A. 一些带立即数和地址偏移量的操作：

```
0x100: 30f3fcffffffffffff |     irmovq $-4,%rbx
0x10a: 4063000800000000000 |     rmmovq %rsi,0x800(%rbx)
0x114: 00                  |     halt
```

B. 包含一个函数调用的代码：

```
0x200: a06f                |     pushq %rsi
0x202: 800c02000000000000  |     call proc
0x20b: 00                  |     halt
0x20c:                     | proc:
0x20c: 30f30a00000000000000 |     irmovq $10,%rbx
0x216: 90                  |     ret
```

C. 包含非法指令指示字节 0xf0 的代码：

```
0x300: 50540700000000000000 |     mrmovq 7(%rsp),%rbp
0x30a: 10                  |     nop
0x30b: f0                  | .byte 0xf0  # Invalid instruction code
0x30c: b01f                |     popq %rcx
```

D. 包含一个跳转操作的代码：

```
0x400:                     | loop:
0x400: 6113                |     subq %rcx, %rbx
0x402: 730004000000000000  |     je loop
0x40b: 00                  |     halt
```

E. pushq 指令中第二个字节非法的代码。

```
0x500: 6362                |     xorq %rsi,%rdx
0x502: a0                  |     .byte 0xa0  # pushq instruction
code
0x503: f0                  |     .byte 0xf0  # Invalid register
specifier byte
```

4.3　使用 iaddq 指令，我们将 sum 函数重新编写为

```
# long sum(long *start, long count)
# start in %rdi, count in %rsi
sum:
        xorq %rax,%rax          # sum = 0
        andq %rsi,%rsi          # Set condition codes
        jmp    test
loop:
        mrmovq (%rdi),%r10      # Get *start
        addq %r10,%rax          # Add to sum
        iaddq $8,%rdi           # start++
        iaddq $-1,%rsi          # count--
test:
        jne    loop             # Stop when 0
        ret
```

4.4　在 x86-64 机器上运行时，GCC 生成如下 rsum 代码：

```
  long rsum(long *start, long count)
  start in %rdi, count in %rsi
rsum:
  movl    $0, %eax
  testq   %rsi, %rsi
  jle     .L9
  pushq   %rbx
  movq    (%rdi), %rbx
  subq    $1, %rsi
  addq    $8, %rdi
```