Project

COAL

Submitted By:

Name: Syed Abdul Rehman Ibrar.

Reg. No: 24-CS-033

Section: C

Submitted to:

SIR Khalid

Department of Computer Science,

HITEC University, Taxila

1. Introduction

In remote areas, internet availability and computing resources are very limited. To address this issue, the government aims to deploy an edge-based solution using low-cost devices such as Raspberry Pi. Due to limited memory and processing power, low-level programming is preferred. This project implements a Remote Area Resource Management System using 8086 Assembly Language to efficiently manage population and resource consumption data.

2. Problem Statement

The objective of this project is to design an assembly language-based system that manages data related to population and essential resources (water, flour, and pulses) of a remote area. The system must support basic database operations while working within strict hardware constraints.

3. Objectives

- To implement a database system using 8086 Assembly Language
- To manage records with minimal memory usage
- To allow adding, updating, and deleting records
- To prevent duplicate entries
- To perform sorting based on resource consumption
- To calculate total consumption of resources

4. System Requirements

Hardware Requirements

- Raspberry Pi / x86 Emulator (EMU8086)
- Minimum RAM usage

Software Requirements

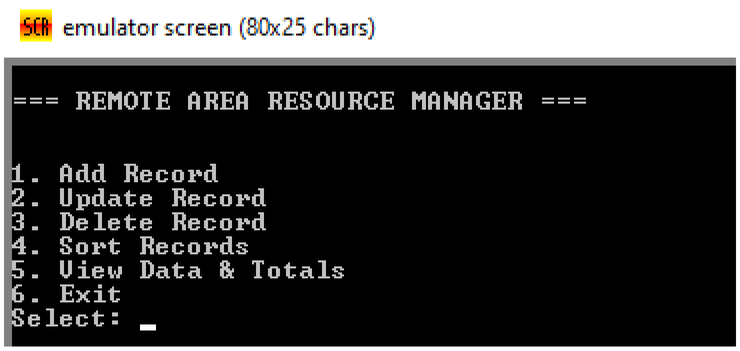
- EMU8086
- DOS Interrupts (INT 21H)
- GitHub for source code hosting

5. Data Description

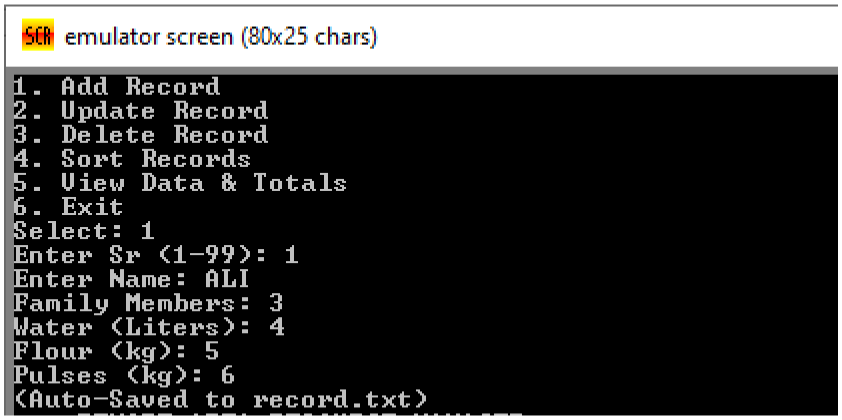
Each record in the system contains the following fields:

- Serial Number (Sr#)
- Name
- Family Members
- Water Consumption (Liters)
- Flour Consumption (Kg)
- Pulses Consumption (Kg)

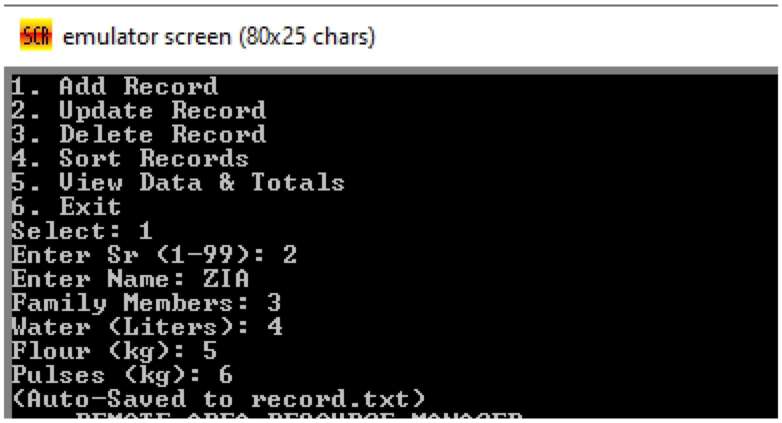
6. Output:



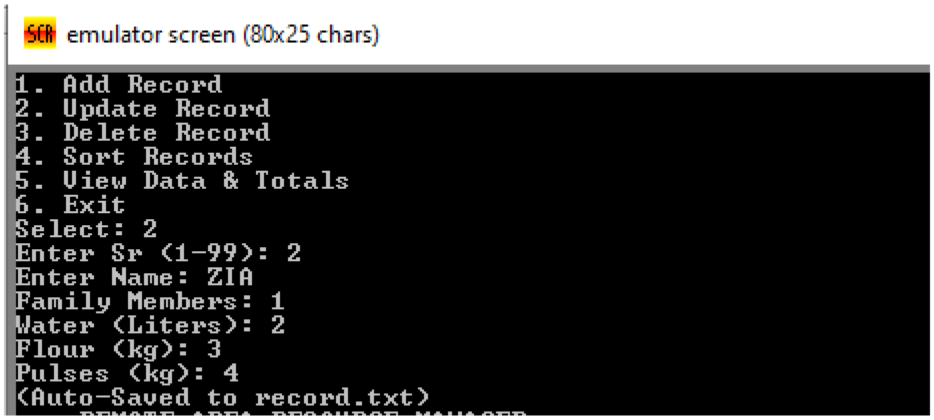
```
=== REMOTE AREA RESOURCE MANAGER ===

1. Add Record
2. Update Record
3. Delete Record
4. Sort Records
5. View Data & Totals
6. Exit
Select: _
```



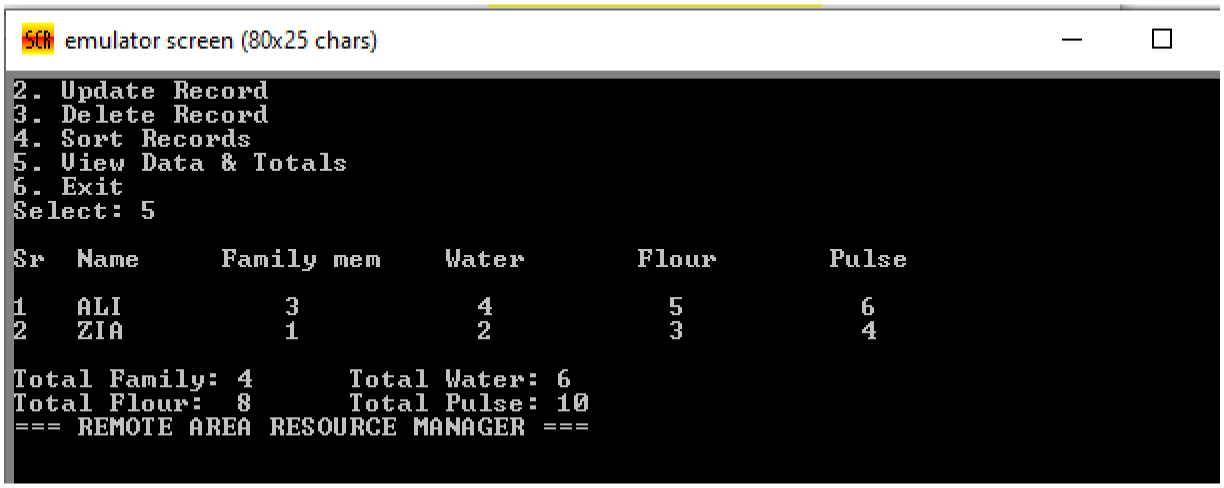
```
1. Add Record
2. Update Record
3. Delete Record
4. Sort Records
5. View Data & Totals
6. Exit
Select: 1
Enter Sr (1-99): 1
Enter Name: ALI
Family Members: 3
Water (Liters): 4
Flour (kg): 5
Pulses (kg): 6
(Auto-Saved to record.txt)
```



```
1. Add Record
2. Update Record
3. Delete Record
4. Sort Records
5. View Data & Totals
6. Exit
Select: 1
Enter Sr (1-99): 2
Enter Name: ZIA
Family Members: 3
Water (Liters): 4
Flour (kg): 5
Pulses (kg): 6
(Auto-Saved to record.txt)
```

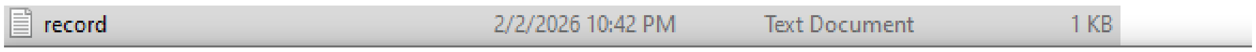


```
1. Add Record
2. Update Record
3. Delete Record
4. Sort Records
5. View Data & Totals
6. Exit
Select: 2
Enter Sr (1-99): 2
Enter Name: ZIA
Family Members: 1
Water (Liters): 2
Flour (kg): 3
Pulses (kg): 4
(Auto-Saved to record.txt)
```

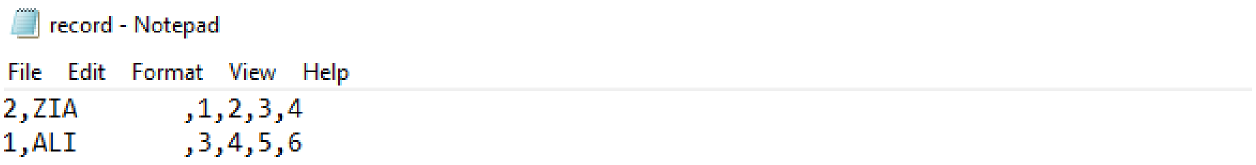


```
2. Update Record
3. Delete Record
4. Sort Records
5. View Data & Totals
6. Exit
Select: 5

Sr   Name       Family mem     Water        Flour        Pulse

1    ALI            3            4            5            6
2    ZIA            1            2            3            4

Total Family: 4       Total Water: 6
Total Flour:  8       Total Pulse: 10
=== REMOTE AREA RESOURCE MANAGER ===
```

TXT FILE:



record	2/2/2026 10:42 PM	Text Document	1 KB



```
2,ZIA        ,1,2,3,4
1,ALI        ,3,4,5,6
```

7. Methodology

The system is implemented using a **menu-driven approach**. Parallel arrays are used to store records to reduce memory overhead. DOS interrupts are used for input/output operations. Sorting is performed using the Bubble Sort algorithm due to its simplicity and suitability for small datasets.

8. Functional Features

- Add new record
- Update existing record
- Delete record
- Duplicate record prevention (using Sr#)
- Sorting based on:
    - Family members
    - Water consumption
    - Flour consumption
    - Pulses consumption
- Display total consumption statistics

9. Algorithm Design

Add Record

1. Check if Sr# already exists
2. If not duplicate, store values in arrays
3. Increment record count

Delete Record

1. Locate record by Sr#
2. Shift remaining records left
3. Decrement record count

Sorting

1. Compare adjacent records
2. Swap all related fields

3. Repeat until sorted

Total Calculation

1. Initialize totals to zero
2. Traverse arrays
3. Accumulate values

10. Implementation Details

- Language: 8086 Assembly Language
- Emulator: EMU8086
- Memory Management: Static arrays
- File Handling: External TXT/CSV (conceptual support)

11. Results and Output

The system successfully manages records and displays correct totals for:

- Family members
- Water consumption
- Flour consumption
- Pulses consumption

12. Conclusion

This project demonstrates how low-level programming can be used to solve real-world problems under hardware constraints. The Remote Area Resource Management System efficiently manages essential data using Assembly Language and fulfills all project requirements.