主管领导审核签字

哈尔滨工业大学 2017 学年　秋 季学期

计算机系统（B）　试　题

题号	一	二	三	四	五	六	七	总分
得分								
阅卷人								

片纸鉴心　诚信不败

授课教师　姓名　学号　院系　密　封　线

一、 单项选择题（每小题 1 分，共 20 分）

1. 操作系统通过提供不同层次的抽象表示来隐藏系统实现的复杂性，其中（ D ）是对实际处理器硬件的抽象。
   A. 进程　B. 虚拟存储器　C. 文件　D. 指令集架构（ISA）
2. 为了使计算机运行得更快，现代 CPU 采用了许多并行技术，将处理器的硬件组织成若干个阶段并让这些阶段并行操作的技术是（ A ），该技术的 CPI 一般不小于 1。
   A. 流水线　B.超线程　C.超标量　D.向量机
3. 在进程的虚拟地址空间中，用户代码不能直接访问的区域是（ D ）
   A.程序代码和数据区　B.栈　C. 共享库　D. 内核虚拟内存区
4. C 语句中的全局变量，在（B ）阶段被定位到一个确定的内存地址
   A.编译　B.链接　C.执行　D.调试
5. 下列 16 进制数值中，可能是 Linux64 系统中 char*类型的指针值是（ C ）
   A.e4f9　B.b4cc2200　C.b811e5ffff7f0000　D.30
6. 关于 IEEE float 类型的数据+0.0 的机器数表示，说法错误的是（ B ）
   A. 是非规格化数　B.不能精确表示　C.+0.0 与-0.0 不同　D. 唯一的
7. 一个子进程终止或者停止时，操作系统内核会发送（ D ）信号给父进程。
   A. SIGKILL　B.SIGQUIT　C.SIGSTOP　D.SIGCHLD
8. Y86-64 的指令编码长度是（ A ）个字节
   A.1～10　B.32　C.64　D.128
9. 在 Y86-64 指令集体系结构中，程序员可见的状态不包括（ B ）
   A. 程序寄存器　B.高速缓存　C.条件码　D.程序状态
10. 下列各种存储器中存储速度最快的是(A)。
    A. 寄存器　B. 主存　C. 磁盘　D. 高速缓存
11. 链接时两个同名的强符号，以哪种方式处理？（ D ）
    A. 链接时先出现的符号为准　B. 链接时后出现的符号为准
    C.任一个符号为准　D. 链接报错
12. 某 CPU 使用 32 位虚拟地址和 4KB 大小的页时，需要 PTE 的数量是（ C ）
    A. 16　B.8　C.1M　D.512K
13. 动态内存分配时的块结构中，关于填充字段的作用不可能的是（C ）
    A. 减少外部碎片　B.满足对齐　C.标识分配状态　D.可选的

14. 链接过程中，带 static 属性的全局变量属于（ B ）
A. 全局符号 B. 局部符号 C. 外部符号 D. 以上都错
15. 虚拟内存系统中的虚拟地址与物理地址之间的关系是（ B ）
A. 1 对 1 B. 多对 1 C. 1 对多 D. 多对多
16. X86-64 中，通过寄存器传递整型参数时，第一个参数用寄存器（ A ）访问
A. %rdi B. %edi C. %rsi D. %edi
17. 虚拟内存发生缺页时，缺页中断是由（ D ）触发
A. 内存 B. Cache L1 C. Cache L2 D. MMU
18. 进程从用户模式进入内核模式的方法不包括（ C ）
A. 中断 B. 陷阱 C. 复位 D. 故障
19. 内核为每个进程维持一个上下文，不属于进程上下文的是（ D ）
A. 寄存器 B. 进程表 C. 文件表 D. 调度程序
20. Linux 进程终止的原因可能是（ D ）
A. 收到一个信号 B. 从主程序返回 C. 执行 exit 函数 D. 以上都是

二、填空题（ 每空 1 分，共 10 分 ）

21. C 语言中 short 类型-2 的机器数二进制表示为_____0xfffffffe_____。
22. C 语言中的 double 类型浮点数用___64___位表示。
23. 64 位 C 语言程序在函数调用时第二个整型参数采用寄存器____rsi____传递。
24. 链接器经过___符号解析_____和重定位两个阶段，将可重定位目标文件生成可执行目标文件。
25. 虚拟内存系统借助____页表_____这一数据结构将虚拟页映射到物理页。
26. Linux 虚拟内存区域可以映射到普通文件和___匿名文件_____，这两种类型的对象中的一种。
27. I7 的 CPU，L2 Cache 为 8 路的 2M 容量，B=64，则其 Cache 组的位数 s=__12___。
28. 非本地跳转中的 setjmp 函数调用一次，返回____多次_____次。
29. 进程加载函数 execve，如调用成功则返回____0____次。
30. 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）

31. （ √ ）现代超标量 CPU 指令的平均周期通常小于 1 个时钟周期。
32. （ x ）CPU 无法判断加法运算的和是否溢出。
33. （ x ）C 浮点常数 IEEE754 编码的缺省舍入规则是向上舍入。
34. （ √ ）C 语言中的有符号数强制转换成无符号数时位模式不会改变。
35. （ x ）Y86-64 的顺序结构实现中，寄存器文件写时是作为组合逻辑器件看待。
36. （ x ）直接映射 Cache 一定会发生冲突不命中的情况。
37. （ x ）进程一旦终止就不再占用内存资源。
38. （ √ ）execve 加载新程序时会覆盖当前进程的地址空间，但不创建新进程。
39. （ x ）动态内存隐式分配是指应用隐式地分配块并隐式地释放已分配块。
40. （ x ）C 语言中从 double 转换成 float 时，值可能溢出，但不可能被舍入。

四、简答题（每小题 5 分，共 20 分）

41. 结合下面的程序段，解释局部性。

```
int cal_array_sum(int *a,int n){
  int sum = 0;
  for (int i = 0; i < n; i++)
    sum += a[i];
  return sum;
}
```

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数据和指令.包括时间局部性和空间局部性。

空间局部性（2 分）：1、对数据的引用：顺序访问数组元素 （步长为 1 的引用模式）

2、对指令的引用：顺序读取指令

时间局部性（2 分）：1、对数据的引用：变量 sum 在每次循环迭代中被引用一次

2、对指令的引用：重复循环执行 for 循环体

42. 什么是静态库？使用静态库的优点是什么？

（定义 3 分）编译时将所有相关的目标模块打包成为一个单独的文件用作链接器的输入，这个文件称为静态库。（优点 2 分）使用时，通过把相关函数编译为独立模块，然后封装成一个单独的静态库文件，应用程序可以通过在命令行上指定单独的文件名来使用这些在库中定义的函数，链接器只需复制被程序引用的目标模块，减少了可执行文件在磁盘和内存中的大小；同时，应用程序员只需包含较少的库文件的名字。

43. 参照 Y86-64 流水线 CPU 的实现，说明流水线如何工作。

概念 2 分，结合示例 3 分。

流水线化的系统，待执行的任务被划分成若干个独立的阶段，将处理器的硬件也组织成若干个单元，让各个独立的任务阶段在不同的硬件单元上一次执行，从而使多个任务并行操作。如 Y86-64 将指令执行分为取指、译码、执行、访存、写回 5 个阶段，通过在每个阶段插入流水线寄存器，利用时钟信号控制流水线的时序和操作，理想情况下可实现 5 条指令的同时运行。

44. 列举几种程序优化的方法，并简述其原理。

至少 2 种，每种 2.5 分

1） 代码移动，通过将代码从循环中移出减少计算执行的频率

2） 用简单方法替代复杂操作，如移位、加替代乘法/除法

3） 共享共用子表达式，重用表达式的一部分

减少过程调用，消除不必要的内存引用，循环展开。。。

五、系统分析题（每小题 5 分，共 20 分）

45. 已知内存和寄存器中的数值情况如下：

内存地址	值
0x100	0xff
0x104	0xAB
0x108	0x13
0x10c	0x11

寄存器	值
%rax	0x100
%rcx	0x1
%rdx	0x3

请填写下表，给出对应操作数的值：

操作数	值
%rax	0x100
(%rax)	0xff
9(%rax,%rdx)	0x11
0xfc(,%rcx,4)	0xff
(%rax,%rdx,4)	0x11

46. 有下列 C 函数:

```
long arith(long x, long y, long z)
{
    long t1 = _______(1)_______;
    long t2 = _______(2)_______;
    long t3 = _______(3)_______;
    long t4 = _______(4)_______;
    ___________(5)_____________;
}
```

函数 arith 的汇编代码如下：

```
arith:
    orq     %rsi,%rdi
    sarq    $3,%rdi
    notq    %rdi
    movq    %rdx,%rax
    subq    %rdi,%rax
    retq
```

请填写出上述 C 语言代码中缺失的部分

（1）____x|y_____ （2）___t1>>3_____ （3）___~t2_____

（4）_____z-t3_____ （5）___return t4_____

47. 假设：某 CPU 的虚拟地址 14 位；物理地址 12 位；页面大小为 64B；TLB 是四路组相联，共 16 个条目；L1 数据 Cache 是物理寻址、直接映射，行大小为 4 字节，总共 16 个组。分析如下项目：

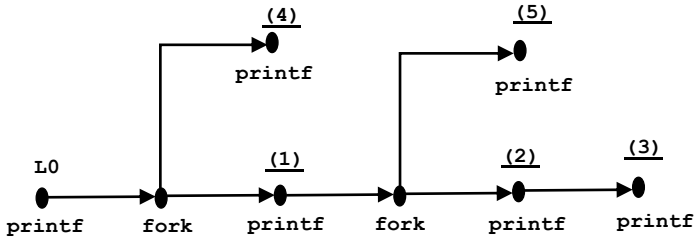
（1）虚拟地址中的 VPN 占_8_位；物理地址的 PPN 占__6____位。

（2）TLB 的组索引位数 TLBI 为___2____位。

（3）用物理地址访问 L1 数据 Cache 时，Cache 的组索引 CI 占___4___位，Cache 标记 CT 占__6___位。

48. C 程序 forkB 的源程序与进程图如下：

```
void forkB()
{
    printf("L0\n");
    if(fork()!=0){
       printf("L1\n");
       if(fork()!=0){
           printf("L2\n");
       }
    }
    printf("Bye\n");
}
```



请写出上述进程图中空白处的内容

（1）____L1_____（2）_____L2_____（3）____Bye_____

（4）___Bye_____（5）____Bye_____

六、综合设计题（每小题 10 分，共 20 分）

49. 请写出 Y86-64 CPU 顺序结构设计与实现中，mrmovq 指令在各阶段的操作。

每个阶段 2 分

阶段	mrmovq D(rB), rB
取指	icode:ifun ← M1[PC] rA:rB ← M1[PC+1] valC ← M8[PC+2] valP ← PC+10
译码	valB ← R[rB]
执行	valE ← valB + valC
访存	valM ← M8[valE]
写回	R[rA] ← valM
更新 PC	PC ← valP

50.向量元素和计算的相关程序如下，请改写或重写计算函数 vector_sum，进行速度优化，并简要说明优化的依据。

```
/*向量的数据结构定义 */
typedef struct{
        int len;      //向量长度，即元素的个数
        float *data; //向量元素的存储地址
} vec;

/*获取向量长度*/
int vec_length(vec *v){return v->len;}

/* 获取向量中指定下标的元素值，保存在指针参数 val 中*/
int get_vec_element(*vec v, size_t idx, float *val){
    if (idx >= v->len)
      return 0;
    *val = v->data[idx];
    return 1;
}

/*计算向量元素的和*/
void vector_sum(vec *v, float *sum){
    long int i;
    *sum = 0;//初始化为 0
    for (i = 0; i < vec_length(v); i++) {
        float val;
        get_vec_element(v, i, &val);//获取向量 v 中第 i 个元素的值，存入 val 中
        *sum = *sum + val;      //将 val 累加到 sum 中
    }
}
```

采用任一种优化方法都 10 分

```
void vector_sum(vec *v, float *sum)
{
   long int i;
   long int length = vec_length(v);
   float *d = get_vec_start(v);
   float t = 0;
   for (i = 0; i < length; i++)
      t = t + d[i];
   *dest = t;
}
```

把函数 vec_length 移到循环外

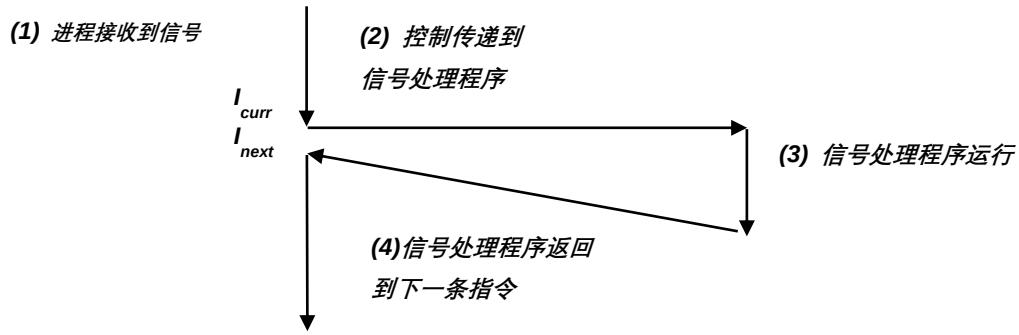
避免每个循环的边界检查

用临时/局部变量累积结果

七、 附加题（共 10 分）

51. Linux 如何处理信号？应当如何编写信号处理程序？谈谈你的理解。

（5 分）Linux 通过软中断（陷阱）方式实现信号机制，包括信号发送和信号接收两个阶段。处理过程如下



信号处理程序的编写原则：5 分

1) 处理程序尽可能简单
2) 在处理程序中只调用异步信号安全的函数
3) 确保其他处理程序不会覆盖当前的 **errno**
4) 阻塞所有信号保护对共享全局数据结构的访问
5) 用 **volatile** 声明全局变量
6) 用 **sig_atomic_t** 声明标志