Student Name: Raghav Taneja Student Number: 20058783

CISC-221 Computer Architecture
Assignment 3 Machine-Level Representation of Programs

Instructor: Dave Dove (dove@cs.queensu.ca).

1 Introduction

The focus of this assignment is to understand the ARM assembly-language representation of a machine-level program. It consists of three problems. The first deals with understanding how to locate the operands of an assembly-language instruction. Once we are able to evaluate the operands of an instruction, we move on to problem 2 where we predict the result of each in a sequence of individual assembly language instructions. We then move to a higher level of abstraction and try to understand what the entire sequence of assembly-language instructions does by completing the C language functional equivalent of the assembly-language sequence.

Developing a full understanding of addressing modes is a fundamental requirement for the rest of the material in this topic. It is a prerequisite for understanding what a sequence of assembly-language (or machine language) instructions does.

Going into this topic, you must also understand how multi-byte data is stored in memory. If you are still unclear about how a short, integer, long, or long long data value is stored in a sequence of memory locations (covered as little endian/big endian issues) , you must resolve that lack of understanding first. Get help from the TAs in the lab to help you resolve this issue once and for all before you continue.

2 Logistics

You must do this assignment individually and submit your solution to the associated drop-box in OnQ under your own name, before the scheduled due date and time.

Clarifications and corrections will be posted to the course web site.

3 Handout Instructions

This assignment is formatted as a MS Word Document. Enter your answers in the spaces provided. Save the document and submit as a pdf file to the course web site.

CISC-221 Assignment 3 Worksheet

Resources:

- Archived Lecture Notes: Instructions: Language of the computer.
- Resources page on course web site. A number of assembly language resources listed for the ARM architecture.
- Extras folder: examples of C programs translated into ARM assembly language.

Complete the following problems and submit this worksheet to the course web site.

1. Assume the following values are stored in memory, and in registers as follows:

A 16 element int array alpha is stored in memory starting with address 0x1000. Assume the current values of the array are:

alpha[0] = 1, alpha[1] = 2, alpha[2] = 3, alpha[3]=4, alpha[4] = 5, all other elements of alpha are 0.

An int variable *i* is stored at address 0x1040 and has a value of 2.

Register values:
r0 =10, r1 = 0x1000, r2 = 3, r3 = 4

Fill in the following table showing the values for the individual operands.

Note: the value of an operand is an input to an instruction and independent of the instruction in which it is used. The *Example use* column below may be one of many instructions where the same operand can be use.

Operand	**Example use**	**Value of Operand (1st column) in hexadecimal**
r0	mov r4, r0	**0xA**
=alpha	ldr r4, =alpha	**0x1000**
[r1]	ldr r4, [r1]	**0x1000**
#16	mov r4, #16	**0x10**
[r1, #4]	ldr r4, [r1, #4]	**0x1004**
[r1,r2, asl #2]	ldr r4, [r1,r2, asl #2]	**0x400C**

2. Using the same initial values in memory and in registers used for the previous problem, fill in the following table showing the effects of each of the following instructions, both in terms of the register or memory location that will be updated and the resulting value. Treat each row independent of any other instructions.

Instruction	**Destination**	**Value (in hexadecimal)**
mov r2, #16	**r2**	**0x10**
ldr r3, = i	**r3**	**0x2**
ldr r3 ,= alpha; ldr r2,[r3, #8]	(of 2nd instr) **r2**	**0x1008**
add r2, r3,#1	**r2**	**0x5**
ldr r0, [r1]	**r0**	**0x1000**
add r2,r2,#1; str r2,[r1,#12]	(of 2nd inst) **r1**	**0x1010**

3. Here is the full program that includes the types of statements that you have decoded:

Fill in the missing information in the following C program that is the functional equivalent of the following assembly language program. (You will have to evaluate the results of each instruction and keep a record of current register and memory values as new instructions are executed.)

```
/*Assignment3 for loop*/
        .data
alpha:  .space 64  @reserves 64 bytes
i:      int 0
        .text
        .align    2
        .global   main
        .func main
main:
        ldr       r3, =i
        mov       r2, #0
        str       r2, [r3]
L1:
        ldr       r3, =i
        ldr       r2, [r3]
        cmp       r2, #16
        bge       exit
        ldr       r3, =i
        ldr       r2, [r3]
        add       r1, r2, #0x200
        ldr       r3, =alpha
        str       r1, [r3, r2, asl #2]
        ldr       r3, =i
        ldr       r2, [r3]
        add       r2, r2, #1
        ldr       r3, =i
        str       r2, [r3]
        b L1
exit: mov r7, #1
          swi 0
```

```
int alpha[16];
int i;
void main(){
        for (i=0; i<16; i++){
           alpha[i] = (i + 0x200);
        }
}
```

The TAs will be happy to confirm your answers.