1. 已知一个硬盘容量为 1TB，由两个盘（4 个盘面）构成，每个盘面由 3200 个磁道构成，每个磁道平均有 128 个扇区。由于磁记录密度的提升，新工艺维持相同的容量下，只需要一个盘（2 个盘面），每个盘面的磁道数为 3200，则新工艺要求每个磁道多少个扇区？
   256

2. 请写出磁盘扇区访问时间的计算公式，并简单解释寻道时间、旋转时间、传送时间。

   一个扇区的平均访问时间=平均寻道时间+平均旋转时间+数据传输时间

3. 如果 cache 块（行）的大小为 64 字节，cache 总容量为 1KB，请问 cache 的块数有多少？如果采用 2 路组相联则 32 位的内存地址，将划分成标记、组索引、快内偏移三个部分各需几个 bit?
   1KB=64 字节/块*16 个块
   块内偏移的比特数=6 比特
   组索引=3 bit
   标记位=32-6-4=23 比特

4. 请简单比较直接映射和全相联映射在硬件实现上的复杂程度的差异
   全相联映射需要进行更多的标记位比较，硬件实现更复杂

5. 请简单分析一下各项因素单独变化时对 cache 命中率的影响：cache 容量大小变化、cache 块大小、组的大小。
   容量变大；命中率提高
   cache 块变大：命中率先变高，但太大会变低，不命中的开销变大
   一组的块数变多：命中率变高

6. 什么是数据访问的时间局部性和空间局部性？
   时间局部性：当前访问的数据和指令，下次还会被访问
   空间局部性：当前访问的数据和指令，其邻近的数据和指令会被访问

7. 请简单说明什么是连接器相关的强符号、弱符号？
   强符号：函数和初始化的全局变量
   弱符号：未初始化的全局变量

8. 对于以下代码：

```
1.  int   a=100;

2.  short proc(int inputarg)
3.  {     int a;
          a=inputarg;
          return a+0;
4.  }
```

请说明边第 1 行变量 a 和 3 行定义变量 a 的作用域。如果将第 3 行变量类型修改为 static int，则第一次调用函数 proc 时传入参数 inputarg 的值为 12，此时返回值为多少？再次调用时传入 inputarg 为 5，则第二次调用的返回值是多少？

12， 5

9. 对于以下 myfile.c 程序代码

```
int hello1;

char myfun(int a)
    {     int b=10;
          hello1=b+a;
          hello1=hello1*f1(a);
          printf("%d\n",hello1);
          return hello1;
    }
```

请指出那些符号是链接器符号，哪些符号需要重定位？

hello1 myfun f1 printf

hello1 f1 printf

10. 假定某计算机的 cache 采用直接映射方式，和主存交换的数据块大小为 1 个字，按字编址，一共能存放 16 个字的数据。CPU 开始执行某程序时，cache 为空，在该程序执行过程中，CPU 依次访问以下地址序列：2、3、11、16、21、13、64、48、19、11、3、22、4、27、6 和 11。

（1）每次访问在 cache 中命中还是缺失？试计算访问上述地址序列的 cache 命中率。1/16

（2）若 cache 容量还是 16 个字，而数据块大小改为 4 个字，则上述地址序列的命中情况又如何？说明块大小和命中率的关系。1/4

11. 假定某计算机的数据 cache 采用全相连映射方式，即内存的任意一个块可以被放置到 cache 中的任意位置。主存与 cache 交换的块大小为 2 个字（2B），数据 cache 一共可以存放 4 个字。当 cache 发生冲突时，选取最近最久不使用（LRU）的那个块被换出。数组按行优先方式存储，请分别计算代码段 1 和代码段 2 的 Cache 命中率，讨论哪一个代码对 cache 更友好，并分析 cache 不友好代码的原因。

```
short sum_array_1 (short a[4][2])
{
  int i, j;
  short sum = 0;

  for (j = 0; j < 2; j++)
    for (i = 0; i < 4; i++)
      sum += a[i][j];
  return sum;
}
```

代码 1

```
long sum_array_2(long a[4][2])
{
  int i, j;
  long sum = 0;

  for (i = 0; i < 4; i++)
    for (j = 0; j < 2; j++)
      sum += a[i][j];
  return sum;
}
```

代码 2

第二个代码对 cache 更友。 第一个代码的数据访问没有好的局部性。