```
; ModuleID = 'Output/test_0.clang.bc'
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

%struct.regs = type <{ i64, i64, i64, i64, i64, i64, i64, i64, i64, i64,
  %i64, i64, i64, i64, i64, i64, i64, i1, i1, i1, i1, i1, i1, i1, [8 x
  %x86_fp80], i1, i1, i3, i1, i1, i1, i1, i1, i1, i1, i1, i1, i1, i1, i1,
  %i2, i2, i1, i1, i1, i1, i1, i1, [8 x i8], i16, i64, i16, i64, i11, i128,
  %i128, i128, i128, i128, i128, i128, i128, i128, i128, i128, i128, i128,
  %i128, i128, i128, i64, i64 }>

; Function Attrs: nounwind
define internal x86_64_sysvcc void @sub_0(%struct.regs*) #0 {
entry:
  %RSP_val = alloca i64
  %RBP_val = alloca i64
  %RDI_val = alloca i64
  %RSI_val = alloca i64
  %RDX_val = alloca i64
  %RCX_val = alloca i64
  %RBX_val = alloca i64
  %RAX_val = alloca i64

  %RAX = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 0
  %foo1 = load i64, i64* %RAX
  store i64 %foo1, i64* %RAX_val
  %RBX = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 1
  %foo2 = load i64, i64* %RBX
  store i64 %foo2, i64* %RBX_val
  %RCX = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 2
  %foo3 = load i64, i64* %RCX
  store i64 %foo3, i64* %RCX_val
  %RDX = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 3
  %foo4 = load i64, i64* %RDX
  store i64 %foo4, i64* %RDX_val
  %RSI = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 4
  %foo5 = load i64, i64* %RSI
  store i64 %foo5, i64* %RSI_val
  %RDI = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 5
  %foo6 = load i64, i64* %RDI
  store i64 %foo6, i64* %RDI_val
  %RSP = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 6
  %foo7 = load i64, i64* %RSP
  store i64 %foo7, i64* %RSP_val
  %RBP = getelementptr inbounds %struct.regs, %struct.regs* %0, i64 0, i32 7
  %foo8 = load i64, i64* %RBP
  store i64 %foo8, i64* %RBP_val

  ; push %RBP ; mov %RSP -> %RBP
  %foo77 = load i64, i64* %RBP_val
  %foo78 = load i64, i64* %RSP_val
  %foo79 = add i64 %foo78, -8
  %foo80 = inttoptr i64 %foo79 to i64*
  store i64 %foo77, i64* %foo80
  store i64 %foo79, i64* %RSP_val
  store i64 %foo79, i64* %RBP_val

  ; lea    -0x10(%RBP),%RAX
  %foo81 = add i64 %foo78, -24
  %foo82 = inttoptr i64 %foo81 to i64*
  %foo83 = ptrtoint i64* %foo82 to i64
  store i64 %foo83, i64* %RAX_val

  ; movl   $0x0,-0x4(%RBP)
  %foo84 = add i64 %foo78, -12
  %foo85 = inttoptr i64 %foo84 to i64*
  %foo86 = bitcast i64* %foo85 to i32*
  store i32 0, i32* %foo86

  ; mov    %RAX,-0x18(%RBP)
  %foo87 = load i64, i64* %RBP_val
  %foo88 = add i64 %foo87, -24
  %foo89 = inttoptr i64 %foo88 to i64*
  %foo90 = load i64, i64* %RAX_val
  store i64 %foo90, i64* %foo89

  ; mov    -0x18(%RBP),%RAX
  %foo91 = load i64, i64* %RBP_val
  %foo92 = add i64 %foo91, -24
  %foo93 = inttoptr i64 %foo92 to i64*
  %foo94 = load i64, i64* %foo93
  store i64 %foo94, i64* %RAX_val

  ; movl   $0x1,(%RAX)
  %foo95 = inttoptr i64 %foo94 to i64*
  %foo96 = bitcast i64* %foo95 to i32*
  store i32 1, i32* %foo96

  ; mov    -0x18(%RBP),%RAX
  %foo97 = load i64, i64* %RBP_val
  %foo98 = add i64 %foo97, -24
  %foo99 = inttoptr i64 %foo98 to i64*
  %foo100 = load i64, i64* %foo99
  store i64 %foo100, i64* %RAX_val

  ; movl   $0x2,0x4(%RAX)
  %foo101 = add i64 %foo100, 4
  %foo102 = inttoptr i64 %foo101 to i64*
  %foo103 = bitcast i64* %foo102 to i32*
  store i32 2, i32* %foo103

  ; mov    -0x18(%RBP),%RAX
  %foo104 = load i64, i64* %RBP_val
  %foo105 = add i64 %foo104, -24
  %foo106 = inttoptr i64 %foo105 to i64*
  %foo107 = load i64, i64* %foo106
  store i64 %foo107, i64* %RAX_val

  ; mov    0x4(%RAX),%fooeax
  %foo108 = add i64 %foo107, 4
  %foo109 = inttoptr i64 %foo108 to i64*
```