一. 实验任务

1. 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。
2. 用上述程序测试以下数组序列：

```
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};
```

二. 实验目的

1. 掌握 MIPS 汇编指令 ；
2. 熟悉 MIPS 汇编语言程序结构 ；
3. 掌握 C 语言语句 MIPS 汇编语言指令实现方案，了解 C 语言编译原理 ；
4. 掌握 MIPS 汇编语言程序数据段、指令段内存映像 ；
5. 熟练掌握使用 MIPS 汇编语言模拟器调试汇编语言程序 ；

三. 实验环境

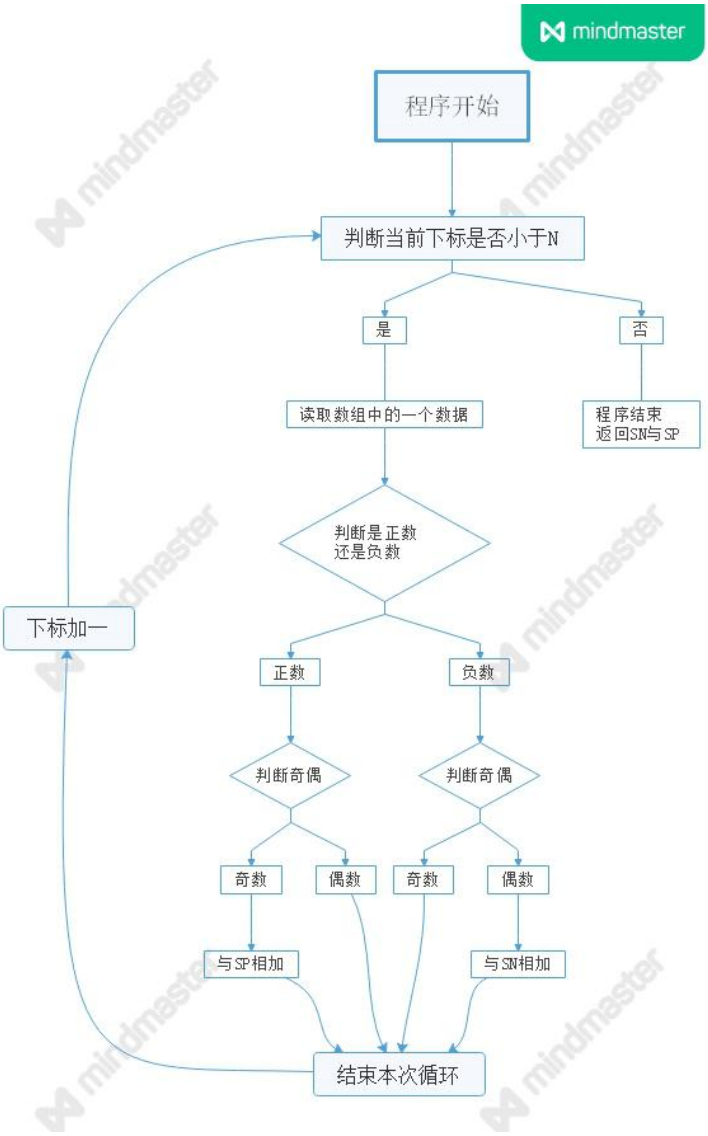
1. Windows 10 操作系统
2. 编辑工具：Mars 4_5
3. MIPS 模拟器：Mars 4_5

四. 汇编程序设计思路

1. 对应 C 语言代码如下：

```
#    PENO(int *X,int N,int sp,int sn){
#    int i;
#    int sp=0;
#    int sn=0;
#    for(i=0;i<n;i++){
#          If(a[i]%2==1&&a[i]>0)
#                sp=sp+a[i];
#    else if(a[i]%2==0&&a[i]<0)
#          sn=sn+a[i];
#    }
```

2. 程序流程图



五. 实现过程

1. 数据段

整型数组变量：array1 与 array2 用于程序功能的测试。

字符串：msg1 与 msg2 用于标注输出数据。

```
.data
        array1:.word 1,-4,8,-9,5,6,-10,19,22,23
        array2:.word 121,-124,-199,255,2566,-1034,1019,2032,2033
        msg1:.asciiz "\n The sum of positive odd numbers is SP = "
        msg2:.asciiz "\n The sum of negative even numbers is SN = "
        .globl main
```

2. 主程序

（1）用于处理和传递输入输出参数，以验证子程序功能。

（2）使用系统功能调用实现将计算结果输出到 console。

```
main:
        li $v0,4 #打印提示字符串1
        la $a0,msg1 #
        syscall #
        la $a0,array1 #初始化地址入口参数
        li $a1,10 # 初始化N
        jal sum
        move $a0,$v0 #处理出口参数正奇数和
        li $v0,1 #
        syscall #
        li $v0,4 #打印提示字符串2
        la $a0,msg2
        syscall #
        li $v0,1#处理出口参数负偶数和
        move $a0,$v1
        syscall
        li $v0,10#回系统
        syscall
    sum:li $v0,0
```

3.子程序

（1）sum：实现$v0 与$v1 的初始化；

（2）loop：实现数组下标的加一与正负数判断与正数的奇偶判断；

（3）negg：实现负数奇偶判断；

（4）addo/adde：实现对正奇数与负偶数的求和；

（5）retzz：返回主程序。

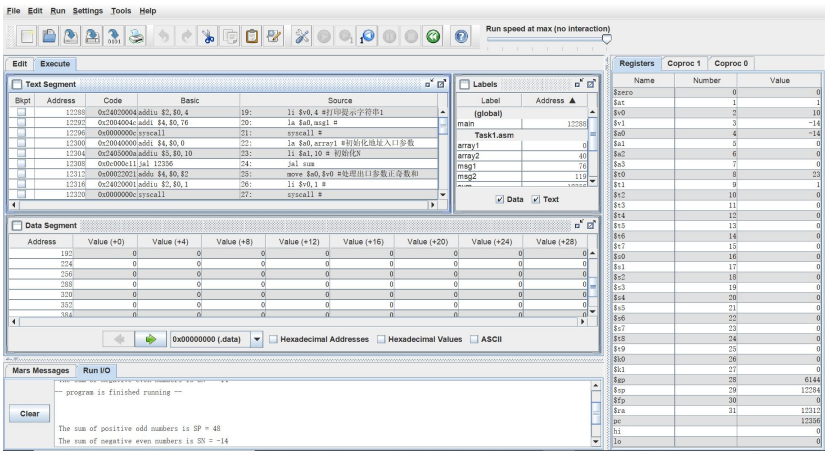
```
sum:li $v0,0
    li $v1,0#初始化v0和v1为0
    #li $s1,2
loop:
    blez $a1,retzz # (a1 <= 0)跳转到 retzz
    addi $a1,$a1, -1 # 减计数器
    lw $t0, 0($a0) #从数矩中获取元素
    addi $a0, $a0, 4 #修改地址指针指向下一个元素
    bltz $t0,negg # 如果负数
    andi $t1,$t0,1
    beq $t1,1,addo
    b loop
addo:
    add $v0, $v0, $t0 #正奇数的和
    b loop # loop
negg:
    andi $t1,$t0,1
    beq $t1,0,adde
    b loop
adde:
    add $v1,$v1,$t0 # 负偶数的和
    b loop #跳转loop
retzz: jr $ra # 返回
```

4. 完整代码与注释

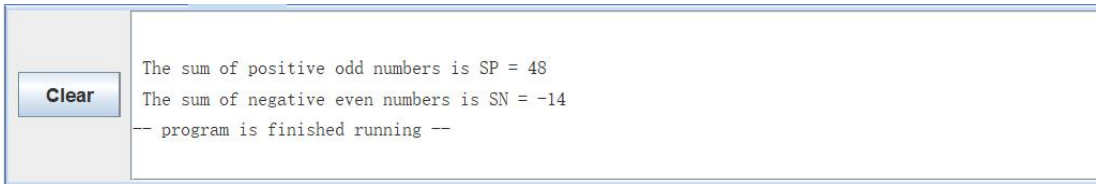
```
.data
array1:.word 1,-4,8,-9,5,6,-10,19,22,23
array2:.word 121,-124,-199,255,2566,-1034,1019,2032,2033
msg1:.asciiz "\n The sum of positive odd numbers is SP = "
msg2:.asciiz "\n The sum of negative even numbers is SN = "
.globl main
.text
main:
li $v0,4 #打印提示字符串 1
la $a0,msg1 #
syscall #
la $a0,array1 #初始化地址入口参数
li $a1,10 #  初始化 N
jal sum
move $a0,$v0 #处理出口参数正奇数和
li $v0,1 #
syscall #
li $v0,4 #打印提示字符串 2
la $a0,msg2
syscall #
li $v0,1#处理出口参数负偶数和
move $a0,$v1
syscall
li $v0,10#回系统
syscall
sum:li $v0,0
li $v1,0#初始化 vO 和 vl 为 0
#li $s1,2
loop:
blez $a1,retzz # (al <= 0)跳转到  retzz
addi $a1,$a1, -1 #  减计数器
lw $t0, 0($a0) #从数矩中获取元素
addi $a0, $a0, 4 #修改地址指针指向下一个元泰
bltz $t0,negg #  如果负数
andi $t1,$t0,1
beq $t1,1,addo
b loop
addo:
add $v0, $v0, $t0 #正奇数的和
b loop # loop
negg:
andi $t1,$t0,1
beq $t1,0,adde
b loop
adde:
add $v1,$v1,$t0 #  负偶数的和
b loop #跳转 loop
retzz: jr $ra #  返回
```

六. 实验结果

1. 运行结果



2. （1）当 X[10]={1, -4, 8, -9, 5, 6, -10, 19, 22, 23}时的输出结果：

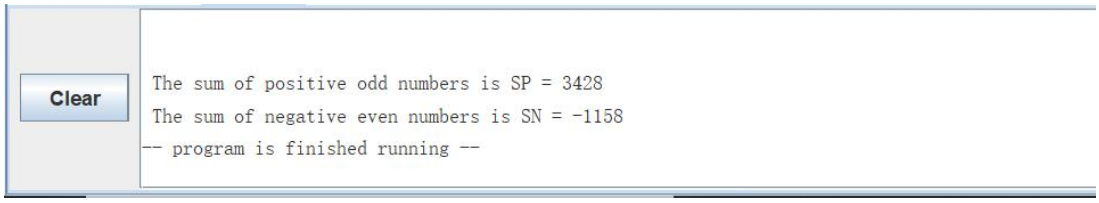


结果分析： 1+5+19+23 = 48

-4-10 = -14

可见，所得结果正确。

（2）X[10]={121, -124, 138, -199, 255, 2566, -1034, 1019, 2032, 2033}时的输出结果：



结果分析： 121+255+1019+2033 = 3428

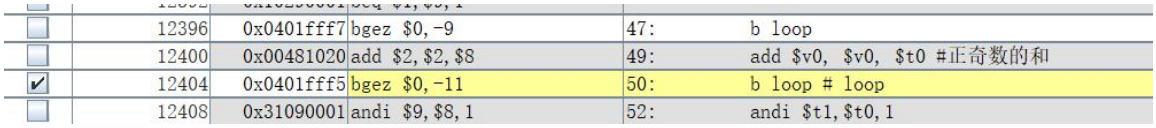
-124-1034 = -1158

可见，所得结果正确。

3. 断点调试

以 X[10]={121, -124, 138, -199, 255, 2566, -1034, 1019, 2032, 2033}的测试为例，设置以下断点：

（1）add $v0, $v0, $t0 #正奇数的和



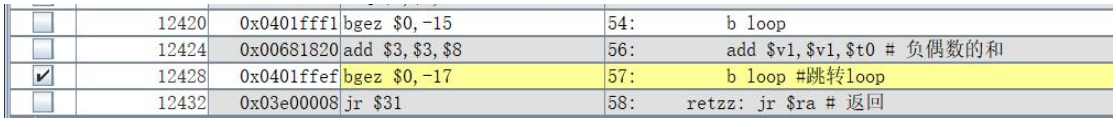
12396	0x0401fff7	bgez $0, -9	47:	b loop
12400	0x00481020	add $2, $2, $8	49:	add $v0, $v0, $t0 #正奇数的和
✔ 12404	0x0401fff5	bgez $0, -11	50:	b loop # loop
12408	0x31090001	andi $9, $8, 1	52:	andi $t1, $t0, 1

可知当程序第一次进入循环时停在此处，且此时有$v0 = 121，$v1= 0 结果如下：

$at	1	1
$v0	2	121
$v1	3	0

可知，程序运行结果正确。

（2）add $v1, $v1, $t0 # 负偶数的和



12420	0x0401fff1	bgez $0, -15	54:	b loop
12424	0x00681820	add $3, $3, $8	56:	add $v1, $v1, $t0 # 负偶数的和
✔ 12428	0x0401ffef	bgez $0, -17	57:	b loop #跳转loop
12432	0x03e00008	jr $31	58:	retzz: jr $ra # 返回

可知当程序第三次进入循环时停在此处，且此时有$v0 = 121，$v1 = -124, 结果如下：

$at	1	0
$v0	2	121
$v1	3	-124
$a0	4	48

可知，程序运行结果正确。

七.实验总结

本次实验是我第一次使用Mars进行汇编程序设计。在本次实验中，我不仅对学到的知识进行了巩固和检测,也让我首次深入体会到了汇编程序设计与C语言程序设计的众多不同之处，下面是我在本次实验中遇到并尝试解决的一些问题与积累与汲取的一些经验。

1. 对汇编设计中几乎每做一步操作（甚至是每一条指令）都要把的内容载入寄存器的设计思想与逻辑有了进一步的认识。这种特性也使得用C语言几行代码就能实现的程序在MIPS中变得相对复杂。
2. 一定注意寄存器编号不能混淆。MIPS架构中，不同的寄存器都有着自己特定的功能，错误的使用可能会带来意想不到的灾难。
3. 有技巧的使用跳转指令。不仅要注意跳转条件的正确用法，更要结合程序来选择合适的跳转程序以简化程序。
   如我在本次实验中使用如下指令实现了较为简洁的奇偶判断。

```
andi $t1,$t0,1
beq $t1,0,adde
```

4. 使用Mars时，在程序出错之后要学会用单步调试和设置断点的方法来找出错误，这样便能高效地找出错误所在，及时改正。
5. 学会充分利用学习资源。本次实验中我不仅学习了老师给的教学视频，也通过相关的电子书、网上博客和电子论坛学到了许多设计技巧。

总之，这次实验我完成了相关实验任务，实现了相关功能并对其进行了检测。本次汇编程序设计巩固了我对一些常用汇编指令的了解和认识;让我对汇编语句实现逻辑功能与编程思想有了一定的了解；同时，也进一步加深了我对计算机工作原理的理解，掌握了 MIPS程序设计的断点调试技能；学会了 Mars这一 MIPS汇编语言模拟器的使用。本次实验，是一次收获满满的历程。