

# 此文档选取一些有代表性的测试程序进行详细说明

## 1.斐波那契数列

- 源程序

```
const int x = 10;
int factorial(int x)
{
    if(x<=0) return (0);
    if(x==1) return (1);
    return (x*factorial(x - 1));
}

void main()
{
    int result;
    result = factorial(x);
    printf(result);
}
```

- 中间产物

```
E:\编译原理\程序记录\4.3.2\bin\Debug\4.3.exe
Please input the absolute path of your source file:E:\新桌面\编译原理提交\源代码\4.3.2\测试程序V2\test3.txt
Would you like to optimize the final code?[default:Y](Y/n):Y
Semantic Analyze Complete...

Global table:
EntryType  name      addr  kind  value  paramum  arraysize
TConst     x          0     1     10     -----  -----
TFunc      factorial  1     1     -----  1         -----
TFunc      main       2     0     -----  0         -----

factorial:
EntryType  name      addr  kind  value  paramum  arraysize
TVariable  $t0        1     1     -----  -----  -----
TVariable  $t1        2     1     -----  -----  -----
TVariable  $t2        3     1     -----  -----  -----
TVariable  $t3        4     1     -----  -----  -----
TVariable  $t4        5     1     -----  -----  -----
TArg       x          0     1     -----  -----  -----

main:
EntryType  name      addr  kind  value  paramum  arraysize
TVariable  $t5        1     1     -----  -----  -----
TVariable  result     0     1     -----  -----  -----

midcode :
Func_begin factorial
{
x          0     $t0
lwp
return
0
Label0
x          1     $t1
lwp
return
1
Label2
x          1     $t2
PARA
$t2
call      factorial
*
x          $t3   $t4   $t3
return
$t4
Func_end factorial
Func_begin main
PARA
10
call      factorial
result    int    result
printf
Func_end   main
```

- 经过优化后的汇编代码

```
.data
    x: .word 10
.text
    j    main
factorial:
    sw   $ra,0($sp)
    sw   $fp,-4($sp)
    add  $fp,$sp,$0
    subi $sp,$sp,24
    lw   $s0,36($fp)
```

```

    li $t2,0
    bgt $s0,$t2,Lab1e0
    li $v1,0
    add $sp,$fp,$0
    lw $ra,0($sp)
    addi $sp,$sp,4
    lw $fp,-8($sp)
    jr $ra
Lab1e0:
    li $t2,1
    bne $s0,$t2,Lab1e2
    li $v1,1
    add $sp,$fp,$0
    lw $ra,0($sp)
    addi $sp,$sp,4
    lw $fp,-8($sp)
    jr $ra
Lab1e2:
    li $t2,1
    sub $s1,$s0,$t2
    subi $sp,$sp,4
    sw $s1,0($sp)
    sw $s0,-4($sp)
    sw $s1,-8($sp)
    sw $s2,-12($sp)
    sw $s3,-16($sp)
    sw $s4,-20($sp)
    sw $s5,-24($sp)
    sw $s6,-28($sp)
    sw $s7,-32($sp)
    subi $sp,$sp,32
    subi $sp,$sp,4
    jal factorial
    lw $s7,0($sp)
    lw $s6,4($sp)
    lw $s5,8($sp)
    lw $s4,12($sp)
    lw $s3,16($sp)
    lw $s2,20($sp)
    lw $s1,24($sp)
    lw $s0,28($sp)
    addi $sp,$sp,32
    li $t1,4
    add $sp,$sp,$t1
    move $s2,$v1
    mul $s3,$s0,$s2
    move $v1,$s3
    add $sp,$fp,$0
    lw $ra,0($sp)
    addi $sp,$sp,4
    lw $fp,-8($sp)
    jr $ra
main:
    sw $ra,0($sp)
    sw $fp,-4($sp)
    add $fp,$sp,$0
    subi $sp,$sp,12
    subi $sp,$sp,4

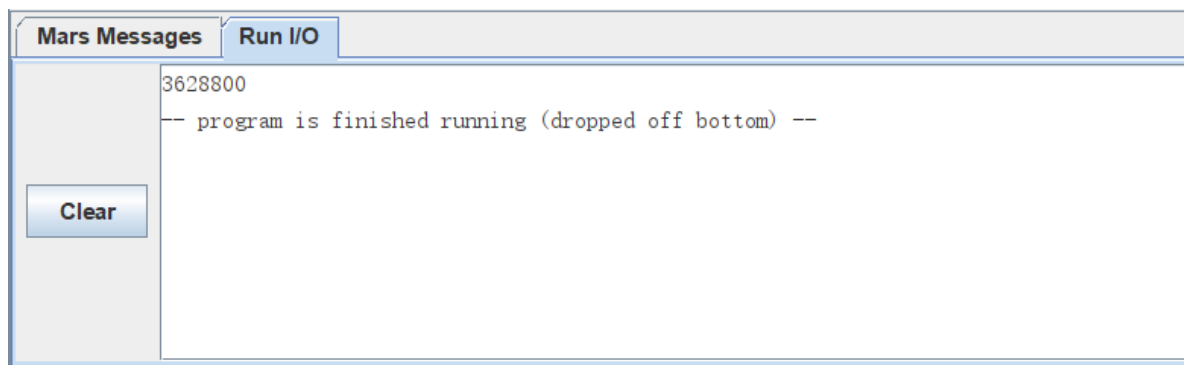
```

```

li    $t1,10
sw    $t1,0($sp)
sw    $s0,-4($sp)
sw    $s1,-8($sp)
sw    $s2,-12($sp)
sw    $s3,-16($sp)
sw    $s4,-20($sp)
sw    $s5,-24($sp)
sw    $s6,-28($sp)
sw    $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal    factorial
lw     $s7,0($sp)
lw     $s6,4($sp)
lw     $s5,8($sp)
lw     $s4,12($sp)
lw     $s3,16($sp)
lw     $s2,20($sp)
lw     $s1,24($sp)
lw     $s0,28($sp)
addi   $sp,$sp,32
li     $t1,4
add    $sp,$sp,$t1
move   $s0,$v1
move   $a0,$s0
li     $v0,1
syscall

```

- Mars运行结果



## 2.汉诺塔问题

- 源代码

```

void move(int n,char a,char b)
{
    printf(n);
    printf(":",a);
    printf("-->",b);
    printf("\n");
    return;
}
void Hanoi(int n,char a,char b,char c)
{
    if(n==1)
        move(n,a,c);
}

```

```

    else
    {
        Hanoi(n - 1,a,c,b);
        move(n,a,c);
        Hanoi(n - 1,b,a,c);
    }
    return;
}

void main()
{
    int n;
    printf("inputnumber:");
    scanf(n);
    Hanoi(n, 'A', 'B', 'C');
}

```

## • 中间产物

```

E:\新桌面\编译原理提交\源代码\4.3.2\bin\Debug\4.3.exe
Please input the absolute path of your source file:E:\新桌面\编译原理提交\源代码\4.3.2\测试程序V2\testnew2.txt
Would you like to optimize the final code?[default:Y](Y/n):
Semantic Analyze Complete...

Global table:
EntryType      name      addr      kind      value      paramum      arraysize
TFunc          Hanoi      1          0          -----      4          -----
TFunc          move       0          0          -----      3          -----
TFunc          main       2          0          -----      0          -----

move:
EntryType      name      addr      kind      value      paramum      arraysize
TArg          n          0          1          -----      -----      -----
TArg          a          1          2          -----      -----      -----
TArg          b          2          2          -----      -----      -----

Hanoi:
EntryType      name      addr      kind      value      paramum      arraysize
TVariable     $t0        4          1          -----      -----      -----
TVariable     $t1        5          1          -----      -----      -----
TVariable     $t2        6          1          -----      -----      -----
TArg          n          0          1          -----      -----      -----
TArg          a          1          2          -----      -----      -----
TArg          b          2          2          -----      -----      -----
TArg          c          3          2          -----      -----      -----

main:
EntryType      name      addr      kind      value      paramum      arraysize
TVariable     n          0          1          -----      -----      -----

midcode :
Func_begin     move      n          int
printf         :         a          char
printf         -->       b          char
printf         \n
return
Func_end       move
Func_begin     Hanoi     1          $t0
==
jmp            Lable0 1          $t0
PARA          n
PARA          a
PARA          c
call          move
jmp            Lable1
Lable         Lable0
n              1          $t1
PARA          $t1
PARA          a
PARA          c
PARA          b
call          Hanoi
PARA          n
PARA          a
PARA          c
call          move
n              1          $t2
PARA          $t2
PARA          b
PARA          a
PARA          c
call          Hanoi
Lable         Lable1
return
Func_end       Hanoi
Func_begin     main
printf         inputnumber:
scanf          n
PARA          n
PARA          'A'
PARA          'B'
PARA          'C'
call          Hanoi
Func_end       main

```

```

Func_end       move
Func_begin     Hanoi
==
n              1          $t0
jmp            Lable0 1          $t0
PARA          n
PARA          a
PARA          c
call          move
jmp            Lable1
Lable         Lable0
n              1          $t1
PARA          $t1
PARA          a
PARA          c
PARA          b
call          Hanoi
PARA          n
PARA          a
PARA          c
call          move
n              1          $t2
PARA          $t2
PARA          b
PARA          a
PARA          c
call          Hanoi
Lable         Lable1
return
Func_end       Hanoi
Func_begin     main
printf         inputnumber:
scanf          n
PARA          n
PARA          'A'
PARA          'B'
PARA          'C'
call          Hanoi
Func_end       main

mips code already generated...

Process returned 0 (0x0)   execution time : 4.108 s
Press any key to continue.

```

## • 经过优化后的汇编代码

```

.data
    msg1: .asciiz ":"
    msg2: .asciiz "-->"
    msg3: .asciiz "\n"
    msg4: .asciiz "inputnumber:"
.text
    j     main
move:
    sw    $ra,0($sp)
    sw    $fp,-4($sp)
    add   $fp,$sp,$0
    subi  $sp,$sp,4
    lw    $s0,44($fp)
    lw    $s1,40($fp)
    lw    $s2,36($fp)
    move  $a0,$s0
    li    $v0,1
    syscall
    li    $v0,4
    la    $a0,msg1
    syscall
    move  $a0,$s1
    li    $v0,11
    syscall
    li    $v0,4
    la    $a0,msg2
    syscall
    move  $a0,$s2
    li    $v0,11
    syscall
    li    $v0,4
    la    $a0,msg3
    syscall
    add   $sp,$fp,$0
    lw    $ra,0($sp)
    addi  $sp,$sp,4
    lw    $fp,-8($sp)
    jr    $ra
    add   $sp,$fp,$0
    lw    $ra,0($sp)
    addi  $sp,$sp,4
    lw    $fp,-8($sp)
    jr    $ra
Hanoi:
    sw    $ra,0($sp)
    sw    $fp,-4($sp)
    add   $fp,$sp,$0
    subi  $sp,$sp,16
    lw    $s0,48($fp)
    lw    $s1,44($fp)
    lw    $s2,36($fp)
    lw    $s4,40($fp)
    li    $t2,1
    bne   $s0,$t2,Lab1e0
    subi  $sp,$sp,4
    sw    $s0,0($sp)
    subi  $sp,$sp,4

```

```

sw    $s1,0($sp)
subi   $sp,$sp,4
sw    $s2,0($sp)
sw    $s0,-4($sp)
sw    $s1,-8($sp)
sw    $s2,-12($sp)
sw    $s3,-16($sp)
sw    $s4,-20($sp)
sw    $s5,-24($sp)
sw    $s6,-28($sp)
sw    $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal move
lw    $s7,0($sp)
lw    $s6,4($sp)
lw    $s5,8($sp)
lw    $s4,12($sp)
lw    $s3,16($sp)
lw    $s2,20($sp)
lw    $s1,24($sp)
lw    $s0,28($sp)
addi   $sp,$sp,32
li     $t1,12
add    $sp,$sp,$t1
j      Label1

```

Label0:

```

li     $t2,1
sub    $s3,$s0,$t2
subi   $sp,$sp,4
sw    $s3,0($sp)
subi   $sp,$sp,4
sw    $s1,0($sp)
subi   $sp,$sp,4
sw    $s2,0($sp)
subi   $sp,$sp,4
sw    $s4,0($sp)
sw    $s0,-4($sp)
sw    $s1,-8($sp)
sw    $s2,-12($sp)
sw    $s3,-16($sp)
sw    $s4,-20($sp)
sw    $s5,-24($sp)
sw    $s6,-28($sp)
sw    $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal Hanoi
lw    $s7,0($sp)
lw    $s6,4($sp)
lw    $s5,8($sp)
lw    $s4,12($sp)
lw    $s3,16($sp)
lw    $s2,20($sp)
lw    $s1,24($sp)
lw    $s0,28($sp)
addi   $sp,$sp,32
li     $t1,16

```

```

add $sp,$sp,$t1
subi $sp,$sp,4
sw $s0,0($sp)
subi $sp,$sp,4
sw $s1,0($sp)
subi $sp,$sp,4
sw $s2,0($sp)
sw $s0,-4($sp)
sw $s1,-8($sp)
sw $s2,-12($sp)
sw $s3,-16($sp)
sw $s4,-20($sp)
sw $s5,-24($sp)
sw $s6,-28($sp)
sw $s7,-32($sp)
subi $sp,$sp,32
subi $sp,$sp,4
jal move
lw $s7,0($sp)
lw $s6,4($sp)
lw $s5,8($sp)
lw $s4,12($sp)
lw $s3,16($sp)
lw $s2,20($sp)
lw $s1,24($sp)
lw $s0,28($sp)
addi $sp,$sp,32
li $t1,12
add $sp,$sp,$t1
li $t2,1
sub $s5,$s0,$t2
subi $sp,$sp,4
sw $s5,0($sp)
subi $sp,$sp,4
sw $s4,0($sp)
subi $sp,$sp,4
sw $s1,0($sp)
subi $sp,$sp,4
sw $s2,0($sp)
sw $s0,-4($sp)
sw $s1,-8($sp)
sw $s2,-12($sp)
sw $s3,-16($sp)
sw $s4,-20($sp)
sw $s5,-24($sp)
sw $s6,-28($sp)
sw $s7,-32($sp)
subi $sp,$sp,32
subi $sp,$sp,4
jal Hanoi
lw $s7,0($sp)
lw $s6,4($sp)
lw $s5,8($sp)
lw $s4,12($sp)
lw $s3,16($sp)
lw $s2,20($sp)
lw $s1,24($sp)
lw $s0,28($sp)

```

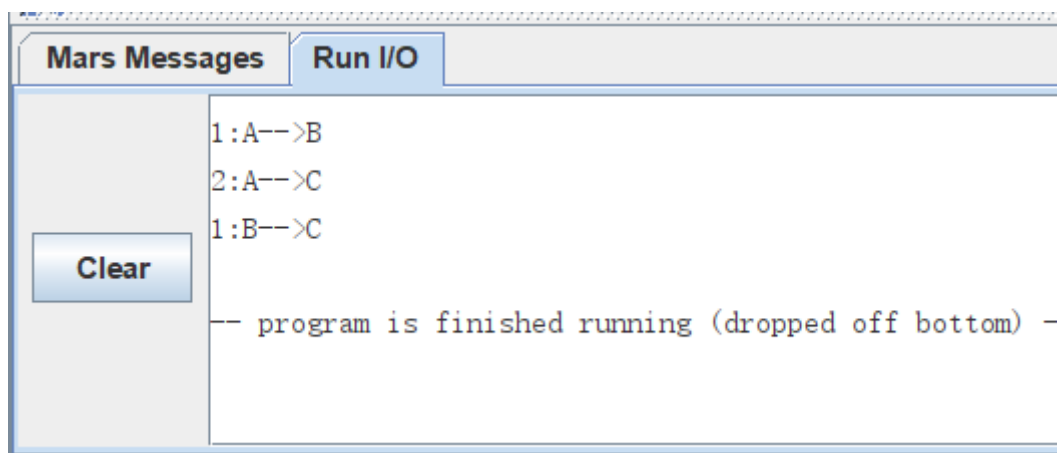
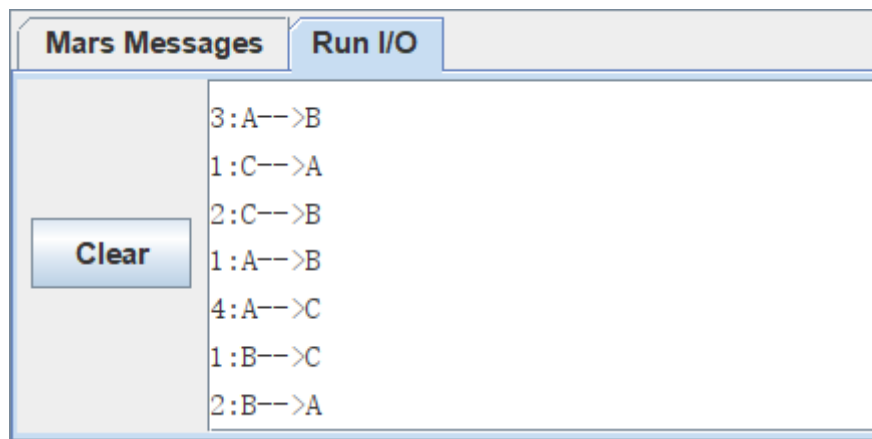
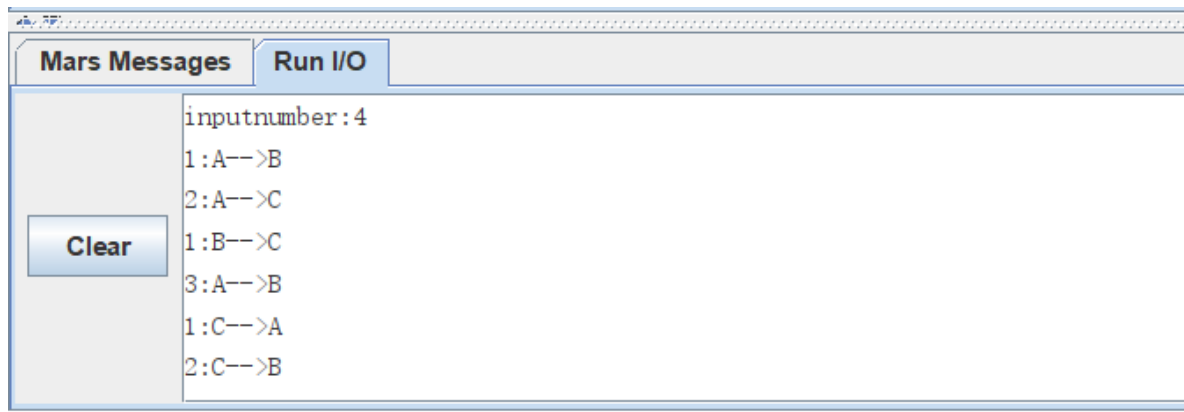
```

    addi    $sp,$sp,32
    li      $t1,16
    add     $sp,$sp,$t1
Label1:
    add     $sp,$fp,$0
    lw      $ra,0($sp)
    addi    $sp,$sp,4
    lw      $fp,-8($sp)
    jr      $ra
    add     $sp,$fp,$0
    lw      $ra,0($sp)
    addi    $sp,$sp,4
    lw      $fp,-8($sp)
    jr      $ra
main:
    sw      $ra,0($sp)
    sw      $fp,-4($sp)
    add     $fp,$sp,$0
    subi    $sp,$sp,8
    li      $v0,4
    la      $a0,msg4
    syscall
    li      $v0,5
    syscall
    move     $s0,$v0
    subi    $sp,$sp,4
    sw      $s0,0($sp)
    subi    $sp,$sp,4
    li      $t1,'A'
    sw      $t1,0($sp)
    subi    $sp,$sp,4
    li      $t1,'B'
    sw      $t1,0($sp)
    subi    $sp,$sp,4
    li      $t1,'C'
    sw      $t1,0($sp)
    sw      $s0,-4($sp)
    sw      $s1,-8($sp)
    sw      $s2,-12($sp)
    sw      $s3,-16($sp)
    sw      $s4,-20($sp)
    sw      $s5,-24($sp)
    sw      $s6,-28($sp)
    sw      $s7,-32($sp)
    subi    $sp,$sp,32
    subi    $sp,$sp,4
    jal     Hanoi
    lw      $s7,0($sp)
    lw      $s6,4($sp)
    lw      $s5,8($sp)
    lw      $s4,12($sp)
    lw      $s3,16($sp)
    lw      $s2,20($sp)
    lw      $s1,24($sp)
    lw      $s0,28($sp)
    addi    $sp,$sp,32
    li      $t1,16
    add     $sp,$sp,$t1

```



- Mars运行结果



### 3.递归与非递归的折半查找算法

- 源代码

```
int IntArray[10];

int RecursiveBinarySearch(int Begin, int End, int Target) {
    int Middle;

    if (Begin >= End)
        return (-1);

    Middle = Begin + (End - Begin) / 2;
    if (IntArray[Middle] > Target)
        return (RecursiveBinarySearch(Begin, Middle, Target));
    if (IntArray[Middle] < Target)
        return (RecursiveBinarySearch(Middle + 1, End, Target));
}
```

```

        return (Middle);
    }

    int IterativeBinarySearch(int Begin, int End, int Target) {
        int Middle;

        if(Begin<End){
            do {
                Middle = Begin + (End - Begin) / 2;
                if (IntArray[Middle] < Target)
                    Begin = Middle + 1;
                else if (IntArray[Middle] > Target)
                    End = Middle;
                else
                    return (Middle);
            }while (Begin < End)
        }

        return (-1);
    }

    void InitializeIntArray(int a) {
        IntArray[0] = 0;
        IntArray[1] = 1;
        IntArray[2] = 2;
        IntArray[3] = 3;
        IntArray[4] = 4;
        IntArray[5] = 5;
        IntArray[6] = 6;
        IntArray[7] = 7;
        IntArray[8] = 8;
        IntArray[9] = 9;
    }

    void TestIBS(int a) {
        printf("Test IterativeBinarySearch:");
        printf("Expected 1: ", IterativeBinarySearch(0, 10, 1));
        printf("Expected 5: ", IterativeBinarySearch(0, 10, 5));
        printf("Expected 7: ", IterativeBinarySearch(0, 10, 7));
        printf("Expected -1: ", IterativeBinarySearch(0, 10, 10));
    }

    void TestRBS(int a) {
        printf("Test RecursiveBinarySearch:");
        printf("Expected 1: ", RecursiveBinarySearch(0, 10, 1));
        printf("Expected 5: ", RecursiveBinarySearch(0, 10, 5));
        printf("Expected 7: ", RecursiveBinarySearch(0, 10, 7));
        printf("Expected -1: ", RecursiveBinarySearch(0, 10, 10));
    }

    void main() {
        InitializeIntArray(0);
        TestIBS(0);
        TestRBS(0);
    }

```

```
}
```

- 中间产物(部分截图)

```

[新桌面]\编译原理提交\源代码\4.3.2\bin\Debug\4.3.exe
Please input the absolute path of your source file:E:\新桌面\编译原理提交\源代码\4.3.2\测试代码\3\algot.txt
Would you like to optimize the final code?[default:Y](Y/n):
Semantic Analyze Complete...

Global table:
EntryType      name      addr  kind  value  paramum  arraysize
TArray         IntArray  0      1      -----  -----  10
TFunc          TFunc     12     0      0       1         -----
TFunc          TestIBS   13     0      1       1         -----
TFunc          RecursiveBinarySearch 10     1      1       3         -----
TFunc          IterativeBinarySearch 11     1      1       3         -----
TFunc          main     15     0      0       0         -----
TFunc          TestRBS   14     0      1       1         -----

RecursiveBinarySearch:
EntryType      name      addr  kind  value  paramum  arraysize
TVariable      $t0       4      1      -----  -----  -----
TVariable      $t1       5      1      -----  -----  -----
TVariable      $t2       6      1      -----  -----  -----
TVariable      $t3       7      1      -----  -----  -----
TVariable      $t4       8      1      -----  -----  -----
TVariable      $t5       9      1      -----  -----  -----
TVariable      $t6      10      1      -----  -----  -----
TVariable      $t7      11      1      -----  -----  -----
TVariable      $t8      12      1      -----  -----  -----
TVariable      $t9      13      1      -----  -----  -----
TVariable      $t10     14      1      -----  -----  -----
TArg           Begin    0      1      -----  -----  -----
TArg           End      1      1      -----  -----  -----
TVariable      Middle   3      1      -----  -----  -----
TArg           Target   2      1      -----  -----  -----

IterativeBinarySearch:
EntryType      name      addr  kind  value  paramum  arraysize
TVariable      $t11     4      1      -----  -----  -----
TVariable      $t20     13     1      -----  -----  -----
TVariable      $t12     5      1      -----  -----  -----
TVariable      $t13     6      1      -----  -----  -----
TVariable      $t14     7      1      -----  -----  -----
TVariable      $t15     8      1      -----  -----  -----
TVariable      $t16     9      1      -----  -----  -----
TVariable      $t17    10      1      -----  -----  -----
TVariable      $t18    11      1      -----  -----  -----
TVariable      $t19    12      1      -----  -----  -----
TArg           Begin    0      1      -----  -----  -----
TArg           End      1      1      -----  -----  -----
TVariable      Middle   3      1      -----  -----  -----
TArg           Target   2      1      -----  -----  -----

InitializeIntArray:

```

- 优化后的汇编代码

```
.data
    IntArray:    .space 40
    msg1:        .ascii "Test IterativeBinarySearch:"
    msg2:        .ascii "Expected 1: "
    msg3:        .ascii "Expected 5: "
    msg4:        .ascii "Expected 7: "
    msg5:        .ascii "Expected -1: "
    msg6:        .ascii "Test RecursiveBinarySearch:"
    msg7:        .ascii "Expected 1: "
    msg8:        .ascii "Expected 5: "
    msg9:        .ascii "Expected 7: "
    msg10:       .ascii "Expected -1: "

.text
    j    main
RecursiveBinarySearch:
    sw   $ra,0($sp)
    sw   $fp,-4($sp)
    add  $fp,$sp,$0
    subi    $sp,$sp,52
    lw   $s1,44($fp)
    lw   $s2,36($fp)
    lw   $s3,40($fp)
    blt  $s1,$s3,Lab1e0
    li   $v1,-1
    add  $sp,$fp,$0
    lw   $ra,0($sp)
    addi    $sp,$sp,4
    lw   $fp,-8($sp)
    jr   $ra
Lab1e0:
    sub  $s4,$s3,$s1
    li   $t2,2
```

```

div $s4,$t2
mflo $s5
add $s0,$s1,$s5
move $t1,$s0
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1
lw $t2,0($t3)
move $s6,$t2
ble $s6,$s2,Label2
subi $sp,$sp,4
sw $s1,0($sp)
subi $sp,$sp,4
sw $s0,0($sp)
subi $sp,$sp,4
sw $s2,0($sp)
sw $s0,-4($sp)
sw $s1,-8($sp)
sw $s2,-12($sp)
sw $s3,-16($sp)
sw $s4,-20($sp)
sw $s5,-24($sp)
sw $s6,-28($sp)
sw $s7,-32($sp)
subi $sp,$sp,32
subi $sp,$sp,4
jal RecursiveBinarySearch
lw $s7,0($sp)
lw $s6,4($sp)
lw $s5,8($sp)
lw $s4,12($sp)
lw $s3,16($sp)
lw $s2,20($sp)
lw $s1,24($sp)
lw $s0,28($sp)
addi $sp,$sp,32
li $t1,12
add $sp,$sp,$t1
move $s7,$v1
move $v1,$s7
add $sp,$fp,$0
lw $ra,0($sp)
addi $sp,$sp,4
lw $fp,-8($sp)
jr $ra

```

Label2:

```

move $t1,$s0
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1
lw $t2,0($t3)
sw $t2,-40($fp)
lw $t1,-40($fp)
bge $t1,$s2,Label4
li $t2,1
lw $t3,-48($fp)
add $t3,$s0,$t2
sw $t3,-48($fp)

```

```

subi    $sp,$sp,4
lw      $t1,-48($fp)
sw      $t1,0($sp)
subi    $sp,$sp,4
sw      $s3,0($sp)
subi    $sp,$sp,4
sw      $s2,0($sp)
sw      $s0,-4($sp)
sw      $s1,-8($sp)
sw      $s2,-12($sp)
sw      $s3,-16($sp)
sw      $s4,-20($sp)
sw      $s5,-24($sp)
sw      $s6,-28($sp)
sw      $s7,-32($sp)
subi    $sp,$sp,32
subi    $sp,$sp,4
jal RecursiveBinarySearch
lw      $s7,0($sp)
lw      $s6,4($sp)
lw      $s5,8($sp)
lw      $s4,12($sp)
lw      $s3,16($sp)
lw      $s2,20($sp)
lw      $s1,24($sp)
lw      $s0,28($sp)
addi    $sp,$sp,32
li      $t1,12
add     $sp,$sp,$t1
sw      $v1,-52($fp)
lw      $v1,-52($fp)
add     $sp,$fp,$0
lw      $ra,0($sp)
addi    $sp,$sp,4
lw      $fp,-8($sp)
jr      $ra

```

Lable4:

```

move    $v1,$s0
add     $sp,$fp,$0
lw      $ra,0($sp)
addi    $sp,$sp,4
lw      $fp,-8($sp)
jr      $ra

```

IterativeBinarySearch:

```

sw      $ra,0($sp)
sw      $fp,-4($sp)
add     $fp,$sp,$0
subi    $sp,$sp,48
lw      $s1,44($fp)
lw      $s2,40($fp)
lw      $s6,36($fp)
bge     $s1,$s2,Lable6

```

Lable7:

```

sub     $s3,$s2,$s1
li      $t2,2
div     $s3,$t2
mflo    $s4
add     $s0,$s1,$s4

```

```

move    $t1,$s0
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1
lw $t2,0($t3)
move    $s5,$t2
bge $s5,$s6,Label8
li $t2,1
add $s1,$s0,$t2
j Label9

```

Label8:

```

move    $t1,$s0
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1
lw $t2,0($t3)
move    $s7,$t2
ble $s7,$s6,Label10
move    $s2,$s0
j Label11

```

Label10:

```

move    $v1,$s0
add $sp,$fp,$0
lw $ra,0($sp)
addi    $sp,$sp,4
lw $fp,-8($sp)
jr $ra

```

Label11:

Label9:

```

blt $s1,$s2,Label7

```

Label6:

```

li $v1,-1
add $sp,$fp,$0
lw $ra,0($sp)
addi    $sp,$sp,4
lw $fp,-8($sp)
jr $ra

```

InitializeIntArray:

```

sw $ra,0($sp)
sw $fp,-4($sp)
add $fp,$sp,$0
subi    $sp,$sp,4
li $t1,0
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1
li $t2,0
sw $t2,0($t3)
li $t1,1
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1
li $t2,1
sw $t2,0($t3)
li $t1,2
sll $t1,$t1,2
la $t2,IntArray
add $t3,$t2,$t1

```

```

li    $t2,2
sw    $t2,0($t3)
li    $t1,3
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,3
sw    $t2,0($t3)
li    $t1,4
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,4
sw    $t2,0($t3)
li    $t1,5
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,5
sw    $t2,0($t3)
li    $t1,6
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,6
sw    $t2,0($t3)
li    $t1,7
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,7
sw    $t2,0($t3)
li    $t1,8
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,8
sw    $t2,0($t3)
li    $t1,9
sll   $t1,$t1,2
la    $t2,IntArray
add   $t3,$t2,$t1
li    $t2,9
sw    $t2,0($t3)
add   $sp,$fp,$0
lw    $ra,0($sp)
addi   $sp,$sp,4
lw    $fp,-8($sp)
jr     $ra

```

TestIBS:

```

sw    $ra,0($sp)
sw    $fp,-4($sp)
add   $fp,$sp,$0
subi   $sp,$sp,20
li    $v0,4
la    $a0,msg1
syscall
subi   $sp,$sp,4

```

```

li    $t1,0
sw    $t1,0($sp)
subi   $sp,$sp,4
li    $t1,10
sw    $t1,0($sp)
subi   $sp,$sp,4
li    $t1,1
sw    $t1,0($sp)
sw    $s0,-4($sp)
sw    $s1,-8($sp)
sw    $s2,-12($sp)
sw    $s3,-16($sp)
sw    $s4,-20($sp)
sw    $s5,-24($sp)
sw    $s6,-28($sp)
sw    $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal    IterativeBinarySearch
lw    $s7,0($sp)
lw    $s6,4($sp)
lw    $s5,8($sp)
lw    $s4,12($sp)
lw    $s3,16($sp)
lw    $s2,20($sp)
lw    $s1,24($sp)
lw    $s0,28($sp)
addi   $sp,$sp,32
li    $t1,12
add    $sp,$sp,$t1
move   $s0,$v1
li    $v0,4
la     $a0,msg2
syscall
move   $a0,$s0
li    $v0,1
syscall
subi   $sp,$sp,4
li    $t1,0
sw    $t1,0($sp)
subi   $sp,$sp,4
li    $t1,10
sw    $t1,0($sp)
subi   $sp,$sp,4
li    $t1,5
sw    $t1,0($sp)
sw    $s0,-4($sp)
sw    $s1,-8($sp)
sw    $s2,-12($sp)
sw    $s3,-16($sp)
sw    $s4,-20($sp)
sw    $s5,-24($sp)
sw    $s6,-28($sp)
sw    $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal    IterativeBinarySearch
lw    $s7,0($sp)

```



```

lw    $s6,4($sp)
lw    $s5,8($sp)
lw    $s4,12($sp)
lw    $s3,16($sp)
lw    $s2,20($sp)
lw    $s1,24($sp)
lw    $s0,28($sp)
addi   $sp,$sp,32
li     $t1,12
add    $sp,$sp,$t1
move   $s1,$v1
li     $v0,4
la     $a0,msg3
syscall
move   $a0,$s1
li     $v0,1
syscall
subi    $sp,$sp,4
li     $t1,0
sw     $t1,0($sp)
subi    $sp,$sp,4
li     $t1,10
sw     $t1,0($sp)
subi    $sp,$sp,4
li     $t1,7
sw     $t1,0($sp)
sw     $s0,-4($sp)
sw     $s1,-8($sp)
sw     $s2,-12($sp)
sw     $s3,-16($sp)
sw     $s4,-20($sp)
sw     $s5,-24($sp)
sw     $s6,-28($sp)
sw     $s7,-32($sp)
subi    $sp,$sp,32
subi    $sp,$sp,4
jal    IterativeBinarySearch
lw     $s7,0($sp)
lw     $s6,4($sp)
lw     $s5,8($sp)
lw     $s4,12($sp)
lw     $s3,16($sp)
lw     $s2,20($sp)
lw     $s1,24($sp)
lw     $s0,28($sp)
addi    $sp,$sp,32
li     $t1,12
add    $sp,$sp,$t1
move   $s2,$v1
li     $v0,4
la     $a0,msg4
syscall
move   $a0,$s2
li     $v0,1
syscall
subi    $sp,$sp,4
li     $t1,0
sw     $t1,0($sp)

```

```

subi    $sp,$sp,4
li      $t1,10
sw      $t1,0($sp)
subi    $sp,$sp,4
li      $t1,10
sw      $t1,0($sp)
sw      $s0,-4($sp)
sw      $s1,-8($sp)
sw      $s2,-12($sp)
sw      $s3,-16($sp)
sw      $s4,-20($sp)
sw      $s5,-24($sp)
sw      $s6,-28($sp)
sw      $s7,-32($sp)
subi    $sp,$sp,32
subi    $sp,$sp,4
jal     IterativeBinarySearch
lw      $s7,0($sp)
lw      $s6,4($sp)
lw      $s5,8($sp)
lw      $s4,12($sp)
lw      $s3,16($sp)
lw      $s2,20($sp)
lw      $s1,24($sp)
lw      $s0,28($sp)
addi    $sp,$sp,32
li      $t1,12
add     $sp,$sp,$t1
move    $s3,$v1
li      $v0,4
la      $a0,msg5
syscall
move    $a0,$s3
li      $v0,1
syscall
add     $sp,$fp,$0
lw      $ra,0($sp)
addi    $sp,$sp,4
lw      $fp,-8($sp)
jr      $ra

```

TestRBS:

```

sw      $ra,0($sp)
sw      $fp,-4($sp)
add     $fp,$sp,$0
subi    $sp,$sp,20
li      $v0,4
la      $a0,msg6
syscall
subi    $sp,$sp,4
li      $t1,0
sw      $t1,0($sp)
subi    $sp,$sp,4
li      $t1,10
sw      $t1,0($sp)
subi    $sp,$sp,4
li      $t1,1
sw      $t1,0($sp)
sw      $s0,-4($sp)

```

```

sw    $s1,-8($sp)
sw    $s2,-12($sp)
sw    $s3,-16($sp)
sw    $s4,-20($sp)
sw    $s5,-24($sp)
sw    $s6,-28($sp)
sw    $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal RecursiveBinarySearch
lw     $s7,0($sp)
lw     $s6,4($sp)
lw     $s5,8($sp)
lw     $s4,12($sp)
lw     $s3,16($sp)
lw     $s2,20($sp)
lw     $s1,24($sp)
lw     $s0,28($sp)
addi   $sp,$sp,32
li     $t1,12
add    $sp,$sp,$t1
move   $s0,$v1
li     $v0,4
la     $a0,msg7
syscall
move   $a0,$s0
li     $v0,1
syscall
subi   $sp,$sp,4
li     $t1,0
sw     $t1,0($sp)
subi   $sp,$sp,4
li     $t1,10
sw     $t1,0($sp)
subi   $sp,$sp,4
li     $t1,5
sw     $t1,0($sp)
sw     $s0,-4($sp)
sw     $s1,-8($sp)
sw     $s2,-12($sp)
sw     $s3,-16($sp)
sw     $s4,-20($sp)
sw     $s5,-24($sp)
sw     $s6,-28($sp)
sw     $s7,-32($sp)
subi   $sp,$sp,32
subi   $sp,$sp,4
jal RecursiveBinarySearch
lw     $s7,0($sp)
lw     $s6,4($sp)
lw     $s5,8($sp)
lw     $s4,12($sp)
lw     $s3,16($sp)
lw     $s2,20($sp)
lw     $s1,24($sp)
lw     $s0,28($sp)
addi   $sp,$sp,32
li     $t1,12

```

```

add $sp,$sp,$t1
move    $s1,$v1
li      $v0,4
la      $a0,msg8
syscall
move    $a0,$s1
li      $v0,1
syscall
subi     $sp,$sp,4
li      $t1,0
sw      $t1,0($sp)
subi     $sp,$sp,4
li      $t1,10
sw      $t1,0($sp)
subi     $sp,$sp,4
li      $t1,7
sw      $t1,0($sp)
sw      $s0,-4($sp)
sw      $s1,-8($sp)
sw      $s2,-12($sp)
sw      $s3,-16($sp)
sw      $s4,-20($sp)
sw      $s5,-24($sp)
sw      $s6,-28($sp)
sw      $s7,-32($sp)
subi     $sp,$sp,32
subi     $sp,$sp,4
jal RecursiveBinarySearch
lw      $s7,0($sp)
lw      $s6,4($sp)
lw      $s5,8($sp)
lw      $s4,12($sp)
lw      $s3,16($sp)
lw      $s2,20($sp)
lw      $s1,24($sp)
lw      $s0,28($sp)
addi     $sp,$sp,32
li      $t1,12
add $sp,$sp,$t1
move    $s2,$v1
li      $v0,4
la      $a0,msg9
syscall
move    $a0,$s2
li      $v0,1
syscall
subi     $sp,$sp,4
li      $t1,0
sw      $t1,0($sp)
subi     $sp,$sp,4
li      $t1,10
sw      $t1,0($sp)
subi     $sp,$sp,4
li      $t1,10
sw      $t1,0($sp)
sw      $s0,-4($sp)
sw      $s1,-8($sp)
sw      $s2,-12($sp)

```

```

sw $s3,-16($sp)
sw $s4,-20($sp)
sw $s5,-24($sp)
sw $s6,-28($sp)
sw $s7,-32($sp)
subi $sp,$sp,32
subi $sp,$sp,4
jal RecursiveBinarySearch
lw $s7,0($sp)
lw $s6,4($sp)
lw $s5,8($sp)
lw $s4,12($sp)
lw $s3,16($sp)
lw $s2,20($sp)
lw $s1,24($sp)
lw $s0,28($sp)
addi $sp,$sp,32
li $t1,12
add $sp,$sp,$t1
move $s3,$v1
li $v0,4
la $a0,msg10
syscall
move $a0,$s3
li $v0,1
syscall
add $sp,$fp,$0
lw $ra,0($sp)
addi $sp,$sp,4
lw $fp,-8($sp)
jr $ra
main:
sw $ra,0($sp)
sw $fp,-4($sp)
add $fp,$sp,$0
subi $sp,$sp,4
subi $sp,$sp,4
li $t1,0
sw $t1,0($sp)
sw $s0,-4($sp)
sw $s1,-8($sp)
sw $s2,-12($sp)
sw $s3,-16($sp)
sw $s4,-20($sp)
sw $s5,-24($sp)
sw $s6,-28($sp)
sw $s7,-32($sp)
subi $sp,$sp,32
subi $sp,$sp,4
jal InitializeIntArray
lw $s7,0($sp)
lw $s6,4($sp)
lw $s5,8($sp)
lw $s4,12($sp)
lw $s3,16($sp)
lw $s2,20($sp)
lw $s1,24($sp)
lw $s0,28($sp)

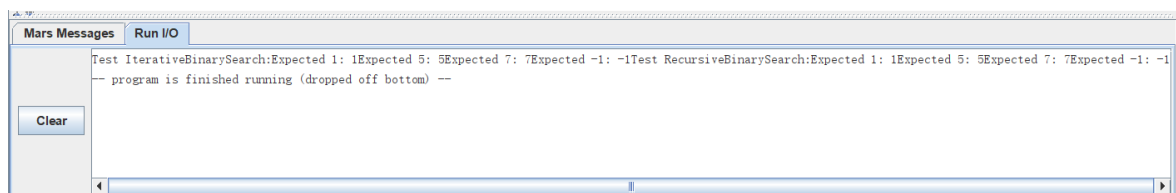
```

```

addi    $sp,$sp,32
li      $t1,4
add     $sp,$sp,$t1
subi    $sp,$sp,4
li      $t1,0
sw      $t1,0($sp)
sw      $s0,-4($sp)
sw      $s1,-8($sp)
sw      $s2,-12($sp)
sw      $s3,-16($sp)
sw      $s4,-20($sp)
sw      $s5,-24($sp)
sw      $s6,-28($sp)
sw      $s7,-32($sp)
subi    $sp,$sp,32
subi    $sp,$sp,4
jal     TestIBS
lw      $s7,0($sp)
lw      $s6,4($sp)
lw      $s5,8($sp)
lw      $s4,12($sp)
lw      $s3,16($sp)
lw      $s2,20($sp)
lw      $s1,24($sp)
lw      $s0,28($sp)
addi    $sp,$sp,32
li      $t1,4
add     $sp,$sp,$t1
subi    $sp,$sp,4
li      $t1,0
sw      $t1,0($sp)
sw      $s0,-4($sp)
sw      $s1,-8($sp)
sw      $s2,-12($sp)
sw      $s3,-16($sp)
sw      $s4,-20($sp)
sw      $s5,-24($sp)
sw      $s6,-28($sp)
sw      $s7,-32($sp)
subi    $sp,$sp,32
subi    $sp,$sp,4
jal     TestRBS
lw      $s7,0($sp)
lw      $s6,4($sp)
lw      $s5,8($sp)
lw      $s4,12($sp)
lw      $s3,16($sp)
lw      $s2,20($sp)
lw      $s1,24($sp)
lw      $s0,28($sp)
addi    $sp,$sp,32
li      $t1,4
add     $sp,$sp,$t1

```

- Mars运行结果



## 4.冒泡排序

- 源代码

```
void main()
{
    int a[10];
    int i, j, temp;
    printf("Please input ten numbers:\n");
    for (i = 0; i < 10; i=i + 1) {
        scanf(temp);
        a[i]=temp;
    }
    for (i = 0; i < 9; i=i + 1) {

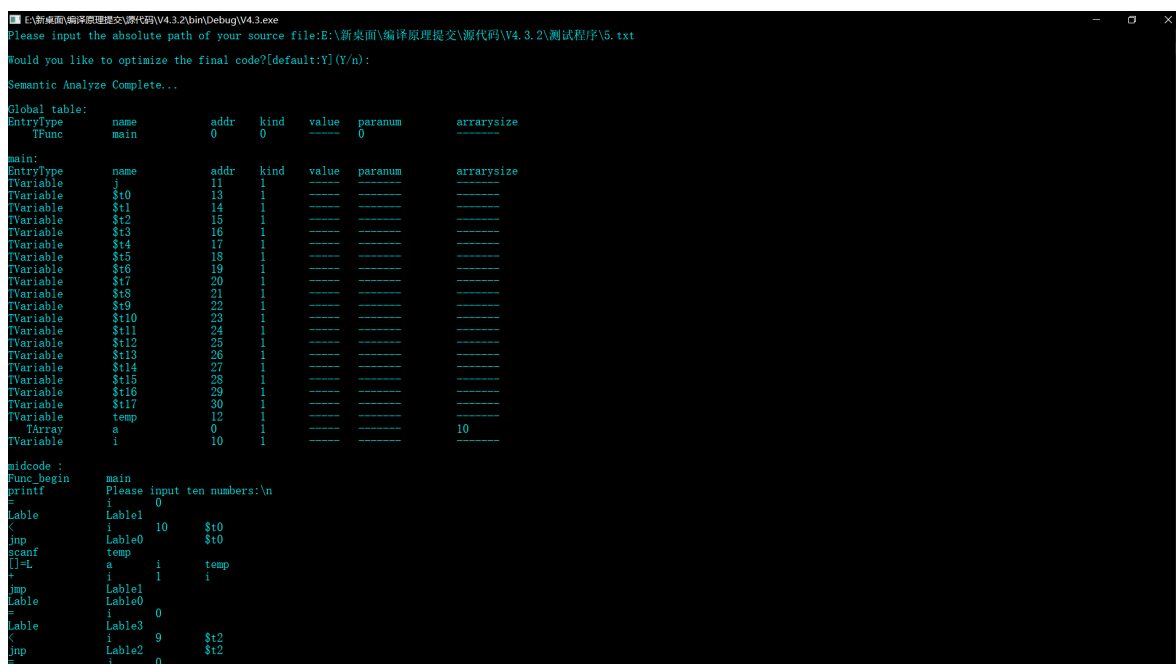
        for (j = 0; j < 9 - i; j=j + 1)
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }

    }

    for (i = 0; i < 10; i=i + 1)
        printf(a[i]);

    return ;
}
```

- 中间产物(部分截图)



- 优化后的汇编

```
.data
    msg1: .asciiz "Please input ten numbers:\n"
.text
    j     main
main:
    sw    $ra,0($sp)
    sw    $fp,-4($sp)
    add   $fp,$sp,$0
    subi  $sp,$sp,128
    li    $v0,4
    la    $a0,msg1
    syscall
    li    $t2,0
    move  $s0,$t2
Lab1e1:
    li    $t2,10
    bge   $s0,$t2,Lab1e0
    li    $v0,5
    syscall
    move  $s2,$v0
    move  $t1,$s0
    sll   $t1,$t1,2
    la    $t2,-8($fp)
    sub   $t3,$t2,$t1
    sw    $s2,0($t3)
    li    $t2,1
    add   $s0,$s0,$t2
    j     Lab1e1
Lab1e0:
    li    $t2,0
    move  $s0,$t2
Lab1e3:
    li    $t2,9
    bge   $s0,$t2,Lab1e2
    li    $t2,0
    move  $s1,$t2
Lab1e5:
    li    $t1,9
    sub   $s4,$t1,$s0
    bge   $s1,$s4,Lab1e4
    move  $t1,$s1
    sll   $t1,$t1,2
    la    $t2,-8($fp)
    sub   $t3,$t2,$t1
    lw    $t2,0($t3)
    move  $s5,$t2
    li    $t2,1
    add   $s6,$s1,$t2
    move  $t1,$s6
    sll   $t1,$t1,2
    la    $t2,-8($fp)
    sub   $t3,$t2,$t1
    lw    $t2,0($t3)
    move  $s7,$t2
    ble   $s5,$s7,Lab1e6
```



```

        move    $t1,$s1
        sll    $t1,$t1,2
        la     $t2,-8($fp)
        sub    $t3,$t2,$t1
        lw     $t2,0($t3)
        sw     $t2,-96($fp)
        lw     $t2,-96($fp)
        move    $s2,$t2
        li     $t2,1
        add    $s3,$s1,$t2
        move    $t1,$s3
        sll    $t1,$t1,2
        la     $t2,-8($fp)
        sub    $t3,$t2,$t1
        lw     $t2,0($t3)
        sw     $t2,-104($fp)
        move    $t1,$s1
        sll    $t1,$t1,2
        la     $t2,-8($fp)
        sub    $t3,$t2,$t1
        lw     $t2,-104($fp)
        sw     $t2,0($t3)
        sw     $s3,-108($fp)
        lw     $t1,-108($fp)
        sll    $t1,$t1,2
        la     $t2,-8($fp)
        sub    $t3,$t2,$t1
        sw     $s2,0($t3)
Lable6:
        li     $t2,1
        add    $s1,$s1,$t2
        j      Lable5
Lable4:
        li     $t2,1
        add    $s0,$s0,$t2
        j      Lable3
Lable2:
        li     $t2,0
        move    $s0,$t2
Lable9:
        li     $t2,10
        bge    $s0,$t2,Lable8
        move    $t1,$s0
        sll    $t1,$t1,2
        la     $t2,-8($fp)
        sub    $t3,$t2,$t1
        lw     $t2,0($t3)
        sw     $t2,-124($fp)
        lw     $a0,-124($fp)
        li     $v0,1
        syscall
        li     $t2,1
        add    $s0,$s0,$t2
        j      Lable9
Lable8:

```

- Mars运行结果

Mars Messages

Run I/O

Clear

Please input ten numbers:  
9  
8  
7  
6  
5  
4

Mars Messages

Run I/O

Clear

4  
3  
2  
1  
0  
0123456789  
-- program is finished running (dropped off bottom) --