4. Cache

课后作业

B.1

> 注
> 下面题目中缓存缺失时间不是表示在未命中情况下增加的访存时间，而是总的时间。当然你可以有不同理解，这样答案就随之有改变。

a. 平均访存时间 = Cache 命中率 *Cache命中访存时间 + Cache缺失率 * Cache缺失访存时间

= 0.95 * 1 + 0.05 * 105 = 6.2

b. 由于是随机生成数组的索引，所以 Cache 的命中率为：64KB/256MB = 0.00025

根据上题的公式，可以得到平均访存时间为 0.00025 *1 + (1-0.00025) * 105 = 104.974

c. 上面是使用了 Cache 的平均访存时间，如果没有使用 Cache 的访存时间是 100，可以看出使用 Cache 之后访存时间反而增大了，因此数据的局部性是 Cache 存在的基础。

d. 设缺失率为 x，那么引入 Cache 不会产生副作用就必须满足：

```
G * (1-x) ≥ L * x
```

得到 x ≤ G/(G+L)。

对于本题来说，如果 G = 99， L = 5，那么最高缺失率为 0.95 才不会产生副作用。

B.3

略。

B.4

a. 对于 PORTION = 1 时，题目给的代码执行的迭代次数为 1 次。那么 Cache 采用写直达策略时，每次写的字节为 4 B，花费 CPU 时间为：10 + 5*([4/8] - 1) = 10.

b. 如果是写回策略，等到被替换出去的时候才将整个 Cache 行写回主存，Cache 行大小为 32 B，所以花费时间为：10 + 5*([32/8] - 1) = 25.

c. 令 PORTION = 8 时，由于是写直达，每次写 4B 都需要往主存写，所以共写主存 8 次，时间为 10 * 8 = 80.

d. 写回策略每次写回要消耗 25 个 cycles，写直达每次要消耗 10 cycles，那么对同一缓存行至少要写 3 次，才会时写回策略占优。

e. 举一个例子说明写直达消耗的总时间比写回少即可。

B.8

a. 指令 Cache 大小为 128B，指令行大小为 4B，对于给定的循环代码大小为 64B，循环代码可以完全装入 Cache 中，因此只有第一次迭代的时候 Cache 会缺失，后面的所有迭代都会命中，所以缺失率趋向于 0.

b. 由于采用的是全相联映射、LRU 替换策略，当循环体大小为 192 B （共 48 条指令）时，下表为 Cache 的映射关系（红色字体表示第二次迭代）：

1（33）	（17）
2（34）	（18）
3（35）	（19）
.....	
16（48）	（32）
17（1）	（33）

18（2）（34）
.....
32（16）（48）

那么对于前 32 条指令全部缺失，第 33 条指令替换的是 Cache 的第一行（最久未使用），第 34 条指令替换的是 Cache 的第二行，...，以此类推，第一次迭代全部未命中。第二次迭代，第 1条指令替换的是第 17 行，第 2 条指令替换的是第 18 行，...，以此类推，发现第二次迭代完仍未有一条指令命中。因此，缺失率为 100%。

同理可以分析当循环体大小为 320B 时，缺失率也为 100%。这也回答了题目中的结论：当一个大于指令 Cache 的循环连续执行时，LRU 策略基于的假设是不成立的。

c. 如果将替换策略变成 MRU 时，

- 当循环体大小为 64B 时，因为 Cache 大小比它大，所以除了第一次迭代缺失之外，后面的迭代都会命中，与 LRU 策略相同。
- 当循环体大小为 192B （共 48 条指令）时，看下面的映射图（黑色字体为第一次迭代，红色字体为第二次迭代，蓝色字体为第三次迭代）：

1（1）（1）
2（2）（2）
3（3）（3）
.....
30（30）（30、31...46）
31（31、32...47）（47）
32（33、34...48）（48）（48）

第一次迭代仍然全部缺失；

第二次迭代，第 1～31 条指令是命中的，第 32～47 条指令缺失，第 48 条指令又命中，因此这次循环共 48 条指令，命中 32 次，缺失 16 次；

第三次迭代，第 1～30 条指令是命中的，第 31～46 条指令缺失，第 47、48 条指令又命中，因此这次循环共 48 条指令，命中 32 次，缺失 16 次；

后面每次迭代都有相似的规律。

- 当循环体大小为 320B 时，同理，这里就不给具体分析过程了。