Department of Engineering Technology and Industrial Distribution
ESET 349 Microcontroller Architecture

Lab 7 – Binary Search

Objectives

1. Understand how to search for a value in an array.
2. Write an assembly language program to perform binary search.

Your Tasks

1. Read and understand how binary search works in a sorted array.
2. Write ARM assembly code based off the code overview below to perform binary search.
3. Demonstrate the functioning of your program in the Keil simulator.

Overview

In this lab, you will be implementing a binary search algorithm in ARM assembly code. Binary search is an efficient algorithm for finding a specific value in a sorted array of values. It works by repeatedly dividing the search interval in half until the target value is found or the search interval is empty.

Given a pre-defined array of integers sorted in ascending order and a target value to search for, your task is to find the index of the target value (if found) using binary search. The resulting index of the target value in the array (0-indexed) will be stored in the R12 register if it is found. If the target value is not found, R12 will be set to the length of the array to indicate failure.

To complete this lab, you will need to have a basic understanding of ARM assembly language, including the instruction set, registers, and memory addressing modes. You will also need to understand the binary search algorithm and its implementation details.

At the end of the lab, you should have a working binary search algorithm in ARM assembly code that can search for a target value in a sorted array of integers. You will also gain a deeper understanding of ARM assembly language and its use in algorithm development.

Algorithm

You may refer to the Wikipedia article on binary search for the algorithm and its explanation.

Code Overview

The following instructions will detail how the code you are to write should work.

The binary search begins by setting the left and right indices of the search space. The left index is initially set to 0, and the right index is initially set to the length of the array minus 1.

The search then enters a loop that continues until either the target value is found, or the search space is exhausted. On each iteration of the loop, the middle index of the search space is calculated as the average of the left and right indices. If the middle index is not integral, round it to the nearest smaller integer (floor).

If the array element at the middle index is equal to the target value, the loop terminates, and the 0-index of the target element (which would be middle) is stored in the R12 register. If the middle element is less than the target value, the search continues in the right half of the search space. If the middle element is greater than the target value, the search continues in the left half of the search space.

If the target value is not found, the loop terminates and the R12 register is set to the length to indicate failure. The code then enters an infinite loop to prevent the program from exiting.

Fig. 1 shows an overview of the structure of your program. Please ensure that your program will continue to work when the words, len or target values defined in lines 5-7 are changed.

```
1          ; perform binary search
2          ; result indicated by R12 (len : fail, other : 0-index of target)
3
4          area lab7, code, readonly
5  words   DCD 0x01, 0x02, 0x04, 0x06, 0x08, 0x10, 0x11
6  len     equ 0x07
7  target  equ 0x11
8          export __main
9
10 __main  proc
11
12         ; Perform binary search
13
14         ; PERFORM BINARY SEARCH ON THE ARRAY OF SIZE "len" BEGINNING AT THE
15         ; MEMORY ADDRESS "sort" TO FIND THE INDEX OF THE TARGET "target".
16         ; IF FOUND, STORE THE INDEX IN R12 (INDEX BEGINS FROM 0).
17         ; IF NOT FOUND, R12 SHOULD BE "len".
18
19 stay    B stay
20
21         endp
22             end
```

Figure 1. Binary search program structure