1. **单项选择题（每小题1分，共20分）**
2. **计算机操作系统抽象表示时( A )是对处理器、主存和I/O设备的抽象表示。**

**A. 进程 B. 虚拟存储器 C. 文件 D. 虚拟机 *教材P10***

1. **每个信号类型都有一个预定义的默认行为，可能是（ D ）**

**A.进程终止 B.进程挂起直到被SIGCONT重启 C.进程忽略该信号 D.以上都是**

1. **当函数调用时，( B )可以在程序运行时动态地扩展和收缩。**

**A. 程序代码和数据区 B. 栈 C. 共享库 D. 内核虚拟存储器**

1. **C语句中的有符号常数，在（ A ）阶段转换成了补码**

**A.编译 B.连接 C.执行 D.调试**

1. **计算机常用信息编码标准中，字符0的编码不可能是16进制数（ C ）**

**A.30 B.30 00 C.00 D.00 30 *ASCII,大小端Unicode***

1. **C语言中float类型的数据0.1的机器数表示，错误的是（ C ）**

**A．规格化数 B.不能精确表示 C.与0.2有1个二进制位不同 D. 唯一的**

1. **递归函数程序执行时，正确的是（ B ）**

**A．使用了堆 B.可能发生栈溢出 C.容易有漏洞 D.必须用循环计数器**

1. **Y86-64的CPU顺序结构设计与实现中，分成（ B ）个阶段**

**A.5 B.6 C.7 D.8**

1. **关于Intel 的现代X86-64 CPU正确的是（ B ）**

**A．属于RISC B. 属于CISC C. 属于MISC D. 属于NISC**

1. **位于存储器层次结构中的最顶部的是( A )。**

**A. 寄存器 B. 主存 C. 磁盘 D. 高速缓存**

1. **连接时两个文件同名的弱符号，以（ C ）为基准**

**A．连接时先出现的 B．连接时后出现的 C.任一个 D.连接报错**

1. **Intel X86-64的现代CPU，采用（ C ）级页表**
2. **2 B.3 C.4 D.由BIOS设置确定**
3. **存储器垃圾回收时，内存被视为一张有向图，不能作为根结点的是（ D ）**
4. **寄存器 B.栈里的局部变量 C.全局变量 D.堆里的变量*教材P606***
5. **连接过程中，赋初值的局部变量名，正确的是（ D ）**

**A.强符号 B.弱符号 C.若是静态的则为强符号 D.以上都错**

1. **CPU一次访存时，访问了L1、L2、L3 Cache所用地址A1、A2、A3的关系（ B ）**

**A.A1>A2>A3 B.A1=A2=A3 C.A1<A2<A3 D.A1=A2<A3**

1. **C程序执行到整数或浮点变量除以0可能发生（ D ）**

**A.显示除法溢出错直接退出 B.程序不提示任何错误**

**C.可由用户程序确定处理办法 D.以上都可能**

1. **“Hello World”执行程序很小不到4k，在其首次执行时产生缺页中断次数（ D ）**

**A.0 B.1 C.2 D.多于2次 *代码、数据、堆栈等至少3个页面***

1. **同步异常不包括（ C ）**

**A.终止 B.陷阱 C.停止 D.故障**

1. **进程上下文切换不会发生在如下（ D ）情况**

**A.当前进程时间片用尽 B.外部硬件中断**

**C.当前进程调用系统调用 D.当前进程发送了某个信号**

1. **Linux下显示当前目录内容的指令为( A C ) //dir在有些系统下也能用**

**A.dir B.man C.ls D.cat**

**二、填空题 ( 每空1 分，共 10 分 )**

1. **64位系统中int数 -2的机器数二进制表示\_11111111 11111111 11111111 11111110\_\_\_**
2. **C语言函数中的整数常量都存放在程序虚拟地址空间的\_\_\_代码/数据\_\_\_\_\_段。**
3. **64位C语言程序中第一个参数采用\_\_\_\_\_\_\_\_RDI/寄存器\_\_\_\_\_\_\_传递。**
4. **C语言程序中的常量表达式的计算是由\_\_\_\_\_编译器\_\_\_\_\_\_完成的。**
5. **TLB(翻译后备缓冲器）俗称快表，是\_\_\_\_\_\_页表/PTE\_\_\_\_\_的缓存。**
6. **虚拟页面的状态有\_\_\_未分配\_\_\_\_\_\_\_\_、已缓存、未缓存共3种**
7. **I7的CPU，L2 Cache为8路的2M容量，B=64，则其Cache组的位数s=\_\_12\_\_。**

**C=S\*E\*B S=2\*1024\*1024/8/64 s=21-3-6 = 12**

1. **程序执行到A处继续执行后，想在程序任意位置还原到执行到A处的状态，通过\_\_\_非本地跳转\_/longjmp\_\_\_\_\_\_\_进行实现。**
2. **进程创建函数fork执行后返回\_\_\_\_2\_\_\_\_\_\_次。**
3. **Intel 桌面X86-64 CPU采用\_\_\_\_\_\_小\_\_\_\_\_\_\_端模式。**

**三、判断对错（每小题1分，共10分，在题前打√ X 符号）**

1. **（ X ）现代超标量CPU指令的平均周期接近于1个但大于1个时钟周期。**
2. **（ √）CPU无法判断参与加法运算的数据是有符号或无符号数。**
3. **（ X ）C浮点常数IEEE754编码的缺省舍入规则是四舍五入。**
4. **（ √ ）对unsigned int x，（x\*x）>=0总成立。**
5. **（ √）Y86-64的顺序结构实现中，寄存器文件读时是作为时序逻辑器件看待。**
6. **（ √）全相联Cache不会发生冲突不命中的情况。**
7. **（ X ）Linux系统调用中的功能号n就是异常号n 。**
8. **（ X ）fork的子进程中与其父进程同名的全局变量始终对应同一物理地址。**
9. **（ X ）动态存储器分配时显式空闲链表比隐式空闲链表的实现节省空间。**
10. **（ √）C语言中从int转换成float时，数字不会溢出，但可能舍入。**

**四、简答题（每小题5分，共20分）**

1. **简述C编译过程对非寄存器实现的int全局变量与非静态int局部变量处理的区别。包括存储区域、赋初值、生命周期、指令中寻址方式等。**

|  |  |  |
| --- | --- | --- |
|  | **int全局变量** | **int局部变量** |
| **存储区域** | 数据段 | 堆栈段 |
| **赋初值** | 编译时 int x=1; | 程序执行时，执行数据传送类指令 如 MOVL $1234, 8(RSP) |
| **生命周期** | 程序整个执行过程中都存在 | 进入子程序后在堆栈中存在(如执行subq $8, %rsp)子程序返回前清除消失 |
| **指令中寻址方式** | 其地址是个常数, 寻址如 movl 0x806808C, %eax | 通过rsp/rbp的寄存器相对寻址方式。 如类似 (%rsp) 或8(%rsp)或-8(%rbp)等 |

**答：**

**1分**

**1分**

**1分**

**1分**

***其他额外给1分：如回答全局变量序号符号解析与重定位，局部变量不需要等***

1. **什么是共享库（动态连接库）？简述动态链接的实现方法。**

**答：共享库（动态连接库）是一个.so的目标模块（elf文件），在运行或加载时，由动态链接器程序加载到任意的内存地址，并和一个和内存中的程序（如当前可执行目标文件）动态*完全连接*为一个可执行程序。使用它可节省内存与硬盘空间，方便软件的更新升级。如标准C库libc.so 。1分**

**2分 加载时动态链接：应用程序第一次加载和运行时，通过ld-linnux.so动态链接器重定位动态库的代码和数据到某个内存段，再重定位当前应用程序中对共享库定义的符号的引用，然后将控制传递给应用程序（此后共享库位置固定了并不变）。**

**2分 运行时动态链接：在程序执行过程中，通过dlopen/dlsym函数加载和连接共享库，实现符号重定位，通过dlclose卸载动态库。**

1. **简述Y86-64流水线CPU中的冒险的种类与处理方法。**

**答：数据冒险：3分 指令使用寄存器R为目的，瞬时之后使用R寄存器为源。处理方法有暂停： 通过在执行阶段插入气泡（bubble/nop），使得当前指令执行暂停在译码阶段；数据转发：增加valM/valE的旁路路径，直接送到译码阶段；加载使用冒险：指令暂停在取指和译码阶段，在执行阶段插入气泡（bubble/nop）**

**控制冒险：2分 分支预测错误：在条件为真的地址target处的两条指令分别插入1个bubble。 ret：在ret后插入3个bubble。**

1. **简述程序的局部性原理，如何编写局部性好的程序？**

**答：局部性原理: 1分 程序倾向于使用与最近使用过数据的地址接近或是相同的的数据和指令.** **时间局部性:** **最近引用的项很可能在不久的将来再次被引用，如代码和变量等；空间局部性:** **与被引用项相邻的项有可能在不久的将来再次被引用。**

**2分让通用或共享的功能或函数—最常见情况运行得快：专注在核心函数和内循环.**

**2分尽量减少每个循环内部的缓存不命中数量：反复引用变量是好的(时间局部性)—寄存器-编译器；步长为1的 参考模式是好的(空间局部性)---缓存是连续块**

**一旦从内存中读入数据对象，尽可能多的使用它，使得程序中时间局部性最大。**

**五、系统分析题（每小题5分，共20分）**

1. **某C程序(64位模式)的main函数参数argv地址为0x0000413433323110，其内容如下：**

**0x0000413433323110：30 31 32 33 34 41 00 00 33 31 32 33 34 41 00 00**

**0x0000413433323120：35 31 32 33 34 41 00 00 00 00 00 00 00 00 00 00**

**0x0000413433323130：31 43 00 30 00 32 42 00 38 00 31 31 32 32 00 30**

**0x0000413433323140：32 33 00 61 41 00 31 00 32 00 33 00 31 00 00 31**

**请写出 程序名：\_\_\_\_\_\_\_\_1C\_\_\_\_\_\_\_\_\_\_\_\_,本程序的参数个数\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**按顺序写出各个参数为\_\_\_\_\_\_\_\_\_\_0\_\_\_\_2B\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**提示： int main(int argc,char \*argv[]); 字符0、A、a的ASCII为0x30、0x41、0x61**

**//注意本题考核的是数据表示--元素为字符串的数组。同题4、5、6、21、22、33、34类**

1. **有下列C函数: 函数arith的汇编代码如下：**

**long arith(long x, long y, long z) arith:**

**{ xorq %rsi,%rdi**

**long t1 = \_\_\_\_\_（1）\_\_\_\_; leaq (%rdi,%rdi,4),%rax**

**long t2 = \_\_\_\_\_（2）\_\_\_\_; leaq (%rax,%rsi,2),%rax**

**long t3 = \_\_\_\_\_（3）\_\_\_\_; subq %rdx,%rax**

**long t4 = \_\_\_\_\_（4）\_\_\_\_; retq**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_（5）\_\_\_\_;**

**}**

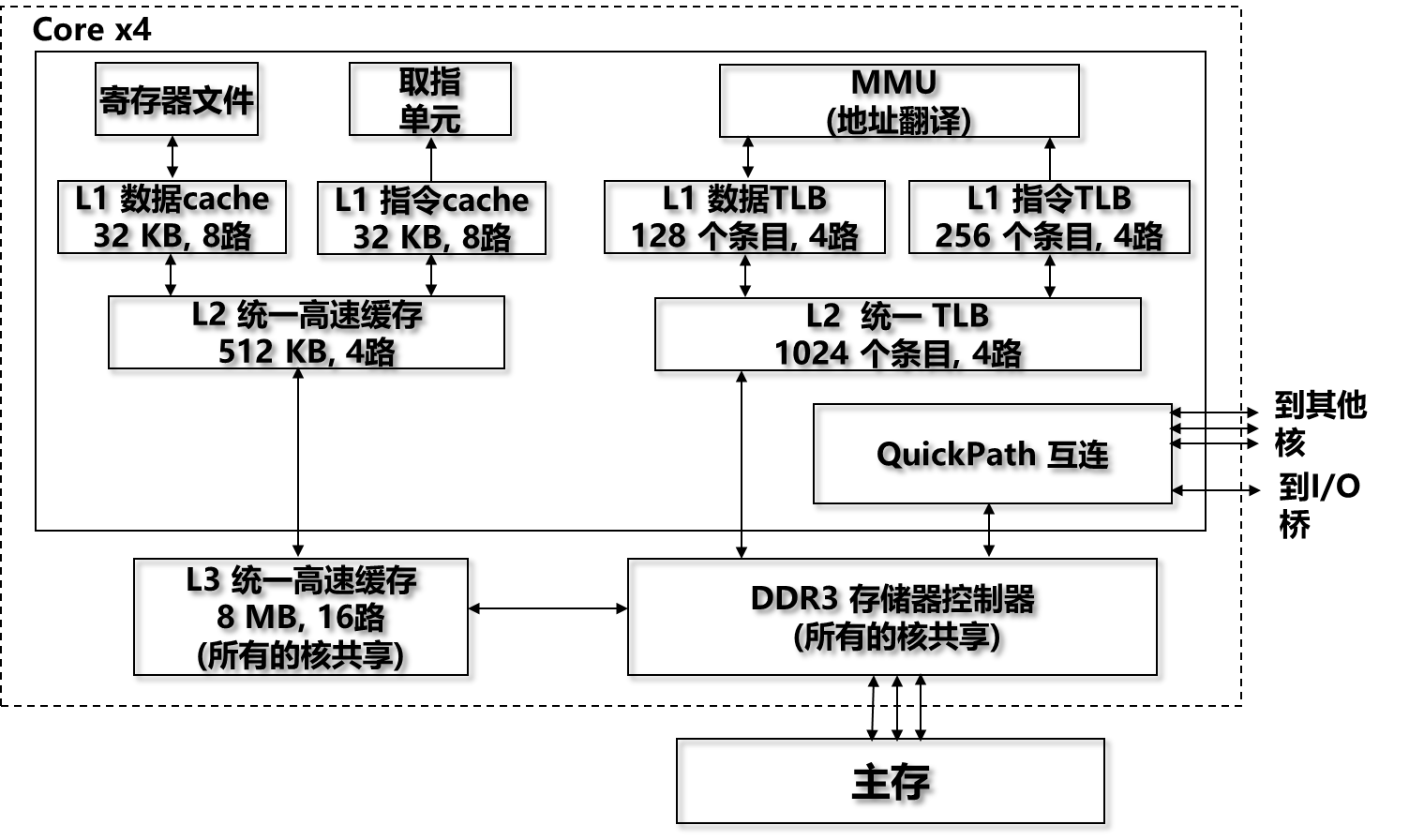
**请填写出上述C语言代码中缺失的部分**

**（1）\_\_\_\_\_x^y\_\_\_\_\_\_\_\_\_（2）\_\_\_\_\_\_t1+ t1 << 2 /5\*t1\_\_\_（3）\_\_\_t2 + y << 1\_\_\_\_\_\_\_\_**

**（4）\_\_\_\_\_t3--z\_\_\_\_\_\_\_（5）\_\_\_\_\_return t4\_\_\_\_\_\_\_\_\_ 4\*t1 2\*y**

**//注意64位参数顺序rdi、rsi、rdx, 这里lea只是把地址送rax而已，相当于计算**

1. **Intel I7 CPU的虚拟地址 48位，物理地址52位。其内部结构如下图所示，依据此结构，**

**每一页面4KB，分析如下项目：**

**虚拟地址中的VPN占\_\_36\_位；**

**其一级页表为\_\_512\_\_项。**

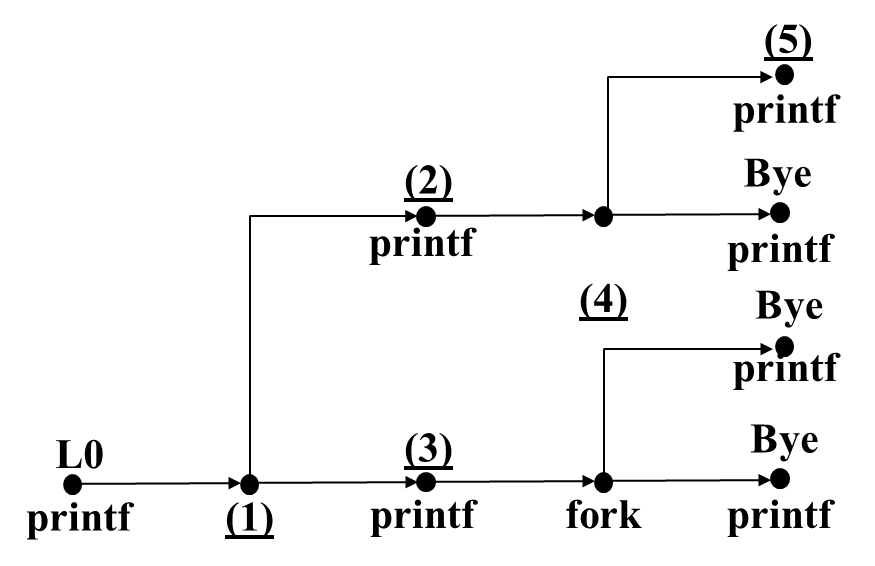
**L1数据TLB的组索引位数 TLBI为\_\_5\_\_\_位。**

***//32 \* 1024 = S \* 8 \* 64***

**L1数据Cache共\_\_64\_组。**

**//52-s6-b6**

**用物理地址访问L1数据Cache时， Cache标记CT占\_\_40\_\_位**

1. **C程序fork2的源程序与进程图如下：**

**void fork2()**

**{**

**printf("L0\n");**

**fork();**

**printf("L1\n");**

**fork();**

**printf("Bye\n");**

**}**

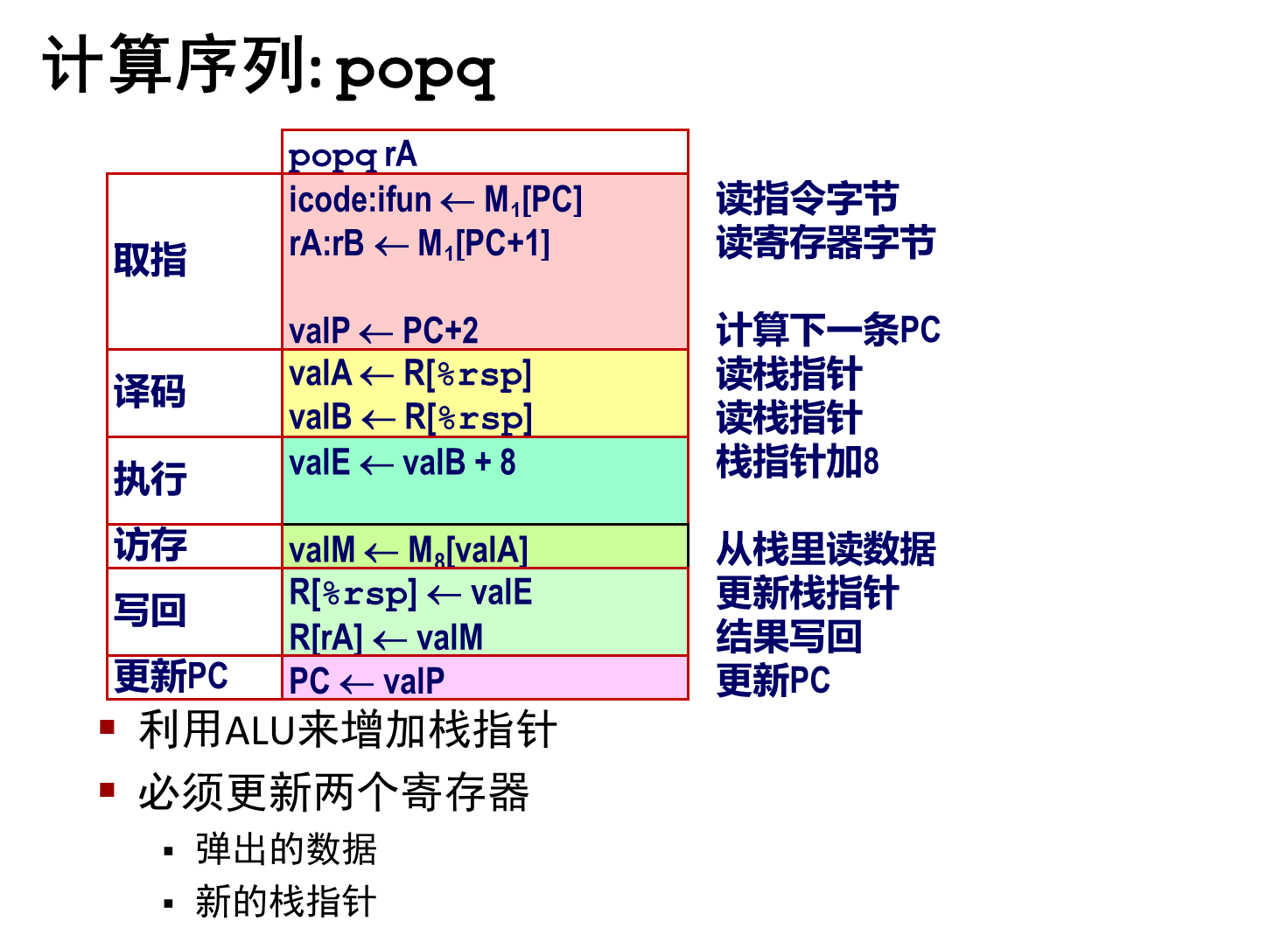
**请写出上述进程图中空白处的内容**

**（1）\_\_fork\_\_\_\_\_\_\_（2）\_\_\_\_L1\_\_\_\_\_（3）\_\_\_\_L1\_\_\_\_\_**

**（4）\_\_fork\_\_\_\_\_\_\_（5）\_\_\_Bye\_\_\_\_\_**

**六、综合设计题（每小题10分，共20分）**

1. **请写出Y86-64 CPU顺序结构设计与实现中，POP指令在各阶段的微操作。**



**1分**

**2分**

**2分**

**2分**

**2分**

**1分**

1. **程序优化: 矩阵c[n,n] = a[n,n] \* b[n,n] ，采用 48题 I7 CPU。块64B。**

**for(int i=0;i<n;i++)**

**for(int j=0;j<n;j++)**

**{**

**c[i,j]=0;**

**for(int k=0; k<n;k++)**

**c[i,j]+=a[i,k]\*b[k][j];**

**}**

**请针对该程序进行速度优化，写出优化后的程序，并说明优化的依据。**

从以下两方面给分：

1、采用的方法：（1）调整循环变量次序（2）循环展开（3）局部变量累积等

2、代码实现：基本正确（重在思路，不计较语法细节）

3、论述分析清晰、有理

1. **附加题（共10分）**
2. **在终端中的命令行运行显示“Hello World”的执行程序hello，结合进程创建、加载、缺页中断、到存储访问（虚存）……等等，论述hello是怎么一步步执行的。**

**包括但不限于以下内容：**

1. **shell接收命令**
2. **用fork创建子进程**
3. **execve函数加载进程**
4. **执行时如何如何会产生缺页异常/中断**
5. **利用VA访存的过程**
6. **缺页中断后的页面换入的方法、如何恢复运行**
7. **printf函数涉及的动态链接库的动态链接**
8. **调用printf函数涉及的"Hello World"字符串的获取**
9. **hello运行完毕后产生SIGCHLD的信号**
10. **父进程对其回收、资源释放等**
11. **…**