1. 选择题



（13期中）1. x86 体系结构中，下面哪些选项是错误的？答：( )



A. leal 指令只能够用来计算内存地址



B. x86\_64 机器可以使用栈来给函数传递参数



C. 在一个函数内， 改变任一寄存器的值之前必须先将其原始数据保存在栈内



D. 判断两个寄存器中值大小关系，只需要 SF (符号) 和 ZF (零) 两个 conditional code



（14期中）2.下列的指令组中，那一组指令只改变条件码，而不改变寄存器的值？

A. CMP, SUB



B. TEST, AND

C. CMP, TEST

D. LEAL, CMP

（15期中）3.设某条 C 语言 switch 语句编译后产生了如下的汇编代码及跳转表：



|  |  |
| --- | --- |
| movl 8(%ebp), %eax | .L7: |
| subl $48, %eax | .long .L3 |
| cmpl $8, %eax | .long .L2 |
| ja .L2 | .long .L2 |
| jmp \*.L7(, %eax, 4) | |  |  | | --- | --- | | .long | .L5 | | .long | .L4 | | .long | .L5 | | .long | .L6 | | .long | .L2 | | .long | .L3 | |

在源程序中，下面的哪些(个) 标号出现过：



A. ‘2’ , ‘7’



B. 1



C. ‘3’

D. 5

（17期中）4.在下列关于条件传送的说法中， 正确的是：



A. 条件传送可以用来传送字节、字、双字、和 4 字的数据



B. C 语言中的“ ?: ”条件表达式都可以编译成条件传送



C. 使用条件传送总可以提高代码的执行效率



D. 条件传送指令不需要用后缀(例如 b, w, l, q)来表明操作数的长度

（17期中）5.有如下代码段：

|  |
| --- |
| int func(int x, int y);  int (\*p) (int a, int b);  p = func;  p(0,0); |

对应的下列 x86-64 过程调用正确的是：

A. call \*%rax B. call (%rax)

C. call \*(%rax) D. call func

（18期中）6.下列关于程序控制结构的机器代码实现的说法中，正确的是：



A) 使用条件跳转( conditional jump)语句实现的程序片段比使用条件赋值 (conditional move)语句实现的同一程序片段的运行效率高



B) 使用条件跳转语句实现的程序片段与使用条件赋值语句实现的同一程序片段 虽然效率可能不同，但在 C 语言的层面上看总是有着相同的行为

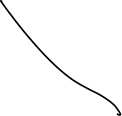


C) 一些 switch 语句不会被 gcc 用跳转表的方式实现

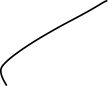


D) 以上说法都不正确

（18期中）7. 下列关于条件码的描述中，不正确的是()



A) 所有算术指令都会改变条件码



B) 所有比较指令都会改变条件码



C) 所有与数据传送有关的指令都会改变条件码



D) 条件码一般不会直接读取，但可以直接修改



1. 填空与简答题

（13期中）1.

/\*

\* Copyright (C) 2013 Davidlohr Bueso <davidlohr.bueso@hp.com> \*

\* Based on the shift-and-subtract algorithm for computing integer

\* square root from Guy L. Steele.

\*/

/\*\*

\* int\_sqrt - rough approximation to sqrt

\* @x: integer of which to calculate the sqrt \*

\* A very rough approximation to the sqrt() function. \*/

unsigned long int\_sqrt(unsigned long x)

{

unsigned long b, m, y = 0;

if (x <= 1)

return x;

m = 1UL << (BITS\_PER\_LONG - 2);

while (m != 0) {

b = y + m;

y >>= 1;

if (x >= b) {



x -= b;

y += m;

}

m >>= 2;

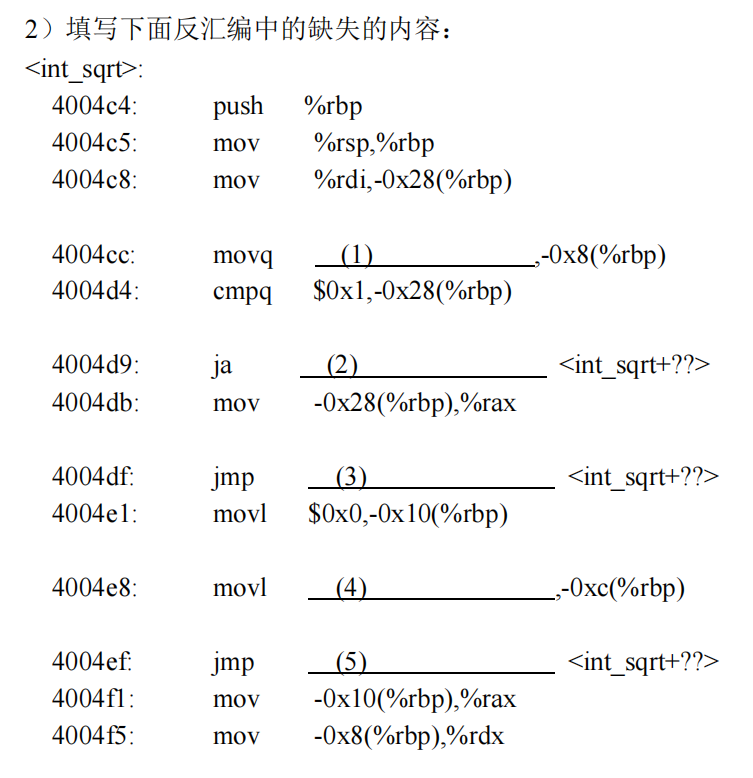
}

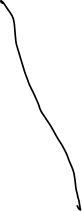
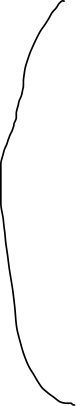
return y;

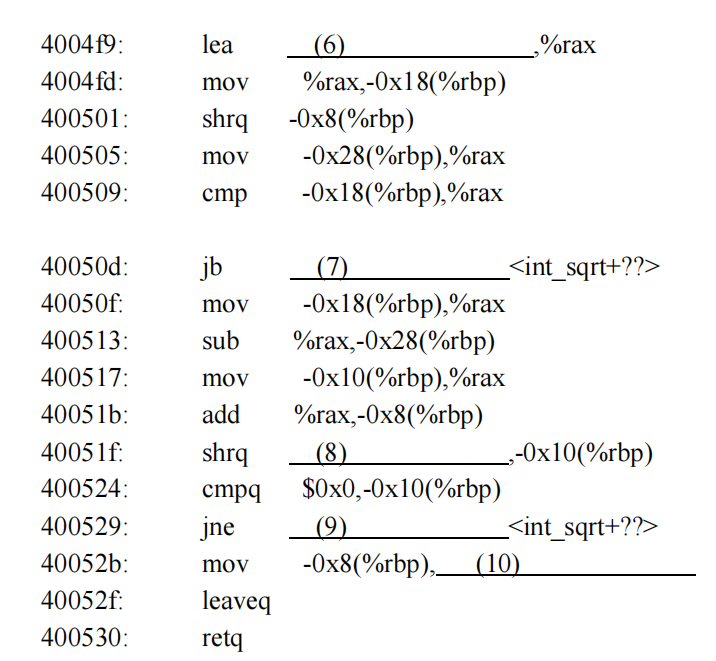
}

1) 在 64 位的机器上 BITS\_PER\_LONG 的定义为 long 类型的二进制位数，它是多少位？











（14期中）2.



一个函数如下， 其中部分代码被隐去，请通过gdb调试信息补全代码( 4分) 。



int f(int n, int m) {



if (m > 0) {



if (n > 1) {



int r = f(n - 1, m);



return (r - 1 + m) % n + 1;



}



else if (n == 1) {



return 1;

}

}

return 0;

}

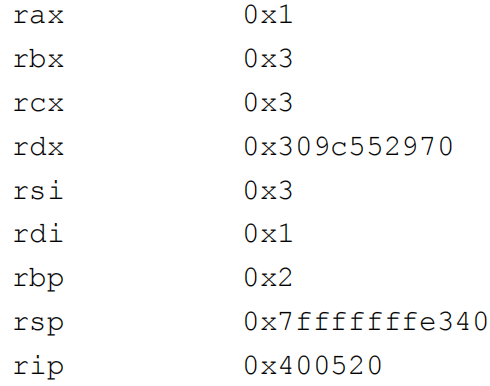
如下是通过“gcc –g –O2”命令编译后， 在gdb中通过“disas f”命令得到 的反汇编代码， 其中有两个汇编指令不全，请补全这两条汇编指令(2分) 。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x00000000004004e0 <f+0>: | mov | | %rbx,-0x10(%rsp) | |
| 0x00000000004004e5 <f+5>: | mov | | %rbp,-0x8(%rsp) | |
| 0x00000000004004ea <f+10>: | xor | | %eax,%eax | |
| 0x00000000004004ec <f+12>: | sub | | $0x10,%rsp | |
| 0x00000000004004f0 <f+16>: | test | | %esi,%esi | |
| 0x00000000004004f2 <f+18>: | mov | | %edi,%ebp | |
| 0x00000000004004f4 <f+20>: | mov | | %esi,%ebx | |
| 0x00000000004004f6 <f+22>: | jle | | 0x400513 <f+51> | |
| 0x00000000004004f8 <f+24>: | cmp | | $0x1,%edi | |
| 0x00000000004004fb <f+27>: | jle | | 0x400521 <f+65> | |
| 0x00000000004004fd <f+29>: | | lea | | -0x1(%rbp),%edi | |
| 0x0000000000400500 <f+32>: | | callq | | 0x4004e0 <f> | |
| 0x0000000000400505 <f+37>: | | lea | | -0x1(%rax,%rbx,1),%edx | |
| 0x0000000000400509 <f+41>: | | mov | | %edx,%eax | |
| 0x000000000040050b <f+43>: | | sar | | $0x1f,%edx | |
| 0x000000000040050e <f+46>: | | idiv | | %ebp | |
| 0x0000000000400510 <f+48>: | | lea | | 0x1(%rdx),%eax | |
| 0x0000000000400513 <f+51>: | | mov | | (%rsp),%rbx | |
| 0x0000000000400517 <f+55>: | | mov | | 0x8(%rsp),%rbp | |
| 0x000000000040051c <f+60>: | | add | | $0x10,%rsp | |
| 0x0000000000400520 <f+64>: | | retq | |  | |
| 0x0000000000400521 <f+65>: | | sete | | %al | |
| 0x0000000000400524 <f+68>: | | movzbl | | %al,%eax | |

0x0000000000400527 <f+71>: jmp 0x400513 <f+51>



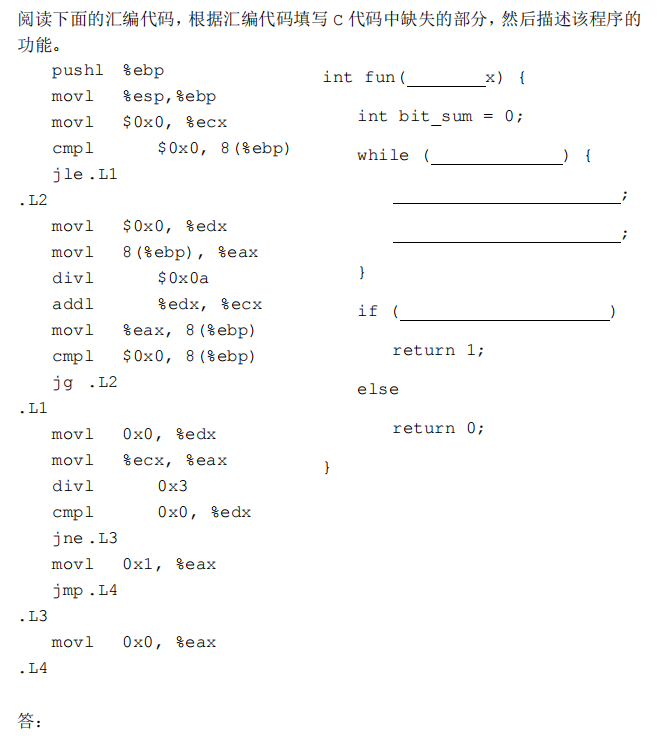
其中，已知在调用函数f(4, 3)时，我们在 函数f中指令retq处设置了断点，下面列 出的是程序在第一次运行到断点处暂停时 时，相关通用寄存器的值。请根据你对函 数及其汇编代码的理解， 填写当前栈中的 内容。如果某些内存位置处内容不确定， 请填写X。

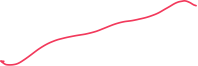


|  |  |
| --- | --- |
| 0x7fffffffe38c | X |
| 0x7fffffffe388 | X |
| 0x7fffffffe384 | X |
| 0x7fffffffe380 | X |
| 0x7fffffffe37c | X |
| 0x7fffffffe378 | X |
| 0x7fffffffe374 | 0x0 |
| 0x7fffffffe370 | 0x00400505 |
| 0x7fffffffe36c | 0x0 |
| 0x7fffffffe368 | 0x4 |
| 0x7fffffffe364 | 0x0 |
| 0x7fffffffe360 | 0x3 |
| 0x7fffffffe35c | 0x0 |
| 0x7fffffffe358 | 0x00400505 |
| 0x7fffffffe354 | 0x0 |
| 0x7fffffffe350 | 0x3 |
| 0x7fffffffe34c | 0x0 |
| 0x7fffffffe348 | 0x3 |
| 0x7fffffffe344 | 0x0 |
| 0x7fffffffe340 | 0x00400505 |
| 0x7fffffffe33c | 0x0 |
| 0x7fffffffe338 | 0x2 |
| 0x7fffffffe334 | 0x0 |
| 0x7fffffffe330 | 0x3 |
| 0x7fffffffe32c | X |
| 0x7fffffffe328 | X |
| 0x7fffffffe324 | X |
| 0x7fffffffe320 | X |



（15期中）3.





1. 答案与解析：



选择：

1.ACD

2.C，SUB 和 AND 都同时会改变条件码和寄存器的值，LEAL 不改变改变条码。

3. C，标号的取值范围为’0’ – ， ’1 ’、 ’2 ’、 ’7 ’、 ’9 ’、…等没有出现

4.D

解析： A.条件传送不支持单字节传送； B.如果“?:”涉及到的两个表达式中有一 个出错或者有副作用， 用条件传送会导致非法行为； C.如果被旁路的分支的计算 量很大，计算就白做了；D.从目标寄存器的名字可以推断出条件传送指令的操作 数长度

5.A/D 都对

解析： 用 -O0 编译可以得到 A，用 -Og 编译可以得到 D。

6. C。条件跳转本身开销大于条件赋值，但条件赋值会将两个分支中的运算都 完成， 故分支中的运算较为复杂时， 使用条件赋值语句实现的程序效率较低， 故 a 错误。分支中的运算带有副作用时， 条件跳转语句和条件赋值语句实现的程序行为 不同， 故 b 错误。 switch 语句中的 case 若比较稀疏， 则不会被用跳转表的方式实 现，故 c 正确。

7. ABCD 均给分。

C，leaq 是 movq 指令的变形，它只传送地址，不改变条件码。

AB 没有指明是 x86 指令系统； x86 指令系统中有 SIMD 类指令不改变条件码； C 数据传送一般不改变条件码；

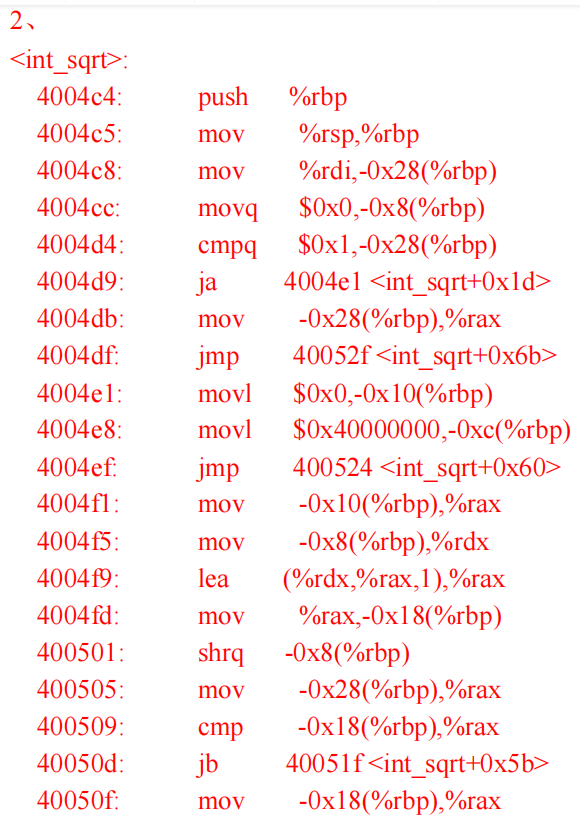
D 正确， SAHF、STC、CLC、CMC 等指令均可以直接写条件码。

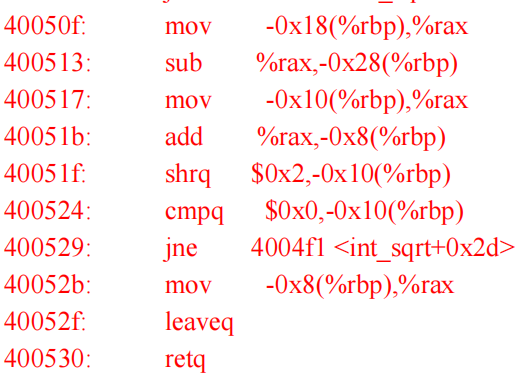
填空与解答：

1.

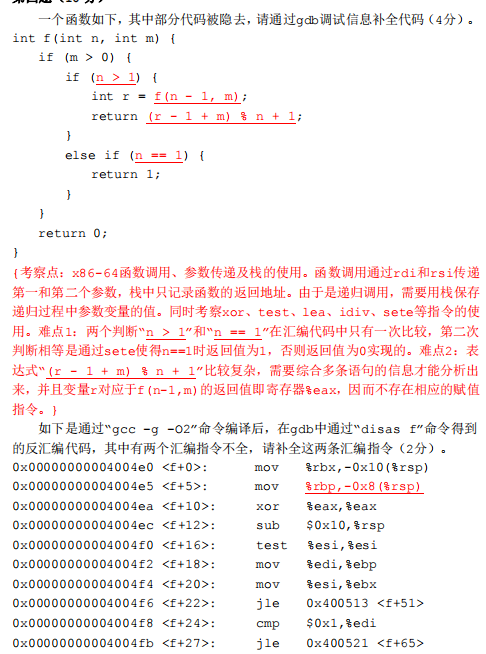
1）64

2）





2.





3.

