Computer Architecture Recitation 3

Abdullah Akgül, Fırat Öncel

27.04.2023

Question 1

A system consists of devices that work as interrupt sources and a CPU.

- The CPU has an Interrupt Request input (IRQ) and an Interrupt Acknowledgment output (INTA). All signals are active at “1”. If the CPU receives a request over the IRQ input, it works with vectored interrupts and reads the interrupt vector number after the acknowledgment (INTA=1).
- Each of the interrupt sources has an Interrupt Request output (IRQ), an Interrupt Acknowledgment input (INTA), and a vector number output (VN). After the interrupt has been acknowledged (INTA=1), the device outputs its vector number at VN and removes its request (IRQ=0).

Question 1

We designed a digital combinational circuit `DC` with four inputs (I_1, I_2, I_3, I_4) and four outputs (O_1, O_2, O_3, O_4) to use as a priority interrupt controller. The logic expressions for the outputs are given below.

- $O_1 = I_2 + I_3 + I_4$
- $O_2 = I_1 \cdot I_2$
- $O_3 = I_1 \cdot I_3$
- $O_4 = I_1 \cdot I_4$

Hint: The output O_1 of `DC` is connected to the IRQ input of the CPU.

a) Since the circuit `DC` was not designed properly, it cannot be used as a priority interrupt controller. Briefly explain the reason.

b) Modify the expressions for the outputs where necessary. Write the modified expressions. Specify which input/output of `DC` will be connected to which input/output of the CPU or interrupt sources.

Solution 1a

- Since output O_1 of the DC is connected to the IRQ input of the CPU, the remaining outputs (O_2, O_3, O_4) can control three interrupt sources.
- Since input variable I_1 appears in the logic expressions for all outputs (O_2, O_3, O_4) that control the devices, it should be connected to the Interrupt Acknowledgment (INTA) output of the CPU.
- The remaining inputs (I_2, I_3, I_4) of DC should be connected to the Interrupt Request (IRQ) outputs of the interrupt sources.

The given logic expressions do not provide priority levels for the outputs. If multiple devices issue interrupt requests, they all get acknowledgments at the same time. For example, if $(I_2 = 1, I_3 = 1, I_4 = 1)$, then after the acknowledgment $(I_1 = 1)$, all outputs are 1, i.e.,$(O_2 = 1, O_3 = 1, O_4 = 1)$.

Solution 1b

Assumption: Order of precedence $I_2 > I_3 > I_4$

- $O_1 = I_2 + I_3 + I_4$
- $O_2 = I_1 \cdot I_2$
- $O_3 = I_1 \cdot \overline{I_2} \cdot I_3$
- $O_4 = I_1 \cdot \overline{I_2} \cdot \overline{I_3} \cdot I_4$

Connections of `DC`:

- O_1: IRQ input of the CPU
- I_1: INTA output of the CPU
- I_2, I_3, I_4: IRQ outputs of the interrupt sources
- O_2, O_3, O_4: INTA inputs of the interrupt sources

Question 2

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 2

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: $A1 > A2 > B1 > B2$

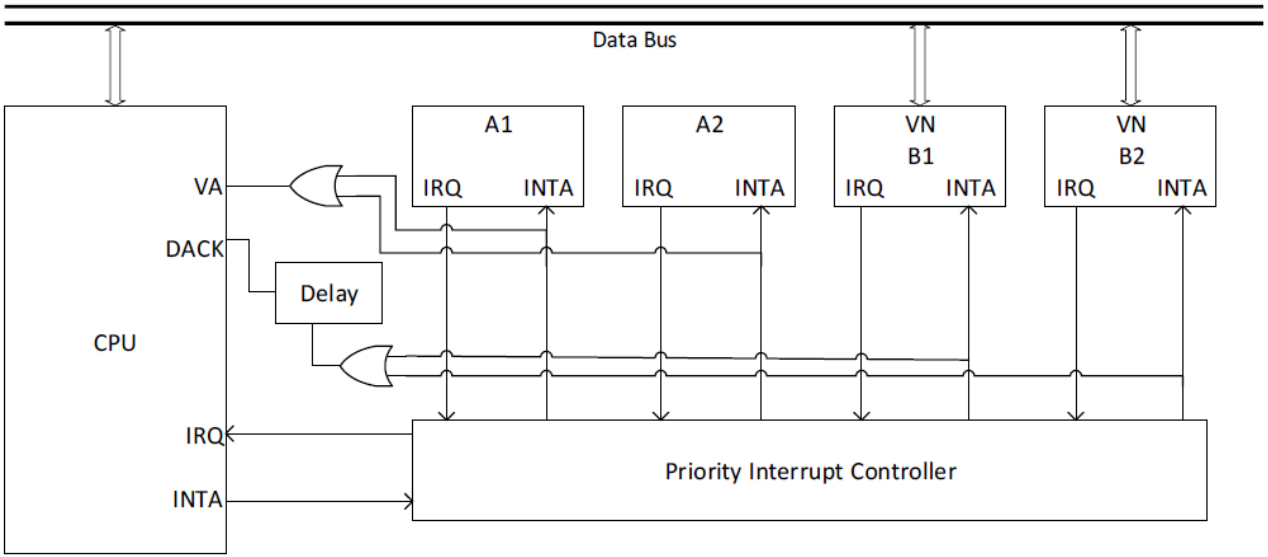
Question 2

a) 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.

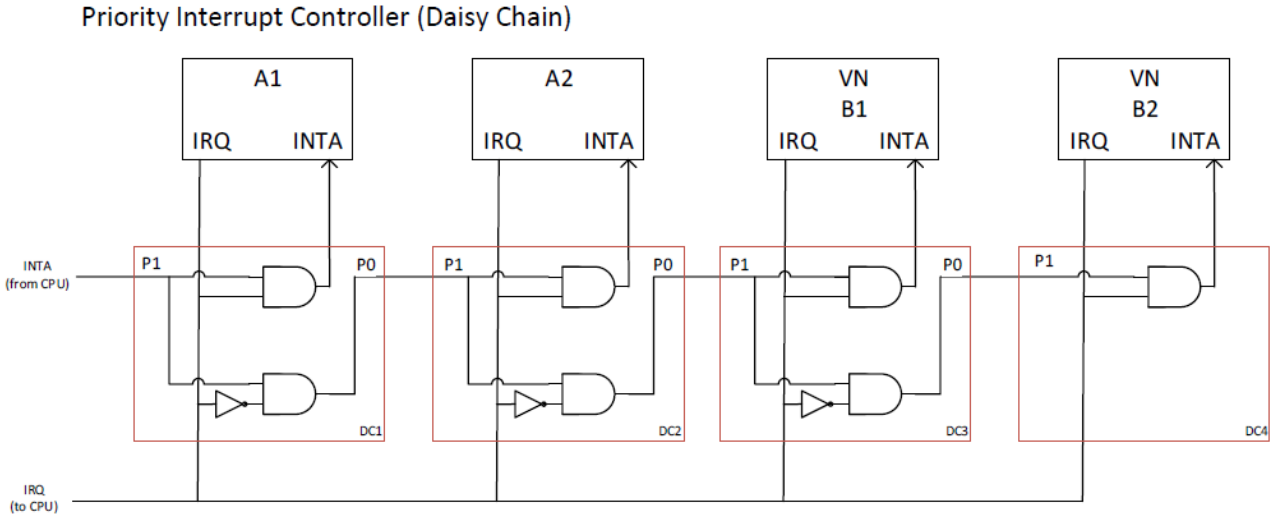
b) Assume that devices A1 and B1 assert their interrupt requests at the same time. Show all the signals that are sent in the system step-by-step until requests of both devices have been fulfilled.

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

Solution 2a


Data Bus
A1
IRQ
INTA
A2
IRQ
INTA
VN
B1
IRQ
INTA
VN
B2
IRQ
INTA
VA
DACK
Delay
CPU
IRQ
INTA
Priority Interrupt Controller

Solution 2a


Priority Interrupt Controller (Daisy Chain)
A1
IRQ
INTA
A2
IRQ
INTA
VN
B1
IRQ
INTA
VN
B2
IRQ
INTA
INTA
(from CPU)
P1
P0
DC1
DC2
DC3
DC4
IRQ
(to CPU)

Solution 2b

- 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.

Solution 2b

- 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 starting address of the ISR from the vector table, and runs the ISR of B1. The ISR returns to the main program.

Solution 2c

- Type A
  - A type A device does not supply the CPU with a vector number, and interrupts are **autovectored**.
  - To determine the source of the interrupt request, the CPU checks the flags of type A devices (software-based polling) in the interrupt service routine for **autovectored** interrupts.
  - The CPU decides that the interrupt is **autovectored** based on the value of the VA input.
  - After determining the source of the request, the CPU jumps to the program of the source device.
- Type B
  - A type B device puts 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 starting address of the corresponding ISR from the vector table.
  - Then, the CPU jumps to this ISR.

Question 3

The instruction cycle of an experimental CPU has these phases:

1. Instruction fetch: 50 ns (memory access needed),
2. Decode: 20 ns (memory access **not** needed),
3. Operand fetch: 70 ns (memory access needed),
4. Execution: 30 ns (memory access **not** needed),
5. Result write: 40 ns (memory access needed),
6. Interrupt housekeeping: 200 ns (memory access needed).

The memory access and I/O interface access times are 50 ns each.

- There is an interrupt source (`IS`). The interrupt service routine for `IS` executes 5 instructions.
- There is a 2-wire DMAC that is configured to transfer words from the I/O interface to memory using **cycle-stealing**. The DMAC type is **fly-by (implicit)** (data does not pass through the DMAC).

Question 3

Assume that we start a clock (Clock = 0) when the CPU begins to run the main program.

- At Clock = 250 ns, the device `IS` sends an interrupt request.
- At Clock = 675 ns, the DMAC attempts to start the data transfer from I/O interface to memory. We assume that the I/O interface is always ready to transfer.

a) When (Clock =?) will the interrupt service routine (ISR) of `IS` start to run? Why?

b) When (Clock =?) will the DMAC complete the transfer of the **first**, **second**, **third**, and **fourth** words? Why?

Solution 3a

a) When (Clock =?) will the interrupt service routine (ISR) of `IS` start to run? Why?

- **Remember**: Interrupt requests are checked after the completion of the instruction.
- In this system, an instruction is completed at Clock = 210 ns $(50 + 20 + 70 + 30 + 40)$.
- `IS` sends the interrupt request at Clock = 250 ns as the CPU is in the fetch cycle of the second instruction.
- The ISR can start after the housekeeping operations if there are no DMA requests.
- The second instruction is completed at Clock = 420 ns $(210 + 210)$.
- The ISR starts at **Clock = 620** ns $(420 + 200)$.

Solution 3b

b) When (Clock =?) will the DMAC complete the transfer of the **first**, **second**, **third**, and **fourth** words? Why?

- DMAC sends bus requests as the CPU is running the ISR.
- The first bus request arrives as the CPU is decoding the first instruction of the ISR. Since the CPU is not using the bus during decoding, the DMAC can immediately start transferring the first word at Clock = 675 ns. As the DMAC type is fly-by (implicit) and the access time is 50 ns, the DMAC completes the transfer of the first word at **Clock = 725** ns $(675 + 50)$.
- Similarly, we can calculate:
  1. 1. Word: at Clock = 725 ns
  2. 2. Word: at Clock = 845 ns $(725 + 70(1.\ \text{IS Operand Fetch}) + 50)$
  3. 3. Word: at Clock = 935 ns $(845 + 40(1.\ \text{IS Result Write}) + 50)$
  4. 4. Word: at Clock = 1035 ns $(935 + 50(2.\ \text{IS Ins. Fetch}) + 50)$

Question 4a

- A CPU normally runs a program in **1000** ns (if there are no DMA transfers).
- In this system, there is a **flow-through (explicit)** (data passes through the DMAC) direct memory access controller (DMAC) that is configured to transfer **10 words** from the I/O interface to memory using **cycle-stealing**.
- Memory access and I/O interface access times are **50** ns each.
- Assume that the DMAC attempts to start the data transfer from I/O interface to memory as the CPU starts the program and the I/O interface is always ready to transfer.
- In this case, how long would it take approximately for the CPU to run the same program? You should give your answer in one of these three forms:
  1. "between X ns and Y ns"
  2. "X ns", or
  3. "longer than X ns"

  Give the duration as precisely as possible, and explain it briefly.

Solution 4a

- There are two cases that we need to consider:
  - There are no memory accesses during the execution of program (i.e., the cache supplies all data and instructions), so memory accesses through the DMAC do not affect program execution. The execution of the program takes 1000 ns. This is the **best case**.
  - In the second case, there is no overlap between program execution and memory accesses using the DMAC. In this case, the execution time of the program is (time to transfer 10 words) + (execution time of the program without DMAC requests) = 2000 ns.
    - Time to transfer 10 words = $10 \cdot (50 + 50) = 1000$ ns
    - Execution time of the program without DMAC requests = 1000 ns
    - This is the **worst case**.
- Therefore, we can say that the execution of the program takes between 1000 ns and 2000 ns.

Question 4b

b) If we want to use a DMAC to transfer data between two memory modules, which type of DMAC should we use (flow-through (explicit) or fly-by (implicit))? Explain briefly.

Solution 4b

b) If we want to use a DMAC to transfer data between two memory modules, which type of DMAC should we use (flow-through (explicit) or fly-by (implicit))? Explain briefly.

- With fly-by, the DMAC provides a single address.
- Accessing two memory units requires specifying both destination and source addresses, and most of the time, it is very unlikely to have the same source and destination addresses.
- Therefore, **flow-through** DMAC accesses must be used.

Question 5

The instruction cycle of a CPU has 5 states with these durations:

1. Instruction Fetch (IF): **40** ns,
2. Operand (Register) Read (OR): **30** ns,
3. Execution (EX): **30** ns,
4. Operand Write (OW) : **30** ns,
5. Interrupt (IR) (if necessary): **200** ns.

- The CPU accesses memory in instruction fetch and operand write cycles but **not** in the operand read and execution cycles.
- The CPU enters the interrupt cycle only if an interrupt request is accepted.
- The housekeeping operations in the interrupt cycle take **200** ns.
- Memory access time and I/O interface access times are **50** ns each.
- There are four 3-wire (BR, BG, BGACK) DMACs ($DMAC_1$, $DMAC_2$, $DMAC_3$, and $DMAC_4$) which are connected to the 68000-like processor over a bus arbiter.
- The precedence order is $DMAC_1 > DMAC_2 > DMAC_3 > DMAC_4$.

Question 5

- For all four DMACs, the DMAC type is **flow-through (explicit)** (i.e., data passes through the DMAC).
- $DMAC_1$ and $DMAC_4$ are in **cycle-stealing** mode, and $DMAC_2$ and $DMAC_3$ are in **burst** mode.
- Assume that we start a clock (Clock = 0) when the CPU begins to run a program and **all** DMACs attempt to start transfer for **5 words** when Clock = **20** ns.

a) When (Clock = ?) will $DMAC_1$ complete the transfer of the **third word**? Why?

b) When (Clock = ?) will the CPU complete the **first instruction cycle**? Why?

c) When (Clock = ?) will $DMAC_4$ complete the transfer of **all 5 words**? Why?

Solution 5

Time (ns)	**CPU**	**DMAC$_1$**	**DMAC$_2$**	**DMAC$_3$**	**DMAC$_4$**
40	IF				
100 (CPU) 140 (DMAC)	OR + EX	1. word			
240 340 440 540 640			1. word 2. word 3. word 4. word 5. word		
740		2. word			
840 940 1040 1140 1240				1. word 2. word 3. word 4. word 5. word	
1340 **(answer of a)**		3. word			
1440					1.word
1540		4. word			

Solution 5

Time (ns)	**CPU**	**DMAC$_1$**	**DMAC$_2$**	**DMAC$_3$**	**DMAC$_4$**
1640					2. word
1740		5. word			
1840					3. word
1870 (answer of b)	OW				
1970					4. word
2010	IF				
2070 (CPU) **2110 (DMAC) (answer of c)**	OR + EX				5. Word