1. 求和

```
main:   cout << “Please input a value for N”
        cin >> v0
        If ( v0 <= 0 ) goto end
        t0 = 0;
        While (v0 > 0 ) do
        {
              t0 = t0 + v0;
              v0 = v0 - 1;
        }
        cout << t0;
        goto main
end:    cout<<“The input is not valid, bye!”
```

```
# Cross References:
# v0: N,
# t0: Sum
```

```
        .data
prompt: .asciiz   “\n     Please Input a value for N =    ”
result:  .asciiz   “     The sum of the integers from 1 to N is ”
bye:    .asciiz   “\n    ****The input is not valid, bye!****”
        .globl main
        .text
main:   li      $v0, 4                  # system call code for Print String
        la      $a0, prompt             # load address of prompt into $a0
        syscall                         # print the prompt message
        li      $v0, 5                  # system call code for Read Integer
        syscall                         # reads the value of N into $v0
        blez    $v0,    end             # branch to end if    $v0<=0
        li      $t0, 0                  # clear register $t0 to zero
loop:   add     $t0, $t0, $v0           # sum of integers in register $t0
        addi    $v0, $v0, -1            # summing integers in reverse order
        bnez    $v0,    loop            # branch to loop if $v0 is != zero
        li      $v0, 4                  # system call code for Print String
        la      $a0, result             # load address of message into $a0
        syscall                          # print the string
        li      $v0, 1                  # system call code for Print Integer
        move    $a0,    $t0             # move value to be printed to $a0
        syscall                          # print sum of integers
        b    main                       # branch to main
end:    li      $v0, 4                  # system call code for Print String
        la      $a0, bye                # load address of msg. into $a0
        syscall                          # print the string
        li      $v0, 10                 # terminate program run and
        syscall                          # return control to system
```

2. 整数的平方根（结果也为整数，使用二分搜索法）

```
sqrt(x)       [0, x/2+1]
i=0;
j=x/2+1;
while    (i<=j)
{
    m=(i+j)/2;
    t=m*m;
    if (t==x) return m;
    else if (t<x) i=m+1;
          else j=m-1;
}
return    j
```

```
main:     cout<< “Please input the X: ”;
          cin>>v0;
          t0=0；t1=v0; t2=t1/2+1;
          while    (t0<=t2)
          {
               t3=(t0+t2)/2;
               t4=t3*t3;
               if (t4==t1) goto end;
               else if (t4<t1) t0=t3+1;
                    else t2=t3-1;
          }
          cout<<t2;
          goto exit
end:      cout<<t3
exit:      return
```

```
          .data
prompt:   .asciiz "\n\n Please Input a value: "
result:   .asciiz "\n The square root is: "
          .globl main
          .text
main:     li $v0, 4
          la $a0, prompt
          syscall
          li $v0, 5
          syscall
          li $t0,0
          move $t1,$v0
          srl $t2,$t1,1
          addi $t2,$t2,1
loop:     bgt $t0,$t2,outj
          add $t3,$t0,$t2
          srl $t3,$t3,1
          mul $t4,$t3,$t3
          beq $t4,$t1,end
          blt $t4,$t1,first
          addi $t2,$t3,-1
          b loop
first:    addi $t0,$t3,1
          b loop
outj:     li $v0,4
          la $a0,result
          syscall
          move $a0,$t2
          li $v0,1
          syscall
          b exit
end:      li $v0,4
          la $a0,result
          syscall
          move $a0,$t3
          li $v0,1
          syscall
exit:     li $v0,10
          syscall
```

3. 将无符号数 x 以某进制（<=10）显示（先看 PPT 算法描述）

```
main:   t1=8;   t2= 0;
        t0=11; Mem(t0)=null;
        while   (x >0) {
            t3=x mod t1;
            x=x /t1;
            t0=t0-1;
            Mem(t0)=t3;
            t2=t2+1;
        }
        Mem(10-t2)=' '
        cout<<Mem(10-t2)开始
        return
```

简化版本

```
main:   t1=8;
        t0=11; Mem(t0)=null;
        while   (x >0) {
            t3=x mod t1;
            x=x /t1;
            t0=t0-1;
            Mem(t0)=t3;
        }
        Mem(t0-1)=' '
        cout<<Mem(t0-1)开始
        return
```

```
        .data
buf:    .space 12           #1 空格+10 字符+1 字符串结束的空字符
        .globl main
        .text
main:
        li    $a1, 1857     #测试数字
        li    $t1, 8        #转换成 8 进制
        li    $t2, 1        #转换后的数值长度计数器，初始为 1
        la    $a0, buf      #缓冲区地址
        addi $t0, $a0, 11   # $t0 存放 buf 的最后一个位置
        sb    $0, ($t0)     #将 0x00 存储到最后一个位置（位置 11）
loop:
        div   $a1, $t1      #除以 8
        mflo  $a1           #商成为新的$t0
        mfhi  $t3           #余数放入$t3
        addi $t3, $t3, 0x30 #余数的 ASCII 码
        addi $t0, $t0, -1   #从倒数第二个位置开始依次向前放 ASCII 码
        sb    $t3, ($t0)
        beqz      $a1, out  #当商为 0 时退出循环
        addi $t2, $t2, 1    #计数器加 1
        b     loop
out:
        addi $a0, $a0,10    #获取 buf 中第一个数字前一位置 10-$t2
        sub  $a0, $a0, $t2
        li    $t4, 0x20
        sb    $t4, ($a0)    #将 0x20 存储到第一个数字位置
        li    $v0, 4
        syscall
        li    $v0, 10
        syscall
```

4. 将无符号数十进制数据以十六进制显示（先看 PPT 算法描述）

```
            .data
hex_tab: .ascii     "0123456789ABCDEF"
result:     .ascii     "Hexadecimal value: 0x"
hex_dig:  .asciiz   "XXXXXXXX"
            .text
            .globl main
main:       li      $v0, 5                  #读入一个整数
            syscall
            move      $t0, $v0
            li      $t1, 7                  #循环次数
hexify:     and     $t2, $t0, 0x0f          #取出低 4 位
            srl     $t0, $t0, 4             #逻辑右移 4 位
            lb      $t3, hex_tab($t2)       #低 4 位对应的十六进制
            sb      $t3, hex_dig($t1)
            sub   $t1, $t1,1
            bgez    $t1, hexify             #$t1>=0 则返回 hexify
            li       $v0, 4                 #打印字符串
            la      $a0, result
            syscall
            li      $v0, 10
            syscall
```

课后写一个对应的伪代码算法

5. 打印十六进制值（先看 PPT 算法描述）

```
main:   cout<< “Please input a value: ”;
        cin>>v0;
        t0=v0;
        t1=&strbuf;
        t2=8;
        t3 = t1+10;
        while (t2>0) do
        {
            t4 = t0 & 0x0f;
            t0 = t0>>4;
            if (t4 >= 10)
            then t4 = t4 + 0x37;
            else t4 = t4+ 0x30;
            Mem(t3) = t4;
            t3 = t3 - 1;
            t2 = t2 - 1;
         }
        Mem(t1+11) = 0x00;
        Mem(t1+2) = 0x78;
        Mem(t1+1) = 0x30;
        Mem(t1) = 0x20;
        cout<<strbuf;
exit:   return
```

```
        .data
prompt: .asciiz "\n\n Please input a value: "
result: .ascii "\n The hex is: "
buf:    .space 12
        .text
main:   li $v0, 4
        la $a0, prompt
        syscall
        li $v0, 5
        syscall
        move $t0, $v0
        la $t1, buf
        li $t2, 8
        addi $t3, $t1, 10
loop:   blez $t2, end
        andi $t4, $t0, 0x0f
        srl  $t0, $t0, 4
        bge  $t4, 10, char
        addi $t4, $t4, 0x30
        b    put
char:   addi $t4, $t4, 0x37
put:    sb   $t4, ($t3)
        addi $t3, $t3, -1
        addi $t2, $t2, -1
        b loop
end:    sb $0, 11($t1)
        li $t0, 0x78
        sb $t0, 2($t1)
        li $t0, 0x30
        sb $t0, 1($t1)
        li $t0, 0x20
        sb $t0, ($t1)
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 10
        syscall
```

6. 右对齐打印十进制值（先看 PPT 算法描述）

```
main:   cout<< “Please input a value: ”;
        cin>>v0;
        t0=v0;
        t1=&strbuf;
        t2=11;
        Mem(t1+12)=0x00;
        for (i=11;i>=0;i--)
        {
            Mem(t1+i)=0x20;
        }
        t3=abs(t0);
        while(t3>0) do
        {
            t4=t3 mod 10;
            t3=t3/10;
            Mem(t1+t2)=t4;
            t2=t2-1;
        }
        if (t0<0)
        then Mem(t1+t2)=0x2d;
        cout<<strbuf;
        return
```

```
        .data
prompt: .asciiz "\n\n Please input a value: "
result: .ascii "\n The dec is: "
buf:    .space 13
        .text
main:   li $v0, 4
        la $a0, prompt
        syscall
        li $v0, 5
        syscall
        move $t0, $v0
        la $t1, buf
        li $t2, 11
        sb $0,12($t1)
        li $t8,0x20
space:  add $t9,$t1,$t2
        sb $t8,($t9)
        addi $t2,$t2,-1
        bgez  $t2,space
        abs $t3,$t0
        li   $t2, 11
        li   $t9,10
        add $t1, $t1, $t2
loop:   div  $t3,$t9
        mflo $t3
        mfhi $t4
        addi $t4,$t4,0x30
        sb   $t4,($t1)
        addi    $t1, $t1, -1
        beqz   $t3,sign
        b    loop
sign:   bgez  $t0,exit
        li    $t8,0x2d
        sb    $t8,($t1)
exit:   li $v0, 4
        la $a0, result
        syscall
        li $v0, 10
        syscall
```

7. 求数组元素的最大值和最小值（先看 PPT 算法描述）

```
int N=10;
int array[N]={100,-10,0,23,35,-67,90,10,65,-87};
int *a=array;
int max=min=a[0]; a++;
for (i=1;i<N;i++)
{
    if (max<*a) max=*a;
    if (min>*a) min=*a;
    a++;
}
```

伪代码中的各个变量对应 MIPS 寄存器，交叉引用表如下：

```
# Cross References:
# t0: a
# t1: max
# t2: min
# t3: i
# t4: N
# t5: *a
```

```
        .data
array:  .word 100,-10,0,23,35,-67,90,10,65,-87
        .globl main
        .text
main:   la   $t0, array          # *a=array
        lw   $t1, ($t0)          # max=a[0]
        move $t2, $t1            # min=max
        li   $t3, 1              # i=1
        li   $t4, 10             # N=10
        addi $t0, $t0, 4         # a++
loop:   lw   $t5, ($t0)          # $t5=*a
        bge  $t1, $t5, minck     # if(max>=*a) goto minck
        move $t1, $t5            # max=*a
minck:  ble  $t2, $t5, next      # if (min<=*a) goto next
        move $t2, $t5            # min=*a
next:   addi $t0, $t0, 4         # a++
        addi $t3, $t3, 1         # i++
        blt  $t3, $t4, loop      # if (i<N) Branch to loop
        li   $v0, 10             # terminate program run and
        syscall                  # return control to system
```

8. 递增序数组的插入操作 （先看 PPT 算法描述）

```
int N=10; x=31;
int array[N]={ 4,8,24,28,35,62,67,78,90,96}
i=9;
while (i>=0){
      k=array[i];
      if (k<=x)
      then {
            array[i+1]=x ;
            break;
      }
      else {
            array[i+1]=k;
            i=i-1;
      }
 }
return
```

伪代码中的各个变量对应 MIPS 寄存器，交叉引用表如下：

Cross References:
t0: array
t2: x
t3: k

```
            .data
            .word -1
array:      .word 4,8,24,28,35,62,67,78,90,96
            .space 8
x:          .word 31
            .globl main
            .text
main:       la     $t0, array
            addi   $t0, $t0, 36          # $t0 point to last unit of array
            la     $t1, x
            lw     $t2, ($t1)            # $t2=x    ( lw $t2, x($0) )
again:      lw     $t3, ($t0)            # $t3=k
            ble    $t3, $t2, insert      # compare k to x，if k<=x,insert
            sw     $t3, 4($t0)           # current unit move back 1 unit
            addi   $t0, $t0, -4          # $t0 point to previous unit
            b      again                 # next unit
insert:     sw     $t2, 4($t0)           # $t2 insert to array
            li     $v0, 10               # terminate program run and
            syscall                      # return control to system
```

9. 简单选择排序 （先看 PPT 算法描述）

```
int N=7;
int array[N]={ 49,38,65,97,76,13,27}
int *a0 =array;
int *a1;
for (t0=6;t0>0;t0--){
      *a1=*a0;
       a1++;
      for (t1=t0;t1>0;t1--){
            t2=*a0;
            t3=*a1;
            if (t2>t3) then {
                  Mem[a0]=t3;
                  Mem[a1]=t2;
            }
            a1++;
      }
      a0++;
}
return
```

```
            .data
array:      .word 49,38,65,97,76,13,27
            .globl main
            .text
main:       la     $a0, array          #a0=array
            li     $t0, 6              #外循环变量 t0 初始化为数组长度-1
lp0:        addi $a1, $a0, 4           #外循环起点。内循环的起始比较单元初始化
            move $t1, $t0              #内循环变量 t1 初始化，t1=t0
lp1:        lw     $t2, ($a0)          #内循环起点。取出本趟比较的单元
            lw     $t3, ($a1)          #逐个取出随后的单元
            ble    $t2, $t3, next      #前面的小，则什么都不做
            sw     $t2, ($a1)          #前面的大，则交换 2 个单元
            sw     $t3, ($a0)
next:       addi $a1, $a1, 4           #a1 指向下一个单元
            addi $t1, $t1,-1           #内循环变量 t1--
            bgt    $t1, $0, lp1        #内循环结束，直到 t1=0 退出循环
            addi $a0, $a0, 4           #准备下一轮，a1 指向下一轮比较的单元
            addi $t0, $t0,-1           #外循环变量 t0--
            bgt    $t0, $0, lp0        #外循环结束，直到 t0=0 退出循环
            li     $v0, 10             # terminate program run and
            syscall                    # return control to system
```

10. 统计$a0 中 1 的个数，结果在 $v0 中返回。

```
Count:
        li      $v0, 0
again:  beqz $a0, ret           # if($a0==0) goto ret
        bgez $a0, next          # if(MSB>=0) goto next
        addi $v0, $v0, 1        # count
next:   sll    $a0, $a0, 1      # Left shift $a0 1 bit
        b       again
ret:    jr      $ra             # return
```

11. 计算以空字符（NULL，即'\0'）结尾的字符串长度

```
string_len：
        li $v0,0                # init $v0 (string length)
loop:
        lb $t0, ($a0)
        beq $t0,0x0a,done       # if nl
        beqz $t0,done           # or NULL, we are done
        addu $a0,$a0,1
        addu $v0,$v0,1
        b loop
 done:
        jr $ra

调用函数语句：
        la $a0,string           # call string length proc.
        jal string_len
        move $t0,$v0            # string length in $v0
```

12. 从键盘输入一个字符串，将其中小写字母转换为大写字母，然后显示。先 看 PPT 算法描述）

```
        .data
str:    .space 100
        .globl main
        .text
main:   la   $a0, str          # Code to read a string
        li   $a1, 100          # $a0:string address, $a1:length
        li   $v0, 8
        syscall
        jal  trans             # call function trans
        li   $v0, 4            # Code to print a string
        syscall                # Output results
        li   $v0, 10           # terminate program run and
        syscall                # return control to system


trans:
        move $a2, $a0          # Keep $a0, use $a2 point to string
        li   $t0, 0x61         # t0='a'
        li   $t1, 0x7a         # t1='z'
        li   $t2, 0            # $t2 is counter
loop:   lb   $t3, ($a2)        # Load char to $t3
        beqz $t3, ret          # if($t3==0) goto ret
        blt  $t3, $t0, next    # if($t3<'a')goto next
        bgt  $t3, $t1, next    # if($t3>'z')goto next
        addi $t3, $t3,-0x20    # $t3-=0x20, lesser to upper
        sb   $t3, ($a2)        # Store char $t3 to string
next:   addi $a2, $a2, 1       # $a2 point to next char
        addi $t2, $t2, 1       # $t2++
        blt  $t2, $a1, loop    # if($t2<$a1) loop
ret:    jr   $ra               # return
```

13. 求长度为 N 的数组中所有正数之和以及所有负数之和。（先看 PPT 中文档化）

```
        .data
array:  .word       -4, 5, 8, -1
msg1:   .asciiz     "\n The sum of the positive values = "
msg2:   .asciiz     "\n The sum of the negative values = "
        .globl main
        .text
main:   li $v0, 4               # system call code for print_str
        la $a0, msg1            # load address of msg1. into $a0
        syscall                 # print the string
        la $a0, array           # Initialize address Parameter
        li $a1, 4               # Initialize length Parameter
        jal sum                 # Call sum
        move $a0, $v0           # move value to be printed to $a0
        li $v0, 1               # system call code for print_int
        syscall                 # print sum of Pos:
        li $v0, 4               # system call code for print_str
        la $a0, msg2            # load address of msg2. into $a0
        syscall                 # print the string
        li $v0, 1               # system call code for print_int
        move $a0, $v1           # move value to be printed to $a0
        syscall                 # print sum of neg
        li $v0, 10              # terminate program run and
        syscall                 # return control to system
```

```
sum:
        li   $v0, 0
        li   $v1, 0             # Initialize v0 and v1 to zero
loop:   blez $a1, retzz         # If (a1 <= 0) Branch to Return
        addi $a1, $a1, -1       # Decrement loop count
        lw $t0, 0($a0)          # Get a value from the array
        addi $a0, $a0, 4        # Increment array pointer to next word
        bltz $t0, negg          # If value is negative Branch to negg
        add $v0, $v0, $t0       # Add to the positive sum
        b loop                  # Branch around the next two instructions
negg:   add $v1, $v1, $t0       # Add to the negative sum
        b loop                  # Branch to loop
retzz:  jr $ra                  # Return
```

14. 打印十六进制值

```
        .data
prompt: .asciiz "\n\n Please input a value: "
result: .ascii "\n The hex is: "
buf:    .space 12
        .text
main:   li $v0, 4
        la $a0, prompt
        syscall
        li $v0, 5
        syscall
        move $a0,$v0
        jal Hexout
        li $v0, 4
        la $a0, result
        syscall
        li $v0, 10
        syscall
```

```
Hexout: move $t0, $a0          #将 a0 保存到 t0 中
        la $t1, buf            #缓冲区地址
        li $t2, 8              #循环次数
        addi $t3, $t1, 10      #从位置 buf+10 处开始存放 16 进制数
loop:   blez $t2,end           #判断循环是否结束
        andi $t4, $t0, 0x0f    #取 t0 的低 4 位
        srl $t0, $t0, 4        #t0 右移 4 位
        bge $t4, 10, char      #t4 大于等于 10 跳转到为 A-F 处理
        addi $t4, $t4, 0x30    #0 的 ASCII 码为 0x30，在原先基础上加 0x30
        b put
char:   addi $t4, $t4, 0x37    #A 的 ASCII 码为 65，在原先基础上加(65-10)
put:    sb $t4, ($t3)          #放置字符
        addi $t3, $t3, -1      #放置位置前移一个字符
        addi $t2, $t2, -1      #将循环次数减 1
        b loop
end:    sb $0, 11($t1)         #将 0x00 存储到最后一个位置（位置 11）
        li $t0, 0x78
        sb $t0, 2($t1)         #将 0x78（字符 x）存储到位置 2
        li $t0, 0x30
        sb $t0, 1($t1)         #将 0x30（字符 0）存储到位置 1
        li $t0, 0x20
        sb $t0, ($t1)          #将 0x20（字符空格）存储到位置 0
        jr $ra
```

15. 右对齐打印十进制值

```
            .data
prompt:     .asciiz "\n\n Please input a value: "
result:     .ascii "\n The dec is: "
buf:        .space 13
#1 个空格+1 个可能的符号+10 个字符+1 个表示字符串结束的空字符
            .globl main
            .text
main:       li $v0, 4
            la $a0, prompt
            syscall
            li $v0, 5
            syscall
            move $a0, $v0
            jal Decout
            li $v0, 4
            la $a0, result
            syscall
            li $v0, 10
            syscall
```

```
Decout:
            move    $t0, $a0
            la    $t1, buf
            li    $t2, 11
            sb    $0,12($t1)
            li    $t8,0x20
space:      add    $t9,$t1,$t2
            sb    $t8,($t9)
            addi    $t2,$t2,-1
            bgez    $t2,space
            abs    $t3,$t0
            li    $t2, 11
            li    $t9,10
            add    $t1, $t1, $t2
loop:       div    $t3,$t9
            mflo    $t3
            mfhi    $t4
            addi    $t4,$t4,0x30
            sb    $t4,($t1)
            addi    $t1, $t1, -1
            beqz    $t3,sign
            b    loop
sign:       bgez    $t0,ret
            li    $t8,0x2d
            sb    $t8,($t1)
ret:        jr    $ra
```

16. 读取十六进制值（先看 PPT 算法描述）

```
#去除 0x 之前的空格
while (1) do {if Mem(a0) == 0x20 then a0= a0 + 1;else break; }
#是否以 0x 开头
if (Mem(a0)!=0x30 or Mem(++a0)!=0x78) then return “不以 0x 开头”
#依次读取字符，判断其 ASCII 是否是有效数字（在 0x30-39 或 0x41-46 或 0x61-66 之间）
t0=0; (实际长度) t3=0; (累加值) ;a0++;
while (Mem(a0)!=0x0a) do{
      if (Mem(a0) >=0x30 and Mem(a0)<=0x39) or (Mem(a0) >=0x41 and Mem(a0)<=0x46)
      or (Mem(a0) >=0x61 and Mem(a0)<=0x66) then {
            t4=getNumber(Mem(a0));
            a0=a0+1 ;
            t0=t0+1;  t3=t3<<4+t4};
      else return “无效数字”;}
#判断长度是否超过 8
If t0>8 then return "长度超过 8 "
return
```

```
            .data
prompt:  .asciiz "\n Please input a hex value: "
result:     .ascii "\n The hex value is: "
buf:        .space 20
#正确的格式允许最初有空格，因此要先处理。
#从第一个非空格开始必须为 2 字符” 0x”+1-8 字符(0`9,A`F)+1 个表示回车字符 0a
#错误的格式包括：（1）不以 0x 开头;（2）每个数字 ASCII 不在 0x30-39 或 0x41-46 之间;
（3）ASCII 在 0x30-39 或 0x41-46 之间的数字超过 8 个;
result0:  .asciiz " The dec value is: "
mess0:    .asciiz    " \n Correct input "
mess1:    .asciiz    " \n Error: Not start with 0x "
mess2:    .asciiz    " \n Error: Invalid number "
mess3:    .asciiz    " \n Error: The length is great than 8 "
            .text
main:       li $v0,4
            la $a0, prompt
            syscall
            la $a0,buf              #缓冲区地址
            li $a1,20
            li $v0,8
            syscall
            jal Hexin
            li $v0,10
            syscall
```

```
Hexin:    li $t1,0x20
do_space:                          #处理最初的空格
          lb $t2,($a0)             #依次取字符
          bne $t2,$t1,do_0x        #若不是空格，退出考虑是否以 0x 开头
          addi $a0,$a0,1
          b do_space               (若全部是空格,结果如何?  out_e1)
do_0x:    li $t1,0x30              #处理是否以 0x 开头
          bne $t2,$t1,out_e1       #若不以 0 开头，则退出返回错误消息"不以 0x 开头"
          li $t1,0x78
          addi $a0,$a0,1
          lb $t2,($a0)
          bne $t2,$t1,out_e1       #若不以 0x 开头，则退出返回错误消息"不以 0x 开头"
          li $t0,0                 #$t0 用于统计以 0x 开头的输入字符长度
          li $t3,0                 #$t3 用于存放最终结果
do_valid: addi $a0,$a0,1           #处理以 0x 开头的输入字符是否有效
          lb $t2,($a0)
          li $t1,0x0a
          beq $t2,$t1,do_length    #是否遇到回车字符“0x0a”，若是处理长度是否<=8
          li $t1,0x20
          beq $t2,$t1,do_space2    #是否遇到空格字符“0x20”，若是处理后续合法空格
          li $t1,0x30
          blt $t2,$t1,out_e2       #是否大于等于 0x30("0")
          li $t1,0x39
          ble $t2,$t1,do_dascii    #是否小于等于 0x39("9")，若是处理数字 0-9
          li $t1,0x41
          blt $t2,$t1,out_e2       #是否大于等于 0x41("A")
          li $t1,0x46
          ble $t2,$t1,do_ucascii   #是否小于等于 0x46("F")，若是则处理字母 A-F
          li $t1,0x61
          blt $t2,$t1,out_e2       #是否大于等于 0x61("a")
          li $t1,0x66
          ble $t2,$t1,do_lcascii   #是否小于等于 0x66("f")，若是则处理字母 a-f
          b out_e2                 #否则退出返回错误消息“无效数字”
do_dascii:                         #处理数字 0-9
          li $t1,0x30
          sub $t4,$t2,$t1
          b do_incre
do_ucascii:                        #处理字母 A-F
          li $t1,0x37
          sub $t4,$t2,$t1
          b do_incre
do_lcascii:                        #处理字母 a-f
          li $t1,0x57
          sub $t4,$t2,$t1
```

```
do_incre:                       #循环累加处理
        addi $t0,$t0,1
        sll $t3,$t3,4
        add $t3,$t3,$t4
        b do_valid
do_length:                      #处理以 0x 开头的输入字符长度是否小于等于 8
        li $t1,8
        bgt $t0,$t1,out_e3      #长度大于 8，则退出返回错误消息
        b out_value             #否则输出原十六和十进制值
do_space2:
        addi $a0, $a0, 1            #下一个字符
        lb   $t2, ($a0)
        beqz $t2, do_length         #扫描到末尾，结束扫描
        li $t1,0x0a
        beq $t2, $t1, do_length     #换行结束扫描
        li $t1,0x20
        beq $t2, $t1, do_space2     #空格继续扫描
        b   out_e2                  #空格之后出现其他字符认为非法
out_e1: li $v0, 4
        la $a0,mess1
        syscall
        b out_r
out_e2: li $v0, 4
        la $a0,mess2
        syscall
        b out_r
out_e3: li $v0, 4
        la $a0,mess3
        syscall
        b out_r
out_value:
        li $v0, 4
        la $a0,mess0
        syscall
        li $v0, 4
        la $a0,result
        syscall
        li $v0, 4
        la $a0,result0
        syscall
        move $a0,$t3
        li $v0,1
        syscall
out_r:  jr $ra
```

17. 读取十进制值及查错（先看 PPT 算法描述）

```
          .data
prompt:   .asciiz "\n Please input a dec value: "
result:   .ascii "\n The value is: "
buffer:   .space 20
endl:     .asciiz "\n"
          .text
main:     li $v0,4
          la $a0, prompt
          syscall
          jal Decin              #调用读十进制数函数，返回结果在 v0 中，状态在 v1
          move $t0, $v0          #输出 v0 值
          li $v0,4
          la $a0, result
          syscall
          li $v0,4
          la $a0, endl
          syscall
          move $a0, $t0
          li $v0, 1
          syscall
          li $v0,4
          la $a0, endl
          syscall
          move $a0, $v1          #输出 v1 值
          li $v0, 1
          syscall
          li $v0, 10
          syscall
```

```
Decin:    la $a0, buffer         #缓冲区地址
          li $a1, 20             #字符串长度
          li $v0, 8              #读入字符串，以换行符或者 0 结束
          syscall
          li   $t4, 0x20         #t4=' '
          li   $t5, 0x0a         #t5 为换行符
          li   $t6, 0x2d         #t6='-'
          li   $t1, 0            #t1 为符号，0 为正数，1 为负数
```

```
do_space1:
        lb    $t0, ($a0)
        beqz $t0, out_1              #扫描到末尾，结束扫描
        addi $a0, $a0, 1             #下一个字符
        beq $t0, $t4, do_space1      #空格继续扫描
        beq $t0, $t5, out_1          #换行结束扫描
        bne $t0, $t6, do_valid       #符号处理（若只输入-，结果为 0 且有效？）
        li    $t1, 1                 #t0 为'-'，t1 置 1
        lb    $t0, ($a0)
        addi $a0, $a0, 1             #下一个字符
do_valid:
        li    $t6, 0x30              #t6='0'
        li    $t7, 0x39              #t7='9'
        li    $v0, 0
        li    $t3, 10                #为乘 10 做准备
loop:
        beq $t0, $t4, do_space2      #空格结束扫描
        beqz $t0, out_3              #扫描到末尾，结束扫描
        beq $t0, $t5, out_3          #换行结束扫描
        blt $t0, $t6, out_2          #小于'0'非法
        bgt $t0, $t7, out_2          #大于'9'非法
        mulo $v0, $v0, $t3           #v0*=10
        addi $t0, $t0, -48
        add $v0, $v0, $t0            #v0+=t0-'0'
        lb    $t0, ($a0)
        addi $a0, $a0, 1
        b    loop
do_space2:
        lb    $t0, ($a0)
        addi $a0, $a0, 1             #下一个字符
        beqz $t0, out_3              #扫描到末尾，结束扫描
        beq $t0, $t5, out_3          #换行结束扫描
        beq $t0, $t4, do_space2      #空格继续扫描
        b    out_2                   #出现其他字符认为非法
out_1:  li $v1, 2                    #空串
        li $v0,0
        b    ret
out_2:  li $v1, 3                    #非法字符
        li $v0,0
        b    ret
out_3:  li $v1, 1                    #有效数字
        beqz $t1, ret                #t1 为 0，正数返回
        neg    $v0, $v0
ret:    jr $ra                       #函数返回
```

18. 输出一个字符串中含有元音字符的数目（函数中使用堆栈）

```
        .data
str :   .asciiz " long time ago in a galaxy
far away "
endl:   .asciiz " \n"
        .globl main
        .text
main:
        la $a0,str
        addiu $sp,$sp,-12
        sw $a0, 0($sp)
        sw $ra, 8($sp)
        jal vcount
        lw $ra, 8($sp)
        lw $v0, 4($sp)
        addiu $sp, $sp, 12
        move $a0,$v0
        li $v0,1
        syscall
        la $a0,endl
        li $v0,4
        syscall
        li $v0,10
        syscall
```

```
vcount:
        lw $a0, 0($sp)
        addiu $sp, $sp, -24
        sw $a0, 0($sp)
        sw $s0, 4($sp)
        sw $s1, 8($sp)
        sw $ra, 20($sp)
        li $s0, 0
        move $s1, $a0
nextc:  lb $a0, ($s1)
        beqz $a0, done
        sw $a0, 12($sp)
        jal   vowelp
        lw $v0, 16($sp)
        add $s0, $s0, $v0
        add $s1, $s1, 1
        b nextc
done:   move $v0,$s0
        lw $a0, 0($sp)
        lw $s0, 4($sp)
        lw $s1, 8($sp)
        lw $ra, 20($sp)
        add $sp, $sp, 24
        sw $v0, 4($sp)
        jr $ra

vowelp:
        lw $a0, 12($sp)
        li $v0, 0
        beq $a0, 'a', yes
        beq $a0, 'e', yes
        beq $a0, 'i', yes
        beq $a0, 'o', yes
        beq $a0, 'u', yes
        beq $a0, 'A', yes
        beq $a0, 'E', yes
        beq $a0, 'I', yes
        beq $a0, 'O', yes
        beq $a0, 'U', yes
        j ret
yes:    li $v0, 1
ret:    sw $v0, 16($sp)
        jr $ra
```

19. 读取十六进制值（函数中使用堆栈）（略）

20. fp 使用示例（略，见讲义）

```
        .text
        .globl    main
main:
        addiu $sp, $sp, -4
        sw $ra, ($sp)
        addiu $sp, $sp, -4
        sw $fp, ($sp)
        addiu $fp, $sp, -4
        move $sp, $fp
        li $a0, 6
        jal mysub
        sw $v0, 0($fp)
        lw $a0, 0($fp)
        li $v0, 1
        syscall
        addiu $sp, $fp, 4
        lw $fp, ($sp)
        addiu $sp, $sp, 4
        lw $ra, ($sp)
        addiu $sp, $sp, 4
        jr $ra
```

```
        .text
        .globl    mysub
mysub:
        addiu $sp, $sp, -4
        sw $ra, ($sp)
        addiu $sp, $sp, -4
        sw $fp, ($sp)
        addiu $sp, $sp, -4
        sw $s1, ($sp)
        addiu $fp, $sp, -8
        move $sp, $fp
        sll $s1, $a0, 1
        sw $s1, 0($fp)
        lw $t0, 0($fp)
        addi $t0, $t0, 7
        sw $t0, 4($fp)
        lw $v0, 4($fp)
        addiu $sp, $fp, 8
        lw $s1, ($sp)
        addiu $sp, $sp, 4
        lw $fp, ($sp)
        addiu $sp, $sp, 4
        lw $ra, ($sp)
        addiu $sp, $sp, 4
        jr $ra
```

21. 计算 N！（递归调用）

```
            .data
prompt:     .asciiz     " \n\n Input 'N' :    "
result:     .asciiz     " N factorial is :    "
bye:        .asciiz     " \n ### Bye ### "
msg:        .asciiz     "Number is too big. "
            .text
main:
            # 分配存储空间
            addiu $sp, $sp,   -8
loop:       li $v0,    4
            la $a0, prompt
            syscall
            li $v0, 5                  # 输入值 N
            syscall
            bltz $v0, exit
            li $t0,13
            bge $v0,$t0, enter
            sw $v0, 0($sp)
            jal fac
            li $v0, 4
            la $a0, result
            syscall
            lw $a0, 4($sp)
            li $v0, 1
            syscall
            b loop
enter:      li $v0, 4
            la $a0, msg
            syscall
            b loop
exit:       addiu $sp, $sp, 8
            li $v0, 4
            la $a0, bye
            syscall
            li $v0, 10
            syscall
```

```
fac:
    #获取输入参数
        lw $a0, 0($sp)
        bltz $a0, out_e
    #在栈上为函数分配临时存储空间
        addiu $sp, $sp, -16
    #保存返回地址
        sw $ra, 12 ($sp )
    #保存输入参数
        sw $a0, 8($sp)
    #如果 N 是 0 或 1，返回 1
        slti $t0, $a0, 2
        beqz $t0, go
        li $v0, 1
        b ret
 go:
    #向函数 fac 传递输入参数 N-1
        addi $a0, $a0, -1
        sw $a0, 0($sp)
        jal fac              #递归调用
        lw $v0, 4 ($sp)      #取（N-1)!
    #取返回地址
        lw $ra, 12 ($sp)
    #取回输入参数
        lw $a0, 8 ($sp)
        mult $v0, $a0        # N* (N-l) !
        mflo $v0
ret :
        addiu $sp, $sp, 16  #释放空间
        sw $v0, 4 ($sp)
        jr $ra
out_e:
        sw $0, 4 ($sp)
        jr $ra
```