

# 第1章 绪论



华中科技大学

- 1.1 什么是汇编语言？
- 1.2 为什么学汇编语言？
- 1.3 如何学习汇编语言？
- 1.4 汇编语言源程序举例





# 1.1 什么是汇编语言？

```
#include <stdio.h>                                // 工程: c_example
int main(int argc, char* argv[])
{
    int    x, y, z;
    x = 10;      y = 20;
    z = 3 * x + 6 * y + 4 * 8;
    printf("3*%d+6*%d+4*8=%d\n", x, y, z);
    return 0;
}
```

## Question:

编译、链接，生成 exe 文件。

源文件、执行文件中的内容是什么样的？

如何看里面的内容？

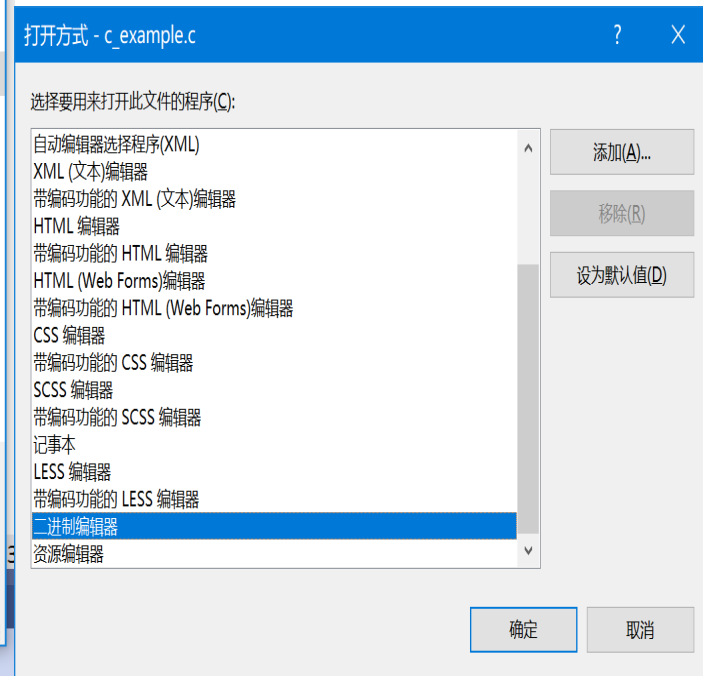
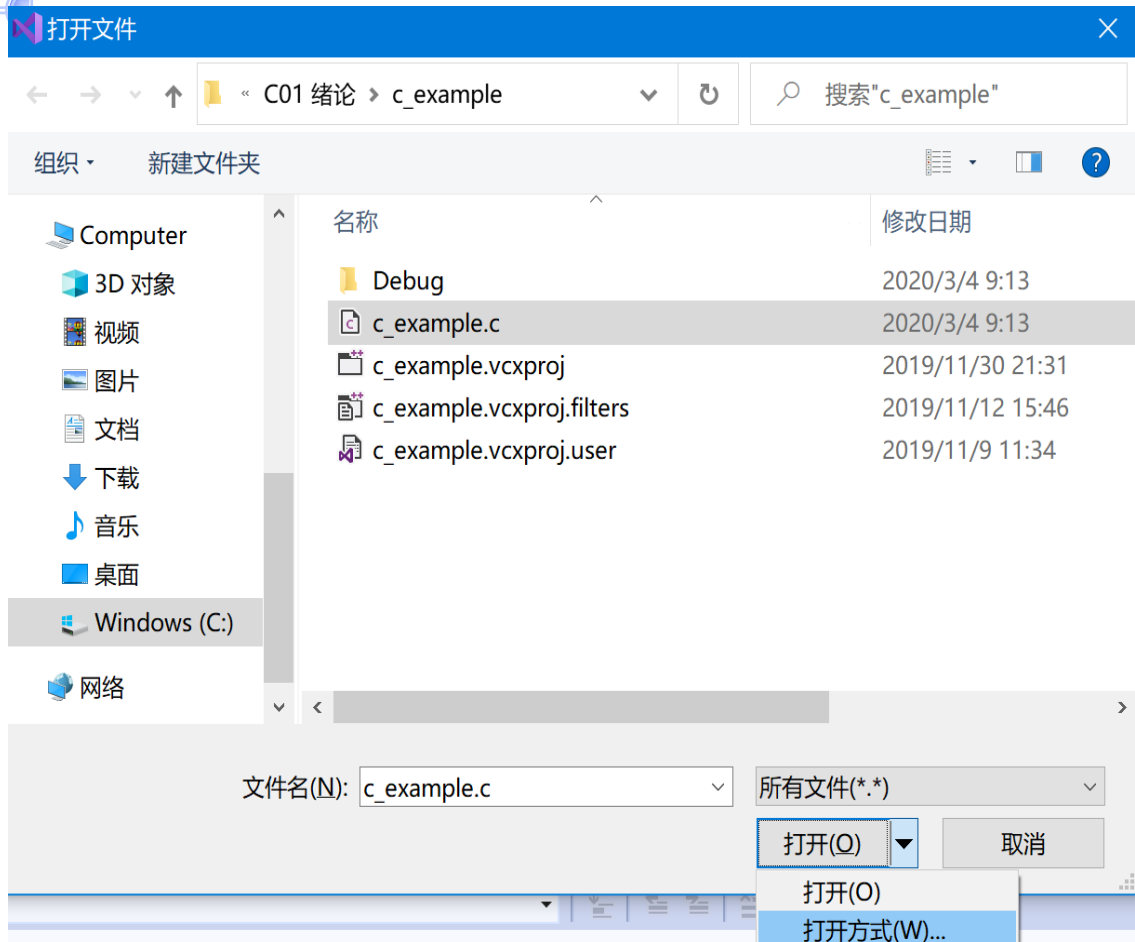




华中科技大学

# 1.1 什么是汇编语言？

操作提示



VS2019，单击“文件”->“打开文件”，在打开文件对话框中选择要打开的文件，并在“打开方式”中选择“二进制编辑器”





# 1.1 什么是汇编语言？

ASCII : American Standard Code for Information Interchange

```
c_example.c  X
00000000  23 69 6E 63 6C 75 64 65 20 3C 73 74 64 69 6F 2E  #include <stdio.
00000010  68 3E 0D 0A 0D 0A 69 6E 74 20 6D 61 69 6E 28 69  h>...int main(i
00000020  6E 74 20 61 72 67 63 2C 20 63 68 61 72 2A 20 61  nt argc, char* a
00000030  72 67 76 5B 5D 29 0D 0A 7B 0D 0A 09 69 6E 74 20  rgv[])..{...int
00000040  20 78 2C 20 79 2C 20 7A 3B 0D 0A 09 78 20 3D 20   x, y, z;...x =
00000050  31 30 3B 0D 0A 09 79 20 3D 20 32 30 3B 0D 0A 09  10;...y = 20;...
00000060  7A 20 3D 20 33 20 2A 20 78 20 2B 20 36 20 2A 20  z = 3 * x + 6 *
00000070  79 2B 20 34 2A 38 3B 0D 0A 09 70 72 69 6E 74 66  y+ 4*8;...printf
00000080  28 22 33 2A 25 64 2B 36 2A 25 64 2B 34 2A 38 3D  ("3*d+6*d+4*8=
00000090  25 64 5C 6E 22 2C 78 2C 79 2C 7A 29 3B 0D 0A 09  %d\n", x, y, z);...
000000a0  72 65 74 75 72 6E 20 30 3B 0D 0A 7D 0D 0A |    return 0;...}
```

用二进制编辑器打开源程序 c\_example.c





# 1.1 什么是汇编语言？

计算机文件：0、1串 二进制编码

4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	E8	00	00	00
0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68
69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F
74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20
6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00

用二进制编辑器打开可执行文件 c\_example.exe





# 1.1.1 机器语言

## 机器指令 (Instruction)

指挥计算机完成某一基本操作的命令，  
也称硬指令。

特点：

- 由0和1组成的二进制代码
- 能为计算机识别并执行
- 依赖于某一类型的机器，**依赖于硬件**

*Question:* ➤ 为什么书名叫<x86汇编语言程序设计>？

➤ 任意一个0,1串都能被计算机识别吗？

有编码规则



有语法规定





# 1.1.1 机器语言

机器语言：机器指令的集合  
指令系统

机器语言程序：用机器语言编写的程序

*Question:*

- 程序员可以使用的语言非常多  
计算机真正懂得的是什么语言？
- 在指令中应包含哪些信息？





## 1.1.1 机器语言

机器指令的一般形式：

操作码

地址码

**操作码**明确了运算种类。

**地址码**指出了操作数和结果存放的位置。

思考：为什么指令中出现的是地址码，  
而不是操作数（标识字符串 或者其值）？

语句  $Z = X + Y;$  形成的指令应是什么样的？



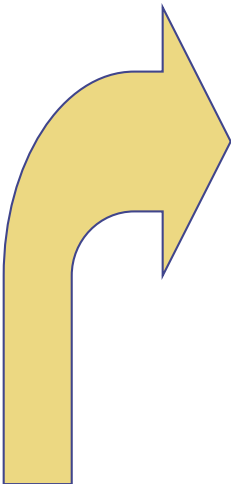


## 1.1.1 机器语言

例：将变量 **x** 中的内容 置为 10。

**C7 45 F8 0A 00 00 00**

思考题：计算机是**0**、**1**的世界，**0**，**1**串可代表什么含义？



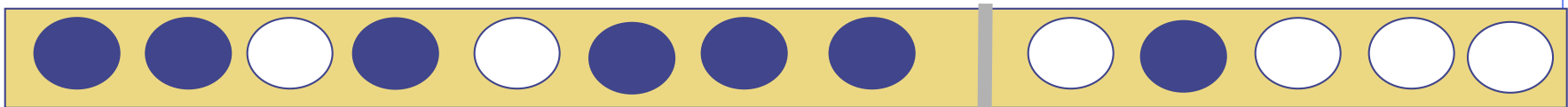
1100	0111
0100	0101
1111	1000
0000	1010
0000	0000
.....	

**0A 00 00 00 -> 10**

**F8 -> -8**，局部变量**x**在栈中的位置；

**45 -> 寻址方式**

**C7 -> 操作码**



**注意：**在不同程序中，相同语句的编码不一定相同；  
变量的地址不是固定的。



## 1.1.2 汇编语言

### 机器语言 >> 汇编语言

### 反汇编

x = 10;

00251828 **C7 45 F8 0A 00 00 00** mov dword ptr [ebp-8],0Ah

y = 20;

0025182F **C7 45 EC 14 00 00 00** mov dword ptr [ebp-14h],14h

z = 3 \* x + 6 \* y + 4 \* 8;

00251836 <b>6B 45 F8 03</b>	imul	eax,dword ptr [ebp-8],3
0025183A <b>6B 4D EC 06</b>	imul	ecx,dword ptr [ebp-14h],6
0025183E <b>8D 54 08 20</b>	lea	edx,[eax+ecx+20h]
00251842 <b>89 55 E0</b>	mov	dword ptr [ebp-20h],edx





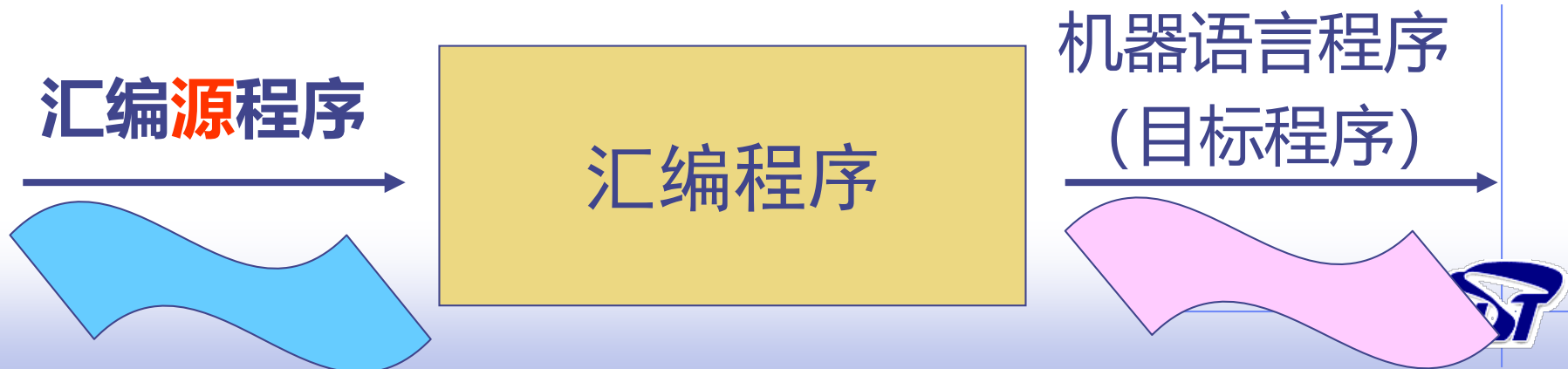
## 1.1.2 汇编语言

**符号：**指令助记符、符号地址、标号、伪指令

**汇编语言：**

用记忆符号书写的、其主要操作与机器指令基本一一对应的、并遵循一定语法规则的语言。

**汇编源程序：**用汇编语言编写的程序。





## 1.1.2 汇编语言

**伪指令：**告诉汇编程序如何进行汇编工作的命令  
也称为汇编控制命令。

- 伪指令也要写在汇编源程序中
- 在汇编后，它们没有对应的目标代码

**问题：**C语言程序中，有无类似的伪指令？





## 1.1.2 汇编语言

**讨论：汇编源程序与反汇编程序有何差异？**

**x = 10;    机器码    C7 45 F8 0A 00 00 00**

**mov    dword ptr [ebp-8], 0Ah**

**mov    dword ptr [x], 0Ah**

**x : 符号地址    对应的地址是 [ebp -8]  
在机器码中使用的不是符号地址。**





## 1.2 为什么学习汇编语言？

$z = 3 * x + 6 * y + 4 * 8;$

```
imul    eax, dword ptr [x], 3
```

```
imul    ecx, dword ptr [y], 6
```

```
lea     edx, [eax+ecx+20h]
```

```
mov     dword ptr [z], edx
```

- 用汇编语言写程序也很麻烦，因而向更高级的语言发展
- 编译器将高级语言程序翻译成机器语言程序





## 1.2 为什么学习汇编语言？

- 解密程序、逆向工程、病毒木马分析和防止的唯一选择
- 理解C语言程序的最好途径
  - 为什么调用一个函数后，能正确返回？
  - 函数之间是如何传递参数的？
  - 为什么局部变量的作用域只在函数内部？
  - 递归程序如何理解？
  - 数组越界访问是怎么回事？
  - 指针是如何指向相应对象的？
  - UNION结构是如何转换的？





## 1.2 为什么学习汇编语言？

- 解密程序、逆向工程、病毒木马分析和防止的唯一选择
- 理解C语言程序的最好途径
- 了解操作系统运行细节的最佳方式
- 特定场合下编写程序的必然选择
- 了解计算机工作原理和后继课程学习的基础







## 1.2 为什么学习汇编语言？

- 了解计算机工作原理和后继课程学习的基础

**C语言程序**

编译、连接

**可执行程序**

**执行程序**

▪ 编译原理

▪ 机器语言程序  
汇编语言

▪ 操作系统

▪ 计算机组成原理

▪ 微机接口技术





# 1.3 如何学习汇编语言?

- ◆态度决定一切，兴趣是最好的老师
- ◆深刻理解计算机工作的本质
- ◆与C语言程序设计进行关联
- ◆把握**语言**学习的要点（学习英语对比）

掌握**基础**：寄存器符号、内存组织方式等

掌握**语法**：指令格式(关键：寻址方式)

掌握**单词**：指令功能(关键：分类记忆)

组成**文章**：阅读、编写程序(关键：实践)





# 32位CPU中的通用寄存器

<b>EAX</b>		<b>AH</b>	<b>AL</b>	累加器 ( <b>AX</b> )
<b>EBX</b>		<b>BH</b>	<b>BL</b>	基址寄存器 ( <b>BX</b> )
<b>ECX</b>		<b>CH</b>	<b>CL</b>	计数寄存器 ( <b>CX</b> )
<b>EDX</b>		<b>DH</b>	<b>DL</b>	数据寄存器 ( <b>DX</b> )
<b>ESI</b>		<b>SI</b>		源变址寄存器
<b>EDI</b>		<b>DI</b>		目的变址寄存器
<b>EBP</b>		<b>BP</b>		堆栈基址寄存器
<b>ESP</b>		<b>SP</b>		堆栈指示器
	<b>31</b>	<b>16 15</b>	<b>8 7</b>	<b>0</b>

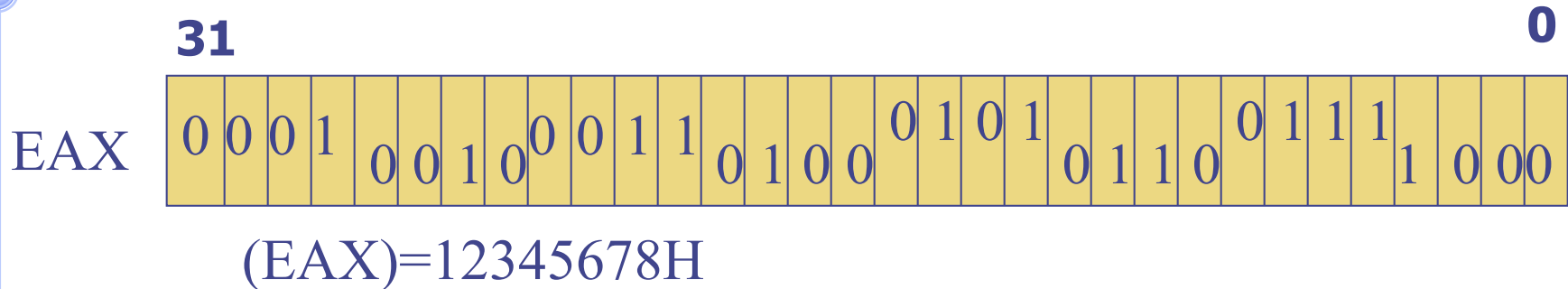
16位寄存器: AX、BX、CX、DX、SI、DI、BP、SP

8位寄存器: AH、AL、BH、BL、CH、CL、DH、DL





# 32位CPU中的通用寄存器



EAX 可以看成是某个存储单元的地址，即一存储单元地址的符号表示。

(XX): 表示XX单元中的内容。

用符号代替数字编码的好处：便于记忆。

思考题：计算机是0、1的世界，EAX中的0，1串可代表什么含义？





# 32位CPU中的通用寄存器

- 为什么要设置寄存器?
- 什么叫通用寄存器?
- 通用寄存器又为何给予特定含义的名称?



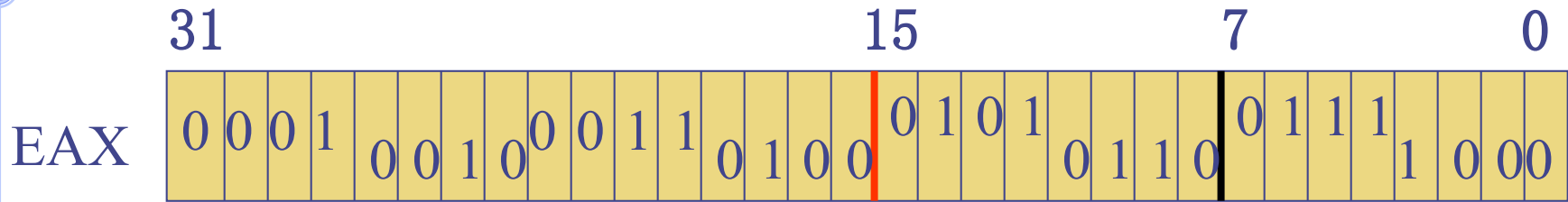


# 32位CPU中的通用寄存器

32 位寄存器	16 位寄存器	8 位寄存器	二进制编码
EAX	AX	AL	000
ECX	CX	CL	001
EDX	DX	DL	010
EBX	BX	BL	011
ESP	SP	AH	100
EBP	BP	CH	101
ESI	SI	DH	110
EDI	DI	BH	111

- 在机器指令码中，若知道使用了一个编号为000的寄存器，还需要什么信息，才能判断出使用的是EAX、AX、还是AL？
- 为什么不给所有的寄存器（24个寄存器）用5位二进制来编码呢？
- 如何理解寄存器的名字也是一个单元的符号地址？







# 1.4 汇编语言源程序举例

求：1+2+3+...+100之和，结果放入eax中。

和： EAX

加数： EBX

```
eax=0;  
for (ebx=1;ebx<=100;ebx++)  
    eax = eax+ebx;
```

工程： asm\_01\_01







# 1.4 汇编语言源程序举例

例1 : 求  $1+2+3+\dots+99$ 之和, 并显示。

```
. 686P
.model flat, c
    ExitProcess proto stdcall :dword
    includelib  kernel32.lib
    printf      proto c :vararg
    includelib  libcmnt.lib
    includelib  legacy_stdio_definitions.lib
.data
    lpFmt       db "%d", 0ah, 0dh, 0
.stack         200
.code
```





# 1.4 汇编语言源程序举例

**main** proc

; **eax=0; for (ebx=1; ebx<=100; ebx++)** **eax=eax+ebx;**

mov eax, 0 ; (eax)=0, 用于存放累加和

mov ebx, 1 ; (ebx)=1, 用于指示当前的加数

lp: cmp ebx, 100 ; 连续两条指令, 等同于  
; if (ebx>100) goto exit

jg exit

**add** **eax, ebx** ; (eax) = (eax) + (ebx)

inc ebx ; (ebx) = (ebx) +1

jmp lp ; goto lp

exit:

invoke printf, offset lpFmt, eax

invoke ExitProcess, 0

**main** endp

end





# 复习

什么是机器语言？ 什么是汇编语言？

机器语言与汇编语言之间有什么关系？

机器语言程序的反汇编与汇编源程序有何差别？

机器语言程序与高级语言程序之间有什么关系？

一条机器指令包括哪些部分？

符号地址是什么意思？采用符号地址有何好处？





# 简化段定义的Win32程序框架

. 686P

.model flat, c

ExitProcess proto stdcall :dword

includelib kernel32.lib

printf proto c :vararg

includelib libcmt.lib

includelib legacy\_stdio\_definitions.lib

.data

lpFmt db "%d", 0ah, 0dh, 0

.stack 200

.code

main proc

.....

invoke ExitProcess, 0

main endp

end

