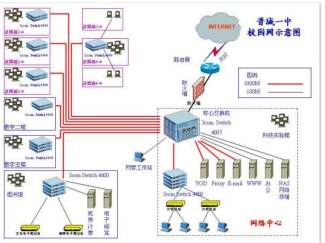
O语言

播报 编辑 讨论 上传视频

计算机语言术语

O语言是一款中间计算机语言（或称套装），它具有传统汇编语言的基本特点，也有与它们诸多不同之处。窗口设计、界面描述语言、O中间语言已经能很好的整合在一起（1.0.2.2版），基本有VB的事件驱动和窗口设计功能（但还不能创建COM控件，自带控件也很少）。

中文名	O语言	外文名	O language
		类　型	计算机语言



O语言的概述图（1张）

目录

简述

播报 编辑

O语言是一款中文计算机语言（或称套装），各部分开发状态：

O汇编语言（简称OASM，√）

O中间语言（简称OML，√）

O高级语言（简称OXX，×）

界面描述语言（简称OFL，√）

窗口设计（√）

窗口设计、界面描述语言、O中间语言已经能很好的整合在一起（1.0.2.2版），基本有VB的事件驱动和窗口设计功能（但还不能创建COM控件，自带控件也很少）。

O汇编语言

顾名思义O汇编语言也是一门汇编语言，它具有传统汇编语言的基本特点，也有与它们诸多不同之处。O汇编语言一个最大显著的特点是支持语言配置，使得它可以支持所有你想支持的语言，当然，O汇编语言的初衷是为了支持中文，所以它可以非常好的支持中文汇编。如果你已经习惯了其它英文的汇编模式，比如你喜欢用EAX、EBP类似这样的方式来命名寄存器，不要紧，只需要修改一下语言配置文件，这可以轻松做到。O汇编语言另一个显著的特点是指令使用非常直观和人性化，在不缺失汇编语言灵活性的情况下，使汇编语句的语意可以很直观地表现出来，这主要得益于用了一些象征性的符号，使得汇编语句不再是千篇一律的 (指令 寄存器，内存操作数) 这样的格式，而是更像 (寄存器 操作符 内存操作数)这样的格式，不但容易理解，而且便于记忆，使汇编语言不再那么枯燥，使人一团雾水。

当然，在这里我还是着重介绍怎么用中文来编写程序，汇编语言通常是分段的，O汇编语言也是如此，下面是O汇编语言的基本格式：

.位模式 32 ‖指明是16位、32位还是64位的代码

文本编写方式 符号编写方式

‖这里包含一些头文件和库的引用

.包含文< 文件名 > 《 》

.引用库< 库文件 > 〈 〉

‖数据段

.数据段 ┍

{

} ┕

‖只读段

词条统计

浏览次数：61168次

编辑次数：26次历史版本

最近更新：1164295378（2023-05-25）

突出贡献榜

linwenbin09

.只读段 ﹁

{

} ﹂

‖外部引用段

.引用段 ︽

{

} ︾

‖代码段开始

.代码段 ︿

{

‖开始函数定义

∏主函数()

{

‖这里添加代码

}

} ‖代码段结束 ﹀

首先介绍注释方式，O汇编语言使用行注释与块注释方式，并分别支持两种符号编写方式。第一种支持C语言模式的注释方式，即：以"//"双左斜杠为行注释符，以"/* */"作为块注释符。第二种方式是以'‖'这个符号作为行注释的开始，以"「 」"这两个符号作为块注释符。在一行中只要遇到行注释符，这行内以这个符号之后的文字都被忽略，在两个注释块符号之间的所有文字也都被忽略。

// 这行内这之后的文字被忽略

/*

这些文字被忽略

*/

‖ 这行内这之后的文字被忽略

「

这些文字被忽略

」

O中间语言

O中间语言可以说是汇编语言的抽象，它和汇编语言一样，使用单句的语法，除了基本的条件句和函数调用外，基本的一条指令对应一条语句，因此，它比C语言在语法上更低级一些。这样设计的目的是为了保持底层足够大的灵活性，使前端代码比较容易地映射到中间语言。C语言毋庸置疑是很强大，Pascal语言也非常强大，但是你很难将两者代码进行相互转换，如果使用中间语言作为中间层，就能够兼容两者的语法。

O中间语言使用了挂载技术，使前端代码的解析与后端代码生成操作进行了分离，使用这一技术只须扩展相应的前端语法就能支持多种语法。所有挂载的接口都封装在OMount.dll的动态链接库中。

与O汇编语言相比之下的特色：

1.去除了汇编中段的概念（方便调用API，仍保留"引用段"）。

2.增加了控制台和动态链接库的创建。

3.增加数组支持。

4.增加for语句，格式：设(;;;){}

5.增加do...while语句，格式：执行{}当()

6.增进循环语句。

7.增加"字符"变量。

8.去除了汇编语言中的寄存器概念。

9.增进了语法，"如果"也可以写成"若"等。

10.支持英文语句，如"函数"可写为function，如果写为if，否则写为else，.包含文写为.include等等。

11.可以直接支持masm32的宏定义。

12.可以直接使用O汇编的函数，须在引用段进行调用，调用方式为API的调用方式。

13.增加了指针支持。

语法基础

播报 编辑

注释文本

O中间语言的注释与C语言注释文本方式相同，行注释为两个英文右斜杠 // 块注释为 /* */

//这是行注释

/*这是块注释

…

*/

运算指令

中间语言共实现了下列格式的指令

变量A = 变量B 赋值指令

变量A += 变量B 加法指令

变量A　++ 自加1指令

变量A －= 变量B 减法指令

变量A －－ 自减1指令

变量A *= 变量B 乘法指令

变量A /= 变量B 除法指令

变量A %= 变量B 求模指令

变量A &= 变量B 与操作指令

变量A |= 变量B 或操作指令

变量A ^= 变量B　异或操作指令

变量A @= 变量B　取地址指令

变量A >>= 变量B　位右移指令

变量A <<= 变量B　位左移指令

变量A~ 求反指令

变量A <=> 变量B 互换指令

其中：

变量A可以是8位、16位和32位数据变量

变量B既可以是8位、16位和32位数据变量，也可以用立即数代替(除互换指令外)，比如：

变量A += 0×1234

条件语句

下面介绍的条件语句中的<条件>可以是：

(变量A == 变量B)

(变量A != 变量B)

(变量A > 变量B)

(变量A < 变量B)

(变量A >= 变量B)

(变量A <= 变量B)

(变量A !> 变量B)

(变量A !< 变量B)

多个条件可以用 或者符号|| 并且符号&& 进行联接，比如：

(((变量A == 变量B) || (变量C > 变量D)) && (变量E <= 变量F))

如果语句如果(<条件>)

{

//语句块

}

类似C/C++的if语句。

如果条件成立就执行语句块，也可以写成：

若(<条件>)

{

//语句块

}

否则如果语句否则如果(<条件>)

{

//语句块

}

类似C/C++的else if语句。

如果条件成立就执行语句块，必须与前一个语句为“如果”语句或“否则如果”语句匹配使用，也可以写成：

又若(<条件>)

{

//语句块

}

否则语句否则

{

//语句块

}

类似C/C++的else语句。

必须与前一个语句为如果语句或否则如果语句匹配使用，也可以写

成：

则

{

//语句块

}

循环语句循环(<条件>)

{

//语句块

跳出; //break

继续; //continue

}

类似C/C++的while语句。

如果条件成立则执行{}中的代码，再判断条件，满足则继续执行语句块，直到不满足才跳出。循环中可以在任何地方使用“跳出;”来跳出循环（不管是否满足）。也可以使用“继续;”语句直接进入下一次条件判断。

执行…当语句执行

{

//语句块

跳出; //break

继续; //continue

}当(<条件>)

类似C/C++的do…while语句。

与循环语句相似，只是判断条件在语句块之后，也就是说语句块至少会被执行一次。

设语句设(语句1;<条件>;语句3)

{

//语句块

跳出; //break

继续; //continue

}

类似C/C++的for循环。

整数 i,j=0;

设(i=0;i<=100;i++)

{

j++;

}

i和j将循环100次

先执行语句1，再判断条件语句，满足，执行语句3，再执行{}中的语句，再判断（注意不再执行语句1），满足，继续……如此循环，直到不满足语句2为止。

标签与跳转

标签 标签A：

跳到 标签A;

跳到语句与C语言的goto语言作用相同，在函数内直接跳转到标签所定义的位置执行。

返回语句

函数可以有多个返回值，这必须在函数定：

有多个函数值时必须使用括号()

函数 函数名(整数32,整数32)=>(整数32,整数32,结构体名)

{

//函数体

返回(0×1234,变量A,结构A);

}

要接收返回值

如果只有一个返回值可以用

变量A = 函数名(1,2); 或者 函数名(1,2)=>变量A;

如果有多个返回值则必须用下列格式：

函数名(1,2)=>(变量A,变量B);

语言示例

播报 编辑

示例1:

《*.\oasm32.oah》//包含头文件

〈*.\user32.lib〉//引用系统API，user32.dll

︽ //引用段

⌠信息框:MessageBoxA(双字,双字,双字,双字)//定义一个函数，MessageBoxA

︾ //引用段结束

︵ //代码段

∏主函数()//定义程序入口

{

信息框(0,&"Hello world",&"你好世界",4) //使用API MessageBoxA

}

⌣ //代码段结束

//这段代码虽然可以显示信息框，但是必须手动结束进程。

示例2：

《*.\oasm32.oah》

〈*.\user32.lib〉

〈*.\kernel32.lib〉

⌒

∫信息框:MessageBoxA(双字,双字,双字,双字)

∫ExitProcess(双字)

⌒

⌢

∏主函数()

{

信息框(0,&"Hello world",&"你好世界",4)

ExitProcess(0)

}

⌣

//这段代码可以运行完毕后自动结束进程。

窗口示例

.包含文<*oasm32.oah>

.引用库<*Kernel32.lib>

.引用库<*user32.lib>

结构 窗体结构

{

双字 结构大小

双字 窗口风格

双字 窗口过程

双字 类名附加空间

双字 窗口附加空间

双字 实例句柄

双字 图标句柄

双字 光标句柄

双字 背景颜色

双字 菜单指针

双字 类名指针

双字 附加图标

}

结构 坐标

{

双字 X轴

双字 Y轴

}

结构 消息结构

{

双字 窗口句柄

双字 消息标识

双字 参数一

双字 参数二

双字 时间

坐标 位置

}

宏定义

{

垂直重绘 0x1

水平重绘 0x2

背景色 0x6

图标形状 32512

鼠标形状 32512

边框样式 0x76c66

标题栏 0x0C00000

系统菜单 0x80000

边界粗细 0x40000

最小化 0x20000

最大化 0x10000

默认坐标X 250

默认坐标Y 394

默认宽度 320

默认高度 185

显示方式 10

关闭消息 0x2

}

.只读段

{

字节 窗口类名.. = "Window"

字节 窗口标题.. = "This are OASM Window"

字节 提醒内容.. = "主窗口创建失败"

字节 提醒标题.. = "警告"

}

.引用段

{

函数 获取模块句柄:GetModuleHandleA(双字)

函数 载入图标:LoadIconA(双字,双字)

函数 载入光标:LoadCursorA(双字,双字)

函数 注册窗口类:RegisterClassExA(双字)

函数 创建窗口:CreateWindowExA(双字,双字,双字,双字,双字,双字,双字,双字,双字,双字,双字,双字)

函数 信息窗口:MessageBoxA(双字,双字,双字,双字)

函数 显示窗口:ShowWindow(双字,双字)

函数 更新窗口:UpdateWindow(双字)

函数 获取消息:GetMessageA(双字,双字,双字,双字)

```
函数 消息翻译:TranslateMessage(双字)
函数 消息调度:DispatchMessageA(双字)
函数 邮送结束消息:PostQuitMessage(双字)
函数 默认窗口过程:DefWindowProcA(双字,双字,双字,双字)
函数 退出进程:ExitProcess(双字)
}
.代码段
{
入口 主函数()
{
主窗口()
退出进程(0)
}
函数 主窗口()
{
双字 模块句柄
获取模块句柄(0)
模块句柄 = 累加32
窗体结构 我的窗口类
我的窗口类.结构大小 = 取大小 我的窗口类
我的窗口类.窗口风格 = 垂直重绘 | 水平重绘
我的窗口类.窗口过程 = 消息处理程序
我的窗口类.类名附加空间 = 0
我的窗口类.窗口附加空间 = 0
压栈 模块句柄
出栈 我的窗口类.实例句柄
我的窗口类.背景颜色 = 背景色
我的窗口类.菜单指针 = 0
我的窗口类.类名指针 = 取地址 窗口类名
载入图标(0,图标形状)
我的窗口类.图标句柄 = 累加32
我的窗口类.附加图标 = 累加32
载入光标(0,鼠标形状)
我的窗口类.光标句柄 = 累加32
累加32 @= 我的窗口类
注册窗口类(累加32)
双字 主窗口句柄
累加32 = 边框样式 | 标题栏 | 系统菜单 | 边界粗细 | 最小化 | 最大化
创建窗口(0,取地址 窗口类名,取地址 窗口标题,累加32,默认坐标X,默认坐标Y,默认宽度,默认高度,0,0,模块句柄,0)
主窗口句柄 = 累加32
如果(主窗口句柄 == 0)
{
信息窗口(0,取地址 提醒内容,取地址 提醒标题,16)
返回 0
}
```

```
显示窗口(主窗口句柄,显示方式)
更新窗口(主窗口句柄)
消息结构 消息
循环(真)
{
累加32 @= 消息
获取消息(累加32,0,0,0)
如果(累加32==0)
{
跳出
}
累加32 @= 消息
消息翻译(累加32)
累加32 @= 消息
消息调度(累加32)
}
}
函数 消息处理程序(双字 句柄,双字 消息,双字 消息参数一,双字 消息参数二)
{<基数32,基址32,源址32>
累加32 = 消息
如果(累加32 == 关闭消息)
{
邮送结束消息(0)
}
否则
{
默认窗口过程(句柄,消息,消息参数一,消息参数二)
返回
}
累加32^=累加32
}
}
//可以显示一个窗口
```

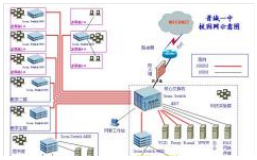
中间语言

播报 编辑

```
Hello,World!示例
.包含文<*视窗32.omh>
入口 主函数()
{
MessageBox(0,&"Hello,World!",&"",0);
ExitProcess(0);
}
```

词条图册

更多图册 >



相关搜索

编程语言有哪些	电脑编程语言	r语言	黄金 交易	一千块的手机哪款好用
干衣机哪个牌子好	专利申请	蓝牙下载安装	烙饼机	凉意在哪直播