微机原理
(计算机原理)

第10讲 MIPS体系结构与编程

第10讲 MIPS体系结构与编程

- MIPS体系结构概述
- MIPS指令集简介
- MIPS汇编语言程序设计

MIPS体系结构概述

- MIPS诞生于1980年代，是最早的RISC处理器之一，也是目前销量最好的RISC处理器之一，从游戏机到路由器，再到SGI的超级计算机，都可以看到MIPS CPU的应用
- MIPS起源于Stanford大学John Hennessy教授的研究成果。Hennessy于1984年在硅谷创立了MIPS公司(www.mips.com)
- John L. Hennessy出版了两本著名的教科书：
  - Computer Organization and Design : The Hardware/Software Interface(计算机组成与设计：硬件/软件接口)
  - Computer Architecture : A Quantitative Approach(计算机体系结构：量化方法)

MIPS体系结构概述

- MIPS=Microprocessor without Interlocked Pipeline Stages，无互锁流水级的微处理器
- MIPS的另一个含义是每秒百万条指令——Millions of instructions per second

MIPS体系结构概述

- MIPS体系结构经历了以下几代
  - MIPS I——该指令集用于最初的32位处理器，至今仍然很流行， R2000、R3000都是MIPS I的实现
  - MIPS II——MIPS I的升级，最初为R6000定义，失败
  - MIPS III——应用于R4000的64位指令集
  - MIPS IV——MIPS III的升级，应用于R5000和R10000

MIPS体系结构概述

MIPS指令集的特点

- 所有指令都是32位的
- 算术/逻辑运算指令为3操作数指令格式，两个源操作数和一个目标操作数只能是寄存器操作数
- 32个通用寄存器
- 寄存器0($0)总是返回0
- 无条件码

MIPS体系结构概述

内存寻址

- 对内存的访问只能通过装入(load)和存储(store)指令进行。算术/逻辑运算指令不能直接访问内存
- 只有一种寻址方式：基址寄存器+16位有符号偏移量
- 数据在内存中存储时必须按边界对齐

MIPS体系结构概述

MIPS不支持的特征

- 不能对字节或半字进行算术/逻辑运算指令
- 没有专门的堆栈支持
- 最小子程序支持。子程序调用通过“跳转与链接指令”进行，将返回地址保存在一个寄存器中($31)。将返回地址保存在寄存器中不如保存在堆栈中灵活
- 最小中断、异常支持：中断发生时，处理器只负责跳到预定义的地址处，其余所有操作都由软件完成

MIPS体系结构概述

寄存器

- 32个通用寄存器可供编程使用：$0～$31，其中
  - $0无论写入什么永远返回0
  - $31被子程序调用指令(“跳转与链接指令”)用来保存返回地址
- 除此以外，所有寄存器都可以在任何指令中以相同方式使用——真正通用

MIPS体系结构概述

- 通用寄存器的习惯用法和命名

寄存器编号	助记符	用法
0	zero	永远为0
1	at	用做汇编器的暂时变量
2-3	v0, v1	子函数调用返回结果
4-7	a0-a3	子函数调用的参数
8-15	t0-t7	临时寄存器。在子函数中使用时不需要保存与恢复
24-25	t8-t9	
16-23	s0-s7	保存寄存器。在子函数中使用这些寄存器时，子函数必须保存和恢复这些寄存器的原值
26,27	k0,k1	通常被中断或异常处理程序使用，用来保存一些系统参数
28	gp	全局指针。一些运行系统维护这个指针来更方便的存取static和extern变量
29	sp	堆栈指针
30	s8/fp	第9个保存寄存器/帧指针
31	ra	子函数的返回地址

MIPS体系结构概述

整数除法相关的寄存器

- 有两个和乘法相关的寄存器**hi**和**lo**，用来保存计算结果
- 它们不是通用寄存器，只能用于乘除操作

MIPS体系结构概述

浮点寄存器

- 浮点加速器(**FPA，floating-point accelerator**)
  如果可用，则增加了**32**个浮点寄存器**$f0～$f31**

MIPS指令集简介

- MIPS指令集包括三类指令：
  - 数据处理指令——实现算术与逻辑运算
  - 数据传送指令——实现寄存器和内存间的数据交换
  - 分支指令——实现程序流程控制

MIPS指令集简介

- 算术运算指令

```
add $t0,$t1,$t2    # $t0 = $t1 + $t2
sub $t2,$t3,$t4    # $t2 = $t3 - $t4

addu $t1,$t6,$t7   # $t1 = $t6 + $t7   无符号整数加法
subu $t1,$t6,$t7   # $t1 = $t6 + $t7   无符号整数减法

addi $t2,$t3, 5    # $t2 = $t3 + 5     加16位立即数
addiu $t2,$t3, 5   # $t2 = $t3 + 5     加16位无符号立即数
```

  - MIPS没有减立即数的指令，如何实现减立即数操作？

MIPS指令集简介

- 算术运算指令

```
mul $t0,$t1,$t2      # $t0 = $t1 * $t2，仅保存乘积的低32位

mult $t3,$t4         # 两个32位的量相乘，结果保存在特殊
                     # 寄存器hi和lo中，(hi,lo) = $t3 * $t4
multu $t3,$t4        # 无符号数乘法

div $t5, $t6         # lo = $t5 / $t6 ， hi = $t5 mod $t6
divu $t5, $t6        #无符号数除法

mfhi $t0             # $t0 = hi
mflo $t1             # $t1 = lo
```

MIPS指令集简介

- 逻辑运算指令

```
and $t0,$t1,$t2        # $t0 = $t1 & $t2
or $t0,$t1,$t2         # $t0 = $t1 | $t2
xor $t0,$t1,$t2        # $t0 = $t1 ⊕ $t2
nor $t0,$t1,$t2        # $t0 = ~($t1 | $t2)
```

  - 如何实现NOT运算？

MIPS指令集简介

- 逻辑运算指令
  - 立即数逻辑运算

andi $t0,$t1,10

ori $t0,$t1,10

xori $t0,$t1,10

MIPS指令集简介

- 逻辑运算指令
  - 移位运算

```
sll $t0, $t1, 10     # $t0 = $t1 << 10，  shift left logical
srl $t0, $t1, 10     # $t0 = $t1 << 10，  shift right logical
sra $t0, $t1, 10     # $t0 = $t1 << 10，  shift right arithm.

sllv $t0, $t1, $t3   # $t0 = $t1 << $t3，  shift left logical
srlv $t0, $t1, $t3   # $t0 = $t1 << $t3，  shift right logical
srav $t0, $t1, $t3   # $t0 = $t1 << $t3，  shift right arithm.
```

MIPS指令集简介

- 比较指令
  - 比较两个寄存器的内容，并根据比较的结果设置第三个寄存器

```
slt $t1,$t2,$t3        # if ($t2 < $t3) $t1=1;
                       # else $t1=0
sltu $t1,$t2,$t3       # 无符号比较
```

  - 寄存器与立即数比较

```
slti $t1,$t2,10        # 与立即数比较
sltu $t1,$t2,$t3       # 与无符号立即数比较
```

MIPS指令集简介

- 数据传送指令——存储(store)指令
  - 实现寄存器到内存的数据传送

```
sw  $t3,  500($t4)          #Store word
sh  $t3,  502($t2)          #Store half
sb  $t2,   41($t3)          #Store byte
```

  - MIPS只支持基址+偏移量的寻址方式

MIPS指令集简介

- 数据传送指令——装入(load)指令
  - 实现内存到寄存器的数据传送

```
lw $t1, 30($t2)        #Load word
lh  $t1, 40($t3)       #Load half word
lhu  $t1, 40($t3)      #Load half word unsigned
lb  $t1, 40($t3)       #Load byte
lbu $t1, 40($t3)       #Load byte unsigned
```

MIPS指令集简介

- 数据传送指令
  - 装入高位立即数(load upper immediate)

lui $t1, 30

  - 例，将32位立即数0x1234abcd装入$t1寄存器

lui $t1, 0x1234

ori $t1, $t1, 0xabcd

MIPS指令集简介

- 分支(branch)指令
  - 比较并分支

```
beq  rs, rt, Target     #如果rs=rt，则分支执行标
                        #号为Target的指令
bne  rs, rt, Target     #如果rs!=rt，则分支执行
                        #标号为Target的指令
```

MIPS指令集简介

- 分支(branch)指令
  - 与0比较并分支

```
blez rs, Target      #如果rs<=0，则分支
bgtz rs, Target      #如果rs>0，则分支
bltz rs, Target      #如果rs<0，则分支
bgez rs, Target      #如果rs>=0，则分支
```

MIPS指令集简介

- 跳转(jump)指令
  - 无条件分支

```
j Exit      #无条件跳转到标号Exit处
```

MIPS指令集简介

堆栈操作

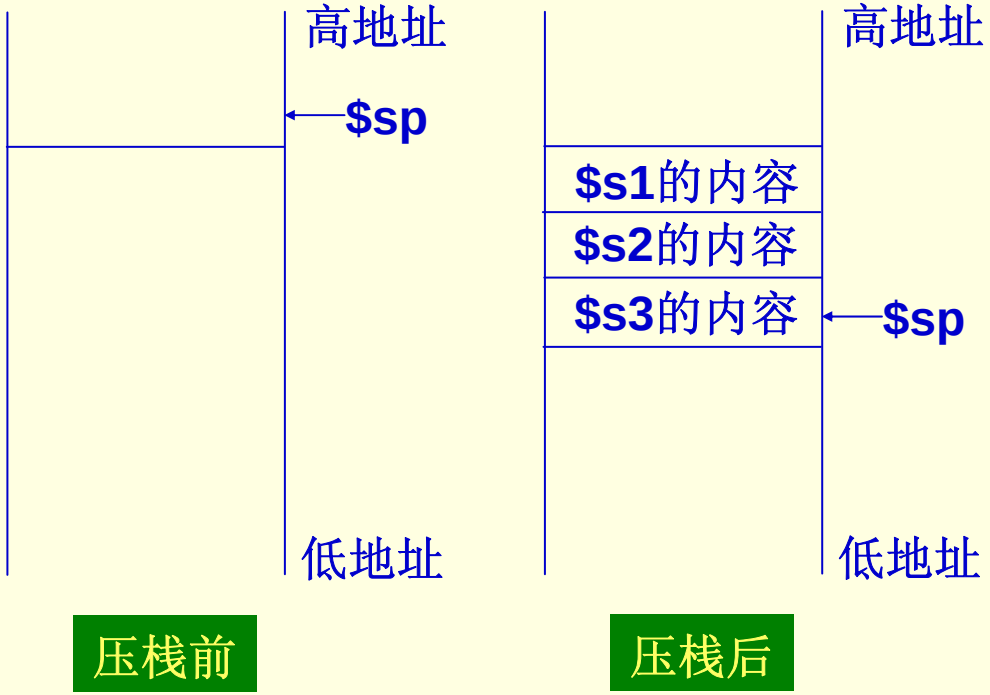
- 虽然MIPS有32个通用寄存器，但是在某些情况下(例如子程序调用)仍然需要将寄存器的内容换出到内存中，在这种情形下，堆栈是保存寄存器内容的理想场所
- MIPS有一个$sp寄存器可以用做堆栈指针，但是MIPS没有PUSH和POP指令，堆栈的操作必须由程序员自己维护

MIPS指令集简介

堆栈操作

- 例，将$s1、 $s2、$s3寄存器的内容压入堆栈

```
addi $sp, $sp, -12
sw $s1, 8($sp)
sw $s2, 4($sp)
sw $s3, 0($sp)
```

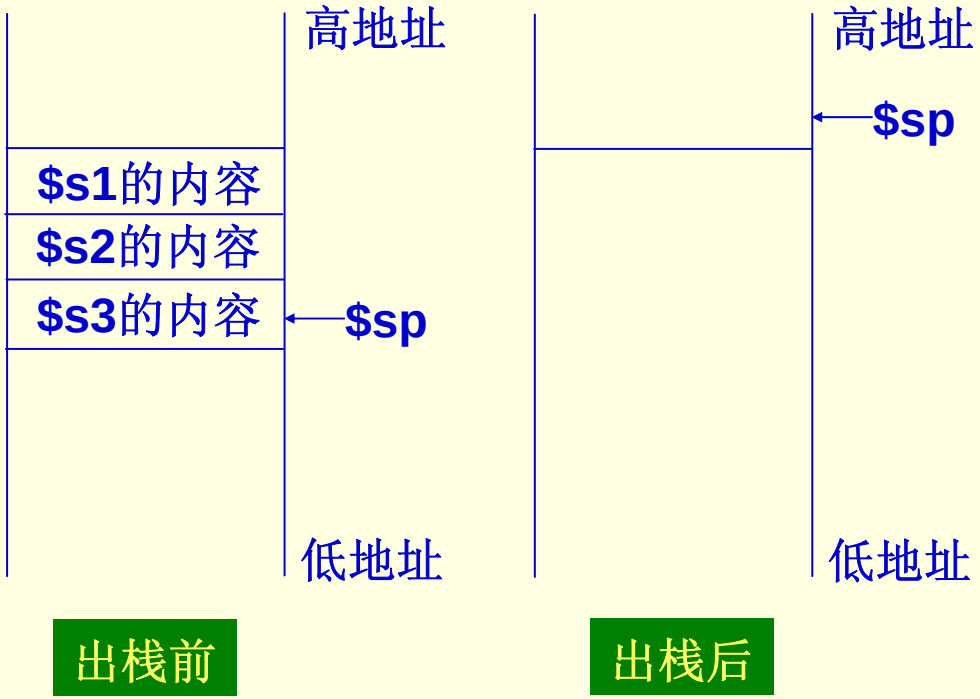


MIPS指令集简介

堆栈操作

- 出栈操作

```
lw $s1, 8($sp)
lw $s2, 4($sp)
lw $s3, 0($sp)
addi $sp, $sp, 12
```



MIPS指令集简介

过程调用

- MIPS的过程调用遵循如下约定：
  - 通过$a0～$a3四个参数寄存器传递参数
  - 通过$v0～$v1两个返回值寄存器传递返回值
  - 通过$ra寄存器保存返回地址

MIPS指令集简介

过程调用

- 子程序调用通过跳转与链接指令**jal**进行

```
jal Procedure   # 将返回地址保存在$ra寄存器
                # 中，程序跳转到过程
                # Procedure处执行
```

- 子程序返回通过寄存器跳转指令**jr**进行

```
jr $ra          # 跳转到寄存器指定的地址
```

MIPS汇编语言程序设计

- SPIM是主要的MIPS模拟器，能够运行和调试MIPS汇编语言程序
- SPIM支持Uinx、Windows等多个操作系统平台

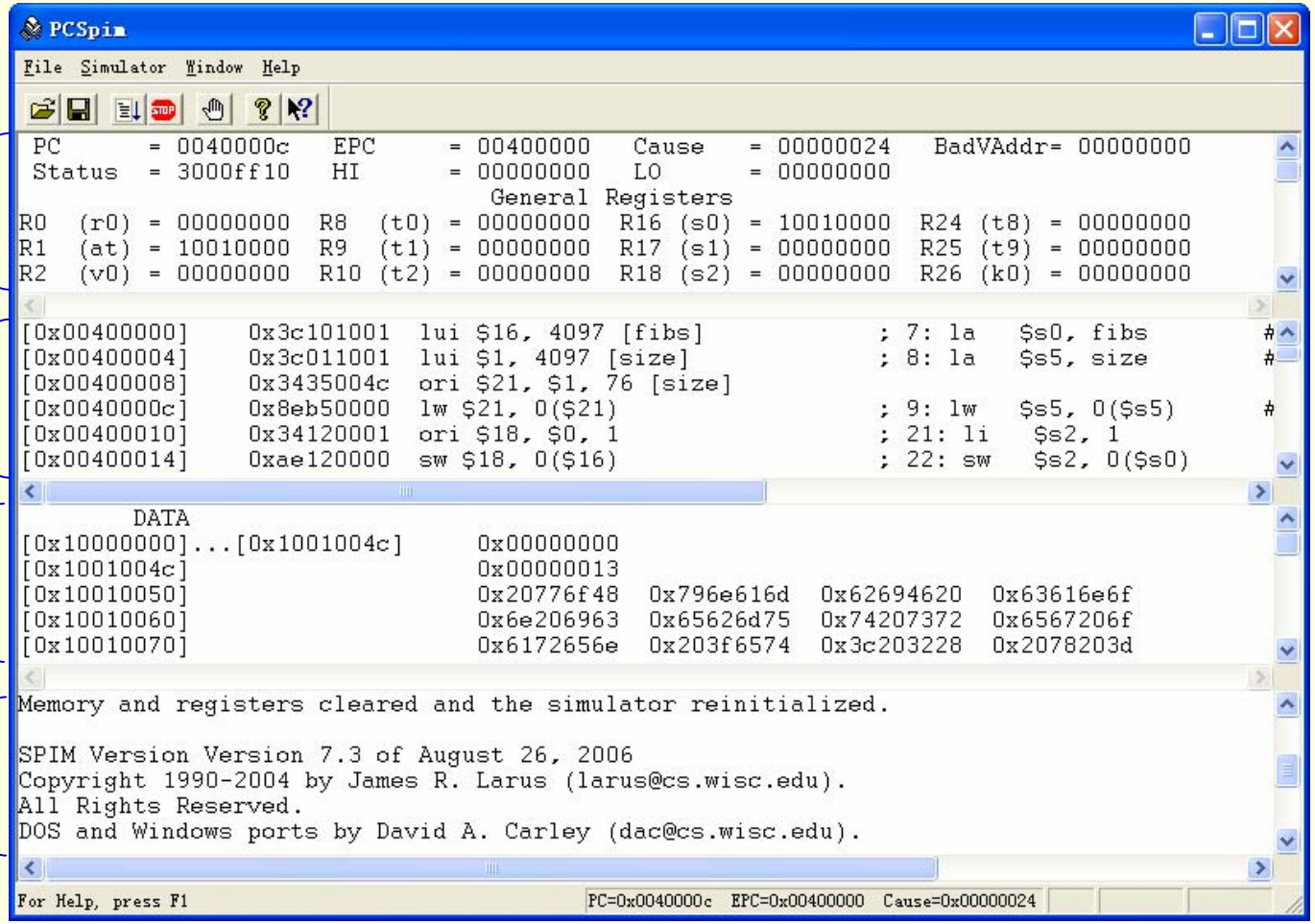
MIPS汇编语言程序设计

寄存器窗口

正文段

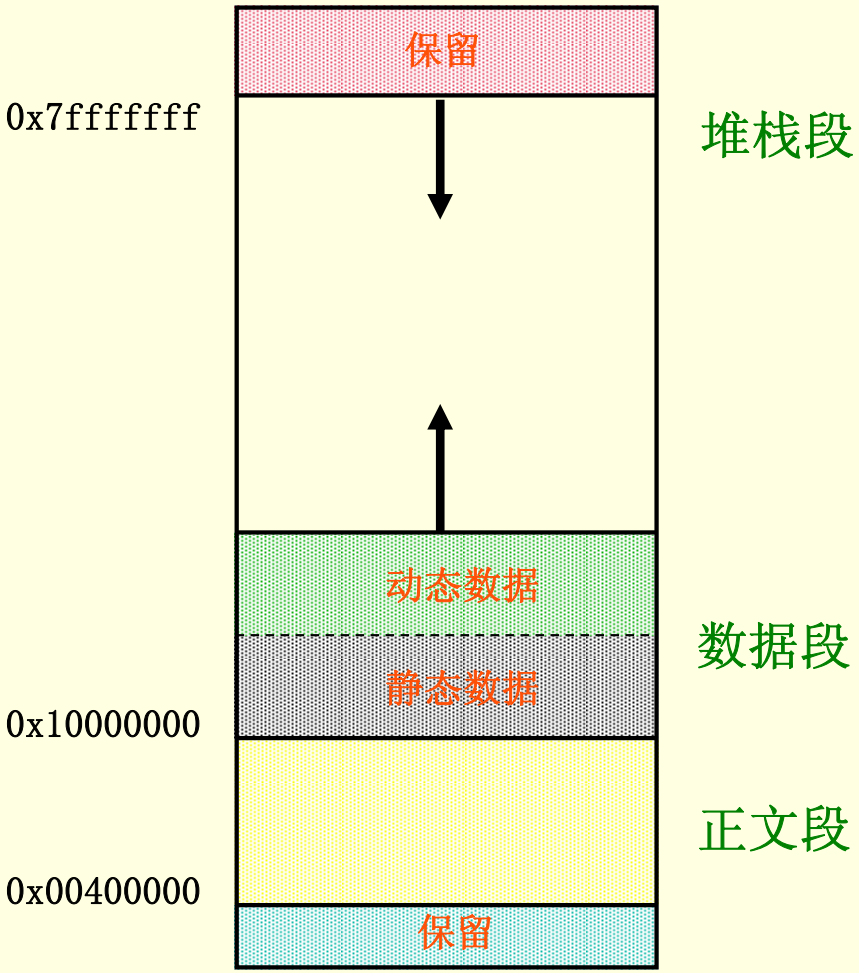
数据与堆栈段

SPIM消息



MIPS汇编语言程序设计

- 内存布局



MIPS汇编语言程序设计

MIPS汇编语言语句格式

- 指令与伪指令语句

  [Label:] <op> Arg1, [Arg2], [Arg3] [#comment]

- 汇编指示语句

  [Label:] .Directive [arg1], [arg2], . . . [#comment]

MIPS汇编语言程序设计

伪指令(Pseudo-instructions)

- 为编程方便而对扩展指令集进行的扩展，并非真正的指令。例如，**ble**、**move**等
- 编程时，伪指令可以和指令一样在程序中使用，在汇编时伪指令将被等效的指令取代

MIPS汇编语言程序设计

伪指令的例子

- move $2, $4
  → add $2, $4, $0 # 寄存器间数据传送，$2 = $4

- li $8, 40
  → addi $8, $0, 40 # 装入立即数，$8 = 40

- sd $4, 0($29) # mem[$29] = $4; # mem[$29+4] = $5
  → sw $4, 0($29)
  sw $5, 4($29)

- la $4, 0x1000056c # 装入地址，$4 = <address>
  → lui $4, 0x1000
  ori $4, $4, 0x056c

MIPS汇编语言程序设计

汇编指示(伪指令)的例子

- .align n　　# 以 2^n字节边界对齐数据.
- .ascii <string>　　# 在内存中存放字符串
- .asciiz <string>　　# 在内存中存放NULL结束的
  # 字符串
- .data　[address]　　# 定义数据段
  # [address]为可选的地址
- .text　[address]　　# 定义代码段
- .word w1, w2, . . . , wn　　# 在内存中存放n个字

MIPS汇编语言程序设计

- Hello world

```
        .text
        .align  2
main:
        la      $4, str
        li      $2, 4
        syscall                 # print string
        li      $2, 10
        syscall                 # exit

        .data
        .align 2
str:
        .asciiz "Hello world."
```

MIPS汇编语言程序设计

系统调用

- **SPIM提供系统调用指令(syscall) 提供了一组类似操作系统的服务**
- 调用方法：
  - 将系统调用代码装入**$v0($2)**寄存器
  - 将参数(如果有)装入**$a0($4)**、**$a1($5)**或**$f12**寄存器
  - **Syscall**
  - 返回值保存在**$v0($2)**或**$f0**寄存器中

MIPS汇编语言程序设计

系统调用

代码	服务	参数	结果
1	print integer	$a0	
2	print float	$f12	
3	print double	$f12	
4	print string	$a0	
5	read integer		integer in $v0
6	read float		float in $f0
7	read double		double in $f0
8	read string	$a0=buffer, $a1=length	
9	sbrk	$a0=amount	address in $v0
10	exit		

MIPS汇编语言程序设计

- 计算$0^2+1^2+2^2+\ldots+100^2$

```
        .text
        .align  2
main:
        la      $10, Temp
loop:
        lw      $14, 4($10)
        mul     $15, $14,$14
        lw      $24, 0($10)
        add     $25, $24,$15
        sw      $25, 0($10)
        addi    $8, $14, 1
        sw      $8, 4($10)
        ble     $8, 100, loop
        la      $4, str
        li      $2, 4
        syscall
        li      $2, 1
        lw      $4, 0($10)
        syscall

        li      $2, 10
        syscall

        .data
        .align 2
Temp:   .word 0, 0
str:    .asciiz "The sum of square
from 0 to 100 is "
```

MIPS汇编语言程序设计

- 计算n!

```
        .data
        .align 2
str:  .asciiz "The factorial is "

        .text
        .align    2
main:
        la        $a0, str
        li        $v0, 4
        syscall
        li        $a0, 6
        jal       factorial
        #nop
        move      $a0, $v0
        li        $v0, 1
        syscall
        li        $v0, 10
        syscall
```

```
factorial:
        addi $sp, $sp, -8
        sw        $ra, 4($sp)
        sw        $a0, 0($sp)
        slti $t0, $a0, 1
        beq       $t0, $zero, L1
        addi $v0, $zero, 1
        addi $sp, $sp, 8
        jr        $ra
L1:
        addi $a0, $a0, -1
        jal       factorial
        #nop
        lw        $a0, 0($sp)
        lw        $ra, 4($sp)
        addi $sp, $sp, 8
        mul $v0, $a0, $v0
        jr $ra
```

小结

- MIPS体系结构概述
- MIPS指令集简介
- MIPS汇编语言程序设计
- 要求理解相关的内容