总结性复习提纲（v1.0）

（绝版！无再版！！【已跑路！！！】）

**二进制部分**

1. 数值的表示方式
   1. 实数：一般采用IEEE754标准进行数据存储。可表示的相邻数值之间等间隔条件不成立（即：等间隔仅在某一区段内成立），在不同区段中可表示的相邻数值之间的间隔不同。整体表现为：所对应数值的绝对值越大，与它相邻的可表示的数值就相隔越远。处于此间隔之区间中的数值不可被精确表示（以就近方式近似表示）。实数计算不会有溢出现象（负数变为正数，或者正数变为负数，或者大数变为小数），但计算结果可能会被表示为正无穷或负无穷。整数的除以0操作会导致异常，但浮点数的除以0操作不会，因为浮点数中有正无穷或负无穷表示方式。
   2. 整数：一般以补码形式存储。可表示的相邻数值之间等间隔，间隔为1。整数计算实际上都是通过逻辑计算来实现。另外，计算机实际上“不会算”减法。
2. 补码的计算方式：
   1. 对于正数：原码本身
   2. 对于负数：该负数所对应的正数的原码取反+1。
3. 位扩展（如：char扩展至short，short扩展至int，int扩展至long）
   1. 有符号数：原bit不变，左边补充的bit都与最高位（符号位）一致，直至扩展至目标类型对应的位数。

例：short型的-1024（二进制表示为：**1111 1100 0000 0000**）扩展为int型，结果为：1111 1111 1111 1111 **1111 1100 0000 0000**

* 1. 无符号数：原bit不变，左边补充0，直至扩展至目标类型对应的bit数。

例：unsigned char型的129（二进制表示为：**1000 0001**）扩展为unsigedn int型，结果为：0000 0000 0000 0000 0000 0000 **1000 0001**

1. 位操作
   1. 按位与 &：任意bit值（0者1）与0相与——结果等于0；任意bit值（0或者1）与1相与——结果等于原bit值。
   2. 按位或 |：任意bit值（0者1）与0相或——结果等于原bit值；任意bit值（0或者1）与1相或——结果等于1。
   3. 按位异或 ^：任意bit值（0者1）与0相异或——结果等于原bit值；任意bit值（0或者1）与1相异或——结果等于原bit值取反。
2. 移位操作

左移<<

对于有符号数和无符号数，左边移出的bit抛弃，右边补0

右移>>

对于有符号数，右边移出的bit抛弃，左边补符号位（正数补0，负数补1）

对于无符号数，右边移出的bit抛弃，左边补0

**X86\_64**体系结构下的**数据类型及所占字节数**

char 1字节

short 2字节

int 4字节

long 8字节

指针 8字节

**数据存储模式**

大端模式：高位地址放低位字节

小端模式：地位地址放高位字节

**汇编部分常用指令**

mov xxx，yyy

数据传送指令，有很多种表示形式。具体参考寻址模式部分。

add var1，var2

加法操作。相当于var2=var2+var1；

sub var1，var2

减法操作。相当于var2=var2-var1；

push xxx 压栈指令

本条指令共执行两个操作。先将栈顶指针寄存器%rsp的值减小8，然后再将xxx存入当前%rsp寄存器所指向的栈内存区域。

pop xxx

本条指令共执行两个操作。先将%rsp寄存器所指向的内存单元中的数据出栈到xxx（这里的xxx应为寄存器），然后再将栈顶指针寄存器%rsp的值增加8。

jmp xxx

功能为跳转到xxx的值所指的位置处继续执行。跳转操作相当于修改%rip的值（也常被叫做PC，指示当前正在执行的指令的下一条指令的位置）。可以为直接跳转（xxx的值表示绝对位置）也可以是相对跳转（xxx值表示目标位置和当前PC间的差值）。此指令只是执行单纯的跳转，并不保存任何返回地址。直接跳转/相对跳转采用不同的机器码表示jmp，但汇编语言方面都写为call。

call xxx

功能为调用xxx的值所指的位置处的函数代码。相当于先将返回地址（call指令的下一条指令的绝对位置）压栈，再jmp到xxx位置处继续执行。调用形式可以是直接调用（xxx的值表示绝对位置）也可以是相对调用（xxx值表示目标位置和当前PC间的差值）。直接调用/相对调用采用不同的机器码表示call，但汇编语言方面都写为call。

ret

功能从当前的函数返回，相当于C语言中的return;。执行过程相当于先将返回地址（配对的call指令所压栈的返回地址）出栈，再jmp到该返回地址处继续执行。

cmp var1，var2

计算var2-var1，不保存计算结果，但根据计算结果改写条件码（也叫做标志位）。

寻址方式：

1. 立即数寻址，如mov $0x3f,%eax

将0x0000003f这个数值装进%rax的第4个字节，%rax高4个字节被自动清0。是否立即数寻址重点查看有无$符号。

1. 直接寻址，mov 0x6001f0,%rax

将0x6001f0单元为起始地址的内容传送至%rax中。

1. 间接寻址，mov (%rcx),%rax

将%rcx的值作为地址，去该地址处取数据，传送至%rax中。

1. 变址寻址，如mov 0x3c(%rbp),%rax

将%rbp的值 + 0x3c作为地址，将该地址所存储的内容传送至%rax

1. BIS（有时也叫做BSI寻址），如：

mov 0x4006c8（%rbx, %rcx, 8），%rax

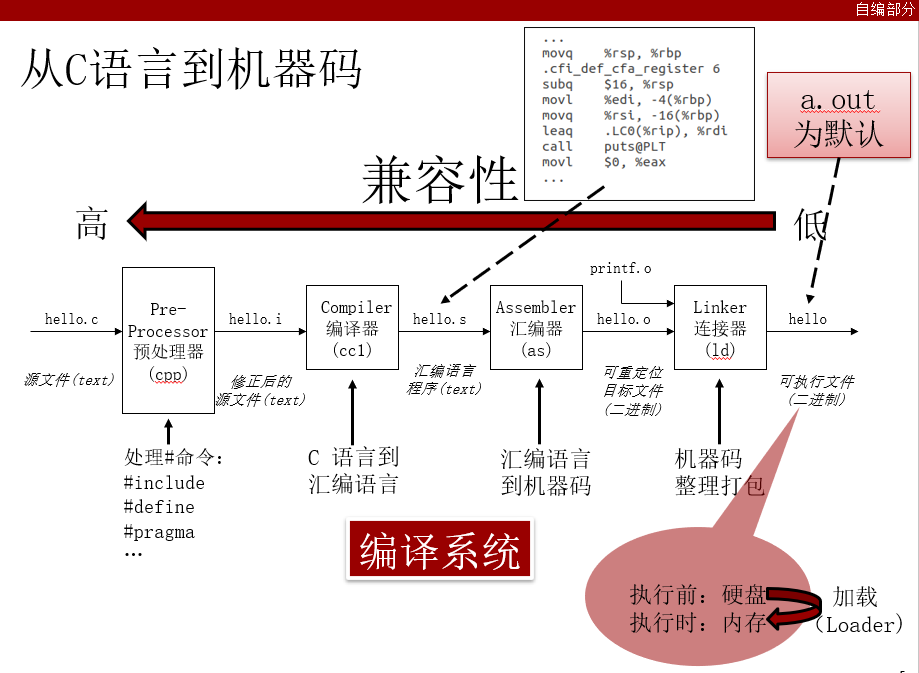
将0x4006c8 + %rbx + %rcx\*8 作为地址，将该地址所存储的内容传送至%rax。

此种寻址模式允许省略，如果用抽象形式表示OFFSET（BASE,INDEX,SCALE）,各缺省值（在寻址时空着不填的情况）。如果OFFSET/BASE/INDEX缺省，意味着这些值取0；如果SCALE缺省，意味着SCALE值取1。

数据类型

1. char/short/int/long \*var[10] 指针数组，共有10个元素，每个元素都是指针变量，每个元素占8字节
2. char/short/int/long (\*var)[10] 指向数组的指针，是一个指针变量，占8个字节

**C代码到机器代码所经历的过程**，参考以下PPT内容，书本上也有对应论述。



**栈、栈帧和调用约定**

1. 与栈相关的常用指令：push，pop，call，ret
2. 栈中存储的典型内容：局部变量、返回地址、寄存器原值（由于寄存器后期将被改写，暂行保存留作恢复时使用），实参（序号超过6的参数部分）
3. 栈的存在形态：

栈位于虚拟内存的最顶端位置（紧挨内核区域）

逆向生长（每次有数据压栈时，栈顶指针会减小）

1. 参数传递方式：

前六个参数分别采用：%rdi,%rsi,%rdx,%rcx,%r8,%r9

第七个及后续的参数传递时使用栈。

1. 任何函数的返回值的传递方式：使用%rax寄存器
2. “调用者”负责保存/恢复的寄存器：%rax,%rdi,%rsi,%rdx,%rcx,%r8,%r9,%r10,%r11

“被调用者”保存/恢复的寄存器：%rbx,%r12,%r13,%r14,%rbp,%rsp(实际上%rsp使用的是call与ret结合的特殊形式保存/恢复)

1. 索引局部变量/参数可以使用%rbp（栈基指针，具体使用方式一般为0xOFFSET(%rbp)），也可以使用%rsp（栈顶指针，具体使用方式一般为0xOFFSET(%rbp)），具体使用哪一个取决于编译时使用的优化级别。
2. 递归函数会在编写时比较简短高效，但不宜大量经常使用。尤其是递归层级很深的函数，因为需要为每一层递归创建一个栈帧，大量消耗栈内存资源。

**符号、链接与ELF**

1. 符号的概念

“符号”指的是进程（虚拟内存）中始终有效的特殊地址与该地址内容的名字的结合体。也就是说符号是将某一个变量名与“始终有效的”地址对应起来。所以，全局变量名、函数名、任何static型变量名都属于符号；而局部变量名（有时也被称为auto型变量）不属于符号，因为局部变量是放在栈里，不具有恒定地址（每次被创建时位置不确定）。

1. 强符号：有函数体的函数名，带初值的全局变量都是强符号。

弱符号：无函数体的函数名，不带初值的全局变量为弱符号。

全局符号：在某一个模块m内被定义，但是除m以外还可以被其他模块所引用符号。例如：非static型函数和非static型全局变量。

外部符号：在某一个模块m中被引用，但是由其他的模块定义的全局符号。

本地符号：在某一个模块m内被定义，但只能在m内部被引用的符号。例如：static型函数和static型全局变量。

1. 链接

链接的本质是“合并相同的节”。

链接把若干个可重定位目标文件变成可执行目标文件。

链接的两个重要步骤：符号解析（将符号的定义和符号引用之间一一确定对应关系）;重定位（调整修改链接前（实际地址无法被确认的）那些调用和引用位置）。重定位是针对符号进行的，局部变量不属于符号，所以不会进行重定位。

1. 符号解析规则：

规则 1: 多个（同名的）强符号定义是不被允许的。链接过程会报错。

规则 2: 当一个强符号和多个（同名的）弱符号同时存在是，以强符号为准。

规则 3: 如果有多个（同名的）弱符号定义（无同名强符号），随机选择一个。

1. ELF头（又称，程序的脸），主要包含内容有：

魔幻数字（magic number），

ELF类型（32bit/64bit等），

数据表示模式（整数存储方式【一般为2进制补码】，大小端模式信息）

OS/ABI（对应的操作系统类型，二进制接口类型）

文件类型（可重定位目标文件/可执行目标文件/共享库）

机器类型（IA32/x86\_64/ARM64等）

入口地址（程序的第一条指令的存放位置）

段（程序）头表位置

节头表位置

等等。。。

此外，还需要自行仔细复习的重点内容还包括：

bomblab实验的基本阶段反汇编部分，

attacklab基本阶段的具体原理部分，

书本上第三章分支/循环语句的泛式/反汇编部分内容，

书本上第三章缓冲区溢出部分内容，

书本上第三章缓冲区对抗技术部分内容

写到这基本已吐血，大家都努力吧，补考/重修一定不再给总结这破玩意了，知识点太多，太累了！！

好好复习，都过了吧，别再折磨我们了

祝大家健康快乐～

——艾你们的课程组老师们