The University of Alabama in Huntsville
ECE Department
CPE 221 01
Fall 2019
Homework #3 Solution

3.3 (20), 3.11(5), 3.13(15), 3.28(5), 3.32(20), 3.36(15) ARM Assembly 3.99(30) All assembly language programs must run without errors on the simulator

3.3 For each of the following 6-bit operations, calculate the values of the C, Z, V, and N flags.

(a) 110101 + 001101

```
   110101
   001101
  ________
   000010   S
  111101    C        C = 1, Z = 0, V = 1 ⊕ 1 = 0, N = 0
```

(b) 111111 + 000001

```
   111111
   000001
  ________
   000000   S
  111111    C        C = 1, Z = 1, V = 1 ⊕ 1 = 0, N = 0
```

(c) 011000 – 111111

```
   011000
   000001
  ________
   011001   S
  000000    C        C = 0, Z = 0, V = 0 ⊕ 0 = 0, N = 0
```

(d) 101101 + 011011

```
   101101
   011011
  ________
   001000   S
  111111    C        C = 1, Z = 0, V = 1 ⊕ 1 = 0, N = 0
```

(e) 000000 – 000001

```
   000000
   111111
  ________
   111111   S
  000000    C        C = 0, Z = 0, V = 0 ⊕ 0 = 0, N = 1
```

3.11 If r1 = 1111 0000 1110 0010 1010 0000 1111 1101 and r2 = 0000 0000 1111 1111 0000 1111 0000 1111, what is the value of r3 after executing `EOR r3, r1, r2`?

3.13 If `r1 = 0xEA00 11FD` and r2 = 16, what is the value of r0 after each of the following instructions has been executed (assume that each instruction uses the same data)?

(a) `ADD r0, r1, r1, LSL #2`

```
r1         =   1110 1010 0000 0000 0001 0001 1111 1101
r1 LSL #2  =   1010 1000 0000 0000 0100 0111 1111 0100
ADD        =   1001 0010 0000 0000 0101 1001 1111 0001
```

(b) `ADD r0, r1, r1, ASR #4`

```
r1         =   1110 1010 0000 0000 0001 0001 1111 1101
r1 ASR #4  =   1111 1110 1010 0000 0000 0001 0001 1111
ADD        =   1110 1000 1010 0000 0001 0010 1101 1100
```

(c) `ADD r0, r1, r1, ROR #7`

```
r1           =    1110 1010 0000 0000 0001 0001 1111 1101
r1 ROR #7    =    1111 1011 1101 0100 0000 0000 0010 0011
ADD          =    1110 0101 1101 0100 0001 0001 1010 0000
```

3.28 What effective address is generated by the instruction `LDR r0, [r2, -r3, LSL #1]`?

EA = r2 – (r3 *2)

3.32 Assume that r2 contains the initial value 0x00001000. Explain the effect of each of the following six instructions, and give the value in r2 after each instruction executes. Use register transfer notation.

(a) `STR r1, [r2]`

M[r2] ← r1, r2 = 0x0000 1000

(b) `STR r1, [r2, #8]`

M[r2 + 8] ← r1, r2 = 0x0000 1000

(c) `STR r1, [r2, #8]!`

M[r2 + 8] ← r1, r2 = 0x0000 1008

(d) `STR r1, [r2] #8`

M[r2] ← r1, r2 = 0x0000 1008

(e) `STR r1, [r2, r0, LSL #8]`

M[r2 + r0*256] ← r1, r2 = 0x0000 1000

3.36 Without using the ARM's multiplication instruction, write one or more instructions (using ADD, SUB, and shifting) to multiply by the following integer.

(d) 109

```
MOV   r1, #0
ADD   r1, r1, r2, LSL #7        ; r1 = 128 * r2
SUB   r1, r1, r2, LSL #4        ; r1 = 128 * r2 – 16 * r2 = 112 * r2
SUB   r1, r1, r2, LSL #1        ; r1 = 112 * r2 – 2 * r2 = 110 * r2
SUB   r1, r1, r2                ; r1 = 110 * r2 – r2 = 109 * r2
```

3.99 Write ARM assembly code to implement the following C statements, assuming all variables are 32-bit integers and a declaration of x[10], y[10]:

```
int x[10] = {8, 2, 9, 6, 7, 0, 1, 3, 5, 4};
for i = 0; i < 10; i++)
  y[x[i]] = i + 20;
```

```
        AREA ARRAY_PLUS_SCALAR, CODE, READWRITE
        ENTRY
        ADR   r0, x                 ; pointer to first element of x
        ADR   r1, y                 ; pointer to first element of y
        LDR   r2, size              ; holds size of x
        LDR   r3, i                 ; holds loop counter
loop    CMP   r3, r2                ; compare i and size
        BGE   done                  ; if i >= size, done
        LDR   r4, [r0, r3, LSL #2]  ; r4 = x[i]
        ADD   r5, r3, #20           ; r5 = i + 20
        STR   r5, [r1, r4, LSL #2]  ; y[x[i]] = i + 20
        ADD   r3, r3, #1            ; i++
        B     loop
done    B     done
size    DCD   10
x       DCD   8, 2, 9, 6, 7, 0, 1, 3, 5, 4
y       SPACE 40
i       DCD   0
        END
```