# Laboratory 4

Title of the Laboratory Exercise: Array manipulation

1. Introduction and Purpose of Experiment

Students will be able to access elements in an array using indexed addressing mode.

1. Aim and Objectives

Aim

To develop assembly language program to access the elements in an array or to access a particular element in an array using indexed addressing mode.

Objectives

At the end of this lab, the student will be able to

* + Perform array manipulation using indexed addressing mode
  + Discuss indexed addressing modes
  + Access the appropriate element in an array

1. Experimental Procedure

1. Write algorithm to solve the given problem

2. Translate the algorithm to assembly language code

3. Run the assembly code in GNU assembler

4. Create a laboratory report documenting the work

1. Questions:
   1. Assume an array of 25 integers. Translate this C statement/assignment using the indexed form:

*x = array [15] + y*

* 1. Create an array with 10 integers. Develop an assembly language program to print the elements of the array.
  2. Develop assembly language program to perform the following array assignment in C:

for ( i = 0; i < 10; i++) {

a[i] = 10;

}

* 1. Develop an assembly language program to generate the first n numbers in Fibonacci series.

1. Calculations/Computations/Algorithms

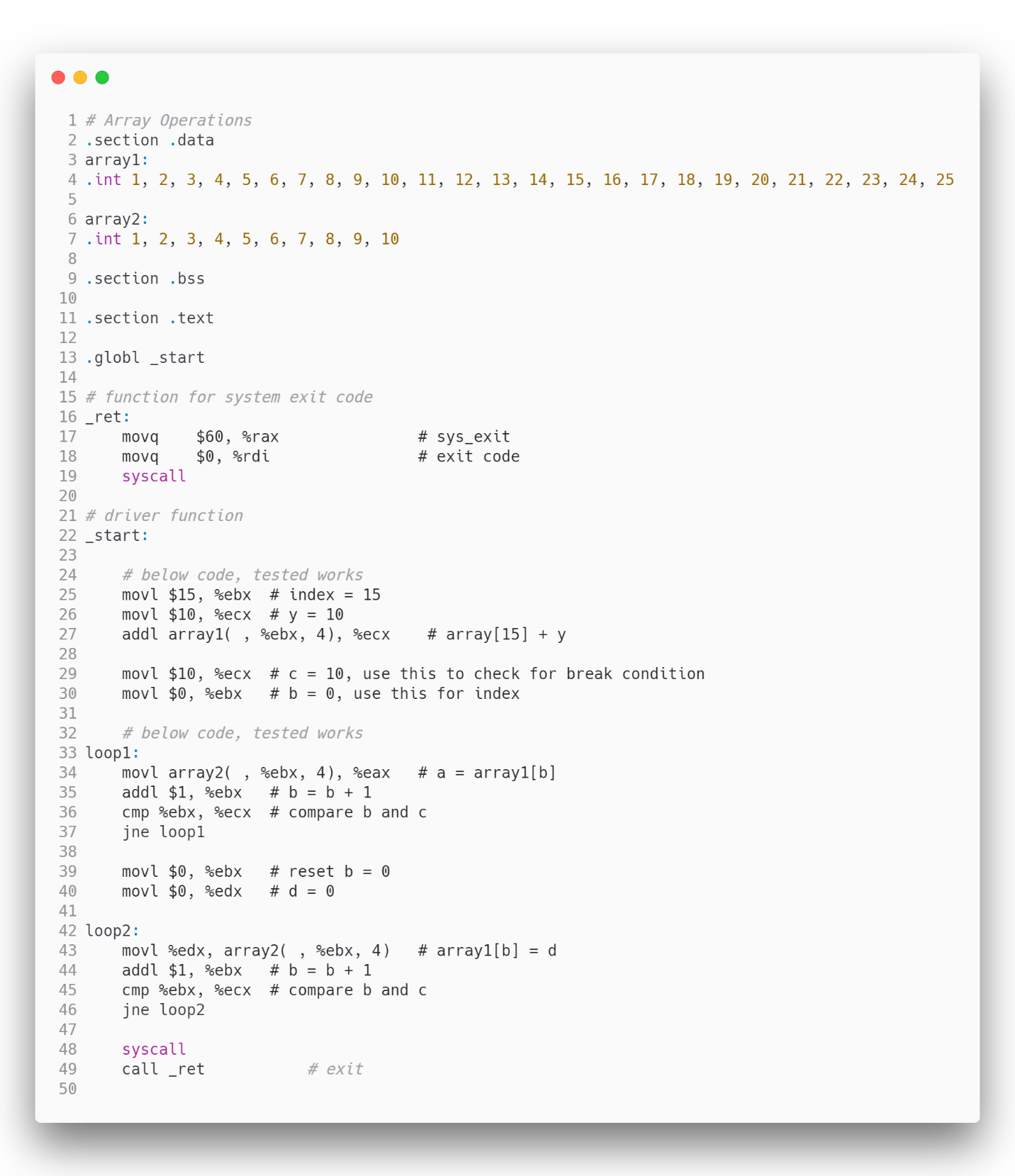


Figure 1 source code for array operations



Figure 2 source code for fibonacci series

1. Presentation of Results

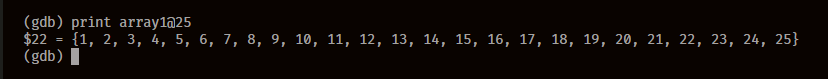


Figure 3 Data stored in array1

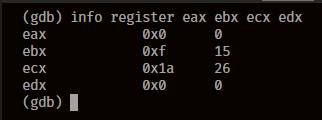


Figure 4 array1[15] + y; y = 10 ; where y = ecx

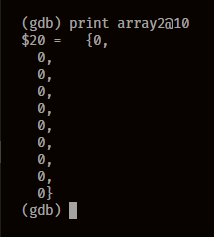
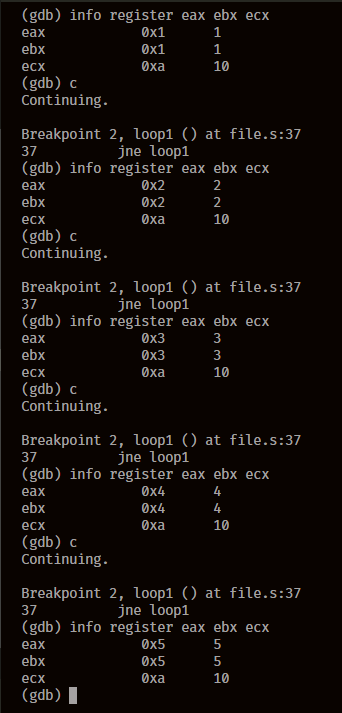


Figure 5 status of array2 after array[i] = 0 operation

Figure 6 Looping over the array and fetching values

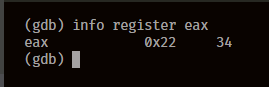


Figure 7 fibonacci number for n = 10

1. Analysis and Discussions

|  |  |
| --- | --- |
| Code | array(base\_offset, index, size) |
| Example | movl $15, %ebx  movl array(, %ebx, 4) %eax |
| Explanation | Performs:  eax = array[15]  Description:  Access the nth element of the array using the above syntax, here the base\_offset is the amount of memory offset from the array starting memory address, the index is the location of the data element to be accessed, and the size is the individual element size of the array, in this case it was taken to be 4 since we have 32-bit data stored in the array.  The index is usually given in a register and immediate values aren’t allowed for the index.  Basically the array name stores the base memory address of the array, by using base\_offset, index and size we are basically performing array + base\_offset + index \* size to obtain the memory location of the element at the index, and then the data at that location is fetched. |

1. Conclusions

To access an array in assembly language, we use a pointer. A pointer is simply a register or variable that contains a memory address.

The value in the pointer is computed as shown in the previous sections by adding the base address of the array and the offset of the desired element.

Part of the computation can be done using offset addressing mode, but note that the offset in offset addressing mode is in bytes, and does not account for the size of an element. For example, if working with words in assembly, we must manually multiply the offset by 4.

1. Comments

1. Limitations of Experiments

The experiment was designed for only preallocated and preinitialized array, i.e. the length of the array is fixed and the values are initialized, there are no cases for uninitialized arrays, and memory reallocation is not used.

2. Limitations of Results

The results are limited to finite number of values in the array since in this experiment we cannot define a n length long array, the length of the array is dependent upon the values given to the variable in the data segment.

3. Learning happened

We learnt ways to access elements of an array, and manipulate values stored in them in various ways. We also learnt the usage of loops to perform operations, such as that of calculating the nth Fibonacci number.

4. Recommendations

When direct access is given to array’s using addresses, it has a very notorious problem of segmentation faults, any address space that is not assigned to the current thread when tried to be dereferenced gives a segmentation fault, this needs to be taken care of by the programmer.

Signature and date Marks