

第一章 计算机系统概述

计算机发展历程

1、硬件

第一代——电子管

第二代——晶体管

第三代——中小规模集成电路

第四代——超大规模集成电路

计算机系统层次结构

1、组成：**硬件系统**与**软件系统**共同组成一个完整的计算机系统

2、计算机硬件

冯诺依曼机基本思想

工作方式：**存储程序**

硬件系统组成：运算器、存储器、控制器、输入设备、输出设备

指令和数据存储在存储器中，存储形式没有区别

指令和数据均由二进制代码表示，指令由操作码和地址码组成

计算机功能部件：

(1)输入设备

(2)输出设备

(3)存储器

分为主存储器（内存）和辅助存储器（外存）

主存使用按地址的存取方式

地址寄存器MAR存放访问地址

数据寄存器MDR存放读取的数据

MAR、MDR、Cache都在CPU中

(4)运算器

核心是算术逻辑单元ALU

还含有累加器ACC、乘商寄存器MQ、操作寄存器X、变址寄存器IX、基址寄存器BR，前三个为必须

PSW标志寄存器，存放处理机状态信息

(5)控制器

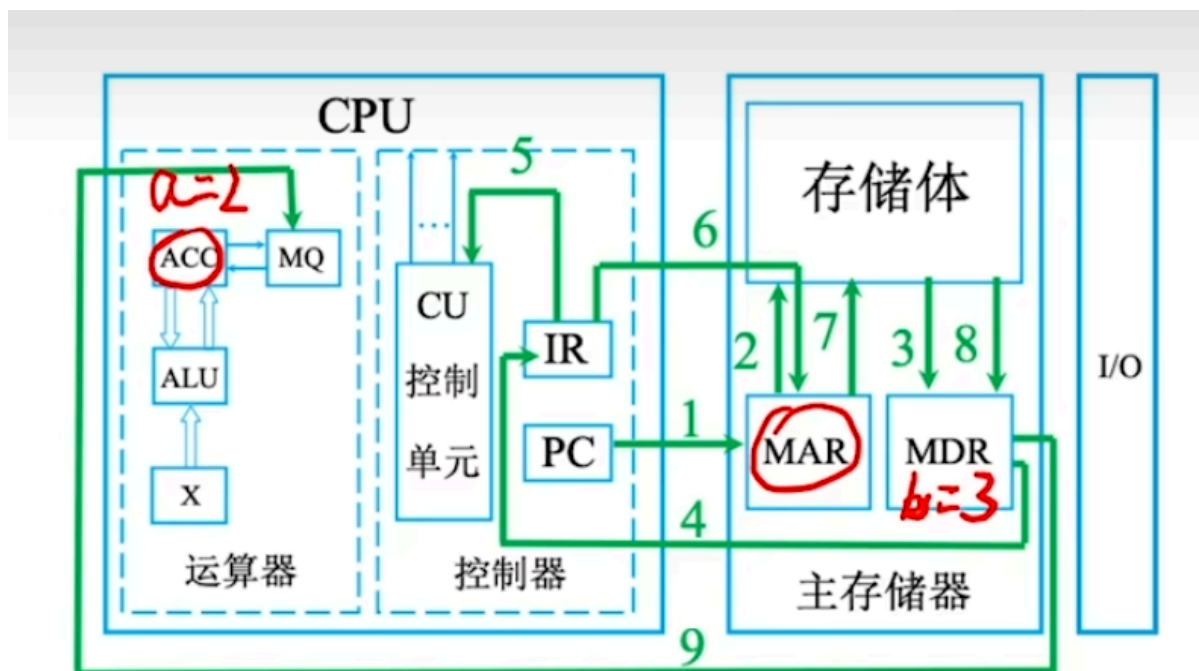
由程序计数器PC、指令寄存器IR、控制单元CU组成

PC来自MAR、IR来自MDR

指令中操作码OP送至CU、地址码Ad(IR)送往MAR取操作数

CPU=运算器+控制器

包含ALU、GPRs（通用寄存器组）、标志寄存器PSW、控制器、指令寄存器IR、程序计数器PC、MAR、MDR



CPU与主存之间通过一条总线相连

总线中有：地址、控制、数据3条

3、计算机软件

系统软件：操作系统OS，数据库管理系统DBMS、语言处理程序、分布式软件、网络软件、标准库等

应用软件：...

3个级别语言：机器语言、汇编语言、高级语言

3种翻译程序：汇编程序（汇编->机器）、解释程序（高级->机器，翻译一句执行一句）、编译程序（高级->机器，一次翻译成目标文件）

逻辑功能等价：某功能既可以由硬件实现也可以由软件实现

4、计算机系统的层次结构

微程序机器层->传统机器语言层->操作系统层->汇编语言层->高级语言层->应用程序层

前3层硬件，后3层软件

软硬件界面：指令集体系结构ISA

5、工作原理

根据PC取指令->指令译码PC+1->取操作数并执行->送结果

hello.c->预处理器cpp->hello.i->编译器ccl->hello.s->汇编器as->hello.o+printf.o可重定位目标程序（二进制）->链接器ld->hello可执行目标程序（二进制）

程序执行过程：

(1)取指令：PC->MAR->M->MDR->IR

(2)分析指令：OP(IR)->CU

(3)执行指令：Ad(IR)->MAR->M->MDR->ACC

计算机性能指标

1、主要性能指标

(1)字长：一次整数运算能处理的二进制数据位数，为字节(B)的倍数

(2)数据通路带宽：数据总线一次能传送信息的位数

(3)主存容量：主存储器的最大容量，以字节来衡量，或者字数*字长

(4)运算速度：

吞吐量：单位时间内处理请求的数量，主要取决于主存的存储周期

响应时间：用户向计算机发出一个请求到获得结果的时间，包括CPU运行的时间和等待时间

CPU时间周期（节拍脉冲/T周期）：CPU最小时间单位，执行指令的每个动作需要一个CPU时间周期，主频的倒数

主频（CPU时间频率）：主时钟的频率

CPI（Cycle Per Instruction）：执行一条指令需要的时钟周期数

CPU执行时间=指令条数*CPI/主频

MIPS（Million Instruction Per Second）：每秒百万条指令， $MIPS = \text{指令条数} / (\text{执行时间} * 10^6) = \text{主频} / (CPI * 10^6)$

MFLOPS：每秒百万次浮点数运算

第二章 数据表示和计算

数制与编码

1、真值和机器数

真值：“+”、“-”表示正负

机器数：0表示正，1表示负

2、定点数的编码表示

通常用**定点补码整数**表示整数

(1)机器数的定点表示：

定点小数：约定小数点在符号位之后有效数值之前， $X = x_0x_1x_2...x_n$ ， x_0 为符号位，后面是有效数值

定点整数：约定小数点在有效数值之后符号位之前， $X = x_0x_1x_2...x_n$ ， x_0 为符号位，后面是有效数值

(2)原码、补码、反码、移码

原码：符号由01表示，其他不变，表示范围： $-(2^n-1) \leq x \leq 2^n-1$ ，零有0,0000和1,0000两种表示

补码：

原码->补码：正数不变，负数符号不变，其余位取反+1

补码->原码：符号为0不变，符号为1，其余位取反+1

[x]补(小数)=

$x, 0 \leq x < 1$

$2+x=2-|x| \quad -1 \leq x < 0$

[x]补(整数)=

$0, x, 0 \leq x < 2^n$

$2^{n+1}+x=2^{n+1}-|x| \quad -2^n \leq x < 0$

表示范围: $-2^n \leq x \leq 2^n - 1$

零的表示唯一: 0,0000

反码: 正数不变, 负数取反

0的表示不唯一

移码: 在真值x上加上一个常数 2^n , n为字长减1

为补码的符号位取反

零的表示唯一: 1,0000

表示范围: $-2^n \leq x \leq 2^n - 1$

3、整数的表示

(1)无符号整数: 全部二进制数都是数值位

(2)带符号整数: 符号位为第一位, 后面是数值位, 用补码表示

运算方法和运算电路

1、基本运算部件

加法器是ALU的核心部件

(1)一位全加器FA

和表达式: $S_i = A_i \oplus B_i \oplus C_{i-1}$

进位表达式: $C_i = A_i B_i \oplus (A_i \oplus B_i) C_{i-1}$

(2)串行进位加法器

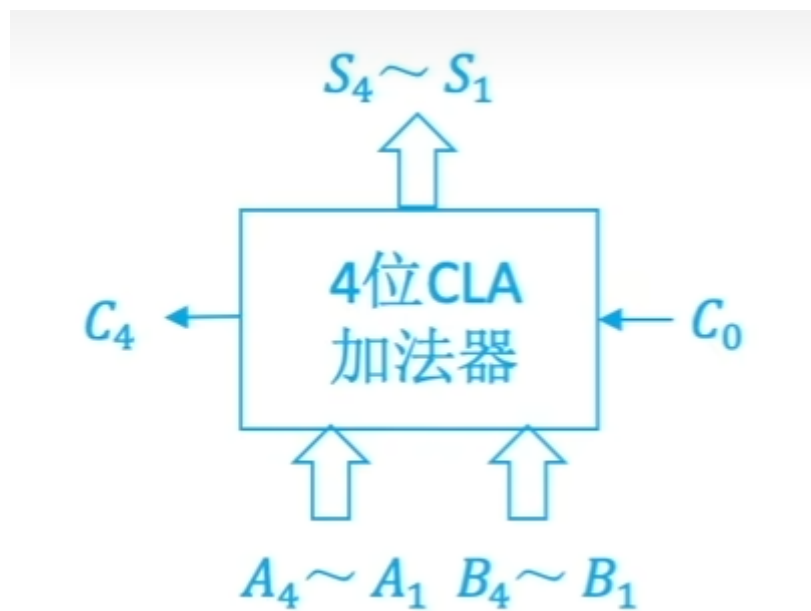
n个全加器相连得到n位加法器

(3)并行进位加法器

进位产生函数 $G_i = A_i B_i$

进位传递函数 $P_i = A_i \oplus B_i$

先行进位部件CLA



(4)带标志加法器

溢出标志 $OF=C_n \oplus C_{n-1}$ ，表示带符号整数运算时发生溢出，对于无符号数无意义

符号标志 $SF=F_{n-1}$ （和的最高位），对于无符号数无意义

零标志 $ZF=1$ 仅当 $F=0$ （和为0），对于两者都有意义

进位/借位标志 $CF=C_{out} \oplus C_{in}$ ，表示无符号数运算的进位/借位，对于带符号数没有意义

(5)算术逻辑单元ALU

输入：n位操作数A、B，进位输入端 C_{in} ，操作控制端 ALU_{op}

输出：ZF、OF、SF、CF、n位结果F

2、定点数的移位运算

(1)算术移位

移位对象是有符号数，符号位不变

	码制	添补代码
正数	原码、反码、补码	0
	原码	0
负数	补码	左移添0，右移添1
	反码	1

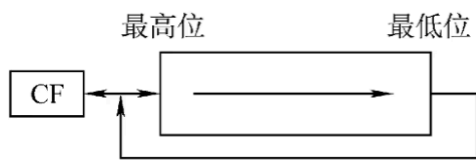
(2)逻辑移位

操作数为无符号数

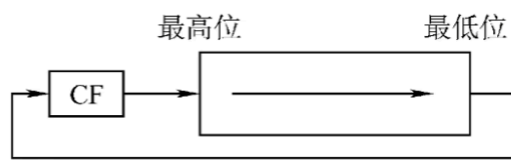
左移右移都添0

(3)循环移位

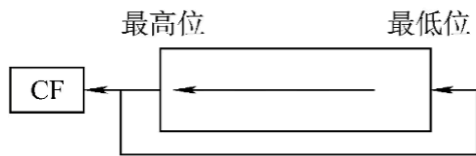
分为带进位标志位CF的循环移位（大循环）和不带进位标志位的循环移位（小循环）



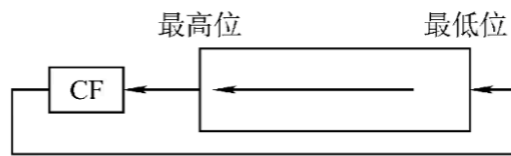
(a) 不带进位位的循环右移



(b) 带进位位的循环右移



(c) 不带进位位的循环左移



(d) 带进位位的循环左移

3、定点数的加减运算

(1) 补码的加减法运算

$$[A+B]_{\text{补}} = [A]_{\text{补}} + [B]_{\text{补}} \pmod{2^{n+1}}$$

$$[A-B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2^{n+1}}$$

(2) 补码的加减运算电路

设置控制端Sub，Sub为0做加法，Sub为1做减法

Sub为1时Y先取反变成-Y再参与运算

$$CF = \text{Sub} \oplus C_{\text{out}}$$

(3) 溢出判别方法

方法一：采用一位符号位：

设A的符号位为 S_A ，B的符号位为 S_B ，运算结果的符号位为 S_S

$$V = A_S B_S \overline{S_S} + \overline{A_S} \overline{B_S} S_S$$

$V=0$ 无溢出， $V=1$ 有溢出

方法二：采用双符号位（模4补码）：

设结果的两个符号位为 S_{S1} 和 S_{S2}

$$V = S_{S1} \oplus S_{S2}$$

$V=0$ 无溢出， $V=1$ 有溢出

方法三：采用一位符号位根据数据位进位情况判断溢出

符号位进位 C_S 与最高数位进位 C_1 相同则无溢出，否则有溢出

$$V = C_S \oplus C_1$$

$V=0$ 无溢出， $V=1$ 有溢出

4、定点数的乘除运算

(1) 定点数的乘法运算

原码一位乘法：

设 $[x]_{\text{原}} = x_s x_1 x_2 \dots x_n$, $[y]_{\text{原}} = y_s y_1 y_2 \dots y_n$

被乘数与乘数都取绝对值参加运算，视作无符号数，符号位为 $x_s \oplus y_s$

部分积初值为0，从乘数最低位 y_n 开始判断：若 $y_n=1$ ，则部分积加上被乘数 $|x|$ ，右移一位；若 $y_n=0$ ，则部分积加上0，右移一位。重复 n 次。

(2)无符号数乘法运算电路

更新中

第三章 存储系统

3.1 存储器概述

1、分类

(1) 按层次

主存储器：主存CPU，随机存取

高速缓冲存储器：Cache

辅助存储器：外存

(2) 存储介质

半导体存储器（主存、cache）、磁表面存储器（磁盘、磁带）、光存储器（光盘）

(3) 存取方式

随机存取存储器RAM：存取时间与存储单元**物理位置无关**（主存、cache）

只读存取存储器ROM：**只能读不能写**，断电不丢失

串行存取存储器：按**物理位置先后**存取，分为顺序存取存储器SAM（磁带）、直接存取存储器DAM（速度大于SAM小于RAM）（磁盘、光盘）

相联存储器CAM：**按内容访问**（快表）

(4) 信息的可更改性

读写存储器RWM（磁盘、内存、Cache）

只读存储器ROM（光盘CD-ROM、BIOS）

(5) 信息的可保存性

易失性存储器：断点**信息消失**（主存，cache）

非易失性存储器：断电信息还在（磁盘、光盘）

破坏性读出：信息读出后被**破坏**（DRAM）

非破坏性读出：信息读出后不被破坏（SRAM，磁盘，光盘）

2、存储器性能指标

(1) 存储容量：存储字数*字长

MAR反映存储字数

MDR反映存储字长

(2) 单位成本：总成本/总容量

(3) 存储速度：数据宽度/存储周期

存取时间Ta：一次存取操作的时间

存取周期Tm：两次存取的间隔，一般>Ta

主存带宽Bm=存储速度=数据宽度/存储周期，单位：B/s、b/s

3.2主存储器

1、SRAM芯片和DRAM芯片

SRAM静态RAM

DRAM动态RAM

地址复用->只需要一半地址线

	SRAM	DRAM
主要用途	cache	主存
存储信息	触发器	电容
破坏性读出	否	是
需要刷新	否	是
地址复用	否	是
运行速度	快	满
集成度	低	高
存储成本	高	低
易失性	是	是

DRAM刷新操作：

刷新周期：1-2ms

刷新单位：**行**

(1) 集中刷新

刷新期间**所有区域**停止读写操作，成为“死时间”/“死区”

(2) 分散刷新

每个存取周期后再用一个存取周期刷新，**没有死区**

(3) 异步刷新

刷新间隔 $t = \text{刷新周期} / \text{行数}$

2、只读存储器

非易失：数据**不会丢失**

(1) **MROM**，掩模式只读存储器

生产过程中写入，**任何人无法改变内容**

(2) **PROM**，可编程只读存储器

写一次后不可更改

(3) **EPROM**，可擦除可编程只读存储器

可多次重写，一次全部擦除

EEPROM，电擦除，可以擦除特定的字

(4) **Flash**，闪存

可多次快速擦除重写，如：U盘，SD卡

写比读更慢

(5) SSD，固态硬盘

可多次擦除重写

控制单元+闪存芯片

3、多模块存储器

(1) 单体多字处理器

每个存储体存 m 个字，一次**并行读出** m 个字

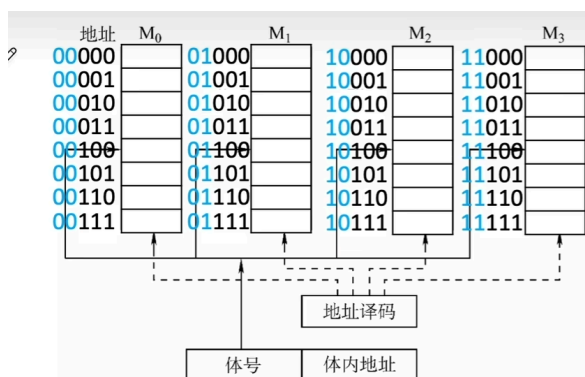
(2) 多体并行存储器

高位交叉编址：高位地址为体号，低位为体内地址

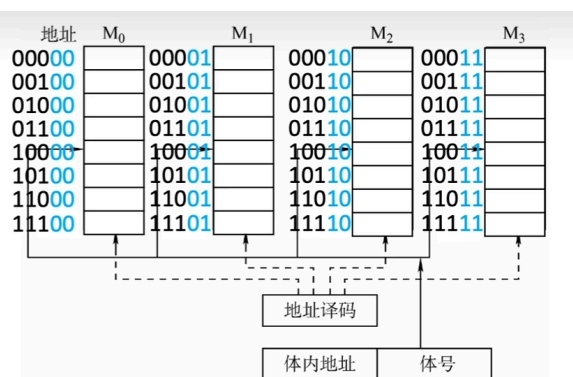
一次读出模 m 相同的全部地址

低位交叉编址（默认）：低位地址为体号，高位为体内地址

一次读出 m 个相邻地址



高位交叉编址的多体存储器



低位交叉编址的多体存储器

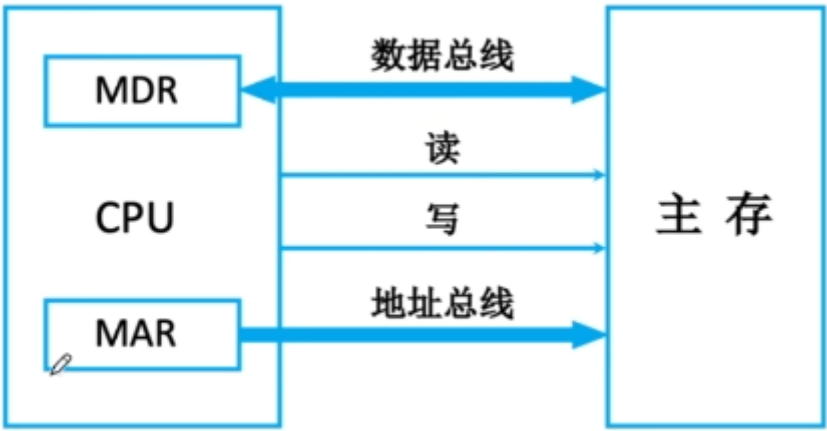
存取周期T，存取时间t，要求 $T \leq mt$ 或者 $m \geq T/r$ ，一般取=

读n个连续地址，低位交叉需要的时间： $T+(n-1)t=T+(n-1)T/m$

3.3主存储器与CPU的连接

1、连接原理

片选线CS，读写线WE（读为RD，写为WR）



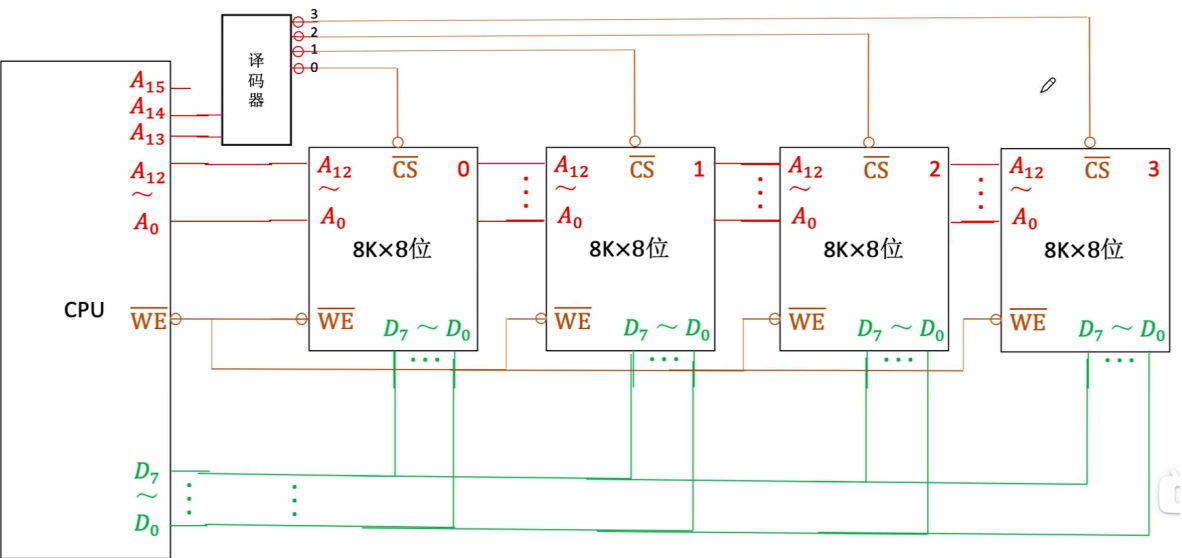
2、主存储容量的拓展

(1) 位拓展

地址线连到所有芯片，需要增加新的数据线连到新芯片

(2) 字拓展

增加新地址线，通过译码器后连到所有芯片，数据线也要连到所有芯片



(3) 字位同时拓展

3、存储芯片的地址分配和片选

(1) 线选法

高位地址线直接连到**片选端**

地址线信息为0时选中对应芯片

增加n个芯片需要n条地址线

(2) 译码片选法

高位地址通过**译码器**连接到芯片地址端

增加n个芯片需要 $\log_2 n$ 条地址线

译码器可能有**使能端G**，信号为1时才工作，连接到CPU的**MREQ**

3.4外部存储器

1、磁盘存储器

磁头数->柱面数->扇区

最小读写单位：**扇区**

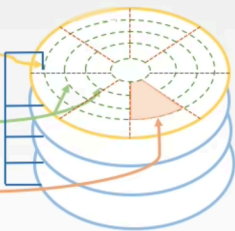
① 存储区域

一块硬盘含有若干个记录面，每个记录面划分为若干条磁道，而每条磁道又划分为若干个扇区，扇区（也称块）是磁盘读写的最小单位，也就是说磁盘按块存取。

磁头数（Heads）	即记录面数，表示硬盘总共有多少个磁头，磁头用于读取/写入盘片上记录面的信息，一个记录面对应一个磁头。
柱面数（Cylinders）	表示硬盘每一面盘片上有多少条磁道。在一个盘组中，不同记录面的相同编号（位置）的诸磁道构成一个圆柱面。
扇区数（Sectors）	表示每一条磁道上有多少个扇区。

② 硬盘存储器

硬盘存储器由磁盘驱动器、磁盘控制器和盘片组成。



2、性能指标

寻道时间：磁头移动到目的磁道的时间，经常取平均值

旋转时间： $1/(2 \times \text{转速})$ ，经常取平均值

存取时间：寻道时间(题目给出)+旋转延迟时间($1/2 \times \text{转速}$)+传输时间(扇区大小 \div 传输速率)

3、磁盘阵列

RAID独立冗余磁盘阵列

措施：奇偶校验、海明纠错、磁盘镜像

效果：提高传输率、数据吞吐量、安全性、容错能力

4、固态硬盘

易磨损

3.5高速缓冲存储器

1、局部性原理

空间局部性：最近的未来要用到的信息，常常是存储空间临近的**数组**。

时间局部性：最近的未来要用到的信息，常常是**重复访问的程序段**（循环）。

2、cache基本原理

cache总长=有效位1位(1有效0无效)+标记位+行长

主存与cache之间以**块**为单位进行数据交换

主存中的块也称为**页/页面/页框**

cache中的块也称为**行**，行长是**数据的长度**不包括前面的标志那些

命中次数 N_c

访问主存次数 N_m

命中率 $H = N_c / (N_c + N_m)$

缺失率 $M = 1 - H$

访问一次cache需要的时间 t_c

访问一次主存需要的时间 t_m

平均访问时间 $t = H t_c + (1 - H)(t_c + t_m)$

3、cache和主存的映射方式

(1) 全相联映射

主存的每一块可装入cache的**任何位置**

标记位=主存块地址

需要依次查找对比标记，访问**最慢**

(2) 直接相联映射

cache共 m 行

主存的第 n 行映射到cache的 **$n \% m$** 行

标记位=主存块地址/ m ，即**少了 $\log_2 m$ 位**

替换时**直接换掉**对应地址的块

256M=2²⁸ 主存的地址共28位：

主存块号		块内地址
22位		6位
19位 标记	3位 行号	6位块内 地址

Cache 共2³ 行

(3) 组相联映射

给cache分组

组内采用全相联映射，组间采用直接相联映射

主存的第n行映射到cache的n%m组

x路组相联映射=x个cache行为一组

4、cache中主存块的替换算法

(1) 随机算法RAND

随机选择一块替换

不稳定

(2) 先进先出算法FIFO

替换**最先调入**的块

不满足局部性

抖动现象：频繁换入换出

(2) 近期最少使用算法LRU

维护一个**计数器**，记录多久没有被访问过

步骤：

命中行计数器清0，比其低的计数器+1

新装入自身清0，全部+1

换出最大的

5、cache的写策略

(1) 写操作命中

全写法：同时写入cache和主存，写回更慢，经常增加一个**写缓存**

回写法：写入cache，当块**被换出**时写入主存，需要增加一个**脏位**

(2) 写操作不命中

写分配法：把主存块调入cache，在cache中修改，与**回写法**一起使用

非写分配法：只更新主存，**不调入**cache，与**全写法**一起使用

多级cache间采用**全写+非写分配**

cache和主存间采用**写回+写分配**

3.6虚拟存储器

1、基本概念

虚拟地址/逻辑地址：用户编程允许使用的地址

实际地址/物理地址：实际的主存单元

页表使用**写回法**

2、页式虚拟存储器

页表内容包括：

虚页号：虚拟地址

有效位：1有效0无效

脏位：1修改过0没改过

引用位：替换算法使用

实页号：物理地址

3、快表（TLB）

快表TLB由**高速缓冲存储器**组成，采用**相联存储器CAM**

满表Page在主存中

快表查不到再查满表

序号	TLB	Page	Cache	说 明
1	命中	命中	命中	TLB 命中则 Page 一定命中，信息在主存，就可能在 Cache 中
2	命中	命中	缺失	TLB 命中则 Page 一定命中，信息在主存，也可能不在 Cache 中
3	缺失	命中	命中	TLB 缺失但 Page 可能命中，信息在主存，就可能在 Cache 中
4	缺失	命中	缺失	TLB 缺失但 Page 可能命中，信息在主存，也可能不在 Cache 中
5	缺失	缺失	缺失	TLB 缺失则 Page 也可能缺失，信息不在主存，也一定不在 Cache

4、段页式虚拟存储器

段是按照**逻辑结构**划分的，便于编程（程序段，数据段）

以**页**为基本单位

一个段包含多个页

虚地址分为：段号，段内页号，页内地址

第四章 指令系统

4.1指令系统

1、概念

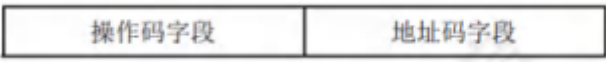
机器指令：二进制指令

指令集/指令系统：一台计算机的所有指令的**集合**

指令字长：一条指令的**长度**

2、指令的基本格式

指令=**操作码字段**+**地址码字段**



指令字长通常为**字节整数倍**

定长指令字结构：所有指令**长度相等**

变长指令字结构：指令长度随功能而异

(1) 零地址指令

如：空操作，停机，关中断等等

对于**堆栈计算机**，两个操作数隐含存放在**栈顶**和**次栈顶**

(2) 一地址指令

OP(A1)->A1 (3次访存：取指令，读A1，写A1)

ACC累加寄存器作为隐含目的地址：(ACC)OP(A1)->ACC

(3) 二地址指令

A1为目的操作数

A2为源操作数

(A1)OP(A2)->A1 (4次访存)

(4) 三地址指令

(A1)OP(A2)->A3 (4次访存)

(5) 四地址指令

(A1)OP(A2)->(A3)

PC=A4 (4次访存)

3、定长操作码指令格式

指令字最高位分配**固定**若干位操作码

4、拓展操作码指令格式

操作码长度**不定**，指令字长**固定**

操作码的位数随地址数的减少而增加

	OP	A ₁	A ₂	A ₃	
4位操作码	0000	A ₁	A ₂	A ₃	15条三地址指令
	0001	A ₁	A ₂	A ₃	
	⋮	⋮	⋮	⋮	
	1110	A ₁	A ₂	A ₃	
8位操作码	1111	0000	A ₂	A ₃	15条二地址指令
	1111	0001	A ₂	A ₃	
	⋮	⋮	⋮	⋮	
	1111	1110	A ₂	A ₃	
12位操作码	1111	1111	0000	A ₃	15条一地址指令
	1111	1111	0001	A ₃	
	⋮	⋮	⋮	⋮	
	1111	1111	1110	A ₃	
16位操作码	1111	1111	1111	0000	16条零地址指令
	1111	1111	1111	0001	
	⋮	⋮	⋮	⋮	
	1111	1111	1111	1111	

图 4.1 扩展操作码技术

地址长度为n，上一层留出m种状态，下一层可拓展 $m \times 2^n$ 种状态

5、指令的操作类型

1、数据传送：CPU、主存直接的数据传送

MOV：寄存器->寄存器

LOAD：主存->寄存器

STORE：寄存器->主存

PUSH：进栈

POP：出栈

2、运算类

算术逻辑操作、移位操作

3、程序控制类

JMP：无条件跳转
BRANCH：条件转移
CALL：调用
RET：返回
TRAP：陷阱

4、输入输出操作

CPU \leftrightarrow IO设备

4.2指令的寻址方式

1、指令寻址

(1) 顺序寻址

指令运行完成后，程序计数器PC+1（一条指令长度）

(2) 跳跃寻址

绝对转移：目标指令地址=地址码

相对转移：目标指令地址=当前地址+地址码+一条指令长度

2、数据寻址

形式地址A：指令中给出的地址

有效地址EA：操作数的**真实地址**

(A)表示地址为A的存储单元的**内容**

(1) 隐含寻址

第一个操作数：(A)

第二个操作数（隐含）：ACC

(2) 立即数寻址

操作数=A

(3) 直接寻址

操作数=(A)

(4) 间接寻址

操作数=((A))

(5) 寄存器寻址

地址字段给出**寄存器编号**

操作数=R（寄存器的内容）

(6) 寄存器间接寻址

操作数=(R)

(7) 相对寻址

广泛用于**转移指令**

一般就是转移到PC+A+1的指令

(取出本条指令后PC自动+1条指令长度)

(8) 基址寻址

基址寄存器BR

操作数=((BR)+A)

一般BR不变，A作为偏移量

用于**多道程序**设计，扩大寻址范围，程序浮动

(9) 变址寻址

变址寄存器IX

操作数=((IX)+A)

一般A不变，IX作为偏移量

用于**数组**寻址

(10) 堆栈寻址

4.3程序的机器级代码表示

1、基础概念

一共**8个**32位寄存器

EAX的低16位为AX，AX分为高8位AH，低8位AL（其他类似）

通用寄存器				16bit	32bit	说明
31	16	15	8 7 0	AX	EAX	累加器 (Accumulator)
				BX	EBX	基地址寄存器 (Base Register)
				CX	ECX	计数寄存器 (Count Register)
				DX	EDX	数据寄存器 (Data Register)
					ESI	变址寄存器 (Index Register)
					EDI	
					EBP	堆栈基指针 (Base Pointer)
					ESP	堆栈顶指针 (Stack Pointer)

图 4.11 x86 处理器中的主要寄存器及说明

两种汇编格式AT&T和Intel，以下只有统考要求的Intel

!规则!:

第一个为**目的操作数**，第二个为**源操作数**

[]表示**存储的内容**，如：[eax]为eax寄存器的内容

操作码后面显式注明**操作数大小**byte ptr、word ptr、dword ptr（字节、字、双字）

2、常用指令

指令说明：

<reg>任意寄存器，<reg32>32位寄存器
<mem>内存地址
<con>常数，<con8>8位常数，<con16>16位常数，<con32>32位常数

(1) mov指令

```
mov eax,<mem>
```

将源操作数复制到目的操作数

不能从内存复制到内存

(2) push指令

```
push [var]      将var的值入栈
```

寄存器ESP是栈顶，入栈前先-4

栈增长方向与内存地址增长方向相反

操作数压入内存的栈

(3) pop指令

```
pop eax      将栈顶内容送到eax
```

将操作数出栈

(4) add/sub指令

将两个操作数相加/相减，送到第一个操作数

(5) inc/dec指令

```
inc <reg>  
dec <mem>
```

操作数自增/自减（只有一个操作数）

(6) imul指令

```
imul <reg>,<mem>,<con>
```

有符号数乘法

两个操作数：相乘后存入第一个操作数，第一个操作数必须为寄存器

三个操作数：第二、三个操作数相乘后存入第1个操作数，第一个操作数必须为寄存器

溢出时OF=1

(7) idiv指令

有符号数除法

```
idiv <reg32>  
idiv <mem>
```

有符号数除法

操作数：除数

EDX:EAX：被除数，共64位

EAX：商

EDX：余数

(8) **and/or/xor**指令

```
and eax,0fh
```

逻辑与/或/异或

两个操作数，操作结果存入**第一个操作数**

(9) **not**指令

```
not <reg>
```

位翻转，1->0，0->1

(10) **neg**指令

```
neg <reg>
```

取负指令

(11) **shl/shr**指令

逻辑移位指令，shl左移，shr右移

左移右移都是**添加0**

第一个是被操作数，第二个是移位的**位数**

(12) **jmp**指令

跳转到指定label的地址

```
begin: mov eax,ebx  
jmp begin
```

(13) **jcondition**指令

经常与cmp连用

```
je      ==跳转
jz      ==0跳转
jne     !=跳转
jg      >跳转
jge     >=跳转
jl      <跳转
jle     <=跳转
```

(14) **cmp/test**指令

cmp相当于sub，比较两个数的值

test相当于and，**按位相与**

不保存结果，仅修改**状态字**

(15) **call/ret**指令

程序调用和返回

call将当前地址入栈并转移到label处

ret出栈栈顶的地址，并跳转

```
call <label>
ret
```

4.4 CISC和RISC的基本概念

	CISC	RISC
指令系统	复杂、庞大	简单、精简
指令数目	一般大于200	一般小于100
指令字长	不固定	固定
可访存指令	不加限制	只有LOAD/STORE
各种指令执行时间	相差较大	一个周期内
各种指令使用频率	相差较大	都比较常用
通用寄存器数量	少	多
目标代码	难以优化	可以优化
控制方式	微程序	组合逻辑
指令流水线	可以实现	必须实现

第五章 中央处理器