TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – GJ (박현우)
uc820@naver.com

목차

1) ARM ASM 예제

(1) add, mov

(2) mul, mla, umull, umlal

(3) ldr , ldreqb

(4) lsl

(5) cpsr

(6) asr

(7) stmia

(8) strb

2) ARM ASM 분석

(1) c코드 register 동작과정

(2) ARM calling convention

1. ARM ASM – add, mov

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```c
#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 r0, #0xff, 8");   // 0x0000 0000 -> 0xff00 0000
    asm volatile("mov r1, #0xf");   // r1 = 0x 0000 000f
    asm volatile("add r2, r1, r0");  // r0 + r1 = r2

    printf("r2 = 0x%x\n", r2);

    return 0;
}
~
~
~
16:31 [모두]                                  [+] ~/arm_asm/asm2/add2.c\
-- 끼워넣기 --
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r2 = 0xff00000f
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```c
#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;

    /* 8 bit 앞으로 rotation   00 00 00 ff  --> ff 00 00 00*/
    asm volatile("mov r0, #0xff, 8");// 버림 없이 shiftg해라 rotation

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

    return 0;
}
~
~
~
~
1:1 [모두]                                    ~/arm_asm/asm2/mov.c\
"mov.c" 18L, 462C
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 0xff000000
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – mul, mla

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
 9      printf("\n");
10 }
11
12 int main(void){
13
14      register unsigned int r0 asm("r0");
15      register unsigned int r1 asm("r1");
16      register unsigned int r2 asm("r2");
17      register unsigned int r3 asm("r3");
18      register unsigned int r4 asm("r4");
19      register unsigned int r5 asm("r5");
20
21
22      asm volatile("mov r2, #3");  // r2 = 3
23      asm volatile("mov r3, #7");   // r3 = 7
24      asm volatile("mul r1, r2, r3");  //  r1 = 3 * 7
25
26      printf("r1 = %d\n", r1);  // r1 = 21
27
28      return 0;
29
30 }
27:1 [바닥]                                  ~/arm_asm/asm2/mul.c\
"mul.c" 30L, 564C 저장 했습니다
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r1 = 21
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
10 }
11
12 int main(void){
13
14      register unsigned int r0 asm("r0");
15      register unsigned int r1 asm("r1");
16      register unsigned int r2 asm("r2");
17      register unsigned int r3 asm("r3");
18      register unsigned int r4 asm("r4");
19      register unsigned int r5 asm("r5");
20
21
22      asm volatile("mov r2, #3");
23      asm volatile("mov r3, #7");
24      asm volatile("mov r4, #33");
25      asm volatile("mla r1, r2, r3, r4");  // mul(r2,r3) + add(r4)
26
27      printf("r1 = %d\n", r1);
28
29      return 0;
30
31 }
27:30 [바닥]                                  [+] ~/arm_asm/asm2/mla.c\
-- 끼워넣기 --
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g mla.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r1 = 54
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – umull, umlal

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
11
12 int main(void){
13
14     register unsigned int r0 asm("r0");
15     register unsigned int r1 asm("r1");
16     register unsigned int r2 asm("r2");
17     register unsigned int r3 asm("r3");
18     register unsigned int r4 asm("r4");
19     register unsigned int r5 asm("r5");
20
21
22     asm volatile("mov r2, #0x44, 4"); // 4bit rotation , r2 = 4000 0004
23     asm volatile("mov r3, #0x200");  // r3 0x200
24     asm volatile("umull r0, r1, r2, r3"); // r2 * r3
25
26     //상위bit r1, 하위 bit r0
27
28     printf("r1 r0 = 0x%x %08x", r1, r0);
29
30     return 0;
31
32 }
32:1 [바닥]                                  ~/arm_asm/asm2/umull.c\
"umull.c" 32L, 635C 저장 했습니다
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r1 r0 = 0x80 00000800hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
12 int main(void){
13
14     register unsigned int r0 asm("r0");
15     register unsigned int r1 asm("r1");
16     register unsigned int r2 asm("r2");
17     register unsigned int r3 asm("r3");
18     register unsigned int r4 asm("r4");
19     register unsigned int r5 asm("r5");
20
21
22     asm volatile("mov r0, #0xf");  // r0 - 0x 0000 000f
23     asm volatile("mov r1, #0x1");  // r1 - 0x 0000 0001
24     asm volatile("mov r2, #0x44, 8"); // r2 - 0x 4400 0000
25     asm volatile("mov r3, #0x200");  //  r3 - 0x 0000 0200
26     asm volatile("umlal r0, r1, r2, r3");
27
28     /* r1 =  r2 * r3  + r1 ,  r0 =  r0*/
29     printf("r1 r0 = 0x%x %08x", r1, r0); // 0x 89 0000000f
30
31     return 0;
32
33 }
31:13 [바닥]                                  [+] ~/arm_asm/asm2/umlal.c\
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g umlal
.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r1 r0 = 0x89 0000000fhyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – ldr

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
15
16      unsigned int arr[5] = {1, 2, 3, 4, 5};
17
18      register unsigned int r0 asm("r0") = 0;
19      register unsigned int *r1 asm("r1") = NULL;
20      register unsigned int *r2 asm("r2") = NULL;
21      register unsigned int r3 asm("r3") = 0;
22      register unsigned int r4 asm("r4") = 0;
23      register unsigned int r5 asm("r5") = 0;
24
25      r1 = arr;
26
27      // load = memory에 있는 정보 불러오기
28
29      asm volatile("mov r2, #0x8");
30      /*byte 이동 , 주소값 시작 0, 4, 8*/
31      asm volatile("ldr r0, [r1,r2]");
32
33      printf("r0 = %u\n", r0);  // arr[2]
34      return 0;
35
36  }
36:1 [바닥]                                  [+] ~/arm_asm/asm2/ldr.c\
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g ldr.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 3
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
12 int main(void){
13
14      unsigned int arr[5] = {1, 2, 3, 4, 5};
15      char test[] = "HelloARM";
16
17      register unsigned int r0 asm("r0") = 0;
18      register char *r1 asm("r1") = NULL;
19      register unsigned int *r2 asm("r2") = NULL;
20      register unsigned int r3 asm("r3") = 0;
21      register unsigned int r4 asm("r4") = 0;
22      register unsigned int r5 asm("r5") = 0;
23
24      r1 = test;
25
26      asm volatile("mov r2, #0x5");
27      asm volatile("ldr r0,[r1,r2]!");  // r2까지 이동하고 그 이후 값으로 셋팅
28
29      printf("test = %s, r1 = %s\n", test, r1);
30
31      return 0;
32
33 }
27:66 [바닥]                                 [+] ~/arm_asm/asm2/ldr2.c\
-- 끼워넣기 --
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g ldr2.
c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
test = HelloARM, r1 = ARM
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – ldr , ldreqb

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
12     printf("\n");
13 }
14
15 int main(void){
16
17     register unsigned int r0 asm("r0") = 0;
18     register unsigned int *r1 asm("r1") = NULL;
19     register unsigned int *r2 asm("r2") = NULL;
20     register unsigned int r3 asm("r3") = 0;
21     register unsigned int r4 asm("r4") = 0;
22     register unsigned int r5 asm("r5") = 0;
23
24     r1 = arr;
25
26     asm volatile("mov r2, #0x4");
27     asm volatile("ldr r0, [r1], r2"); // r1 -> r0,  r2 -> r1 *따로 따로임
28     /* r0 = r1[0],   시작 주소 r1 = r1[1]*/
29     printf("r0 = %u, r1 = %u\n", r0, *r1); //  1, 2
30
31     return 0;
32
33 }
33:1 [바닥]                                  [+] ~/arm_asm/asm2/ldr3.c\
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g ldr3.
c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 1, r1 = 2
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
10         printf("%d", (reg >> i--) & 1);
11     }
12     printf("\n");
13 }
14
15 int main(void){
16
17     register unsigned int r0 asm("r0") = 0;
18     register char *r1 asm("r1") = NULL;
19     register unsigned int *r2 asm("r2") = NULL;
20     register unsigned int r3 asm("r3") = 0;
21     register unsigned int r4 asm("r4") = 0;
22     register unsigned int r5 asm("r5") = 0;
23
24     r1 = test;
25
26     asm volatile("ldreqb r0, [r1,#0x5]");  // load equal byte
27
28     printf("r0 = %c\n", r0);
29     return 0;
30
31 }
31:2 [바닥]                                  ~/arm_asm/asm2/ldreqb.c\
-- 끼워넣기 --
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
yunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = A
yunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – lsl

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
1 #include<stdio.h>
2
3 int main(void){
4
5     register unsigned int r0 asm("r0") = 0;
6     register unsigned int r1 asm("r1") = 0;
7     register unsigned int r2 asm("r2") = 0;
8     register unsigned int r3 asm("r3") = 0;
9     register unsigned int r4 asm("r4") = 0;
10    register unsigned int r5 asm("r5") = 0;
11
12    asm volatile("mov r1, #7");
13    asm volatile("mov r2, #3");
14
15    // 7  + r2 = 3 *  logical shift left 7  == 2^7
16    asm volatile("add r0, r1, r2, lsl #7");
17
18    printf("r0 = 0x%x\n", r0);
19
20    return 0;
21 }
~
18:27 [모두]                                  ~/arm_asm/asm2/lsl.c
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g lsl.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 0x187
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
1 #include<stdio.h>
2
3 int main(void){
4
5     register unsigned int r0 asm("r0") = 0;
6     register unsigned int r1 asm("r1") = 0;
7     register unsigned int r2 asm("r2") = 0;
8     register unsigned int r3 asm("r3") = 0;
9     register unsigned int r4 asm("r4") = 0;
10    register unsigned int r5 asm("r5") = 0;
11
12    asm volatile("mov r1, #7");
13    asm volatile("mov r2, #3");
14    asm volatile("mov r3, #2");
15    asm volatile("add r0, r1, r2, lsl r3");
16    // r0 =  7 + 3 * 2^2   = 19
17
18    printf("r0 = 0x%x\n", r0);
19
20    return 0;
21 }
~
16:31 [모두]                                  ~/arm_asm/asm2/lsl2.c
"lsl2.c" 21L, 485C
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g lsl2.
c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 0x13
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – lsl ,cpsr

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
1 #include<stdio.h>
2
3 int main(void){
4
5     register unsigned int r0 asm("r0") = 0;
6     register unsigned int r1 asm("r1") = 0;
7     register unsigned int r2 asm("r2") = 0;
8     register unsigned int r3 asm("r3") = 0;
9     register unsigned int r4 asm("r4") = 0;
10    register unsigned int r5 asm("r5") = 0;
11
12    asm volatile("mov r1, #2");
13    asm volatile("add r0, r1, r1, lsl #2");
14    // r0 = 2 + 2 * 4
15    printf("r0 = 0x%x\n", r0);
16
17    return 0;
18 }
~
~
~
~
14:21 [모두]                                  ~/arm_asm/asm2/lsl3.c\
"lsl3.c" 18L, 415C 저장 했습니다
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g lsl3.
c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 0xa
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
11
12 int main(void){
13
14     register unsigned int r0 asm("r0") = 0;
15     register unsigned int r1 asm("r1") = 0;
16     register unsigned int r2 asm("r2") = 0;
17     register unsigned int r3 asm("r3") = 0;
18     register unsigned int r4 asm("r4") = 0;
19     register unsigned int r5 asm("r5") = 0;
20
21     asm volatile("mov r1, #32");
22     asm volatile("add r0, r1, asr #2");
23     asm volatile("mrs r0, cpsr");
24     /*
25      *mrs ---  cpsr레지스터를 특정 레지스터에 집어넣는다.(intel pushf 역할)
26      *eflags 레지스터랑 같은 역할을 하는 걸 뭘게 가져올 수 있음.
27      */
28     show_reg(r0);
29 //  printf("r0 = 0x%x\n", r0);
30
31     return 0;
32 }
21:4 [바닥]                                   [+] ~/arm_asm/asm2/mrs.c\
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g mrs.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
00000000000000000000000000001000
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – asr, stmia

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
#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;

    // 오른쪽으로 쉬프트
    asm volatile("mov r1, #32");   // 0x0000 0020 -->  10 0000 >> 2
    asm volatile("add r0, r1, asr #2");  // r0 = 1000

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

    return 0;
}
~
~
~
16:30 [모두]                                  ~/arm_asm/asm2/asr.c\
"asr.c" 19L, 474C 저장 했습니다
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r2 = 0xff00000f
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g asr.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
r0 = 0x8
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
    register unsigned int r4 asm("r4") = 0;
    register unsigned int r5 asm("r5") = 0;

    r0 = test_arr;

    asm volatile("mov r1, #0x3"); // r1 =3
    asm volatile("mov r2, r1, lsl #2"); // r2 = 12
    asm volatile("mov r4, #0x2");  // r4 = 2
    asm volatile("add r3, r1, r2, lsl r4");  // r3 = 48 + 3
    asm volatile("stmia r0!, {r1, r2, r3}"); // 3, 12, 51

    /*
       store(stack에 집어넣음)  muliple increment after
       (stack 증가 후 값 넣기)
     */

    for(i = 0; i<5; i++)
        printf("test_Arr[%d] = %d\n", i,test_arr[i]);

    return 0;

}
35:7 [바닥]                                   ~/arm_asm/asm2/stmia.c\
"stmia.c" 42L, 895C 저장 했습니다
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g stmia
.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
test_Arr[0] = 3
test_Arr[1] = 12
test_Arr[2] = 51
test_Arr[3] = 0
test_Arr[4] = 0
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – stmia

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
23
24      r0 = test_arr;
25
26      asm volatile("mov r1, #0x3");
27      asm volatile("mov r2, r1, lsl #2");
28      asm volatile("mov r4, #0x2");
29      asm volatile("add r3, r1, r2, lsl r4");
30      asm volatile("stmia r0!, {r1, r2, r3}");
31      // r0 = {3, 12 , 51}
32      asm volatile("str r4, [r0]"); // r0 시작이 4번째 , r0 = {3, 12,51, 2}
33
34      for(i = 0; i<5; i++)
35          printf("test_Arr[%d] = %d\n", i,test_arr[i]);
36
37      return 0;
38
39 }
~
~
~
~
~
32:73 [바닥]                                    ~/arm_asm/asm2/stmia2.c\
"stmia2.c" 39L, 834C 저장 했습니다
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g stmia
2.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
test_Arr[0] = 3
test_Arr[1] = 12
test_Arr[2] = 51
test_Arr[3] = 2
test_Arr[4] = 0
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
17      register unsigned int *r0 asm("r0") = 0;
18      register unsigned int r1 asm("r1") = 0;
19      register unsigned int r2 asm("r2") = 0;
20      register unsigned int r3 asm("r3") = 0;
21      register unsigned int r4 asm("r4") = 0;
22      register unsigned int r5 asm("r5") = 0;
23
24      r0 = test_arr;
25
26      asm volatile("mov r1, #0x3\n"
27              "mov r2, r1, lsl #2\n"
28              "mov r4, #0x2\n"
29              "add r3, r1, r2, lsl r4\n"
30              "stmia r0!, {r1, r2, r3}\n"
31              "str r4, [r0]");
32
33      for(i = 0; i<5; i++)
34          printf("test_Arr[%d] = %d\n", i,test_arr[i]);
35
36      return 0;
37
38 }
38:1 [바닥]                                    [+] ~/arm_asm/asm2/stmia3.c\
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g stmia
3.c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
test_Arr[0] = 3
test_Arr[1] = 12
test_Arr[2] = 51
test_Arr[3] = 2
test_Arr[4] = 0
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

1. ARM ASM – strb

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
    printf("\n");
}

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];

    asm volatile("mov r0, #61");  // r0 = '='
    asm volatile("strb r0,[r1] ");  // register to memory (str) r0 -> r1

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

    return 0;

}
33:1 [바닥]                                    ~/arm_asm/asm2/strb.c\
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ arm-linux-gnueabi-gcc -g strb.
c
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -L /usr/arm-li
nux-gnueabi ./a.out
test = Hello=RM
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$
```

2. ARM ASM 분석 – c코드 register 동작과정

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```c
#include<stdio.h>

int my_func(int num){
    return num * 2;
}

int main(void){

    int res, num = 2;
    res = my_func(num);
    printf("res = %d\n", res);
    return 0;
}
```

1:1 [모두] ~/arm_asm/asm2/arm_func.c\
"arm_func.c" 13L, 159C

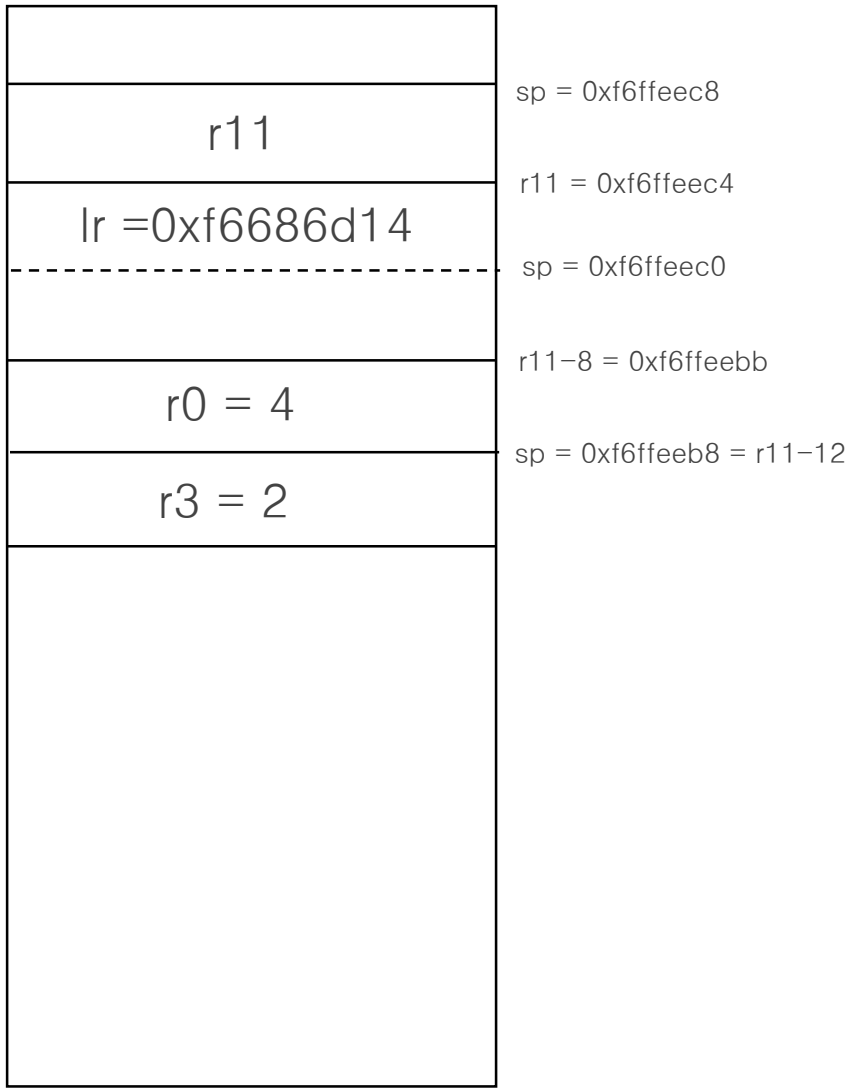
hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
hyunwoopark@hyunwoopark-P65-P67SG:~/arm_asm/asm2$ qemu-arm-static -g 1234 -L /usr/arm-linux-gnueabi ./a.out
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/arm_asm/asm2

```
Do you need "set solib-search-path" or "set sysroot"?

Breakpoint 2, main () at arm_func.c:7
7       int main(void){
(gdb) disas
Dump of assembler code for function main:
=> 0x00010460 <+0>:     push   {r11, lr}
   0x00010464 <+4>:     add    r11, sp, #4
   0x00010468 <+8>:     sub    sp, sp, #8
   0x0001046c <+12>:    mov    r3, #2
   0x00010470 <+16>:    str    r3, [r11, #-12]
   0x00010474 <+20>:    ldr    r0, [r11, #-12]
   0x00010478 <+24>:    bl     0x10438 <my_func>
   0x0001047c <+28>:    str    r0, [r11, #-8]
   0x00010480 <+32>:    ldr    r1, [r11, #-8]
   0x00010484 <+36>:    ldr    r0, [pc, #16]   ; 0x1049c <main+60>
   0x00010488 <+40>:    bl     0x102e0 <printf@plt>
   0x0001048c <+44>:    mov    r3, #0
   0x00010490 <+48>:    mov    r0, r3
   0x00010494 <+52>:    sub    sp, r11, #4
   0x00010498 <+56>:    pop    {r11, pc}
   0x0001049c <+60>:    andeq  r0, r1, r0, lsl r5
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0xf6fff014       -150999020
r2             0xf6fff01c       -150999012
r3             0x10460  66656
r4             0x104a0  66720
r5             0x0      0
r6             0x10310  66320
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0x0      0
r12            0xf6ffef40       -150999232
sp             0xf6ffeec8       0xf6ffeec8
lr             0xf6686d14       -160928492
pc             0x10460  0x10460 <main>
cpsr           0x60000010       1610612752
```



2. ARM ASM 분석 – calling convention

ARM calling convention

- Register usage:

Registers	Function	Value preserved during call
R0-R3	Arguments / Return values	No
R4-R11	Local variables	Yes
R12 (IP)	Intra-procedure-call scratch reg.	No
R13 (SP)	Stack Pointer	Yes
R14 (LR)	Link register	No
R15 (PC)	Program Counter	No

- If a routine has more than 4 arguments R0-R3 are used for the first 4 arguments and the rest are placed on the stack before the call
- The stack must be of the Full-Descending type
- Local variables can also be stored in R0-R3, R12, and even LR, specially in "leaf" subroutines (no other subroutine call)

arm(Load store architecture)
함수의 리턴 값 r0 ,
r7 = system call
---
r11 = bp
lr = r15 --- 복귀 주소 pc
---
arm에서 함수의 인자는 레지스터로 받는다.
하지만,
인자를 4개 이상 쓰면 스택을 사용하게 되어서
속도가 떨어진다.