TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 - 이상훈
gcccompil3r@gmail.com

학생 – 이우석
colre99@naver.com

[5/1(화) – 45일차]

[ARM_ARCHITECTURE] - ASSEMBLY 2일차

[add_mov3]

```
#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, #7");                  // 상수 7 을 r1 에 넣는다.
        asm volatile("mov r2, #3");                  // 상수 3 을 r2 에 넣는다.
        asm volatile("add r0, r1, r2, lsl #7");      // lsl #7 = 상수 2^7 을 r2 인 3 과 곱한다.
                                                     //  그런후, r1 인 7 을 더한다. 그 값을 r0 에 넣는다.
        printf("r0 = 0x%x\n", r0);         // 출력형태는 %x = 16 진수이다.
                                           // 계산값: 3 (=r2) x 128(=2^7) + 7 (=r1) = 391 (10 진수)
        return 0;                          // 391 (10 진수) 를 16 진수로 바꾸면  187.
}                                          // 결국 출력값은   'r0 = 0x187'
```

* ARM 에서 #은 상수를 의미한다.
* lsl 은 logical shift left = 왼쪽으로 민다. (왼쪽으로 Shift)

[add_mov4]

```
#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, #7");                     // 상수 7 을 r1 에 대입.
        asm volatile("mov r2, #3");                     // 상수 3 을 r2 에 대입.
        asm volatile("mov r3, #2");                     // 상수 2 를 r3 에 대입.
        asm volatile("add r0, r1, r2, lsl r3");     // lsl r3 (= 2^2) 를 r2 (= 3) 와 곱한다. = 3 x 2^2 = 12
                                                    //  12 와 r1 (= 7) 과 더한다. = 7 + 12 = 19 를 r0 에 대입.
        printf("r0 = 0x%x\n", r0);                  // 출력값 형태는  %x(16 진수).  10 진수인 19 를
                                                    // 16 진수로 변환하면 13. 즉, 출력값은  'r0 = 0x13'
        return 0;
}
```

[add_mov5]

```
#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, #2");		// 상수 2 를 r1 에 대입.
	asm volatile("add r0, r1, r1, lsl #2");	// lsl 상수 2^2 (= 4) x r1 (= 2) = 8 + r1 (= 2) = 10
						// 최종값 10 을 r0 에 대입.
	printf("r0 = 0x%x\n", r0);	//출력 형태는 %x (16 진수).  10 을 16 진수로 바꾸면 a.
					//16 진수는 9 이후부터 알파벳이기에  'a'로 출력.
	return 0;			// 결국 최종 출력값은  ' r0 =0xa'
}
```

[add_mov6]

```
#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");		//상수 32 를 r1 에 대입.
	asm volatile("add r0, r1, asr #2");	// asr #2 는 2^2 (= 4) 를 r1(= 32)에 나누기를 해준다.
						// 그럼, 32 / 4 = 8 (=10 진수)이 나온다. 이 값을 r0 에 대입.

	printf("r0 = 0x%x\n", r0);		//출력형태는 %x(16 진수). 최종 출력값은 'r0 = 0x8'

	return 0;
}
```

* asr 은 arithmetic shift right = 오른쪽으로 민다. (오른쪽으로 shift)

[add_mul]

```
#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, #3");		// 상수 3 을 r2 에 대입.
	asm volatile("mov r3, #7");		// 상수 7 을 r3 에 대입.
	asm volatile("mul r1, r2, r3");		// mul 은 r2 (= 3) x r3 (= 7) 을 한 후, 곱한 값을 r1 에 대입.
						//즉, r2 (= 3) x r3 (= 7) => 3 x 7 = 21.  이 값을 r1 에 대입.
	printf("r1 = %d\n", r1);		// 출력형태는 %d (=10 진수). 최종 출력값은  'r1 = 21'

	return 0;				// mul 의 문법형태는  MUL  Rd, Rm, Rs  →  Rm x Rs => Rd
}
```

[add_mla]

```
#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, #3");             // 상수 3 을 r2 에 대입.
        asm volatile("mov r3, #7");             // 상수 7 을 r3 에 대입.
        asm volatile("mov r4, #33");            // 상수 33 을  r4 에 대입.
        asm volatile("mla r1, r2, r3, r4");     //mla 는 r2, r3 를 곱한 후, r4 에 더한다. 그 값은 최종 r1 에 대입.
                                                // 즉, ( r2 (= 3)  x r3 (= 7) ) + r4 (= 33) => (3 x 7) + 33  = 54.
        printf("r1 = %d\n", r1);                //출력형태는 %d (=10 진수). 최종 출력값은  'r1 = 54'
        return 0;                               // mla 의 문법형태는 MLA  Rd, Rm, Rs, Rn   →  (Rm x Rs) + Rn => Rd
}
```