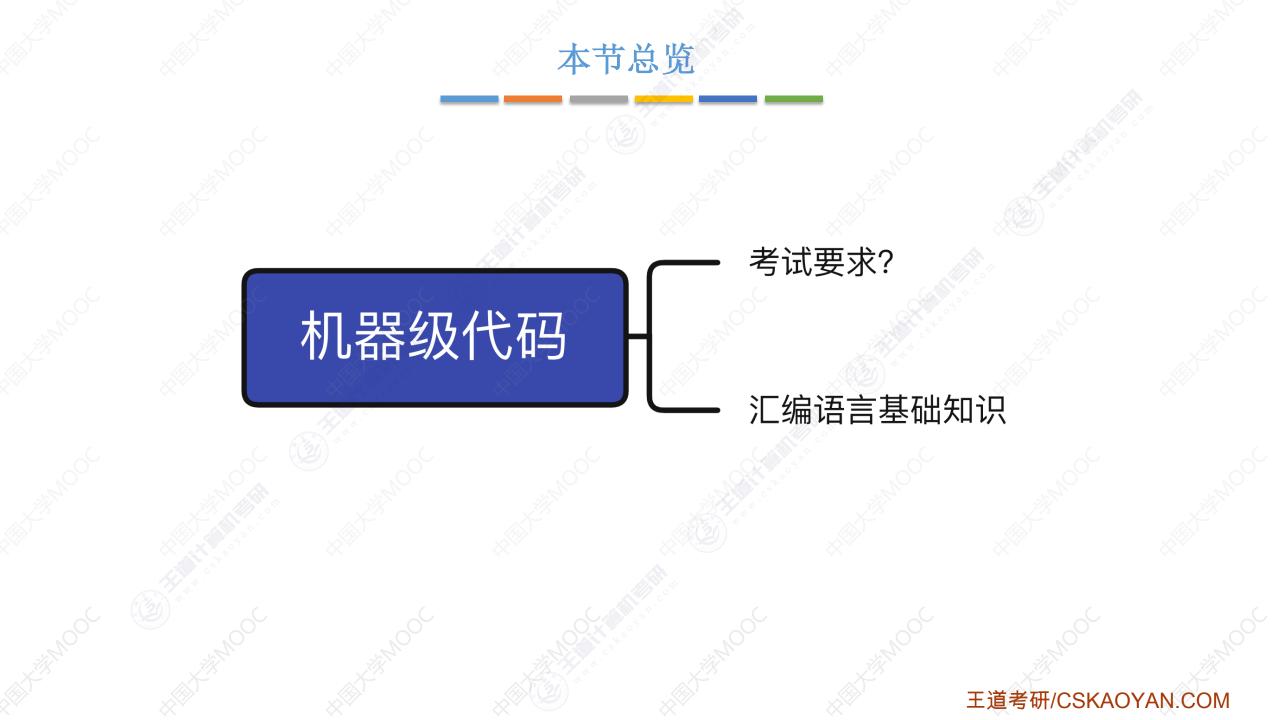
本节内容

高级语言与机器级代码之间的对应

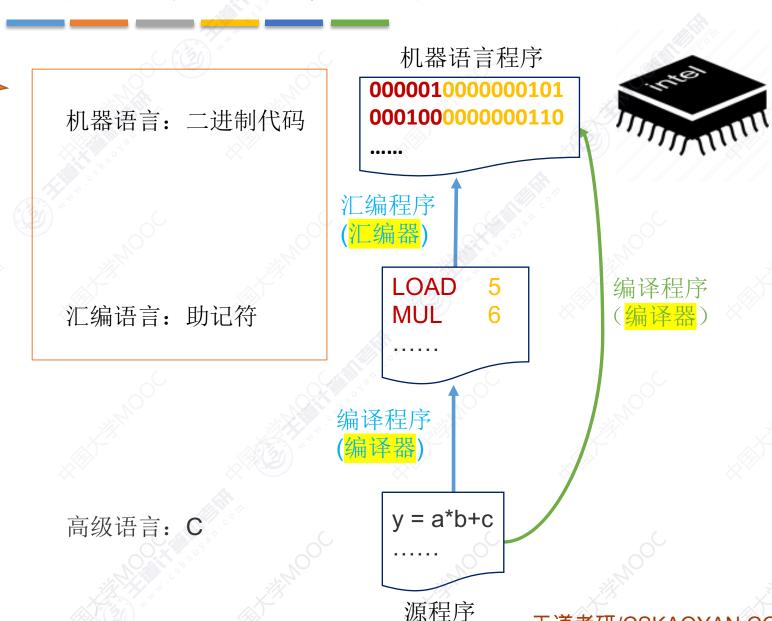


高级语言→汇编语言→机器语言

机器级 代码

考试要求:

- 只需关注x86汇编语言;若考 察其他汇编语言题目会详细注 释
- 题目给出某段简单程序的C语言、汇编语言、机器语言表示。 能结合C语言看懂汇编语言的 关键语句(看懂常见指令、选择结构、循环结构、函数调用)
- 汇编语言、机器语言 ——对 应,要能结合汇编语言分析机 器语言指令的格式、寻址方式
- 不会考:将C语言人工翻译为 汇编语言或机器语言



王道考研/CSKAOYAN.COM

历年真题

2. 【2019 统考真题】已知 $f(n) = n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$, 计算 f(n)的 C 语言函数 f1 的源程序(阴影部分)及其在 32 位计算机 M 上的部分机器级代码如下:

```
int f1(int n) {
      00401000
                            push ebp
   if(n>1)
               83 7D 08 01 cmp dword ptr [ebp+8],1
 11 00401018
      0040101C
                     jle f1+35h (00401035)
 12
               7E 17
    return n*f1(n-1);
 13 0040101E
             8B 45 08
                            mov eax, dword ptr [ebp+8]
 14 00401021 83 E8 01 sub eax, 1
 15 00401024 50
                            push eax
 16
    00401025
               E8 D6 FF FF FF call f1 ( 00401000)
 19 00401030 OF AF C1 imul eax, ecx
 20 00401033 EB 05 jmp f1+3Ah (0040103a)
    else return 1;
      00401035 B8 01 00 00 00 mov eax,1
      00401040 3B EC
                            cmp ebp, esp
      0040104A C3
                            ret
```

其中, 机器级代码行包括行号、虚拟地址、机器指令和汇编指令, 计算机 M 按字节编址, int 型数据占 32 位。请回答下列问题:



不要慌,问题不大



x86汇编语言指令基础

Intel x86架构CPU



起源: 代号为 8086 的CPU



数据在哪儿?

怎么处理?

在寄存器里

在主存里

在指令里

在指令中给出"寄存器名" x86架构的CPU有哪些寄存器?

在指令中给出"主存地址" 怎么在指令中指明读写长度?

直接在指令中给出要操作的数 也就是"立即寻址"

<mark>d</mark>estination: 目的地

以mov指令为例

source:来源、发源地



mov 目的操作数d, 源操作数s

#mov指令功能:将源操作数<mark>s</mark>复制到目的操作数<mark>d</mark>所指的位置

mov eax, ebx

mov eax, 5

mov eax, dword ptr [af996h]

mov byte ptr [af996h], 5

如何指明内存的读写长度:

#将寄存器 ebx 的值复制到寄存器 eax

#将立即数 5 复制到寄存器 eax

#将内存地址 af996h 所指的32bit值复制到寄存器 eax

#将立即数 5 复制到内存地址 af996h 所指的一字节中

dword ptr——双字,32bit word ptr——单字,16bit byte ptr——字节,8bit



adj. hexadecimal;

王道考研/CSKAOYAN.COM

x86架构CPU,有哪些寄存器?

每个寄存器都是32bit

EAX **EBX ECX EDX**

通用寄存器(X=未知) E = Extended = 32bit

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

#寄存器→寄存器 #立即数→寄存器

ESI

EDI

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

EBP

ESP

堆栈基指针 (Base Pointer)

堆栈顶指针 (Stack Pointer)

堆栈寄存器用于实现函数调用

x86架构CPU,有哪些寄存器?

每个寄存器都是32bit

31 0

| AX | (低16bit) |
|----|----------|
| вх | (低16bit) |
| CX | (低16bit) |
| DX | (低16bit) |
| | DX |

通用寄存器,使用低16bit

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

ESI (2)

EBP

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)

堆栈顶指针 (Stack Pointer)

两个堆栈寄存器只能固定使用32bit

x86架构CPU,有哪些寄存器?

每个寄存器都是32bit

31 // 0

| | AH | AL |
|-------------|------|------|
| × | 8bit | 8bit |
| | BH | BL |
| | 8bit | 8bit |
| 70 | CH | CL |
| <i>20</i> 7 | 8bit | 8bit |
| | DH | DL |
| × ' | 8bit | 8bit |

通用寄存器,使用指定8bit

mov ah, bl mov ah, byte ptr [af996h] mov ah, 5 #寄存器→寄存器 #主存→寄存器 #立即数→寄存器

| ESI | |
|-----|---------|
| EDI | ,7/1/NO |
| | |

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

EBP

堆栈基指针 (Base Pointer)

堆栈顶指针 (Stack Pointer)

两个堆栈寄存器只能固定使用32bit

更多例子

mov eax, dword ptr [ebx]
mov dword ptr [ebx], eax
mov eax, byte ptr [ebx]
mov eax, [ebx]
mov [af996h], eax

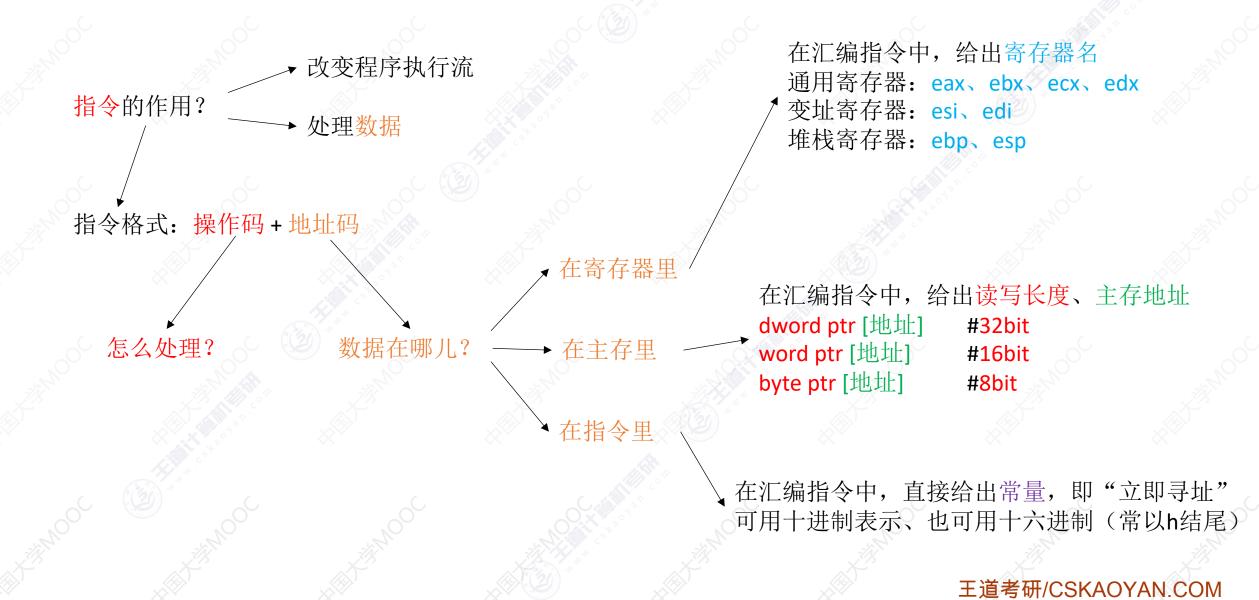
mov eax, dword ptr [ebx+8] mov eax, dword ptr [af996-12h] #将 ebx 所指主存地址的 32bit 复制到 eax 寄存器中
#将 eax 的内容复制到 ebx 所指主存地址的 32bit
#将 ebx 所指的主存地址的 8bit 复制到 eax
#若未指明主存读写长度,默认 32 bit
#将 eax 的内容复制到 af996h 所指的地址(未指明长度默认32bit)

#将 ebx+8 所指主存地址的 32bit 复制到 eax 寄存器中 #将 af996-12 所指主存地址的 32bit 复制到 eax 寄存器中





总结





△ 公众号: 王道在线





抖音: 王道计算机考研