课程导学

汇编课对于学生来说是一门全新的课程，尽管在此之前他们大多学过一门或两门高级语言，有一定的编程经历和程序设计能力。但汇编这种低级语言比较独特，从形式到结构与高级语言有很大不同，它不像高级语言那样容易“入门”，学生在学习过程中尤其在课程的前半段充满了疑惑，几乎每一章都有难点。

遇到的难点主要分两种：一是基本概念方面的，对这些概念和基础知识正确而清晰的理解非常重要；二是应用和编程方面的（主要体现在程序格式或结构上），下面分别加以说明：

（1）基本概念方面的难点及解决办法

● 第 1 章介绍基础知识，数的补码表示是这一章的主要难点。“补码”是绝大多数处理器采用的码制，它在表示机器数时，不及“原码”和“反码”直观，但‘0’的补码表示唯一；在算术运算中，补码减法还可以通过对减数求补转化为加法来进行，这样一套运算器就可以既实现加法又实现减法。

需要注意区分无符号数和有符号数的补码表示，强调有符号数的正负号需要用二进制数的最高有效位来“体现”。通过大量示例说明十进制数和补码表示的相互转换规则，。

● 第 2 章介绍 80x86 计算机组织，在这一章里我们开始提到寄存器、内存单元的地址和内容、实模式下段的概念、存储器的逻辑分段、物理地址、逻辑地址（段地址和偏移地址）等概念，学生往往不理解什么是“段”？为什么要“分段”？“物理”和“逻辑”有什么区别？“实模式”与“保护模式”是怎么回事？等等。

此时课程刚刚开始，还没有接触编程，没有感性认识，可以通过类比、图示、举例等通俗形式尽可能讲清楚这些问题，比如：8086/8088 之所以只能工作在“实模式”，是因为它的地址线是 20 根，而一个寄存器只有 16 位长，容纳不下 20 位的物理地址，必须使用两个寄存器；这两个寄存器一个放段地址，一个放偏移地址（二者构成逻辑地址），此时顺便讲解每个寄存器的专门用途，给出实模式下物理地址的计算公式；同时结合存储器中存放数据的不同，引出“逻辑分段”的概念。处理器发展到 80386 之后，寄存器达到 32 位，已经能够放下 32 位物理地址。但对于新增的保护工作模式而言，并不是简单地把物理地址放到寄存器中，此时存储器中的代码和数据为适应多任务的需求包含各自的“保护”信息，这些信息存放在“描述符”中，而“描述符”又构成“描述符表”保存在内存的某块区域。此时的逻辑地址由 16 位选择器（原来的段寄存器充当）和 32 位寄存器构成，代码和数据共用 4GB 的内存空间，已经不再有“段”的概念。

随着课程的深入，在学习完下一章“寻址方式”并经过一段时间的编程体验后，本章的难点基本上迎刃而解。

● 第 3 章（通过 C 语言对应的汇编）介绍 80x86 的指令系统和寻址方式，这一章篇幅大，内容多。“寻址方式”由于种类多、组合规则多、使用的灵活性大，所以初学时会感到比较繁杂，不好理解。为此，需要在重点理解每个寄存器的功能和作用以及存储单元地址与内容的关系之外，还通过大量示例的讲解和练习，搞清楚各种寻址方式的适用场合、所用寄

存器限制、扩展后寻址方式的新特征等，加深理解。在弄懂基本规则的基础上，通过课外作业练习和上机编程逐步攻克这一难点。

“指令系统”部分涉及近百条指令，会觉得汇编语言指令比高级语言限制多，不好记，因为有些指令隐含使用寄存器（比如乘法除法指令、串处理指令等），所以对指令的功能不明确，更不知如何使用。要求特别关注指令的汇编格式、指令的基本功能、指令支持的寻址方式、指令的执行对标志位的影响、指令的特殊要求等几方面关键问题，并做大量练习加深理解。同时要求尽早上机，尽管此时还没有讲到程序格式和伪操作部分，但在教师提供的程序框架中，可以先验证和查看指令的执行结果。

对于比较重要的指令，如堆栈操作的 PUSH POP 指令、子程序调用与返回的 CALL RET 指令等，它们往往一条指令对应多项操作，学生通过上机编程调试、查看内存和寄存器的值，可以更好地理解指令的执行过程。对于算术运算指令，要求重点掌握运算结果对标志位的影响，因为这是考查处理器状态的重要指标，有助于正确理解溢出、进位等概念。

另一难点是栈与过程调用的机器表示：用栈传递参数或参数地址是学生比较难掌握的，尤其是 EBP 寄存器的使用和带立即数的 RET 指令的使用。更进一步地，嵌套和递归子程序设计对学生而言也有一定难度。我们在平时练习和上机中加重了这方面的训练，在实验中也特别设计了这方面的内容，并要求学生在实验报告中分析子程序功能、参数传递机制、栈变化情况等。

涉及到 MIPS 汇编后，又会碰到概念难点：

- 异常处理程序设计

通过 SPIM 模拟器的实际单步运行，一步步的演示 MIPS 下异常处理的流程与相关寄存器的变化，给出直观的异常处理概念。

- MIPS 内存管理

学生基本上对于系统虚存没有概念。因此必须首先通过通俗的原理讲解与虚存访问流程示例来赋予基本概念，然后才能针对特定的 TLB 处理流程、相关指令进行详细介绍。

（2）编程应用方面的难点及解决办法

这部分难点主要体现在程序格式和结构方面，一方面是对汇编语言的编程不习惯（与高级语言相比），另一方面要接触一些硬件编程（之前未接触过），还有就是前期基础知识掌握不好带来的困难。

- 第 4 章介绍汇编语言程序格式，在遇到一些段定义、数据定义等伪操作时不理解，此时有些问题可以解释清楚（比如与高级语言进行类比），有些可以直接告诉结论。比如可能最不理解的是为何程序进入退出时为何要写如下的代码：

```
push    ds
mov     ax, 0
push    ax
……
ret
```

这就要涉及 PSP（程序段前缀）方面的内容，要搞明白，当程序装到内存准备执行时，

只有代码段寄存器的内容不用调整，此时数据段、附加段、堆栈段的段寄存器都另有它用，DS 和 ES 指向 PSP，而不是真正的数据段；把 DS 和 0 压入堆栈是为了在段间返回 ret 时把 DS 和 0 存入 CS 和 IP，因为 PSP 的第一条指令就是 int 20h，恰好返回操作系统。还需给学生指出的是伪指令是由汇编器在汇编时处理的指令，与上一章讲的“指令系统”中的指令不同，后者是真正在机器硬件上执行的指令。

- 第 5 章介绍循环与分支程序设计，从这章开始涉及编程方法与技巧，尽管所举例子可能在高级语言中也出现过，并不陌生，但这儿的困难在于对这种表现形式的不习惯，特别是转移指令、循环指令的用法。这儿需要理解的是高级语言的 for while if else 翻译过来也是一大堆这样的形式，直接用汇编这样写，会比高级语言效率更高；另外对于处理一些特定的问题如地址表、逻辑尺，汇编有其独特的优势。也会涉及一些经典算法如冒泡排序、折半查找，重点需要感受汇编与高级语言在表现形式上的相同与不同。

- 第 6 章介绍子程序结构，在介绍子程序的参数传递时，用堆栈传递参数或参数地址是学生比较难掌握的，尤其是 BP/EBP 寄存器的使用和带立即数的 RET 指令的使用。更进一步地，嵌套和递归子程序设计也有一定难度。在平时练习和上机中需加重这方面的训练，在实验中也特别设计了这方面的内容，并要求在实验报告中分析子程序功能、参数传递机制、堆栈变化情况等。