HW3

1. 다음과 같은 main.c 함수를 갖는 project를 MDK-ARM tool에서 생성한 다음, compile 후에 실 습 kit로 download하시오.

1. 17번, 19번, 22번과 25번 라인의 코드의 동작을 각각 설명하시오. (8점)

17번: AIRCR[10:8]에 주어진 값을 write하게 되며, 17번 코드는 AIRCR[18:8]=5 가 되게 한다. 그 결과는 priority 8bits 중에서 [7:6] bits는 group priority 부분이고, [5:0] bits는 subpriority 부분이 되도록 한다. (그룹 1점, sub 1점 (or 전체 8bit명시 + 우선순위 중 하나만 명시한 경우 정답으로 인정))

19번 라인은: external interrupt 6번 (STM32F411에서는 EXTI Line 0 interrupt)의 우선 순위를 0x40으로 설정하는 코드이다. STM32F411에서는 상위 4 bits만 priority로 사용 하기 때문에 (0x40>>4) 즉 0x4 (01002) 값을 주고 있다. 이 값은 IPR1 register의 [23:20]에 write되게 된다.

22번 라인은 EXTI0 interrupt에 대해서 enable 시키라는 코드이다. EXTI0에 인터럽트가 발생하면 그것이 CPU core에 전달되게 된다. NVIC의 ISER0 register의 bit [6]를 set 시 키는 기능을 수행한다.

25번 라인은 SW를 이용하여 EXTI0 interrupt를 발생시키는 코드이다. 즉 NVIC의 ISPR0 register의 bit [6]를 set 시키는 기능을 수행한다.

=> line당 2점 4개

1. 디버거를 시작한 다음, 25번 라인과 5번 라인에 breakpoint를 설정하고, Peripherals -> Core Peripherals -> NVIC를 선택 다음, Idx 22번과 23번의 상태를 설명하시오. (E, P, A, Priority) 그리고 Interrupt Control & State register (scb->ICSR)의 주요 필드들의 상태를 설명하시오 (VECTACTIVE, RETTOBASE, VECTPENDING, ISRPREEMPT, ISRPENDING)

EXTI0와 EXTI1 모두 E=0 (disable 상태), P=0 (no pending 상태) A=0 (inactive 상태, 즉 interrupt가 실행 중이 아님) 그리고 priority는 0 (즉 group priority = 0, sub-priorirty = 0)인 상태를 나타냄. (4점) => EPA priority 각각 1점씩

VECACTIVE = 0 (현재 서비스(active 상태) 받고 있는 interrupt는 없음)

VECPENDING = 0 (현재 pending 상태인 interrupt는 없음)

RETTOBASE = 0 (handler mode에서 thread 모드로 돌아가는 상태가 아님)

ISRPREEMPT = 0 (다음 cycle 때 interrupt service 받을 interrupt는 없음)

ISRPENDING = 0 (pending 되어 있는 interrupt 없음) (5점) => 값 2점(사진도 인정), 의미 3점 (의미 2개당 +1점)

1. Run 버튼을 클릭하시오. 그러면 25번 breakpoint에서 실행이 일단 멈출 것이다. 이때 SP, 값은 무엇인가? 그리고 CONTROL register 값이 무엇인지 기록하고, 그 값의 의미 를 설명하시오. 그리고 Peripherals -> Core Peripherals -> NVIC에서 Idx 22번과 23번 의 바뀐 상태를 설명하시오. (E, P, A, Priority)

SP = 0x2000\_0648 (1점) (사진도 인정)

CONTROL = 0x04 즉 CONTROL[2]=1 즉 FPU 관련된 명령어가 사용되었음을 나타내고 있음. (3점) => 값 1점, 의미 2점

EXTI0와 EXTI1 interrupt 모두 E=1 즉 interrupt enable되었음. EXTI0의 우선 순위가 64 (0x40) 따라서 group priority = 01b = 1, sub-priority = 0 으 로 설정되었음. (4점, E설명:3점, 그룹우선순위표현:1점)

1. Run 버튼을 클릭하시오. 그러면 5번 breakpoint에서 실행이 일단 멈출 것이다. 이때 SP 값은 무엇인가? Stack에 저장된 내용들을 설명하시오. 그리고 Peripherals -> Core Peripherals -> NVIC와 Interrupt Control & State register (scb->ICSR)의 주요 필 드들에서 바뀐 상태와 LR에 나타난 EXC\_RETURN 값의 의미를 설명하시오.

SP = 0x2000\_05D8 (3점) (사진도 인정)

Stack 메모리에 저장된 값 (사진 또는 어떤 주소에 어떤 값이 있는지 전부 설명 (아래 같이) or 주소의 시작점부터 순서대로 의미를 나열) => 4점(1, 1, 1, 1)

0x20000644 0x60000005 NVIC\_SetPendingIRQ(6) 루틴에서 사용하고 있는 stack

0x20000640 0x000000000 (8-byte) Alignment을 위한 추가 영역. xPSR[9] 확인!

0x2000063C 0x000000005 (extended stack frame) Alignment을 위한 영역

0x20000638 0x00000000 FPSCR

0x20000634 0x00000000 S15 저장을 위한 영역

0x20000630 0x20000260 S14 저장을 위한 영역

0x2000062C 0x00000000 S13 저장을 위한 영역

0x20000628 0x00000000 S12 저장을 위한 영역

0x20000624 0x00000000 S11 저장을 위한 영역

0x20000620 0x00000000 S10 저장을 위한 영역

0x2000061C 0x00000000 S9 저장을 위한 영역

0x20000618 0x00000000 S8 저장을 위한 영역

0x20000614 0x00000000 S7 저장을 위한 영역

0x20000610 0x00000000 S6 저장을 위한 영역

0x2000060C 0x00000000 S5 저장을 위한 영역

0x20000608 0x00000000 S4 저장을 위한 영역

0x20000604 0x00000000 S3 저장을 위한 영역

0x20000600 0x00000000 S2 저장을 위한 영역

0x200005FC 0x00000000 S1 저장을 위한 영역

0x200005F8 0x00000000 S0 저장을 위한 영역

0x200005F4 0x41000200 xPSR, xPRST[9] == 1

0x200005F0 0x08000368 PC (return address)

0x200005EC 0x08000423 LR

0x200005E8 0x20000040 R12

0x200005E4 0x20000260 R3

0x200005E0 0x00000000 R2

0x200005DC 0xE000E200 R1

0x200005D8 0x00000040 R0

EXTI0의 A = 1 (즉 EXTI0의 interrupt가 active 즉 서비스 중이라는 의미임.)

VECTACTIVE = 0x16 (=22) 즉 현재 active 상태인 interrupt가 22번이라는 의미임.

RETTOBASE = 1 (즉 현재 서비스를 받고 있는 interrupt가 종료되면 thread 모드로 돌 아 간다는 의미임.) (4점) => (값과 의미 별로 1~2개 틀리면 -1, 전부 틀리면 0점 )

(특정 값을 가정하고 의미를 설명한 경우 의미만 있어도 인정, 단 사진 있는 경우만)

EXC\_RETURN = 0xFFFF\_FFE9 (4점) => 값 2점, 설명 틀린 항목 1~2개당 -1

* Bit 4 = 0 (extended stack frame is used)
* Bit 3 = 1 (interrupt에서 복귀 시 thread 모드)
* Bit 2 = 0 (interrupt에서 복귀 시 main stack)
* Bit 1 = 0 (reserved bit) => 설명 없어도 됨
* Bit 0 = 1 (interrupt에서 복귀 시 thumb 모드)

1. 디버거를 reset시키고 (RSR 버튼 클릭) Run 버튼을 클릭하시오. 그러면 25번 breakpoint에서 실행이 일단 멈출 것이다. ((c)에서의 SP 값이 같음을 일단 확인하시 오.) 이 때 CONTROL register의 값을 0x00으로 변경한 다음, Run 버튼을 클릭하시오. 그러면 5번 breakpoint에서 실행이 일단 멈출 것이다. 이때 SP 값은 무엇인가? 그 리고 Stack에 저장된 내용들을 설명하시오. 그리고 LR에 나타난 EXC\_RETURN 값의 의미를 설명하시오.

SP = 0x2000\_0620 => 2점

Stack 메모리에 저장된 값 사진 또는 주소별 값 정확히 설명 => 3점, 위에 2line은 1점, 그 아래 3줄 1점, R 1점

0x20000644 0x06000005 NVIC\_SetPendingIRQ(6) 루틴에서 사용하고 있는 stack

0x20000640 0x00000000 8-byte) Alignment을 위한 추가 영역. xPSR[9] 확인!

0x2000063C 0x41000200 xPSR

0x20000638 0x08000368 PC (return address)

0x20000634 0x08000423 LR

0x20000630 0x20000040 R12

0x2000062C 0x20000260 R3

0x20000628 0x00000000 R2

0x20000624 0xE000E20000 R1

0x20000620 0x00000040 R0

EXC\_RETURN = 0xFFFF\_FFF9 (2점) => 값 1점, 의미 1점(의미 부분 없음)

* Bit 4 = 1 (standard stack frame is used)
* Bit 3 = 1 (interrupt에서 복귀 시 thread 모드)
* Bit 2 = 0 (interrupt에서 복귀 시 main stack)
* Bit 1 = 0 (reserved bit) => 해당 부분은 설명 없어도 됨
* Bit 0 = 1 (interrupt에서 복귀 시 thumb 모드)

1. (d)와 (e)의 결과가 다른 이유를 설명하시오. (5점)

CONTROL[2]=1 인 경우는 FPU 관련된 명령어가 사용되었음을 의미하므로 ISR routine에서 FPU 관련 instruction을 사용하면 FPU 관련 register 중에서 S0 ~ S15까지의 값을 stack에 저장하여 값이 변하지 않도록 동작하도록 되어 있다. (하지만 S0 ~ S15 register 값을 무조건 저장하지는 않는다. ISR에서 FPU 관련 명령어를 사용하게 되면 그때, S0 ~ S15 register 값이 현재 예비된 stack 영역에 저장되게 된다. 이와 같은 mechanism을 last stacking이라고 한다.)

하지만 CONTROL[2]=0 인 경우는 FPU 관련 명령어가 사용되지 않았기 때문에 ISR에서 FPU 관련 명령어를 사용한다고 하더라도 굳이 이전 FPU register 값을 저장할 필요 가 없다. 따라서 FPU register S0 ~ S15 저장을 위한 공간을 stack에 확보할 필요는 없다.

(control reg가 달라져 FPU관련 명령어를 사용하지 않기 때문에 달라진다.)