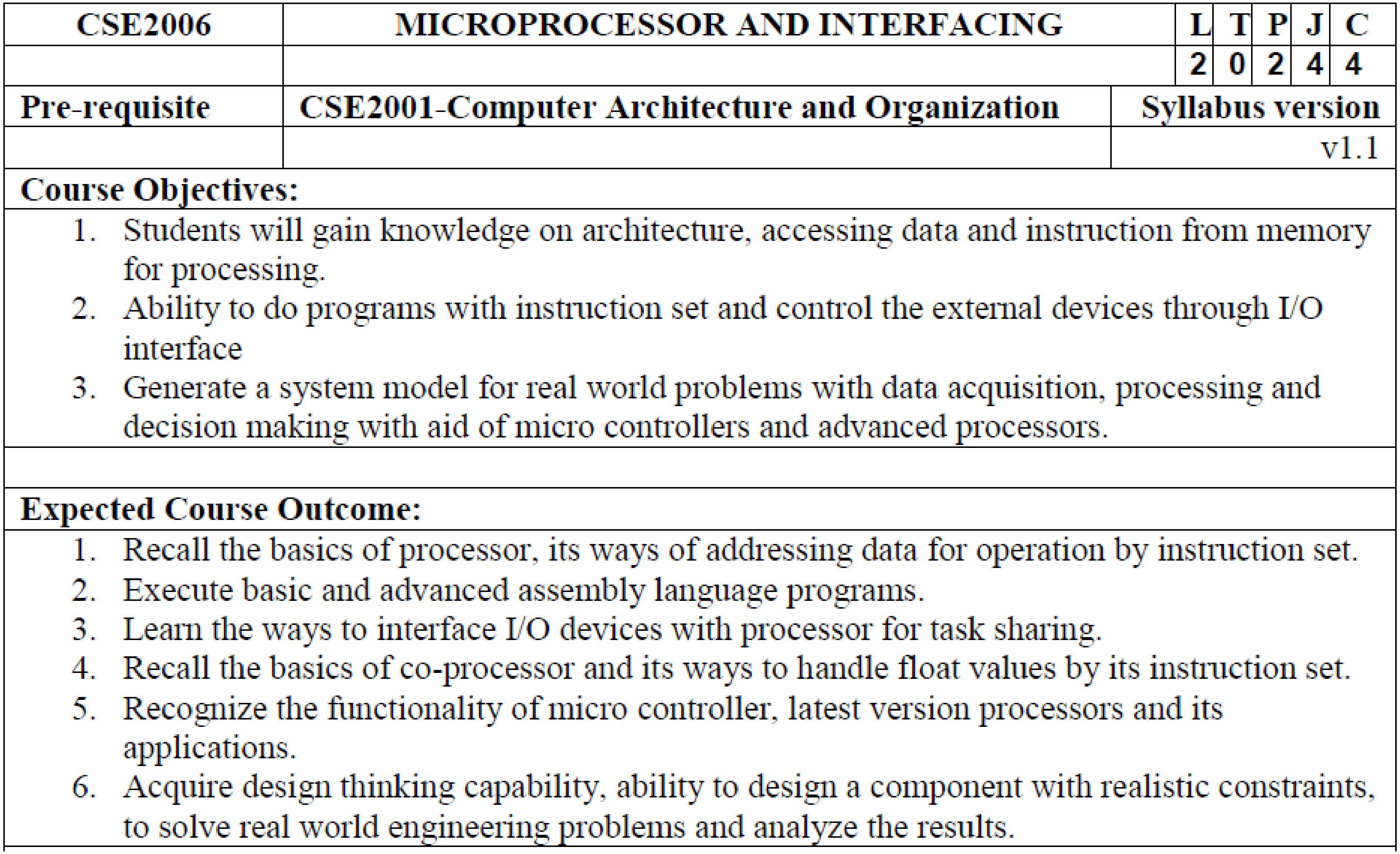
# CSE2006 Microprocessor & Interfacing

## Module – 2
## Introduction to ALP

**Dr. E. Konguvel**
Assistant Professor (Sr. Gr. 1),
Dept. of Embedded Technology,
School of Electronics Engineering (SENSE),
konguvel.e@vit.ac.in
9597812810

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

# Syllabus



| CSE2006 | MICROPROCESSOR AND INTERFACING | L | T | P | J | C |
|---|---|---|---|---|---|---|
| | | 2 | 0 | 2 | 4 | 4 |
| Pre-requisite | CSE2001-Computer Architecture and Organization | Syllabus version | | | | |
| | | v1.1 | | | | |

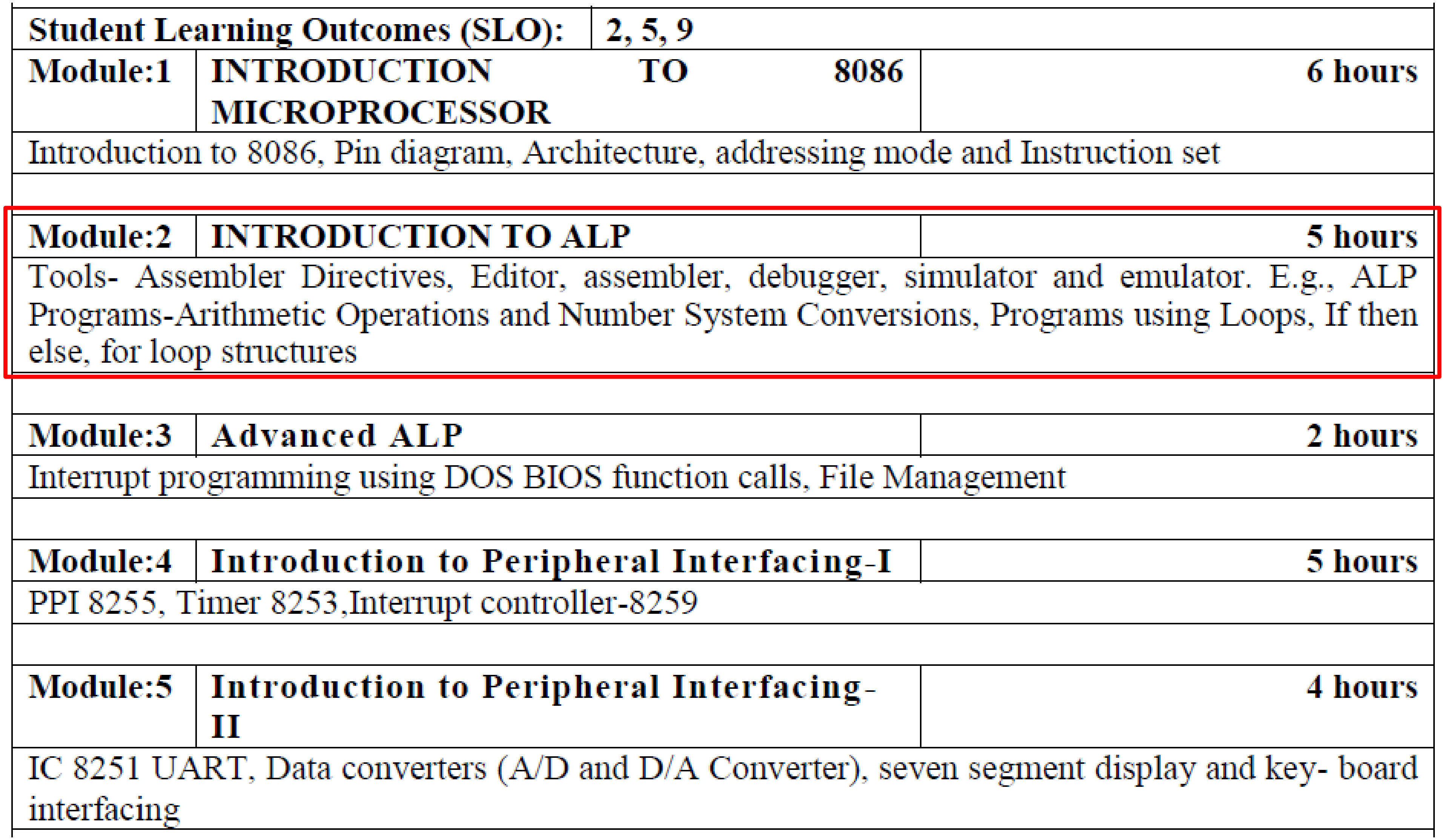
**Course Objectives:**

1. Students will gain knowledge on architecture, accessing data and instruction from memory for processing.
2. Ability to do programs with instruction set and control the external devices through I/O interface
3. Generate a system model for real world problems with data acquisition, processing and decision making with aid of micro controllers and advanced processors.

**Expected Course Outcome:**

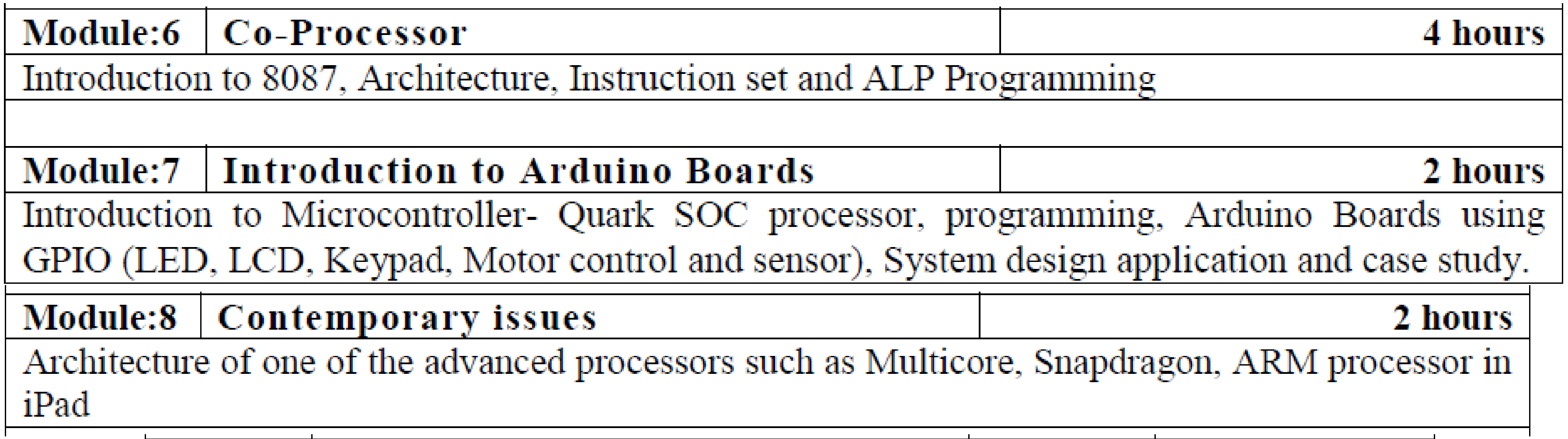
1. Recall the basics of processor, its ways of addressing data for operation by instruction set.
2. Execute basic and advanced assembly language programs.
3. Learn the ways to interface I/O devices with processor for task sharing.
4. Recall the basics of co-processor and its ways to handle float values by its instruction set.
5. Recognize the functionality of micro controller, latest version processors and its applications.
6. Acquire design thinking capability, ability to design a component with realistic constraints, to solve real world engineering problems and analyze the results.

# Syllabus

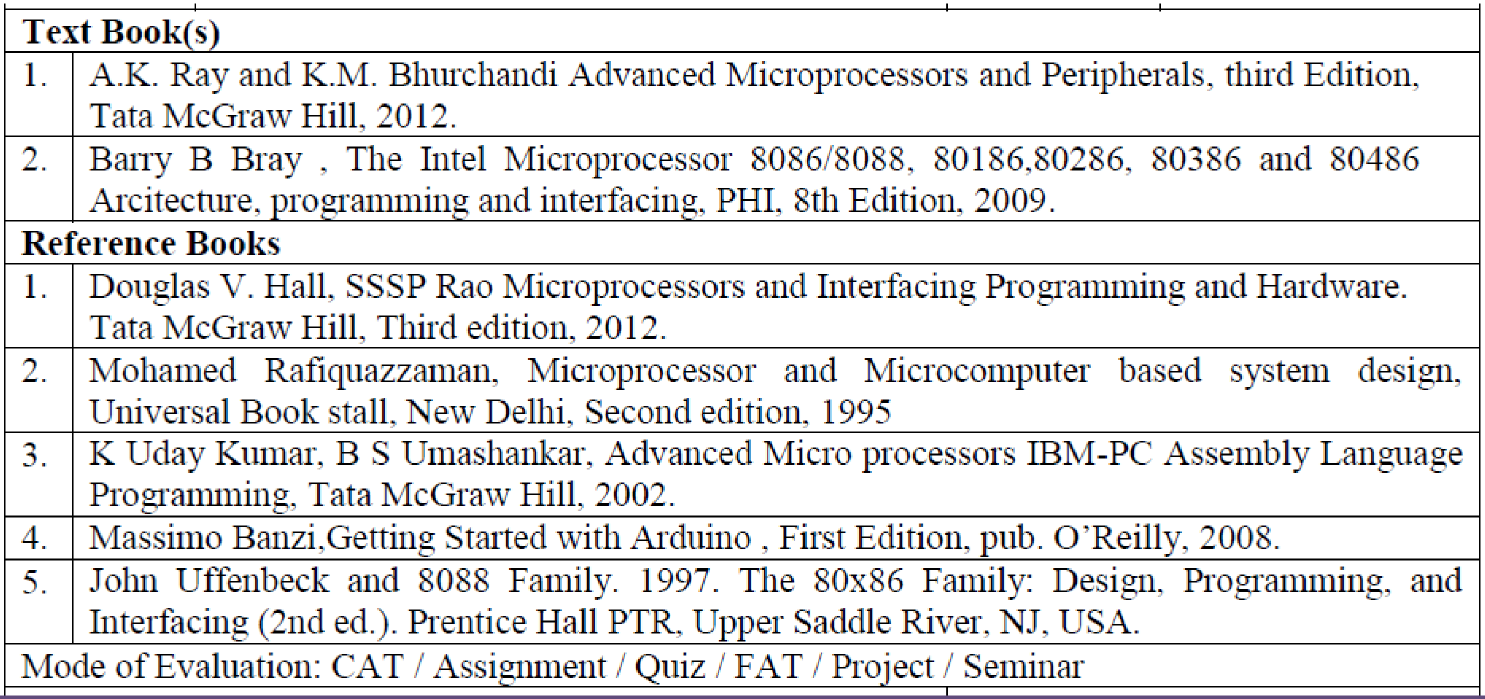


| | | |
|---|---|---|
| **Student Learning Outcomes (SLO):** | **2, 5, 9** | |
| **Module:1** | **INTRODUCTION TO 8086 MICROPROCESSOR** | **6 hours** |
| Introduction to 8086, Pin diagram, Architecture, addressing mode and Instruction set | | |
| | | |
| **Module:2** | **INTRODUCTION TO ALP** | **5 hours** |
| Tools- Assembler Directives, Editor, assembler, debugger, simulator and emulator. E.g., ALP Programs-Arithmetic Operations and Number System Conversions, Programs using Loops, If then else, for loop structures | | |
| | | |
| **Module:3** | **Advanced ALP** | **2 hours** |
| Interrupt programming using DOS BIOS function calls, File Management | | |
| | | |
| **Module:4** | **Introduction to Peripheral Interfacing-I** | **5 hours** |
| PPI 8255, Timer 8253,Interrupt controller-8259 | | |
| | | |
| **Module:5** | **Introduction to Peripheral Interfacing-II** | **4 hours** |
| IC 8251 UART, Data converters (A/D and D/A Converter), seven segment display and key- board interfacing | | |

# Syllabus

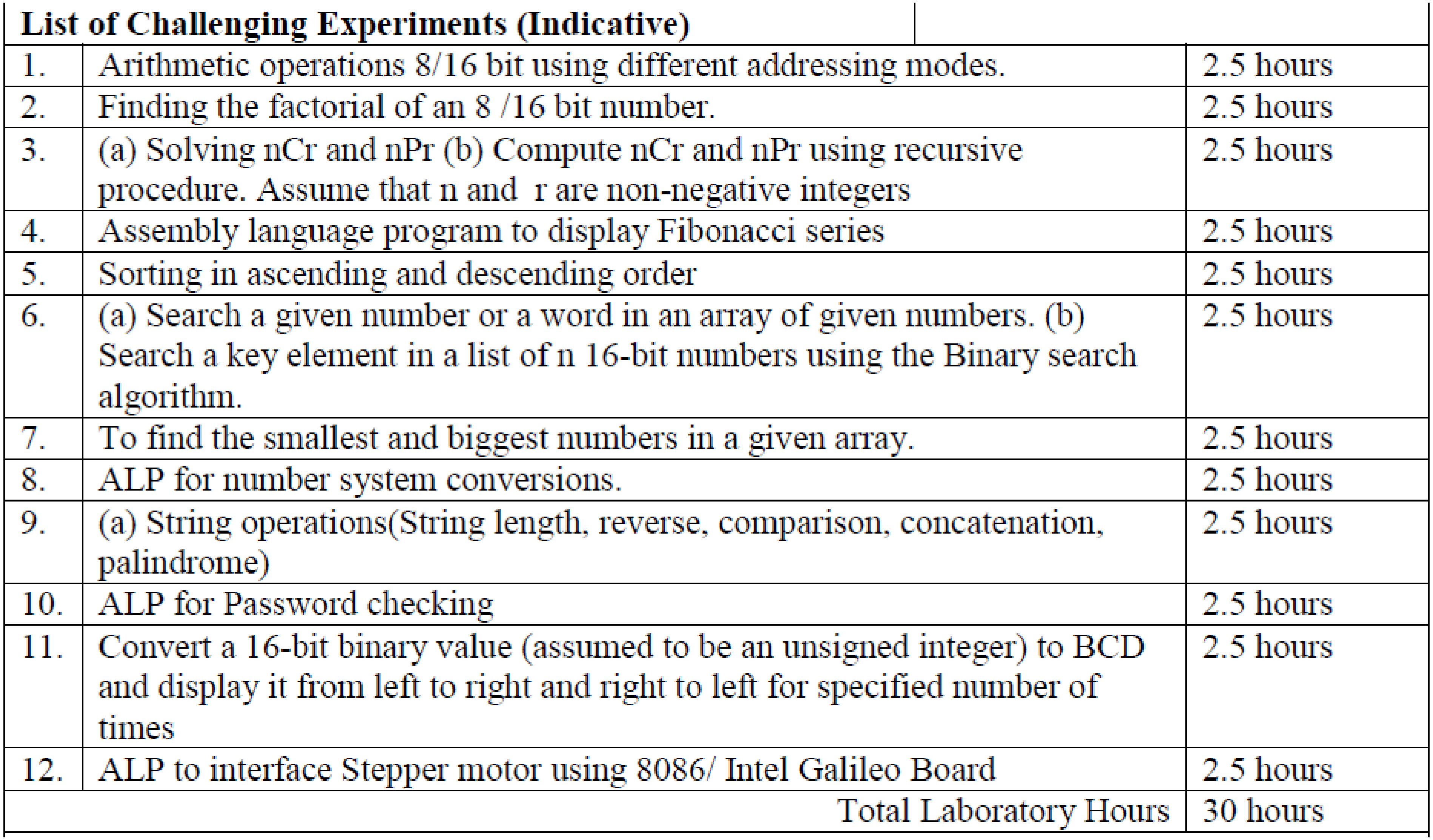


| Module:6 | Co-Processor | 4 hours |
|---|---|---|
| Introduction to 8087, Architecture, Instruction set and ALP Programming | | |
| | | |
| **Module:7** | **Introduction to Arduino Boards** | **2 hours** |
| Introduction to Microcontroller- Quark SOC processor, programming, Arduino Boards using GPIO (LED, LCD, Keypad, Motor control and sensor), System design application and case study. | | |
| **Module:8** | **Contemporary issues** | **2 hours** |
| Architecture of one of the advanced processors such as Multicore, Snapdragon, ARM processor in iPad | | |



| **Text Book(s)** | |
|---|---|
| 1. | A.K. Ray and K.M. Bhurchandi Advanced Microprocessors and Peripherals, third Edition, Tata McGraw Hill, 2012. |
| 2. | Barry B Bray , The Intel Microprocessor 8086/8088, 80186,80286, 80386 and 80486 Arcitecture, programming and interfacing, PHI, 8th Edition, 2009. |
| **Reference Books** | |
| 1. | Douglas V. Hall, SSSP Rao Microprocessors and Interfacing Programming and Hardware. Tata McGraw Hill, Third edition, 2012. |
| 2. | Mohamed Rafiquazzaman, Microprocessor and Microcomputer based system design, Universal Book stall, New Delhi, Second edition, 1995 |
| 3. | K Uday Kumar, B S Umashankar, Advanced Micro processors IBM-PC Assembly Language Programming, Tata McGraw Hill, 2002. |
| 4. | Massimo Banzi,Getting Started with Arduino , First Edition, pub. O'Reilly, 2008. |
| 5. | John Uffenbeck and 8088 Family. 1997. The 80x86 Family: Design, Programming, and Interfacing (2nd ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA. |
| Mode of Evaluation: CAT / Assignment / Quiz / FAT / Project / Seminar | |

# Syllabus



| List of Challenging Experiments (Indicative) | | |
|---|---|---|
| 1. | Arithmetic operations 8/16 bit using different addressing modes. | 2.5 hours |
| 2. | Finding the factorial of an 8 /16 bit number. | 2.5 hours |
| 3. | (a) Solving nCr and nPr (b) Compute nCr and nPr using recursive procedure. Assume that n and r are non-negative integers | 2.5 hours |
| 4. | Assembly language program to display Fibonacci series | 2.5 hours |
| 5. | Sorting in ascending and descending order | 2.5 hours |
| 6. | (a) Search a given number or a word in an array of given numbers. (b) Search a key element in a list of n 16-bit numbers using the Binary search algorithm. | 2.5 hours |
| 7. | To find the smallest and biggest numbers in a given array. | 2.5 hours |
| 8. | ALP for number system conversions. | 2.5 hours |
| 9. | (a) String operations(String length, reverse, comparison, concatenation, palindrome) | 2.5 hours |
| 10. | ALP for Password checking | 2.5 hours |
| 11. | Convert a 16-bit binary value (assumed to be an unsigned integer) to BCD and display it from left to right and right to left for specified number of times | 2.5 hours |
| 12. | ALP to interface Stepper motor using 8086/ Intel Galileo Board | 2.5 hours |
| | Total Laboratory Hours | 30 hours |

# Module 2: Introduction to ALP

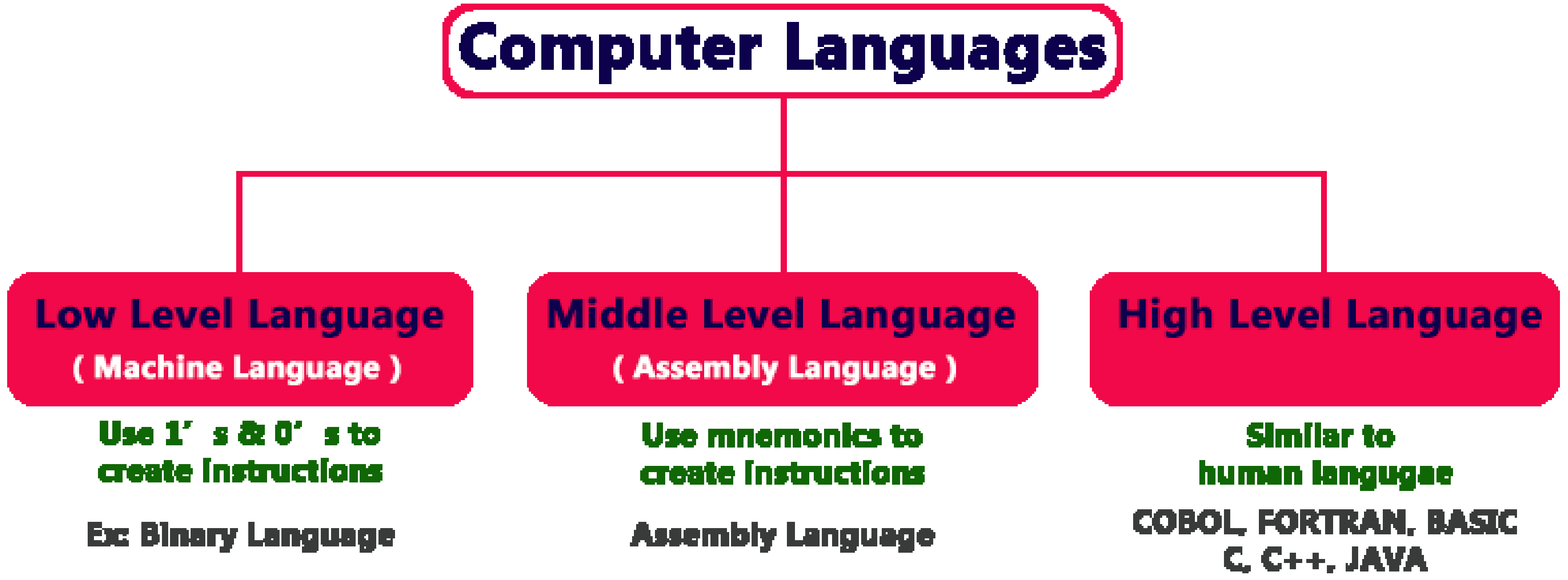
- **Introduction**
- ALP Development Tools
  - Editor
  - Assembler
  - Library Builder
  - Linker
  - Debugger
  - Simulator
  - Emulator
- Assembler Directives
- ALP Programs
  - Arithmetic Operations
  - Number System Conversions
  - Programs using Loops
  - If then else
  - For loop structures

## Introduction: Levels of Programming

**Definition:**

- A program is a *set of instructions or commands* needed for performing a specific task by a programmable device such as a microprocessor.

**Levels:**



## Introduction: Levels of Programming

**Machine Level:**

- Instructions - ***Binary codes***: '0' and '1'.
- Binary codes represent each operation to be performed.
- If program is developed using binary codes, it is called machine level programming.
- Microprocessor can understand and execute the machine language programs directly.
- Machine Dependent: Binary instructions of each microprocessors will be different.
- Highly tedious for a programmer to write machine language programs.

## Introduction: Levels of Programming

**Assembly Level:**

- Instructions - ***mnemonics*** - Few letters of English language which represent the operation performed by the instruction.

  Example: Addition operation: ADD.

- Each mnemonic has its own binary code

- If the program is developed using mnemonics, it is called assembly level programming.

- Microprocessors cannot execute the assembly language programs directly. (Machine dependent / upward compatibility)

- The assembly language programs have to be converted to machine language for execution, performed using a software tool, called ***assembler***.

## Introduction: Levels of Programming

**High Level:**

- Instructions - Form of statements written using symbols, English words and phrases.

  Examples: class triangle { … }

- Each high level language will have its own syntax, vocabulary of words, symbols, phrases and sentences.

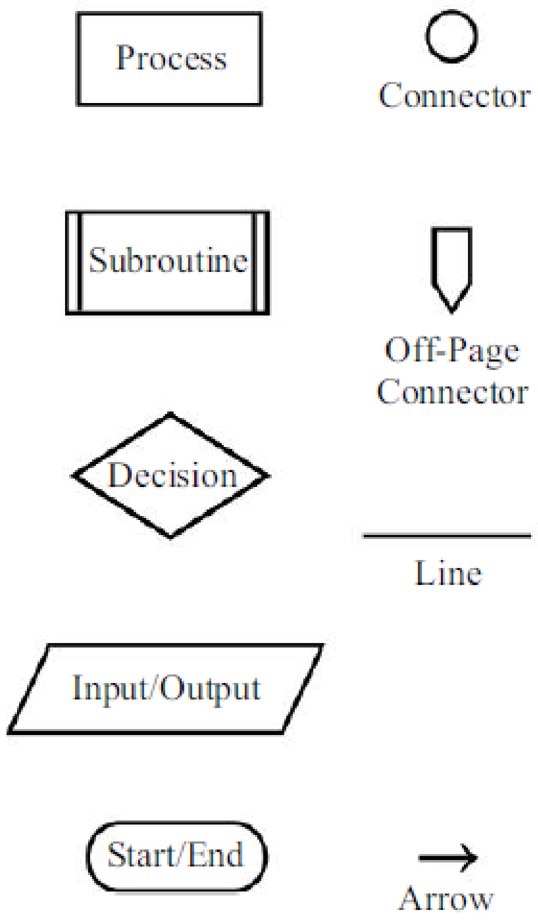
- High level languages are easy to understand, machine independent and portable.

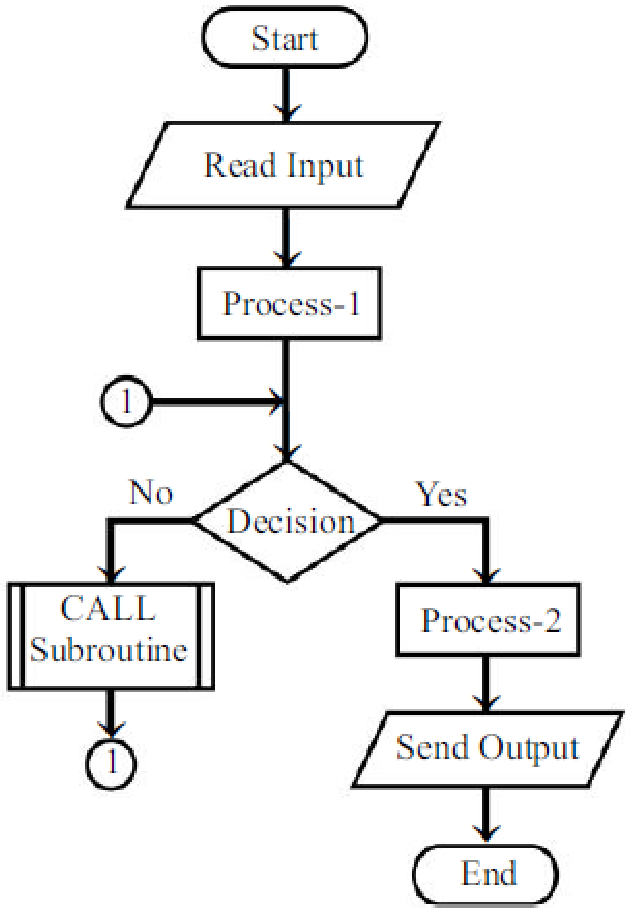
- High level language programs have to be converted into machine language for execution, performed using a software tool called ***compiler***.

- Two Types: Compiled Language & Interpreted Language

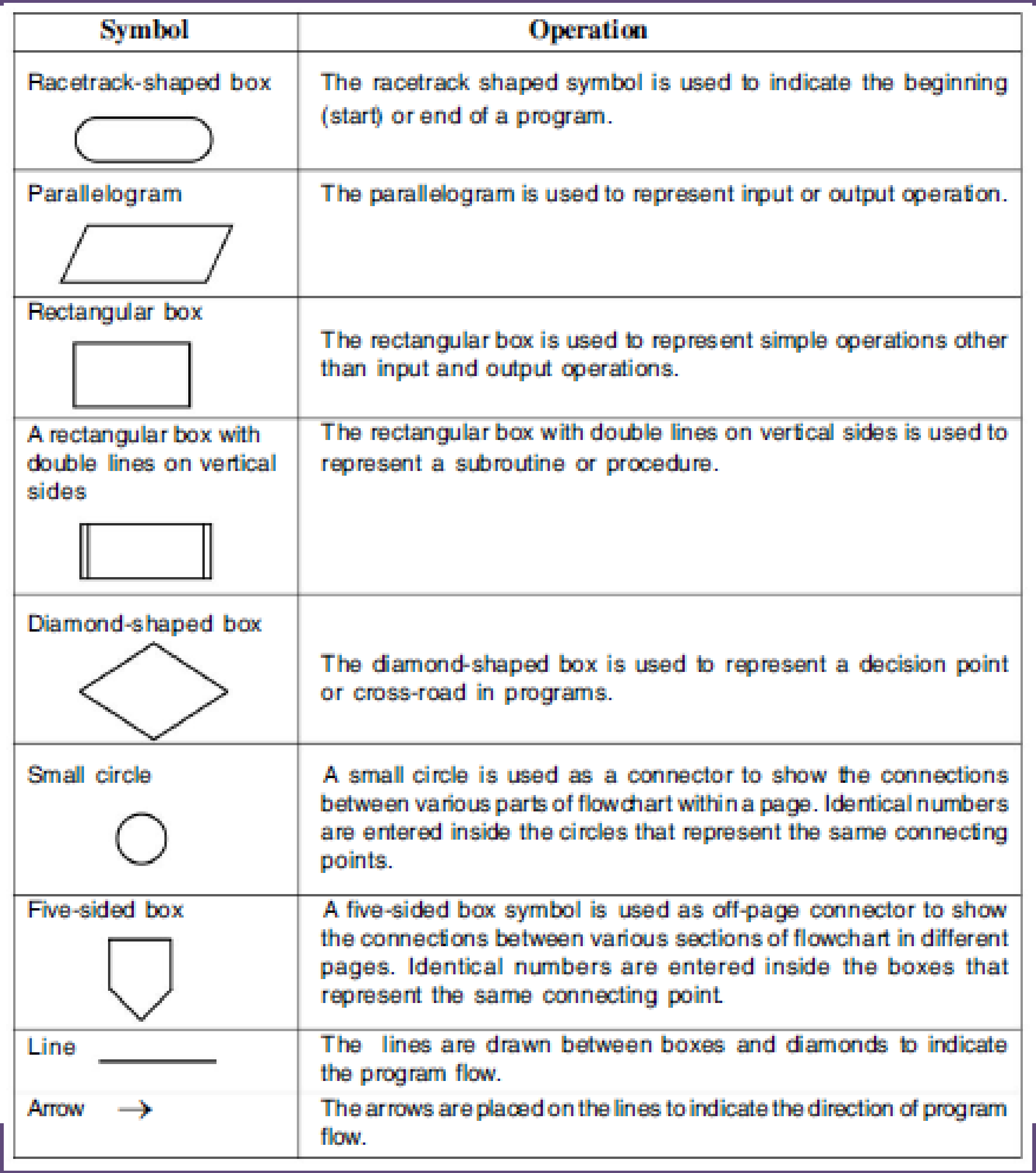
# Introduction: Flowchart

- Graphical representation of the operation flow of the program or algorithm is flowchart.
- Flowcharts are valuable aid in visualizing programs.





# Introduction: Flowchart



| Symbol | Operation |
|---|---|
| Racetrack-shaped box | The racetrack shaped symbol is used to indicate the beginning (start) or end of a program. |
| Parallelogram | The parallelogram is used to represent input or output operation. |
| Rectangular box | The rectangular box is used to represent simple operations other than input and output operations. |
| A rectangular box with double lines on vertical sides | The rectangular box with double lines on vertical sides is used to represent a subroutine or procedure. |
| Diamond-shaped box | The diamond-shaped box is used to represent a decision point or cross-road in programs. |
| Small circle | A small circle is used as a connector to show the connections between various parts of flowchart within a page. Identical numbers are entered inside the circles that represent the same connecting points. |
| Five-sided box | A five-sided box symbol is used as off-page connector to show the connections between various sections of flowchart in different pages. Identical numbers are entered inside the boxes that represent the same connecting point. |
| Line | The lines are drawn between boxes and diamonds to indicate the program flow. |
| Arrow → | The arrows are placed on the lines to indicate the direction of program flow. |

## Module 2: Introduction to ALP

- Introduction
- **ALP Development Tools**
  - **Editor**
  - **Assembler**
  - **Library Builder**
  - **Linker**
  - **Debugger**
  - **Simulator**
  - **Emulator**
- Assembler Directives
- ALP Programs
  - Arithmetic Operations
  - Number System Conversions
  - Programs using Loops
  - If then else
  - For loop structures

## ALP Development Tools

- To design and test the software and hardware of a microprocessor system before going for practical implementation (or fabrication).
- Contains a set of ***hardware*** and ***software tools***.
- The hardware tools includes a standard PC (Personal Computer), printer and an emulator.
- The software tools includes editor, assembler, library builder, linker, debugger and simulator.
- Software tools run on hardware tools to write or edit, assemble, debug, modify and test the assembly language programs.

# ALP Development Tools

**Editor (Text Editor)**

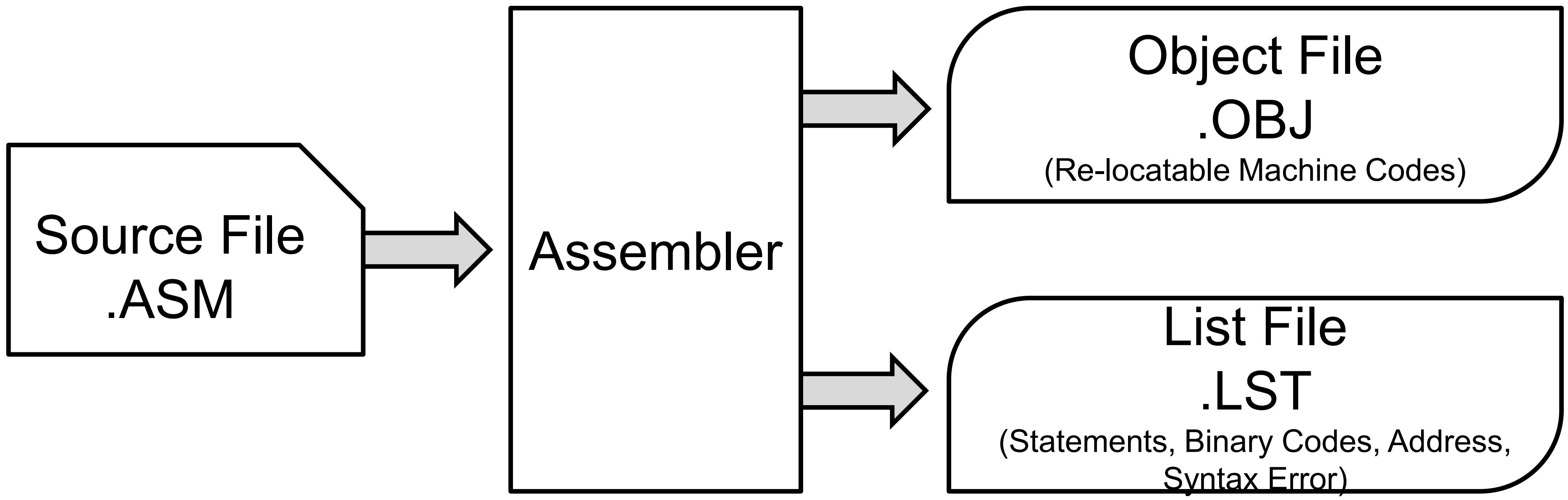
- A software tool run on a PC, allows the user to type/enter and modify the ALP.
- The editor provides a set of commands for insertion, deletion, modification of letters, characters, statements, etc.
- To construct the ALP in the right format.
- The program created using an editor is known as source program and it is saved with file extension ".ASM".

  Example: "ADDITION.ASM"
- Some examples of editors are NE (NASM - Norton Editor), EDIT (DOS Editor).

## ALP Development Tools

**Assembler**

- A software tool run on a PC, converts the assembly language program to a machine language program.
- Types: One-pass, Two-pass, Macro, Cross, Resident and Meta assembler.


Source File
.ASM
Assembler
Object File
.OBJ
(Re-locatable Machine Codes)
List File
.LST
(Statements, Binary Codes, Address, Syntax Error)

## ALP Development Tools

**Assembler**

- Examples: TASM (Borland's Turbo Assembler), MASM (Microsoft's Macro Assembler), ASM85 (INTEL'S 8085 Assembler), ASM86 (INTEL'S 8086 Assembler).
- Translates mnemonics into binary code with speed and accuracy.
- Assigns appropriate values to the variables used in a program.
- Easy to insert or delete instructions in a program and reassemble the entire program quickly with new memory locations and modified addresses for jump locations.
- Reserve memory locations for data or results.
- List file can be used for documentation

# ALP Development Tools

**Library Builder**

- To create library files which are collections of procedures of frequently used functions (object files)

- When the library file is linked with a program, only the procedure required by the program are copied from the library file and added to the program.

- The library builder combines the program modules/procedures into a single file known as library file and it is saved with file extension ".LIB".

- Examples: Microsoft's LIB, Borlands TLIB.

## ALP Development Tools

**Linker**

- A software tool to combine re-locatable object files of program modules and library functions to a single executable file (.EXE).
- Modules: Entire task of the program can be divided into smaller tasks and procedures that can be developed individually (using libraries)
- Object files and Library files can be linked to get an Executable file.
- Generates a link map file which contains the address information about the linked files.
- Examples: Microsoft's linker LINK, Borland's Turbo linker TLINK.

## ALP Development Tools

**Debugger**

- A software tool that allows the execution of a program in single-step or breakpoint mode under the control of the user.
- Process of locating and correcting the errors using a debugger is debugging.
- Allows the designer to examine the contents of registers and memory locations after running the program.
- Helps the user to isolate & rectify a problem in the program,
- User can use the editor to correct the source program, reassemble the corrected source program, relink and run the program again.

## ALP Development Tools

**Simulator**

- A program run on PC to simulate the operations of the newly designed system.

- Operations of Simulator:
    1. Execute a program and display result
    2. Single-step execution of a program
    3. Breakpoint execution of a program
    4. Display the contents of register/memory

## ALP Development Tools

**Simulator**

- Not only shows the contents of registers and memory locations, but allows to watch data change as the program operates.
- Saves considerable time in not using separate commands to view the contents.
- Gives a better feel for what is taking place in execution.
- Simulators do not have the ability to perform actual IO or internal hardware operations such as timing or data transmission and reception.

## ALP Development Tools

**Emulator**

- A mixture of hardware and software.
- To test and debug the hardware and software of a newly designed microprocessor-based system.
- Has a multicore cable which connects the PC of a development system with the newly designed hardware of the microprocessor system.
- Allows the designer to download the object code program into RAM, to test and run.
- Like a debugger, Allows user to load and run programs, examine and change the contents of registers and memory locations and insert breakpoints in the program.

# ALP Development Tools

**Emulator**

- Stores the trace data of the contents of registers, activity on the address and data bus and the state of the flags as each instruction executes.
- User can have a printout of the trace data to see the results that the program produced on a step-by-step basis.
- Powerful feature: Ability to use either development system memory or the memory on the hardware under test for the program that is being debugged.

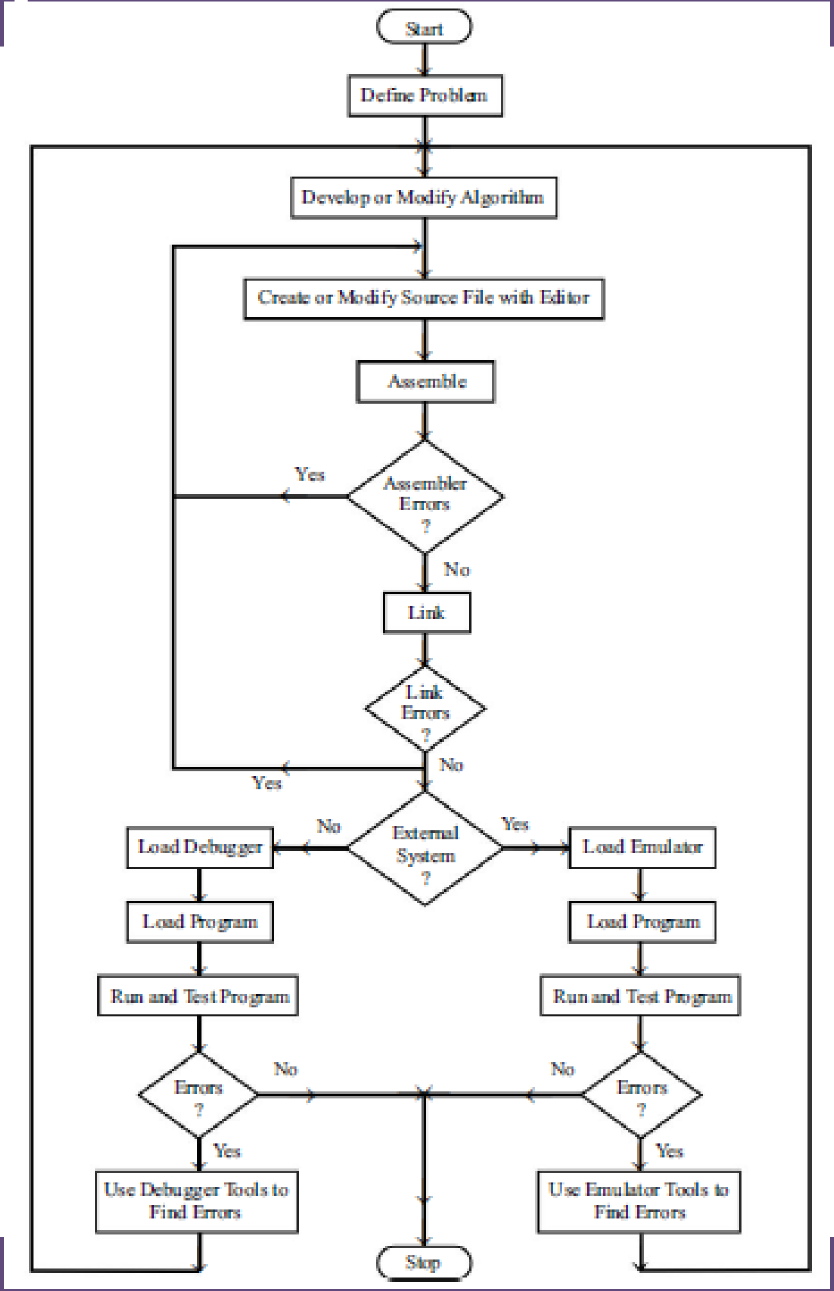
# ALP Development Tools

**Summary of ALP Development Tools**

1. Define the problem carefully.
2. Create the ALP source file using editor.
3. Assemble the source file using assembler.
4. If assembler indicates errors, use editor and correct the errors.
5. Cycle through the edit-assemble loop until all errors are cleared.
6. Link all object files of the program modules and library files into a single executable file using linker.
7. If linker indicates any error then modify the source program, reassemble and relink it to correct the errors.
8. If developed program does not interact with any external hardware as intended, use debugger to run and debug.
9. If designed program is intended to work with external hardware, use an emulator to run and debug.

# ALP Development Tools

## Summary



Start
Define Problem
Develop or Modify Algorithm
Create or Modify Source File with Editor
Assemble
Assembler Errors ?
Yes
No
Link
Link Errors ?
Yes
No
External System ?
No
Yes
Load Debugger
Load Emulator
Load Program
Load Program
Run and Test Program
Run and Test Program
Errors ?
No
Yes
Errors ?
No
Yes
Use Debugger Tools to Find Errors
Use Emulator Tools to Find Errors
Stop