Restaurant Management System

Group - 14 Presentation

- **Isha Jangir** (202211031)
- **Tejas Pakhale** (202211061)
- **Rahul Gupta** (202211069)
- **Rajat Kumar Thakur** (202211070)
- **Sanskar Koserwal** (202211077)

Mentor - Dr. Varun Kumar

Indian Institute of Information Technology Vadodara
International Campus Diu
Education Hub, Kevdi Diu(U.T) - 362520

IIITV-ICD

Outline

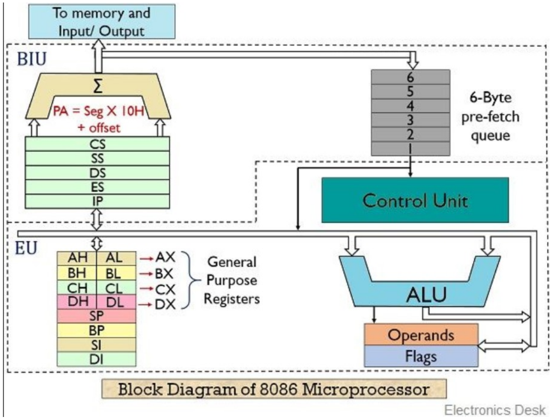
Introduction

- Our Restaurant Management System is a robust solution designed specifically for the food service industry. It optimizes restaurant operations, from inventory tracking to customer service, ensuring maximum effectiveness.
- Harnessing the power of assembly language x86, our system guarantees unparalleled performance, responsiveness, and stability. This advanced technology enhances efficiency and productivity, giving your restaurant a competitive edge in today’s market.

Benefits Of RMS using Assembly Language

- Performance Efficiency
- Resource Utilization
- Customization and Optimization
- Educational Value

8086 Architecture



Block Diagram of 8086 Microprocessor

Developing a RMS in Assembly Language

Modular Development:

Break down the RMS into manageable components for guest users and restaurant owners.

Implement functionalities separately, focusing on interactive menu browsing, seamless ordering, bill generation, inventory management, and order history.

Restaurant Owners' Implementation:

Implement inventory management for efficient stock updates.
Provide order history functionality for convenient bill retrieval.

Developing a RMS in Assembly Language

Functionality Implementation:

Develop interactive menu browsing for guests to visualize items dynamically.

Enable seamless ordering, allowing users to specify quantities and handle errors gracefully.

Implement bill generation for accurate and detailed receipts.

Error Handling:

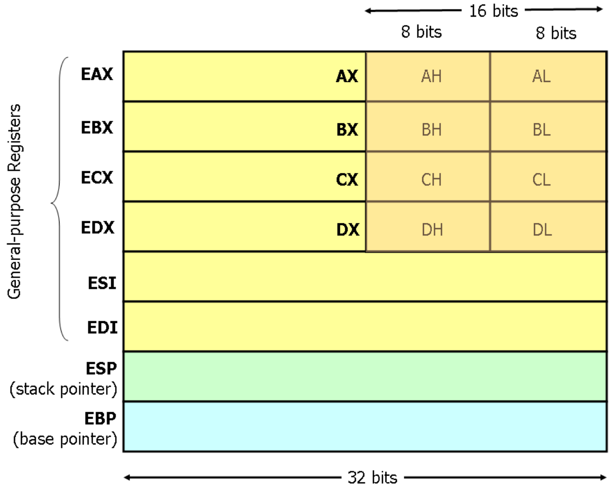
Implement robust error handling for edge cases like incorrect orders or out-of-stock items.

About X86 Assembly Language.

Technical Insights:

- Mnemonics and opcodes:
  - Mnemonics and opcodes in x86 assembly represent instructions through mnemonics translated to opcodes, like NOP (0x90) and HLT (0xF4). Some opcodes lack mnemonics, leading to varied processor interpretations, often used in coding contests for optimization or to showcase skill.

x86 Registers



About X86 Assembly Language.

- Instruction types:
    - Compact encoding: Efficiently encoded instructions.
    - General and implicit register usage: Registers can be freely used but are affected by certain instructions.
    - Conditional flags production: Integer ALU instructions generate conditional flags.
    - Various addressing modes: Immediate, offset, and scaled index addressing supported.
    - Special instructions: File read-modify-write, SIMD instructions for parallel operations, and stack instructions.

Why X86 Assembly Language?

- Direct Hardware Interaction
- Performance Optimization
- Compatibility with Legacy Systems
- Resource Efficiency
- Versatility and Customization

About MASM

The Microsoft Macro Assembler (MASM) provides several advantages over inline assembly. MASM contains a macro language that has features such as looping, arithmetic, and text string processing. MASM gives you greater control over the hardware. By using MASM, you also can reduce time and memory overhead in your code.

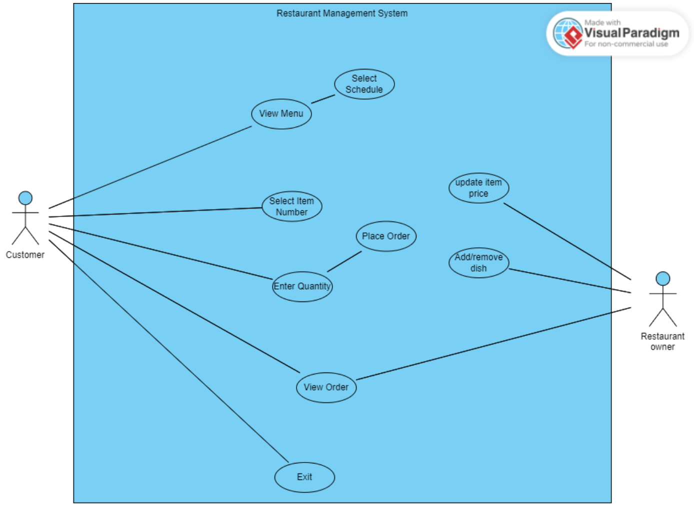
How we will implement RMS in MASM

- Data Structures:
  - Define data structures for menu items, inventory items, and orders.
- Menu and Inventory Management:
  - Implement functions to update the inventory when orders are placed.
- Order Placement:
  - Display the menu items and allow the user to select items and specify quantities.
  - Check the availability of items in the inventory before confirming the order.

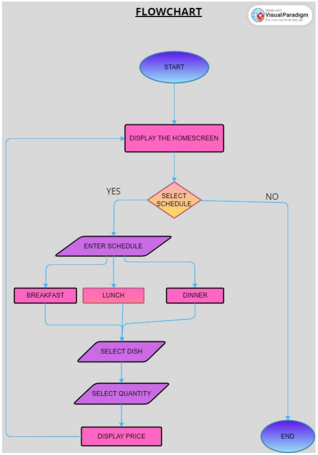
How we will implement RMS in MASM

- Bill Generation:
  - Calculate the total price of the order based on selected items and quantities.
- Order History:
  - Use file I/O or database operations to store order history data.
- Error Handling:
  - Implement robust error handling mechanisms to deal with edge cases such as incorrect orders, out-of-stock items, and invalid inputs.

Use Case Diagram



FlowChart for Customer



Conclusion

In conclusion, developing a Restaurant Management System (RMS) in x86 Assembly Language offers a tailored solution to address specific restaurant needs. Leveraging the low-level capabilities of x86 Assembly Language ensures unparalleled performance efficiency, precise hardware control, and seamless integration with restaurant hardware.

Thank You