Laboratory 2

Compare-Two-Tables Program

Due Date: Beginning of Week 4 Lab

Introduction

An array is just a group of items or elements stored in consecutive memory locations. Arrays are also called tables, lists, and sometimes strings, and we will often use these terms interchangeably. There are two pieces of information that define an array: where the array is stored in memory and the size and type of each array element. For this course, we will assume that arrays only contain one type of element. For example, we may have an array of all one-byte unsigned numbers, an array of all two-byte signed numbers, etc.

There are three main methods of defining where an array is stored in memory.

- Starting address and length: We supply the memory address where the first element is located in memory and the number of items (sometimes the number of bytes or addresses, and this does not always yield the same number).
- Starting address and final address: We supply the memory address of the first element and the memory address of the last element.
- Starting address with end-of-array value: We supply the memory address of the first element of the array. After the last element of the array, a predefined value is stored indicating that the end of the array has been reached.

Assignment

The flowchart in Figure 1 represents an algorithm for counting how many $64's appear in a list of unsigned one-byte values. The constructs are labeled to show that this is a structured program, but this is typically not notated. Notice that the only assumption made was that the starting address of the array was available (since it is supplied in all three methods above) and that the algorithm can determine when it has reached the end of the array. Also, none of the S12's registers were referenced in the flowchart. This is a decision that should be postponed until writing the code whenever possible.

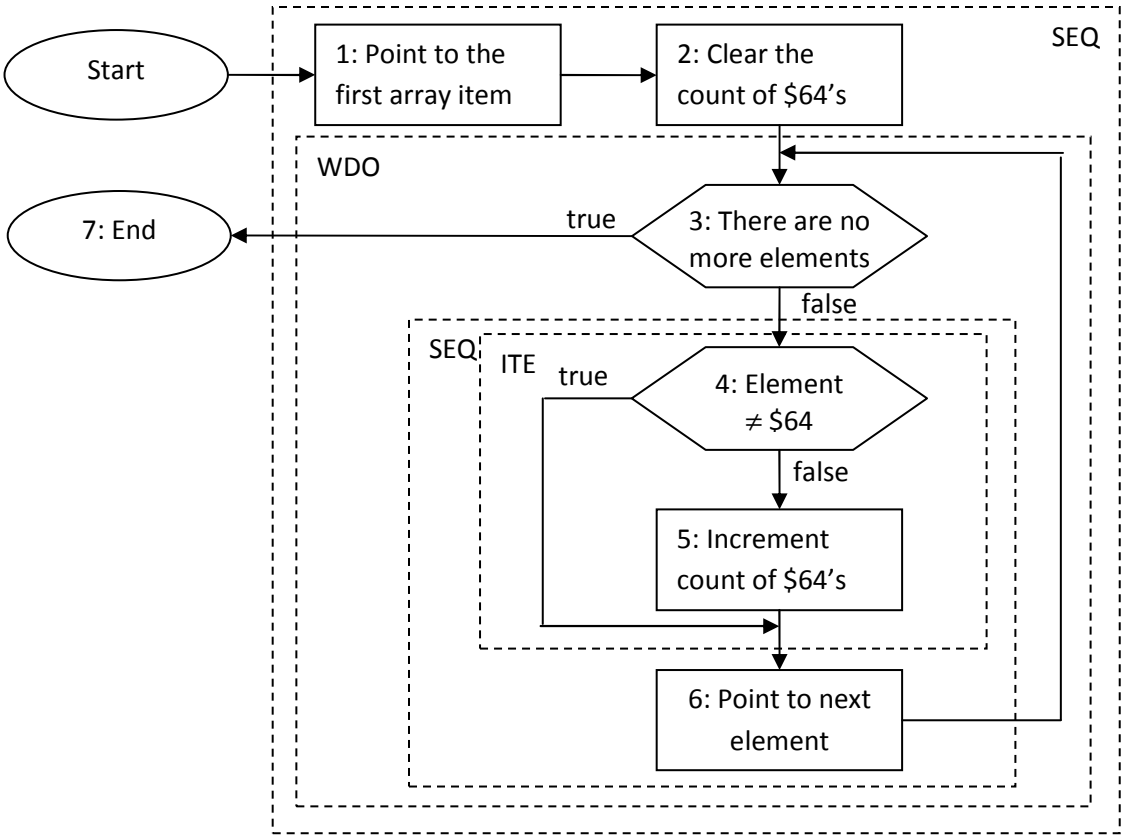

Figure 1 – Flowchart to count the number of 64h's in a list

To convert the flowchart into code, more details are needed. First, this list corresponds to the grades for a Microcomputer's I quiz, and only values between $00 and $64 are possible. The value $FF can be used as an end-of-array value, since there is no possibility for $FF to be a valid number in the array. Therefore, the array of values will be specified by supplying the address of the first element in address $1000, and a value of $FF will indicate the end.

Next, we need a means of reporting the answer which is usually done by storing it to a predetermined address. Addresses $1000 and $1001 are already used (addresses require two bytes), so we will return the answer in $1002. By only reserving one byte for the answer, our program will be limited to arrays up to 255 elements.

The details we've just discussed are typically represented as a list of requirements, shown below. Each requirement item discusses only one very specific aspect of the program listed in Figure 2.

1. The starting address of the input array is supplied in address $1000.
2. The end of the array is indicated by the value $FF.
3. The program must correctly handle arrays up to 255 elements not including the end-of-array value.
4. The correct number of locations containing $64 is returned as a one-byte value in address $1002.

Code Line	Memory Address	Contents	Assembly
1:	2000	FE	LDX $1000
	2001	10	
	2002	00	
2:	2003	18	MOVB #0,$1002
	2004	0B	
	2005	00	
	2006	10	
	2007	02	
3:	2008	A6	LDAA 0,X
	2009	00	
4:	200A	81	CMPA #$FF
	200B	FF	
5:	200C	27	BEQ $0A
	200D	0A	
6:	200E	81	CMPA #$64
	200F	64	
7:	2010	26	BNE $03
	2011	03	
8:	2012	72	INC $1002
	2013	10	
	2014	02	
9:	2015	08	INX
10:	2016	20	BRA -16
	2017	F0	
11:	2018	3F	SWI

Figure 2 - Program to count $64's

The program in Figure 2 correctly meets these requirements. Perform the following tasks.

1. Enter the program into the memory locations specified.
2. Enter the values $64, $45, $FF, and $64 in locations $3000 to $3003.
3. Enter the value $3000 into locations $1000 and $1001.
4. Complete a program trace, including the PC, and registers used, and the N, Z, V, and C condition code bits.

   Question 1: Notice that the items in the flowchart have been numbered. For each line of the program, give the number of the flowchart object that corresponds to the assembly code. For example, Code Line 11 corresponds to flowchart item 7.

   Question 2: What changes must be made to the program to process a list that begins at address $3800?

   Question 3: What changes must be made to the program to store the answer at address $1100?

   Question 4: What changes must be made to the program to handle a list with a two-byte length? (Hint: This means that the answer may require two bytes, and two changes are required.)
5. Write a program that compares two equal-length arrays of numbers byte-by-byte and determines if all pairs of bytes are the same (i.e. the arrays are identical). Your program must meet the following requirements.
   a. The address of the first array is supplied in location $1000.
   b. The address of the second array is supplied in location $1002.
   c. The two-byte length of both arrays is supplied in location $1004.
   d. If the two arrays contain the same bytes, the program should store the result $FF in location $1006, otherwise it should store the result $00 to location $1006.
   e. Arrays with lengths of $0000 should be reported as identical.
6. Enter your program into the Dragon12+ board and test your program. When it is working, demonstrate it to the instructor. **Important**: Only addresses $1000 to $35FF are available for use in lab.

What to Demonstrate/Submit

- Demonstrate the program to compare two tables
- Typed answers to all questions, including program trace
- Printout of the .lst file and email a copy of the .lst file to dfoster@kettering.edu