北京大学信息科学技术学院考试试卷

**考试科目：** 计算机系统导论  **姓名：**  **学号：**

**考试时间：** 2014 年 1 月 7 日 **任课教师:**

## 第一题 单项选择题（每小题1.5分，共30分）

1、对于IEEE浮点数，如果减少1位指数位，将其用于小数部分，将会有怎样的效果？答：（ ）

A. 能表示更多数量的实数值，但实数值取值范围比原来小了。  
B. 能表示的实数数量没有变化，但数值的精度更高了。  
C. 能表示的最大实数变小，最小的实数变大，但数值的精度更高。  
D. 以上说法都不正确。

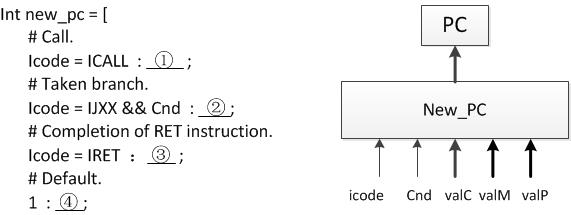
2、按照教材描述的原则，对于x86\_64程序，在callq指令执行后，函数的第一个参数一般存放在哪里？答：（ ）

A. 8(%rsp) B. 4(%rsp) C. %rax D. %rdi

3、已知变量x的值已经存放在寄存器eax中，现在想把5x+7的值计算出来并存放到寄存器ebx中，如果不允许用乘法和除法指令，则至少需要多少条IA-32指令完成该任务？答：（ ）

A. 1条 B. 3条 C. 2条 D. 4条

4、 在Y86的SEQ实现中，PC（Program Counter，程序计数器）更新的逻辑结构如下图所示，请根据HCL描述为①②③④选择正确的数据来源。



其中：Icode为指令类型，Cnd为条件是否成立，valC表示指令中的常数值，valM表示来自返回栈的数据，valP表示PC自增。

① ：A）valC B）valM C）valP

② ：A）valC B）valM C）valP

③ ：A）valC B）valM C）valP

④ ：A）valC B）valM C）valP

（5~6） 如果直接映射高速缓存（Cache）的大小是4KB，并且块大小（block）大小为32字节。

5、请问它每路（way）有多少行（line）？ 答：（ ）

A. 128 B. 64 C. 32 D. 1

6、如果数据访问的地址序列为0->4->16->132->232->4096->160（以字节为单位），请问一共发生多少次替换？答：（ ）

A. 0 B. 1 C. 2 D. 3

7、下列程序运行的结果是什么？答：（ ）

/\* main.c \*/

int i=0;

int main()

{

foo();

return 0;

}

/\* foo.c \*/

int i=1;

void foo()

{

printf(“%d”, i);

}

A. 编译错误 B. 链接错误

C. 段错误 D. 有时打印输出1，有时打印输出0；

8、在链接时，对于什么样的符号一定不需要进行重定位？答：（ ）

A．不同C语言源文件中定义的函数

B．同一C语言源文件中定义的全局变量

C．同一函数中定义时不带static的变量

D．同一函数中定义时带有static的变量

9、关于信号的描述，以下不正确的是哪一个？答：（ ）

A. 在任何时刻，一种类型至多只会有一个待处理信号

B. 信号既可以发送给一个进程，也可以发送给一个进程组

C. SIGTERM和SIGKILL信号既不能被捕获，也不能被忽略

D. 当进程在前台运行时，键入Ctrl-C，内核就会发送一个SIGINT信号给这个前台进程

10、下面关于非局部跳转的描述，正确的是（ ）

A. setjmp可以和siglongjmp使用同一个jmp\_buf变量

B. setjmp必须放在main()函数中调用

C. 虽然longjmp通常不会出错，但仍然需要对其返回值进行出错判断

D. 在同一个函数中既可以出现setjmp，也可以出现longjmp

11、假设有一台64位的计算机的物理页块大小是8KB，采用三级页表进行虚拟地址寻址，它的虚拟地址的VPO（Virtual Page Offset，虚拟页偏移）有13位，问它的虚拟地址的VPN（Virtual Page Number，虚拟页号码）有多少位？答：（ ）

A. 20 B. 27 C. 30 D. 33

12、进程P1通过fork()函数产生一个子进程P2。假设执行fork()函数之前，进程P1占用了53个（用户态的）物理页，则fork函数之后，进程P1和进程P2共占用\_\_\_\_\_\_\_个（用户态的）物理页；假设执行fork()函数之前进程P1中有一个可读写的物理页，则执行fork()函数之后，进程P1对该物理页的页表项权限为\_\_\_\_\_\_\_\_。上述两个空格对应内容应该是（ ）

A. 53，读写 B. 53，只读 C. 106，读写 D. 106，只读

13、下列哪个例子是外部碎片？答：（ ）

A. 分配块时为了字节对齐而多分配的空间

B. 空闲块中互相指向的指针所占据的空间

C. 多次释放后形成的不连续空闲块

D. 用户分配后却从未释放的堆空间

14、用带有header和footer的隐式空闲链表实现分配器时，如果一个应用请求一个3字节的块，下列说法哪一项是错误的？答：（ ）

A. 搜索空闲链表时，存储利用率为：best fit > next fit > first fit

B. 搜索空闲链表时，吞吐率为：next fit > first fit > best fit

C. 在x86机器上，malloc(3)实际分配的空闲块大小可能为8字节

D. 在x64机器上，malloc(3)返回的地址可能为2147549777

15、考虑如下代码，假设result.txt的初始内容是“123”。

int main(int argc, char\*\* argv)

{

int fd1 = open("result.txt", O\_RDWR);

char str[] = "abc";

char c;

write(fd1, str, 1);

read(fd1, &c, 1);

write(fd1, &c, 1);

return 0;

}

在这段代码执行完毕之后，result.txt的内容是什么？（假设所有的系统调用都会成功）答：（ ）

A．a22 B．a21 C．a13 D．abb

16、已知如下代码段

write(fd1, str1, strlen(str1));

write(fd2, str2, strlen(str2));

可以在原本为空的文件ICS.txt中写下字符串 I love ICS!

对于下面这些对于变量fd1, fd2, str1, str2的定义：

(1)

int fd1 = open("ICS.txt", O\_RDWR);

int fd2 = open("ICS.txt", O\_RDWR);

char \*str1 = "I love ";

char \*str2 = "ICS!";

(2)

int fd1 = open("ICS.txt", O\_RDWR);

int fd2 = dup(fd1);

char \*str1 = "I love ";

char \*str2 = "ICS!";

(3)

int fd1 = open("ICS.txt", O\_RDWR);

int fd2 = open("ICS.txt", O\_RDWR);

char \*str1 = "I love ";

char \*str2 = "I love ICS!";

(4)

int fd1 = open("ICS.txt", O\_RDWR);

int fd2 = dup(fd1);

char \*str1 = "I love ";

char \*str2 = "I love ICS!";

下面哪一个组合是正确的：（ ）

A．(1)(4) B．(2)(3) C．(1)(2)(3)(4) D．都不正确

17、下列关于计算机网络概念的说法中，哪一项是正确的？答：（ ）

A. 全球最大的计算机网络是互联网Internet，所以计算机网络协议是Internet Protocol即IP协议。

B. 计算机之间的网络通信是一个机器上的一个 process (如client process) 与另一个机器上的 process (如server process) 之间的通信。

C. 网络应用程序有默认的端口号，大部分应用的端口号可以修改，而少部分知名应用如Web服务程序的端口号80是无法修改的。

D. 一个域名只能对应一个IP地址；而一个IP地址可以对应多个域名。

18、在 client-server 模型中，一个连接（connection）可以由IP地址，端口号的组合来表示。假设一个访问网页服务器的应用，客户端IP地址为128.2.194.242，目标服务器端IP地址为208.216.181.15，用户设置的代理服务器IP地址为155.232.108.39。目标服务器同时提供网页服务（默认端口80），和邮件服务（默认端口25）。当客户端向目标服务器发送访问网页的请求时，下面connection socket pairs 正确的一组是？答：（ ）

|  |  |  |
| --- | --- | --- |
|  | 客户端请求 | 代理请求 |
| A | (128.2.194.242:25, 155.232.108.39:80) | (128.2.194.242:51213, 208.216.181.15:80) |
| B | (128.2.194.242:51213, 155.232.108.39:80) | (128.2.194.242:12306, 208.216.181.15:80) |
| C | (128.2.194.242:25, 208.216.181.15:80) | (155.232.108.39:51213, 208.216.181.15:80) |
| D | (128.2.194.242:51213, 208.216.181.15:80) | (155.232.108.39:12306, 208.216.181.15:80) |

19、在Pthread线程包使用中，下列代码输出正确的是：（ ）

void \*th\_f(void \* arg)

{

printf("Hello World") ;

pthread\_exit(0) ;

}

int main(void)

{

pthread\_t tid;

int st;

st = pthread\_create(&tid,NULL,th\_f,NULL);

if(st<0) {

printf("Oops, I can not create thread\n");

exit(-1);

}

sleep(1);

exit(0);

}

A. Oops, I can not create thread

B. Hello World

Oops, I can not create thread

C. Hello World

D. 不输出任何信息

20、有四个信号量，初值分别为：a=1, b=1, c=1, d=1

|  |  |  |  |
| --- | --- | --- | --- |
| 线程1：  ① P(a);  ② P(b);  ③ P(d);  ④ P(c);  ⑤ V(d);  ⑥ V(a);  ⑦ V(b);  ⑧ V(c); | 线程2：  ⑨ P(c);  ⑩ P(b);  ⑪ V(c);  ⑫ V(b);  ⑬ P(a);  ⑭ V(a); | 线程3：  ⑮ P(d);  ⑯ P(a);  ⑰ V(d);  ⑱ P(b);  ⑲ V(a);  ⑳ V(b); |  |

上面的程序执行时，下列哪些执行轨迹不会产生死锁?

A. ①②③⑨⑩⑮④ B. ⑨①②③⑩

C. ①②⑮⑨③ D. ⑨①⑮②③

|  |
| --- |
| 得分 |
|  |

## 第二题（8分）

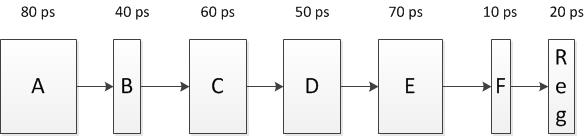
考虑一种新的遵从IEEE规范的浮点的格式，包含3位指数位，4位小数位和1位符号位。请填写下面的表格。

|  |  |  |
| --- | --- | --- |
| **描述** | **十进制数(或分数)** | **二进制表示** |
| Bias |  | -------- |
| 最小的正数 |  |  |
| 最小的有限数 |  |  |
| 最小规格正数 | 1/4 |  |
| --- | -11/32 |  |
| --- | 7/2 |  |
| --- | 63/128 |  |
| --- | 13.25 |  |

|  |
| --- |
| 得分 |
|  |

## **第三题 （12分）**

如图所示，每个模块表示一个单独的组合逻辑单元，每个单元的延迟已在图中标出。通过在两个单元间添加寄存器的方式，可以对该数据通路进行流水化改造。假设每个寄存器的延迟为20ps。



1）如果改造为一个二级流水线（只插入一个寄存器），为获得最大的吞吐率，该寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。

2）如果改造为一个三级流水线（插入两个寄存器），为获得最大的吞吐率，寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。

3）如果改造为一个四级流水线（插入三个寄存器），为获得最大的吞吐率，寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。

4）不改变单元划分，为获得最大性能，该设计至少需要划分成几级？请计算对应的吞吐率，并说明计算过程。

## **第四题（10分）**

考虑如下两个程序（fact1.c和fact2.c）：

/\* fact1.c \*/

#define MAXNUM 12

int table[MAXNUM];

int fact(int n);

int main(int argc, char \*\*argv) {

int n;

table[0] = 0;

table[1] = 1;

if (argc == 1) {

printf("Error: missing argument\n");

exit (0);

}

argv++;

if (sscanf(\*argv, "%d", &n) != 1 || n < 0 || n >= MAXNUM) {

printf ("Error: %s not an int or out of range\n", \*argv);

exit (0);

}

printf("fact(%d) = %d\n", n, fact(n));

}

/\* fact2.c \*/

int\* table;

int fact(int n) {

static int num = 2;

if (n >= num) {

int i = num;

while (i <= n) {

table[i] = table[i-1] \* i;

i++;

}

num = i;

}

return table[n];

}

（1）对于每个程序中的相应符号，给出它的属性（局部变量、强全局变量或弱全局变量），以及它在链接后位于ELF文件中的什么位置？（提示：如果某表项中的内容无法确定，请画X）（6分）

fact1.c

|  |  |  |
| --- | --- | --- |
| 变量 | 类型 | ELF Section |
| table |  |  |
| fact |  |  |
| num |  |  |

fact2.c

|  |  |  |
| --- | --- | --- |
| 变量 | 类型 | ELF Section |
| table |  |  |
| fact |  |  |
| num |  |  |

（2）对上述两个文件进行链接之后，会对每个符号进行解析。请给出链接后下列符号被定义的模块（fact1 or fact2）。（2分）

|  |  |
| --- | --- |
|  | 定义模块 |
| table |  |
| fact |  |
| num |  |

（3）使用gcc（命令：gcc -o fact fact1.c fact2.c）来编译之后得到的可执行文件是否能够正确执行？为什么？（2分）

## 第五题（10分）

**Part I**

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

int main() {

printf("A\n");

if (fork() == 0) {

printf("B\n");

}

else {

printf("C\n");

**A**

}

printf(“D\n");

exit(0);

}

（1）如果程序中的A位置的代码为空，列出所有可能的输出结果：（1分）

（2）如果程序中的A位置的代码为：

waitpid(-1, NULL, 0);

列出所有可能的输出结果：（2分）

（3）如果程序中的A位置的代码为：

printf(“E\n”);

列出所有可能的输出结果：（2分）

**Part II**

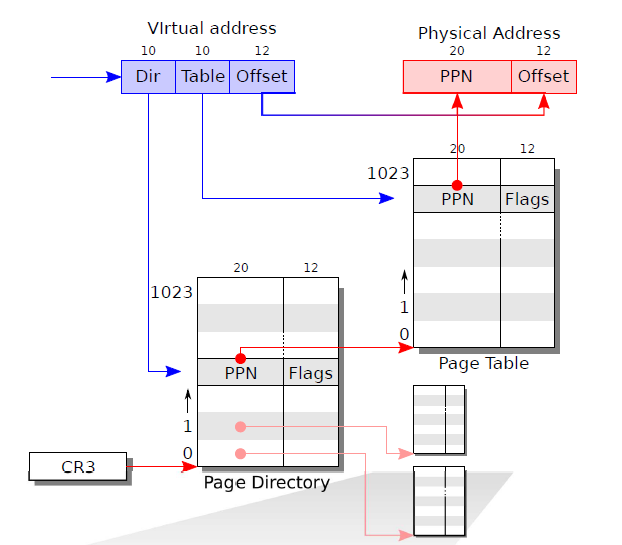
请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行，且每条语句都是原子动作）：

|  |  |
| --- | --- |
| pid\_t pid;  int even = 0;  int counter1 = 0;  int counter2 = 1;  void handler1(int sig) {  if (even % 2 == 0) {  printf(“%d\n”, counter1);  counter1 = A ;  } else {  printf(“%d\n”, counter2);  counter2 = B ;  }  even = C ;  }  void handler2(int sig) {  if (\_\_\_\_\_D\_\_\_\_\_) {  counter1 = even \* even;  } else {  counter2 = even \* even;  }  }  int main() {  signal(SIGUSR1, handler1);  signal(SIGUSR2, handler2);  if ((pid = fork()) == 0) {  while (1) {};  }  while (even < 20) {  kill(pid, E );  sleep(1);  kill(pid, F );  sleep(1);  even += 2;  }  kill(pid, SIGKILL);  exit(0);  } | 1). 完成程序，使得程序在输出前20个斐波那契(Fibonacci)数列，即F0=0, F1=1, …, Fn=Fn-1+Fn-2。（如果存在对本次程序执行结果没有影响的语句，请在相应位置填写“无关”）（3分）  A: \_\_\_ \_\_  B: \_\_\_ \_\_  C: \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_  D: \_\_\_\_ \_\_\_\_\_\_\_\_\_\_  E: \_\_\_ \_\_\_\_\_\_\_\_\_\_\_  F: \_\_\_ \_\_\_\_\_\_\_\_\_\_\_  2). 完成程序，其中A, B处保持不变，使得程序可以分别输出前几个奇数或偶数的平方和。（如果存在对本次程序执行结果没有影响的语句，请在相应位置填写“无关”）（2分）  其中：  若要输出奇数的平方和：even的初始值为**3**；  若要输出偶数的平方和，even的初始值为**2**。  C: \_\_\_\_ \_\_\_\_\_\_\_\_\_\_  D: \_\_\_\_ \_\_\_\_\_\_  E: \_\_\_\_ \_\_\_\_\_\_\_\_\_\_  F: \_\_\_\_ \_\_\_\_\_\_\_\_\_\_ |

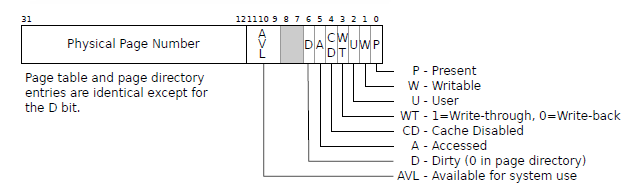
|  |
| --- |
| 得分 |
|  |

## 第六题（10分）

Intel的IA32 体系结构采用二级页表，称第一级页表为页目录(Page Directory)，第二级页表为页表(Page Table)。其虚拟地址到物理地址的翻译方式如下图。先根据CR3找到页目录地址，然后依据偏移Dir找到一个页目录项，页目录项的高20位(PPN)为二级页表地址；在二级页表中根据偏移Table找到页表项，页表项中的高20位(PPN)即为物理地址的高20位，将这20位与虚拟地址的低12位拼在一起形成完整的物理地址。



页目录和页表均有1024项，每一项为4字节，含义如下：



页目录和页表由操作系统维护，通常情况下只能在内核态下访问，为了给用户提供一个访问页表项和页目录项内容的接口，假设操作系统中已经执行过如下代码段：

#define UVPT 0xef400000

#define PDX(la) ((((unsigned int) (la)) >> 22) & 0x3FF)

……

kern\_pgdir[PDX(UVPT)] = PADDR(kern\_pgdir) | PTE\_U | PTE\_P;

其中kern\_pgdir是操统系统维护的页目录数组，共1024项，每一项的类型为unsigned int。PADDR(kern\_pgdir)用于获得kern\_pgdir的物理地址，页目录在物理内存中正好占一页，所以kern\_pgdir的物理地址是4KB对齐的。PTE\_U和PTE\_P代表了这个页目录项的权限，即用户态可访问(只读)。可以看到，这条语句将页目录的第PDX(UVPT)项指向了页目录自身。

利用这一点，对于给定的虚拟地址va，可以获得va对应的页目录项和页表项内容，分别对应于函数get\_pde和get\_pte，请完成这两个函数（每空2分）：

#define UVPT 0xef400000

// get\_pde(va)：获取虚拟地址va对应的一级页表(页目录)中的页目录项内容

unsigned int get\_pde(unsigned int va) {

unsigned int pdx = (va >> \_[1]\_\_\_ \_\_\_\_ ) & \_[2]\_ \_;

unsigned int addr = UVPT + (\_[3]\_ \_\_\_) + pdx \* 4;

return \*((unsigned int \*)(addr));

}

// get\_pte(va)：获取虚拟地址va对应的二级页表中的页表项内容

unsigned int get\_pte(unsigned int va) {

unsigned int PGNUM = va >> \_\_\_[4]\_ \_\_;

unsigned int addr = \_\_\_[5]\_ \_ + PGNUM;

return \*((unsigned int \*)(addr));

}

|  |
| --- |
| 得分 |
|  |

## 第七题（10分）

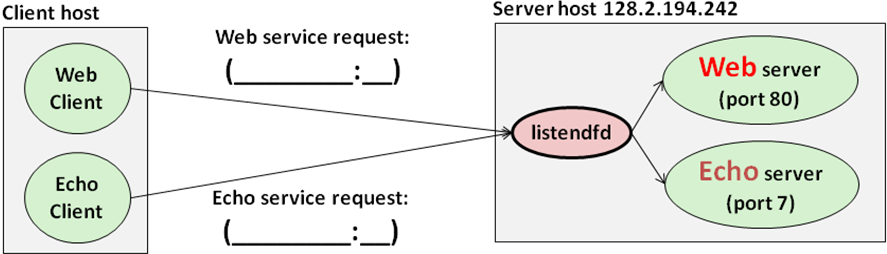
（1） 一个服务器拥有两个独立的固定IP地址，那么它在web应用端口80上最多可以监听多少个独立的 socket 连接？（2分）

|  |  |  |
| --- | --- | --- |
| **服务器端** | **客户端** | **结果** |
|  |  |  |

（2） 该服务器在所有有web应用端口上最多可以监听多少个独立的 socket 连接？（2分）

|  |  |  |
| --- | --- | --- |
| **服务器端** | **客户端** | **结果** |
|  |  |  |

（3） 在下图中连线上填入正确的目标服务器的socket 标识符（2分）



4) 在Echo server 范例中，server端通过accept函数接受了一个 client的连接请求，从而将网络描述符与该网络连接、socket绑定，然后进行网络数据传输。在下面的空格处填写正确的网络描述符，每个空填写**listenfd** 或**connfd**（共4分，每空1分）。

int main(int argc, char \*\*argv) {

int **listenfd**, **connfd**, port, clientlen;

struct sockaddr\_in clientaddr;

struct hostent \*hp;

char \*haddrp;

unsigned short client\_port;

…………

while (1) {

clientlen = sizeof(clientaddr);

\_\_\_\_\_\_\_\_\_ = Accept(\_\_\_\_\_\_\_\_\_, (SA \*)&clientaddr,

&clientlen);

hp = Gethostbyaddr((const char\*)

&clientaddr.sin\_addr.s\_addr,

sizeof(clientaddr.sin\_addr.s\_addr), AF\_INET);

haddrp = inet\_ntoa(clientaddr.sin\_addr);

client\_port = ntohs(clientaddr.sin\_port);

printf("server connected to %s (%s), port %u\n",

hp->h\_name, haddrp, client\_port);

echo(\_\_\_\_\_\_\_\_\_);

Close(\_\_\_\_\_\_\_\_\_);

}

}

|  |
| --- |
| 得分 |
|  |

## 第八题（10分）

某个城市为了解决市中心交通拥堵的问题，决定出台一项交通管制措施，对进入市中心区的机动车辆实行单双日限制行驶的办法。具体要求是，逢单日，只允许车辆牌号号码为单数的机动车进入市中心区；同样，逢双日，只允许车辆牌号号码为双数的机动车进入市内中心区。有一个进入市中心区的交通路口，进入该路口的道路有一条，离开该路口的道路有两条，其中一条是通往市中心区的道路，而另一条是绕过市中心区的环路，在进入路口处设置了自动识别车辆牌号的识别设备与放行栅栏控制设备。在单日，遇到单号车辆进入路口车辆号码识别区，号码识别设备打开通往市中心区道路的放行栅栏；而遇到双号车辆，则打开绕过市中心区环路的放行栅栏。反之亦然。显然，只有在该路口车辆号码识别区中无车时，才允许一辆车进入车辆号码识别区。同时为了防止有车辆混过路口，两个放行栅栏平时处于关闭状态，只有在车辆号码识别区中的车辆已被识别出单双号之后，放行栅栏才会在识别设备的控制下，打开对应的放行栅栏，在车辆通过之后，该放行栅栏自行关闭。

vehicle\_n int； /\* 车辆号码 \*/

检查车辆牌号线程T1：

while (1) {

车辆到达识别区路口；  
①

车辆进入号码识别区；

if (vehicle\_n == 奇数)

{ ② }  
 else

{ ③ }  
 }；

市区放行栅栏线程T2：

while (1) {

④

允许车辆进入市中心区；

⑤  
 }；

环路放行栅栏线程T3：

while (1) {

⑥

允许车辆绕行环路；

⑦  
}；

1）请设计若干信号量，给出每一个信号量的作用和初值。（3分）

2）请将信号量上对应的PV操作填写在代码中适当位置。（7分）