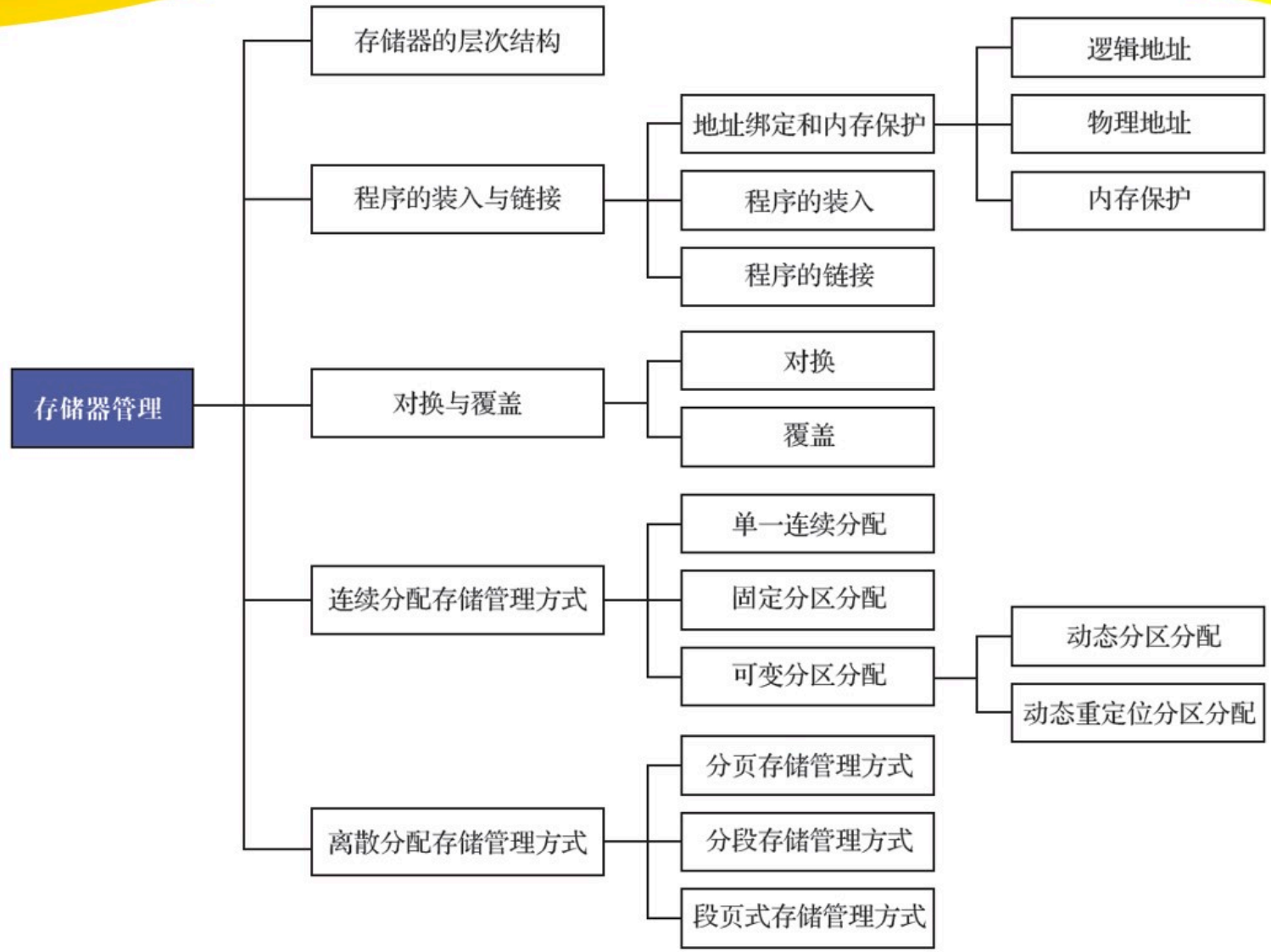
第5章知识导图

第1章	操作系统引论
第2章	进程的描述与控制
第3章	处理机调度与死锁
第4章	进程同步
第5章	**存储器管理**
第6章	虚拟存储器
第7章	输入/输出系统
第8章	文件管理
第9章	磁盘存储器管理
第10章	多处理机操作系统
第11章	虚拟化和云计算
第12章	保护和安全

第5章知识导图


存储器管理
存储器的层次结构
程序的装入与链接
地址绑定和内存保护
逻辑地址
物理地址
内存保护
程序的装入
程序的链接
对换与覆盖
对换
覆盖
连续分配存储管理方式
单一连续分配
固定分区分配
可变分区分配
动态分区分配
动态重定位分区分配
离散分配存储管理方式
分页存储管理方式
分段存储管理方式
段页式存储管理方式

Part 1 基础概念

学习目标

- 能够理解各层存储器的作用
- 理解存储管理的基本任务
- 理解逻辑地址、物理地址的概念
- 理解程序装入的实现方式及特点
- 理解程序链接的实现方式及特点

知识回顾

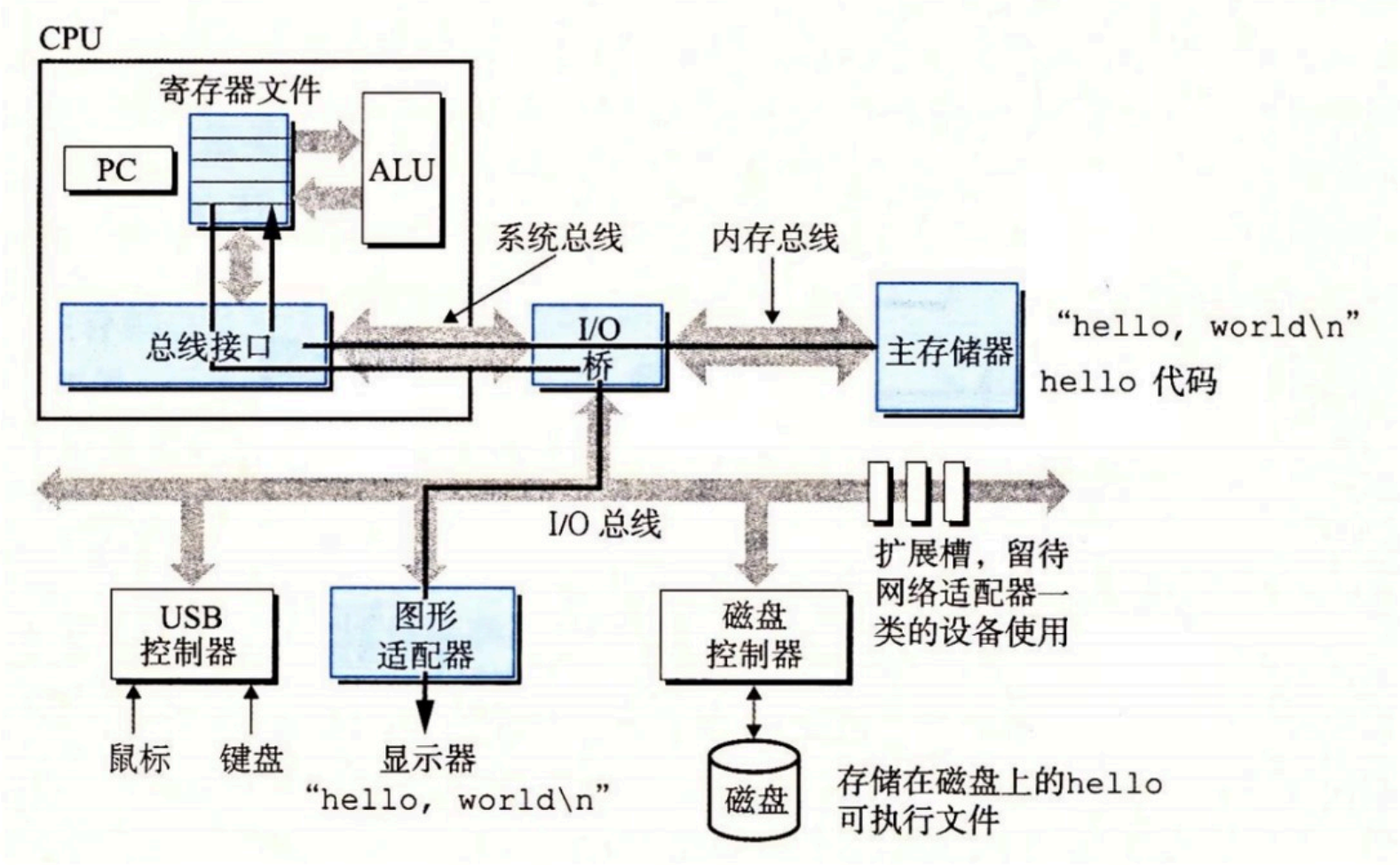
Q1：并发环境下程序如何执行?

Q2：CPU如何执行一条指令?

Q3：如何表示跳转地址?

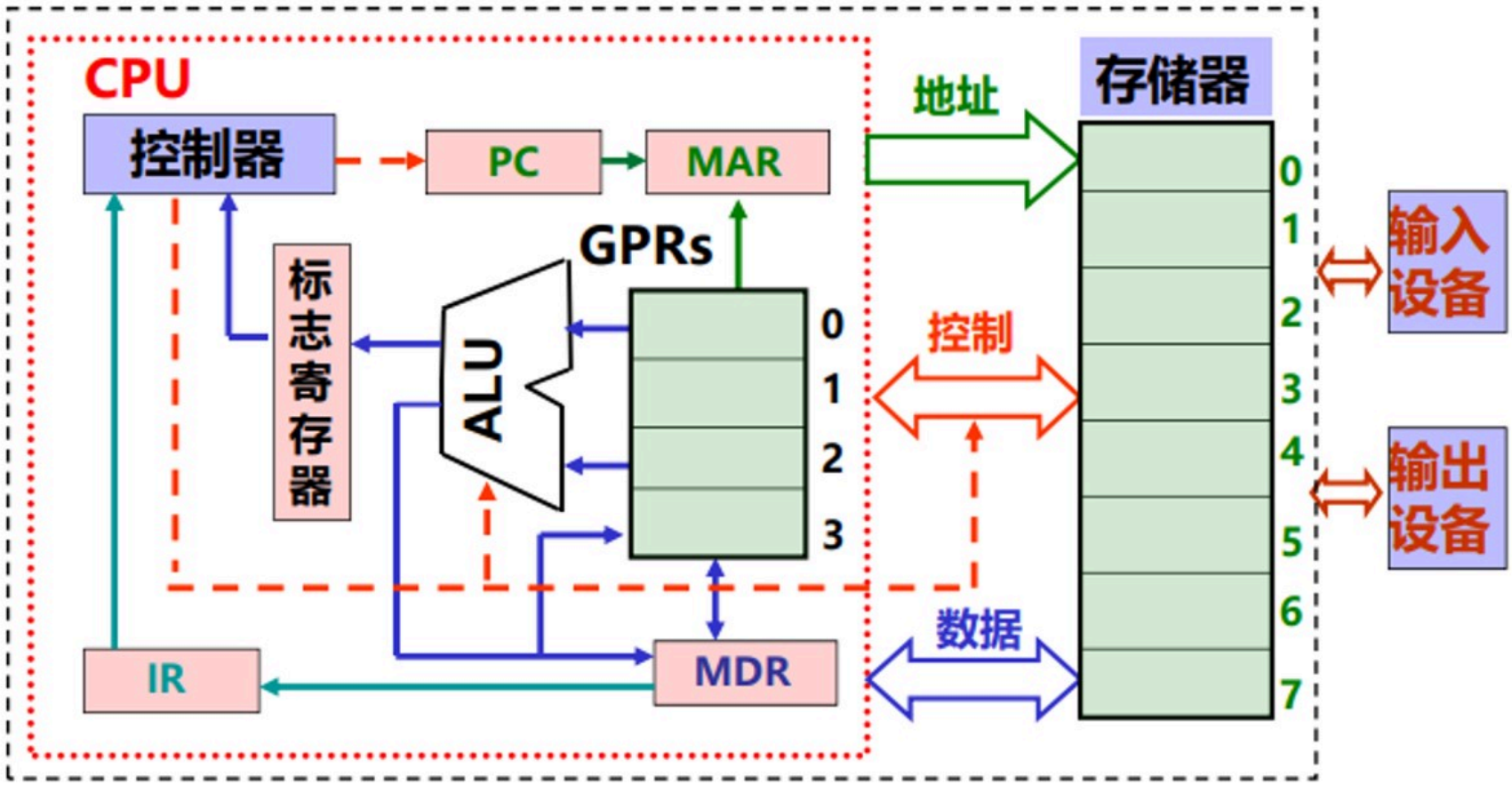
知识回顾

hello程序执行



知识回顾

CPU执行程序



1.1 内存管理的任务

内存分配回收

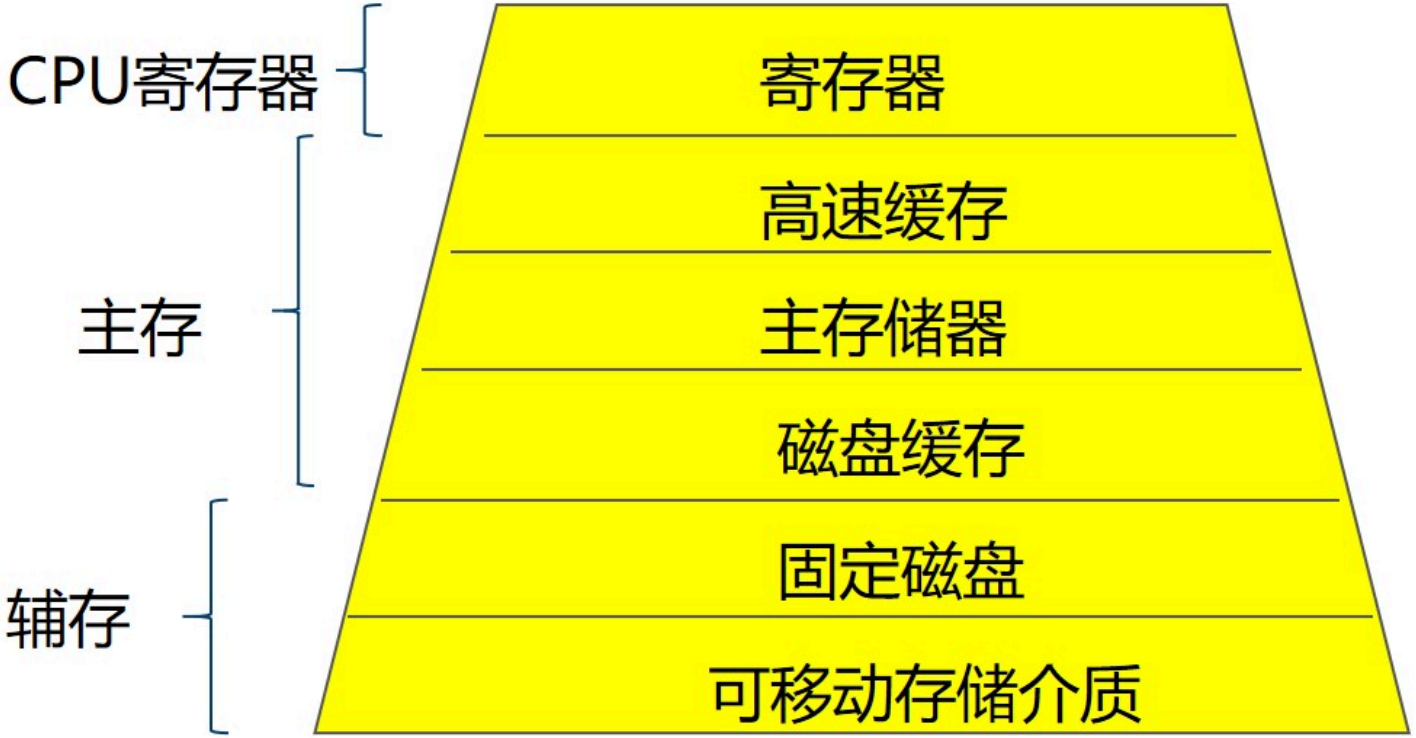
地址转换

存储扩充

存储保护

1.2 存储器的层次结构

计算机存储层次示意图



1.2 存储器的层次结构

寄存器

- 存放CPU执行时的数据和指令

1.2 存储器的层次结构

高速缓存：介于寄存器和存储器之间。

- 备份主存主常用数据和指令，减少对主存储器的访问次数；
- 缓和内存与处理机之间的矛盾。

1.2 存储器的层次结构

高速缓存

L1d： 一级数据缓存
L1i： 一级指令缓存
L2 cache： 二级缓存
L3 cache： 三级缓存

```
[mylinux@localhost ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:   0
L1d 缓存:              32K
L1i 缓存:              32K
L2 缓存:               256K
L3 缓存:               3072K
```

1.2 存储器的层次结构

磁盘缓存

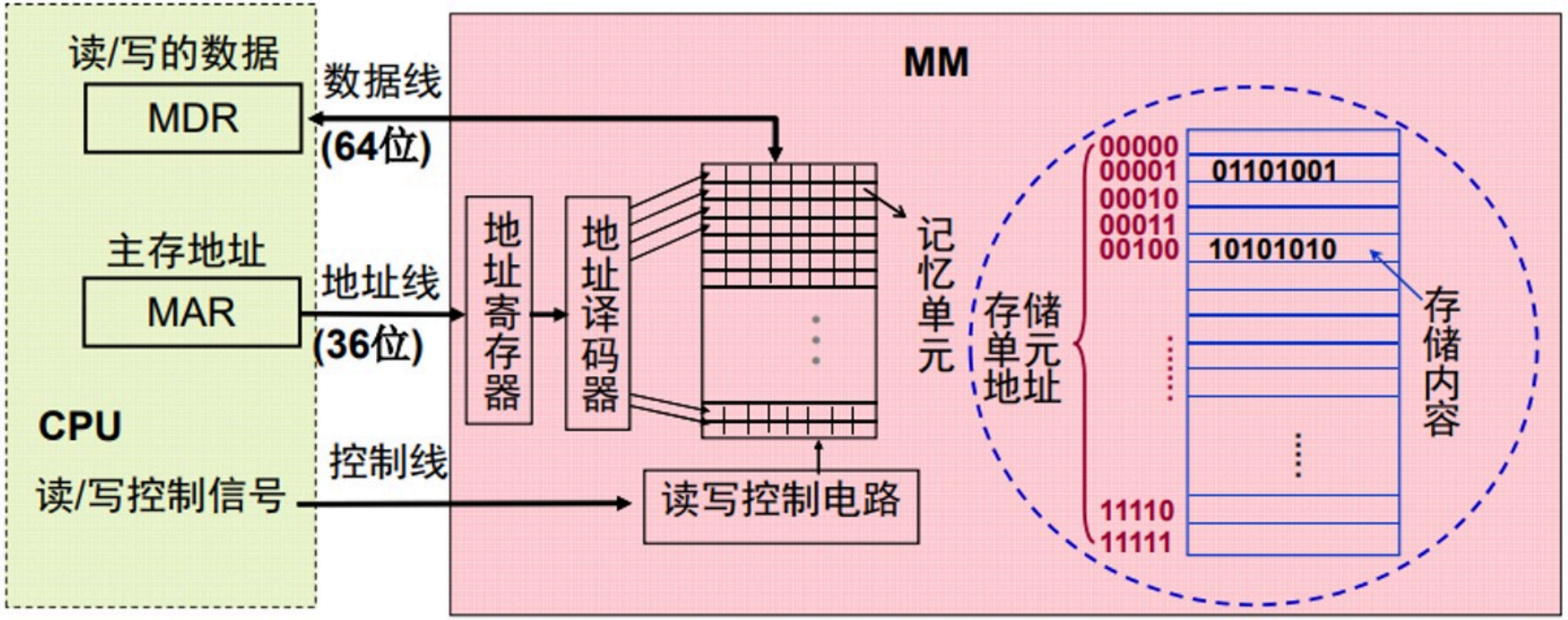
- 暂时存放频繁使用的一部分磁盘数据和信息；
- 缓和主存和I/O设备在速度上的不匹配；
- 利用主存的部分空间，主存可看成辅存的高速缓存。

1.2 存储器的层次结构

主存

性能指标：

- **按字节连续编址，每个存储单元为1个字节（8个二进位）**
- **存储容量：所包含的存储单元的总数（单位：MB或GB）**



1.2 存储器的层次结构

某台计算机存储器层次配置

- CPU中的寄存器100个字，存取时间1ns；
- 高速缓存512KB，存取周期15ns；
- 主存储器4GB，存取周期60ns；
- 磁盘容量500GB，存取周期毫秒级；
- 后援存储容量1TB，存取周期秒级。

1.3 程序地址

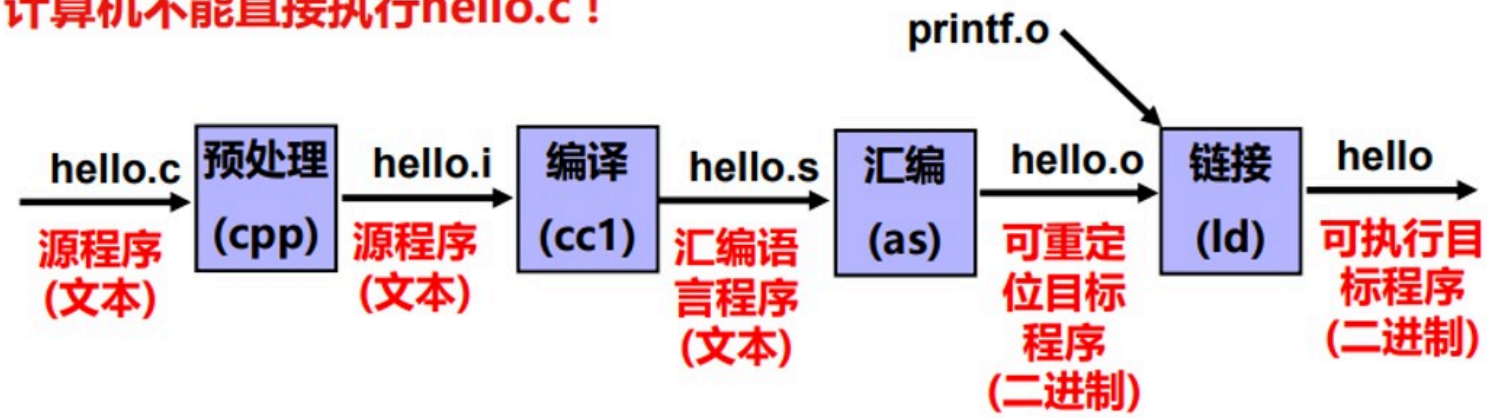
- **程序在成为进程前的准备工作**

Linux系统gcc编译过程：

```
#include <stdio.h>

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

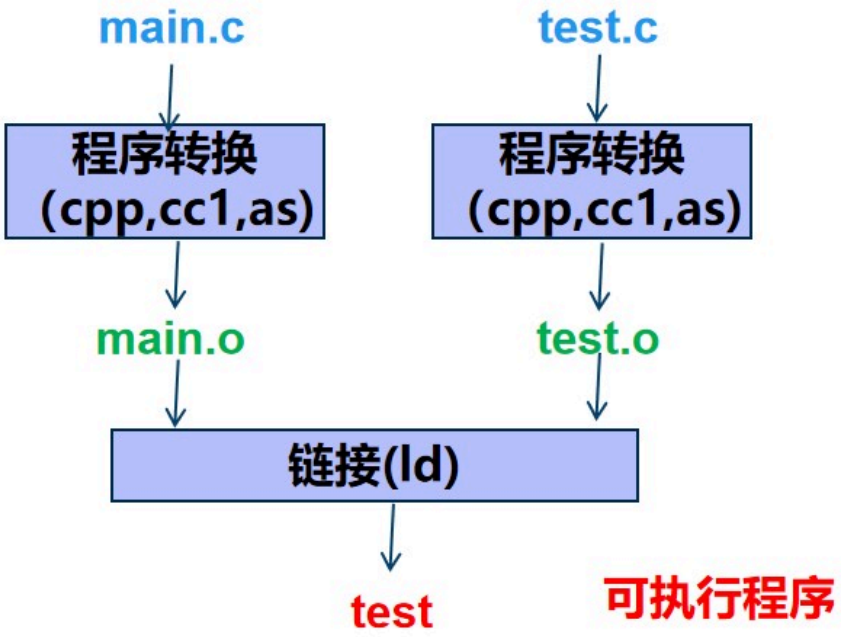
计算机不能直接执行hello.c！



1.3 程序地址

- **程序在成为进程前的准备工作**

Linux系统gcc编译过程：




```
/* main.c */
int add(int, int);
int main( )
{
  return add(20, 13);
}
/* test.c */
int add(int i, int j)
{
    int x = i + j;
    return x;
}
```

1.3 程序地址

程序各阶段的地址变化

1.源程序编译后的main.s文件

add：符号表示

符号地址

```
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $13, %esi
        movl    $20, %edi
        call    add
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
```

1.3 程序地址

程序各阶段的地址变化

2.源程序汇编后的main.o和test.o反汇编后的文件

```
0000000000000000 <main>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   be 0d 00 00 00          mov    $0xd,%esi
   9:   bf 14 00 00 00          mov    $0x14,%edi
   e:   e8 00 00 00 00          callq  13 <main+0x13>
  13:   5d                      pop    %rbp
  14:   c3                      retq
0000000000000000 <add>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   89 7d ec                mov    %edi,-0x14(%rbp)
   7:   89 75 e8                mov    %esi,-0x18(%rbp)
   a:   8b 45 e8                mov    -0x18(%rbp),%eax
   d:   8b 55 ec                mov    -0x14(%rbp),%edx
  10:   01 d0                   add    %edx,%eax
  12:   89 45 fc                mov    %eax,-0x4(%rbp)
  15:   8b 45 fc                mov    -0x4(%rbp),%eax
  18:   5d                      pop    %rbp
  19:   c3                      retq
```

13：整数值

相对地址

1.3 程序地址

程序各阶段的地址变化

3.可执行程序myprog反汇编后　4004e2：整数值

相对地址

```
00000000004004cd <main>:
  4004cd:       55                      push   %rbp
  4004ce:       48 89 e5                mov    %rsp,%rbp
  4004d1:       be 0d 00 00 00          mov    $0xd,%esi
  4004d6:       bf 14 00 00 00          mov    $0x14,%edi
  4004db:       e8 02 00 00 00          callq  4004e2 <add>
  4004e0:       5d                      pop    %rbp
  4004e1:       c3                      retq

00000000004004e2 <add>:
  4004e2:       55                      push   %rbp
  4004e3:       48 89 e5                mov    %rsp,%rbp
  4004e6:       89 7d ec                mov    %edi,-0x14(%rbp)
  4004e9:       89 75 e8                mov    %esi,-0x18(%rbp)
  4004ec:       8b 45 e8                mov    -0x18(%rbp),%eax
  4004ef:       8b 55 ec                mov    -0x14(%rbp),%edx
  4004f2:       01 d0                   add    %edx,%eax
  4004f4:       89 45 fc                mov    %eax,-0x4(%rbp)
  4004f7:       8b 45 fc                mov    -0x4(%rbp),%eax
```

1.3 程序地址

- **程序各阶段的地址变化**

4.可执行程序装入内存执行时的地址?

内存单元地址 —— CPU对内存的访问方式决定

物理地址