X86组合语言/X86架构及寄存器解释

维基教科书，自由的教学读本
< X86組合語言

目录

X86架构

x86架构有8个通用寄存器（GPR）、6段寄存器、1个标志寄存器和指令指针。 64位的x86有附加的寄存器。

通用寄存器（GPR） – 32位命名约定

8个GPR是：

1. 累加器寄存器（AX）。用在算术运算。
2. 基址寄存器（BX）。作为一个指向数据的指针（在分段模式下，位于段寄存器DS）。
3. 计数器寄存器（CX）。用于移位/循环指令和循环。
4. 数据寄存器（DX）。用在算术运算和I/O操作。
5. 堆栈指针寄存器（SP）。用于指向堆栈的顶部。
6. 栈基址指针寄存器（BP）。用于指向堆栈的底部。
7. 源变址寄存器（SI）。在流操作中用作源的一个指针。
8. 目标索引寄存器（DI）。用作在流操作中指向目标的指针。

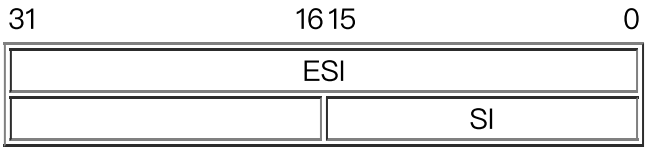
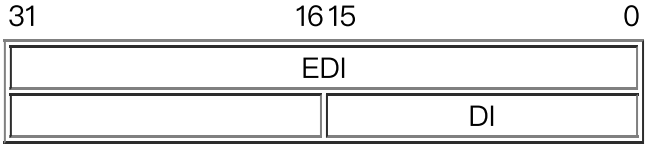
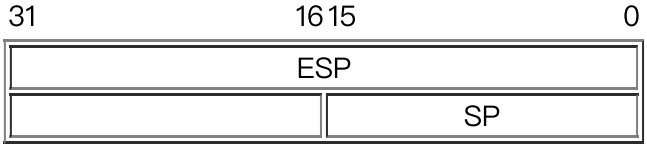
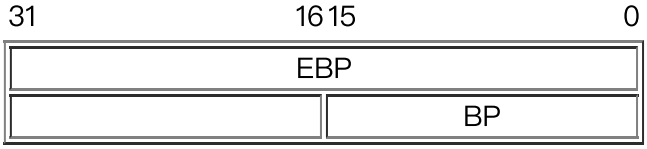
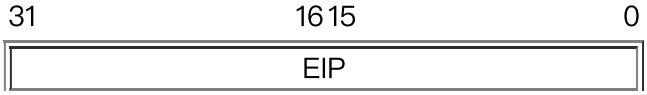
将它们以这样的顺序列出是有原因的：这个顺序和堆栈操作中推入栈中的次序相同，我们以后会讲到。

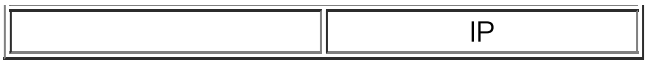
所有寄存器都可以在16位和32位模式下被访问。在16位模式下，通过上面的列表中两个字母的缩写来确定该寄存器。在32位模式下，这两个字母的缩写名字前有“E”（*extended*，延伸）。例如，“EAX'是累加器寄存器作为一个32位的值。

类似地，在64位的版本，“E”被替换为“R”，所以在64位版本“EAX'被称为'RAX'。

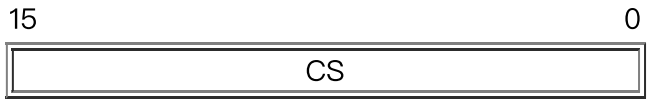
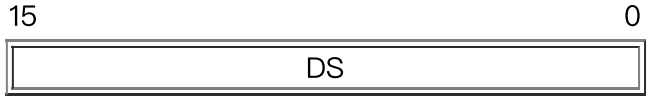
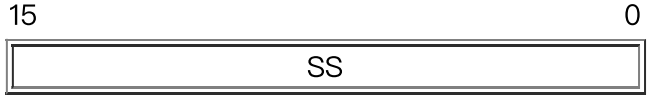
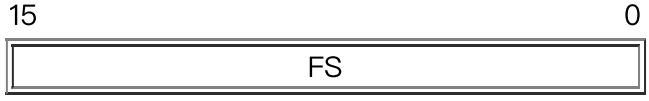
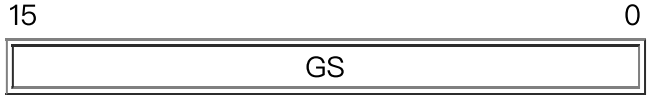
寄存器	累加器	计数器	数据	基址	堆栈指针	栈基址指针	源变址	目标索引
64-bit	RAX	RCX	RDX	RBX	RSP	RBP	RSI	RDI
32-bit	EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
16-bit	AX	CX	DX	BX	SP	BP	SI	DI
8-bit	AH AL	CH CL	DH DL	BH BL				

指标暂存器

暂存器别名	暂存器	预设的区段暂存器	暂存器示意图
来源索引暂存器	ESI = 32 bits SI = 16 bits		
目的索引暂存器	EDI = 32 bits DI = 16 bits		
堆叠指标暂存器	ESP = 32 bits SP = 16 bits	SS	
基底指标暂存器	EBP = 32 bits BP = 16 bits		
程式指标暂存器	EIP = 32 bits IP = 16 bits	CS	



区段暂存器

暂存器别名	暂存器	暂存器示意图
程式区段暂存器	CS = 16 bits	
资料区段暂存器	DS = 16 bits	
堆叠区段暂存器	SS = 16 bits	
额外区段暂存器	ES = 16 bits	
额外区段暂存器 (80386以后出现)	FS = 16 bits	
额外区段暂存器 (80386以后出现)	GS = 16 bits	

- 在真实模式或虚拟86模式时，‘区段暂存器’用来扩展定址的范围由 64 KByte 到 1 MByte 。
- 在保护模式时，‘区段暂存器’将变成‘选择子暂存器’，用为对于程序与记忆体存取权限的控管的索引。

大多现代操作系统（如FreeBSD、Linux或Windows）的大多应用程序使用将几乎所有段寄存器都指向同一位置（并使用页面），从而便捷地停用这些寄存器。

EFLAGS寄存器

EFLAGS寄存器，在台湾也称为“旗标暂存器”，是用一系列表示布尔值的位来存储操作的结果和处理器状态的32位寄存器。

这些位的名称是：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	ID	VIP	VIF	AC	VM	RF
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

- AF：辅助进位旗标
- CF：进位旗标
- OF：溢位旗标
- SF：符号(负号)旗标
- PF：奇偶旗标
- ZF：零值旗标
- DF：方向旗标
- IF：中断旗标
- TF：单步旗标

指令指针

如果没有创建过分支，EIP寄存器包含下一条将要执行的指令的地址。

EIP只能在一个`call`指令后从堆栈读出。

内存中字的存储

x86架构是小端序的，即多字节数值的最低位字节首先写入。(这只针对字节的排序，不对于位。)

所以x86上的32位数值$B_3B_2B_1B_0{}_{16}$在内存中会表示为：

小端序表示法

B0	B1	B2	B3

例如，32位的双字0x1BA583D4 (**0x**表示十六进制)在内存中写为：

小端序例子

D4	83	A5	1B

执行内存转储时会把其视为`0xD4 0x83 0xA5 0x1B`。

二进制补码表示

二进制补码是以二进制表示负整数的标准方式。符号是由对所有的位取反并加一来改变的。

二进制补码例子

开始:	0001
取反:	1110
加一:	1111

0001表示十进制的1

1111表示十进制的-1

寻址方式

寻址方式给出了操作数给出的方式。

寄存器寻址

（操作数地址R所在地址字段）

```
mov ax, bx  ; moves contents of register bx into ax
```

Immediate

（实际地址所在字段）

```
mov ax, 1   ; moves value of 1 into register ax
```

或

```
mov ax, 010Ch ; moves value of 0x010C into register ax
```

直接存储器寻址

（操作数地址所在地址字段）

```
.data
my_var dw 0abcdh ; my_var = 0xabcd
.code
mov ax, my_var ; copy my_var content in ax (ax=0xabcd)
mov ax, [my_var] ; idem as above
```

直接偏移寻址

（通过算数改变地址）

```
byte_tbl db 12,15,16,22,..... ; Table of bytes
mov al,[byte_tbl+2]
mov al,byte_tbl[2] ; same as the former
```

寄存器间接寻址

（field points to a register that contains the operand address）

```
mov ax,[di]
```

The registers used for indirect addressing are BX, BP, SI, DI

基址

```
mov ax,[bx + di]
```

例如，如果我们是在讨论一个数组，BX包含数组的起始地址和DI包含数组的索引。

基址变址：

```
mov ax,[bx + di + 10]
```

栈

栈是一个后进先出(LIFO)的数据结构，数据被压到栈顶并会以逆序出栈。

```
mov ax, 006Ah
mov bx, F79Ah
mov cx, 1124h
push ax
```

你将AX中的数值压入栈顶， 它现在的数值为$006A。

```
push bx
```

你对BX中的数值做相同的操作，现在栈中有$006A和$F79A。

```
push cx
```

如今栈中有$006A、$F79A和$1124。

```
call do_stuff
```

Do some stuff. The function is not forced to save the registers it uses, hence us saving them.

```
pop cx
```

取出最后一个压入栈中的元素到CX，即$1124；现在栈中有$006A和$F79A.

```
pop bx
```

取出最后一个压入栈中的元素到BX，即$F79A；现在栈中只有$006A。

```
pop ax
```

取出最后一个压入栈中的元素到AX，即$006A；现在栈是空的。

栈通常用来向函数或过程传递参数，并用来记录`call`指令使用时的控制流。栈的另一个常见的用途是临时存储寄存器。

CPU的工作模式

真实模式 (Real mode)

- 参照: 真实模式 (维基百科)
- 特性:
  - 记忆体定址限制: 1 MByte (每一个记忆体区段限定 64 Kbyte, 使用区段暂存器扩展可存取的记忆体范围到 1 MByte)
  - 暂存器与资料操作宽度: 16 bits 与 8 bits

保护模式 (Protected Mode)

- 参照: 保护模式 (http://zh.wikipedia.org/wiki/%E4%BF%9D%E8%AD%B7%E6%A8%A1%E5%BC%8F) (维基百科)

16位元保护模式 (16-bit protected mode)

- 特性:
  - 记忆体定址限制: 16 MByte
  - 暂存器与资料操作宽度: 16 bits 与 8 bits

32位元保护模式 (32-bit protected mode)

- 特性:
  - 记忆体定址限制: 4 GByte
  - 暂存器与资料操作宽度: 32 bits 与 16 bits 与 8 bits

虚拟86模式