Ve370 Introduction to Computer Organization

Project 1

Project Description

Develop a RISC-V assembly program that operates on a data segment consisting of an array of 32-bit signed integers. In the text(program) segment of memory, write a procedure called "main" that implements the "main()" function and as well as procedures for other subroutines described below. Assemble, simulate, and carefully comment the file.

```
main() {
    int arraySize = ...; //determine the size of the array here
    int positive, negative, small, average;
    int Array[arraySize] = {10, -5, 0, ...}; //compose the array here
    // Note: The tested Array will be given to you,
    //       the size of the array will be smaller than 100,
    //       and the absolute value of the array will be smaller than 64.
    positive = countArray(Array, arraySize, 1);
    negative = countArray(Array, arraySize, -1);
    small = countArray(Array, arraySize, 0);
    average = calculateAverage(Array, arraySize, arrayPosition);
    // Note: The following is just a demonstration, you do not need to implement
the        // following three lines as they are written
    long long product;
    product = calculateProduct(Array, arraySize);
    std::cout << product << endl;

}

int countArray(int A[], int arraySize, int cntType) {
    /*
     * Desciption: Count sepcific elements in the integer array A[]
     * Parameters:  array A[], size of array A[] arraySize, operation mode
cntType
     * Return values: When cntType = 1, return number of elements greater than
0;
     *                When cntType = -1, return number of elements less than 0;
     *                When cntType = 0, return number of elements greater than
or equal
     *                to -10, and less than or equal to 10;
     */
    int i, cnt = 0;
    for (i = arraySize - 1; i >= 0; i--) {
        switch (cntType) {
            case '1': cnt += isPositive(A[i]);
                break;
            case '-1': cnt += isNegative(A[i]);
```

```
                break;
            default: cnt += isSmall(A[i]);
                break;
        }
    }
    return cnt;
}

int isPositive(int x) {
    if (x > 0) return 1;
    else return 0;
}

int isNegative(int x) {
    if (x < 0) return 1;
    else return 0;
}

int isSmall(int x) {
    if (x >= -10 && x <= 10) return 1;
    else return 0;
}

int calculateAverage(int A[], int arraySize, int position) {
    /*
     * Desciption: Using recursion to calculate the average of the elements in
array
     * Parameters: Mostly same as countArray, position indicates a pointer to an
     *             element in the array
     * Return value: The average
     * Note: To make your life easier, you will not need to consider float
numbers as
     *         long as you implement the function correctly, and divide numbers
in the
     *         designed order
     */
    if (position == arraySize - 1) {
        return *(A + position);
    }
    else {
        int sum = (calculateAverage(A, arraySize, position + 1)
                   * (arraySize - position - 1)) + *(A + position);
        return sum/(arraySize - position);
    }
}

long long calculateProduct(int A[], int arraySize) {
    /*
     * Description: calculate the product of all the elements
     * Note: IMPORTANT! The following codes only serves as an example, you will
need to
     *       consider the boundary conditions of the array, and modify the
function
     *       by yourself in assembly to calculate the product of the elements in
A.
     *       ONLY IMPLEMENTING THE FOLLOWING WILL NOT PASS SOME OF THE TESTS!
     *       Also, to make your life easier, the length of the Array will be
restricted
```

```
     *       to less than or equal 20 elements
     */
    long long int product = 1;
    for (int i = 0; i < arraySize, i++) {
        product = product * A[i];
    }
    return product;
}
```

Notes

In this project, you will be asked to implement 7 functions about array calculation, namely, isPositive, isNegative, isSmall, countArray, calculateAverage, calculateProduct, and main.

For the functions: isPositive, isNegative, isSmall, countArray, calculateAverage, you NEED to STRICTLY implement the functions as they are written. Passing different parameters, looping the function in a different way will lead to DEDUCTIONS in the Coding Style grading segment.

For the function calculateProduct, you are suggested to implement and write the function yourself, with the following ability:

Calculate the product of all elements in the array, and print out (or store in the Memory) the results in BINARY form.

(You will be given a 10% bonus to the total score of Project 1 if you are able to print out (or store in the Memory) the Decimal form of the result, with the correct sign)

Note that the code above is only a demonstration of the ability of the function and will definitely not pass some of the tests.

For the function main: You need to strictly implement the codes before "long long product", and can modify the last three lines according to your will.

Restrictions

In this project, you will only be allowed to use RV64I BASE INTEGER INSTRUCTIONS. Any Extention Instruction Sets or PSEUDO INSTRUCTIONS are not allowed.

Specifically speaking, the instructions you are able to use includes:

add, addw, addi, addiw, and, andi, auipc, beq, bge, bgeu, blt, bltu, bne, csrrc, csrrci, csrrs, csrrsi, csrrw, csrrwi, ebreak, ecall, fence, fence.i, jal, jalr, lb, lbu, ld, lh, lhu, lui, lw, lwu, or, ori, sb, sd, sh, sll, sllw, slli, slliw, slt, slti, sltiu, sltu, sra, sraw, srai, sraiw, srl, srlw, orli, orliw, sub, subw, sw, xor, xori.

Deliverables

The deliverables for this product are divided into three parts: A personal demonstration, a written report, and a milestone.

For the milestone, you will be asked to submit a program consisting of implementations of main, countArray, isPositive, isNegative, and isSmall functions. The program will be submitted via Canvas.

In the personal demonstration, offline students will be required to come personally, with laptop, and demonstrate the result of their program with the given test cases. Two oral questions will be asked, about the implementation and coding style of the program. If the student failed both questions, three more questions will be asked. Failing 4 out of 5 questions will lead to Honor Code Violation. Examples of the questions include "How did you pass the average back to the main() function?"

For the written report, NO FORMAT is required. You will need to submit a file including the your source file, which is clearly commented for readability through Canvas. Directly coping the code into RIPES and running the test case should lead to the same results as your personal demonstration. Difference between the results might also lead to Honor Code Violation, according to the situation.

Grading Policy

Grading Factors	Weight
Milestone	5%
Written Report & Readability	5%
Oral Presentation	5%
isPositive and isNegative function	10%
isSmall function	10%
main function	10%
countArray function	15%
calculateAverage function	15%
calculateProduct function	15%
Coding Style	10%
Bonus (mentioned in “Notes”)	10%

Due Date

This project is due by 5th Jun.

Milestone submission is due by **29th May. 11:59 pm**.

Written Report submission is due by **5th Jun. 11:59 pm**. Late submission will result in the lost of all points in written report grading. The project will only be graded after the report submission.

The time and location for Personal Demonstration will be posted on Canvas later, offline students must come personally, and online students can attend on Feishu. If you can not attend, please contact me or the professor via email ASAP. Detailed explanation will be posted on Canvas later.