**Ve370 Introduction to Computer Organization**

**Project 1 Report**

**Ye Weijie 515370910174**

**OBJECTIVE**

Develop a MIPS assembly program that operates on a data segment consisting of an array of 32-bit unsigned integers. In the text (program) segment of memory, write a procedure called main that implements the main() function and other subroutines described below. Assemble, simulate, and carefully comment the file. Screen print your simulation results and explain the results by annotating the screen prints. You should compose an array whose size is determined by you in the main function and is not less than 20 elements.

main() {

int size = ...; //determine the size of the array here

int PassCnt, FailCnt;

int testArray[size] = { 55, 83,

... //compose your own array here

};

PassCnt = countArray(testArray, size, 1);

FailCnt = countArray(testArray, size, -1);

}

int countArray(int A[], int numElements, int cntType) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Count specific elements in the integer array A[] whose size is \*

\* numElements and return the following: \*

\* \*

\* When cntType = 1, count the elements greater than or equal to 60; \*

\* When cntType = -1, count the elements less than 60; \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int i, cnt = 0;

for(i=numElements-1, i>0, i--) {

switch (cntType) {

case '1' : cnt += Pass(A[i]); break;

otherwise: cnt += Fail(A[i]);

}

}

return cnt;

}

int Pass(int x) {

if(x>=60) return 1;

else return 0;

}

int Fail(int x) {

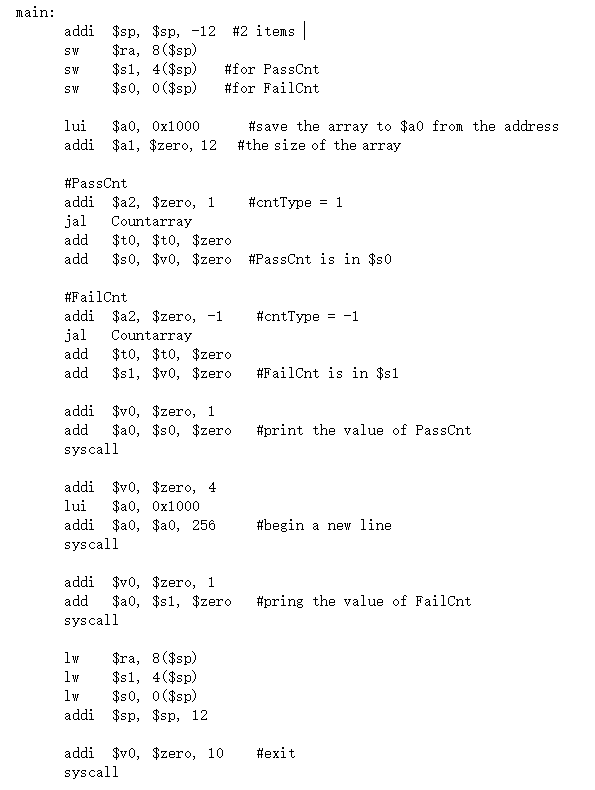
if (x<60) return 1;

else return 0;

}

**PROCEDURE**

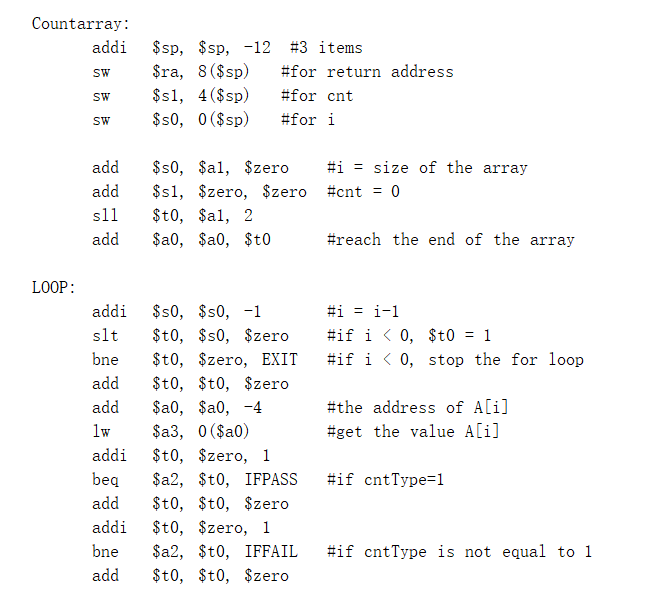
1. **Main Function**

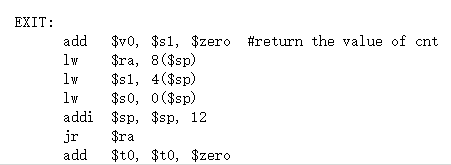


The main function can be separated into two parts. In the first part, we store things we need in the stack. $ra for the returning address, $s1 for PassCnt and $s0 for FailCnt. Then we store the address of array to $a0, and set the size ($a1) to 12. Then we set the cntType ($a2) to 1 and jump to countArray and get the value of PassCnt. Change cntType ($a2) to -1 we can get the value of FailCnt.

In the second part, we output the PassCnt and FailCnt in two lines, and give back the stack.

**2. CountArray Function**



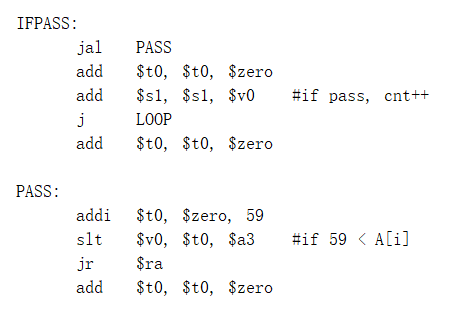


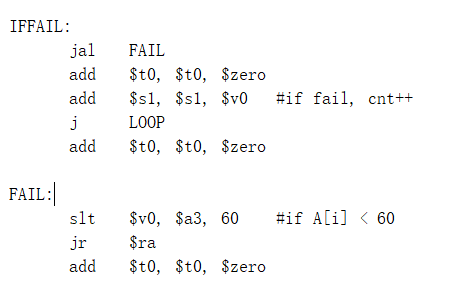
First of all, we store the things we need in the stack. $ra for the returning address, $s1 for cnt and $s0 for i. Then we set i($s0) to numElement(size of the array), cnt($s0) to 0, and the array($a0) to the end(+ $t0).

Then we come to the FOR loop. Before the loop, we will make i--, and judge whether it is smaller than 0. If i is smaller than 0, we will skip out this loop and to the EXIT part. If i is larger than 0, we will get the value of A[i] to $a3. Then we judge the cntType ($a2). If cntType = 1, we will jump to the IFPASS function. Else, we will jump to the IFFAIL function.

In the EXIT part, we will give back the stacks and jump back to the main function.

**3. Pass & Fail Function**



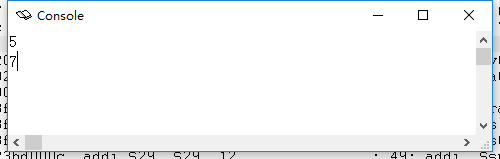


In the Pass function, I separate it into two parts. The PASS function is to judge whether A[i] ($a3) is larger than or equal to 60. If so, $v0 will be 1 and add to cnt ($s1) in IFPASS part. Then it will jump back to the FOR loop again.

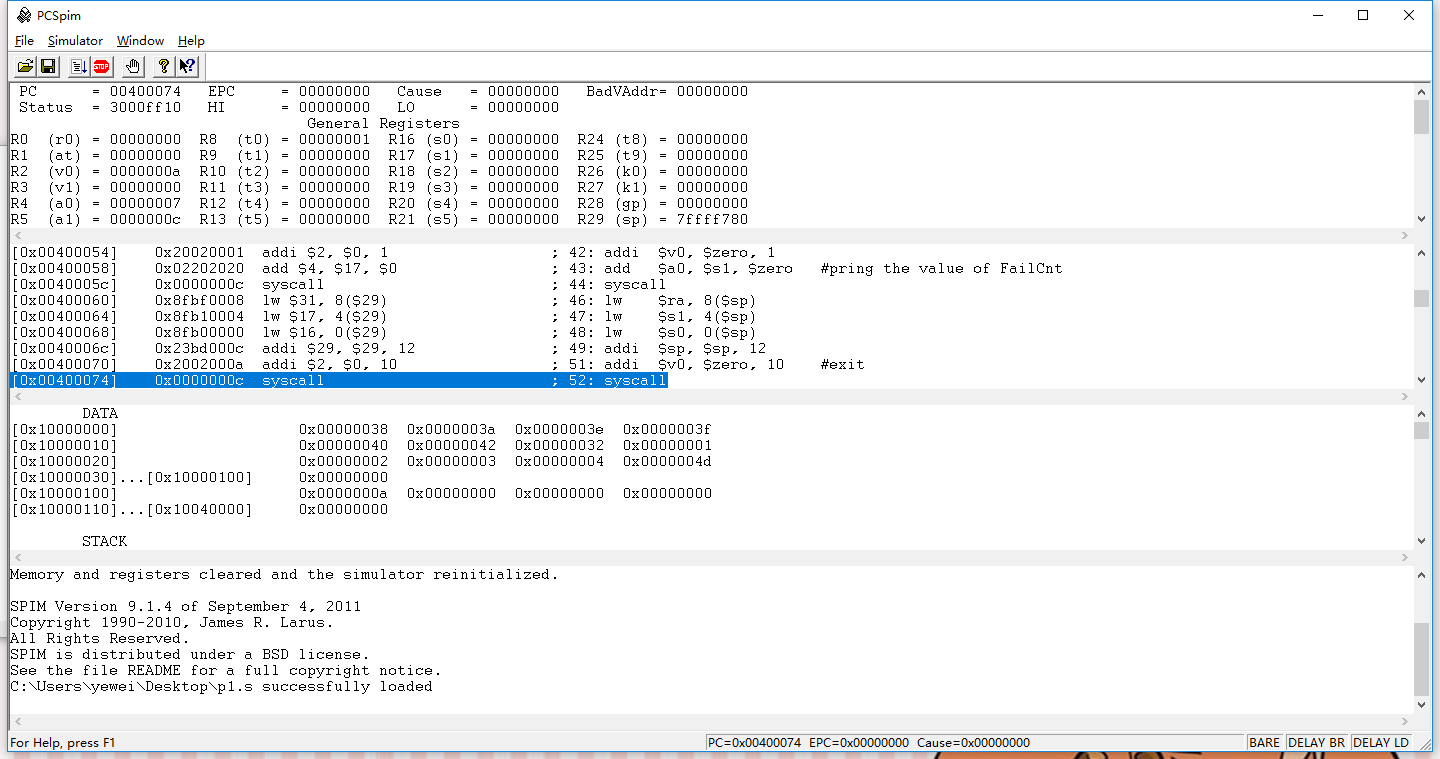
The Fail function works in the same way, it has has two parts: FAIL and IFFAIL. If A[i] ($a3) is smaller than 60, cnt ($s1) will add 1.

**RESULT:**

Console:



Register:



The array is: 56,58,62,63,64,66,50,1,2,3,4,77

So the PassCnt is 5, and the FailCnt is 7.

**CONCLUSION:**

In this project, we first use the Spim software and use assembly language to write a programme.

1.It is very important to put things into stacks at the beginning of the function.

2.A right use of different kind of registers is also very necessary. $a0 is for parameter of the function, $v0 is for returning value of the function, $t0 is for temporary number.

3.The most difficult part is to call the subfunction and jump back to the original function. We should use jr, j, jal correctly to avoid the problems.

4.The output part has some convention. We will put the instruction to $v0: 1 is for print int, 10 is for exit. Then we will put the output value to $a0, and syscall is for output the number to the console.

5.The data part is very important. It can save the array and some other instructions.

It is a relatively simple project, but the first time to use Spim. We can know the internal operation when we operate programs in other languages, like C++. We also know more about the operation of the registers in the computer.

**NOTICE:**

The notice I want to mention are mainly in two aspects.

1. Besides, it is very important to know how to write the data part. It is as important as the function part for a program.
2. The comments are very useful both for the reader and programmer. With the comments, we can know the function of each instruction and the register.