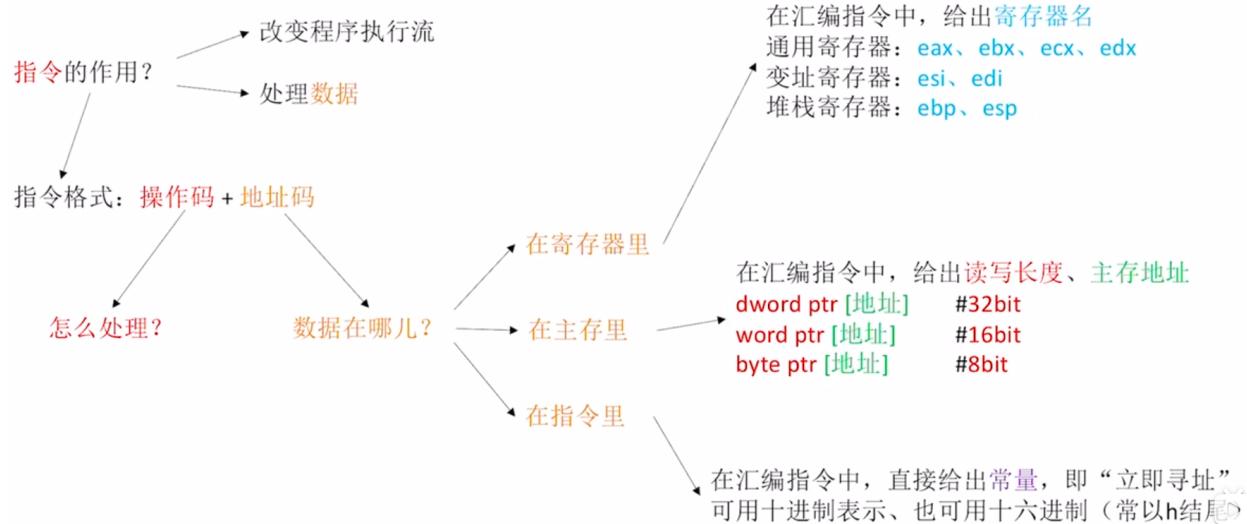
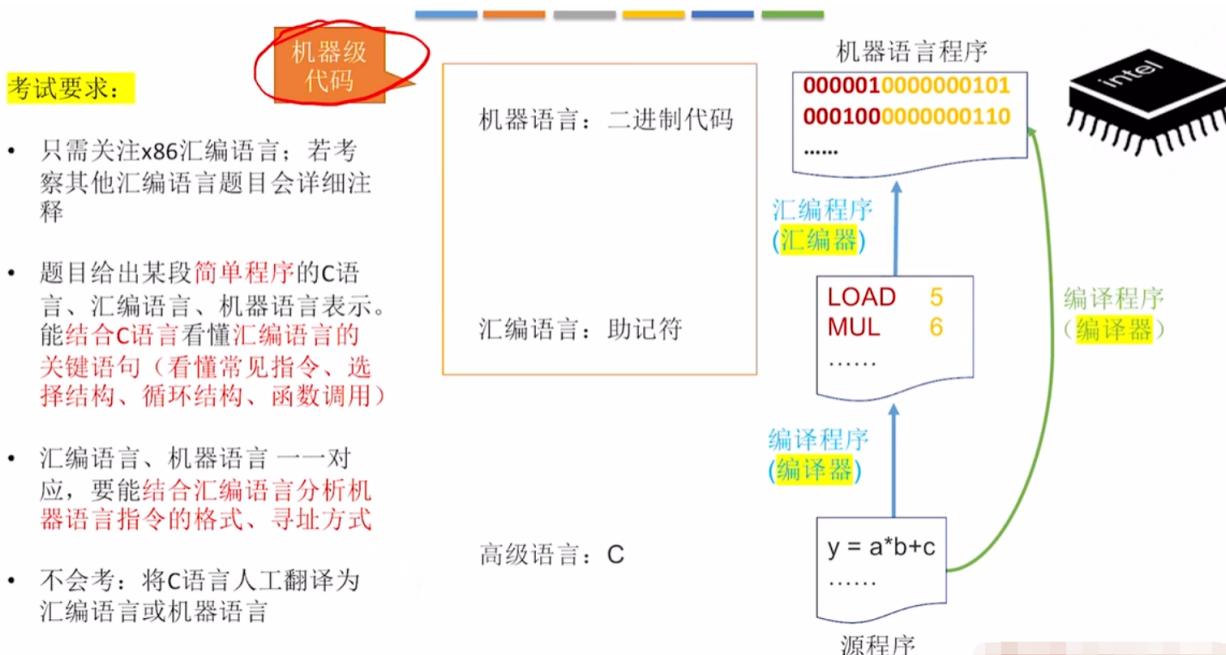


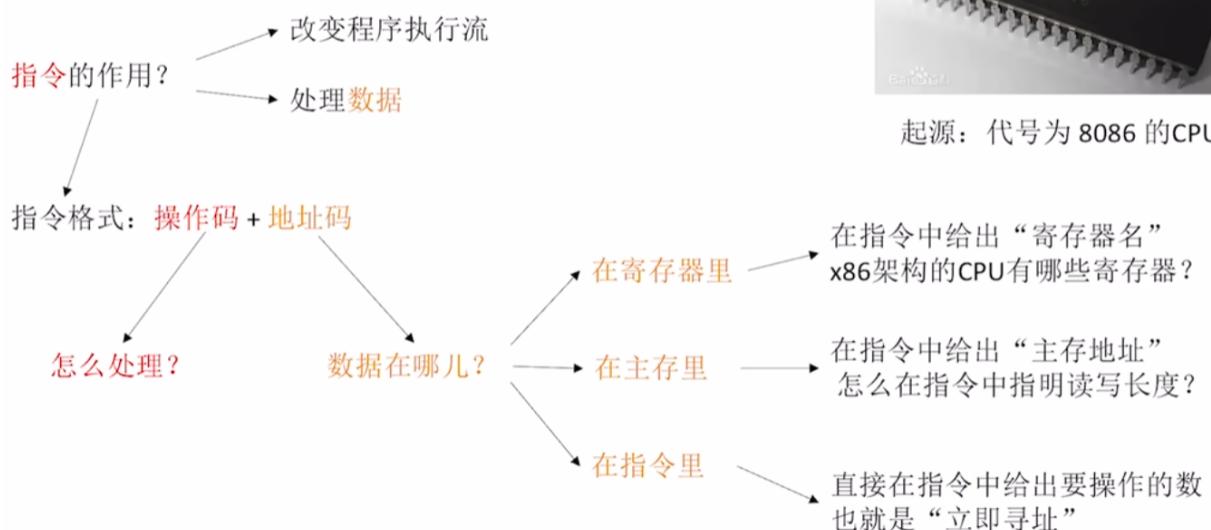
常用汇编指令介绍



高级语言->汇编语言->机器语言



x86 汇编语言指令基础



起源：代号为 8086 的CPU

以 mov 指令为例

mov 目的操作数 d, 源操作数 s

#mov指令功能：将源操作数 s 复制到目的操作数 d 所指的位置

destination: 目的地

source: 来源、发源地

`mov eax, ebx` #将寄存器 ebx 的值复制到寄存器 eax

`mov eax, 5` #将立即数 5 复制到寄存器 eax

`mov eax, dword ptr [af996h]` #将内存地址 af996h 所指的32bit值复制到寄存器 eax

`mov byte ptr [af996h], 5` #将立即数 5 复制到内存地址 af996h 所指的一字节中

如何指明内存的读写长度：

dword ptr——双字，32bit

word ptr——单字，16bit

byte ptr——字节，8bit

十六进制

adj. **hexadecimal** ;

x86架构CPU，有哪些寄存器？

每个寄存器都是32bit

31	0
EAX	
EBX	
ECX	
EDX	
ESI	
EDI	
EBP	
ESP	

通用寄存器 (X = 未知)
E = Extended = 32bit

mov eax, ebx #寄存器→寄存器
mov eax, dword ptr [af996h] #主存→寄存器
mov eax, 5 #立即数→寄存器

变址寄存器 (I = Index)
S = Source, D=Destination

变址寄存器可用于线性表、字符串的处理

堆栈基指针 (Base Pointer)

堆栈寄存器用于实现函数调用

堆栈顶指针 (Stack Pointer)

每个寄存器都是32bit

31	0
AX (低16bit)	
BX (低16bit)	
CX (低16bit)	
DX (低16bit)	
ESI	
EDI	
EBP	
ESP	

通用寄存器，使用低16bit

mov ax, bx #寄存器→寄存器
mov ax, word ptr [af996h] #主存→寄存器
mov ax, 5 #立即数→寄存器

变址寄存器 (I = Index)
S = Source, D=Destination

两个变址寄存器只能固定使用32bit

堆栈基指针 (Base Pointer)

两个堆栈寄存器只能固定使用32bit

堆栈顶指针 (Stack Pointer)

每个寄存器都是32bit

31	0
AX (低16bit)	
BX (低16bit)	
CX (低16bit)	
DX (低16bit)	
ESI	
EDI	
EBP	
ESP	

通用寄存器，使用低16bit

mov ax, bx #寄存器→寄存器
mov ax, word ptr [af996h] #主存→寄存器
mov ax, 5 #立即数→寄存器

变址寄存器 (I = Index)
S = Source, D=Destination

两个变址寄存器只能固定使用32bit

堆栈基指针 (Base Pointer)

两个堆栈寄存器只能固定使用32bit

堆栈顶指针 (Stack Pointer)

mov eax, dword ptr [ebx]	#将 ebx 所指主存地址的 32bit 复制到 eax 寄存器中
mov dword ptr [ebx], eax	#将 eax 的内容复制到 ebx 所指主存地址的 32bit
mov eax, byte ptr [ebx]	#将 ebx 所指的主存地址的 8bit 复制到 eax
mov eax, [ebx]	#若未指明主存读写长度, 默认 32 bit
mov [af996h], eax	#将 eax 的内容复制到 af996h 所指的地址 (未指明长度默认32bit)
mov eax, dword ptr [ebx+8]	#将 ebx+8 所指主存地址的 32bit 复制到 eax 寄存器中
mov eax, dword ptr [af996-12h]	#将 af996-12 所指主存地址的 32bit 复制到 eax 寄存器中

常用的x86汇编指令

常见的算术运算指令

功能	英文	汇编指令	注释	目的地 d 不可以是常量
加	add	add d,s	#计算d+s, 结果存入d	destination: 目的地 (d 目的操作数) source: 来源地 (s 源操作数)
减	subtract	sub d,s	#计算d-s, 结果存入d	
乘	multiply	mul d,s imul d,s	#无符号数d*s, 乘积存入d #有符号数d*s, 乘积存入d	
除	divide	div s idiv s	#无符号数除法 edx:eax/s, 商存入eax, 余数存入edx #有符号数除法 edx:eax/s, 商存入eax, 余数存入edx	
取负数	negative	neg d	#将d取负数, 结果存入d	
自增++	increase	inc d	#将d++, 结果存入d	
自减--	decrease	dec d	#将d--, 结果存入d	

1) **add/sub 指令。** add 指令将两个操作数相加, 相加的结果保存到第一个操作数中。sub 指令用于两个操作数相减, 相减的结果保存到第一个操作数中。

它们的语法如下:

```
add <reg>,<reg> / sub <reg>,<reg>
add <reg>,<mem> / sub <reg>,<mem>
add <mem>,<reg> / sub <mem>,<reg>
add <reg>,<con> / sub <reg>,<con>
add <mem>,<con> / sub <mem>,<con>
```



没有.....你听我解释

举例:

```
sub eax, 10          #eax ← eax-10
add byte ptr [var], 10 #10 与 var 值指示的内存地址的一字节值相加, 并将结果
                      #保存在 var 值指示的内存地址的字节中
```

<reg> **寄存器**

<mem> **内存**

<con> **常数**

n. register

n. memory

n. constant

常见的逻辑运算指令

功能	英文	汇编指令	注释
与	and	and d,s	#将 d、s 逐位相与，结果放回d
或	or	or d,s	#将 d、s 逐位相或，结果放回d
非	not	not d	#将 d 逐位取反，结果放回d
异或	exclusive or	xor d,s	#将 d、s 逐位异或，结果放回d
左移	shift left	shl d,s	#将d逻辑左移s位，结果放回d（通常s是常量）
右移	shift right	shr d,s	#将d逻辑右移s位，结果放回d（通常s是常量）

其他指令

用于实现分支结构、循环结构的指令：cmp、test、jmp、jxxx

用于实现函数调用的指令：push、pop、call、ret

用于实现数据转移的指令：mov

AT&T格式v.s Intel格式

AT&T Unix、Linux的常用格式

Windows 的常用格式

	AT&T 格式	Intel 格式
目的操作数d、源操作数s	op s, d 注：源操作数在左，目的操作数在右	op d, s 注：源操作数在右，目的操作数在左
寄存器的表示	mov %ebx, %eax 注：寄存器名之前必须加“%”	mov eax, ebx 注：直接写寄存器名即可
立即数的表示	mov \$985, %eax 注：立即数之前必须加“\$”	mov eax, 985 注：直接写数字即可
主存地址的表示	mov %eax, (af996h) 注：用“小括号”	mov [af996h], eax 注：用“中括号”
读写长度的表示	movb \$5, (af996h) movw \$5, (af996h) movl \$5, (af996h) addb \$4, (af996h) 注：指令后加 b、w、l 分别表示读写长度为 byte、word、dword	mov byte ptr [af996h], 5 mov word ptr [af996h], 5 mov dword ptr [af996h], 5 add byte ptr [af996h], 4 注：在主存地址前说明读写长度byte、word、dword
主存地址偏移量的表示	movl -8(%ebx), %eax 注：偏移量(基址) movl 4(%ebx, %ecx, 32), %eax 注：偏移量(基址, 变址, 比例因子)	mov eax, [ebx - 8] 注：[基址+偏移量] mov eax, [ebx + ecx * 32 + 4] 注：[基址+变址*比例因子+偏移量]

	AT&T 格式	Intel 格式
主存地址偏移量的表示	<code>movl \$4(%ebx, %ecx, 32), %eax</code> 注: 偏移量(基址, 变址, 比例因子)	<code>mov eax, [ebx + ecx*32 + 4]</code> 注: [基址+变址*比例因子+偏移量]

