ICS Seminar Week9 Prep

刘昕垚 杨斯淇 许珈铭

2023.11.20

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
```

Q1

4. 下列关于链接技术的描述，错误的是（ ）

A. 在 Linux 系统中，对程序中全局符号的不恰当定义，会在链接时刻进行报告。

B. 在使用 Linux 的默认链接器时，如果有多个弱符号同名，那么会从这些弱符号中任意选择一个占用空间最大的符号。

C. 编译时打桩（interpositioning）需要能够访问程序的源代码，链接时打桩需要能够访问程序的可重定位对象文件，运行时打桩只需要能够访问可执行目标文件。

D. 链接器的两个主要任务是符号解析和重定位。符号解析将目标文件中的全局符号都绑定到唯一的定义，重定位确定每个符号的最终内存地址，并修改对那些目标的引用。

答案：选 A。参考机械工业出版社第三版中文教材，四个选项分别对应 P464 第 5 自然段、P471 页文末、P494 页文末、P496 文末的文字。其中选项 B 略有调整，增加了“占用空间最大的”，是 Linux binutils 中链接器实现的原则，仍是正确选项。A 错误是因为全局符号的不恰当定义并不总会报警（甚至不报告 warning）。

A

Q2

4. 下列关于链接技术的描述，错误的是（ ）

A. 在 Linux 系统中，对程序中全局符号的不恰当定义，会在链接时刻进行报告。

B. 在使用 Linux 的默认链接器时，如果有多个弱符号同名，那么会从这些弱符号中任意选择一个占用空间最大的符号。

C. 编译时打桩（interpositioning）需要能够访问程序的源代码，链接时打桩需要能够访问程序的可重定位对象文件，运行时打桩只需要能够访问可执行目标文件。

D. 链接器的两个主要任务是符号解析和重定位。符号解析将目标文件中的全局符号都绑定到唯一的定义，重定位确定每个符号的最终内存地址，并修改对那些目标的引用。

答案：选 A。参考机械工业出版社第三版中文教材，四个选项分别对应 P464 第 5 自然段、P471 页文末、P494 页文末、P496 文末的文字。其中选项 B 略有调整，增加了“占用空间最大的”，是 Linux binutils 中链接器实现的原则，仍是正确选项。A 错误是因为全局符号的不恰当定义并不总会报警（甚至不报告 warning）。

A

Q3

4. 下列关于链接技术的描述，错误的是（ ）

A. 在 Linux 系统中，对程序中全局符号的不恰当定义，会在链接时刻进行报告。

B. 在使用 Linux 的默认链接器时，如果有多个弱符号同名，那么会从这些弱符号中任意选择一个占用空间最大的符号。

C. 编译时打桩（interpositioning）需要能够访问程序的源代码，链接时打桩需要能够访问程序的可重定位对象文件，运行时打桩只需要能够访问可执行目标文件。

D. 链接器的两个主要任务是符号解析和重定位。符号解析将目标文件中的全局符号都绑定到唯一的定义，重定位确定每个符号的最终内存地址，并修改对那些目标的引用。

答案：选 A。参考机械工业出版社第三版中文教材，四个选项分别对应 P464 第 5 自然段、P471 页文末、P494 页文末、P496 文末的文字。其中选项 B 略有调整，增加了“占用空间最大的”，是 Linux binutils 中链接器实现的原则，仍是正确选项。A 错误是因为全局符号的不恰当定义并不总会报警（甚至不报告 warning）。

A

Q4

4. 下列关于链接技术的描述，错误的是（ ）

A. 在 Linux 系统中，对程序中全局符号的不恰当定义，会在链接时刻进行报告。

B. 在使用 Linux 的默认链接器时，如果有多个弱符号同名，那么会从这些弱符号中任意选择一个占用空间最大的符号。

C. 编译时打桩（interpositioning）需要能够访问程序的源代码，链接时打桩需要能够访问程序的可重定位对象文件，运行时打桩只需要能够访问可执行目标文件。

D. 链接器的两个主要任务是符号解析和重定位。符号解析将目标文件中的全局符号都绑定到唯一的定义，重定位确定每个符号的最终内存地址，并修改对那些目标的引用。

答案：选 A。参考机械工业出版社第三版中文教材，四个选项分别对应 P464 第 5 自然段、P471 页文末、P494 页文末、P496 文末的文字。其中选项 B 略有调整，增加了“占用空间最大的”，是 Linux binutils 中链接器实现的原则，仍是正确选项。A 错误是因为全局符号的不恰当定义并不总会报警（甚至不报告 warning）。

A

Q5

12.下列说法中哪一个是错误的？（　　）

A. 中断一定是异步发生的

B. 异常处理程序一定运行在内核模式下

C. 故障处理一定返回到当前指令

D. 陷阱一定是同步发生的

答案：C

A.D. 正确 （书p504）

B. 正确（书p503）

C. 如果可以修正，则返回当前指令；否则回到内核中的abort例程，终止引起故障的程序

C

Q6

7. 学完本课程后，几位同学聚在一起讨论有关异常的话题，请问你认为他们中谁学习的结果有错误？

A. 发生异常和异常处理意味着控制流的突变。

B. 与异常相关的处理是由硬件和操作系统共同完成的。

C. 异常是由于计算机系统发生了不可恢复的错误导致的。

D. 异常的发生可能是异步的，也可能是同步的。

答案：C。
只有会导致终止的异常才是不可恢复的错误。

C

Q7

7. 关于 x86-64 系统中的异常，下面那个判断是正确的：
   A. 除法错误是异步异常，Unix 会终止程序；
   B. 键盘输入中断是异步异常，异常服务后会返回当前指令执行；
   C. 缺页是同步异常，异常服务后会返回当前指令执行；
   D. 时间片到时中断是同步异常，异常服务后会返回下一条指令执行；

答案：C
A. 除法错误是故障，同步
B. 键盘输入中断是I/O 中断，异步，总是返回到下一条指令
C. 缺页异常需要重新执行遇到问题的访存指令，为同步异常
D. 时间片到时中断属于时钟中断，属于异步异常

C

Q8

10. 当一个网络数据包到达一台主机时，会触发以下哪种异常：

A. 系统调用

B. 信号

C. 中断

D. 缺页异常

答案：C

外部 I/O 会触发中断，CPU 执行完当前指令之后可能会去处理该中断

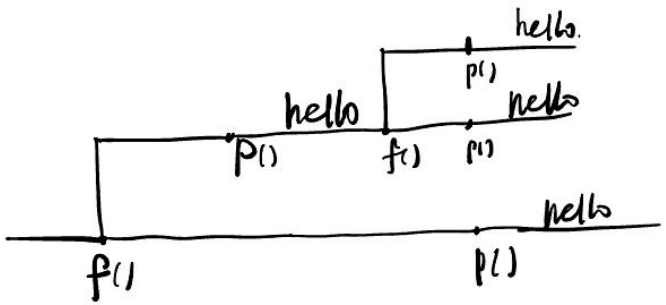
C

Q9

11.在系统调用成功的情况下，下列代码会输出几个 hello? （　　）

```
void doit()
{
    if ( fork() == 0 ) {
       printf("hello\n");
       fork();
    }
    return ;
}

int main()
{
    doit();
    printf("hello\n");
    exit(0) ;
}
```



B

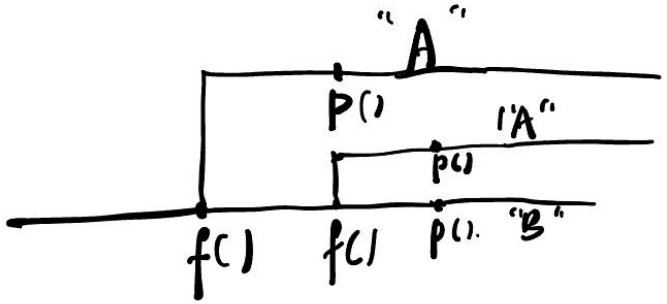
A. 3　　B. 4　　C. 5　　D. 6

Q10

9. 在系统调用成功的情况下，下面哪个输出是可能的？

```
int main() {
    int pid = fork();
    if (pid == 0) {
        printf("A");
    } else {
        pid = fork();
        if (pid == 0) {
            printf("A");
        } else {
            printf("B");
        }
    }
    exit(0);
}
```

A. AAB
B. AAA
C. AABB
D. AA



AAB

ABA

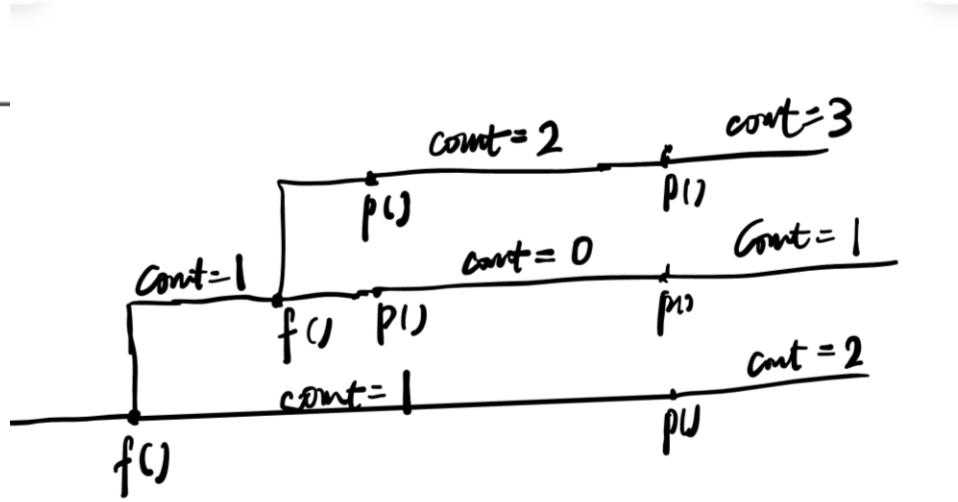
BAA.

A

Q11

9. 下列程序输出的数字顺序可能是：

```
int count = 1;
if (fork() == 0) {
    if (fork() == 0) {
        printf("%d\n", ++count);
    }
    else {
        printf("%d\n", --count);
    }
}
printf("%d\n", ++count);
```



C

至少一个2在3前面，且0在1前面

A. 0 1 3 2 2　　B. 0 3 2 2 1

C. 2 0 1 3 2　　D. 2 1 0 2 3

C

Q12

9. C 语言中的代码如下：

```
fork() && fork();
printf("-");
fork() || fork();
printf("-");
```

这段代码一共输出（ ）个“-”字符。

A. 12

B. 18

C. 20

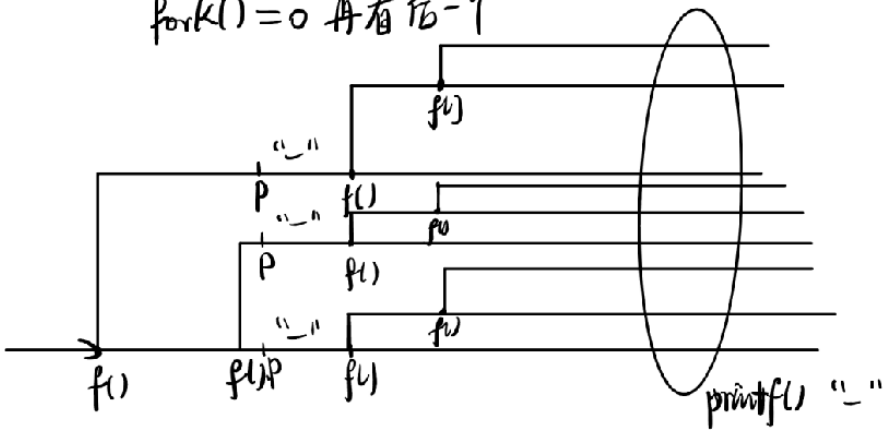
D. 32

&&：fork()=0 一定是0，下一条语句

fork()≠0 不确定，再看后一个操作数

||：fork()≠0 一定是1

fork()=0 再看后一个



- 但这里没有换行符没有fflush函数，到进程结束时才刷新缓冲区；
- 已知创建子进程时会拷贝缓冲区，最后每一个子进程和父进程的缓冲区中都有两个“-”
- 一共九个进程，18个

B

Q13

第四題（10 分）

在 x86_64 **環境下，考慮如下** 2 **个文件：**main.c **和** foo.c **：**

```
/* main.c */
#include <stdio.h>

long long ____________________________;
const char* foo(int);

int main(int argc, char **argv){
    int n = 0;
    sscanf(argv[1], "%d", &n);
    printf(foo(n));
    printf("%llx\n", a);
}
```

```
/* foo.c */
#include <stdio.h>

int a[2];

static void swapper(int num){
    int swapper;
    if (num % 2){
        swapper = a[0];
        a[0] = a[1];
        a[1] = swapper;
    }
}

const char* foo(int num){
    static char out_buf[50];
    swapper(num);
    sprintf(out_buf,                                    "%x\n",
____________________________);
    return out_buf;
}
```

Q13

1. 对于每个程序中的相应符号，给出它的属性（局部或全局，强符号或弱符号）（提示：如果某表项中的内容无法确定，请画 X。）

main.c

	局部或全局？	强或弱？
a		
foo		

foo.c

	局部或全局？	强或弱？
a		
foo		
out_buf		

2. 根据如下的程序运行结果，补全程序【在程序空白处填空即可】。

```
$ gcc -o test main.c foo.c
$ ./test 1
bffedead
cafebffedeadbeef

$ ./test 2
beefcafe
deadbeefcafebffe
```

3. 现在有一位程序员要为这个程序编写头文件。假设新的头文件名称为 foo.h，内容如下：

```
extern long long a;
extern char *foo(int);
```

然后在 main.c 和 foo.c 中分别引用该头文件，请问编译链接能通过吗？请说明理由。

Q13

main.c

	局部或全局？	强或弱？
A	全局	强
foo	全局	弱

foo.c

	局部或全局？	强或弱？
A	全局	弱
foo	全局	强
out_buf	局部	X

```
long longa = 0xdeadbeefcafebffe; （1 分）
*(int*)((unsigned long long)a + 2) （2 分）
```

不能。无论如何声明 a 的类型都会造成在至少一个文件内引起声明和定义冲突。
结论 1 分，理由 2 分。（结论错不得分）

Q14

第四题（10 分）

在 x86_64 环境下，考虑如下 2 个文件：main.c 和 foo.c：

```
/* main.c */
#include <stdio.h>

long long _____________________________;
const char* foo(int);

int main(int argc, char **argv){
    int n = 0;
    sscanf(argv[1], "%d", &n);
    printf(foo(n));
    printf("%llx\n", a);
}
```

```
/* foo.c */
#include <stdio.h>

int a[2];

static void swapper(int num){
    int swapper;
    if (num % 2){
        swapper = a[0];
        a[0] = a[1];
        a[1] = swapper;
    }
}

const char* foo(int num){
    static char out_buf[50];
    swapper(num);
    sprintf(out_buf,                                    "%x\n",
_____________________________);
    return out_buf;
}
```

Q14

1. 对于每个程序中的相应符号，给出它的属性（局部或全局，强符号或弱符号）（提示：如果某表项中的内容无法确定，请画 X。）

main.c

	局部或全局？	强或弱？
a		
foo		

foo.c

	局部或全局？	强或弱？
a		
foo		
out_buf		

2. 根据如下的程序运行结果，补全程序【在程序空白处填空即可】。

```
$ gcc -o test main.c foo.c
$ ./test 1
bffedead
cafebffedeadbeef

$ ./test 2
beefcafe
deadbeefcafebffe
```

3. 现在有一位程序员要为这个程序编写头文件。假设新的头文件名称为 foo.h，内容如下：

```
extern long long a;
extern char *foo(int);
```

然后在 main.c 和 foo.c 中分别引用该头文件，请问编译链接能通过吗？请说明理由。

Q14

main.c

	局部或全局？	强或弱？
A	全局	强
foo	全局	弱

main.c中：a：根据后续题的分析，有初始值，因此为强符号

foo.c

	局部或全局？	强或弱？
A	全局	弱
foo	全局	强
out_buf	局部	X

foo.c中：outbuf：为static，是局部符号

```
long longa = 0xdeadbeefcafebffe; （1 分）
*(int*)((unsigned long long)a + 2) （2 分）
```

根据test2：因为2%2==0，因此不会进行swap，因此此时输出的为原始的a，可知，long long a = 0xdeadbeefcafebffe（第二次的第二个输出），此时用test1验证也是正确的（即后4字节和前4字节进行交换）

不能。无论如何声明 a 的类型都会造成在至少一个文件内引起声明和定义冲突。
结论 1 分，理由 2 分。（结论错不得分）

a有4*2个字节，而foo(n)返回的是在判断swap并swap后第3,4,5,6字节，因此是转化为unsigned long long类型+2，再强转为int*，即向后取2个字节。因此答案为 *(int*)((unsigned long long)a + 2)（即取3,4,5,6字节）

extern a 会与main.c 中的a产生声明和定义冲突

Q15

3. 有下面两个程序。将他们先分别编译为.o 文件，再链接为可执行文件。

main.c

```
#include <stdio.h>
        A
int foo(int n) {
    static int ans = 0;
    ans = ans + x;
    return n + ans;
}
int bar(int n);
void op(void) {
    x = x + 1;
}
int main() {
    for (int i = 0; i < 3; i++) {
        int a1 = foo(0);
        int a2 = bar(0);
        op();
        printf("%d %d ", a1, a2);
    }
    return 0;
}
```

count.c

```
        B
int bar(int n) {
    static int ans = 0;
    ans = ans + x;
    return n + ans;
}
```

(1) 当 A 处为 `int x = 1;`，B 处为 `int x;` 时，完成下表。如果某个变量不在符号表中，那么在名字那一栏打×；如果它在符号表中的名字含有随机数字，那么请用不同的四位数字区分多个不同的符号。对于局部符号，不需要填最后一栏。

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x			
	bar			
	ans			
count.o	x			
	bar			
	ans			

程序能够链接成功吗？如果可以，程序的运行结果是什么？如果不可以，链接器报什么错？

(2) 当 A 处为 `static int x = 1;`，B 处为 `static int x = 1;` 时，完成下表。

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x			
	bar			
	ans			
count.o	x			
	bar			
	ans			

程序能够链接成功吗？如果可以，程序的运行结果是什么？如果不可以，链接器报什么错？

(3) 当 A 处为 `int x = 1;`，B 处为 `int x = 1;` 时。程序能够链接成功吗？如果可以，程序的运行结果是什么？如果不可以，链接器报什么错？

Q15

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x	x	×	√
	bar	bar	×	×
	ans	ans.1597	√	
count.o	x	x	×	×
	bar	bar	×	√
	ans	ans.0344	√	

1 1 3 3 6 6

ans在两个板块中都是static，且都是过程间的静态变量，是不一样的，因此在符号表中要用不同的四位数字来区分

第1层循环后：ans.1597=0+1，ans.0344=0+1，x=2
第2层循环后：ans.1597=1+2，ans.0344=1+2，x=3
第3层循环后：ans.1597=3+3，ans.0344=3+3，x=4

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x	x	√	
	bar	bar	×	×
	ans	ans.1597	√	
count.o	x	x	√	
	bar	bar	×	√
	ans	ans.0344	√	

1 1 3 2 6 3。两个 x 在各自的.o 文件中的名字都为 x，因为它们不是过程中的静态变量。思考：对于非过程间的静态变量，为什么**编译器**不需要作这样的区分？

x都不是过程间的静态变量，因此编译器不需要对此进行区分

第1层循环后：ans.1597=0+1，ans.0344=0+1，x_1=2
第2层循环后：ans.1597=1+2，ans.0344=1+1，x_1=3
第3层循环后：ans.1597=3+3，ans.0344=2+1，x_1=4

一个文件里面过程间的静态变量可能不止一个
但是非过程间的静态变量只有一个

链接错误，x 被定义多次。

两处都为x=1，都为强符号，因此链接错误

Q16

3. 有下面两个程序。将他们先分别编译为.o 文件，再链接为可执行文件。

main.c

```
#include <stdio.h>
        A
int foo(int n) {
    static int ans = 0;
    ans = ans + x;
    return n + ans;
}
int bar(int n);
void op(void) {
    x = x + 1;
}
int main() {
    for (int i = 0; i < 3; i++) {
        int a1 = foo(0);
        int a2 = bar(0);
        op();
        printf("%d %d ", a1, a2);
    }
    return 0;
}
```

count.c

```
        B
int bar(int n) {
    static int ans = 0;
    ans = ans + x;
    return n + ans;
}
```

(1) 当 A 处为 `int x = 1;`，B 处为 `int x;` 时，完成下表。如果某个变量不在符号表中，那么在名字那一栏打×；如果它在符号表中的名字含有随机数字，那么请用不同的四位数字区分多个不同的符号。对于局部符号，不需要填最后一栏。

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x			
	bar			
	ans			
count.o	x			
	bar			
	ans			

程序能够链接成功吗？如果可以，程序的运行结果是什么？如果不可以，链接器报什么错？

(2) 当 A 处为 `static int x = 1;`，B 处为 `static int x = 1;` 时，完成下表。

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x			
	bar			
	ans			
count.o	x			
	bar			
	ans			

程序能够链接成功吗？如果可以，程序的运行结果是什么？如果不可以，链接器报什么错？

(3) 当 A 处为 `int x = 1;`，B 处为 `int x = 1;` 时。程序能够链接成功吗？如果可以，程序的运行结果是什么？如果不可以，链接器报什么错？

Q16

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x	x	×	√
	bar	bar	×	×
	ans	ans.1597	√	
count.o	x	x	×	×
	bar	bar	×	√
	ans	ans.0344	√	

1 1 3 3 6 6

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.o	x	x	√	
	bar	bar	×	×
	ans	ans.1597	√	
count.o	x	x	√	
	bar	bar	×	√
	ans	ans.0344	√	

1 1 3 2 6 3。两个 x 在各自的.o 文件中的名字都为 x，因为它们不是过程中的静态变量。思考：对于非过程间的静态变量，为什么**编译器**不需要作这样的区分？

链接错误，x 被定义多次。

Q17

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。

答案：D
D. RISC的寄存器数目一般更多

CISC和RISC详见书p249

D

Q18

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

答案：B
A. 错误。不同指令的机器码长度是不一定相同的（相信大家做了那么多题也知道，嗯）
B. 正确。都是相当于将%rax的值与0进行比较，并不更改%rax的值
C. 错误。如果switch的case比较稀疏，就不会生成跳转表

B

Q19

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];};

答案：B
sizeof(P) = 4 + 1 + 3(对齐) + 4 + 1 + 3(对齐) = 16
A. sizeof(P1) = 3 * 16 = 48
B. sizeof(P2) = 3 * 4 + 3 * 1 + 1(对齐) + 3 * 4 + 3 * 1(对齐) = 32
C. sizeof(P3) = 3 * 8 + 3 * 8 = 48
D. sizeof(P3) = 3 * 8 + 3 * 8 = 48
所以B不同，答案为B

B

Q20

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

答案：A
可以只使用一条指令：leal 7(%eax, %eax, 4), %ebx

A