计算机组成原理

合肥工业大学
计算机与信息学院

陈 田

提　纲

1. 考试形式和试卷结构
2. 考查目标
3. 参考书目
4. 考点及重点难点分析

大纲要求

- 计算机组成原理考查目标
  - 理解单处理器计算机系统中各部件的内部工作原理、组成结构以及相互连接方式，具有完整的计算机系统的整机概念。
  - 理解计算机系统层次化结构概念，熟悉硬件与软件之间的界面，掌握指令集体系结构的基本知识和基本实现方法。
  - 能够运用计算机组成的基本原理和基本方法，对有关计算机硬件系统中的理论和实际问题进行计算、分析，并能对一些基本部件进行简单设计，并能对高级程序设计语言（如C语言）中的相关问题进行解析。

第一、二章　计算机系统概述

- 计算机发展历程
- 计算机系统的层次结构
  - 1.计算机系统的基本组成
  - 2.计算机硬件的基本组成
  - 3.计算机软件和硬件的关系
  - 4.计算机系统的工作过程
  - "存储程序"工作方式，*高级语言程序与机器语言程序之间的转换*，程序和指令的执行过程
- 计算机性能指标

  吞吐量；响应时间；CPU时钟周期、主频、CPI、CPU执行时间；MIPS、MFLOPS 、GFLOPS、TFLOPS、PFLOPS EFLOPS、ZFLOPS。

重点和难点

- 重点：计算机组成的概貌和框图；衡量计算机的性能指标；主机完成一条指令的信息流程

- 难点思考：如何区分存放在存储器中的指令和数据

知识点回顾

1. 怎样衡量CPU性能?

- 一般把程序的响应时间划分为CPU时间和等待时间，CPU时间又分为用户CPU时间和系统CPU时间
- 因为操作系统对自己所花费的时间进行测量时，不十分准确，所以对CPU性能的测量一般通过程序运行的用户CPU时间进行

知识点回顾

2. 什么是CPI？响应时间（执行时间）与CPI的关系是什么？

- CPI是每条指令执行所用的时钟周期数。
- 通常，一条特定指令的CPI是一个确定的值，而某个程序的CPI则是一个平均值。
- 一个程序的执行时间取决于该程序所包含的指令数、CPI和时钟周期。
- 当指令条数和时钟周期一定时，CPI越大，执行时间越长。

CPU 执行时间 = CPU时钟周期数 / 程序 X 时钟周期

= CPU时钟周期数 / 程序 ÷ 时钟频率

= 指令条数 / 程序 X CPI X 时钟周期

知识点回顾

3. 计算机的MIPS数越大，说明性能越好，对吗？

答：不对。MIPS数反映的是机器执行定点指令的速度。但是，不同机器的指令集不同，指令的功能也不同，也许一个机器上一条指令的功能，在另外一个机器上要用多条指令完成，这样，同样的指令条数所完成的功能可能完全不同。因此用MIPS书对不同机器进行性能比较是不太客观的。

4. 将一个程序在一台计算机上编译，如果生成的指令条数少，代码执行时间是否就短?

例：假设计算机M的指令集中包含A、B、C三类指令，其中CPI分别为1、2、4。某个程序P在M上被编译成两个不同的目标代码序列P1和P2，P1所含A、B 、C三类指令的条数分别为8、2、2，P2所含A、B、C三类指令的条数分别为2、5、3。请问哪个代码序列指令条数少？哪个执行速度快？它们的CPI分别是多少？

解：P1和P2指令执行条数分别为12和10，所以P2指令条数少。

P1时钟周期数为8×1+2 ×2+2 ×4=20

P2时钟周期数为2×1+5 ×2+3 ×4=24

两个程序在同一台机器上运行，所以时钟周期一样，故时钟数少的代码序列所用的时间短、执行速度快。P1比P2快。由上可知，指令条数少的代码序列执行时间并不一定短。

CPI=程序总时钟周期数÷程序所含指令条数

P1的CPI=20/12=1.67

P2的CPI为24/10=2.4

第三章 总线

（一） 总线概述

1. 总线的基本概念

2.总线的组成及性能指标

（二） 总线仲裁

1. 集中仲裁方式

2. 分布仲裁方式

（三） 总线事务和定时

1. 同步定时方式

2. 异步定时方式

重点和难点

重点：总线的功能与结构；总线的性能指标；总线的判优控制和通信控制；如何克服总线的瓶颈

难点：总线的通信控制

设一个32位微处理器配有16位外部数据总线，时钟频率为50MHz，若总线传输的最短周期为4个时钟周期，试问处理器的最大数据传输率是多少？若想提高一倍数据传输率，可采用什么措施?

解：根据题意，总线最短传输周期为：

$T=4/(50MHz)=4\times20\times10^{-9}=80\times10^{-9}$ s

对于外部总线为16位的处理器，最大数据传输率为

$2B/T=25\times10^{6}$Bps

如果想提高一倍数据传输率，可采用的措施

(1) 外部数据总线宽度改为32位，CPU时钟频率仍为50MHz,则数据传输率为$4B/T=50\times10^{6}$ Bps

(2) 时钟频率加倍至100MHz，外部数据总线宽度仍为16位，则数据总线的传输周期为$T'=4/(100MHz)=40\times10^{-9}$ s

数据传输率为$2B/T'=50\times10^{6}$ Bps

第六章　数据的表示和运算

(一)数制与编码

1. 进位计算制及其数据之间的相互转换
2. 定点数的编码表示

(二)运算方法和运算电路

1. 基本运算部件

   加法器，算术逻辑部件（ALU）

2. 加/减法运算

   补码加减法运算器，

   标志位的生成

3. 乘除法运算的基本原理，

   乘法电路和除法电路的基本结构

(三)整数的表示和运算

1. 无符号整数的表示和运算
2. 带符号整数的表示和运算

(四) 浮点数的表示和运算

1.浮点数的表示

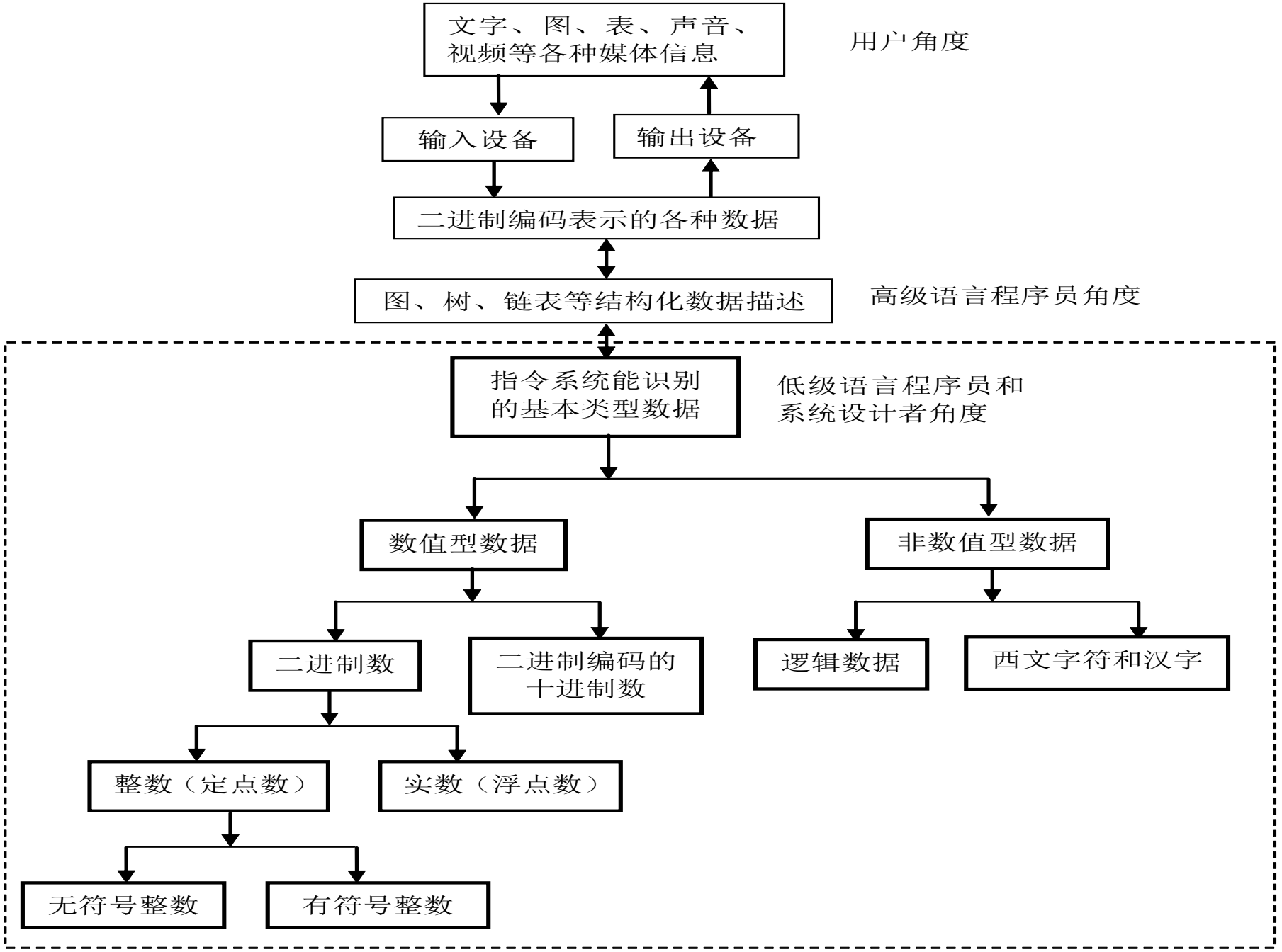
IEEE754标准

2. 浮点数的加/减运算

重点和难点

重点：有符号数、无符号数、定点数和浮点数的表示；移位运算、定点补码加、减、乘、除运算；IEEE754浮点数加减运算；提高运算速度的措施。

难点：机器字长相同的条件下，补码比原码和反码能多表示一个负数；区分浮点数和补码表示的浮点规格化数；在定点机和浮点机中，如何判断运算结果溢出；不同的机器数运算规则不同，直接影响运算器的硬件组成。


文字、图、表、声音、
视频等各种媒体信息
用户角度
输入设备
输出设备
二进制编码表示的各种数据
图、树、链表等结构化数据描述
高级语言程序员角度
指令系统能识别
的基本类型数据
低级语言程序员和
系统设计者角度
数值型数据
非数值型数据
二进制数
二进制编码的
十进制数
逻辑数据
西文字符和汉字
整数（定点数）
实数（浮点数）
无符号整数
有符号整数

知识点回顾

- 数据的表示：数值数据、非数值数据
  - 数值数据：在数轴上有对应点，二进制、十进制
    - 二进制表示的数：直接用二进制
      无符号数：正整数，表示地址等
      有符号数：常用补码
      浮点数：表示实数，多用IEEE754标准表示
    - 用十进制表示的数：用二进制进行编码，称BCD码，常用8421码表示
  - 非数值数据：在数轴上无对应点
    - 逻辑值：只有两个状态取值，按位运算
    - 西文字符：7位ASCII码表示
    - 汉字字符：输入码、内码、字模码
- 数据的宽度：以字节为基本单位表示
- 数据的排列：大端方式（较低的有效字节存放在较高的存储器地址）、小端方式（较低有效字节存放在较低的存储器地址）

知识点回顾

- 算术逻辑单元ALU：在超前进位加法器的基础上增加其它逻辑，实现基本的算术和逻辑运算部件。有两个操作数输入、低位进位输入、一个操作控制输入、一个结果输出、一位高位进位输出和相等标志输出等。
- 定点数运算：由专门的定点运算器实现，核心部件是带快速加法器的ALU
- 移位运算
  - 逻辑移位：对无符号数移位，补0，低（高）位移出
  - 算术移位：带符号整数移位，符号位不变，否则溢出
  - 循环移位：最左（右）边位移到最低（高）位，其它位左（右）移一位。
- 加减运算：
  - 补码加减：用于整数加减运算。符号位和数值位一起运算，减法用加法实现。同号相加时若结果的符号不同于加数符号，则溢出。
  - 原码加减：

知识点回顾

- 加减运算：
  - 补码加减
  - 加减法器：在基本加法器基础上增加进位/加减控制、求补电路、溢出判断电路等
- 乘法运算（用加法和右移实现）
  - 原码乘法：符号位和数值位分开运算,数值部分用无符号数乘法实现
  - 补码乘法：用于整数乘法运算,符号位和数值位一起运算,采用Booth算法
  - 快速乘法器

知识点回顾

- 除法运算（用加减法和左移实现）
  - 补码除法：符号位和数值位一起参加运算，有恢复余数法和不恢复余数法两种
  - 原码除法：符号和数值分开运算，用无符号数除法实现，有恢复余数法和不恢复余数法两种
- 浮点数运算（由专门的浮点运算器实现）
  - 加减运算：
    - 对阶：小阶向大阶看齐，阶小的尾数右移，右移时保留附加位
    - 尾数相加减：定点数加减运算，隐藏位和附加位一起参加运算。
    - 规格化处理：左规、右规
    - 舍入
    - 溢出判断

知识点回顾

求特殊数的补码（假定机器数有n位）

① $[-2^{n-1}]_{补} = 2^n - 2^{n-1} = 10\ldots0$（n-1个0） （mod 2^n）

② $[-1]_{补} = 2^n - 0\ldots01 = 11\ldots1$（n个1） （mod 2^n） 整数补码

③ $[-1.0]_{补} = 2 - 1.0 = 1.00\ldots0$（n-1个0） （mod 2） 小数补码

④ $[+0]_{补} = [-0]_{补} = 00\ldots0$（n个0）

设补码的位数为n，求“-1”的补码

解：对于小数补码$[-1]_{补} = 2 - 1 = 1.0000$

对于整数补码$[-1]_{补} = 2^n - 1 = 11\ldots1$

知识点回顾　IEEE 754 浮点数标准

Single Precision： **(Double Precision is similar)**

1 bit S	8 bits Exponent	23 bits Significand

- **Sign bit: 1 表示negative ; 0表示 positive**
- **Exponent（阶码 / 指数）：**

  全0和全1用来表示特殊值！

  - **SP规格化数阶码范围为0000 0001 (-126) ~ 1111 1110 (127)**
  - **bias为127 (single), 1023 (double)**

- **Significand（尾数）：**
  - **规格化尾数最高位总是1，所以隐含表示，省1位**
  - **1 + 23 bits （ single），1 + 52 bits （double）**

SP: $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$

DP: $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-1023)}$

0的表示

How to represent 0?

exponent: all zeros

significand: all zeros

What about sign? Both cases valid.

+0: 0 00000000 00000000000000000000000

-0: 1 00000000 00000000000000000000000

如何表示+∞/-∞

In FP, 除数为0的结果是 +/- ∞, 不是溢出异常. ∞：无穷(infinity)

为什么要这样处理?

- **可以利用+∞/-∞作比较。 例如：X/0>Y可作为有效比较**

How to represent +∞/-∞?

- **Exponent : all ones (11111111B = 255)**
- **Significand: all zeros**

+∞ : 0 11111111 00000000000000000000000

-∞ : 1 11111111 00000000000000000000000

Operations

5 / 0 = +∞, -5 / 0 = -∞

5+(+∞) = +∞, (+∞)+(+∞) = +∞

5 - (+∞) = -∞, (-∞) - (+∞) = -∞ etc

IEEE754标准规定的五种异常情况

① 无效运算（无意义）

- 运算时有一个数是非有限数，如：

  加 / 减∞、0 x ∞、 ∞/∞等

- 结果无效，如：

  源操作数是NaN、0/0、x REM 0、 ∞ REM y 等

② 除以0（即：无穷大）

③ 数太大（阶码上溢）: 对于SP，指阶码 E >1111 1110 （指数大于127）

④ 数太小（阶码下溢）: 对于SP，指阶码 E < 0000 0001（指数小于-126 ）

⑤ 结果不精确（舍入时引起），例如1/3，1/10等不能精确表示成浮点数

上述情况硬件可以捕捉到，因此这些异常可设定让硬件处理，也可设定让软件处理。让硬件处理时，称为硬件陷阱。

注：硬件陷阱：事先设定好是否要进行硬件处理（即挖一个陷阱），当出现相应异常时，就由硬件自动进行相应的异常处理（掉入陷阱）。

规格化数

规格化数的定义：　　$r=2$　　$\frac{1}{2} \leq |S| < 1$

$$S = -\frac{1}{2} = -0.100\cdots0$$

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = 1.100\cdots0$$

∴ $[-\frac{1}{2}]_{补}$ 不是规格化的数

$$S = -1$$

机器判别方便

$$[S]_{补} = 1.000\cdots0$$

∴ $[-1]_{补}$ 是规格化的数

设机器数字长为 8 位（其中一位为符号位）对于整数，当其分别代表无符号数、原码、补码和反码时，对应的真值范围各为多少？

二进制代码	无符号数对应的真值	原码对应的真值	补码对应的真值	反码对应的真值
00000000	0	+0	±0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111111	127	+127	+127	+127
10000000	128	-0	-128	-127
10000001	129	-1	-127	-126
⋮	⋮	⋮	⋮	⋮
11111101	253	-125	-3	-2
11111110	254	-126	-2	-1
11111111	255	-127	-1	-0

- 与原码和反码相比，补码多表示一个最小负数：-2^{n-1}，即“10…0”

例：无符号数在C语言中对应unsigned short、unsigned int(unsigned)、unsigned long，带符号整数表示为short、int、long类型，求以下程序段在一个32位机器上运行时输出的结果，并说明为什么。

```
1 int x=-1;
2 unsigned u=2 147 483 648;
3 printf (“x=%u=%d\n”, x , x);
4 printf (“u=%u=%d\n”, u , u);
```

说明：

•其中**printf**为输出函数，指示符**%u**、**%d**分别表示以无符号整数和 有符号整数形式输出十进制数的值，

• 2 147 483 648=2^{31}

解：因为现代计算机中带符号整数都是用补码表示的， **-1**的补码整数表示为**“11…1”**（共**32**位），当作**32**位无符号数时，因此，十进制表示为 **2^{32}-1=4 294 967 296-1= 4 294 967 295**

2^{31}的无符号整数在计算机中表示为**“10…0”**，当作为有符号整数输出时，其值为最小负数**-2^{32-1}=-** 2 147 483 648

输出结果：

x= 4 294 967 295=-1

u=2 147 483 648=- 2 147 483 648

例：以下为C语言程序，用来计算一个数组a中每个元素的和，当参数len为0时，返回值应该为0，却发生了存储器访问异常。请问这是什么原因造成的？说明如何修改。

```
1   float  sum_elements  ( float a[ ], unsigned len)
2   {
3          int     i;
4          float  result=0;
5           for ( i=0; i<=len-1; i++ )
6                  result+=a[ i ] ;
7            return result;
8   }
```

解： 存储器访问异常是由于对数组**a**访问时产生了越界错误造成的。循环变量**i**是**int**型，而**len**是unsigned型，当**len**为**0**时，执行**len-1**的结果为**32**个**1**，是最大可表示的**32**位无符号数，任何无符号数都比它小，使循环体不断被执行，导致数组访问越界，因而发生存储器访问异常，应当将**len**声明为**int**型。

第四章　存储器层次结构

三. 存储器层次结构

(一)存储器的分类

(二)层次化存储器的基本结构

(三)半导体随机存取存储器

1. SRAM存储器

2.DRAM存储器

3. Flash存储器

(四)主存储器

1. DRAM芯片和内存条

2. 多模块存储器

3.主存储器与CPU的连接

(五）外部存储器

1. 磁盘存储器

2. 固态硬盘（SSD）

(六）高速缓冲存储器(Cache)

1. Cache的基本工作原理

2. Cache和主存之间的映射方式

4. Cache中主存块的替换算法

5.Cache写策略

(七） 虚拟存储器

1. 虚拟存储器的基本概念

2. 页式虚拟存储器

基本原理，页表，地址转换，TLB（快表）

3. 段式虚拟存储器

4. 段页式虚拟存储器

重点和难点

重点：存储系统的层次结构；主存、Cache 的工作原理及技术指标；半导体体存储芯片的外特性及与 CPU的连接；提高访存速度的措施。

难点：对于一定容量的存储器，按字节或字访问的寻址范围；多体并行结构存储器顺序编址和交叉编址对访存速度的影响；Cache —主存地址映射。

知识点回顾

- 存储器分类
  - 按存取方式：随机、顺序、直接、相联
  - 按存储介质：半导体、磁表面、激光盘
  - 按信息是否可更改：可读可写、只读
  - 按断电后可否保存：易失、非易失
  - 按功能/容量/速度分：寄存器、cache、主存、辅存
- 存储器层次结构
- 半导体随机存取存储器的组织

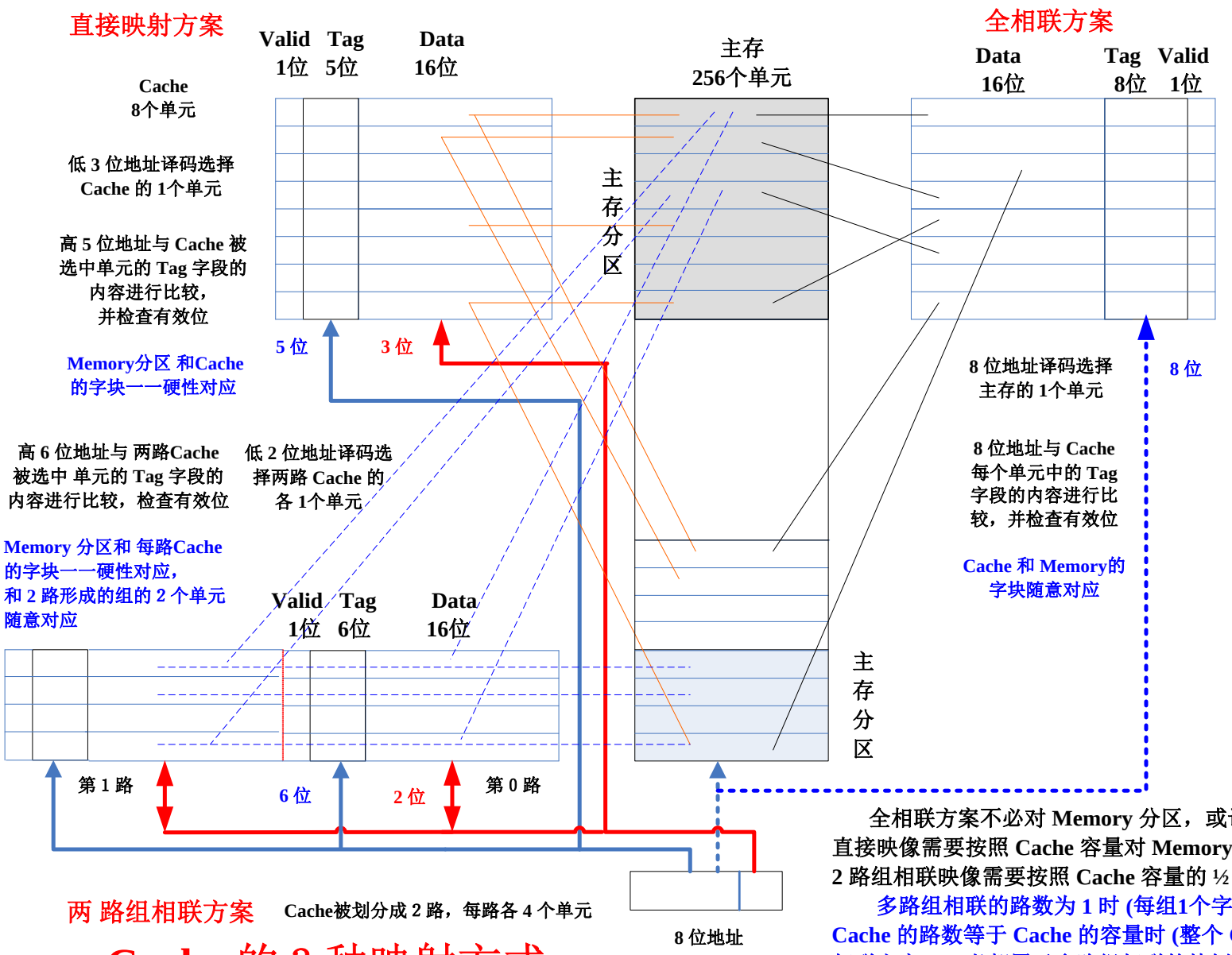
  存储元→存储芯片→存储模块（内存条）→存储器
- 只读存储器
- 存储器芯片与CPU连接：地址线、数据线、控制线连接
- 主存的主要技术指标：存取时间、存储周期、存储器带宽

知识点回顾

- 多模块存储器：连续编址、交叉编址
- Cache：
  - 使用cache的基本原理：程序访问的局部性
    - 时间局部性
    - 空间局部性
  - Cache基本工作原理
    - Cache和主存空间被划分为相等的区域。主存中的区域称为块(block)，是cache与主存间信息交换的单位；cache中存放一个主存块的区域称为行（line）或槽（slot）
    - Cache的有效位：系统启动或复位时，每个cache行都是空的，其中的信息无效，只有装入主存块后信息才有效，因此每个cache行需要一个有效位（valid bit）
    - Cache的容量
    - Cache命中率、cache-主存层次平均访问时间
  - Cache与主存间的映射：直接映射、全相联映射、组相联映射

知识点回顾

- Cache：
  - 替换算法：FIFO、LRU、LFU（最不经常使用）、随机法
  - 写策略：
    - 写回法：暂时只写cache，替换时一次性写回主存
    - 全写法：每次写cache时也写主存，可在cache和主存间加写缓存
- 虚拟存储器（操作系统）
  - 基本原理：
  - 虚拟存储器的实现方案：分页式、分段式、段页式
  - 地址转换
  - 页表和页表项
  - 缺页
  - TLB(快表)：用来存放常用页表项，减少主存访问页表次数



Cache 的 3 种映射方式

把Cache与主存的三种地址映像方式统一画在一张图上，有利于对比它们的相同与差异之处。

为突出基本原理并容易理解，画面清楚，只画出主存分区的首个分区和尾个分区，中间部分从略。并假定主存由 256 个字块组成，因此要用 8 位地址访问主存字块，假定主存字长16位。

Cache容量为 8 个字块，因此要使用 3 位（直接映射） 或 2位（两路组相联）地址访问 Cache 字块，这样 Cache 的 Tag 字段的位数应为 5 或 6 。使用一位有效位， 数据段 是 16 位。

全相联方案不必对 Memory 分区，或认为每个字块是一个区；
直接映像需要按照 Cache 容量对 Memory分区；
2 路组相联映像需要按照 Cache 容量的 ½ 对 Memory分区。

多路组相联的路数为 1 时 (每组1个字块) 就是直接映像方案; Cache 的路数等于 Cache 的容量时 (整个 Cache 仅有 1 组) 就是全相联方案；二者都属于多路组相联的特例。 否则是 n 路组相联，每组由 n (路数) 个字块构成， Cache 的组数应为 Cache容量 / n。

例题：有三个处理器，带有以下不同cache:

cache1：采用直接映像方式，块大小为1个字，指令和数据的缺失率分别为4%和6%；

cache2：采用直接映像方式，块大小为4个字，指令和数据的缺失率分别为2%和4%；

cache3：采用2路组相联映像方式，块大小为4个字，指令和数据的缺失率分别为2%和3%；

在些处理器上运行相同的程序，该程序的CPI为2.0，其中有一半是访存指令，若缺失损失为“块大小+6”个时钟周期，处理器1和处理器2的时钟周期都为420ps，带有cache3的处理器3的时钟周期为450ps。问：哪个处理器因cache缺失而引起的额外开销最大？哪个处理器执行速度最快？

解：假设所运行的程序共执行N条指令，每条访存指令仅读写一次内存数据，则在该程序执行过程中各个处理器因cache缺失而引起的额外开销和执行时间计算如下：

处理器1：额外开销为N×（4%+6%×50%）×（1+6）=0.49N个时钟周期，执行程序所需时间为（N×2.0+0.49N）×420ps=1045.8N（ps）；

处理器2：额外开销为N×（2%+4%×50%）×（4+6）=0.40N个时钟周期，执行程序所需时间为（N×2.0+0.40N）×420ps=1008N（ps）；

处理器3：额外开销为N×（2%+3%×50%）×（4+6）=0.35N个时钟周期，执行程序所需时间为（N×2.0+0.35N）×450ps=1057.5N（ps）.

由此可见，处理器1cache缺失引起的额外开销最大，处理器2执行速度最快。

第五章　输入输出系统

(一)　I/O系统基本概念

(二)　I/O接口（I/O控制器）

1.　I/O接口的功能和基本结构

2.　I/O端口及其编址

(四)　I/O方式

1.　程序查询方式

2.　程序中断方式

中断的基本概念；中断响应过程；中断处理过程

3.　DMA方式

DMA控制器的组成；DMA传送过程

重点和难点

重点：主机与 I/O 交换信息的三种控制方式；程序查询方式、程序中断方式和 DMA 方式的各自特点、接口电路和工作原理、中断系统需要解决的问题及实施方案。

难点：处理 I/O 中断的各类软、硬件技术的运用；DMA与主存交换数据的三种方法各自的特点；周期窃取的含义； CPU 响应中断请求和 DMA 请求的时间。

第7章　指令系统

（一）　指令系统的基本概念

（二）　指令格式

（三）　寻址方式

（四）数据的对齐和大/小端存放方式（在第四章）

（五）　CISC和RISC的基本概念

（六）高级语言程序与机器级代码之间的对应

1. 编译器，汇编器和链路器的基本概念

2.选择结构语句的机器级表示

3.循环结构语句的机器级表示

4.过程（函数）调用对应的机器级表示

重点和难点

重点：指令格式、寻址方式

地址格式对访存次数、寻址范围的影响

寻址方式对操作数寻址范围、信息加工流程、所需硬件支持及编程的影响

RISC

难点：设计指令格式

扩展操作码技术的运用

在可按字节和字寻址的存储器中不同的机器其数据的存放方式是不同的

例：设相对寻址的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，用补码表示。每当CPU从存储器取出一个字节时，即自动完成(PC)+1→PC。

（1）设当前PC值为3000H，问转移后的目标地址范围是多少？

解：（1）由于相对寻址的转移指令为两个字节，第一个字节为操作码，第二个字节为相对位移量，且用补码表示，故其范围为-128～+127，即80H～7FH。又因PC当前值为3000H,且CPU取出该指令后，PC已修改为3002H，因此最终的转移目标地址范围为3081H～2F82H，即3002H + 7FH =3081H至3002H-80H=2F82H

思考：若PC为16位，位移量可正可负，PC相对寻址范围为多大？

解：相对寻址中，PC提供基准地址，位移量提供修改量，位移量为16位可正可负，则相对寻址范围为：（PC）-2^{15}～（PC）$+2^{15}-1$

例：设相对寻址的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，用补码表示。每当CPU从存储器取出一个字节时，即自动完成(PC)+1→PC。

（2）若当前PC值为2000H,要求转移到01BH，则转移指令第二字节的内容是什么?

解：（2）若PC当前值为2000H，取出该指令后PC值为2002H，故转移指令第二字节应为

201BH－002H=19H。

例：设相对寻址的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，用补码表示。每当CPU从存储器取出一个字节时，即自动完成(PC)+1→PC。

（3）若当前PC值为2000H，指令JMP * -9（*为相对寻址特征）的第二字节的内容是什么？

解：根据汇编语言指令JMP* - 9，即要求转移后的目标地址为2000H - 09H=1FF7H，但因为CPU取出该指令后PC值已修改为2002H，故转移指令的第二字节的内容应为-11（十进制），写成补码为F5H。

第八、九、十章　中央处理器

(一)　CPU的功能和基本结构

(二)　指令执行过程

(三)　数据通路的功能和基本结构

(四)　控制器的功能和工作原理

1. 硬布线控制器

2. 微程序控制器

微程序、微指令和微命令；微指令格式；微命令的编码方式；微地址的形成方式

重点和难点

- CPU的功能和结构

重点：CPU 的功能和硬件组成； CPU 工作周期和指令周期的概念；一个完整的指令周期中的信息流程。

难点：各种中断技术和实现多重中断所需配置的硬件；响应优先级和处理优先级的区别。

重点和难点

- 控制单元的功能

重点：控制单元对不同的指令在取指、间址和中断周期中，发出哪些相同的操作命令；多级时序系统；控制单元的控制方式（同步、异步、联合）。

难点：指令周期、机器周期、时钟周期与控制信号的关系；中央控制和局部控制相结合的同步控制方式；不同结构的计算机（总线结构和非总线结构） 控制信号的特点。

重点和难点

- 控制单元的设计

重点：微操作命令的节拍安排；两种控制器的设计思想、设计步骤、组成原理；两种控制单元微操作命令节拍安排的区别。

难点：微指令的控制方式（编码方式）及后续微指令地址的形成方式；编写微指令的码点。

知识点回顾

- CPU的基本功能
  - 控制程序的执行顺序
  - 控制指令进行什么操作
  - 控制每个操作什么时候进行
  - 对数据进行算术或逻辑运算
  - 控制对存储器或I/O的访问
  - 判断有无异常或中断并调出相应处理程序
- CPU基本结构：由数据通路和控制单元组成
  - 数据通路指执行过程中数据所经过的路径，其中包含组合逻辑部件和时序逻辑部件
  - 控制单元对取出的指令进行译码，与指令执行得到的条件码或当前机器状态、时序信号等组合，生成对数据通路进行控制的控制信号

知识点回顾

- CPU中的寄存器：用户可见、用户部分可见、用户不可见寄存器
- 指令执行过程：取指、译码、取数、运算、存结果、查中断
  - 指令周期
  - 机器周期
  - 时钟周期
- 数据通路中信息的流动过程
- 控制单元的实现方式
  - 硬连线控制器
  - 微程序控制器
  - 基本概念：微程序、微指令和微命令；微指令格式；微命令的编码方式；微地址的形成方式

第十一章　指令流水线

1.指令流水线的基本概念

2. 指令流水线的基本实现

3. 结构冒险、数据冒险和控制冒险的处理

4. 超标量和动态流水线的基本概念

多处理器基本概念

（自学）

1.SISD、SIMD、MIMD、向量处理器的基本概念

2. 硬件多线程的基本概念

3.多核处理器（multi-core）的基本概念

4. 共享内存多处理器（SMP）的基本概念

重点和难点

重点：指令流水线。

难点：影响指令流水线性能的因素；超标量、超流水、超长指令字的特点；指令流水线的实现。

知识点回顾

- 指令流水线的基本概念
  - 将每条指令的执行可分为若干个流水阶段。
  - 每个流水阶段的执行时间以最慢的流水段所需时间为准。
  - 理想情况下，每个时钟周期有一条指令进入流水线，并有一条指令执行结束。
  - 每个流水段中的部件都是组合逻辑部件，流水线段之间需要加流水段寄存器，组合逻辑中产生的结果在时钟到来时被存储到流水段寄存器中。流水段寄存器用以记录所有流到后面阶段要用的各种信息，例如控制信号、指令、新的PC值、参加运算的操作数、指令运算结果、指令异常信息、寄存器读口地址、寄存器写口地址、存储地址等。
  - 指令译码得到的控制信号通过流水段寄存器与本指令的数据信息一起，同步传送到后面各个流水段。

知识点回顾

- 指令流水线的局限性
  - 不同的指令功能不同，并不是每条指令都能划分成相同多个阶段，按最复杂指令所需规划流水段后，有些指令的某些流水段执行的可能是空操作。
  - 不同流水阶段的功能不同，并不是每个流水段所用的时间都一样长，按最长时间流水段设置时钟周期后，某些流水段可能会有时间浪费。
  - 随着流水段深度的增加，流水段寄存器的额外开销比例也增大。
  - 指令在资源冲突、数据相关或控制相关事发生流水线阻塞，因而影响指令执行效率。

知识点回顾

- 指令流水线的执行效率
  - 吞吐率：比非流水线方式下提高若干倍，理想情况下，其倍数为划分的流水段个数。
  - 指令执行时间：由于流水段划分要求的一致性，以及流水段寄存器的额外开销，使得流水段方式下一条指令的执行时间更长了。

知识点回顾

- 流水线冒险的种类及其处理基本思想
- 结构冒险（资源冲突）：多条指令同时要求使用同一个功能部件。所用解决策略如下：
  - 规定每个功能部件在一条指令中只能被使用一次。
  - 规定每个功能部件只能在某个特定的阶段被使用。
  - 指令存储器（code cache）和数据存储器（data cache）分开。
- 数据冒险（数据相关）：前面指令的目的操作数是后面指令的源操作数。解决策略如下：
  - 用软件（如编译器）在数据相关指令前插入nop指令。
  - 在硬件检测到数据相关时，使后面的数据相关指令进入停顿状态，业绩在特定的流水端插入“气泡”以“阻塞”指令继续执行，直到取得所需数据为止。
  - 利用“转发（旁路）”技术把前面指令执行过程中得到的数据直接传送到后面指令需要使用数据的地方。
  - 对于取数后直接使用的情况（如Load指令取出的数据是随后下一条运算指令的操作数），则采用“阻塞加转发”的方式解决数据冒险。

知识点回顾

- 控制冒险（控制相关）：有两种情况会发生控制冒险
  - 1.转移指令引起。分支、调用、返回、跳转等指令（统称为转移指令）需要计算目标转移地址，分支指令还需要根据条件确定新的PC值。由于获取转移目标地址的时间较长，使得在目标地址产生前，后续指令已被取到流水线中。如果已经取出并正在执行的指令不是应该执行的指令，则发生了控制冒险。分支指令引起的控制冒险称为分支冒险。
  - 第二种是异常和中断引起的。当处理器检测到发生异常和中断时，可能有多条不该执行的指令已在流水线中执行，因而会导致控制冒险。

知识点回顾

- 控制冒险（控制相关）：

  解决策略如下：

  - 由编译器在转移指令后插入nop指令(插入指令条数等于延迟损失时间片)。
  - 在硬件检测到有关转移指令时，使后面若干条指令（插入指令条数等于延迟损失时间片）进入停顿状态，也即在特定的流水段中插入“气泡”以“阻塞”指令继续执行，直到能确定正确的PC值为止。
  - 对于分支冒险，可结合采用“分支预测”技术。有简单（静态）预测和动态预测两种方式。静态预测状态下，预测的结果是固定的，总是预测转移（taken）或不转移（not taken）；动态预测方式下，预测位根据分支指令执行的历史情况进行调整。动态预测比静态预测成功率高得多。
  - 采用延迟分支技术。将前面几条与转移指令无关的指令放到分支指令后面的延迟槽中执行，无关指令不够时用nop指令填满延迟槽。这样，硬件不需对流水线进行阻塞，对指令顺序进行调整的工作在编译阶段完成。

知识点回顾

- 高级流水线技术。在时间和空间上进一步提高指令级并行性
  - 超流水线：将指令执行过程化分得更细，采用更多级数的流水线，着重在于时间上并行。
  - 多发射流水线（超标量处理器）：同时发射多条指令，并有多个功能部件并行执行，着重在于空间上并行。“指令打包”和“冒险处理”是实现多发射的两个基本任务。
    - 静态多发射：“指令打包”和“冒险处理”任务主要由编译器静态完成，打包后的指令相当于一条由多条指令组成的长指令，被同时发射执行，因此，这类处理器有时被称为VLIW处理器。Intel的IA-64架构采用这种方法，Intel称其为EPIC技术。
    - 动态多发射：由处理器硬件在指令执行时动态确定那些指令被同时发射，如果没有可以被同时发射的指令或遇到指令相关时，则某些流水段空闲。目前超标量处理器多指采用动态发射流水线的处理器，并且通常会结合采用动态流水线调度技术。
  - 动态流水线调度：处理器通过指令相关性检测和动态分支预测等手段，投机性的不按指令顺序执行。即当发生流水线阻塞时，根据指令的依赖关系，动态地到后面找一些没有依赖关系的指令提前执行。在动态流水线调度下，指令被“乱序”执行。

设某机平均执行一条指令需要两次访问内存，平均需要3个CPU周期，每个CPU周期平均包含4个节拍周期。若机器主频为240MHz，问：

（1）若主存为“0等待”（即不需要插入等待周期），问执行一条指令的平均时间为多少？

（2）若每次访问内存需要插入2个等待周期，问执行一条指令的平均时间又是多少？

解：因为主频为240MHz，所以节拍周期=（1/240）μs每个

因为每个CPU周期平均包含4个节拍周期，所以：

CPU周期=节拍周期×4 = 4/240MHz=(1/60)μs

若访存不需要插入等待周期，则执行一条指令平均需要3个CPU周期，所以：

指令周期 = 3×CPU周期 = 3×（1/60）μs=(1/20)μs=0.05μs

机器平均速度 = 1/0.05μs=20 MIPS

（2）平均执行一条指令需要两次访问内存，每次访问内存需要插入2个等待周期，所以：

指令周期 = 0.05μs + 2×(1/240)μs = (1/20)μs +(1/120)μs

=(7/120)μs

机器平均速度 = 120/7≈17MIPS

$$\frac{MIPS_1}{MIPS_2} = \frac{f_1}{f_2}$$

若某机主频为200MHZ，每个指令周期平均为2.5CPU周期，每个CPU周期平均包括2个主频周期，问：
(1)该机平均指令执行速度为多少MIPS?
(2)若主频不变，但每条指令平均包括5个CPU周期，每个CPU周期又包含4个主频周期，平均指令执行速度又为多少MIPS?
由此可得出什么结论?

解：（1）主频为为200MHz，所以主频周期 = 1/200MHz=0.005μs

每个指令周期平均为2.5CPU周期，每个CPU周期平均包括2个主频周期，所以一条指令的执行时间 = 2.5×2×0.005μs = 0.025μs

该机平均指令执行速度 = 1/0.025 = 40MIPS。

（2）每条指令平均包括5个CPU周期，每个CPU周期又包含4个主频周期，所以一条指令的执行时间 = 4×5×0.005μs = 0.1μs

该机平均指令执行速度 = 1/0.1 = 10MIPS

(3)说明指令的复杂程度会影响指令的平均执行速度。

例：假设数据通路中主要功能部件的操作时间如下：存储器200ps；ALU和加法器：100ps；寄存器堆读口或写口：50ps。程序中指令的组成比例为取数25%、存数10%、ALU52%、分支11%、跳转2%。假设非单周期处理器的时钟周期为存储器存取时间的一半，MUX、控制单元、PC、扩展器和传输线路等的延迟都忽略不计，则下面的实现方式中哪个最快？快多少？

1. 单周期方式：每条指令在一个固定长度的时钟周期内完成。
2. 多周期方式：每类指令的时钟数是为取数：7，存数：6， ALU：5，分支：4，跳转：4
3. 流水线方式：采用取指1、取指2、取数/译码、执行、存取1、存取2、写回七段流水线；没有结构冒险；数据冒险采用转发（旁路）技术处理；load指令与后续各指令之间存在依赖关系的概率分别为1/2、1/4、1/8、……；分支延迟损失时间片为2，分支预测准确率为75%；不考虑异常、中断和访存缺失引起的流水线冒险。

解：(1) 单周期方式下，时钟周期为200+50+100+200+50=600ps，所以，一条指令执行时间为600ps

(2)多周期方式下，CPI=0.25×7+0.10×6+0.52×5+0.11×4+0.02×4=5.47，因为存储器操作变为在两个时钟周期内完成，所以多周期数据通路的时钟周期为100ps，故平均一条指令执行时间为100×5.47=547ps

(3)流水线方式下，存储器操作变为在两个时钟周期内完成，流水线包含七个分段，对于分支指令，若预测正确，则不需要额外时钟周期，故只需1个时钟周期；若预测错误，则因为分支延迟损失2个时间片，所以应该将错误预取的2条指令冲刷掉，额外多用了2个时钟周期，因此预测错误共需3个时钟周期，CPI=25%×3+75×1=1.5。对于load指令，因为一个存储操作需要2个时钟周期，所以随后的1条指令共需3个时钟周期（其中阻塞2个时钟周期）；随后第二条指令共需2个时钟周期（其中阻塞1个时钟周期），以后的指令不需要阻塞，所以CPI=1/2×3+1/4×2+1/8×1=2.25.对于ALU指令随后的数据相关指令都可以通过转发解决，故CPI=1；对于store指令，不会发生数据冒险，故CPI=1；对于jump指令，最快也要在译码阶段才能确定转移地址，故CPI=3.因此综合CPI=0.25×2.25+0.10×1+0.52×1+0.11×1.5+0.02×3=1.41。所以平均一条指令的执行时间为1.41×100=141ps。

综上，流水线处理器的指令执行速度最快是单周期的600/141=4.26倍；是多周期的3.84倍 。

计算机组成原理课程教学内容的知识结构图

数据表示、运算

数据类型 (整型、浮点型、逻辑型、BCD码、字符型) 的表示，中文的输入码、内码、字形码
进位计数制，二进制与十进制数据之间的转换，二进制数据运算规则
整数、小数的原、反、补码的编码与表示，机器数和数的真值
补码加、减法运算规则与实现算法，原码一位乘、除法的规则与实现算法
3 种常用的检错纠错码的实现原理与应用场所

运算器部件

算术与逻辑运算部件ALU的原理性线路设计，串行进位与并行进位的概念与实现思路
定点运算器的功能与组成（执行运算的部件ALU、暂存数据的寄存器组REGs、特征位寄存器）
补码加减法的线路实现与溢出判别方案、原码一位乘除法的实现原理性线路
浮点数的表示、编码，它的数据范围和精度，IEEE-754标准，浮点数的加、减、乘、除运算规则

控制器部件

控制器的功能与组成（PC、IR、步骤标记与转换线路、产生与提供控制信号的部件）
两种控制器类型：组合逻辑控制器和微程序控制器，同异之处与应用场合
指令的执行过程（步骤），CPU周期，指令周期，指令的串行或并行执行
指令的串行执行过程：单指令周期方案方案，多单指令周期
典型指令的执行过程（步骤），与指令格式和功能、计算机的硬件组成有关
通用原理需用一个简单实际系统来加深理解，CPU的性能计算：主频、CPI、MIPS
RISC 和 CISC 两种计算机追求的基本目标和实现途径（RISC机的特点 ）
指令的并行执行：指令流水线技术，追求的主要目标，实现思路，表示方法
性能指标：吞吐率，加速比，效率
流水线中的结构相关、数据相关、控制相关的产生原因与解决思路

输入设备

输入设备的功能和分类，最常用的输入设备
键盘的组成与运行原理，鼠标设备的功能和工作原理

输出设备

以点阵方式显示或打印字符、图形或图像设备的共性原理
输出设备的功能和分类，最常用的输出设备
CRT和液晶显示器的一般构成和运行原理
针式、喷墨式和激光打印机的一般构成和运行原理

指令、指令系统和汇编语言程序设计

读写外设，外设编址方式：与主存统一编址， IO端口地址
读写内存，寻址方式，CPU与主存交换数据或指令
数据传送，数据运算，数据地址计算，指令地址计算
使用硬件的命令，**指令系统—机器语言**，写程序很难
机器语言符号化并扩展得到**汇编语言**，写程序容易些
针对解题算法而不是硬件本身提供高级语言，最好用

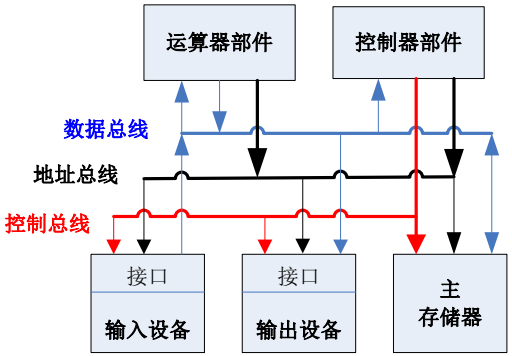
指令的功能与格式：操作码与数据地址字段
寻址方式
形式地址与实际 (物理) 地址的概念及其相互关系
基本寻址方式： 寄存器寻址， 寄存器间接寻址
直接寻址，变址寻址，相对寻址，基地址寻址
堆栈寻址，间接寻址， 立即数寻址
用几种基本寻址方式复合而成的更复杂的寻址方式

接口

接口的功能，经过总线连接外设与 CPU
希望把多种设备和不同的CPU都能连接起来 (个性化处理)
通用可编程接口的一般组成与初始化操作，中断有关线路
接口标准：串行接口，并行接口，USB 接口，...

输入输出方式

程序查询方式，程序中断方式，直接存储器访问方式
通道方式， 外围处理机方式
中断源的分类，中断优先级与中断排队，中断屏蔽
中断请求、响应和处理（保存与恢复中断现场）过程
DMA接口卡的构成，DMA传送的请求，响应与执行过程



存储器系统

存储介质的分类与存储原理，存储器芯片的内部构成
存储器的性能指标：速度、容量、成本
解决问题的途径：构建 3 级结构的存储器系统
程序运行的局部性原理，一致性原则和包含性原则

主存储器：保存CPU正在运行、使用中的程序和数据，通常情况下以字为单位与CPU交换信息，以字块为单位与高速缓存交换信息。
主存与CPU的连接，读写操作过程和访存指令
字位扩展技术，一体多字、多体单字技术

高速缓冲存储器：缓冲最近使用的主存中的程序和数据，由硬件直接控制以字块为单位完成交换。
3 种地址映射方式与地址变换
高速缓存的命中率与加速比，Cache的替换算法

辅助(外)存储器：以文件形式比较长时间的保存计算机系统中的信息，通常以几个扇区为单位执行读写。
磁盘的平均寻址时间、磁盘读写的数据传输率计算
温盘与光盘的基本组成与读写原理，磁盘阵列技术

虚拟存储器：是高速磁盘上的一片存储空间，暂存目前主存中存放不下的的程序和数据，依 据系统运行情况随时与主存交换信息，可能以页或段为单位进行交换，也可以兼用二者。 段表和页表的组成（内容）和地址变换过程，快表的作用。 内存中的换页算法。

总线

总线的功能和基本结构，对总线线路的基本要求
总线类型（数据、地址、控制 3 类总线）及各自的作用
总线的竞争使用：总线仲裁、定时和总线使用
总线周期，总线状态，总线传送方式（正常或成组传送）
总线标准，常用的几种总线概况
总线的性能指标：总线的带宽

计算机的5个功能部件及其连接关系

运算器：暂存运送数据、中间结果，完成数据运算并暂存
控制器：指挥、控制计算机各部件协同运行以便执行程序
主存储器：存储正在使用的程序和数据
输入设备：输入程序和数据， **输出设**备：输出计算结果
总线：连接 5个功能部件，以便以分时方式传接送信息
接口线路：经总线连接CPU和设备，解决二者识别配合

数据表示、运算方法
数据编码与检错纠错
二 <—> 十进制转换
整数的原、反、补码
补码加法与溢出检查
乘除法运算实现原理
浮点数的表示与运算
浮点数的IEEE754标准

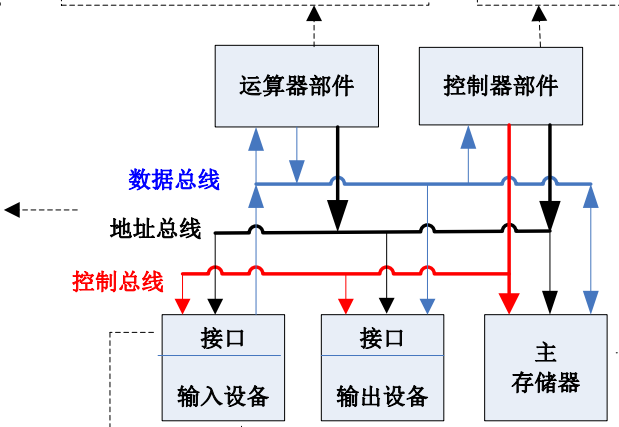
(1)

定点运算器
功能：执行数据运算，暂存运算数据和中间结果 (值和标志),
组成：ALU，执行运算
REGS，暂存数据
Flags, 存标志位信息
浮点运算器概述

硬连线控制器
功能：按指令及执行步骤控制执行程序
组成：PC, 提供指令地址
IR, 保存指令内容
Timing, 给执行步骤信号
CU, 产生各部件的控制信号
微程序控制器，指令流水线

(2)

指令的功能、格式
(操作码字段、操作数地址字段)
形式地址 有效地址
寻址方式
(Reg，Mem，IO)
指令系统实例
RISC、SISC



(3)

传送数据信号
传送地址信号
传送总线周期与入出操作结束时刻等信号
总线性能、类型、结构
总线仲裁、定时
总线标准

层次机构的存储器系统
功能：保存程序和数据的系统，
组成:主存，存正运行的程序数据
高速缓存， 降低平均访问时间
虚存， 作为主存的后援存储区
外存设备，以文件形式保存信息
温盘，光盘，磁带

(4)

连接CPU和输入、输出设备，
机械与电气要匹配、连通，
识别、缓冲、控制、状态 ，中断请求
通用可编程接口

用于输入程序和数据信息，
鼠标
键盘

用于输出计算机内的信息，
显示器
打印机

程序直接控制方式
程序中断方式
通道方式
直接存储器访问方式

1. 计算机相关的寄存器：PC　IR　MAR　MDR　ACC
2. CPI及其相关计算
3. 存储单元 存储字 存储元　存储字长 存储容量
4. 总线分类　系统总线 波特率 比特率
5. 存储器编址范围　大端存储 小端存储 奇偶校验　各类存储器件的存取速度 存储周期
6. 存储器容量 如何选择芯片
7. SRAM DRAM比较　刷新及其分类
8. 多体并行（概念 计算）
9. CACHE　目的　原理　映射方式（直接映射 全相连映射 组相连映射 地址如何变换）写一致性

- 10. 程序查询 程序中断 DMA方式 向量中断 中断隐指令
- 11. 原码 反码 补码 移码
- 13. 浮点数表示范围 规格化 浮点数运算
- 14. 寻址方式（目的 有哪些寻址方式 每种寻址方式如何得到有效地址 适用于什么情况）
- 15. 中断屏蔽字 CPU程序运行轨迹
- 16. 指令周期 取值 间址 执行周期数据通路
- 17.微程序 微指令 机器指令 水平型微指令 垂直型微指令
- 18. 流水线中多发技术 超标量 超流水线 超长指令字
- 19. 流水线中 结构 数据 控制相关

考试题型： 选择题（40分） 填空题（10分） 简答题（10分） 计算题（40分）

注：题型分值可能会调整 但考察的知识点不变