

**TI DSP, MCU 및 Xilinx Zynq  
FPGA  
프로그래밍 전문가 과정**

강사 – Innova Lee(이상훈)

[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

학생 – 문한나

[mhn97@naver.com](mailto:mhn97@naver.com)

예제 1)

```
#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");

    asm volatile("mov r0,#0xff,8");
    printf("r0 = 0x%x\n",r0);

    return 0;

}
```

```
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ arm-linux-gnueabi-gcc -g arm1.c
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
r0 = 0xff000000
```

barrel shifter - 버림없이 쉬프트하라  
arm은 4 바이트(32 비트) 단위로 움직인다.

0x 00 00 00 ff 여기서 8 비트를 로테이션 한다.  
0x ff 00 00 00 이된다.

## Barrel Shifter

LSL - Logical Shift Left

상수 시프트의 허용범위 : 0 ~ 31

LSR - Logical Shift Right

상수 시프트의 허용범위 : 1 ~ 32

ASR - Arithmetic Shift Right

상수 시프트의 허용범위 : 1 ~ 32

ROR - Rotate Right

상수 시프트의 허용범위 : 1 ~ 31

RRX - Rotate Right Extended

예제 2)

```
#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){

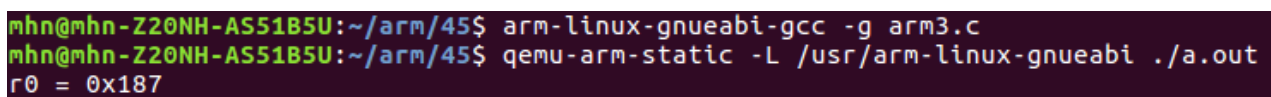
    register unsigned int r0 asm("r0");
    register unsigned int r1 asm("r1");
    register unsigned int r2 asm("r2");
    register unsigned int r3 asm("r3");
    register unsigned int r4 asm("r4");
    register unsigned int r5 asm("r5");

    asm volatile("mov r1,#7");
    asm volatile("mov r2,#3");
    asm volatile("add r0,r1,r2, lsl #7");

    printf("r0 = 0x%x\n",r0);

    return 0;

}
```



```
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ arm-linux-gnueabi-gcc -g arm3.c
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
r0 = 0x187
```

lsl = logical shift left

r2 = 0x 0 0 1 1 ->7비트 쉬프트

0x 0 0 0 1 1 0 0 0 0 0 0 0 0 => 125+128 = 384  
384 + 7 = 391

예제 3)

```
#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;
```

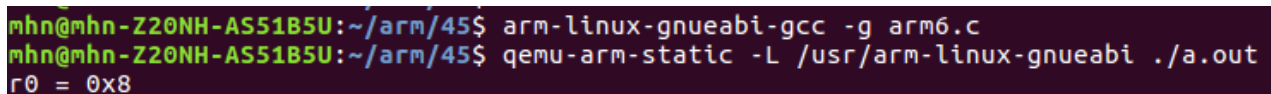
```
    asm volatile("mov r1, #32");
    asm volatile("add r0, r1, asr #2");
```

```
    printf("r0 = 0x%x\n",r0);
```

```
    return 0;
```

```
}
```

ASR - Arithmetic Shift Right



```
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ arm-linux-gnueabi-gcc -g arm6.c
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
r0 = 0x8
```

r1 = 0x 0 0 1 0 0 0 0 0 → 2비트 쉬프트

0x 0 0 0 0 1 0 0 0 => 8

예제 4)

```
#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("mov r1,#32");
asm volatile("add r0,r1,asr #2");
asm volatile("mrs r0,cpsr");
```

```
show_reg(r0);
```

```
return 0;
```

}

```
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out  
0110000000000000000000000000010000
```

mr\_s? PSR의 내용을 범용 레지스터로 이동

이 예제에서는 cpsr 레지스터를 r0 레지스터에 넣음

예제 5)

```
#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("mov r2,#3");
    asm volatile("mov r3,#7");
    asm volatile("mov r4,#33");
    asm volatile("mla r1,r2,r3,r4");

    printf("r1 = %d\n", r1);

    return 0;

}
```

```
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ arm-linux-gnueabi-gcc -g arm9.c
mhn@mhn-Z20NH-AS51B5U:~/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
r1 = 54
```

MUL{S}{cond} {Rd}, Rn, Rm → Rn의 값과 Rm의 값을 곱하고 결과의 최하위 32 비트를 Rd에 배치

MLA{S}{cond} Rd, Rn, Rm, Ra → Rn의 값과 Rm의 값을 곱하고 Ra의 값을 더한 다음 결과의 최하위 32 비트를 Rd에 배치

MLS{cond} Rd, Rn, Rm, Ra → Rn의 값과 Rm의 값을 곱하고 Ra의 값에서 결과를 뺀 다음 최종 결과의 최하위 32 비트를 Rd에 배치

예제 6)

```
#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("mov r2, #0x44, 8");
    asm volatile("mov r3, #0x200");
    asm volatile("umull r0,r1,r2,r3");

    printf("r1r0 = 0x%x%.8x\n", r1, r0);

    return 0;

}
```

```
00X3L:~/my_proj/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
r1r0 = 0x8800000000
```

곱한 후 상위비트를 r1에, 하위비트를 r0에 저장

```
r2 = 0x 44 00 00 00
r3 = 0x          2 00
-----
      0x 88 00 00 00 00
```

예제 7)

```
#include <stdio.h>

char test[] = "HelloARM";

int main(void){

    register unsigned int r0 asm("r0")=0;
    register char *r1 asm("r1")=NULL;
    register unsigned int *r2 asm("r2")=NULL;
    register unsigned int r3 asm("r3")=0;
    register unsigned int r4 asm("r4")=0;
    register unsigned int r5 asm("r5")=0;

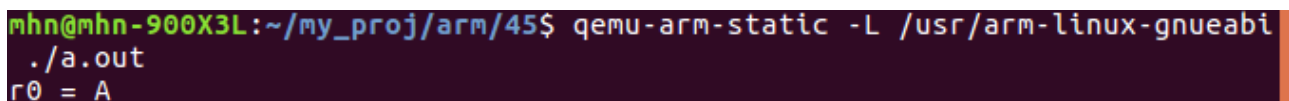
    r1 = test;

    asm volatile("ldreqb r0, [r1, #0x5]");

    printf("r0 = %c\n", r0);

    return 0;

}
```



```
mhn@mhn-900X3L:~/my_proj/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi
./a.out
r0 = A
```

ldr → 메모리에서 레지스터로

b → 바이트 단위로 이동

5 바이트 이동하여 r0 에 저장



예제 8)

```
#include <stdio.h>

char test[] = "HelloARM";

int main(void){

    register unsigned int r0 asm("r0")=0;
    register char *r1 asm("r1")=NULL;
    register unsigned int *r2 asm("r2")=NULL;
    register unsigned int r3 asm("r3")=0;
    register unsigned int r4 asm("r4")=0;
    register unsigned int r5 asm("r5")=0;

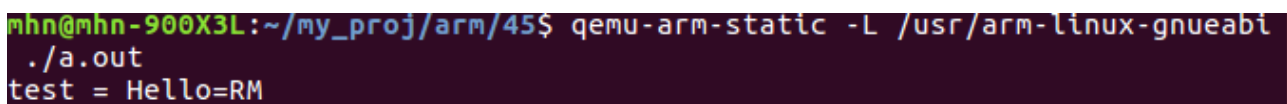
    r1 = &test[5]; //r1 = test;

    asm volatile("mov r0, #61");
    asm volatile("strb r0, [r1]"); //strb r0, [r1, #5]

    printf("test = %s\n", test);

    return 0;

}
```



```
mhn@mhn-900X3L:~/my_proj/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi
./a.out
test = Hello=RM
```

str → 레지스터에서 메모리로

61 → 아스키코드로 =

예제 9)

```
#include <stdio.h>

unsigned int arr[5] = {1, 2, 3, 4, 5};

int main(void){

    register unsigned int r0 asm("r0")=0;
    register unsigned int *r1 asm("r1")=NULL;
    register unsigned int *r2 asm("r2")=NULL;
    register unsigned int r3 asm("r3")=0;
    register unsigned int r4 asm("r4")=0;
    register unsigned int r5 asm("r5")=0;

    r1 = arr;

    asm volatile("mov r2, #0x4");
    asm volatile("ldr r0, [r1], r2");

    printf("r0 = %u, r1 = %u\n", r0, *r1);

    return 0;

}
```

```
mhn@mhn-900X3L:~/my_proj/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
r0 = 1, r1 = 2
```

r2는 r1으로, r1은 r0으로 저장

예제 10)

```
#include <stdio.h>
```

```
int main(void){
```

```
    int i;
```

```
    unsigned int test_arr[7] = {0};
```

```
    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;
```

```
    register unsigned int r6 asm("r6")=0;
```

```
    r0 = test_arr;
```

```
    asm volatile("mov r1, #0x3\n"
```

```
                "mov r2, r1, lsl #2\n"
```

```
                "mov r4, #0x2\n"
```

```
                "add r3, r1, r2, lsl r4\n"
```

```
                "stmia r0!, {r1, r2, r3}\n"
```

```
                "str r4, [r0]\n"
```

```
                "mov r5, #128\n"
```

```
                "mov r6, r5, lsr #3\n"
```

```
                "stmia r0, {r4, r5, r6}\n"
```

```
                "sub r0, r0, #12\n"
```

```
                "ldmia r0, {r4, r5, r6}");
```

```
    for(i=0; i<7; i++)
```

```
        printf("test_arr[%d] = %d\n", i, test_arr[i]);
```

```
    printf("r4 = %u, r5 = %u, r6 = %u\n", r4, r5, r6);
```

```
    return 0;
```

```
}
```

```
mhn@mhn-900X3L:~/my_proj/arm/45$ qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out
test_arr[0] = 3
test_arr[1] = 12
test_arr[2] = 51
test_arr[3] = 2
test_arr[4] = 128
test_arr[5] = 16
test_arr[6] = 0
r4 = 3, r5 = 12, r6 = 51
```

```
r1 = 3
r2 = 12
r3 = 51
r4 = 2
```

```
"stmia r0!, {r1, r2, r3}\n"
```

```
arr[0] = 3
arr[1] = 12
arr[2] = 51
arr[3] = 0 → !는 연산한 위치까지 포인터를 옮김
arr[4] = 0
arr[5] = 0
arr[6] = 0
```

```
"str r4, [r0]\n"
```

```
arr[0] = 3
arr[1] = 12
arr[2] = 51
arr[3] = 2 → r4의 값이 들어감
arr[4] = 0
arr[5] = 0
arr[6] = 0
```

```
r1 = 3
r2 = 12
r3 = 51
r4 = 2
r5 = 128
r6 = 16
```

```
"stmia r0, {r4, r5, r6}\n"
```

```
arr[0] = 3  
arr[1] = 12  
arr[2] = 51  
arr[3] = 2 → 여기부터 다시 채워짐  
arr[4] = 128  
arr[5] = 16  
arr[6] = 0
```

```
"sub r0, r0, #12\n"
```

```
arr[0] = 3 → 포인터가 옮겨감  
arr[1] = 12  
arr[2] = 51  
arr[3] = 2  
arr[4] = 128  
arr[5] = 16  
arr[6] = 0
```

```
"ldmia r0, {r4, r5, r6}");
```

```
r1 = 3  
r2 = 12  
r3 = 51  
r4 = 3  
r5 = 12  
r6 = 51
```

메모리에서 다시 레지스터로 옮긴다.

포인터가 옮겨졌으므로 다시 3, 12, 51이 레지스터의 r4, r5, r6에 저장된다.