实验一：MIPS汇编程序设计

专业班级：**提高2201班**

姓名：　　**王翎羽**

学号：　　**U202213806**

实验名称

MIPS汇编程序设计

实验目的：

1. 熟悉常见的MIPS汇编指令
2. 掌握MIPS汇编程序设计
3. 了解MIPS汇编语言与机器语言之间的对应关系
4. 了解C语言语句与汇编指令之间的关系
5. 掌握MARS的调试技术
6. 掌握程序的内存映像

实验环境

Mars MIPS汇编编译器、Windows 11操作系统

实验任务

- 编写子程序PENO(&X,N,SP,SN)求长度为N 的字类型数组X中所有正奇数的和和所有负偶数的和，并分别保存到SP和SN中。已知a0保存X的地址，a1保存数组长度N，正奇数的和保存在v0,负偶数的和保存在v1中。并编写主程序验证子程序功能,要求将计算结果输出到console。
- 测试以下数组序列

  int X[10]={1,-4,8,-9,5,6,-10,19,22,23};

  int X[10]={121,-124,138,-199,255,2566,-1034,1019,2032,2033};

实验思路

1. 在主程序中，首先打印提示信息，然后调用FUNC函数分别计算两个数组中的正奇数和负偶数的和，并打印结果。
2. FUNC函数中，使用循环遍历数组中的每个元素，判断元素的正负和奇偶性，并将符合条件的元素加到相应的和中。
3. 循环直到遍历完数组所有元素，最后返回主程序继续执行。

细节思考

- 由于寄存器v0需要用来控制syscall指令的操作，所以我改用寄存器s0和s1来存储计算后的结果。
- 在判断元素的奇偶性时，采用最后一位和1进行与运算，运算结果为1，说明为偶数，反之为奇数。

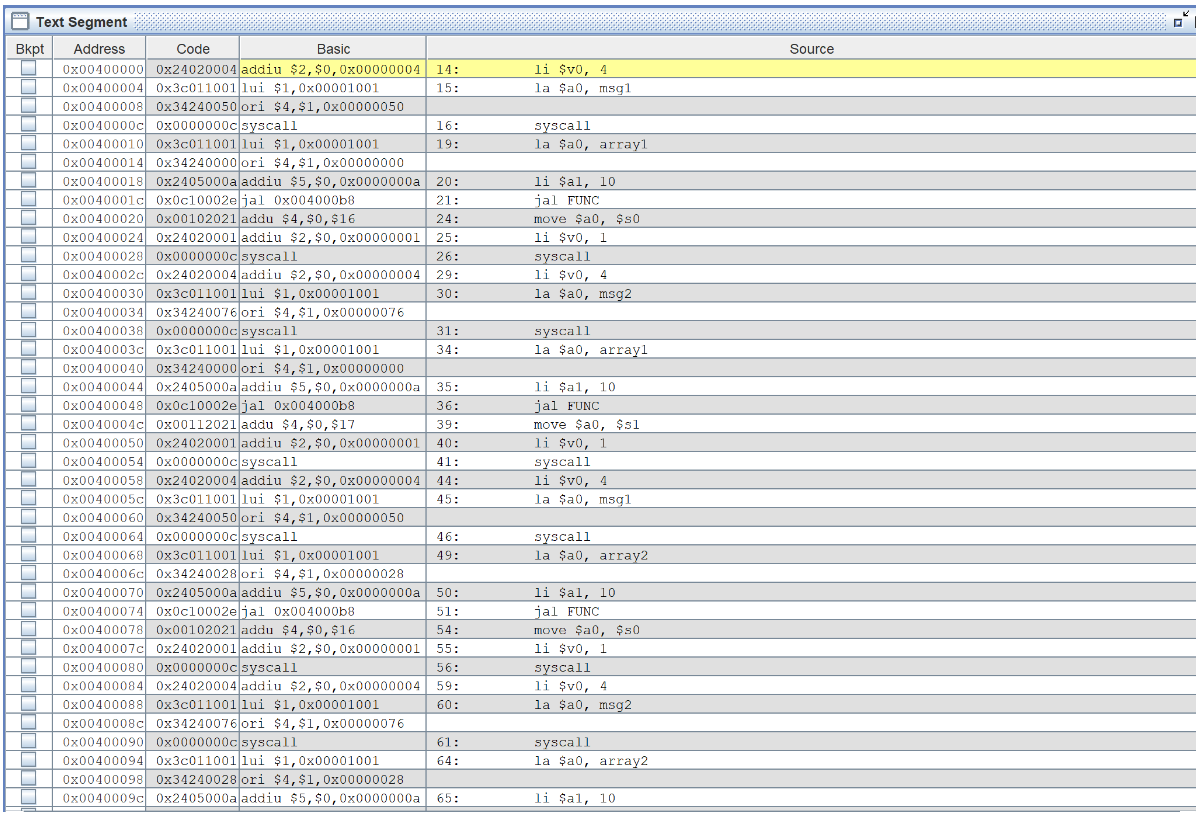
实验源代码及注释

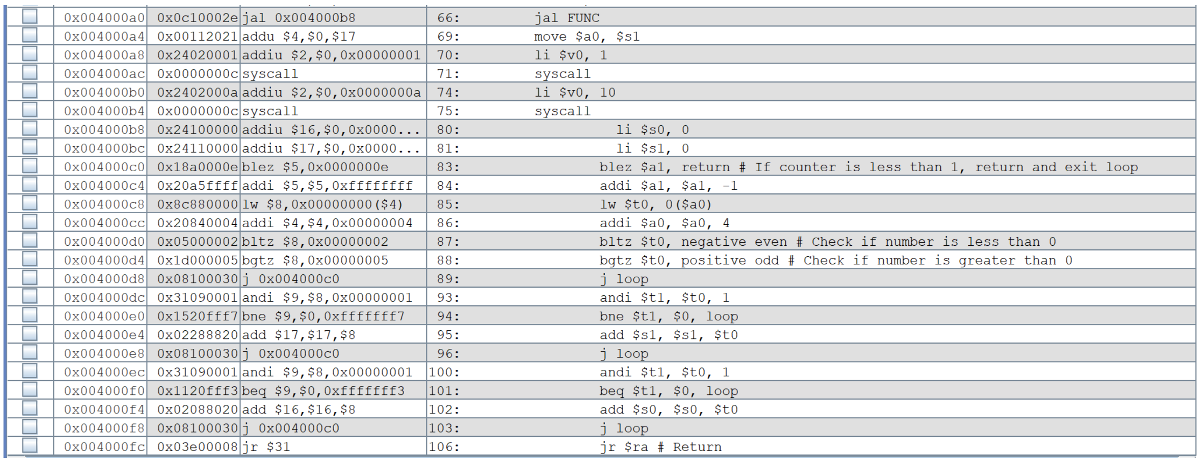
```
.data
    array1:.word 1,-4,8,-9,5,6,-10,19,22,23
    array2:.word 121,-124,138,-199,255,2566,-1034,1019,2032,2033

    msg1: .asciiz "\n Sum of these positive odd values = "
    msg2: .asciiz "\n Sum of these negative even values = "

    .globl main

.text

main:
    # Print message for sum of positive odd values in array1
    li $v0, 4
    la $a0, msg1
    syscall

    # Initialize address entry parameters for array1
    la $a0, array1
    li $a1, 10
    jal FUNC

    # Print sum of positive odd values in array1
    move $a0, $s0
    li $v0, 1
    syscall

    # Print message for sum of negative even values in array1
    li $v0, 4
    la $a0, msg2
    syscall

    # Print sum of negative even values in array1
    move $a0, $s1
    li $v0, 1
    syscall

    # Print message for sum of positive odd values in array2
    li $v0, 4
    la $a0, msg1
    syscall

    # Initialize address entry parameters for array2
    la $a0, array2
    li $a1, 10
    jal FUNC
```

```
# Print sum of positive odd values in array2
move $a0, $s0
li $v0, 1
syscall

# Print message for sum of negative even values in array2
li $v0, 4
la $a0, msg2
syscall

# Print sum of negative even values in array2
move $a0, $s1
li $v0, 1
syscall

# Exit the program
li $v0, 10
syscall

# Function to calculate sum of positive odd and negative even numbers
FUNC:
    # Initialize $s0 and $s1 to store the sums
    li $s0, 0
    li $s1, 0
loop:
    blez $a1, return    # If counter is less than 1, return and exit loop
    addi $a1, $a1, -1
    lw $t0, 0($a0)
    addi $a0, $a0, 4
    bltz $t0, negative_even     # Check if number is less than 0
    bgtz $t0, positive_odd  # Check if number is greater than 0
    j loop

negative_even:
    # Add negative even number to sum
    andi $t1, $t0, 1
    bne $t1, $0, loop
    add $s1, $s1, $t0
    j loop

positive_odd:
    # Add positive odd number to sum
    andi $t1, $t0, 1
    beq $t1, $0, loop
    add $s0, $s0, $t0
    j loop

return:
    jr $ra  # Return
```

实验结果

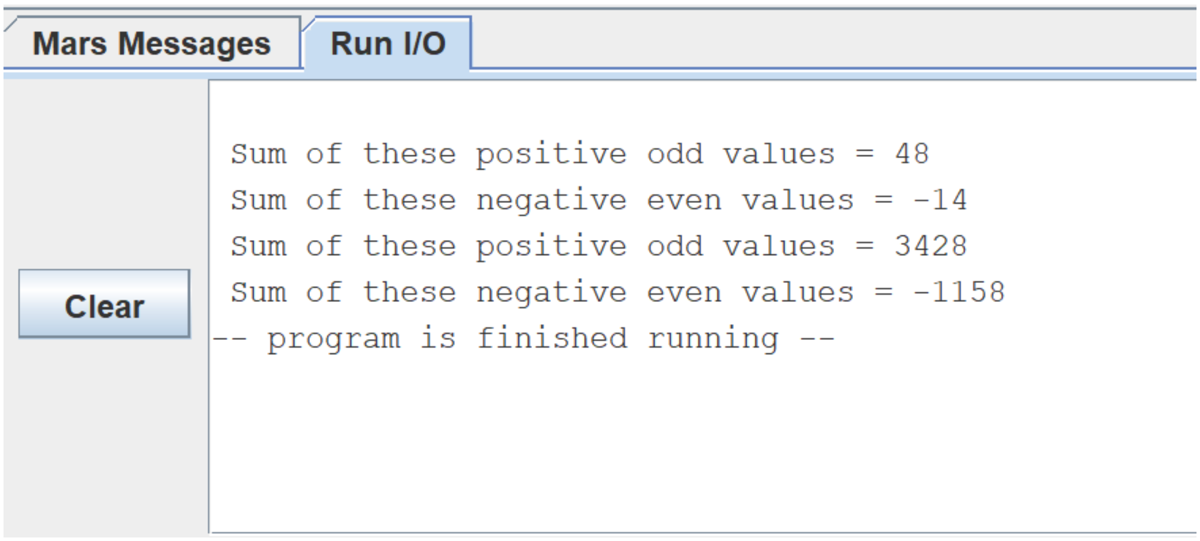
程序代码段映像





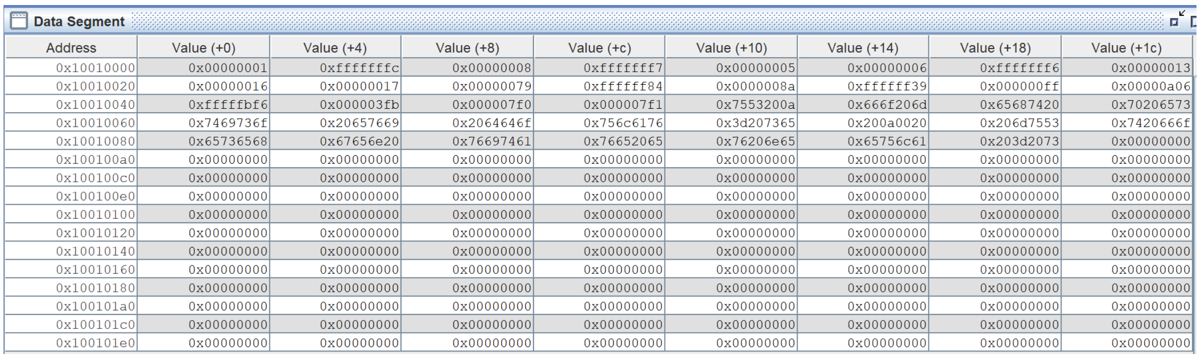
Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020004	addiu $2,$0,0x00000004	14: li $v0, 4
	0x00400004	0x3c011001	lui $1,0x00001001	15: la $a0, msg1
	0x00400008	0x34240050	ori $4,$1,0x00000050	
	0x0040000c	0x0000000c	syscall	16: syscall
	0x00400010	0x3c011001	lui $1,0x00001001	19: la $a0, array1
	0x00400014	0x34240000	ori $4,$1,0x00000000	
	0x00400018	0x2405000a	addiu $5,$0,0x0000000a	20: li $a1, 10
	0x0040001c	0x0c10002e	jal 0x004000b8	21: jal FUNC
	0x00400020	0x00102021	addu $4,$0,$16	24: move $a0, $s0
	0x00400024	0x24020001	addiu $2,$0,0x00000001	25: li $v0, 1
	0x00400028	0x0000000c	syscall	26: syscall
	0x0040002c	0x24020004	addiu $2,$0,0x00000004	29: li $v0, 4
	0x00400030	0x3c011001	lui $1,0x00001001	30: la $a0, msg2
	0x00400034	0x34240076	ori $4,$1,0x00000076	
	0x00400038	0x0000000c	syscall	31: syscall
	0x0040003c	0x3c011001	lui $1,0x00001001	34: la $a0, array1
	0x00400040	0x34240000	ori $4,$1,0x00000000	
	0x00400044	0x2405000a	addiu $5,$0,0x0000000a	35: li $a1, 10
	0x00400048	0x0c10002e	jal 0x004000b8	36: jal FUNC
	0x0040004c	0x00112021	addu $4,$0,$17	39: move $a0, $s1
	0x00400050	0x24020001	addiu $2,$0,0x00000001	40: li $v0, 1
	0x00400054	0x0000000c	syscall	41: syscall
	0x00400058	0x24020004	addiu $2,$0,0x00000004	44: li $v0, 4
	0x0040005c	0x3c011001	lui $1,0x00001001	45: la $a0, msg1
	0x00400060	0x34240050	ori $4,$1,0x00000050	
	0x00400064	0x0000000c	syscall	46: syscall
	0x00400068	0x3c011001	lui $1,0x00001001	49: la $a0, array2
	0x0040006c	0x34240028	ori $4,$1,0x00000028	
	0x00400070	0x2405000a	addiu $5,$0,0x0000000a	50: li $a1, 10
	0x00400074	0x0c10002e	jal 0x004000b8	51: jal FUNC
	0x00400078	0x00102021	addu $4,$0,$16	54: move $a0, $s0
	0x0040007c	0x24020001	addiu $2,$0,0x00000001	55: li $v0, 1
	0x00400080	0x0000000c	syscall	56: syscall
	0x00400084	0x24020004	addiu $2,$0,0x00000004	59: li $v0, 4
	0x00400088	0x3c011001	lui $1,0x00001001	60: la $a0, msg2
	0x0040008c	0x34240076	ori $4,$1,0x00000076	
	0x00400090	0x0000000c	syscall	61: syscall
	0x00400094	0x3c011001	lui $1,0x00001001	64: la $a0, array2
	0x00400098	0x34240028	ori $4,$1,0x00000028	
	0x0040009c	0x2405000a	addiu $5,$0,0x0000000a	65: li $a1, 10
	0x004000a0	0x0c10002e	jal 0x004000b8	66: jal FUNC
	0x004000a4	0x00112021	addu $4,$0,$17	69: move $a0, $s1
	0x004000a8	0x24020001	addiu $2,$0,0x00000001	70: li $v0, 1
	0x004000ac	0x0000000c	syscall	71: syscall
	0x004000b0	0x2402000a	addiu $2,$0,0x0000000a	74: li $v0, 10
	0x004000b4	0x0000000c	syscall	75: syscall
	0x004000b8	0x24100000	addiu $16,$0,0x0000...	80: li $s0, 0
	0x004000bc	0x24110000	addiu $17,$0,0x0000...	81: li $s1, 0
	0x004000c0	0x18a0000e	blez $5,0x0000000e	83: blez $a1, return # If counter is less than 1, return and exit loop
	0x004000c4	0x20a5ffff	addi $5,$5,0xffffffff	84: addi $a1, $a1, -1
	0x004000c8	0x8c880000	lw $8,0x00000000($4)	85: lw $t0, 0($a0)
	0x004000cc	0x20840004	addi $4,$4,0x00000004	86: addi $a0, $a0, 4
	0x004000d0	0x05000002	bltz $8,0x00000002	87: bltz $t0, negative even # Check if number is less than 0
	0x004000d4	0x1d000005	bgtz $8,0x00000005	88: bgtz $t0, positive odd # Check if number is greater than 0
	0x004000d8	0x08100030	j 0x004000c0	89: j loop
	0x004000dc	0x31090001	andi $9,$8,0x00000001	93: andi $t1, $t0, 1
	0x004000e0	0x1520fff7	bne $9,$0,0xfffffff7	94: bne $t1, $0, loop
	0x004000e4	0x02288820	add $17,$17,$8	95: add $s1, $s1, $t0
	0x004000e8	0x08100030	j 0x004000c0	96: j loop
	0x004000ec	0x31090001	andi $9,$8,0x00000001	100: andi $t1, $t0, 1
	0x004000f0	0x1120fff3	beq $9,$0,0xfffffff3	101: beq $t1, $0, loop
	0x004000f4	0x02088020	add $16,$16,$8	102: add $s0, $s0, $t0
	0x004000f8	0x08100030	j 0x004000c0	103: j loop
	0x004000fc	0x03e00008	jr $31	106: jr $ra # Return

输入输出端口测试



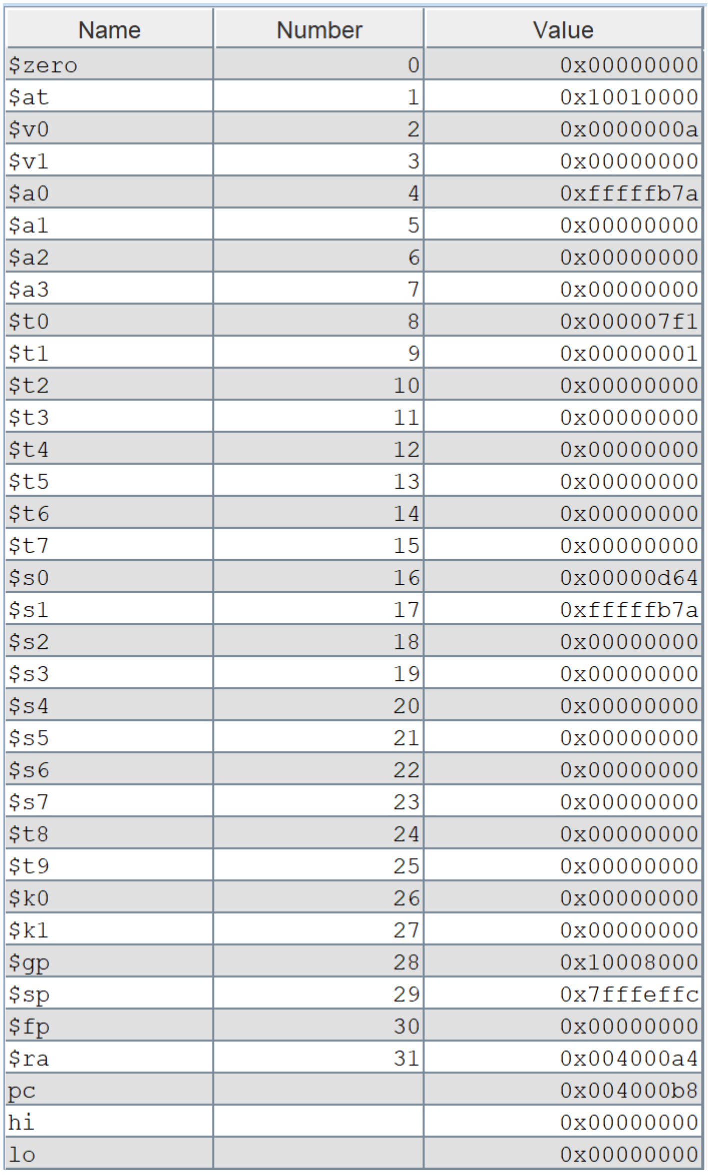
```
 Sum of these positive odd values = 48
 Sum of these negative even values = -14
 Sum of these positive odd values = 3428
 Sum of these negative even values = -1158
-- program is finished running --
```

程序数据段映像



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0xfffffffc	0x00000008	0xfffffff7	0x00000005	0x00000006	0xfffffff6	0x00000013
0x10010020	0x00000016	0x00000017	0x00000079	0xffffff84	0x0000008a	0xffffff39	0x000000ff	0x00000a06
0x10010040	0xfffffbf6	0x000003fb	0x000007f0	0x000007f1	0x7553200a	0x666f206d	0x65687420	0x70206573
0x10010060	0x7469736f	0x20657669	0x2064646f	0x756c6176	0x3d207365	0x200a0020	0x206d7553	0x7420666f
0x10010080	0x65736568	0x67656e20	0x76697461	0x76652065	0x76206e65	0x65756c61	0x203d2073	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000



Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0xfffffb7a
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x000007f1
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000d64
$s1	17	0xfffffb7a
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x004000a4
pc		0x004000b8
hi		0x00000000
lo		0x00000000

结果分析

从I/O端口输出得到的结果正确，满足实验要求。

实验小结

本次实验我使用了Mars软件进行汇编语言的练习，学会了使用syscall来进行数据的输出，最后实验结果正确，收获很大！写代码的过程中我也感受到了在C语言中一行代码就可以完成的工作，在汇编语言中可能就需要好几行才能完成。这让我更深刻理解了高级语言的代码可读性和开发效率方面上的巨大优势。