第一章 计算机系统概述

1. 计算机的诞生史

* 世界上第一台真正意义的电子数字计算机：ABC（Atanasoff-Berry Computer，阿塔那索夫-贝瑞计算机），1935~1939年由美国艾奥瓦州立大学物理系副教授约翰·文森特·阿塔那索夫和克里福特·贝瑞研制成功。国际计算机界公认约翰·文森特·阿塔那索夫被称为“电子计算机之父”。
* 世界上第一台真正实用的电子计算机：ENIAC（Electronic Numerical Integrator And Computer，电子数字积分计算机），1946年由美国宾夕法尼亚大学莫齐利、艾克特研制。
* 现代计算机结构思想的诞生：冯·诺依曼于1945年发表“关于EDVAC的报告草案”的全新“存储程序通用电子计算机方案”。该报告中提出的计算机结构被称为冯·诺依曼结构，标志着现代计算机结构思想的诞生。
* 1946年，普林斯顿高等研究院（the Institute for Advance Study at Princeton，IAS ）开始按照冯·诺依曼的设计实现“存储程序”式计算机，被称为IAS计算机。
* 世界上第一台存储程序计算机：1949年由英国剑桥大学完成的EDSAC。

2. 简述“存储程序”工作方式的基本思想。

3. 简述冯·诺依曼结构计算机的五个基本组成部分。

4. 简述ALU、控制器、主存储器、通用寄存器、标志寄存器、指令寄存器、程序计数器、总线的作用。CPU由上述的哪几部分组成？CPU送到地址线的主存地址要首先存放在哪个寄存器中？发送到或从数据线取来的信息要先存放在哪个寄存器中？

CPU送到地址线的主存地址应先存放在存储器地址寄存器（Memary Adies Regster，简称MAR）中，发送到或从数据线取来的信息存放在存储器数据寄存器（Memory Data Register，简称MDR）中。

5. 图1-1：解释冯·诺依曼结构计算机的基本组成和相关。工作原理。

6. 机器指令的0/1序列通常被划分成哪些字段？各字段的含义是什么？

7. 一条机器指令的执行过程通常包含哪几个阶段？

8. 图1-9：简述程序的执行过程。

9. 什么是ISA？ISA主要包含哪些内容？

第二章 数据的机器级表示与处理

1. 信息编码的两大要素是什么？现实世界中的各种媒体信息要怎么样才能在计算机内部进行存储、运算和传送？什么是数字化编码过程？

1.标准和系统化

2.被转化成二进制编码的数字化信息后

3.现实世界中的所有信息经过采样、编码，在计算机内部以二进制编码形式表示

2. 了解什么是数值数据和非数值数据？

3. 数值数据表示的三要素是什么？

1.进位计数制 2.定、浮点表示 3.二进制编码

4. R进制数和十进制数之间的相互转换。

5. 什么是定点数？定点数有哪两种？

小数点位置约定在固定位置

定点整数和定点小数

6. 什么是真值、机器数、原码、补码、反码、移码？

7. 原码、补码的相互转换：给出一个n位整数，能够正确计算它的原码和补码。

8. 以时钟为例说明为什么在模运算系统中，减去一个数等于加上（这个）负数的补码？

9. 为什么说计算机内部的运算电路是一个模运算系统？

10. 对给定的C语言运算表达式，正确计算结果并解释原因：例2.21和习题34、35。

11. 浮点数的表示

（1）为什么要进行尾数的规格化？为什么规格化后尾数部分可以表示多一位的精度：用23个数位表示24位尾数？

（2）浮点数的编码需要对哪几个部分进行编码？

（3）什么是移码、偏置常数？阶码的移码和真值之间的换算关系是什么？

（4）IEEE754浮点数编码标准：

（a）IEEE754浮点数的尾数规格化形式是什么样的？

（b）单/双精度浮点数的尾数、阶码的位数、偏置常数各是多少？

（c）给出一个用IEEE754标准表示的浮点数，可以换算其十进制真值，反之亦然。

（d）IEEE754标准中非规格化模式：

* 0的表示
* 非规格化数：作用、特征是什么？默认阶码是多少？
* +∞/-∞：特征是什么？解释：5/0和5.0/0的区别。
* 非数：什么是非数？非数有什么用处？了解静止的NaN和通知的NaN的特征格式什么？

12. 了解BCD码、中文编码（区位码、国标码、机内码）。

13. 什么是数据的宽度？

14. 什么是数据通路？计算机系统中字长是指什么？字和字长有什么区别？

15. 什么是最高有效字节（MSB）和最低有效字节（LSB）？

16. 什么是大端方式、小端方式？对于一个已知数据的字节数据，能够分别写出它在大端方式和小端方式下的字节排列。C语言中数据的地址指的是MSB的地址还是LSB的地址？存放方式不同的机器间程序移植或数据通信会存在什么问题？

17. 逻辑左移的溢出判定条件是什么？算术左移的溢出判定条件是什么？

18. 了解C语言的基本运算与机器级运算之间的对应关系。

19. 基于n位整数加减运算器（原理图）简述n位整数加减的原理：

（1）如何在同一个电路上实现加、减两种运算？

* sub输入端的作用
* result、Cout的输出
* ZF、CF、SF、OF的设置

（2）一般了解反向器、多路选择器的作用

20. 整数加减：结果必须在可表示范围内，超出范围的需要加2n或减2n。

21. 整数的乘运算：操作数长度为n，乘积长度为2n，数据截断

（1）原码乘法和补码直接相乘

（1）符号数乘法溢出的判断

（2）无符号数乘法溢出的判断

22. 整数的除运算：n位整数除以n位整数，结果还是整数

（1）不能整除时需要进行舍入。舍入规则：朝0方向舍入。

（2）利用右移实现除2k：

不能整除时的舍入处理：低位截断、朝零舍入

* 无符号数、带符号正整数：移出的低位直接丢弃
* 带符号负整数：先加偏移量(2*k*-1)，然后再右移*k* 位。为什么？

23. 浮点数运算

（1）了解浮点数加减法的基本要点：对阶、尾数加减、尾数规格化

（2）了解附加位的设置：Guard (保护位)、Round (舍入位)、sticky（粘位）

（3）四种舍入方式：就近舍入、朝+∞方向舍入、朝-∞方向舍入、朝0方向舍入。重点了解就近舍入的规则（舍入为最近可表示的数（包括舍入为偶数））。

（4）浮点数溢出的判定：阶码溢出。

（5）了解浮点运算中“大数吃小数”现象:浮点数运算不满足加法结合律。

24.爱国者导弹定位错误问题：理解造成错误的根本原因

第三章 程序的转换及机器级表示

1. 什么是微程序、微指令、机器指令？机器指令和微程序是什么关系。

2. 描述机器指令的执行过程：取指、译码、执行。

3. 了解IR、IP寄存器的作用。

4. 了解机器指令的一般格式：操作码+操作数

5. IA-32指令系统：操作数类型、几种基本寻址方式、常用指令（汇编指令）（了解）

6. 什么是有效地址、线性地址？

7. 了解Linux 32位线性地址空间的划分：用户空间、内核空间

用户空间的分布：只读代码段、读写数据段、堆、栈、共享库

8. 应用程序的栈区从哪里开始、向什么方向生长？函数的栈帧由哪个寄存器指示栈帧底、又由哪个寄存器指示栈帧顶？

9. 调用者保存寄存器和被调用者保存寄存器各有哪些？为减少被调用函数的准备和结束阶段的开销，应先使用哪些寄存器？

10. 过程调用中栈和栈帧的变化：设P为调用过程，Q为被调用过程，正确描述P调用Q的过程中P和Q的栈帧变化（保存调用者保存寄存器、参数准备、CALL命令的执行、Q建立自己的栈帧、保存被调用者保存寄存器、Q开辟临时工作区和对局部变量的操纵、入口参数的获取等，重点掌握call、ret、push ebp、leave等指令）。

11. C语言两种参数传递的方式：传值和传地址，从机器级解释这两种参数传递方式的不同，以及被调用函数在获取参数及获取参数后对值参/变参操纵方式上的不同。

12. C语言选择结构的机器级表示

（1）if-else语句的机器级表示：cmp+jmp

（2）switch语句的机器级表示：跳转表。

（3）如何利用跳转表实现switch-case语句的跳转？

13. C语言循环结构的机器级表示

14. 为什么说递归程序的时、空效率较差？以递归和迭代实现的int nn\_sum(int n)为例说明其机器级的根本原因。

15. 逆向工程：例题和相关习题

16. 数组的分配和访问

（1）数组元素的寻址：基址+比例变址。基址寄存器、变址寄存器存放什么数据？比例因子代表什么？

（2）分配在静态区和栈区的数组分别怎么寻址？

17. 结构体数据的分配和访问

（1）结构体成员如何寻址：基址+偏移

（2）结构体数据作为函数的入口参数，在传值和传地址两种参数传递方式下有什么不同？哪种效率高？

18. 联合体数据的分配和访问

（1）联合体数据有什么特点？各数据成员的首地址都等于什么？

（2）如何对联合体成员寻址？

19. 数据对齐

（1）什么是数据对齐？为什么要数据对齐？

（2）了解交叉编址的基本原理。

（3）简单数据类型的数据对齐策略是什么？

（4）结构体数据的对齐策略是什么？正确计算不同对齐策略下一个结构体数据的字节长度。通过调整数据项的顺序优化结构体数据的存储。

20. 什么是缓冲区溢出？造成缓冲区溢出的根本原因是什么？

超越数组存储区范围的写入操作称为缓冲区溢出

没有对栈中作为缓冲区的数组的访问进行越界检查。

21. 什么是缓冲区溢出攻击？简单的缓冲区溢出攻击的基本原理是什么？结合实验lab3（阶段1~阶段4）掌握相关内容。

22. 了解缓冲区溢出攻击的防范措施。

第四章 程序的链接

1. 了解从源程序到可执行目标程序的转换过程：预处理、编译、汇编、链接等各阶段的输入和输出各是什么？

2. 有哪三类目标文件？

可重定位目标文件

可执行目标文件

共享的

3. 什么是可重定位目标文件？产生于哪个阶段？内部编址有什么特点？

4. 什么是可执行目标文件？产生于哪个阶段？内部编址有什么特点？

5. 链接器的主要工作是什么？

将所有关联的可重定位目标文件组合起来，以生成一个可执行文件。

6. ELF格式的全称是什么？ELF目标文件格式有两哪种视图？

可执行可链接格式。链接视图和执行视图。

* 链接视图：什么是节？可链接目标文件由不同的什么组成？有哪些主要的节？

1.是 ELF 文件中具有相同特征的最小可处理单位，不同的节描述了目标文件中不同类型的信息及其特征。

2.节

3. .text节:代码节

.data节: 已初始化的全局数据节

.rodata: 只读数据节

.bss: 未初始化的全局数据节

* 执行视图：什么是段？可执行目标文件由不同的什么组成？有哪些主要的段？

1. 可执行目标文件存储映像中不同性质的一块区域。

2. 段

3. 如只读代码段、读写数据段等。

7. ELF头

（1）了解ELF头包含的主要内容

（2）对比可重定位目标文件和可执行目标文件的ELF头，二者主要有哪些不同？

8. 什么是节头表？了解节头表数据结构各数据项的含义。关注.bss节的特点。

1.描述每个节的基本信息。

2.未初始化全局变量，仅是占位符，不占据任何实际磁盘空间。区分初始化和非初始化是为了空间效率

9. 什么是程序头表？了解程序头表数据结构各数据项的含义。为什么有些段的FileSize和MemSize大小不同？解释其原因（主要针对.bss节的数据）。

1. 程序头表用来说明段信息

2.

10. 什么是静态共享库？静态共享库由什么组成？创建共享库文件的命令是什么？C语言的标准静态共享库叫什么名字？

1.可以将一些相关的目标模块（.o）打包为一个称之为静态库文件的文件（.a）

2.可重定位目标文件

3.gcc –c myproc1.c myproc2.c => ar rcs mylib.a myproc1.o myproc2.o

4.libc.a

11. 链接分哪两步进行？

符合解析和重定位。

12. 符号解析

（1）什么是符号解析？符号解析的对象是什么？编译器将符号的相关信息保存在目标文件的哪个结构里？

是将引用符号与定义符号建立关联的过程。

引用符号和定义符号。

符号表。

（2）什么是符号表（.symtab）？了解符号表数据结构各数据项的含义。对用readelf –s读出来的某目标文件的符号表，能够正确识别其中各个符号的名称、类型、位置（所在节、偏移）、大小等。

包含了在程序模块中被定义和引用的所有符号的相关信息。

（3）什么是Global symbols、Local symbols、External symbols？对一个具体的程序，能够正确指出其中各符号的类型。链接器主要对哪两类符号进行处理？

全局符号、本地符号、外部符号。

全局符号和本地符号。

（4）什么是定义符号和引用符号？对一个具体的程序实例，能够正确指出其中各符号是定义符号还是引用符号。

（5） 什么是强符号定义和弱符号定义？一般程序中哪些位置出现的符号是强符号定义或弱符号定义？对一个具体的程序实例，能够正确指出其中定义的符号哪些是强定义的、哪些是弱定义的。

函数名和已初始化的全局变量名是强符号定义，未初始化的全局变量名是弱符号定义。

（7）链接器对单一定义的符号的解析规则是什么？

单一定义的符号，只要建立引用符号和定义符号的关联即可。

（8）链接器对多重定义的符号的解析规则是什么？对一个具体的程序实例，能够正确写出符号解析后符号定义和符号引用之间的关联关系，及因此而造成程序中的变量之间关联性改变，对出现的显式或隐式错误能够指出产生问题的原因。

Rule 1: 强符号不能多次定义，强符号只能被定义一次，否则链接错误。

Rule 2: 若一个符号被定义为一次强符号和多次弱符号，则按强定义为准，对弱符号被解析为其强定义符号的引用。

Rule 3: 若有多个弱符号定义，则任选其中一个。

（9）能够概述链接器符号解析的全过程

* E、D、U三个集合中分别存放什么对象？

E 将被合并以组成可执行文件的所有目标文件集合

U 当前所有未被解析的引用符号的集合

D 当前所有定义符号的集合

* 对一个具体的程序实例，概述其符号解析的过程，重点描述E、D、U三个集合的变化

P179

* 对静态共享库的操作：其模块被使用或不被使用是怎么判别的？使用到的模块怎么处理？没有被使用到的模块又怎么处理？

（10）为什么符号解析成功与否与命令行中文件的顺序有关？举例说明。

P181

13. 重定位

（1）重定位过程：1）合并相同的节、2）对定义符号进行重定位、3）对引用符号进行重定位，各做什么具体工作？

1. 将集合E的所有目标模块中相同的节合并成同一类型的新节。

2. 根据每个新节在虚拟地址空间中的起始位置以及新节中每个定义符号的位置，为新节的每个定义符号确定其在统一虚拟地址空间中的存储地址。

3. 链接器对合并后新代码节(.text)和新数据节(.data)中的引用符号进行重定位，使其指向对应的定义符号起始处。

（2）链接器怎么知道目标文件中有哪些位置需要重定位？编译器将重定位信息记录在目标文件的什么结构中？

编译器在对源文件进行编译时，会把每个全局符号输出到汇编代码文件中。汇编器遇到引用时，为每个引用生成一个重定位条目记录重定位信息，并保存到目标文件的.rel.data和.rel.text节中。

（3）什么是重定位信息表？了解重定位信息表数据结构各项的含义。.rel.data节和.rel.text节各保存哪种数据的重定位信息。

数据引用的重定位条目在.rel.data节中

指令中引用的重定位条目在.rel.text节中

（4）两种重定位类型：R\_386\_PC32和R\_386\_32

* R\_386\_PC32重定位：
* 相对寻址方式下，有效地址如何计算？

有效地址 = PC的内容+重定位后的32位地址（偏移）

* 什么是重定位的初始值？

此处重定位时，PC距离需要重定位的地址偏移量

* R\_386\_PC32方式下，如何计算重定位值（偏移量）？

ADDR(r\_sym) – ( ( ADDR(.text) + r\_offset ) – init )

* R\_386\_32重定位：R\_386\_32方式下，有效地址等于什么？

初始值加所引用符号地址。

（5）对于一个具体的程序实例，能够根据每处的重定位信息计算重定位（地址）值

14. 了解可执行文件执行时的加载过程。

1）通过execve系统调用函数来启动加载器;

2）加载器(loader)根据可执行文件的程序头表中的信息，将可执行文件的相关节的内容与虚拟地址空间中的只读代码段和可读写数据段建立映射

作业（新版书习题）：

所有布置过的作业 + 适当扩展（如奇数题、偶数题）

实验：

lab1、lab2前四阶段、lab3前四阶段