Trac Duc Anh Luong - 103488117

Lab 8

Timers

6. Try compiling and see the error message that comes back.

```
ducanh DESKTOP-RN4OP9Q ~ Downloads FASMARM $ ./fasmarm OK2.ASM kernel7.img
flat assembler for ARM  version 1.43 (built on fasm 1.73.02)  (16384 kilobytes memory)
OK2.ASM [17]:
   mov r2,#103488
processed: mov r2,#103488
error: Immediate value cannot be encoded.
```

7. Convert your student number to Hex, and enter it in your submission document.
#103488 = $19440

8.1. Why does MOV only work with numbers with 24 bits set to 0?
Because of the 20 op-code bits and the 4 remaining bits for the ROR, there are 24 bits total. The barrel shifter uses these 24 bits. Only 8 bits hold the necessary value to transfer.

8.2. How can MOV still be used for numbers that do not satisfy this?
The numerical value to be moved can be stored in more bits using the 64-bit and 84-bit mov instructions, respectively.

8.3. Identify the three bytes (as hex digits) needed to construct your student number, and write the code to load the entire number into a register.

OK2.ASM

```
12   loop$:              ;outer loop - repeat LED on, wait, LED off, wait
13     mov r1,#1
14     lsl r1,#18
15     str r1,[r0,#28]  ;turn LED on
16
17      mov r2,$194
18     wait1$:
19       sub r2,#1
20       cmp r2,#0
21       bne wait1$    ;count from  983040 to 0 (busy wait)
22
23     mov r1,#1  ;can be omitted
24     lsl r1,#18 ;can be omitted
25     str r1,[r0,#40]   ;turn LED off (writing to the pull up register)
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL bash

```
ducanh DESKTOP-RN4OP9Q ~ Downloads FASMARM $ ./fasmarm OK2.ASM kernel7.img
flat assembler for ARM  version 1.43 (built on fasm 1.73.02)  (16384 kilobytes memory)
1 passes, 80 bytes.
```

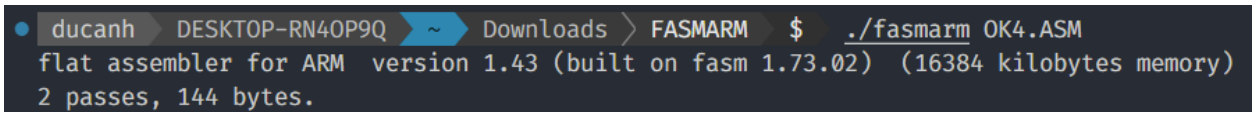
Some Patterned LED Flashing

15.

- Add r2 and assign binary
- Finish one outer loop, r2-1
- Add timer loop outside outer loop (timerloop3)
- If r2 = 0 , go to timerloop3

18.

Output



Code

```
format binary as 'img'  ;must be first
BASE = $FE000000 ; Use $3F000000 for 2B, 3B, 3B+
GPIO_OFFSET = $200000
mov r0,BASE
orr r0,GPIO_OFFSET ;Base address of GPIO
mov r1,#1
lsl r1,#24; GPIO18
str r1,[r0,#4] ;enable output
mov r1,#1
lsl r1,#18
mov r8,BASE
orr r8,TIMER_OFFSET ;store base address of timer (r3)
mov r9,$2D0000
orr r9,$00C600
orr r9,$0000C0   ;TIMER_MICROSECONDS = 3 second
timerloop3:
mov r2,#11
loop$:
 str r1,[r0,#28] ;Turn on LED
 ;new timer
TIMER_OFFSET = $3000
;TIMER_MICROSECONDS = 524288 ; $0080000 ;0.524288 s
mov r3,BASE
orr r3,TIMER_OFFSET ;store base address of timer (r3)
mov r4,$70000
orr r4,$0A100
orr r4,$00020   ;TIMER_MICROSECONDS = 500,000
  ;store delay (r4)
 ldrd r6,r7,[r3,#4]
 mov r5,r6 ;store starttime (r5)(=currenttime (r6))
 timerloop:
```

```
  ldrd r6,r7,[r3,#4] ;read currenttime (r6)
  sub r8,r6,r5  ;remainingtime (8)= currenttime (r6) - starttime (r5)
  cmp  r8,r4  ;compare remainingtime (r8), delay (r4)
  bls timerloop ;loop if LE (reamainingtime <= delay)
 str r1,[r0,#40]  ;turn off LED
 ;re-use timer
 ldrd r6,r7,[r3,#4]
 mov r5,r6 ;store starttime (r5)(=currenttime (r6))
 timerloop2:
  ldrd r6,r7,[r3,#4] ;read currenttime (r6)
  sub r8,r6,r5  ;remainingtime (8)= currenttime (r6) - starttime (r5)
  cmp  r8,r4  ;compare remainingtime (r8), delay (r4)
  bls timerloop2 ;loop if LE (reamainingtime <= delay)
sub r2,r2,#1
 b loop$
cmp r2,#0
beq timerloop3
```