COS 10004 Computer Systems

Assignment 3 – Raspberry PI Led display

Student ID – 103636725

Name – John Hewitt

September 27 2021

Contents

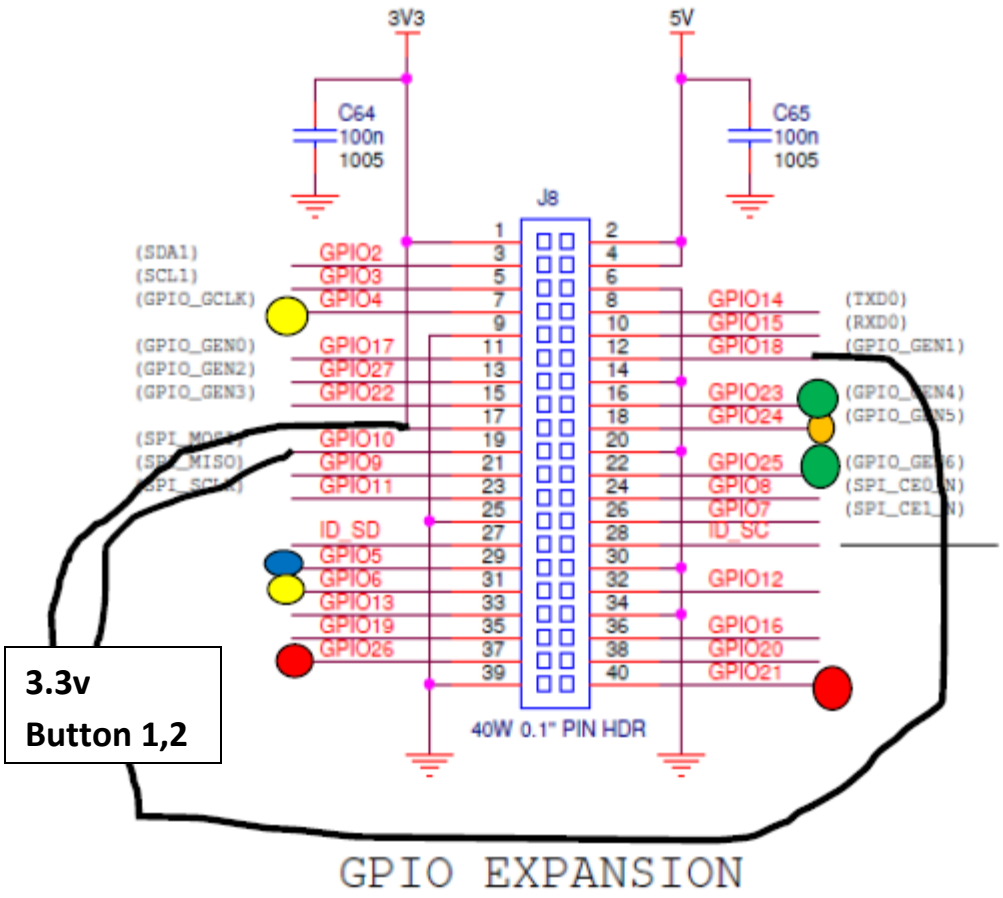
1.Description of Project.

For this project I have produced a multi LED display using a Raspberry PI4, 8 LED lights, breadboard, and 2 buttons to trigger a 4 mode LED flashing pattern. The project also connects to a HDMI display which has a screen to show the name of the project. Screen prints of some of the asm assembly code is also included with the project.

2. Block Diagram

The diagrams below show a block diagram and GPIO details of what LED is connected to which GPIO pin. The circuit also includes 2 buttons which is connected to GPIO pin 10 and GPIO18 and provide a switch to connect to 3.3volt pin.

GPIO	GPIO function enable bit (lsl)	GPIO function enable offset (add to 0x20200000)	R/W GPIO bit (lsl)	R/W GPIO offset (add to register offset)	Led No.	button no.
4	12	0	4	0	3	
5	15	0	5	0	1	
6	18	0	6	0	6	
7	21	0	7	0		
10	0	4	10	0		1
18	24	4	18	0		2
21	3	8	21	0	4	
23	9	8	23	0	2	
24	12	8	24	0	8	
25	15	8	25	0	7	
26	18	8	26	0	5	
3.3v						1,2
gnd						



The following ASM files are used for this project

Kernel81.asm	**Drawpixel.asm**
Led_disp.asm	**Drawchar.asm**
Timer2.asm	**Fbinit8.asm**

Originally my project was only going to include led_disp.asm plus the timer but just to add a screen with the programs name and a box around the name I added led_disp as a function to the kernel81.asm file we had from the lab which was used to draw characters and mixed it with the one to draw lines. I altered slightly to print name of program to screen plus I added some code to draw 3 lines to make a box around the title. That is the only purpose of including the drawpixel, drawchar and fbinit8.asm files the main part of my project is included in the led_disp.asm file.

The main parts of Led_disp operation is detailed below(not all code is shown)

Setup of GPIO 10,18 for buttons

```
orr r0,GPIO_OFFSET ;Base address of GPIO

ldr r1,[r0,#4] ;read function register for GPIO 10 - 19
bic r1,r1, #7 ;     ; #bit clear bits 0,1,2 FOR gpio 10    = read access
bic r1,r1,$7000000 ; #bit clear bits 18,19,20 FOR gpio 18
str r1,[r0,#4];10 input
```

Setup GPIO pins for 8 LEDS

```
mov r10,BASE
mov r2,BASE
orr r2,GPIO_OFFSET
orr r10,GPIO_OFFSET

ldr r2,[r0,#0]
orr r2,r2, $40000 ; make R1 =    00100100000000000000000  GPIO6
orr r2,r2, $8000 ; make R1 =     00100100100000000000000 add GPIO5
orr r2,r2, $1000 ; make R1 =     00100100100100000000000 add GPIO4
str r2,[r0,#0];set GPIO4,6 to output

ldr r10,[r0,#8]
orr r10,r10, $40000 ;make R1 =  01000000000000000000  GPIO26
orr r10,r10, $8000 ; make R1 = 01001000000000000000  add GPIO25
orr r10,r10, $1000 ; make R1 = 01001000000000000000  add GPIO24
orr r10,r10, $200 ; make R1 =  01001000001000000000  add GPIO23
orr r10,r10, $08 ; make R1 =  01001000001000000000  add GPIO21
str r10,[r0,#8];set GPIO23,25&26 to output  ;store r1 to r0
```

Test value of Button 1 GPIO 10 and branch to led2 or else mode 1 display

```
;poll GPIO10 and swap LEDS if high
loopx$:
;read first block of GPIOs
ldr r9,[r0,#52] ;read gpios 0-31
tst r9,#1024  ; use tst to check bit 10
bne led2 ;if ==0 branch to led2 when gpio 10 connects with 3.3v
;else LED 1
```

Turning Leds on or off for mode 1 display

```
;else LED mode_1
 mov r2,#1
 lsl r2,#5   write  1 into r2, lsl 5 times to move the 1 to bit 5
 str r2,[r0,#28] ;turn LED on
  mov r10,#1
  lsl r10,#23;write  1 into r10, lsl 23 times to move the 1 to bit 23
  str r10,[r0,#28]  ;turn LED on
  mov r2,#1
  lsl r2,#4;write  1 into r2, lsl 4 times to move the 1 to bit 4
  str r2,[r0,#40]  ; turn led off
   mov r10,#1
  lsl r10,#21;write  1 into r10, lsl 21 times to move the 1 to bit 21
  str r10,[r0,#40]    ; turn led off
  mov r10,#1
 lsl r10,#26;write  1 into r10, lsl 26 times to move the 1 to bit 26
 str r10,[r0,#40]    ; turn led off
  mov r2,#1
 lsl r2,#6;write  1 into r2, lsl 6 times to move the 1 to bit 6
 str r2,[r0,#40]      ; turn led off
  mov r10,#1
 lsl r10,#25;write  1 into r10, lsl 25 times to move the 1 to bit 25
 str r10,[r0,#40]    ; turn led off
  mov r10,#1
 lsl r10,#24;write  1 into r2, lsl 24 times to move the 1 to bit 24
 str r10,[r0,#40]     ; turn led off
 push {r0,r1,r2,r7,r10,r11,lr} ;r0,r1,r2,r7 in use push and then set parameters
 mov r0,BASE

mov r1,$040000 ; value passed to delay about .25 second
 bl Delay
 pop {r0,r1,r2,r7,r10,r11,lr}
 mov r2,#1
 lsl r2,#5   write  1 into r2, lsl 5 times to move the 1 to bit 5
 str r2,[r0,#40] ;turn LED off
  mov r10,#1
```

At led2 test for Button 2 GPIO18 and branch to led3 or else mode 2 display

```
b cont1
led2:   ; comes here when GPIO10 connect 3.3v
ldr r9,[r0,#52] ;read gpios 0-31
tst r9,#262144  ; use tst to check bit 18--disabled
bne led3 ;if ==0 branch to Led 3 if gpio 10 &18 cnnected to 3.3v
;else if only GPIO pin 10 connected to 3.3v
```

Turning Leds on or off for mode 2 display

```
; else if both are connected
mov r10,#1
lsl r10,#21   ;bit 21 to write to GPIO21
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#23   ;bit 23 to write to GPIO23
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#25   ;bit 23 to write to GPIO25
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#26   ;bit 23 to write to GPIO26
str r10,[r0,#28] ;Turn on LED 1
mov r2,#1
 lsl r2,#5
 str r2,[r0,#28] ;turn LED on
mov r2,#1
lsl r2,#4;write  1 into r2, lsl 4_times to move the 1 to bit 4
```

At led3 test for Button 1 and 2 and branch to led4 or else mode 3 display

```
cont1:

b cont2
led3:
ldr r9,[r0,#52] ;read gpios 0-31
teq r9,#262144   ; use tst to check only  bit 18 enabled
beq led4 ; branch if equal ie gpio10 not connected to 3.3v but gpio 18 is
; else if both are connected
```

Turning Leds on or off for mode 3 display

```
; else if both are connected
mov r10,#1
lsl r10,#21   ;bit 21 to write to GPIO21
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#23   ;bit 23 to write to GPIO23
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#25   ;bit 23 to write to GPIO25
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#26   ;bit 23 to write to GPIO26
str r10,[r0,#28] ;Turn on LED 1
mov r2,#1
 lsl r2,#5
 str r2,[r0,#28] ;turn LED on
mov r2,#1
lsl r2,#4;write  1 into r2, lsl 4_times to move the 1 to bit 4
```

Turning Leds on or off for mode 4 display (not working)

```
cont2:
b cont3
led4:
mov r10,#1
lsl r10,#21   ;bit 23 to write to GPIO23
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
lsl r10,#23   ;bit 23 to write to GPIO23
str r10,[r0,#28] ;Turn on LED 1
mov r10,#1
  mov r0,BASE
 mov r1,$0F0000
  bl Delay
 pop {r0,r1,r2,r7,r10,r11,lr}
 mov r10,#1
lsl r10,#21   ;bit 23 to write to GPIO23
str r10,[r0,#40] ;Turn on LED 1
mov r10,#1
lsl r10,#23   ;bit 23 to write to GPIO23
str r10,[r0,#40] ;Turn on LED 1

  push {r0,r1,r2,r7,r10,r11,lr}
```

Sending parameter half second to delay function

```
cont3:

;call timer
push {r0-r3}
mov r0,BASE
mov r1,$70000
orr r1,$0A100
orr r1,$00020    ;TIMER_MICROSECONDS = 500,000
bl Delay
pop {r0-r3}
```

4. Assembly code Description

Assembly Code	Brief Description	file
ldr r9,[r0,#52]	Loads the number in r9 at the offset 52 in r0.	Led_disp
bic r1,r1,$7000000	#bit clear bits 18,19,20 for GPIO 18	Led_disp
orr r10,r10, $40000	performs bitwise OR operations on the values in R10 and Hex40000. (add 1 to bit 18)	Led_disp
tst r9,#262144	use tst to check bit 18—disabled	Led_disp
bne led2	Branch if the value not equal ..else next	Led_disp
pop {r0,r1,r2,r7,r10,r11,lr}	Pop registers listed off a full descending stack.	Led_disp
lsl r10,#23	Logical Shift Left.. moves bit 1 in r10 23 bits left.	Led_disp
str r10,[r0,#28]	Stores offset in r10 to r0. Turns Leds on	Led_disp
mov r10,#1	copies the value decimal 1 into r10	Led_disp
push {r0,r1,r2,r7,r10,r11,lr}	copies the registers in the list onto the top of the stack.	Led_disp
bl Delay	Save address of next instruction & branch to Delay.	Led_disp
b loopx$	Branch to loopx in program	Led_disp
org $8000	ORG is used to set the address of the *counter*	kernel
mov sp,$1000	Moves the stack address pointer.	kernel
cmp r9,#8;	Compare value in r9 to decimal 8	kernel
add r0,r1	Add r0 and r1 and store in r0.	kernel
include "drawpixel.asm"	Include file drawpixel.asm file when compiling	kernel
sub r4,#1	Subtract value 1 from the value in r4.	kernel
ldrd r6,r7,[r3,#4]	Load r6 and r7 with the value in r3 at offset4	kernel
mul r8,r9	Multiply r8 x r9 value and store in r8	kernel

5. Reflection and Conclusion

Achievements

Although I did not achieve everything I wanted with this project, I did manage to accomplish a couple of significant things. Firstly, to have 8 led lights working at the one time required special assembly code as did getting the second button to work. Although the screen display did not add to the led display the bit of understanding to have the screen display work with the led display was significant.

Difficulties

With this project I did have some challenges which I have yet to solve. The main one was that my branch to go to mode 4 (button 2 on button 1 off) I was not able to complete despite trying a few ways with BNE, BEQ, TST and TEQ code. I was also not able to do one of the lines on my box on the

display screen. I think I will achieve this with extra practice. Overall I am happy with the project and I have achieved what could be developed into a Christmas light display.

6. Appendix

The following ASM files are included with this submission:

- Kernel81.asm
- Led_disp.asm
- Timer2.asm
- Drawpixel.asm
- Drawchar.asm
- Fbinit8.asm