Penalo B. Thibeaud 40212017

**Question 1)**
Suppose
R0 = 0xFFFFFFFF, N
R1 = 0x00000001,
R2 = 0x00000000,
and initially N, Z, C, V flags are zero. Find out the value of the NZCV flags of the following instructions. (Assume each instruction runs individually, i.e., these instructions are not part of a program.) (10 points)

a) ADD R3, R0, R2
b) SUBS R3, R0, R0
c) ADDS R3, R0, R2
d) LSL R3, R0, #1
e) LSRS R3, R1, #1
f) ANDS R3, R0, R2

N: if the result is negative
Z: if the result is zero
C: if there is a carry out, borrow
V: if the results of an operation cannot fit in 32 bits

| | | | | | | N | Z | C | V |
|---|---|---|---|---|---|---|---|---|---|
| A) | ADD | $R_3$, | $R_0$, | $R_2$ | Result = 0xFFFF + 0x0000 = 0xFFFF | 1 | 0 | 0 | 0 |
| B) | SUBS | $R_3$, | $R_0$, | $R_0$ | Result = 0xFFFF + 0x0001 = 0x0000 (U) | 0 | 1 | 1 | 0 |
| C) | ADDS | $R_3$, | $R_0$, | $R_2$ | Result = 0xFFFF + 0x0000 = 0xFFFF (U) | 1 | 0 | 0 | 0 |
| D) | LSL | $R_3$, | $R_0$, | #1 | Result = 0xFFFF ← 1 = 0xFFF1 | 1 | 0 | 1 | 0 |
| E) | LSRS | $R_3$, | $R_1$, | #1 | Result = 0x0001 → 1 = 0x0000 (U) | 0 | 1 | 1 | 0 |
| F) | ANDS | $R_3$, | $R_0$, | $R_2$ | Result = 0xFFFF ∩ 0x0000 = 0x0000 (U) | 0 | 1 | 0 | 0 |

**Question 2)**

Suppose R0 = 0x20000000, R1 = 0x12345678 and R2 = 0x00000001. All bytes in memory are initialized to 0x00. The following assembly program has been executed successfully.

```
STR R1, [R0], #4
STR R1, [R0, #4]!
STR R1, [R0, #4]
LDR R1, [R0], -R2, LSL#3
LDR R1, [R0, R2, LSR#1]
```

a) What are the register R0, R1 and R2 values after each instruction.

b) Show the corresponding memory values after each instruction if the processor uses **Big Endian**. (20 points)

| | $R_0$ | $R_1$ | $R_2$ | 0x20000000 | 0x20000008 | 0x2000000C | |
|---|---|---|---|---|---|---|---|
| initial value | 0x20000000 | 0x12345678 | 0x00000001 | 0x00 | 0x00 | 0x00 | |
| STR R1, [R0], #4 | 0x20000004 | 0x12345678 | 0x00000001 | 0x12345678 | 0x00 | 0x00 | |
| STR R1, [R0, #4]! | 0x20000008 | 0x12345678 | 0x00000001 | 0x12345678 | 0x12345678 | 0x00 | |
| STR R1, [R0, #4] | 0x20000008 | 0x12345678 | 0x00000001 | 0x12345678 | 0x12345678 | 0x12345678 | |
| LDR R1, [R0], -R2, LSL#3 | 0x20000008 | 0x00 | 0x00000001 | 0x12345678 | 0x12345678 | 0x12345678 | |
| LDR R1, [R0, R2, LSR#1] | 0x20000008 | 0x00 | 0x00000001 | 0x12345678 | 0x12345678 | 0x12345678 | |

$R_0 + (-2 \ll 3) = R_0 - 16 = 0x20000008 - 0x00000010 = 0x1FFFFFF8$ is 0x00

$R_0 + (R_2 \gg 1) = R_0 + 0x00 = R_0 = 0x20000008 = 0x12345678$ is 0x00

Big Endian is MSB first

**Question 3)**

Given an array of characters terminated with a zero. For example,
Array1: .word ‘h’,‘e’,’l’,’l’, ‘o’, 0
Write an assembly program using ARM instructions that copies the source string at address Array1 to a destination string at address Array2.
Note that characters are byte size. (20 points)

```
.Syntax unified
.Cpu Cortex-m4
.thumb

.global start
.data
    Array1: .byte 'h','e','l','l','o', 0
    Array2: .space 6

.text
start:
    ldr r0, =Array1
    ldr r1, =Array2
    mov r5, #0

copy_loop:
    ldrb r2, [r0, r5]
    strb r2 [r0, r5]

    cmp r2, #0
    beq copy_end

    add r5, r5, #1
    b copy_loop

copy_end: b.
```

**Question 4)**

An assembly program is to be written to calculate the Greatest Common Divider (gcd) using the following Euclid's Algorithm: (20 points)

```
uint32_t a, b;
while (a != b ) {
    if (a > b)
        a = a – b;
    else
        b = b – a;
}
```

Given register R0 = a and register R1 = b

a) Write an assembly program using SIMARM instruction set.

b) Rewrite the program for ARM processor using conditional execution of Data instructions.

A)

```
.syntax unified
.cpu cortex-m4
.thumb

.global start
.text

start:
    cmp  R0, R1
    Beq  end

loop-start:
    cmp  R0, R1
    BGT  sub_A
    BLT  sub_B

sub_A:
    suB  R0, R0, R1
    b  loop_start

sub_B:
    suB  R1, R1, R0
    b  loop-start

end: b end
```

B)

```
.syntax unified
.cpu cortex-m4
.thumb

.global start
.text

start:
    cmp  R0, R1
    beq  end

Loop:
    Cmp  R0, R1
    subGt  R0, R0, R1
    sublt  R1, R1, R0
    BNE  Loop

end:  b.
```

**Question 5)**

Write a program that compares two arrays, A and B. Each array contains 100 8-bit signed numbers. Compare corresponding elements of the two arrays until either two elements are found to be unequal or all elements of the arrays have been compared and found to be equal. Assume the arrays start at addresses 0xA000 and 0xB000, respectively. If the two arrays are found to be unequal save the address of the first unequal element of A array in the memory location with address FOUND, otherwise write all 0s in this location. (30 points)

```
.syntax unified
.cpu cortex-m4
.thumb

.global start
.data
   Found: .word 0X00000000

.text
start:
    ldr R0, =0XA000
    ldr R1, =0XB000
    mov R3, #100
    ldr R2, =Found

loop:
    ldrb R4, [R0], #1
    ldrb R5, [R1], #1
    cmp R4, R5
    bne unequal
    subs R3, R3, #1
    bne loop

    mov R4, #0
    str R4, [R2]
    b end

unequal:
    sub R0, R0, #1
    str R0, [R2]

end: b.
```