MSU CSC 285
Handout: Arrays, lw and sw instruction

Java	MIPS assembly

`int grades[] = {92, 79, 86, 82, 95};`	`.data`
	`grades: .word 92, 79, 86, 82, 95`

`grades.length` has value 5 Valid indices are 0, 1, 2, 3, 4 (That is, 0 to `grades.length - 1`) Invalid indices cause exception	**No equivalent operation!** **No boundary checking!** • *Programmer's responsibility to use valid indices*

`nextValue = grades[0];`
`nextValue = grades[1];`
`nextValue = grades[2];`
`OR`
`nextValue = grades[i];`
 	
*The expresssion "`grades[i]`" contains **two** variables (`grades` and `i`)*
	`la $s7, grades`
	`lw $s0, 0($s7) # grades[0]`
	`lw $s0, 4($s7) # grades[1]`
	`lw $s0, 8($s7) # grades[2]`
	• *The expresssion "0($s7)" contains **one** variable ($s7) and **one** constant*
	• *The register is called the "base register" and holds the "base address"*
	• *The constant is the "offset"*
	• *The address of the element we're accessing is (base address + offset)*
	• *The constant is "always" a multiple of 4 because there are four bytes in a word*

`int sum = 0;`
`for (i = 0; i < grades.length; i++)`
`{`
`sum += grades[i];`
`}`
 	
The variable `grades` stays the same and the variable `i` changes value
	`add $s2, $zero, $zero # sum = 0`
	`la $s7, grades`
	`# top of loop`
	`lw $s0, 0($s7) # next element`
	`add $s2, $s2, $s0 # sum += g[i]`
	`addi $s7, $s7, 4 #increment addr.`
	`# bottom of loop`
	
	• *In the expresssion "0($s7)" it's the constant that stays the same, and the address in the register that changes!*
	• *In effect, we're always accessing element 0 but the array is in a different place!*

`grades[4] = 88;`	`la $s7, grades`
	`addi $s0, $zero, 88`
	`sw $s0, 16($s7) # grades[4] = 88`
	
	• *The order of operands to `sw` is the same as the order of operands to `lw`*

```
# First MIPS program, shows MIPS registers and operations
# -----------------------------------

.data

array: .word 0x100

.text

addi $s0, $zero, 42

la $s1, array

sw $s0, 4($s1)
```