ICS Seminar Week10 Prep

王善上 倪嘉怡 许珈铭

2023.11.27

Rules

```
remainder <- ordinal number in WeChat Group % 4
for all questions do
    if question number % 4 == remainder then
        you should work on it
    end
end
```

signal

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

C

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

D

6. 一段程序中阻塞了 SIGCHLD 和 SIGUSR1 信号。接下来，向它按顺序发送 SIGCHLD，SIGUSR1，SIGCHLD 信号，当程序取消阻塞继续执行时，将处理这三个信号中的哪几个？

A. 都不处理

B. 处理一次 SIGCHLD

C. 处理一次 SIGCHLD，一次 SIGUSR1

D. 处理所有三个信号

C

8. 下列说法正确的是：

A. SIGTSTP 信号既不能被捕获，也不能被忽略

B. 存在信号的默认处理行为是进程停止直到被 SIGCONT 信号重启

C. 系统调用不能被中断，因为那是操作系统的工作

D. 子进程能给父进程发送信号，但不能发送给兄弟进程

B

10．下列哪一事件不会导致信号被发送到进程？

A．新连接到达监听端口

B．进程访问非法地址

C．除零

D．上述情况都不对

A

10．下列关于信号的说法不正确的是：

A．在键盘上输入 Ctrl-C 会导致内核发送一个 SIGINT 信号到前台进程组中的每个进程

B．每种类型最多只能有一个未处理的信号

C．SIGINT 的处理函数不能被另一个 SIGINT 信号中断

D．进程可以通过使用 signal 函数修改和 SIGSTOP 相关联的默认行为

D

5. 关于进程，以下说法正确的是：

A. 没有设置模式位时，进程运行在用户模式中，允许执行特权指令，例如发起 I/O 操作。

B. 调用 waitpid(-1, NULL, WNOHANG & WUNTRACED)会立即返回：如果调用进程的所有子进程都没有被停止或终止，则返回 0；如果有停止或终止的子进程，则返回其中一个的 ID。

C. execve 函数的第三个参数 envp 指向一个以 null 结尾的指针数组，其中每一个指针指向一个形如”name=value”的环境变量字符串。

D. 进程可以通过使用 signal 函数修改和信号相关联的默认行为，唯一的例外是 SIGKILL，它的默认行为是不能修改的。

C

9. 进程管理相关函数的调用和返回行为，下列那些函数都是可能返回多于一次的？

A. longjmp 和 fork

B. execve 和 longjmp

C. fork 和 setjmp

D. setjmp 和 execve

C

11. 在键盘上输入 Ctrl+C 会导致内核发送一个（ ）信号到（ ）进程组中的每个进程

A. SIGINT，前台

B. SIGTSTP，前台

C. SIGINT，后台

D. SIGTSTP，后台

A

13. 对于 Linux 系统，下列说法错误的是:

A. 在单核 CPU 上，所有看似并行的逻辑流实际上是并发的。

B. 用户模式不可访问/proc 文件系统中包含的内核数据结构的内容。

C.在 execve 函数参数的 argv 数组加入”> 1.txt”，不能自动实现 IO 重定向。

D. 即便在信号处理程序中调用 printf 前阻塞所有信号，也不一定安全。

B

1、关于进程和异常控制流，以下说法正确的是：

A、调用 waitpid (-1, NULL, WNOHANG & WUNTRACED) 会立即返回：如果调用进程的所有子进程都没有被停止或终止，则返回 0；如果有停止或终止的子进程，则返回其中一个的 ID。

B、进程可以通过使用 signal 函数修改和信号相关联的默认行为，唯一的例外是 SIGKILL，它的默认行为是不能修改的。

C、从内核态转换到用户态有多种方法，例如设置程序状态字；从用户态转换到内核态的唯一途径是通过中断/异常/陷入机制。

D、中断一定是异步发生的，陷阱可能是同步发生的，也可能是异步发生的。

C

1. 下列关于系统 I/O 的说法中，正确的是（）：

A. Linux shell 创建的每个进程开始时都有三个打开的文件：标准输入（描述符为 0），标准输出（描述符为 1），标准错误（描述符为 2），这使得程序始终不能使用保留的描述符 0,1,2 读写其他文件。

B. Unix I/O 的 read/write 函数是异步信号安全的，故可以在信号处理函数中使用。

C. RIO 函数包的健壮性保证了对于同一个文件描述符，任意顺序调用 RIO 包中的任意函数不会造成问题。

D. 使用 int fd1 = open("ICS.txt", O_RDWR); 打开 ICS.txt 文件后，再用 int fd2 = open("ICS.txt", O_RDWR); 再次打开文件，会使得 fd1 对应的打开文件表中的引用计数 refcnt 加一。

B

io

19.下列系统 I/O 的说法中，正确的是(　　)

A. C 语言中的标准 I/O 函数在不同操作系统中的实现代码一样

B. 对于同一个文件描述符，混用 RIO 包中的 rio_readnb 和 rio_readn 两个函数不会造成问题

C. C 语言中的标准 I/O 函数是异步线程安全的

D. 使用 I/O 缓冲区可以减少系统调用的次数，从而加快 I/O 的速度

D

15. 关于 IO 操作，以下说法中正确的是（）

A. 由于 RIO 包的健壮性，所以 RIO 中的函数都可以交叉调用

B. 成功调用 open 函数后，返回一个不小于 3 的文件描述符

C. 调用 Unix I/O 开销较大，标准 I/O 库使用缓冲区来加快 I/O 的速度

D. 和描述符表一样，每个进程拥有独立的打开文件表

C

14. 以下关于文件 I/O 的说法中，正确的是：

A. 文件重定向（dup 和 dup2）操作仅仅改变了文件描述符的指向，不会改变打开文件表中的内容

B. 进程调用 fork()时，可能对文件描述符表和打开文件表采用写时拷贝（Copy on Write）机制

C. 对同一描述符，rio_readlineb 和 rio_readnb 可以任意交叉使用，rio_readn 和 rio_writen 也可以任意交叉使用

D. RIO 中包括无缓冲的输入输出函数和带缓冲的输入输出函数，使用带缓冲的输入输出函数时，要先声明一个 rio_t 类型变量并调用 rio_readinitb 函数

C

6. 假设某进程有且仅有五个已打开的文件描述符：0~4，分别引用了五个不同的文件，尝试运行以下代码：

dup2(3,2)； dup2(0,3)； dup2(1,10)； dup2(10,4)； dup2(4,0)；

关于得到的结果，说法正确的是：

A. 运行正常完成，现在有四个描述符引用同一个文件

B. 运行正常完成，现在进程共引用四个不同的文件

C. 由于试图从一个未打开的描述符进行复制，发生错误

D. 由于试图向一个未打开的描述符进行复制，发生错误

A

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

A

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

B

13.下列这段代码的输出不可能是(　　)

```
void handler()
{
    printf("h");
}

int main()
{
    signal(SIGCHLD, handler) ;

    if ( fork() == 0 ) {
        printf("a") ;
    } else {
        printf("b") ;
    }
    printf("c") ;
    exit(0) ;
}
```

A. abcc　　B. abch　　C. bcach　　D. bchac

D

18.ICS.txt 中包含 3000 个字符，考虑如下代码段：

```
int main(int argc, char** argv) {
   int fd = open("ICS.txt", O_CREAT | O_RDWR, S_IRUSR |
S_IWUSR);
   write(fd, "ICS", 3);

   char buf[128];
   int i;
   for (i = 0; i < 10; i++) {
      int fd1 = open("ICS.txt", O_RDWR);
      int fd2 = dup(fd1);

      int cnt = read(fd1, buf, 128);
      write(fd2, buf, cnt);
   }
   return 0;
}
```

上述代码执行完后，ICS.txt 中包含多少个字符？(　　)（假设所有系统调用都成功）

A. 3　　B. 256　　C. 3000　　D. 3072

C

14. 考虑以下代码，假设 result.txt 中的初始内容为“666666”

```
char *str1 = "6666";
char *str2 = "2333";
char *str3 = "hhhh";
int fd1, fd2, fd3, i;

fd1 = open("result.txt", O_RDWR);
fd2 = open("result.txt", O_RDWR);
dup2(fd1, fd2);

for (i = 0; i < 5; ++i) {
    fd3 = open("result.txt", O_RDWR);
    write(fd1, str1, 4);
    write(fd2, str2, 4);
    write(fd3, str3, 4);
    close(fd3);
}

close(fd1); close(fd2);
```

假设所有系统调用均成功，则这段代码执行结束后，result.txt 的内容中有（）个“6”

A. 6

B. 16

C. 20

D. 22

B

15. 以下程序执行完成后，ICS.txt 文件中的内容是：

```
int main(int argc, char** argv) {
    int fd1 = open("ICS.txt", O_CREAT|O_RDWR,
                               S_IRUSR|S_IWUSR);
    write(fd1, "ics ", 4);
    int fd2 = fd1;
    int fd3 = dup(fd2);
    int fd4 = open("ICS.txt", O_APPEND|O_RDWR);
    write(fd2, "segmentation fault ", 19);
    write(fd4, "tao", 3);
    int fd5 = fd4;
    dup2(fd3, fd5);
    write(fd4, "lab", 3);
    close(fd1);
    return 0;
}
```

A. ics segmentation fault tao

B. ics segmentation fault lab

C. ics taomentation fault lab

D. tao segmentation fault lab

B

2. 考虑以下代码，假设 ICS.txt 中的初始内容为"ICS!!!ics!!!"：

```
int fd = open("ICS.txt", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
for (int i = 0; i < 2; ++i){
    int fd1 = open("ICS.txt", O_RDWR | O_APPEND);
    int fd2 = open("ICS.txt", O_RDWR);
    write(fd2, "!!!!!!", 6);
    write(fd1, "ICS", 3);
    write(fd, "ics", 3);
}
```

假设所有系统调用均成功，则这段代码执行结束后，ICS.txt 的内容为（）：

A．ICSics

B．!!!icsICS

C．!!!icsics!!!ICSICS

D．!!!icsICSICS

D

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 = 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 < 30) {
            kill(pid, ___E___);
            sleep(1);
            kill(pid, ___F___);
            sleep(1);
            even = even+___G____ ;
        }
        kill(pid, SIGKILL);
        exit(0);
    }
```

（1）完成程序，使得程序在输出的数字为以下 Q 队列的前 30 项，Q 队列定义如下：

$$Q_0 = 0,\ Q_1 = 1,\ Q_{n+2} = \begin{cases} Q_n + 1, & \text{n\%2} = 0 \\ Q_n \times 2, & \text{n\%2} \neq 0 \end{cases} \quad (n = 0,1,2,3,\ldots)$$

（注：若某个位置中的程序内容，对本次程序执行结果没有影响，请在相应位置填写“无关”）

A: _____

B: _____

C: _____

D: _____

E: _____

F: _____

G: _____

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 = 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 < 30) {
            kill(pid, ___E___);
            sleep(1);
            kill(pid, ___F___);
            sleep(1);
            even = even+___G___ ;
        }
        kill(pid, SIGKILL);
        exit(0);
}
```

（1）完成程序，使得程序在输出的数字为以下 Q 队列的前 30 项，Q 队列定义如下：

$$Q_0 = 0,\ Q_1 = 1,\ Q_{n+2} = \begin{cases} Q_n + 1, & \text{n\%2} = 0 \\ Q_n \times 2, & \text{n\%2} \neq 0 \end{cases} \quad (n = 0,1,2,3,\ldots)$$

（注：若某个位置中的程序内容，对本次程序执行结果没有影响，请在相应位置填写“无关”）

A: _____

B: _____

C: _____

D: _____

E: _____

F: _____

G: _____

counter1 + 1
counter2 * 2
1
无关
SIGUSR1
SIGUSR1
2

第四题（10 分）

分析以下C程序，其中f1.txt和f2.txt为已有用户有读写的文件，初始文件内容为空。

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <sys/types.h>
4.  #include <sys/stat.h>
5.  #include <fcntl.h>
6.  #include <unistd.h>
7.  #include <sys/types.h>
8.  #include <sys/wait.h>
9.
10. int main()
11. {
12.   int fd1,fd2,fd3,fd4;
13.   int pid;
14.   int c=1;
15.   fd1=open("./f1.txt",O_WRONLY,0);
16.   fd2=open("./f1.txt",O_WRONLY,0);
17.
18.   printf("fd1=%d,fd2=%d;\n",fd1,fd2);
19.
20.   write(fd1,"EECSPKU",7);
21.   write(fd2,"2019",4);
22.
23.   close(fd2);
24.
25.   fd3=open("./f2.txt",O_WRONLY,0);
26.   fd4=dup(fd3);
27.
28.   printf("fd3=%d,fd4=%d;\n",fd3,fd4);
29.
30.   pid=fork();
31.   if((pid==0)) {
32.     c--;
33.     write(fd3,"PKU",3);
34.     write(fd4,"ICS",3);
35.     printf("c= %d\n",c);
36.   }
37.   else {
38.     waitpid(-1,NULL,0);
39.     c++;
40.     write(fd3,"2019",4);
41.     close(fd3);
42.     close(fd4);
43.     printf("c= %d\n",c);
44.   }
45.
46.   if(c)
47.    write(fd1,"CS",2);
48.   c++;
49.   close(fd1);
50.   printf("c=%d\n",c);
51. }
```

当程序正确运行后，填写输出结果：

（1） 程序第 18 行：fd1=__①__，fd2=__②__；

（2） 程序第 28 行：fd3=__①__，fd4=__②__；

（3） 程序第35、43、50 行输出c的值依次分别为：_____；

（4） 文件 f1.txt 中的内容为：_____；

（5） 文件 f2.txt 中的内容为：_____。

第四题（10 分）

分析以下C程序，其中f1.txt和f2.txt为已有用户有读写的文件，初始文件内容为空。

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <sys/types.h>
4.  #include <sys/stat.h>
5.  #include <fcntl.h>
6.  #include <unistd.h>
7.  #include <sys/types.h>
8.  #include <sys/wait.h>
9.
10. int main()
11. {
12.   int fd1,fd2,fd3,fd4;
13.   int pid;
14.   int c=1;
15.   fd1=open("./f1.txt",O_WRONLY,0);
16.   fd2=open("./f1.txt",O_WRONLY,0);
17.
18.   printf("fd1=%d,fd2=%d;\n",fd1,fd2);
19.
20.   write(fd1,"EECSPKU",7);
21.   write(fd2,"2019",4);
22.
23.   close(fd2);
24.
25.   fd3=open("./f2.txt",O_WRONLY,0);
26.   fd4=dup(fd3);
27.
28.   printf("fd3=%d,fd4=%d;\n",fd3,fd4);
29.
30.   pid=fork();
31.   if((pid==0)) {
32.     c--;
33.     write(fd3,"PKU",3);
34.     write(fd4,"ICS",3);
35.     printf("c= %d\n",c);
36.   }
37.   else {
38.     waitpid(-1,NULL,0);
39.     c++;
40.     write(fd3,"2019",4);
41.     close(fd3);
42.     close(fd4);
43.     printf("c= %d\n",c);
44.   }
45.
46.   if(c)
47.    write(fd1,"CS",2);
48.   c++;
49.   close(fd1);
50.   printf("c=%d\n",c);
51. }
```

当程序正确运行后，填写输出结果：

（1） 程序第 18 行：fd1=__①__，fd2=__②__；

（2） 程序第 28 行：fd3=__①__，fd4=__②__；

（3） 程序第35、43、50 行输出c的值依次分别为：_____；

（4） 文件 f1.txt 中的内容为：_____；

（5） 文件 f2.txt 中的内容为：_____。

3, 4
4, 5
0, 1, 2, 3
2019PKUCS
PKUICS2019

5. 下面对指令系统的描述中，错误的是：（ ）

A. 通常 CISC 指令集中的指令数目较多，有些指令的执行周期很长；而 RISC 指令集中指令数目较少，指令的执行周期较短。

B. 通常 CISC 指令集中的指令长度不固定；RISC 指令集中的指令长度固定。

C. 通常 CISC 指令集支持多种寻址方式，RISC 指令集支持的寻址方式较少。

D. 通常 CISC 指令集处理器的寄存器数目较多，RISC 指令集处理器的寄存器数目较少。

D

6. Y86 指令 rmmovl 的 SEQ 实现如下图所示，其中①和②分别为：

	rmmovl rA, D(rB)
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_4[PC+2] valP ← ①
Decode	valA ← R[rA] valB ← R[rB]
Execute	valE ← ②
Memory	M_4[valE] ← valA
Write back	
PC update	PC ← valP

A．PC + 4，valB + 4　　B．PC + 4，valB + valC

C．PC + 6，valB + 4　　D．PC + 6，valB + valC

D

2. 在本课程的 PIPE 流水线中，下列情况会出现数据冒险的是：

A．当前指令会改变下一条指令的目的操作数

B．当前指令会改变下一条指令的源操作数

C．下一条指令会改变当前指令的目的操作数

D．下一条指令会改变当前指令的源操作数

B

4. 下述关于 RISC 和 CISC 的讨论，哪个是错误的

A．RISC 指令集包含的指令数量通常比 CISC 的少

B．RISC 的寻址方式通常比 CISC 的寻址方式少

C．RISC 的指令长度通常短于 CISC 的指令长度

D．手机处理器通常采用 RISC，而 PC 采用 CISC

C

2. 有如下结构定义和程序片段

```
struct A
{
    char c;
    int i;
    double d;
    int array[10];
};

struct B
{
    int array[10];
    double d;
    char c;
    int i;
};

void foo(struct A *pa, struct B *pb, int index)
{
    pb->i = pa->array[index];
}
```

在 Linux 下使用 GCC 编译器，仅采用-O2 选项，上述代码对应的汇编语言是：（将选项依次填入空格内）

```
movslq %edx, %rdx
movl __(%rdi,%rdx,__), %eax
movl %eax, __(%rsi)
```

A. (16, 4, 52)　B. (24, 4, 52)　C. (16, 4, 49)　D. (24, 4, 49)

A

3. 下面说法正确的是：

A. 不同指令的机器码长度是相同的

B. `test %rax, %rax` 恒等于 `cmp $0, %rax`

C. switch 编译后总是会产生跳转表

D. 以上都不对

4. 分析下图的指令执行步骤，请问这是 Y86 指令系统的哪条指令？

Fetch	icode:ifun ← M_1[PC]
Decode	valA ← R[%esp] valB ← R[%esp]
Execute	valE ← valB + 4
Memory	valM ← M_4[valA]
Write back	R[%esp] ← valE
PC update	PC ← valM

A. call

B. ret

C. pushl

D. popl

3． 缓冲区溢出会带来程序风险，下列避免方法中错误的是：

A．在栈中存放特殊字段用于检测是否发生缓冲区溢出

B．避免使用有风险的库函数，如 gets 等

C．随机设置栈的偏移地址

D．分配尽可能大的缓冲区数组

D

4．现有四级指令流水线，分别完成取指、取数、运算、传送结果 4 步操作。若完成上述操作的时间依次为 9ns、10ns、6ns、8ns，则流水线的操作周期应设计为 _____ ns。

A. 6　　B. 8　　C. 9　　D. 10

D

3. 下面哪条指令不是 X86 正确的寻址方式

A. movl $34, (%eax)

B. movl (%eax), %eax

C. movl $23, 10(%edx, %eax)

D. movl (%eax), 8(%ebx)

D

3. 左边的 C 函数中，在 x86_64 服务器上采用 GCC 编译产生的汇编语言如右边所示。那么（1）和（2）的内容分别是：（　　）

```
int arith(int x, int y) {
   return (x < y) ? ( 1 ) : ( 2 );
}
```

```
<arith>:
  lea    (%rsi,%rdi,1),%eax
  mov    %esi,%edx
  sub    %edi,%edx
  cmp    %esi,%edi
  cmovge %edx,%eax
  retq
```

（提示：第一个参数放在 rdi 寄存器中，第二个参数放在 rsi 寄存器中）

A. x-y，x+y　B. x+y，x-y　C. x+y，y-x　D. y-x，x+y

C

4. 假定 struct P {int i; char c; int j; char d;}; 在 x86_64 服务器的 Linux 操作系统上，下面哪个结构体的大小与其它三个不同：答：（　　）

A. struct P1 {struct P a[3]};
B. struct P2 {int i[3]; char c[3]; int j[3]; char d[3]};
C. struct P3 {struct P *a[3]; char *c[3];};
D. struct P4 {struct P *a[3]; int *f[3];};

B

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

D

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

A