CS312 : MIPS Assembly Programming
Assignment 6: Arrays and Strings

Instructor: Dr. Sukarn Agarwal,
Assistant Professor,
Department of CSE,
IIT (BHU) Varanasi

March 10, 2021

Arrays

An array is a collection of similar data-type variables. In MIPS assembly programming, accessing an array requires loading the base address into the register.

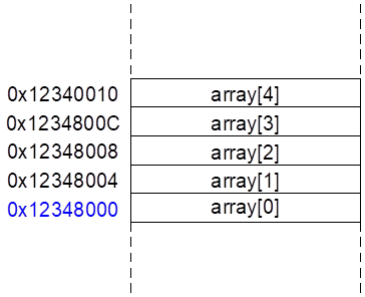

Figure 1: Representational view of array elements

An array can store a large amount of data. Storing these data in the limited number of registers (32) is infeasible. Hence, arrays are stored in the data segment of the program. Essentially, there are three different operations that can perform on the array:

- Receiving the data from the particular index of array into the variable, e.g., `x = list[i];`
- Writing the data from the variable to the particular index of the array, e.g., `list[i] = x;`
- Knowing the length of the array, i.e. `list.length`

To access the data elements in the array, you need to know the array's data address and then use the load word (`lw`) or store word (`sw`) instructions. Recall that the size of the words in the MIPS is 32 bits or 4 bytes. One of the way of declaring an array is

```
array: .word 5, 4, 2, 1, 3, 4, -2, -1, 0, 9
```

The following snippet of the code will place the value of array[4] into the `$s3`:

```
la $s2, array # Assign the address of array into $s2
li $s1, 4 # Place the element index into $s1
add $s1, $s1, $s1  # double the index
add $s1, $s1, $s1   # double the index again (now 4 times)
add $s4, $s1, $s2   # access the index element by adding into the base addresss
lw $s3, 0($s4)  # Assign the value fom the index element of the array to $s3
```

In case if you want to assign the content of `$s3` to array[4], use the following line in place of the load word instruction

```
sw $s3, 0($s4)
```

Some other ways for 1-Dimensional and 2-Dimensional array declaration

1-Dimensional Array

- Different ways to declare string array of 10 characters are as follows:

  ```
  1. my_char_array:    .byte   0:10 (intial value of each element
   in the array: Null \0)
  2. my_char_array:    .space   10
  ```

  The difference between the two declarations is that the first way initializes the memory space before the program begins execution. In contrast, the second way does not initialize memory. It allocates the 10 bytes but does not change their contents before the program begins its execution.

- In case if you need to declare the string array of 10 characters with their initial value as 'A'. There are two different ways:

  ```
  my_char_array:    .byte   'A':13

  Alternatively:

  my_char_array:    .byte   65:13 (usig ASCII Format)
  ```

- Different ways to declare integer array of size 36 are as follows:

  ```
  int_array:  .word   0:36  (intial value of each element: 0)
  int_array:  .word   2:36  (intial value of each element: 2)
  ```

  The other different way to initialize the array element with different values are given at the above code snippet.

2-Dimensional Array

- A declaration of a 2-dimensional array reduces to the allocation of the correct amount of contiguous memory. The base address identifies the first element of the first row within the first column. Consider an example of a string array of 2 by 4. The declaration is as follows:

```
char_2D_Array:    .space   8     # 2 by 4 = 8 bytes of allocated space
```

- An alternative realistic way of declaring an array is given below. Consider an example of an integer array of 5 by 6, where each element of the array is initialized to 9.

```
int_arr:    .word   9:6
.word   9:6
.word   9:6
.word   9:6
.word   9:6
```

Problem 1: For a given two arrays (size 6) of name and student marks (out of 100) of four subjects, write a MIPS assembly program that lets the user enter the name of a student as an input. The program checks whether the input name is matched with the given string array of the names. If matched, the program prints the marks list of the corresponding student. The two possible output formats of the program are as follows:

```
Output 1: Enter Student Name: Abhilash
The marks of student is as follows:
Physics: 85
Chemistry: 70
Mathematics: 95
Biology: 90

Output 2: Enter Student Name: Tanmay
  Student name is not matched
```

Problem 2: Write a MIPS assembly program that implements the bubble sort for an integer array of size 10. The user provides the input value of the integers for the array. The program prompts the following output on the screen:

```
Before Bubble Sort: 2, 3, 1, 0, 5, 4, 7, 6, 9, 8
Iteration 1: 2, 1, 0, 3, 4, 5, 6, 7, 8, 9
Iteration 2: 1, 0, 2, 3, 4, 5, 6, 7, 8, 9
..
..
..
After Bubble Sort: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

Problem 3: Write a MIPS assembly program that convert all lower case user input string to upper case string. The string may contain any ASCII character (For e.g., the string can include ,.+-1,2). The possible output format for the given program are as follows:

```
Enter the input string: +@3d,Fgh
Output String with Upper Case Letter: +@3D,FGH
```

Problem 4: Write a MIPS assembly program that multiply two double precision floating point matrix of order 3 by 3. The two matrix are stored in the row major order. The input for the matrices are provided by the user.

Note: Submit all of your source code and final screen shot of the register panels (both integer and floating point) and console output to the google classroom portal on the end of the day of 20th March 2021 (Indian Standard Time). Further any copy case between the assignments results into the zero marks. In case of any doubt(s) regarding the assignment, you can contact TA: Nirbhay and Deepika.