



南京大學
NANJING UNIVERSITY



程序开发和执行过程简介

南京大学

计算机科学与技术系

袁春风

email: cfyuan@nju.edu.cn

2015.6

最早的程序开发过程

- 用机器语言编写程序，并记录在纸带或卡片上



穿孔表示0，未穿孔表示1

输入：按钮、开关；所有信息都是0/1序列！
输出：指示灯等

假设：0010-jc

0：0101 0110

1：0010 0100

2：.....

3：.....

4：0110 0111

5：.....

6：.....



太原始了，无法忍受，咋办？

用符号表示而不用0/1表示！

若在第4条指令前加入指令，则需重新计算地址码（如jxx的目标地址），然后重新打孔。不灵活！

书写、阅读困难！

用汇编语言开发程序

- 若用**符号**表示跳转位置和变量位置，是否简化了问题？

- 于是，汇编语言出现
 - 用**助记符**表示操作码
 - 用**标号**表示位置
 - 用**助记符**表示寄存器
 -

0 : 0101 0110

1 : 0010 0100

2 :

3 :

4 : 0110 0111

5 :

6 :

7 :

add B

jc L0

.....

.....

L0 : sub C

.....

B :

C :

你认为用汇编语言编写的优点是：

不会因为增减指令而需要修改其他指令

不需记忆指令编码，编写方便

可读性比机器语言强

不过，这带来新的问题，是什么呢？

人容易了，可机器不认识这些指令了！

需将汇编语言转换为机器语言！


用汇编程序转换

在第4条指令前加指令时不用改变
add、jxx和sub指令中的地址码！

进一步认识机器级语言

- 汇编语言(源)程序由**汇编指令**构成
- 你能用一句话描述**什么是汇编指令**吗？
 - 用助记符和标号来表示的指令（与机器指令一一对应）
- **指令**又是什么呢？
 - 包含操作码和操作数或其地址码
（**机器指令**用**二进制**表示，**汇编指令**用**符号**表示）
 - 只能描述：取（或存一个数）
两个数加（或减、乘、除、与、或等）
根据运算结果判断是否转移执行
- 想象用**汇编语言**编写复杂程序是怎样的情形？
 - （例如，用汇编语言实现排序（sort）、矩阵相乘）
 - 需要描述的细节太多了！程序会很长很长！而且在不同结构的机器上就不能运行！

```
add B
jc L0
.....
.....
L0 : sub C
.....
B : .....
C : .....
```



机器语言和汇编语言都是面向机器结构的语言，故它们统称为**机器级语言**

SKIP

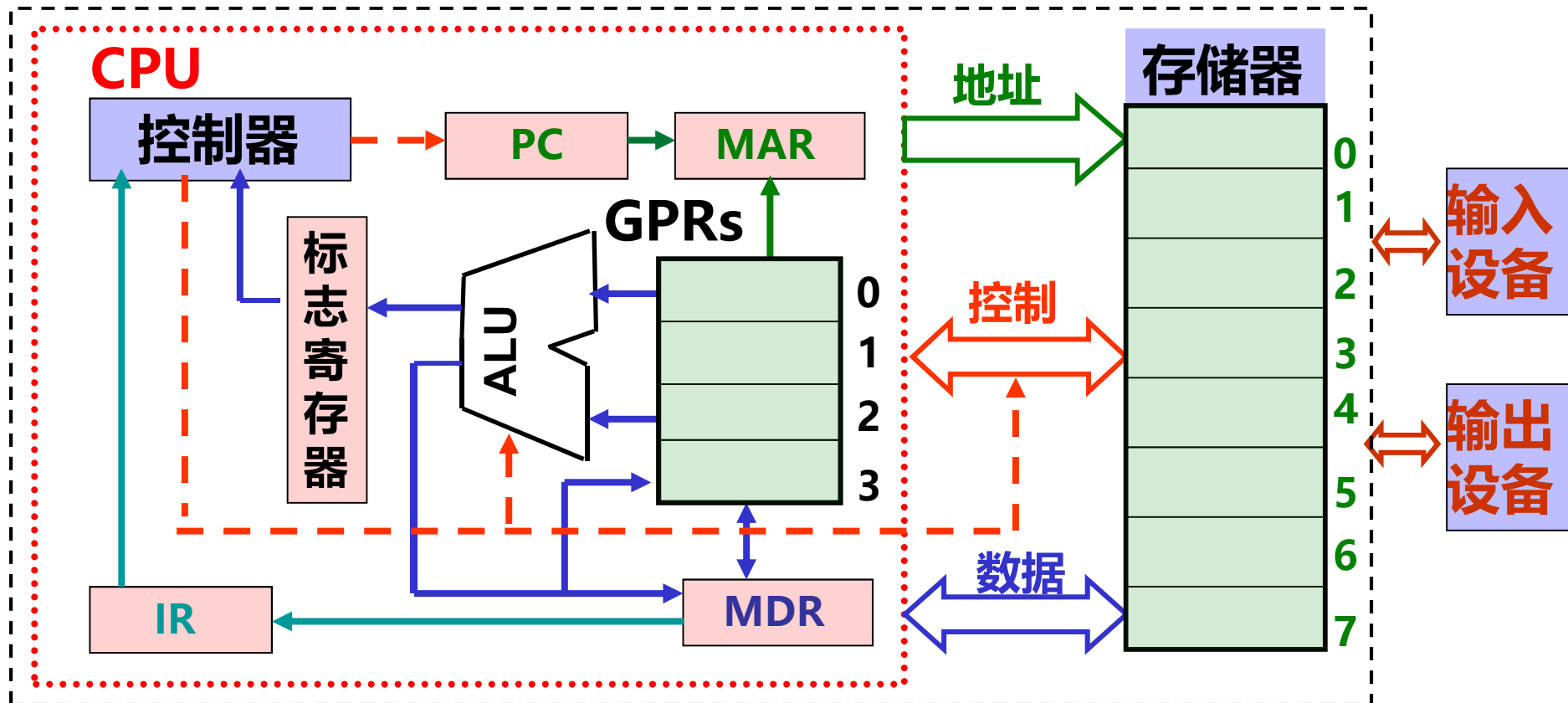
结论：用汇编语言比机器语言好，但是，还是很麻烦！

指令所能描述的功能

对于以下结构的机器，你能设计出几条指令吗？

[BACK](#)

Ld M# , R# (将存储单元内容装入寄存器)
St R# , M# (将寄存器内容装入存储单元)
Add R# , M# (类似的还有Sub , Mul等 ; 操作数还可 “R# , R#” 等)
Jxx M# (若满足条件 , 则转移到另一处执行)
.....



用高级语言开发程序

- 随着技术的发展，出现了许多高级编程语言
 - 它们与具体机器结构无关
 - 面向算法描述，比机器级语言描述能力强得多
 - 高级语言中一条语句对应几条、几十条甚至几百条指令
 - 有“面向过程”和“面向对象”的语言之分
 - 处理逻辑分为三种结构
 - 顺序结构、选择结构、循环结构
 - 有两种转换方式：“编译”和“解释”
 - 编译程序(Compiler)：将高级语言源程序转换为机器级目标程序，执行时只要启动目标程序即可
 - 解释程序(Interpreter)：将高级语言语句逐条翻译成机器指令并立即执行，不生成目标文件。
- 现在，几乎所有程序员都用高级语言编程，但最终要将高级语言转换为机器语言程序

一个典型程序的转换处理过程

经典的 “hello.c” C-源程序

```
#include <stdio.h>

int main()
{
    printf("hello, world\n");
}
```

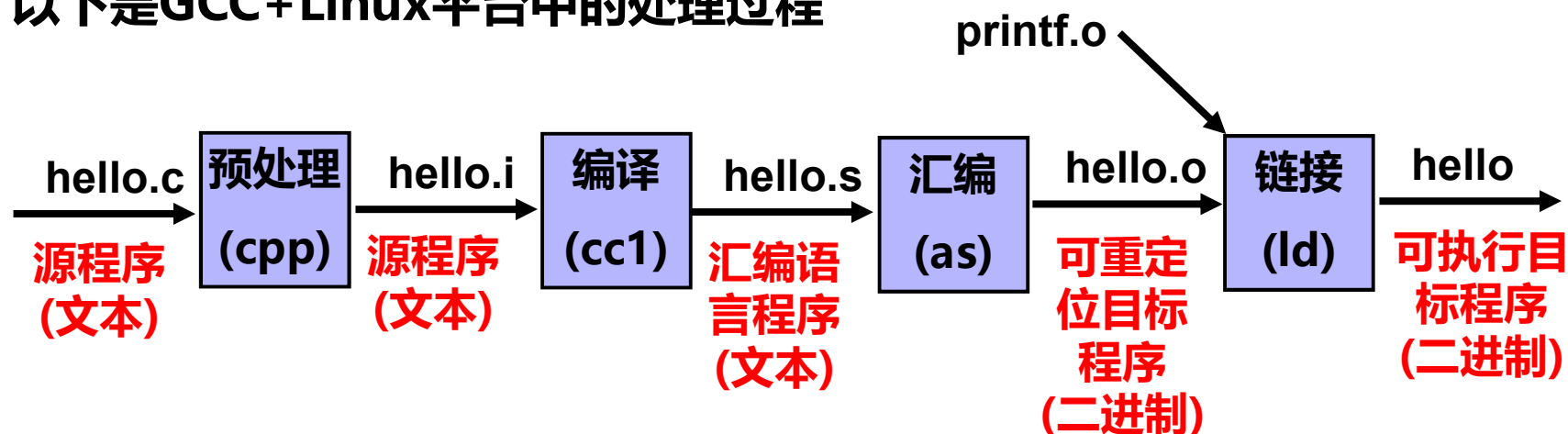
hello.c的ASCII文本表示

```
# i n c l u d e < s p > < s t d i o .
35 105 110 99 108 117 100 101 32 60 115 116 100 105 111 46
h > \n \n i n t < s p > m a i n ( ) \n {
104 62 10 10 105 110 116 32 109 97 105 110 40 41 10 123
\n < s p > < s p > < s p > < s p > p r i n t f ( " h e l
10 32 32 32 32 112 114 105 110 116 102 40 34 104 101 108
l o , < s p > w o r l d \n " ) ; \n }
108 111 44 32 119 111 114 108 100 92 110 34 41 59 10 125
```

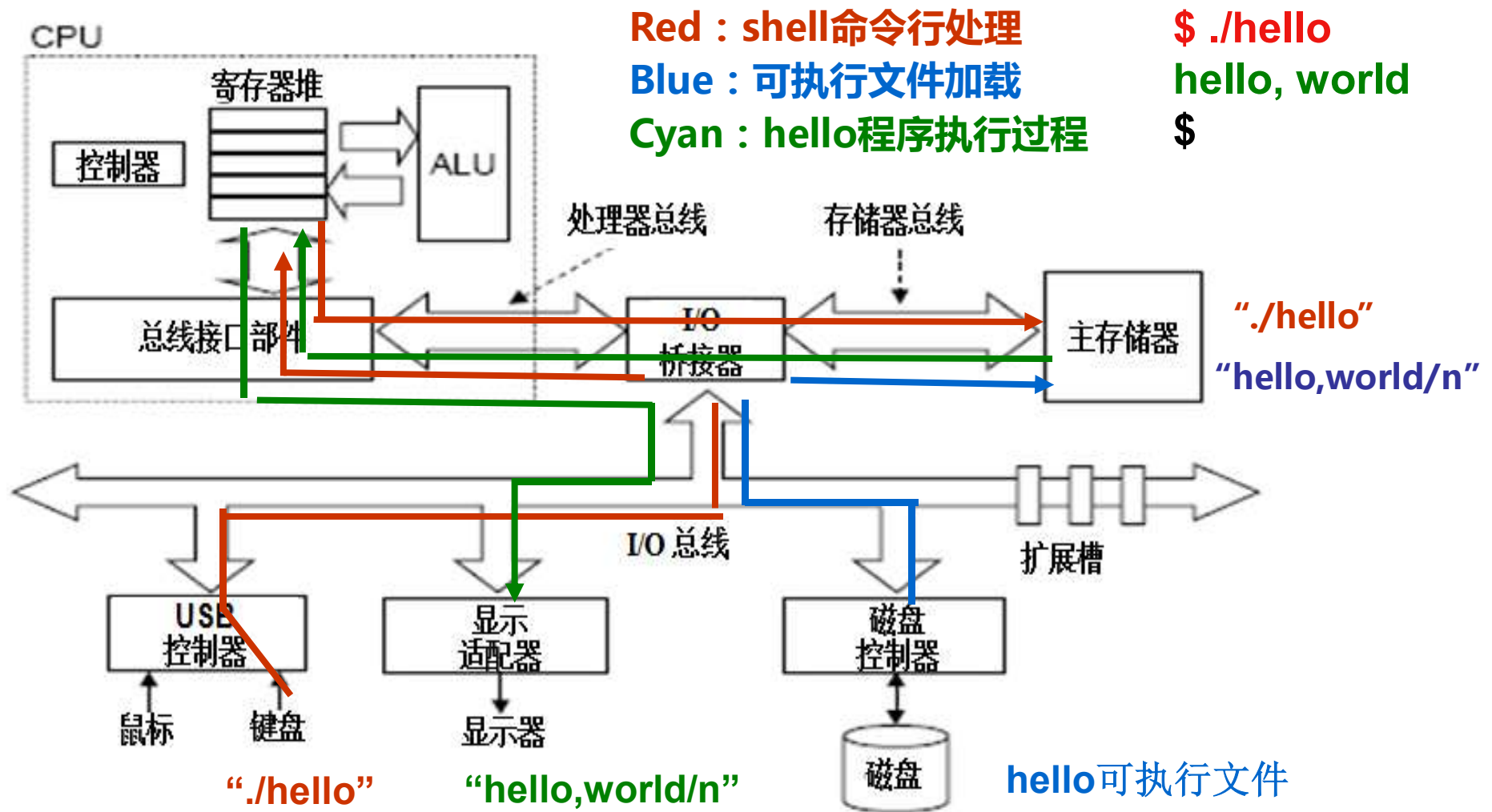
功能：输出 “hello,world”

计算机不能直接执行hello.c！

以下是GCC+Linux平台中的处理过程



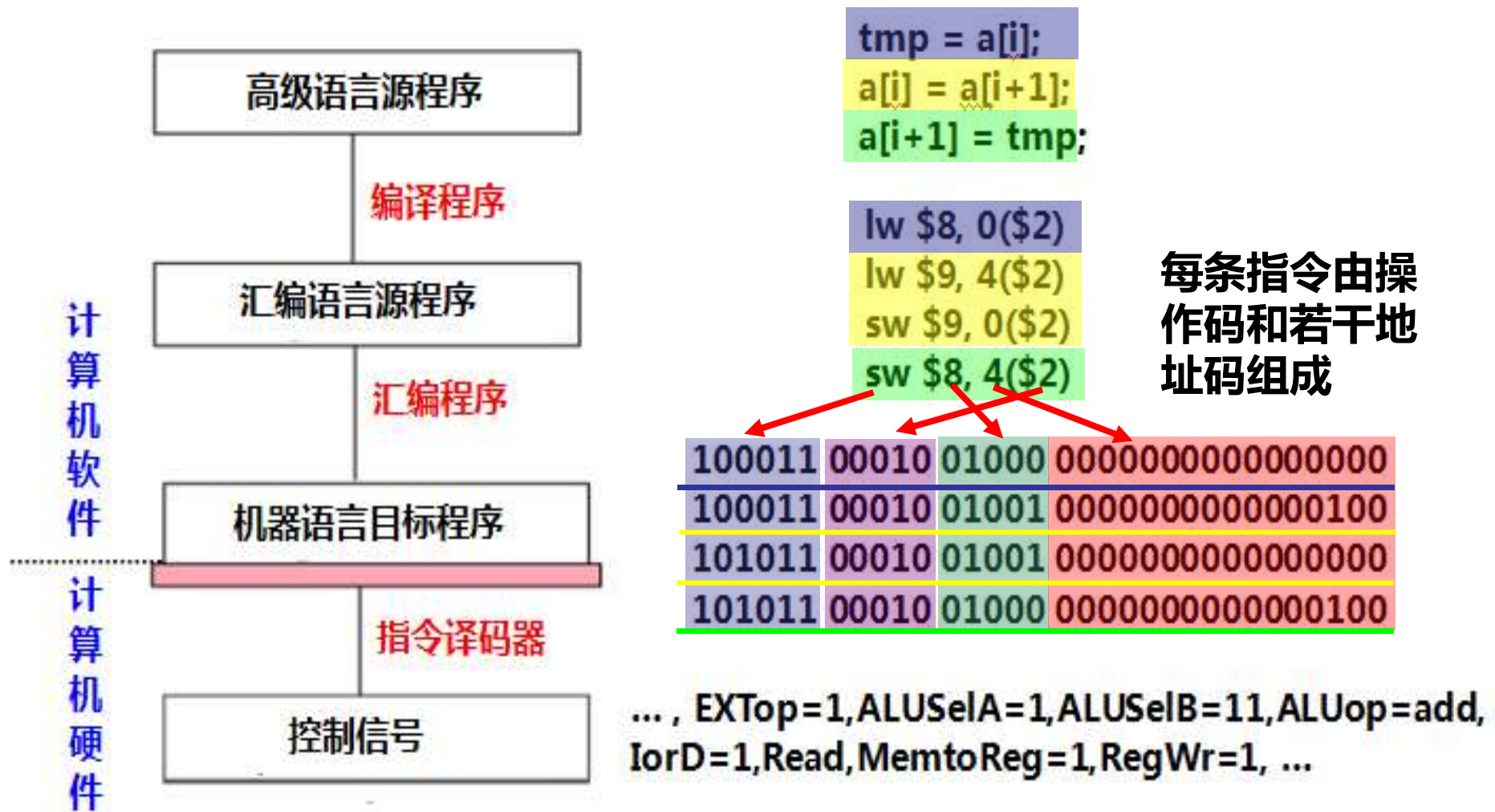
Hello程序的数据流动过程



数据经常在各存储部件间传送。故现代计算机大多采用“缓存”技术！

所有过程都是在CPU执行指令所产生的控制信号的作用下进行的。

不同层次语言之间的等价转换



任何高级语言程序最终通过执行若干条指令来完成！

开发和运行程序需什么支撑？

- 最早的程序开发很简单（怎样简单？）
 - 直接输入指令和数据，启动后把第一条指令地址送PC开始执行
- 用高级语言开发程序需要复杂的支撑环境（怎样的环境？）
 - 需要编辑器编写源程序
 - 需要一套翻译转换软件处理各类源程序
 - 编译方式：预处理程序、编译器、汇编器、链接器
 - 解释方式：解释程序
 - 需要一个可以执行程序的界面（环境）
 - GUI方式：图形用户界面
 - CUI方式：命令行用户界面

语言
处理
程序

语言处理系统 +

语言的运行时系统

操作系统内核

指令集体系结构

计算机硬件

人机
接口

+
操作
系统

支撑程序开发和运行的环境由系统软件提供

最重要的系统软件是操作系统和语言处理系统

语言处理系统运行在操作系统之上，操作系统利用指令管理硬件