Exercises

1. Which modules generate the IRQ0, IRQ10 and IRQ16 interrupt requests? Examine the Interrupt Vector Assignments table in the reference manual and the appropriate *device*.h file.

The next several questions involve configuring registers so that if interrupts IRQ0, IRQ10 and IRQ16 are requested simultaneously, the CPU responds as requested. For each question, explain what values must be loaded into which registers, and then write C code which uses the CMSIS functions to perform that operation.

1. Interrupts are serviced in order IRQ10, IRQ0, IRQ16.
2. Interrupts are serviced in order IRQ0, IRQ16, IRQ10.
3. No interrupts are serviced.
4. We wish to enable IRQ13 but disable IRQ24. What value needs to be loaded into which register bits, and what is the CMSIS code call to accomplish the same?
5. We wish to determine if IRQ7 has been requested. Which register and which bit will indicate this? What is the CMSIS call which will reveal the information?
6. Which register can an exception handler use to determine if it is servicing exception number 0x21? What value will the register have? What is the CMSIS interface code to read the IPSR?
7. Fill in the table below to show the contents of memory and registers after the CPU responds to an IRQ2 exception request but before it executes the first instruction of the exception handler.

Initial State:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Register** | **Value** | **Register** | **Value** | **Register/State** | **Value** |
| R0 | 10 | R8 | 18 | xPSR | 0x0100\_0000 |
| R1 | 11 | R9 | 19 | MSP | 0x2000\_0FF4 |
| R2 | 12 | R10 | 20 | PSP | n/a |
| R3 | 13 | R11 | 21 | PRIMASK | 0x0000\_0000 |
| R4 | 14 | R12 | 22 |  |  |
| R5 | 15 | R13 (SP) | 0x2000\_0FF4 | Mode | Thread |
| R6 | 16 | R14 (LR) | 0x1000\_0000 | Privilege | Privileged |
| R7 | 17 | R15 (PC) | 0x0000\_0220 | Stack | Main |

|  |  |
| --- | --- |
| **Exception No.** | **Value** |
| 12 | 0x0000\_1441 |
| 13 | 0x0000\_1211 |
| 14 | 0x0000\_1335 |
| 15 | 0x0000\_1669 |
| 16 | 0x0000\_120D |
| 17 | 0x0000\_1129 |
| 18 | 0x0000\_1351 |
| 19 | 0x0000\_1539 |
| 20 | 0x0000\_1285 |
| 21 | 0x0000\_183D |
| 22 | 0x0000\_1849 |
| 23 | 0x0000\_1661 |

After responding to exception request, before executing exception handler:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Register** | **Value** | **Register** | **Value** | **Register/State** | **Value** |
| R0 |  | R8 |  | xPSR |  |
| R1 |  | R9 |  | MSP |  |
| R2 |  | R10 |  | PSP |  |
| R3 |  | R11 |  | PRIMASK |  |
| R4 |  | R12 |  |  |  |
| R5 |  | R13 (SP) |  | Mode |  |
| R6 |  | R14 (LR) |  | Privilege |  |
| R7 |  | R15 (PC) |  | Stack |  |

|  |  |  |
| --- | --- | --- |
| **Address** | **Value** | **Description** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. What is the return address will an exception handler return to, given the following register and memory contents? Assume the instruction for exiting the exception handler is POP {R4,PC}.

Graphical user interface, application, table

Description automatically generated with medium confidence

Consider a system built on a 48 MHz processor with a main loop and an interrupt which occurs at a 10 kHz frequency. Assume the ISR take 14.9 us to execute and there is a total of 1 us of response and return time overhead.

1. What percentage of the processor’s time is spent in interrupt response and return overhead?
2. What percentage of the processor’s time is left for the main loop to execute?
3. If the main loop requires 37 ms of computation to execute one iteration, what is the minimum main loop update rate (in Hz)?

Consider the system above again, but now with the interrupt running at 25 kHz and the ISR taking 34 us to execute.

1. What percentage of the processor’s time is spent in interrupt response and return overhead?
2. What percentage of the processor’s time is left for the main loop to execute?
3. If the main loop requires 37 ms of computation to execute one iteration, what is the minimum main loop update rate (in Hz)?