

# 《计算机组成原理实验》

(第四次) (答案)

厦门大学信息学院软件工程系 曾文华

2023年5月4日

# 目录

## 一、验证实验

1. MIPS汇编语言程序的运行
2. 在Logisim上运行MIPS程序
3. RISC-V汇编语言程序的运行
4. 在Logisim上运行RISC-V程序
5. Intel x86汇编语言程序的运行
6. ARMv7汇编语言程序的运行

## 二、设计实验

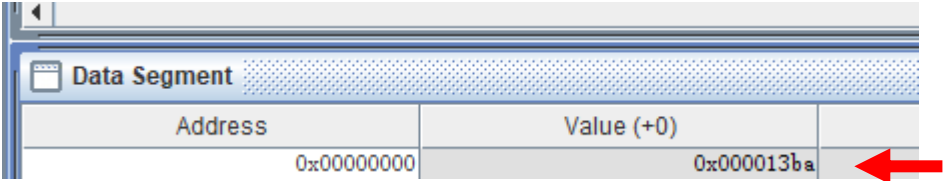
1. MIPS汇编语言程序设计（排序程序）
2. RISC-V汇编语言程序设计（排序程序）
3. Intel x86汇编语言程序设计（排序程序）

## 三、挑战性实验

- ARMv7汇编语言程序设计（排序程序）

- 请同学们修改“**sum\_mips.asm**”程序，计算 $1+2+3+\dots+100=5050=13bah$ ，然后在MARS 4.5汇编仿真器中运行，并将该程序的机器码保存到**sum100\_mips.hex**文件中。

答案



Data Segment	
Address	Value (+0)
0x00000000	0x000013ba

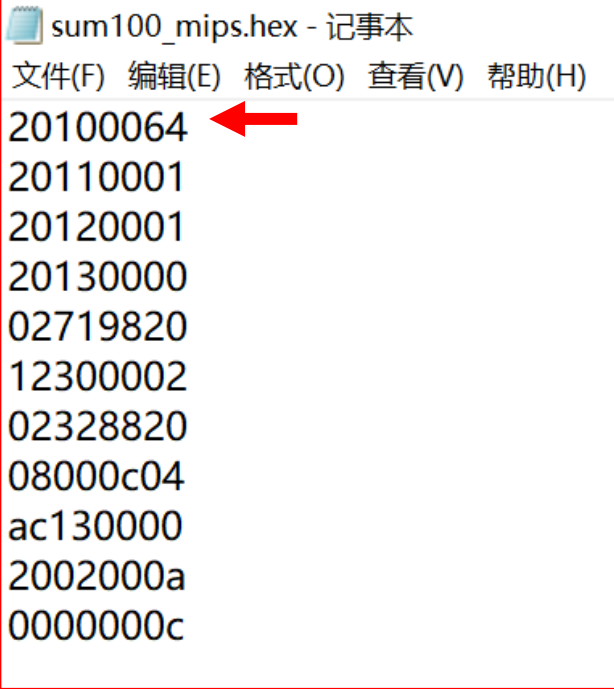
```
sum100_mips.asm - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#MIPS程序 求累加和 sum100_mips.asm
#求累加和: 1+2+.....+n, n的值为100 (可以改变), 累加和的结果存放到地址为0的数据存储器中

main:
    addi $s0,$zero,100      # n=100 -> s0
    addi $s1,$zero,1        # 1 -> s1
    addi $s2,$zero,1        # 1 -> s2
    addi $s3,$zero,0        # 0 -> s3

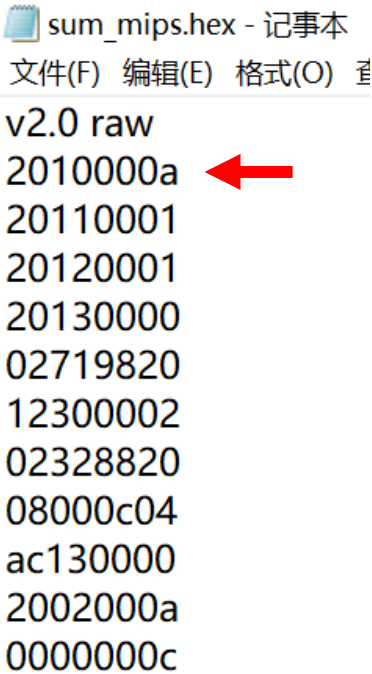
loop:
    add $s3,$s3,$s1         # s3+s1 -> s3
    beq $s1,$s0,finish      # 如果s1=s0, 则转finish
    add $s1,$s1,$s2         # s1+s2 -> s1
    j loop

finish:
    sw $s3,0($zero)         # s3 存到地址为0的存储单元中

    addi $v0,$zero,10       # 10号系统调用
    syscall                 # 程序退出
```



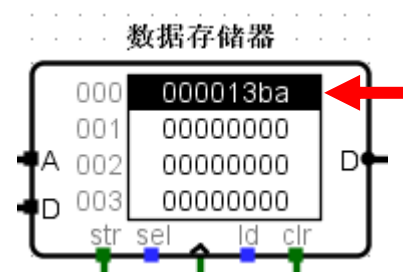
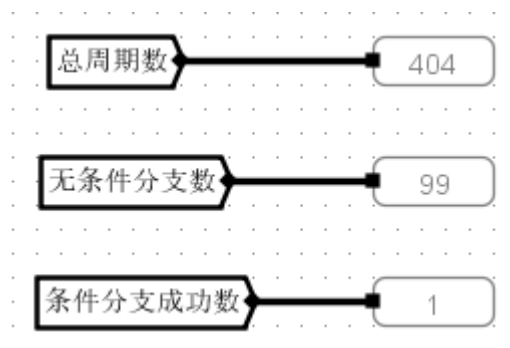
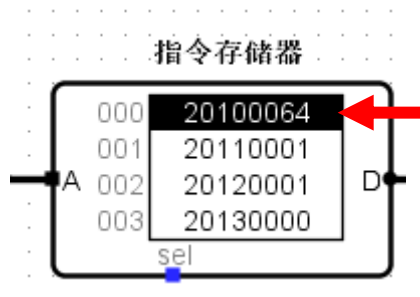
```
sum100_mips.hex - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
20100064
20110001
20120001
20130000
02719820
12300002
02328820
08000c04
ac130000
2002000a
0000000c
```



```
sum_mips.hex - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
v2.0 raw
2010000a
20110001
20120001
20130000
02719820
12300002
02328820
08000c04
ac130000
2002000a
0000000c
```

- 请同学们在Logisim实现的单周期MIPS处理器上，运行计算 $1+2+3+\dots+100=5050=13bah$ 的机器语言程序，观看数据存储器0号单元的值是不是**13bah**？程序运行的总周期数、无条件分支数、条件分支成功数分别是多少？

## 答案



- 请同学们修改sum\_riscv.asm程序，计算 $1+2+3+\dots+100=5050=13bah$ ，然后在RARS 1.5汇编仿真器中运行，并将该程序的机器码保存到sum100\_riscv.hex文件中。

## 答案

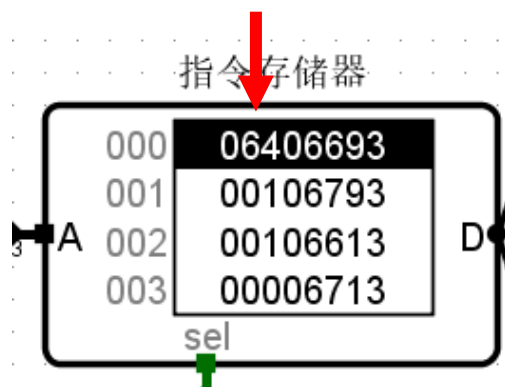
Data Segment	
Address	Value (+0)
0x00000000	0x000013ba

sum100\_riscv.hex - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
v20 raw  
06406693  
00106793  
00106613  
00006713  
00f70733  
00d78663  
00c787b3  
ff5ff06f  
00e02023  
0000006f

sum\_riscv.hex - 记事本  
文件(F) 编辑(E) 格式(O)  
v20 raw  
00a06693  
00106793  
00106613  
00006713  
00f70733  
00d78663  
00c787b3  
ff5ff06f  
00e02023  
0000006f

- 请同学们在Logisim实现的单周期RISC-V处理器上，运行计算 $1+2+3+\dots+100=5050=13bah$ 的程序，观看数据存储器0号单元的值是不是**13bah**？

答案



# 1、MIPS汇编语言程序设计实验

- 请使用单周期MIPS处理器的24条指令，编写**排序程序（设计实验）**
  - 假设有10个数据，这10个数据的顺序是随机的，例如：8、1、5、2、7、9、6、4、3、10；这10个数据是通过指令存放在数据存储器的第0、4、8、…、36号地址的存储单元中。
  - 要求：（1）对这10个数据按照**从大到小**的顺序（降序）进行排序，排序后的数据仍然放在数据存储器的第0、4、8、…、36号地址的存储单元中。（2）对这10个数据按照**从小到大**的顺序（升序）进行排序，排序后的数据仍然放在数据存储器的第0、4、8、…、36号地址的存储单元中。请分别编写程序。
  - 要求：编写好的程序先在MIPS汇编器（**MARS 4.5**）上运行通过；然后再到Logisim实现的单周期MIPS处理器（**单周期MIPS处理器.circ**）上运行通过，观察程序运行的总周期数、无条件分支数、条件分支成功数分别是多少，并进行分析。

核心指令（8条）			
序号	MIPS指令	RTL描述	功能说明
1	add \$rd,\$rs,\$rt	$R[rd] \leftarrow R[rs] + R[rt]$	寄存器加法指令，不考虑溢出
2	slt rd,\$rs,\$rt	$R[rd] \leftarrow (R[rs] < R[rt]) ? 1:0$	小于置1指令，有符号数比较
3	addi \$rt,\$rs,imm	$R[rt] \leftarrow R[rs] + \text{SignExt}(imm)$	立即数加法指令，不考虑溢出
4	lw \$rt,imm(\$rs)	$R[rt] \leftarrow M[R[rs] + \text{SignExt}(imm)]$	取数指令
5	sw \$rt,imm(\$rs)	$M[R[rs] + \text{SignExt}(imm)] \leftarrow R[rt]$	存数指令
6	beq \$rs,\$rt,imm	$\text{if}(R[rs] == R[rt]) \quad PC \leftarrow PC + 4 + \text{SignExt}(imm) < 2$	条件分支指令：如果rs == rt，则跳转
7	bne \$rs,\$rt,imm	$\text{if}(R[rs] != R[rt]) \quad PC \leftarrow PC + 4 + \text{SignExt}(imm) < 2$	条件分支指令：如果rs != rt，则跳转
8	syscall	系统调用指令，用于停机	系统调用指令，用于停机

24条指令

基础指令（16条）			
序号	MIPS指令	RTL描述	功能说明
1	addu \$rd,\$rs,\$rt	$R[rd] \leftarrow R[rs] + R[rt]$	无符号寄存器加法指令
2	addiu \$rd,\$rs,imm	$R[rd] \leftarrow R[rs] + \text{SignExt}(imm)$	无符号立即数加法指令
3	and \$rd,\$rs,\$rt	$R[rd] \leftarrow R[rs] \& R[rt]$	逻辑与指令
4	andi \$rd,\$rs,imm	$R[rd] \leftarrow R[rs] \& (16\text{个}0, imm)$	立即数逻辑与指令
5	slti \$rd,\$rs,imm	$R[rd] \leftarrow R[rs] < \text{short}$	逻辑左移
6	slti \$rd,\$rs,imm	$R[rd] \leftarrow R[rs] > \text{short}$	逻辑左移
7	sll \$rd,\$rs,imm	$R[rd] \leftarrow R[rs] >> \text{short}$	逻辑左移
8	srl \$rd,\$rs,\$rt	$R[rd] \leftarrow R[rs] \gg R[rt]$	寄存器移位指令
9	srl \$rd,\$rs,\$rt	$R[rd] \leftarrow R[rs] \gg R[rt]$	逻辑右移指令
10	sllt \$rd,\$rs,\$rt	$R[rd] \leftarrow (R[rs] < R[rt]) ? 1:0$	立即数逻辑左移指令
11	sllt \$rd,\$rs,\$rt	$R[rd] \leftarrow (R[rs] < R[rt]) ? 1:0$	逻辑右移指令
12	sra \$rd,\$rs,\$rt	$R[rd] \leftarrow (R[rs] < R[rt]) ? 1:0$	小于置1指令，无符号数比较
13	sra \$rd,\$rs,\$rt	$R[rd] \leftarrow (R[rs] < \text{SignExt}(imm)) ? 1:0$	小于置1指令，无符号数比较
14	j address	$PC \leftarrow (PC \> 4)_{12-31} \& address < 2$	无条件分支指令
15	jal address	$PC \leftarrow PC + 4 \quad PC \leftarrow (PC \> 4)_{12-31} \& address < 2$	子程序调用指令
16	jr \$rs	$PC \leftarrow R[rs]$	寄存器跳转指令

# 答案

sort1\_mips.asm - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

#MIPS程序 冒泡法排序（降序排列，从大到小） sort1\_mips.asm  
#先将10个数（8、1、5、2、7、9、6、4、3、10）存放地址为0开始的数据存储器中，然后对这10个数进行排序。

main:  
    addi \$s0,\$zero,8                   #第1个数=8（可以修改）保存到(0)  
    sw \$s0,0(\$zero)  
    addi \$s0,\$zero,1                   #第2个数=1（可以修改）保存到(4)  
    sw \$s0,4(\$zero)  
    addi \$s0,\$zero,5                   #第3个数=5（可以修改）保存到(8)  
    sw \$s0,8(\$zero)  
    addi \$s0,\$zero,2                   #第4个数=2（可以修改）保存到(12)  
    sw \$s0,12(\$zero)  
    addi \$s0,\$zero,7                   #第5个数=7（可以修改）保存到(16)  
    sw \$s0,16(\$zero)  
    addi \$s0,\$zero,9                   #第6个数=9（可以修改）保存到(20)  
    sw \$s0,20(\$zero)  
    addi \$s0,\$zero,6                   #第7个数=6（可以修改）保存到(24)  
    sw \$s0,24(\$zero)  
    addi \$s0,\$zero,4                   #第8个数=4（可以修改）保存到(28)  
    sw \$s0,28(\$zero)  
    addi \$s0,\$zero,3                   #第9个数=3（可以修改）保存到(32)  
    sw \$s0,32(\$zero)  
    addi \$s0,\$zero,10                  #第10个数=10（可以修改）保存到(36)  
    sw \$s0,36(\$zero)  
  
    addi \$s0,\$zero,0                   #s0=0                   排序区间开始地址  
    addi \$s1,\$zero,36                  #s1=36=10\*4-4       排序区间结束地址       10个数       如果不是10个数，这里要修改，例如20个数，这里修改为76  
  
|  
sort\_loop:  
    lw \$s3,0(\$s0)                   #\$s3=(\$s0)  
    lw \$s4,0(\$s1)                   #\$s4=(\$s1)  
    slt \$t0,\$s3,\$s4                  #如果s3<s4，则置t0=1；否则，置t0=0       降序排序       从大到小  
    beq \$t0,\$zero,sort\_next        #如果t0=0，则转sort\_nent  
    sw \$s3,0(\$s1)                   #交换(\$s0)和(\$s1)  
    sw \$s4,0(\$s0)                   #交换(\$s0)和(\$s1)  
  
sort\_next:  
    addi \$s1,\$s1,-4                  # \$s1-4 -> \$s1  
    bne \$s0,\$s1,sort\_loop           #如果s0不等于s1，则转sort\_loop  
    addi \$s0,\$s0,4                   #\$s0+4 -> \$s0  
    addi \$s1,\$zero,36                #s1=36=10\*4-4       排序区间结束地址       10个数       如果不是10个数，这里要修改，例如20个数，这里修改为76  
    bne \$s0,\$s1,sort\_loop           #如果s0不等于s1，则转sort\_loop  
  
    addi \$v0,\$zero,10                # 10号系统调用  
    syscall                          # 程序退出



# 答案

置10个数，并保存到存储器中

main:

```
addi $s0,$zero,8
sw $s0,0($zero)
addi $s0,$zero,1
sw $s0,4($zero)
addi $s0,$zero,5
sw $s0,8($zero)
addi $s0,$zero,2
sw $s0,12($zero)
addi $s0,$zero,7
sw $s0,16($zero)
addi $s0,$zero,9
sw $s0,20($zero)
addi $s0,$zero,6
sw $s0,24($zero)
addi $s0,$zero,4
sw $s0,28($zero)
addi $s0,$zero,3
sw $s0,32($zero)
addi $s0,$zero,10
sw $s0,36($zero)
```

#第1个数=8 (可以修改) 保存到(0)

#第2个数=1 (可以修改) 保存到(4)

#第3个数=5 (可以修改) 保存到(8)

#第4个数=2 (可以修改) 保存到(12)

#第5个数=7 (可以修改) 保存到(16)

#第6个数=9 (可以修改) 保存到(20)

#第7个数=6 (可以修改) 保存到(24)

#第8个数=4 (可以修改) 保存到(28)

#第9个数=3 (可以修改) 保存到(32)

#第10个数=10 (可以修改) 保存到(36)

# 答案

## 从大到小排序

addi \$s0,\$zero,0 addi \$s1,\$zero,36	#\$s0=0 #\$s1=36=10*4-4	排序区间开始地址 排序区间结束地址	10个数	如果不是10个数，这里要修改，例如20个数，这里修改为76
sort_loop: lw \$s3,0(\$s0) lw \$s4,0(\$s1) → slt \$t0,\$s3,\$s4 beq \$t0,\$zero,sort_next sw \$s3,0(\$s1) sw \$s4,0(\$s0)	#\$s3=(\$s0) #\$s4=(\$s1) #如果\$s3<\$s4，则置\$t0=1；否则，置\$t0=0 #如果\$t0=0，则转sort_next #交换(\$s0)和(\$s1) #交换(\$s0)和(\$s1)		降序排序	从大到小
sort_next: addi \$s1,\$s1,-4 bne \$s0,\$s1,sort_loop addi \$s0,\$s0,4 addi \$s1,\$zero,36 bne \$s0,\$s1,sort_loop	# \$s1-4 -> \$s1 #如果\$s0不等于\$s1，则转sort_loop #\$s0+4 -> \$s0 #\$s1=36=10*4-4 #如果\$s0不等于\$s1，则转sort_loop	排序区间结束地址	10个数	如果不是10个数，这里要修改，例如20个数，这里修改为76
addi \$v0,\$zero,10 syscall	# 10号系统调用 # 程序退出			

程序退出

# 答案

sort2\_mips.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

#MIPS程序 冒泡法排序(升序排列,从小到大) sort2\_mips.asm

#先将10个数(8、1、5、2、7、9、6、4、3、10)存放地址为0开始的数据存储器中,然后对这10个数进行排序。

main:

```
addi $s0,$zero,8      #第1个数=8 (可以修改) 保存到(0)
sw $s0,0($zero)
addi $s0,$zero,1      #第2个数=1 (可以修改) 保存到(4)
sw $s0,4($zero)
addi $s0,$zero,5      #第3个数=5 (可以修改) 保存到(8)
sw $s0,8($zero)
addi $s0,$zero,2      #第4个数=2 (可以修改) 保存到(12)
sw $s0,12($zero)
addi $s0,$zero,7      #第5个数=7 (可以修改) 保存到(16)
sw $s0,16($zero)
addi $s0,$zero,9      #第6个数=9 (可以修改) 保存到(20)
sw $s0,20($zero)
addi $s0,$zero,6      #第7个数=6 (可以修改) 保存到(24)
sw $s0,24($zero)
addi $s0,$zero,4      #第8个数=4 (可以修改) 保存到(28)
sw $s0,28($zero)
addi $s0,$zero,3      #第9个数=3 (可以修改) 保存到(32)
sw $s0,32($zero)
addi $s0,$zero,10     #第10个数=10 (可以修改) 保存到(36)
sw $s0,36($zero)
```

```
addi $s0,$zero,0      #s0=0      排序区间开始地址
addi $s1,$zero,36     #s1=36=10*4-4 排序区间结束地址      10个数      如果不是10个数,这里要修改,例如20个数,这里修改为76
```

sort\_loop:

```
lw $s3,0($s0)          #s3=(s0)
lw $s4,0($s1)          #s4=(s1)
slt $t0,$s4,$s3        #如果s4<s3,则置t0=1;否则,置t0=0      升序排序      从小到大
beq $t0,$zero,sort_next #如果t0=0,则转sort_next
sw $s3,0($s1)          #交换(s0)和(s1)
sw $s4,0($s0)          #交换(s0)和(s1)
```

sort\_next:

```
addi $s1,$s1,-4        # $s1-4 -> $s1
bne $s0,$s1,sort_loop  #如果s0不等于s1,则转sort_loop
addi $s0,$s0,4          #s0+4 -> $s0
addi $s1,$zero,36      #s1=36=10*4-4 排序区间结束地址      10个数      如果不是10个数,这里要修改,例如20个数,这里修改为76
bne $s0,$s1,sort_loop  #如果s0不等于s1,则转sort_loop
```

```
addi $v0,$zero,10      # 10号系统调用
syscall                # 程序退出
```

升序排列,从小到大

就这个地方不一样!

# 答案

## MARS 4.5上的运行结果

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x00000000	0x0000000a	0x00000009	0x00000008	0x00000007	0x00000006	0x00000005	0x00000004	0x00000003	
0x00000020	0x00000002	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x00000000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007	0x00000008	
0x00000020	0x00000009	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	

总周期数	340
无条件分支数	0
条件分支成功数	79

## Logisim上的运行结果

000 0000000a 00000009 00000008 00000007 00000006 00000005 00000004 00000003 00000002 00000001 00000000 00000000 00000000 00000000 00000000 00000000

总周期数	338
无条件分支数	0
条件分支成功数	80

000 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 0000000a 00000000 00000000 00000000 00000000 00000000 00000000

## 2、RISC-V汇编语言程序设计实验

- 请使用RISC-V处理器的9条指令，编写**排序程序（设计实验）**
  - 假设有10个数据，这10个数据的顺序是随机的，例如：8、1、5、2、7、9、6、4、3、10；这10个数据是通过指令存放在数据存储器的第0、4、8、…、36号地址的存储单元中。
  - 要求：（1）对这10个数据按照**从大到小**的顺序（降序）进行排序，排序后的数据仍然放在数据存储器的第0、4、8、…、36号地址的存储单元中。（2）对这10个数据按照**从小到大**的顺序（升序）进行排序，排序后的数据仍然放在数据存储器的第0、4、8、…、36号地址的存储单元中。请分别编写程序。
  - 要求：编写好的程序先在RISC-V汇编器（**RARS 1.5**）上运行通过；然后再到Logisim实现的单周期RISC-V处理器（**单周期RISC-V处理器.circ**）上运行通过。

• RISC-V核心指令集RV32I的9条指令（我们设计的RISC-V单周期处理器仅支持该9条指令）：

- |                     |  |
|---------------------|--|
| ① add rd,rs1,rs2    | ; rs1+rs2->rd                          |
| ② slt rd,rs1,rs2    | ; 带符号数比较指令 if rs1<rs2 1->rd else 0->rd |
| ③ sltu rd,rs1,rs2   | ; 无符号数比较指令 if rs1<rs2 1->rd else 0->rd |
| ④ ori rd,rs1,imm12  | ; rs1 或 imm12 -> rd                    |
| ⑤ lw rd,rs1,imm12   | ; M[rs1+imm12] -> rd                   |
| ⑥ lui rd,imm20      | ; imm20 -> rd                          |
| ⑦ sw rs2,rs1,imm12  | ; rs2 -> M[rs1+imm12]                  |
| ⑧ beq rs1,rs2,imm12 | ; if rs1=rs2 then goto imm12           |
| ⑨ jal rd,imm20      | ; goto imm20                           |

# 答案

sort1\_riscv.asm - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

#RISC-V冒泡方法排序程序 sort1\_riscv.asm 将n个数据进行排序 (从大到小, 降序排列)  
# n为数据个数 (可以改变) n个数据由程序设置存放在地址为0、4、8、12、.....、36的数据存储器中  
# 排列好的数据仍然放在地址为0、4、8、12、.....、36的数据存储器中

```
main:
    ori    a2,zero,10          #a2=10=n

    ori    a1, zero, 8          # a1 = 0
    sw     a1, 0(zero)          # 0 -> (0)
    ori    a1, zero, 1          # a1 = 1
    sw     a1, 4(zero)          # 1 -> (4)
    ori    a1, zero, 5          # a1 = 5
    sw     a1, 8(zero)          # 0 -> (8)
    ori    a1, zero, 2          # a1 = 2
    sw     a1, 12(zero)         # 1 -> (12)
    ori    a1, zero, 7          # a1 = 7
    sw     a1, 16(zero)         # 0 -> (16)
    ori    a1, zero, 9          # a1 = 9
    sw     a1, 20(zero)         # 1 -> (20)
    ori    a1, zero, 6          # a1 = 6
    sw     a1, 24(zero)         # 0 -> (24)
    ori    a1, zero, 4          # a1 = 4
    sw     a1, 28(zero)         # 1 -> (28)
    ori    a1, zero, 3          # a1 = 3
    sw     a1, 32(zero)         # 0 -> (32)
    ori    a1, zero, 10         # a1 = 10
    sw     a1, 36(zero)         # 1 -> (36)
```

## 降序排列，从大到小

```
    ori    a4,zero,1
    ori    a5,zero,4
    ori    a6,zero,-1

loop1:
    beq     a2,a4,finish
    ori     a3,zero,1
    ori     a7,zero,0
    ori     s8,zero,4

loop2:
    sltu    s11,a3,a2
    beq     s11,zero,loop3
    lw      s9,0(a7)
    lw      s10,0(s8)
    sltu    s11,s10,s9
    beq     s11,a4,loop4
    sw      s10,0(a7)
    sw      s9,0(s8)
    jal     zero,loop4

loop3:
    add     a2,a2,a6
    jal     zero,loop1

loop4:
    add     a3,a3,a4
    add     a7,a7,a5
    add     s8,s8,a5
    jal     zero,loop2

finish:
    jal     zero,finish
```

#降序排列，从大到小

# 答案

## 升序排列，从小到大

```
sort2_riscv.asm - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#RISC-V冒泡方法排序程序      sort2_riscv.asm      将n个数据进行排序 (从小到大, 升序排列)
# n为数据个数 (可以改变)      n个数据由程序设置存放在地址为0、4、8、12、.....、36的数据存储器中
# 排列好的数据仍然放在地址为0、4、8、12、.....、36的数据存储器中
```

```
main:
    ori    a2,zero,10                #a2=10=n

    ori    a1, zero, 8                # a1 = 0
    sw     a1, 0(zero)                # 0 -> (0)
    ori    a1, zero, 1                # a1 = 1
    sw     a1, 4(zero)                # 1 -> (4)
    ori    a1, zero, 5                # a1 = 5
    sw     a1, 8(zero)                # 0 -> (8)
    ori    a1, zero, 2                # a1 = 2
    sw     a1, 12(zero)               # 1 -> (12)
    ori    a1, zero, 7                # a1 = 7
    sw     a1, 16(zero)               # 0 -> (16)
    ori    a1, zero, 9                # a1 = 9
    sw     a1, 20(zero)               # 1 -> (20)
    ori    a1, zero, 6                # a1 = 6
    sw     a1, 24(zero)               # 0 -> (24)
    ori    a1, zero, 4                # a1 = 4
    sw     a1, 28(zero)               # 1 -> (28)
    ori    a1, zero, 3                # a1 = 3
    sw     a1, 32(zero)               # 0 -> (32)
    ori    a1, zero, 10               # a1 = 10
    sw     a1, 36(zero)               # 1 -> (36)
```

```

    ori    a4,zero,1
    ori    a5,zero,4
    ori    a6,zero,-1

loop1:
    beq     a2,a4,finish
    ori     a3,zero,1
    ori     a7,zero,0
    ori     s8,zero,4

loop2:
    sltu    s11,a3,a2
    beq     s11,zero,loop3
    lw      s9,0(a7)
    lw      s10,0(s8)
    sltu    s11,s9,s10
    beq     s11,a4,loop4
    sw      s10,0(a7)
    sw      s9,0(s8)
    jal     zero,loop4

loop3:
    add     a2,a2,a6
    jal     zero,loop1

loop4:
    add     a3,a3,a4
    add     a7,a7,a5
    add     s8,s8,a5
    jal     zero,loop2

finish:
    jal     zero,finish
```

#升序排列，从小到大

# 答案

## RARS 1.5上的运行结果

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x0000000a	0x00000009	0x00000008	0x00000007	0x00000006	0x00000005	0x00000004	0x00000003
0x00000020	0x00000002	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007	0x00000008
0x00000020	0x00000009	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

## Logisim上的运行结果

```
000000 0000000a 00000009 00000008 00000007 00000006 00000005 00000004 00000003 00000002 00000001 00000000 00000000 00000000 00000000 00000000
000000 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 0000000a 00000000 00000000 00000000 00000000 00000000
```



# 3、Intel x86汇编语言程序设计实验

- 请使用Intel x86的指令，编写**排序程序（设计实验）**
  - 假设有10个数据，这10个数据的顺序是随机的，例如：8、1、5、2、7、9、6、4、3、10。
  - 要求：（1）对这10个数据按照**从大到小**的顺序（降序）进行排序。（2）对这10个数据按照**从小到大**的顺序（升序）进行排序。请分别编写程序。
- 在masm32汇编工具上运行上述2个程序。

```
.486 ; create 32 bit code
.model flat, stdcall ; 32 bit memory model
option casemap :none ; case sensitive
include \masm32\macros\macros.asm
includelib \masm32\lib\masm32.lib
includelib \masm32\lib\gdi32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\wsock32.lib
includelib \masm32\lib\msvcrt.lib
include \masm32\include\msvcrt.inc
include \masm32\include\masm32.inc
include \masm32\include\gdi32.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
```

```
.data
arr dd 8, 1, 5, 2, 7, 9, 6, 4, 3, 10 ; 10个原始数据, 数据可以改变, 排好后, 还放在这里
len1 byte ?
len2 byte ?
fmt byte '%d ',0
```

```
.code
main: ;-----显示原始数据-----
mov len1,lengthof arr ;获取数据长度
mov ebx,offset arr ;获取数据的起始地址
xor ecx,ecx
mov al,len1
```

```
p1:
movsx ebx,al
cmp ecx,ebx
jnb fina1
mov edx, arr[(type arr)*ecx]
pushad
invoke crt_printf,offset fmt,edx ;显示一个数据
popad
inc ecx
jmp p1
```

```
fina1:
print chr$(" ",13,10) ;显示回车、换行
print chr$(" ",13,10) ;显示回车、换行
```

```
;-----排序-----
mov len1,lengthof arr ;数据长度 -> len1
mov ebx,offset arr ;数据偏移地址(首地址) -> ebx
mov al,0h ; al=0
```

答案

降序排列, 从大到小

运行结果

```
C:\x86>sort1_x86.exe
8 1 5 2 7 9 6 4 3 10
10 9 8 7 6 5 4 3 2 1
C:\x86>
```

```
lp:
cmp al,len1
jnb done ;排序结束, 转done

inner:
mov ah, 1h ; ah=1

mov cl,len1 ; cl <- len1=数据长度
mov len2,cl ; len2 <- cl
sub len2,al ; len2 <- len2-al
cmp ah,len2 ; 比较ah 和 len2
jnb last ; 跳出内循环
movsx esi,ah ; esi <- ah
mov bl,ah ; bl = ah
sub bl,1 ; bl < 1 bl-1
movsx edi,bl ; edi <- bl
mov ecx, arr[(type arr)*esi] ;ecx <- [esi]
mov edx, arr[(type arr)*edi] ;edx <_ [edi]
cmp ecx,edx ;比较ecx和edx
jb follow ;小于则转 降序排序 从大到小
mov edx,arr[(type arr)*esi] ;edx <- [esi]
xchg edx,arr[(type arr)*edi] ;edx与[edi]交换
xchg edx,arr[(type arr)*esi] ;edx与[esi]交换

follow:
inc ah ; ah +1 -> ah
jmp inner ; 转内循环

last:
inc al ; al+1 -> al
jmp lp ;转外循环

;-----显示排序后的数据-----
done:
xor ecx,ecx
mov al,len1

prt:
movsx ebx,al
cmp ecx,ebx
jnb fina
mov edx, arr[(type arr)*ecx]
pushad
invoke crt_printf,offset fmt,edx
popad
inc ecx
jmp prt

fina:
exit

end main
```

```
.486 ; create 32 bit code
.model flat, stdcall ; 32 bit memory model
option casemap :none ; case sensitive
include \masm32\macros\macros.asm
includelib \masm32\lib\masm32.lib
includelib \masm32\lib\gdi32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\wsock32.lib
includelib \masm32\lib\msvcrt.lib
include \masm32\include\msvcrt.inc
include \masm32\include\masm32.inc
include \masm32\include\gdi32.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
```

```
.data
arr dd 8, 1, 5, 2, 7, 9, 6, 4, 3, 10 ; 10个原始数据, 数据可以改变, 排好后, 还放在这里
len1 byte ?
len2 byte ?
fmt byte '%d ',0
```

```
.code
main: ;-----显示原始数据-----
mov len1,lengthof arr ;获取数据长度
mov ebx,offset arr ;获取数据的起始地址
xor ecx,ecx
mov al,len1
```

```
prt1:
movsx ebx,al
cmp ecx,ebx
jnb fina1
mov edx, arr[(type arr)*ecx]
pushad
invoke crt_printf,offset fmt,edx ;显示一个数据
popad
inc ecx
jmp prt1
```

```
fina1:
print chr$(" ",13,10) ;显示回车、换行
print chr$(" ",13,10) ;显示回车、换行
```

```
;-----排序-----
mov len1,lengthof arr ;数据长度 -> len1
mov ebx,offset arr ;数据偏移地址(首地址) -> ebx
mov al,0h
```

答案

升序排列, 从小到大

运行结果

```
C:\x86>sort2_x86.exe
8 1 5 2 7 9 6 4 3 10
1 2 3 4 5 6 7 8 9 10
C:\x86>
```

```
lp:
cmp al,len1
jnb done ;排序结束, 转done

mov ah, 1h ; ah=1

inner:
mov cl,len1 ; cl <- len1=数据长度
mov len2,cl ; len2 <- cl
sub len2,al ; len2 <- len2-al
cmp ah,len2 ; 比较ah 和 len2
jnb last ; 跳出内循环
movsx esi,ah ; esi <- ah
mov bl,ah ; bl = ah
sub bl,1 ; bl < 1 bl-1
movsx edi,bl ; edi <- bl
mov ecx, arr[(type arr)*esi] ;ecx <- [esi]
mov edx, arr[(type arr)*edi] ;edx <_ [edi]
cmp ecx,edx ;比较ecx和edx
jnb follow ;大于等于则转
mov edx,arr[(type arr)*esi] ;edx <- [esi]
xchg edx,arr[(type arr)*edi] ;edx与[edi]交换
xchg edx,arr[(type arr)*esi] ;edx与[esi]交换

follow:
inc ah ; ah +1 -> ah
jmp inner ; 转内循环

last:
inc al ; al+1 -> al
jmp lp ;转外循环
```

```
;-----显示排序后的数据-----
done:
xor ecx,ecx
mov al,len1

prt:
movsx ebx,al
cmp ecx,ebx
jnb fina
mov edx, arr[(type arr)*ecx]
pushad
invoke crt_printf,offset fmt,edx
popad
inc ecx
jmp prt

fina:
exit

end main
```

# ARMv7汇编语言程序设计实验

- 请使用ARMv7的指令，编写**排序程序（挑战性实验）**
  - 假设有10个数据，这10个数据的顺序是随机的，例如：8、1、5、2、7、9、6、4、3、10。
  - 要求：（1）对这10个数据按照**从大到小**的顺序（降序）进行排序。（2）对这10个数据按照**从小到大**的顺序（升序）进行排序。请分别编写程序。
- 在ARMv7汇编工具上运行上述2个程序。

#ARMv7 冒泡方法排序程序      sort1\_armv7.s      将n个数据进行排序      从大到小, 降序排列  
 # n存放在地址为0x100的存储器中      | n个数据存放在地址为0x200、0x204、0x208、.....的存储器中  
 # 排列好的数据也放在地址为0x200、0x204、0x208、.....的存储器中

```
.global _start
_start:
.org 0

    ldr r0,=num
    ldr r1,=result

    ldr r9,[r0]

    lsl r9,r9,#2
    add r8,r9,#-4

    mov r4,#0
    mov r5,r8
```

# 答案

## 降序排列，从大到小

```
sort_loop:
    add r10,r4,r1
    add r11,r5,r1
    ldr r6,[r10]
    ldr r7,[r11]
    cmp r7,r6
    blt sort_next
    add r10,r4,r1
    add r11,r5,r1
    str r6,[r11]
    str r7,[r10]

sort_next:
    add r5,r5,#-4
    cmp r4,r5
    bne sort_loop
    add r4,r4,#4
    mov r5,r8
    cmp r4,r5
    bne sort_loop

end:
    b end

.org 0x100
num:
    .word 10

.org 0x200
result:
    .word 8, 1, 5, 2, 7, 9, 6, 4, 3, 10
```

@读取第j个元素  
 @读取第j+1个元素  
 @ 从大到小, 降序排列

@交换存储  
 @交换存储

## 运行结果

		result:	
00000200	0000000a	andeq	r0, r0, r10
00000204	00000009	andeq	r0, r0, r9
00000208	00000008	andeq	r0, r0, r8
0000020c	00000007	andeq	r0, r0, r7
00000210	00000006	andeq	r0, r0, r6
00000214	00000005	andeq	r0, r0, r5
00000218	00000004	andeq	r0, r0, r4
0000021c	00000003	andeq	r0, r0, r3
00000220	00000002	andeq	r0, r0, r2
00000224	00000001	andeq	r0, r0, r1

sort2\_armv7.s - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
#ARMv7 冒泡方法排序程序      sort2\_armv7.s      将n个数据进行排序      从小到大, 升序排列  
# n存放在地址为0x100的存储器中      n个数据存放在地址为0x200、0x204、0x208、.....的存储器中  
# 排列好的数据也放在地址为0x200、0x204、0x208、.....的存储器中

```
.global _start
_start:
.org 0

    ldr r0,=num
    ldr r1,=result

    ldr r9,[r0]

    lsl r9,r9,#2
    add r8,r9,#-4

    mov r4,#0
    mov r5,r8
```

```
sort_loop:
    add r10,r4,r1
    add r11,r5,r1
    ldr r6,[r10]
    ldr r7,[r11]
    cmp r6,r7
    blt sort_next
    add r10,r4,r1
    add r11,r5,r1
    str r6,[r11]
    str r7,[r10]

sort_next:
    add r5,r5,#-4
    cmp r4,r5
    bne sort_loop
    add r4,r4,#4
    mov r5,r8
    cmp r4,r5
    bne sort_loop

end:
    b end
```

```
.org 0x100
num:
    .word 10

.org 0x200
result:
    .word 8, 1, 5, 2, 7, 9, 6, 4, 3, 10
```

答案

升序排列，从小到大

运行结果

00000200	00000001	andeq	r0, r0, r1
00000204	00000002	andeq	r0, r0, r2
00000208	00000003	andeq	r0, r0, r3
0000020c	00000004	andeq	r0, r0, r4
00000210	00000005	andeq	r0, r0, r5
00000214	00000006	andeq	r0, r0, r6
00000218	00000007	andeq	r0, r0, r7
0000021c	00000008	andeq	r0, r0, r8
00000220	00000009	andeq	r0, r0, r9
00000224	0000000a	andeq	r0, r0, r10

**Thanks**