CCCN221 – ComputerArchitecture

Lab 7 – Arrays in MIPS

In a High-Level Language, an array is a multi-valued variable: a single array variable can contain many values. Arrays in MIPS assembly will be similar; however, the abstraction of an array is very much constrained in assembly. In MIPS assembly an array is implemented by storing multiple values in contiguous areas of memory and accessing each value in the array as an offset of the array value.
An array is a multivalued variable stored in a contiguous area of memory that contains elements that are all the same size.

Allocating Array in Memory

In some languages, such as Java, arrays can only be allocated on the heap. Others, such as C/C++ or C#, allow arrays of some types to be allocated anywhere in memory. In MIPS assembly, arrays can be allocated in any part of memory.

To allocate an array in static data, a label is defined to give the base address of the array, and enough space for the array elements is allocated. Note also that the array must take into account any alignment consideration (e.g. words must fall on word boundaries). The following code fragment allocates an array of 10 integer words in the data segment.

```
.data
array: .space 40
```

Printing an Array

This first program presented here shows how to access arrays by creating a PrintIntArray subprogram that prints the elements in an integer array. Two variables are passed into the subprogram, $a0 which is the base address of the array, and $a1, which is the number of elements to print. The subprogram processes the array in a counter loop, and prints out each element followed by a ",". The pseudo code for this subprogram follows.

```
Subprogram PrintIntArray(array, size)
{
print("[")
for (int i = 0; i < size; i++)
{
print("," + array[i])
}
print("]")
}
```

Example 1: Declaring Array and printing first value.

```
.data
myArray: .space 12

.text
.globl main

main:

addi $s0, $zero,4     # $s0=0+4=4
addi $s1, $zero,10   # $s0=0+10=10
addi $s2, $zero,12   # $s0=0+12=12

# Index = $t0
addi $t0, $zero, 0     # $t0=0+0=0

sw $s0, myArray($t0)    # 4 will be stored at 0 index
addi $t0, $t0, 4        #

sw $s1, myArray($t0)   # 10 will be stored at 1 index
addi $t0, $t0, 4       # $t0=0+4=4
sw $s2, myArray($t0)    # 12 will be stored at 2 index


li $v0, 1             # To display the fetched value
syscall

li $v0, 10          # To terminate the program
syscall

.end main
```

Example 2: Printing Array values

```
.data
        myArray: .space 12
        newline: .asciiz "\n"

.text
.globl main
main:
            addi $s0, $zero,4
            addi $s1, $zero,10
```

```
    addi $s2, $zero,12

# Index = $t0
    addi $t0, $zero, 0

    sw $s0, myArray($t0)
    addi $t0, $t0, 4
    sw $s1, myArray($t0)
    addi $t0, $t0, 4
    sw $s2, myArray($t0)

    #clear $t0 to 0
    addi $t0,$zero,0
   while:
     beq $t0,12, exit

     lw $t6, myArray($t0)
     addi $t0, $t0, 4

#print the numbers
    li $v0,1
    move $a0, $t6
    syscall

# print a new line
   li $v0, 4
   la $a0, newline
   syscall

   j while

   exit:
    #tell system, this is end of program
     li $v0, 10
     syscall
```

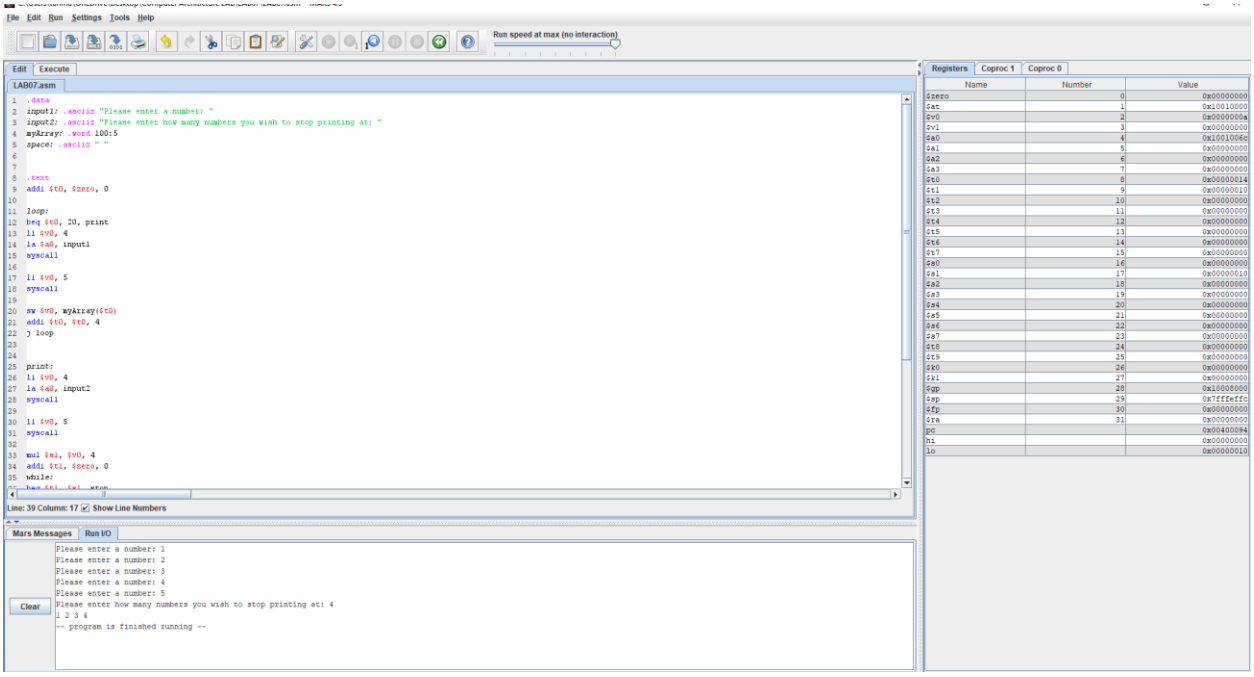
Example 3: Array Initialization

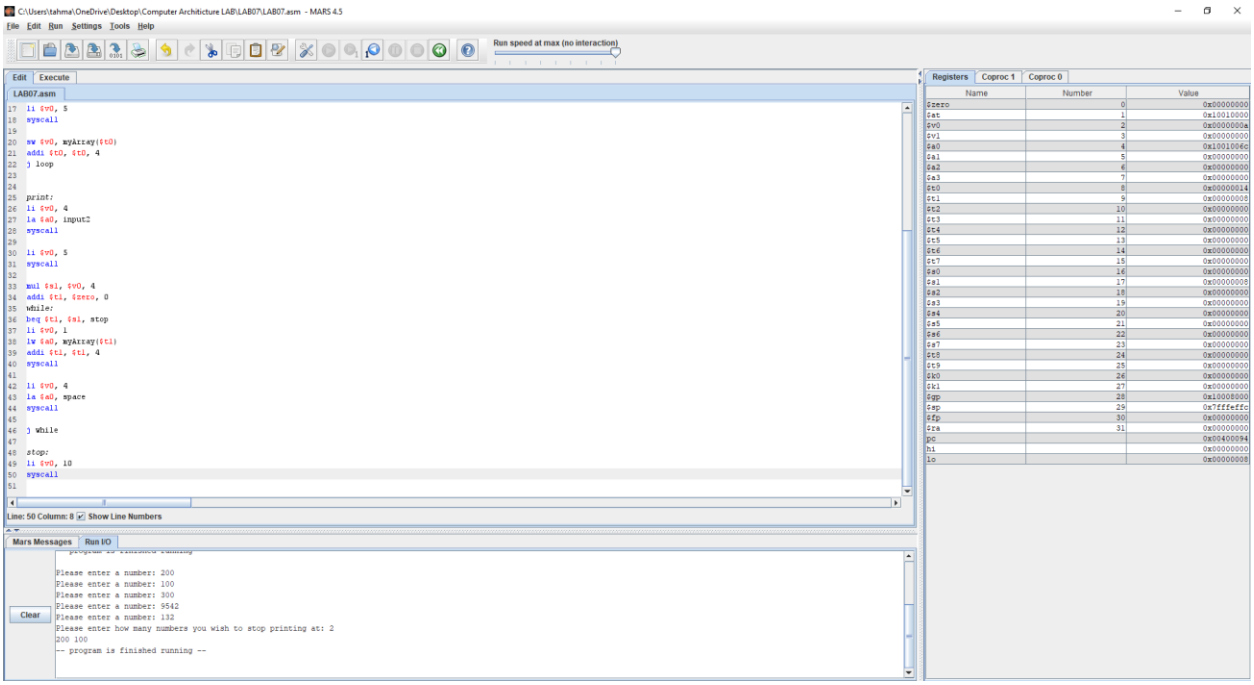
```
.data
        myArray: .word 100:5
        newline: .asciiz "\n"

.text
.globl main
main:
```

```
    #clear $t0 to 0
    addi $t0,$zero,0
   while:
     beq $t0,20, exit

     lw $t6, myArray($t0)
     addi $t0, $t0, 4

#print the numbers
    li $v0,1
    move $a0, $t6
    syscall

# print a new line
   li $v0, 4
   la $a0, newline
   syscall

   j while

   exit:
    #tell system, this is end of program
     li $v0, 10
     syscall
```

Practice

- Write a program that prompts the user to enter an integer value in an array, and user can also print out some values.





.data

input1: .asciiz "Please enter a number: "

input2: .asciiz "Please enter how many numbers you wish to stop printing at: "

```
myArray: .word 100:5

space: .asciiz " "



.text

addi $t0, $zero, 0



loop:

beq $t0, 20, print

li $v0, 4

la $a0, input1

syscall



li $v0, 5

syscall



sw $v0, myArray($t0)

addi $t0, $t0, 4

j loop
```

```
print:
li $v0, 4
la $a0, input2
syscall

li $v0, 5
syscall

mul $s1, $v0, 4
addi $t1, $zero, 0
while:
beq $t1, $s1, stop
li $v0, 1
lw $a0, myArray($t1)
addi $t1, $t1, 4
syscall
```

```
li $v0, 4

la $a0, space

syscall


j while


stop:

li $v0, 10

syscall
```