BLG 322E - Recitation 1 (Part 2)

Doğukan Arslan
arsland15@itu.edu.tr

Question 1

A CPU with an 8-bit data bus has an Interrupt Request input (**IRQ**) and an Interrupt Acknowledgment output (**INTA**). Both signals are active at “1”. If the vectored/autovectored input (**VA**) of the CPU is “0”, the CPU works with vectored interrupts, so that it reads the interrupt vector number after the acknowledgment of the interrupt, when its Data Acknowledgment input (**DACK**) is “1”. The CPU also supports autovectored interrupts. After the acknowledgment of an interrupt request (**INTA=1**), if the vectored/autovectored input (**VA**) of the CPU is set to “1”, then the CPU does not read a vector number and works in autovectored mode.

Question 1

In this system, there are **four interrupt sources** (A1, A2, B1, and B2) of **two different types** (A and B):

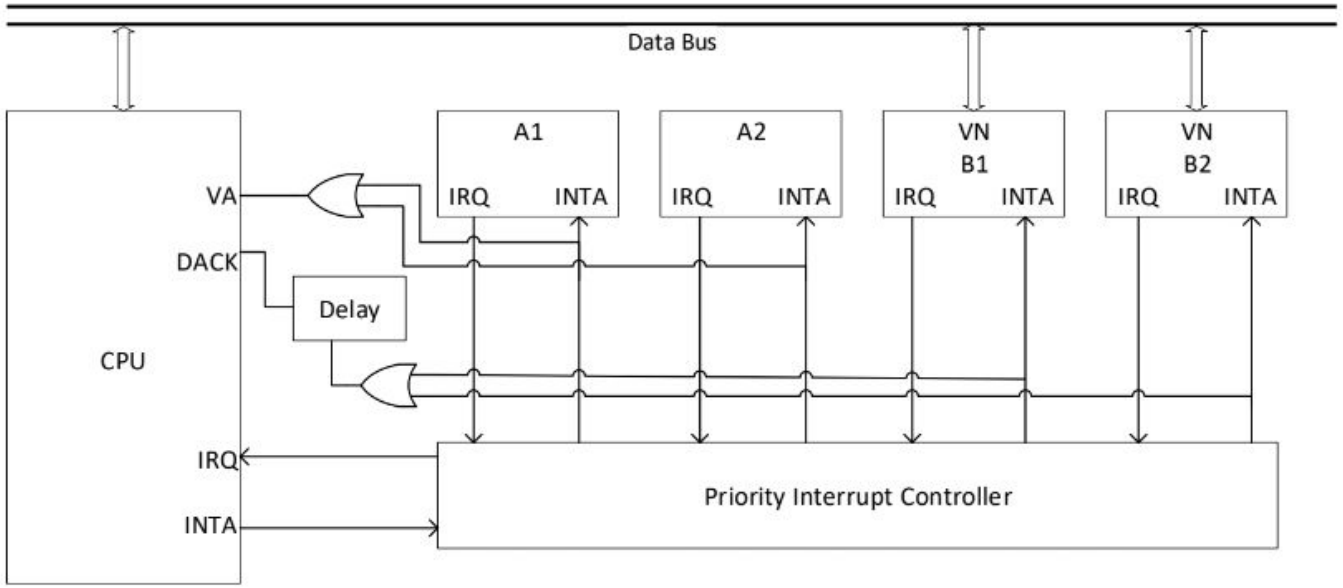
Type A (A1 and A2): Each one of these devices has an Interrupt Request output (**IRQ**) and an Interrupt Acknowledgment input (**INTA**). They do not have vector number outputs. They work in **autovectored mode**.

Type B (B1 and B2): Each one of these devices has an Interrupt Request output (**IRQ**), an Interrupt Acknowledgment input (**INTA**), and an 8-bit vector number output (**VN**).

Priority (precedence) order of the devices: *A1* > *A2* > *B1* > *B2*

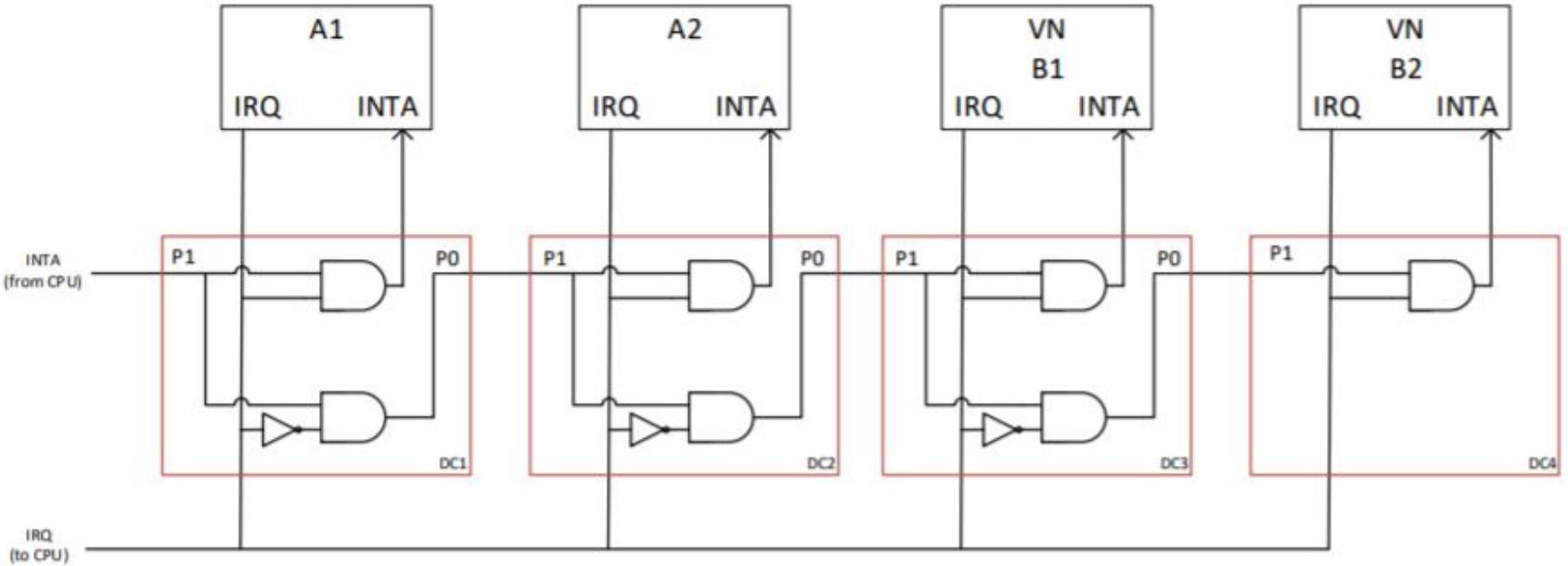
Question 1

➔ Design and draw the system with the **CPU**, the **four devices** (A1, A2, B1, and B2), and the **priority interrupt controller**. First, show the priority interrupt controller only as a black box. Then, design and draw the internal structure of the priority interrupt controller using logic gates.



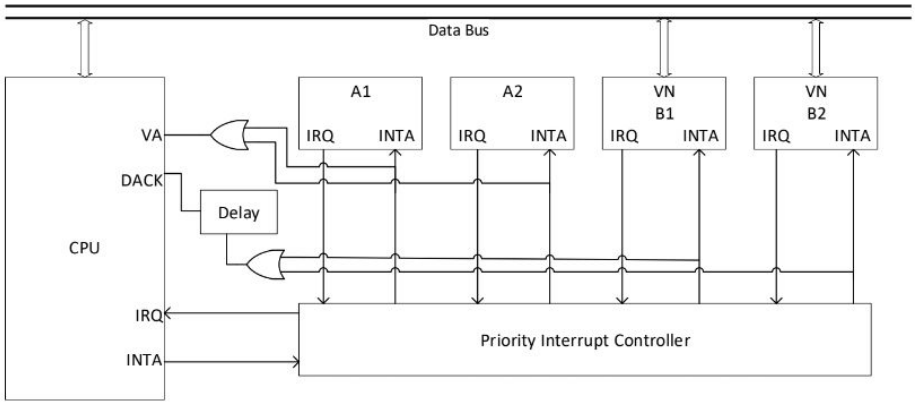
Question 1

➔ Design and draw the system with the **CPU**, the **four devices** (A1, A2, B1, and B2), and the **priority interrupt controller**. First, show the priority interrupt controller only as a black box. Then, design and draw the internal structure of the priority interrupt controller using logic gates.



Question 1

➔ Assume that devices **A1** and **B1** assert their interrupt requests **at the same time**. Show all the signals that are sent in the system step-by-step until requests of both devices have been fulfilled.



IRQ(**A1**) = 1, IRQ(**B1**) = 1

IRQ(**CPU**) = 1

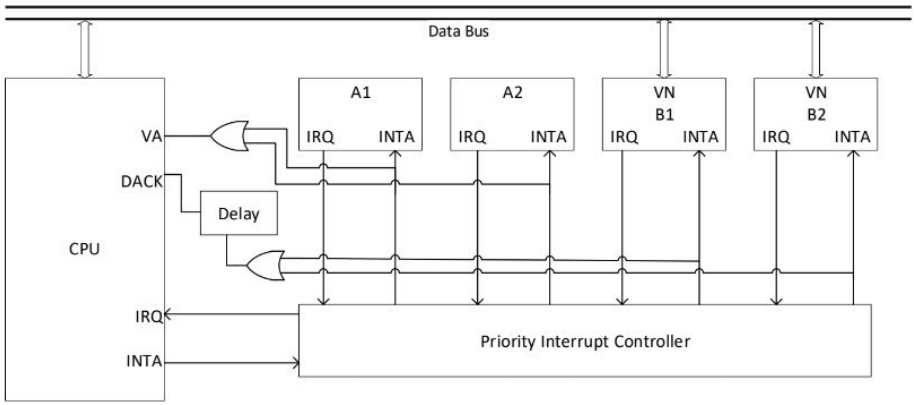
INTA(**CPU**) = 1 (Interrupt request accepted)

PI(**DC1**) = 1, PO(**DC1**) = 0, INTA(**DC1**) = 1 (Source of the request is A1)

INTA(**A1**) = 1, **VA** = 1. The CPU determines that the interrupt is autovectored from **VA** and checks the flags of A1 and A2 (**software-based polling**). Then, it determines that the source is **A1**. The request of **A1** is removed (IRQ(**A1**) = 0). The ISR jumps to the procedure of A1 and returns after the operation.

Question 1

➔ Assume that devices **A1** and **B1** assert their interrupt requests **at the same time**. Show all the signals that are sent in the system step-by-step until requests of both devices have been fulfilled.



IRQ(**A1**) = 0, IRQ(**B1**) = 1, IRQ(**CPU**) = 1

INTA(**CPU**) = 1 (Interrupt request accepted)

PI(**DC1**) = 1, PO(**DC1**) = 1 (Source of the request is not A1)

PI(**DC2**) = 1, PO(**DC2**) = 1 (Source of the request is not A2)

PI(**DC3**) = 1, PO(**DC3**) = 0 (Source of the request is B1)

INTA(**B1**) = 1, **VA** = 0 (Vectored interrupt. The vector number is put on the data bus)

IRQ(**B1**) = 0 (B1 removes request)

DACK = 1. CPU reads vector number of B1, gets the start address of the ISR from the vector table, and runs the ISR of B1. The ISR returns to the main program.

Question 1

➔ How does the CPU determine the start address of the interrupt service routine that will be run if the interrupt source is a device of type **A** or of type **B**?

Type A devices do not supply the CPU with a vector number, and interrupts are **autovectored**. In order to determine the source of the interrupt request, the CPU checks the **flags** of type A devices (software-based polling) in the interrupt service program for the autovectored interrupts. The CPU decides that the interrupt is autovectored based on the value of VA input. After determining the source of the request, the CPU jumps to the program of the source device.

Type B devices put the vector number on the data bus. After determining that the interrupt is **vectored** and **acknowledging** the interrupt, the CPU reads the vector number and fetches the beginning address of the corresponding ISR from the vector table. Then, the CPU jumps to this ISR.

Question 2 (2019 2nd Midterm)

A CPU has an Interrupt Request (**IRQ**) input that is active at “1”. It does not have an Interrupt Acknowledgement (**INTAK**) output. If the CPU accepts a request from the IRQ, next requests are disabled by clearing the interrupt mask (**IM=0**). Then the CPU runs a fixed interrupt service routine (**ISR**) that is assigned to the IRQ.

In this system, there are four serial communication interfaces (SCI1, SCI2, SCI3, SCI4). Each of these devices has an Interrupt Request output (**IRQx**; x=1,2,3,4) that is active at “1”. When a SCI receives data, the “received” bit R in its status register is set (**R=1**), and its Interrupt Request output (IRQx) becomes “1”. If the data register of a SCI is read, the R is cleared (**R=0**) and IRQ of the SCI becomes “0” (**IRQx=0**).

Priority (precedence) order of the devices: SCI1 > SCI2 > SCI3 > SCI4.

Question 2 (2019 2nd Midterm)

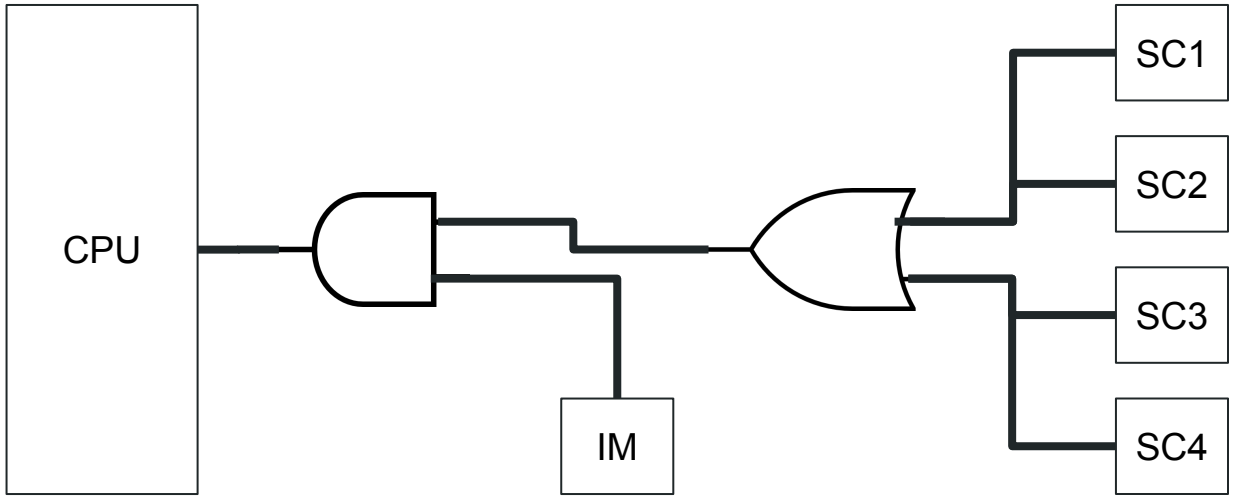
➔ Which type of interrupt priority controller can (or cannot) be used in this system to connect four SCIs to the CPU? Why?

Since there is no **INTACK** from CPU and vector numbers are not read from interrupt sources, we cannot use a hardware based priority controller. Instead, we can use a **software-based** approach.

Interrupt Requests outputs (IRQx) from devices can be **or-gated** to have a single IRQ output. On the CPU, a **software based polling mechanism** can be used to determine the source of interrupt request and control the priority of these Interrupt Sources.

Question 2 (2019 2nd Midterm)

➔ Design and draw the system with the **CPU**, four **SCIs devices** (SCI1, SCI2, SCI3, SCI4) and the necessary **logic gates**. Use the minimum number of elements to obtain a circuit with the lowest cost.



Question 2 (2019 2nd Midterm)

➔ How does this system determine the **interrupt source with the higher precedence** order, when more than one devices send requests?

In the ISR, the **order** in which the interrupt sources are polled determined the priority. Based on the given priority order, first IS1 must be polled (remember that **polling is done by reading the device's status register**). If the IS1 has an outstanding request, then the IS1's request will be served. Otherwise, IS2 will be polled. Similarly, if IS2 has a request then it will be served. Otherwise, IS3 will be polled. IS4 will be the last to be polled.

.

Question 2 (2019 2nd Midterm)

➔ How does the CPU identify the **interrupt source** and run the **proper service routine** for the requesting device?

Since there is a fixed ISR and the interrupt sources do not have a capability to provide a vector number, the interrupt source is identified by **reading the device's status register (software polling)**. This fixed ISR must include code to provide the service for each device; or the ISR can include "Jump to Subroutine" instructions that call the procedures related to the interrupting devices.

Question 3

➔ Suppose that you have a processor connected to an I/O device that can transfer data at an average of **2 MB/s** using interrupt-driven I/O (1 Megabyte = 10^6 Bytes). The interrupt service program transfers **two bytes** each time. Interrupt processing (including the interrupt service program that transfers two bytes and housekeeping operations) takes **740 ns**. Then, what is the percentage of time used for data transfer?

$2x10^6$ bytes transferred **per second**.

10^6 CPU cycle **per second** (since 2 bytes transferred for every interrupt)

So, 1 CPU cycle takes **10^3 ns** (since 1 s = 10^9 ns)

Since 740 ns of 10^3 ns is consumed by I/O device, fraction is **%74**.