Concordia University

Lab 3: Addressing Modes and 2-D arrays

COEN 311

Lab Section: SN-X

Computer Organization and Software

Penoelo Thibeaud

40212017

Lab Instructor: T. Obuchowicz and Milad Khanchi

Performed on : October 25, 2023

Submitted on : November 8, 2023

I certify that this submission is my original work and meets the Faculty’s Expectations of Originality

# Objective:

This report aims to explore various addressing modes in assembly language programming, focusing on accessing elements of a two-dimensional array stored in memory. Different addressing modes, such as immediate, register, register indirect, post-indexed, pre-indexed, scaled register indirect, and immediate offset, are discussed. The report delves into how these modes facilitate memory access and arithmetic operations, providing insights into the tasks performed by high-level programming language compilers. This exploration allows for an appreciation of the nuances of assembly language and its efficiency in memory access.

# Introduction:

The experiments involve writing assembly language programs to access elements of a 2D array using various addressing modes. The practical application of addressing modes like immediate, register, and register indirect modes is investigated. The main objective is to understand and apply different addressing modes in assembly language to efficiently access memory locations. The experiments are expected to reveal the efficiency and utility of using different addressing modes for specific tasks, leading to a deeper understanding of memory access in assembly language programming.

# Procedure (Methods):

The goal of this lab was to write an ARM assembly language program utilizing the 2-D array address translation formula to access array elements through the register indirect addressing mode, specifically "register indirect with register immediate offset" mode.

## Steps Followed:

1. **Setup of Data Section:**

The .data section of the program was defined with an array representing 3 rows and 2 columns of integers:

.data

array: .byte 3,2,4,1,5,6

1. **Initialization of Registers:**

Two registers were initialized to hold the row and column indices of the desired array element. For example:

mov r1, #1 @ r1 will hold the row index

mov r2, #1 @ r2 will hold the column index

1. **Computation of Offset:**

The mul and add instructions were used to compute the offset value based on the row and column indices.

1. **Accessing the Array Element:**

The calculated offset was used along with the base address of the array to load the desired element into a register through the register indirect addressing mode with offset.

1. **Testing:**

The program was tested using gdb for several typical values of row and column indices to ensure correct functionality. The source code was edited with new values for the row and column indices, re-assembled, re-loaded, and re-run with gdb for each test case. It was not necessary to use loops to iterate over each array element for this lab.

# Results and Discussion:

Understanding different addressing modes is crucial for efficient programming, especially when dealing with memory-intensive tasks.

## Questions:

1. **Determine the value loaded into register r2 by the following ARM assembly language program:**

@ Ted Obuchowicz @ May 23, 2023 @ lab3\_question.s

.syntax unified .cpu cortex-m4 .thumb

.word 0x20000400 .word 0x800000ed .space 0xe4

.data @ declare your data here mydata: .word 0xdeadbeef address\_of\_mydata: .word mydata

.text start: @ assembly code goes here ldr r1, =address\_of\_mydata ldr r1, [r1] ldr r2, [r1]

stop: b stop .end

1. **Consider the following line from the .lst file for the above program:**

15 0000 EFBEADDE mydata: .word 0xdeadbeef

# Conclusion:

The experiments successfully showcased the utility and efficiency of different addressing modes in assembly language programming. The findings align with the objectives stated, providing a comprehensive understanding of how high-level programming language compilers access memory.

# Appendix: