

P2课下

P2课下

[P2_L0_judge](#)
[P2_L0_full_1](#)
[P2_L0_matrix](#)
[P2_L0_conv](#)
[P2_L1_puzzle](#)
[P2_L1_factorial](#)
[P2_L1_calculate\(推荐题\)](#)
[P2_L1_hanoi\(推荐题\)](#)

P2_L0_judge

汇编代码:

```
1  .data
2      string: .space 1000
3
4
5
6  .macro getChar(%dest)
7      li $v0,12
8      syscall
9      move %dest,$v0
10 .end_macro
11
12 .macro printChar(%src)
13     move $a0,%src
14     li $v0,11
15     syscall
16 .end_macro
17
18 .macro getInt(%dest)
19     li $v0,5
20     syscall
21     move %dest,$v0
22 .end_macro
23
24 .macro printInt(%src)
25     move $a0,%src
26     li $v0,1
27     syscall
28 .end_macro
29
30 .macro get_vector_addr(%index, %x)
31     sll     %index, %x, 2
32 .end_macro
33
34
```

```

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

```

```

90     if_end:
91
92     # exit
93     li $v0, 10
94     syscall
95

```

P2_L0_full_1

C代码:

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  #include<ctype.h>
5  #include<string.h>
6  int used[100],num[100],n;
7  void solve(int k){
8      if(k==n+1){
9          for(int i=1;i<=n;i++){
10             printf(i==n?"%d\n":"%d ",num[i]);
11         }
12         return;
13     }
14     for(int i=1;i<=n;i++){
15         if(!used[i]){
16             num[k]=i;
17             used[i]=1;
18             solve(k+1);
19             used[i]=0;
20         }
21     }
22 }
23 int main()
24 {
25     scanf("%d",&n);
26     solve(1);
27     return 0;
28 }

```

MIPS汇编代码:

```

1  .data
2      rec:      .space 1000
3      used:     .space 1000
4      space:    .asciiz " "
5      enter:    .asciiz "\n"
6
7  #push
8  .macro push(%src)
9      sw        %src, 0($sp)
10     subi      $sp, $sp, 4

```

```

11 .end_macro
12 #pop
13 .macro pop(%des)
14     addi    $sp, $sp, 4
15     lw      %des, 0($sp)
16 .end_macro
17
18 .macro printStrOf(%src)
19     la $a0, %src
20     li $v0, 4
21     syscall
22 .end_macro
23
24 .macro getInt(%dest)
25     li $v0, 5
26     syscall
27     move %dest, $v0
28 .end_macro
29
30 .macro printInt(%src)
31     move $a0, %src
32     li $v0, 1
33     syscall
34 .end_macro
35
36 .macro get_vector_addr(%index, %x)
37     sll    %index, %x, 2
38 .end_macro
39
40
41
42 .text
43     getInt($s0) # $s0 = n
44
45     li $t0, 1
46     move $a0, $t0
47     jal solve
48
49     li $v0, 10
50     syscall
51
52     solve:
53     # 1: push
54         push($ra)
55         push($t0)
56         push($t1)
57         push($t2)
58         push($t3)
59
60     # 2: move args
61         move $t0, $a0
62     # 3 body
63         ble $t0, $s0, continue
64
65         li $t1, 1

```

```

66     for_1_begin:
67         bgt $t1,$s0,for_1_end
68
69         get_vector_addr($t2,$t1)
70         lw $t3,rec($t2)
71         printInt($t3)
72         printStrOf(space)
73
74         addi $t1,$t1,1
75         j for_1_begin
76     for_1_end:
77
78     printStrOf(enter)
79
80     pop($t3)
81     pop($t2)
82     pop($t1)
83     pop($t0)
84     pop($ra)
85     jr $ra
86
87     continue:
88
89     li $t1,1
90     for_2_begin:
91         bgt $t1,$s0,for_2_end
92
93         get_vector_addr($t2,$t1)
94         lw $t3,used($t2)
95         bnez $t3,go
96         li $t3,1
97         sw $t3,used($t2)
98
99         get_vector_addr($t2,$t0)
100        sw $t1,rec($t2)
101
102        move $a0,$t0
103        addi $a0,$a0,1
104
105        jal solve
106
107        get_vector_addr($t2,$t1)
108        li $t3,0
109        sw $t3,used($t2)
110
111
112
113        go:
114
115
116
117        addi $t1,$t1,1
118        j for_2_begin
119    for_2_end:
120

```

```

121
122     # 4: pop
123     pop($t3)
124     pop($t2)
125     pop($t1)
126     pop($t0)
127     pop($ra)
128     # 5:
129     jr $ra

```

P2_L0_matrix

MIPS汇编代码：

```

1  .data
2      A:      .space 400
3      B:      .space 400
4      C:      .space 400
5      space:  .asciiz " "
6      enter:  .asciiz "\n"
7
8  .macro getInt(%dest)
9      li $v0, 5
10     syscall
11     move %dest, $v0
12 .end_macro
13
14 .macro printInt(%src)
15     move $a0, %src
16     li $v0, 1
17     syscall
18 .end_macro
19
20 .macro printStrOf(%src)
21     la $a0, %src
22     li $v0, 4
23     syscall
24 .end_macro
25
26 .macro get_matrix_addr(%ans, %row, %col)
27     li %ans, 10
28     multu %row, %ans
29     mflo %ans
30     addu %ans, %ans, %col
31     sll %ans, %ans, 2
32 .end_macro
33
34
35
36
37
38 .text
39     getInt($s0) # $s0 = n

```

```

40
41 ##### read A #####
42 li $t0,0
43 for_1_i_begin:
44     bge $t0,$s0,for_1_i_end
45
46     li $t1,0
47     for_1_j_begin:
48         bge $t1,$s0,for_1_j_end
49
50         get_matrix_addr($t2,$t0,$t1)
51         getInt($t3)
52         sw $t3,A($t2)
53
54
55         addi $t1,$t1,1
56         j for_1_j_begin
57     for_1_j_end:
58
59
60     addi $t0,$t0,1
61     j for_1_i_begin
62 for_1_i_end:
63
64 ##### read B #####
65 li $t0,0
66 for_2_i_begin:
67     bge $t0,$s0,for_2_i_end
68
69     li $t1,0
70     for_2_j_begin:
71         bge $t1,$s0,for_2_j_end
72
73         get_matrix_addr($t2,$t0,$t1)
74         getInt($t3)
75         sw $t3,B($t2)
76
77
78         addi $t1,$t1,1
79         j for_2_j_begin
80     for_2_j_end:
81
82
83     addi $t0,$t0,1
84     j for_2_i_begin
85 for_2_i_end:
86
87 ##### calculate: A * B #####
88 li $t0,0
89 for_3_i_begin:
90     bge $t0,$s0,for_3_i_end
91
92     li $t1,0
93     for_3_j_begin:
94         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

```



```

9
10 int main(){
11     int row1,row2,col1,col2;
12     int M[20][20]={0};
13     int core[20][20]={0};
14     scanf("%d%d%d%d",&row1,&col1,&row2,&col2);
15
16     for(int i =1; i<= row1 ; i++){
17         for(int j = 1;j<= col1;j++){
18             scanf("%d",&M[i][j]);
19         }
20     }
21
22     for(int i=1;i<=row2;i++){
23         for(int j=1;j<= col2;j++){
24             scanf("%d",&core[i][j]);
25         }
26     }
27
28     for(int i = 1;i<=row1-row2+1;i++){
29         for(int j=1;j<=col1-col2+1;j++){
30             int tmp = 0;
31             for(int k=1;k<=row2;k++){
32                 for(int l=1;l<=col2;l++){
33                     tmp+=M[i+k-1][j+l-1]*core[k][l];
34                 }
35             }
36             printf("%d ",tmp);
37
38         }
39         printf("\n");
40     }
41
42     return 0;
43 }

```

MIPS汇编代码:

```

1 .data
2     M:      .space 1000
3     core:   .space 400
4     space:  .asciiz " "
5     enter:  .asciiz "\n"
6
7 .macro getInt(%dest)
8     li $v0, 5
9     syscall
10    move %dest, $v0
11 .end_macro
12
13 .macro printInt(%src)
14    move $a0, %src
15    li $v0, 1
16    syscall
17 .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)

```

```

73         getInt($t3)
74         sw $t3,core($t2)
75
76         addi $t1,$t1,1
77         j for_2_j_begin
78     for_2_j_end:
79
80
81
82         addi $t0,$t0,1
83         j for_2_i_begin
84     for_2_i_end:
85
86     ##### calculate #####
87     subu $s6,$s0,$s2
88     addi $s6,$s6,1
89
90     subu $s7,$s1,$s3
91     addi $s7,$s7,1
92
93     li $t0,1
94     for_3_i_begin:
95         bgt $t0,$s6,for_3_i_end
96
97         li $t1,1
98         for_3_j_begin:
99             bgt $t1,$s7,for_3_j_end
100
101             li $s5,0
102
103             li $t2,1
104             for_3_k_begin:
105                 bgt $t2,$s2,for_3_k_end
106
107                 li $t3,1
108                 for_3_l_begin:
109                     bgt $t3,$s3,for_3_l_end
110
111                     addu $t4,$t0,$t2
112                     subi $t4,$t4,1
113                     addu $t5,$t1,$t3
114                     subi $t5,$t5,1
115                     get_matrix_addr($t6,$t4,$t5)
116                     lw $t7,M($t6)
117
118                     get_matrix_addr($t6,$t2,$t3)
119                     lw $t8,core($t6)
120
121                     multu $t7,$t8
122                     mflo $t7
123
124                     addu $s5,$s5,$t7
125
126
127                     addi $t3,$t3,1

```

```

128             j for_3_l_begin
129             for_3_l_end:
130
131
132             addi $t2,$t2,1
133             j for_3_k_begin
134             for_3_k_end:
135
136             printInt($s5)
137             printStrOf(space)
138
139
140             addi $t1,$t1,1
141             j for_3_j_begin
142             for_3_j_end:
143
144             printStrOf(enter)
145
146             addi $t0,$t0,1
147             j for_3_i_begin
148             for_3_i_end:
149             # exit
150             li $v0,10
151             syscall
152

```

P2_L1_puzzle

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
9
10 int map[10][10];
11 int used[10][10];
12 int cnt;
13 int target_x,target_y;
14 int src_x,src_y;
15 int n,m;
16
17 void dfs(int x,int y){
18     if(x <= 0 || x > n || y <= 0 || y > m){
19         return;
20     }
21     if(x == target_x && y == target_y){
22         cnt++;
23         return;

```

```

24     }
25     used[x][y]=1;
26
27     if(!used[x+1][y] && !map[x+1][y])dfs(x+1,y);
28     if(!used[x-1][y] && !map[x-1][y])dfs(x-1,y);
29     if(!used[x][y+1] && !map[x][y+1])dfs(x,y+1);
30     if(!used[x][y-1] && !map[x][y-1])dfs(x,y-1);
31
32     used[x][y]=0;
33 }
34 int main(){
35     scanf("%d%d",&n,&m);
36     for(int i=1;i<=n;i++){
37         for(int j=1;j<=m;j++){
38             scanf("%d",&map[i][j]);
39         }
40     }
41     scanf("%d%d%d%d",&src_x,&src_y,&target_x,&target_y);
42     dfs(src_x,src_y);
43     printf("%d\n",cnt);
44
45     return 0;
46 }

```

MIPS汇编代码:

```

1  .data
2      map:      .space 4000
3      used:     .space 4000
4      space:    .asciiz " "
5      enter:    .asciiz "\n"
6      mark1:    .asciiz "fail"
7      mark2:    .asciiz "succeed"
8
9  .macro getInt(%dest)
10     li $v0, 5
11     syscall
12     move %dest, $v0
13 .end_macro
14
15 .macro printInt(%src)
16     move $a0, %src
17     li $v0, 1
18     syscall
19 .end_macro
20
21 .macro printStrOf(%src)
22     la $a0, %src
23     li $v0, 4
24     syscall
25 .end_macro
26
27 .macro get_matrix_addr(%ans, %row, %col)
28     li %ans,10
29     multu %row, %ans

```

```

30     mflo %ans
31     addu %ans,%ans,%col
32     sll %ans,%ans,2
33 .end_macro
34
35 #push
36 .macro push(%src)
37     sw     %src, 0($sp)
38     subi   $sp, $sp, 4
39 .end_macro
40 #pop
41 .macro pop(%des)
42     addi   $sp, $sp, 4
43     lw     %des, 0($sp)
44 .end_macro
45
46
47
48
49 .text
50     getInt($s0) #n
51     getInt($s1) #m
52
53     li $s7,0 # ans = 0
54
55     li $t0,1
56     for_i_begin:
57         bgt $t0,$s0,for_i_end
58
59         li $t1,1
60         for_j_begin:
61             bgt $t1,$s1,for_j_end
62
63             get_matrix_addr($t2,$t0,$t1)
64             getInt($t3)
65             sw $t3,map($t2)
66
67
68             addi $t1,$t1,1
69             j for_j_begin
70         for_j_end:
71
72
73         addi $t0,$t0,1
74         j for_i_begin
75     for_i_end:
76
77
78     getInt($s2) #src_x
79     getInt($s3) #src_y
80     getInt($s4) #target_x
81     getInt($s5) #target_y
82
83     move $a0,$s2
84     move $a1,$s3

```

```

85     jal dfs
86
87     printInt($s7)
88
89     li $v0,10
90     syscall
91
92     dfs:
93     # 1: push
94         push($ra)
95         push($t0)
96         push($t1)
97         push($t2)
98         push($t3)
99         push($t4)
100        push($t5)
101
102    # 2: move args
103        move $t0,$a0
104        move $t1,$a1
105    # 3: function body
106        #printStrOf(space)
107        #printInt($t0)
108        #printInt($t1)
109        ble $t0,$zero,if_1
110        ble $t1,$zero,if_1
111        bgt $t0,$s0,if_1
112        bgt $t1,$s1,if_1
113
114        j if_1_end
115
116    if_1:
117        #printInt($t0)
118        #printInt($t1)
119        #printStrOf(mark1)
120        #printStrOf(enter)
121        pop($t5)
122        pop($t4)
123        pop($t3)
124        pop($t2)
125        pop($t1)
126        pop($t0)
127        pop($ra)
128        jr $ra
129
130    if_1_end:
131
132    bne $t0,$s4,if_2_end
133    bne $t1,$s5,if_2_end
134
135    if_2:
136        addi $s7,$s7,1
137        #printInt($t0)
138        #printStrOf(space)
139        #printInt($t1)

```

```

140         #printStrOf(mark2)
141         #printStrOf(enter)
142         pop($t5)
143     pop($t4)
144     pop($t3)
145     pop($t2)
146     pop($t1)
147     pop($t0)
148     pop($ra)
149     jr $ra
150
151     if_2_end:
152
153     ##### used[x][y]=1; #####
154     get_matrix_addr($t2,$t0,$t1)
155     li $t3,1
156     sw $t3,used($t2)
157
158     ##### [x+1][y] #####
159     addi $t4,$t0,1
160     move $t5,$t1
161     get_matrix_addr($t2,$t4,$t5)
162     lw $t3,used($t2)
163     bnez $t3 judge_1_end
164     lw $t3,map($t2)
165     bnez $t3 judge_1_end
166
167     move $a0,$t4
168     move $a1,$t5
169     jal dfs
170
171
172
173     judge_1_end:
174     ##### [x-1][y] #####
175     subi $t4,$t0,1
176     move $t5,$t1
177     get_matrix_addr($t2,$t4,$t5)
178     lw $t3,used($t2)
179     bnez $t3 judge_2_end
180     lw $t3,map($t2)
181     bnez $t3 judge_2_end
182
183     move $a0,$t4
184     move $a1,$t5
185     jal dfs
186
187
188
189     judge_2_end:
190     ##### [x][y+1] #####
191     move $t4,$t0
192     addi $t5,$t1,1
193     get_matrix_addr($t2,$t4,$t5)
194     lw $t3,used($t2)

```



```

195     bnez $t3 judge_3_end
196     lw $t3,map($t2)
197     bnez $t3 judge_3_end
198
199     move $a0,$t4
200     move $a1,$t5
201     jal dfs
202
203
204
205     judge_3_end:
206     ##### [x][y-1] #####
207     move $t4,$t0
208     subi $t5,$t1,1
209     get_matrix_addr($t2,$t4,$t5)
210     lw $t3,used($t2)
211     bnez $t3 judge_4_end
212     lw $t3,map($t2)
213     bnez $t3 judge_4_end
214
215     move $a0,$t4
216     move $a1,$t5
217     jal dfs
218
219
220
221     judge_4_end:
222     ##### used[x][y]=0; #####
223     get_matrix_addr($t2,$t0,$t1)
224     li $t3,0
225     sw $t3,used($t2)
226     # 4: pop
227     pop($t5)
228     pop($t4)
229     pop($t3)
230     pop($t2)
231     pop($t1)
232     pop($t0)
233     pop($ra)
234     # 5: return
235     jr $ra
236

```

P2_L1_factorial

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
9
10 int a[1001],h=1,n;
11
12
13
14 int main(){
15     a[1]=1;
16     scanf("%d",&n);
17     for(int i=2;i<=n;i++){
18         for(int j=1;j<=h;j++){
19             a[j]=a[j]*i;
20         }
21         for(int j=1;j<=h;j++){
22             if(a[j]>9){
23                 a[j+1]=a[j+1]+a[j]/10;
24                 a[j]=a[j]%10;
25             }
26         }
27         while(a[h+1]>0){
28             h++;
29             a[h+1]=a[h]/10;
30             a[h]=a[h]%10;
31         }
32     }
33     int flag=0;
34     for(int i=1000;i>=1;i--){
35         if(a[i]!=0)flag=1;
36         if(flag)printf("%d",a[i]);
37     }
38
39     return 0;
40 }

```

C代码: by GPT

```

1  #include <stdio.h>
2
3  #define MAX_DIGITS 10000
4
5  int main() {
6      int n;
7      printf("Enter a number to calculate its factorial: ");
8      scanf("%d", &n);
9
10     if (n < 0) {
11         printf("Factorial is not defined for negative numbers.\n");
12         return 1; // 返回错误码
13     }
14
15     int result[MAX_DIGITS] = {0};
16     result[0] = 1;
17     int length = 1;
18

```

```

19     for (int i = 2; i <= n; i++) {
20         int carry = 0;
21         for (int j = 0; j < length; j++) {
22             int product = result[j] * i + carry;
23             result[j] = product % 10;
24             carry = product / 10;
25         }
26         while (carry > 0) {
27             result[length] = carry % 10;
28             carry /= 10;
29             length++;
30         }
31     }
32
33     printf("Factorial of %d is: ", n);
34     for (int i = length - 1; i >= 0; i--) {
35         printf("%d", result[i]);
36     }
37     printf("\n");
38
39     return 0;
40 }

```

MIPS汇编代码:

```

1  .data
2      res: .space 8000
3
4
5
6  .macro getInt(%dest)
7      li $v0,5
8      syscall
9      move %dest,$v0
10 .end_macro
11
12 .macro printInt(%src)
13     move $a0,%src
14     li $v0,1
15     syscall
16 .end_macro
17
18 .macro get_vector_addr(%index, %x)
19     sll    %index, %x, 2
20 .end_macro
21
22
23 .text
24     getInt($s0) #n
25
26     li $s1,1      #len
27
28     li $t1,0
29     get_vector_addr($t2,$t1)
30     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 = remainder
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

```

```

86     for_1_i_end:
87
88
89     move $t0,$s1
90     subi $t0,$t0,1
91     for_2_begin:
92         blt $t0,$zero,for_2_end
93
94         get_vector_addr($t2,$t0)
95         lw $t3,res($t2)
96
97         printInt($t3)
98
99         subi $t0,$t0,1
100        j for_2_begin
101    for_2_end:
102
103
104    li $v0,10
105    syscall

```

P2_L1_calculate(推荐题)

MIPS汇编代码：

```

1  .data
2      letter: .space 1000
3      cnt: .space 1000
4      space: .asciiz " "
5      enter: .asciiz "\n"
6
7
8
9  .macro getChar(%dest)
10     li $v0,12
11     syscall
12     move %dest,$v0
13 .end_macro
14
15 .macro printChar(%src)
16     move $a0,%src
17     li $v0,11
18     syscall
19 .end_macro
20
21 .macro getInt(%dest)
22     li $v0,5
23     syscall
24     move %dest,$v0
25 .end_macro
26
27 .macro printInt(%src)
28     move $a0,%src

```

```

29     li $v0,1
30     syscall
31 .end_macro
32
33 .macro printStrOf(%src)
34     la $a0,%src
35     li $v0,4
36     syscall
37 .end_macro
38
39 .macro get_vector_addr(%index, %x)
40     sll    %index, %x, 2
41 .end_macro
42
43 .text
44     getInt($s0)
45
46     li $s2,10
47     li $s3,0
48
49     li $t0,0
50     for_1_begin:
51         bge $t0,$s0,for_1_end
52
53         getChar($t1)
54
55
56         li $s4,0 #flag
57         li $t2,0
58         for_j_begin:
59             bge $t2,$s3,for_j_end
60
61             # if letter[j]==char
62             get_vector_addr($t3,$t2)
63             lw $t4,letter($t3)
64
65             bne $t4,$t1,else
66                 lw $t5,cnt($t3)
67                 addi $t5,$t5,1
68                 sw $t5,cnt($t3)
69                 li $s4,1
70                 j for_j_end
71             else:
72
73                 addi $t2,$t2,1
74                 j for_j_begin
75         for_j_end:
76
77         bnez $s4,flag_is_1
78         get_vector_addr($t3,$s3)
79         sw $t1,letter($t3)
80         li $t1,1
81         sw $t1,cnt($t3)
82
83         addi $s3,$s3,1

```

```

84
85     flag_is_1:
86
87
88     addi $t0,$t0,1
89     j for_1_begin
90 for_1_end:
91
92     li $t0,0
93 for_2_begin:
94     bge $t0,$s3,for_2_end
95
96     get_vector_addr($t1,$t0)
97
98     lw $t2,letter($t1)
99     printChar($t2)
100    printStrOf(space)
101    lw $t2,cnt($t1)
102    printInt($t2)
103    printStrOf(enter)
104
105    addi $t0,$t0,1
106    j for_2_begin
107 for_2_end:
108
109    li $v0,10
110    syscall
111
112

```

P2_L1_honoi(推荐题)

MIPS汇编代码：

```

1  .data
2  mark1: .asciiz "move disk "
3  mark2: .asciiz " from "
4  mark3: .asciiz " to "
5  enter: .asciiz "\n"
6
7  .macro getChar(%dest)
8      li $v0,12
9      syscall
10     move %dest,$v0
11 .end_macro
12
13 .macro printStrOf(%src)
14     la $a0, %src
15     li $v0, 4
16     syscall
17 .end_macro
18
19 .macro printChar(%src)
20     move $a0,%src
21     li $v0,11

```

```

22     syscall
23 .end_macro
24
25 .macro getInt(%dest)
26     li $v0,5
27     syscall
28     move %dest,$v0
29 .end_macro
30
31 .macro printInt(%src)
32     move $a0,%src
33     li $v0,1
34     syscall
35 .end_macro
36
37 .macro get_vector_addr(%index, %x)
38     sll    %index, %x, 2
39 .end_macro
40
41 #push
42 .macro push(%src)
43     sw      %src, 0($sp)
44     subi    $sp, $sp, 4
45 .end_macro
46 #pop
47 .macro pop(%des)
48     addi    $sp, $sp, 4
49     lw      %des, 0($sp)
50 .end_macro
51
52
53 .text
54     getInt($s0) #n
55     move $a0,$s0
56     li $t1,65
57     move $a1,$t1
58     li $t1,66
59     move $a2,$t1
60     li $t1,67
61     move $a3,$t1
62
63     jal hanoi
64
65
66     li $v0,10
67     syscall
68
69
70 hanoi:
71 #1: push
72     push($ra)
73     push($t0)
74     push($t1)
75     push($t2)
76     push($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

```

```
132     pop_and_return:
133     #4: pop
134         pop($t3)
135         pop($t2)
136         pop($t1)
137         pop($t0)
138         pop($ra)
139     #5: return
140         jr $ra
141
142
143
144     move_:
145     #1: push
146         push($ra)
147         push($t0)
148         push($t1)
149         push($t2)
150     #2: move args
151         move $t0,$a0
152         move $t1,$a1
153         move $t2,$a2
154     #3: body
155         printStrOf(mark1)
156         printInt($t0)
157         printStrOf(mark2)
158         printChar($t1)
159         printStrOf(mark3)
160         printChar($t2)
161         printStrOf(enter)
162
163     #4: pop
164         pop($t2)
165         pop($t1)
166         pop($t0)
167         pop($ra)
168     #5: return
169         jr $ra
170
```