附录 C　模拟器 MIPSsim 的汇编语言

1. 汇编程序语法结构

可先定义数据段，再定义代码段；或者直接定义代码段（数据段在程序最后定义）

代码段以“.text”开头，含有指令。

数据段以“.data”开头，含 byte、half、word、dword、single、double、space 等数据子节。

除 space 外，各类数据子节可含若干数据列表，其每个数据皆属于该类型。每个数据间可用逗号隔开。

以类似正则表达式的方式表示（其中“*”表示 0 到多个匹配，“[ ]”表示将其中内容作为整体匹配，“|”表示“或”关系，“?”表示 1 到多个匹配，“RT”表示换行符）：

```
sparc_assemblly: [ text_sec | data_sec ] *
text_sec:'.text '[ addr ]? RT [ text_line [ comment ]? RT ] *
text_line: instr | label | align | (blank)
data_sec: [ data_line [ comment ]? RT] *
data_line:'.data '[ addr ]? RT [ sub_sec | align ] *
sub_sec: byte_sec | half_sec | word_sec | dword_sec | single_sec | double_sec | ascii_sec | asciiz_
sec | space_sec
word_sec:'.word 'word_list RT
word_list: word [','word ] *
(byte、half、dword、single、double、ascii、asciiz 子节与 word 子节类似)
space_sec:'.space 'number RT
align:'.align '[ 0 | 1 | 2 | 3 ]
```

2. 详细说明

① 指令段：含有指令，'.text '后若指定了地址，则该地址表示该段起始地址。

② 数据段：含有数据，'.data '后若指定了数字，则该地址表示该段起始地址。

③ 标签：本身代表一个地址，即其所在处。

④ 对齐（align）：指示接下来的单元如何对齐，设所指定的数字为 n，则在 2^n 边界处对齐。

⑤ 数据列表：逗号隔开的每个数据依次写入地址空间，每个数据在写入前都要

对齐。

⑥ 换行(RT):即回车。

⑦ 注释(comment):以“#”开头,以换行(RT)结束。

⑧ 标签(label):一个标识符加一个冒号;标识符以字母或下划线开头,接着若干字母、数字或下划线。

⑨ 指令(instr):即指令列表的汇编语句列中所列的语句。

在汇编语句中,各词法要素实际写法如下:

- 整数寄存器:$r0 ~ $r31。
- 浮点寄存器:$f0 ~ $f31,
- 立即数:其书写规则与普通数字相同,分为十进制和十六进制两种。
- 字符:与 C/C++对于字符的定义一致,以单引号为标记。
- 字符串:与 C/C++对于字符串的定义一致,以双引号为标记,特殊转义字符需要多加一个\。

3. 伪指令

名称及缩写	格式	描述与说明
.align	.align *n*	将下一个数据的起始点对准 2 的 *n* 次方字节地址的边界。例如,.align 2 将下一个数据对准字边界
.ascii	.ascii str	在内存中存储字符串 str, 但不以 null 结尾
.asciiz	.asciiz str	在内存中存储字符串 str, 并以 null 结尾
.byte	.byte b1, b2,…, .b*n*	在内存的连续空间内存储 *n* 个值(字节):b1, b2, …, b*n*
.data	.data <addr>	随后定义的数据被存放到数据段。如果参数 addr 存在,那么这些数据将被存放到以 addr 作为起始地址的一片内存单元中
.double	.double d1, d2, …, dn	在连续的内存区中存储值为 d1, d2, …, d*n* 的 *n* 个双精度浮点数
.extern	.extern sym size	声明存储在 sym 中的数据是 size 个字节大小,并且是一个全局标记,该宏指令使得数据可以存储在凭借 $gp 可以有效访问的数据段中
.float	.float f1, f2, …, f*n*	在连续的内存区中存储值为 f1, f2, …, f*n* 的 *n* 个单精度浮点数

续表

名称及缩写	格式	描述与说明
.globl	.globl sym	将 sym 声明为全局标号，并且可以被其他文件引用
.half	.half h1，…，h*n*	在连续的内存区中存储值为 h1，h2，…，h*n* 的 *n* 个半字(16 位)的数
.kdata	.kdata <addr>	随后的数据被存放到核心数据段，如果参数 addr 存在，那么这些数据将被存放到以 addr 作为起始地址的一片内存单元中
.ktext	.ktext <addr>	随后的数据被存放到核心代码段，如果参数 addr 存在，那么这些代码将被存放到以 addr 作为起始地址的一片内存单元中
.set	.set noat .set at	第一条宏指令拒绝接下来使用 $at 的指令，第二条宏指令恢复该警告
.space	.space *n*	在当前段中分配 *n* 个字节的空间
.text	.text <addr>	随后的项目被存放到代码段。如果参数 addr 存在，那么这些项目将被存放到以 addr 地址开始的内存中
.word	.word w1，…，w*n*	在连续的内存区中存储值为 w1，w2，…，w*n* 的 *n* 个字(32 位)的数