

同济大学计算机系

计算机组成原理实验报告



学 号 2151140

姓 名 王谦

专 业 信息安全

授课老师 张冬冬

一、实验内容

实验介绍：

本实验通过编写几个汇编小程序来帮助各位熟悉常用的 MIPS 汇编指令

实验目标：

1. 学习使用 MARS 模拟器
2. 熟悉常用的 MIPS 指令
3. 编写几个 MIPS 汇编程序
 - a) Fibonacci 数列
 - b) 冒泡排序
 - c) Booth 乘法

实验原理：

1. MIPS 汇编基本格式
 - a) 代码段由.text 开头
 - b) 数据段以.data 开头（本次实验可以不适用数据段）
 - c) 跳转标记格式如“label:”，为标记名+冒号
2. MARS 是一个 MIPS 模拟器，可以使用其来编写并调试 MIPS 汇编程序
3. MIPS 程序要求
 - a) Fibonacci 数列：将\$2,\$3 寄存器初始化为 fibonacci 数列的前两个数 0, 1；\$4 为数列中所需得到的数字的序号（\$4=4 即表示得到第四个 fibonacci 数）；最后得到的结果存入\$1
 - b) 将一串数列输入\$2-\$6，用冒泡排序算法对其进行排序
 - c) 运用布斯乘法算法实现两个数的乘法，结果用两个寄存器表示，具体算法可参考 wikipedia 上的相关词条

PS：由于 MIPS 的一些默认操作会改变\$1 的值，所以运算时尽量不要使用\$1
4. 我们必须仅使用以下指令来编写 MIPS 的汇编程序。

Mnemonic Symbol	Format						Sample
Bit #	31..26	25..21	20..16	15..11	10..6	5..0	
R-type	op	rs	rt	rd	shamt	func	
add	000000	rs	rt	rd	0	100000	add \$1,\$2,\$3
addu	000000	rs	rt	rd	0	100001	addu \$1,\$2,\$3
sub	000000	rs	rt	rd	0	100010	sub

							\$1,\$2,\$3
subu	000000	rs	rt	rd	0	100011	subu \$1,\$2,\$3
and	000000	rs	rt	rd	0	100100	and \$1,\$2,\$3
or	000000	rs	rt	rd	0	100101	or \$1,\$2,\$3
xor	000000	rs	rt	rd	0	100110	xor \$1,\$2,\$3
nor	000000	rs	rt	rd	0	100111	nor \$1,\$2,\$3
slt	000000	rs	rt	rd	0	101010	slt \$1,\$2,\$3
sltu	000000	rs	rt	rd	0	101011	sltu \$1,\$2,\$3
sll	000000	0	rt	rd	shamt	000000	sll \$1,\$2,10
srl	000000	0	rt	rd	shamt	000010	srl \$1,\$2,10
sra	000000	0	rt	rd	shamt	000011	sra \$1,\$2,10
sllv	000000	rs	rt	rd	0	000100	sllv \$1,\$2,\$3
srlv	000000	rs	rt	rd	0	000110	srlv \$1,\$2,\$3
srav	000000	rs	rt	rd	0	000111	srav \$1,\$2,\$3
jr	000000	rs	0	0	0	001000	jr \$31

Bit #	31..26	25..21	20..16	15..0	
I-type	op	rs	rt	immediate	
addi	001000	rs	rt	immediate	addi \$1,\$2,100
addiu	001001	rs	rt	immediate	addiu \$1,\$2,100
andi	001100	rs	rt	immediate	andi \$1,\$2,10
ori	001101	rs	rt	immediate	andi \$1,\$2,10
xori	001110	rs	rt	immediate	andi \$1,\$2,10
lw	100011	rs	rt	immediate	lw \$1,10(\$2)
sw	101011	rs	rt	immediate	sw

					\$1,10(\$2)
beq	000100	rs	rt	immediate	beq \$1,\$2,10
bne	000101	rs	rt	immediate	bne \$1,\$2,10
slti	001010	rs	rt	immediate	slti \$1,\$2,10
sltiu	001011	rs	rt	immediate	sltiu \$1,\$2,10
lui	001111	00000	rt	immediate	Lui \$1, 10
Bit #	31..26	25..0			
J-type	op	Index			
j	000010	address			j 10000
jal	000011	address			jal 10000

实验步骤：

1. 下载并打开 MARS
2. 在 MARS 中编写汇编程序
3. 运行并调试汇编程序

二、逻辑图

—— 本次实验不要求 ——

三、汇编代码

Fibonacci

```
.text
    #0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89
    #初始化
    addiu $2,$0,0
    addiu $3,$0,1

    #设定第几个数
    addiu $4,$0,11 #默认大于 1
    beq $4,$0,end

    and $5,$4,1    #if($4 mod 2 == 1) $5 = 0
    srl $4,$4,1    #$4-- 判断循环次数，每次循环得到后面两个值

loop:
    beq $4,$0,main #if($4 == 0) goto end
```

```

    addiu $4,$4,-1 #$4--
    add $6,$2,$3 #6 = 2 + 3
    add $7,$3,$6 #7 = 3 + 6
    addiu $2,$6,0 #2 = 6
    addiu $3,$7,0 #3 = 7
    j loop

```

main:

```

    addiu $1,$2,0 #偶数 $2 => $1
    beq $5,$0,end
    addiu $1,$3,0 #奇数 $3 => $1

```

end:

Bubble_Sort

.text

```

    #待排序
    addiu $2,$0,13
    addiu $3,$0,11
    addiu $4,$0,7
    addiu $5,$0,5
    addiu $6,$0,3

```

```

    #辅助参数
    addiu $10,$0,1
    addiu $12,$0,5
    addiu $13,$0,4
    addiu $14,$0,3

```

loop:

```

    beq $10,$12,end #if($10 == $12) goto end
    addiu $10,$10,1 #$10++
    slt $7,$3,$2 #if($3 < $2) continue
    beq $7,$0,loop1 #if($3 >= $2) goto loop1
    addiu $9,$2,0 #交换 23
    addiu $2,$3,0
    addiu $3,$9,0

```

loop1:

```

    beq $10,$12,loop #if($10 == $12) goto loop
    slt $7,$4,$3 #if($4 < $3) continue
    beq $7,$0,loop2 #if($4 >= $3) goto loop2
    addiu $9,$3,0 #交换 34
    addiu $3,$4,0
    addiu $4,$9,0

```

```

loop2:
    beq $10,$13,loop #if($10 == $13) goto loop
    slt $7,$5,$4 #if($5 < $4) continue
    beq $7,$0,loop3 #if($5 >= $6) goto loop3
    addiu $9,$4,0 #交换 45
    addiu $4,$5,0
    addiu $5,$9,0

```

```

loop3:
    beq $10,$14,loop #if($10 == $14) goto loop
    slt $7,$6,$5 #if($6 < $5) continue
    beq $7,$0,loop #if($6 >= $5) goto loop
    addiu $9,$5,0 #交换 56
    addiu $5,$6,0
    addiu $6,$9,0
    j loop #回到开头

```

```

end:

```

Mult

```

.text
main:
    addiu $s0,$0,114514 #A
    addu $s5,$0,$s0
    addiu $s1,$0,1919810 #B
    addu $s6,$0,$s1

    j Mult

```

```

Mult:
    addiu $s3,$0,0
    addiu $s4,$0,0

    beq $s1,$0,done
    beq $s0,$0,done

    addiu $s2,$0,0

```

```

loop:
    andi $t0,$s0,1
    beq $t0,$0,next
    addu $s3,$s3,$s1
    sltu $t0,$s3,$s1
    addu $s4,$s4,$t0

```

```
    addu $s4,$s4,$s2

next:
    srl $t0,$s1,31
    sll $s1,$s1,1
    sll $s2,$s2,1
    addu $s2,$s2,$t0

    srl $s0,$s0,1
    bne $s0,$0,loop

done:
    j end

end: # A x B = $s4 $s3
```

四、实验结果

Fibonacci
0-11 依次为： 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

示例一： 11—89

```
#设定第几个数
addiu $4,$0,11 #默认大于1
```

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000059
\$v0	2	0x00000037
\$v1	3	0x00000059
\$a0	4	0x00000000
\$a1	5	0x00000001
\$a2	6	0x00000037
\$a3	7	0x00000059

☰ 程序员

59

HEX 59

DEC 89

示例二： 6—8

```
#设定第几个数
addiu $4,$0,6 #默认大于1
beq $4,$0,end
```

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000008
\$v0	2	0x00000008
\$v1	3	0x0000000d
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000008
\$a3	7	0x0000000d
\$t0	8	0x00000000

≡ 程序员

8

HEX 8

DEC 8

Bubble_Sort

示例一： 13 11 7 5 3 => 3 5 7 11 13

```
#待排序
addiu $2,$0,13
addiu $3,$0,11
addiu $4,$0,7
addiu $5,$0,5
addiu $6,$0,3
```

\$v0	2	0x00000003
\$v1	3	0x00000005
\$a0	4	0x00000007
\$a1	5	0x0000000b
\$a2	6	0x0000000d

示例二：256 512 64 128 32 => 32 64 128 256 512

#待排序

addiu \$2, \$0, 256

addiu \$3, \$0, 512

addiu \$4, \$0, 64

addiu \$5, \$0, 128

addiu \$6, \$0, 32

\$v0	2	0x00000020
\$v1	3	0x00000040
\$a0	4	0x00000080
\$a1	5	0x00000100
\$a2	6	0x00000200

20

HEX 20

DEC 32

40

HEX 40

DEC 64

80

HEX 80

DEC 128

100

HEX 100

DEC 256

200

HEX 200

DEC 512

Mult

示例一： $123 * 246 = (30258)_{10} = (7632)_{16}$

```
addiu $s0, $0, 123 #A
addu $s5, $0, $s0
addiu $s1, $0, 246 #B
addu $s6, $0, $s1
```

```
end: # A x B = $s4 $s3
```

\$s3	19	0x00007632
\$s4	20	0x00000000

程序员

$123 \times 246 =$

30,258

HEX 7632

DEC 30,258

示例一： $114514 * 1919810 = (219845112340)_{10} = (33_2FCA5924)_{16}$

```
addiu $s0, $0, 114514 #A
addu $s5, $0, $s0
addiu $s1, $0, 1919810 #B
addu $s6, $0, $s1
```

```
end: # A x B = $s4 $s3
```

\$s3	19	0x2fca5924
\$s4	20	0x00000033

程序员

$114514 \times 1919810 =$

219,845,122,340

HEX 33 2FCA 5924

DEC 219,845,122,340