과제: ARM 프로세서 실습(1)

실습 내용

1. 온라인 시뮬레이터(https://cpulator.01xz.net/)에서 Architecture를 ARMv7으로, System을 ARMv7 generic으로 선택하여 시뮬레이터를 실행시켜보시오.

2. 다음 프로그램을 작성하여 single step(F2)으로 실행하여 동작을 확인하시오.
   (변경된 레지스터값, 플래그값 확인)

```
_start:
        mov r0, #0x10
        mov r1, #0x20
        mov r2, #0              // r2
        add r3, r0, r1          // r3
        sub r4, r0, r1          // r4, NZCV(CPSR의 상위4비트)
        movlt r2, #1            // r2, conditional move
        subs r4, r0, r1         // r4, NZCV
        movlt r2, #2            // r2, conditional move
        rsb r4, r0, r1          // r4, NZCV
        movlt r2, #3            // r2
        rsbs r4, r0, r1         // r4, NZCV
        movlt r2, #4            // r2
stop:
        b stop
```

3. 다음과 같이 동작하는 프로그램을 작성하고 시뮬레이터에서 실행시켜 보시오.
   r0에 16진수 75 (0111 0101)를 저장하시오.
   r0의 bit 5와 4를 0으로 변경하여 r1에 저장하시오.
   r0의 bit 3과 2를 1로 변경하여 r2에 저장하시오.
   r0의 bit 1과 0를 반대로 변경하여 r3에 저장하시오.
   r0의 모든 비트를 반대로 변경하여 r4에 저장하시오.
   r0와 r4가 같은지 여부를 비교하고, 다르면 r4를 1 증가시키시오.
   r0과 r4가 서로 다른 부호이면 r0를 1 증가시키시오.

4. 다음 프로그램을 각각 작성하여 single step(F2)으로 실행하여 동작을 확인하시오. (변경된 레지스터값, 플래그값 확인, 필요한 경우 동작 설명)

```
.global _start
_start:
// shifted operand
        mov r0, #0x10           // r0
        mov r1, #4              // r1

        mov r2, r0, lsl #3      // r2,
        lsl r2, r0, #3          // 앞명령어와 비교
        mov r3, r0, lsr r1      // r3,
        lsr r3, r0, r1          // 앞명령어와 비교

        mov r0, #-8             // r0
        mov r4, r0, asl #2      // r4, lsl을 사용한 경우와 비교
        mov r5, r0, asr #3      // r5
        asr r5, r0, #3          // 앞명령어와 비교
        mov r6, r0, lsr #3

// multiplication using shifted operand
        mov r0, #10
        add r1, r0, r0, lsl #2  // r1 = r0*5 = r0*(1+4)
        rsb r2, r0, r0, lsl #4  // r2 = r0*15 = r0*(16-1)
        rsb r2, r2, r2, lsl #3  // r2 = r2*7 = r2*(8-1)
// multiplication
        mov r0, #10
        mov r1, #15
        mul r2, r0, r1          // r2 = r0 * r1
        mov r4, #4;
        mla r2, r4, r4, r2      // r2 = r2 + r4*r4

        mov r0, #0x80000001
        mov r1, #8
        mul r2, r0, r1          // r2, multiply
        umull r3, r4, r0, r1    // r3,r4, multiply-long (unsigned)
        smull r5, r6, r0, r1    // r5,r6, multiply-long (signed)
stop:
        b stop
```

```
.global _start
_start:
        mov r0, #0x12           // r0 (small number)
        ldr r1, data1           // r1 (large number)

        ldr r2, =0x104          // 실제 코드
        ldr r3, =0xffffff55     // 실제 코드,
        ldr r4, =0x102          // 실제 코드, PC offset, 숫자 저장위치
        ldr r5, =0x77777777     // 실제 코드, PC offset, 숫자 저장위치
stop:
        b stop

data1:  .word 0x12345678        // 32-bit data
```