Mohammad Iqbal

CSC 210

12/02/2024

**Detailed Analysis of MIPS Bubble Sort Code**

**General Overview**

The MIPS assembly code implements a bubble sort algorithm to sort an array of integers. The program uses structured procedures for modularity and incorporates macros for reusable functionalities like printing. The design separates the data (array and messages) from the execution logic, enhancing clarity.

**Key Components**

**1. Macros**

The macros provide reusable functionality to improve code readability and reduce redundancy:

* **printString Macro**: Prints a string to the console using system calls.
* **printArray Macro**: Iterates through an array and prints its elements, showcasing the intermediate steps of sorting.

These macros encapsulate commonly used operations, making the main execution logic concise.

**2. Data Segment**

Defined under the .data section, this segment contains:

* **Array**: The input array to be sorted.
* **Messages**: Strings for prompts, like instructions and output results.
* **Temporary Variables**: For storing size and intermediary data.

Example:

.data

array: .word 5, 3, 8, 6, 2

size: .word 5

msg\_prompt: .asciiz "Array before sorting:"

**3. Text Segment**

The .text section begins the program logic. It includes:

* Loading array and size.
* Printing the array before and after sorting.
* Invoking sorting and swapping procedures.

Example:

main:

la $a0, msg\_prompt

li $v0, 4

syscall

la $a0, array

lw $a1, size

jal printArray

**4. Procedures**

**Bubble Sort Procedure**

Implements the core logic:

1. Iterates through the array multiple times.
2. Compares adjacent elements.
3. Swaps elements if they are in the wrong order using the swap procedure.

Example:

sort:

lw $t0, size # Load size of array

sub $t0, $t0, 1

outer\_loop:

...

jal swap

...

b outer\_loop

**Swap Procedure**

Handles swapping of two elements:

1. Takes addresses of the two elements.
2. Performs the exchange using temporary registers.

Example:

swap:

lw $t1, 0($a0)

lw $t2, 0($a1)

sw $t1, 0($a1)

sw $t2, 0($a0)

jr $ra

**Execution Flow**

1. The program initializes the environment and prints the unsorted array.
2. The sort procedure is called to sort the array using bubble sort logic.
3. The sorted array is printed.
4. The program terminates gracefully.

**Strengths and Weaknesses**

**Strengths**

* **Modular Design**: Clear separation of concerns (main logic, sorting, and swapping).
* **Reusable Macros**: Improves readability and reduces redundancy.
* **Dynamic Output**: Prints the array before and after sorting for verification.

**Weaknesses**

* **Inefficiency**: Bubble sort has complexity, making it less suitable for large arrays.
* **Verbose Code**: Assembly inherently involves more instructions compared to high-level languages.
* **Lack of Error Handling**: No checks for invalid inputs or edge cases like empty arrays.

**Conclusion**

This MIPS assembly code demonstrates a well-structured implementation of the bubble sort algorithm. While the use of macros and procedures enhances clarity, the inherent inefficiency of the sorting algorithm and verbosity of assembly language are limitations.