



第11章 程序设计的其他方法

一、学习内容

汇编语言多模块化程序设计

C程序和汇编语言程序的混合

内嵌汇编

模块程序设计中的注意事项

宏功能程序设计

目标：提高编程效率和质量，简化程序设计工作。





第11章 程序设计的其他方法

二、本章的学习重点

- (1) 模块程序设计的方法
- (2) 简单宏指令的定义与调用方式





11.1 多模块程序设计

1. 模块的划分与设计

按照功能划分，每个功能放在一个模块内，一个模块可以由一个人实现，程序放在一个文件内，独立调试通过后再联调

```
.....  
.stack 200  
.data  
BUF1 DB 'A'  
.code  
main proc c  
.....  
main endp  
END  
主模块main.asm
```

```
.686P  
....  
.data  
BUF3 DB 0DH  
.code  
QUEUE PROC NEAR  
.....  
QUEUE ENDP  
END  
排序子模块 queue.asm
```

2. 通讯方式



11.1 多模块程序设计——通信方式



华中科技大学

1. 公共符号与外部符号

外部符号:

在一个模块内访问而不在该模块内定义的符号。

语句格式:

EXTERN 符号:类型 [, 符号:类型]

例如:

```
EXTERN  AVG : WORD, COUNT : WORD  
EXTERN  SUB_P : NEAR
```



11.1 多模块程序设计

1. 公共符号与外部符号

公共符号：在一个模块中定义，其它模块要用到的符号。

PUBLIC 符号 [,符号]

```
public asm_avg
```

```
asm_avg proc num1:dword, num2:dword
```

```
.....
```

```
RET
```

```
asm_avg endp
```

```
extern "C" int asm_avg(int num1, int num2);
```

11.2 C程序和汇编语言程序的混合



华中科技大学

函数的申明和调用

变量的申明和调用

- 在汇编语言程序中，调用 C 库函数
- 在 C 语言程序中，调用汇编语言编写的函数



11.2 C程序和汇编语言程序的混合

.686P

.model flat, c

ExitProcess proto stdcall :dword

includelib kernel32.lib

printf proto c :vararg

includelib libcmtd.lib

includelib legacy_stdio_definitions.lib

...

- 在汇编程序中，函数原型说明伪指令proto。

11.2 C程序和汇编语言程序的混合



华中科技大学

编写一个程序，输入5个整型数据，对它们按从小到大的顺序排序，输出排序结果。

- 主程序实现数据的输入和输出，用C语言编写。
- 排序函数sort用汇编语言编写



11.2 C程序和汇编语言程序的混合



华中科技大学

主程序

mainp.c

```
#include <stdio.h>
#include <conio.h>
void sort (int *, int);
int main()
{
    int a[5], i;
    for (i = 0; i < 5; i++)
        scanf_s("%d", &a[i]);
    printf("\n result after sort \n");
    sort(a, 5);
    for (i = 0; i < 5; i++)
        printf("%d ", a[i]);
    _getch();
    return 0;
}
```



11.2 C程序和汇编语言程序的混合



华中科技大学

.686P

.model flat, c

.code

子程序

sort.asm

; sort : 对一个双字类型的数组按从小到大的顺序排序

; buf : 输入缓冲区的首地址, 也是排序结果存放的地址

; num : 元素的个数

sort proc buf:dword, num:dword

local outloop_num:dword

.if (num<2) ; 元素少于2个, 不用排序

ret

.endif



11.2 C程序和汇编语言程序的混合



华中科技大学

```
mov  eax, num
dec  eax
mov  outloop_num, eax ; 外循环的次数
mov  ebx, buf ; 数据缓冲区的首地址在 ebx中
mov  esi, 0 ; 外循环的控制指针
Out_Loop: ; 外循环
    cmp  esi, outloop_num
    jae  exit

    ; 下面是内循环
    lea  edi, [esi+1]
    Inner_Loop:
        cmp  edi, num
        jae  Inner_Loop_Over
```

51	32	55	48	22	11	20	15
----	----	----	----	----	----	----	----



11.2 C程序和汇编语言程序的混合



华中科技大学

```
mov    eax, [ebx][esi*4]
cmp    eax, [ebx][edi*4]
jle    Inner_Modify
xchg   eax, [ebx][edi*4]
mov    [ebx][esi*4], eax
```

Inner_Modify: ; 修改内循环的控制变量

```
inc    edi
jmp    Inner_Loop
```

51	32	55	48	22	11	20	15
----	----	----	----	----	----	----	----

Inner_Loop_Over:

32	51	55	48	22	11	20	15
----	----	----	----	----	----	----	----

```
inc    esi
jmp    Out_Loop
```

exit:

```
ret
sort endp
end
```



11.2 C程序和汇编语言程序的混合



华中科技大学

- 在C语言程序的**文件后缀名为cpp**时，编译时没有报错，但在链接时会报错，“无法解析的外部符号 `void _cdecl sort(int *,int) (? sort@@YAXPAHH@Z)`”。
- 编译器看到文件是cpp时，按照C Plus Plus (C++)的规范解析符号，会产生一个新的名称（换名机制）。
- 对于汇编语言程序在编译时保持了原有的名字，因而链接时出现了找不到符号的情况。
- 在C++程序中，使用**`extern "C"`**，说明按C语言的规则解析符号。

原说明: `void sort (int *, int);`

修改后: **`extern "C"`** `void sort (int *, int);`



11.2 C程序和汇编语言程序的混合



华中科技大学

函数名的大小写要一致

- 在C、C++程序设计中，函数名称是区分大小写的
- 在汇编语言程序中，默认状态下名称不区分大小写
- 为了让C语言程序能调用汇编语言写的函数，要求两者的函数命名一致。



11.2 C程序和汇编语言程序的混合



华中科技大学

语言类型声明要一致

- 在汇编语言程序中，优先采用函数定义伪指令proc中指定语言类型。
- 当proc中未指明语言类型时，使用模型说明伪指令.model中的语言类型。
- 函数定义与函数说明中的语言类型要相同

对于语言类型 C，说明为：

```
extern "C" void sort (int *, int);  
extern "C" void __cdecl sort(int *,int);
```

对于语言类型stdcall，说明为：

```
extern "C" void __stdcall sort(int *,int);
```



11.2 C程序和汇编语言程序的混合



华中科技大学

变量的申明和调用

- 在C程序中，按C语言的语法申明引用的外部全局变量；
- 在汇编语言程序中，按汇编语言的语法规则来写。

在 .c 文件中有:	<code>int x;</code>
	<code>extern int y;</code>
在 .cpp 文件中有:	<code>extern "C" int z;</code>
在汇编源程序中有:	<code>public y</code>
	<code>public z</code>
	<code>extern x:sdword</code>
	<code>y sdword 0</code>
	<code>z sdword 0</code>



11.3 内嵌汇编

__asm

{

汇编语言指令序列

}

__asm 汇编语言指令



11.3 内嵌汇编

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    int sum;
    sum=0;
    __asm {
        mov  eax, sum      ; eax 用来存放和
        mov  ebx, 1        ; ebx 为循环计算器
L1:   cmp   ebx, 100
        jg   L2
        add  eax, ebx
        inc  ebx
        jmp  L1
L2:   mov   sum, eax
    }
    printf("%d\n", sum);
    return 0;
}
```

计算从1累加
到100的和，
并且显示出和





华中科技大学

11.5 宏功能程序设计

1. 宏定义
2. 宏调用
3. 宏定义和宏调用中的参数
4. 宏指令与子程序的比较





11.5.1 宏定义

宏指令名 **MACRO** [形式参数 [, 形式参数]]
 宏体
 ENDM

例：将字类型数据 $(X) + (Y) \rightarrow Z$
 WORD_ADD MACRO X, Y, Z
 MOV AX, X
 ADD AX, Y
 MOV Z, AX
 ENDM

特别注意： ENDM前有什么？





11.5.1 宏定义

宏定义中注意的问题：

- (1) 宏段的结束处，没有宏指令名
- (2) 形参可有可无，有多个时，之间以逗号分隔
- (3) ENDM与MACRO必须成对出现
- (4) 宏名字可以与其它变量、标号、保留字同名，
汇编程序在处理时，**宏名字优先级最高**，利用
这一特点，可设计新的指令系统。
- (5) 宏指令在使用之前要先定义，与子程序可写在
调用指令后不同。



11.5.2 宏调用

调用格式：宏指令名 [实在参数[, 实在参数]]

- (1) 宏指令名要与原宏定义的名字一致;
- (2) 实参与形参应按位置关系一一对应:
 - a. 实参个数多于形参, 多余实参被忽略;
 - b. 实参个数小于形参, 缺少的实参被处理为空白 (没有字符)。



11.5.2 宏调用

```
WORD_ADD    MACRO    X, Y, Z
              MOV     AX,  X
              ADD     AX,  Y
              MOV     Z,   AX
              ENDM
```

```
BUF1  DW    10, 30, 0
BUF2  DW    20, 40, 0
```

.....

```
WORD_ADD    BUF1, BUF1+2, BUF1+4
```

.....

```
WORD_ADD    BUF2, BUF2+2, BUF2+4
```



11.5.2 宏调用

宏调用经汇编程序扩展后的形式

```
WORD_ADD  BUF1, BUF1+2, BUF1+4  
+  MOV     AX,  BUF1  
+  ADD     AX,  BUF1+2  
+  MOV     BUF1+4, AX
```

.....

```
WORD_ADD  BUF2, BUF2+2, BUF2+4  
+  MOV     AX,  BUF2  
+  ADD     AX,  BUF2+2  
+  MOV     BUF2+4, AX
```


11.5.2 宏调用

printf 函数会改变一些寄存器的值，
写一个不改变寄存器的宏，方便使用

```
PRINT_MYSELF MACRO A, B  
    PUSHAD  
    invoke printf, A, B  
    POPAD  
ENDM
```

```
.data  
    x dd -3  
    y dd 5  
    outfmt db '%d ', 0dh, 0ah, 0  
  
PRINT_MYSELF offset outfmt, x
```

11.5.2 宏调用

```
.data  
x dd -3  
y dd 5  
outfmt db '%d %d', 0dh,0ah,0
```

invoke printf, offset outfmt, x, y

参数个数不定

```
PRINT_MYSELF MACRO A  
    PUSHAD  
    invoke printf, A  
    POPAD  
ENDM
```

带间隔符的参数，用<...>

```
PRINT_MYSELF <offset outfmt, x, y>
```



11.5.2 宏调用

内存 1 寄存器

对象浏览器反汇编c_main.cppasm_fun.asmMAIN.C

```
PRINT_MYSELF MACRO A
    PUSHAD
    invoke printf, A
    POPAD
ENDM

my_printf proc
    PRINT_MYSELF <offset outfmt, X,Y>
    ret
my_printf endp
```

H:\教学\汇编程序示例\

-3 5



华中科技大学

11.5.3 宏指令与子程序的比较

- 处理时间不同
- 处理方式不同
- 目标程序的长度不同
- 执行速度不同
- 参数传递方式不同

