狂神聊汇编先导课

bilibili关注:遇见狂神说

绝对是你听过最通俗有趣的底层技术讲解

不为任何机构站台，编程是爱好，恭喜你发现宝藏男孩一枚~希望你们关注我是因为喜欢我

概述

语言

进制

进制如何运算

二进制

数据宽度

有符号数和无符号数

原码反码补码

位运算

位运算计算

汇编

寄存器

内存

汇编指令

内存复制

堆栈的指令

汇编如何写函数

堆栈传参

堆栈平衡

外挂

机器语言

人和人沟通？ 语言！老外！计算机！ 学习计算机的语言！

什么是机器语言?

```
# 我们目前主流的电子计算机！
状态：0 和 1
# 最早的程序员：穿孔卡带！
加 0100 0000
减 0100 1000
乘 0100 1000 0100 1000
除 0100 1000 1100 1000
```

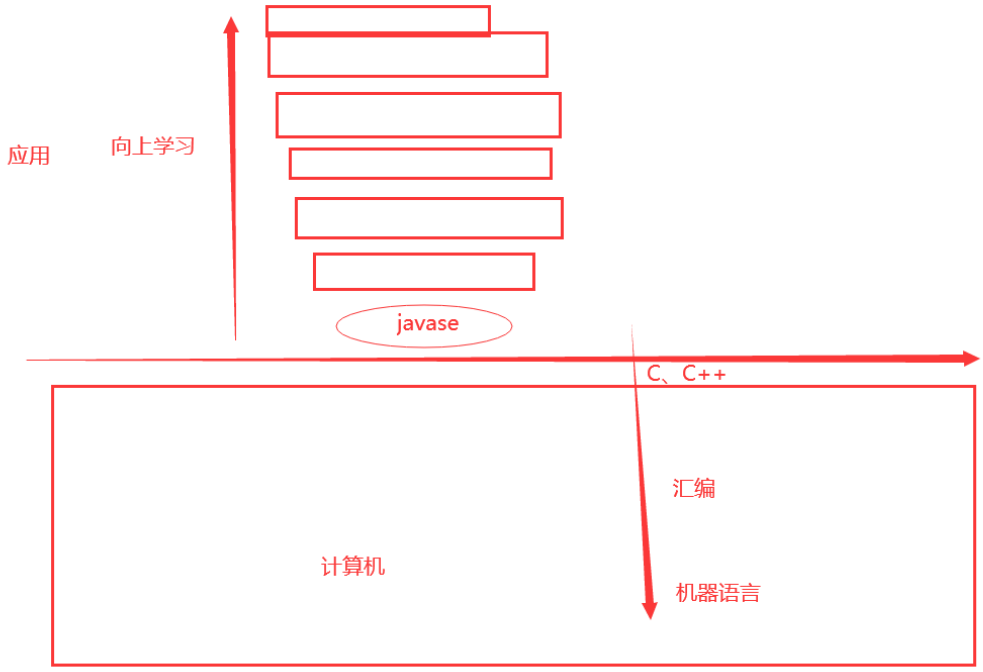
这些复杂的机器语言，能简化吗？助记符！**汇编语言**！ 人能够理解的语言转换成为机器能够理解的语言！

```
加 INC -编译器->  0100 0000
减 DEC            0100 1000
乘 MUL            0100 1000 0100 1000
除 DIV            0100 1000 1100 1000
```

离程序的本质：隔阂！汇编一般用于底层的编写，单片机..

C语言

```
加 A+B -编译器->  0100 0000
减 A-B            0100 1000
乘 A*B            0100 1000 0100 1000
除 A/B            0100 1000 1100 1000
```



进制

二进制? 0 1

学习进制的障碍?

10进制!

人类天然的选择的就是10进制，10个指头。跳出固有思维的方法！"屈指可数"

二进制

思想：每一种进制都是完美的，都有自己的计算方式!

进制?

1进制：一进一，结绳记事。1 1

2进制：二进一，计算机

八进制：八进一。8个符号组成：0 1 2 3 4 5 6 7

10进制：10进一。10个符号组成：0 1 2 3 4 5 6 7 8 9

16进制：16进一。16个符号组成：0 1 2 3 4 5 6 7 8 9 a b c d e f

进制远远没有大家想的那么复杂。**查数**

> 测试

```
# 一进制  1~20
1
1 1
1 1 1
1 1 1 1
.....

# 三进制  1~20
十进制: 0   1   2   3   4   5   6   7   8   9   10
三进制: 0   1   2   10  11  12  20  21  22  100 101  102 100 101  102
 110 111  112 120 121  122
# 二进制
      0   1   10  11  100  101  110 111  1000

# 七进制  1~20
0   1   2   3   4   5   6
10  11  12  13  14  15  16
20  21  22  23  24  25  26

```

```
# 一组符号： 逢几进几
```

问题：你真的理解进制了吗？ 1 + 1 = 3 对吗？！ 如果你可以使用进制来解答这个问题，那么你就学会了！

十进制：0 1 2 3 4 5 6 7 8 9

狂神的十进制：0 2 4 7 8 a b r d f， 可以自己随便定义的，学习，创造者！

加密解密：程序员，破解程序的人！ **进制的加密**

数字量一大，总是有规律的！

进制怎么运算

```
# 八进制计算下面的结果
2+3=5
2*3=6
4+5=11
4*5=24

# 运算的本质就是查数
0  1  2  3  4  5  6  7  10  11  12  13  14  15  16  17  20  21 22 23  24 25
 26  27

# 八进制计算下面的结果  九九乘法表=加法表！
277+333 =
276*54 =
237-54 =
234/4 =
```

八进制的乘法表

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7
2*2=4	2*3=6	2*4=10	2*5=12	2*6=14	2*7=16	
3*3=11	3*4=14	3*5=17	3*6=22	3*7=25		
4*4=20	4*5=24	4*6=30	4*7=34			
5*5=31	5*6=36	5*7=43				
6*6=44	6*7=52					
7*7=61						

八进制的加法表：

1+1=2						
1+2=3	2+2=4					
1+3=4	2+3=5	3+3=6				
1+4=5	2+4=6	3+4=7	4+4=10			
1+5=6	2+5=7	3+5=10	4+5=11	5+5=12		
1+6=7	2+6=10	3+6=11	4+6=12	5+6=13	6+6=14	
1+7=10	2+7=11	3+7=12	4+7=13	5+7=14	6+7=15	7+7=16

```
# 运算的本质就是查数

277
333   +
--------
632


    276
     54   *
--------
   1370
  1666     +
--------
  20250

# 减法的本质其实就是加法！   237-54 = 237 + (-54)
--------
# 除法的本质，除数乘以那个数最接近结果即可！
234
  4
----
 47
```

结论：无论是什么进制，本身都是有一套完美的运算体系的，我们都可以通过列表的方式将它计算出来！

二进制

计算机使用二进制 0 1 ！ 状态！电子！ 物理极限：摩尔定律！硬操作！ 追求语言的极限！并发语言！软操作！

量子计算机：(传道)

可以实现量子计算的机器。

传统的计算机：集成电路！0 1 。 硅晶片！

量子计算机的单位：昆比特。（量子比特！）量子的两态来表示。

光子：正交偏振方向。

磁场：电子的自旋方向。

21世纪。计算力。快到尽头了！【落伍】本质问题！

量子计算机！ 提高计算机的计算力。

量子比特、量子叠加态、量子纠缠、量子并行原理.....

2019年，Google研究人员展示其最新54比特量子计算机，该计算机只用200秒便可计算完毕当前世界最大的超级计算机需1万年进行的运算。

2020年，6.18。量子体积64的量子计算机！！！

霍尼韦尔还表示，将在一年之内得到至少10个有效量子比特，相当于1024个量子体积。量产！

电子计算机 == 量子计算机！

回到我们的电子计算机！0 1！

```
二进制： 0    1111
0  1  10  11  100  101  110  111 1000  1001 1010 1011 1100 1101  1110 1111
```

二进制这么去写很麻烦！二进制能否简写！

```
0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
```

这就是我们的16进制。

课程上的教学！2进制转换为10进制。然后计算！

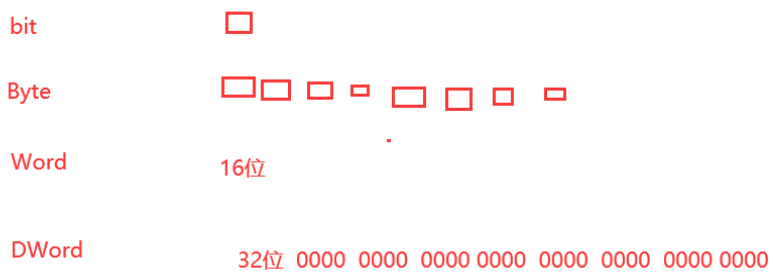
为什么要学习理解二进制?

寄存器、内存、位！底层的每一个位都是有含义的。汇编入门理解的基础！

汇编高级：了解程序的深层！操作系统的内核?

数据宽度

计算机：内存！给数据增加数据宽度。



C 和 C++ Java都需要定于数据的类型。计算机底层需要我们给这些数据定义宽度。

位 0 1

字节　0~0xFF

字　　0~0xFFFF

双字　0~0xFFFFFFF

在计算机中，每一个数据都需要给它定义类型。给它定义宽度。在内存中的宽度。

有符号数无符号数

数据都是有宽度的。每个数据代表什么意思呢？，二进制

```
0 1 0 1 0 1 0 1
```

规则，二进制解码增加一个规则?

无符号数规则

你这数字是什么，那就是什么。

```
1 0 0 1  1 0 1 0 十六进制：0x9A  十进制 154
```

有符号数规则

最高位是符号位：1 (负数) 0（正数）

```
1 0 0 1  1 0 1 0 如何转换?
```

原码反码补码

之后要用它来计算。

编码规则

有符号数的编码规则

原码： 最高位符号位，对齐它的为进行本身绝对值即可。

反码：

- 正数：反码和原码相同
- 负数：符号位一定是1，其余位对原码取反。

补码：

- 正数：补码和原码相同
- 负数：符号位一定是1，反码+1

> 测试

```
# 现在我说的这些都是 8 位
# 如果是正数，那都是一样的。
1
#原码 0 0 0 0  0 0 0 1
```

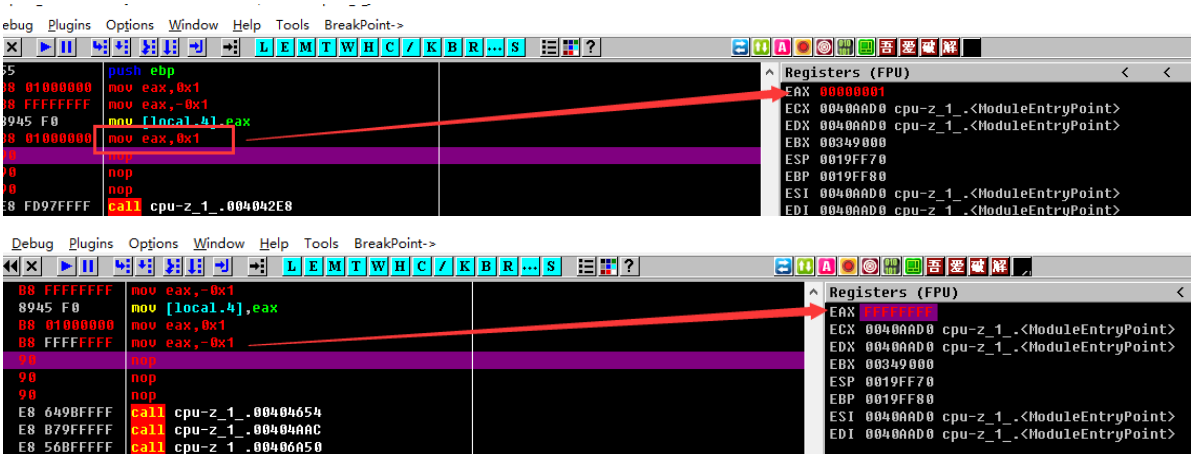
```
#反码 0 0 0 0  0 0 0 1
#补码 0 0 0 0  0 0 0 1

# 现在我说的这些都是 8 位
# 如果是负数
-1
#原码 1 0 0 0  0 0 0 1
#反码 1 1 1 1  1 1 1 0
#补码 1 1 1 1  1 1 1 1
-7
#原码 1 0 0 0  0 1 1 1
#反码 1 1 1 1  1 0 0 0
#补码 1 1 1 1  1 0 0 1

```

如果看到一个数字，二进制的，需要了解它是有符号数还是无符号数。

寄存器：mov 寄存器，值



学习通过直接操作来查看是最有效的。

位运算

计算机现在可以存储所有的数字(整数，浮点数，字符)的，运算。！

0 1

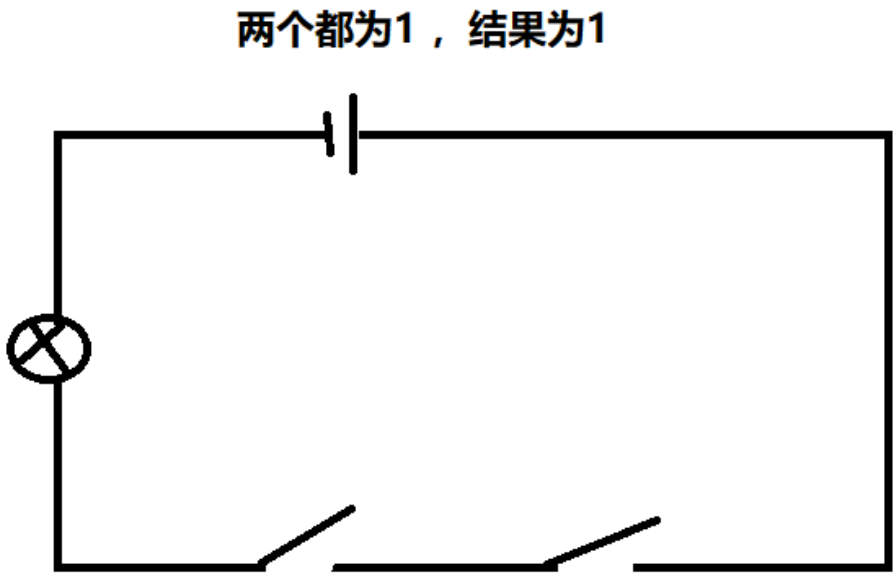
位运算?

2*8 最高效计算方式。

很多底层的调试器。需要通过位来判断CPU的状态。

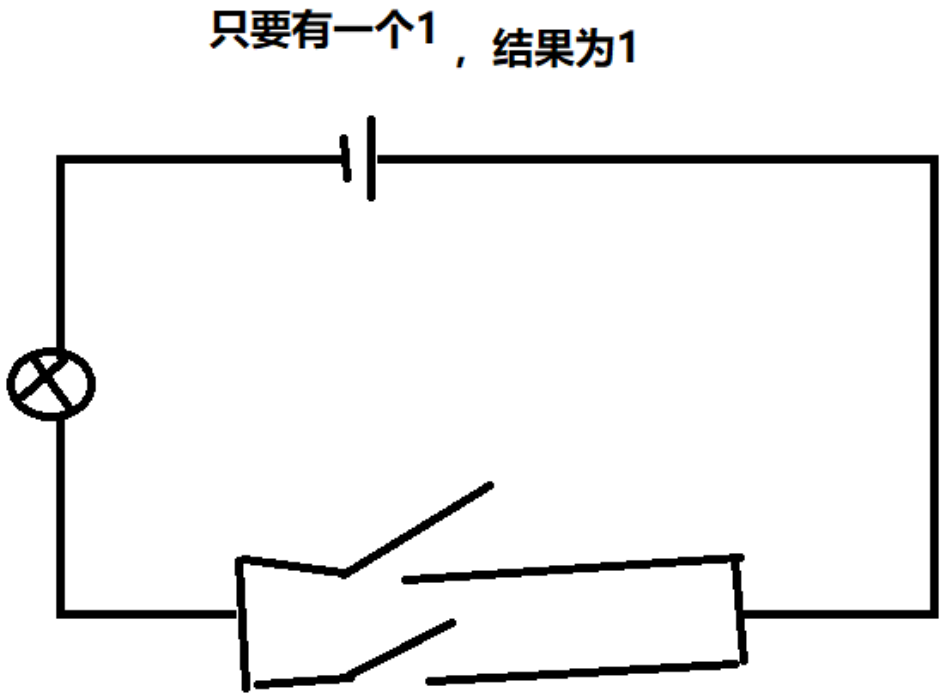
与运算(and &)

计算机的本质。




```
1011 0001
1101 1000
----------  与运算。
1001 0000
```

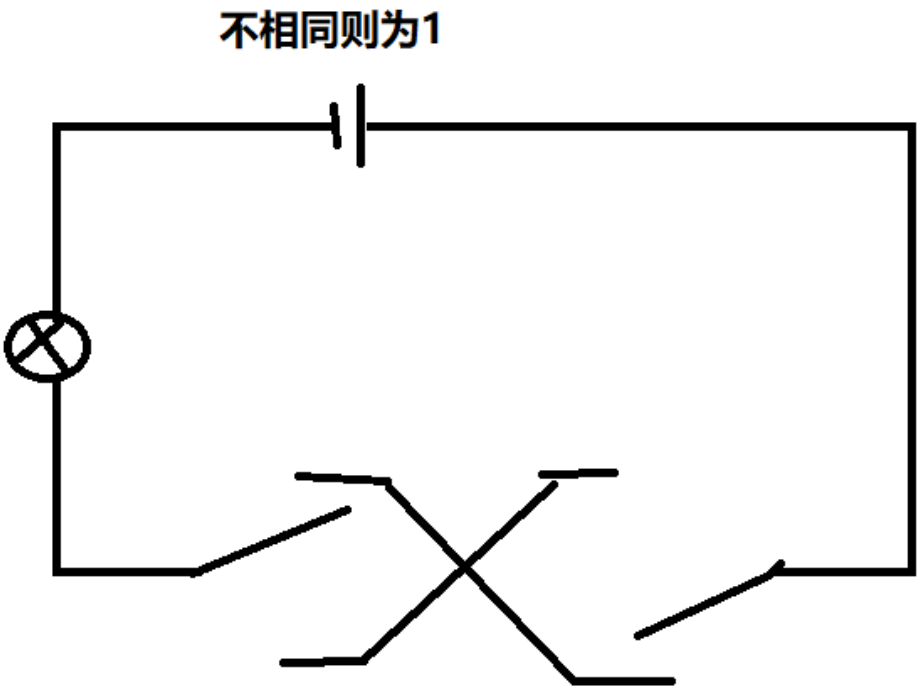
或运算（or |）




```
1011 0001
1101 1000
---------- 或运算
1111 1001
```

异或运算（xor ^）

不一样就是1。




```
1011 0001
1101 1000
----------- 异或运算
0110 1001
```

非运算（单目运算符 not ~）

0就是1,1就是0， 取反！

```
1101 1000
----------
0010 0111
```

通过这些可以完成加减乘除！位运算来实现加减乘除！

位运算(移动位)

左移：(shl <<)

```
0000 0001   @ 所有二进制位全部左移若干位，高位就丢弃了，低位补0
0000 0010
```

右移：(shr >>)

```
0000 0001   @ 所有二进制位全部右移若干位，低位就丢弃了，高位就需要补0,1(符号位决定。)
0000 0000

int a = 10;
printf("%d\n",a>>2);
```

二进制、位运算=> 加减乘除

位运算的加减乘除

计算机只认识 0 1

基本数学是建立在 加减乘除。（加法）

4+5?

```
# 计算机是怎么操作的!
0000 0100
0000 0101
---------- （加法：计算机是不会直接加的）
0000 1001

# 计算机的实现原理

# 第一步：异或： 如果不考虑进位，异或就可以直接出结果。
0000 0100
0000 0101
------------
0000 0001

# 第二步：与运算（判断进位，如果与运算结果为0，没有进位。）
0000 0100
0000 0101
------------
0000 0100

# 第三步：将与运算的结果，左移一位。  0000 1000 # 进位的结果

# 第四步：异或!
0000 0001
0000 1000
-------------
0000 1001

# 第五步：与运算（判断进位，如果与运算结果为0，没有进位。）
0000 0001
0000 1000
-----------
0000 0000

# 所以最终的结果就是与运算为0的结果的上一个异或运算。
```

4-5?

```
# 计算机是怎么操作的!
4+(-5)

0000 0100
1111 1011
---------- （减法：计算机是不会直接减的）
1111 1111


0000 0100
1111 1011
```

```
----------- 异或(如果不考虑进位，异或就可以直接出结果。)
1111 1111

0000 0100
1111 1011
----------- 与（判断进位，如果与运算结果为0，没有进位。）
0000 0000

最终结果 1111 1111  16 ff   10  -1
```

乘：x*y， 就是 y 个 x 相加，还是加法

除：x/y， 本质就是减法，就是X 能减去多少个Y。

计算机只会做加法！

机器语言就是位运算。都是电路来实现的。这就是计算机的最底层的本质。

通过机器语言来实现加法计算器。设计电路。

汇编语言环境说明

通过指令来代替我们的二进制编码！

4-5?

```
# 计算机是怎么操作的！
4+(-5)

0000 0100
1111 1011
---------- （减法：计算机是不会直接减的）
1111 1111


0000 0100
1111 1011
----------- 异或(如果不考虑进位，异或就可以直接出结果。)
1111 1111

0000 0100                          ADD指令转换为二进制操作
1111 1011
----------- 与（判断进位，如果与运算结果为0，没有进位。）
0000 0000

最终结果 1111 1111  16 ff   10  -1
```

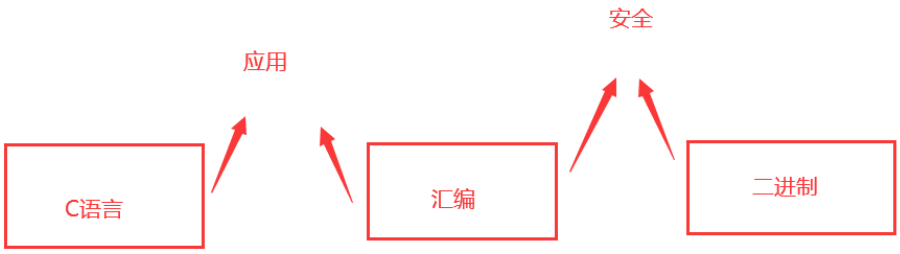
乘：x*y， 就是 y 个 x 相加，还是加法

通过汇编指令可以给计算机发一些操作，然后让计算机执行。编译器的发展，底层的大佬，几乎都是最原始的IDE。

在学习汇编之前，大家需要先掌握环境的配置（1、Vc6（程序到汇编的理解），2、OD！3、抓包工具、4、加密解密工具）

学汇编不是为了写代码

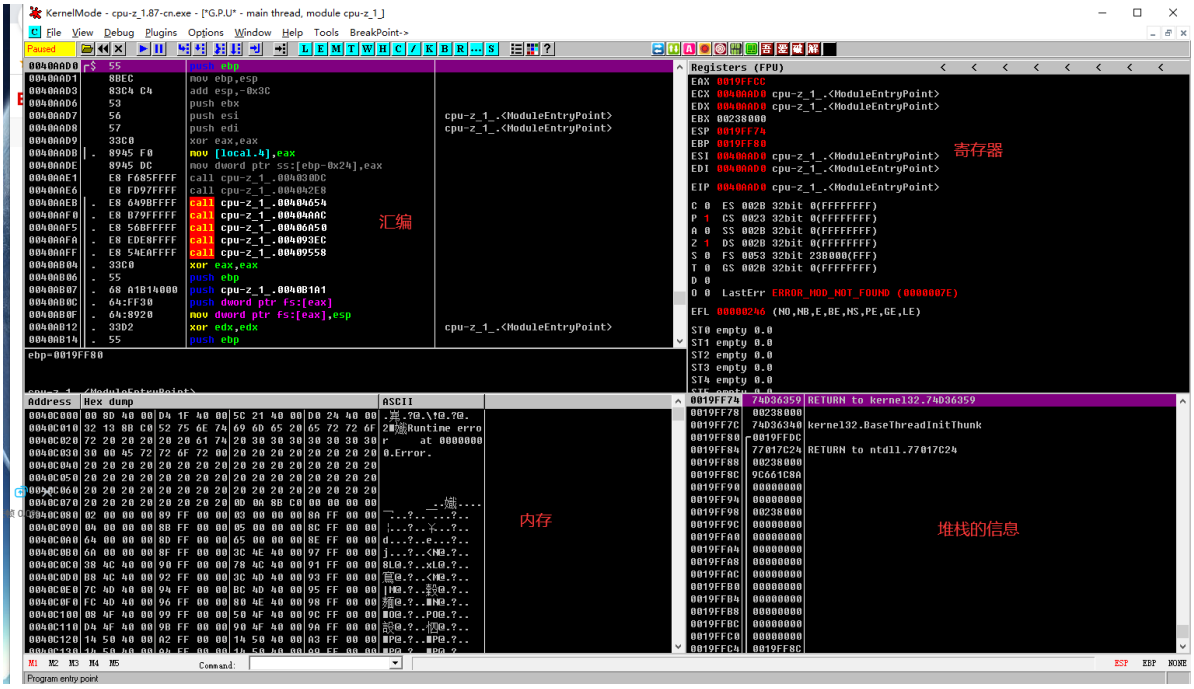
理解程序的本质。



《汇编语言》 16位的汇编 32位 64位（本质架构区别不大，寻址能力增加。）

建议大家可以直接学习32位汇编！

汇编入门：了解汇编和程序的对应关系，程序的本质即可！



通用寄存器

寄存器：

存储数据：CPU > 内存 > 硬盘

32位 CPU 8 16 32

64位 CPU 8 16 32 64

通用寄存器，可以存储任意的东西

```
1  # 32位的通用寄存器只有8个
```

存值的范围 0 ~ FFFFFFFF

对于二进制来说，直接修改值

计算机如果像寄存器存值。

mov指令

```
mov 存的地址,存的数
mov 存的地址1,存的地址1
```

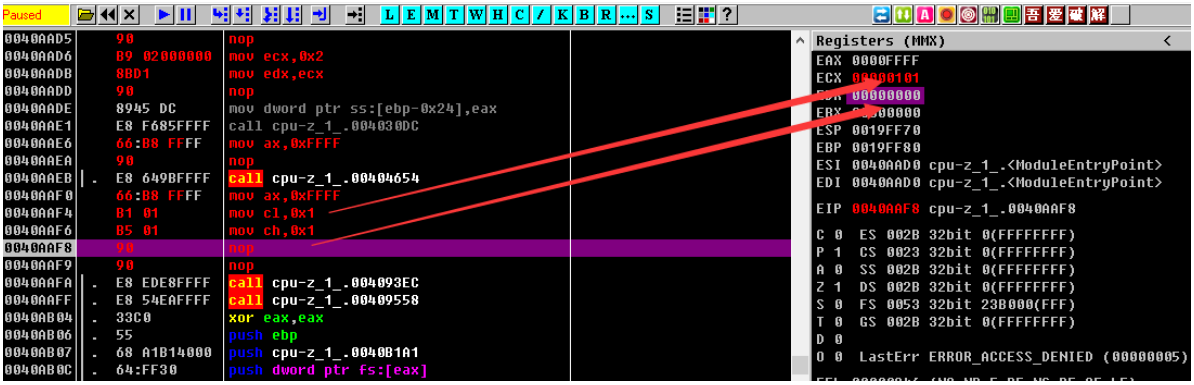
可以将数字写入到寄存器，可以将寄存器中的值写到寄存器。

计算机：计算力！

不同的寄存器

```
        FFFF    FF  0000 0000
32位    16位    8位
EAX     AX      AL
ECX     CX      CL
EDX     DX      DL
EBX     BX      BL
ESP     SP      AH
ENP     NP      CH
ESI     SI      DH
EDI     DI      BH
```

8位：L低8位，H 高8位



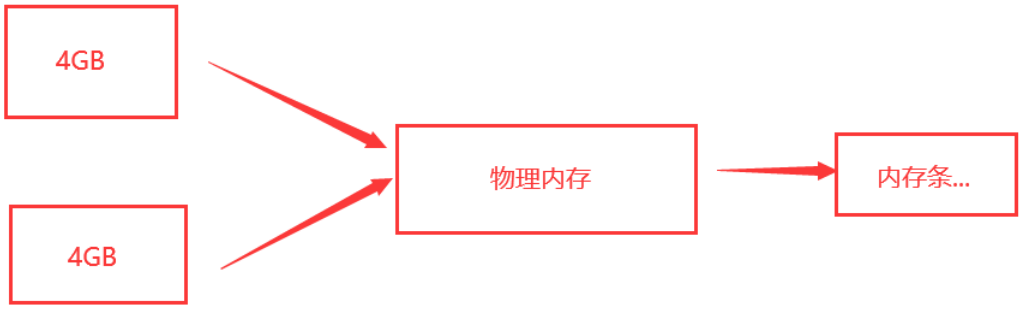
除了这些通用寄存器之外，那么其他的寄存器每一位都有自己特定的功能！

内存

寄存器很小，不够用。所以说，数据放到内存！

平时买的内存条！

每个应用程序进程都有4GB的内存空间，空头支票。



程序真正运行的时候，才会用到物理内存。

1B = 8bit

1KB = 1024B

1MB = 1024KB

1GB = 1024MB

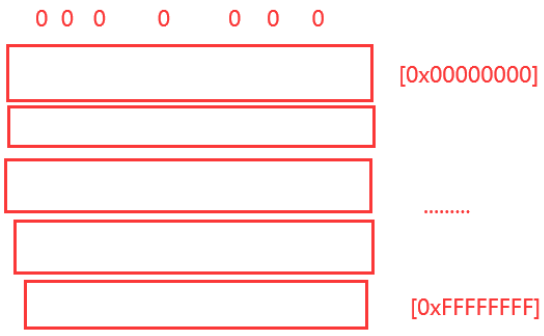
4G 的内存，4096m => 最终计算为位，就是这个可以存储的最大容量的。

计算机中内存地址很多，空间很大。

内存地址

存一个数：占用的大小，数据宽度！ 存到哪里?

计算机中内存地址很多，空间很大， 每个空间分配一个地址，名字。



这些给内存起的编号，就是我们的内存地址。 32位 8个 16进制的值。

32位：寻址能力！4GB 。

FFFFFFFF+1 = 100000000，最大的值。

位是怎么限制内存大小的。

100000000 内存地址 * 8 = 位：800000000

转换为10进制/8；4,294,967,296 字节

按照规则/1024, 最终发现就是4GB!

64位，绰绰有余!

所以每个内存地址都有一个编号! 可以通过这些编号想里面存值。！

内存如何存值?

数据宽度：byte word dword

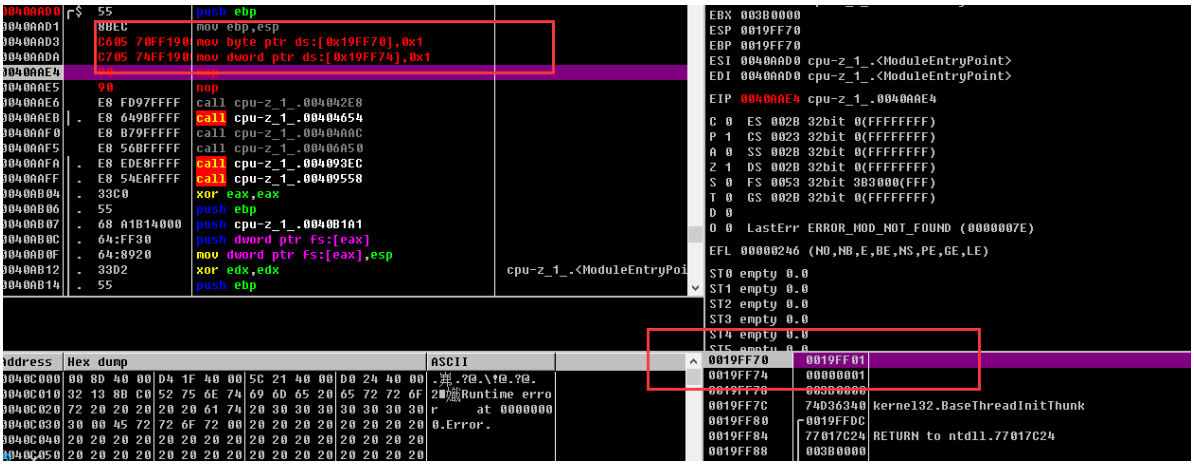
地址的位置：0xFFFFFFFF

不是任意的地址都可以写东西的，申请使用的。只有程序申请过的内存地址我们才可以使用。

```
汇编如何向内存中写值。
mov 数据宽度 内存地址，1

mov  byte ptr ds:[0x19FF70],1

传递的值的大小一定要和数据宽度相等。
```



内存地址有多种写法

ds:[0x19FF70+4] 内存地址偏移

ds:[eax] 寄存器

ds:[eax+4] 寄存器偏移

数组 []

ds:[reg+reg*{1,2,4,8}] 数组!

ds:[reg+reg*{1,2,4,8}+4] 偏移 !