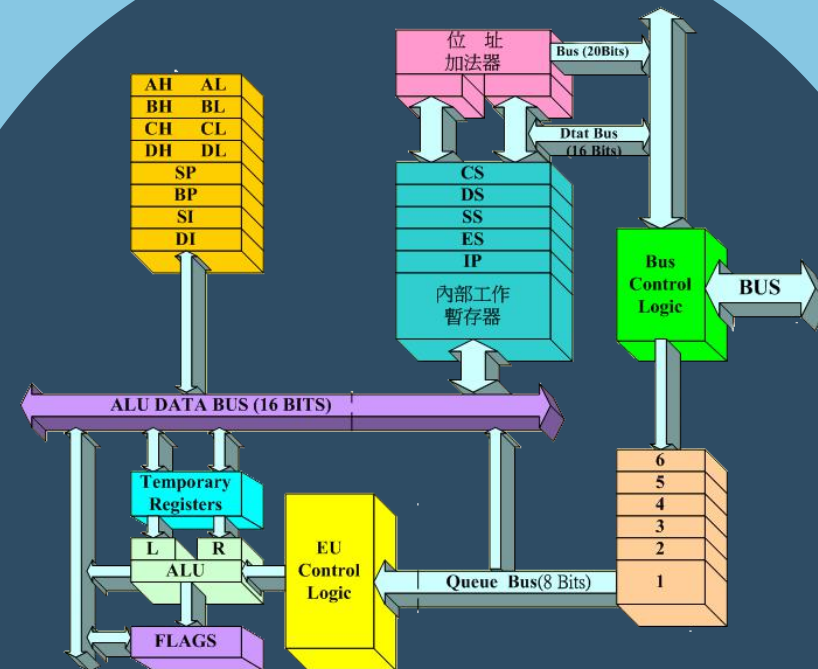


宏汇编

贺利坚 主讲

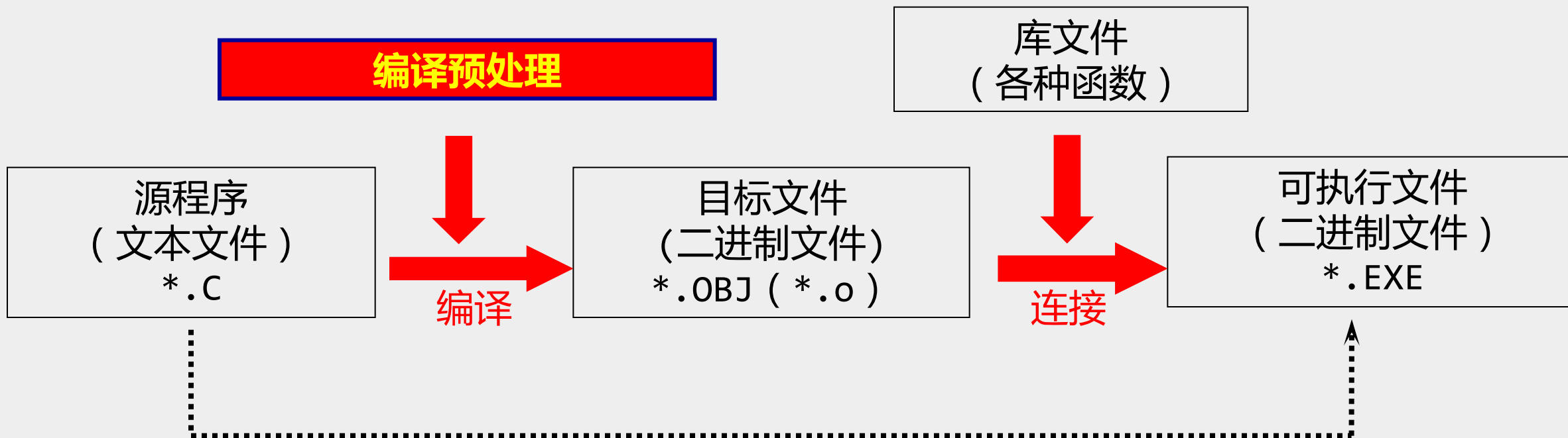


汇编语言程序设计
Assembly Language

C语言中的预处理命令

💻 C语言中以符号"#"开头的命令

- (1) 宏定义: #define ...
- (2) 文件包含: #include ...
- (3) 条件编译: #ifdef ...



预处理含义:

- 📁 在对程序进行编译之前，根据预处理命令对程序作相应处理；
- 📁 经过预处理后编译程序才可以对程序进行编译等处理，得到可供执行的目标代码。

汇编中的宏——由C中的宏定义说起



宏

📄 源程序中一段有独立功能的程序代码。



宏指令

📄 用户自定义的指令。

📄 在编程时，将多次使用的功能用一条宏指令来代替。

汇编语言指令 { 指令
伪指令（伪操作）
宏指令

```
#include <stdio.h>
#define S(a,b) a*b
int main()
{
    printf("面积1: %d\n", S(2, 4));
    printf("面积2: %.2f\n", S(2.3, 4.5));
    return 0;
}
```

例:使用宏的程序

```
assume cs:code
data segment
    n1 dw 1
data ends
code segment
mproc macro p1, p2, p3
```

```
    mov ax,p1
    mov cx,p2
    mov dx,p3
endm
```

```
code ends
end start
```

先定义,
后调用

```
start:mov ax,data
      mov ds,ax
      lea bx, n1
      mproc [bx], bx, 100h
      mproc 1, n1, cx
      mov ax,4c00h
      int 21h
```

```
assume cs:code
data segment
    n1 dw 1
data ends
code segment
mproc macro p1, p2, p3
    mov ax,p1
    mov cx,p2
    mov dx,p3
endm
```

```
code ends
end start
```

宏展开

```
start:mov ax,data
      mov ds,ax
      lea bx, n1
      mov ax,[bx]
      mov cx,bx
      mov dx,100h
      mov ax,1
      mov cx,n2
      mov dx,cx
      mov ax,4c00h
      int 21h
```

语法与求语


宏定义

```
macro_name MACRO [哑元表] ; 形参/虚参  
.....  
..... ; 宏定义体  
ENDM
```

宏调用（必须先定义后调用）

```
macro_name [实元表] ; 实参
```

宏展开：汇编程序把宏调用展开

 将宏定义体复制到宏指令位置，
实参代替虚参

```
assume cs:code
```

```
data segment
```

```
    n1 dw 1
```

```
data ends
```

```
code segment
```

```
mproc macro p1, p2, p3
```

```
    mov ax,p1
```

```
    mov cx,p2
```

```
    mov dx,p3
```

```
endm
```

```
start:mov ax,data
```

```
    mov ds,ax
```

```
    lea bx, n1
```

```
mproc [bx], bx, 100h
```

```
mproc 1, n1, cx
```

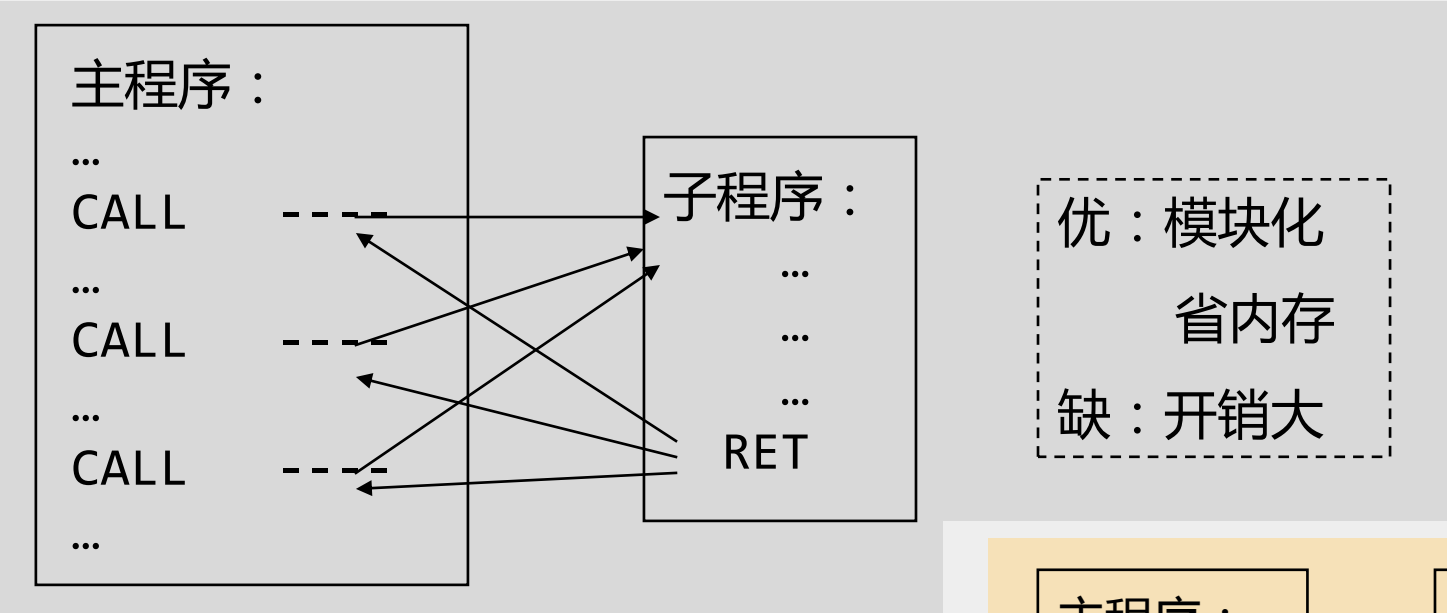
```
    mov ax,4c00h
```

```
    int 21h
```

```
code ends
```

```
end start
```

子程序 vs 宏定义



宏定义：

```
Q macro x,y
```

```
...
```

```
endm
```

主程序：

```
...
```

```
Q a, b
```

```
...
```

```
Q c, d
```

```
...
```

```
Q e, f
```

```
...
```

宏展开的程序：

```
...
```

```
...
```

```
...
```

```
...
```

```
...
```

```
...
```

```
...
```

优：

参数传送简单
执行效率高

缺：

代码占用内存
空间大

例：宏的应用

宏定义：

savereg **MACRO**

```
push ax
push bx
push cx
push dx
push si
push di
ENDM
```

宏调用：
savereg

宏展开：

```
push ax
push bx
push cx
push dx
push si
push di
```

宏定义：

multiply **MACRO** opr1,opr2,result

```
push dx
push ax
mov ax,opr1
imul opr2
mov result,ax
pop ax
pop dx
ENDM
```

宏调用：

multiply cx,var,xyz[bx]

宏展开：

```
push dx
push ax
mov ax,cx
imul var
mov xyz[bx],ax
pop ax
pop dx
```

宏中的局部标号——以求绝对值为例

宏定义：

```
absol MACRO oper
      LOCAL next
      cmp oper,0
      jge next
      neg oper
```

next:

ENDM

宏调用：

.....

absol var

.....

absol bx

.....

宏展开：

.....

cmp var,0

jge ??0000

neg var

??0000:

.....

.....

cmp bx,0

jge ??0001

neg bx

??0001:

.....

变元是操作码中一部分

宏定义：

```
leap  macro   cond,lab  
      j&cond  lab  
      endm
```

宏调用：

```
leap  z, there  
.....  
leap nz, here
```

宏展开：

```
  jz   there  
  .....  
  jnz  here
```