Xilinx Zynq FPGA, TI DSP, MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)
gcccompil3r@gmail.com
학생 - 이유성
dbtjd1102@naver.com

목차

1.

movr.c movr2.c

lsl.c lsl2.c

asr.c asr2.c(+ msr.)

mul.c

mla.c

umull.c

umlal.c

ldr.c ldr2.c ldr3.c ldr(메모리 ->레지스터)

strb.c strb2.c strb3.c str(레지스터 ->메모리)

stmia.c stmia2.c stmia3.c

ldmia.c

2.

func1.c func2.c arm asm 분석.

movr.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");

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

	return 0;
}


r0 = 0xff000000
```

:r0 = 0xff000000

8비트단위로 오른쪽 로테이션

2^n 단위로 로테이션 가능한 것 같다.

```
#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");
	asm volatile("mov r1,#0xf");
	asm volatile("add r2,r1,r0");
	printf("r2 = 0x%x\n",r2);

	return 0;
}
```

:r2 = 0xff00000ff

8비트 로테이션 후 더함.

lsl.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 r1,#7");
	asm volatile("mov r2,#3");
	asm volatile("add r0 , r1 ,r2, lsl #7");  //  3<<7  3x128 +7   r2, lsl #7 해서 r2의 값이
바뀌었다고 r2,r2, lsl #7 경우 앞의 r2의 값이 바뀌지는 않음 ->병렬

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

	return 0;
}
```

r2를 왼쪽으로 7만큼 쉬프트 하고 r1과 더한 후 r0에 저장.

lsl2.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 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;
}
```

r2을 r3만큼 왼쪽 쉬프트하고 r1과 더한 후 r0에 저장

asr.c

```
#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");  //오른쪽 쉬프트 연산

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

	return 0;
}
```

r1을 오른쪽으로 2번 쉬프트하고 r0에 저장

asr2.c (+ msr.)

```
#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");  //msr  cpsr레지스터 값을 특정 레지스터에 저장함. pushf

	show_reg(r0);

	return 0;
}
```

msr cpsr레지스터 값을 특정 레지스터에 저장함. pushf

인터럽트를 끄고 킬 때 많이 사용

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

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

	return 0;
}
```

r2 * r3 값을 r1에 저장

mla.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 r2,#3");
	asm volatile("mov r3,#7");
	asm volatile("mov r4, #33");
	asm volatile("mla r1,r2,r3,r4");  // dsp: 곱셈+덧셈 동시 기능   r2 *r3 + r4  <<이게 1clock에
끝남.

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

	return 0;
}
```

r2 * r3 + r4 값을 r1에 저장.

umull.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 r2,#0x44,8");
	asm volatile("mov r3,#0x200");
	asm volatile("umull r0,r1,r2,r3");// r2*r3 0x 88 00 00 00 00  상위비트 r1에 88 하위비트 r0에
00 00 00 00
				//그런데 r0에 0이 나옴 00 00 00 00 이 나와야하는데 맞게 볼 수 있고
틀리게 볼 수 있음
```

```
                                // %08x 으로 해결 가능.
        printf("r1r0 =0x%x%x\n",r1,r0 );

        return 0;
}
```

r2 * r3 연산후 0x 88 / 00 00 00 00   슬래쉬 기준으로 하위비트는 r0에 저장 r1에는 상위비트가 저장.

umlal.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,#0xf");
        asm volatile("mov r1,#0x1");
        asm volatile("mov r2,#0x44,8");
        asm volatile("mov r3,#0x200");
        asm volatile("umlal r0,r1,r2,r3"); // mla 곱하고 더해,  r1=88+1 ,r0=00 00 00 00+f


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

        return 0;
}
```

mla = 곱하고 더하는 것을 1clock에 처리

r2 * r3 연산후 하위비트를 r0의 값과 더한 후 r0에 저장

r2 * r3 연산후 상위비트를 r1의 값과 더한 후 r1에 저장

ldr.c

```
#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,#0x8");
	asm volatile("ldr r0,[r1,r2]"); //load 메모리에있던 정보를 레지스터로 가져옴. r1주소값에서 r2byte만큼 이동



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

	return 0;
}
```

r0=3

r1에서 8byte이동한 값을 r0에 저장

ldr2.c

```
#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("ldr r0,[r1,#0x4]");  //4byte만큼 이동.

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

	return 0;
```

```
}
```

r0=2

r1에서 4byte 이동한 값을 r0에 저장

ldr3.c

```
#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]");  //eq 위에 뭔가 조건이 있어야 좋음. b는 byte 1바이트씩 처리하겠다.

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

	return 0;
}
```

r1에서 5바이트 이동한 정보를 r0에 저장.

stmia.c

```
#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("stmia r0,{r1,r2,r3}"); // store multiple increment after  : stack 증가 후 값을
집어넣겠다.

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

	return 0;
}
```

```
test_arr[0]=3
test_arr[1]=12
test_arr[2]=51
test_arr[3]=0
test_arr[4]=0
```

r0에 차례로 r1,r2,r3정보를 저장 .

stmia2.c

```
#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("stmia r0!,{r1,r2,r3}"); //! 이동한 값을 갱신
        asm volatile("str r4,[r0]");


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

        return 0;
}
```

r0에 차례로 r1,r2,r3 정보를 저장하고 그 다음 주소를 !로 r0을 셋팅

셋팅 된 r0에 r4의 값을 저장.

stmia3.c

```
#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\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]");


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

        return 0;
}
```

test_arr[0]=3
test_arr[1]=12
test_arr[2]=51
test_arr[3]=2
test_arr[4]=0

stmia2.c 를 좀 더 간편하게 적을 수 있음.

strb.c

```
#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]   //ldr의 반대 레지스터에서 메모리로

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

	return 0;
}
```

test=Hello=RM

61값을 r1에 저장

strb2.c

```
#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]!"); // ! A에 r1을 픽스 시켜라, 그전꺼는 갱신

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

	return 0;
}
```

test=HelloARM,r1=ARM

r1에서 5byte 이동 시켜서 !로 픽스 시킴.

r1에서 5byte이동 시킨 정보를 r0에 저장

strb3.c

```
#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");  //r0에 [r1]  , [r1]에 r2바이트만큼 갱신되면 2를 가르킴.  이 둘은
따로 놈.

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

        return 0;
}
```

r0 =1,r1 = 2

r0에 [r1] , [r1]에 r2바이트만큼 갱신되면 2를 가르킴. 이 둘은 따로 놈.

ldmia.c

```
#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" //r2 = 12
                "mov r4, #0x2\n"            // r4 = 2
                "add r3,r1,r2,lsl r4\n"  // r3 = 51
                "stmia r0!,{r1,r2,r3}\n"// 배열 인덱스 0,1,2에 값넣고 3에 셋팅.

                "str r4,[r0]\n" // 배열 인덱스 3에 2를 넣음.
                "mov r5, #128\n"  //r5 =128
                "mov r6,r5,lsr #3\n" //r6 =16
                "stmia r0,{r4,r5,r6}\n" //배열 인덱스 3부터 2,128,16을 넣음
                "sub r0,r0,#12\n"// 인덱스 3부터 12byte를 빼서 시작주소로.
                "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;
}
```

```
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
```

ldmia . r0값을 차례로 r4,r5,r6에 저장시킴.