1 选择C语言

C语言是一门较少依赖操作系统的语言，适合用来开发操作系统。

C语言中无与“OUT/IN”、“CLI/STI”、“PUSHFD/POPFD”、“HLT”等汇编指令相对应的C语句。在C程序中，编译器会选择一些寄存器来保存一些重要的值，C语言不是可以访问到所有的寄存器。

2 C程序到机器码

C程序只有经过C编译器和链接器才能被CPU执行。“书”中利用以下工具，最终将C语言转换成32位机器指令（CPU已经被切换到32位保护模式，导入C语言后用C语言来设置与中断相关的东西，再用汇编指令STI开启中断）：

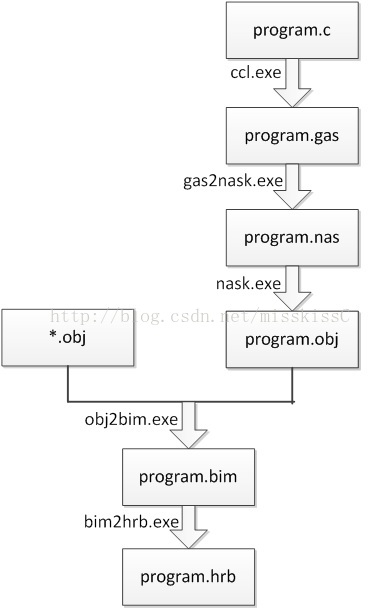


Figure1. C语言到机器语言

3 C程序与汇编程序的合并（机器码）

写给操作系统的C程序，用链接器将C的各个目标文件（包含唯一程序入口）连接在一块形成可执行程序，用加载器将其加载到内存中即可运行这部分C程序。

在无操作系统的情况下，CPU先执行IPL，再跳转去执行x86实模式切换到保护模式的代码（称为R2P），这时才能跳转去执行C程序的机器码（32位机器码）。那么就应该在编辑器中组织IPL、R2P以及C程序及它们在内存（偏移）地址的关系，让IPL、R2P、C依次得到执行。按照“书”中的方法来：

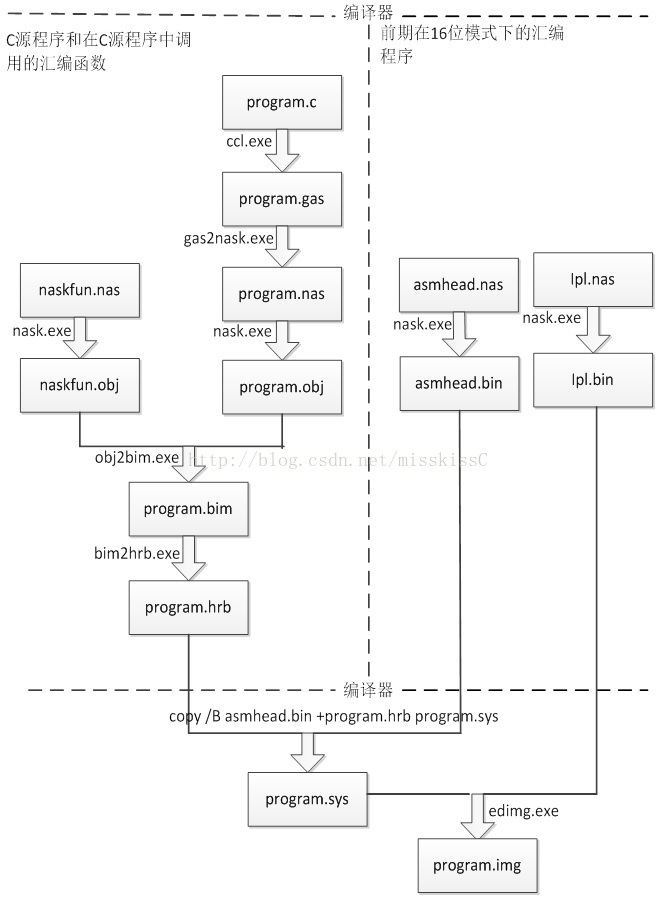


Figure 2. 组织文件结构

用copy命令合并asmhead.bin和program.hrb得到program.sys，program.hrb的内容紧跟asmhead.bin内存。

在用eding.exe工具得到的program.img文件中，ipl.bin文件的内容从program.img文件偏移量为0处开始；program.sys文件名从program.img文件偏移量处0x002600处开始，program.sys内容从偏移量0x4200处开始。

那么对于IPL程序末尾来说，asmhead.nas程序在内存中的偏移应为0x8000+ 0x4200 = 0xc200。（IPL将程序对应内存0x8000~ 0x81ff）

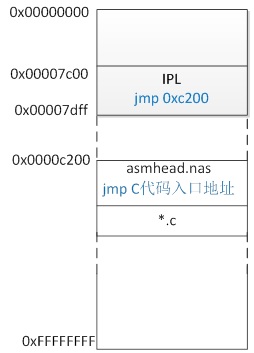


Figure3. IPL asmhea.nas \*.c程序在内存地址空间中的布局（实模式）

4 进入保护模式后，将代码拷贝到设定的段中

32为的机器指令不能在实模式下运行，进入保护模式后，C语言程序需要运行到已经设定的段之上，并找到C代码的入口地址。

|  |
| --- |
|  |

1.        BOTPAK    EQU          0x00280000              ; C代码程序的起始内存地址空间

2.        DSKCAC    EQU          0x00100000              ;IPL及asmhead.nas程序起始的内存地址空间

3.        DSKCAC0 EQU          0x00008200      ;磁盘源程序的起始地址空间

4.

5.        org 0xc200

6.        ;画面设置，实模式到保护模式的代码

7.

8.                 ;拷贝32位C代码到设定的段3中

9.                 MOV         ESI,bootpack             ;保存32位C指令的内存起始地址

10.             MOV         EDI,BOTPAK              ;32位C代码位于的段

11.             MOV         ECX,512\*1024/4

12.             CALL         memcpy

13.

14.             ;随便将曾运行在保护模式下的汇编代码拷贝到内存地址在1M以后的内存中

15.             MOV         ESI,0x7c00        ;IPL程序起始地址

16.             MOV         EDI,DSKCAC     ;将IPL程序拷贝到新的内存中

17.             MOV         ECX,512/4

18.             CALL         memcpy

19.

20.             MOV         ESI,DSKCAC0    ;0x8200

21.             MOV         EDI,DSKCAC     ;紧跟IPL

22.             MOV         ECX,0

23.             MOV         CL,BYTE[CYLS]

24.             IMUL        ECX,512\*18\*2/4

25.             SUB  ECX,512/4

26.             CALL        memcpy

27.

28.

29.    ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

30.    ;C代码的启动

31.             MOV         EBX,BOTPAK

32.             MOV         ECX,[EBX+16]  ;.hrb文件内容（二进制编辑器可查看）

33.             ADD ECX,3

34.             SHR  ECX,2

35.             JZ     skip           ;无传送的东西

36.             MOV         ESI,[EBX+20]    ;

37.             ADD ESI,EBX              ;0x00280000 + ...

38.             MOV         EDI,[EBX+12]   ;

39.             CALL         memcpy

40.    skip:

41.             MOV         ESP,[EBX+12]    ;

42.             JMP DWORD 2\*8:0x0000001b        ;**HariMain函数首地址，之前一段是作者开发的链接组织的内容（Page-466）**

43.

44.    ;将拷贝起始ESI的4\*ECX字节内容到起始于EDI的内存中

45.    memcpy:

46.             MOV         EAX,[ESI]

47.             ADD ESI,4

48.             MOV         [EDI],EAX

49.             ADD EDI,4

50.             SUB  ECX,1

51.             JNZ  memcpy

52.             RET

53.

54.             ALIGNB    16

55.    bootpack:

|  |
| --- |
|  |

C代码的启动那部分代码是依据编译器、链接器往C代码中所加入的信息而编写的。

这段代码执行过后，各程序在内存中的分布为：

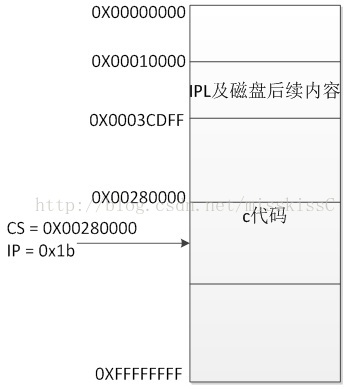


Figure4. 进入保护模式后各个代码在内存中的位置

C代码的入口地址是“书”者从二进制文件中看来的，C代码正好运行在段号为2的段中。

5 编写C程序

5.1 C程序入口

“书”者的C编译器由gcc改编而来，将C程序的入口地址改为了HariMain()。

5.2 让CPU休眠

C语言不含让CPU休眠的语句，这个功能需要用汇编指令HLT来实现。由2，“书”者在naskfunc.nas中用汇编语言编写供C语言调用的函数，遵循之。

|  |
| --- |
|  |

1.       ; naskfunc

2.       ; TAB=8

3.

4.       [FORMAT "WCOFF"]     ;生成的目标文件的格式

5.       [BITS 32]            ;生成32位机器码

6.

7.

8.       [FILE "naskfunc.nas"]             ;本文件名

9.

10.           GLOBAL    \_io\_hlt          ;全局函数声明

11.

12.

13.    [SECTION.text]       ;段

14.

15.    \_io\_hlt:  ; void io\_hlt(void);

16.           HLT

 17.      RET

|  |
| --- |
|  |

要在naskfun.nas文件中声明所编写的汇编函数。

然后在C程序中调用在naskfunc.nas中编写的汇编函数：

**[cpp]** [view plain](http://blog.csdn.net/misskissC/article/details/45097363) [copy](http://blog.csdn.net/misskissC/article/details/45097363)

1. **void** io\_hlt(**void**);
3. **void** HariMain(**void**)
4. {
5. io\_hlt();
6. }

C编译器将C程序转换为汇编代码时，会在个标识符前面加下划线。

6 运行

此次文件跟“书”第3天的最后一个文件内容相同，打开“!cons\_nt.bat”，运行“makerun”，在QEMU的显示如下：



Figure5. 程序停留在C程序的中

此时，还未用STI指令开启中断。还在未运行中断做准备工作。

总结

此处导入C语言的过程跟工具“链接器”、“加载器”导入C语言的过程应该类似。