E02：并发进程（UNIX 进程与中断）

一、单项选择题

1. 响应外部中断的过程中，下列不是中断隐指令完成的工作的一项__B__。
A. 关中断　B. 保存通用寄存器的内容
C. 形成中断入口程序地址并送程序计数器　D. 保护断点外

2. 下列选项中，会导致进程从用户态切换到核心态的操作是__B__。
I. 整数除以零　II. sin()函数调用　III. IO 操作结束
A. I，II　B. I，III
C. II，III　D. I，II 和 III

3. 下列选项中，在用户态执行的是__A__。
A. 命令解释程序　B. 缺页中断处理程序
C. 进程调度程序　D. 时钟中断处理程序

4. UNIX V6++中，决定代码段是否与进程图象的可交换部分一起被换出到盘交换区的因素是__B__。
A. 内存空间的大小
B. 内存中共享该代码段的进程数量
C. 盘交换区的大小
D. 进程图象可交换部分的大小

5. 若共享某一代码段的进程共有 5 个，其中有三个进程图象的可交换部分在内存，其余 2 个进程图象的可交换部分在盘交换区，则该代码段的 Text 结构中，x_count 和 x_ccount 的值分别为__A__。
A. 5，3
B. 5，2
C. 5，0
D. 5，5

二、简答题：

6. UNIX V6++中，如果已知一个进程的 ID 号，如何获得该进程图象可交换部分和代码段的起始地址？如果当前进程图象的可交换部分在内存，且需要换到盘交换区，如何确定是否需要将代码段同时换出？

答：首先在 ProcessManager 类中的 process 数组中通过 ID 号找到对应的进程控制块 PCB（Process 类）。根据 proc 结构中的 p_flag 是否具有 SLOAD 标志位判断该进程的可交换部分是否在内存中。如果在内存，在 PCB 中的 p_addr 存放着可交换部分在物理内存中的起始地址；如果在盘交换区，则 p_addr 为进程图像可交换部分在盘交换区上的盘块号。

PCB 中的 p_textp 指向代码段 TEXT 结构的地址。如果 x_ccount>=1，则代码段在内存，x_caddr 为其内存起始地址；如果 x_ccount==0，则代码段在盘交换区，x_daddr 为其在盘交换区的起始盘块号。如果当前进程图像的可交换部分在内存需要换到盘交换区，需要检查其 text 类中的 x_ccount，若仅有该进程一个在内存中的进程，则将代码段同时换出，否则不交换代码段。

7. 多道计算机系统中为什么要有中断？

答：1. 保证了 CPU 和设备之间的并行操作；

2. 提供了进程执行内核代码的机会；
3. 多道程序并发的硬件基础。

三、应用分析题

8. 如果某一程序执行如下代码：

```
==========================================================
……
int main()
{
    int i , j;
    for (i=0;i<M;i++)
        for (j=0;j<M;j++)
              produce(i , j);
    …;
}
void produce ( int row, int column )
{
    int i;
    for (i=0;i<M;i++)
        matrixDes[row][column] += matrixOriginal[row][i] *
        matrixOriginal[i][column];
}
==========================================================
```

假设汇编后形成的机器指令为 1K，DATA 段长度为 0，BSS 段长度为 64 字节，

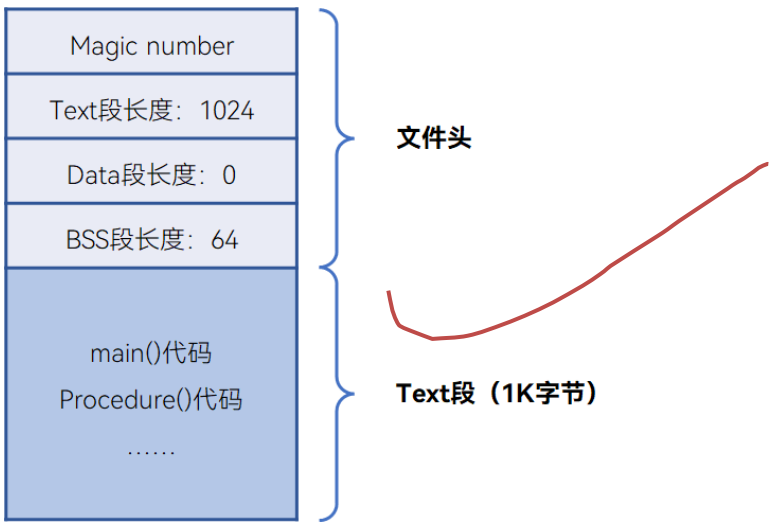
（1） 请绘制上述代码汇编后形成的可执行文件的结构；

（2） 分别创建进程 pa，pb，pc 执行上述可执行文件，其中，pa，pb 在内存，pc 在盘交换区上，请绘制所有进程的图象。

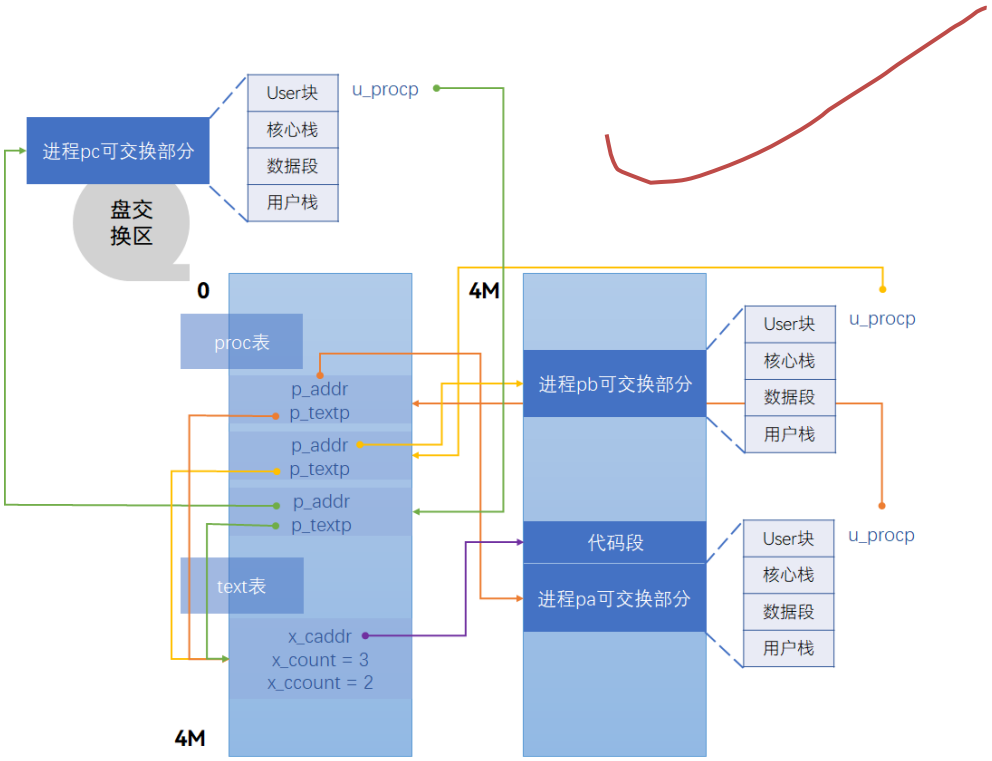
（3） 当 pa 执行到 produce 函数的 for 语句时，请绘制该进程的用户栈和核心栈的构成（图中需标出 main 函数中的 i 和 j，produce 函数中的 row 和 column，i 和 j 的位置）。

（4） 如果此时中断控制器发来一个中断请求，且 CPU 开中断，请绘制进程 pa 响应中断后，核心栈的变化情况。

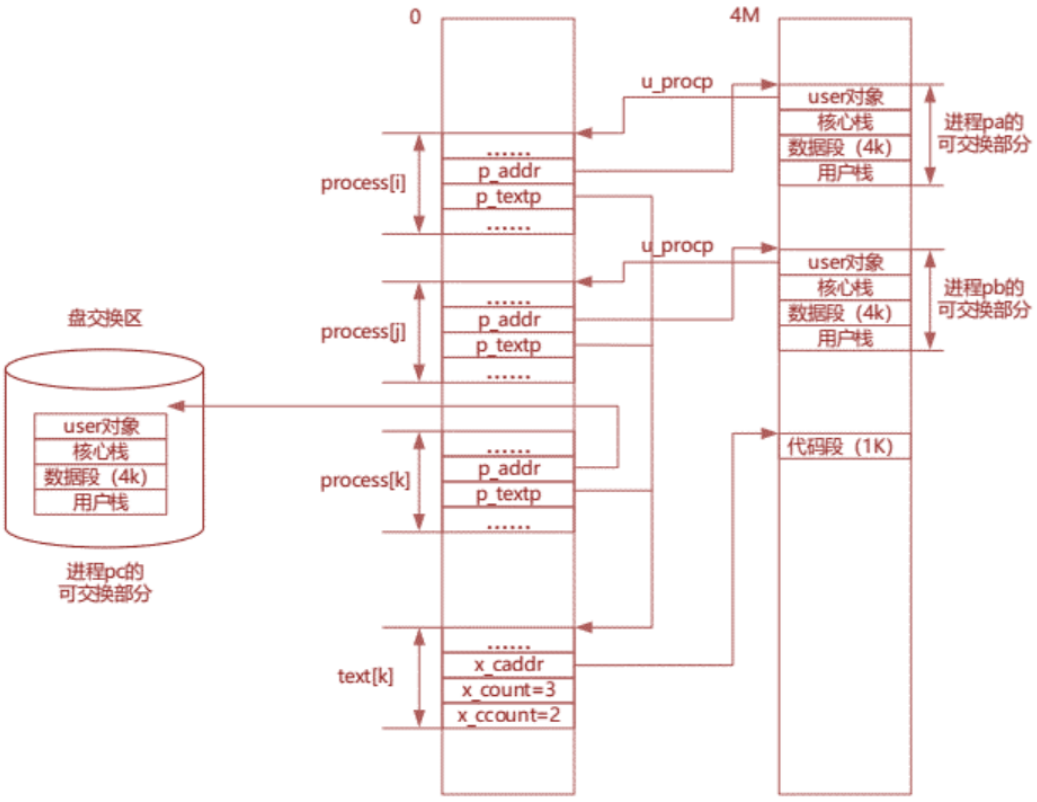
答：(1)



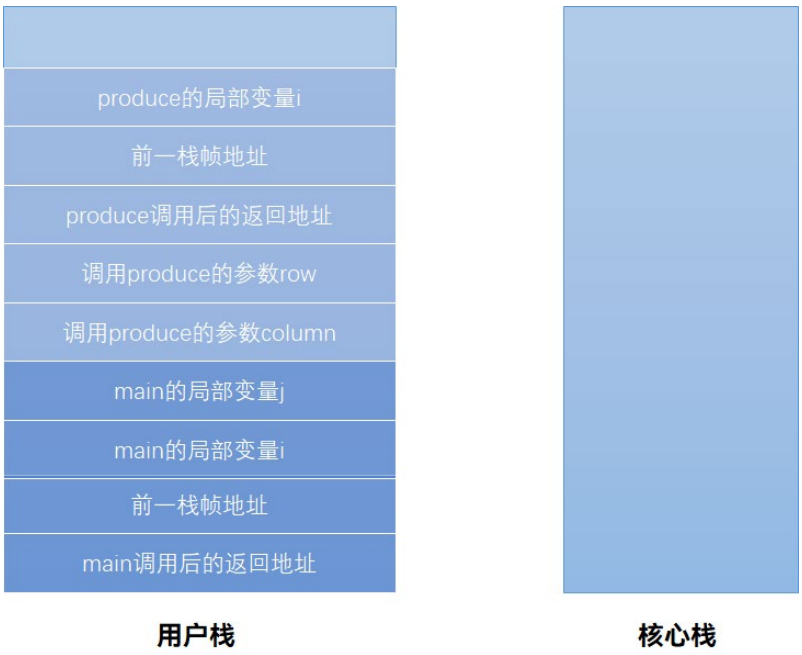
(2)


User块
核心栈
数据段
用户栈
u_procp
进程pc可交换部分
盘交换区
0
4M
proc表
p_addr
p_textp
p_addr
p_textp
p_addr
p_textp
text表
x_caddr
x_count = 3
x_ccount = 2
4M
进程pb可交换部分
User块
核心栈
数据段
用户栈
u_procp
代码段
进程pa可交换部分
User块
核心栈
数据段
用户栈
u_procp

标准答案：


0
4M
u_procp
user对象
核心栈
数据段（4k）
用户栈
进程pa的
可交换部分
process[i]
.....
p_addr
p_textp
.....
u_procp
user对象
核心栈
数据段（4k）
用户栈
进程pb的
可交换部分
process[j]
.....
p_addr
p_textp
.....
盘交换区
user对象
核心栈
数据段（4k）
用户栈
进程pc的
可交换部分
process[k]
.....
p_addr
p_textp
.....
代码段（1K）
text[k]
.....
x_caddr
x_count=3
x_ccount=2

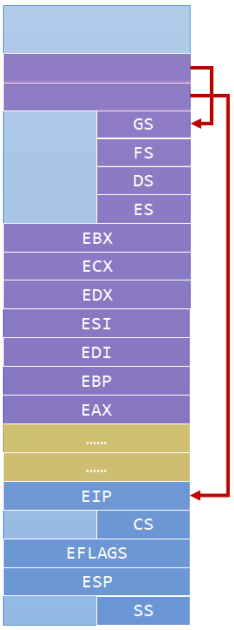
(3)


produce的局部变量i
前一栈帧地址
produce调用后的返回地址
调用produce的参数row
调用produce的参数column
main的局部变量j
main的局部变量i
前一栈帧地址
main调用后的返回地址
用户栈
核心栈

标准答案：

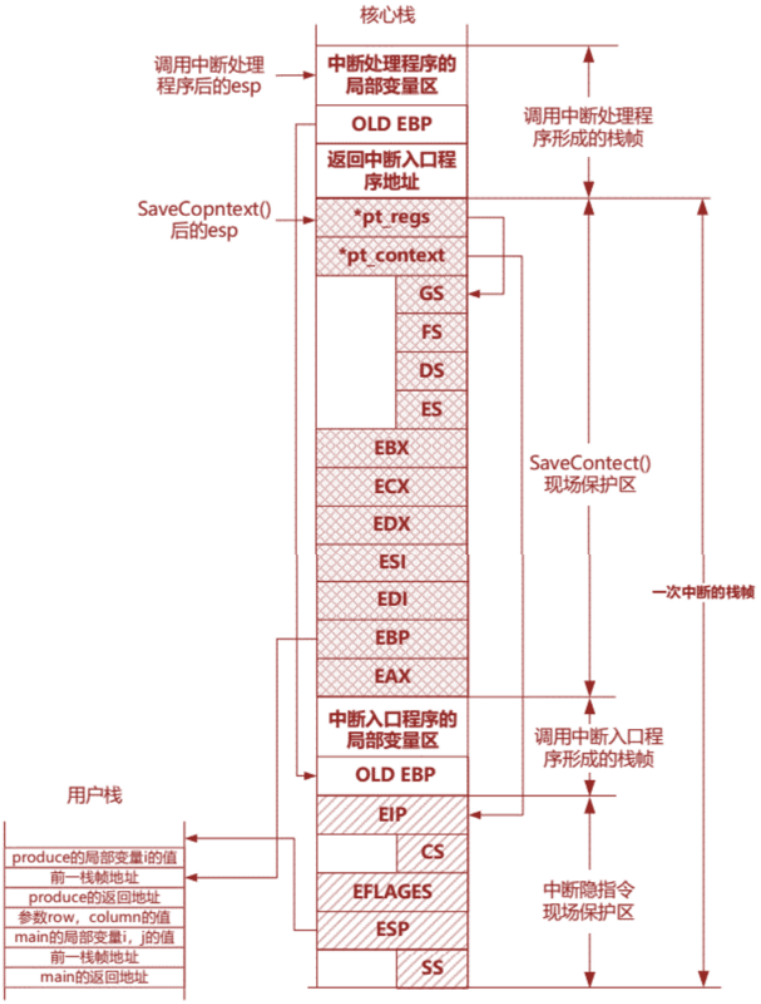

用户栈
esp
ebp
produce的局部变量i的值
前一栈帧地址
produce的返回地址
参数row，column的值
main的局部变量i，j的值
前一栈帧地址
main的返回地址
produce
栈帧
main栈帧
(无参数)

(4)


GS
FS
DS
ES
EBX
ECX
EDX
ESI
EDI
EBP
EAX
.....
.....
EIP
CS
EFLAGS
ESP
SS

核心栈

标准答案：


核心栈
调用中断处理程序后的esp
中断处理程序的局部变量区
OLD EBP
返回中断入口程序地址
调用中断处理程序形成的栈帧
SaveCopntext()后的esp
*pt_regs
*pt_context
GS
FS
DS
ES
EBX
ECX
EDX
ESI
EDI
EBP
EAX
SaveContect()现场保护区
一次中断的栈帧
中断入口程序的局部变量区
OLD EBP
调用中断入口程序形成的栈帧
用户栈
EIP
CS
EFLAGES
ESP
SS
中断隐指令现场保护区
produce的局部变量i的值
前一栈帧地址
produce的返回地址
参数row，column的值
main的局部变量i，j的值
前一栈帧地址
main的返回地址