**C语言解释器**

* ***第一节***

1. **注解**

解释器：意味着可以像运行脚本一样运行C语言的源文件代码。

1. 解释器 与 编译器 的异同：

异：在代码生成阶段有区别。

同：语法分析、词法分析都是一样的。

1. 解释器 需要实现自己的虚拟机、指令集

（有助于了解 计算机工作原理）

1. **主要步骤**
2. 词法分析器

用于将 字符串转换成内部的表示结构。

(已有的现成工具flex 用于词法分析)

1. 语法分析器

将词法分析得到的 标记流(token)生成一棵语法树。

(已有的现成工具bison 用于语法分析)

1. 目标代码的生成

将语法树 转换成 目标代码。

1. **解释器的框架**
2. next()

用于词法分析，获取下一个标记，它将自动忽略空白字符。

1. program()

语法分析的入口，分析整个C语言程序。

1. expression()

用于解析一个表达式

1. eval()

虚拟机的入口，用于解释目标代码

* ***第二节***

1. **相关理论**

CPU、内存、寄存器

(1) 可执行代码 以 二进制形式保存在 内存中。

(2) CPU 从 内存中 一条条加载指令执行。

(3) 程序运行的状态保存在寄存器中

1、内存

现代操作系统 都不直接使用内存，而是使用虚拟内存，虚拟内存 可以理解为是一种映射，虚拟内存的作用是屏蔽了物理的细节。在程序看来，可以访问全部的内存地址，而OS需要将它映射到实际内存上。

一般来说，进程的内存分布：

1. 代码段 text：用于存放代码(指令)
2. 数据段 data：用于存放已经初始化的数据
3. 未初始化数据段 bss：用于存放未初始化的数据
4. 栈 stack：用于处理函数调用相关数据、局部变量
5. 堆 heap：用于程序动态分配内存

大致类似于：

+------------------+  
| stack | | high address  
| ... v |  
| |  
| |  
| |  
| |  
| ... ^ |  
| heap | |  
+------------------+  
| bss segment |  
+------------------+  
| data segment |  
+------------------+  
| text segment | low address  
+------------------+

自己要实现的虚拟机并不模拟完整的计算机，只关心三个内容：代码段、数据段、栈。其中：数据段 只存放字符串。

理论上要实现的虚拟器需要维护自己的堆用于内存分配，但实际实现上较为复杂且与编译无关，故引入一个指令MSET，使程序能直接使用解释器中的内存。

因此，需要添加全局变量：

int \*text, // text segment  
 \*old\_text, // for dump text segment  
 \*stack; // stack  
char \*data; // data segment

注意：这里的类型，虽然是int型，但理解起来应该作为无符号的整型，因为我们会在代码段（text）中存放如指针/内存地址的数据，它们就是无符号的。

其中数据段（data）由于只存放字符串，所以是 char \* 型的。

接着，在main()中添加代码，分配内存。

（具体代码见second\_version.c）

2、寄存器

计算机中的寄存器 用于存放计算机的运行状态，PC中有许多寄存器，但是自己实现的虚拟机 只是用4个寄存器：

1. pc 程序计数器：存放的是一个内存地址，该地址中存放的是 下一条要执行的指令。
2. sp 指针寄存器：指向当前栈顶。栈是由高地址 向 低地址增长的，所以 入栈时sp是减小的。
3. bp 基址指针：指向栈底
4. ax 通用寄存器：在自己实现的虚拟机中，它用于存放 一条指令执行的后的结果。

要理解这些寄存器的作用，需要去理解程序运行中会有哪些状态。而这些寄存器只是用于保存这些状态的。

在全局中加入如下定义：

int \*pc, \*bp, \*sp, ax, cycle; //virtual machine registers

在main函数中加入初始化代码，如下，注意一点：pc的初始值应该指向目标代码中的main函数，由于目前还没有写关于编译相关的代码，暂时先不处理。

3、指令集

与CPU相关的是指令集。指令集 是 CPU能够识别的命令的集合，即CPU能理解的语言。

这里要为虚拟机构建指令集，基于X86架构，比计算机的指令更加简单。在全局中加入一个枚举数据结构，存放虚拟机支持的全部指令。

//