1.
a)4
b)4096

2.
a)T
处理完异常后回不到触发异常处
b)T
无法正确切换堆栈
c)F

d)T
恢复不到正常运行状态
e)F

3. B
4. ???

A->1	A->1	无变化
缺页异常	缺页异常	无变化
缺页异常 缺页异常	无变化	无变化
缺页异常 缺页异常		无变化

二

1)
a)管理一个内核线程的线程控制块数据结构为proc_struct
bcd)pid kstack mm cr3

2)
a)不会
b)是

3)
不
是

4)
不需要
需要
不需要
需要

在fork之后exec之前两个进程用的是相同的物理空间（内存区），子进程的代码段、数据段、堆栈都是指向父进程的物理空间，
也就是说，两者的虚拟空间不同，但其对应的物理空间是同一个。
当父子进程中有更改相应段的行为发生时，再为子进程相应的段分配物理空间，
如果不是因为exec，内核会给子进程的数据段、堆栈段分配相应的物理空间（至此两者有各自的进程空间，互不影响），
而代码段继续共享父进程的物理空间（两者的代码完全相同）。而如果是因为exec，由于两者执行的代码不同，子进程的代码段也会分配单独的物理空间。

在网上看到还有个细节问题就是，fork之后内核会通过将子进程放在队列的前面，以让子进程先执行，以免父进程执行导致写时复制，而后子进程执行exec系统调用，因无意义的复制而造成效率的下降。

三
1)
位于磁盘第一个扇区不超过512

MBR特征：

启动代码446字节: 检查分区表正确性,加载并跳转到磁盘上的引导程序

硬盘分区表：64字节,描述分区状态和位置,每个分区描述信息占据16字节

结束标志字：2字节(55AA),主引导记录的有效标志

2)
bootasm.S第16行cli

61行

使得80386的全部32根地址线有效，可寻址高达4G字节的线性地址空间和物理地址空间

3)
语句((void (*)(void))(ELFHDR->e_entry & 0xFFFFFF))();
跳转至ELF header中存储的ucore入口地址

四
(1)
功能:
fork() 创建一个继承的子进程，复制父进程的所有变量和内存，复制父进程的所有CPU寄存器(有一个寄存器例外)

调用接口

子进程的fork()返回0，父进程的fork()返回子进程标识符
无参数
(2)

五
(1)
函数调用: 返回地址压栈，跳转至函数入口
返回指令: 弹出栈顶并跳转至该值
(2)
记有n个参数:arg1...argn
argn,arg(n-1),...,arg1依次压入栈

(3)

```
void print_stackframe(void) {
    uint32_t ebp = read_ebp();
    uint32_t eip = read_eip();
```

```
for (int i = 0; i < STACKFRAME_DEPTH && ebp; ++ i) {
    cprintf("ebp:0x%08x eip:0x%08x args:", ebp, eip);

    for (int j = 0; j < 4; ++ j) {
        cprintf("0x%08x ", ((uint32_t *)ebp+2)[j]);
    }

    cprintf("\n");
    print_debuginfo(eip - 1);
    eip = ((uint32_t *)ebp)[1];
    ebp = ((uint32_t *)ebp)[0];
}
```

}

六
1)
硬件中断: 来自硬件设备的处理请求
软件中断: int指令触发的中断
系统调用：应用程序主动向操作系统发出的服务请求

2)
硬件将(SS,EFLAGS),CS,EIP,CS中断号压栈,切换堆栈，跳转到中断服务例程

软件:
关中断
保存断点，保护现场(通用寄存器,ds,es,fs,gs)
开中断
执行中断服务程序
关中断
恢复断点，恢复现场
开中断
返回断点

硬件将(SS,EFLAGS),CS,EIP,CS,中断号出栈

硬件 宿主机 虚拟机

001D1A70 55 push ebp
001D1A71 8B EC mov ebp,esp
001D1A73 81 EC CC 00 00 00 sub esp,0CCh
001D1A79 53 push ebx
001D1A7A 56 push esi
001D1A7B 57 push edi
001D1A7C 8D BD 34 FF FF FF lea edi,[e
....