大计基第七次作业

刘天瑞

院 （系）：英才学院 专 业：未来技术模块

学 号：7203610121 指导教师：史先俊

**2020年11月**



**大计基**

**题目：第七章作业**

**专 业：英才**

**学 号：7203610121**

**学 生：刘天瑞**

**指导教师：史先俊**

**答辩日期：无**

**目 录**

**第一题……....………………………………………………………………….4**

**第二题…………………………………………………………………….7**

**第三题…………………………………………………………………….8**

**第四题…………………………………………………………………….9**

**第五题…………………………………………………………………….12**

**第六题…………………………………………………………………….13**

**参考文献……………………………………………………………………………16**

**第一题**

**题面：C语言源程序，经过哪些步骤，各产生什么中间产物，最后生成了机器语言的执行程序？**

答：详细过程如下所示：

1.预处理

在这一阶段，源码中的所有预处理语句得到62616964757a686964616fe58685e5aeb931333433646532处理，例如：#include语句所包含的文件内容替换掉语句本身，所有已定义的宏被展开。

根据#ifdef，#if等语句的条件是否成立取舍相应的部分，预处理之后源码中不再包含任何预处理语句。

GCC预处理阶段可以生成.i的文件，通过选项-E可以使编译器在预处理结束时就停止编译。例如：gcc -E -o hello.i hello.c

2.编译

这一阶段，编译器对源码进行词法分析、语法分析、优化等操作，最后生成汇编代码。这是整个过程中最重要的一步，因此也常把整个过程称为编译。

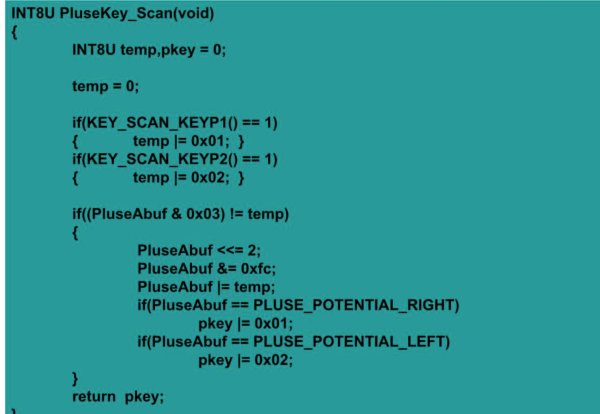
可以通过选项-S使GCC在进行完编译后停止，生成.s的汇编程序。例如：gcc -S -o hello.s hello.c

3.汇编

这一阶段使用汇编器对汇编代码进行处理，生成机器语言代码，保存在后缀为.o的目标文件中。

当程序由多个代码文件构成时，每个文件都要先完成汇编工作，生成.o目标文件后，才能进入下一步的链接工作。

目标文件已经是最终程序的某一部分了，只是在链接之前还不能执行。可以通过-c选项生成目标文件：gcc -c -o hello.o hello.c

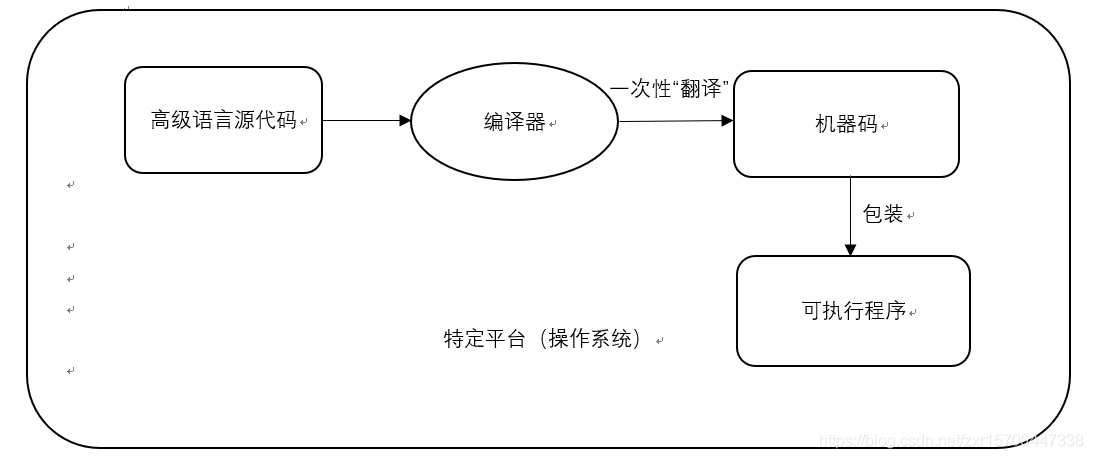


4.链接

经过汇编以后的机器代码还不能直接运行。为了使操作系统能够正确加载可执行文件，文件中必须包含固定格式的信息头，还必须与系统提供的启动代码链接起来才能正常运行，这些工作都是由链接器来完成的。gcc -o hello hello.c

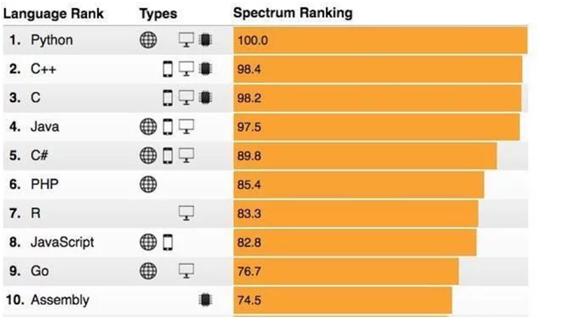
5.运行：执行.EXE文件，得到运行结果。

其实C语言过程可以简化成如下所示的流程图：

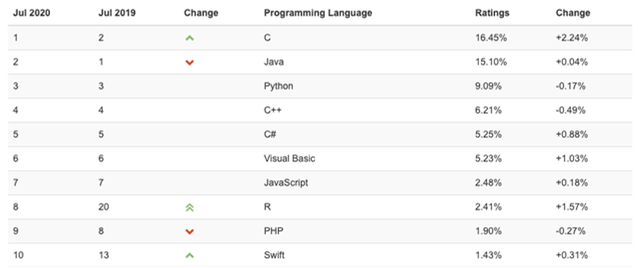


**第二题**

**题面：目前TOP10语言是哪些？截图验证。**

答：截图如下所示：

或者：



**第三题**

**题面：C语言的函数有哪些构成要素？举例并标注。**

答：1.数据：类型、常量、变量(全局/局部/静态)、表达式（常量/变量、算术/关系=比较、逻辑=条件）、复合数据类型（数组、指针、结构/联合-嵌套即链表、树、图）

2.赋值 = ,逗号操作符，变量的赋初值/不赋初值

3.类型转换(隐式或显式) unsigned/char/int/long/float/double

4.sizeof 伪操作符、宏定义与宏调用

5.算术操作：+ - \* / % ++ -- 取正/负+- 复合“+=”等

6.逻辑/位操作：逻辑&& || ! 位 & | ~ ^ 移位>> << 复合操作如 “|=” 或“<<=”等

7.关系操作：== != > < >= <=

8.数组/指针/结构操作：A[i] &v \*p s.id p->id

9.控制转移：分支(if/else switch ?:) 循环( for while do/while continue break)

10.函数操作：参数传递(地址/值)、函数调用()、函数返回 return。迭代与递归。

11.简单语句、复合语句、文件、项目Project、库Lib、工作空间WorkSpace。

**第四题**

**题面：函数在机器底层是如何实现的？传值与传地址在实现上有什么不同吗？**

**答：1.函数在机器底层实现：**

函数体：是全部机器程序代码的一部分，函数名就是函数的起始地址。

参数传递：32位使用堆栈，倒序入栈，函数/子程序内使用[BP+n]操作参数。

64位用寄存器，参数多于4/6个的再用堆栈。

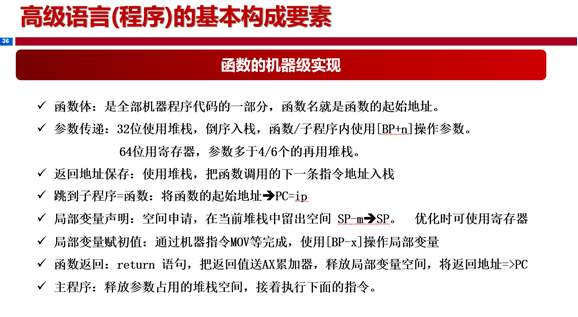
返回地址保存：使用堆栈，把函数调用的下一条指令地址入栈跳到子程序=函数：将函数的起始地址🡺PC=ip。

局部变量声明：空间申请，在当前堆栈中留出空间 SP-m🡺SP。优化时可使用寄存器。

局部变量赋初值：通过机器指令MOV等完成，使用[BP-x]操作局部变量。

函数返回：return 语句，把返回值送AX累加器，释放局部变量空间，将返回地址=>PC。

主程序：释放参数占用的堆栈空间，接着执行下面的指令。

如图：

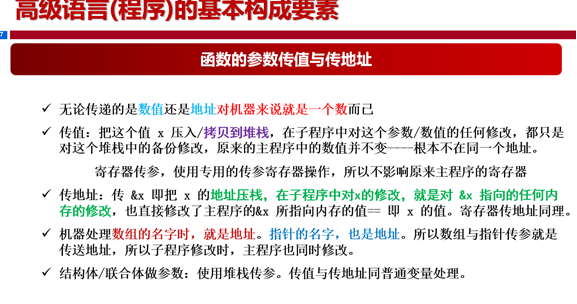
**2.传值与传地址区别：**

传值：把这个值 x 压入/拷贝到堆栈，在子程序中对这个参数/数值的任何修改，都只是对这个堆栈中的备份修改，原来的主程序中的数值并不变----根本不在同一个地址。

寄存器传参，使用专用的传参寄存器操作，所以不影响原来主程序的寄存器

传地址：传 &x 即把 x 的地址压栈，在子程序中对x的修改，就是对 &x 指向的任何内存的修改，也直接修改了主程序的&x 所指向内存的值== 即 x 的值。寄存器传地址同理。

机器处理数组的名字时，就是地址。指针的名字，也是地址。所以数组与指针传参就是传送地址，所以子程序修改时，主程序也同时修改。

如图：

**第五题**

**题面：编译与解释有什么区别？每种实现各举两种不同编程语言。**

答：解释程序不产生目标代码，它逐条地取出源程序中的语句，边解释，边执行；解释器把源代码文件边解释成机器语言边交给CPU执行。即解释器每次接收一行源程序，翻译成机器语言执行。如js、sql、html、Python语言。

编译是将源程序翻译成可执行的目标代码（二进制执行代码），执行可执行程序文件，翻译与执行是分开的。如C、java语言。

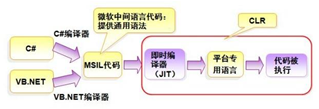
其中编译效率高、解释效率低。编译后只需要执行程序，解释需要源程序+解释器运行。编译后不需源程序，而解释需要源程序（或简单加密后）。而且除此之外编译与解释也在融合，如matlab、basic等都支持。

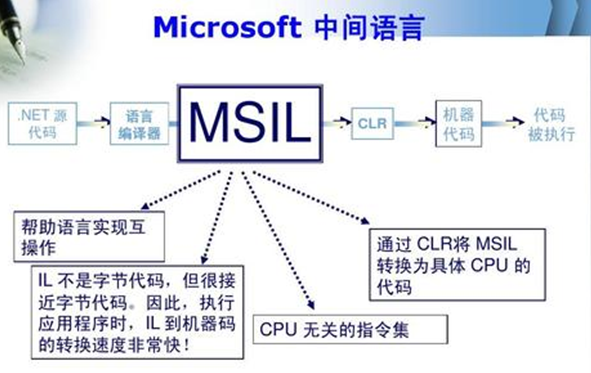
**第六题**

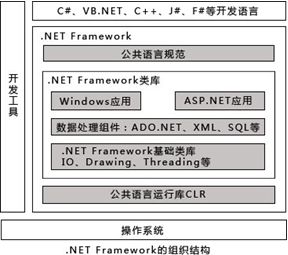
**题面：什么是中间层语言？以微软和SUN为例说明其实现方式。**

答：中间层语言：即虚拟机运行时的机器语言。Java/c/c#等编译后生成的语言。有其对应的汇编语言。

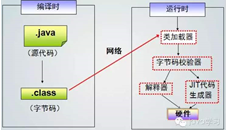
微软的跨平台解决方案：

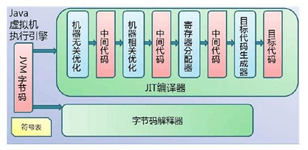


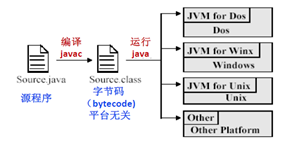


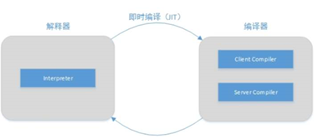


SUN公司的解决方案：









**参考文献**

百度百科

知乎

百度贴吧

百度文库

大计基PPT