

COS10004: Computer Systems

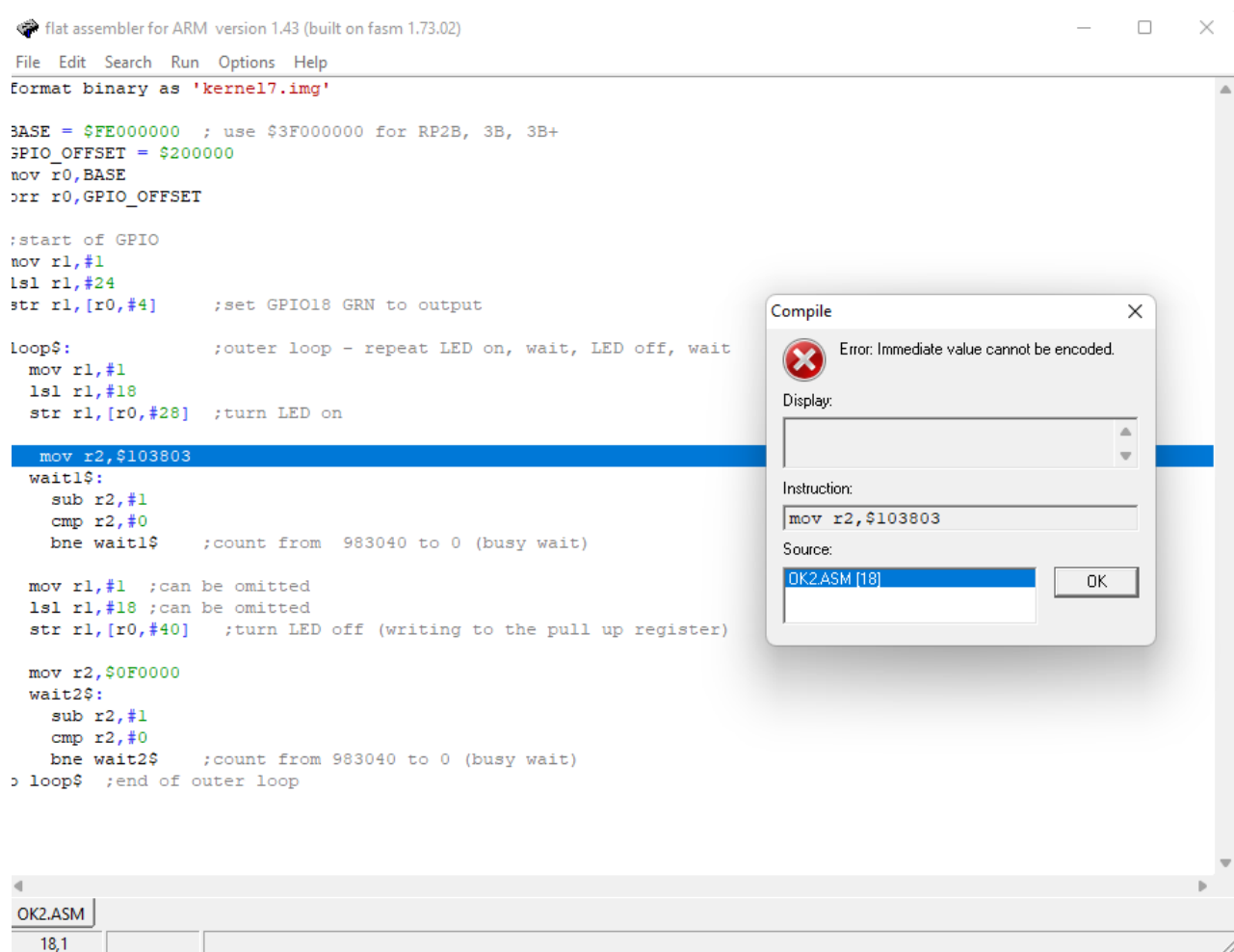
Lab 8

Name: SWH00420 Tran Quoc Dung

Student ID: 103803891

Timers

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



The screenshot shows the flat assembler for ARM version 1.43 (built on fasm 1.73.02) interface. The main window displays an assembly file named 'kernel7.img'. The code includes several instructions, with the line 'mov r2, \$103803' highlighted in blue. A 'Compile' dialog box is open, displaying the error message 'Error: Immediate value cannot be encoded.' The 'Display' field is empty, the 'Instruction' field shows 'mov r2, \$103803', and the 'Source' field shows 'OK2.ASM [18]'. The 'OK' button is visible in the dialog box.

```
flat assembler for ARM version 1.43 (built on fasm 1.73.02)
File Edit Search Run Options Help
Format binary as 'kernel7.img'

BASE = $FE000000 ; use $3F000000 for RP2B, 3B, 3B+
GPIO_OFFSET = $200000
mov r0, BASE
orr r0, GPIO_OFFSET

;start of GPIO
mov r1, #1
lsl r1, #24
str r1, [r0, #4] ;set GPIO18 GRN to output

loop$:
;outer loop - repeat LED on, wait, LED off, wait
mov r1, #1
lsl r1, #18
str r1, [r0, #28] ;turn LED on

mov r2, $103803
wait1$:
sub r2, #1
cmp r2, #0
bne wait1$ ;count from 983040 to 0 (busy wait)

mov r1, #1 ;can be omitted
lsl r1, #18 ;can be omitted
str r1, [r0, #40] ;turn LED off (writing to the pull up register)

mov r2, $0F0000
wait2$:
sub r2, #1
cmp r2, #0
bne wait2$ ;count from 983040 to 0 (busy wait)
loop$ ;end of outer loop
```

7. Convert your student number to Hexadecimal and enter it in your submission document.

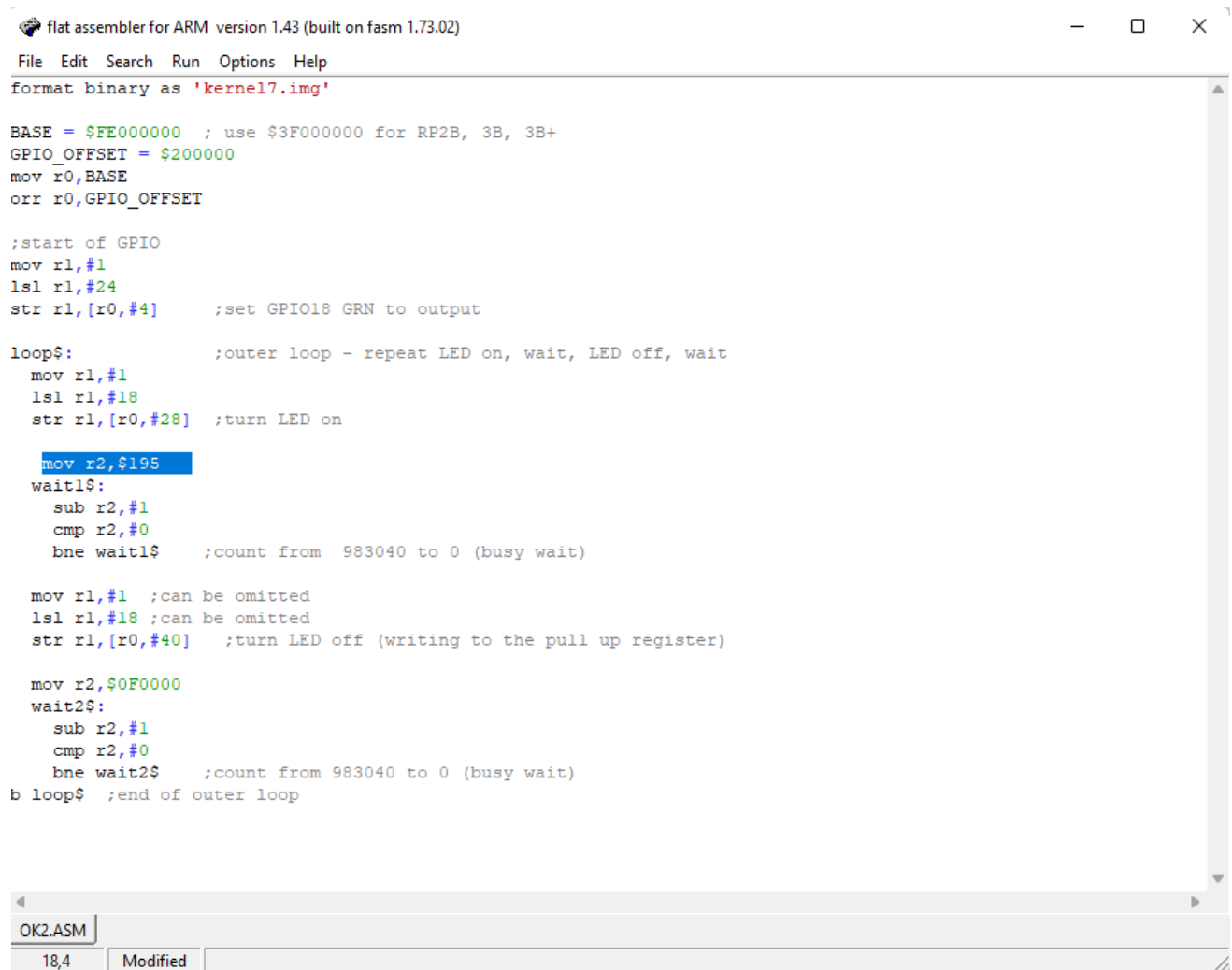
#103803 = \$1957 (1957B)

8.1. Why does MOV only work with numbers with 24 bits set to 0 ? Because only 8 bits contains value that need to move. While 20 bits are used for op-code, 4 other bits left used for a the ROR, both used for barrel shifter.

8.2. How can MOV still be used for numbers that do not satisfy this ? 64bit and 84bit mov instructions that can use more bits to store the number value to move.

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.

Code:



```
flat assembler for ARM version 1.43 (built on fasm 1.73.02)
File Edit Search Run Options Help
format binary as 'kernel7.img'

BASE = $FE000000 ; use $3F000000 for RP2B, 3B, 3B+
GPIO_OFFSET = $2000000
mov r0,BASE
orr r0,GPIO_OFFSET

;start of GPIO
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ;set GPIO18 GRN to output

loop$: ;outer loop - repeat LED on, wait, LED off, wait
    mov r1,#1
    lsl r1,#18
    str r1,[r0,#28] ;turn LED on
    mov r2,$195
wait1$:
    sub r2,#1
    cmp r2,#0
    bne wait1$ ;count from 983040 to 0 (busy wait)

    mov r1,#1 ;can be omitted
    lsl r1,#18 ;can be omitted
    str r1,[r0,#40] ;turn LED off (writing to the pull up register)

    mov r2,$0F0000
wait2$:
    sub r2,#1
    cmp r2,#0
    bne wait2$ ;count from 983040 to 0 (busy wait)
b loop$ ;end of outer loop
```

OK2.ASM
18,4 Modified

Some Patterned LED Flashing

format binary as 'kernel7.img'

BASE = \$FE000000 ; use \$3F000000 for RP2B, 3B, 3B+

GPIO_OFFSET = \$200000

mov r0,BASE

orr r0,GPIO_OFFSET

;start of GPIO

mov r1,#1

lsl r1,#24

str r1,[r0,#4] ;set GPIO18 GRN to output

loop\$: ;outer loop - repeat LED on, wait, LED off, wait

mov r1,#1

lsl r1,#18

str r1,[r0,#28] ;turn LED on

mov r2,\$0F0000

wait1\$:

sub r2,#1

cmp r2,#0

bne wait1\$;count from 983040 to 0 (busy wait)

mov r1,#1 ;can be omitted

lsl r1,#18 ;can be omitted

str r1,[r0,#40] ;turn LED off (writing to the pull up register)

mov r2,\$0F0000

```
wait2$:  
    sub r2,#1  
    cmp r2,#0  
    bne wait2$ ;count from 983040 to 0 (busy wait)  
b loop$ ;end of outer loop
```