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

강사 - Innov (이상훈) gcccompil3r@gmail.c

학생 - 이유성 dbtjd1102@naver.com

Arm architecture

소스코드 분석

```
add.c

#include<stdio.h>

int main(void)
{
       register unsigned int r0 asm("r0");        //레지스터 사용하겠다 .셋팅.
       register unsigned int r1 asm("r1");
       register unsigned int r2 asm("r2");

        r1 = 77;
        r2 = 37;

        asm volatile("add r0,r1,r2");// 더해서 r0 에 넣어라

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

결과 r0 = 114

```
subgt.c

#include<stdio.h>

int main(void)
{
        register unsigned int r0 asm("r0");
        register unsigned int r1 asm("r1");
        register unsigned int r2 asm("r2");
        register unsigned int r3 asm("r3");

        r1 = 77;
        r2 = 37;
        r3 = 34;

        if(r1 > r2)
                asm volatile("subgt r3,r3,#1");
        printf("r3 = %d\n",r3);
        return 0;
}
```

결과 r3 = 33

```
rsble.c


#include<stdio.h>

int main(void)
{
          register unsigned int r0 asm("r0");
           register unsigned int r1 asm("r1");
           register unsigned int r2 asm("r2");
          register unsigned int r3 asm("r3");
           register unsigned int r4 asm("r4");
          register unsigned int r5 asm("r5");

           r1 = 77;
           r2 = 37;
           r3 = 34;
           r5 = 3;

           if(r2 <= r1)
                     asm volatile("rsble r4,r5,#5"); //reverse sub 숫자 5 – r5  원래는 이것의 역방향이 정방향
           printf("r4 = %d\n",r4);
           return 0;
}
```

결과 r4 = 2

```
And.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");
        register unsigned int r1 asm("r1");
        register unsigned int r2 asm("r2");
        register unsigned int r3 asm("r3");
        register unsigned int r4 asm("r4");
        register unsigned int r5 asm("r5");

        r1 = 34;
```

```
	r2 = 37;
	r5 = 3;
	asm volatile("and r0,r1,r2");

	show_reg(r0);
	return 0;
}
```

결과 00000000000000000000000000100000

```
Bic.c

#include<stdio.h>  //bit clear      r0 = r1 같은 경우에만 동작하도록

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;

	r0 = 7;
	r1 = 7;

	if(r0 == r1)
	{
		r3 = 42;
		asm volatile("biceq r2,r3,#7");
	}

	show_reg(r2);
	return 0;
}

// 42 & ~(2^3 -1)
// 42 & ~(7)          42 를 2^3 의 배수로 정렬 ->40
```

결과 00000000000000000000000000101000

```
Orr.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;

	r5 = 3;

	if(r0 == r1)
	{
		r3 = 44;
		asm volatile("orr r2,r3,r5");
	}
	show_reg(r2);
	return 0;
}
```

결과 00000000000000000000000000101111

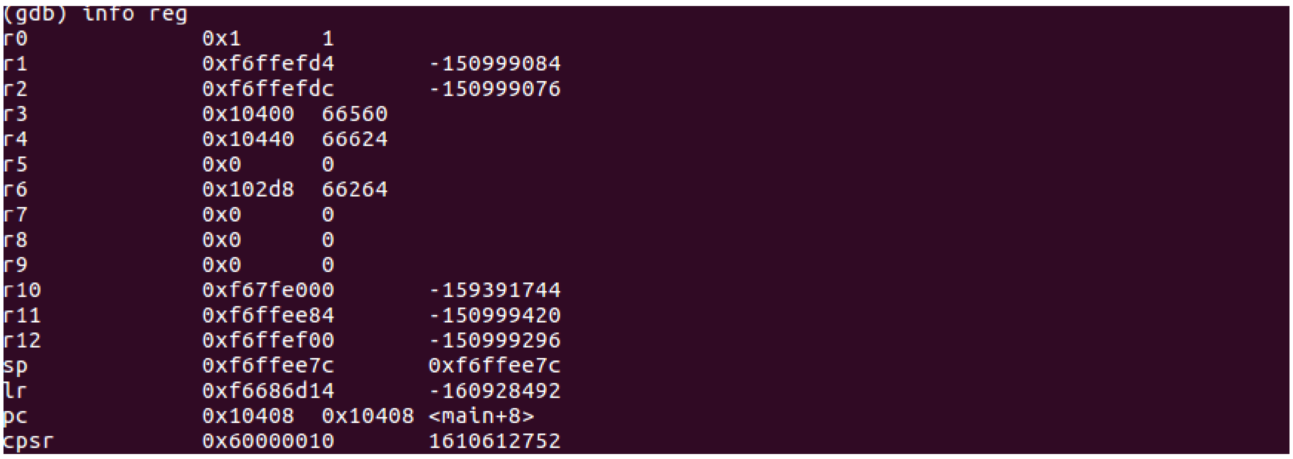
```
Eors.c

#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;

        if(r0 == r1)
        {
                r0 = 10;
                r3 = 5;
                asm volatile("eors r1,r3,r0");
        }
        show_reg(r1);
        return 0;
}
```

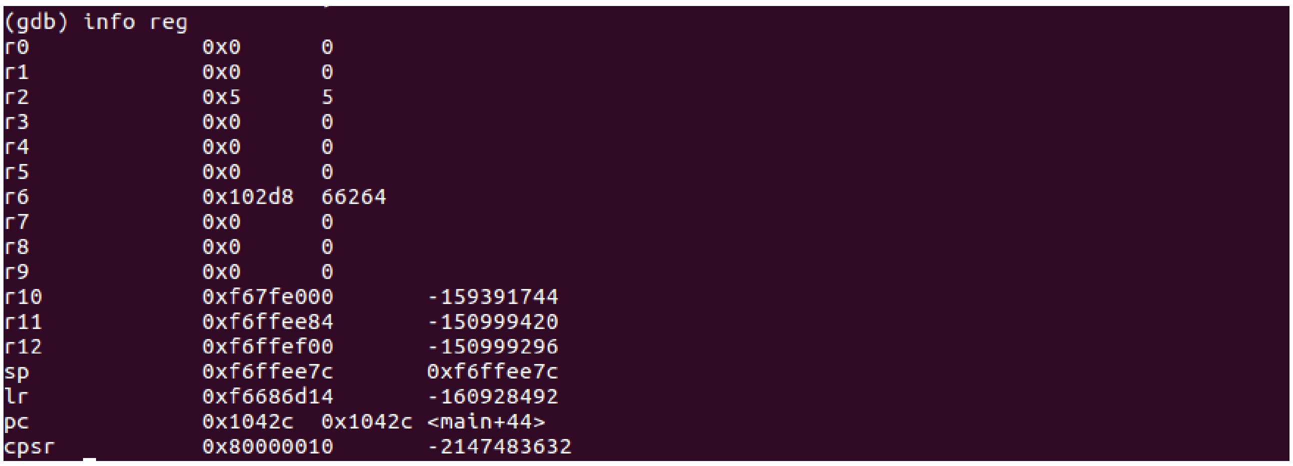
결과 00000000000000000000000000001111

```
Cmp.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("cmp r0,r1");
        asm volatile("mov r2,#5");
        asm volatile("cmp r0,r2");

        return 0;
}
```

→ qemu-arm-static -g 1234 -L /usr/arm-linux-gnueabi ./a.out
→ 터미널 창 하나 더 띄우고 gdb-multiarch
→ file a.out
→ target remote localhost:1234
→ b main
→ c
→ info reg
→ cpsr   0x60000010   확인
→ ni 후 info reg
→ cpsr 0x80000010 확인 N C 1 셋팅.



```
(gdb) info reg
r0             0x1      1
r1             0xf6ffefd4       -150999084
r2             0xf6ffefdc       -150999076
r3             0x10400  66560
r4             0x10440  66624
r5             0x0      0
r6             0x102d8  66264
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee84       -150999420
r12            0xf6ffef00       -150999296
sp             0xf6ffee7c       0xf6ffee7c
lr             0xf6686d14       -160928492
pc             0x10408  0x10408 <main+8>
cpsr           0x60000010       1610612752
```



```
(gdb) info reg
r0             0x0      0
r1             0x0      0
r2             0x5      5
r3             0x0      0
r4             0x0      0
r5             0x0      0
r6             0x102d8  66264
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee84       -150999420
r12            0xf6ffef00       -150999296
sp             0xf6ffee7c       0xf6ffee7c
lr             0xf6686d14       -160928492
pc             0x1042c  0x1042c <main+44>
cpsr           0x80000010       -2147483632
```

Cmp2.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("cmp r0,r1");

        asm volatile("mov r2,#3");
        asm volatile("tsteq r2,#5"); //eq 동작 시키는 애 cmp        0 이 아니기 때문에 zero 꺼짐.

        show_reg(r2);
        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("cmp r0 , r1");
        asm volatile("mvneq r1, #0"); //eq->cmp 에서 r0==r1 같으면 zero->1->동작  /eq 없으면 언제나 동
작  0xffffffff 랑
 xor 시키는게 mvn

        printf("r1 = 0x%x\n",r1);
        show_reg(r2);
        return 0;
}
```