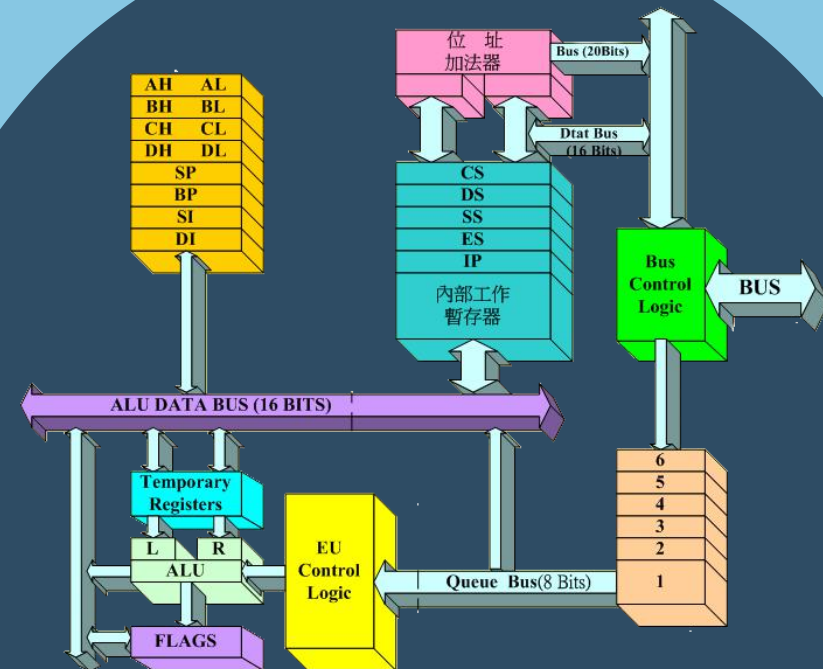


jmp指令

贺利坚 主讲



汇编语言程序设计
Assembly Language

jmp指令——无条件转移

jmp指令的功能

 无条件转移，可以只修改IP，也可以同时修改CS和IP

jmp指令要给出两种信息：

 转移的目的地址

 转移的距离

- 段间转移（远转移）：`jmp 2000:1000`
- 段内短转移：`jmp short 标号`；IP的修改范围为 -128~127，8位的位移
- 段内近转移：`jmp near ptr 标号`；IP的修改范围为 -32768~32767，16位的位移



jmp指令：依据位移进行转移

```
-a 073f:0100
073F:0100 mov ax, 0123
073F:0103 mov ax, [0123]
073F:0106 push [0123]
073F:010A
-u 073f:0100
073F:0100 B82301      MOV     AX,0123
073F:0103 A12301      MOV     AX,[0123]
073F:0106 FF362301  PUSH    [0123]
```

```
1  assume cs:codesg
2  codesg segment
3  start: mov ax,0
4          jmp short s
5          add ax,1
6          s: inc ax
7  codesg ends
8  end start
```

```
C:\>debug p9-2.exe
-u
076f:0000 B80000      MOV     AX,0000
076f:0003 EB03      JMP     0008
076f:0005 050100      ADD     AX,0001
076f:0008 40          INC     AX
076f:000B 40          INC     AX
```

引子：常见指令中的立即数均在机器指令中有体现

问题：jmp short 指令中，转移到了哪里？

👉 jmp short 的机器指令中，包含的是跳转到指令的相对位置，而不是转移的目标地址。

```
1  assume cs:codesg
2  codesg segment
3  start: mov ax,0
4          jmp short s
5          add ax,1
6          nop
7          nop
8          s: inc ax
9  codesg ends
10 end start
```

👉 左边程序jmp short s指令的读取和执行：

(1) (IP)=0003，CS:IP指向EB 05(jmp 的机器码)

(2) 读取指令码EB 05进入指令缓冲器；

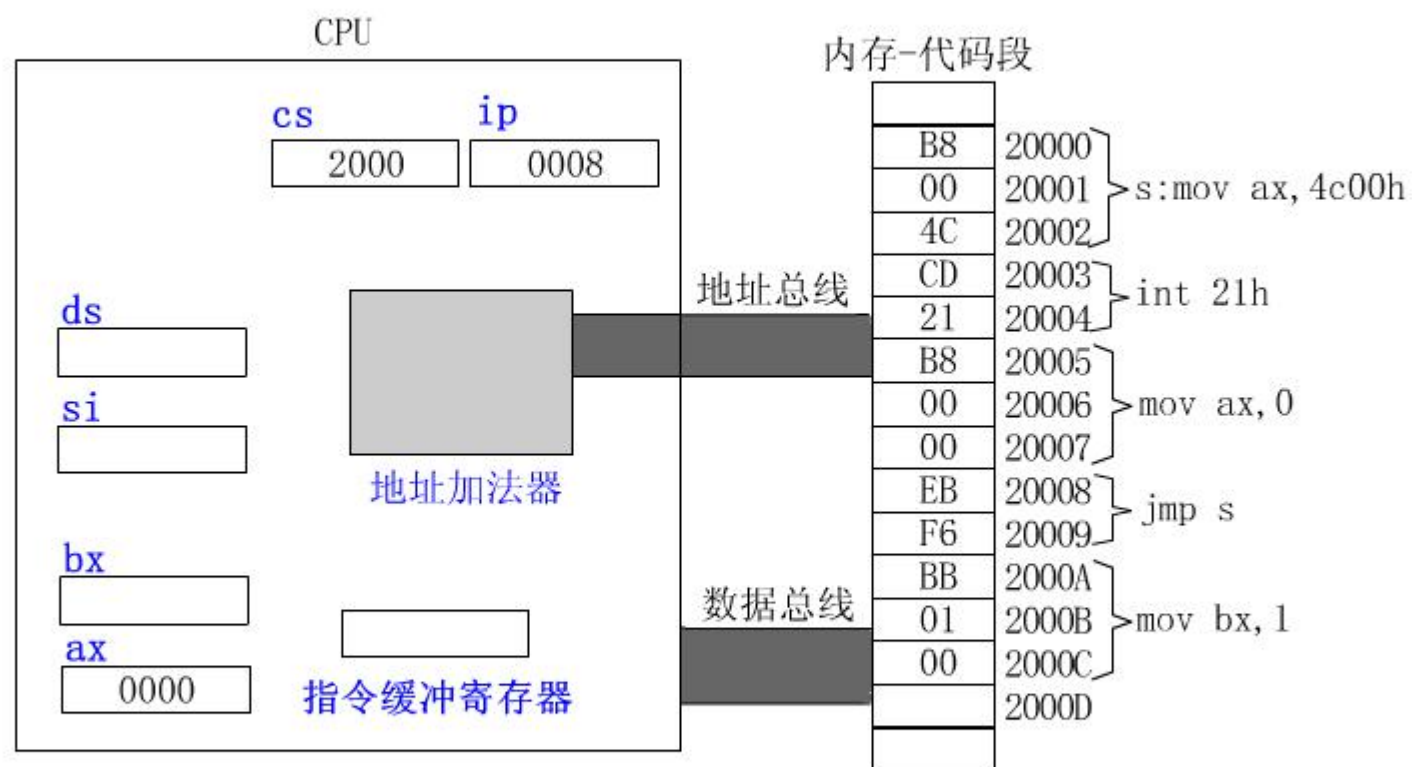
(3) (IP)=(IP)+所读取指令的长度
=(IP)+2=0005，CS:IP指向add ax, 0001；

(4) CPU执行指令缓冲器中的指令EB05；

(5) 指令EB 05执行后，
(IP)=(IP)+05=000AH，CS:IP指向inc ax

```
C:\>debug p9-3.exe
-u
076a:0000 B80000      MOV     AX,0000
076a:0003 EB05      JMP     000A
076a:0005 050100      ADD     AX,0001
076a:0008 90          NOP
076a:0009 90          NOP
076a:000A 40          INC     AX
076a:000B 40          INC     AX
```

依据位移进行转移的 jmp 指令



play



step




step



stop

两种段内转移

 **短转移**：“jmp short 标号”


 功能： $(IP) = (IP) + 8\text{位位移}$

 原理

- (1) 8位位移=“标号”处的地址-jmp指令后的第一个字节的地址；
- (2) short指明此处的位移为8位位移；
- (3) 8位位移的范围为-128~127，用**补码**表示；
- (4) 8位位移由编译程序在编译时算出。

```
1  assume cs:codesg
2  codesg segment
3  start: mov ax,0
4          jmp short s
5          add ax,1
6  s: inc ax
7  codesg ends
8  end start
```

 **近转移**：指令“jmp near ptr 标号”

 功能： $(IP) = (IP) + 16\text{位位移}$

 原理

- (1) 16位位移=“标号”处的地址-jmp指令后的第一个字节的地址；
- (2) near ptr指明此处的位移为16位位移，进行的是段内近转移；
- (3) 16位位移的范围为-32769~32767，用补码表示；
- (4) 16位位移由编译程序在编译时算出。



我要这样！

```
P9-error.asm - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
assume cs:codesg
codesg segment
start: jmp short s
      db 128 dup (0)
      s: mov ax,0ffffh
codesg ends
end start
```

```
C:\>masm p9-error;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

p9-error.ASM(3): error A2053: Jump out of range by 1 byte(s)

51738 + 464806 Bytes symbol space free

0 Warning Errors
1 Severe Errors
```

转移位移超界!

远转移： jmp far ptr 标号

远转移jmp far ptr 标号

段间转移

far ptr指明了跳转到的**目的地址**，即包含了标号的段地址CS和偏移地址IP。

```
1  assume cs:codesg
2  codesg segment
3  start: mov ax,0
4          mov bx,0
5          jmp far ptr s
6          db 256 dup (0)
7  s: add ax,1
8      inc ax
9  codesg ends
10 end start
```

C:\>debug p9-5.exe

-u

076A:0000	B80000	MOV	AX,0000
076A:0003	BB0000	MOV	BX,0000
076A:0006	EA0B016A07	JMP	076A:010B
076A:000B	0000	ADD	[BX+SI],AL
076A:000D	0000	ADD	[BX+SI],AL

-u 109

076A:0109	0000	ADD	[BX+SI],AL
076A:010B	050100	ADD	AX,0001
076A:010E	40	INC	AX
076A:010F	7DFF	JGE	0110
076A:0111	50	PUSH	AX

近转移jmp near ptr 标号

段内转移

near ptr 指明了相对于当前IP的**转移位移**，而不是转移的目的地址。

```
1  assume cs:codesg
2  codesg segment
3  start: mov ax,0
4          mov bx,0
5          jmp near ptr s
6          db 256 dup (0)
7  s: add ax,1
8      inc ax
9  codesg ends
10 end start
```

C:\>debug p9-4.exe

-u

076A:0000	B80000	MOV	AX,0000
076A:0003	BB0000	MOV	BX,0000
076A:0006	E90001	JMP	0109
076A:0009	0000	ADD	[BX+SI],AL
076A:000B	0000	ADD	[BX+SI],AL

-u 109

076A:0109	050100	ADD	AX,0001
076A:010C	40	INC	AX
076A:010D	8D867DFF	LEA	AX,[BP+FF7D]
076A:0111	50	PUSH	AX

转移地址在寄存器中的jmp指令

🖥️ 指令格式：jmp 16位寄存器

📁 功能：IP = (16位寄存器)

📁 举例：

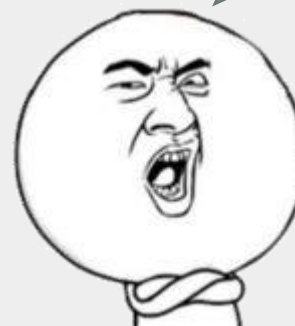
jmp ax

jmp bx

```
1  assume cs:codesg
2  codesg segment
3  start: mov ax,0
4          mov bx,ax
5          jmp bx
6          mov ax,0123H
7  codesg ends
8  end start
```

```
C:\>debug p9-6.exe
-u
076A:0000 B80000  MOV  AX,0000
076A:0003 8BD8     MOV  BX,AX
076A:0005 FFE3     JMP  BX
076A:0007 B82301  MOV  AX,0123
076A:000A 07      POP  ES
```

跳到哪儿由变量定，
就这样自在！



转移地址在内存中的jmp指令

jmp word ptr 内存单元地址	jmp dword ptr 内存单元地址
段内转移	段间转移
功能：从内存单元地址处开始存放着 一个字 ，是转移的目的 偏移地址 。	功能：从内存单元地址处开始存放着 两个字 ，高地址处的字是转移的 目的段地址 ，低地址处是转移的目的 偏移地址 。 <div><div>00</div><div>(IP)</div><div>02</div><div>(CS)</div></div>
<div><div>mov ax,0123H</div><div>mov ds:[0],ax</div><div>jmp word ptr ds:[0]</div><div>执行后，(IP)=0123H</div></div> <div><div>mov ax,0123H</div><div>mov [bx],ax</div><div>jmp word ptr [bx]</div><div>执行后，(IP)=0123H</div></div>	<div><div>mov ax,0123H</div><div>mov ds:[0],ax</div><div>mov word ptr ds:[2],0</div><div>jmp dword ptr ds:[0]</div><div>执行后，</div><div>(CS)=0</div><div>(IP)=0123H</div><div>CS:IP指向0000:0123</div></div> <div><div>mov ax,0123H</div><div>mov [bx],ax</div><div>mov word ptr [bx+2],0</div><div>jmp dword ptr [bx]</div><div>执行后，</div><div>(CS)=0</div><div>(IP)=0123H</div><div>CS:IP指向0000:0123</div></div>

jmp指令小结

jmp指令格式	示例
jmp 标号	<ul style="list-style-type: none">– 段间转移（远转移）： jmp far ptr 标号– 段内短转移： jmp short 标号 ; 8位的位移– 段内近转移： jmp near ptr 标号 ; 16位的位移
jmp 寄存器	<ul style="list-style-type: none">– jmp bx ; 16位的位移
jmp 内存单元(表示跳转到的地址)	<ul style="list-style-type: none">– 段内转移： jmp word ptr 内存单元地址 ; jmp word ptr [bx]– 段间转移： jmp dword ptr 内存单元地址 ; jmp dword ptr [bx]

在源程序中，不允许使用“jmp 2000:0100”的转移指令实现段间转移

- 这是在 Debug 中使用的汇编指令，汇编编译器并不认识
- 如果在源程序中使用，编译时也会报错。