Chapter 4 Homework

Charles Oroko

8 November 2018

Problem 4.1. If $r0$ initially contains 1, what will it contain after the third instruction in the sequence below?

```
        add     r0, r0, #1          @r0=2
        mov     r1, r0              @r1=2
        add     r0, r1, r0 lsl #1   @r0=6
```

$r0 = 6$

Problem 4.2. What will r0 and r1 contain after each of the following instructions? Give your answers in base 10.

```
        mov     r0, #1      @r0=1
        mov     r1, #0x20   @r1=32
        orr     r1, r1, r0  @r1=33
        lsl     r1, #0x2    @r1=132
        orr     r1, r1, r0  @r1=133
        eor     r0, r0, r1  @r0=132
        lsr     r1, r0, #3  @r1=1056
```

$$r0 = 132$$
$$r1 = 1056$$

Problem 4.3. What is the difference between *lsr* and *asr*?

The *lsr* and *asr* operations do similar things. They both shifts each bit n bits to the right, losing the least significant n bits.

With the *lsr* operation, zero is shifted into the n most significant bits. However, with the *asr* operation, the n most significant bits become copies of the sign bit (bit 31).

Problem 4.4. Write the ARM assembly code to load the numbers stored in $num1$ and $num2$, add them together, and store the result in $numsum$. Use only $r0$ and $r1$.

```
    ldr     r0, =num1
    ldr     r1, =num2
    ldr     r0, [r0]
    ldr     r1, [r1]
    add     r0, r0, r1

    ldr     r1, =numsum
    str     r0, [r1]
```

Problem 4.5. Given the following variable definitions:

```
        num1:   .word x
        num2:   .word y
```

where you do not know the values of x and y, write a short sequence of ARM assembly instructions to load the two numbers, compare them, and move the largest number into register r0.

```
    ldr     r1, =num1
    ldr     r2, =num2
    ldr     r0, [r1]     @assume x>=y
    ldr     r2, [r2]
    cmp     r0, r2       @change r0=y if x<y

    bge     done
    mov     r0, r2

    done:
```

Problem 4.6. Assuming that a is stored in register r0 and b is stored in register r1, show the ARM assembly code that is equivalent to the following C code.

```
        if ( a & 1 )
        a = -a;
        else
        b = b+7;
```

```
@if (a&1)
and    r2, r0, #1
cmp     r2, #0

@a=-a
beq     else
equal:
rsb     r0, r0, #0
b       end

@else b=b+7
else:
add     r1, r1, #7

end:
```