TI DSP, MCU, Xilinx Zynq FPGA 기반의 프로그래밍 전문가 과정

<ARM Architecture>
2018.05.02 - 46 일차

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 - 안상재
sangjae2015@naver.com

1. 부트 코드

```
Reset_Handler PROC
            EXPORT Reset_Handler          [WEAK]
     IMPORT SystemInit                  ; SystemInit 함수를 선언함
     IMPORT __main                      ; __main 함수를 선언함

         ;FPU settings                    ; 어셈블리에서는  ';' 가 주석임
         LDR R0, = 0xE000ED88       ; R0 에 0xE000ED88 값을 저장
                                          (0xE000ED88 은 CPACR 레지스터의 주솟값)*/
         LDR R1, [R0]                     ; R0 의 데이터 값을 R1 에 저장
         ORR R1, R1, #(0xF <<20)          ; R1 = R1 | 0x00F0 0000
         STR R1, [R0]                     ; R1 을 R0 에 저장함

         LDR R0, =SystemInit              ; R0 에 SystemInit 값을 저장함
         BLX R0                           ; R0 로 branch 함
         LDR R0, =__main                  ; R0 에 __main 값을 저장함
         BX R0                            ; R0 로 branch 함
         ENDP

NMI_Handler      PROC
            EXPORT NMI_Handler
            B
            ENDP
HardFault_Handler\
                PROC
                EXPORT HardFault_Handler        [WEAK]
                B
```

2. SystemInit 함수

```
void SystemInit(void)
{
        RCC->CR |= (uint32_t)0x00000001;
        RCC->CFGR = 0x00000000;
        RCC->CR &= (uint32_t)0xFEF6FFFF;
        RCC->PLLCFGR = 0x24003010;
        RCC->CR &= (uint32_t)0xFFFBFFFF;
        RCC->CIR = 0x00000000;

#ifdef DATA_IN_ExtSRAM
        SystemInit_ExtMemCtl();
#endif

        SetSysClock();

#ifdef VECT_TAB_SRAM
        SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET;
#else
```

```
        SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET;
#endif
}
```

1) RCC→CR |= (uint32_t)0x00000001
- HSEON (HSE clock enable) 1: HSE oscillator ON

2) RCC→CR &= (uint32_t)0xFEF6FFFF

- PLLON (Main PLL enable)
0 : PLL OFF

- CSSON (Clock security system enable)
0 : Clock security system OFF (Clock detector OFF)

- HSEON (HSE clock enable)
0 : HSE oscillator OFF

3) RCC→CR &= (uint32_t)0xFFFBFFFF

- HSEBYP (HSE clock bypass)
0 : HSE oscillator not bypassed

3. 시스템 콜 어셈블리

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.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;
        register int r7 asm("r7") = 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]\n"
                        "mov r5, #128\n"
                        "mov r6, r5, lsr #3\n"
```

```
                "stmia r0, {r4,r5,r6}\n"
                "sub r0,r0, #12\n"
                "ldmia r0, {r4,r5,r6}\n"
                "swp r6,r3,[r0]");

        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);

        r7 = 2;// 2 번은 sys_fork()

        asm volatile("swi #0" : "=r" (r0) : "r" (r7) : "memory");
        ; "=r" : 레지스터(r0)에 집어넣어라, "memory" => memory barrier 를 집어넣는다(이 명령어 끝날
때까지 instruction scheduling 를 하지마라 r7 의 순서가 바뀔 수도 있어서)
        if(r0 > 0)
                printf("r0 = %p, Parent\n", r0);
        else if(r0 == 0)
                printf("r0 = %p, Child\n", r0);
        else
        {
                perror("fork()");
                exit(-1);
        }

        return 0;
}

/*
r0,r1,r2,r3 : 함수의 인자로 사용하는 레지스터
r0 : 함수의 리턴값
r7 : 시스템 콜
r13,r14,r15 : sp,lr,pc
r11 : 스택의 베이스 포인터
명령어 : 출력 : 입력 : 특수지시어
*/
```