EEE.4.2 Computer Systems Architecture

COURSE CONTENT

- Background – History
- Introduction to the Architecture of Computer Systems
- Arithmetic Issues (Number systems, numerical representations, etc.)
- Systems Organization – Computer Architecture
- Memory organization and operation - Addressing
- Central Processing Unit Architectures
- Basic functions of the Central Processing Unit
- Techniques to increase the efficiency of the Central Processing Unit
- Programming in machine language
- Interrupt System
- Data I/O Techniques
- Communications systems
- Parallel architectures
- Distributed system architectures
- Evolutionary trends (ANN, Fuzzy, Dataflow, Quant)

Hands on Laboratory Exercises

The laboratory training incorporates 13 laboratory exercises focused on the main subjects of theoretical teaching. The exercises will cover the following fields:

- Familiarization with programming environments
- 8086 Assembly Instruction Set – Arithmetic and Logical Operations
- Program outline in symbolic language – Directives
- Introduction to Software Interrupts
- Printing of constant and parametric messages
- Printing of Register contents
- Loops implementation using comparison – branch commands
- Transfer and processing of data blocks
- Data entry
- Video Functions – Use of graphics
- Subroutines – Macros – Stack

LEARNING OUTCOMES

The aim of this course is to understand the internal architecture of modern computer systems and their communication with external I/O devices. The main architectures of all the basic subsystems of modern computer systems and their operation are presented. At the same time, the programming of the systems in machine language is presented and the concepts of command coding are analyzed, as well as the basic elements of compilation in machine language.

Upon successful completion of the course the student will be able to:

- Recognize and understand basic computer system architectures.
- Understand the operation of basic subsystems (Central Processing Unit, Memory, etc.)
- Program processors in assembly language.
- Simulate the operation of a computer system using reverse engineering for debugging.