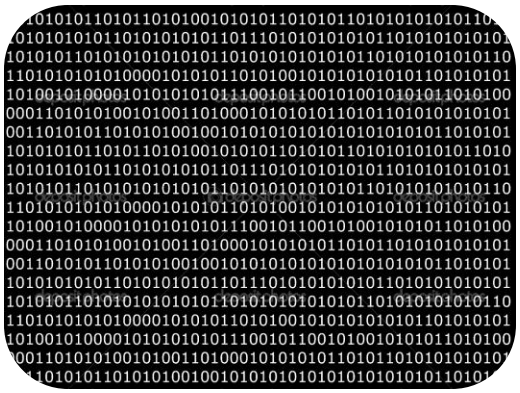
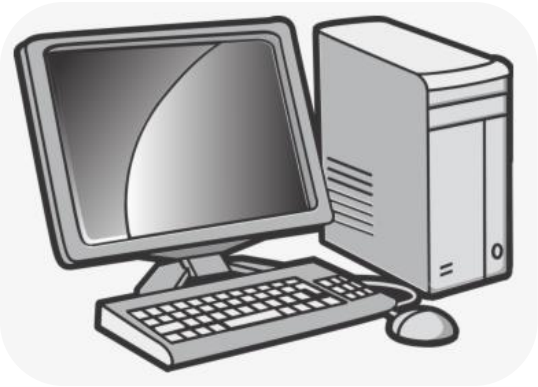
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

Unit-3

Assembly Language Programming





Subject:- COA Code:-3140707	Prepared by: Asst. Prof. S Y JOSHI (CSE Department,ACET)	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Topics to be covered

- Introduction,
- Machine Language,
- Assembly Language Programming
- Arithmetic and logic operations,
- looping constructs,
- Subroutines,
- I-O Programming

Introduction

- A total computer system includes both hardware and software. Hardware consists of the physical components and all associated equipment. Software refers to the programs that are written for the computer. It is possible to be familiar with various aspects of computer software without being concerned with details of how the computer hardware operates.
- Writing a program for a computer consists of specifying, directly or indirectly, a sequence of machine instructions. Machine instructions inside the computer form a binary pattern which is difficult to remember. There is a need for translating user-oriented symbolic programs into binary programs recognized by the hardware.

Machine Language

- A program is list of instructions or statements for directing the computer to perform a required data processing task.
- Programs written for a computer may be in one of the following categories:

1. **Binary code**. This is a sequence of instructions and operands in binary that list the exact representation of instructions as they appear in computer memory.
2. **Octal or hexadecimal code**. This is an equivalent translation of the binary code to octal or hexadecimal representation.
3. **Symbolic code**. The user employs symbols (letters, numerals, or special characters) for the operation part, the address part, and other parts of the instruction code. Each symbolic instruction can be translated into one binary coded instruction. This translation is done by a special program called an assembler. Because an assembler translates the symbols, this type of symbolic program is referred to as an assembly language program.

Machine Language

.4. **High-level programming languages** .

These are special languages developed to reflect the procedures used in the solution of a problem rather than be concerned with the computer hardware behavior. An example of a high-level programming language is Fortran. It employs problem oriented symbols and formats. The program is written in a sequence of statements in a form that people prefer to think in when solving a problem. However, each statement must be translated into a sequence of binary instructions before the program can be executed in a computer. The program that translates a high-level language program to binary is called a compiler.

.

Assembly Language Programming

A programming language is defined by a set of rules.

- The basic unit of an assembly language program is a line of code.
- set of rules specify the symbols that can be used and how they may be combined to form a line of code.
- Each line of an assembly language program is arranged in three columns called fields.

The fields specify the following information.

1. The label field may be empty or it may specify a symbolic address.
2. The instruction field specifies a machine instruction or a pseudo instruction.
3. The comment field may be empty or it may include a comment.

- A symbolic address consists of one, two, or three, but not more than three alphanumeric characters.

Assembly Language Programming

- The instruction field in an assembly language program may specify one of the following items:
- 1. A memory-reference instruction (MRI)
- 2. A register-reference or input-output instruction (non-MRI)
- 3. A pseudo instruction with or without an operand
- A memory-reference instruction occupies two or three symbols separated by spaces. The first must be a three-letter symbol defining an MRI operation. The second is a symbolic address. The third symbol, which may or may not be present, is the letter
- non-MRI is defined as an instruction that does not have an address part
- A pseudoinstruction is not a machine instruction but rather an instruction to the assembler giving information about some phase of the translation

Assembly Language Programming

MRI/NON-MRI instructions

Symbol	Hexadecimal code	Description
AND	0 or 8	AND M to AC
ADD	1 or 9	Add M to AC, carry to E
LDA	2 or A	Load AC from M
STA	3 or B	Store AC in M
BUN	4 or C	Branch unconditionally to m
BSA	5 or D	Save return address in m and branch to $m + 1$
ISZ	6 or E	Increment M and skip if zero
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right E and AC
CIL	7040	Circulate left E and AC
INC	7020	Increment AC,
SPA	7010	Skip if AC is positive
SNA	7008	Skip if AC is negative
SZA	7004	Skip if AC is zero
SZE	7002	Skip if E is zero
HLT	7001	Halt computer
INP	F800	Input information and clear flag
OUT	F400	Output information and clear flag
SKI	F200	Skip if input flag is on
SKO	F100	Skip if output flag is on
ION	F080	Turn interrupt on
IOF	F040	Turn interrupt off

Assembly Language Programming

Pseudo instructions

Symbol	Information for the Assembler
ORG N	Hexadecimal number N is the memory location for the instruction or operand listed in the following line
END	Denotes the end of symbolic program
DEC N	Signed decimal number N to be converted to binary
HEX N	Hexadecimal number N to be converted to binary

An assembler is a program that accepts a symbolic language program and produces its binary machine language equivalent.

Arithmetic and logic operations

- Some computers have machine instructions to add, subtract, multiply, and divide. Others, such as the basic computer, have only one arithmetic instruction, such as ADD.
- Operations not included in the set of machine instructions must be implemented by a program

looping constructs

- A program loop is a sequence of instructions that are executed many times, each time with a different set of data.
- System program that translates a program written in a high-level programming language such as the above to a machine language program is compiler called a compiler .

Exercise-1

Perform Multiplication Program using repeated addition loop in

Exercise-2

Perform double precision addition using ADD and loop

Subroutines,

- Frequently, the same piece of code must be written over again in many different parts of a program.
- Instead of repeating the code every time, needed, there is an obvious advantage if the common instructions are written only once.
- A set of common instructions that can be used in a program many times is called a subroutine.
- Each time that a subroutine is used in the main part of the program, a branch is executed to the beginning of the subroutine. After the subroutine has been executed, a branch is made back to the main program.
- A subroutine consists of a self-contained sequence of instructions that carries out a given task. A branch can be made to the subroutine from any part of the main program

Subroutines,

- the link between the main program and a subroutine is the BSA instruction (branch and save return address).

- **Exercise**
- Write a Subroutine to Move a Block of Data

I-O Programming

- Users of the computer write programs with symbols that are defined by the programming language employed. The symbols are strings of characters and each character is assigned an 8-bit code so that it can be stored in computer memory.
- A binary-coded character enters the computer when an INP (input) instruction is executed.
- A binary-coded character is transferred to the output device when an OUT (output) instruction is executed.
- The output device detects the binary code and types the corresponding character.

References

- **Images , descriptive Tables , from Computer System Architecture, Morris Mano, 3rd edition Prentice Hall**

- **Note: These pdf/ppt notes are for purpose of teaching aids to classroom/online sessions study, and in no case imply for GTU syllabus or GTU exam. For GTU syllabus or exam related preparation, one may, however will need to attend college/online lectures and refer books given by GTU in their official syllabus.**