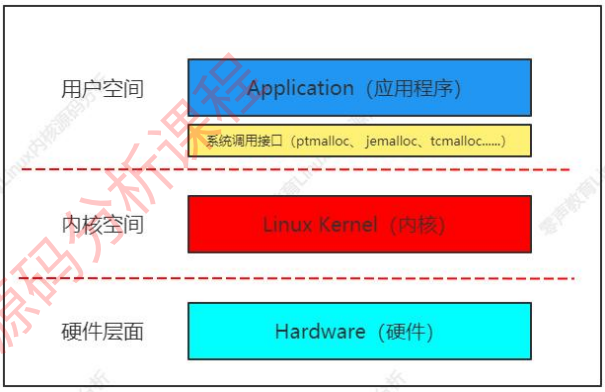
第 0076 讲 1 汇编基础与寻址方式



一、汇编基础部分

1、汇编语言是软件开发工程师信息处理全部为物理级数据，必须熟练掌握检测内存部分和相关的寄存器。计算机内部描述为二进制数据，也有十进制数及十六进制数（每种数制都有自己的基数），各种数字格式的掌握有助于快速进行数字的格式转换。

进程名称	进程基数	数字表示范围
二进制数	2	01
十进制数	10	0123456789
八进制数	8	01234567
十六进制数	16	0123456789ABCDEF

2、x86-64 处理器指令集的 64 位基本架构为 Intel64/AMD64。具有特性：

- 向后兼容 x86 指令集；

- 地址长度 64 位，虚拟地址空间为 2 的 64 个字节；
- 使用 64 位通用寄存器；
- 比 x86-32 多 8 个通用寄存器；
- 物理地址为 48 位，高达 256TB 的 RAM。

3、64 位通用寄存器仍可访问 8 位、16 位、32 位和 64 位数据：

操作数	可以访问的寄存器
8 位	AL,BL,CL,DL,DIL,SIL,BPL,SPL,R8L,R9L.....还有后缀把 L 改为 R
16 位	AX,BX,CX,DX,DI,SI,BP,SP,R8W,R9W,R10W,R11W,R12W,R13W,R14W,R15W
32 位	EAX,EBX,ECX,EDX,EDI,ESI,EBP,ESP,R8D,R9D,R10D,R11D,R12D,R13D,R14D,R15D
64 位	RAX,RBX,RCX,RDX,RDI,RSI,RBP,RSP,R8,R9,R10,R11,R12,R13,R14,R15

4、汇编语言基础知识

A.汇编语言程序

程序功能为执行两个数字相加，将结果存储至寄存器当中，程序名称为 TwoNumAddd（源码）：

```
main PROC
    mov eax, 9                  ;将数字 9 送入 eax 寄存器
    add eax, 8                  ;eax 寄存器加 8
    INVOKE ExitProcess, 0       ;程序结束
main ENDP
```

我们也可以为程序添加变量，将计算结果存储到 tsum 当中，所以我们需要增加一些标记、或者声明，用来标识程序的代码和数据区操作：

```
.data                          ;此为数据区
tsum DWORD 0                   ;定义名为 tsum 的变量

.code                          ;此为代码区
main PROC
    mov eax,9                  ;将数字 9 送入而 eax 寄存器
    add eax,8                  ;eax 寄存器加 8
    mox sum,eax
    INVOKE ExitProcess,0       ;结束程序
main ENDP
```

B.汇编语言常量

∑ 整型常量（又称整数常量）由一个可选前置符号、一个或多个数字，以及一个指明其基数的可选基数字符构成。是一种算术表达式，它包含了整数常量和算术运算符。

∑ 实型常量（又称为浮点数常量）用于表示十进制实数和编码（十六进制）实数。

∑ 字符及字符串常量：字符常量是指：用单引号或双引号包含的一个字符。字符串常量是指用单引号或双引号包含的一个字符（ 含空格符 ）序列组成。

C.保留字

汇编语言保留字没有大小写之分，比如 Mov 和 MOV 为同等。

保留字有不同类型：

- 指令助记符：比如 MOV、ADD 和 MUL。
- 寄存器名称：比如 AX、BX、CS 等等。
- 伪指令：告诉汇编器如何汇编程序（比如：segment、DB）。
- 属性：提供变量和操作数的大小与使用信息。例如 BYTE 和 WORD。
- 运算符：在常量表达式中使用。
- 预定义符号：比如 @data，它在汇编时返回常量的整数值。

常用保留字如下：

$	PARITY?	DWORD	STDCALL
?	PASCAL	FAR	SWORD
@B	QWORD	FAR16	SYSCAL
@F	REAL4	FWORD	TBYTE
ADDR	REAL8	FORTRAN	VARARG
BASIC	REAL10	NEAR	WORD
BYTE	SBYTE	NEAR16	ZERO?
C	SDORD	OVERFLOW ?	SIGN ?
CARRY?			

标识符及命名规则

标识符（identifier）是由程序员选择的名称，它用于标识变量、常数、子程序和代码标签。

标识符书写规则

- 标识符不能与汇编器保留字相同
- 不区分大小写
- 第一个字符必须为字母（A---Z, a---z）、下划线（_）、@、？或 $。其后的字符也可以是数字。
- 从第二个字符开始，可以是字母、数字、@、”或问号 ？

一些命名良好的名称 例：

lineCount firstValue index line_count

下面的名称合法，但是不可取：

_lineCount $first @myFile;

应避免用符号 @ 和下划线 _ 作为第一个字符。因为它们既用于汇编器，也用于高级语言编译器。

D.EQU 指令及 TEXTEQU 伪指令

EQU 伪指令把一个符号名称与一个整数表达式或一个任意文本连接起来，它有 3 种格式：

- name EQU expression
- name EQU symbol
- name EQU <text>

TEXTEQU 伪指令，类似于 EQU，创建文本宏（text macro）。它有 3 种格式：

- name TEXTEQU <text>
- name TEXTEQU textmacro
- name TEXTEQU %constExpr

二、常用寻址方式

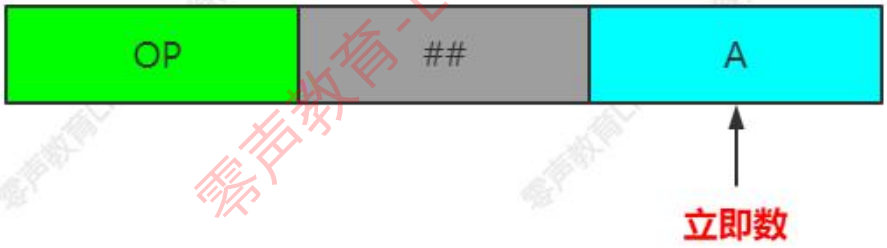
1、立即寻址：指令当中直接给出相应的操作数。在这种方式中，指令的操作数部分就是指令的操作数，而不是操作数的地址。

优点：取指同时取得操作数，提高指令的运行速度。

缺点：操作数的长度受指令长度的影响，且不便修改。适合操作数固定的情况。

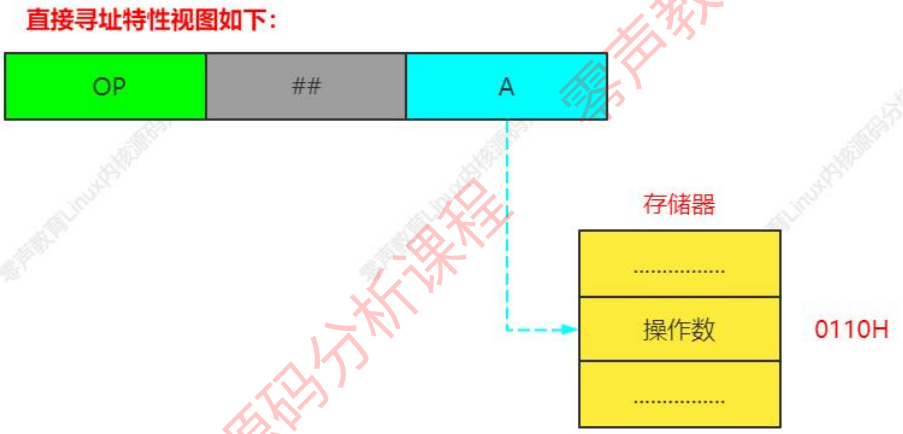
比如：MOV AX,2AH　// 2AH 就是使用立即寻址。

立即寻址特性视图如下：



2、直接寻址（寄存器直接寻址）：指令的操作数部分给出的就是操作数在存储器中的地址。特点是简单直观，便于硬件实现，但操作数地址是指令的一部分，只能用于访问固定的存储　器单元。

比如：　MOV AX，[0110H]

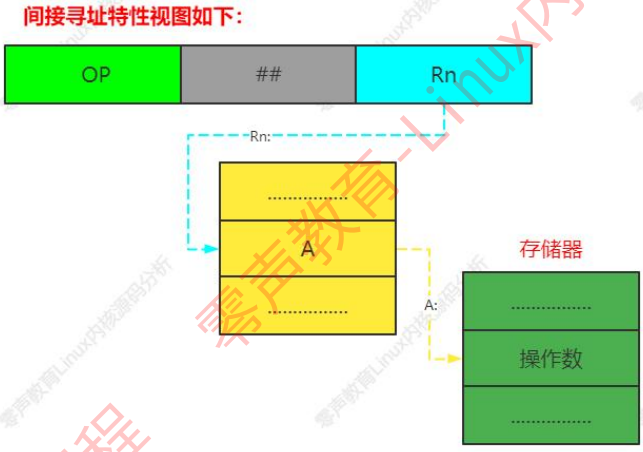


3、间接寻址（寄存器间接寻址）：在指令的操作数部分直接给出的既不是操作数也不是操作数的地址，而是操作数地址的地址。

优点：改变寄存器 Rn 中的内容就可访问内存的不同地址。修改十分方便。

缺点：二次寻址速度慢。

比如：MOV AX,[BX]



4、变址寻址：把 CPU 中变址寄存器的内容和指令操作数部分给出的地址之和作为操作数的地址来获得操作数。这种方式多用于字串处理、矩阵运算和成批数据处理。

5、相对寻址：把程序计数器 PC 的内容(即当前执行指令的地址)与指令的地址码部分给出的位移量(disp)之和作为操作数的地址或转移地址，称为相对寻址。

位移量可正可负，通常用补码表示。

相对寻址方式主要应用于相对转移指令。由于目的地址随 PC 变化不固定，所以非常适用于浮动程序的装配与运行。

6、基址寻址：在计算机中设置了一个专用的基址寄存器，操作数存储单元的实际有效地址就等于基址寄存器的内容与指令操作数部分给出的地址之和。改变基址寄存器的内容(基准量)并由指令提供位移量就可以访问存储器的任一单元。基址寄存器用于程序装配可为浮动程序分配存储单元。