Xilinx Zynq FPGA, TI DSP MCU 기반의

프로그래밍 및 회로 설계 전문가

강사 이상훈

(Innova Lee)

Gcccompil3r@gmail.com

학생 김민호

minking12@naver.com

ARM Add 함수

```
#include<stdio.h>

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

   r1=77;
   r2=37;

   asm volatile("add r0,r1,r2");//r1+r2 -> r0 에 저장

   printf("r0=%d\n",r0);//r0 프린트문.
   return 0;

}
```

ARM sub 함수

```
#include<stdio.h>

int main()
{
   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");//sub 는 뺄셈 , gt 는 greater than r3(34)에서 #1(1)을 빼고 , r3 에
저장

   printf("r3=%d\n",r3);//r3 프린트문.
   return 0;
}
```

ARM rsble 함수

```
#include<stdio.h>
```

```
int main()
{

   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");//le 니까 r1 이 크거나 같다. rsb 는 reverse #5(5)에서 r5(3)을 뺄셈 후에 r4 에 저장

   printf("r4=%d\n",r4);
   return 0;
}
```

ARM and 함수

```
#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()
{

    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");//r1(34)과 r2(37)을 and 연산 -> r0 에 저장.

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

ARM biceq 함수

```
#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()
{

    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");

    r0=7;
    r1=7;

    if(r0==r1)
    {
        r3=42;
```

```
        asm volatile("biceq r2,r3,#7");//eq= equality r3(42)와 #7(7)을 BIC(And Not) 한다.
                                42 & ~ 2^3 -1 -> r2 에 저장
    }

    show_reg(r2);
    return 0;
}
```

ARM cmp 함수

```
#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("cmp r0,r1");//cmp 양수값 비교 명령어 , r0 과 r1 비교 0-0 같으니 zbit 값이 1 이다.(cmp
결과가 0 이 아니면 zbit 는 0 nbit 는 음수일때 1 nzcb ->1248)
    asm volatile("mov r2,#5");//r2 에 5 를 저장
    asm volatile("cmp r0,r2");//cmp r0 r2 양수값 비교 명령어

    return 0;
}
```

ARM eors 함수

```
#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()
{
```

```
    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");

    if(r0==r1)
    {
        r0=10;
        r3=5;
        asm volatile("eors r1,r3,r0");//r3(5)과 r0(10)을 XOR 연산(두 개의 비트가 서로 다르면 "1", 서로 같으면 "0"이 되는 XOR 연산을 수행합니다.)
    }

    show_reg(r1);
    return 0;
}
```

ARM orr 함수

```
#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()
{

    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");//r3 과 r5 를 or 연산(두 개의 비트중 하나라도 "1"이면 "1", 모두
"0"이면 "0"이 나오는 OR 연산을 수행합니다.)
    }

    show_reg(r2);
    return 0;
}
```

ARM mvneq 함수

```
#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()
{

    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");//r0 과 r1 양수 비교
    asm volatile("mvneq r1,#0");//0 을 not 연산

    printf("r1=0x%x\n",r1);

    return 0;
}
```

ARM tst 함수

```
#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()
{

   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");//cmp -> 두 변수?의 뺄셈이 0 이면 zbit 는 1 , 값이 0 이 아니면 zbit 은 0
nbit 는 음수일때 1 c 는 자릿수 올라갈때 대문자로 변한다.(flag 값이 1)
  asm volatile("mov r2,#3");//r2 에 3 을 저장
  asm volatile("tsteq r2,#5");//eq 는 equality 3 이랑 5 랑 and 연산하면 1 이니까 값이 0 이 아니니 zbit
가 0

  return 0;
}
```