微机原理与汇编语言

1. 计算机语言

- 是人与计算机进行交流的工具
- 计算机语言分为：

机器语言	机器指令（由0 和 1组成），可直接执行	难学、难记 依赖机器的类型
汇编语言	用助记符代替机器指令，用变量代替各类地址。	克服记忆的难点 其他与机器语言类似
高级语言	类似数学语言、接近自然语言、具有通用性和可移植性，不依赖具体的计算机类型。	

不同程序设计语言编写的程序示例

- 机器语言程序

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

- 汇编语言源程序

```
MOV   AX, 300H
ADD    BX,  AX
MOV  [2100H], BX
HLT
```

- 高级语言源程序 (C)

```
main( )
{ int a, b, c;
   a=300;  b=18;
   c=a+b;
   printf(“ a+b= %d\n”, c);
}
```

源程序需要翻译

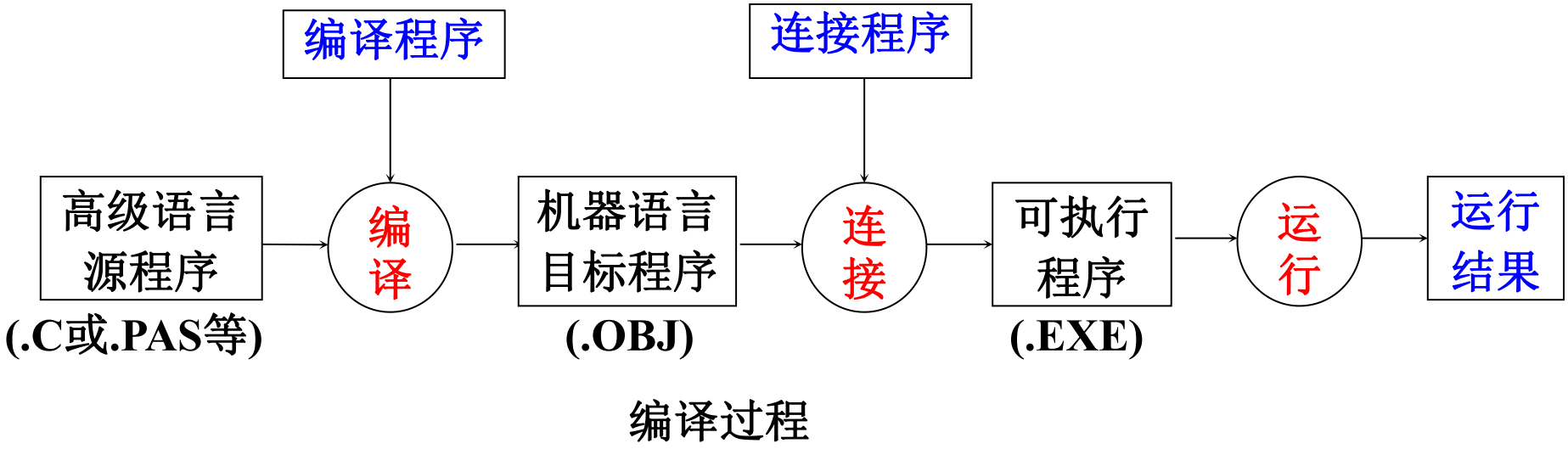
```
$P 00 70 ; START: IN R0,00H 从 IN 单元读入计数初值
$P 01 00
$P 02 A1 ; LDI R1,0FH 立即数 0FH 送 R1
$P 03 0F
$P 04 24 ; AND R0,R1 得到 R0 低四位
$P 05 A1 ; LDI R1,00H 装入和初值 00H
$P 06 00
$P 07 F0 ; BZC RESULT 计数值为 0 则跳转
$P 08 16
$P 09 A2 ; LDI R2,60H 读入数据始地址
$P 0A 60
$P 0B CB ; LOOP: LAD R3,[RI],00H 从 MEM 读入数据送 R3，变址寻址，偏移量为 00H
$P 0C 00
```

汇编语言语句

机器语言程序

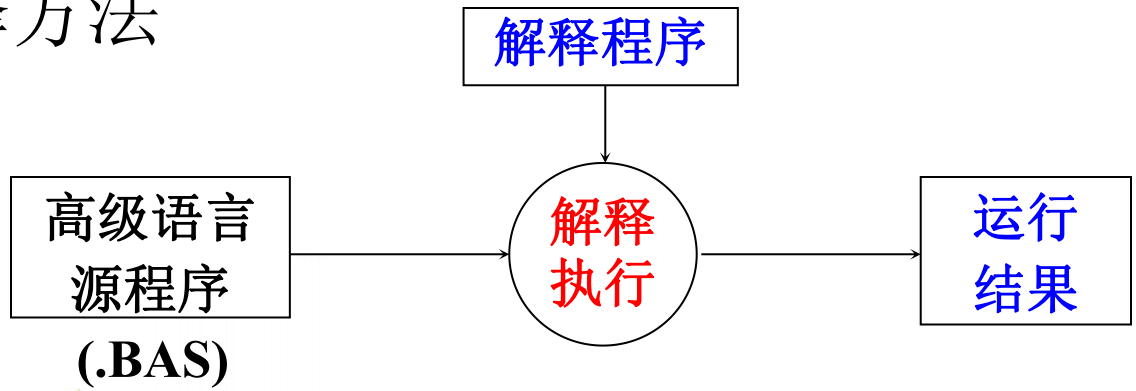
高级语言的翻译

- 编译方法



编译过程

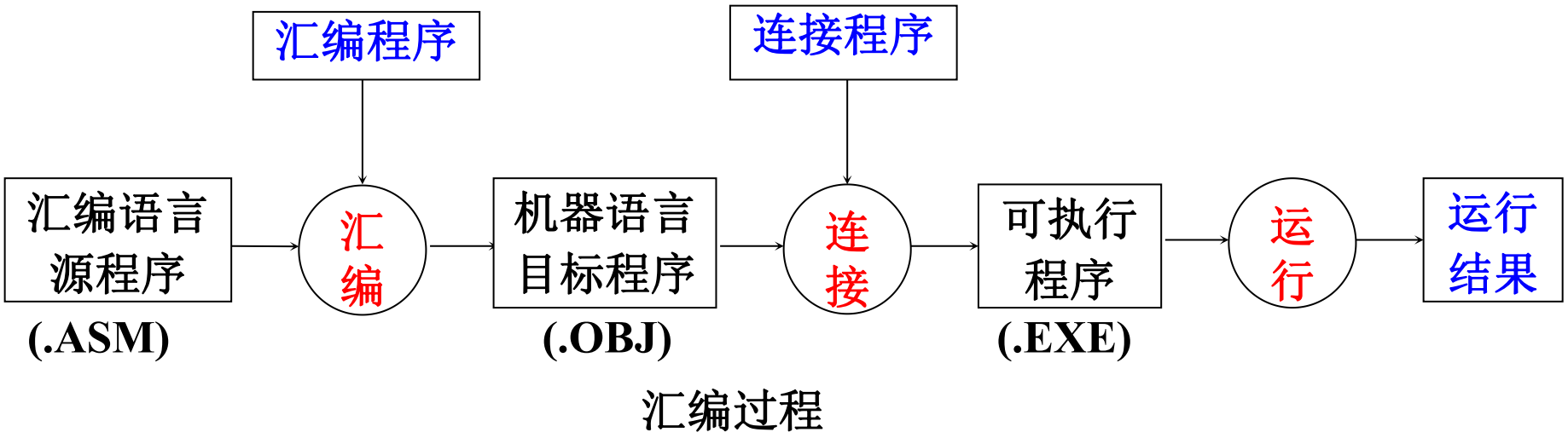
- 解释方法



解释过程

汇编源程序的翻译

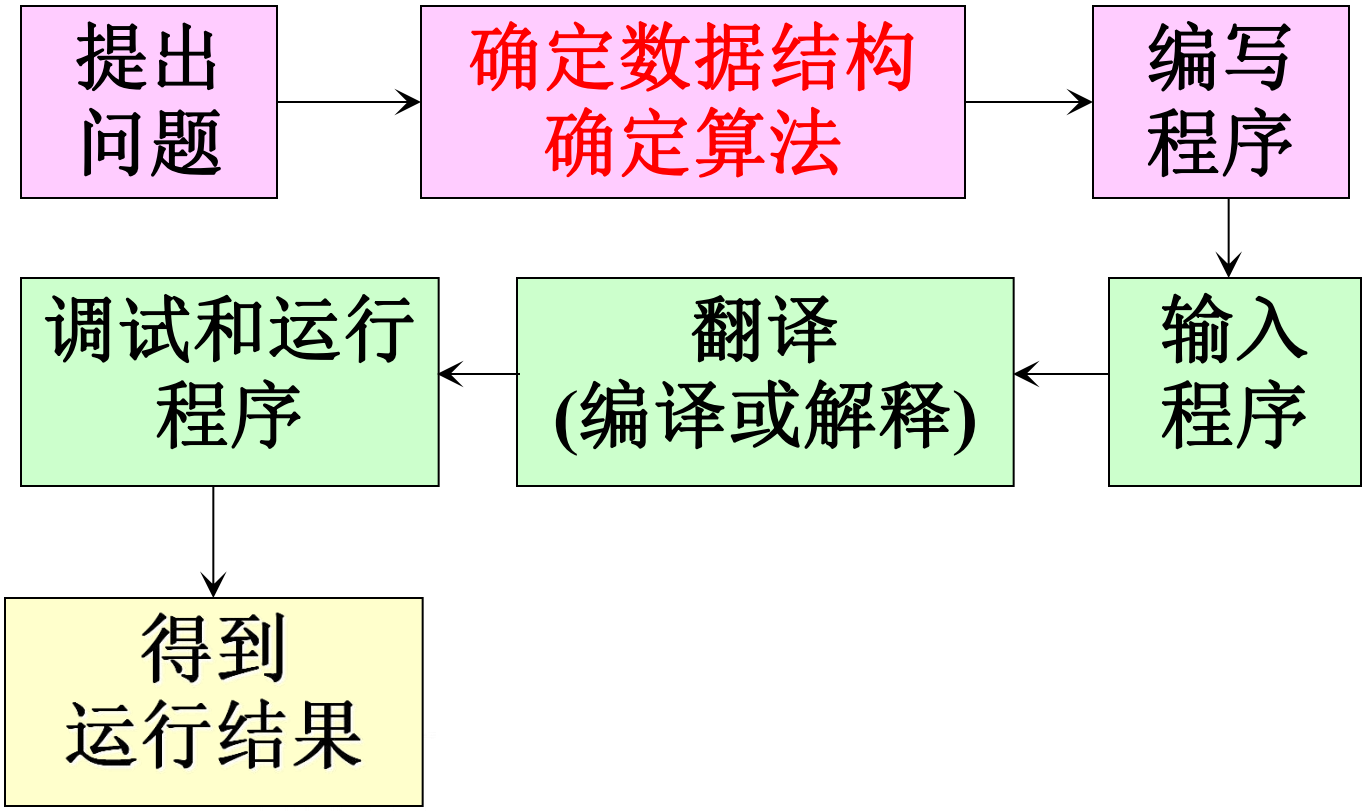
- 将汇编源程序翻译为目标程序的过程称为汇编
- 汇编过程：



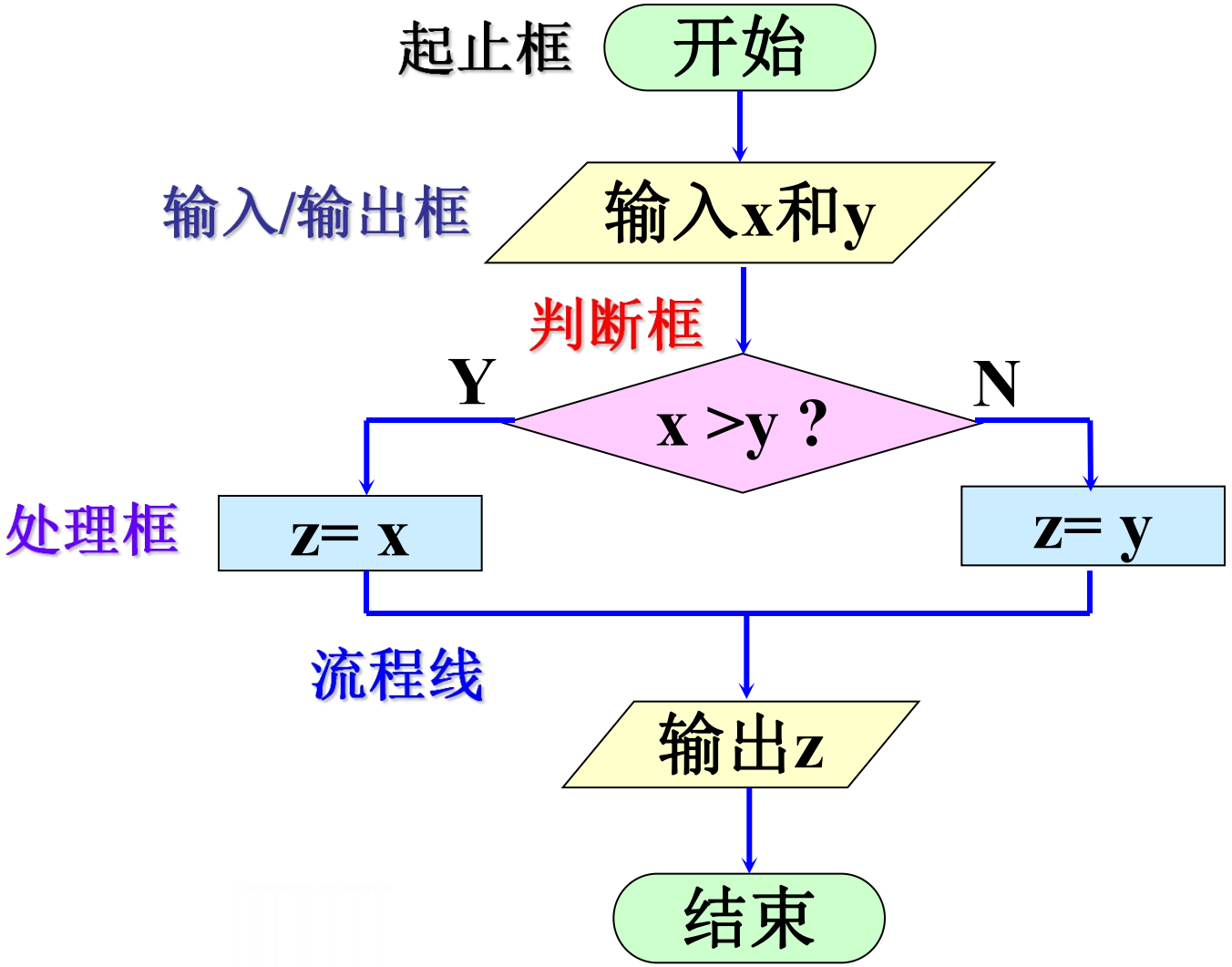
汇编过程

程序设计

- 用计算机解决一个实际应用问题时的整个处理过程称为程序设计



用流程图描述算法



用流程图描述算法

第一个汇编语言程序：Hello，world！

```
.model small ;定义程序的存储模式，small表示小型模式
.stack  2048    ;定义堆栈段，默认是1KB空间
.data           ;定义数据段
string     db 'Hello, world !', 0dh,0ah, '$'
.code
start:  mov ax,@data
        mov ds,ax

        mov dx,offset string    ;9号DOS功能调用，显示字符串
        mov ah,9
        int 21h

        mov ax,4c00h            ;系统功能调用，返回系统
        int 21h

        end start
```

汇编语言程序开发过程

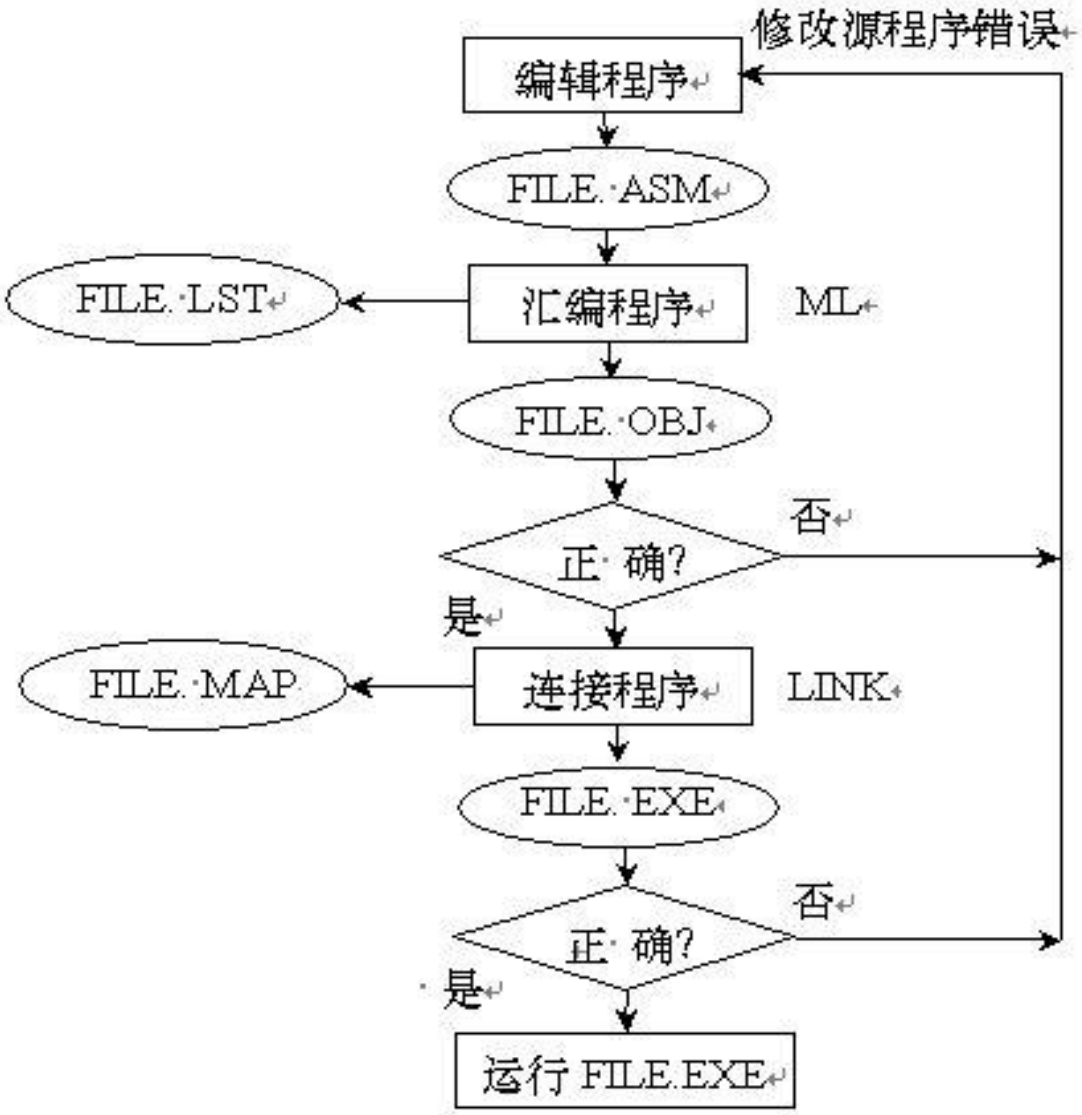


图 3.1 汇编语言程序调试上机过程

What's Wrong With Assembly Language

Assembly language has a pretty bad reputation. The common impression about assembly language programmers today is that they are all **hackers** or **misguided individuals** who need **enlightenment**. Here are the reasons people give for *not* using assembly language

- Assembly is hard to learn.
- Assembly is hard to read and understand.
- Assembly is hard to debug.
- Assembly is hard to maintain.
- Assembly is hard to write.
- Assembly language programming is time consuming.
- Improved compiler technology has eliminated the need for assembly language.
- Today, machines are so fast that we no longer need to use assembly. If you need more speed, you should use a better algorithm rather than switch to assembly language.
- Machines have so much memory today, saving space using assembly is not important.
- Assembly language is not portable.

of course, all of these points are wrong!!!

What's Right With Assembly Language?

An old joke goes something like this: "There are three reasons for using assembly language: speed, speed, and more speed." Even those who absolutely hate assembly language will admit that if speed is your primary concern, assembly language is the way to go. Assembly language has several benefits:

- **Speed**. Assembly language programs are generally the fastest programs around.
- **Space**. Assembly language programs are often the smallest.
- **Capability**. You can do things in assembly which are difficult or impossible in HLLs.
- **Knowledge**. Your knowledge of assembly language will help you write better programs, even when using HLLs.

Speed

Assembly language is the uncontested speed champion among programming languages. An expert assembly language programmer will almost always produce a faster program than an expert C programmer. While certain programs may not benefit much from implementation in assembly, you can speed up many programs by a factor of five or ten over their HLL counterparts by careful coding in assembly language; even greater improvement is possible if you're not using an optimizing compiler. Alas, speedups on the order of five to ten times are generally not achieved by beginning assembly language programmers. However, if you spend the time to learn assembly language really well, you too can achieve these impressive performance gains.

space

- **Despite some people's claims that programmers no longer have to worry about memory constraints, there are many programmers who need to write smaller programs.**
- **Assembly language programs are often less than *one-half the size* of comparable HLL programs.**
- **Saving space saves money. Pure and simple. If a program requires 1.5 megabytes, it will not fit on a 1.44 Mbyte floppy.**
- **Most users put more large memory in their machines so they can run *multiple* programs at one time. The bigger a program is, the fewer applications will be able to coexist in memory with it. Virtual memory isn't a particularly attractive solution either. With virtual memory, the bigger an application is, the slower the system will run as a result of that program's size.**

Capability

- Capability is another reason people resort to assembly language.
- HLLs are an abstraction of a typical machine architecture. They are designed to be independent of the particular machine architecture. As a result, they rarely take into account any special features of the machine. *If you want to use such features, you will need to use assembly language.* A really good example is the input/output instructions available on the 80x86 microprocessors. These instructions let you directly access certain I/O devices on the computer. In general, such access is not part of any high level language. In assembly language you have no such restrictions.
- Anything you can do on the machine you can do in assembly language. This is definitely *not* the case with most HLLs.

knowledge

Now some of you may be thinking, "Gee, that would be wonderful, but I've got lots to do. My time would be better spent writing code than learning assembly language." There are some practical reasons for learning assembly, even if you never intend to write *a single line of assembly code*.

- If you know assembly language well, you'll have an appreciation for the compiler, and you'll know exactly what the compiler is doing with all those HLL statements.

- Good assembly language programmers make better HLL programmers because they understand the limitations of the compiler and they know what it's doing with their code. Those who don't know assembly language will accept the poor performance their compiler produces and simply shrug it off.

Work hard on the assembly studying—you will get as much as you can imagine from it

- ✧ Yes, assembly language is definitely worth the effort.
- ✧ The only scary thing is that once you learn it really well, you'll probably start using it far more than you ever dreamed you would. That is a common malady among assembly language programmers. Seems they can't stand what the compilers are doing with their programs.