Q1.关于多核处理器的Cache结构，介绍UCA与NUCA的特点

UCA是集中式共享结构。每一个处理器核拥有自己独立的一级Cache，多个核之间通过总线或者交叉开关连接二级Cache（假设二级Cache是LLC），这样所有处理器核队LLC的访问延迟相同。这种结构导致核的数量不可以过多，否则访问延迟会很大。此外由于使用总线核交叉开关互联，可扩展性不佳。

NUCA是分布式共享结构。每一个处理器核拥有自己独立的一级Cache和二级Cache（LLC），并通过片上互连访问其他处理器核的LLC。在NUCA结构中，处理器核可以访问所有的LLC，并且拥有不同的访问延迟。因此可扩展性良好，能支持的核数量也较大。

Q2.

P1 P2

X = 1； 1 Y = 1； 3

print Y； 2 print X； 4

顺序一致性要求每一次读都必须能读取到最近一次更新到的值，而弱一致性允许读取到更新之前的值。由于P1和P2之间并没有同步操作，于是可行的执行顺序有如下几种：

对于顺序一致性：

1234 正确结果：打印 0 1

1324 正确结果：打印 1 1

1342 正确结果：打印 1 1

3412 正确结果：打印 0 1

3142 正确结果：打印 1 1

3124 正确结果：打印 1 1

对于弱一致性，由于两个线程间没有同步操作，故执行顺序可以乱序执行。

可行的执行顺序有4！= 24种

正确结果可以是： 打印 0 0

打印 0 1

打印 1 0

打印 1 1

Q3.MESI在ESI的基础上增加了M（Modified状态 ）。Modified状态表示当前Cache行被当前处理器核独占并且已经被修改过了，如果处理器核想替换该Cache行，需要将该Cache行写回内存。增加该位的优点是减少了Cache到内存的数据传输次数，Cache只需要将Modified状态的Cache行写回内存。

Q4.自旋锁的实现：

fetch\_and\_increment：

void acquire(&lock)

{

while(fetch\_and\_increment(lock)!=0);

临界区；

}

void release(&lock)

{

lock = 0;

}

compare\_and\_swap：

void acquire(&lock)

{

while(compare\_and\_swap (lock,1)!=0);

临界区；

}

void release(&lock)

{

lock = 0;

}

Q5. 当两个以上CPU都要访问同一个缓存行大小的内存区域时，就会引起冲突，这种情况就叫“共享”。但是，这种情况里面又包含了“其实不是共享”的“假共享”情况。比如，两个处理器各要访问一个word，这两个word却存在于同一个cache line大小的区域里，这时，从应用逻辑层面说，这两个处理器并没有共享内存，因为他们访问的是不同的内容（不同的word）。但是因为cache line的存在和限制，这两个CPU要访问这两个不同的word时，却一定要访问同一个cache line块，产生了事实上的“共享”。显然这种假共享会浪费系统资源。

写程序时的方法：

1）让不同线程操作的对象处于不同的缓存行。

可以进行缓存行填充（Padding）。例如，如果一条缓存行有 64 字节，而 Java 程序的对象头固定占 8 字节(32位系统)或 12 字节( 64 位系统默认开启压缩, 不开压缩为 16 字节)，所以我们只需要填 6 个无用的长整型补上6\*8=48字节，让不同的 VolatileLong 对象处于不同的缓存行，就避免了伪共享( 64 位系统超过缓存行的 64 字节也无所谓，只要保证不同线程不操作同一缓存行就可以)。

2）使用汇编指导，强制使每一个变量对齐。

强制使对象按照缓存行的边界对齐。例如可以使数据按64位对齐，那么一个缓存行只有一个可操作对象，这样发生假共享之后，也只是对应缓存行的数据变化，并不影响其他的对象。

Q6.

一个虚拟通道程序需要一个客户端组件和一个服务器端组件构成：  
1.服务器端组件运行在终端服务服务器上，它可以是一个用户程序或是一个核心设备。  
2.客户端组件是一个动态连接库，当终端服务的客户端程序运行时必须把它加载到内存中。

在片上网络路由结构中，承担终端服务服务器作用的是分配器。分配器包括路由计算、虚通道分配、交叉开关分配三个作用。

片上路由数据传输简述如下：客户端（单个处理器核心）将需要传输的数据输入路由器中的缓冲区（Buffer），随后分配器进行路由计算，计算虚拟信道的VPI和VCI，分配数据传输的虚通道，交叉开关分配最后传输出的资源端口，全部过程执行一次实现了一条信息从一个核到另一个核的传输。

Q7.

Fermi的存储结构：

对于单个线程，线程可见的存储器如下：

1. 寄存器

每一个SM有32K个32位寄存器，每个线程（SP）可以访问自己的私有寄存器。私有寄存器数目随线程数目的变化而变化，在21~63之间。

1. 一级Cache和共享存储。

一级Cache线程间独立，共享存储线程间共享。另外，每个SM块上均有片上高速存储，用于一级Cache与共享存储之间的交换。每组SM有16KB的共享存储。

1. 二级Cache

二级Cache位768KB，在16个SM间共享。该Cache可访问全局存储器。

1. 全局存储

所有线程共享的片外存储。

可共享的器件：共享存储、二级Cache、全局存储（主存）

访问延迟从低到高排序为寄存器、一级Cache、共享存储、二级Cache、全局存储。

访存带宽依据具体显卡不同而不同。以GTX 480为例，其访存带宽为177.4GB/s。

以下是GTX 480的其他参数：

GPU Engine Specs:

CUDA Cores 480

Graphics Clock (MHz) 700 MHz

Processor Clock Tester(MHz) 1401 MHz

Texture Fill Rate (billion/sec) 42

Memory Specs:

Memory Clock 1848 MHz (3696 data rate)

Standard Memory Config 1536 MB

Memory Interface GDDR5

Memory Interface Width 384-bit

Memory Bandwidth (GB/sec) 177.4