DHA Suffa University
Department of Computer Science
Computer Organization & Assembly Language
Spring 2021
Lab # 12 (2D- Arrays)

Objective:

Representation and manipulation of two dimensional arrays in MIPS.

Multidimensional Arrays:

Memory is organized as a single-dimensional array. Two dimensional arrays must be treated as simple single-dimensional arrays. Then, in assembly language to declare two-dimensional arrays, we have to arrange the arrays as single-dimensional arrays. To do this, we have to know how to organize all elements of an array. There are two different ways to organize the elements of two-dimensional array:

1. Row-major order:

The array is organized as a sequence of ROWS. Most of programming languages such as C follow this method.

2. Column-major order:

The array is organized as a sequence of COLUMNS. This ordering is being implemented in FORTRAN.

Let we have an array A:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Row-major order,
e.g., C

Address	Value
0	a_{11}
1	a_{12}
2	a_{13}
3	a_{21}
4	a_{22}
5	a_{23}

Column-major order,
e.g., Fortran

Address	Value
0	a_{11}
1	a_{21}
2	a_{12}
3	a_{22}
4	a_{13}
5	a_{23}

Address Calculation:

Assume the row and column index starts from 0. The general formula to calculate the byte address of the element [a, b] can be expressed as:

Row-major order: Starting Address + Size-of-element * (a * number-of-columns + b)

Column-major order: Starting Address + Size-of-element * (b * number-of-rows + a)

Example 1: Extracting a value at index [2,3]

```
.data
Array:   .word  10,20,30,40
         .word  50,60,70,80
         .word  90,100,110,120
         .word  130,140,150,160
.text
.globl main

main:
         la     $t0 , Array
         li     $t1 , 2   # Row no.
         li     $t2 , 3   # Column no.
         li     $t4 , 4   # No. of columns in the array

         mul    $t3 , $t1, $t4    # (Row No. * No. of columns in the array)
         add    $t3 , $t3, $t2    # [(Row No. * No. of columns in the array) + Column No.]
         mul    $t3 , $t3, 4      # (((Row No. * No. of columns in the array) + Column No.) * (Size of
element))
         add    $t3 , $t3, $t0    # [(((Row No. * No. of columns in the array) + Column No.) * (Size of
element)) + Base address of array]
         lw     $t5 , ($t3)

         li     $v0 , 1
         move   $a0 , $t5
         syscall
```

```
        li      $v0 , 10
        syscall
```

Example 2: Sum of values in Row 2

```
.data
Array:  .word  10,20,30,40
        .word  50,60,70,80
        .word  90,100,110,120
        .word  130,140,150,160
.text
.globl main

main:
        la      $t0 , Array
        li      $t1 , 1  # Row no.
        li      $t2 , 0  # Column no.
        li      $t4 , 4  # No. of columns in the array
        li      $t7, 0

        mul     $t3 , $t1, $t4   # (Row No. * No. of columns in the array)
        add     $t3 , $t3, $t2   # [(Row No. * No. of columns in the array) + Column No.]
        mul     $t3 , $t3, 4     # (((Row No. * No. of columns in the array) + Column No.) *
(Size of element))
        add     $t3 , $t3, $t0   # [(((Row No. * No. of columns in the array) + Column No.) *
(Size of element)) + Base address of array]



        Sum:
        beq $t7,$t4 , exit
        lw  $t6 ,($t3)
        add $t5, $t5, $t6
        add $t3, $t3, $t4
        add $t7, $t7, 1
        b Sum


        exit:
        li      $v0 , 1
        move    $a0 , $t5
        syscall
        li      $v0 , 10
        syscall
```

Lab Task:

(1) Using the following 2D array declaration in which each row represents the marks of each student in the four subjects. You are now required to print the maximum score of each student on the console.

```
class_marks:    .word 10,18,12,13
                .word 20,21,28,23
                .word 32,31,34,33
                .word 40,41,42,48
                .word 50,51,52,55
                .word 60,61,66,63
                .word 70,71,76,73
                .word 80,88,82,83
                .word 90,94,92,93
                .word 99,96,97,98
```

(2) Let the following array as matrix; write the MIPS code to transpose it.

```
class_marks:    .word 10,11,12,13
                .word 20,21,22,23
                .word 30,31,32,33
                .word 40,41,42,43
```