# 🧠 8086 Microprocessor System Project (Part I-B)

## 📘 Project Summary

This project presents the complete design, integration, and initialization of a microcomputer system based on the Intel **8086 microprocessor running at 10 MHz**, with support for all major peripherals and coprocessing capabilities. The system includes:

* Math Coprocessor **8087**
* **1MB total memory** (RAM and ROM)
* **Diskette controller 8272** (added in Part I-B)
* **Parallel**, **Serial**, **USB** (DMA-based) communication
* **ADC/DAC** for analog/digital conversion
* **16-digit seven-segment display**
* **64-key matrix keyboard**
* **Printer interface**
* **8237 DMA Controller**
* **8259A Interrupt Controller**

The full system includes memory decoding logic, I/O mapping, assembly-level routines, pseudocode for all initialization steps, and team member task distribution.

## 🧩 Design Objectives

* 🧠 Construct a working microcomputer based on **8086** + **8087**
* 🗺️ Design and implement a **1MB memory map** with DRAM, ROM, I/O
* 🔌 Integrate peripherals via **memory-mapped I/O**
* 🧮 Use **8237 DMA** for high-speed USB/disk transfers
* ⚡ Use **8259A PIC** for interrupt-driven I/O handling
* 💾 Add **8272 diskette controller** (Part I-B requirement)
* 🔧 Provide full initialization and test routines in **pseudocode + assembly**

## 🧱 System Architecture

* Full block diagram of CPU, memory, coprocessor, and peripheral buses
* Unified memory and I/O space using **address decoding logic** with **74LS138**
* All data/control signals routed and labeled

📄 Refer to: CPU\_Memory.md, CPU\_Memory.asm, CPU\_Memory\_pseudocode.md

## 💾 Memory System Design

* 1MB memory space divided as:
  + **768KB RAM** using 4164 DRAM chips (3 banks)
  + **128KB ROM** for BIOS and routines
  + **128KB I/O-mapped area** for peripherals
* I/O Address Summary: | Address Range | Assigned To | |——————|——————————-| | 0xC0000 | Interrupt controller (PIC) | | 0xC0100 | DMA | | 0xC0200 | Serial port (8251) | | 0xC0300 | Parallel port (8255) | | 0xC0400 | Keyboard (8279) | | 0xC0500 | Display (8279) | | 0xC0600 | ADC/DAC | | 0xC0700 | USB Interface (DMA) | | 0xC0800 | Printer (8255) | | 0xC0900 | Diskette Controller (8272) |

📄 Refer to: CPU\_Memory.md, Storage\_DMA\_Integration.md

## 🛠️ Software & Initialization

* All system modules are initialized via assembly and/or pseudocode:
  + System startup and memory test
  + 8087 coprocessor detection and setup
  + DMA configuration and buffer setup
  + Peripheral setup (keyboard, display, printer, ADC/DAC, disk)
  + USB + Floppy transfer routines using **DMA Channels 1–3**

📄 Refer to: - CPU\_Memory.asm, CPU\_Memory\_pseudocode.md - IO\_Peripherals.md - Data\_Conversion\_Interrupts.md - Storage\_DMA\_Integration.md —

## 📚 Team Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Member | Area | Key Components | Deliverables | Files Submitted |
| **Jared** | CPU & Memory Architecture | • 8086 CPU• 8087 Coprocessor• 1MB RAM/ROM• Address decoding | • Block diagrams• Memory map• Initialization code• Assembly routines | **CPU\_Memory.md**CPU\_Memory.asmCPU\_Memory\_pseudocode.mdCPU\_Memory.txt |
| **Jesmarie** | User I/O Interface | • 16-digit 7-segment display• 64-key matrix keyboard• Printer | • Display driver• Keyboard scanner• Printer interface• Assembly examples | **IO\_Peripherals.md** |
| **Valeria** | Communications & Interrupts | • RS-232 serial port• Parallel port• USB+DMA• 8259A interrupt controller | • Communication drivers• DMA controller• Interrupt handlers• USB routines | **Data\_Conversion\_Interrupts.md** |
| **Giovanny** | Data Conversion & Storage | • ADC (Analog-to-Digital)• DAC (Digital-to-Analog)• 8272 Floppy controller | • ADC/DAC drivers• Disk controller• Conversion routines• Storage examples | **Storage\_DMA\_Integration.md** |

All files are modular and follow shared address conventions.

## ✅ Final Integration Test

* System performs correct memory initialization with RAM test and ROM checksum
* All I/O devices respond correctly to their mapped addresses
* DMA transfers are verified between floppy <-> memory <-> USB
* Interrupts are handled using vector table 0x0000–0x03FF and managed by 8259A

## 📎 Getting Started

To replicate or run the project: 1. Load CPU\_Memory.asm and simulate on 8086-compatible assembler 2. Refer to each .md file for subsystem details 3. Use pseudocode as reference for embedded system implementation 4. Test each module independently before full integration

## 📚 References (with PDF Filenames)

* [1] **8086 Memory and I/O Interfacing**  
    • 1.8086 Memory and I\_O Interfacing.pdf  
    • 2.8086 Memory and I\_O Interfacing. Part II.pdf  
    • 3.8086 Memory and I\_O Interfacing.Part III and Case of Studies.pdf
* [2] **System Bus Structure**  
    • 4.CAP 8 - SYSTEM BUS STRUCTURE.pdf
* [3] **I/O Interfaces and Interrupts**  
    • 5.CAP 9 - I.O INTERFACES.pdf
* [4] **Semiconductor Memory**  
    • 6.CAP 10 Semiconductor Memory.pdf
* [5] **Multiprocessor Configurations & Datasheets**  
    • 7.8086 Multiprocessor Configurations-1.pdf  
    • Manufacturer datasheets: 8086, 8087, 8237, 8259A, 8255, 8251, 8279, 8272, ADC0808, DAC0800