United International University

Department of CSE

Course Code: CSE 1326

Course Name: Digital Logic Design Laboratory

Experiment no: 06

Experiment Name: Implementing different kinds of registers to accomplish different functions.

Submitted by:

Name: Talha Jubayer

Student ID: 0112410062

Submitted to:

Fahim Hafiz [Lecturer, Dept. of CSE]

Date of Performance: 13-01-2025

Date of Submission: 19-01-2025

Objective:

This experiment focuses on implementing various types of registers, each designed to perform specific tasks. These include creating standard registers, registers capable of performing left-right shifts, multifunctional registers, and registers built using multiple individual registers.

Apparatus/Instruments used in the Lab:

- Trainer Board
- Power supply
- NOT Gate (74LS04)
- AND Gate (74LS08)
- X-OR gate IC (74LS86)
- X-NOR gate IC (74LS266)
- NAND gate (74LS00)
- Dual D flip-flop IC (74AHC74)
- Logisim

Theory:

When we provide input to the system, that input is stored in registers. Similarly, when the system returns results after processing, those results are also retrieved from the registers, enabling the CPU to utilize them for processing the data provided by the user.

Registers function based on three primary operations:

- **Fetch:** The Fetch Operation is used to retrieve user-provided instructions that have been stored in the main memory. Registers are used to fetch these instructions.

- **Decode:** The Decode Operation is used to interpret the Instructions, which means that the CPU will determine which Operation has to be carried out on the Instructions after the Instructions have been decoded.
- **Execute:** The CPU manages the Execute Operation. The results that the CPU generates are then stored in the memory before being presented on the user screen.

Types of Registers:

1. **Status and Control Register:** Status and Control registers report and allow modification of the state of the processor and the program being executed.
2. **General-Purpose Data Registers:** General purpose registers are extra registers in the CPU and are utilized anytime data or a memory location is required. These registers are used for storing operands and pointers. These are mainly used for holding the following:
    i. Operands for logical and arithmetic operations.
    ii. Operands for address calculation.
    iii. Memory pointers.

There are 3 types of **General-purpose** data registers:

1. **Data registers:** Data registers consist of four 32-bit data registers, which are used for arithmetic, logic and other operations.
    i. **AX:** This is referred to as the accumulator register. Its 16 bits are divided into two 8-bit registers, AH and AL, enabling it to execute both 16-bit and 8-bit instructions. In 8086 microprocessors, the accumulator register plays a key role in arithmetic, logic, and data transfer operations. For certain operations, such as multiplication and division, one of the operands must be stored in AX or AL.
    ii. **BX:** This is known as the base register. It is a 16-bit register divided into two 8-bit registers, BH and BL. The BX register functions as an address register and is commonly used as a data pointer for indirect addressing, which can be based, based-indexed, or register-based.
    iii. **CX:** This is referred to as the Count register. It is a 16-bit register divided into two 8-bit registers, CH and CL, enabling it to execute both 16-bit and 8-bit instructions. The Count register serves as a counter for loops, aiding in the creation of program loops. Additionally, it is used in shift/rotate instructions and string manipulation operations, where it functions as a counter.

iv. **DX:** This is known as the Data register. Its 16 bits are split into two 8-bit registers DH and DL so that it can execute 8-bit instructions as well. In I/O operations, the data register can be used as a port number. It is also applied to division and multiplication.

2. **Pointer registers:** The pointer registers consist of 16-bit left sections (SP and BP) and 32-bit ESP and EBP registers.
   i. **SP:** This is known as the Stack Pointer, which is used to indicate the location of the program stack. It works in conjunction with the Stack Segment (SS) register to access the stack segment. The Stack Pointer is 16 bits in size and points to the item at the top of the stack. If the stack is empty, the Stack Pointer will hold the value (FFFE)H. The stack segment address is determined relative to its offset address.
   ii. **BP:** This is known as the Base pointer used to point data in the stack segments. We can utilize BP to access data in the other segments. It has a 16-bit size. It mostly serves as a way to access parameters given via the stack. The stack segment is relative to its offset address.
3. **Index registers:** The 16-bit rightmost bits of the 32-bit ESI and EDI index registers.
   i. SI: This source index register is used to identify memory addresses in the data segment that DS is addressing. Therefore, it is simple to access successive memory locations when we increment the contents of SI. It has a 16-bit size. Relative to the data segment, it has an offset.
   ii. DI: The function of this destination index register is identical to that of SI. String operations are a subclass of instructions that employ DI to access the memory addresses specified by ES. It is generally used as a Destination index for string operations.

SI and DI are sometimes employed in addition and sometimes in subtraction as well as for indexed addressing.

4. **Special Purpose Registers:** Special purpose registers are employed to store machine state data and change state configuration. It is also defined as the CPU has several registers that are used to carry out instruction execution these registers are called special purpose registers. Special purpose registers are of 8 types:
   i. **CS (Code Segment register):** The Code Segment (CS) is a 16-bit register that holds the starting address of a 64 KB memory segment containing CPU instructions. The CPU uses the CS register in conjunction with the Instruction Pointer (IP) register to access

instructions. Direct modifications to the CS register are not allowed. Instead, the CS register is automatically updated during the execution of far jump, far call, and far return instructions.

ii. **DS (Data Segment register):** The Data Segment (DS) is a 16-bit register used to address a 64 KB segment of program data. By default, the processor assumes that all data referenced by general-purpose registers (AX, BX, CX, and DX) and index registers (SI and DI) resides in the data segment. The DS register can be directly modified using instructions such as POP and LDS.

iii. **SS (Stack Segment register):** The Stack Segment (SS) is a 16-bit register that holds the starting address of a 64 KB segment used for the software stack. By default, the CPU assumes that the stack segment contains all data referenced by the Stack Pointer (SP) and Base Pointer (BP) registers. The SS register can be directly modified using the POP instruction.

iv. **ES (Extra Segment register):** The Extra Segment (ES) is a 16-bit register that holds the starting address of a 64 KB segment, typically used for storing program data. In string manipulation instructions, the CPU assumes by default that the Destination Index (DI) register refers to the ES segment. The ES register can be directly updated using the POP and LES instructions.

v. **FS (File Segment register):** The FS register does not have a predetermined purpose assigned by the CPU; rather, its function is defined by the operating system that utilizes it. In Windows processes, the FS register is typically used to point to the Thread Information Block (TIB).

vi. **GS (Graphics Segment register):** The GS register in Windows 64-bit is utilized to point to structures defined by the operating system. It is commonly used by OS kernels to access memory specific to individual threads. In Windows, the GS register manages thread-specific memory. In contrast, the Linux kernel uses the GS register to access CPU-specific memory. Additionally, GS often serves as a pointer to Thread Local Storage (TLS).

The CS, DS, SS, ES, FS, and GS registers come under segment registers. These registers hold up to six segment selectors.

vii. **IP (Instruction Pointer register):** The 8086 microprocessor uses the CS and IP registers to retrieve instructions. The CS register holds the segment number of the next instruction, and the IP register stores the offset. After each instruction is executed, the IP is updated to indicate the next instruction's location. Unlike other registers, the IP cannot be directly altered by an instruction and cannot serve as an operand in any operation.

viii. **Flag register:** The FLAGS register serves as the status register for an x86 CPU, containing information about its current state. The size and significance of the flag bits can vary depending on the architecture. Typically, it includes details about the current operational limitations of the CPU and the results of arithmetic operations. Some limitations may prevent the execution of certain "privileged" instructions and inhibit specific interrupts from occurring. Other status flags can alter memory mapping and dictate the CPU's response in cases of arithmetic overflow.

➢ Shift Register

A shift register is a collection of flip-flops used to store multiple bits of data. The bits in these registers can be moved internally or shifted in and out by applying clock pulses. An N-bit shift register is created by linking N flip-flops, with each flip-flop storing one bit of data. Registers that shift bits to the left are known as "shift left registers," while those that shift bits to the right are called "shift right registers."

➢ Types of Shift Registers:

1. **Serial-In Serial-Out Shift Register (SISO):** A Serial-In Serial-Out (SISO) shift register allows for serial input, where one bit is entered at a time through a single data line, and it produces serial output. With only one output, data exits the shift register one bit at a time, which is why it's called a Serial-In Serial-Out shift register. The logic circuit shown represents a SISO shift register made up of four D flip-flops connected in series, all synchronized to the same clock signal.
2. **Serial-In Parallel-Out Shift Register (SIPO):** The shift register that allows serial input (one bit at a time through a single data line) and produces parallel output is known as a Serial-In Parallel-Out (SIPO) shift register. The logic circuit illustrated represents a SIPO shift register, consisting of four interconnected D flip-flops. In addition to the clock signal, a clear (CLR) signal is connected to all four flip-flops to reset them. The output of the first

flip-flop is linked to the input of the next flip-flop, and this continues sequentially. All these flip-flops operate synchronously, as they receive the same clock signal.

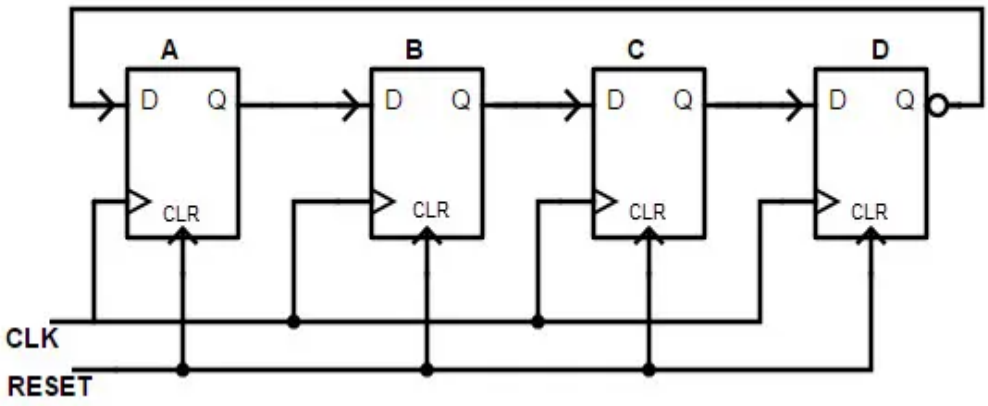
3. **Parallel-In Serial-Out Shift Register (PISO):** A Parallel-In Serial-Out (PISO) shift register allows for parallel input, where data is fed to each flip-flop simultaneously, and produces serial output. The logic circuit shown represents a PISO shift register made of four D flip-flops connected in series. The clock signal is supplied to all flip-flops, while the input data is directed to each flip-flop through a multiplexer. The multiplexer connects both the output from the previous flip-flop and the parallel input data to feed into the next flip-flop. All flip-flops operate synchronously, receiving the same clock signal. A PISO shift register is typically used to convert parallel data into serial form.
4. **Parallel-In Parallel-Out Shift Register (PIPO):** The shift register that allows parallel input (where data is provided separately and simultaneously to each flip-flop) and also produces parallel output is known as a Parallel-In Parallel-Out (PIPO) shift register. The logic circuit illustrated below represents a PIPO shift register, consisting of four interconnected D flip-flops. Both the clear (CLR) and clock signals are connected to all four flip-flops. In this type of register, there are no interconnections between the individual flip-flops, as no serial shifting of the data is required. Data is input separately to each flip-flop, and similarly, the output is collected individually from each flip-flop. A PIPO shift register is used as a temporary storage device and, like a Serial-In Serial-Out (SISO) shift register, it functions as a delay element.
5. **Bidirectional Shift Register:** Shifting a binary number one position to the left is equivalent to multiplying it by 2, while shifting it one position to the right is equivalent to dividing it by 2. To perform these operations, a register capable of shifting data in both directions is needed. Bidirectional shift registers are designed to shift data either left or right, depending on the selected mode. If the mode is set to 1 (high), the data shifts to the right; if the mode is set to 0 (low), the data shifts to the left. The logic circuit shown represents a bidirectional shift register made up of four interconnected D flip-flops. Input data is provided at both ends of the circuit, and only one gate is activated based on the selected mode.
6. **Universal Shift Register:** A Universal Shift Register is a type of register that supports both right and left shifting operations, along with parallel loading capabilities. These registers are commonly used as memory elements in

computers. However, a limitation of this type of register is that it shifts in only one direction at a time. In essence, a universal shift register combines the features of both bidirectional and unidirectional shift registers. An N-bit universal shift register consists of N flip-flops and N multiplexers. All N multiplexers share the same select lines, allowing the select input to determine the appropriate input for the flip-flops.

7. **Shift Register Counter:** Shift Register Counters are the shift registers in which the outputs are connected back to the inputs to produce particular sequences.

   There are two types:

   i. **Ring Counter:** A ring counter is a type of shift register counter where the output of the first flip-flop is connected to the input of the next flip-flop, and the output of the last flip-flop is fed back to the input of the first flip-flop, forming a "ring." As long as clock pulses are applied, the data pattern within the shift register will circulate. The provided logic circuit and truth table demonstrate a Ring Counter:



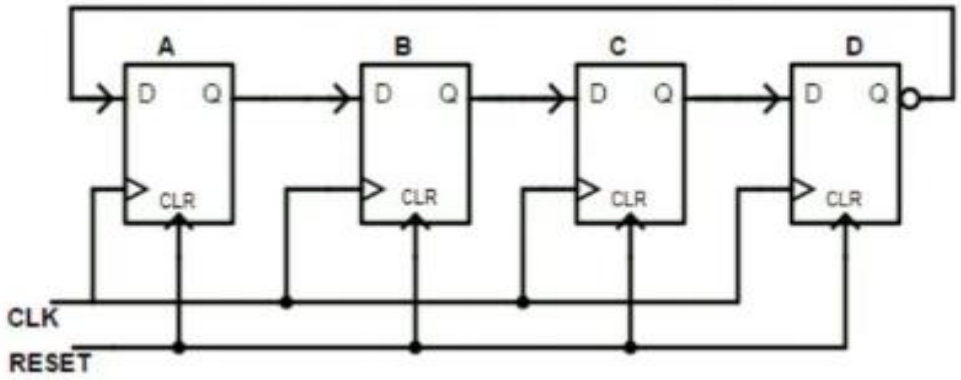
Clock Input	Q3	Q2	Q1	Q0
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0
5	0	0	0	1
6	0	0	1	0
7	0	1	0	0
8	1	0	0	0

The circuit consists of four interconnected D flip-flops. Since there are four flip-flops, the data pattern will repeat every four clock pulses, as shown in the truth table. A ring counter is commonly used because it is self-decoding, meaning no additional decoding circuit is

required to identify the counter's current state.

ii. **Johnson Counter:** A Johnson counter is a shift register counter in which the output of the first flip-flop is connected to the next flip-flop and so on and the inverted output of the last flip-flop is again fed back to the input of the first flip-flop. They are also known as twisted ring counters.

The logic circuit & truth table given below shows a Johnson Counter:



The circuit consists of four D flip-flops which are connected. An n-stage Johnson counter yields a count sequence of 2n different states, thus also known as a mod-2n counter. Since the circuit consists of four flip-flops the data pattern will repeat every eight clock pulses as shown in the truth table. The main advantage of the Johnson counter is that it only needs no number of flip-flops compared to the ring counter to circulate given data to generate a sequence of 2n states.

Clock Input	Q3	Q2	Q1	Q0
1	0	0	0	0
2	0	0	0	1
3	0	0	1	1
4	0	1	1	1
5	1	1	1	1
6	1	1	1	0
7	1	1	0	0
8	1	0	0	0

Solution of the mentioned problems:

1) Construct a 3-bit register:

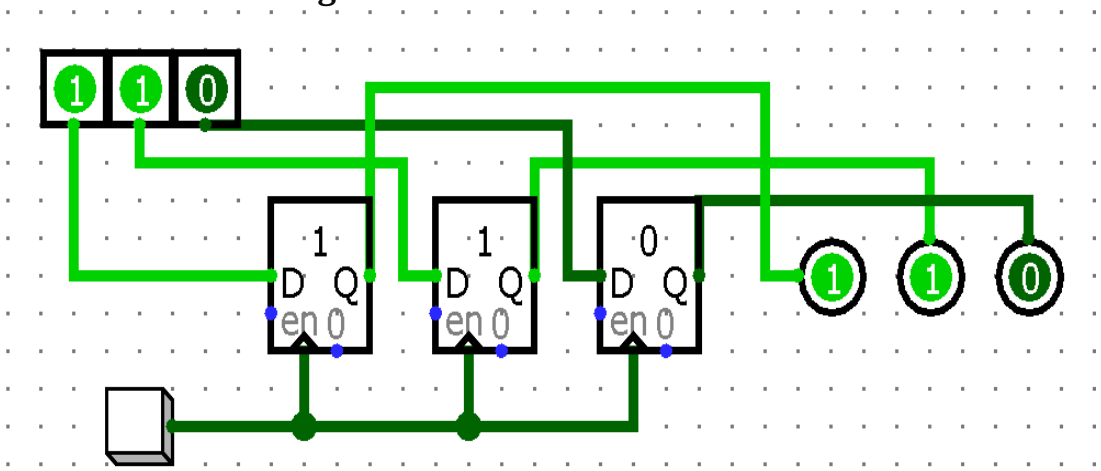


Figure 1: 3 bit register

2) Select between two 4-bit registers using MUX:

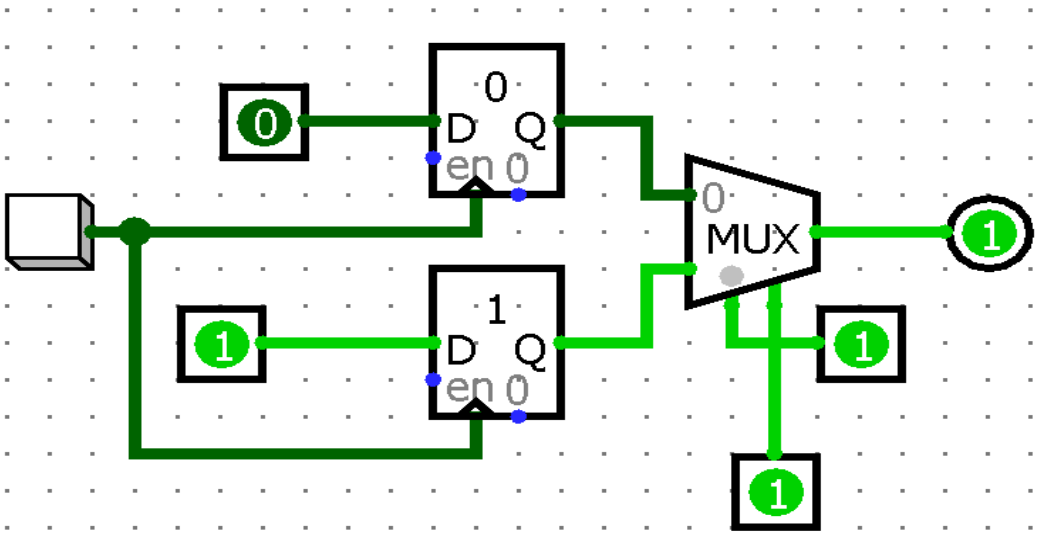


Figure 2: Two 4-bit Register using MUX

3) Construct a right shift register:

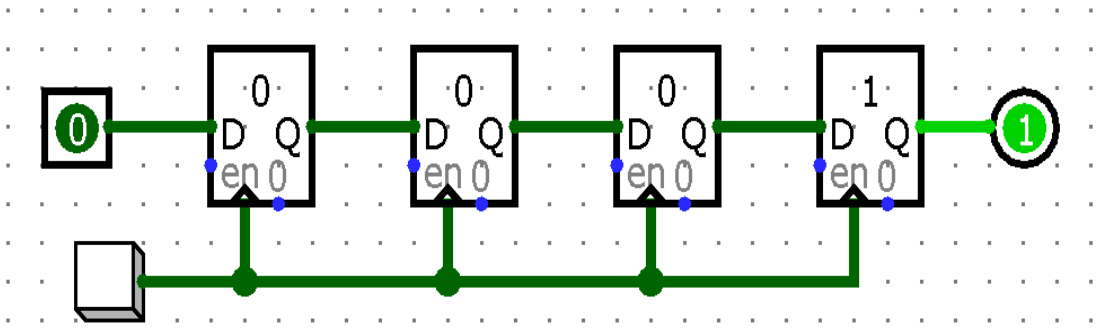


Figure 3: Right Shift Register

4) Construct a left shift register:

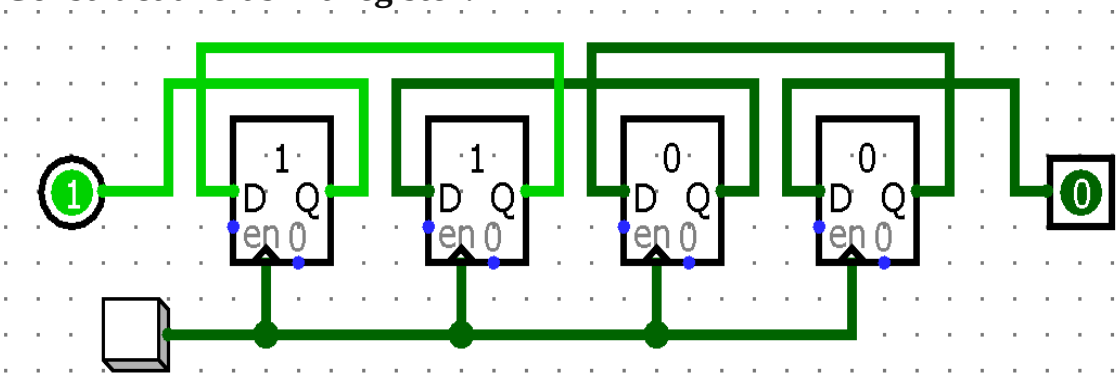
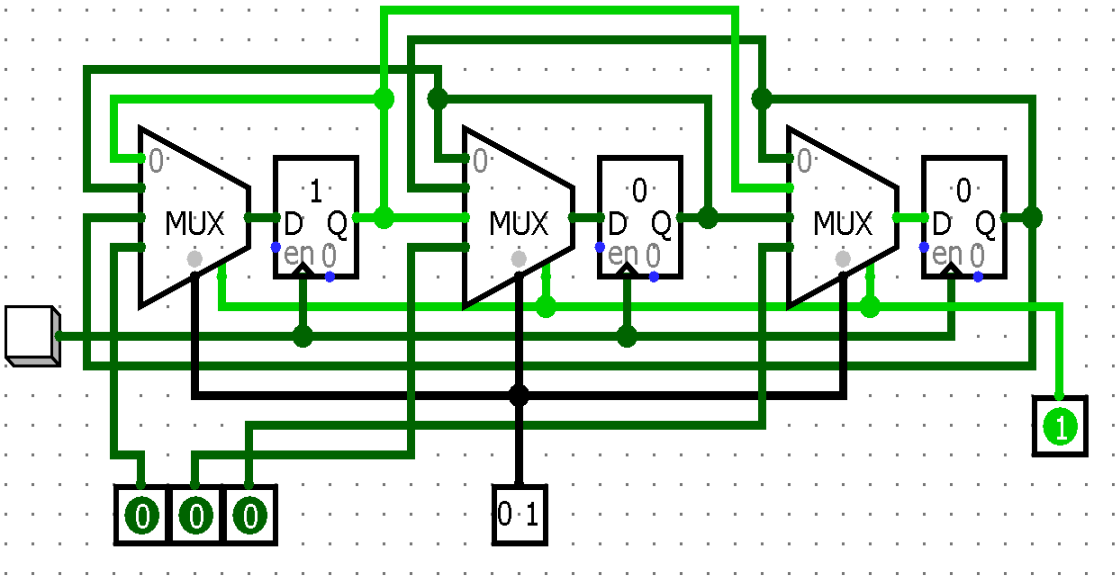


Figure 4: Left Shift Register

5) Construct a 3-bit register with the functions given in the function table:



Discussion:

After completing the laboratory experiment on Registers, I gained an understanding of how registers function as small memory cells used to temporarily store data within a computer's central processing unit (CPU). During the experiment, we observed the operation of registers and learned about the various types and their specific functions. One of the key insights from the experiment was the critical role registers play in enhancing computer performance. By storing data in registers instead of retrieving it from memory, the computer can execute instructions much more quickly, leading to significant improvements in processing speed. Overall, the experiment provided valuable insight into the vital role registers have in computer architecture.