1. **单项选择题（每小题1分，共20分）**
2. **printf等函数的格式串在elf文件的（ C ）节**

**A. data B. bss C. rodata D. stack**

1. **两个有符号数运算结果是否超出范围，可查看CPU的（ B ）**

**A. CF B.OF C.SF D.ZF**

1. **在Linux下，汉字用（ D ）个字节编码。**

**A. 2 B. 3 C. 4 D. >=3**

1. **float浮点数1的阶码是（ C/A ）**

**A.0 B.1 C.127 D.128**

1. **局部变量数组赋初值0，通过（ D ）操作完成**

**A.预处理或编译 B.请求二进制0 C.OS加载时 D.机器指令**

1. **缓冲器溢出漏洞防范方法错误的是（ C ）**

**A．金丝雀 B.使用安全函数 C.加大局部变量占用空间 D. 编译加安全选项**

1. **流水线CPU设计中数据转发时增加旁路路径是为了避免（ A ）**

**A.数据冒险 B.加载/使用冒险 C.控制冒险 D.处理器异常**

1. **计算机是64位是指（ B）**

**A.数据总线64根 B.CPU中通用寄存器是64位的**

**C.安装的操作系统是64位的 D.CPU中所有寄存器都是64位的**

1. **进程的虚拟地址空间到磁盘文件的映射通过（ D ）**

**A．fork B. execve C.loader D.mmap**

1. **局部变量是 ( D )**

**A. 局部符号 B. 本地符号 C. 全局符号 D.以上都不是**

1. **CPU访问TLB、Cache时使用的地址分别是（ B ）**

**A．虚拟地址、虚拟地址 B．虚拟地址、物理地址**

**C. 物理地址、虚拟地址 D. 物理地址、物理地址**

1. **固态硬盘访问，修改数据时正确的是（ D ）**

**A.修改原来位置的页面数据 B. 修改原来位置的整块数据**

**C.写在已擦除的新页上 D. 写在已擦除的新块上**

1. **Intel X86-64的现代处理器，采用（ B ）级页表**
2. **2 B.4 C.8 D.可在OS中配置**
3. **如下数据在内存中地址最高的是（ B ）**
4. **命令行参数 B.环境变量 C.main的栈帧 D.当前子程序栈帧**
5. **库打桩不会发生在（ C ）**

**A.编译时 B.静态链接时 C.静态链接/运行时 D.动态链接/运行时**

1. **程序运行中按下Ctrl-Z,会发生（ B ）**

**A.当前进程终止 B. 当前进程停止 C.父进程停止 D.父进程终止**

1. **调用read()函数产生（ B ）**

**A.故障 B.陷阱 C.异步异常 D.进程切换**

1. **用fopen两次打开同一个文件，会产生两个（ B ）**

**A.描述符表 B.文件表项 C.v-node表项 D.stat结构**

1. **执行一条指令最不幸时需要访问（ D ）次各类存储器**

**A. 2 B.4 C.<=8 D.> 8**

1. **Hello World! 执行程序占3186字节，运行时占( D )页内存**
2. **1 B.2 C.4 D. > 4**

**二、填空题 ( 每空1 分，共 10 分 )**

1. **float 数 0的机器数有\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_个。**
2. **switch语句的机器级实现中，采用的跳转表存在elf文件的\_\_\_\_\_.rodata\_节。**
3. **C语言64位系统中函数超过6个以上的参数采用\_\_\_\_\_堆栈/+寄存器\_\_\_\_传递。**
4. **缓冲器溢出漏洞中，是用黑客程序的地址覆盖了\_\_\_\_\_返回地址\_\_\_\_\_完成的**
5. **电脑主板上内存条中的每个二进制位信息采用\_\_\_\_\_\_电容/(dram??)\_来存储。**
6. **fork后创建的子进程与父进程不同的信息是\_\_\_\_\_\_\_进程ID\_\_\_\_\_\_\_\_\_\_\_\_。**
7. **隐式空闲链表的空闲块合并有\_\_\_4\_/2/3（立即合并/延迟合并-有2）\_种策略。**
8. **Intel I7的CPU其TLB的每行的存储块Block是\_\_\_\_\_8\_\_\_字节。**
9. **运行一次，可返回多次的函数是\_\_\_\_\_\_setjmp/fork/sigsetjmp\_\_\_\_\_\_\_。**
10. **Intel I7 CPU的各级页表的元素个数为\_\_\_\_\_\_\_512\_\_\_\_\_\_\_\_\_\_\_\_。**

**三、判断对错（每小题1分，共10分，在题前打√ X 符号）**

1. **（ X ）无符号int整数比有符号int整数多。**
2. **（ √ ）C语言函数scanf是不安全的。**
3. **（ X ）浮点数计算后的舍入规则是四舍五入。**
4. **（ X ）流水线处理器的级数越多越好。**
5. **（ X ）Cache的行数必须是2的n次幂。**
6. **（ √ ）Ubuntu中数据段、代码段在内存的起始地址（段基址）是一样的。**
7. **（ X ）C语言程序中的内存垃圾可以完全回收。**
8. **（ X ）Intel 64位系统页表中PTE的物理页号PPN是64位的，占8个字节。**
9. **（ √ ）缺页异常处理子程序完成虚拟内存到物理内存的映射。**
10. **（ √ ）C的标准IO函数不能用在信号处理子程序中**

**四、简答题（每小题5分，共20分）**

1. **针对有符号及无符号整数的加法运算，CPU、编译器、程序员是怎么配合完成不同类型整数的数据表示、数据运算，并如何判断其结果是否超出范围的？**

**答：C程序员用unsigned 或signed（缺省，可不用）来区分数据类型，常数后加U表示无符号数。程序中可以自由进行比较、赋值、运算等。**

**CPU并不知道数据类型，只是按位进行加法操作，并按照逻辑规定设置CF、OF、ZF、SF、PF、AF等标志位。**

**编译器会将数据转换成相应的二进制编码（无符号数）或补码（有符号数），如果类型不一致，都转换成无符号数再进行操作。**

**数组操作之后，编译器根据不同数据类型选择不同的分支转移指令，可按照如上标志位，或无符号数用JA/JB等、有符号数用JG/JL等判断数据大小，并进行跳转。无符号溢出用JC、有符号溢出用JO判断。**

1. **Unix IO函数与C标准IO函数能否混合使用？为什么？并说明各自的适用范围。**

**答：不能。**

**应为UnixIO函数是不带缓冲的，C标准IO函数是带缓冲的。如混合使用会导致数据输出顺序与程序中发送顺序不一致的情况，从而出错。**

**UnixIO函数是异步信号安全的函数，可用于信号处理程序中，并适用于一些实时性要求高（高性能）的IO应用场合。**

**C标准IO函数带缓冲，能减少对IO设备的访问次数，大大改善IO的效率，如磁盘文件和终端文件等。**

1. **以Intel64位现代处理器为例，简述加快页表PTE访问、大大降低页表占用空间的相关技术。**

**答：采用TLB加快页表PTE的访问：采用高速缓冲存储器作为页表的Cache。TLB中保存最近常用的虚拟页号对应的页表条目PTE（含物理页号）。虚拟页数较少的进程页表可以完全在TLB中。**

**采用多级页表大大降低页表占用的空间：由于原有页表中大量的连续PTE条目都是未分配的，Intel64位CPU采用4级页表后，一级页表的大量条目其内容为NULL，同样二、三级页表也是如此，这样只有已分配页表条目采用4、3、2、1级页表。大大节省了页表空间。**

1. **程序执行 int x=y/c语句时，当c=0时程序执行结果是什么？并请结合异常、信号的概念及处理机制解释原因。**

**答：异常是指为响应某个事件将控制权转移到操作系统内核中的情况，每种异常都有一个异常号及对应的异常处理子程序（内核态）**

**信号是一条消息，它通知进程系统中发生了一个某种类型的事件，在进程进入运行态前由OS内核检查信号并执行其对应的信号处理子程序（用户态）。**

**程序执行到这条语句时会产生整数除法出错异常（异常号0），执行Divide\_Eror( )异常处理子程序,在异常处理子程序中，向当前进程发送一个SIGFPE信号（8号信号），而SIGFPE（信号处理子程序）的默认行为就是显示“Floating point exception (core dumped)”,终止并转储内存。**

**五、系统分析题（每小题5分，共20分）**

1. **请分析如下C程序，指出各符号对应的运行时内存区域**

**int i=100; .data \_\_\_\_\_\_\_\_\_i\_\_\_\_\_\_\_\_\_\_\_\_**

**char \*s=“Hello World!\n”; (i,s,a)**

**short a[1<<30]; .rodata\_\_\_\_\_\_\_\_s\_\_\_\_\_\_\_\_\_\_\_\_\_**

**void main(int argc,char \*argv[]) (“He……”“12^”)**

**{ ………………………… .bss \_\_\_\_\_\_\_a,count\_\_\_\_\_\_\_\_\_\_**

**static int count=0;**

**int \*p1,\*p2; .stack\_\_\_\_\_c,argc,argv\_\_\_\_\_\_\_**

**char c[100]=“1234567890ABCDEF”; (p1,p2)**

**p1=malloc(4096); .heap(brk堆)\_\_\_\_\_p1\_\_\_\_\_\_\_\_\_**

**p2=malloc(256\*1024); （多写p2 ???,\*p1,\*p2）**

**………………………… 注（有1个就算对）**

1. **某C函数的反汇编结果分别如下：**
   1. **401160: 83 ee 01 sub $0x1,%esi**
   2. **401163: b8 00 00 00 00 mov $0x0,%eax**
   3. **401168: 85 f6 test %esi,%esi**
   4. **40116a: 78 0b js 401177 <f+0x17>**
   5. **40116c: 48 63 d6 movslq %esi,%rdx**
   6. **40116f: 03 04 97 add (%rdi,%rdx,4),%eax**
   7. **401172: 83 ee 01 sub $0x1,%esi**
   8. **401175: eb f1 jmp 401168 <f+0x8>**
   9. **401177: c3 retq**

**请写出 函数f的返回值类型是\_\_\_\_\_\_int\_\_\_\_,参数的类型\_\_int \* 或int[]\_\_\_\_，\_\_int\_\_**

**第3、4条指令的功能\_\_\_\_\_\_若esi<0则程序结束并返回\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**第5、6条指令的功能\_\_\_\_\_参数1的第esi号元素内容加到累加器eax上\_\_\_\_\_\_**

1. **写出46题函数的C语言实现**

**答：**

**int sum(int a[],int n) 1分**

**{**

**int res=0;**

**int i; 1分**

**for(i=n-1;i>=0;i- -) 2分**

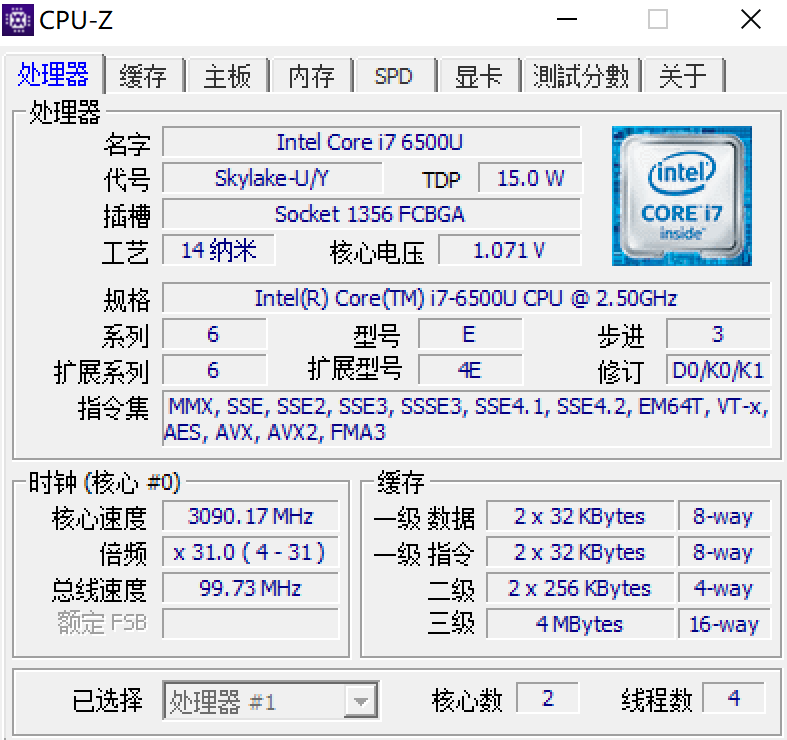
**res+=a[i]; 1分**

**return res;**

**}**

**while ???? 也算对 goto 也算对**

1. **某电脑使用CPUZ查看结果如下图所示，虚拟地址48位，物理地址47位，每页4KB，每块64B，分析并填写如下内容**



**1.此CPU的3级Cache都采用了**

**\_\_组相联\_\_类型的高速缓冲器。**

**(SRAM??)**

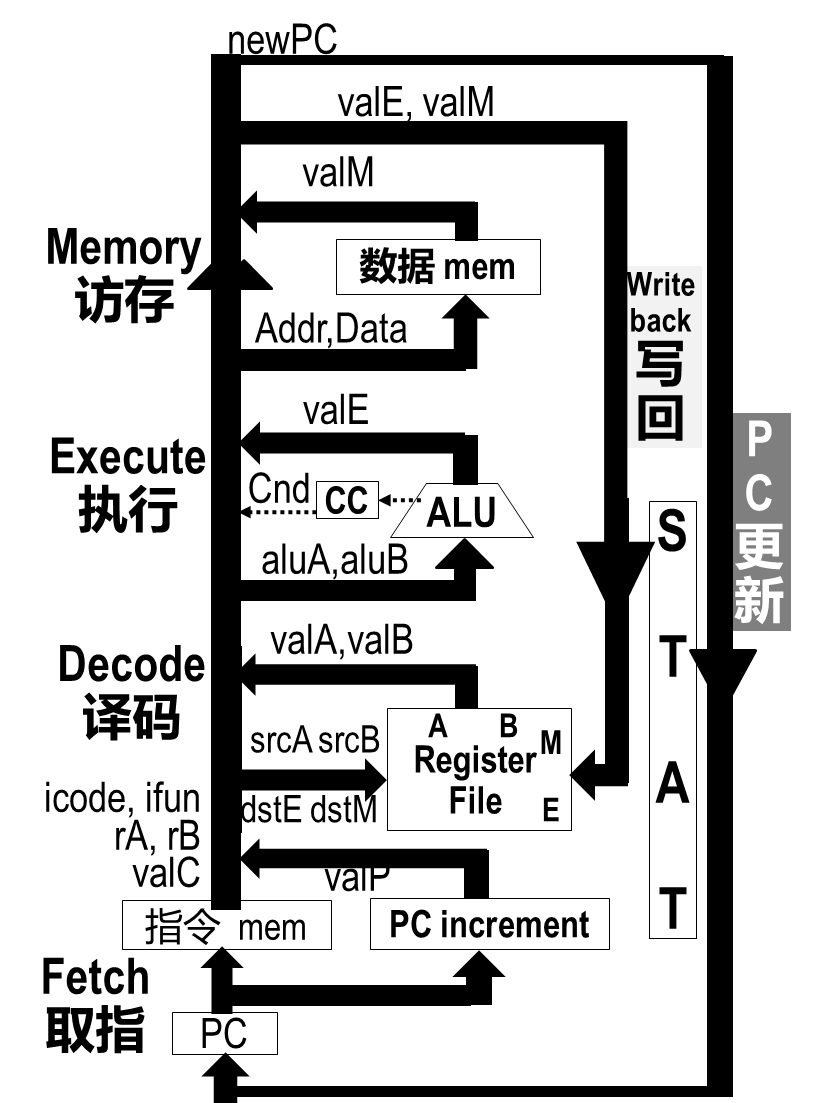
**2.L3Cache有\_\_4096/4K\_\_组**

**3.L2Cache的Tag有\_\_31\_\_\_位**

**4.L1Cache的组索引位数\_\_6\_\_\_\_**

**5.L1Cache的行数为\_8/512/1024**

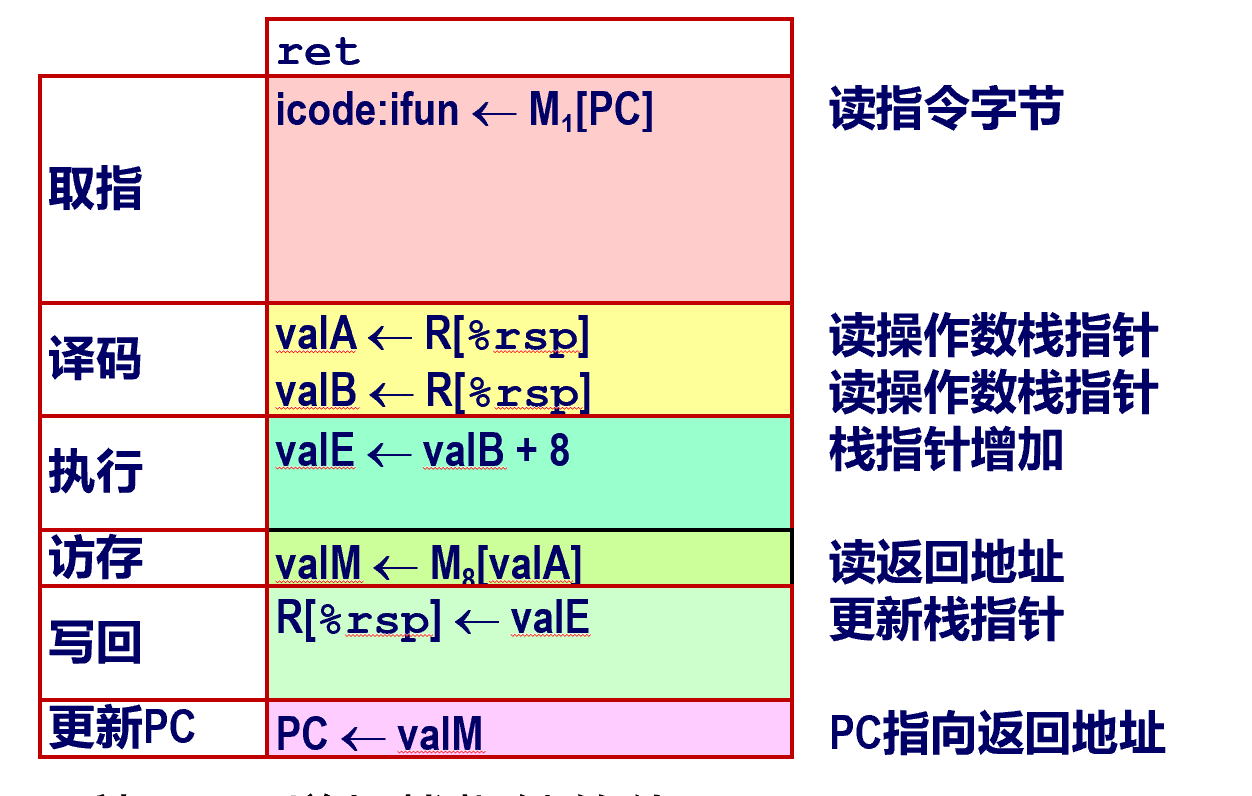
**/2048**

** 六、综合设计题（每小题10分，共20分）**

1. **Y86-64顺序结构的CPU，请按顺序写出ret（机器语言 0x90）**

**指令在各阶段的微操作**

**答：**



取指：1分 可以有valp🡨pc+1

更新PC：1分

其他每个阶段2分

1. **一个图像处理程序实现图像的平滑，其图像分辨率为1920\*1080，每一点颜色值为long类型，用long img[1920][1080]存储屏幕上的所有点颜色值。（long为64位）**

**平滑算法为：任一点的颜色值为其上下左右4个点颜色的平均值，即：**

**val[i][j]=(img[i-1][j]+img[i+1][j]+img[i][j-1]+img[i][j+1])/4。**

**请面向 48题的CPU，利用本课程学过的优化技术，编写程序，并说明你所采用的优化方法。**

**答：**

（1）一般有用的优化，共享共用子表达式、复杂指令简化

up = img[i-1][j] = img[(i-1)\*n + j ];

down = img[i+1][j] = img[(i+1)\*n + j ];

left = img[i][j-1] = img[i\*n + j-1 ];

right = img[i][j+1] = img[i\*n + j+1];

valij = (up + down + left + right)/4

优化为：

long \*inj = img+i\*n + j;

up = \*( inj – n);

down = \*( inj + n);

left = \*( inj – 1);

right = \*( inj + 1);

valij = (up + down + left + right) >> 2

（2）面向编译器的优化：用局部变量

（3）面向标量CPU优化：带分离的累加器的循环展开。通过比较不同展开因子L时的最小CPE，从而确定最优的L展开因子。

面向向量CPU优化：采用vaddpd及YMMi寄存器编程

（4）面向Cache优化：

空间局部性：重新排列（局部变量、循环变量顺序重排）提高空间局部性

时间局部性：分块，考虑到Cache 32K。

注意：如上四类优化方法，任选2种即可满分。第3种、第4种又可各自有两个方法任选。

有程序实现有瑕疵，若能多种方法集成在一起形成一个程序可加分直至本题目满分。

边缘点的外邻点颜色值为0，如没有考虑，不扣分。