CSE331/Computer Organization-HW2 Report

The problem in the homework is known as “Subset Sum Problem”. Subset Sum Problem is one of the most famous algorithms. The best way to implement it in a code is by using a recursioın function.

C++

- In the main function, array is filled by the numbers which are taken from user, by using the loop below

```
for(int i = 0; i < arraySize; ++i){
    cin >> arr[i];
}
```

- And it is sent to the checking function with the needed arguaments to make the function return a result value.

```
returnVal = CheckSumPossibility(num,arr,arraySize);
```

- According to returned value, the result is printed to the screen.

```
if(returnVal == 1){
    cout << "Possible!" << endl;
}
else
    cout << "Not possible!" << endl;
```

- Here is the definition of the checking function:

```
int CheckSumPossibility(int num, int arr[], int arraySize)
```

- Base cases are declared in the code below.

```
if (num == 0){
    return 1;
}
if (arraySize == 0){
    return 0;
}
```

- If not, function is called and returned as recursively as below

```
return CheckSumPossibility(num, arr, arraySize - 1)
        || CheckSumPossibility(num - arr[arraySize - 1], arr, arraySize - 1);
```

MIPS/.asm

- In the main function, size and sum values are taken from the user as shown in below.

```
main:
        la $t0,arr      #stores the address of arr in $t0

        li $v0,4
        la $a0,str1
        syscall         #prints str1

        li $v0, 5
        syscall         #gets input from the user and stores it to $t1
        move $t1, $v0

        li $v0,4
        la $a0,str2
        syscall         #prints str2

        li $v0, 5
        syscall         #gets input from the user and stores it to $t2
        move $t2, $v0
```

- Then, the integer inputs are taken from the user to fill the array

```
li $t3,0        #sets iterator to zero
input:
        beq $t1,$t3,continue
        li $v0, 5
        syscall         #gets integer from the user
        sw $v0, 0($t0)  #stores it in the address of arr.
        addi $t0,$t0,4  #increases the addres of word for storing other integers.
        addi $t3,$t3,1  #increases the iterator
        j input
continue:
```

- To obey the contract arguaments of the recursive function are assigned to $a registers. Then checkSum procedure is called. If the returned value is 1 it means subset sum is possible. Otherwise it is not possible.

```
continue:
        move $s1,$t1            #$s1 is the temp size for checkSum procedure

        move $a0,$t2            #num is assigned to a0
        la $a1,($t0)            #arrays last address is assigned to a1
        move $a2,$t1            #size is assigned to a2

        jal checkSum            #jumps checkSum function and stores the next command in the program counter.
        bne $zero,$v0,printTrue #if $s0(returned value) is 1,then it is possible.
        beq $zero,$v0,printFalse#if $s0(returned value) is not 1,then it is 0. So sum is not possible.
```

- Here is the recursion procedure. I thşnk it explains itself with commands.

```
checkSum:
        beq $a0,$zero,returnTrue        # $a0=sum value
        beq $a2,$zero,returnFalse       # $a2=size value
        addi $sp,$sp,-20                # creates a space in the stack
        sw $ra, 0($sp)                  # keeps the return adress in the 0($sp)
        sw $a0, 4($sp)                  # keeps the sum number in the 4($sp)
        sw $a1, 8($sp)                  # keeps the arr address in the 8($sp)
        sw $a2, 12($sp)                 # keeps the size in the 12($sp)
        addi $a2,$a2,-1                 # decrease the size
        jal checkSum                    # jump (and link) to checkSum procedure
        sw $v0, 16($sp)                 # keeps the return value of first function in the 16($sp)
        lw $a0, 4($sp)                  # loads the first value
        lw $a1, 8($sp)                  # loads the first value
        lw $a2, 12($sp)                 # loads the first value
        addi $a2,$a2,-1                 # decrease the size
        lw $s7,-4($a1)                  # loads the last element of array to $s7
        sub $a0,$a0,$s7                 # number-lastelement
        lw $a1, -4($a1)                 # changes the last element of array to 1 before
        jal checkSum                    # calls itself again
        lw $s6, 16($sp)                 # loads the first v0 value to $s6 for or command
        or $v0,$v0,$s6                  # "or"s them and assign the result value to $v0 (according to contracc
        lw $ra,0($sp)                   # original ra value is called

        addi $sp,$sp,20                 # stack is as same as before.
        jr $ra                          #go back the line that PC keeps.
```

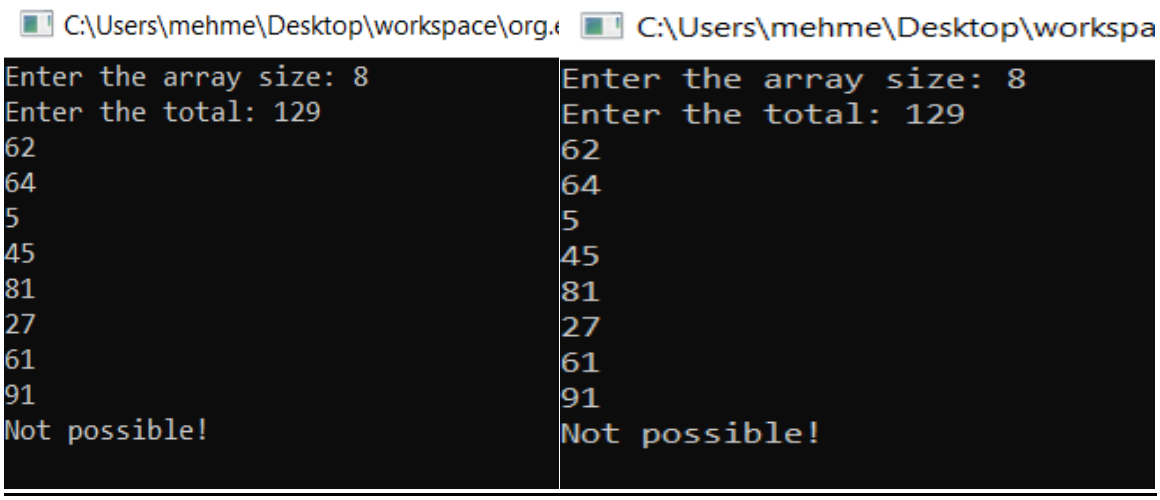
ERRORS:

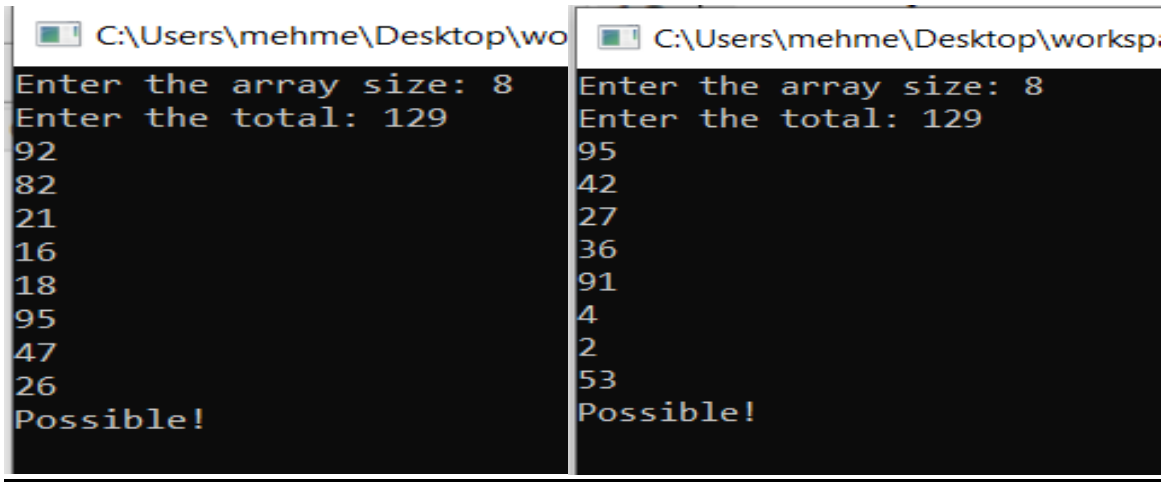
- There are some problems at the recursion procedure of asm code. I couldnt find how to fix it but i think i know where it is. The problem does not always ocur. When sum is not possible it gives error. When sum is possible sometimes it shows the correct answer.

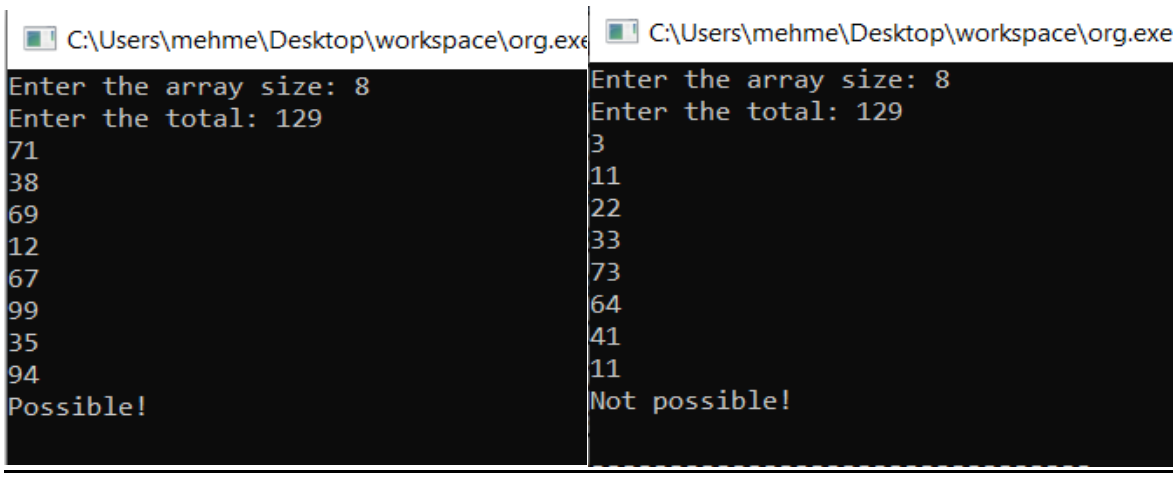
```
lw $s7,-4($a1)                  # loads the last element of array to $s7
sub $a0,$a0,$s7                 # number-lastelement
lw $a1, -4($a1)                 # changes the last element of array to 1 before
```

İn this part of code, at the end of the program i am out of range the address. I couldnt fix it.

Sample I/O (C++)


C:\Users\mehme\Desktop\workspace\org.e
Enter the array size: 8
Enter the total: 129
62
64
5
45
81
27
61
91
Not possible!
C:\Users\mehme\Desktop\workspa
Enter the array size: 8
Enter the total: 129
62
64
5
45
81
27
61
91
Not possible!


C:\Users\mehme\Desktop\wo
Enter the array size: 8
Enter the total: 129
92
82
21
16
18
95
47
26
Possible!
C:\Users\mehme\Desktop\workspa
Enter the array size: 8
Enter the total: 129
95
42
27
36
91
4
2
53
Possible!


C:\Users\mehme\Desktop\workspace\org.exe
Enter the array size: 8
Enter the total: 129
71
38
69
12
67
99
35
94
Possible!
C:\Users\mehme\Desktop\workspace\org.exe
Enter the array size: 8
Enter the total: 129
3
11
22
33
73
64
41
11
Not possible!

Sample I/O (MIPS)

```
Enter the Array Size: 5
Enter the sum: 5
1
2
3
4
5
Possible
-- program is finished running --
```

```
Enter the Array Size: Enter the Array Size: 8
Enter the sum: 8
1
2
3
4
5
6
7
8
Possible
```

Error Message:

- For many other I/O's i get this message but i dont know that according to what it gives me this message. Thats why i didnt use same inputs and outputs for sample screenshoots.

```
Assemble: assembling C:\Users\mehme\Desktop\workspace\mips1.asm

Assemble: operation completed successfully.

Go: running mips1.asm

Error in C:\Users\mehme\Desktop\workspace\mips1.asm line 70: Runtime exception at 0x004000b8: fetch address not aligned on word boundary 0x0000005b

Go: execution terminated with errors.
```

Except the recursive procedure everything works fine.

Thank You...

☹

Mehmet Avni ÇELİK

1801042630