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（A）　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. 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*
2. 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
3. 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
4. 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
5. 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*
6. 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
7. 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
8. 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
9. 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
10. 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
11. 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
12. Intel X86-64 的现代 CPU，采用（ C ）级页表
    A. 2　B. 3　C. 4　D. 由 BIOS 设置确定
13. 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 ）
    A. 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*
14. 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
15. 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

16. 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

17. “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*

18. 同步异常不包括（ C ）
A. 终止　B. 陷阱　C. 停止　D. 故障

19. 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

20. 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 ）

21. 64 位系统中 int 数 -2 的机器数二进制表示 11111111 11111111 11111111 11111110

22. 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。

23. 64 位 C 语言程序中第一个参数采用 RDI/寄存器 传递。

24. C 语言程序中的常量表达式的计算是由 编译器 完成的。

25. TLB(翻译后备缓冲器）俗称快表，是 页表/PTE 的缓存。

26. 虚拟页面的状态有 未分配、已缓存、未缓存共 3 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 。
C=S*E*B　S=2*1024*1024/8/64　s=21-3-6 = 12

28. 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。

29. 进程创建函数 fork 执行后返回 2 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. （ 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。
32. （ √）CPU 无法判断参与加法运算的数据是有符号或无符号数。
33. （ X ）C 浮点常数 IEEE754 编码的缺省舍入规则是四舍五入。
34. （ √）对 unsigned int x，（x*x）>=0 总成立。
35. （ X ）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。
36. （ √）全相联 Cache 不会发生冲突不命中的情况。
37. （ X ）Linux 系统调用中的功能号 n 就是异常号 n 。
38. （ 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。
39. （ 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。
40. （ √）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）

41. 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。

答：

		int 全局变量	int 局部变量
1 分	**存储区域**	数据段	堆栈段
1 分	**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)
1 分	**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
1 分	**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

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

42. 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。

43. 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。

44. 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）

45. 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

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:
xorq    %rsi,%rdi
leaq    (%rdi,%rdi,4),%rax
leaq    (%rax,%rsi,2),%rax
subq    %rdx,%rax
retq
```

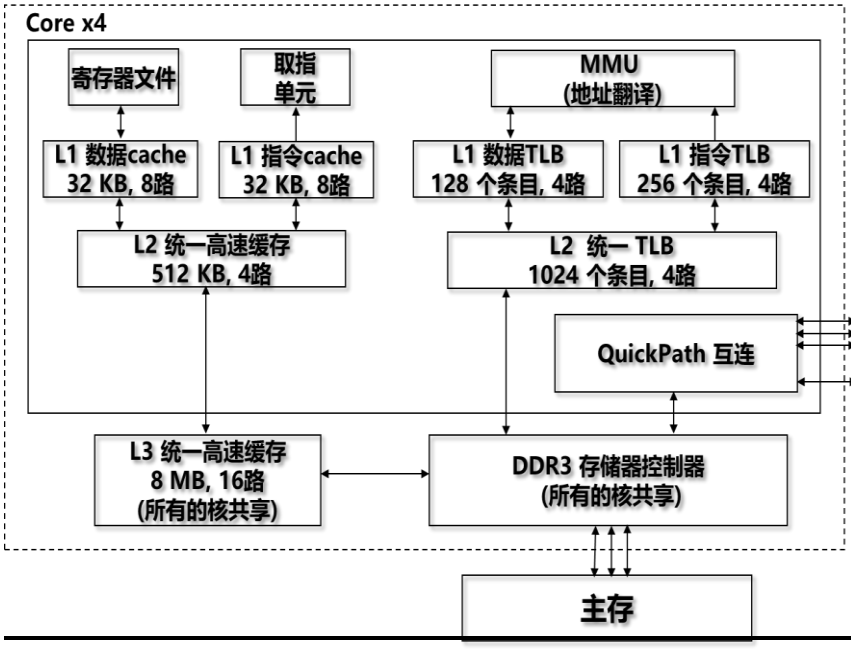
请填写出上述 C 语言代码中缺失的部分

(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

47. 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*

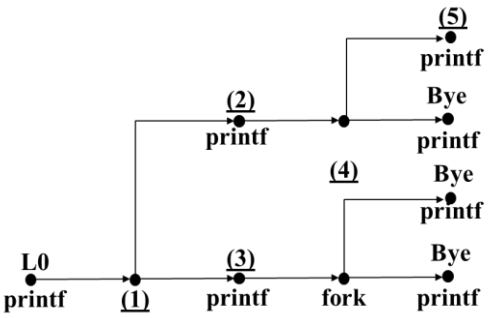
L1 数据 Cache 共__64__组。

//52-s6-b6

用物理地址访问 L1 数据 Cache 时， Cache 标记 CT 占__40__位

48. 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）

49. 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。

计算序列: popq

	popq rA	
取指	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2	读指令字节 读寄存器字节 计算下一条PC
译码	valA ← R[%rsp] valB ← R[%rsp]	读栈指针 读栈指针
执行	valE ← valB + 8	栈指针加8
访存	valM ← M_8[valA]	从栈里读数据
写回	R[%rsp] ← valE R[rA] ← valM	更新栈指针 结果写回
更新PC	PC ← valP	更新PC

1 分　2 分　2 分　2 分　2 分　1 分

- 利用ALU来增加栈指针
- 必须更新两个寄存器
  - 弹出的数据
  - 新的栈指针

50. 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

院系_____ 学号_____ 姓名_____ 授课教师_____ 密 封 线

七、 附加题（共 10 分）

51. 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) …