

# HW2

21800471 유준호

1. 다음 (a), (b) (d) 문제는 별개의 program이 아니고, 단계적으로 실행되는 문제이다.

(a) 0x20001010 번지에 1 byte 0x0F 값을 저장시키는 assembly program을 작성하고, 디버거를 이용하여 확인하시오. (5점)

The screenshot displays a debugger interface with three main panels. The 'Registers' panel on the left shows the 'Core' registers, with R15 (PC) highlighted at 0x080001A0. The 'Disassembly' panel on the right shows assembly instructions, with the instruction 'ldrb r1, [r0]' at address 0x080001A0 highlighted in yellow. The 'Memory 1' panel at the bottom shows the memory address 0x20001010 with a value of 0F, indicating that the byte 0x0F has been successfully stored at that location.

Register	Value
R0	0x20001010
R1	0x0000000F
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000600
R14 (LR)	0x080001B1
R15 (PC)	0x080001A0
xPSR	0x01000000

```
0x08000192 0800 DCW 0x0800
0x08000194 01C7 DCW 0x01C7
0x08000196 0800 DCW 0x0800
4: ldr r0, =0x20001010
5:
0x08000198 4803 LDR r0, [pc, #12] ; @0x080001A8
6: ldr r1, =0x0F
7:
0x0800019A F04F010F MOV r1, #0x0F
8: strb r1, [r0]
9:
0x0800019E 7001 STRB r1, [r0, #0x00]
10: ldrb r1, [r0]
->0x080001A0 7801 LDRB r1, [r0, #0x00]
<
```

hw1.s

```
1 AREA |.text|, CODE, READONLY, ALIGN=2
2 EXPORT __main
3 main PROC
4 ldr r0, =0x20001010
5
6 ldr r1, =0x0F
7
8 strb r1, [r0]
9
10 ldrb r1, [r0]
```

Memory 1

Address: 0x20001010

Address	Value
0x20001010	0F 00
0x20001032	00 00
0x20001054	00 00

0x20001010 번지에 0x0F값이 들어가 있음을 확인하였다. Ldr 명령어를 이용하여 0x20001010이라는 주소값을 r0에 넣어준다. 또한 0x0f 값을 r1에 넣어준다. 이 때 16진수 숫자를 주소값으로 안보고 상수로 보는 이유는 상수 앞에 '='를 넣어서 그렇다. strb 명령어를 이용하여 r0에 있는 가장 낮은 주소의 1바이트를 R0에 넣은 주소에 넣어준다.

(b) 0x20001010 번지의 bit 3를 clear시키는 assembly program을 작성하고, 디버거를 이용하여 확인하시오. (bit-band alias 주소를 이용하지 말 것!) (10 점)

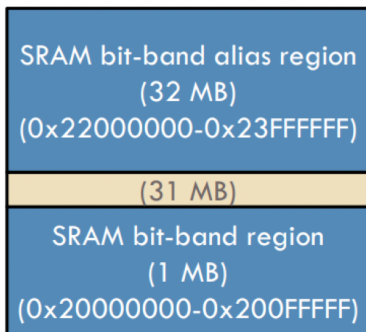
The screenshot displays a debugger interface with three main panels:

- Register Panel:** Shows the state of various registers. R15 (PC) is highlighted with a value of 0x080001AC. Other registers like R0-R14, xPSR, Banked, System, Internal, and FPU are also listed.
- Assembly Panel:** Shows a list of assembly instructions. The instruction at address 0x080001A0, `AND r1, r1, #0xF7`, is highlighted in yellow. Other instructions include MOV, STRB, and LDR.
- Memory Panel:** Shows a memory dump starting at address 0x20001010. The dump displays hexadecimal values and their corresponding ASCII representations. The value at 0x20001010 is 0x07, which corresponds to the ASCII character 'G'.

R1에 있는 0000 1111(2진수)의 값과 1111 0111(2진수)의 값을 and연산하여 bit 3의 값만 clear(=0)해주고 다른 비트는 보존하였다.

(c) 0x20001010 번지의 bit 3의 bit-band alias address는 무엇인가? 5 (점)

$$\text{Bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4)$$



```
C main.c ×
C main.c > main()
1  #include <stdio.h>
2
3  int main(){
4
5      unsigned int bit_band_base = 0x22000000;
6      unsigned int byte_offset = 0x1010;
7      unsigned int bit_number = 3;
8
9      unsigned int Bit_word_addr = bit_band_base + (byte_offset * 32) + (bit_number * 4);
10
11     printf("Bit_word_addr = 0x%x\n", Bit_word_addr);
12
13
14
15     return 0;
16 }
```

문제    출력    디버그 콘솔    터미널

```
● juno@yujunhoui-MacBookPro-2 HW2 % gcc main.c -o main
● juno@yujunhoui-MacBookPro-2 HW2 % ./main
  Bit_word_addr = 0x2202020c
○ juno@yujunhoui-MacBookPro-2 HW2 %
```

0x20001010번지의 Bit 3의 bit-band alias address는 0x2202020c이다.

Bit\_band\_base\_addr은 SRAM인 경우 0x22000000이다. byte\_offset 은 0x1010이고, bit\_number는 3이다.

(d) 0x20001010 번지의 bit 3를 set시키는 assembly program을 작성하고, 디버거를 이용하여 확인하시오. (bit-band alias 주소를 이용하시오.) (5점)

The screenshot displays a debugger interface with three main panels:

- Registers:** Shows the state of various registers. R15 (PC) is highlighted with a value of 0x08001B8. Other registers like R0-R14, xPSR, Banked, System, Internal, and FPU are also listed.
- Disassembly:** Shows the assembly code for a file named `hw1.s`. The code includes:
 

```

1      AREA |.text|, CODE, READONLY, ALIGN=2
2      EXPORT __main
3      __main PROC
4      ldr r0, =0x20001010
5
6      ldr r1, =0x0F
7
8      strb r1, [r0]
9
10     and r1, 0xF7 ; 0000 1111 & 1111 0111 = 0000 0111
11
12     strb r1, [r0]
13
14     ldr r0, =0x2202020c
15     mov r2, #1
16     str r2, [r0]
      
```
- Memory 1:** Shows the memory dump starting at address 0x20001010. The first line shows the value 0x0F at address 0x20001010, which is the target of the bit-band alias operation.

12번째 줄을 지났을 때 0x20001010주소값의 가장 낮은 바이트의 값은 0x0F였다. 그 후 위에서 구한 bit-band alias값을 이용하여 0x20001010번지의 bit 3의 값을 1로 set 하였다. 이를 통해 0x20001010의 값 0x07이 0x0F가 되었다.(0000 0111 -> 0000 1111)

2. 1번의 프로그램을 디버깅하기 위해서 디버거를 동작시켰을 때 (혹은 디버거 동작 중에 reset 버튼을 눌렀을 때)

(a) R13 (SP)의 값은 무엇인가? 그 값은 어떻게 정해진 것인가? (5점)

Registers	
Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000600
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x080001B8
xPSR	0x01000000

R13(SP) = 0x20000600

## Registers

17

### □ R13: SP (stack pointer) (MSP or PSP)

- R13 (SP) means the current SP. The other one can be accessed by special instructions MSR/MRS.

- MSP (Main SP): Default stack pointer used by the OS kernel and exception handler
- PSP (Process SP): Used by user applications

- ✎ On reset, the processor loads the MSP with the value from address 0x00000000.

- The lowest 2 bits of the SP are always 0. (word aligned)

- Stack operation: full-descending stack

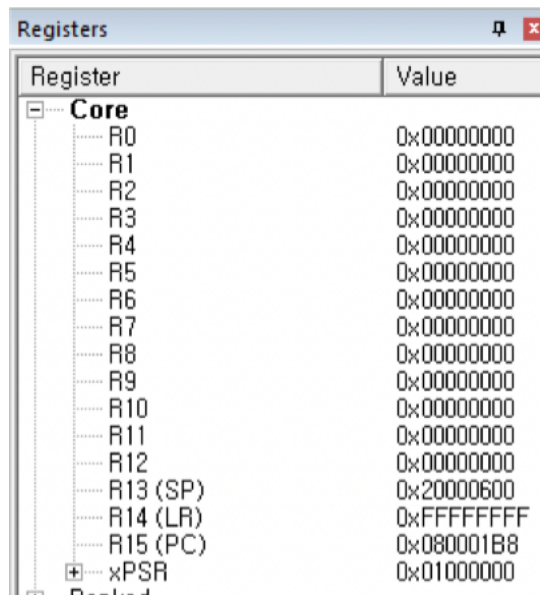
```
subroutine:
    PUSH    {R0-R7, R12, R14}
    ...
    POP     {R0-R7, R12, R14}
    BX      R14
```

만일 2 bits는 0  
주소 받아오는 쪽으로  
32 bits

메모리 0x00000000번지에 저장된 값을 읽어서 그 값을 MSP 레지스터에 로드한다. 즉 사용할 스택의 메모리 주소값을 0x00000000에 저장시켜놓고, 리셋 후에 0x00000000에 저장되어있는 값을 읽어서 MSP에 저장하는 방식이다. 참고로 MSP는 R13(SP) 레지스터에 속하며 이와 함께 PSP가 있다.

(b) R15 (PC)의 값은 무엇인가? 그 값은 어떻게 정해진 것인가?

(10점)



Register	Value
<b>Core</b>	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000600
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x080001B8
xPSR	0x01000000

R15(PC) = 0x080001B8

리셋 후에 PC에는 0x00000004번지에 저장되어있는 값이 로드된다. 이 때 로드된 값은 리셋된 후에 실행될 명령어의 주소값이다.

3. 2번과 보는 바와 같은 결과가 어떻게 초래되는 것인지 (즉 0x00번지와 0x04번지에 특정 값이 주어지도록 code가 작성된 부분을 찾고, 그 부분을 설명하시오. (startup\_stm32f411xe.s 를 참조하여야 한다) (10 점)

```
171 ; Reset handler
172 Reset_Handler    PROC
173                 EXPORT Reset_Handler            [WEAK]
174                 IMPORT SystemInit
175                 IMPORT __main
176
177                 LDR    R0, =SystemInit
178                 BLX    R0
179                 LDR    R0, =__main
180                 BX     R0
181                 ENDP
182
```

외부에 정의된 함수 SystemInit을 IMPORT를 통해 불러온다. 177번줄을 이용하여 SystemInit 함수의 주소값을 R0에 로드한다. 178번 줄에서 R0에 들어있는 주소로 분기한다. SystemInit 함수로 분기하는 것이다. 여기에서 0x00과 0x04의 초기화가 진행된다. 그 후에 \_\_main 함수의 주소값을 불러와서 분기한다. 이렇게 리셋은 끝난다.



4. (a) SYSCFG memory remap register의 현재 값은 무엇인가? (디버거에서 Peripherals -> System viewer -> SYSCFG -> MEMRM 값을 보면 된다.) "stm32f411 reference manual"을 참고하여 SYSCFG memory remap register의 현재 값이 의미하는 것이 무엇인지 설명하시오. (10 점)

SYSCFG	
Property	Value
MEMRM	0
ME...	0x00

MEMRM = 0x00 (2 비트)

### 7.2.1 SYSCFG memory remap register (SYSCFG\_MEMRMP)

This register is used for specific configurations on memory remap:

- Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.
- After reset these bits take the value selected by the BOOT pins. When booting from main Flash memory with BOOT0 pin set to 0 this register takes the value 0x00.

In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT pins)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														MEM_MODE	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MEM\_MODE**: Memory mapping selection

Set and cleared by software. This bit controls the memory internal mapping at address 0x0000 0000. After reset these bits take the value selected by the Boot pins .

00: Main Flash memory mapped at 0x0000 0000

01: System Flash memory mapped at 0x0000 0000

11: Embedded SRAM mapped at 0x0000 0000

Note: Refer to [Section 2.3: Memory map](#) for details about the memory mapping at address 0x0000 0000.

physical remap을 정하는 과정에서 이 두 개의 비트가 사용된다. 매핑 모드에서 CPU는 외부 메모리에 시스템 버스 대신 ICode를 통해 접근한다. 이 두 비트를 통해 CPU는 main flash memory에서 부팅하게 된다.



(b) 메모리 0x00000000 ~ 0x0000000F 영역의 값과 0x08000000 ~ 0x0800000F 영역의 값을 비교하시오. (5 점)

Memory 1															
Address: 0x00000000															
0x00000000: 00 06 00 20 B9 01 00 08 C1 01 00 08 C3 01 00 08															

Memory 1															
Address: 0x08000000															
0x08000000: 00 06 00 20 B9 01 00 08 C1 01 00 08 C3 01 00 08															

두 영역의 값이 같음을 확인하였다.(0x0800 01c3 0800 01c1 0800 01b9 2000 0600)