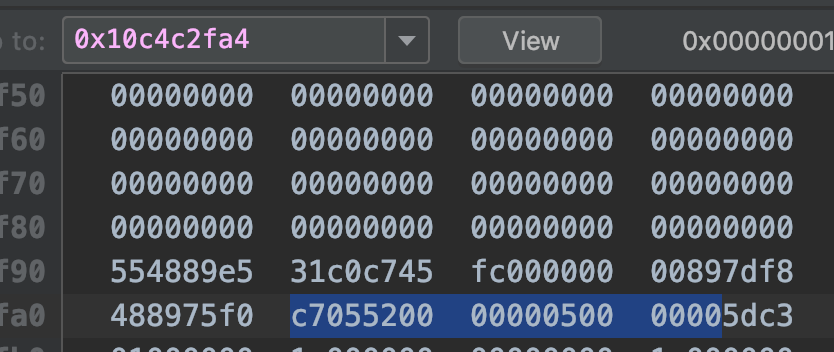
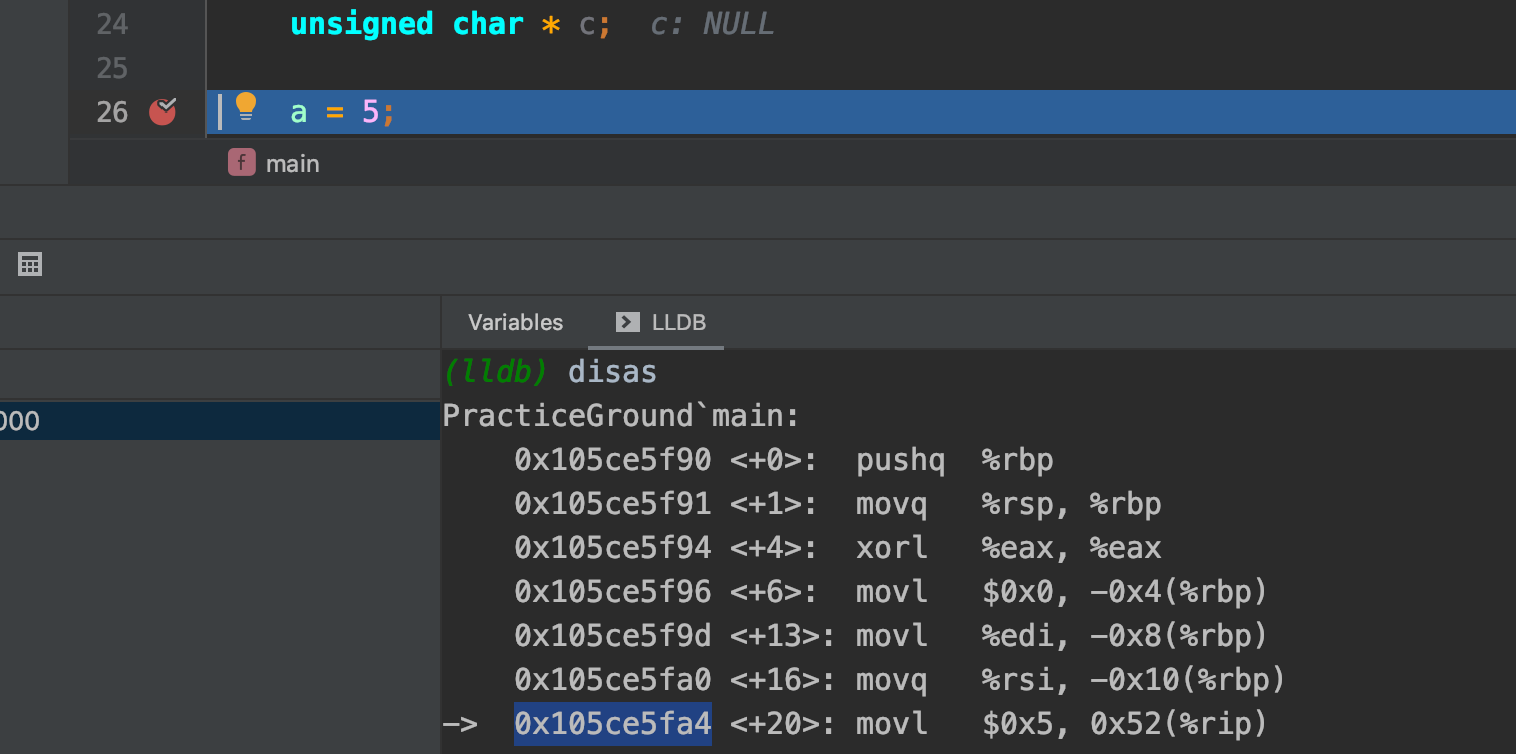
### 程序1:查看a = 5的机器码





### 程序2：一个带有算术运算的程序：

对应的汇编代码为如下

0x108e73f5d <+29>: movl 0xb5(%rip), %edi ; b

0x108e73f63 <+35>: imull $0x3, 0xb2(%rip), %ecx ; b + 3

0x108e73f6a <+42>: addl %ecx, %edi

0x108e73f6c <+44>: movl %edi, 0xae(%rip) ; a

使用了指令寄存器rip 变址寄存器edi 以及计数器 ecx

### 程序3：一个带有位运算的程序：

c = a&b; 对应的汇编代码是：

0x10bae4f6d <+29>: movl 0xa5(%rip), %edi ; a

0x10bae4f73 <+35>: andl 0xa3(%rip), %edi ; b

0x10bae4f79 <+41>: movl %edi, 0xa1(%rip) ; c

### 程序4：一个带有逻辑运算的程序：

0x103f0af4a <+26>: cmpl $0x0, 0xc7(%rip) ; (void \*)0x0000050000000400

0x103f0af51 <+33>: movb %cl, -0x11(%rbp)

0x103f0af54 <+36>: je 0x103f0af67 ; <+55> at main.cpp

0x103f0af5a <+42>: cmpl $0x0, 0xbb(%rip) ; a + 3

0x103f0af61 <+49>: setne %al

0x103f0af64 <+52>: movb %al, -0x11(%rbp)

0x103f0af67 <+55>: movb -0x11(%rbp), %al

0x103f0af6a <+58>: leaq 0x41(%rip), %rdi ; "'\n'"

0x103f0af71 <+65>: andb $0x1, %al

0x103f0af73 <+67>: movzbl %al, %ecx

0x103f0af76 <+70>: movl %ecx, 0xa4(%rip) ; c

### 程序5：一个if分支的程序

if对应的汇编代码为：

0x1003d0f56 <+22>: movl $0x5, 0xb8(%rip)

0x1003d0f60 <+32>: cmpl $0x0, 0xb1(%rip) ; (void \*)0x0000000000000500

0x1003d0f67 <+39>: jle 0x1003d0f7e ; <+62> at main.cpp

0x1003d0f7e <+62>: xorl %eax, %eax

其框架是，首先比较0与后面指令寄存器所保存的地址通过间接寻址方式得到的地址中存放的值，其中是之前通过movl指令将常数5存放进去了的，然后条件跳转，若不符合条件则跳转到地址0x103f0af67执行所包含的命令，其中我们可以知道xorl %eax, %eax

这一句指令代表对应return0之后函数执行完毕的汇编代码。

### 程序6：一个if-else分支的程序

0x101690f36 <+22>: movl $0x5, 0xd8(%rip)

-> 0x101690f40 <+32>: cmpl $0x0, 0xd1(%rip)

0x101690f47 <+39>: jle 0x101690f63 ; <+67> at main.cpp

0x101690f4d <+45>: leaq 0x4a(%rip), %rdi ; "a>0 \n"

0x101690f54 <+52>: movb $0x0, %al

0x101690f56 <+54>: callq 0x101690f7c ; symbol stub for: printf

0x101690f5b <+59>: movl %eax, -0x14(%rbp)

0x101690f5e <+62>: jmp 0x101690f74 ; <+84> at main.cpp

0x101690f63 <+67>: leaq 0x3a(%rip), %rdi ; "a<0 \n"

0x101690f6a <+74>: movb $0x0, %al

0x101690f6c <+76>: callq 0x101690f7c ; symbol stub for: printf

0x101690f71 <+81>: movl %eax, -0x18(%rbp)

0x101690f74 <+84>: xorl %eax, %eax

其框架是：首先同样对比0与通过简介寻址ip寄存器得到的地址中的值，然后条件不满足则跳转到0x100000f63这个地址的指令，从上面我们可以看到这个这个地址对应的是printf函数部分相关的指令，而若满足条件则进行另一段printf的汇编，并且通过跳转指令跳转到函数结束部分0x101690f74而越过else的代码部分执行。

### 程序7：一个switch分支的程序

0x10b32af32 <+18>: movl $0x2, 0xc4(%rip)

-> 0x10b32af3c <+28>: movl 0xbe(%rip), %edi ; a

0x10b32af42 <+34>: movl %edi, %eax

0x10b32af44 <+36>: subl $0x1, %eax

0x10b32af47 <+39>: movl %edi, -0x14(%rbp)

0x10b32af4a <+42>: movl %eax, -0x18(%rbp)

0x10b32af4d <+45>: je 0x10b32af80 ; <+96> at main.cpp:168

0x10b32af53 <+51>: jmp 0x10b32af58 ; <+56> at main.cpp:165

0x10b32af58 <+56>: movl -0x14(%rbp), %eax

0x10b32af5b <+59>: subl $0x2, %eax

0x10b32af5e <+62>: movl %eax, -0x1c(%rbp)

0x10b32af61 <+65>: je 0x10b32af8f ; <+111> at main.cpp:172

0x10b32af67 <+71>: jmp 0x10b32af6c ; <+76> at main.cpp:165

0x10b32af6c <+76>: movl -0x14(%rbp), %eax

0x10b32af6f <+79>: subl $0x3, %eax

0x10b32af72 <+82>: movl %eax, -0x20(%rbp)

0x10b32af75 <+85>: je 0x10b32af99 ; <+121> at main.cpp:175

0x10b32af7b <+91>: jmp 0x10b32afa8 ; <+136> at main.cpp:179

0x10b32af80 <+96>: movl $0x1, 0x76(%rip)

0x10b32af8a <+106>: jmp 0x10b32afb2 ; <+146> at main.cpp

0x10b32af8f <+111>: movl $0x2, 0x67(%rip)

0x10b32af99 <+121>: movl $0x3, 0x5d(%rip)

0x10b32afa3 <+131>: jmp 0x10b32afb2 ; <+146> at main.cpp

0x10b32afa8 <+136>: movl $0x5, 0x4e(%rip)

0x10b32afb2 <+146>: xorl %eax, %eax

程序的框架为：首先先将a的值取出来然后做减法，如果结果为零则跳转到给a赋值为一的语句，否则将跳转到继续将a与case含有的值减法并判是否为0（这个过程不改变a的值）.并且如果在case中含有break则跳转到最后的return代表的汇编代码处，然后按顺序将a减去case后面跟的值2和3，都是当结果为零的时候跳转到对应的case需要执行的代码处 然后最后default是如果之前的case都没有break则会自然执行到default所对应的汇编 然后直接跳转到接下来的代码也就是return 0.

### 程序8：一个while循环的程序

0x109b6cf7c <+28>: cmpl $0x0, 0x7d(%rip)

0x109b6cf83 <+35>: jle 0x109b6cfa6 ; <+70> at main.cpp

0x109b6cf89 <+41>: movl -0x14(%rbp), %eax

0x109b6cf8c <+44>: addl $0x1, %eax

0x109b6cf8f <+47>: movl %eax, -0x14(%rbp)

0x109b6cf92 <+50>: movl 0x68(%rip), %eax ; a

0x109b6cf98 <+56>: addl $-0x1, %eax

0x109b6cf9b <+59>: movl %eax, 0x5f(%rip) ; a

0x109b6cfa1 <+65>: jmp 0x109b6cf7c ; <+28> at main.cpp:205

0x109b6cfa6 <+70>: xorl %eax, %eax

整体框架为先比较a和零的值，若不符合a>0则跳转到return0对应的汇编指令所在的地址，如果符合则执行对于b的加法以及a的减法操作，然后跳转会cmpl指令再次进行比较

### 程序9：一个do-while循环的程序

0x107e7cf82 <+18>: movl $0x5, 0x74(%rip)

0x107e7cf8c <+28>: movl -0x14(%rbp), %eax

0x107e7cf8f <+31>: addl $0x1, %eax

0x107e7cf92 <+34>: movl %eax, -0x14(%rbp)

0x107e7cf95 <+37>: movl 0x65(%rip), %eax ; a

0x107e7cf9b <+43>: addl $-0x1, %eax

0x107e7cf9e <+46>: movl %eax, 0x5c(%rip) ; a

0x107e7cfa4 <+52>: cmpl $0x0, 0x55(%rip)

0x107e7cfab <+59>: jg 0x107e7cf8c ; <+28> at main.cpp:233

0x107e7cfb1 <+65>: xorl %eax, %eax

与while循环相近，但是do-while循环首先执行do中的代码对应的汇编代码，然后进行比较，如果符合a>0则跳转到地址0x107e7cf8c也就是do的代码部分，否则继续执行到达return0的汇编代码

### 程序10：一个for循环的程序

0x10e5aff6c <+28>: movl $0x5, 0x8a(%rip)

0x10e5aff76 <+38>: cmpl $0x0, 0x83(%rip)

0x10e5aff7d <+45>: jle 0x10e5affa6 ; <+86> at main.cpp

0x10e5aff83 <+51>: movl 0x7b(%rip), %eax ; b

0x10e5aff89 <+57>: addl $0x1, %eax

0x10e5aff8c <+60>: movl %eax, 0x72(%rip) ; b

0x10e5aff92 <+66>: movl 0x68(%rip), %eax ; a

0x10e5aff98 <+72>: addl $-0x1, %eax

0x10e5aff9b <+75>: movl %eax, 0x5f(%rip) ; a

0x10e5affa1 <+81>: jmp 0x10e5aff76 ; <+38> at main.cpp:260

0x10e5affa6 <+86>: xorl %eax, %eax

框架为，首先执行0x10e5aff6c地址处的a=5对应的赋值操作，然后进行将0与间接寻址rip得到的地址中的值作比较，如果不符合大于条件则跳转到xorl也就是return0所对应的汇编，否则继续执行对于b的加法操作，以及对a的减法操作，然后跳转回0x10e5aff76

也就是循环开始的地方。继续下一次的比较操作。

### 程序11：一个函数调用的程序

每一个函数在{之前都需要为函数开辟函数栈，具体的操作为：

pushq %rbp

movq %rsp, %rbp

subq $0x10, %rsp

也就是将先push一个栈底指针，然后先将栈顶指针指向栈底，再将栈顶指针减少16个字节个单位地址，获得这个函数的栈，如果有传入参数则通过di和si两个寄存器来传入，我们可以看到，main函数通过将要传递给add函数的参数传入ei和si两个寄存器，然后add函数通过从eisi中取出他们的值来实现参数传递

0x10589ff94 <+36>: movl -0x8(%rbp), %edi

0x10589ff97 <+39>: movl -0xc(%rbp), %esi

0x10589ff9a <+42>: callq 0x10589ff50

0x1006e8f50 <+0>: pushq %rbp

0x1006e8f51 <+1>: movq %rsp, %rbp

0x1006e8f54 <+4>: movl %edi, -0x4(%rbp)

0x1006e8f57 <+7>: movl %esi, -0x8(%rbp)

最后通过将结果放入加法寄存器ax然后main函数再从中取出ax的值来实现函数值返回

0x1006e8f63 <+19>: movl -0xc(%rbp), %eax

0x1006e8fa1 <+49>: movl %eax, -0x10(%rbp)

### 5.xaddy的汇编代码

.global xaddy,\_main

xaddy:

movl %ecx,%eax

addl %edx,%eax

retq

\_main:

movl $1,%ecx

movl $2,%edx

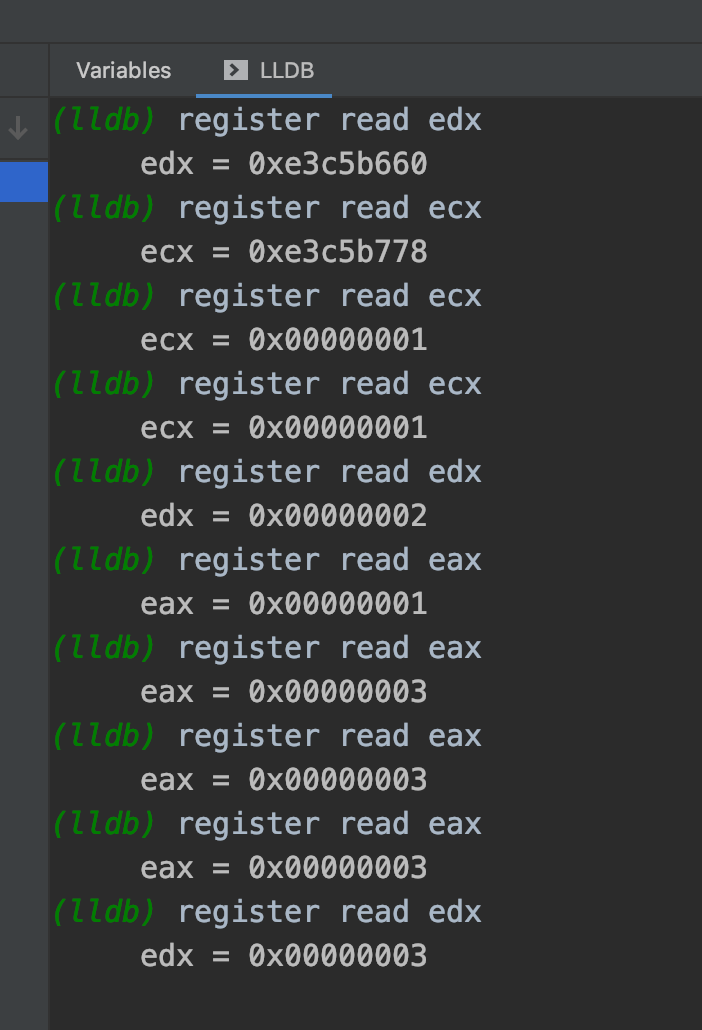
call xaddy

movl %eax,%edx

xorq %rax,%rax

retq

编写平台：mac



### 解码汇编对应的机器码

mov EAX, 0xdeadbeaf -> b8afbeadde

push 0X0028FE60 -> 6860fe2800

Ret -> c3

