深入理解计算机系统

信息：对事物运动状态 和变化的反映

是事物联系和相互作用的表征

包含了运动变化的内容

信息和信号

信息存在于通讯系统 ，消息系统 音讯系统中 是这些系统传递的对象

通讯和控制系统 信息是消除不确定 万物之间都可以传递信息

、

寻址：信息装入记忆库或存储装置 还是说寻找地址？

中断：cpu 收到信号，停止当前程序，执行中断处理程序

中断源：来自程序 或者来自外部io设备（如鼠标键盘）

计算机系统

查词学习

文件是 符号名 文件内容 和文件信息的 二进制序列，是磁盘等io设备中数据的抽象

数据结构： 数据的组织方式 和数据的操作 的集合 例如 数据库表 数组 链表 栈 二叉树

类型

字节序列

文本文件：存在不可见的换行符/n 值为10

预处理器，把头文件代码插入到原文件 .i 编译 s转汇编

一根总线传递一位，64根总线传递64位。。。。

Io设备控制器（芯片）或者通过主板插槽（适配器）和io总线相连

动态随机存取存储器dram :大的字节数组 地址是数组索引 int short double

程序计数器Pc (eip) 指向指令 算术逻辑单元alu 寄存器

操作: 加载 存储 操作: 两个寄存器值放入alu，存回 跳转

直接存储器存取DMA 磁盘和主存直接io,.无需占用cpu

高速缓存（存储器）：静态随机访问存储器 sram 多级缓存

操作系统管理硬件（文件是对io的抽象）

汇编器 转可重定位目标程序o (二进制机器码 ) 链接 printf.o 链接系统运行库

静态变量 全局变量

缓冲区溢出

存储器 io设备

外壳shell 字word

Dram主存

进程上下文切换 线程

用户代码内核代码

虚拟存储器virtual memory 虚拟地址空间virtual address space 堆 程序代码和数据.data .text 共享库（静态动态连接） 栈 内核虚拟存储器（为内核保留）

文件 就是字节序列 i/o设备 磁盘，键盘 显示器 网络都可以看作文件

并发 并行

线程：并发 超线程

指令级并行：同时多个指令

单指令多数据并行：一个指令多个数据 （影音，向量？）simd

抽象： 为一组函数规定一个接口api 进程是cpu,memory和io的抽象

二、 程序 无符号数表示正数 补码 表示有符号 浮点数表示实数，科学计数法（二进制） 溢出 浮点运算只是近似的

内存空间是ram 磁盘（虚拟内存）和硬件（显存等）结合起来的、、、

十六进制hexadecimal (hex,h) 0~f 十转16 /16移位 %16 取末位

数据大小：1字节char byte short2 int4 long8 longlong

指针 4 浮点4 double 8

寻址 字节序 多字节对象比如 unicode 比如Int float 只要不是char,ascii都是多字节

小端 大端 lltie big endian intel的基本是反序存入的。。。。

这里的对象指的是单个对象。。。如果是数组或者字符串，每个int是逆序，但数组元素是正序

强制类型转换cast 导致错误

格式化打印 指针和数组 ，2x 以二进制输出

程序show bytes

码表 man ascii

布尔代数 位向量（01组成的串） 用01串表示不同的信息状态（不同整数），然后信息之间还能交并补成另一种状态

位运算

掩码

逻辑运算符和位运算不同，逻辑中所有非0都是true 0是false 运算后返回结果是0或1 10&100===1;

移位运算 方便有符号整数 逻辑右移 左边补0 算术右移 补最高位的值

左移直接补0 无符号必须用逻辑 有符号可以用任何右移 **移动时只会考虑低位，不能超过位数**

默认是有符号 -128 127 long long long

Java只支持有符号

补码 最高位是负权 符号位 1为负 0为非负 ？？？？？？

有符号转无符号 隐式强制转换

不同长度转换 符号扩展 零扩展（补0）zero extension

截断数字 丢弃高位 Mod 2^k

整数运算 是模运算

无符号加法

补码加法 注意临界值

补码的非

无符号乘法 补码乘 溢出

乘常数 用位移和加法代替乘法

除法 右移

浮点数 ieee 二进制小数 1\*2（-1） +0\*2^(-2)

舍入 round

三、汇编级表示

浮点指令

汇编代码中夹杂变量可以吗？

push register

sub esp,4 mov [esp],register

sub esp,4 esp= esp -4 此时能用的空间是[esp+0~4]

局部和参数 的差别

mov ebp esp (把栈底指向栈顶，用来索引相对位置)

push ebp 保存栈底指针 pop ebp

ret 默认返回

此时ebp是栈顶

mov edx,[ebp+8] 栈方向

表达式操作，默认在eax中进行

a+&b 会用两个寄存器 保存两个变量 eax 二元表达式默认？

取指针地址[ edx]

字节 字2 双字4 四字8 （双精度） 寄存器不用指定长度，内存需要指定

数据传送 字节Movb 字w 双字l

eax ecx edx ebx esi edi esp栈指针 可以独立访问寄存器前几位el

操作数：立即数（immediate常数） 寄存器 存储器

寻址方式 间接 [register] [register+立即] 寄存器是栈基址， 偏移量

变 址 [ebx+esi+数] 比例变址[ebx+esi\*4]???

数据传送 mov movsx

栈底在上，栈顶在下 pop时esp上移 pop值保存在eax

加载地址lea 必须存到寄存器内 lea eax

一元操作 inc ++ --

二元操作 mov add 第一个操作数是自己 eax+=eax。。。。当语句到了分号的时候，寄存器就可以放开了？不用保存任何值？ Eax

所有表达式最后的值都用eax保存

ret命令最后只是长度单位？需要吗？

移位（右移时要求区分有符号和无符号，其他不需要） sal和 shl一样 sar算术移位，填符号

and or not

乘法imul 除法 div 高位扩展

条件码寄存器 所有的指令都会改变cf 无符号进位 sf zf of

sef指令 cmp test cmp+jnz/jb组合

跳转和循环 if else 多层，二义性了； while for do-while

栈帧：

浮点体系结构 x87 sse