CS1021 Tutorial #3
Condition Code Flags and Basic Flow Control

1 Condition Code Flags

(a) Determine whether the Overflow flag would be set after performing each of the following 4-bit arithmetic operations. (Assume a 4-bit microprocessor and a 4-bit 2's Complement system are being used.) Explain your answers.

(i) $0100_2 + 0010_2$

(ii) $0101_2 + 0100_2$

(iii) $1110_2 + 0101_2$

(iv) $1101_2 + 1001_2$

(b) Consider each of the highlighted instructions in the ARM Assembly Language program below. In each case, calculate the value stored in the destination register and state whether each of the N (Negative), Z (Zero), C (Carry) and V (oVerflow) flags would be set or clear (true or false) after the execution of the highlighted instruction. Provide a detailed explanation for your answer in each case.

```
    LDR     R0, =0xC0001000
    LDR     R1, =0x51004000
    ADDS    R2, R0, R1    ; result? flags?
    LDR     R3, =0x92004000
    SUBS    R4, R3, R3    ; result? flags?
    LDR     R5, =0x74000100
    LDR     R6, =0x40004000
    ADDS    R7, R5, R6    ; result? flags?
    LDR     R1, =0x6E0074F2
    LDR     R2, =0x211D6000
    ADDS    R0, R1, R2    ; result? flags?
    LDR     R1, =0xBE2FDD2E
    LDR     R2, =0x41D022D2
    ADDS    R0, R1, R2    ; result? flags?
```

(c) Find pairs of 32-bit values which, when added together using the ADDS instruction, cause the following combinations of the conditions code flags to be set or cleared (1 or 0).

(i) N = 0; Z = 0; C = 0; V = 0

(ii) N = 1; Z = 0; C = 0; V = 0

(iii) N = 0; Z = 0; C = 1; V = 0

(iv) N = 1; Z = 0; C = 1; V = 0

(v) N = 0; Z = 1; C = 1; V = 0

(vi) N = 0; Z = 1; C = 0; V = 0

(vii) N = 1; Z = 0; C = 0; V = 1

(viii) N = 0; Z = 0; C = 1; V = 1

(ix) N = 0; Z = 1; C = 1; V = 1

2 Flow Control

(a) The ARM Assembly Language program shown below computes x^y. The program uses branch instructions to implement a loop construct, similar to a *for* or *while* loop in Java. Modify the program so it computes $x!$ instead of x^y. Begin by modifying the pseudo-code and be sure to provide pseudo-code comments for your Assembly Language program.

```
        LDR     r1, =3          ; test with x = 3
        LDR     r2, =4          ; test with y = 4
        MOV     r0, #1          ; result = 1

while
        CMP     r2, #0
        BEQ     endwh           ; while (y != 0) {
        MUL     r0, r1, r0      ;   result = result * x
        SUB     r2, r2, #1      ;   y = y - 1
        B       while           ; }
endwh

stop    B       stop
```

(b) Translate each of the following pseudo-code constructs into ARM Assembly Language. You should assume that all values are unsigned. (i.e. Do not interpret any values as signed 2's complement values.)

(i) Assume `x` is stored in `R0`.

```
if (x == 0)
{
    x = x + 5;
}
```

(ii) Assume `x` is stored in `R0`.

```
if (x != 0)
{
    x = 1;
}
else
{
    x = x * 2;
}
```

(iii) Assume `x` is stored in `R0`.

```
if (x >= 5)
{
    x = 0;
}
```