==========
==========
==========《嵌入式技术 基础课程》之 X86 汇编
==========
==========
==========

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。

第一部分：预备知识

1、为什么要推出 X86 汇编

1.0 嵌入式技术课程特点（基于计算机体系结构）

1.1 推出精华版为后续课程准备（对比 51 汇编，ARM 汇编）

1.2 理解计算机底层运行原理（直接与机器中的寄存器打交道）

1.3 有利用理解高级语言的语法特性（编译器，隐晦语法）

1.4 学习目标：利用几节课完成汇编的学习，能看懂 X86 汇编代码

2、计算机体系结构

2.0 冯式结构与哈弗结构

独立的存储架构和信号通道（代码与数据是否统一编址）

与是否统一编址没有关系，也与 CPU 没有关系，与计算机整体设计有关

CACHE 的引入

2.1 CPU 模型

运算器和控制器

运算器（ALU） 全加器

a + b

a - b = a + (-b)

a * b = a + a ...+a

a / b = a - b ...- b

控制器：（控制存储器[指令]）

将机器码转换成电平信号

2.2 X86 微处理器

2.2.1 三大总线

地址总线的宽度决定了 CPU 的寻址能力

数据总线的宽度决定了 CPU 与其它器件进行数据传送时的一次数据送送量

控制总线的宽度决定了 CPU 对系统中其它器件的控制能力

2.2.2 X86 寻址范围

在逻辑编程中，内存的编址是以字节为单位的，与 X 位机没有关系！

8086CPU 有 20 位地址总线，达到 1MB 的寻址能力，但其数据总线是 16 位的，

在内部一次性处理只能为 16 位，表现出的寻址能力只有 64KB,所以有段地址

和偏移地址分两次处理物理地址（内存地址），段地址 16 位(变成 20 位，左移 4 位)，

偏移地址 16 位，物理地址= 段地址*16(变成 20 位) + 偏移地址,物理上是通过 20 位

去访问的，但在逻辑上我们只关心其偏移地址，物理地址的计算由硬件自动完成。

内存并没有分段，只是为了 20 位地址访问设置的。一个物理地址可以有很多种表示

物理地址 10080 可以用 1008：0 和 1000：80 来表示

3、计算机指令系统与程序语言

2.0 CISC 与 RISC

复杂指令集和精简指令集(取决于 N)

N=111（8051） 复杂指令集

N=34 （ARM） 精简指令集

SWAP（1） <---> MOV （3） 2/8 定律

2.1 程序语言

编译型 与 解释型 -->机器码

JAVA：一次编译到处运行

源文件--编译器--字节码---JVM（解释）--- 机器码

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。

第二部分：寄存器

4、X86 寄存器分类

8086 寄存器(14 个)：AX，BX，CX，DX，SP，BP，SI，DI，DS，ES，SS，CS，IP，PSW

ARM(37 个)：R0,R1.....R15 CPSR SPSR（不同的模式） 8051:R0,R1,...R7(四组)，ACC，B，PSW,SP,DP

分类：(都是 16 位)

4.1 数据寄存器:AX，BX，CX，DX

AX（AH，AL）：累加器(accumulator),它是很多加法乘法指令的缺省寄存器。

BX（BH，BL）：基地址(base)寄存器

CX（CH，CL）：计数(Counter)寄存器，LOOP 指令的内定寄存器

DX（DH，DL）：数据(Data)寄存器

4.2 指针，变址寄存器：SP，BP，SI，DI

SP（stack pointer）：堆栈指针寄存器

BP（base pointer）：基指针寄存器

SI（source index）:源变址寄存器

DI（destination index）：目标变址寄存器

4.3 段寄存器：ES，SS，CS，DS

CS（code segment）:代码段寄存器

DS（data segment）:数据段寄存器

SS（stack segment）:堆栈段寄存器

ES（extra segment）:附加段寄存器

注意：段寄存器是唯一的，但是我们可以定义多个代码段和数据段，可重合，可不重合，

只需要在访问哪个段是改变相应段寄存器中的值

4.4 控制寄存器：PSW(FLAG) IP

IP（instruction pointer）:指令指针寄存器,存放下一条指令的地址

CS*16 + IP <---->PC

关于等长指令(PC 加的值是一样的)与非等长指令（IP 加的值不一样）

PSW(FLAG) :程序状态寄存器

状态标志位：（控制转移指令相关）

进位标志 C(Carry Flag) 当结果的最高位产生进位或借位时 C=1；符号描述：CY(1)NC(0)

溢出标志 O(Overflow Flag) 算术运算中，带符号数运算超出了 8 位或 16 位时 O=1;符号描述：OV(1) NV(0)

符号标志 S(Sign Flag) 结果的最高位为 1 时 S=1；符号描述：NG(1) PL(0)

零标志 Z(Zero Flag) 若运算结果为 0 时 Z=1；符号描述：ZR(1) NZ(0)

奇偶标志 P(parity Flag) 若结果中 1 的个数为偶数时 P=1；符号描述：PE(1)PO(0)

辅助进位标志 A(Auxitiary Flag)字节操作时，低半字向高半字进位或借位 A=1；符号描述：AC(1)NA(0)

控制标志位：

方向标志 D(Direction Flag) D=1 串操作时地址自动减；D=0 串操作时地址自动增；符号描述：DN(1) UP(0)

中断允许标志 I(interrupt enable Flag) I=1 允许中断 ；I=0 屏蔽中断；符号描述：EI(1) DI(0)

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。

5、debug 使用

debug 是一个交互式的机器语言的调试程序。Windows64 位系统是没有这个命令的，需要安装 dosbox。

常用调试命令：(r d e u t a g) --修改/查看 寄存器/内存数据 与 执行/查看汇编代码

5.1: 输入 -r 查看或者修改寄存器(Register command)

-r 回车 查看各个寄存器

-r ax 回车 修改 ax 的值

-r cs 回车 修改 cs 的值 (注意：不能通过指令修改 mov cs,xxx)

-r ip 回车 修改 ip 的值

-r f 回车 修改状态寄存器的值

5.2: 输入-D 查看内存中的信息（Dump command）

-d 1000:0000 查看段地址为 1000H ,偏移地址为 0 开始的内存信息

-d 1000:0000 00ff 就是查看 1000:0000 到 1000:00ff 的内存信息

查看主板生产日期

-d fff0:0000 00ff 12/22/11 格式月/日/年， 11 年 12 月 22 号生产的主板

这里的生产日期是只读的，无法修改，通过-e 修改表面上看是改过了，

下次打开后又恢复之前数据了

5.3：输入-E 修改内存中位置的信息(Edit command)

-e 1000:0000 1a 00 09 10 a1 61

这样就可以修改对应段地址为 1000H ,偏移地址为 0 开始的内存信息

-e 1000:0000 'a+b'

这样就会在内存 1000:0000 处写入 61H 1000:0001 处+ 1000：0002 62H

5.4：输入-u 查看指定位置对应的汇编指令(Un-assemble command)

-u 1000:0000 查看该处的汇编指令

-u cs:0000 查看段地址为 cs 处的汇编指令

5.5：输入-t 执行指令(Trace command)

-t 执行 CS:IP 所指向的第一个指令，如果要执行自己写入内存的指令

应该先通过-r 来改变 CS IP 的值，然后再执行 -t 执行

-t 指令默认执行第一条指令后停止。但是当遇到修改 SS 的指令时，

会把紧接的下一条指令也执行了

5.6：输入-a 输入汇编指令(Assemble command)

-a 1000:0000 回车

这样就可以直接在 1000:0000 处输入汇编指令了，注意再输入完之后，输入回车，表示输入完毕，输入完毕后，通过-r 修改 CS ip 来指向对应的内存地址，然后就可以通过-t 逐行执行了，每执行一次通过-r 来看对应的寄存器的变化。

-a 回车 直接输入汇编指令，默认会在 cs:ip 处输入，所以直接用-a 回车输入汇编指令，然后用-t 执行那么执行的就是刚刚输入的汇编指令，这样可以用来调试汇编指令

5.7:输入-g 直接跳转到哪步执行(Go command)

用-t 是需要一步一步执行，但是往往遇到 loop 循环时候，次数很多，必须要用-g 跳到指定行执行

-g 0019 直接跳到偏移地址为 0019h 处执行

5.8：用 DEBUG 加载.exe 文件

debug D:\RadASM\Masm\Projects\test\test.exe 回车

-u 查看对应 test.exe 里的汇编指令

-t 执行

-g 跳转到哪个偏移地址执行

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。

6、helloworld

6.1 编辑，编译，链接

编辑器：记事本，notepad++，IDE 等

编译器：masm(microsoft) tasm(Borland 的 Turbo 系列) nasm

MASM 编译 LINK 链接 ML 组合命令

6.2 helloworld

6.2.1，格式 （源文件扩展名：asm）

```
DSEG SEGMENT                                ;伪指令，声明段名
MESSAGE DB 'Hello World',0DH,0AH,'$'   ;字符串以$结尾(9 号调用)
DSEG ENDS
CSEG SEGMENT
    ASSUME CS:CSEG,DS:DSEG              ;假设关联
BEGIN:
    MOV AX,DSEG                         ;设置数据段（代码段的设置由编译器设置）
    MOV DS,AX
```

```
    MOV DX,OFFSET MESSAGE          ;操作符 OFFSET 取得标号的偏移地址
    MOV AH,9                       ;显示字符串的 DOS 系统功能调用
    INT 21H
    MOV AH,4CH                     ;退出程序
    INT 21H
CSEG    ENDS
    END BEGIN                      ;程序结束，第一条指令标号，不带标
```
号表示是一个子模块

6.2.2，编译过程(源文件:hello.asm)

MASM 编译　　LINK 链接

步骤 1：输入 MASM hello.asm (可不用打入附加名.ASM)

注意：如果没有一个错误存在，即可生成 OBJ 文件。OBJ 中包含的是编译后的二进制结果，

它还无法被 DOS 载入内存中加以执行，必须加以链结（Linking）。

步骤 2：输入 LINK hello.obj (可不用附加名 OBJ) ---全部按回车即可

6.3 debug 调试 helloworld

debug hello.exe 回车

-u 查看对应 hello.exe 里的汇编指令

-t 单步执行

-g xxx 跳转到 xxx 偏移地址执行

-r -d -e -a

6.4 内存布局(memory map)（图示描述）

6.4.1 8086 地址线性空间（1MB）

总线空间：线性地址空间上有：RAM， BIOS 的 ROM， 显卡(RAM ,ROM)，网卡的(ROM)

主存储器地址空间(中断向量 （软中断，BIOS 子程序中断）+ BIOS/DOS 数据区+dos 操作系统+用户区)

显存地址空间 + BIOS ROM 地址空间

注意：这里的 ROM 可以理解成驱动程序，内存条只是地址空间的一部分。

IO 空间：（IN/OUT 命令进行数据读写） CPU--IO 接口--外部设备

IO 接口的编址方式：IO 接口中能被 CPU 访问的寄存器称为端口

端口与存储器统一编址（ARM）uart gpio

端口与存储器独立编址（X86）8259A 8255(并口) 8253(定时器)

通过 MEMR/MEMW 和 IOR/IOW 两组控制信号来实现对 I/O 端口和存储器的不同寻址

6.4.2 程序是如何操作硬件的

中断：用于调用别人写的程序。

软件中断：大多数的系统调用使用 INT21，通过调用号来区分功能

BIOS 中断，调用相应的驱动程序

对比：AH 9,INT 21H (打印字符串) / INT 10H (显示器驱动程序)

用户程序 DOS 程序 BIOS 程序

用户程序：用户自己编写的代码

DOS 程序：完成设备管理，文件管理，目录管理等功能，比如：键盘输入，显示输出，打印输出

DOS 系统功能调用（相当于系统调用）通过 INT 21H 软中断进入（调用号放入 AH 中）

系统 BIOS：提供常用外设的驱动(键盘，打印机，磁盘，串口等)

显卡 BIOS ROM 地址空间 、网卡 BIOS ROM 地址空间、系统 BIOS ROM 地址空间

注意：所有外设都是通过 IO 方式访问的，而 ROM 里存放的是操作 IO 寄存器的命令(IN/OUT),也就是驱动程序

ROM 中的驱动程序是通过中断方式（BIOS 中断）进行调用的。DOS 可以调用，用户程序也可以调用

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。

第三部分：X86 指令系统

7、汇编语句格式

7.1 语句格式：

[标号：] 操作符 OPD,OPS [;注解]

MOV AX，00 -----> B8 +rw(?) + 0000

机器码格式：opcode, ModR/M SIB P562

注意：不能虚构指令.

7.2 符号说明：

(...)表示地址"..."中的内容。 (ax)--->bx

[...]表示以地址"..."中的内容为偏移地址(指针)。 ([ax])----->bx

EA 表示偏移地址，SA 表示段地址，PA 表示物理地址

7.3 数据处理的两个问题：

7.2.1,处理的数据在什么地方？(首地址)。寄存器,内存单元(段地址+偏移地址),立即数(idata)

一般情况不指定段地址，SA 为 DS。　　mov ax,[5]

7.2.2,要处理的数据有多长？(类型匹配)

byte ptr word ptr 或 dword ptr

注意：类型匹配,OPS 和 OPD 必须得一个类型明确，如果两个都模糊，是无法操作的

例如：mov ax,10　ax 是字类型。10 可当字类型也可当 byte 类型，取明确类型的

mov ax,bx　　ax 是字类型，bx 也是字类型。两个都明确

mov [si],100 [si]只是给出了在哪里，没有确定数据长度，而 100 类型也是模糊的

可以使用 mov word ptr [si],100

8、寻址方式

8.1 寻址方式：源和目标的数据从哪里来的。寄存器，内存地址(IO 地址)，立即数。分类：是按源地址而言的；

注意：段地址 SA,偏移地址 EA 物理地址=SA*16+EA，但在实际应用中，当要访问某段一存储单元时，

不用理会其 PA，而是要关心其类型及偏移地址的表示形式

8.2 立即数寻址：就是指令中有一个操作数是立即数，mov ax,5

8.3 寄存器寻址：就是指令中有一个操作数是寄存器。mov ax,bx

8.4 直接寻址：　格式为：[idata]. EA=立即数，SA=(ds) mov ax,[5]

8.5 寄存器间接寻址：mov ax,[bx] 把 bx 寄存器的值看作地址　(指针)

8.6 寄存器相对寻址：mov ax,[bx,idata] EA=(bx)+idata,SA=(ds)　（数组元素 a[5]）

注意：能够用来间接寻址的寄存器只能是寄存器 SI,DI,BP,BX 其中之一

8.7 基址变址寻址：　mov ax,[bx,si]　EA=(bx)+(si) ,SA=(ds)　（遍历数组）

8.8 相对基址变址寻址：mov ax,[bx,si,idata] EA=(bx)+(si)+idata ,SA=(ds)

注意：BX,BP 只能作基址，SI,DI 只能作变址

9、功能分类

8086 指令大约为 100 多条。可以根据功能分为下面几类：

9.1，数据传送类 mov push pop lea (传送偏移地址)

9.1.1 mov 指令中不能向 CS 作传送操作。

9.1.2 push ops　/　pop　opd

push AX　　sp-1->sp; (AH)->[SP]　　sp-1->sp;(AL)->[SP]

POP　AX　　([sp])->al ;(sp)+1->sp　　([sp])->ah ;(sp)+1->sp

9.1.3 lea opd,ops //opd 是一个 16 位通用寄存器，ops 是一个存储器的地址

mov ax,offset num　/ lea ax,num

lea ax,[si] / mov ax,[si]

9.2，算术和逻辑指令 add sub inc dec imul idiv and or xor not test neg shl shr

加类指令(add,adc,inc) add ax,[10] ; ([10]) +(ax)---> ax

减类指令(dec,sub,cmp) dec ax

乘类指令(imul,mul) :mul ops (al)*(ops)->ax (ax)*(ops)->dx,ax

除类指令(idiv,div) :div ops (ax)/(ops)->al(商) ah(余数)

9.3，控制转移指令 jmp cmp je(ne,z,g,ge,l,le) call ret(f)

9.4，串操作指令（略）

9.5，伪指令 equ db dw dd dq assume proc org

伪指令：用来辅助编译器工作的，它不是指令，不会生成机器码。

段定义伪指令：segment ends(段结束) end(程序结束)

假定伪指令：assume 注意：cs 和 ss 段由系统自动设置，而 ds 和 es 段需要在程序中显式的说明

数据定义伪指令： db dw dd

10、汇编中表达式

对一个汇编语句而言，除了正确地选择操作符外，其主要的问题在于如何正确地表示操作数据的地址。

操作数据地址在汇编中体现在表达式中。常用的下面两类表达式：

10.1，常量（EQU）与数值表达式

数值表达式(只有大小没有属性) MOV ax,(1+2)

算术：+ - * / MOD SHR SHL （区分指令--编译器来操作的）

逻辑：AND OR XOR NOT

关系: EQ NE LT GT LE GE

10.2，变量，标号与地址表达式

变量:(段属性，偏移地址，类型：长度 PTR) ，一般在数据段和附加段中定义

格式：变量名 数据定义伪指令 表达式

数据定义伪指令：db dw dd (类型)

表达式用来确认变量初值

A dw M ;定义变量 A，初值为 M 的 EA（不能写成 db,偏移地址是双字节的）

BUF db "hello",0dH,0AH ;定义以变量 BUF 为首的 7 个字节

CON EQU 50H ; 定义常量，不分配内存单元

M DB 2 dup(1),2 dup(2,'0'),'123',1,2,3 ; n dup(表达式) 表示重复语句

例如：MOV BL,BUF+2

MOV SI,OFFSET BUF

MOV BL,[SI,2]

标号:(段属性，偏移地址，类型：NEAR/FAR)机器指令前存放的地址符号，

表示当前机器指令的偏移量，也可以是过程名(函数名):ADDFUN PROC FAR
MOV AX,BEGIN ;标号取的是 EA

地址表达式：它由变量，标号，常量，寄存器（BP,BX,SI,DI 加方括号表示的）和运算符组成的有意义的式子

地址表达式中的出现变量和标号，则均是取它们的 EA 参加运算，绝不可以理解为取其存储单元中的内容。

地址表达式一般都是段内偏移地址，它具有段地址，偏移地址及类型三个属性。

注意：单个变量或寄存器取的是值，它是地址表达式的特例，不是取它们的 EA 参与运算，而是直接取其值。

MOV AL,BUF (BUF)--> AL ;注意类型匹配
MOV AL,[BUF] ;直接寻址
MOV AX,BP
例如：mov byte ptr[si+4+BEGIN],'x' ;改变了段属性 CS:[xx]，默认是 DS，而 BEGIN 是 CS 段的

总结：表达式是为了更方便的描述相应的偏移地址，把一些工作交给编译器去做。

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。

第四部分：程序控制

在汇编语言程序中，汇编语句群最常见的结构有以下几种：顺序，选择，循环。

注意：在汇编程序设计中，最重要的是如何合理的分配存储区并给变量命名(包含分配寄存器)

11、顺序与选择控制

11.1 顺序结构:程序流程无分支，无循环，无转移，以直线方式一条指令接着一条指令顺序执行。

习题：从键盘输入 0-9 中任一自然数 x,求其立方值。输入提示信息为"PLEASE INPUT X(0...9):"

分析：TAB DW 0,1,8,27,64,125,216,343,512,729

11.2 选择结构：根据不同的条件，分成若干个分支路。

转移指令分为：条件转移和无条件转移--->修改 ip 的值

控制转移指令：jmp cmp je(ne,z,g,ge,l,le)---[equal zero greater less]

前一次操作影响状态标志位(cmp ax,bx)

操作符 + 短标号(je lable)

注意：大多数情况，在 cmp 指令之后执行条件跳转指令。cmp 指令设置标志位，以便判断小于、大于、等于等情况，进行比较运算时，

不区分比较对象是有符号还是无符号，它是机械的从左边的操作数中减去右边的操作数，并根据运算结果设置各标志的状态。

习题：从键盘输入 0-9 中任一自然数 x,求其立方值，若输入的字符不是 0-9 中某个数字，则显示”INPUT ERROR!”

12、循环控制

在顺序和选择控制中，任一语句执行次数最多一次，循环控制中可以多次。

循环结构一般分成四部分：1，初值 2，循环体 3，修改初值 4，判断条件

循环控制方法：计数控制(循环次数明确)，条件控制(循环次数无法确定)---实质-->条件控制转移

Loop 指令等效于 dec cx jnz l <--等效->loop l

习题 1：1+2+3...+100

习题 2：假设有一串数字{2,4,10,8,14,1,20}，用汇编语言实现冒泡排序算法（双重循环）

13、子程序/IO/中断

13.1 子程序(函数)

13.1.1 子程序格式

```
FUN PROC     [NEAR/FAR]
    .....
FUN ENDP
```

13.1.2 子程序的调用与返回

call (IP)---> (SP) 目标的偏移地址--->IP 注意：段间调用还要压 CS

ret (SP)---> (IP)

13.1.3 现场保护与恢复

PUSH/POP

13.1.4 参数传递：寄存器，约定单元，堆栈 --（汇编是自己规定）

习题：把上面的题子改成子程序的形式

13.2 IO:CPU 与外设之间的数据通信

明确一点：WIN2000 以后,系统硬件的访问做了保护,必须通过 WDM(Windows Driver Model)去访问硬件。

13.2.1 输入指令 IN

```
IN AL,PORT      ;(PORT)--->AL
IN AX,PORT      ;(PORT)--->AX
IN AL,DX        ;([DX])--->AL
IN AX,DX        ;([DX])--->AX
```

13.2.2 输出指令 OUT

```
OUT PORT,AL     ;(AL)--->PORT
```

OUT PORT,AX ;(AX)--->PORT

OUT DX，AL ;(AL)--->[DX]

OUT DX，AX ;(AX)--->[DX]

注意：PORT 和[DX]均指外设寄存器地址，若外设地址为 0-255 时，直接将地址码写在指令中

当大于 255 时，先装地址码装入 DX 之中，然后将 DX 写入指令中

13.2.3 计算机与外设传送数据的方式

查询/中断:CPU 的利用率，中断需要中断系统支持

直接存储器传送(DMA):配置外设寄存器(传送双方的起始位置和个数)，DMA 准备就绪，CPU 让出总线控制权。

外设与存储器很快完成数据交换，CPU 又收回控制权。

13.3 中断

13.3.1 中断的分类

内部中断(除法出错，溢出中断，软中断，单步中断)和外部中断(8259A)

软中断：INT n ,执行 INT 时会产生中断。其中 INT 21H 是用来调用 DOS 系统功能的，还有各种 BIOS 中断

DOC 提供了 75 个系统调用，编号从 0-57H,主要分为：设备管理，文件管理，目录管理等。

使用系统调用一般过程为，将调用号放入寄存器 AH 中，写好入口参数，然后执行软中断

INT 21H，比如显示字符串就是 9 号调用。

13.3.1 中断向量表

存放中断处理程序的入口地址信息,响应相应中断时，CPU 会从相应的入口地址处把中断服务程序地址放入 CS:IP 中

有的处理器是 PC 直接跳到相应的中断向量位置去。(8051/ARM)

14、AT&T 与 Intel 语法差异

注意：AT&T 与 Intel 语法只是在书写方式上不同，最终生成的机器码是一样的。（参考 codeblocks 上汇编窗口）

14.1，前缀方向：AT&T 语法下，寄存器加前缀%,立即数加前缀$

14.2, 操作方向：两者相反 Intel 语法，第一个是目的操作数，第二个是源操作数

14.3，间接寻址方向：Intel 语法用[]，AT&T 语法用()

14.4, 基址变址方向：[base+index*scale+disp]，disp(base,index,scale)

14.5，操作码的后缀：AT&T 的操作码后加后缀"l"(32 位),"w"(16 位),"b"(8 位)

Intel 在操作数的前加 byte ptr word ptr 或 dword ptr

例如：sub $0x30,%esp

mov %eax,(%esp)

movl $0x0,0x74(%esp)

15、Win32/64 汇编

Win32/64 汇编，会涉及到 Windows 编程方面的知识，各种 WinAPI 的使用。这样会陷

入 Windows 编程技术用去

而学习 8086 汇编，可以更多的关注汇编语言本身，理解在实模式下的计算机运算原理。

相比于 16 位汇编，Win32/64 增加了一些寄存器，指令和寻址方式。(AL AX EAX RAX)

预告:Win32 汇编编程

目的：了解 Windows 系统的消息驱动机制，用户界面，多线程，图形操作系统内存管理，注册表和 INI 文件。动态链接库，TCP/IP 网络通信和 PE 文件使用 Win32 汇编是了解操作系统运行细节的最佳方式!

最后：

学习目标：能否看懂 X86 汇编代码

学习建议：

1，知识就是一张网，相互联系的,不管是从哪个方向突破，都会遇到不熟悉的问题。承认没有学习过的知识点。

2，源点(熟悉知识点)+推理----->新知识。完善自己的知识体系结构

3，熟悉知识<---相互-->应用知识

出品：佳嵌工作室

“佳嵌工作室”致力于物联网、嵌入式产品以及‘物联网、嵌入式技术课程’的研发，希望所推出的技术课程，能够帮助到正在努力

学习物联网、嵌入式的同学，欢迎大家和我们一起共同学习，共同进步。
