Laboratory 1: Basic Assembly Commands

Experiment Sheet

Purpose

The purpose of this laboratory is to learn how to perform operations using basic assembly commands on the ARM architecture. The commands to be used include MOV, LDR, STR, ADD, SUB, MUL, SDIV, AND, and ORR. Additionally, the organization and structure of the program in memory will be discussed.

Essential Knowledge

1. **AREA**: This directive specifies the areas in which the code will reside. The `CODE` area is defined to contain executable code. The `READONLY` statement indicates that this area is read-only.
2. **ALIGN**: Specifies memory alignment. `ALIGN=2` indicates alignment to two bytes.
3. **PROC** and **ENDP**: `PROC` defines the beginning of a procedure, while `ENDP` marks the end of the procedure. This encourages modular programming.
4. **THUMB**: Activates the Thumb mode of the ARM architecture; this mode provides a smaller code size.
5. **EXPORT**: This directive indicates that the following label (e.g., `Start`) can be accessed by other modules or the operating system, marking it as a public entry point for the program.
6. **Label (Start)**: The label `Start` denotes the entry point of the program. Execution begins at this label, allowing for structured flow and clarity in the organization of the code.

Feel free to modify or expand on any of these points!

Used Commands

1. **MOV**: Loads a value into a register. For example, it copies a specific numerical value or the content of another register into the target register.
2. **LDR**: Used to load data from a memory address or to load an immediate value. It transfers a specific value or data obtained from memory into the target register.
3. **STR**: Writes a value from a register to a specified memory address. This allows data to be stored in memory during program execution.
4. **ADD**: Used to add two values. It adds the contents of two registers and stores the result in the target register.
5. **SUB**: Used to compute the difference between two values. It subtracts the value of one register from another and places the result in the target register.
6. **MUL**: Computes the product of two values. The result of the multiplication is stored in a specified register.
7. **SDIV**: Used to divide two values. The result of the division is assigned to the target register.
8. **AND**: Performs a bitwise AND operation on two values. The result is stored in the target register.
9. **ORR**: Performs a bitwise OR operation on two values. The result is transferred to a specified register.
10. **B**: Performs an unconditional branch to a specified label. It is used to control the program flow.