Assembly 2nd

노트북:	SW		
만든 날짜:	2018-05-01 오후 5:09	**수정한 날짜:**	2018-05-02 오전 9:35
작성자:	정상요		
URL:	https://drive.google.com/drive/folders/1qnLSAN1-5ZJc9_1TRyFBEYD2mSWUjl6Z		

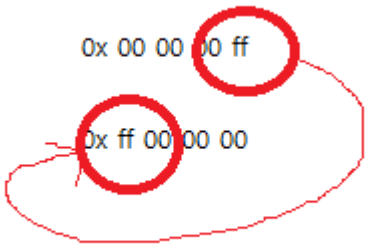
2018. 5. 1 화 – 45회차

과정 : TI, DSP, Xilinx Znq FPGA, MCU 기반의 프로그래밍 전문가 과정

Prof. 이상훈
gcccompil3r@gmail.com
Stu. 정상용
fstopdg@gmail.com

1. Rotation(#이 없는 숫자, #숫자 : 상수)

```
/* Rotation */

#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");

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

	return 0;
}
```



-> mov r0, #0xff, 8 :
0x 00 00 00 ff 에서 8bit만 rotation 시켜서 0x ff 00 00 00 로 만들어준다.

2. Rotation 응용

```
/* Rotation 응용 */

#include <stdio.h>

int main()
{
	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");
	asm volatile ("mov r1, #0xf");
	asm volatile ("add r2, r1, r0");

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

	return 0;
}
```

r0 = 0x ff 00 00 00
r1 = 0x 00 00 00 0f
r2 = r1 + r0 = 0x ff 00 00 0f

3. lsl = Logical Shift Left

```
/* lsl */

#include <stdio.h>

int main()
{
	register unsigned int r0 asm("r0") = 0;
	register unsigned int r1 asm("r1") = 0;
	register unsigned int r2 asm("r2") = 0;

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

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

	return 0;
}
```

mov r1 -> r1 = 7
mov r2 -> r2 = 3
add r0, r1, r2, lsl#7 -> First, r2 << 7 : 11 << 7 : 110000000 = 384(= 3*x^7 = 384)
Second : 7 + 384 = 391
*주의할 점
printf 할 경우, printf 에 레지스터가 사용되므로 두 번째 printf에는 전과 다른 레지스터값이 출력될 가능성이 있다.

4. lsl(lsl #숫자 대신에 lsl '레지스터' 가 오는 경우)

```
#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;

	asm volatile ("mov r1, #7");
	asm volatile ("mov r2, #3");
	asm volatile ("mov r3, #2");
```

```
	asm volatile ("add r0, r1, r2, lsl r3");

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

	return 0;
}
```

5. lsl(2nd paraemter, 3rd parameter 가 같은 경우)

```
#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;

	asm volatile ("mov r1, #2");
	asm volatile ("add r0, r1, r1, lsl #2"); // 첫 번째 r1은 fixed, FPGA는 병렬 계산하므로 두번째 r1에 lsl #2를 해도 첫 번째 r1에 영향을 주지 않는다.

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

	return 0;
}
```

mov r1, #2 -> r1 = 2
add r0, r1, r1, lsl#2 -> 2nd parameter,r1 is fixed!!! Only 3rd parameter,r1 is changed.(FPGA는 병령 계산하므로 두 개의 r1은 서로 영향을 주지 않는다.)

6. asr(Arithmetic Shift Right)

```
#include <stdio.h>

int main(void)
{
	register unsigned int r0 asm("r0") = 0;
	register unsigned int r1 asm("r1") = 0;

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

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

mov r1, #32 -> r1 = 32
add r0, r1, asr #2 -> r0 = r0 + r1, asr #2 (r1, asr #2 : 100000 << 10 : 1000 = 8 = 32/4)
r0 = 0 + 8 = 8
*참고
add r0, r1, r2 -> r0 = r1 + r2
add r0, r1 -> r0 = r0 + r1

7. mrs

```
#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;

        asm volatile ("mov r1, #32");
        asm volatile ("add r0, r1, asr #2"); //add r0, r1 -> r0 = r0 + r1
        asm volatile ("mrs r0, cpsr");

        show_reg(r0);
        printf("%d\n", r0);

        return 0;
}
```

mrs r0, cpsr -> cpsr 레지스터를 특정한 레지스터(지금의 경우, r0)에 넣어준다. : pushf
interrupt시 껐다 켰다를 하므로 mrs를 자주 보게 된다.

pushf(stack에 넣어줬다가 pop해서 fadsflkjasdlf)

eflags _ 9bit : interrupt를 받을지 안받을지 정한다.

8. mul(multiply

```
#include <stdio.h>

int main()
{
        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;

        asm volatile ("mov r2, #3");
        asm volatile ("mov r3, #7");
        asm volatile ("mul r1, r2, r3");

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

        return 0;
}
```

mul r1

Q.
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0204j/Cjacbgca.html
ASR, LSL, LSR, ROR and RRX
ASR : Arithmetic Shift Right
LST : Logical Shift Right
차이??