

Arrays

Integer Array

Without initialization

.data

arr: .space 40

integer
is 4
bytes

```
int arr[10];
```

With initialization

.data

arr: .word 1, 2, 3, 4

```
int arr[] = {1, 2, 3, 4};
```

With default value

.data

arr: .word 50:3

```
int arr[3] = {50, 50, 50};
```

Accessing Elements in the Integer Array

- To access the i^{th} element, displace by $i*4$ from the starting address

.text

addi R1, \$zero, 4	# size of integer
addi R2, \$zero, 9	# index (9) of element to be accessed
mul R3, R1, R2	# displacement from starting index

- $\text{arr}[9] \leftarrow R4$

sw R4, arr(R3)

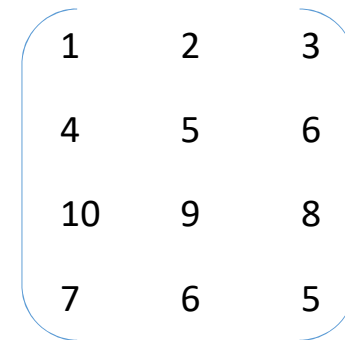
- $R4 \leftarrow \text{arr}[9]$

lw R4, arr(R3)

2 Dimensional Integer Array

.data

```
arr:      .word    1, 2, 3
          .word    4, 5, 6
          .word   10, 9, 8
          .word    7, 6, 5
numCol:   .word    3
numRow:   .word    4
```



1	2	3
4	5	6
10	9	8
7	6	5

Accessing a 2D Integer Array

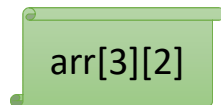
- To access element in row 'i' and column 'j', displace by the following from the start index

$$(i * \text{numCol} + j) * 4$$



.text

```
addi R1, $zero, 3
addi R2, $zero, 2
lw R3, numCol
mul R1, R3, R1
add R1, R2, R1
mul R1, R1, 4
```



- $\text{arr}[3][2] \leftarrow R4$
sw R4, arr(R1)
- $R4 \leftarrow \text{arr}[3][2]$
lw R4, arr(R1)

Assignment

- Floating point array
- Char array
- Array of Strings