TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 윤연성
whatmatters@naver.com

```
#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%x\n", r1, r0);
        return 0;
}
```

umull

0x44 00 00 00
0x   200

곱하기 하면

00x88 00 00 00 00

상위비트가 r1
하위비트 r0

0x88 까지는 나오는데 뒤에 00 00 00 00 은 짤린다

```
 printf("r1r0 = 0x%x %08x\n", r1, r0);    //자리값을 계산
        return 0;
```

이렇게 할경우 에는
r1r0 = 0x88 00000000 정상적으로 나옴

r1r0 = 0x880

```
#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 r0, #0xf");
          asm volatile("mov r1, #0x1");
          asm volatile("mov r2, #0x44, 8");
          asm volatile("mov r3, #0x200");
          asm volatile("umlal r0, r1, r2, r3");

        printf("r1r0 = 0x%x%x\n", r1, r0);
        return 0;
}
```

곱하고 더해서 확장

```
r1r0 = 0x89f
```

```
printf("r1r0 = 0x%x%08x\n", r1, r0);
        return 0;
```

```
r1r0 = 0x890000000f
```

인텔 arm 차이
어셈블리에서볼때
노드스토어아키텍처

b *xxxxxxxxxx

```
0x00010488 <+0>:  push   {r11, lr}
   0x0001048c <+4>:      add    r11, sp, #4
   0x00010490 <+8>:      sub    sp, sp, #32
=> 0x00010494 <+12>:     mov    r3, #2
   0x00010498 <+16>:     str    r3, [r11, #-28]      ; 0xffffffe4
   0x0001049c <+20>:     mov    r3, #3
   0x000104a0 <+24>:     str    r3, [r11, #-24]      ; 0xffffffe8
   0x000104a4 <+28>:     mov    r3, #4
   0x000104a8 <+32>:     str    r3, [r11, #-20]      ; 0xffffffec
   0x000104ac <+36>:     mov    r3, #5
   0x000104b0 <+40>:     str    r3, [r11, #-16]
   0x000104b4 <+44>:     mov    r3, #6
   0x000104b8 <+48>:     str    r3, [r11, #-12]
   0x000104bc <+52>:     ldr    r3, [r11, #-12]
   0x000104c0 <+56>:     str    r3, [sp]
   0x000104c4 <+60>:     ldr    r3, [r11, #-16]
   0x000104c8 <+64>:     ldr    r2, [r11, #-20]      ; 0xffffffec
   0x000104cc <+68>:     ldr    r1, [r11, #-24]      ; 0xffffffe8
   0x000104d0 <+72>:     ldr    r0, [r11, #-28]      ; 0xffffffe4
   0x000104d4 <+76>:     bl     0x10438 <my_func>
   0x000104d8 <+80>:     str    r0, [r11, #-8]
   0x000104dc <+84>:     ldr    r1, [r11, #-8]
```

(함수호출규약)
r7 = system call
특수용도 sp , lr , pc

함수의 리턴값은 항상 r0 에 저장됨

4 개가 넘어가면 스택을 쓴다
(sp 에다 저장)

```
#include <stdio.h>

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

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") = 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, #0x8");
          asm volatile("ldr r0, [r1, r2]");
//         asm volatile("ldr r0, [r1, #0x4]");

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


r0 = 3
```

```
#include <stdio.h>

char test[] ="HelloARM";

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  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 = arr;


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

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

```
test = HelloARM, r1 =ARM
```

```
#include <stdio.h>

char test[] ="HelloARM";

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 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]
//      asm volatile("ldr r0, [r1, #0x4]");
//str 레지스터에서 메모리로 ndr 은 메모리에서 레지스터로
        printf("test = %s\n",test);

        return 0;
}
```

r0 = 1, r1 =2

```
#include <stdio.h>

char test[] ="HelloARM";

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 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("mov r2, #0x5");
           asm volatile("ldr r0, [r1, r2]!");



        printf("test = %s, r1 =%s\n", test,r1 );

        return 0;
}
```

```
#include <stdio.h>

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

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") = 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");
				//[r1] r0 에 따로들어가고 r2 가 r1 에 따로들어감  ( 따로놀음)


	printf("r0 = %u, r1 =%u\n", r0, *r1);				//r0  = 1   r1 = 2

	  return 0;
}
```

```
include <stdio.h>



int main(void)
{
        int i;
        unsigned int test_arr[5] = {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;

      r0 = test_arr;




         asm volatile("mov r1, #0x3");
         asm volatile("mov r2, r1, lsl #2");
         asm volatile("mov r4, #0x2");
       asm volatile("add r3, r1, ,r2, lsl r4");
      asm volatile("stmla r0,{r1, r2, r3}");




        for(i = 0; i<5 i++)

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

      return 0;
}
```

```
#include <stdio.h>



int main(void)
{
        int i;
        unsigned int test_arr[5] = {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;

        r0 = test_arr;



         asm volatile("mov r1, #0x3");
         asm volatile("mov r2, r1, lsl #2");
         asm volatile("mov r4, #0x2");
          asm volatile("add r3, r1, ,r2, lsl r4");
        asm volatile("stmla r0,{r1, r2, r3}");
           asm volatile("str r4, [r0]);


        for(i = 0; i<5 i++)

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

        return 0;
}
```

```
#include <stdio.h>



int main(void)
{
        int i;
        unsigned int test_arr[5] = {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;
}
```

```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;
}
```

```c
#include <stdio.h>

int my_func(int n1 ,int n2, int n3, int n4, int n5 )
	{
		return n1+n2+n3+n4+n5;

	}

int main(void)
{
	int res, n1 = 2, n2 =3, n3 =4, n4 =5, n5=6;
	res = my_func(n1, n2, n3, n4, n5);
	printf("res = %d\n", res);
	return 0;
}
```

```
#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 r0, #0xf");
          asm volatile("mov r1, #0x1");
             asm volatile("mov r2, #0x44, 8");
             asm volatile("mov r3, #0x200");
             asm volatile("umlal r0, r1, r2, r3");

        printf("r1r0 = 0x%x%08x\n", r1, r0);
        return 0;
}
```