

강의명 : 마이크로프로세서
실습 번호 : 3
실습 제목 : APSR register
학생 이름 : 이학민
학번 : 201910906

1. APSR register의 bit 출력

1.1

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
for(int cnt=0; cnt<9; cnt++)
{
    switch(cnt)
    {
        case 0:
            n=apsr&(1<<31)?1:0;
            break;
        case 1:
            z=apsr&(1<<30)?1:0;
            break;
        case 2:
            c=apsr&(1<<29)?1:0;
            break;
        case 3:
            v=apsr&(1<<28)?1:0;
            break;
        case 4:
            q=apsr&(1<<27)?1:0;
            break;
        case 5:
            ge1=apsr&(1<<19)?1:0;
            break;
        case 6:
            ge2=apsr&(1<<18)?1:0;
            break;
        case 7:
            ge3=apsr&(1<<17)?1:0;
            break;
        case 8:
            ge4=apsr&(1<<16)?1:0;
            break;
```

```

        default:
            break;
    }
}

printf("N=%d,    Z=%d,    C=%d,    V=%d,    Q=%d,    GE[3:0]=%d%d%d%d\n",
n,z,c,v,q,ge1,ge2,ge3,ge4);

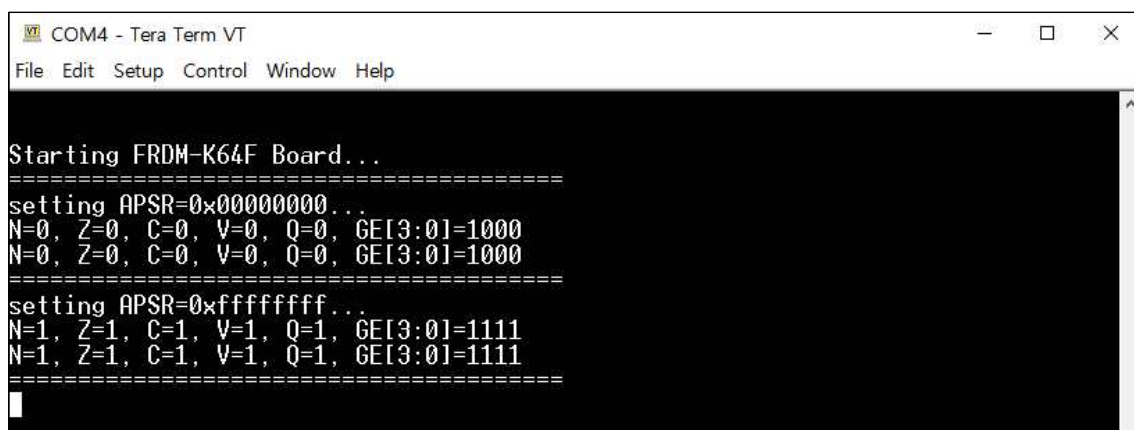
__asm(" mov    r0, %0": :r"(apsr));
__asm(" msr    apsr_nzcvqg, r0");

```

<프로그램 설명>

우선, APSR register에서 총 9개의 status를 표현하는 bit을 저장하기 위하여 int형 변수 9개를 선언하였다. 총 9개의 bit를 알아내기 위하여 switch문이 9번 반복되도록 반복문을 수행하였다. switch문 안에서 shift 연산자를 이용하여 원하는 주소의 값을 뽑아낼 수 있도록 만들었다. 예를 들어, N(negative) bit는 APSR register의 31번지에 위치해 있는데 이 값을 알아내기 위하여 31번지만 1이고 나머지는 모두 0인 32bit 길이의 비트와 AND 계산을 한다. N bit의 값이 0이었을 경우 AND 계산을 한 결과 32bit 모두 0, 즉 false가 되기 때문에 삼항 연산자에 의해 변수 n에 0이 대입된다. 반대로 N bit의 값이 1이었을 경우 AND 계산을 한 결과 31번지에는 1, 나머지는 모두 0이 된다. 계산 결과인 32bit 길이의 비트가 0이 아니므로 true가 되기 때문에 삼항 연산자에 의해 변수 n에 1이 대입된다. 이를 일반화하여 x번지의 비트는 $apsr \& (1 \ll (x-1)) ? 1 : 0$ 을 사용하여 얻을 수 있다. 따라서 각 조건에 맞는 bit의 주소를 적용하여 case를 총 9개로 나눈다. 마지막으로 도출한 비트의 값을 printf를 이용하여 출력한다. apsr_print() 함수가 수행되고 나면 aspr register의 값이 변하므로 inline assembly 2줄을 사용하여 원래의 값으로 되돌려 준다.

1.2



```

COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=1000
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=1000
=====
setting APSR=0xffffffff...
N=1, Z=1, C=1, V=1, Q=1, GE[3:0]=1111
N=1, Z=1, C=1, V=1, Q=1, GE[3:0]=1111
=====

```

2. APSR register의 bit N set

2.1

apsr_print() 함수 내에서의 작성 :

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
for(int cnt=0; cnt<9; cnt++)
{
    switch(cnt)
    {
        case 0:
            n=apsr&(1<<31)?1:0;
            break;
        case 1:
            z=apsr&(1<<30)?1:0;
            break;
        case 2:
            c=apsr&(1<<29)?1:0;
            break;
        case 3:
            v=apsr&(1<<28)?1:0;
            break;
        case 4:
            q=apsr&(1<<27)?1:0;
            break;
        case 5:
            ge1=apsr&(1<<19)?1:0;
            break;
        case 6:
            ge2=apsr&(1<<18)?1:0;
            break;
        case 7:
            ge3=apsr&(1<<17)?1:0;
            break;
        case 8:
            ge4=apsr&(1<<16)?1:0;
            break;
        default:
            break;
    }
}

printf("N=%d,      Z=%d,      C=%d,      V=%d,      Q=%d,      GE[3:0]=%d%d%d%d\n",
```

```
n,z,c,v,q,ge1,ge2,ge3,ge4);
__asm(" mov    r0, %0": :r"(apsr));
__asm(" msr     apsr_nzcvqg, r0");
```

<프로그램 설명>

1.1에서 작성한 내용과 같습니다.

main 함수 내에서의 작성 :

```
__asm("mov r1, #10");
__asm("subs r2, r1, #20");
```

<프로그램 설명>

N bit를 1로 만들기 위해 계산 결과가 음수가 되도록 한다. 따라서 $10-20=-10$ 을 수행하는 과정을 나타냈다. inline assembly를 사용해 상수 10을 r1에 넣고, $r2=r1-20$ 을 통해 최종적으로 $-10=10-20$ 을 구현하였다. 작은 숫자를 이용하는 것이므로 mov를 사용하였다. 또한 sub 뒤에 붙은 s는 status register에 반영되도록 붙인 suffix이다.

2.2

```
COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
=====
setting N=1 of APSR...
N=1, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
N=1, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
=====
```

3. APSR register의 bit Z set

3.1

apsr_print() 함수 내에서의 작성 :

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
for(int cnt=0; cnt<9; cnt++)
{
    switch(cnt)
    {
        case 0:
            n=apsr&(1<<31)?1:0;
            break;
        case 1:
            z=apsr&(1<<30)?1:0;
            break;
        case 2:
            c=apsr&(1<<29)?1:0;
            break;
        case 3:
            v=apsr&(1<<28)?1:0;
            break;
        case 4:
            q=apsr&(1<<27)?1:0;
            break;
        case 5:
            ge1=apsr&(1<<19)?1:0;
            break;
        case 6:
            ge2=apsr&(1<<18)?1:0;
            break;
        case 7:
            ge3=apsr&(1<<17)?1:0;
            break;
        case 8:
            ge4=apsr&(1<<16)?1:0;
            break;
        default:
            break;
    }
}

printf("N=%d,      Z=%d,      C=%d,      V=%d,      Q=%d,      GE[3:0]=%d%d%d%d\n",
```

```
n,z,c,v,q,ge1,ge2,ge3,ge4);
__asm(" mov    r0, %0": : "r"(apsr));
__asm(" msr    apsr_nzcvqg, r0");
```

<프로그램 설명>

1.1에서 작성한 내용과 같습니다.

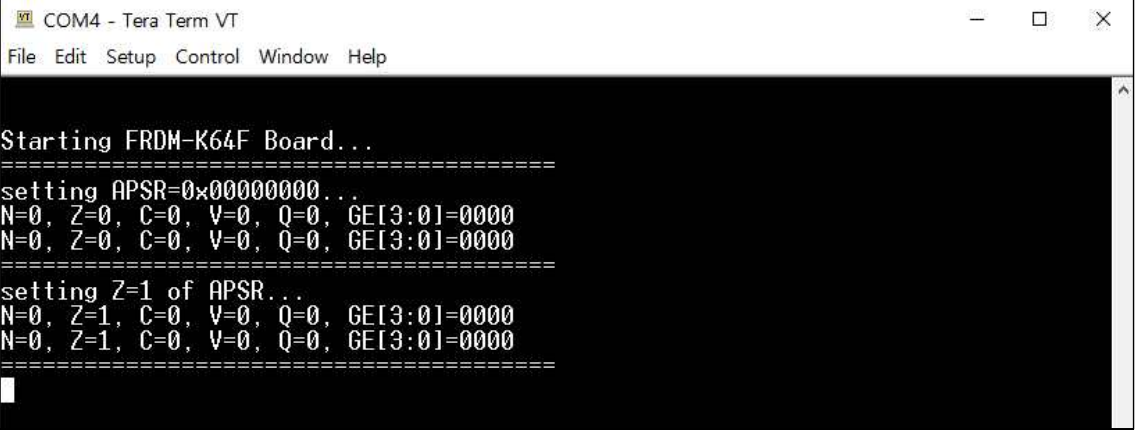
main 함수 내에서의 작성 :

```
__asm("mov r1, #0");
__asm("adds r2, r1, #0");
```

<프로그램 설명>

Z bit를 1로 만들기 위해 계산 결과가 0이 되도록 한다. 따라서 $0+0=0$ 을 수행하는 과정을 나타냈다. inline assembly를 사용해 상수 0을 r1에 넣고, $r2=r1+0$ 을 통해 최종적으로 $0=0+0$ 을 구현하였다. 작은 숫자를 이용하는 것이므로 mov를 사용하였다. 또한 add뒤에 붙은 s는 status register에 반영되도록 붙인 suffix이다.

3.2



```
COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
=====
setting Z=1 of APSR...
N=0, Z=1, C=0, V=0, Q=0, GE[3:0]=0000
N=0, Z=1, C=0, V=0, Q=0, GE[3:0]=0000
=====
```

4. APSR register의 bit C set

4.1

apsr_print() 함수 내에서의 작성 :

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
```

```
for(int cnt=0; cnt<9; cnt++)
```

```
{
```

```
    switch(cnt)
```

```
    {
```

```
        case 0:
```

```
            n=apsr&(1<<31)?1:0;
```

```
            break;
```

```
        case 1:
```

```
            z=apsr&(1<<30)?1:0;
```

```
            break;
```

```
        case 2:
```

```
            c=apsr&(1<<29)?1:0;
```

```
            break;
```

```
        case 3:
```

```
            v=apsr&(1<<28)?1:0;
```

```
            break;
```

```
        case 4:
```

```
            q=apsr&(1<<27)?1:0;
```

```
            break;
```

```
        case 5:
```

```
            ge1=apsr&(1<<19)?1:0;
```

```
            break;
```

```
        case 6:
```

```
            ge2=apsr&(1<<18)?1:0;
```

```
            break;
```

```
        case 7:
```

```
            ge3=apsr&(1<<17)?1:0;
```

```
            break;
```

```
        case 8:
```

```
            ge4=apsr&(1<<16)?1:0;
```

```
            break;
```

```
        default:
```

```
            break;
```

```
    }
```

```
}
```

```
printf("N=%d,      Z=%d,      C=%d,      V=%d,      Q=%d,      GE[3:0]=%d%d%d%d\n",
```

```
n,z,c,v,q,ge1,ge2,ge3,ge4);
__asm(" mov    r0, %0" : "r"(apsr));
__asm(" msr    apsr_nzcvqg, r0");
```

<프로그램 설명>

1.1에서 작성한 내용과 같습니다.

main 함수 내에서의 작성 :

```
__asm("ldr r1, =#0xffffffff");
__asm("ldr r2, =#0x00100000");
__asm("adds r3, r1, r2");
```

<프로그램 설명>

C bit를 1로 만들기 위해 계산 시 캐리가 발생하도록 한다. 두 숫자를 더할 때 MSB+1 bit에 1이 생성되어야 하므로 0xffffffff와 0x00100000의 덧셈이 실행되게 하였다. 큰 숫자를 이용하는 것이므로 ldr을 사용하여 각 숫자를 r1, r2에 대입하였고 adds를 통해 두 숫자를 더해 r3에 저장하였다. 이때 add뒤에 붙은 s는 status register에 반영되도록 붙인 suffix이다.

4.2



```
COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
=====
setting C=1 of APSR...
N=0, Z=0, C=1, V=0, Q=0, GE[3:0]=0000
N=0, Z=0, C=1, V=0, Q=0, GE[3:0]=0000
=====
```


5. APSR register의 bit V set

5.1

apsr_print() 함수 내에서의 작성 :

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
for(int cnt=0; cnt<9; cnt++)
{
    switch(cnt)
    {
        case 0:
            n=apsr&(1<<31)?1:0;
            break;
        case 1:
            z=apsr&(1<<30)?1:0;
            break;
        case 2:
            c=apsr&(1<<29)?1:0;
            break;
        case 3:
            v=apsr&(1<<28)?1:0;
            break;
        case 4:
            q=apsr&(1<<27)?1:0;
            break;
        case 5:
            ge1=apsr&(1<<19)?1:0;
            break;
        case 6:
            ge2=apsr&(1<<18)?1:0;
            break;
        case 7:
            ge3=apsr&(1<<17)?1:0;
            break;
        case 8:
            ge4=apsr&(1<<16)?1:0;
            break;
        default:
            break;
    }
}
printf("N=%d,      Z=%d,      C=%d,      V=%d,      Q=%d,      GE[3:0]=%d%d%d%d\n",
```

```
n,z,c,v,q,ge1,ge2,ge3,ge4);
__asm(" mov    r0, %0" : "r"(apsr));
__asm(" msr    apsr_nzcvqg, r0");
```

<프로그램 설명>

1.1에서 작성한 내용과 같습니다.

main 함수 내에서의 작성 :

```
__asm("ldr r1, =#0x40000000");
__asm("ldr r2, =#0x40000000");
__asm("adds r3, r1, r2");
```

<프로그램 설명>

V bit를 1로 만들기 위해 계산 시 overflow가 발생하도록 한다. overflow가 발생하는 경우는 2가지로 나눌 수 있다. 첫 번째로 MSB가 0인 두 숫자를 더한 결과 값의 MSB가 1일 때 발생한다. 두 번째로 MSB가 1인 두 숫자를 더한 결과 값의 MSB가 0일 때 발생한다. 문제의 조건에서 N bit와 V bit 이외의 bit는 변경되면 안 된다고 했는데 두 번째 경우는 C bit가 1로 바뀌므로 첫 번째 경우를 채택하였다. 따라서 두 숫자는 0x40000000으로 설정하였다. 큰 숫자를 이용하는 것이므로 ldr을 사용하여 각 숫자를 r1, r2에 대입하였고 adds를 통해 두 숫자를 더해 r3에 저장하였다. 이때 add뒤에 붙은 s는 status register에 반영되도록 붙인 suffix이다.

5.2

```
COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
=====
setting V=1 of APSR...
N=1, Z=0, C=0, V=1, Q=0, GE[3:0]=0000
N=1, Z=0, C=0, V=1, Q=0, GE[3:0]=0000
=====
```

6. APSR register의 bit Q set

6.1

apsr_print() 함수 내에서의 작성 :

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
for(int cnt=0; cnt<9; cnt++)
{
    switch(cnt)
    {
        case 0:
            n=apsr&(1<<31)?1:0;
            break;
        case 1:
            z=apsr&(1<<30)?1:0;
            break;
        case 2:
            c=apsr&(1<<29)?1:0;
            break;
        case 3:
            v=apsr&(1<<28)?1:0;
            break;
        case 4:
            q=apsr&(1<<27)?1:0;
            break;
        case 5:
            ge1=apsr&(1<<19)?1:0;
            break;
        case 6:
            ge2=apsr&(1<<18)?1:0;
            break;
        case 7:
            ge3=apsr&(1<<17)?1:0;
            break;
        case 8:
            ge4=apsr&(1<<16)?1:0;
            break;
        default:
            break;
    }
}

printf("N=%d,      Z=%d,      C=%d,      V=%d,      Q=%d,      GE[3:0]=%d%d%d%d\n",
```

```
n,z,c,v,q,ge1,ge2,ge3,ge4);
__asm(" mov    r0, %0" : "r"(apsr));
__asm(" msr    apsr_nzcvqg, r0");
```

<프로그램 설명>

1.1에서 작성한 내용과 같습니다.

main 함수 내에서의 작성 :

```
__asm("ldr r1, =#0x1f1f1f1f");
__asm("SSAT r1, #16, r1");
```

<프로그램 설명>

Q bit을 1로 만들기 위해 16진수 0x1f1f1f1f를 saturation 하는 프로그램을 작성하였다.
r1의 값인 0x1f1f1f1f가 2번째 줄의 SSAT 명령어를 거치면 16bit로 표현할 수 있는 가장 가까운 값으로 변경되게 된다. 이때 r1의 값이 변경되기 때문에 Q bit이 1이 되고 캐리가 발생하여 C bit 또한 1이 된다. 강의 자료에서는 [SSAT R1, R1, #16]로 작성하라고 되어 있었지만 이대로 작성 시 컴파일이 되지 않았다. 따로 인터넷에 SSAT syntax를 검색하여 올바른 작성법을 알아냈고 [SSAT R1, #16, R1]의 방식으로 작성하여 문제를 해결하였다.

10.126 SSAT

Signed Saturate to any bit position, with optional shift before saturating.

Syntax

```
SSAT{cond} Rd, #sat, Rm[, shift]
```

where:

cond

is an optional condition code.

Rd

is the destination register.

sat

specifies the bit position to saturate to, in the range 1 to 32.

Rm

is the register containing the operand.

6.2

```
COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GEI3:01=0000
N=0, Z=0, C=0, V=0, Q=0, GEI3:01=0000
=====
setting Q=1 of APSR...
N=0, Z=0, C=1, V=0, Q=1, GEI3:01=0000
N=0, Z=0, C=1, V=0, Q=1, GEI3:01=0000
=====
```

7. APSR register의 bit GE set

7.1

apsr_print() 함수 내에서의 작성 :

```
int n,z,c,v,q,ge1,ge2,ge3,ge4;
for(int cnt=0; cnt<9; cnt++)
{
    switch(cnt)
    {
        case 0:
            n=apsr&(1<<31)?1:0;
            break;
        case 1:
            z=apsr&(1<<30)?1:0;
            break;
        case 2:
            c=apsr&(1<<29)?1:0;
            break;
        case 3:
            v=apsr&(1<<28)?1:0;
            break;
        case 4:
            q=apsr&(1<<27)?1:0;
            break;
        case 5:
            ge1=apsr&(1<<19)?1:0;
            break;
        case 6:
            ge2=apsr&(1<<18)?1:0;
            break;
        case 7:
            ge3=apsr&(1<<17)?1:0;
            break;
        case 8:
            ge4=apsr&(1<<16)?1:0;
            break;
        default:
            break;
    }
}

printf("N=%d,      Z=%d,      C=%d,      V=%d,      Q=%d,      GE[3:0]=%d%d%d%d\n",
```

```
n,z,c,v,q,ge1,ge2,ge3,ge4);
__asm(" mov    r0, %0" : "r"(apsr));
__asm(" msr     apsr_nzcvqg, r0");
```

<프로그램 설명>

1.1에서 작성한 내용과 같습니다.

main 함수 내에서의 작성 :

```
__asm("ldr r1, =#0x00010001");
__asm("ldr r2, =#0x01000100");
__asm("SSUB8 r3, r1, r2");
```

<프로그램 설명>

GE[3:0]중 GE[2]와 GE[0]의 bit을 1로 바꾸기 위한 프로그램이다. 수행 결과 값이 0이상인 경우 GE bit값이 1으로 변경되고 그렇지 않으면 0으로 유지되므로 GE[3]과 GE[1]은 음수가 되도록하고 GE[2]와 GE[0]은 0 또는 양수가 되도록 값을 설정한다.

레지스터 r1에 0x00010001을 대입하고 레지스터 r2에 0x01000100을 대입한 뒤 8bit씩 4개의 영역으로 나누어 각각 빼기 계산을 하고 r3에 값을 저장한다. r1과 r2에 값을 대입한 근거는 다음과 같다.

GE[3]과 GE[1]의 경우, 00000000에서 00000001을 뺀으므로 음수가 나와 비트 값이 0이 되고, GE[2]와 GE[0]의 경우, 00000001에서 00000000을 뺀으므로 양수가 나와 비트값이 1로 변경된다. 이와 같은 과정을 통해 GE[3:0]의 비트값을 0101로 변경할 수 있다.

7.2

```
COM4 - Tera Term VT
File Edit Setup Control Window Help

Starting FRDM-K64F Board...
=====
setting APSR=0x00000000...
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
N=0, Z=0, C=0, V=0, Q=0, GE[3:0]=0000
=====
setting GE[3:0]=0101 of APSR...
N=0, Z=0, C=1, V=0, Q=0, GE[3:0]=0101
N=0, Z=0, C=1, V=0, Q=0, GE[3:0]=0101
=====
█
```

끝.