

# *Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 전문가 과정*

<ARM\_Architecture>  
2018.04.30 - 44 일차

강사 - Innova Lee(이상훈)  
[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

학생 - 안상재  
[sangjae2015@naver.com](mailto:sangjae2015@naver.com)

# <소스코드 분석>

## 1. add 명령어

```
#include <stdio.h>

int main(void)
{
    register unsigned int r0 asm("r0"); // r0 을 레지스터로 정의
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");

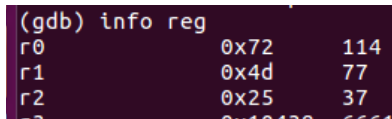
    r1 = 77;
    r2 = 37;

    asm volatile("add r0,r1,r2"); // r0 = r1+r2

    printf("r0 = %d\n", r0);
    return 0;
}
```

### 1-1. 결과 분석

- 디버깅으로 레지스터 내부 값을 확인. r0 는 결과값.



```
(gdb) info reg
r0          0x72      114
r1          0x4d      77
r2          0x25      37
r3          0x10428   66648
```

## 2. biceq 명령어

- biceq 명령어는 & ~ 을 의미함.

```
#include <stdio.h>

void show_reg(unsigned int reg) // 십진수를 이진수로 바꾸어서 출력함
{
    int i;

    for(i=31;i>=0;)
        printf("%d", (reg >> i--)&1);
    printf("\n");
}

int main(void)
{
    register unsigned int r0 asm("r0");
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");
    register unsigned int r3 asm("r3");
    register unsigned int r4 asm("r4");
    register unsigned int r5 asm("r5");

    r0 = 7;
    r1 = 7;

    if(r0 == r1)
    {
        r3 = 42;
    }
}
```

```
asm volatile("biceq r2, r3, #7"); /* r2 = r3 & ~7 (r3 을 8 단위로 정렬함)
                                   r0 가 r1 과 같으면 eq 로 이동함 */
}

show_reg(r2);
return 0;
}
```

## 2-1. 결과 분석

```
josephn@gjosephann-ZZ0NH-A5S1BSU:~/code/arm$ qemu-arm-static -g 1234 -L /usr/arm-linux-gnueabi ./a.out  
0000000000000000000000000101000
```

### 3. cmp 명령어

```
#include <stdio.h>

int main(void)
{
    register unsigned int r0 asm("r0") = 0;
    register unsigned int r1 asm("r1") = 0;
    register unsigned int r2 asm("r2") = 0;
    register unsigned int r3 asm("r3") = 0;
    register unsigned int r4 asm("r4") = 0;
    register unsigned int r5 asm("r5") = 0;

    asm volatile("cmp r0,r1");
    asm volatile("mov r2,#5");
    asm volatile("cmp r0,r2");

    return 0;
}
```

### 3-1. 결과 분석

- 초기 cpsr 값 : 0x60000010

```
cpsr                                0x60000010
(gdb) n
```

- mov r2, #5 와 cmp r0, r2 를 실행하고 나서 cpsr 값 : 0x80000010  
r0 - r2 = -5 가 나오므로 negative 가 set 됨.

```
pc          0x1042c   0x1042c
cpsr       0x80000010
(qdb)
```

#### 4. eors 명령어

- eors r1, r3, r0 => r3 과 r0 를 xor 취함

```
#include <stdio.h>

void show_reg(unsigned int reg)
{
    int i;

    for(i=31;i>=0;)
        printf("%d", (reg >> i--)&1);
    printf("\n");
}

int main(void)
{
    register unsigned int r0 asm("r0") = 0;
    register unsigned int r1 asm("r1") = 0;
    register unsigned int r2 asm("r2") = 0;
    register unsigned int r3 asm("r3") = 0;
    register unsigned int r4 asm("r4") = 0;
    register unsigned int r5 asm("r5") = 0;

    if(r0 == r1)
    {
        r0 = 10;
        r3 = 5;
        asm volatile("eors r1,r3,r0"); // xor
    }

    show_reg(r1);
    return 0;
}
```

##### 4-1. 결과 분석

r1 = 0x0000000f

#### 5. mvneq 명령어

```
#include <stdio.h>

int main(void)
{
    register unsigned int r0 asm("r0") = 0;
    register unsigned int r1 asm("r1") = 0;
    register unsigned int r2 asm("r2") = 0;
    register unsigned int r3 asm("r3") = 0;
    register unsigned int r4 asm("r4") = 0;
    register unsigned int r5 asm("r5") = 0;

    asm volatile("cmp r0,r1");
    asm volatile("mvneq r1, #0"); // mvn : 0xFFFFFFFF 랑 0 이랑 XOR 시킴(eq 지우면 언제나 실행)

    printf("r1 = 0x%x\n", r1);

    return 0;
}
```

## 6. orr 명령어

- or 연산과 동일

```
#include <stdio.h>

void show_reg(unsigned int reg)
{
    int i;

    for(i=31;i>=0;)
        printf("%d", (reg >> i--) & 1);
    printf("\n");
}

int main(void)
{
    register unsigned int r0 asm("r0");
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");
    register unsigned int r3 asm("r3");
    register unsigned int r4 asm("r4");
    register unsigned int r5 asm("r5");

    r5 = 3;

    if(r0 == r1)
    {
        r3 = 44;
        asm volatile("orr r2,r3,r5"); // r2 = r3 | r5
    }

    show_reg(r2);
    return 0;
}
```

### 6-1. 결과 분석

→ r2 = 0x0000002f

## 7. rsble 명령어

- rsb 는 뒤에서 거꾸로 빼기 연산을 함.

```
#include <stdio.h>

int main(void)
{
    register unsigned int r0 asm("r0");
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");
    register unsigned int r3 asm("r3");
    register unsigned int r4 asm("r4");
    register unsigned int r5 asm("r5");

    r1 = 77;
    r2 = 37;
    r3 = 34;
    r5 = 3;

    if(r2 <= r1)
        asm volatile("rsble r4,r5,#5"); /* r4 = 5 - r5 , if 문이 less than 조건을 충족하면 들어옴 */
    printf("r4 = %d\n", r4);
    return 0;
}
```

```
}
```

### 7-1. 결과 분석

- r4 = 2

### 8. subgt 명령어

```
#include <stdio.h>

int main(void)
{
    register unsigned int r0 asm("r0");
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");
    register unsigned int r3 asm("r3");

    r1 = 77;
    r2 = 37;
    r3 = 34;

    if(r1 > r2)
        asm volatile("subgt r3,r3,#1"); // r3 = r3-1 , if 문안이 greather than 조건을 충족하면 들어옴

    printf("r3 = %d\n", r3);
    return 0;
}
```

### 8-1. 결과분석

- r3 = 33

### 9. tsteq 명령어

```
#include <stdio.h>

void show_reg(unsigned int reg)
{
    int i;

    for(i=31;i>=0;)
        printf("%d", (reg >> i--)&1);
    printf("\n");
}

int main(void)
{
    register unsigned int r0 asm("r0") = 0;
    register unsigned int r1 asm("r1") = 0;
    register unsigned int r2 asm("r2") = 0;
    register unsigned int r3 asm("r3") = 0;
    register unsigned int r4 asm("r4") = 0;
    register unsigned int r5 asm("r5") = 0;

    asm volatile("cmp r0, r1");
    asm volatile("mov r2, #3");
    asm volatile("tsteq r2, #5"); // eq 는 cmp 가 동작, tst 는 and

    show_reg(r2);
    return 0;
}
```

### 9-1. 결과 분석

- tst 는 실제 레지스터를 변화시키는 것이 아니라 CPSR 값을 변화 시킴. "tsteq r2, #5" 를 수행하고 나면 결과 값이 0 이 아니므로 cpsr 의 Z 비트가 0 으로 바뀜.

```
23      asm volatile("tsteq r2, #5");
(gdb) n
25      show_reg(r2);
(gdb) info reg
r0      0x0      0
r1      0x0      0
r2      0x3      3
r3      0x0      0
r4      0x0      0
r5      0x0      0
r6      0x10340  66368
r7      0x0      0
r8      0x0      0
r9      0x0      0
r10     0xf67fe000 -159391744
r11     0xf6ffeee4 -150999324
r12     0xf6ffef60 -150999200
sp      0xf6ffeed8 0xf6ffeed8
lr      0xf6684d14 -160936684
pc      0x104f8 0x104f8 <main+44>
cpsr    0x20000010 536870928
(gdb) n
```

## 10. subgt 명령어

```
#include <stdio.h>

int main(void)
{
    register unsigned int r0 asm("r0");
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");
    register unsigned int r3 asm("r3");

    r1 = 77;
    r2 = 37;
    r3 = 34;

    if(r1 > r2)
        asm volatile("subgt r3,r3,#1"); // r3 = r3-1 , gt 는 if 문 안에 greater than 조건을 만족하면 들어옴

    printf("r3 = %d\n", r3);
    return 0;
}
```

### 10-1. 결과 분석

- r3 = 33