Assignment 1

Name: Darshit Jain
Roll Number: B19EE024

- **Programming language used:** C++
- **Category:** Interrupt drive I/O connection (fixed ISR location)
- **Input format:**
    - The user is expected to provide an input *(to trigger an interrupt)*
    - The **input** must be an **integer value**
- **Output format:**
    - At each step, the **Program Counter** is *displayed* along with the **instruction** being executed (**NOTE**: The **Program Counter** is **initialized** to **PC = 90** and the ISR is fixed at location PC = 16)
    - Since a **Logical Left Shift** operation is performed on the input data, the **output** displayed will be equal to the **input << 1** *(input logically left shifted by 1 bit)*
- **Resources:**
    - Multithreading input code snippet: https://stackoverflow.com/a/26128185

The input is taken on a thread different from the main thread (the main thread continues its execution), running **asynchronously**. ***task.wait_for(2s)*** blocks the thread until the specified duration *(2 seconds in this case)* has elapsed or the result becomes available, whichever comes first. This mechanism is the same as **Interrupt Polling** (periodically checking peripheral devices for any interrupts). ***std::future_status::ready*** checks if the task (input from the user) is finished or not. ***task.get()*** will return the result of the asynchronously run task.

- This modeling **completely mimics** the given Interrupt drive I/O connection with a fixed ISR location. The **input is handled asynchronously**, and hence, the **interrupt can occur anytime** (this is exactly what happens in actual devices).
- Before servicing the interrupt, the current microprocessor state, including the Program Counter, is stored in a stack and then the Program Counter moves to the fixed ISR location. The **output** is **stored** in the **peripheral 2 device**.
- Once the ISR is executed, the previous microprocessor state is restored and the processor resumes its pending execution.
- The program was written using the concept of **Object-oriented programming**.

Sample input/output is shown below:

1) When an input is given:

```
Interrupt Service Routine location in Program Memory: 16
Main application program location in Program Memory: 90
Initialized Program Counter to: 90

Connected Peripheral 1 to the processor
Connected Peripheral 2 to the processor

Initiated Program Memory with size: 110

Current Program Counter (PC): 90
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 91
Bitwise AND Instruction: dataMemory[address1] = dataMemory[address1] & dataMemory[address2]

Current Program Counter (PC): 92
Add Instruction: dataMemory[address1] = dataMemory[address1] + dataMemory[address2]

Current Program Counter (PC): 93
Jump Instruction: Program Counter (PC) = address1

Current Program Counter (PC): 94
Bitwise XOR Instruction: dataMemory[address1] = dataMemory[address1] ^ dataMemory[address2]

32
Received input from Periheral 1: 32

Storing previous state (Program Counter, Interrupt Flag and Register) in memory stack

Initiating interrupt

*** EXECUTING INTERRUPT SERVICE ROUTINE ***

Current Program Counter (PC): 16
Move Instruction: registerR0 = dataMemory[address1]

Current Program Counter (PC): 17
Logical Left Shift (by 1 bit) Instruction: registerR0 = registerR0 << 1

Current Program Counter (PC): 18
Move Instruction: dataMemory[address2] = registerR0
```

```
*** EXECUTING INTERRUPT SERVICE ROUTINE ***

Current Program Counter (PC): 16
Move Instruction: registerR0 = dataMemory[address1]

Current Program Counter (PC): 17
Logical Left Shift (by 1 bit) Instruction: registerR0 = registerR0 << 1

Current Program Counter (PC): 18
Move Instruction: dataMemory[address2] = registerR0

Current Program Counter (PC): 19
Halt Instruction: Stop current instructions execution


Output stored in Peripheral 2: 64

Restoring microprocessor state and continuing execution

Current Program Counter (PC): 95
Bitwise OR Instruction: dataMemory[address1] = dataMemory[address1] | dataMemory[address2]

Current Program Counter (PC): 96
Jump Instruction: Program Counter (PC) = address1

Current Program Counter (PC): 97
NOP Instruction: No Operation

Current Program Counter (PC): 98
Jump Instruction: Program Counter (PC) = address1

Current Program Counter (PC): 99
Add Instruction: dataMemory[address1] = dataMemory[address1] + dataMemory[address2]

Current Program Counter (PC): 100
Add Immediate Instruction: dataMemory[address1] = dataMemory[address1] + Immediate data

Current Program Counter (PC): 101
Logical Right Shift (by 1 bit) Instruction: dataMemory[address1] = dataMemory[address1] >> 1

Current Program Counter (PC): 102
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 103
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)
```

```
Current Program Counter (PC): 103
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 104
Add Immediate Instruction: dataMemory[address1] = dataMemory[address1] + Immediate data

Current Program Counter (PC): 105
Bitwise AND Instruction: dataMemory[address1] = dataMemory[address1] & dataMemory[address2]

Current Program Counter (PC): 106
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 107
NOP Instruction: No Operation

Current Program Counter (PC): 108
Move Instruction: dataMemory[address1] = dataMemory[address2]

Current Program Counter (PC): 109
Logical Left Shift (by 1 bit) Instruction: dataMemory[address1] = dataMemory[address1] << 1

Current Program Counter (PC): 110
Divide Instruction: dataMemory[address1] = dataMemory[address1] / dataMemory[address2]


***Program reached end of the Program Memory. Stopping execution!***
```

2) When no input is given:

```
Interrupt Service Routine location in Program Memory: 16
Main application program location in Program Memory: 90
Initialized Program Counter to: 90

Connected Peripheral 1 to the processor
Connected Peripheral 2 to the processor

Initiated Program Memory with size: 110

Current Program Counter (PC): 90
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 91
Bitwise AND Instruction: dataMemory[address1] = dataMemory[address1] & dataMemory[address2]

Current Program Counter (PC): 92
Add Instruction: dataMemory[address1] = dataMemory[address1] + dataMemory[address2]

Current Program Counter (PC): 93
Jump Instruction: Program Counter (PC) = address1

Current Program Counter (PC): 94
Bitwise XOR Instruction: dataMemory[address1] = dataMemory[address1] ^ dataMemory[address2]

Current Program Counter (PC): 95
Bitwise OR Instruction: dataMemory[address1] = dataMemory[address1] | dataMemory[address2]

Current Program Counter (PC): 96
Jump Instruction: Program Counter (PC) = address1

Current Program Counter (PC): 97
NOP Instruction: No Operation

Current Program Counter (PC): 98
Jump Instruction: Program Counter (PC) = address1

Current Program Counter (PC): 99
Add Instruction: dataMemory[address1] = dataMemory[address1] + dataMemory[address2]

Current Program Counter (PC): 100
Add Immediate Instruction: dataMemory[address1] = dataMemory[address1] + Immediate data

Current Program Counter (PC): 101
Logical Right Shift (by 1 bit) Instruction: dataMemory[address1] = dataMemory[address1] >> 1
```

```
Current Program Counter (PC): 101
Logical Right Shift (by 1 bit) Instruction: dataMemory[address1] = dataMemory[address1] >> 1

Current Program Counter (PC): 102
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 103
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 104
Add Immediate Instruction: dataMemory[address1] = dataMemory[address1] + Immediate data

Current Program Counter (PC): 105
Bitwise AND Instruction: dataMemory[address1] = dataMemory[address1] & dataMemory[address2]

Current Program Counter (PC): 106
Store Pogram Counter Instruction: dataMemory[address1] = PC (Program Counter)

Current Program Counter (PC): 107
NOP Instruction: No Operation

Current Program Counter (PC): 108
Move Instruction: dataMemory[address1] = dataMemory[address2]

Current Program Counter (PC): 109
Logical Left Shift (by 1 bit) Instruction: dataMemory[address1] = dataMemory[address1] << 1

Current Program Counter (PC): 110
Divide Instruction: dataMemory[address1] = dataMemory[address1] / dataMemory[address2]


***Program reached end of the Program Memory with no input received. Stopping execution!***
```