1. cache是什么？为什么一般不能将CPU内部的寄存器存贮器设计得很大而使用寄存器作为cache？为什么利用cache能提高程序的运行效率？cache 命中率指的是什么意思？如何计算cache命中率？什么是cache抖动？编写程序时，提高cache命中率的原则是什么？

- cache是什么？
    - cache是一种小容量高速缓存存储器，由快速的SRAM构成，用于存放CPU近期可能需要的指令和数据。
- 为什么一般不能将CPU内部的寄存器存贮器设计得很大而使用寄存器作为cache？
    - 寄存器的成本较高，且寄存器的容量有限，不能存放大量的数据。
    - 寄存器的结构复杂，导致CPU体积会变大，功耗会增加。
- 为什么利用cache能提高程序的运行效率？
    - cache的存取速度比主存快几个数量级。cache存放CPU近期可能需要的指令和数据，减少了CPU访问主存的次数，提高了程序的运行效率。
- cache命中率指的是什么意思？如何计算cache命中率？
    - 若CPU访问单元所在的内存块在cache中，则称为cache命中(hit)。命中的概率称为命中率(hit rate)。
    - 命中率 $= \frac{\text{命中次数}}{\text{访问总次数}} \times 100\%$
- 什么是cache抖动？
    - cache抖动是指cache中的数据被频繁替换，导致cache中的数据无法被重复利用，从而降低了cache的效率。
- 编写程序时，提高cache命中率的原则是什么？
    - 空间局部性原则：程序中的数据和指令往往在空间上是连续的，因此在访问一个数据或指令时，很可能会访问附近的数据或指令。
    - 时间局部性原则：程序中的数据和指令往往在时间上是连续的，因此在访问一个数据或指令时，很可能会再次访问这个数据或指令。
    - 程序的局部性原则：程序中的数据和指令往往在空间和时间上是连续的，因此在访问一个数据或指令时，很可能会访问附近的数据或指令，并且会再次访问这个数据或指令。

2. 写数据时怎样保证cache和主存的一致性?

- 全写法：当CPU执行写操作时，若写命中，则同时将数据写入cache和主存；若写不命中，则再分为两种策略：
    - 写分配法：先在主存块中更新相应存储单元，然后分配一个cache行，将更新后的主存块装入分配的cache行中。
    - 非写分配法：仅更新主存单元，不把主存块装入cache中。
- 回写法：当CPU执行写操作时，若写命中，则信息只被写入cache而不被写入主存；若写不命中，则在cache中分配一行，将主存块调入该cache行中，并更新cache行中相应单元的内容。只有在cache行中的主存块被替换时，才将该主存块内容一次性写回主存。

3. 在cache映射时, cache中每一行需要一个标记信息（用于确定主存的哪一块）。假设主存空间1M字节, cache数据区4K字节, 块大小为512字节, 在直接映射和全相联映射方式下, 一个cache行的标记域需要多少个二进制位?

主存空间2^{20}字节，块大小2^9字节，则主存空间分为$2^{20-9}=2^{11}$块，cache数据区分为$2^{12}/2^9=2^3$行。

- 直接映射：cache中共2^3行，故cache行号字段需要3位，因此标记字段需要$11-3=8$位。
- 全相联映射：主存空间分为2^{11}块，因此标记字段需要11位。

4. 根据本章的知识，列出几种利用CPU硬件特性的优化方法。

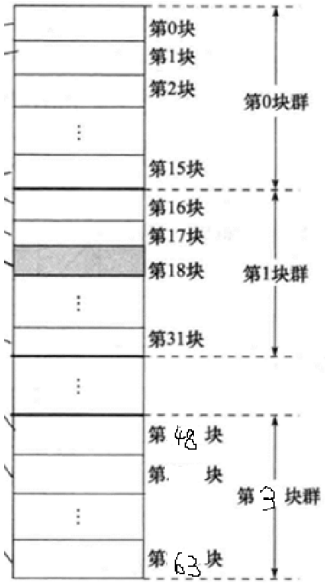
- 提高cache命中率：利用空间局部性、时间局部性和程序局部性原则。
- SIMD指令：一次性处理多个数据。
- 多线程：提高程序的并行度。

5. 假定主存按字（4字节）编址。cache数据区容量为8K字。块大小为512字。采用直接映射方式。分析下面程序效率不高的原因。

```
int a[4][512 * 16]; // 假定a的起始地址为 00000

int main()
{
    int sum = 0;
    for (int c = 0; c < 512 * 16; c++) {
        for (int r = 0; r < 4; r++) {
            sum += a[r][c];
        }
    }
    printf("sum = %d\n", sum);
}
```

直接映射方式下，块大小为$512 = 2^9$字节，cache数据区容量为$8K = 2^{13}$字节，故cache数据区分为$2^{13}/2^9 = 2^4$行。则数组a在主存中分为64个块、4个块群。



- a[0][0]位于主存块0，读入cache行0。
- a[1][0]位于主存块16，读入cache行0。
- a[2][0]位于主存块32，读入cache行0。
- a[3][0]位于主存块48，读入cache行0。
- a[0][1]位于主存块0，读入cache行0。
- a[1][1]位于主存块16，读入cache行0。
- a[2][1]位于主存块32，读入cache行0。
- a[3][1]位于主存块48，读入cache行0。

- ...
- a[0][511]位于主存块0，读入cache行0。
- a[1][511]位于主存块16，读入cache行0。
- a[2][511]位于主存块32，读入cache行0。
- a[3][511]位于主存块48，读入cache行0。
- a[0][512]位于主存块1，读入cache行1。
- a[1][512]位于主存块17，读入cache行1。
- a[2][512]位于主存块33，读入cache行1。
- a[3][512]位于主存块49，读入cache行1。
- ...

可见，cache命中率为0。每次读取a[r][c]时，都会导致cache行替换，无法重复利用cache行中的数据，导致cache抖动，程序的运行效率极低。