

幻灯片 1

幻灯片 2

**补充：大端小端模式**

- 大端模式将高位存放在低地址，小端模式将高位存放在高地址。
- 将一个 32 位的整数 0x12345678 放到一个整型变量（int）中，（0P0 表示一个 32 位数据的最高字节 MSB，0P3 表示一个 32 位数据最低字节 LSB）。

幻灯片 3

**补充：大端小端模式**

- 将一个 16 位的整数 0x1234 放到一个短整型变量（short）中。这个短整型变量在内存中的存储：

- x86、DEC 是小端模式，很多的 ARM，DSP 都为小端模式。
- PowerPC、IBM、Sun、MIPS 是大端模式。
- 有些 ARM 处理器可以由硬件来选择是大端模式还是小端模式。
- 大多数的操作系统（如 windows, FreeBSD, Linux）是小端方式。
- 少部分，如 MAC OS ,是大端方式。

幻灯片 4

**补充：MIPS 系统的数据存储**

- **MIPS 系统加载/存储必须对齐地址**
  - 字节在任何地址都可以被访问
  - 半字（16 位）必须按偶数字节对齐
  - 字（32 位）必须按 4 字节对齐
  - 双字（64 位）必须按 8 字节对齐

幻灯片 5

**补充：MIPS 系统的数据存储**

- **MIPS 的加载/存储指令**
  - 读取字的指令 lw
  - 读取字节的指令 lb（寄存器高位填入符号位）、lbu（寄存器高位填入 0）
  - 读取半字的指令 lh（寄存器高位填入符号位）、lhu（寄存器高位填入 0）
  - 存储字节/半字/字的指令 sb、sh、sw
  -

幻灯片 6

**补充：MIPS 系统的数据存储**

- 例如：

- lw rd, RAM\_source
  - # 从存储器中读出一个字 word (4 bytes)到目的寄存器中
  - lb rd, RAM\_source
  - # 从存储器中读出一个字节到目的寄存器的低位字节中，目的寄存器的高位部分填入符号位
  - sw rs, RAM\_destination
- # 把源寄存器中的字 word (4 bytes)写入存储器

幻灯片 7

#### 补充：MIPS 系统的数据存储

- sb rs, RAM\_destination
  - # 把源寄存器低位字节 byte 写入存储器
- 
- load / store 指令支持多种寻址方式，例如：
  - sw \$t2, -12(\$t0)
  - # 把\$t2 寄存器中的字写入存储器地址为\$t0-12 的单元
  - 以上指令要求地址对齐，否则将引起地址错误异常。

幻灯片 8

#### 补充：MIPS 系统的数据存储

- 地址非对齐加载/存储
  - 非对齐加载/存储指令：ldr/ldl/lwr/lwl/sdr/sdl/swr/swl
  - ldr/ldl/lwr/lwl/sdr/sdl/swr/swl 指令只能访问内存的始地址到下一个对齐地址处。
  - 例如：
    - lwl \$t0,2(\$0)
    - lwr \$t0,5(\$0)
    - 大端模式下以上 2 条指令顺序执行，完成加载一个非对齐地址字
    -

幻灯片 9

#### 大端模式下，加载未对齐字举例

幻灯片 10

#### 小端模式下，加载未对齐字举例

- 小端机器上，始地址为  $t1 = 0x1022$ ，则：
  - ldr t0, 0(\$t1) 取 0x1022-0x1027 到 t0 的右部
  - ldl t0, 7(\$t1) 取 0x1028-0x1029 到 t0 的左部

0AH
9
8
7

## t0寄存器初始值

0EH	0FH	0CH	3
-----	-----	-----	---

## 执行lw1 \$t0,2(\$0)后的t0寄存器

2	3	0CH	3
---	---	-----	---

大端\_百度百科 - 搜狗高速浏览器

账户(U) 文件(F) 查看(V) 收藏(O) 工具(T) 帮助(H)

输入文字搜索 高速

http://baike.baidu.com/view/911873.htm

工具 迅雷下载支持 消息盒子 听音乐 看视频 玩游戏 new 看新闻 截图

主页 - 网址大全 大端小端 - 搜狗搜索 大端\_百度百科

Bai 百科 大端

编辑 收藏 赞 登录

如果将一个32位的整数0x12345678存放在一个**整型变量** (int) 中, 这个整型变量采用大端或者小端模式在内存中的存储由下表所示。为简单起见, 本书使用OP0表示一个32位数据的最高字节MSB (Most Significant Byte), 使用OP3表示一个32位数据最低字节LSB (Least Significant Byte)。

地址偏移	大端模式	小端模式
0x00	12 (OP0)	78 (OP3)
0x01	34 (OP1)	56 (OP2)
0x02	56 (OP2)	34 (OP1)
0x03	78 (OP3)	12 (OP0)

如果将一个16位的整数0x1234存放在一个**短整型变量** (short) 中。这个短整型变量在内存中的存储在大小端模式由下表所示。

地址偏移	大端模式	小端模式
0x00	12 (OP0)	34 (OP1)
0x01	34 (OP1)	12 (OP0)

由上表所知, 采用大小模式对数据进行存放的主要区别在于在存放的**字节顺序**, 大端方式将高位存放在低地址, 小端方式将高位存放在高地址。采用大端方式进行数据存放符合人类的正常思维, 而采用小端方式进行数据存放利于计算机处理。到目前为止, 采用大端或者小端进行数据存放, 其孰优孰劣也没有定论。

有的处理器系统采用了小端方式进行数据存放, 如Intel的奔腾。有的处理器系统采用了大端方式进行数据存放, 如IBM半导体和Freescle的PowerPC处理器。不仅对于处理器, 一些外设的设计中也存在着使用大端或者小端进行数据存放的选择。

因此在一个处理器系统中, 有可能存在大端和小端模式同时存在的现象。这一现象为系统的软硬件设计带来了不小的麻烦, 这方面系统级设计工程师必须深入了解大端和小端模式的差别。大端与小端模式的差别性现在一个处理器的寄存器, 每个寄存器

1 详细解释

1.1 1.1.1 ; 从软件的角度理解端模式

1.2 1.1.2 ; 从系统的角度理解端模式

城市百科

0.42s 14:12 2014/3/4

大端模式的情况则相反: `ldl t0, 0($t1)`

`ldr t0, 7($t1)`