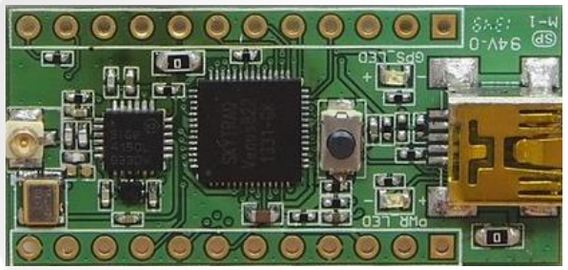


计算机组成原理

第四章 存储系统

4.2 主存中的数据组织

1 存储字长

- 主存的一个存储单元所包含的二进制位数；

- 目前大多数计算机的主存按字节编址，存储字长也不断加大,如16位字长、32位字长和64位字长；

- ISA设计时要考虑的两个问题：

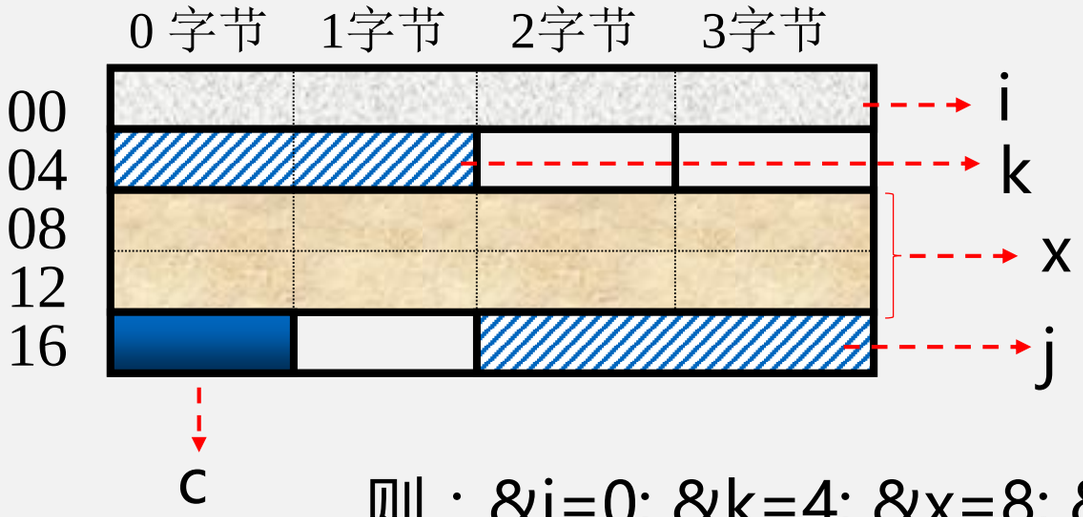
  a)如何根据字节地址读取一个32位的字？ - 字的存放问题

  b)一个字能否存放在主存的任何字节边界？ - 字的边界对齐问题

2 数据存储与边界的关系

1)按边界对齐的数据存储

int i, short k, double x, char c, short j,……（32位系统中）

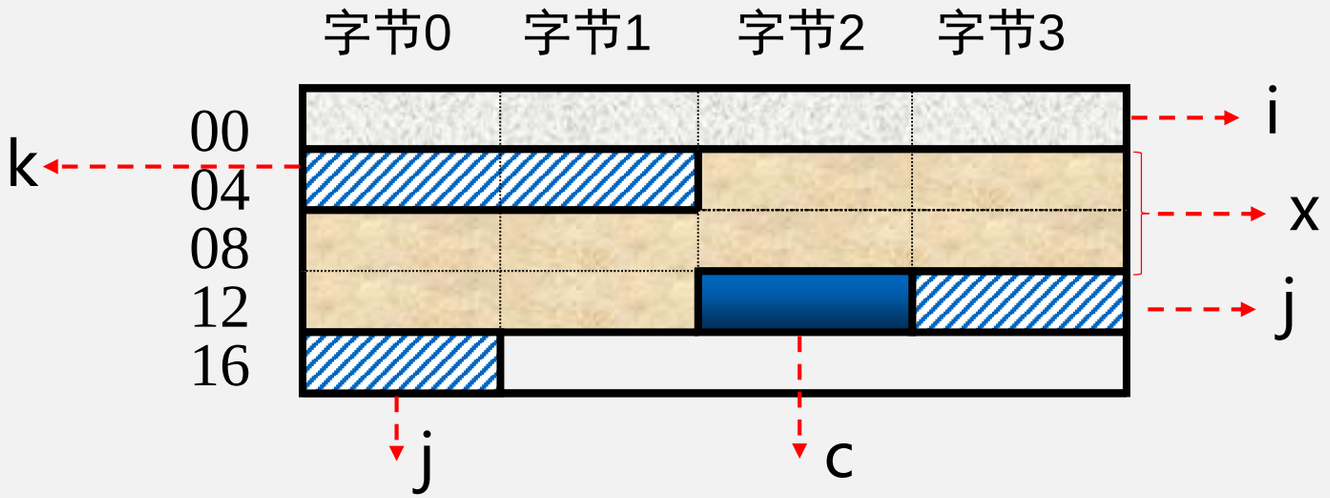


则：&i=0; &k=4; &x=8; &c=16; &j=18;……

2 数据存储与边界的关系

2)未按边界对齐的数据存储

int i, short k, double x, char c, short j,……（32位系统中）



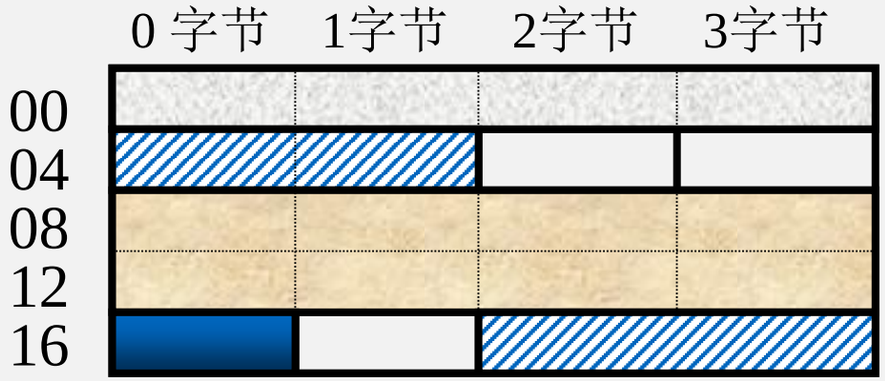
则： &i=0; &k=4; &x=6; &c=14; &j=15;……

虽节省了空间，但增加了访存次数！

需要在性能与容量间权衡！

2 数据存储与边界的关系

3)边界对齐与存储地址的关系（以32位为例）



- 双字长数据边界对齐的起始地址的最末三位为000(8字节整数倍；
- 单字长边界对齐的起始地址的末二位为00(4字节整数倍)；
- 半字长边界对齐的起始地址的最末一位为0(2字节整数倍)。

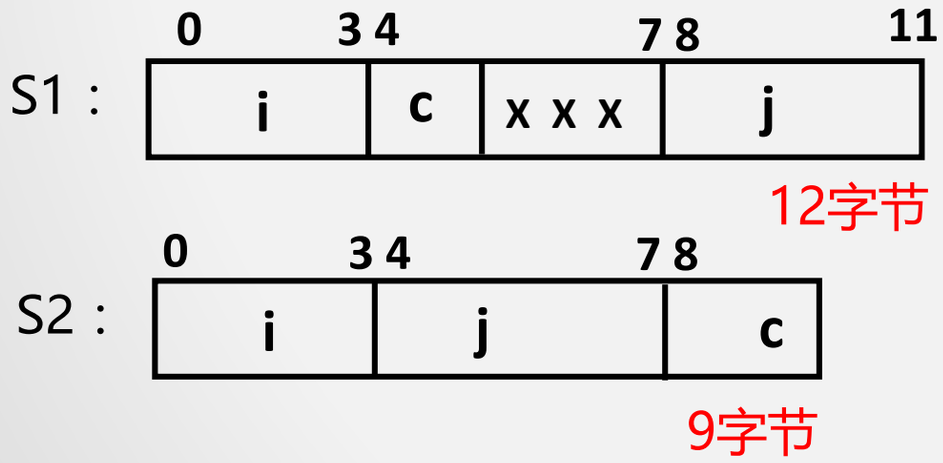
2 数据存储与边界的关系

考虑下列两个结构声明（以32位为例）：

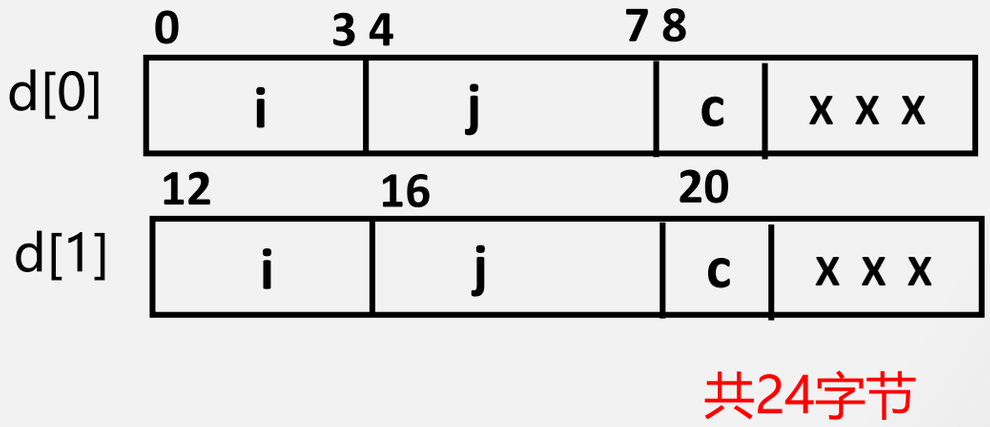
```
struct  S1 {
        int     i ;
        char    c ;
        int     j ; } ;
```

```
struct  S2 {
        int     i ;
        int     j ;
        char    c ; } ;
```

在对齐情况下，哪种结构声明更好？



struct S2 d[2]满足对齐要求的字节数？

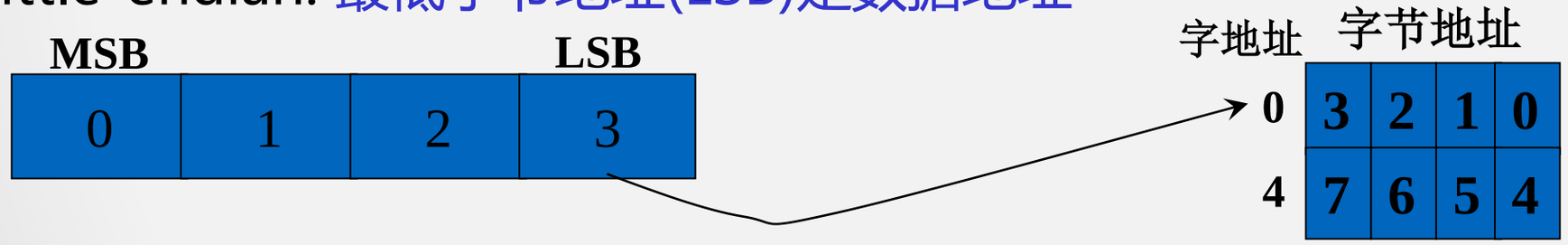


3 大端与小端存储方式

- Big-endian: 最高字节地址(MSB)是数据地址

IBM 360/370, Motorola 68k, MIPS, Sparc

- Little-endian: 最低字节地址(LSB)是数据地址



Intel 80x86, DEC， VAX

有的机器两者都支持，但需要设定.

3 大端与小端存储方式

设某程序执行前　r0 =0x 11223344

执行下列指令：

```
r1=0x100
STR r0, [r1]
LDRB r2 ,[r1]
```

小端模式下：　r2=0x44

大端模式下：　r2=0x11

无论是大端还是小端，每个系统内部是一致的，但在系统间通信时可能会发生问题！因为顺序不同，需要进行顺序转换；