Kunal Mukherjee
CS 6332
Assignment 5
Dr. Kangkook Jee

Part A

How to:

* You run through modules by using runOnModule() and functions using runOnFunction()
* use M.getGlobalList(): to get the global variables
* use G.hasInitializer(): to see if it is initialized
* use G.getInitializer(): to get the initialized value
* use F.getType(): to get the func proto type
* use F.getName(): to get the func name
* use F.getReturnType(): to get the func return type
* use i.getOpcodeName(): to get the opcode
* use opCounter.begin()->second: to get the operand

Output:

```
cs6332@cs6332-kxm180046:/home/cs6332/tools/llvm $ ./run_myllvmpass1.sh cs6332.001-f20-assign0x5-master/fib/fib.ll
==================PART1=================
Running on Module
Constants in Module

stderr
Global Name: .str
Global Var Type: [19 x i8]*
Global Initializer: 0x55555726cfb0

Global Name: .str1
Global Var Type: [7 x i8]*
Global Initializer: 0x55555726d0c0

Global Name: .str2
Global Var Type: [29 x i8]*
Global Initializer: 0x555557270cd0

Global Name: .str3
Global Var Type: [37 x i8]*
Global Initializer: 0x55555725e250

Global Name: .str4
Global Var Type: [4 x i8]*
Global Initializer: 0x55555725e130
```

***Function in Module
fib
Function Prototype: i32 (i32)*

fib_logger
Function Prototype: i32 (i32, i32)*

main
Function Prototype: i32 (i32, i8**)*

fprintf
Function Prototype: i32 (%struct._IO_FILE*, i8*, ...)*

exit
Function Prototype: void (i32)*

strtol
Function Prototype: i32 (i8*, i8**, i32)*

__errno_location
Function Prototype: i32* ()*

perror
Function Prototype: void (i8*)*

printf
Function Prototype: i32 (i8*, ...)*

***Function fib
Function Argument: size: 1 : i
Function Return Type: i32
Function Number of BB: 4

Function's BB done: 0
InstName: alloca, Operand #: 2
InstName: br, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 1
InstName: store, Operand #: 1

Function's BB done: 1

InstName: br, Operand #: 1
InstName: store, Operand #: 1

Function's BB done: 2
InstName: add, Operand #: 1
InstName: br, Operand #: 1
InstName: call, Operand #: 3
InstName: load, Operand #: 3
InstName: store, Operand #: 1
InstName: sub, Operand #: 3

Function's BB done: 3
InstName: load, Operand #: 1
InstName: ret, Operand #: 1

***Function main
Function Argument: size: 2 : argc: argv
Function Return Type: i32
Function Number of BB: 14

Function's BB done: 0
InstName: alloca, Operand #: 5
InstName: br, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 1
InstName: store, Operand #: 3

Function's BB done: 1
InstName: call, Operand #: 2
InstName: getelementptr, Operand #: 1
InstName: load, Operand #: 3
InstName: unreachable, Operand #: 1

Function's BB done: 2
InstName: br, Operand #: 1
InstName: call, Operand #: 2
InstName: getelementptr, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 3
InstName: store, Operand #: 1

Function's BB done: 3
InstName: br, Operand #: 1
InstName: icmp, Operand #: 1

InstName: load, Operand #: 1

Function's BB done: 4
InstName: br, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 1

Function's BB done: 5
InstName: br, Operand #: 1
InstName: call, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 1

Function's BB done: 6
InstName: br, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 1

Function's BB done: 7
InstName: call, Operand #: 2
InstName: unreachable, Operand #: 1

Function's BB done: 8
InstName: br, Operand #: 1
InstName: getelementptr, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 3

Function's BB done: 9
InstName: call, Operand #: 2
InstName: getelementptr, Operand #: 1
InstName: load, Operand #: 3
InstName: unreachable, Operand #: 1

Function's BB done: 10
InstName: br, Operand #: 1
InstName: icmp, Operand #: 1
InstName: load, Operand #: 2
InstName: sext, Operand #: 1

Function's BB done: 11
InstName: call, Operand #: 2
InstName: load, Operand #: 2
InstName: unreachable, Operand #: 1

Function's BB done: 12

InstName: call, Operand #: 3
InstName: load, Operand #: 2
InstName: store, Operand #: 1
InstName: unreachable, Operand #: 1

Function's BB done: 13
InstName: load, Operand #: 1
InstName: ret, Operand #: 1

Part B

How to:
* we will iterate over the whole module by runOnModule()
* we will find if it a "ret" or a "fib" call inst
* ret check: if `i.getOpcodeName().compare("ret") == 0 `
* call check: `isa<CallInst>(i) && cast<CallInst>(i).getCalledFunction()->getName().compare("fib") == 0`

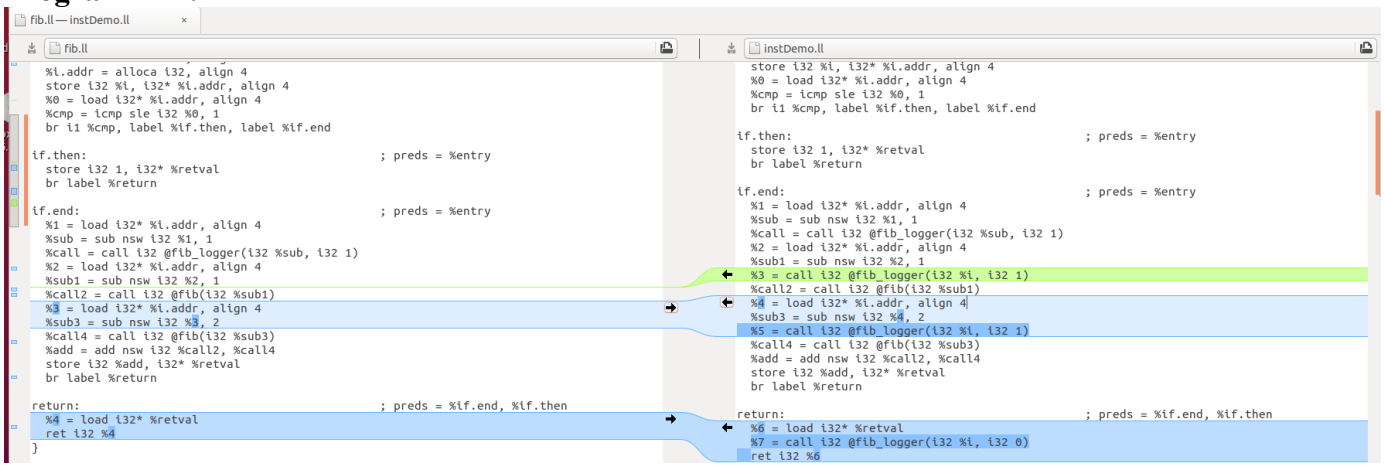
* then we will call the fiblogger func:
```
... create a funcTy a fiblogger definition
Constant* hook = M.getOrInsertFunction("fib_logger", funcTy);
std::vector<Value*> args;

... populate the args and insert the new call inst. before call to fib()
auto *newInst = CallInst::Create(hook,args);
bb.getInstList().insert(i, newInst);
```

Output:
cs6332@cs6332-kxm180046:/home/cs6332/tools/llvm $ ./run_myllvmpass2.sh cs6332.001-f20-assign0x5-master/fib/fib.ll 5
==========PART2==========
Inside opcode: call calling: fib
Inside opcode: call calling: fib
Inside opcode: ret
Inside opcode: call calling: fib
Inside opcode: ret
call fib(2)
call fib(4)
call fib(5)
call fib(3)
call fib(4)
call fib(2)
call fib(3)
call fib(1)

call fib(2)
fib() returns 1
call fib(2)
fib() returns 0
fib() returns 2
call fib(3)
fib() returns 1
fib() returns 3
call fib(4)
call fib(1)
call fib(2)
fib() returns 1
call fib(2)
fib() returns 0
fib() returns 2
fib() returns 4
call fib(5)
call fib(2)
call fib(3)
call fib(1)
call fib(2)
fib() returns 1
call fib(2)
fib() returns 0
fib() returns 2
call fib(3)
fib() returns 1
fib() returns 3
fib() returns 5
8

Program Diff:



Part C
How to:

Encryption
* Go through the entire module: runOnModule
* get the global list: M.getGlobalList()
* see if it is initialized: G.hasInitializer()
* and if the global is a string `G.getName().str().find("str")`
* get the initializer: `Value* newValue = G.getInitializer();`
* call the xor: `xorCipher(_dst, _src)`;
* set the new encrypted init: `G.setInitializer(const_array);`

Decryption
* go though the inst int BB in the functions in the module: `Instruction &i: bb`
* see if it is a load or call: `isa<LoadInst>(i)` or `isa<CallInst>(i)`
* see the operand : `i.getOperand(0)`
* also check if it is a GOP `isa<GEPOperator>(i.getOperand(0)`
* get the operator to find the operand array: `GEPOperator *gepo = dyn_cast<GEPOperator>(i.getOperand(0));`
* * then we will call the xor func:
```
... create a funcTy a fiblogger definition
Constant* hook = M.getOrInsertFunction("xor", funcTy);
std::vector<Value*> args;

... populate the args and insert the new call inst. before call to fib()
auto *newInst = CallInst::Create(hook,args);
bb.getInstList().insert(i, newInst);
```

Output:
cs6332@cs6332-kxm180046:/home/cs6332/tools/llvm $ ./run_myllvmpass3.sh cs6332.001-f20-assign0x5-master/db_connector/db_connector.ll
==========PART3==========
Running on Module
Constants in Module

Global Name: .str
Global Var Type: [7 x i8]*
Global Initializer: [7 x i8] c"cs6332\00"
src: cs6332 : dest: dt1445
New Global Var Type: [7 x i8]
Initialization value changed
New Global Initialization value: [7 x i8] c"dt1445\00"New Global Initialization value: [7 x i8] c"dt1445\00"

Global Name: .str1
Global Var Type: [15 x i8]*
Global Initializer: [15 x i8] c"mysecretpasswd\00"
src: mysecretpasswd : dest: j~tbdubswfttpc
New Global Var Type: [15 x i8]
Initialization value changed
New Global Initialization value: [15 x i8] c"j~tbdubswfttpc\00"New Global Initialization value: [15 x i8] c"j~tbdubswfttpc\00"

Global Name: cred_default
Global Var Type: %struct.cred_t*
Global Initializer: %struct.cred_t { i8* getelementptr inbounds ([7 x i8]* @.str, i32 0, i32 0), i8* getelementptr inbounds ([15 x i8]* @.str1, i32 0, i32 0) }

Global Name: pass_buf
Global Var Type: [16 x i8]*
Global Initializer: [16 x i8] zeroinitializer

Global Name: .str2
Global Var Type: [17 x i8]*
Global Initializer: [17 x i8] c"Input username: \00"
src: Input username: : dest: Niwrs'rtbuifjb='
New Global Var Type: [17 x i8]
Initialization value changed
New Global Initialization value: [17 x i8] c"Niwrs'rtbuifjb='\00"New Global Initialization value: [17 x i8] c"Niwrs'rtbuifjb='\00"

stdin
Global Name: .str3
Global Var Type: [25 x i8]*
Global Initializer: [25 x i8] c"Input error, exiting...\0A\00"
src: Input error, exiting...
 : dest: Niwrs'buuhu+'bnsni`)))
New Global Var Type: [25 x i8]
Initialization value changed
New Global Initialization value: [25 x i8] c"Niwrs'buuhu+'b\7Fnsni`)))\0D\00"New Global Initialization value: [25 x i8] c"Niwrs'buuhu+'b\7Fnsni`)))\0D\00"

Global Name: .str4
Global Var Type: [10 x i8]*
Global Initializer: [10 x i8] c"Password:\00"
src: Password: : dest: Wfttphuc=
New Global Var Type: [10 x i8]
Initialization value changed

New Global Initialization value: [10 x i8] c"Wfttphuc=\00"New Global Initialization value: [10 x i8] c"Wfttphuc=\00"

stderr
Global Name: .str5
Global Var Type: [42 x i8]*
Global Initializer: [42 x i8] c"Invalid credential input, using defaults\0A\00"
src: Invalid credential input, using defaults
 : dest: Niqfknc'dubcbisnfk'niwrs+'rtni`'cbafrkst
New Global Var Type: [42 x i8]
Initialization value changed
New Global Initialization value: [42 x i8] c"Niqfknc'dubcbisnfk'niwrs+'rtni`'cbafrkst\0D\00"New Global Initialization value: [42 x i8] c"Niqfknc'dubcbisnfk'niwrs+'rtni`'cbafrkst\0D\00"

Global Name: .str6
Global Var Type: [50 x i8]*
Global Initializer: [50 x i8] c"can we proceed with username %s and password %s? \00"
src: can we proceed with username %s and password %s? : dest: dfi'pb'wuhdbbc'pnso'rtbuifjb'"t'fic'wfttphuc'"t8'
New Global Var Type: [50 x i8]
Initialization value changed
New Global Initialization value: [50 x i8] c"dfi'pb'wuhdbbc'pnso'rtbuifjb'\22t'fic'wfttphuc'\22t8'\00"New Global Initialization value: [50 x i8] c"dfi'pb'wuhdbbc'pnso'rtbuifjb'\22t'fic'wfttphuc'\22t8'\00"

Global Name: .str7
Global Var Type: [13 x i8]*
Global Initializer: [13 x i8] c"Exiting ...\0A\00"
src: Exiting ...
 : dest: Bnsni`')))
New Global Var Type: [13 x i8]
Initialization value changed
New Global Initialization value: [13 x i8] c"B\7Fnsni`')))\0D\00"New Global Initialization value: [13 x i8] c"B\7Fnsni`')))\0D\00"

Global Name: .str8
Global Var Type: [31 x i8]*
Global Initializer: [31 x i8] c"PGPASSWORD=%s psql -h %s -U %s\00"
src: PGPASSWORD=%s psql -h %s -U %s : dest: W@WFTTPHUC:"t'wtvk'*o'"t'*R'"t
New Global Var Type: [31 x i8]
Initialization value changed
New Global Initialization value: [31 x i8] c"W@WFTTPHUC:\22t'wtvk'*o'\22t'*R'\22t\00"New Global Initialization value: [31 x i8] c"W@WFTTPHUC:\22t'wtvk'*o'\22t'*R'\22t\00"

Global Name: .str9
Global Var Type: [14 x i8]*

Global Initializer: [14 x i8] c"10.176.150.50\00"
src: 10.176.150.50 : dest: 67)601)627)27
New Global Var Type: [14 x i8]
Initialization value changed
New Global Initialization value: [14 x i8] c"67)601)627)27\00"New Global Initialization value: [14 x i8] c"67)601)627)27\00"

**

Call GEPOperator == GlobalVariable
inside GEPOperator: i8* getelementptr inbounds ([16 x i8]* @pass_buf, i32 0, i32 0)
XOR Called to decrypt

Call GEPOperator == GlobalVariable
inside GEPOperator: i8* getelementptr inbounds ([17 x i8]* @.str2, i32 0, i32 0)
XOR Called to decrypt

Load Operand contain a global var: @stdin = external global %struct._IO_FILE*
Call GEPOperator == GlobalVariable
inside GEPOperator: i8* getelementptr inbounds ([25 x i8]* @.str3, i32 0, i32 0)
XOR Called to decrypt

Call GEPOperator == GlobalVariable
inside GEPOperator: i8* getelementptr inbounds ([10 x i8]* @.str4, i32 0, i32 0)
XOR Called to decrypt

Call GEPOperator == GlobalVariable
inside GEPOperator: i8* getelementptr inbounds ([25 x i8]* @.str3, i32 0, i32 0)
XOR Called to decrypt

Load Operand contain a global var: @stderr = external global %struct._IO_FILE*
GEPOperator == GlobalVariable
inside GEPOperator: i8** getelementptr inbounds (%struct.cred_t* @cred_default, i32 0, i32 0)
XOR Called to decrypt

GEPOperator == GlobalVariable
inside GEPOperator: i8** getelementptr inbounds (%struct.cred_t* @cred_default, i32 0, i32 1)
XOR Called to decrypt

Call GEPOperator == GlobalVariable
inside GEPOperator: i8* getelementptr inbounds ([50 x i8]* @.str6, i32 0, i32 0)
XOR Called to decrypt

Load Operand contain a global var: @stderr = external global %struct._IO_FILE*
```
========================================
=           DB Connector               =
========================================
Niwrs'rtbuifjb='^C
```