Phase 2 Documentation

1. Design Document

This document outlines the design and structure of the ARMv7 assembly programs provided, which are divided into three main parts:

- basic_ops.s - Basic Operations Library
- string_ops.s - String Processing Library
- array_ops.s - Array Processing Library

Each part contains specific functions that perform mathematical operations, string manipulations, and array processing tasks.

1.1 Overall Program Structure

The program is structured as a modular assembly codebase where different operations are grouped into separate sections for better organization and reusability.

- Data Section (.data): Stores memory-resident data such as predefined strings, variables, and arrays.
- Text Section (.text): Contains the main execution logic of the program.
- Global Labels (.global): Ensures functions are accessible from other parts of the program.

1.2 basic_ops.s - Basic Operations Library

This section contains fundamental arithmetic, bitwise, memory, and stack operations.

Key Features:

1. Arithmetic Operations: Addition, Multiplication, Division, Overflow handling.
2. Bitwise Operations: XOR, Logical Shift, OR, AND.

3. Memory Operations: Load and Store.

4. Stack Operations: Push and Pop.

Basic Arithmetic Operations

```
MOV R0, #0
MOV R1, #0
MOV R2, #0
MOV R3, #0

ADD R0, R4, R5     @ R0 = R4 + R5
MUL R1, R4, R5     @ R1 = R4 * R5
BVS Overflow_error  @ Check for overflow
```

Division Loop

```
MOV R0, #0
MOV R2, R4
B divlooptest

divloop:
    ADD R0, R0, #1
    SUB R2, R2, R5

divlooptest:
    CMP R2, R5
    BGE divloop  @ Repeat loop until R2 < R5
```

1.3 string_ops.s - String Processing Library

This section implements common string manipulations such as length calculation, copying, comparison, and concatenation.

Key Features:

1. String Length: Iterates through a string and counts characters until a null terminator ( ) is found.

2. String Copy: Copies one string to another memory location.

3. String Compare: Compares two strings and returns 0 if equal, 1 if greater, -1 if smaller.

4. String Concatenation: Merges two strings and stores the result in a predefined memory space.

String Length Function

```
str_length:
    MOV R0, #0
    LDR R1, =lenstring

loop_length:
    LDRB R2, [R1, R3]
    CMP R2, #0
    BEQ str_copy
    ADD R0, #1
    ADD R3, #1
    B loop_length
```

String Concatenation

```
str_concat:
    MOV R4, #0
    LDR R0, =concatstring3
    LDR R1, =concatstring1
    LDR R2, =concatstring2

loop3:
    LDRB R3, [R1, R4]
    STRB R3, [R0, R4]
    ADD R4, #1
    CMP R3, #0x00
    BNE loop3

loop4:
    LDRB R3, [R2, R4]
```

```
    STRB R3, [R0, R4]
    ADD R4, #1
    CMP R3, #0x00
    BNE loop4
```

1.4 array_ops.s - Array Processing Library

This section includes operations related to array manipulation such as sorting, searching, statistics computation, and rotation.

Key Features:

1. Sorting: Implements insertion sort algorithm.
2. Searching: Searches for a predefined value (2 in this case) in the array.
3. Statistics: Computes Minimum, Maximum, and Sum of array elements.
4. Array Rotation: Rotates elements of the array to the right by one position.

Array Sorting Function

```
array_sort:
    PUSH {R4, R5, LR}
    MOV R5, #1

outer_loop_test:
    CMP R5, R1
    BGE sort_done
    B outer_loop

outer_loop:
    LDR R3, [R0, R5, LSL #2]
    MOV R2, R5
    CMP R2, #1
    BGE sort

no_sort:
```

```
    ADD R5, R5, #1
    B outer_loop_test

sort:
    SUB R2, R2, #1
    CMP R2, #0
    BLT no_sort
    LDR R4, [R0, R2, LSL #2]
    CMP R4, R3
    BLE no_sort
    STR R4, [R0, R5, LSL #2]
    STR R3, [R0, R2, LSL #2]
    B sort

sort_done:
    POP {R4, R5, LR}
    BX LR
```

2. Test Plan and Results

Each function is tested with different inputs to verify correctness. The expected behavior is observed using a debugger (such as gdb or qemu-arm).

Test Cases

Test Case	Function	Input	Expected Output
String Length	str_length	"count length"	12
Array Sorting	array_sort	[3, -1, 2]	[-1, 2, 3]
Array Search	array_search	[3, -1, 2], search 2	Index 2
Array Statistics	array_stats	[3, -1, 2]	min: -1, max: 3, sum: 4