计算机组成
课程总复习
(2017学年)

课程成绩及期终考试

- 课程成绩
  - 平时成绩15%
  - 期终笔试成绩85%
- 试题类型
  - 选择题、简答题、设计题、分析题、编程题
- 试题知识点分布
  - 数字逻辑部分： 约25%
  - 汇编语言部分： 约15%
  - 组成原理部分： 约60%

课程内容概况

序号	内容
第一讲	计算机组成概述
第二讲	组合逻辑设计
第三讲	时序逻辑设计
第四讲	主存储器
第五讲	指令系统与MIPS汇编语言
第六讲	MIPS处理器设计
第七讲	高速缓冲存储器
第八讲	虚拟存储系统
第九讲	总线与输入输出方式

第一讲：计算机组成概述（4学时）

- 目标
  - 了解计算机系统的基本功能、组成框架、典型结构及层次关系，掌握计算机中数的表示方法及常用编码。
- 主要内容
  - 计算机系统的基本组成
  - 计算机系统的典型架构与层次关系
  - 计算机中数的表示
    - 定点数的表示（原码、反码、补码）
    - 浮点数的表示
    - 其他编码（格雷码、循环码、ASCII码、汉字编码）
  - 计算机的程序执行原理简介
    - 指令的含义简介
    - 程序的执行过程简介

第二讲：组合逻辑设计（8学时）

- 目标
  - 了解门电路的基本结构，掌握布尔代数的理论及其门电路实现方法，进而掌握布尔方程表示、转换及化简等方法，以及运算单元、译码器等基本组合逻辑部件设计方法，学习并掌握Verilog HDL。
- 主要内容
  - 逻辑门电路
    - 非门、与门、或门、复合逻辑门电路及其性能指标
    - TTL、MOS集成门电路
  - 布尔代数
    - 布尔代数基本原理
    - 逻辑函数表达式：标准表达式（最小项表达式、最大项表达式）
    - 逻辑函数表达式的简化法：（合并乘积项法、吸收项法、配项法）
  - Verilog HDL介绍（自学）
  - 基本组合逻辑部件设计与分析
    - 运算单元电路（加法器、比较器）
    - 多路选择器，译码器，编码器

内容概要

- 逻辑门电路：数字电路中的基本逻辑单元电路
  - 由晶体管和MOS管（晶体二极管、晶体三极管、NMOS、PMOS）构建门电路（与、或、非、与非、或非等）
- 布尔代数：分析与设计数字系统的重要理论工具
  - 逻辑代数基本概念：逻辑常量/变量，典型逻辑运算
  - 逻辑代数的运算法则：公理、定律、定理、基本公式及其推论
  - 逻辑函数的表达式：真值表 -> 最小项表达式、最大项表达式
  - 逻辑函数的简化法：合并乘积项法、吸收项法、配项法
- 硬件描述语言：Verilog HDL（自学）
  - Verilog HDL的模块、词法、常用语句
  - 不同抽象级别的Verilog HDL模型：行为描述和结构描述
- 基本组合逻辑部件设计
  - 运算单元电路： 加法、减法、乘法、比较器、ALU
  - 编码器/译码器： 三种编码器/译码器
  - 多路选择器： 数据选择、多功能运算

最小项表达式和最小项推导法

1. 最小项表达式

- 全部由最小项构成的与或式，也称标准与或式，可由最小项推导法直接从真值表中导出。
- 例如：三人表决器设计的输出表达式

$$F = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$F(A,B,C) = m_3 + m_5 + m_6 + m_7$$

$$F(A,B,C) = \sum m(3,5,6,7)$$

最简略

最小项表达式

真值表

A B C	F
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

- 最小项推导法 —— 从真值表推出逻辑函数表达式的一种方法
  - 使输出为1的输入组合写成乘积项的形式，其中取值为1的输入用原变量表示，取值为0的输入用反变量表示，然后把这些乘积项加起来。

逻辑函数的简化法 —— 逻辑函数的公式简化法

- 逻辑函数的公式简化法的原理是反复使用逻辑代数的基本公式、基本定理和常用公式，消去函数中多余的乘积项和因子，以求得最简形式。

一、“与或”表达式的化简

- 最简与或表达式
  - 1、乘积项的个数最少（用门电路实现，用的与门数最少）；
  - 2、在满足1的条件下，乘积项中的变量最少（与门的输入端最少）。
  - 省器件：用最少的门，门的输入也最少。
- 常用的化简方法有：合并乘积项法、吸收项法和配项法

逻辑函数的简化法 —— 合并乘积项法

❖逻辑函数的公式简化常用的方法（以与或表达式的化简为例）有：合并乘积项法、吸收项法、配项法、消除冗余项法

1、合并乘积项法——利用互补律消去1个变量

化简 $F = A(BC + \overline{B}\,\overline{C}) + AB\overline{C} + A\overline{B}C$

解：
$$F = ABC + A\overline{B}\,\overline{C} + AB\overline{C} + A\overline{B}C$$ 利用分配律展开
$$= (ABC + A\overline{B}C) + (A\overline{B}\,\overline{C} + AB\overline{C})$$ 合并
$$= AC(B + \overline{B}) + A\overline{C}(\overline{B} + B)$$ 互补律
$$= AC + A\overline{C}$$ 互补律
$$= A(C + \overline{C}) = A$$

逻辑函数的简化法 —— 吸收项法和配项法

2、吸收项法——利用吸收律和包含律减少“与”项

化简 $F = A\overline{B} + \overline{A}B + ABCD + \overline{A}\,\overline{B}CD$

解：
$$F = (A\overline{B} + \overline{A}B) + (AB + \overline{A}\,\overline{B})CD$$ 合并乘积项
$$= (A\overline{B} + \overline{A}B) + \overline{A\overline{B} + \overline{A}\,\overline{B}}\,CD$$ “同或”和“异或”互为反函数
$$= A\overline{B} + \overline{A}B + CD$$

由吸收律3 $A+\overline{A}B=A+B$

3、配项法——利用互补律，配在乘积项上

化简 $F = AB + \overline{A}\,\overline{B}C + BC$

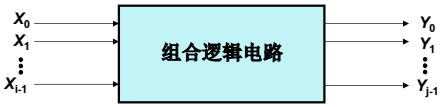
解：
$$F = AB + \overline{A}\,\overline{B}C + BC(A + \overline{A})$$ 配项
$$= AB + \overline{A}\,\overline{B}C + ABC + \overline{A}BC$$ 展开
$$= (AB + ABC) + (\overline{A}\,\overline{B}C + \overline{A}BC)$$ 合并
$$= AB(1 + C) + \overline{A}C(B + \overline{B})$$ 1律、互补律
$$= AB + \overline{A}C$$

组合逻辑电路的结构和特点

❖ 数字电路分类：组合逻辑电路和时序逻辑电路

❖ 组合逻辑电路

- 是将逻辑门以一定的方式组合在一起，使其具有一定逻辑功能的数字电路。
- 是一种无记忆电路 —— 任一时刻的输出信号仅取决于该时刻的输入信号，而与信号作用前电路原来所处的状态无关。
- 常用的组合逻辑电路：算术逻辑运算电路、编码器/译码器、数据选择器、数值比较器、奇偶校验器等



❖ 特点

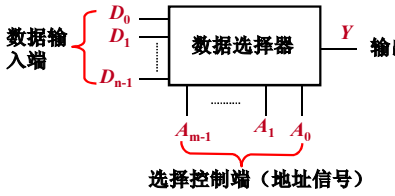
- 由逻辑门电路组成
- 输出不能再直接反馈到输入（不能有环路），没有存储电路
- 当时的输出仅由当时的输入决定——速度快

数据选择器

❖ 从一组输入数据选出其中需要的一个数据作为输出的过程叫做数据选择，具有数据选择功能的电路称为数据选择器（Data Selector）。

❖ 数据选择器又称多路选择器（Multiplexer，多路器），它是以“与或非”门 或 以“与或”门为主体的组合逻辑电路。它在选择控制信号的作用下，能从多路平行输入数据中任选一路数据作为输出。

❖ 常用的集成数据选择器有四2选1（74××157）、双4选1（74××153）、8选1（74××151）及16选1（74××150）数据选择器等。

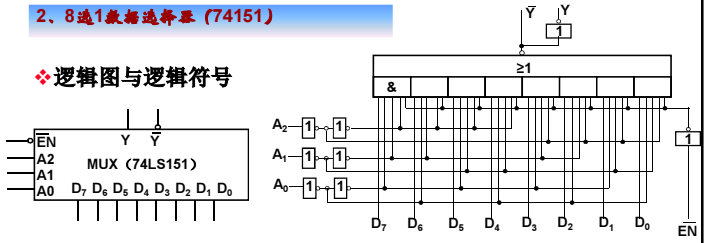


若有n个输入，则称为n选1数据选择器。

8选1数据选择器的功能（1）

2、8选1数据选择器（74151）

❖逻辑图与逻辑符号



$$Y = \overline{A_2}\,\overline{A_1}\,\overline{A_0}D_0 + \overline{A_2}\,\overline{A_1}A_0D_1 + \overline{A_2}A_1\overline{A_0}D_2 + \overline{A_2}A_1A_0D_3 + A_2\overline{A_1}\,\overline{A_0}D_4 + A_2\overline{A_1}A_0D_5 + A_2A_1\overline{A_0}D_6 + A_2A_1A_0D_7 \quad (\overline{EN} = 0)$$
$$= m_0D_0 + m_1D_1 + m_2D_2 + m_3D_3 + m_4D_4 + m_5D_5 + m_6D_6 + m_7D_7$$

❖功能①

$A_2A_1A_0$为控制端——

8选1数据选择器（8路开关）

在/EN=0时

8选1数据选择器的功能（2）

功能表（EN=0）

A_2	A_1	A_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

功能②：$D_7 \sim D_0$为控制端——多功能运算电路

- 通过D_7~D_0取不同的值，从输入变量A_2、A_1、A_0的各个最小项中选取某几个最小项输出，实现不同的运算电路
- 有2^8=256种功能——包含3变量的各种最小项表达式——可实现任意组合逻辑电路的设计。

$$Y = D_0m_0 + D_1m_1 + D_2m_2 + D_3m_3 + D_4m_4 + D_5m_5 + D_6m_6 + D_7m_7$$

当$D_7 \sim D_0$为0000_0000时，Y=0；
当$D_7 \sim D_0$为1111_1111时，Y=1；
当$D_7 \sim D_0$为0000_0001时，Y=m_0；
当$D_7 \sim D_0$为1010_0101时，Y=m_7+m_5+m_2+m_0。

利用数据选择器实现逻辑函数

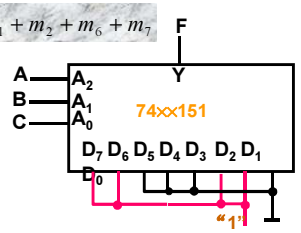
利用数据选择器实现逻辑函数

$$F(A,B,C) = \overline{A}BC + A\overline{B}C + AB$$

解：使用8选1数据选择器74151
将逻辑函数的输入变量作为数据选择器的地址输入。
首先将组合逻辑函数变换为最小项之和的标准形式：

$$F = \overline{A}BC + A\overline{B}C + AB(\overline{C}+C) = m_1 + m_2 + m_6 + m_7$$

74151 输出 $Y = D_0m_0 + D_1m_1 + D_2m_2 + D_3m_3 + D_4m_4 + D_5m_5 + D_6m_6 + D_7m_7$



比较*F*和*Y*，得：
D_0=0，D_1=1，D_2=1，D_3=0，
D_4=0，D_5=0，D_6=1，D_7=1

第三讲：时序逻辑设计（8学时）

❖ 目标
- 掌握触发器、寄存器的结构和工作原理，掌握有限状态机、同步时序逻辑电路的设计方法和分析方法，具备使用仿真工具开发时序逻辑电路的能力。

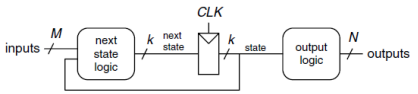
❖ 主要内容
- 锁存器和触发器
  - SR锁存器、D锁存器
  - D触发器
  - JK触发器
- 有限状态机（FSM）
  - Moore型FSM
  - Mealy型FSM
- 时序逻辑电路设计分析
  - 数据寄存器
  - 移位寄存器
  - 计数器

锁存器和触发器

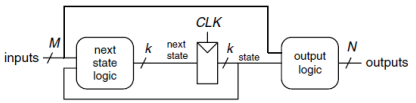
- ❖基本RS锁存器：具有保持、置0、置1功能，其输入信号可以直接控制锁存器的输出
- ❖钟控RS锁存器：CP有效（高电平或低电平）时，锁存器的状态随输入变化（约束条件R、S不能同时为1）
- ❖钟控D锁存器：为消除钟控RS锁存器的不定状态，将钟控RS锁存器双端输入改为单端输入（D），即D锁存器
- ❖D触发器：两个反相钟控D锁存器构成D触发器。时钟信号CP的边沿（上升沿或下降沿）触发
- ❖JK触发器：是一种功能全面，而且没有约束条件的FF
- ❖寄存器：由共享时钟信号CLK的多个D触发器构成

有限状态机

- ❖根据输出信号产生的机理不同，有限状态机可分成两类：
  - ➢摩尔(Moore)型状态机： 输出信号仅与当前状态有关
  - ➢米里(Mealy)型状态机： 输出信号与当前状态及输入信号有关



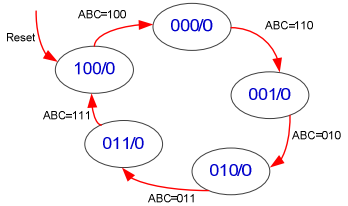
Moore型有限状态机



Mealy型有限状态机

Moore型FSM设计

- ❖ Moore型FSM的表示方法
  - ➢状态图（State Diagram）：圆圈表示状态，圈内“Q2Q1Q0 / D”分别表示状态组合Q2Q1Q0（或状态编码）及输出信号D；带箭头的线段表示状态转移，线段上的文字表示转移发生时的信号输入
  - ➢状态表（State Table）：状态转换表，反映下一状态与当前状态和输入的关系

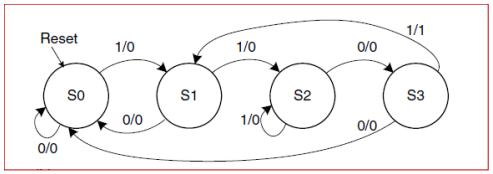


$Q_2Q_1Q_0$ / D

Moore型FSM状态转换图

Mealy型FSM设计

- ❖ Mealy型FSM的表示方法 —— 状态图
  - ➢圆圈表示状态，带箭头的线段表示状态转移
  - ➢状态圈内“S0”、“S1”等代表状态名（对应状态编码）
  - ➢与Moore型FSM不一样的是，输出信号不再标注在圈内，而是以“输入/输出”的形式标注在状态转移线上：“输入”表示引起状态转换的输入信号； “输出”表示状态转换同时产生的输出信号



Mealy型FSM状态转换图

FSM设计

【例】二进制序列检测器：检测器接收到二进制序列“1101”时，输出检测标志为1，否则输出检测标志为0。注: 不重复检测，即收到1101输出1后，下一次从下一个输入信号开始检测。

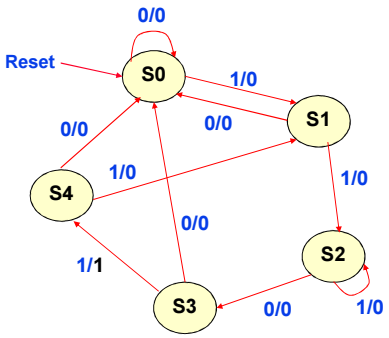
解：

（1）检测器FSM模型

- 输入：二进制序列输入信号A，1位
- 输出：检测标志信号Y，1位
- 状态：共5个不同状态
  - S0：未收到第一个有效位（输入为0，输出0）
  - S1：收到第一个有效位（输入为1，输出0）
  - S2：收到第二个有效位（即S1后输入为1，输出0）
  - S3：收到第三个有效位（即S2后输入为0，输出0 ）
  - S4：连续收到四个有效位（即S3后输入为1，输出1 ）
- 状态寄存器：3位

FSM设计

（2）画出状态转换图



“1101”检测器FSM模型：

S0：未收到第一个有效位（输入为0，输出0）

S1：收到第一个有效位（输入为1，输出0）

S2：收到第二个有效位（即S1后输入为1，输出0）

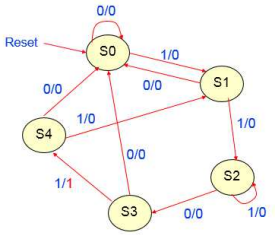
S3：收到第三个有效位（即S2后输入为0，输出0 ）

S4：连续收到四个有效位（即S3后输入为1，输出1 ）

FSM设计

（3）根据状态转换图得到状态转换表

当前状态（$S_2S_1S_0$）	输入（A）	下一状态（$S'_2S'_1S'_0$）	输出（Y）
S0（000）	0	S0（000）	0
S0（000）	1	S1（001）	0
S1（001）	0	S0（000）	0
S1（001）	1	S2（010）	0
S2（010）	0	S3（011）	0
S2（010）	1	S2（010）	0
S3（011）	0	S0（000）	0
S3（011）	1	S4（100）	1
S4（100）	0	S0（000）	0
S4（100）	1	S1（001）	0



FSM设计

（4）根据状态转换表写出 次态逻辑表达式 和 输出逻辑表达式

$$S_2' = \overline{S}_2 S_1 S_0 A$$

$$S_1' = \overline{S}_2\overline{S}_1 S_0 A + \overline{S}_2 S_1 \overline{S}_0 \overline{A} + \overline{S}_2 S_1 \overline{S}_0 A = \overline{S}_2\overline{S}_1 S_0 A + \overline{S}_2 S_1 \overline{S}_0$$

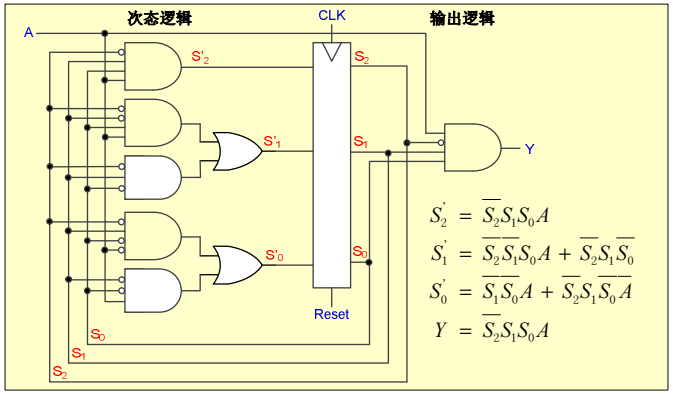
$$S_0' = \overline{S}_2 S_1 \overline{S}_0 A + \overline{S}_2 S_1 \overline{S}_0 \overline{A} + S_2 \overline{S}_1 \overline{S}_0 A = \overline{S}_1 S_0 A + \overline{S}_2 S_1 \overline{S}_0 \overline{A}$$

$$Y = \overline{S}_2 S_1 S_0 A$$

当前状态（$S_2S_1S_0$）	输入（A）	下一状态（$S'_2S'_1S'_0$）	输出（Y）
S0（000）	0	S0（000）	0
S0（000）	1	S1（001）	0
S1（001）	0	S0（000）	0
S1（001）	1	S2（010）	0
S2（010）	0	S3（011）	0
S2（010）	1	S2（010）	0
S3（011）	0	S0（000）	0
S3（011）	1	S4（100）	1
S4（100）	0	S0（000）	0
S4（100）	1	S1（001）	0

FSM设计

（4）根据逻辑表达式画出 逻辑图



第四讲：主存储器（4学时）

- 目标
  - 了解存储单元电路的工作原理，掌握主存储器的结构特点、工作原理和构造方法。
- 主要内容
  - 存储单元电路
    - SRAM存期单元电路
    - DRAM存储单元电路
    - ROM存储单元电路
  - 主存储器的结构
    - SRAM芯片的内部结构
    - DRAM芯片的内部结构
  - 存储器的扩展
    - 芯片容量的基本描述（字单元数 X 每个字单元的位数，2n x m）
    - 存储器的扩展方法
  - DRAM的刷新

存储芯片和存储器的容量描述

- 存储芯片容量的基本描述（字单元数×每个字单元的位数）
  - 1K × 2：1024 个字单元，每个字单元 2 位（二进制位）

意味着任一时刻可以（也只能）访问1024个独立字单元中的

任意一个，每次读写的数据位数是一个字单元的容量（2位）

对于1K X 2的存储芯片：

有多少个存储位元？共1K个（1024个）字单元，每个字单元2位 2048

需多少条地址线？按字单元寻址，1024个（2^{10}个）字单元 10

需要多少条数据线？一次访问一个字单元，每个字单元是2位 2

  - 64K×8：65536（64K）个字单元，每个字单元8位，也即64KB

有多少个存储位元？需要多少条地址线？多少条数据线？

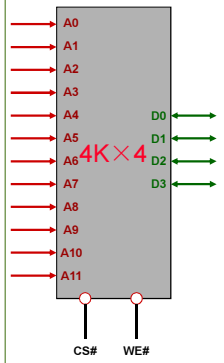
主存储器扩展方法

- 混合扩展的基本思路
  1. 确定每个芯片的地址位数、数据位数。
  2. 确定整个存储空间所需的地址总线和数据总线的数量。
  3. 计算所需芯片的数量，确定每个芯片在整个存储空间中的地址空间范围、位空间范围
  4. 所有芯片的地址全部连接到地址总线对应的地址线上。
  5. 同一字空间的存储芯片CS信号连在一起。
  6. 同一位空间的数据线连在一起，并连接到对应的数据总线上。
  7. 根据每个芯片的地址空间范围，设计芯片所需要的片选信号逻辑，CS逻辑电路的输入一定是地址总线中没有连接到芯片的地址管脚上的那部分地址线。
  8. 统一读写控制。

存储器芯片的扩展 —— 混合扩展

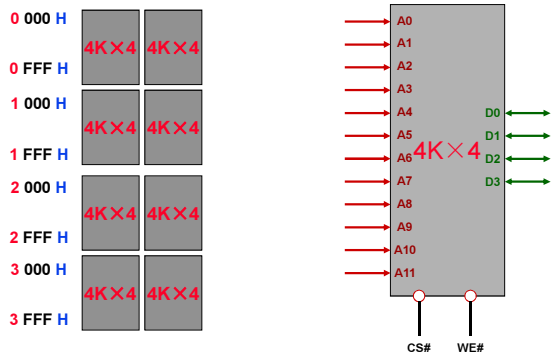
例：4Kx4 SRAM存储芯片构成16Kx8的存储器

- 4K×4芯片：
  - 12个地址管脚 A11～A0
  - 4个数据管脚 D3～D0
  - 1个片选输入管脚 CS＃
  - 1个读写控制管脚 WE#
  - 芯片地址空间：000H～FFF H
- CPU向存储器提供：
  - 地址总线14根：AB13～AB0
  - 数据总线8根：DB7～DB0
  - 读写控制信号：MemW
  - 存储器地址空间：0000H～3FFF H
- 需要芯片数：（16K×8)/(4K×4）= 4X2= 8片
  - 分4组（字扩展），每组2个芯片（位扩展）
- 一个2-4译码器产生4个片选信号
  - 译码器输入：AB13~AB12
  - 译码器输出：分别接4组芯片片选管脚



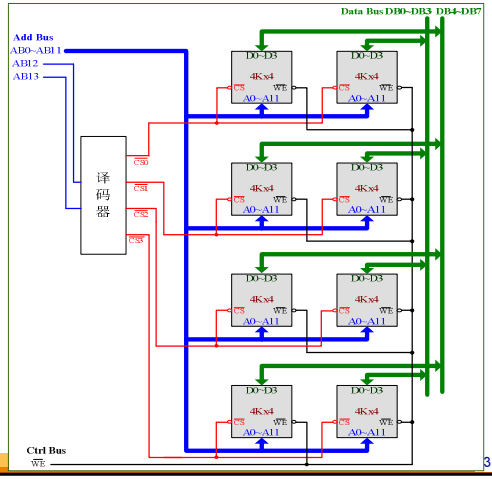
存储器芯片的扩展 —— 混合扩展

4Kx4 SRAM存储芯片构成16Kx8的存储器地址空间划分



存储器芯片的扩展 —— 混合扩展

4Kx4 SRAM存储芯片构成16Kx8的存储器连接图



存储器芯片的扩展示例 —— 异种芯片

CPU地址线A15～A0，数据线D7～D0，$\overline{WR}$为读/写信号，$\overline{MREQ}$为访存请求信号。0000H～3FFFH为系统程序区，4000H～FFFFH为用户程序区。用8K×4位ROM芯片和16K×8位RAM芯片构成该存储器，要求说明地址译码方案，并将ROM芯片、RAM芯片与CPU连接。

解：因为0000H～3FFFH为系统程序区，ROM区高两位总是00，低14位为全译码。

ROM区大小为：2^{14}×8位=16K×8位=16KB

ROM芯片数为：16K×8位 / 8K×4位 = 2×2 = 8，字方向扩展2倍，位方向扩展2倍

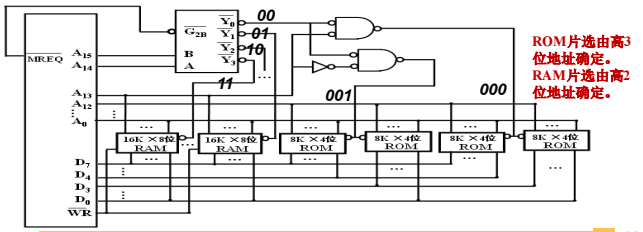
ROM芯片内地址位数为13位，连到CPU低13位地址线A12～A0

因为4000H～FFFFH为用户程序区，RAM区高两位是01、10、11，低14位为全译码。

RAM区大小为：3×2^{14}×8位=3×16K×8位= 48KB

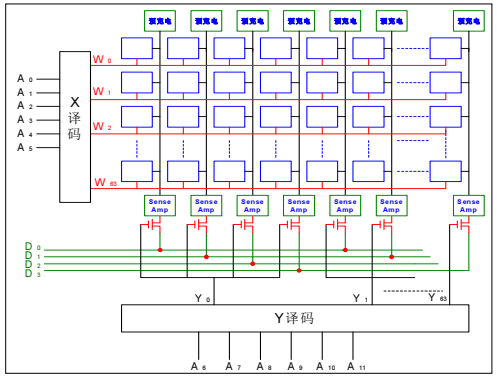
RAM芯片数为：48K×8位 / 16K×8位 = 3×1 = 3，字方向上扩展3倍，位方向上不扩展。

RAM芯片内地址位数为14位，连到CPU低14位地址线A13～A0。



DRAM刷新

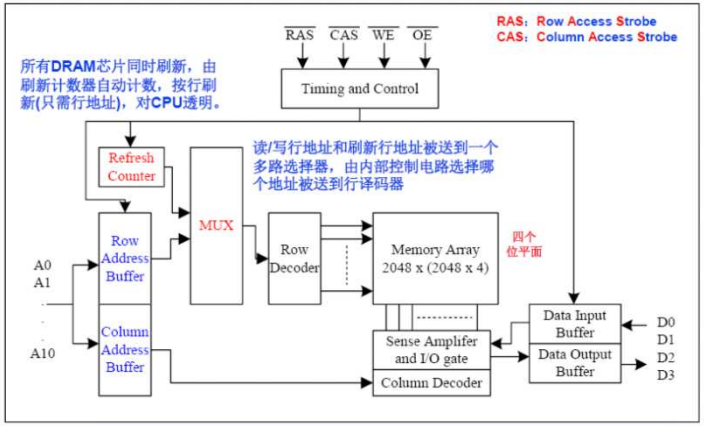
❖ 二维地址结构（4096* 4 DRAM）



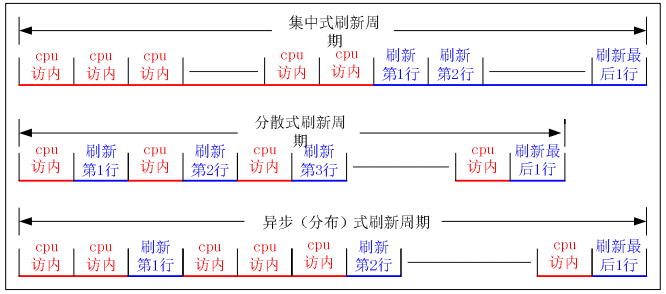
按行刷新，每次刷新1行

存储芯片示例

❖ DRAM 4M*4 DRAM芯片结构



DRAM刷新方式



第五讲：指令系统与MIPS汇编语言（6学时）

❖ 目标

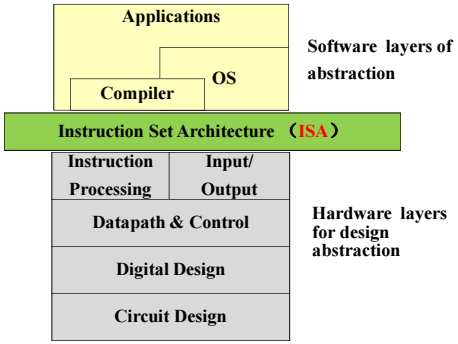
- 以MIPS两种指令系统为研究对象，学习并掌握计算机指令系统的格式、寻址方式和设计方法；学习并掌握MIPS汇编语言编程。

❖ 主要内容

- 指令系统概述
  - 指令系统的基本要素
  - 指令格式
  - 寻址方式
- MIPS指令系统
- MIPS汇编语言编程

指令系统概述

- 指令集系统结构(ISA)
  - 机器语言编程者的视角，机器内部结构和行为能力的指令级抽象



指令格式

- 机器指令：计算机硬件可以执行的表示一种基本操作的二进制代码
  - 指令格式：操作码 + 操作数（操作数地址）
  - 操作码：指明指令的操作性质
  - 操作数（地址）：指明操作数的位置（或操作数本身）

操作码	操作数地址
11010101 10000100 01010001 10100000	

- 指令的表示
  - 机器表示：二进制代码
  - 符号化表示：助记符，如： MOV AX, BX

操作数

- 操作数的类型
  - 数值（无符号、定点、浮点）
  - 逻辑型数、字符
  - 地址（操作数地址、指令地址）
- 操作数的位置
  - 存储器（存储器地址）
  - 寄存器（寄存器地址）
  - 输入输出端口（输入输出端口地址）
- 操作数的存储方式
  - 大端（big-endian）次序：最高有效字节存储在地址最小位置
  - 小端（little-endian）次序：最低有效字节存储在地址最小位置

例： Int a; //0x12345678

地址	值
a+0	12
a+1	34
a+2	56
a+3	78

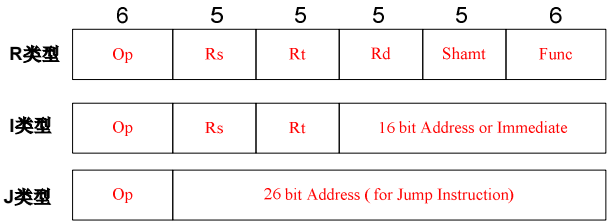
大端次序

地址	值
a+0	78
a+1	56
a+2	34
a+3	12

小端次序

指令系统举例 —— MIPS指令系统

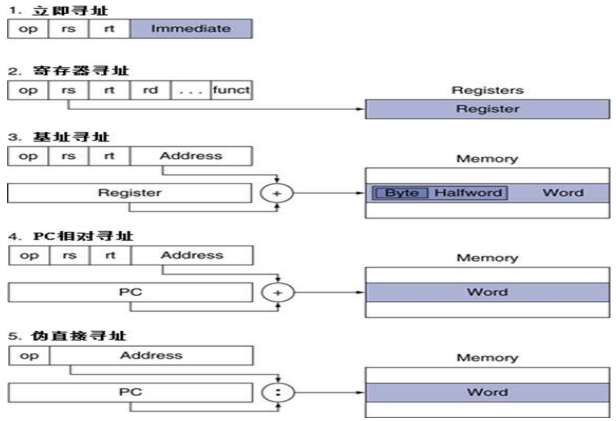
- MIPS 指令格式



- MIPS 指令格式举例
  - R类型：add, addu, sub, subu, and, or, jr 等
  - I 类型： addi, ori, lw, sw, beq, bnq 等
  - J类型：j, jal 等

指令系统举例 —— MIPS指令系统

❖ MIPS寻址方式



指令系统举例 —— MIPS指令系统

❖ Load/Store（取数/存储）指令

- I类型指令，存储器与通用寄存器之间传送数据
- 支持唯一的存储器寻址方式：Base+Index
- 取数指令：LB（取字节）、LBU（取不带符号字节）、LH（取半字）、LHU（取不带符号的半字）、LW（取字）、LWL、LWR
- 存储指令：SB（存字节）、SH（存半字）、SW（存字）、SWL、SWR

❖ 运算指令

- R类型指令 和 I类型指令
- 算术运算：add, addu, addi, addiu, sub, subu, mul, mulu, div, divu, mfhi, mflo等
- 逻辑运算：and, andi, or, ori, xor, xori, nor等
- 移位指令：sll, srl, sra, sllv, srlv, srav等

指令系统举例 —— MIPS指令系统

❖ 跳转和转移指令：控制程序执行顺序

- 跳转指令：J类型指令（26位绝对转向地址）或R类型指令（32位的寄存器地址）
- 转移指令：I 类型指令，PC-relative寻址方式，相对程序计数器的16位位移量（立即数）。
- 跳转：J 、JAL、JR、JALR
- 转移：BEQ（相等转移）、BNE（不等转移）、BLEZ（小于或等于0转移）、BGTZ（大于0 转移）、BLTZ（小于0转移）、BLTZAL、BGEZAL

❖ 特殊指令

- R类型指令
- 系统调用SYSCALL
- 断点BREAK

第六讲：MIPS处理器设计（16学时）

❖ 目标

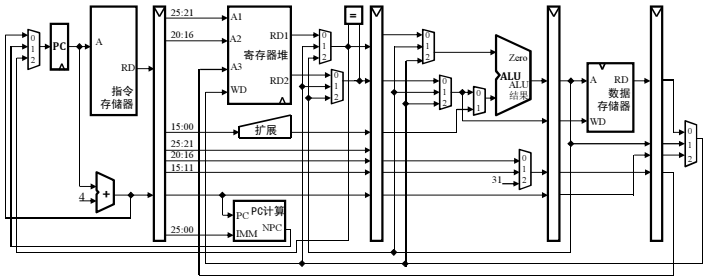
- 以小型MIPS处理器为研究对象，学习并掌握基于指令执行分析的数据通路构造方法、基于与或逻辑阵列为基础的MIPS控制器设计方法，进而掌握MIPS处理器设计方法。

❖ 主要内容

- 处理器的功能、组成、一般设计方法等
- MIPS处理器设计概述
  - 结构、指令集、数据通路的基本组件
- 单周期处理器设计
  - 单周期数据通路和控制器设计
  - 单周期处理器性能分析
- 流水线处理器设计
  - 流水线数据通路和控制器设计
  - 流水线处理器性能分析
  - 流水线冒险及其处理

标准流水线

- 流水线：以性能为目标的标准流水线
  - 数据冒险：转发、暂停
  - 控制冒险：分支比较前移、转发、暂停



时钟驱动的流水线时空图

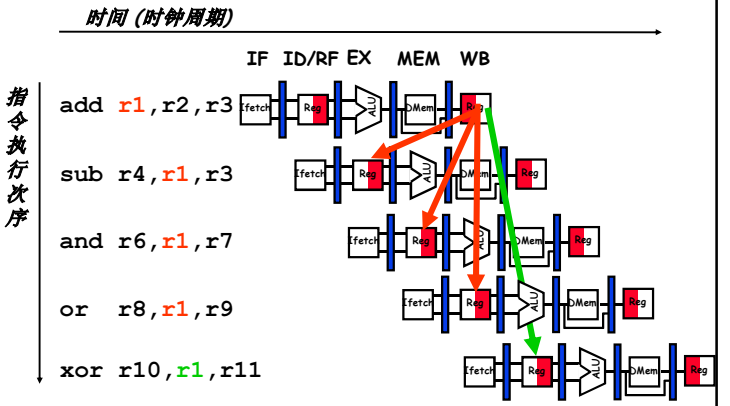
- 本图用途：需精确分析指令/时间/流水线3者关系时
  - 行：某个时钟，指令流分别处于哪些阶段
  - 列：某个部件，在时间方向上执行了哪些指令
- 注意区分流水阶段与流水线寄存器的关系
- 可以看出，在CLK5后，流水线全部充满
  - 所有部件都在执行指令
    - 只是不同的指令

相对PC的地址偏移	指令		IF级	ID/RF级	EX级	MEM级	WB级	
		CLK	PC	IM	IF/ID	ID/EX	EX/MEM	MEM/WB	RF
0	Instr 1	1	0→4	Instr 1	Instr 1				
4	Instr 2	2	4→8	Instr 2	Instr 2	Instr 1			
8	Instr 3	3	8→12	Instr 3	Instr 3	Instr 2	Instr 1		
12	Instr 4	4	12→16	Instr 4	Instr 4	Instr 3	Instr 2	Instr1	
16	Instr 5	5	16→20	Instr 5	Instr 5	Instr 4	Instr 3	Instr 2	Instr1
20	Instr 6	6	20→24	Instr6	Instr6	Instr5	Instr4	Instr3	Instr2

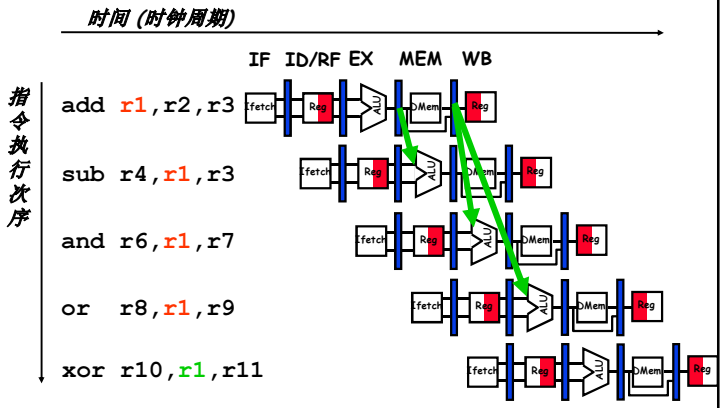
流水线冒险

- 结构冒险
- 数据冒险
  - 指令之间的数据相关
  - 转发
  - Load延迟槽
- 控制冒险
  - 分支和跳转延迟槽

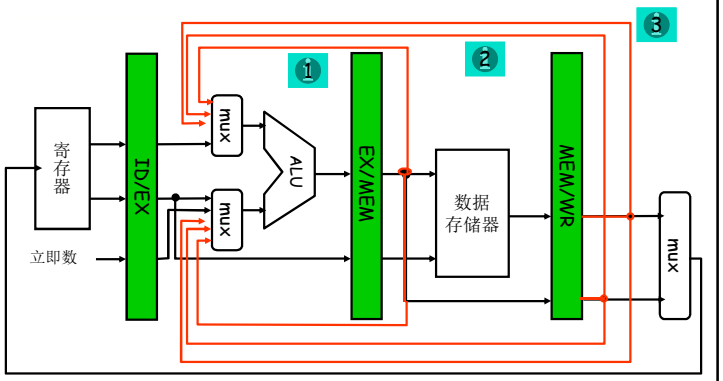
数据冒险 —— 数据相关性



数据冒险 —— 转发策略

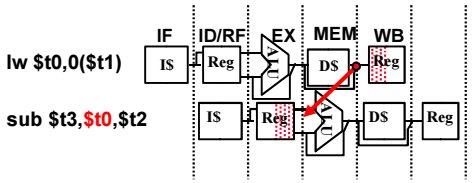


调整硬件结构支持转发



数据冒险：停顿

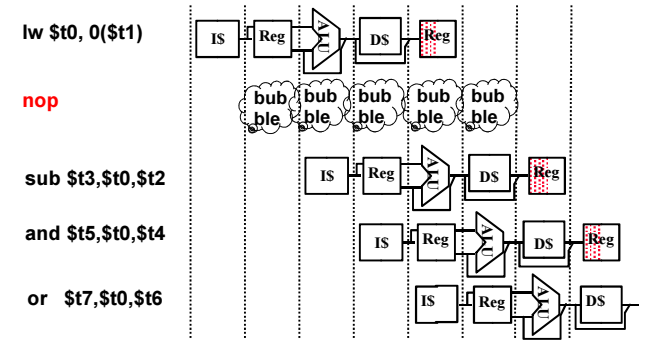
- 数据流在时间尺度上回流会带来风险



- 无法利用转发解决所有问题
  - 必须暂停与load相关的指令，然后再依靠转发

数据冒险：load延迟槽

- 停顿等价于 nop

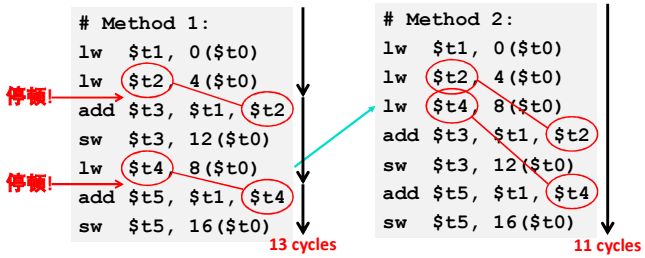


数据冒险： 代码调度

- Load之后的时隙称为 *load 延迟槽*
  - 如果延迟槽中的指令需要load的结果，硬件会互锁并停顿1个时钟周期
  - 用硬件暂停延迟槽中的指令等价于在延迟槽中插入一条nop指令
- 思路： 由编译器选择一条不相关的指令放入延迟槽→无需停顿!

数据冒险：通过代码调度避免流水线停顿

- 重排序代码避免load延迟槽中的指令使用load的结果!
- MIPS 代码： A=B+E； C=B+F；



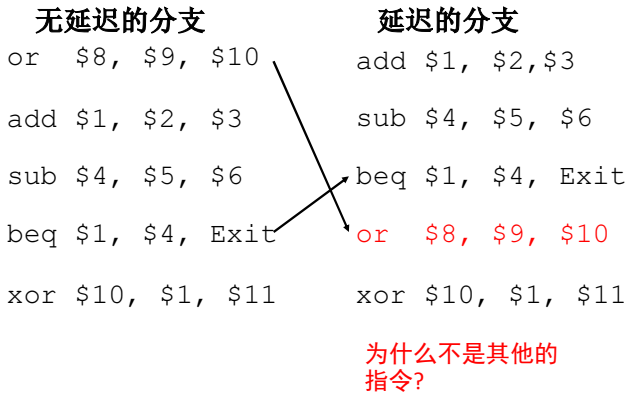
控制冒险：MIPS中延迟的跳转

- MIPS Green Sheet：jal

R[31]=PC+8； PC=JumpAddr

  - PC+8 因为有 *jump 延迟槽*!
  - PC+4 处的指令总是在jal跳转到label之前执行，所以返回地址是 PC+8

控制冒险：延迟的分支

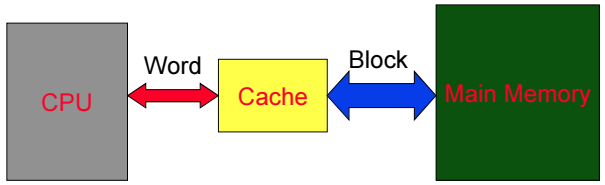


第七讲：高速缓冲存储器（CACHE）（6学时）

- 目 标
  - 掌握高速缓冲存储器（Cache）的结构特点和工作原理，以及多级Cache层次关系，掌握Cache的映射机制、Cache的命中与缺失分析及其性能计算方法。
- 主要内容
  - 程序执行局部性原理
  - Cache的结构与工作原理
  - Cache的映射机制
    - 直接映射
    - 全相联映射
    - 组相联映射
  - Cache的替换策略
  - Cache性能分析与其他
    - 容量计算
    - 性能分析
    - Cache数据一致性问题

高速缓冲存储器(Cache)的动机与原理

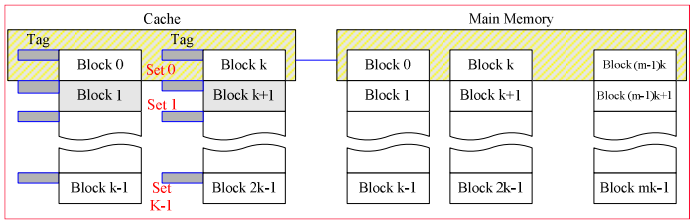
- 动机： 解决CPU和主存储器之间的性能差距问题
- Cache：CPU和主存间的一容量较小的高速缓存，其中总是存放最活跃（被频繁访问）的程序块和数据，大多数情况下，CPU能直接从这个高速缓存中取得指令和数据，而不必访问主存。
- Cache与主存之间按照数据块（Block）为单位进行数据交换。



Cache与主存之间的映射 —— 组相联

- 组相联（Set Associative）
  - 映射关系：Cache 分成 K 组，每组分成 L 块；主存的块 J 以下列原则映射到 Cache 的组 I 中的任何一块。

    $I = J \bmod K$
  - 实际上主存与Cache都分成 K 组，主存每一组内的块数与Cache一组内的块数不一致，主存组M内的某一块只能映射到Cache组M内，但可以是组M内的任意一块。



Cache与主存之间的映射 —— 组相联

- 组相联映射
  - 主存的地址格式：

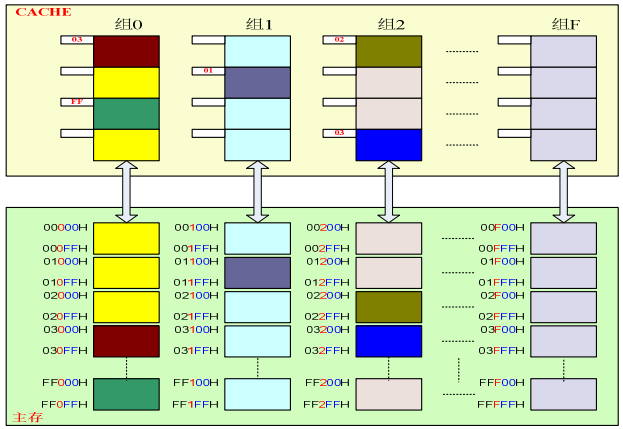
组内块地址(Tag)	组地址Set #	Offset

  - Tag的内容：主存中与该Cache数据块对应的数据块的组内块地址。
- 举例
  - 主存容量1M 字节，4路组相联（每组包含4个Block）Cache容量16K字节，Block大小256 字节
  - Cache分多少组？每组包含多少块？
  - Cache的Tag需要多少位？

解：

- Cache 组数＝ $2^{14} \div (2^8 \times 2^2) = 2^4 = 16$ 组
- 主存每组块数＝ $2^{20} \div (2^8 \times 2^4) = 2^8 = 256$ 块/组
- 主存地址：20 位，其中高8 位为组内块地址，中间4 位为组地址，低 8位为块内地址
- Cache的Tag应该为 8 位。

Cache与主存之间的映射 —— 组相联



Cache的性能计算

- **存储访问时间：对于cache和主存组成的两级存储系统**

  若：T_m为主存储器的访问周期；

  T_c为Cache的访问周期；

  H为Cache命中率

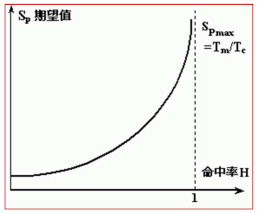
  则存储系统的等效访问周期T为：

$$T = T_c \times H + T_m \times (1-H)$$

- **加速比SP（Speedup）**

  存储系统的加速比S_p为：

$$S_p = \frac{T_m}{T} = \frac{T_m}{H \times T_c + (1-H) \times T_m} = \frac{1}{(1-H) + H \times \frac{T_c}{T_m}}$$



加速比与命中率的关系

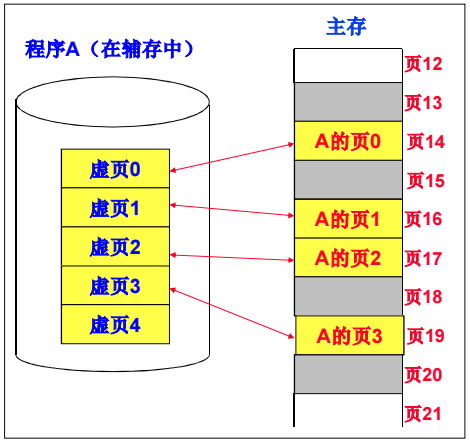
第八讲：虚拟存储系统（6学时）

- **目标**
  - 掌握虚拟存储器概念、工作原理、虚实地址转换与页表工作原理、TLB工作原理，具备进行虚拟存储器性能分析的能力。
- **主要内容**
  - 辅助存储器
  - 虚拟存储器的概念和作用
  - 虚拟存储器工作原理
  - 虚实地址转换
  - 页表工作原理
  - TLB工作原理
  - 虚拟存储器性能分析

页式虚拟存储器

虚页	实页号	有效位
0	14	1
1	16	1
2	17	1
3	19	1
4	XX	0
	XX	0
	XX	0

程序A的页表



页式虚拟存储器

❖ 举例

某计算机虚拟地址32位，物理内存128MB，页大小4KB。

（1）程序虚拟空间最多可有多少页？

（2）页表项共有多少位？

（3）每个页表占多少内存空间？

❖ 解答

虚地址32位：虚页号（20位）+ 页内偏移（12位）

实地址27位：实页号（15位）+ 页内偏移（12位）

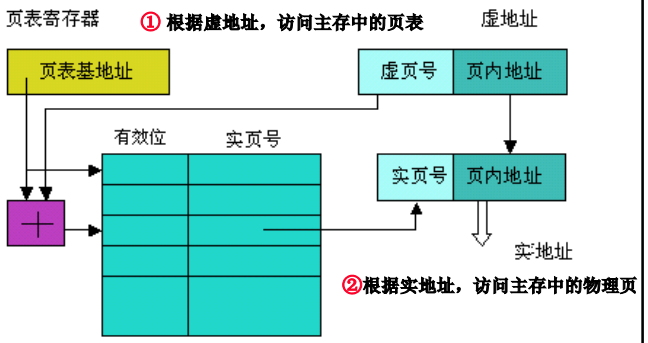
每个程序虚拟空间最多可有：2^{20}个虚页

每个页表项：1位（有效位）+ 15位（实页号）= 16位

每个页表所占空间：$2^{20} \times 16 = 16Mb = 2MB$

页式虚拟存储器

❖ 虚实地址的转换 —— 访存次数问题



页式虚拟存储器

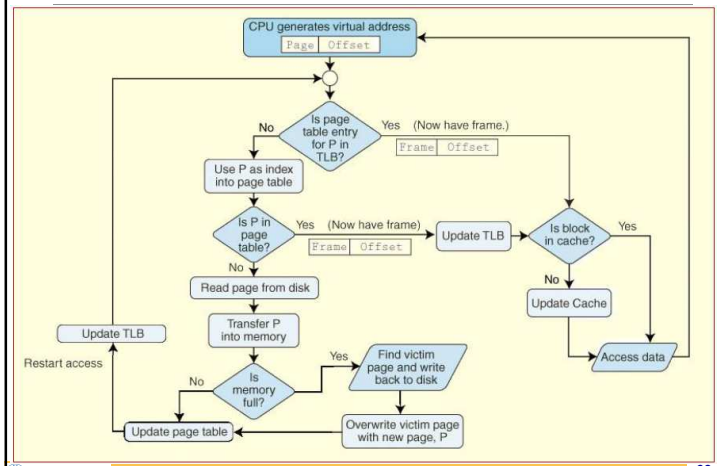
❖ 快表TLB（Translation Lookaside Buffer，转换后备缓冲器）

- 问题：每次虚拟存储器的访问带来两次存储器访问，一次访问页表，一次访问所需的数据（或指令），简单的虚拟存储器速度太慢
- 解决办法： 使用Cache存储部分活跃的页表项，称为TLB（快表），它包含了最近使用的那些页表项
- TLB内容：虚页号（标记）、对应实页号（数据）、有效位、修改位
- TLB一般采用全相联模式

有效位	修改位	标记（tag）	数据
		虚页号	实页号
		虚页号	实页号
		虚页号	实页号
		虚页号	实页号

快表（TLB）

页式虚拟存储器



页式虚拟存储器

假定页式虚拟存储系统按字节编址，逻辑地址36位，页大小16KB，物理地址32位，页表中包括有效位和修改位各1位、使用位和存期方式位各2位，且所有虚拟页都在使用中。请问：

（1）每个进程的页表大小为多少？

（2）如果所使用的快表（TLB）总表项数为256项，且采用2路组相联Cache实现，则快表大小至少为多少？

❖ 解答（1）

页面大小：16KB=2^{14}，页内偏移14位

虚地址36位：虚页号=36-14=22位

实地址32为：实页号=32-14=18位

每个进程最多可有：2^{22}个虚页

每个页表项：1 +1+2 +2+ 18=24位

每个页表所占空间：$2^{22}\times24$=12MB

（2）

- TLB：256个表项，2路组相联，所以共有128组
- 22位虚页号：7位组地址，15位Tag
- TLB每个表项：15+24=39位
- TLB容量：39×256=9984位=1248字节

第九讲：总线与输入输出方式（4学时）

❖ 目标

- 掌握程序查询I/O、中断I/O和DMA I/O等输入输出方式的工作原理。

❖ 主要内容

- 计算机I/O系统
- 总线
- I/O方式
  - 程序查询方式
  - 中断方式
  - DMA方式
  - I/O通道

I/O与主机信息交换的控制方式

- 程序查询方式
- 程序中断方式
- 直接内存访问(DMA)方式
- 通道方式

IO占用CPU时间

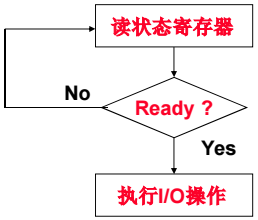
程序查询方式

❖I/O接口设置状态寄存器以表示外部设备的工作状态

❖CPU通过不断读取状态寄存器以查询外部设备的状态

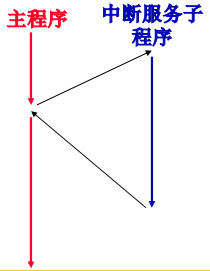
❖在外部设备准备就绪的时候，CPU通过I/O接口中的数据寄存器与外设完成数据交换。

```
RdSta:  MOV DX,3FDH
        IN  AL,DX
        CMP AL,61H
        JNE  RdSta
        MOV DX,3F8H
        IN AI,DX
```



中断方式

- 中断的概念
  - 概念：机器出现紧急事务，CPU不得不停下当前正在执行的程序，转去处理紧急事务，事务处理完后，继续执行被中断的程序
  - 作用：主机与外设并行、实时处理和过程控制、硬件故障处理、多道程序和分时操作
  - 一般情况下，中断是随机的
  - 主程序：被中断的程序
  - 中断服务子程序：处理中断事务的程序
  - 中断向量：中断服务子程序的入口地址
  - 中断向量表：保存所有中断向量的内存区域，一般固定。



DMA方式

- 程序I/O与中断I/O的不足
  - I/O传送速度受处理器测试和给设备提供服务的速度的限制
  - 处理器直接负责管理I/O，对于每一次I/O传送，处理器必须执行一些指令
- DMA（Direct Memory Access）
  - CPU对总线的控制被临时禁止。DMA控制器接管总线控制权，控制数据直接在存储器与外设之间高速交换
  - CPU不再介入具体的I/O操作，由DMA控制器来负责提供存储器地址信号、读写控制信号等。
  - CPU与I/O设备在更大的程度上并行工作，效率更高。
  - DMA方式适合高速批量的数据传输，如视频显示刷新、磁盘存储系统的读写、存储器到存储器的传输等。