1. C语言中char 默认是有符号类型的，但程序对char是否有符号并不敏感
2. 文本数据比二进制数据有更强的平台独立性，在所有用ASCII编码的系统上都有相同的结果
3. 可以执行命令man ascii来得到一张ascii表
4. 计算机系统的一个基本概念是从机器的角度来看，程序仅仅只是字节序列
5. 在java中，x>>k执行逻辑右移，而x>>>k执行算术右移
6. 0的位表示在补码和反码的意义下都是唯一的，而在原码的意义下不唯一，因为原码有一个符号位
7. C标准要求用补码形式来表示有符号整数，用<limits.h>库引入常量来标记有符号数和无符号数的范围
8. 大多数默认输入的数字是有符号的，创建无符号常量需要在数字后加上u
9. 在同时进行大小和有无符号的类型转换时，我们应当先转换大小，再转换有无符号
10. 需要注意一些宏和库函数会隐式地采用无符号数作为返回值，比如size\_t,sizeof等，而strlen则是用有符号数作为返回值
11. 很多编程语言只支持有限精度的整数运算，比如C、C++、Lisp等，但也有一些语言支持无限精度的整数运算（当然不能超过内存大小）
12. 有符号加法是否溢出可以通过减法来检验，即若x+y=s且发生了溢出，那么会有x!=s-y或y!=s-x
13. 一个补码的相反数是将这个补码的每一位取反得到的数+1
14. 无符号乘积和对应的补码乘积会得到相同的位表示，然后截断成合适的长度按照无符号或补码的规则进行解释就能得到正确的结果
15. 补码乘法是否溢出可以通过除法来检验，即若x\*y=p且没有发生溢出，那么会有x=0或p/x=y
16. 乘以一个整型常数一定可以通过移位和加减法的组合来实现优化
17. x>>k的值会向下取整，而在补码除法中执行向零舍入，因此如果想用移位代替除法，在处理负数时需要加上偏置量bias=(1<<k)，这样(x+bias)>>k=x/2^k
18. 单精度浮点数的bias=127，双精度浮点数的bias=2047
19. 在IEEE规则下，浮点数数值越大，能精确表示的数值越稀疏
20. 浮点数的加法可交换但不可结合，无穷和NAN没有负元
21. 除NaN外，浮点加法和乘法满足单调性
22. C语言中规定浮点数使用IEEE规则表示，float为单精度浮点数，double为双精度浮点数
23. 在命令行中使用-S选项能看到编译出的汇编代码，使用-c选项会编译并汇编该代码得到.o文件
24. 在.o文件上运行的gdb中执行命令x/pxb func表示显示从函数func所处地址开始的p个16进制表示的字节
25. 带”-c”标识的objdump可以反汇编机器代码
26. 使用命令”-masm=Intel”可以生成Intel格式的汇编代码，其省略了指示大小的后缀、寄存器名字前面的%，在多个操作数的命令中列出操作数的顺序则与ATT格式一致。
27. Intel用术语word字表示8位数据类型，称32位为4字，64位为8字，要区别其与机器字的不同
28. 只有用于传递参数和返回值的寄存器是调用者保存（caller saved）的，其余寄存器（除%rsp）都是被调用者保存（callee saved）
29. 变址偏移量寻址的伸展量s只能是1,2,4,8
30. 数据传送指令不能直接将一个立即数传递给内存，必须通过寄存器中转
31. movq指令若传递立即数，只能以32位补码数字作为源，然后将其零扩展至64位，如果想传递64位立即数需要用movabsq指令，但这里的目的只能是寄存器
32. cltq S指令的操作数S必须是一个32位寄存器，其把32位寄存器的值符号扩展到对应的64位寄存器
33. 可以用leaq指令实现一些简单的计算，其目的操作数必须是寄存器
34. 移位操作的移位量可以是一个立即数或存放在单字节寄存器%cl中
35. imulq既可以有两个操作数，计算两个64位数的乘积，也可以只有一个操作数，计算%rdx中数值与操作数的全128位乘积，将低64位储存在%rax中，高64位储存在%rdx中
36. 128位全除法指令是(i)divq，其只有一个操作数，将储存在%rdx中的高64位与储存在%rax中的低64位除以操作数S，将商放在%rdx，将余数放在%rax，可以用cqto将%rax符号扩展到%rdx来实现普通的64位除法
37. 默写条件码
38. 逻辑操作会把进位、溢出和符号标志设置成0，移位操作会把溢出标志设置成最后一个被移出的位，而INC和DEC操作会设置溢出和零标志，但不会设置进位标志
39. 默写条件判断规则
40. 跳转指令可以是直接跳转，也可以是间接跳转（用\*表示从寄存器或内存中读出的地址）
41. 跳转指令的一种编码方式是PC相对的，其会将跳转的目标地址与跳转指令地址的差作作为编码
42. C语言标准规定了for循环与while循环的等价性，但需要特殊处理continue的情况
43. switch在开关情况的数量比较多时就会使用跳转表
44. 过程P调用过程Q，过程Q的返回地址会被储存在P的栈帧中，而Q所需要的超过寄存器能存储数量的参数则会放在Q的栈帧中
45. 通过栈传递参数时，所有数据大小都向8的倍数对齐
46. 同一数据结构中两个指针之差结果的类型是long，其值等于两个地址之差除以该数据类型的大小
47. 对于一个结构指针obj，引用其元素时必须写成\*(obj).name或obj->name而不能写成\*obj.name
48. 为了让x86-64系统正常工作并提高效率，应当使用数据对齐，对齐原则是任何K字节的基本对象的地址都必须是K的倍数。
49. 使用gets或其他能导致缓冲区溢出的函数（如strcpy,strcat）等都是不好的习惯，应当使用fgets,sscanf,sprintf等限制了缓冲区的函数
50. 病毒可以自己运行并将自己的等效副本传递给其他机器
51. leave指令会恢复帧指针的值，等价于popq %rbp;movq %rbp,%rsp