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 存放到一个整型变量（int）中，（OP0 表示一个 32 位数据的最高字节 MSB，OP3 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 存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

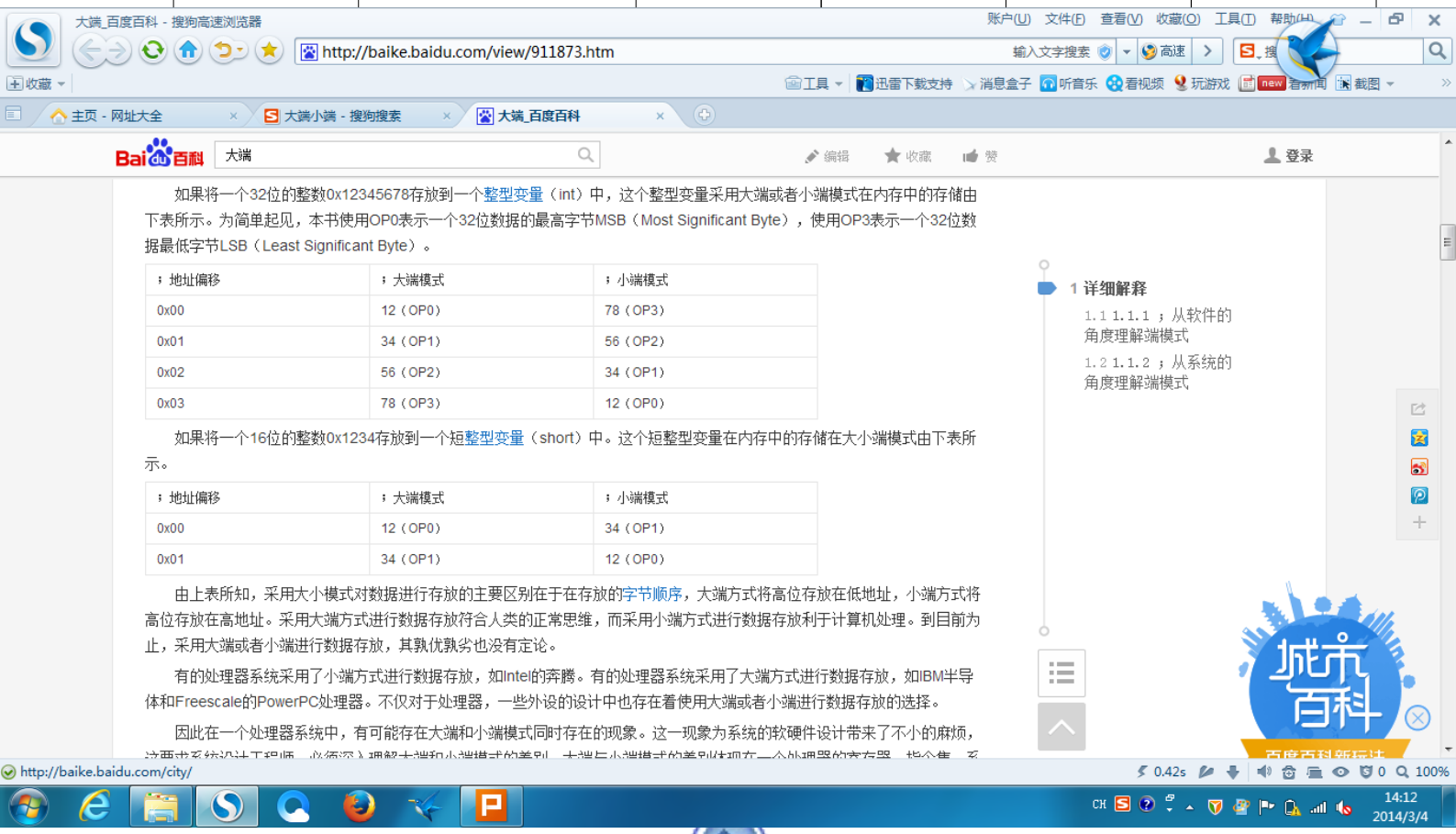
0AH	
9	
8	8
	7

t0寄存器初始值

0EH	0FH	0CH	3

执行lwl $t0,2($0)后的t0寄存器

2	3	0CH	3



大端模式的情况则相反： `ldl t0, 0($t1)`

ldr t0, 7($t1)