Pre-lab Assignment 4

Due: Thu, 19 Sep 2019 14:30:00 (approximately 15 hours from the time this page was loaded)
[15 points possible]

An interrupt is a hardware-invoked subroutine. Some form of trigger causes execution of the normal program to be suspended. A subset of the *state* of the CPU is saved on the stack. In particular, registers R0-R3, R12, LR, PC, and xPSR are saved. This stack content is called an exception frame. A special value is loaded into the LR register to indicate that the stack contains an exception frame. This tells a subsequent **`BX LR`** or **`POP {...,LR}`** instructions to load the exception frame contents back into their respective registers. In this way the interrupted execution can be treated just like a subroutine that can be *returned* from.

The type of trigger decides what form of *exeception vector* the CPU should read from the vector table. The 32-bit value read from the vector table is put into the PC and execution continues in this new subroutine. Since the subroutine is meant to handle some kind of interrupt, it is called an Interrupt Service Routine (ISR). Since R0-R3, R12, LR, PC, and xPSR have been saved, the subroutine can modify all of the registers that the ABI normally allows. If it modifies R4-R7, they should be saved on the stack in an initial **`PUSH {R4-R7,LR}`** instruction at the start of the subroutine.

---

(1) [1 point]

Look at startup/stm32f0xx.S in any Standard Peripheral project in System Workbench. What is the address range of the STM32 exception vector table? (State it in the form of two hexadecimal numbers like 0x***FROM*** - 0x***TO***) Include the reset vector and initial stack pointer value. Include the space up to, but not including, the word for BootRAM. This is not necessarily the same value you saw in the lecture notes.

0x00 - 0x144

Save

---

(2) [1 point]

Each value label used in a vector table entry in startup/stm32f0xx.S is defined further down in the file with a **.weak** directive. What does this mean?

It means that any other .global symbols with the same name will be override it but it none is available it will be used.

Save

---

(3) [1 point]

Each value label used in a vector table entry in startup/stm32f0xx.S is defined further down in the file with a **.thumb_set** directive. What does this mean?

This tells the linker the linker that the address is for a function. Without this the address in the vector table would be even and the CPU would consider the label as ARM rather than Thumb code.

Save

---

(4) [1 point]

In addition to declaring it .global, we recommend, for any label of a subroutine used as an ISR, to also declare it with the following directive:

```
.type some_handler, %function
```

What does this do? (Hint: It does the same thing as .thumb_set)

This tells the linker the linker that the address is for a function. Without this the address in the vector table would be even and the CPU would consider the label as ARM rather than Thumb code.

Save

---

(5) [1 point]

Why are things like .thumb_set needed? I.e., what happens if you omit them?

Without this the address in the vector table would be even and the CPU would consider the label as ARM rather than Thumb code.

Save

---

(6) [1 point]

How many distinct exception vectors does a STM32F0 microcontroller support? (Hint: Take a look at the g_pfnVectors table in startup/startup_stm32f0xx.S and count them. Note that 0 entries do not represent a supported handler.) Some of them may not be supported by the specific microcontroller we use for ECE 362. For instance, we have no Consumer Electronic Control (CEC) peripheral in the STM32F051R8T6. The F0 family still has a defined vector location for it.

36

Save

---

(7) [1 point]

Which bit in which register enables the SysTick countdown timer?

The lower 3 bits in the SYST_CSR

Save

---

(8) [1 point]

What is the address of the SysTick_CSR register? (Indicate the absolute address, not the offset from the SysTick register base.)

0xe000e010

Save

---

(9) [1 point]

Write a simple ISR for the interrupt generated by SysTick. You should make it just like the one you saw in the lecture notes, except that your ISR should toggle PC8. You do not need to initialize the SysTick subsystem---just write the ISR. You may assume PC8 is already set up for output. Assume the availability of an .equs named GPIOC and ODR that represent the address of the GPIOC port registers and the offset of the Output Data Register, respectively. (This is easy enough to try on your own for practice before your lab. Make that blue LED blink, and build your confidence. Keep in mind that a Standard Peripheral project in System Workbench makes the CPU run at 48 MHz. The **maximum** value you can store in the SysTick reload value register is 16,777,215. That would generate a SysTick interrupt every 16,777,215/48,000,000 = 0.35 seconds.)

```
SysTick_Handler:
        push {lr}
        ldr r0, =GPIOC
        ldr r1, [r0, #ODR]
        movs r2, #0x100
```

Save

---

(10) [1 point]

Why is it that we need to enable the EXTI interrupt in the NVIC ISER register but not for the SysTick interrupt?

```
SysTick interrupt is a non-maskable exception.
```

Save

---

(11) [1 point]

Which register is used to un-mask the EXTI interrupt for a particular pin?

EXTI_IMR

Save

---

(12) [1 point]

Name the register that can be used to set the EXTI interrupt to be triggered on a rising edge?

EXTI_RTSR

Save

---

(13) [1 point]

Which EXTI interrupt handler are pins 0 and 1 of any port required to use? (Hint: Look at the vector table in startup/stm32f0xx.S to find the exact name of the interrupt handler. Copy and paste it below. You must always use the exact name of the ISR to override the (weakly definined) default handler. If you don't, it won't even give you a warning. Always copy and paste the value from startup/stm32f0xx.S file. Did we mention you can always find the exact name in startup/stm32f0xx.S? Lots of students forget this, and they try to type it in from memory or a guess.)

EXTI0_1_IRQHandler

Save

---

(14) [1 point]

What register must you write to to change the port used for an EXTI on pin 0? (E.g., if you wanted to use PB0 instead of PA0 for an EXTI.)

SYSCFG

Save

---

(15) [1 point]

The subsystem for the register you named in question 14 must have a clock enabled for it before it can be modified. This is similar to how you must set the GPIOCEN bit in the RCC_AHBENR to enable the clock to Port C. (Almost every subsystem in the STM32 other than the NVIC and SysTick needs to have a clock enabled before it will do anything.) What bit in what register must you set to '1' to enable that clock before you update the register in question 14? (Write the symbol name of the register and bit.)

NVIC_ISER bit 5

Save