrite Setup for (pin,1)
    ldr r0, =pin
    ldr r0, [r0]
    mov r1, #1
    bl  digitalWrite
@delay(200)
    ldr r0, =delay_time
    ldr r0, [r0]
    bl  delay
@digitalWrite Setup for (pin,0)
    ldr r0, =pin
    ldr r0, [r0]
    mov r1, #0
    bl  digitalWrite
@delay(200)
    ldr r0, =delay_time
    ldr r0, [r0]
    bl  delay
@Adding 1 to i for for loop count
    add r4, r4, r6
    b   loop

exit:
    pop {r12,pc}
```