CS 226

Computer Organization and Design

Fall 2024

Assignment #6

Assembly Language

Operations on Arrays, Conditional & Unconditional Branching,

External Calls, Subroutines, Stack

Due date: Wednesday, October 30, 2024

This assignment is similar to the exercise we worked on in class on Wednesday, October 9, and includes the programming concepts mentioned in the above assignment title. This assignment will have you create a RISC-V assembly language program to process an array to produce the sum of all values, a sum of the positive values and a sum of the negative values, storing the results to memory and outputting the results to the Run I/O console.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I have included a Java program for reference that does what the assembly language program should do. Also included is an outline of the general structure of the program on the last page.

Follow a similar approach to this assignment that was implemented in the class exercise. Of importance is the planning of your register usage and making sure you include the register usage comment section in your program.

Define the values in the data segment of your program in the same order as in the Java program. The string values used for the output should also be defined in the data segment and should include the tab escape sequence or control character as demonstrated in the example programs in lecture. There should be eight items defined in the data segment.

Note how the loop is structured in the Java program and define the appropriate values in your assembly language program to accommodate the index, memory offset and memory address calculations of the array elements. All numbers in the array are defined as integers representing 32-bit two’s complement word values.

There will be one subroutine that will print one line of output each time it is called. The subroutine should be structured such that the values required for output, specifically the string addresses and sum values, will be read from the stack. As in the Java program, the output subroutine will be called three times. Therefore, in the main program, prior to each call, the two values should be pushed onto the stack in the appropriate order so that the subroutine will read them for producing the output. All external calls, except for the program exit, will be in the subroutine.

Remember to store the final calculated values to the appropriate memory locations that are to be allocated and initialized to zero in the data segment. Storing the values should be done in the main program.

**Note:** the **la (load address)** pseudoinstruction may be used to place the addresses for the strings into a register. This is the ***only*** pseudoinstruction allowed in the program.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

After successfully running the program, create a memory dump and delete all the empty lines of data keeping only the data from your program. Label or indicate in some way the location of the three result values computed in your program. Also, copy/paste the output from the Run I/O console window into the same text file as your memory dump. An example of how your memory dump should look can be found at the end of this assignment document.

Submit on Blackboard:

1. Your assembly language source program file (.asm)
2. The text file containing the memory dump with the copy/paste of the Run I/O output. (.txt)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Grading criteria:

Check your program to make sure you have included the following list of requirements:

Header comments 5 points

Register usage comment section 5 points

Required data definitions in data segment 10 points

Correct structure of the loop 5 points

Correct conditional/unconditional logic 10 points

Calculated values stored correctly in memory 5 points

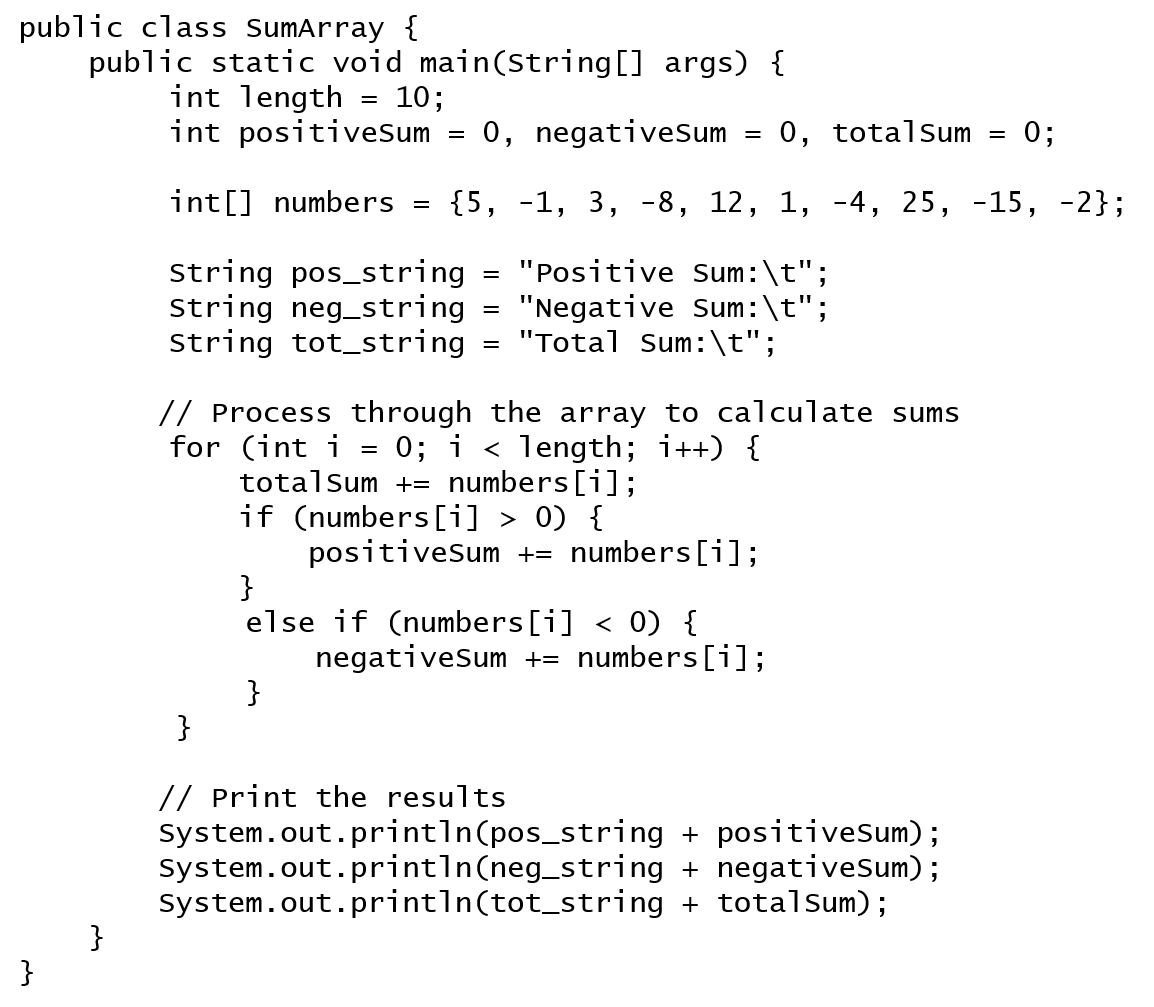
Subroutine producing correctly formatted output 10 points

Appropriate use of stack to pass values to subroutine 5 points

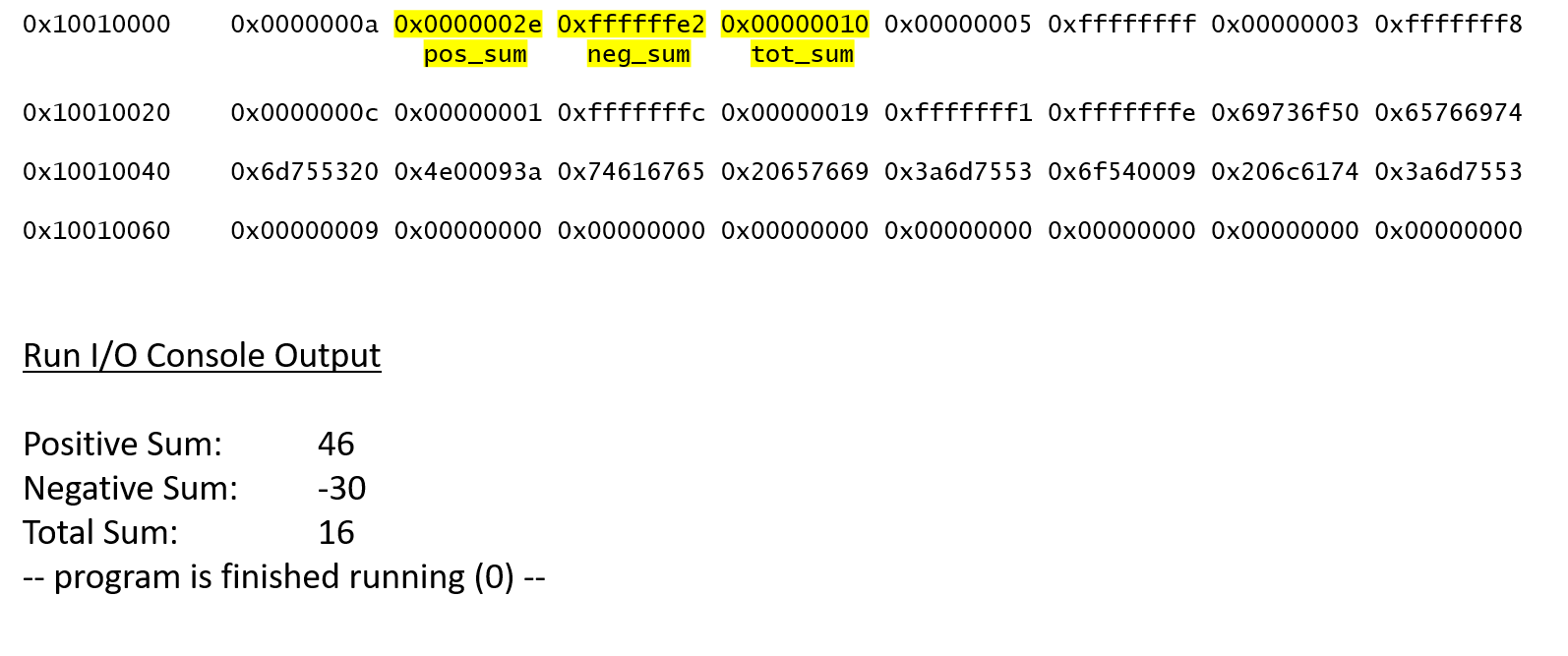
Labeled memory dump with copy of output 5 points

Total points for assignment 60 points

Reference Java program



Example of a Memory Dump with Console Output Added



General Program Structure

Data Segment

Define the 8 data items in the same order as the Java program

Text Segment

Initialize values:

Memory base address

Array base address

Array length (read from memory)

Index = 0

Loop:

Calculate array offset

Add offset to array base address

Read array element

Add to total

If element > 0, add to positive

Else if element < 0, add to negative

Else next array element

Increment Index

Repeat loop if not done

Store sums to memory

Push string address & integer values onto stack (first output line)

Call subroutine for output

Push string address & integer values onto stack (second output line)

Call subroutine for output

Push string address & integer values onto stack (third output line)

Call subroutine for output

Exit program

Subroutine

External call code to print string

Pop string address from stack

Print

External call code to print integer

Pop integer from stack

Print

External call code to print newline

ASCII newline code as argument

Print

Return from subroutine