

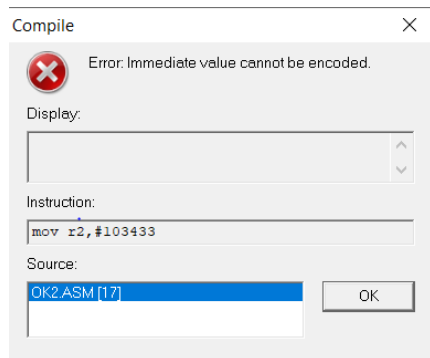
LAB 08

COS10004 – Computer System

NGO CONG THANH

11/14/2021

Insert the error message into your submission document



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

#103433 = \$19409

8.1. Why does MOV only work with numbers with 24 bits set to 0?

Because in that 24 bits, there are 20 bits for op-code, 4 left is for the ROR. These 24 bits use for barrel shifter. Only 8 bits contains value that need to move.

8.2. How can MOV still be used for numbers that do not satisfy this?

We have 64bit and 84bit mov instructions that can use more bits to store the number value to move.

8.3

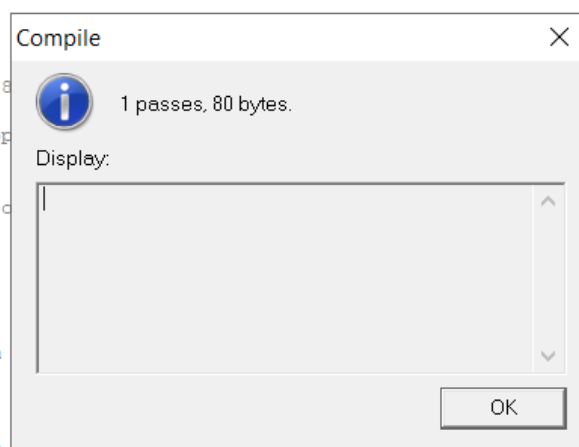
```
GPIO_OFFSET = $200000
mov r0, BASE
orr r0, GPIO_OFFSET
;start of GPIO
mov r1, #1
lsl r1, #24
str r1, [r0, #4] ;set GPIO18

loop$:
    mov r1, #1
    lsl r1, #18
    str r1, [r0, #28] ;turn LED on

    mov r2, $194
wait1$:
    sub r2, #1
    cmp r2, #0
    bne wait1$ ;count from 194 to 0

mov r1, #1 ;can be omitted
lsl r1, #18 ;can be omitted
str r1, [r0, #40] ;turn LED on

mov r2, $0F0000
wait2$:
```



Include your notes or comments in your submission document:

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

Demonstrate your program to your lab demonstrator and copy your code into your submission document:

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

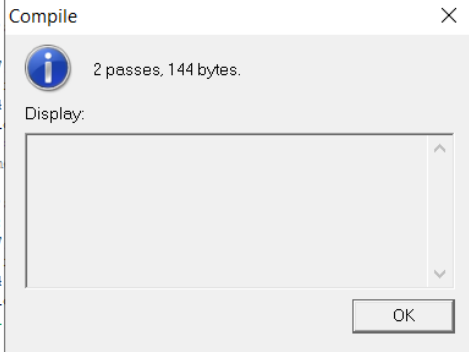
```

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,$00020 ;TIMER_MICROSECONDS = 500,000
;store delay (r4)
ldrd r6,r7,[r3,#4]
mov r5,r6 ;store starttime (r5)(=currenttime (r6))
timerloop:
ldrd r6,r7,[r3,#4] ;read currenttime (r6)
sub r8,r6,r5 ;remainingtime (8)= currenttime (r6) - starttime (r5)
cmp r8,r4 ;compare remainingtime (r8), delay (r4)
bls timerloop ;loop if LE (remainingtime <= delay)
str r1,[r0,#40] ;turn off LED
;re-use timer
ldrd r6,r7,[r3,#4]
mov r5,r6 ;store starttime (r5)(=currenttime (r6))
timerloop2:
ldrd r6,r7,[r3,#4] ;read currenttime (r6)
sub r8,r6,r5 ;remainingtime (8)= currenttime (r6) - starttime (r5)
cmp r8,r4 ;compare remainingtime (r8), delay (r4)
bls timerloop2 ;loop if LE (remainingtime <= delay)
sub r2,r2,#1
b loop$
cmp r2,#0
beq timerloop3

```

File Edit Search Run Options Help

```
orr r4,$0A100
orr r4,$00020 ;TIMER_MICROSECONDS = 500,000
;store delay (r4)
ldrd r6,r7,
mov r5,r6 ;
timerloop:
ldrd r6,r7
sub r8,r6,
cmp r8,r4
bls timerl
str rl,[r0,
;re-use tim
ldrd r6,r7,
mov r5,r6 ;
timerloop2:
ldrd r6,r7
sub r8,r6,
cmp r8,r4
bls timerl
sub r2,r2,#1
b loop$
cmp r2,#0
beq timerloop3
```



time (r5)

time (r5)

<

OK2.ASM | Untitled | OK4.ASM |