3.6 Monitoring and control systems

3.6.2 Bit manipulation to monitor and control devices

As you probably know by now, converting positive numbers into negative numbers (and vice versa) involves changing bits. As does converting a capital ASCII letter into a lower case ASCII letter. How exactly does a computer perform this action? The answer lies with bitwise operators:

Bitwise Operator	NOT (P`)	AND (P.Q)	OR (P + Q)	XOR
Description	Invert input	Where exactly two 1s	Where one or more 1s	Where exactly one 1
Input P	01001010	01001010	01001010	01001010
Input Q		11110000	11110000	11110000
Output	10110101	01000000	11111010	10111010

So how can we use these for useful tasks in a computer? If we look closer at the examples above, we can see that setting input 2 bits to 1s or 0s has a direct impact on the output. We'll call input 2 as a **mask**, and we apply this mask to change the values in input 1 in certain ways. Consider these questions about masks:

A **bitmask** is a way of accessing a particular bit. The bitmask is a number which has **0** in all bits that we don't care about, and a **1** for the bit(s) that we want to examine. By **AND**ing the bitmask with the original number, we can "extract" the bit(s) – if that bit was **0**, then the new number will be completely zero; if the bit was **1**, then the new number will be non-zero.

Consider these questions about Masks:

- If we have an AND mask bit as 0 what is the output in all cases?
- If we have an AND mask bit as 1 what is the output in all cases?
- If we have an OR mask bit as 0 what is the output in all cases?
- If we have an OR mask bit as 1 what is the output in all cases?
- If we have an XOR mask bit as 0 what is the output in all cases?
- If we have an XOR mask bit as 1 what is the output in all cases?

3.6.2 Bit manipulation to monitor and control devices

In summary:

	AND	OR	XOR
0	Clears the value	Retains the value	Retains the value
1	Retains the value	Sets the value	Inverts the value
uses	Setting chosen bits to 0	Setting chosen bits to 1	Inverting chosen bits, finding differences between bit sets

Example: Input data is **11011011**, we want an output with bits 3,4,5,6 set to 1, and bits 1 and 8 set to 0. We don’t care about the others.

(0?1111?0)

Solution: we can AND **011111110** followed by ORing **00111100**

Alternatively, we could **XOR 10100101**

Example: Input data is **11011011**, we want an output with bits 4 and 5 set to 0. We don’t care about the others.

(???00???)

Solution: By using an AND I can set bits to 0 and keep other bits. The mask I need is: **11100111**

Example: Input data is **11011011**, we want an output with bits 1 and 8 set to 1. We don’t care about the others.
(1??????1)

Solution: By using an OR, we can set bits to 1 and keep other bits as **0**. The mask we need is: **10000001**

The simplest form of Bit manipulation has been explained above. It has a number of applications in real life. They are used to monitor systems and make changes accordingly and they are used in the vending machines.

Here’s an example:

A vending machine has different items which it can dispense:

- Tea
- Coffee
- Chocolate
- Milk
- Sugar

3.6.2 Bit manipulation to monitor and control devices

Each button on the vending machine will give one item. This sort of system is known as an **embedded system.** A system which is programmed once and cannot be changed. Another example of an embedded system could be a microwave.

So according to the vending machine embedded system, each button could be used to alter one bit input to the vending machine.

- Bit 0: Tea
- Bit 1: Coffee
- Bit 2: Chocolate
- Bit 3: Milk
- Bit 4: Sugar

It is up to the user as to how many inputs he/she wishes to choose (or how many bits he/she wishes to alter.

If the user would like a coffee with no milk and sugar, the input would look as shown below.

Sugar	Milk	Chocolate	Coffee	Tea
4	3	2	1	0
1	1	0	1	0

The bits which were set to one are those button which the user pressed. The rest of the bits stay unchanged.

Similarly, if the user would like a tea with milk but no sugar, the input would look as shown below.

Sugar	Milk	Chocolate	Coffee	Tea
4	3	2	1	0
0	1	0	0	1

The bits which were set to one are those button which the user pressed. The rest of the bits stay unchanged.

Example:

A zoo needs to control the climate to accommodate the living conditions for different animals. The zoo uses a computerized system which in turn makes use of actuators. These actuators will operate devices which adjust the microclimate.

3.6.2 Bit manipulation to monitor and control devices

Actuators can be in 2 states, on or off. Whether an actuator is on or off is determined by a single bit value **(0 means off, 1 means on)** in a specific 8-bit memory location.

The actuators to control the climate in a certain Tank uses memory location 0804. Bit 5 of this memory location controls the heater.

7	6	**5**	4	3	2	1	0	bit number
0	0	1	1	0	1	0	1	value

3.6.2 Bit manipulation to monitor and control devices

Use some of the assembly language instructions to write the instructions that will ensure bit 5 of location 0804 is set to 1

Instruction		Explanation
Op Code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC
LDD	<address>	Direct addressing. Load the contents of the given address to ACC
STO	<address>	Store the contents of ACC at the given address
OUT		Output to the screen the character whose ASCII value is stored in ACC
AND	#n	Bitwise AND operation of the contents of ACC with the operand
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand
OR	#n	Bitwise OR operation of the contents of ACC with the operand

Solution:

- First, we need to find out which tank we want to make changes to. In this case, tank 4 which is address 0804 **so we have to load the address 0804** to the actuator.

- Then we need to make sure that bit 5 is switched on. (Turn it on if it is off and Keep it on if it is already on)
  We can use the **OR** operation for this task as it matches our requirement

  our mask will be **00100000**. The instruction is **OR #B00100000**

- Finally, we have to give this instruction to the actuator so that it can start to alter the temperature of the tank. In other words, we need to store the instruction in the actuator's memory and more specifically, we store it at address location 0804. The instruction is **STO 0804**.

  - **LDD 0804**
  - **OR #B00100000**
  - **STO 0804**