SAE2B
DIGITAL ELECTRONICS & MICROPROCESSOR
Unit : I - V
Unit : I – Overview

- Number System
- Binary System
- Binary Code
- Logic Gates
- Boolean Algebra
- Truth Tables
- Universal Gates
- Simplification of Boolean functions
- Karnaugh Map
- Combinational Logic
Number System

- Number System is a system of representing letters or numbers in computer understandable form.
- Examples – Binary, Decimal, Octal and Hexadecimal

<table>
<thead>
<tr>
<th>Number System</th>
<th>Base</th>
<th>Digits or Symbols included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>2</td>
<td>0,1</td>
</tr>
<tr>
<td>Decimal</td>
<td>10</td>
<td>0,1,2,3,4,5,6,7,8,9</td>
</tr>
<tr>
<td>Octal</td>
<td>8</td>
<td>0,1,2,3,4,5,6,7</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>16</td>
<td>0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F</td>
</tr>
</tbody>
</table>
Binary System

A binary number system is made up of only 0s and 1s. Example : 001010

Code Conversion

• Conversion from any Number system to Decimal
• Conversion from decimal to other system
• Conversion from Binary to Octal / Hexadecimal
• Conversion from Octal / Hexadecimal to binary
• Conversion from Decimal to other system
Binary Codes

- Codes are alternate representations for the binary numbers.
- Different codes are used for binary numbers. Commonly used are
  1. BCD (Binary Coded Decimal) Code
     a) 8421
     b) 2421
     c) 4221
  2. Excess-3 Code
  3. Gray Code
Logic Gates

**Symbol set 1**

- **AND**
  \[ a \land b \]

- **OR**
  \[ a \lor b \]

- **NOT**
  \[ \overline{a} \]

- **NAND**
  \[ \overline{a \land b} \]

- **NOR**
  \[ \overline{a \lor b} \]

- **EXCLUSIVE OR**
  \[ a \oplus b \]

**Symbol set 2**


- **AND**
  \[ a \& b \]

- **OR**
  \[ a \lor b \]

- **NOT**
  \[ \overline{a} \]

- **NAND**
  \[ \overline{a \& b} \]

- **NOR**
  \[ \overline{a \lor b} \]

- **EXCLUSIVE OR**
  \[ a \oplus b \]
Boolean Algebra

◆ Commutative Law
  \[ A + B = B + A \]
  \[ A \cdot B = B \cdot A \]

◆ Associative Law
  \[ A + (B + C) = (A + B) + C \]
  \[ A (BC) = (AB) C \]

◆ Distributive Law
  \[ A (B + C) = AB + AC \]
Rules specific to Boolean Algebra

1a. \( A + 0 = A \) 
2a. \( A + 1 = 1 \) 
3a. \( A + A = A \) 
4a. \( A + A' = 1 \) 
5a. \( A'' = A \) 
6a. \( A + AB = A \) 
7a. \( A + A'B = A + B \) 
8a. \( A + BC = (A + B) (A + C) \) 
9a. \( AB + A'C + BC = AB + A'C \)

1b. \( A \cdot 1 = A \) 
2b. \( A \cdot 0 = 0 \) 
3b. \( A \cdot A = A \) 
4b. \( A \cdot A' = 0 \) 
5b. \( A'' = A \) 
6b. \( A (A + B) = A \) 
7b. \( A (A' + B) = AB \) 
8b. \( A (B + C) = AB + AC \) 
9b. \( (A+B) (A'+C) (B+C) = (A+B) (A' + C) \)
Demorgan’s Laws

• The AND and OR functions can be shown to be related to each other through the following equations.

Theorem 1:
“The complement of a product is equal to the sum of individual complements.” In other words,

\[(AB)' = A' + B'\]
or
NAND = Bubbled OR

Theorem 2:
“The complement of a sum is equal to the product of individual complements.” In other words,

\[(A + B)' = A' \cdot B'\]
or
NOR = Bubbled AND
A truth table is a mathematical table used in logic—specifically in connection with Boolean algebra and Boolean functions - which sets out the functional values of logical expressions on each of their functional arguments, that is, for each combination of values taken by their logical variables.
Universal Gates

Universal gates are the ones which can be used for implementing any gate like AND, OR and NOT, or any combination of these basic gates.

NAND and NOR gates are universal gates.

But there are some rules that need to be followed when implementing NAND or NOR based gates.
Sum of Products and Product of Sums

Any given truth table can be converted into a logical expression, by either SOP or POS method.

• To obtain SOP expression,
  a) Take the cases where output is a logical 1
  b) Represent each case as a product of the variables, such that output is 1. This product is known as a minterm.
  c) ORing the minterms gives us the SOP expression.

• To obtain POS expression,
  a) Take the cases where output is a logical 0.
  b) Represent each case as a sum of the variables, such that output is 0. This product is known as a maxterm.
  c) ANDing the maxterms gives us the POS expression.
Karnaugh Maps (K-Maps)

• Karnaugh maps provide a systematic method to obtain simplified sum-of-products (SOPs) Boolean expressions.

• This is a compact way of representing a truth table and is a technique that is used to simplify logic expressions.

• It is ideally suited for four or less variables, becoming cumbersome for five or more variables.

• A K-map of n variables will have $2^n$ squares.

• Each square represents either a minterm or maxterm.

• For a Boolean expression, product terms are denoted by 1's, while sum terms are denoted by 0's - but 0's are often left blank.
A combinational circuit can have an \( n \) number of inputs and \( m \) number of outputs.

Some of the **Combinational circuits** are
1. Half Adder
2. Full Adder
3. Half Subtractor
4. Full subtractor
Combinational Logic Design

Combinational Logic Design also holds
1. Decoder
2. Encoder
3. Multiplexer
4. Demultiplexer
Unit : II – Overview

- Sequential Logic
- Flip-Flops
- Shift Register
- Counters
Sequential Logic

- The outputs of a sequential logic circuit depend on both the current inputs and on previous inputs and outputs of the circuit.

- Sequential elements have storage elements that record the state of the circuit. In other words, the state information combined with the inputs is generating the outputs.

- The state and inputs also combine to generate a new state of the circuit.

- The same inputs in a sequential circuit may generate different outputs and different new states, depending on the circuit’s current state.
Flip-Flops

- A bi-stable device i.e. a circuit with only 2 stable states, namely the ‘0’ state and the ‘1’ state.
- Ability to retain its state and store a bit of information.
- It is one-bit memory cell.
- A flip-flop has 2 outputs and they complement each other.
A common form of register used in computers and in many other types of logic circuits is a shift register.

It is simply a set of flip flops (usually D latches or RS flip-flops) connected together so that the output of one becomes the input of the next, and so on in series.

It is called a shift register because the data is shifted through the register by one bit position on each clock pulse.
Serial-in Serial-out Register

- On the leading edge of the first clock pulse, the signal on the D input is latched in the first flip flop.

- On the leading edge of the next clock pulse, the contents of the first flip-flop is stored in the second flip-flop, and the signal which is present at the D input is stored in the first flip-flop, etc.

- Because the data is entered one bit at a time, this called a serial-in shift register. Since there is only one output, and data leaves the shift register one bit at a time, then it is also a serial out shift register.
Parallel-in Parallel-out Register

- Parallel input can be provided through the use of the preset and clear inputs to the flip-flop.
- The parallel loading of the flip-flop can be synchronous (i.e., occurs with the clock pulse) or asynchronous (independent of the clock pulse) depending on the design of the shift register.
- Parallel output can be obtained from the outputs of each flip-flop as shown in Figure.
Counters

- Counter is a register which counts the sequence in binary form.
- The state of counter changes with application of clock pulse.
- The counter is binary or non-binary.
- The total number of states in counter is called as modulus.
- If counter is modulus-n, then it has n different states.
- State diagram of counter is a pictorial representation of counter states directed by arrows in graph.
Asynchronous (Ripple) Counters

- All Flip-Flops are in toggle mode.
- The clock input is applied.
- Count enable = 1.
- Counter counts from 0000 to 1111.
Synchronous Counter

In *synchronous counters*, the clock inputs of all the flip-flops are connected together and are triggered by the input pulses. Thus, all the flip-flops change state simultaneously (in parallel).

- The J and K inputs of FF0 are connected to HIGH. FF1 has its J and K inputs connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate that is fed by the outputs of FF0 and FF1.
- After the 3rd clock pulse, both outputs of FF0 and FF1 are HIGH. The positive edge of the 4th clock pulse will cause FF2 to change its state due to the AND gate.
Up / Down Counter

Bidirectional counters, also known as Up/Down counters, are capable of counting in either direction through any given count sequence and they can be reversed at any point within their count sequence by using an additional control input.
Unit : II – Overview

- Microprocessors
- Microprocessor Architecture
- Peripheral/Externally Initiated Operations
- Memory and its classification
- 8085 Instruction Set
- Addressing Modes
Microprocessors

- Multipurpose, clock-driven, register-based electronic device.
- Reads binary instructions from memory.
- Accepts data as input.
- Process data according to instructions.
- Provides result as output.
Micro Computer

- A microcomputer is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU).
- It includes a microprocessor, memory, and minimal input/output (I/O) circuitry mounted on a single printed circuit board.

Assembly Language

- An assembly language is a low-level programming language for microprocessors and other programmable devices.
- An assembly language implements a symbolic representation of the machine code needed to program a given CPU architecture.
- Assembly language is also known as assembly code.
Microprocessor Architecture

- Microprocessor is digital device designed with
  - Register
  - Flip-flop
  - Timing element
8085 Bus Structure
Address Bus

- Group of 16 lines generally identified as A0 to A15.
- It is unidirectional (bits flow in one direction).
- Identifies the peripherals or a memory location through these line.
- Carry a 16-bit address.
- Capable of identifying $2^{16} = 65,536$ (64K) memory locations.
Data Bus

- Group of 8 lines used for data flow (D0 to D7)

- Bidirectional: data flow both direction between MPU and memory and peripherals.

- The largest number that can appear on the data bus is 1111 1111 (i.e. 25510)

- Handles up to 2^8=256 (i.e. 00 to FF) numbers.
Control Bus

➢ Comprised of various signal lines that carry synchronization signals.

➢ MPU generates specific control signal for every operation.

➢ Used to identify the device type which MPU intends to communicate.

➢ Eg: To read data from the memory MPU sends the control signal called Memory Read.
8085 Pin Diagram and Signals
Classifications of the functions

1. Microprocessor-initiated operations.
2. Internal operations.
3. Peripheral (or externally) initiated operations.
Internal Data Operations

- Internal architecture of the 8085 microprocessor determines how and what operations can be performed.

- The operations are:
  - Store 8-bit data.
  - Perform arithmetic and logical operations.
  - Test for conditions.
  - Sequence the execution of instruction.
  - Store data temporarily in the Stack.
Peripheral/Externally Initiated Operations

- External devices can initiate the MPU operations.

- Individual pins on the MPU chip are assigned for various operations like:
  - Reset
  - Interrupt
  - Ready
  - Hold
Memory and its classification

- Two basic categories of computer memory:
  - Primary stores small amounts of data and information that will be immediately used by the CPU.
  - Secondary stores much larger amounts of data and information (an entire software program, for example) for extended periods of time.
Memory and Instruction Fetch

- All instructions are stored in memory.

- To run a program, the individual instructions must be read from the memory in sequence, and executed.
  - Instruction fetch
  - Decode instruction
  - Get operands
  - Execute operation
Instruction Fetch Operation
8085 Instruction Set

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.

- The entire group of instructions that a microprocessor supports is called Instruction Set.

- 8085 has 246 instructions.

- Each instruction is represented by an 8-bit binary value.

- These 8-bits of binary value is called Op-Code or Instruction Byte.
Addressing Modes

- Every instruction has to operate on a data.

- The method of specifying the data to be operated by the instruction is called Addressing.

- The 8085 has 5 types of addressing:
  1. Immediate Addressing
  2. Direct Addressing
  3. Register Addressing
  4. Register Indirect Addressing.
  5. Implied Addressing
Arithmetic Operations

The 8085 microprocessor performs various arithmetic operations, such as addition, subtraction, increment, and decrement.

1. ADD
2. ADI
3. SUB
4. SUI
5. INR
6. DCR
Logical Operations

- Microprocessor is basically a programmable logic chip.

- It can perform all the logic functions of the hard-wired logic through its instruction set.

- They are:
  - AND
  - OR
  - Ex OR
  - NOT
Branching Instructions

- Most powerful instructions because they allow the microprocessor to change the sequence of a program.

- Change may be unconditional or under certain test conditions.

- Instruct the microprocessor to go to a different memory location.

- Types:
  1. Jump Instructions.
  2. Call and Return instructions.
  3. Restart instructions.
Jump Instructions

- Specify the memory location explicitly.
- They are 3-byte instructions.
- One byte for operation code, followed by a 16-bit memory address.
- Classified into:
  1. Unconditional Jump.
  2. Conditional Jump.
Unconditional Jump

JMP 16-bit address:

- Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

Example: JMP 2034H or JMP XYZ(Label name)
Conditional Jumps

- Allow the microprocessor to make decisions based on certain conditions indicated by flags.

- Check the flag conditions to change or not.

- Flags used by jump instructions:
  1. Carry flag
  2. Zero flag
  3. Sign flag
  4. Parity flag
Unit : IV – Overview

- Time Delay Using One Register
- Time Delay Using a Register Pair
- Using a Loop within Loop Technique
- Counter Design with Time Delay
- Stack and Subroutines
- BCD to Binary Conversion and Vice-versa
- BCD to HEX Conversion and Vice-versa
- Binary to ASCII Conversion and Vice-versa
- BCD Addition and Subtraction
Time Delay

- Procedure used to design a specific delay.
- A register is loaded with a number, depending on the time delay required and then the register is decremented until it reaches zero by setting up a loop with conditional jump instruction.

- Time delay using
  One register:
## Time Delay using Register Pair

<table>
<thead>
<tr>
<th>Label</th>
<th>Opcode</th>
<th>Operand</th>
<th>Comments</th>
<th>T state</th>
</tr>
</thead>
<tbody>
<tr>
<td>LXI</td>
<td>B,2384H</td>
<td></td>
<td>Load BC with 16-bit count</td>
<td>10</td>
</tr>
<tr>
<td>LOOP:</td>
<td>DCX</td>
<td>B</td>
<td>Decrement BC by 1</td>
<td>6</td>
</tr>
<tr>
<td>MOV</td>
<td>A,C</td>
<td></td>
<td>Place contents of C in A</td>
<td>4</td>
</tr>
<tr>
<td>ORA</td>
<td>B</td>
<td></td>
<td>OR B with C to set Zero flag</td>
<td>4</td>
</tr>
<tr>
<td>JNZ</td>
<td>LOOP</td>
<td></td>
<td>if result not equal to 0 , jump back to loop</td>
<td>10/7</td>
</tr>
</tbody>
</table>

Time Delay in Loop \( TL = T \times \text{Loop T states} \times N10 \)

\[
= 0.5 \times 24 \times 9092 \\
= 109 \text{ ms}
\]

## Time Delay using LOOP within a LOOP

<table>
<thead>
<tr>
<th>Label</th>
<th>Opcode</th>
<th>Operand</th>
<th>Comments</th>
<th>T state</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVI</td>
<td>B,38H</td>
<td></td>
<td>Delay in Loop ( TL1 = 1783.5 ) μs</td>
<td></td>
</tr>
<tr>
<td>LOOP2:</td>
<td>MVI</td>
<td>C,FFH</td>
<td>Delay in Loop ( TL2 = (0.5\times21+TL1)\times56 )</td>
<td>100.46ms</td>
</tr>
<tr>
<td>LOOP1:</td>
<td>DCR</td>
<td>C</td>
<td></td>
<td>4T</td>
</tr>
<tr>
<td></td>
<td>JNZ</td>
<td>LOOP1</td>
<td></td>
<td>10/7T</td>
</tr>
<tr>
<td></td>
<td>DCR</td>
<td>B</td>
<td></td>
<td>4T</td>
</tr>
<tr>
<td></td>
<td>JNZ</td>
<td>LOOP2</td>
<td></td>
<td>10/7T</td>
</tr>
</tbody>
</table>
Flowchart of a counter with time delay

- Initialize
- Display
- Update the count
- Is this Final Count?
  - No
  - Delay
  - Initialize Register
  - Decrement Delay register
  - Is Delay register = 0?
    - No
      - Delay register ≠ 0
    - Yes
      - RET
  - Yes

The Stack

- STACK is a group of memory location in the R/W memory that is used for temporary storage of binary information during the execution of a program.

- The programmer can store and retrieve the contents of a register pair by using PUSH and POP.
SUBROUTINE

- A subroutine is a group of instructions that will be used repeatedly in different locations of the program.

  - Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.

  - Instructions used in subroutine are CALL, RET, RTE and RST.
BCD - TO - BINARY CONVERSION

Example: $72_{10} = 01110010_{BCD}$

**SOLUTION:**

- **Step 1:** 0111 0010
  - $\rightarrow$ 0000 0010 Unpacked BCD$_1$.
  - $\rightarrow$ 0000 0111 Unpacked BCD$_2$.

- **Step 2:** Multiply BCD$_2$ by 10 = (7x10)

- **Step 3:** Add BCD$_1$ to the answer in step 2
BCD - TO - BINARY CONVERSION

Example: \(72_{10} = 01110010_{BCD}\)

**SOLUTION:**

- **Step 1:** 0111 0010
  - \(\rightarrow 0000 0010\) Unpacked BCD\(_1\).
  - \(\rightarrow 0000 0111\) Unpacked BCD\(_2\).

- **Step 2:** Multiply BCD\(_2\) by 10 = \((7 \times 10)\)

- **Step 3:** Add BCD\(_1\) to the answer in step 2
BINARY-TO-BCD CONVERSION

Example: Assume the binary number is

1111 1111 \(2(\text{FFH})=255_{10}\)

To represent this number in BCD requires twelve bits or three BCD digits, labeled here as \(\text{BCD}_3\) (MSB), \(\text{BCD}_2\) and \(\text{BCD}_1\) (LSB).

\[=0010 \ 0101 \ 0101\]

\(\text{BCD}_3 \ \text{BCD}_2 \ \text{BCD}_1\)
BCD to HEX conversion

- Initialize memory pointer to 4150H.
- Get the most significant Digit (MSD)
- Multiply the MSD by ten using repeated addition
- Add the least significant digit (LSD) to the result obtained in previous step.
- Store hex data in memory.

- Input: 4150:02 (MSD), 4151:09 (LSD)
- Output: 4152:1DH
HEX to BCD Conversion

- Initialize memory pointer to 4150H.
- Get the hexadecimal number.
- Perform repeated addition for n number of times.
- Adjust for BCD in each step.
- Store BCD data in memory.

Input: 4150:FF

Output: 4151:55 (LSD)
         4152:02 (MSD)
BINARY TO ASCII

LDA 2050H : Take the binary number in the accumulator
CPA 0AH   : Compare the given number with 0AH
JC SKIP 7 : If number < 9 no need to add 7
ADI 07H   : Else, add 7 to the accumulator
SKIP 7 : ADI 30H : Add 30h to accumulator
STA 2051H : Store the ASCII result in memory
HLT       : End the program
BCD ADDITION

- The 8085 provides a special instruction DAA (decimal adjust accumulator) to perform BCD addition.

- The DAA instruction is included immediately after an addition or increment instruction.

- The maximum in two digit BCD (8 bits) is 99.
BCD Subtraction

- The DAA cannot be used directly to perform BCD subtraction because DAA instruction requires an addition to be performed first.
- So, 10’s complement method is employed.

Example: 85 - 39

9’s complement of 39 = 99 - 39 = 60
(each digit is subtracted from 9)

10’s complement of 39 = 9’s complement + 1 = 60 + 1 = 61

Therefore 85 + 61 = 46 (with carry 1 (should be omitted))
Unit : V – Overview

- Interrupt
- Vectored Interrupts
- Interfacing I/O Devices
- Basic Interfacing Concepts
- DMA
Interrupts

- Interrupt is a process where an external device can get the attention of the microprocessor.
  - The process starts from the I/O device
  - The process is asynchronous.

TYPES OF INTERRUPT

SOFTWARE

HARDWARE

VECTORED AND NON VECTORED
### The 8085 Interrupts

<table>
<thead>
<tr>
<th>Interrupt name</th>
<th>Maskable</th>
<th>Vectored</th>
<th>VECTOR ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAP</td>
<td>No</td>
<td>Yes</td>
<td>0024H</td>
</tr>
<tr>
<td>RST 7.5</td>
<td>Yes</td>
<td>Yes</td>
<td>003CH</td>
</tr>
<tr>
<td>RST 6.5</td>
<td>Yes</td>
<td>Yes</td>
<td>0034H</td>
</tr>
<tr>
<td>RST 5.5</td>
<td>Yes</td>
<td>Yes</td>
<td>002CH</td>
</tr>
<tr>
<td>INTR</td>
<td>Yes</td>
<td>No</td>
<td>--</td>
</tr>
</tbody>
</table>
8085 Interrupts
The 8085 Vectored Interrupt Process

In vectored interrupts, the processor automatically branches to the specific address in response to an interrupt.
The 8085 Non-Vectored Interrupt Process

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If INTR is high, MP completes current instruction, disables the interrupt and sends INTA (Interrupt acknowledge) signal to the device that interrupted.
4. INTA allows the I/O device to send a RST instruction through data bus.
5. Upon receiving the INTA signal, MP saves the memory location of the next instruction on the stack and the program is transferred to ‘call’ location (ISR Call) specified by the RST instruction.
The 8085 Non-Vectored Interrupt Process

6. Microprocessor Performs the ISR.
7. ISR must include the ‘EI’ instruction to enable the further interrupt within the program.
8. RET instruction at the end of the ISR allows the MP to retrieve the return address from the stack and the program is transferred back to where the program was interrupted.
Basic Interfacing Concepts

**Interface** is the path for communication between two components. Interfacing is of two types, memory interfacing and I/O interfacing.
Direct Memory Access

- Process of communication or data transfer controlled by an external peripheral.

- Ex: Data transfer between a floppy and R/W memory of the system.

- 8085A has two pins for this type of communication
  - HOLD
  - HLDA
Direct Memory Access

CPU sends a **starting address**, direction (R/W), and word count to DMAC. Then issues "start".

DMAC provides;
- Peripheral controller ← Handshake signals
- Memory ← Addresses
- Memory ← Handshake signals

**Time to do 1000 xfers in 1 msec:**

1 DMA set-up sequence: @ 50 μsec
1 interrupt: @ 2 μsec
1 interrupt service sequence: @ 48 μsec

100 μsec

.0001 second of CPU time

---

**Diagram:**

- CPU
- Memory
- DMAC
- IOC
- I/O
- Device

**Memory Mapped I/O**

- ROM
- RAM
- Peripherals
- DMA