花开一季 叶落一地 P2 课上

花开一季 叶落一地 P2 课上
- 冒泡排序
  - C语言（针对升序）
  - MIPS
  - 拓展—选择排序C
  - 初次快排（判断条件竟然不反过来写？亏你写的出来）
- 斐波那契数列
  - C代码
  - 虚假的递归
  - 真正的递归
  - 数列求解
- 约瑟夫
  - 递归+从k开始+有中间输出
  - 迭代+ 从k开始+无中间输出
  - 数组遍历
- 全排列
  - C代码
  - MIPS
- 矩阵乘法
- 高精度阶乘
  - C++
  - MIPS

冒泡排序

C语言（针对升序）

```c
void bubbleSort(int k[], int n)
{
    int i, j, temp, flag;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0, flag = 0; j < n - i - 1; j++)
            if (k[j] > k[j + 1])
            { /*这里是升序，如果是降序都会改吧*/
                temp = k[j];
                k[j] = k[j + 1];
                k[j + 1] = temp; /* 交换两个元素的位置 */
                flag = 1;
            }
        if (flag == 0)
```

```
            break;
    }
}
```

MIPS

```
.data
ans: .space 4080
space: .asciiz " "

.macro read(%n)
    li $v0, 5
    syscall
    move %n, $v0
.end_macro

.macro write(%n, %h)
    move $a0, %n
    li $v0, %h
    syscall
.end_macro

.text
read($s0) # $s0 = n
li $s1, 0 # $s1 = i
li $s2, 0 # $s2 = 4 * i

input:
    bge $s1, $s0, sort
    read($s3)
    sw $s3, ans($s2)
    addi $s1, $s1, 1
    addi $s2, $s2, 4
    j input

sort:
    li $s1, 0
    subi $s3, $s0, 1 # $s3 = n - 1

    for_i:
        bge $s1, $s3, end_i
        li $s4, 0 # $ s4 = flag
        li $s5, 0 # $ s5 = j
        sub $s7, $s3, $s1 # $s7 = n - 1 - i
```

```
            for_j:
            bge $s5, $s7, end_j
            sll $t2, $s5, 2
            addi $t3, $t2, 4
            lw $t0, ans($t2)
            lw $t1, ans($t3)
            ble $t0, $t1, end_swap
            sw $t0, ans($t3)
            sw $t1, ans($t2)
            li $s4, 1

        end_swap:
            addi $s5, $s5, 1
            j for_j

    end_j:
        beqz $s4, end_i
        addi $s1, $s1, 1
        j for_i

end_i:
    li $s1, 0 # $s1 = i
    li $s2, 0 # $s2 = 4 * i

output:
    bge $s1, $s0, end
    lw $s3, ans($s2)
    write($s3, 1)
    la $s4, space
    write($s4, 4)
    addi $s1, $s1, 1
    addi $s2, $s2, 4
    j output

end:
    li $v0, 10
    syscall
```

拓展—选择排序C

```c
void selectSort(int k[], int n)
{
    int i, j, d, temp;
    for (i = 0; i < n - 1; i++)
    {
        d = i;
        for (j = i + 1; j < n; j++)
            if (k[j] < k[d])
                d = j;
        if (d != i)
        {
            temp = k[d]; /* 最小值元素非未排序元素的第一个元素时 */
            k[d] = k[i];
            k[i] = temp;
        }
    }
}
```

初次快排（判断条件竟然不反过来写？亏你写的出来）

```
.data
  ans: .space 4096
  space: .asciiz ","
  left: .asciiz "["
  right: .asciiz "]"
.text
  li $v0, 5
  syscall
  move $t0, $v0 # t0, n

  la $s0, ans # s0, ans
  move $t1, $zero # t1, i

input:
  beq $t1, $t0, sort
  li $v0, 5
  syscall
  sll $t2, $t1, 2 # t2  i * 4
  add $t2, $s0, $t2
  sw $v0, 0($t2)
  addi $t1, $t1, 1
  j input
```

```
23
24  sort:
25    move $a0, $zero # a0 i
26    subi $a1, $t0, 1 # a1 j
27    jal quicksort
28    move $t1, $zero
29
30    la $a0, left
31    li $v0, 4
32    syscall
33
34    j print
35
36  quicksort:
37    blt $a0, $a1, owo
38  owon:
39    jr $ra
40
41  owo:
42
43    move $t5, $a0
44    move $t6, $a1
45    sll $t2, $t5, 2
46    add $t2, $s0, $t2
47    lw $t4, 0($t2) # t4 key
48
49  while1:
50    bge $t5, $t6, get2
51
52  while2:
53    blt $t5, $t6, con1
54    j while1
55
56  con1:
57    sll $t2, $t6, 2
58    add $t2, $s0, $t2
59    lw $t9, 0($t2)
60
61    bge $t9, $t4, con2
62
63    sll $t2, $t5, 2
64    add $t2, $s0, $t2
65    sw $t9, 0($t2)
66
67    j while3
```

```
68
69  con2:
70    subi $t6, $t6, 1
71    j while2
72
73  while3:
74    blt $t5, $t6, con3
75    j while1
76
77
78  con3:
79    sll $t2, $t5, 2
80    add $t2, $s0, $t2
81    lw $t8, 0($t2)
82
83    ble $t8, $t4, con4
84
85    sll $t2, $t6, 2
86    add $t2, $s0, $t2
87    sw $t8, 0($t2)
88
89    j while1
90
91  con4:
92    addi $t5, $t5, 1
93    j while3
94
95  get2:
96    sll $t2, $t5, 2
97    add $t2, $s0, $t2
98    sw $t4, 0($t2)
99
100   move $v0, $t5
101
102   subi $sp, $sp, 32
103   sw $ra, 0($sp)
104   sw $a0, 4($sp)
105   sw $a1, 8($sp)
106   sw $v0, 12($sp)
107   subi $a1, $v0, 1
108   jal quicksort
109   lw $ra, 0($sp)
110   lw $a0, 4($sp)
111   lw $a1, 8($sp)
112   lw $v0, 12($sp)
```

```
  addi $sp, $sp, 32

  subi $sp, $sp, 32
  sw $ra, 0($sp)
  sw $a0, 4($sp)
  sw $a1, 8($sp)
  sw $v0, 12($sp)
  addi $a0, $v0, 1
  jal quicksort
  lw $ra, 0($sp)
  lw $a0, 4($sp)
  lw $a1, 8($sp)
  lw $v0, 12($sp)
  addi $sp, $sp, 32
  j owon

print:
  beq $t1, $t0, exit
  sll $t2, $t1, 2
  add $t2, $s0, $t2
  lw $a0, 0($t2)
  li $v0, 1
  syscall
  addi $t1, $t1, 1
  beq $t1, $t0, exit
  la $a0, space
  li $v0, 4
  syscall
  j print


exit:
  la $a0, right
  li $v0, 4
  syscall
  li $v0, 10
  syscall
```

斐波那契数列

C代码

```c
int f(int n)
{
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return f(n - 1) + f(n - 2);
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("%d", f(n));
    return 0;
}
```

虚假的递归

```
.data

.macro read(%n)
    li $v0, 5
    syscall
    move %n, $v0
.end_macro

.macro write(%n)
    move $a0, %n
    li $v0, 1
    syscall
.end_macro

.text
read($s0) # $s0 = n
move $a0, $s0 # $a0 = n
li $s1, 0 # $s1 = ans
jal f
write($s1)
li $v0, 10
syscall

```

```
f:
    beq $a0, 0, f0
    beq $a0, 1, f1
    subi $sp, $sp, 8
    sw $a0, 0($sp)
    sw $ra, 4($sp)
    subi $a0, $a0, 1
    jal f
    subi $a0, $a0, 1
    jal f
    lw $a0, 0($sp)
    lw $ra, 4($sp)
    addi $sp, $sp, 8
    jr $ra

f0:
    addi $s1, $s1, 0
    jr $ra

f1:
    addi $s1, $s1, 1
    jr $ra
```

真正的递归

```
.data

.macro read(%n)
    li $v0, 5
    syscall
    move %n, $v0
.end_macro

.macro write(%n)
    move $a0, %n
    li $v0, 1
    syscall
.end_macro

.text
read($s0) # $s0 = n
move $a0, $s0 # $a0 = n
li $s1, 0 # $s1 = ans
jal f
```

```
write($v0)
li $v0, 10
syscall

f:
    beq $a0, 0, f0
    beq $a0, 1, f1
    subi $sp, $sp, 8
    sw $a0, 0($sp)
    sw $ra, 4($sp)
    subi $a0, $a0, 1
    jal f
    subi $sp, $sp, 4
    sw $v0, 0($sp)
    subi $a0, $a0, 1
    jal f
    lw $s1, 0($sp)
    addi $sp, $sp, 4
    move $s2, $v0
    add $v0, $s1, $s2 # f(n) = f(n - 1) + f(n - 2)
    lw $a0, 0($sp)
    lw $ra, 4($sp)
    addi $sp, $sp, 8
    jr $ra

f0:
    li $v0, 0
    jr $ra

f1:
    li $v0, 1
    jr $ra
```

数列求解

```
.data
    ans: .space 4080

.macro read(%n)
    li $v0, 5
    syscall
    move %n, $v0
.end_macro

```

```
.macro write(%n)
    move $a0, %n
    li $v0, 1
    syscall
.end_macro

.text
read($s0) # $s0 = n
li $s1, 2 # $s1 = i
li $s2, 0 # $s2 = i - 2
li $s3, 1 # $s1 = i - 1
sw $s2, ans
sw $s3, ans+4

loop:
    bgt $s1, $s0, end
    sll $s1, $s1, 2
    sll $s2, $s2, 2
    sll $s3, $s3, 2
    lw $s4, ans($s2)
    lw $s5, ans($s3)
    add $s6, $s4, $s5
    sw $s6, ans($s1)
    srl $s1, $s1, 2
    srl $s2, $s2, 2
    srl $s3, $s3, 2
    addi $s1, $s1, 1
    addi $s2, $s2, 1
    addi $s3, $s3, 1
    j loop

end:
    sll $s0, $s0, 2
    lw $s1, ans($s0)
    write($s1)
    li $v0, 10
    syscall
```

约瑟夫

递归+从k开始+有中间输出

- C

```cpp
#include<bits/stdc++.h>
using namespace std;
//用递归实现约瑟夫环问题

int cir(int n, int m, int i)
{
    if (i == 1)
        return (m - 1 + n) % n;
    return (cir(n - 1, m, i - 1) + m) % n;
}
int main()
{
    int n, m, k;
    cin >> n >> m >> k;
    for (int i = 1; i <= n; i++)
    {
        int tmp = (cir(n, m, i) + k) % n;
        if (tmp == 0)
            printf("%d\n", n);
        else
            printf("%d\n", tmp);
    }
    return 0;
}
```

- MIPS(zes)

```
.data
  Enter:.asciiz "\n"
.macro read(%n)
  li $v0, 5
  syscall
  move %n, $v0
.end_macro

.macro write(%n)
  move $a0, %n
  li $v0, 1
  syscall
.end_macro

```

```
.text
  read($s0) #s0-n
  read($s1) #s1-m
  read($s7) #s7-k

  li $t0, 1 #t0-1
  li $s2, 0 #s2-ans
For:
  bgt $t0, $s0, End
  move $a0, $s0   #n
  move $a1, $s1   #m
  move $a2, $t0   #i
  jal cir
  add $s2, $s2, $s7 #ans = (cir(n,m,i) + k)
  div $s2, $s0  #ans = (cir(n,m,i) + k) % n
  mfhi $s2
  beq $s2, $0, isZero
  write($s2)  #ans = (cir(n,m,i) + k) % n
  j continue
isZero:
  write($s0)  #ans = n
continue:
  la $a0, Enter
  li $v0, 4
  syscall
  li $s2, 0

  addi $t0,$t0, 1
  j For
cir:
  addi $sp, $sp, -16
  sw $ra, 0($sp)
  sw $a0, 4($sp)
  sw $a1, 8($sp)
  sw $a2, 12($sp)

  beq $a2, 1, i_1
  addi $a0, $a0, -1
  addi $a2, $a2, -1
  jal cir

  lw $ra, 0($sp)
  lw $a0, 4($sp)
  lw $a1, 8($sp)
  lw $a2, 12($sp)
```

```
  addi $sp, $sp, 16
  add $s2, $s2, $a1
  div $s2, $a0
  mfhi $s2
  jr $ra
i_1:
  add $s2, $s2, $a1
  addi $s2, $s2, -1
  add $s2, $s2, $a0
  div $s2, $a0
  mfhi $s2
  lw $ra, 0($sp)
  lw $a0, 4($sp)
  lw $a1, 8($sp)
  lw $a2, 12($sp)
  addi $sp, $sp, 16
  jr $ra
End:
  li $v0, 10
  syscall
```

迭代+ 从k开始+无中间输出

- C

```c
#include <stdio.h>
int n, m, k;
int cir(int n, int m)
{
    int p = 0;
    int i;
    for (i = 2; i <= n; i++)
        p = (p + m) % i;
    return p + 1;
}
int main()
{
    while (~scanf("%d%d%d", &n, &m, &k))
        printf("%d\n", (cir(n, m) + k) % n == 0 ? n : (cir(n, m) + k) % n);
}
```

- MIPS(lxy)

```
.macro read_int(%n)
```

```
  li $v0, 5
  syscall
  move %n, $v0
.end_macro

.macro write_int(%n)
  move $a0, %n
  li $v0, 1
  syscall
.end_macro

.text
main:
  read_int($t0)  # n
  read_int($t1)  # m
  read_int($t2)  # k

  move $a0, $t0
  move $a1, $t1
  move $a2, $t2

  jal ysf

  move $t0, $a0
  write_int($t0)

  li $v0, 10
  syscall

ysf:
  move $t0, $a0 # n
  move $t1, $a1 # m
  move $t4, $a2 # k

  li $t2, 0  # ans = 0
  li $t3, 2  # i = 2

  loop:
  bgt $t3, $t0, end_loop
    add $t2, $t2, $t1  #ans = ans + m
    div $t2, $t3
    mfhi $t2
    addi $t3, $t3, 1
  j loop

```

```
  end_loop:
  add $t2, $t2, $t4  #ans = ans + x
  div $t2, $t0  #ans + x % n
  mfhi $t2
  beqz $t2,moven  # ans ? ans : n
  move $a0, $t2
  j return
  moven:
    move $a0, $t0
  return:
  jr $ra
```

数组遍历

- C (lh)

```
#include <iostream>
using namespace std;
bool visit[200];
int main()
{
    int m, n;
    cin >> m >> n;
    int s = 0, num = 0;
    for (int i = 1; num < m; i++)
    {
        if (i > m)
            i %= m;
        if (visit[i])
            continue;
        s++;
        if (s == n)
        {
            cout << i << " ";
            visit[i] = true;
            s = 0;
            num++;
        }
    }
    return 0;
}
```

- MIPS(lh)

```
.data
  array:.space 4000
  b:.asciiz" "
.text
Input:
  li $v0,5
  syscall
  move $s0,$v0#m个人
  li $v0,5
  syscall
  move $s1,$v0#n为所报数字
  li $t0,0
  li $t4,1#i
  li $t5,0#s
  li $t6,0#num
  li $s2,1#与1比较
Init:
  beq $t0,$s0,For
  move $t1,$t0
  sll $t1,$t1,2
  sw $0,array($t1)
  addi $t0,$t0,1
  j Init
For:
  bge $t6,$s0,END
  bgt $t4,$s0,DIV
After_DIV:
  move $t1,$t4
  sll $t1,$t1,2
  lw $t2,array($t1)
  beq $t2,$s2,continue
  addi $t5,$t5,1
  beq $t5,$s1,Print
  addi $t4,$t4,1
  j For
DIV:
  div $t4,$s0
  mfhi $t4
  j After_DIV
continue:
  addi $t4,$t4,1
  j For
Print:
  li $v0,1
  move $a0,$t4
```

```
  syscall
  li $v0,4
  la $a0,b
  syscall
  #visited[i] = true
  move $t1,$t4
  sll $t1,$t1,2
  sw $s2,array($t1)
  li $t5,0
  addi $t6,$t6,1
  addi $t4,$t4,1
  j For
END:
  li $v0,10
  syscall
```

- MIPS(lsr)

```
.data
n: .word 0
m: .word 0
a: .space 1024
char: .asciiz "\n"
.text
la $t0, n
la $t1, m
la $s0, a
li $t2, 0#cnt
li $t3, 0#i
li $t4, 0#k
#输入n
li $v0, 5
syscall
sw $v0, ($t7)
lw $t0, ($t7)
#输入m
li $v0, 5
syscall
sw $v0, ($t7)
lw $t1, ($t7)
#跳转进入循环
jal choose

choose:
```

```
beq $t2, $t0, end#所有人均出局
#cnt!=n
addi $t3, $t3, 1#i+=1
bgt $t3, $t0, restart#判断i>n?
jal if_change

restart:
li $t3, 1
jal if_change

if_change:
#t6=a[i]
mul $t5, $t3, 4
add $t5, $t5, $s0
lw $t6, 0($t5)

beq $t6, 0, change#a[i]==0?
jal choose

change:
addi $t4, $t4, 1
beq $t4, $t1, change_#if k==m?
jal choose#k!=m

change_:
#t6=a[i]
mul $t5, $t3, 4
add $t5, $t5, $s0
lw $t6, 0($t5)
#a[i]=1
li $t8, 1
sw $t8, 0($t5)

addi $t2, $t2, 1#cnt+=1
#输出
li $v0, 1
move $a0, $t3
syscall
li $v0, 4
la $a0, char
syscall

li $t4, 0#k=0
jal choose

```

```
end:
li $v0, 10
syscall
```

全排列

C代码

```c
#include <stdio.h>
#define MAXL (25)

int symbol[MAXL], array[MAXL], n;

void FullArray(int index)
{
    int i;
    if (index >= n)
    {
        for (i = 0; i < n; i++)
            printf("%d ", array[i]);
        putchar('\n');
        return;
    }
    for (i = 0; i < n; i++)
        if (symbol[i] == 0)
        {
            array[index] = i + 1;
            symbol[i] = 1;
            FullArray(index + 1);
            symbol[i] = 0;
        }
}

int main()
{
    scanf("%d", &n);
    FullArray(0);
    return 0;
}
```

MIPS

```
.data
symbol: .space 100
array: .space 100
space: .asciiz " "
enter: .asciiz "\n"

.macro read(%n)
    li $v0, 5
    syscall
    move %n, $v0
.end_macro

.macro write(%n, %h)
    move $a0, %n
    li $v0, %h
    syscall
.end_macro

.text
read($s0) # $s0 = n
li $a0, 0 # $a0 = index
jal dfs
li $v0, 10
syscall

dfs:
    blt $a0, $s0, no_print
    li $t0, 0 # $t0 = i

print:
    bge $t0, $s0, print_enter
    sll $t1, $t0, 2
    lw $t2, array($t1)
    write($t2, 1)
    la $t2, space
    write($t2, 4)
    addi $t0, $t0, 1
    j print

print_enter:
    la $t2, enter
    write($t2, 4)
    jr $ra
```

```
44
45  no_print:
46      li $t0, 0
47      loop:
48          bge $t0, $s0, return
49          sll $t1, $t0, 2
50          lw $t2, symbol($t1)
51          bnez $t2, end_if
52          addi $t3, $t0, 1
53          sll $t4, $a0, 2
54          sw $t3, array($t4)
55          li $t4, 1
56          sw $t4, symbol($t1)
57
58          subi $sp, $sp, 12
59          sw $a0, 0($sp)
60          sw $t0, 4($sp)
61          sw $ra, 8($sp)
62          addi $a0, $a0, 1
63          jal dfs
64          lw $a0, 0($sp)
65          lw $t0, 4($sp)
66          lw $ra, 8($sp)
67          addi $sp, $sp, 12
68
69          sll $t1, $t0, 2
70          li $t2, 0
71          sw $t2, symbol($t1)
72
73  end_if:
74      addi $t0, $t0, 1
75      j loop
76
77  return:
78      jr $ra
```

矩阵乘法

```
1  .data
2  a: .space 256
3  B: .space 256
4  c: .space 256
5
6  str_enter: .asciiz "\n"
```

```
7  str_space: .asciiz " "
8
9  .macro RI(%i)
10   li $v0, 5
11   syscall
12   move %i, $v0
13 .end_macro
14
15 .macro PI(%i)
16   li $v0, 1
17   move $a0, %i
18   syscall
19 .end_macro
20
21 .macro Pstr(%i)
22   la $a0, %i
23   li $v0, 4
24   syscall
25 .end_macro
26
27 .macro getIndex(%ans, %i, %j)
28   sll %ans, %i, 3
29   add %ans, %ans, %j
30   sll %ans, %ans, 2
31 .end_macro
32
33 .macro LOAD_LOCAL(%i)
34   addi $sp, $sp, 4
35   lw %i, 0($sp)
36 .end_macro
37
38 .macro SAVE_LOCAL(%i)
39   sw $i, 0($sp)
40   subi $sp, $sp, 4
41 .end_macro
42
43 .text
44   RI($s0)
45
46   li $t0, 0
47 in_1_i:
48   beq $t0, $s0, in_1_i_end
49   li $t1, 0
50   in_1_j:
51     beq $t1, $s0, in_1_j_end
```

```
    RI($v0)
    getIndex($t2, $t0, $t1)
    sw $v0, a($t2)
    addi $t1, $t1, 1
    j in_1_j
  in_1_j_end:
  addi $t0, $t0, 1
  j in_1_i
in_1_i_end:

  li $t0, 0
in_2_i:
  beq $t0, $s0, in_2_i_end
  li $t1, 0
  in_2_j:
    beq $t1, $s0, in_2_j_end
    RI($v0)
    getIndex($t2, $t0, $t1)
    sw $v0, B($t2)
    addi $t1, $t1, 1
    j in_2_j
  in_2_j_end:
  addi $t0, $t0, 1
  j in_2_i
in_2_i_end:

  li $t0, 0 # i
for_i:
  beq $t0, $s0, for_i_end
  li $t1, 0 # j
  for_j:
    beq $t1, $s0, for_j_end
    li $t2, 0 # k
    li $t3, 0 ##c[i][j]
    for_k:
      beq $t2, $s0, for_k_end
      getIndex($t4, $t0, $t2)
      lw $t6, a($t4)
      getIndex($t5, $t2, $t1)
      lw $t7, B($t5)
      mul $t4, $t6, $t7
      add $t3, $t3, $t4
      add $t2, $t2, 1
      j for_k
    for_k_end:
```

```
    getIndex($t4, $t0, $t1)
    sw $t3, c($t4)
    addi $t1, $t1, 1
    j for_j
  for_j_end:
  addi $t0, $t0, 1
  j for_i
for_i_end:

  li $t0, 0
out_i:
  beq $t0, $s0, out_i_end

  li $t1, 0
  out_j:
    beq $t1, $s0, out_j_end
    getIndex($t2, $t0, $t1)
    lw $t3, c($t2)
    PI($t3)
    Pstr(str_space)
    addi $t1, $t1, 1
    j out_j
  out_j_end:

  Pstr(str_enter)
  addi $t0, $t0, 1
  j out_i
out_i_end:

li $v0, 10
syscall
```

高精度阶乘

C++

```cpp
#include <iostream>
using namespace std;
const int length = 2500; //这个值经过多次调整，才过了10000!
int a[length];
int main()
{
    a[0] = 1; //首位置为1;
    int n;
```

```
    cin >> n; //输入要计算阶乘的数
    for (int i = 2; i <= n; i++)
    {
        int jinwei = 0;
        int j = 0;
        int temp;
        while (j < length)
        {
            temp = a[j] * i + jinwei;
            jinwei = temp / 10;
            a[j] = temp % 10;
            j++;
        }
    }
    int k = length - 1;
    while (!a[k])
    { //将为0的数全跳过，不输出
        k--;
    }
    // printf("%d\n", k);
    while (k >= 0)
    { //输出正确的阶乘结果
        cout << a[k];
        k--;
    }
    return 0;
}
```

MIPS

```
.data
a: .word 0:2500

.macro RI(%i)
  li $v0, 5
  syscall
  move %i, $v0
.end_macro

.macro PI(%i)
  li $v0, 1
  move $a0, %i
  syscall
.end_macro
```

```
15
16  .macro getIndex(%ans, %i)
17    sll %ans, %i, 2
18  .end_macro
19
20  .text
21  main:
22    RI($s0)
23    li $s1, 1 # len
24    li $t6, 10
25    li $t0, 1
26    sw $t0, a($0)
27    addi $t0, $t0, 1 # i
28
29    for:
30      bgt $t0, $s0, endfor
31      li $t2, 0 # carry
32      li $t1, 0 # j
33      while:
34        bge $t1, $s1, endwhile
35        getIndex($t4, $t1)
36        lw $t5, a($t4)
37        mul $t5, $t5, $t0
38        add $t5, $t5, $t2
39        div $t5, $t6
40        mflo $t2
41        mfhi $t5
42        sw $t5, a($t4)
43        addi $t1, $t1, 1
44        j while
45      endwhile:
46
47      whileC:
48        beqz $t2, endwhileC
49        div $t2, $t6
50        mflo $t2
51        mfhi $t5
52        getIndex($t4, $s1)
53        sw $t5, a($t4)
54        addi $s1, $s1, 1
55        j whileC
56      endwhileC:
57
58      addi $t0, $t0, 1
59      j for
```

```
endfor:

move $t0, $s1
whileO:
  getIndex($t1, $t0)
  lw $t2, a($t1)
  bgtz $t2, endwhileO
  subi $t0, $t0, 1
  j whileO
endwhileO:

whileOut:
  bltz $t0, endwhileOut
  getIndex($t1, $t0)
  lw $t2, a($t1)
  PI($t2)
  subi $t0, $t0, 1
  j whileOut
endwhileOut:

li $v0, 10
syscall
```