P2课下

P2_L0_judge

汇编代码：

```
.data
    string: .space 1000



.macro getChar(%dest)
    li $v0,12
    syscall
    move %dest,$v0
.end_macro

.macro printChar(%src)
    move $a0,%src
    li $v0,11
    syscall
.end_macro

.macro getInt(%dest)
    li $v0,5
    syscall
    move %dest,$v0
.end_macro

.macro printInt(%src)
    move $a0,%src
    li $v0,1
    syscall
.end_macro

.macro get_vector_addr(%index, %x)
    sll     %index, %x, 2
.end_macro


```

```
35
36  .text
37      getInt($s0)    # $s0 = n
38
39      ################## read string[n] ###################
40      li $t0,0
41      for_1_begin:
42          bge $t0,$s0,for_1_end
43
44          getChar($t1)
45          get_vector_addr($t2,$t0)
46          sw $t1,string($t2)
47
48
49          addi $t0,$t0,1
50          j for_1_begin
51      for_1_end:
52
53      ################### judge ####################
54      li $s1,0            # fail_flag = 0
55      li $t0,0        # i = 0
56      move $t1,$s0
57      subi $t1,$t1,1      # j = n - 1
58      for_2_begin:
59          bgt $t0,$t1,for_2_end
60
61          get_vector_addr($t2,$t0)
62          lw $t3,string($t2)
63          get_vector_addr($t2,$t1)
64          lw $t4,string($t2)
65
66          beq $t3,$t4,not_fail
67          # fail:
68              li $s1,1
69              j for_2_end
70
71          not_fail:
72
73          addi $t0,$t0,1
74          subi $t1,$t1,1
75          j for_2_begin
76      for_2_end:
77
78      ################### output according to fail_flag ###################
79      li $t0,1
80      beq $s1,$t0,no
81          li $t0,1
82          printInt($t0)
83          j if_end
84      no:
85          li $t0,0
86          printInt($t0)
87          j if_end
88
89
```

```
    if_end:

    # exit
    li $v0, 10
    syscall

```

P2_L0_full_1

C代码：

```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
int used[100],num[100],n;
void solve(int k){
    if(k==n+1){
        for(int i=1;i<=n;i++){
            printf(i==n?"%d\n":"%d ",num[i]);
        }
        return;
    }
    for(int i=1;i<=n;i++){
        if(!used[i]){
            num[k]=i;
            used[i]=1;
            solve(k+1);
            used[i]=0;
        }
    }
}
int main()
{
    scanf("%d",&n);
    solve(1);
    return 0;
}
```

MIPS汇编代码：

```
.data
    rec:    .space 1000
    used:   .space 1000
    space:  .asciiz " "
    enter:  .asciiz "\n"

#push
.macro push(%src)
    sw      %src, 0($sp)
    subi    $sp, $sp, 4
```

```
.end_macro
#pop
.macro pop(%des)
    addi    $sp, $sp, 4
    lw      %des, 0($sp)
.end_macro

.macro printStrOf(%src)
    la $a0, %src
    li $v0, 4
    syscall
.end_macro

.macro getInt(%dest)
    li $v0,5
    syscall
    move %dest,$v0
.end_macro

.macro printInt(%src)
    move $a0,%src
    li $v0,1
    syscall
.end_macro

.macro get_vector_addr(%index, %x)
    sll     %index, %x, 2
.end_macro



.text
    getInt($s0) # $s0 = n

    li $t0,1
    move $a0,$t0
    jal solve

    li $v0,10
    syscall

    solve:
    # 1: push
        push($ra)
        push($t0)
        push($t1)
        push($t2)
        push($t3)

    # 2: move args
        move $t0,$a0
    # 3 body
        ble $t0,$s0,continue

        li $t1,1
```

```
        for_1_begin:
            bgt $t1,$s0,for_1_end

            get_vector_addr($t2,$t1)
            lw $t3,rec($t2)
            printInt($t3)
            printStrOf(space)

            addi $t1,$t1,1
            j for_1_begin
        for_1_end:

        printStrOf(enter)

        pop($t3)
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
        jr $ra

        continue:

        li $t1,1
        for_2_begin:
            bgt $t1,$s0,for_2_end

            get_vector_addr($t2,$t1)
            lw $t3,used($t2)
            bnez $t3,go
                li $t3,1
                sw $t3,used($t2)

                get_vector_addr($t2,$t0)
                sw $t1,rec($t2)

                move $a0,$t0
                addi $a0,$a0,1

                jal solve

                get_vector_addr($t2,$t1)
                li $t3,0
                sw $t3,used($t2)



            go:



            addi $t1,$t1,1
            j for_2_begin
        for_2_end:

```

```
    # 4: pop
        pop($t3)
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
    # 5:
    jr $ra
```

P2_L0_matrix

MIPS汇编代码：

```
.data
    A:      .space 400
    B:      .space 400
    C:      .space 400
    space:  .asciiz " "
    enter:  .asciiz "\n"

.macro getInt(%dest)
    li $v0, 5
    syscall
    move %dest, $v0
.end_macro

.macro printInt(%src)
    move $a0, %src
    li $v0, 1
    syscall
.end_macro

.macro printStrOf(%src)
    la $a0, %src
    li $v0, 4
    syscall
.end_macro

.macro get_matrix_addr(%ans, %row, %col)
    li %ans,10
    multu %row, %ans
    mflo %ans
    addu %ans,%ans,%col
    sll %ans,%ans,2
.end_macro









.text
    getInt($s0) # $s0 = n
```

```
    ################### read A ####################
    li $t0,0
    for_1_i_begin:
        bge $t0,$s0,for_1_i_end

        li $t1,0
        for_1_j_begin:
            bge $t1,$s0,for_1_j_end

            get_matrix_addr($t2,$t0,$t1)
            getInt($t3)
            sw $t3,A($t2)


            addi $t1,$t1,1
            j for_1_j_begin
        for_1_j_end:


        addi $t0,$t0,1
        j for_1_i_begin
    for_1_i_end:

    ################### read B ####################
    li $t0,0
    for_2_i_begin:
        bge $t0,$s0,for_2_i_end

        li $t1,0
        for_2_j_begin:
            bge $t1,$s0,for_2_j_end

            get_matrix_addr($t2,$t0,$t1)
            getInt($t3)
            sw $t3,B($t2)


            addi $t1,$t1,1
            j for_2_j_begin
        for_2_j_end:


        addi $t0,$t0,1
        j for_2_i_begin
    for_2_i_end:

    ################### calculate: A * B ####################
    li $t0,0
    for_3_i_begin:
        bge $t0,$s0,for_3_i_end

        li $t1,0
        for_3_j_begin:
            bge $t1,$s0,for_3_j_end
```

```
95
96              li $t3,0 # C[i][j] = 0
97
98              li $t2,0
99              for_3_k_begin:
100                 bge $t2,$s0,for_3_k_end
101
102                 get_matrix_addr($t4,$t0,$t2) # (i,k)
103                 lw $t5,A($t4)   # $t5 = A[i][k]
104                 get_matrix_addr($t4,$t2,$t1) # (k,j)
105                 lw $t6,B($t4)   # $t6 = B[k][j]
106
107                 multu $t5,$t6
108                 mflo $t5
109                 addu $t3,$t3,$t5
110
111
112                 addi $t2,$t2,1
113                 j for_3_k_begin
114             for_3_k_end:
115
116             printInt($t3)
117             printStrOf(space)
118
119
120             addi $t1,$t1,1
121             j for_3_j_begin
122         for_3_j_end:
123
124         printStrOf(enter)
125
126         addi $t0,$t0,1
127         j for_3_i_begin
128     for_3_i_end:
129
130
131     # exit
132     li $v0, 10
133     syscall
134
```

P2_L0_conv

C代码：

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  #include<ctype.h>
5  #include<string.h>
6  #define max(a,b) (((a)<(b))?(b):(a))
7  #define min(a,b) (((a)<(b))?(a):(b))
8  #define LL long long
```

```c

int main(){
    int row1,row2,col1,col2;
    int M[20][20]={0};
    int core[20][20]={0};
    scanf("%d%d%d%d",&row1,&col1,&row2,&col2);

    for(int i =1; i<= row1 ; i++){
        for(int  j = 1;j<= col1;j++){
            scanf("%d",&M[i][j]);
        }
    }

    for(int i=1;i<=row2;i++){
        for(int j=1;j<= col2;j++){
            scanf("%d",&core[i][j]);
        }
    }

    for(int i = 1;i<=row1-row2+1;i++){
        for(int j=1;j<=col1-col2+1;j++){
            int tmp = 0;
            for(int k=1;k<=row2;k++){
                for(int l=1;l<=col2;l++){
                    tmp+=M[i+k-1][j+l-1]*core[k][l];
                }
            }
            printf("%d ",tmp);

        }
        printf("\n");
    }

    return 0;
}
```

MIPS汇编代码:

```assembly
.data
    M:      .space 1000
    core:   .space 400
    space:  .asciiz " "
    enter:  .asciiz "\n"

.macro getInt(%dest)
    li $v0, 5
    syscall
    move %dest, $v0
.end_macro

.macro printInt(%src)
    move $a0, %src
    li $v0, 1
    syscall
.end_macro
```

```
18
19  .macro printStrOf(%src)
20      la $a0, %src
21      li $v0, 4
22      syscall
23  .end_macro
24
25  .macro get_matrix_addr(%ans, %row, %col)
26      li %ans,13
27      multu %row, %ans
28      mflo %ans
29      addu %ans,%ans,%col
30      sll %ans,%ans,2
31  .end_macro
32
33
34
35  .text
36      getInt($s0)
37      getInt($s1)
38      getInt($s2)
39      getInt($s3)
40      ################# read M #################
41      li $t0,1
42      for_1_i_begin:
43          bgt $t0,$s0,for_1_i_end
44
45          li $t1,1
46          for_1_j_begin:
47              bgt $t1,$s1,for_1_j_end
48
49              get_matrix_addr($t2,$t0,$t1)
50              getInt($t3)
51              sw $t3,M($t2)
52
53              addi $t1,$t1,1
54              j for_1_j_begin
55          for_1_j_end:
56
57
58
59          addi $t0,$t0,1
60          j for_1_i_begin
61      for_1_i_end:
62
63      ################# read core #################
64      li $t0,1
65      for_2_i_begin:
66          bgt $t0,$s2,for_2_i_end
67
68          li $t1,1
69          for_2_j_begin:
70              bgt $t1,$s3,for_2_j_end
71
72              get_matrix_addr($t2,$t0,$t1)
```

```
            getInt($t3)
            sw $t3,core($t2)

            addi $t1,$t1,1
            j for_2_j_begin
        for_2_j_end:



        addi $t0,$t0,1
        j for_2_i_begin
    for_2_i_end:

    ################ calculate #################
    subu $s6,$s0,$s2
    addi $s6,$s6,1

    subu $s7,$s1,$s3
    addi $s7,$s7,1

    li $t0,1
    for_3_i_begin:
        bgt $t0,$s6,for_3_i_end

        li $t1,1
        for_3_j_begin:
            bgt $t1,$s7,for_3_j_end

            li $s5,0

            li $t2,1
            for_3_k_begin:
                bgt $t2,$s2,for_3_k_end

                li $t3,1
                for_3_l_begin:
                    bgt $t3,$s3,for_3_l_end

                    addu $t4,$t0,$t2
                    subi $t4,$t4,1
                    addu $t5,$t1,$t3
                    subi $t5,$t5,1
                    get_matrix_addr($t6,$t4,$t5)
                    lw $t7,M($t6)

                    get_matrix_addr($t6,$t2,$t3)
                    lw $t8,core($t6)

                    multu $t7,$t8
                    mflo $t7

                    addu $s5,$s5,$t7


                    addi $t3,$t3,1
```

```
                    j for_3_l_begin
                for_3_l_end:


                addi $t2,$t2,1
                j for_3_k_begin
            for_3_k_end:

            printInt($s5)
            printStrOf(space)


            addi $t1,$t1,1
            j for_3_j_begin
        for_3_j_end:

        printStrOf(enter)

        addi $t0,$t0,1
        j for_3_i_begin
    for_3_i_end:
    # exit
    li $v0,10
    syscall

```

P2_L1_puzzle

C代码:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
#define max(a,b) (((a)<(b))?(b):(a))
#define min(a,b) (((a)<(b))?(a):(b))
#define LL long long

int map[10][10];
int used[10][10];
int cnt;
int target_x,target_y;
int src_x,src_y;
int n,m;

void dfs(int x,int y){
    if(x <= 0 || x > n || y <= 0 || y > m){
        return;
    }
    if(x == target_x && y == target_y){
        cnt++;
        return;
```

```
    }
    used[x][y]=1;

    if(!used[x+1][y] && !map[x+1][y])dfs(x+1,y);
    if(!used[x-1][y] && !map[x-1][y])dfs(x-1,y);
    if(!used[x][y+1] && !map[x][y+1])dfs(x,y+1);
    if(!used[x][y-1] && !map[x][y-1])dfs(x,y-1);

    used[x][y]=0;
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            scanf("%d",&map[i][j]);
        }
    }
    scanf("%d%d%d%d",&src_x,&src_y,&target_x,&target_y);
    dfs(src_x,src_y);
    printf("%d\n",cnt);

    return 0;
}
```

MIPS汇编代码：

```
.data
    map:    .space 4000
    used:   .space 4000
    space:  .asciiz " "
    enter:  .asciiz "\n"
    mark1:  .asciiz "fail"
    mark2:  .asciiz "succeed"

.macro getInt(%dest)
    li $v0, 5
    syscall
    move %dest, $v0
.end_macro

.macro printInt(%src)
    move $a0, %src
    li $v0, 1
    syscall
.end_macro

.macro printStrOf(%src)
    la $a0, %src
    li $v0, 4
    syscall
.end_macro

.macro get_matrix_addr(%ans, %row, %col)
    li %ans,10
    multu %row, %ans
```

```
    mflo %ans
    addu %ans,%ans,%col
    sll %ans,%ans,2
.end_macro

#push
.macro push(%src)
    sw      %src, 0($sp)
    subi    $sp, $sp, 4
.end_macro
#pop
.macro pop(%des)
    addi    $sp, $sp, 4
    lw      %des, 0($sp)
.end_macro




.text
    getInt($s0) #n
    getInt($s1) #m

    li $s7,0 # ans = 0

    li $t0,1
    for_i_begin:
        bgt $t0,$s0,for_i_end

        li $t1,1
        for_j_begin:
            bgt $t1,$s1,for_j_end

            get_matrix_addr($t2,$t0,$t1)
            getInt($t3)
            sw $t3,map($t2)


            addi $t1,$t1,1
            j for_j_begin
        for_j_end:


        addi $t0,$t0,1
        j for_i_begin
    for_i_end:


    getInt($s2) #src_x
    getInt($s3) #src_y
    getInt($s4) #target_x
    getInt($s5) #target_y

    move $a0,$s2
    move $a1,$s3
```

```
    jal dfs

    printInt($s7)

    li $v0,10
    syscall

    dfs:
    # 1: push
        push($ra)
        push($t0)
        push($t1)
        push($t2)
        push($t3)
        push($t4)
        push($t5)

    # 2: move args
        move $t0,$a0
        move $t1,$a1
    # 3: function body
        #printStrOf(space)
        #printInt($t0)
        #printInt($t1)
        ble $t0,$zero,if_1
        ble $t1,$zero,if_1
        bgt $t0,$s0,if_1
        bgt $t1,$s1,if_1

        j if_1_end

        if_1:
            #printInt($t0)
            #printInt($t1)
            #printStrOf(mark1)
            #printStrOf(enter)
        pop($t5)
        pop($t4)
        pop($t3)
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
            jr $ra

        if_1_end:

        bne $t0,$s4,if_2_end
        bne $t1,$s5,if_2_end

        if_2:
            addi $s7,$s7,1
            #printInt($t0)
            #printStrOf(space)
            #printInt($t1)
```

```
            #printStrOf(mark2)
            #printStrOf(enter)
                    pop($t5)
        pop($t4)
        pop($t3)
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
            jr $ra

        if_2_end:

        ############### used[x][y]=1; ###############
        get_matrix_addr($t2,$t0,$t1)
        li $t3,1
        sw $t3,used($t2)

        ######## [x+1][y] #######
        addi $t4,$t0,1
        move $t5,$t1
        get_matrix_addr($t2,$t4,$t5)
        lw $t3,used($t2)
        bnez $t3 judge_1_end
        lw $t3,map($t2)
        bnez $t3 judge_1_end

        move $a0,$t4
        move $a1,$t5
        jal dfs



        judge_1_end:
        ######## [x-1][y] #######
        subi $t4,$t0,1
        move $t5,$t1
        get_matrix_addr($t2,$t4,$t5)
        lw $t3,used($t2)
        bnez $t3 judge_2_end
        lw $t3,map($t2)
        bnez $t3 judge_2_end

        move $a0,$t4
        move $a1,$t5
        jal dfs



        judge_2_end:
        ######## [x][y+1] #######
        move $t4,$t0
        addi $t5,$t1,1
        get_matrix_addr($t2,$t4,$t5)
        lw $t3,used($t2)
```

```
        bnez $t3 judge_3_end
        lw $t3,map($t2)
        bnez $t3 judge_3_end

        move $a0,$t4
        move $a1,$t5
        jal dfs



        judge_3_end:
        ######## [x][y-1] #######
        move $t4,$t0
        subi $t5,$t1,1
        get_matrix_addr($t2,$t4,$t5)
        lw $t3,used($t2)
        bnez $t3 judge_4_end
        lw $t3,map($t2)
        bnez $t3 judge_4_end

        move $a0,$t4
        move $a1,$t5
        jal dfs



        judge_4_end:
        ############### used[x][y]=0; ###############
        get_matrix_addr($t2,$t0,$t1)
        li $t3,0
        sw $t3,used($t2)
    # 4: pop
        pop($t5)
        pop($t4)
        pop($t3)
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
    # 5: return
        jr $ra

```

P2_L1_factorial

C代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
#define max(a,b) (((a)<(b))?(b):(a))
```

```c
#define min(a,b) (((a)<(b))?(a):(b))
#define LL long long

int a[1001],h=1,n;



int main(){
    a[1]=1;
    scanf("%d",&n);
    for(int i=2;i<=n;i++){
        for(int j=1;j<=h;j++){
            a[j]=a[j]*i;
        }
        for(int j=1;j<=h;j++){
            if(a[j]>9){
                a[j+1]=a[j+1]+a[j]/10;
                a[j]=a[j]%10;
            }
        }
        while(a[h+1]>0){
            h++;
            a[h+1]=a[h]/10;
            a[h]=a[h]%10;
        }
    }
    int flag=0;
    for(int i=1000;i>=1;i--){
        if(a[i]!=0)flag=1;
        if(flag)printf("%d",a[i]);
    }

    return 0;
}
```

C代码：by GPT

```c
#include <stdio.h>

#define MAX_DIGITS 10000

int main() {
    int n;
    printf("Enter a number to calculate its factorial: ");
    scanf("%d", &n);

    if (n < 0) {
        printf("Factorial is not defined for negative numbers.\n");
        return 1; // 返回错误码
    }

    int result[MAX_DIGITS] = {0};
    result[0] = 1;
    int length = 1;

```

```
    for (int i = 2; i <= n; i++) {
        int carry = 0;
        for (int j = 0; j < length; j++) {
            int product = result[j] * i + carry;
            result[j] = product % 10;
            carry = product / 10;
        }
        while (carry > 0) {
            result[length] = carry % 10;
            carry /= 10;
            length++;
        }
    }

    printf("Factorial of %d is: ", n);
    for (int i = length - 1; i >= 0; i--) {
        printf("%d", result[i]);
    }
    printf("\n");

    return 0;
}
```

MIPS汇编代码：

```
.data
    res: .space 8000



.macro getInt(%dest)
    li $v0,5
    syscall
    move %dest,$v0
.end_macro

.macro printInt(%src)
    move $a0,%src
    li $v0,1
    syscall
.end_macro

.macro get_vector_addr(%index, %x)
    sll     %index, %x, 2
.end_macro


.text
    getInt($s0) #n

    li $s1,1        #len

    li $t1,0
    get_vector_addr($t2,$t1)
    sw $s1,res($t2)
```

```
31
32
33      li $t0,2
34      for_1_i_begin:
35          bgt $t0,$s0,for_1_i_end
36
37
38          li $t3,0    # carry
39
40
41          li $t1,0
42          for_1_j_begin:
43              bge $t1,$s1,for_1_j_end
44
45              get_vector_addr($t2,$t1)
46              lw $t4,res($t2)
47              multu $t4,$t0
48              mflo $t4
49              addu $t4,$t4,$t3        # $t4 = product
50
51              li $t5,10
52              div $t4,$t5
53
54              mfhi $t6        # $t6 = remainer
55              mflo $t7        # $t6 = group  ->  carry
56
57              sw $t6,res($t2)
58
59              move $t3,$t7
60
61              addi $t1,$t1,1
62              j for_1_j_begin
63          for_1_j_end:
64
65          while:
66              ble $t3,$zero,while_end
67
68              get_vector_addr($t2,$s1)
69              li $t5,10
70              div $t3,$t5
71
72              mfhi $t6
73              mflo $t7
74
75              sw $t6,res($t2)
76              move $t3,$t7
77
78              addi $s1,$s1,1
79
80              j while
81          while_end:
82
83
84          addi $t0,$t0,1
85          j for_1_i_begin
```

```
    for_1_i_end:


    move $t0,$s1
    subi $t0,$t0,1
    for_2_begin:
        blt $t0,$zero,for_2_end

        get_vector_addr($t2,$t0)
        lw $t3,res($t2)

        printInt($t3)

        subi $t0,$t0,1
        j for_2_begin
    for_2_end:


    li $v0,10
    syscall
```

P2_L1_calculate(推荐题)

MIPS汇编代码：

```
.data
    letter: .space 1000
    cnt: .space 1000
    space: .asciiz " "
    enter: .asciiz "\n"



.macro getChar(%dest)
    li $v0,12
    syscall
    move %dest,$v0
.end_macro

.macro printChar(%src)
    move $a0,%src
    li $v0,11
    syscall
.end_macro

.macro getInt(%dest)
    li $v0,5
    syscall
    move %dest,$v0
.end_macro

.macro printInt(%src)
    move $a0,%src
```

```
    li $v0,1
    syscall
.end_macro

.macro printStrOf(%src)
    la $a0,%src
    li $v0,4
    syscall
.end_macro

.macro get_vector_addr(%index, %x)
    sll     %index, %x, 2
.end_macro

.text
    getInt($s0)

    li $s2,10
    li $s3,0

    li $t0,0
    for_1_begin:
        bge $t0,$s0,for_1_end

        getChar($t1)


        li $s4,0 #flag
        li $t2,0
        for_j_begin:
            bge $t2,$s3,for_j_end

            # if letter[j]==char
            get_vector_addr($t3,$t2)
            lw $t4,letter($t3)

            bne $t4,$t1,else
                lw $t5,cnt($t3)
                addi $t5,$t5,1
                sw $t5,cnt($t3)
                li $s4,1
                j for_j_end
            else:

            addi $t2,$t2,1
            j for_j_begin
        for_j_end:

        bnez $s4,flag_is_1
            get_vector_addr($t3,$s3)
            sw $t1,letter($t3)
            li $t1,1
            sw $t1,cnt($t3)

            addi $s3,$s3,1
```

```
        flag_is_1:


        addi $t0,$t0,1
        j for_1_begin
    for_1_end:

    li $t0,0
    for_2_begin:
        bge $t0,$s3,for_2_end

        get_vector_addr($t1,$t0)

        lw $t2,letter($t1)
        printChar($t2)
        printStrOf(space)
        lw $t2,cnt($t1)
        printInt($t2)
        printStrOf(enter)

        addi $t0,$t0,1
        j for_2_begin
    for_2_end:

    li $v0,10
    syscall


```

P2_L1_honoi(推荐题)

MIPS汇编代码:

```
.data
    mark1: .asciiz "move disk "
    mark2: .asciiz " from "
    mark3: .asciiz " to "
    enter: .asciiz "\n"

.macro getChar(%dest)
    li $v0,12
    syscall
    move %dest,$v0
.end_macro

.macro printStrOf(%src)
    la $a0, %src
    li $v0, 4
    syscall
.end_macro

.macro printChar(%src)
    move $a0,%src
    li $v0,11
```

```
    syscall
.end_macro

.macro getInt(%dest)
    li $v0,5
    syscall
    move %dest,$v0
.end_macro

.macro printInt(%src)
    move $a0,%src
    li $v0,1
    syscall
.end_macro

.macro get_vector_addr(%index, %x)
    sll     %index, %x, 2
.end_macro

#push
.macro push(%src)
    sw      %src, 0($sp)
    subi    $sp, $sp, 4
.end_macro
#pop
.macro pop(%des)
    addi    $sp, $sp, 4
    lw      %des, 0($sp)
.end_macro


.text
    getInt($s0) #n
    move $a0,$s0
    li $t1,65
    move $a1,$t1
    li $t1,66
    move $a2,$t1
    li $t1,67
    move $a3,$t1

    jal hanoi


    li $v0,10
    syscall


    hanoi:
    #1: push
        push($ra)
        push($t0)
        push($t1)
        push($t2)
        push($t3)
```

```
77
78
79      #2: move args
80          move $t0,$a0
81          move $t1,$a1
82          move $t2,$a2
83          move $t3,$a3
84      #3: body
85          bnez $t0,if_end
86
87              move $a0,$t0
88              move $a1,$t1
89              move $a2,$t2
90              jal move_
91
92              move $a0,$t0
93              move $a1,$t2
94              move $a2,$t3
95              jal move_
96
97              j pop_and_return
98
99
100         if_end:
101
102             move $a0,$t0
103             subi $a0,$a0,1
104             move $a1,$t1
105             move $a2,$t2
106             move $a3,$t3
107             jal hanoi
108
109             move $a0,$t0
110             move $a1,$t1
111             move $a2,$t2
112             jal move_
113
114             move $a0,$t0
115             subi $a0,$a0,1
116             move $a1,$t3
117             move $a2,$t2
118             move $a3,$t1
119             jal hanoi
120
121             move $a0,$t0
122             move $a1,$t2
123             move $a2,$t3
124             jal move_
125
126             move $a0,$t0
127             subi $a0,$a0,1
128             move $a1,$t1
129             move $a2,$t2
130             move $a3,$t3
131             jal hanoi
```

```
    pop_and_return:
    #4: pop
        pop($t3)
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
    #5: return
        jr $ra



    move_:
    #1: push
        push($ra)
        push($t0)
        push($t1)
        push($t2)
    #2: move args
        move $t0,$a0
        move $t1,$a1
        move $t2,$a2
    #3: body
        printStrOf(mark1)
        printInt($t0)
        printStrOf(mark2)
        printChar($t1)
        printStrOf(mark3)
        printChar($t2)
        printStrOf(enter)

    #4: pop
        pop($t2)
        pop($t1)
        pop($t0)
        pop($ra)
    #5: return
        jr $ra

```