**Chapter 3 Example**

Assume the following 8-bit processor architecture. You are given the task of implementing a **prioritization scheme** among 7 I/O devices where an I/O device A has a higher priority than an I/O device B, etc (A > B > C > D > E > F > G). Main program has the lowest priority (<G).

The mapping for the IPENDING and IENABLE registers is the same and shown below. The bit 0 in PS register is the interrupt enable bit and bits 3,4,5 are the priority level bits, where Pr0 is the LSB and Pr2 is the MSB. Assume the lowest value (000) indicates the highest priority, i.e., device A has a priority level 000, main program has priority level 111.

Also shown are the control/status registers for devices A, C, and G, where the interrupt enable bits and the three bits that identify the priority level are shown. For efficiency, control bits (IE) and the status bits (priority) are held in the same register.

In the following, you will implement the necessary code to handle interrupts from devices A, C, and G.

**IPENDING and IENABLE register**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B | D | A | G | C | F | E |  |

**PS register**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Pr2 | Pr1 | Pr0 |  |  | IE |

**I/O Device A Control/Status Register (A\_CONT)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 0 | 0 |  | AIE |  |  |

**I/O Device C Control/Status Register (C\_CONT)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | CIE |  | 0 | 1 | 0 |  |

**I/O Device G Control/Status Register (G\_CONT)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | GIE | 1 | 1 | 1 |

A, C, and G are all input devices. When interrupt occurs on one of the devices, the corresponding bit in the IPENDING register is set and the data is available in the associated data register A\_DATA, C\_DATA, or G\_DATA (not shown).

1. What is the 8-bit pattern for IENABLE to enable only the devices A, C, and G?
2. What is the appropriate number in **decimal** (**not** binary) for the third operand in the second instruction below, to enable interrupts in device C interface?

LoadByte R2, C\_CONT

Or R2, R2, #\_\_\_\_\_\_\_\_ (?)

StoreByte R2, C\_CONT

1. Can we use ‘Move’ instruction instead of ‘Or’ in question (b) above? Why? Why not? Explain.
2. The main program is shown below with some of the code missing. Fill in the missing instructions or empty locations with (?) based on the comments. Use **decimal** numbers (**not** binary).

|  |  |  |  |
| --- | --- | --- | --- |
| **Main program:** |  |  |  |
| START: | … |  | Set up parameters for ISRs |
|  | LoadByte | R2, A\_CONT |  |
|  | Or | R2, R2, #\_\_\_ (?) | Enable interrupts |
|  | StoreByte | R2, A\_CONT | in device A interface |
|  | LoadByte | R2, C\_CONT |  |
|  | Or | R2, R2, #\_\_\_ (?) | Enable interrupts |
|  | StoreByte | R2, G\_CONT | in device C interface |
|  | Or | R2, R2, #\_\_\_ (?) | Enable interrupts |
|  | StoreByte | R2, G\_CONT | in device G interface |
|  | MoveControl | R2, IENABLE | Enable interrupts for the |
|  | \_\_\_\_\_\_\_\_\_\_\_\_(?) | \_\_, \_\_, \_\_ (?) | three devices in processor |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ (?) | IENABLE, R2 | control register |
|  | MoveControl | R2, PS | Set the priority level to 111 |
|  | Or | R2, R2, #\_\_\_ (?) | and enable interrupts |
|  | MoveControl | PS, R2 | in PS |
|  | next instruction.. |  |  |

1. In this part, you are asked to implement the interrupt handler. When interrupt handler is called, the IE bit in PS is automatically reset. It is set automatically after return-from-interrupt. The interrupt handler should perform the following:

* Save appropriate registers
* Check the current priority level
* Check the contents of IPENDING
* Check the priority level(s) of the device(s) that raised the request
* If a device with a priority lower than the current priority raised the request, clear the corresponding IPENDING bit (ignore interrupt) and return
* If a device with a priority higher than the current priority raised the request, branch to its ISR (AISR, CISR, or GISR).
* To enable interrupt nesting, set the IE bit in PS at the appropriate time (Hint: you would not want to set it immediately at the beginning to take care of required operations before an interrupt can …well… interrupt)
* Load appropriate registers back from stack

You are expected to use at least the stack; LINK\_reg, IPENDING, PS, A\_CONT, C\_CONT, and G\_CONT registers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interrupt Handler:** |  |  |  |
| ILOC: |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Return-from-interrupt | | |