



《计算机组成原理与接口技术实验》 实验报告

(实验一)

学 院 名 称 : 数据科学与计算机学院

专业 (班级) : 16 软件工程二 (4) 班

学 生 姓 名 : 刘亚辉

学 号 : 16340157

时 间 : 2018 年 4 月 10 日

成绩：

实验一：MIPS汇编语言程序设计实验

一. 实验目的

1. 认识和掌握MIPS汇编语言程序设计的基本方法；
2. 熟悉PCSpim模拟器的使用；

二. 实验内容

编写一程序，实现将既包含在数组 A 中又包含在数组 B 中的无符号字数取出并存储于内存中，其中数组 A 包含 20 个数，数组 B 包含 30 个数。如找不到相同的数则显示“No same!”。

三. 实验器材

电脑一台，PCSpim仿真器软件一套。

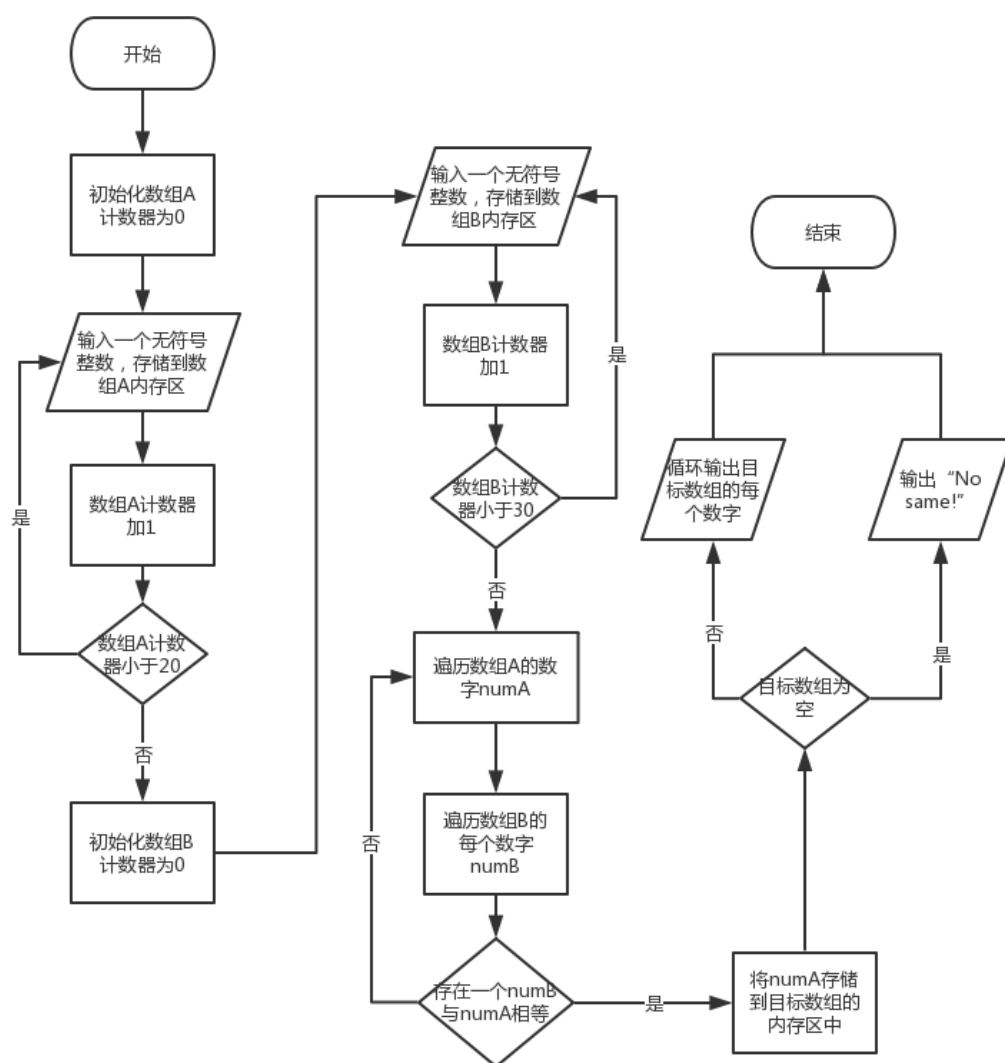
四. 实验过程与结果

1.设计的思想与方法：

从实验所要实现的要求来看，可以将整个程序分为三个部分：输入存储阶段，遍历查询相同的数字并存储，输出阶段（若目标数组不为空则还需要进行去重操作）。

通过各种汇编指令：加法操作：add，addi，取指令：li，区地址：la，加载字：lw，存储字：sw，以及beq，slt等比较大小的指令来实现条件分支语句，循环语句等复杂模块。采用模块化编程，将程序分为一个个更小的模块来进行实现，最终即可完成想要的功能。

2.程序流程图：



3. 实验步骤:

首先进行输入存储, 由于输入的数字较多, 所以采用循环输入, 并且将listA和listB分别输入。通过采用计数操作, 和间接寻址访问内存的方法, 每输入一个数组, 将计数器加1, 存储数字之后, 将偏移量加4; 然后通过beq或者slt等比较操作来判断是否达到输入数字个数的上限, 若没达到, 则继续进行下一个数字的输入, 否则就顺序执行下面的代码。

输入完成之后, 通过一个双重循环, 外层循环遍历listA的每个数numA (计数器从1-20), 内层循环遍历listB的每个数字numB (计数器从1-30): 若存在一个numB=numA则将numA (或者numB) 存储到目标数组中, 并记录其长度; 否则的话, 将外层循环计数器加1, 判断下一个数字是否也出现在listB中。直到外层循环结束就得到完整的目标数组。

最后，进行输出操作。首先判断目标数组是否为空，若为空则输出 “No same!” 即可；否则的话，遍历整个目标数组（计数器从1-目标数组的长度），去重操作可以嵌套一个内层循环，遍历范围是已经输出的数字，即从1-外层循环计数器，左闭右开，这样即可在内层循环判断当前数字是否已经输出过，输出的话就跳转到外层循环执行，相当于一次continue操作，没有输出的话就将当前数字输出即可。

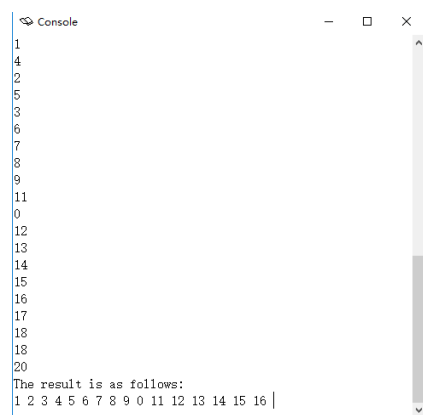
在所有代码执行完毕之后，调用系统调用退出程序。

4.实验结果及分析

实验输入样例1:

```
listA: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0,
10, 11, 12, 13, 14, 15, 16, 1, 70, 19
listB: 11, 22, 33, 44, 55, 66, 77,
88, 99, 100, 1, 4, 2, 5, 3, 6, 7, 8,
9, 11, 0, 12, 13, 14, 15, 16, 17, 18,
18, 20
```

样例1结果:

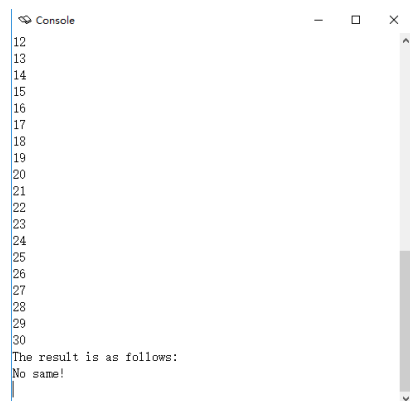


```
Console
1
4
2
5
3
6
7
8
9
11
0
12
13
14
15
16
17
18
18
20
The result is as follows:
1 2 3 4 5 6 7 8 9 0 11 12 13 14 15 16
```

实验样例输入2:

```
listA: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0,
1, 2, 3, 4, 5, 6, 7, 8, 9, 0
listB: 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25,
26, 27, 28, 29, 30
```

样例2结果:



```
Console
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
The result is as follows:
No same!
```

实验心得

这次实验对我来说是一次全新的体验，之前只是知道汇编语言，知其然但不知所以然，这次实验让我从实际出发，用汇编语言来编写一个简单功能的程序，切身感受汇编语言，从而对汇编语言有一个深刻的认识 and 了解。

在编写代码之前，我主要学习了老师所提供的几个实例代码，通过一个个简单功能模块的实现，了解汇编语言的执行过程及其语法规范。熟悉了几个常用的命令 add, lw, sw,

li, la, j, deq, slt 之后我便开始尝试书写代码。正如之前所写的详细流程那样，将整个代码分为三个模块，每实现一个模块的功能，就通过常用的打点输出检验模块的正确性，从而减轻完整代码实现之后 debug 的任务量。在整个实现过程中，遇到的比较严重的问题是在代码运行之后报错：“Exception 5 [Address error in store] occurred and ignored”，通过在 stackoverflow 中查找解决方案，发现是因为数据区分配空间的时候内存区没有对齐，通过尝试，发现分配空间时加上.align 来进行对齐操作即可解决。另一种解决方案是，将需要分配的内存区放在数据区的前面位置，并且尽量是 4 的倍数即可。

在最终的测试阶段也遇到了一个一开始没有在意的细节，没有进行去重操作。有了之前查询相同数字的经验，通过嵌套一个内层循环在输出阶段进行去重即可实现该功能。这次实验我收获最大的就是感受到了一个之前没有接触的程序环境，汇编语言更加接近底层，相信通过研究学习汇编语言，可以更加清楚的认识高级语言与系统进行交互的过程，并且进一步学习内存区的分配问题，让我们对数据的存储有一个清醒的认识，从而为提高代码效率找到一个合适出路：进行数据对齐等等。

【程序代码】

```

1.  #####
2.  #                实验三                #
3.  #####
4.
5.  #-----数据 segment-----#
6.
7.  .data
8.      listA:
9.          .space 80
10.     listB:
11.         .space 120
12.     outList:
13.         .space 80
14.     msgA:
15.         .asciiiz "Please input 20 numbers as the listA: \n"
16.     msgB:
17.         .asciiiz "Please input 30 numbers as the listB: \n"
18.     msgOutput:
19.         .asciiiz "The result is as follows: \n"
20.     msgNoSame:
21.         .asciiiz "No same!\n"
22.     str:
23.         .asciiiz " "
24.
25. #-----代码 segment-----#
26.
27. .text
28. .globl main

```

```

29.
30. storeNum:                                #存储相同的数字代码
31.     addi    $t4,    $t4,    1              #outList 计数器加 1
32.     lw      $s7,    0($t3)
33.     sw      $s7,    0($t9)                #存放相同的数字
34.     addi    $t9,    $t9,    4              #outList 偏移量加 4
35.     j       queryOutLoop
36.
37. main:                                     #主程序入口
38.     addi    $s1,    $zero, 20              #ListA 总数初始化
39.     addi    $t1,    $zero, 1              #ListA 计数器
40.     la      $t2,    listA                  #listA 首地址
41.     add     $t3,    $t2,    $zero          #将 listA 首地址$t2 赋值给变量
    $t3
42.
43.     addi    $s2,    $zero, 30              #ListB 总数初始化
44.     addi    $t5,    $zero, 1              #ListB 计数器
45.     la      $t6,    listB                  #listB 首地址
46.     add     $t7,    $t6,    $zero          #将 listB 首地址$t6 赋值给变量
    $t7
47.
48.     add     $t4,    $zero, $zero           #outList 计数器
49.     la      $t8,    outList                #outList 首地址
50.     add     $t9,    $t8,    $zero          #将 outList 首地址$t8 赋值给变量
    $t9
51.     li      $v0,    4                      #打印提示信息, 字符串
52.     la      $a0,    msgA                  #读取字符串地址, 输出
53.     syscall
54.
55. #-----代码 输入-----#
56.
57. loopAinput:                               #循环输入 listA
58.     li      $v0,    5                      #接收一个整数
59.     syscall
60.
61.     #move    $a0,    $v0                    #Debug 输出当前输入的数字
62.     #li      $v0,    1
63.     #syscall
64.
65.     sw      $v0,    0($t3)                #存放输入的数字
66.
67.     beq     $t1,    $s1,    tipsMsg        #判断输入的数字是否足够
68.     addi    $t1,    $t1,    1              #计数器加 1
69.

```

```

70.    addi    $t3,    $t3,    4           #偏移加 4
71.    j      loopAinput
72.
73. tipsMsg:                                #listB 输入的提示信息
74.    li      $v0,    4                   #打印提示信息, 字符串
75.    la      $a0,    msgB                #读取字符串地址, 输出
76.    syscall
77.
78. loopBinput:
79.    li      $v0,    5                   #接收一个整数
80.    syscall
81.
82.    sw      $v0,    0($t7)               #存放输入的数字
83.
84.    beq     $t5,    $s2,    queryInit    #判断输入的数字是否足够
85.    addi    $t5,    $t5,    1           #计数器加 1
86.
87.    addi    $t7,    $t7,    4           #偏移加 4
88.    j      loopBinput
89.
90. #-----代码 查找相同的数字-----#
91.
92. queryInit:                                #listA、B 输入完成, 检索相同数字
    的循环初始化
93.    addi    $t1,    $zero,    0         #listA 计数器置 0
94.    addi    $t3,    $t2,    -4         #listA 首地址
95.
96. queryOutLoop:                            #检索外层循环
97.    addi    $t1,    $t1,    1         #ListA 计数器加 1
98.    addi    $t3,    $t3,    4         #偏移加 4
99.    addi    $t5,    $zero,    0       #listB 计数器置 0
100.    addi    $t7,    $t6,    -4       #将 listB 首地址$t6 赋值给变量
    $t7
101.    beq     $t1,    21,    outputInit  #判断退出循环的条件
102.
103. queryInLoop:                            #检索内层循环
104.    addi    $t5,    $t5,    1         #listB 计数器加 1
105.    addi    $t7,    $t7,    4         #listB 指针偏移
106.
107.    lw      $s4,    0($t3)             #取出 listA 当前的数字
108.    lw      $s5,    0($t7)             #去除 listB 当前的数字
109.    beq     $s4,    $s5,    storeNum   #判断两个数字是否相同
110.
111.    beq     $t5,    $s2,    queryOutLoop #判断内部循环是否完成

```

```

112.    slt    $s3,    $t5,    $s2
113.    beq    $s3,    1,      queryInLoop    #未完成继续内部循环
114.
115.    #-----代码 输出-----#
116.
117.    outputInit:                                #输出循环初始化
118.    li      $v0,    4                          #打印提示信息，字符串
119.    la      $a0,    msgOutput                  #读取字符串地址，输出
120.    syscall
121.    beq     $t4,    0,      noSame              #判断是否有相同的数字
122.    add     $t0,    $zero, $zero                #初始化输出列表计数器$t0
123.    addi    $t1,    $t4,    1                  #初始化输出列表总数+1（左闭右
    开集合）
124.    addi    $t9,    $t8,    -4                #将 outList 首地址$t8 向后偏移
    -4 赋值给变量$t9
125.
126.    loopOutput:                                #输出循环
127.    addi    $t0,    $t0,    1                  #listB 计数器加 1
128.    addi    $t9,    $t9,    4                  #listB 指针偏移
129.
130.    beq     $t0,    $t1,    Quit              #判断输出循环是否结束
131.
132.    add     $s7,    $zero, $zero                #去重计数器
133.    addi    $s6,    $t8,    -4                #去重指针
134.
135.    duplicateRomoveLoop:                        #去重循环
136.    addi    $s7,    $s7,    1                  #去重计数器加 1
137.    beq     $s7,    $t0,    output            #输出当前数字
138.
139.    addi    $s6,    $s6,    4                  #去重指针偏移
140.    lw      $s4,    0($t9)                    #将当前数字存放在$s4 寄存器
    中
141.    lw      $s5,    0($s6)                    #将输出的数字放在$s5 寄存器中
    备用
142.
143.    beq     $s4,    $s5,    loopOutput        #判断当前的数字之前是否输出过，
    如果为假，就尝试输出下一个数字
144.    slt     $s3,    $s7,    $t0
145.    beq     $s3,    1,      duplicateRomoveLoop #如果为真，和下一个数字比较判断
    是否出现过
146.
147.    output:                                #输出去重后的数字
148.    li      $v0,    1                          #调用指令输出一个数字
149.    lw      $a0,    0($t9)

```



```
150.    syscall
151.
152.    la    $a0,    str                #调用指令输出一个空格
153.    li    $v0,    4
154.    syscall
155.
156.    slt    $s3,    $t0,    $t4
157.    beq    $s3,    1,    loopOutput    #判断所有数字是否输出完毕
158.
159. Quit:                                #退出
160.    li    $v0,    10
161.    syscall
162.
163. noSame:
164.    li    $v0,    4                #打印提示信息，字符串
165.    la    $a0,    msgNoSame        #读取字符串地址，输出
166.    syscall
167.    j      Quit
```