第三题（15 分）请阅读并分析下面的 C 语言程序和对应的 x86-64 汇编代码。

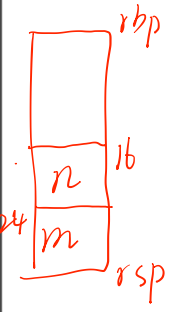
1．其中，有一部分缺失的代码（用标号标出），请在标号对应的横线上填写缺失的内容。注:汇编与机器码中的数字用 16 进制数填写。

C 代码如下：

```
long f(long n, long m)
{
  if (n == 0 || (1)m==1 )
  return m;
  if ((2)m&1 )
  {
    long ret = (3)f(n-1,3m+1);
    return ret;
  }
  else
  {
    long ret = f(n - 1, m >> 1);
    return ret;
  }
}
```

x86-64 汇编代码如下（为简单起见，函数内指令地址只给出后四位，需要时可补全）：

```
0x0000555555555149 <f>:
    5149:  f3 0f 1e fa          endbr64
    514d:  55                   push   %rbp
    514e:  48 89 e5             mov    (4)%rsp,%rbp
    5151:  48 83 ec 20          sub    $0x20,%rsp
    5155:  48 89 7d e8          mov    %rdi,-0x18(%rbp)
    5159:  48 89 75 e0          mov    %rsi,-0x20(%rbp)
    515d:  48 83 7d e8 00       cmpq   $0x0,-0x18(%rbp)n==0?
    5162:  74 (5)07             je     (6)516b
```




```
    5164:  48 83 7d e0 01       cmpq   $0x1,-0x20(%rbp)
    5169:  75 06                jne    5171 <f+0x28>
    516b:  48 8b 45 e0          mov    (7)-0x20(%rbp),%rax
    516f:  eb 5f                jmp    51d0 <f+0x87>
    5171:  48 8b 45 e0          mov    -0x20(%rbp),%rax(m)
    5175:  83 e0 01             and    $0x1,%eax
    5178:  48 85 c0             test   %rax,%rax
    517b:  74 ??          m为偶 je     (8)51ab
    517d:  48 8b 55 e0          mov    -0x20(%rbp),%rdx(m)
```

6b
-64
7

d0
-5f
71

```
5181:  48 89 d0           mov    %rdx,%rax
5184:  48 01 c0           add    %rax,%rax
5187:  48 01 d0           add    %rdx,%rax
518a:  48 8d 50 01        lea    0x1(%rax),%rdx
518e:  48 8b 45 e8        mov    -0x18(%rbp),%rax
5192:  48 83 e8 01        sub    $0x1,%rax
5196:  48 89 d6           mov    (9) %rdx.  ,%rsi
5199:  48 89 c7           mov    %rax,%rdi
519c:  e8 a8 ff ff ff     callq  5149 <f>
51a1:  48 89 45 f8        mov    %rax,-0x8(%rbp)
51a5:  48 8b 45 f8        mov    -0x8(%rbp),%rax
51a9:  eb 25              jmp    51d0 <f+0x87>
51ab:  48 8b 45 e0        mov    -0x20(%rbp),%rax
51af:  48 d1 f8           (10) sar    %rax
51b2:  48 89 c2           mov    %rax,%rdx
51b5:  48 8b 45 e8        mov    -0x18(%rbp),%rax
51b9:  48 83 e8 01        sub    $0x1,%rax
51bd:  48 89 d6           mov    (9) %rdx.  ,%rsi
51c0:  48 89 c7           mov    %rax,%rdi
51c3:  e8 81 ff ff ff     callq  5149 <f>
51c8:  48 89 45 f0        mov    %rax,-0x10(%rbp)
51cc:  48 8b 45 f0        mov    -0x10(%rbp),%rax
51d0:  c9                 leaveq
51d1:  c3                 retq
```

2. 已知在调用函数 f(7,6)时，我们在 gdb 中使用 b f 指令在函数 f 处加上了断点，下面是程序某一次运行到断点时从栈顶开始的栈的内容，请在空格中填入相应的值。（U 表示不要求填写）

0x7fffffffe558	0x00005555555551c8
0x7fffffffe550	(11) 0x7fff ffff ffff e580
0x7fffffffe548	U
0x7fffffffe540	U
0x7fffffffe538	U
0x7fffffffe530	(12) 0x0000 0000 0000 0003
0x7fffffffe528	(13) 0x0000 5555 5555 51c8
0x7fffffffe520	0x00007fffffffe550
0x7fffffffe518	U

0x7fffffffe510	U
0x7fffffffe508	(14) 0x00 00 00 00 00 00 00 03
0x7fffffffe500	0x0000000000000010
0x7fffffffe4f8	0x00005555555551c8 偶

3．运行函数 f(7,6)后得到的值是多少？ （15） 7

第三题（15 分）

分析下面C语言程序和相应的x86-64汇编程序。其中缺失部分代码（被遮挡），请在对应的横线上填写缺失的内容。

```
#include <stdio.h>
#include "string.h"

void myprint(char *str)
{
    char buffer[16];
    [redacted](buffer,str);
    printf("%s \n",buffer);
}

void alert(void)
{
    printf("[redacted]\n");
}

int main(int argc,char *argv[])
{
    myprint("12345671234567123456712345671234567\xaa\x84\x04\x08");
    return 0;
}

*****************************************************************
    .section    .rodata
.LC0:
    .string "[redacted]"
    .text
    .globl  myprint
    .type   myprint, @function
myprint:
.LFB0:

    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $48, [redacted]
    movq    %rdi, -40(%rbp)
    movq    %fs:40, [redacted]
    movq    %rax, -8(%rbp)
    xorl    %eax, %eax
```

① strcpy

② Where am I?

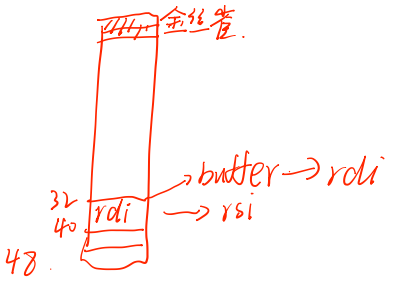
③ %s \n

④ %rsp.

⑤ %rax

疑问：
Where am I?
没加 \n.
这个是否要加 \n.

rax=0.



```
    movq    [redacted], %rdx
    leaq    -32(%rbp), %rax
    movq    %rdx, [redacted]
    [redacted]
    call    strcpy
    [redacted]  -32(%rbp), %rax
    movq    %rax, %rsi
    movl    $.LC0, %edi
    movl    $0, %eax
    call    printf
```

⑥ -40(%rbp)

⑦ %rsi.

⑧ movq %rax %rdi.

⑨ leaq.

```
    nop
    [redacted], %rax
    xorq    [redacted]
    je  [redacted]
    call    __stack_chk_fail
.L2:
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   myprint, .-myprint
    .section    .rodata
.LC1:
    .string "Where am I?"
    .text
    .globl  alert
    .type   alert, @function
alert:
.LFB1:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    movl    $.LC1, %edi
    call    puts
    nop
    popq    %rbp
    .cfi_def_cfa 7, 8
    [redacted]
    .cfi_endproc
.LFE1:
    .size   alert, .-alert
    .section    .rodata
    .align 8
.LC2:
    .string "1234567123456712345671234567\252\[redacted]\004\b"
    .text
    .globl  main
    .type   main, @function
main:
.LFB2:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $16, %rsp
```

⑩ %fs:40.

⑪ -8(%rbp), %rax.

⑫ .L2.

⑬ rt.

⑭ 204

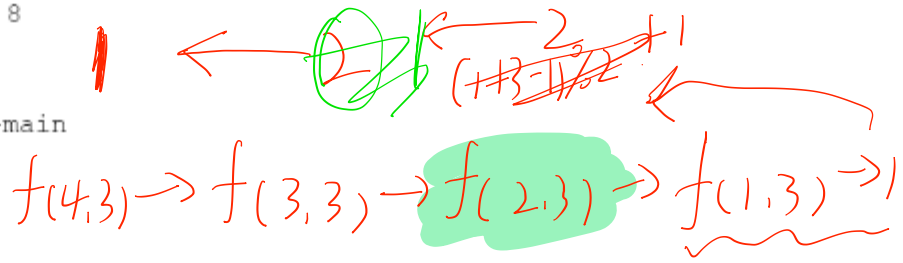
84 = 8×16+4

= 8×8×2+4

= 204

```
    movl    %edi, -4(%rbp)
    movq    %rsi, -16(%rbp)
    movl    $.LC2, %edi
    [redacted]
    movl    $0, %eax
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE2:
    .size   main, .-main
```

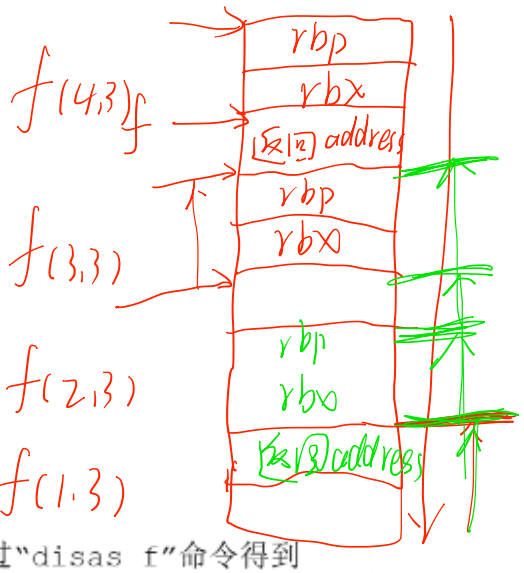
call myprint



第四题（10 分）

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）。

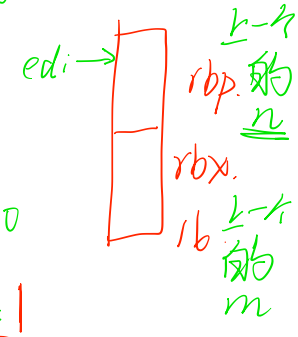
```
int f(int n, int m) {
    if (m > 0) {
        if (  n>0 1  ) {
            int r =  f(n-1,m)  ;
            return  (r+m-1)%n+1  ;
        }
        else if (  n==1  ) {
            return 1;
        }
    }
    return 0;
}
```



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>
```

rdi = n
rsi = m.
rax = 0.
m ≤ 0
n ≤ 1
n-1



f(n-1, m)

```
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>
```

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。（4分）

rax	0x1
rbx	0x3
rcx	0x3
rdx	0x309c552970
rsi	0x3
rdi	0x1
rbp	0x2
rsp	0x7fffffffe340
rip	0x400520

0x7fffffffe38c	X
0x7fffffffe388	X
0x7fffffffe384	X
0x7fffffffe380	X
0x7fffffffe37c	X
0x7fffffffe378	X
0x7fffffffe374	X
0x7fffffffe370	X
0x7fffffffe36c	X
0x7fffffffe368	X
0x7fffffffe364	X
0x7fffffffe360	X
0x7fffffffe35c	0
0x7fffffffe358	0x400540
0x7fffffffe354	0
0x7fffffffe350	4
0x7fffffffe34c	0
0x7fffffffe348	3
0x7fffffffe344	0
0x7fffffffe340	0x400540
0x7fffffffe33c	0
0x7fffffffe338	3
0x7fffffffe334	0
0x7fffffffe330	3
0x7fffffffe32c	0
0x7fffffffe328	0x400540
0x7fffffffe324	0
0x7fffffffe320	2