1. 判断以下描述更符合 SRAM 还是 DRAM，还是都符合.

	SRAM	DRAM
(1)访问速度更快	✔	
(2)每比特需要的晶体管数目少		✔
(3)单位容量造价更便宜		✔
(4)常用作主存		✔
(5)需要定期刷新		✔
(6)断电后失去存储的信息	✔	✔
(7)支持随机访问	✔	✔

2. 已知一个双面磁盘有 2 个盘片、10000 个柱面，每条磁道有 400 个扇区，每个扇区容量为 512 字节，则它的存储容量是_**8.192**_GB

   2*2*10000*400*512 = 8,192,000,000 Byte = 8.192GB

3. 已知一个磁盘的平均寻道时间为 6ms，旋转速度为 7500RPM，那么它的平均访问时间大约为_**10**_ms

   6ms + 0.5*(60/7500*1000)ms = 10ms

4. 已知一个磁盘每条磁道平均有 400 个扇区，旋转速度为 6000RPM，那么它的平均传送时间大约为_**0.025**_ms

5. 考虑如下程序

```
for (int i = 0; i < n; i++) {
   B[i] = 0;
   for (int j = 0; j < m; j++)
      B[i] += A[i][j];
}
```

判断以下说法的正确性

(**N**) 对于数组 A 的访问体现了时间局部性。

(**Y**) 对于数组 A 的访问体现了空间局部性。

(**Y**) 对于数组 B 的访问体现了时间局部性。

(**Y**) 对于数组 B 的访问体现了空间局部性。

6. 高速缓存

   a) 一个容量为 8K 的直接映射高速缓存，每一行的容量为 32B，那么它有_**256**_组，每组有_**1**_行。

   b) 一个容量为 8K 的全相联映射高速缓存，每一行的容量为 32B，那么它有_**1**_组，每组有_**256**_行。

   c) 一个容量为 8K 的 4 路组相联映射高速缓存，每一行的容量为 32B，那么它有_**64**_组，每组有_**4**_行。

   d) 一个容量为 16K 的 4 路组相联高速缓存，每一行的容量为 64B，那么一个 16 位地址 0xCAFE 应映射在第_**43**_组内。

7. 判断以下说法的正确性

(**Y**)	保持块大小与路数不变，增大组数，命中率一定不会降低。
(**N**)	保持总容量与块大小不变，增大路数，命中率一定不会降低。
(**N**)	保持总容量与路数不变，增大块大小，命中率一定不会降低。
(**Y**)	使用随机替换代替 LRU，期望命中率可能会提高。

8. 有以下定义：

```
// 以下都是局部变量
int i, j, temp, ians;
int *p, *q, *r;
double dans;
// 以下都是全局变量
int iMat[100][100];
double dMat[100][100];
// 以下都是函数
int foo(int x);
```

如果将下列左侧代码优化为右侧代码，有没有可能有副作用？如果有请说明.

(1)	`ians = 0;`
	`for (j = 0; j < 100; j++)`
	`  for (i = 0; i < 100; i++)`
	`  ians += iMat[i][j];`
		`ians = 0;`
		`for (i = 0; i < 100; i++)`
		`  for (j = 0; j < 100; j++)`
		`  ians += iMat[i][j];`

(2)	`dans = 0;`
	`for (j = 0; j < 100; j++)`
	`  for (i = 0; i < 100; i++)`
	`  dans += dMat[i][j];`
		`dans = 0;`
		`for (i = 0; i < 100; i++)`
		`  for (j = 0; j < 100; j++)`
		`  dans += dMat[i][j];`

(3)	`for (i = 0;i < foo(100);i++)`
	`  ians += iMat[0][i];`
		`temp = foo(100);`
		`for (i = 0; i < temp; i++)`
		`  ians += iMat[0][i];`

(4)	`*p += *q;` `*p += *r;`	`temp = *q + *r;` `*p += temp;`

答：
(1) 会
(2) 不会，因为浮点数不能结合
(3) 不会，因为foo 可能有副作用
(4) 不会，如果 pqr 指向同一个元素那么两个运算不等价

9. 假设已有声明 `int i, int sum, int *p, int *q, int *r, const int n = 100, float a[n], float b[n], float c[n], int foo(int), void bar()`，以下哪种优化编译器总是可以进行？

A	`for(i = 0; i < n; ++i){`
	`  a[i] += b[i];`
	`  a[i] += c[i];`
	`}`
		`float temp;`
		`for(i = 0; i < n; ++i){`
		`  temp = b[i] + c[i];`
		`  a[i] += temp;`
		`}`

B	`*p += *q;`
	`*p += *r;`
		`int temp;`
		`temp = *q + *r;`
		`*p += temp;`

C	`for(i = 0; i < n; ++i)` `  sum += i*4;`	`int N = n * 4;` `for(i = 0; i < N; i += 4)`

		sum += i;

D	for(i = 0; i < foo(n); ++i)
	bar();
		int temp = foo(n);
		for(i = 0; i < temp; ++i)
		bar();

C

10. 阅读下列 C 代码以及它编译生成的汇编语言

```
long func() {
    long ans = 1;
    long i;
    for (i = 0; i < 1000; i += 2)
        ans = ans ?? (A[i] ?? A[i+1]);
    return ans;
}
```

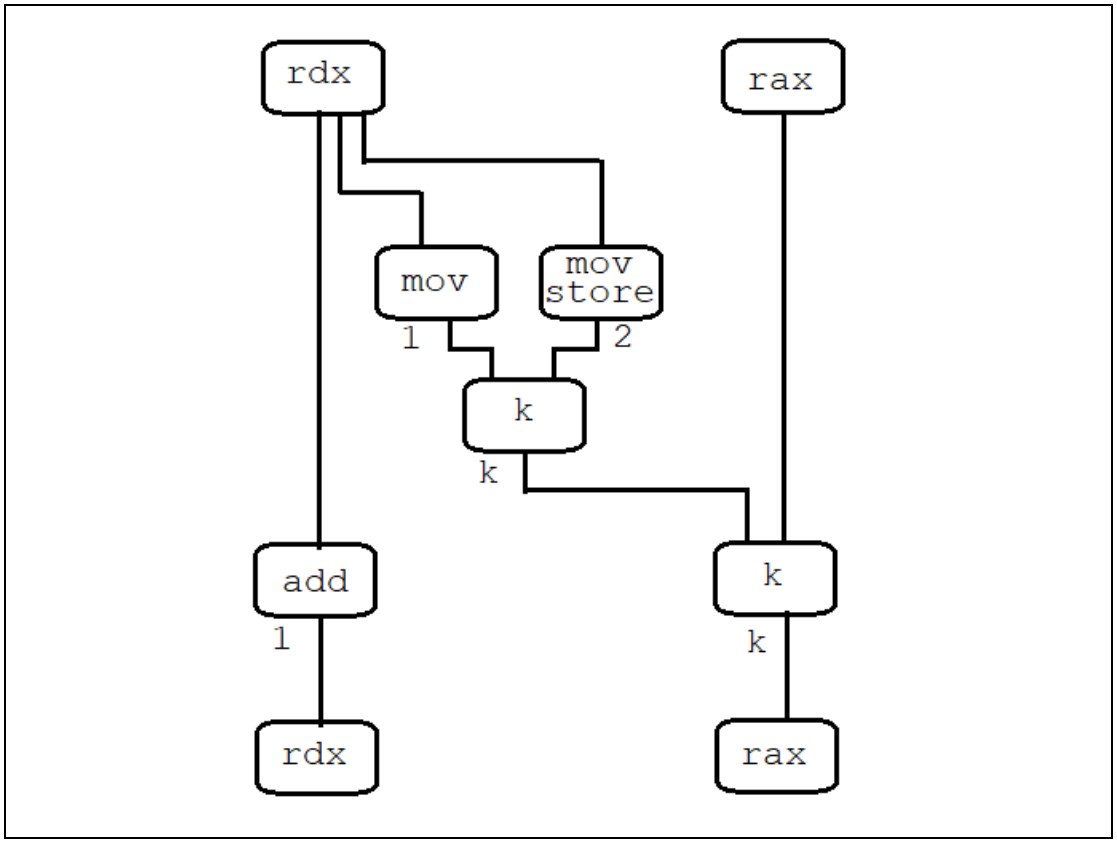
```
func:
    movl $0, %edx
    movl $1, %eax
    leaq A(%rip), %rsi
    jmp .L2
.L3:
    movq 8(%rsi,%rdx,8), %rcx // 2 cycles
    ?? (%rsi,%rdx,8), %rcx    // k + 1 cycles
    ?? %rcx, %rax             // k cycles
    addq $2, %rdx             // 1 cycles
.L2:
    cmpq $999, %rdx           // 1 cycles
    jle .L3
    rep ret
```

该程序每轮循环处理两个元素。在理想的机器上（执行单元足够多），每条指令消耗的时间周期如右边所示。

(1) 当问号处为乘法时，k = 8。此时这段程序的 CPE 为__**4**__

(2) 当问号处为加法时，k = 1。此时这段程序的 CPE 为__**0.5**__

数据相关图：



现有一个能够存储4个Block的Cache，每一个Cache Block的长度为2 Byte（既B = 2）。内存空间的大小是32Byte，既内存空间地址范围如下：

0_{10} (00000_2) -- 31_{10} (11111_2)

现有一程序，访问内存地址序列如下所示，单位是Byte。

2_{10} 23_{10} 10_{10} 9_{10} 9_{10} 11_{10} 3_{10}

1. Cache的结构如下图所示（S=2， E=2），初始状态为空，替换策略LRU。请在下图空白处填入上述数据访问后Cache的状态。
（TAG使用二进制格式；Data Block使用10进制格式，例：M[6-7]表示地址6_{10}-7_{10}对应的数据）

	V	TAG	Data Block
set0	1	010	M[8-9]
	0		

	V	TAG	Block
set1	1	010	M[10-11]
	1	000	M[2-3]

上述数据访问一共产生了多少次 Hit: __2__ (2pts)

2．如果Cache的替换策略改成MRU（即Most Recently Used，最近使用的数据被替换出去），请在下图空白处填入访问上述数据访问后Cache的状态。

set0

V	TAG	Data Block
1	010	M[8-9]
0		

set1

V	TAG	Block
1	000	M[2-3]
1	010	M[10-11]

上述数据访问一共产生了多少次 Hit：__3___ (2pts)

3．现增加一条新规则：地址区间包含 5 的倍数的 block 将不会被缓存，仍旧使用 MRU 替换策略，这 7 次数据访问一共产生了多少次 Hit __2___ (2pts)

4．在第 3 小题的基础上，现又增加一条数据预取规则：每当地址为 10 的数据被访问时，地址为 8 的数据将会被放入缓存，这 7 次数据访问一共产生了多少次 Hit __3___ (2pts)