Obfuscation

Paolo Falcarin

Ca' Foscari University of Venice

Department of Enviromental Sciences, Informatics and Statistics

paolo.falcarin@unive.it

CM0626 – Software Security

CM0631-2 – Software Security

4th March 2025

Obfuscation Goal

The goal of code obfuscation is to make a program more

- difficult to understand for a human than the original code
- difficult to analyse automatically for a reverse engineering tool in order to hide some asset in the program.

Vague definition of difficult:

- The obfuscated program requires more human time, more money, or more computing power to analyse than the original program.

Example: obfuscated code

Ca' Foscari University of Venice

```
public class C {
 static Object get0(Object[] I) {
  Integer I7, I6, I4, I3; int t9, t8;
  I7=new Integer(9);
  for (;;) {
   if (((Integer)I[0]).intValue()%((Integer)I[1]).intValue()==0)
       {t9=1; t8=0;} else {t9=0; t8=0;}
   I4=new Integer(t8);
   I6=new Integer(t9);
   if ((I4.intValue()^I6.intValue())!=0)
     return new Integer(((Integer)I[1]).intValue());
   else {
     if ((((I7.intValue()+ I7.intValue()*I7.intValue())%2!=0)?0:1)!=1)
        return new Integer(0);
     I3=new Integer(((Integer)I[0]).intValue()%
           ((Integer)I[1]).intValue());
     I[0]=new Integer(((Integer)I[1]).intValue());
     I[1]=new Integer(I3.intValue());
    }


                  }
```

Example: original code

- An **obfuscation tool** turns the original code into obfuscated code.
- We want that **obfuscating transformations** make the program as hard to understand as possible.

```
public class C {
   static int gcd(int x, int y) {
      int t;
      while (true) {
         boolean b = x % y == 0;
         if (b) return y;
         t = x % y; x = y; y = t;
      }
   }
}
```

This code computes the Greatest Common Denominator of its arguments...

Code Obfuscations

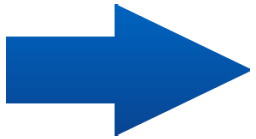
- Lexical transformations
  - Modify variable names (Identifier-Renaming)
  - Strings scrambling
- Control-flow transformations
  - Opaque Predicates -Redundant Code
  - Increase Indirection Levels
- Data transformations
  - Modify data structures
- Anti-disassembly
- Anti-debugging
- Code/Data Encoding or encryption

Identifier Renaming

Ca' Foscari University of Venice

Basic Obfuscation based on lexical transformations of all names (classes, methods, variables) into random strings with no semantics

```
public void addUserToList(String strRoomName, String strUser)
{
    RoomTabItem tab = getRoom(strRoomName);
    if(tab != null)
        tab.addUserToList(strUser);
}

public void removeUserFromList(String strRoomName, String strUser)
{
    RoomTabItem tab = getRoom(strRoomName);
    if(tab != null)
        tab.removeUserFromList(strUser);
}
```



```
public void k(String s, String s1)
{
    h h1 = h(s);
    if(h1 != null)
        h1.k(s1);
}

public void l(String s, String s1)
{
    h h1 = h(s);
    if(h1 != null)
        h1.l(s1);
}
```

Lexical Transformations

Ca' Foscari University of Venice

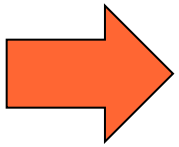
- The Stunnix obfuscator targets at obfuscating only the layout of the JavaScript code
- As the obfuscator parses the code, it removes spaces, comments and new line feeds
- While doing so, as it encounters user defined names, it replaces them with some random string
- It replaces print strings with their hexadecimal values
- It replaces integer values with complex equations

From Stunnix to Javascript

Ca' Foscari University of Venice

- **Actual code**:

```
function foo( arg1)
{
  var myVar1 =  "some string";  //first
    comment
  var intVar =  24 *  3600; //second
    comment
  /* here is a
  multi-line comment */
  document. write( "vars are:" +
    myVar1 +  " " +  intVar +  " " +  arg1)
    ;
} ;
```

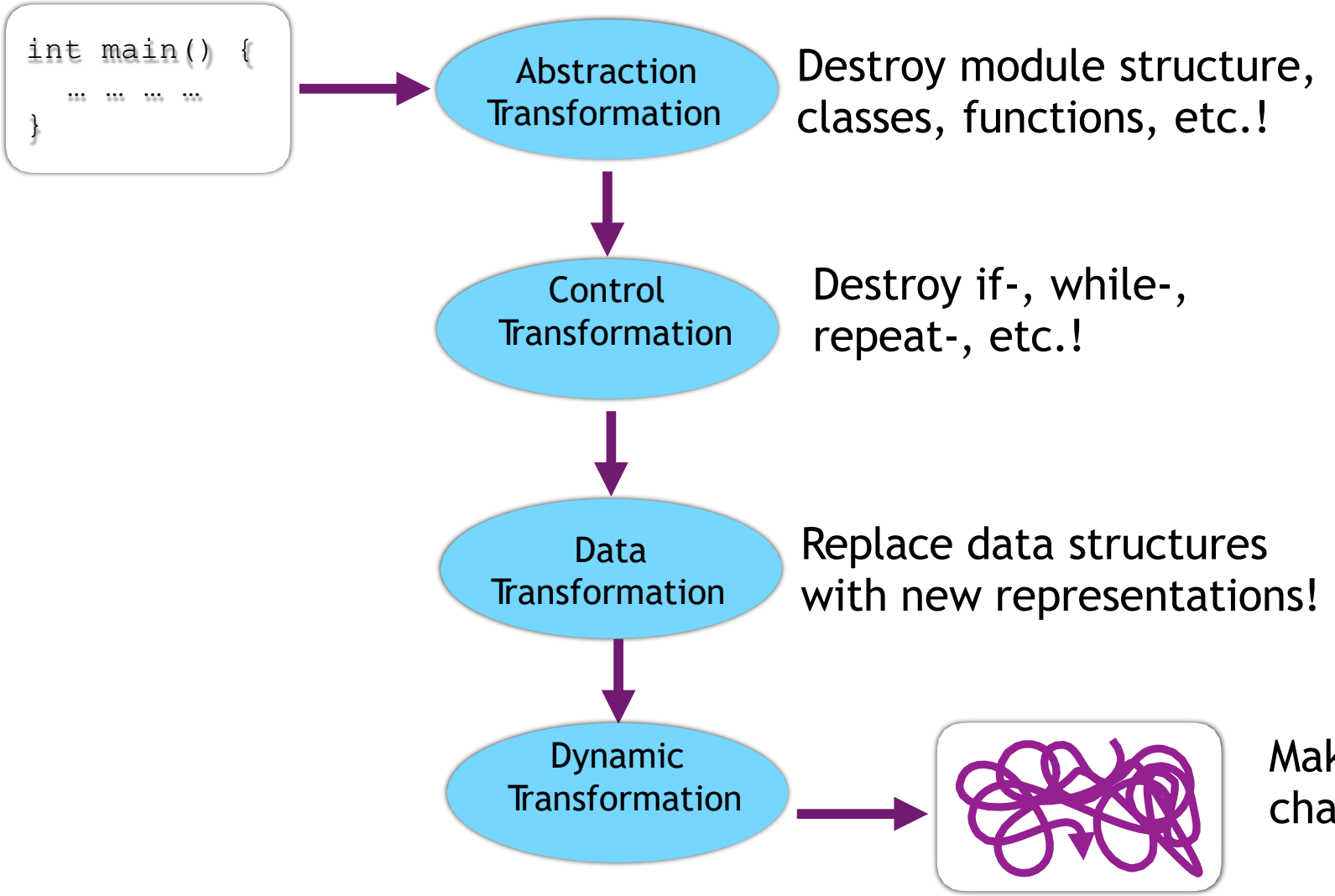


- **Obfuscated code**:

```
function z001c775808(
z3833986e2c) { var
z0d8bd8ba25=
"\x73\x6f\x6d\x65\x20\x73\x74\
x72\x69\x6e\x67"; var
z0ed9bcbcc2= (0x90b+785-
0xc04)* (0x1136+6437-0x1c4b);
document. write(
"\x76\x61\x72\x73\x20\x61\x72\
x65\x3a"+ z0d8bd8ba25+ "\x20"+
z0ed9bcbcc2+ "\x20"+
z3833986e2c);};
```

What is Obfuscation?

Ca' Foscari University of Venice



Example

Ca' Foscari University of Venice

```
int main() {
    int y = 6;
    y = foo(y);
    bar(y,42);
}
int foo(int x)
    return x*7;
}
void bar(int x, int z) {
    if (x==z)
        printf("%i\n",x);
}
```

Abstract Transformation

Ca' Foscari University of Venice

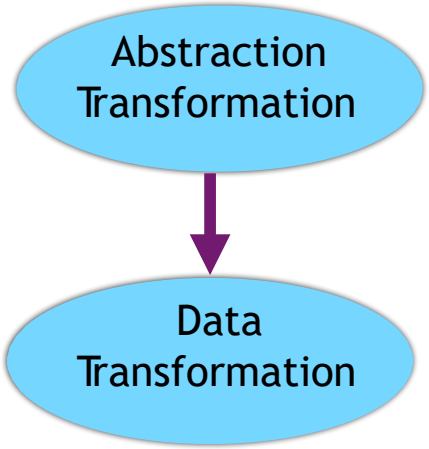
```
int main() {
    int y = 6;
    y = foo(y);
    bar(y,42);
}
int foo(int x)
    return x*7;
}
void bar(int x, int z) {
    if (x==z)
        printf("%i\n",x);
}
```


```
int main() {
    int y = 6;
    y = foobar(y,99,1);
    foobar(y,42,2);
}

int foobar(int x, int z, int s)
{
    if (s==1)
        return x*7; else
    if (s==2)
        if (x==z)
            printf("%i \n",x);
}
```

Data Transformation

Ca' Foscari University of Venice

```
int main() {
    int y = 6;
    y = foobar(y,99,1);
    foobar(y,42,2);
}

int foobar(int x, int z, int s) {
    if (s==1)
        return x*7; else if
    (s==2)
        if (x==z)
            printf("%i \n",x);
}
```

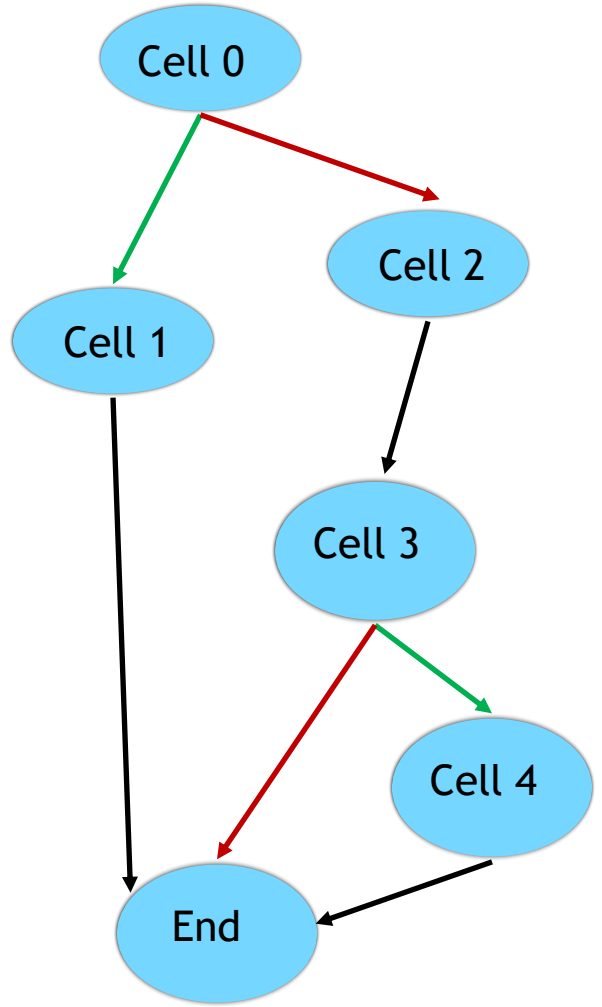



```
int main () {
    int y = 12;
    y = foobar(y,99,1);
    foobar(y,36,2);
}
int foobar(int x, int z, int s) {
    if (s==1)
        return (x*7)%51;
     else if (x==z) {
      int x2=x*x%51,x3=x2*x%51;
      int x4=x2*x2%51,x8=x4*x4%51;
      int x11=x8*x3%51;
      printf("%i\n",x11);
}
}
```

Control Flow Transformation

Ca' Foscari University of Venice

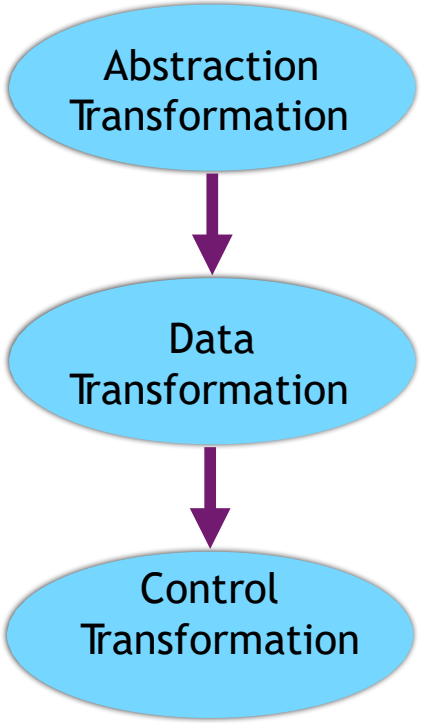
```
int main () {
     int y = 12;
     y = foobar(y,99,1);
     foobar(y,36,2);
}
int foobar(int x, int z, int s) {
        if (s==1) /cell 0
                return (x*7)%51; /cell 1
       else  /cell 2
        if (x==z) {    /cell 3
        int x2=x*x%51,x3=x2*x%51; /cell 4
        int x4=x2*x2%51,x8=x4*x4%51;
        int x11=x8*x3%51;
        printf("%i\n",x11);
        }
} /end
```



Control Flow Transformation

Ca' Foscari University of Venice

```
int foobar(int x, int z, int s) {
char* next= &&cell0; int retVal = 0;
cell0:  {next=(s==1)? &&cell1: &&cell2;
goto *next;}
cell1: {retVal=(x*7)%51; goto end;}
cell2: {next=(s==2)? &&cell3: &&end;
goto *next;}
cell3: {next=(x==z)? &&cell4: &&end;
goto *next;}
cell4: {
      int x2=x*x%51,x3=x2*x%51;
      int x4=x2*x2%51,x8=x4*x4%51;
      int x11=x8*x3 % 51;
      printf("%i \n",x11); goto end;
} end: return retVal;
}
```



Tigress

Tigress Obfuscator

Ca' Foscari University of Venice

- Open-Source obfuscator for C language from University of Arizona
- It works on linux, Darwin (Mac O.S.), Android
- It works with Intel, ARM, web assembly,
- Can deal with 32 or 64 instruction set,
- Can be compiled with gcc, clang, emcc

References

Tigress - https://tigress.wtf

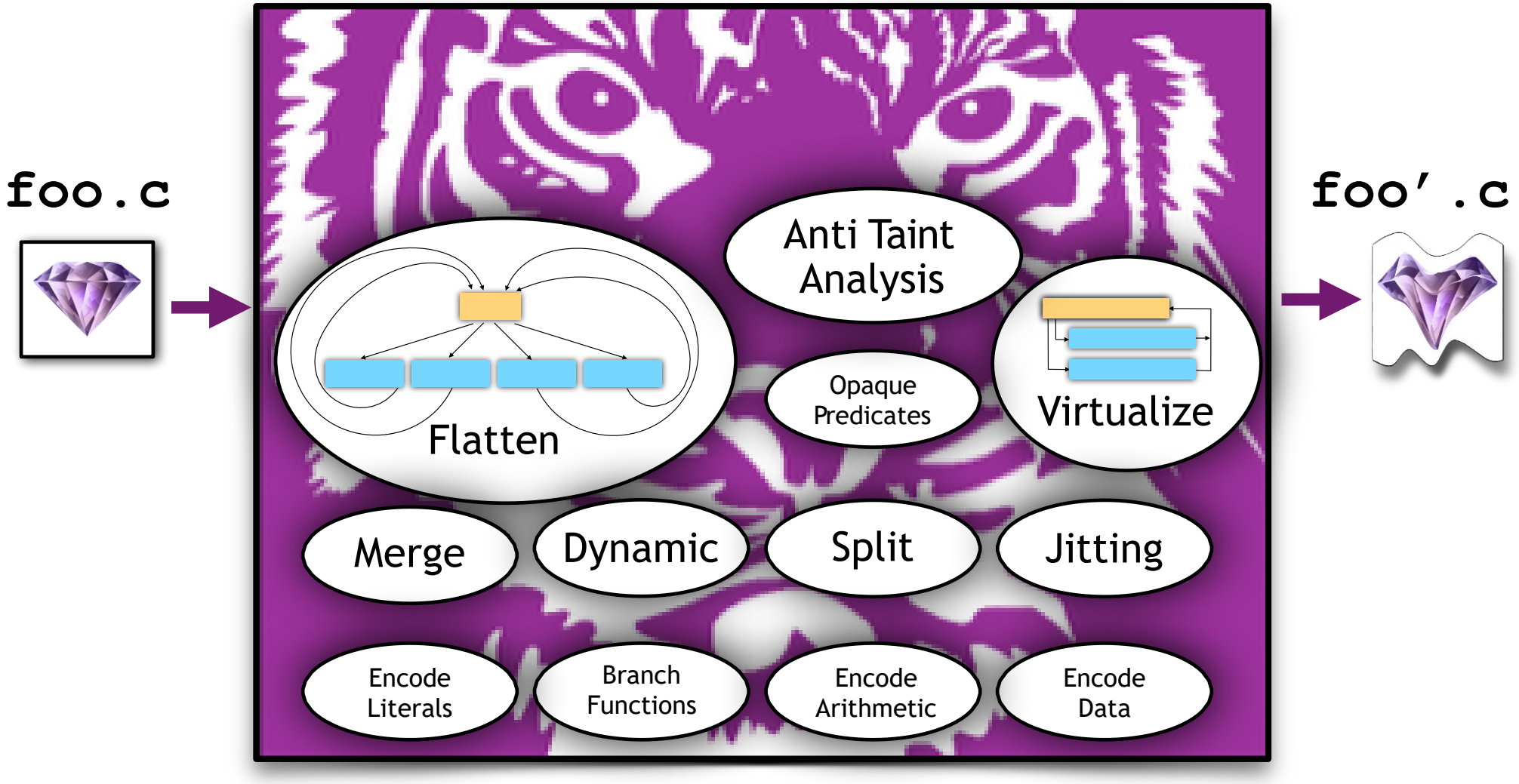
Clang- https://clang.llvm.org/

EMCC - https://emscripten.org/

Web Assembly - https://webassembly.org/

Tigress Obfuscations

Ca' Foscari University of Venice



Tigress obfuscation script

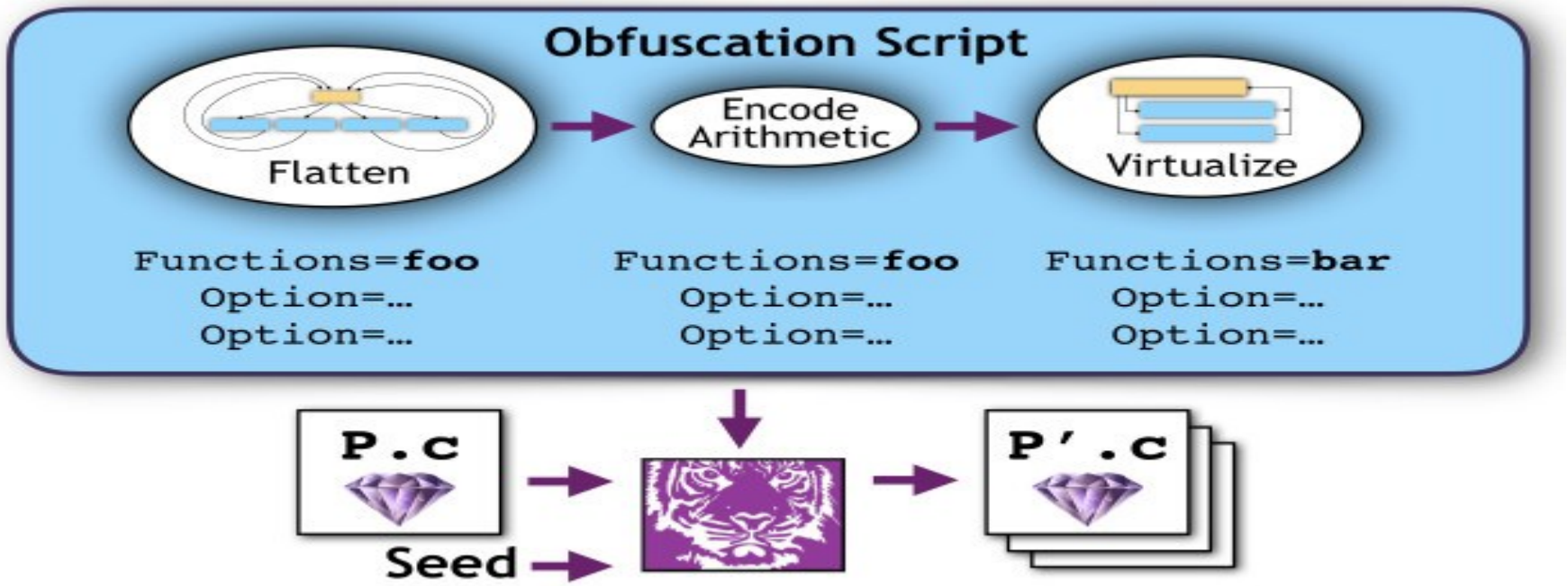
Ca' Foscari University of Venice

- Tigress is a diversifying virtualizer/obfuscator for the C language that supports many protections against both static and dynamic reverse engineering and de-virtualization attacks.
- Tigress is a source-to-source transformer - it takes a C source program as input and returns a new C program as output.

Obfuscation Script

Ca' Foscari University of Venice

- An *obfuscation script* (actually, a long sequence of command line options) describes the sequences of transformations that should be applied to the functions of the program:



Example of Tigress transformations

Ca' Foscari University of Venice

- Transform a function bar in a program foo.c using Tigress.
- The first transformation flattens the code, the second turns the flattened function into an interpreter:

```
tigress --Verbosity=1 --Environment=x86_64:Linux:Gcc:4.6 --Seed=42 \
 --Transform=Flatten \
     --Functions=bar \
     --FlattenDispatch=switch \       --FlattenObfuscateNext=false \
     --FlattenRandomizeBlocks=true \       --FlattenConditionalKinds=branch,compute,flag \
     --FlattenImplicitFlowNext=true \
  --Transform=Virtualize \
     --Functions=bar \      --VirtualizeDispatch=direct \
   --Transform=CleanUp \
      --CleanUpKinds=annotations,constants,names \
   --out=foo_out.c foo.c
```

Web Assembly

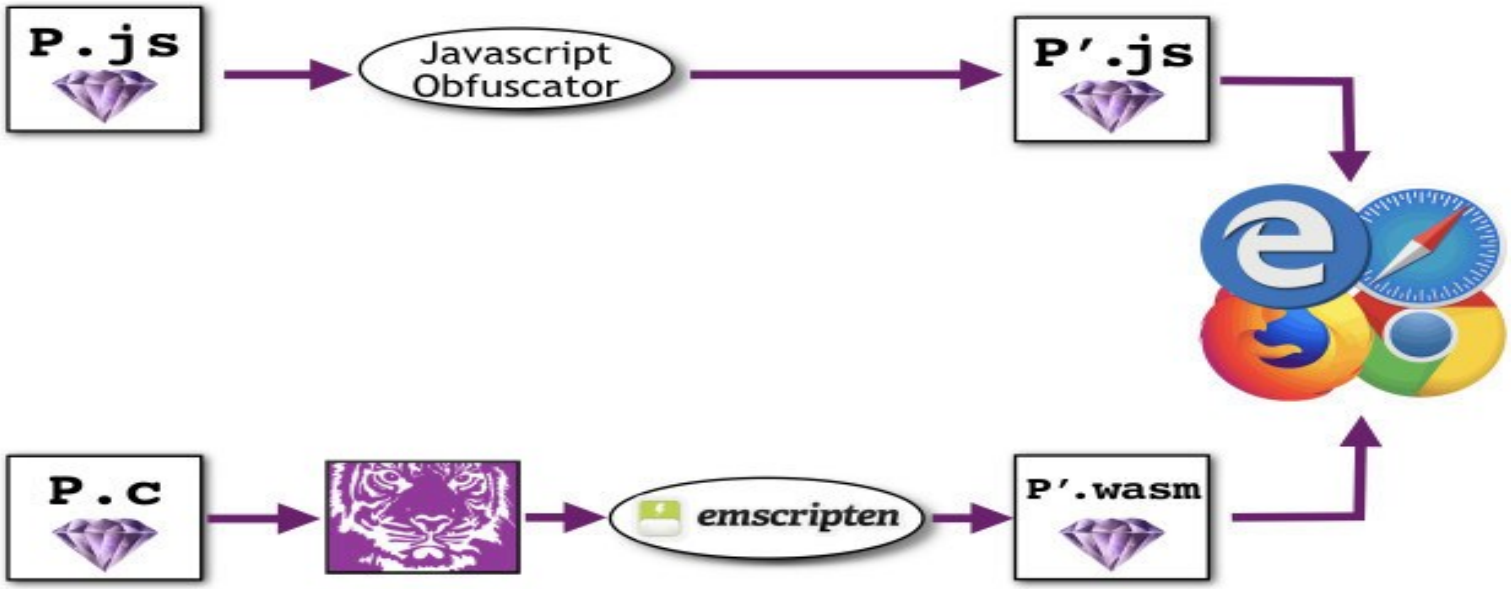
Ca' Foscari University of Venice

- The typical way to protect code running in the browser from attacks by the user is to write the code in Javascript and transform it with a Javascript obfuscator.
- An alternative approach is to write the code in C, transform it with Tigress, and to compile the resulting code using Emscripten (a C-to-WASM compiler) to a WebAssembly/html/Javascript package.

Web Assembly

Ca' Foscari University of Venice

- The WebAssembly/html/Javascript package can be loaded by most browsers.
- The advantage is that the resulting code can run at near-native speeds
- Tigress supports more powerful obfuscating transformations than most Javascript obfuscators.



Obfuscating Arithmetic

Encoding Integer Arithmetic

Ca' Foscari University of Venice

- $x+y = x- \neg y-1$
- $x+y = (x \oplus y) + 2\cdot(x \wedge y)$
- $x+y = (x \vee y) + (x \wedge y)$
- $x+y = 2\cdot(x \vee y) - (x \oplus y)$

Logic Operator	Math symbol	C language operator
OR	$\vee$	**¦**
XOR	$\oplus$	**^**
AND	$\wedge$	**&**
NOT	$\neg$	**!**

Example

Ca' Foscari University of Venice

- One possible encoding of

  z = x+y+w

- is

```
z = (((x ^ y) + ((x & y) << 1)) | w) +
      (((x ^ y) + ((x & y) << 1)) & w);
```

- Many others are possible, which is good for diversity.

Exercise

Ca' Foscari University of Venice

- The virtualizer's **add** instruction handler could still be identified by the fact that it uses a + operator!
- Try adding an arithmetic transformer:

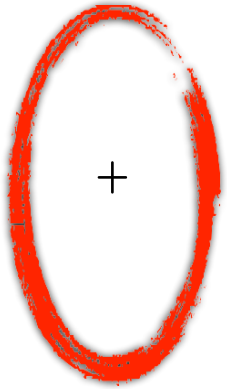
```
tigress --Environment=x86_64:Linux:Gcc:4.6\
   --Transform=Virtualize \
      --Functions=fib \
      --VirtualizeDispatch=switch\
   --Transform=EncodeArithmetic \
      --Functions=fib \
   --out=fib5.c fib.c
```

- What differences do you notice between before and after arithmetic encoding?

Virtualized Code

Ca' Foscari University of Venice

```
int fib(int n ) {
  ...
  while (1) {
    switch (*(_1_fib_$pc[0])) {
    case  PlusA:  {
    (_1_fib_$sp[0] + -1)->_int =
       (_1_fib_$sp[0] + -1)->_int
       (_1_fib_$sp[0] + 0)->_int;
   break;
    }
}
```



Virtualization and Obfuscating Arithmetic

Ca' Foscari University of Venice

```
int fib(int n ) {
   ...
   while (1) {
     switch (*(_1_fib_$pc[0])) {
     case  PlusA:  {
     (_1_fib_$sp[0] + -1)->_int =
        ((_1_fib_$sp[0] + -1)->_int      ^
        (_1_fib_$sp[0] + 0)->_int)       +
        (((_1_fib_$sp[0] + -1)->_int     &
        (_1_fib_$sp[0] + 0)->_int) << 1);
     break;
     }
}
```

Virtualization and Obfuscating Arithmetic

Ca' Foscari University of Venice

$$x+y = (x \oplus y)+2\cdot(x \wedge y)$$

```
int fib(int        n ) {
   ...
   while (1) {
     switch (*(_1_fib_$pc[0])) {
     case  PlusA:  {
     (_1_fib_$sp[0] + -1)->_int =
        ((_1_fib_$sp[0] + -1)->_int      ^
        (_1_fib_$sp[0] + 0)->_int)       +
        (((_1_fib_$sp[0] + -1)->_int     &
        (_1_fib_$sp[0] + 0)->_int) << 1);
     break;
     }
}
```

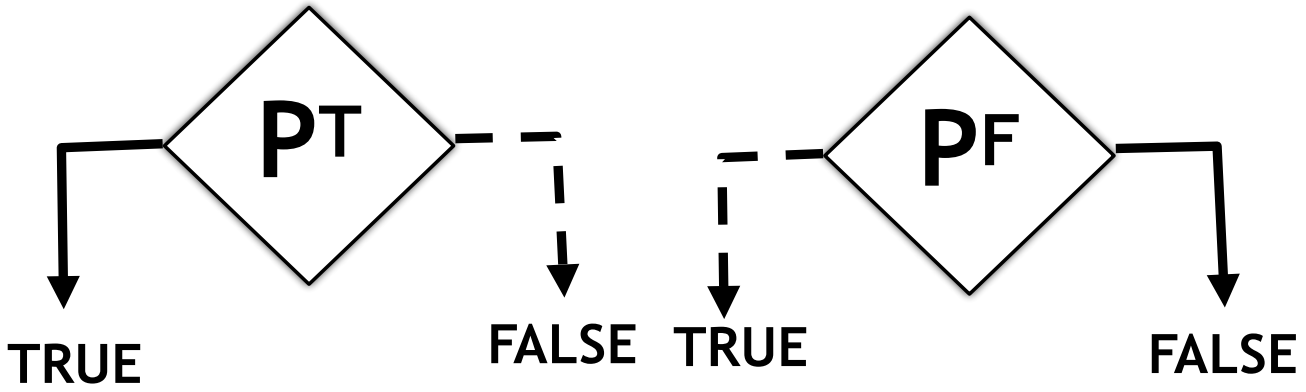
Opaque Expressions

Opaque Expressions

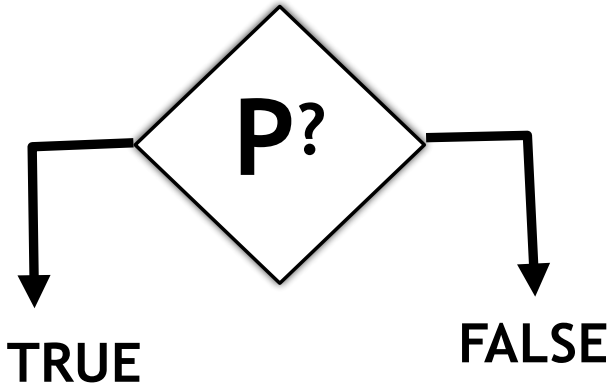
Ca' Foscari University of Venice

- An expression whose value is known to you as the defender (at obfuscation time) but which is difficult for an attacker to figure out

Opaque Predicates

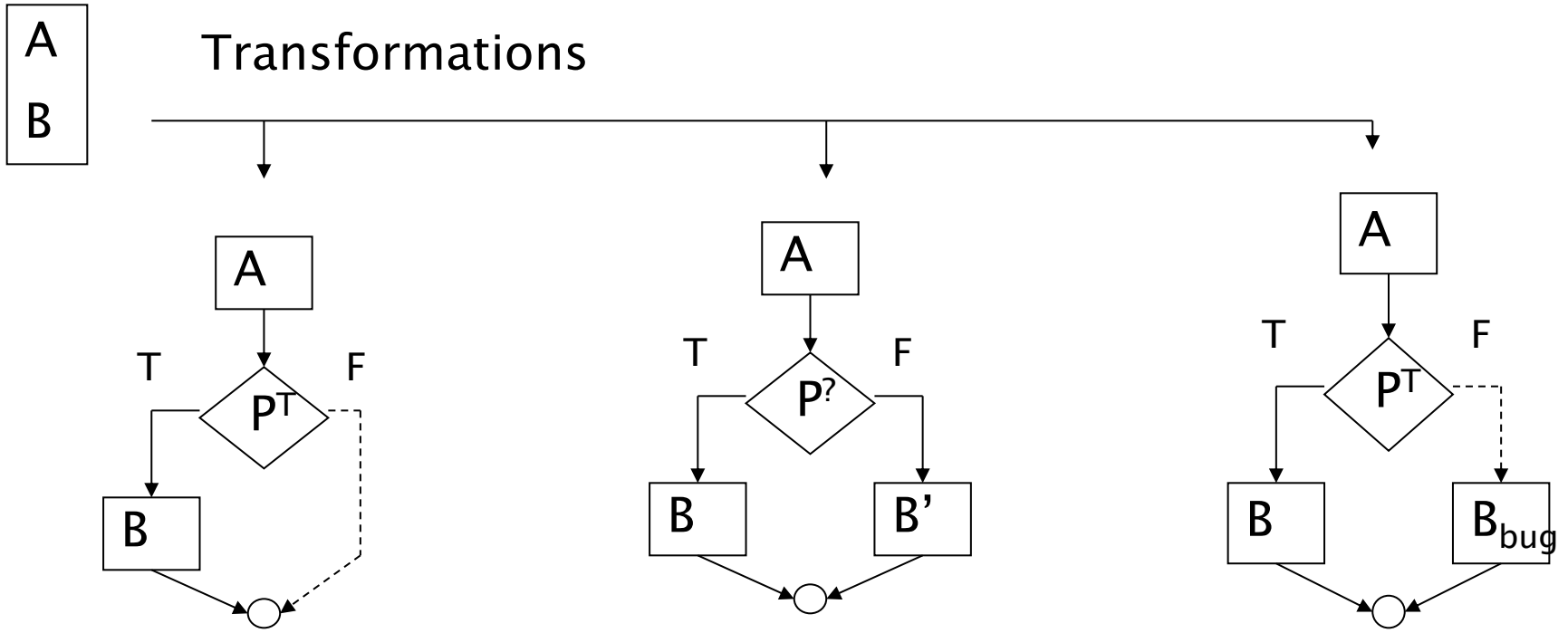


Opaquely true/ false predicate



Opaquely indeterminate predicate

Opaque Predicates Strategies



P^T: Predicate always TRUE

$P^?$: Predicate can be TRUE or FALSE

B and B' have same effect

P^T: Predicate always TRUE

B_{bug}: A buggy version of B

Inserting Bogus Control Flow

Ca' Foscari University of Venice

```
if (x[k] == 1)
    R = (s*y) % n
else
    R = s;
s = R*R % n;
L = R;
```

```
if (x[k] == E=1)
    R = (s*y) % n
else
    R = s;
s = R*R % n;
L = R;
```

Inserting Bogus Control Flow

Ca' Foscari University of Venice

```
if (x[k] == 1)
    R = (s*y) % n
else
    R = s;
s = R*R % n;
L = R;
```

```
if (x[k] == 1)
    R = (s*y) % n
else
    R = s;
if (expr=T)
    s = R*R % n;
else
    s = R*R * n;
L = R;
```

Inserting Bogus Control Flow

Ca' Foscari University of Venice

```
if (x[k] == 1)
    R = (s*y) % n
else
    R = s;
s = R*R % n;
L = R;
```

```
if (x[k] == 1)
    R = (s*y) % n
else
    R = s;
if (expr=?)
    s = R*R % n;
else
    s = (R%n)*(R%n)%n;
L = R;
```

Opaque Predicates

- Predicate is opaque if its outcome is known at obfuscation time, but difficult to deduce later
- Implementation of opaque predicates uses pointer-based structures
  - Static code analysis is harder
  - But Dynamic Analysis (debugging) can identify the actual running paths and the ones which are never executed!

Exercise

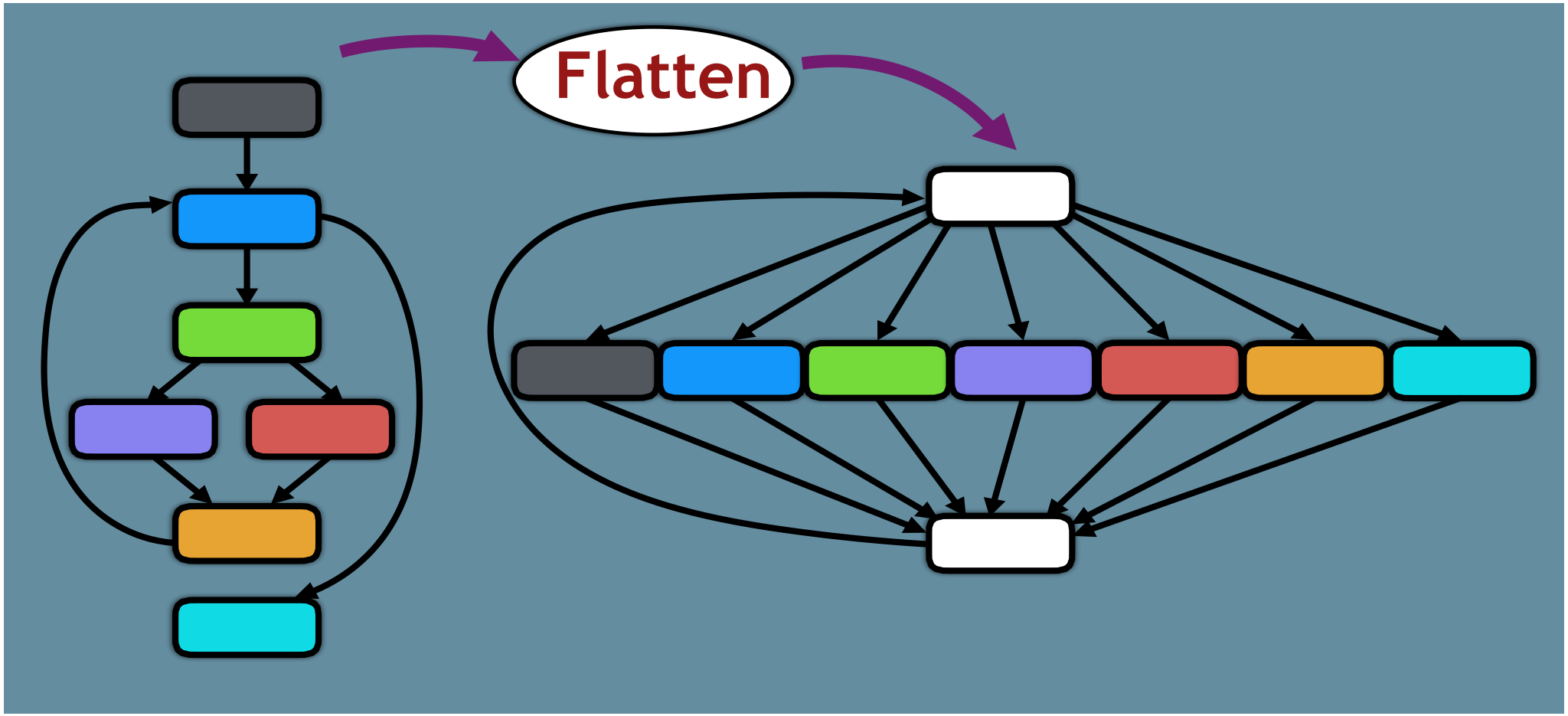
Ca' Foscari University of Venice

```
*************************************************************
* 1) Opaque Predicates
*************************************************************
tigress --Environment=x86_64:Linux:Gcc:4.6 --Seed=0 \
   --Transform=InitEntropy \
      --InitEntropyKinds=vars \
   --Transform=InitOpaque \
      --Functions=main\
      --InitOpaqueCount=2\
      --InitOpaqueStructs=list,array \
   --Transform=AddOpaque\
      --Functions=fib\
      --AddOpaqueKinds=question \
      --AddOpaqueCount=10 \
    --out=fib1.c fib.c
```

Control Flow Flattening

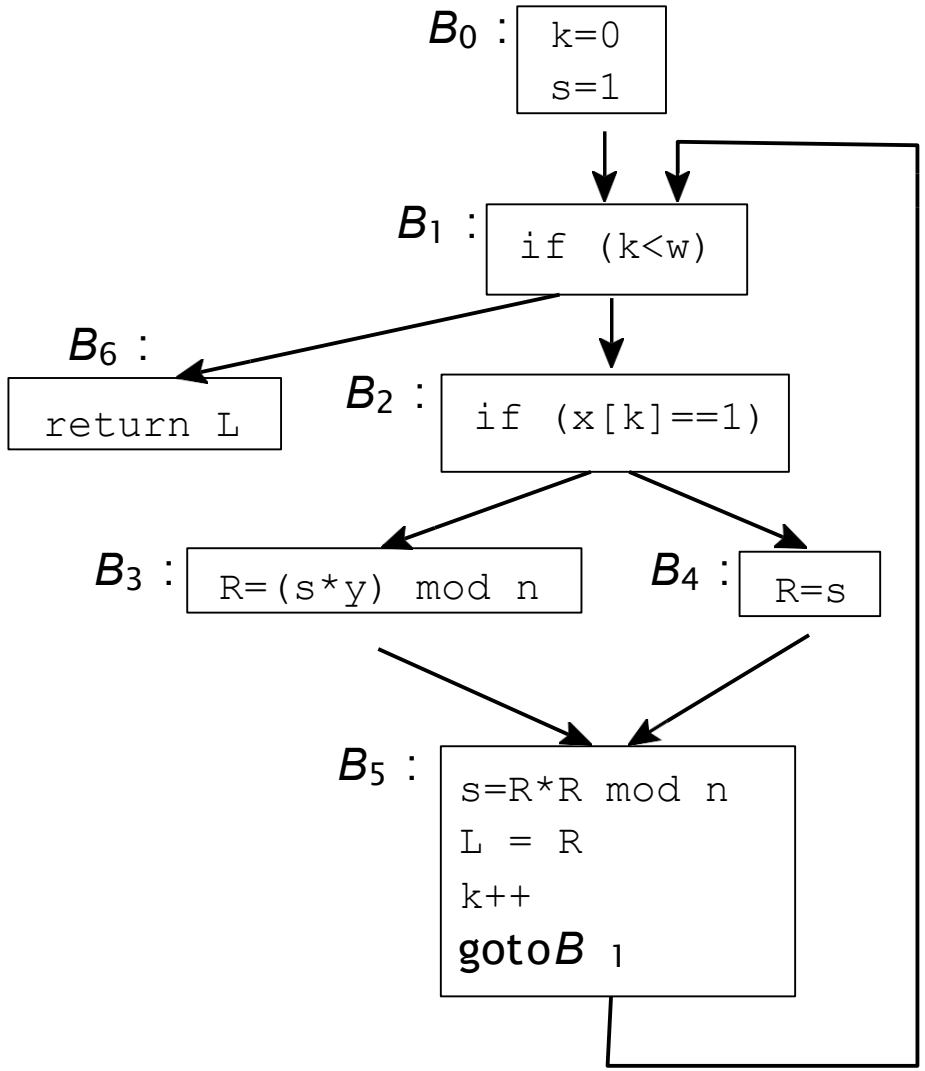
Control Flow Flattening

Ca' Foscari University of Venice



Example

```
int modexp(
    int y,int x[],
    int w,int n){
    int R, L;
    int k=0; int s=0;
    while (k < w) {
        if (x[k] == 1)
            R = (s*y) % n
        else
            R = s;
        s = R*R % n;
        L = R;
        k++;
    }
    return L;
}
```



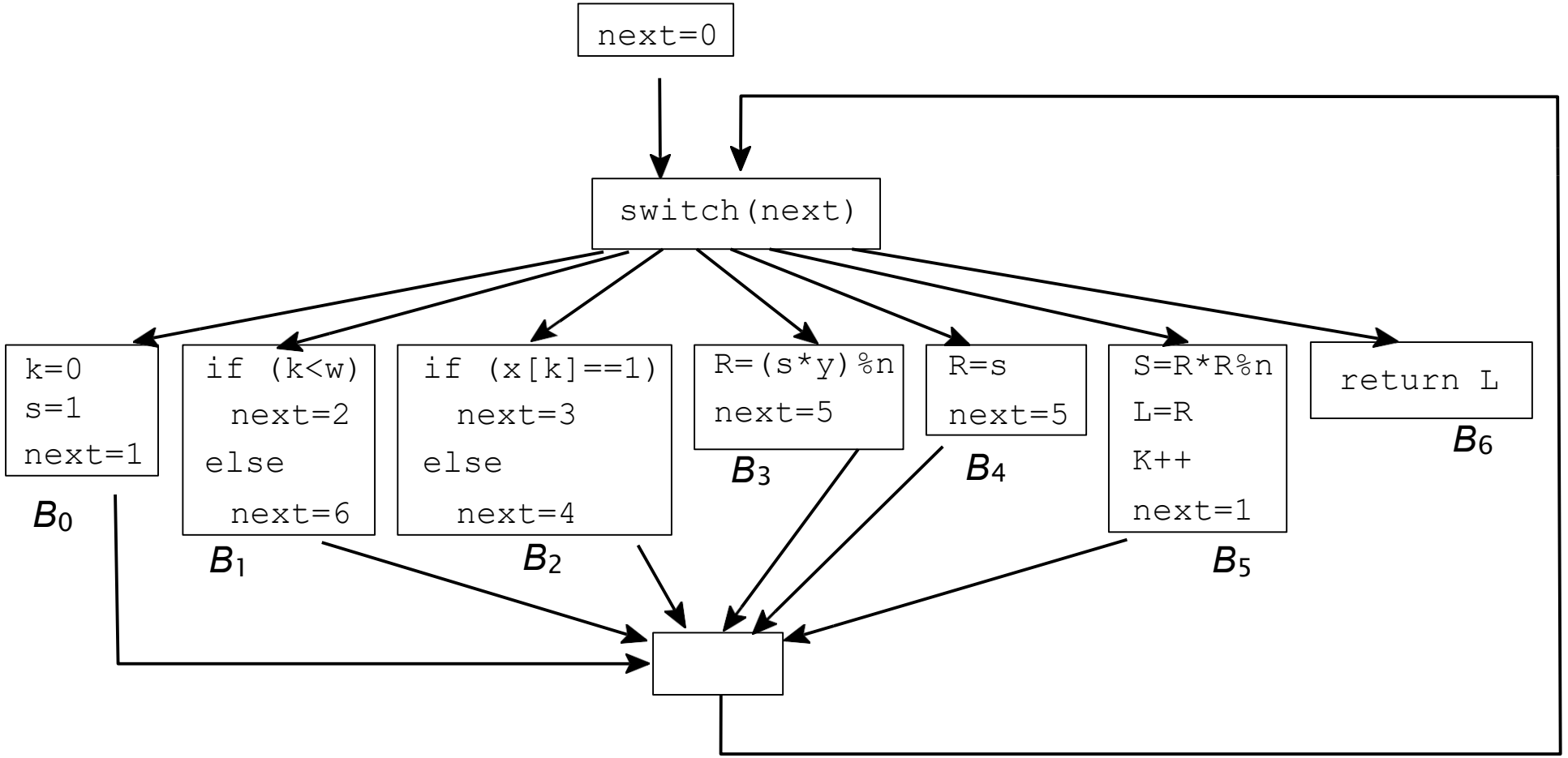
Example flattened

Ca' Foscari University of Venice

```
int modexp(int y, int x[], int w, int n) {
int R, L, k, s;
    int next=0;
    for(;;)
        switch(next) {
            case 0 : k=0; s=1; next=1; break;
            case 1 : if (k<w) next=2; else next=6; break;
            case 2 : if (x[k]==1) next=3; else next=4; break;
            case 3 : R=(s*y)%n; next=5; break;
            case 4 : R=s; next=5; break;
            case 5 : s=R*R%n; L=R; k++; next=1; break;
            case 6 : return L;
    }
}
```

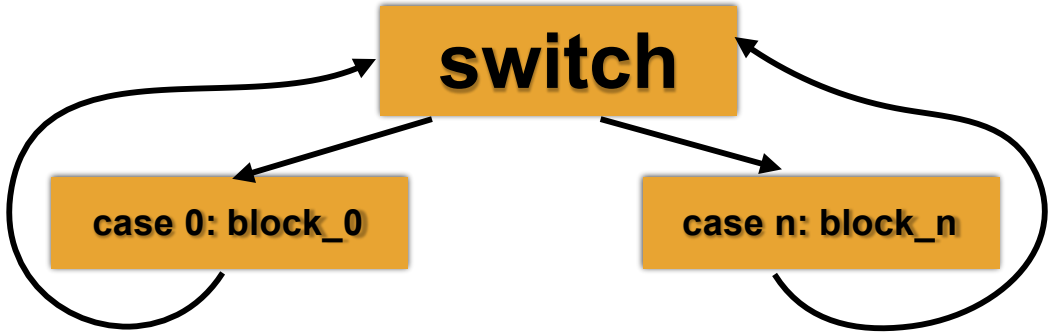
Flattened Control Flow Graph

Ca' Foscari University of Venice



Flattening Algorithm

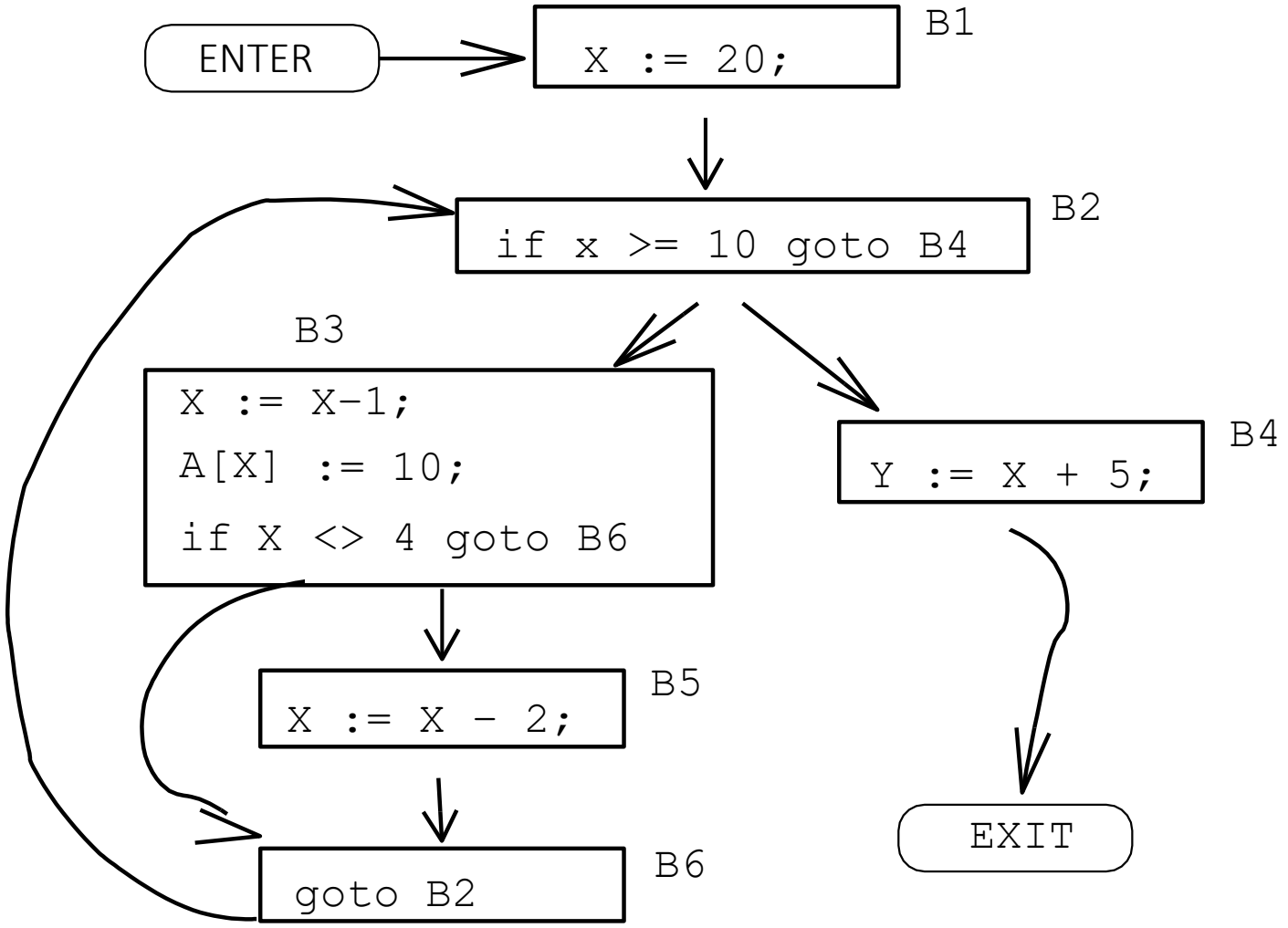
1. Construct the CFG
2. Add a new variable **int next=0;**
3. Create a switch inside an infinite loop, where every basic block is a case:



4. Add code to update the **next** variable

```
case n: {
  if (expression)
    next = …
  else
    next = …
}
```

Flatten this CFG! Work with your friends!



Virtualization Obfuscation

Virtualize: Turn Code Into Data

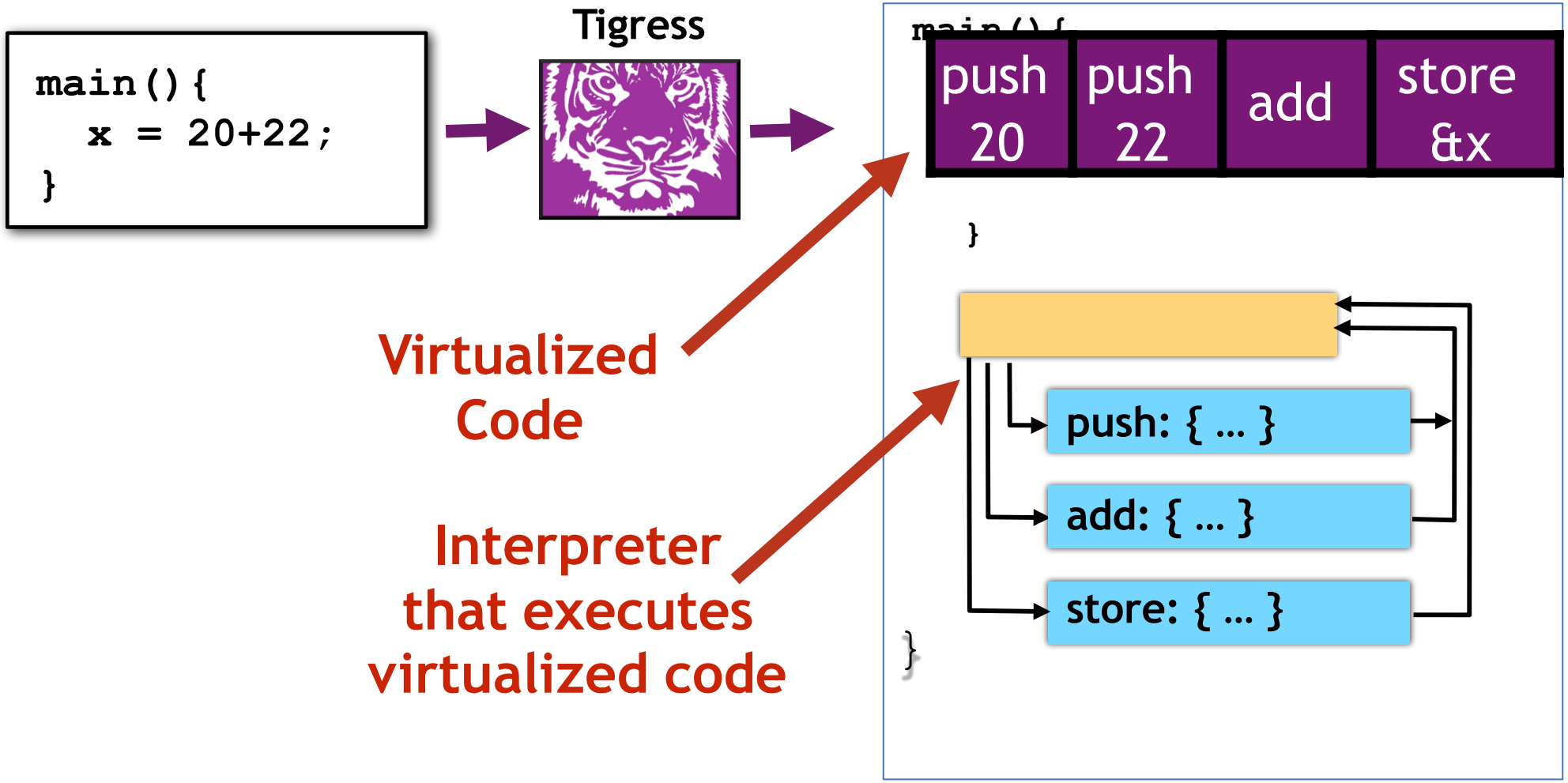
Ca' Foscari University of Venice

```
main(){
   x = 20+22;
}
```


```
main(){

   ??????????

   }
```

Virtualize: Turn Code Into Data

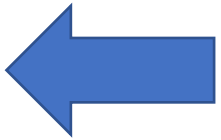
Ca' Foscari University of Venice



Virtualize: Turn Code Into Data

Ca' Foscari University of Venice

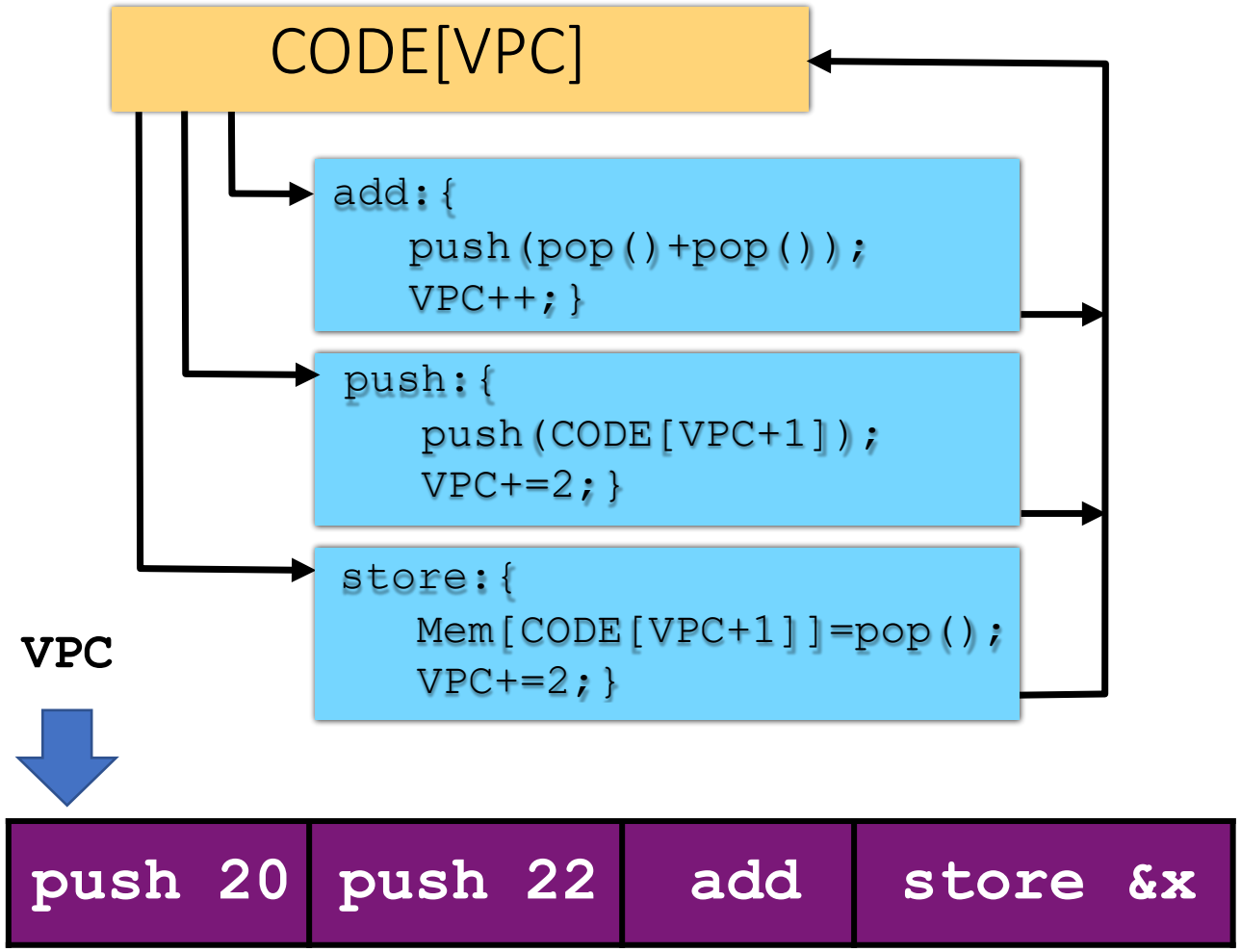
- Example: Translate code into high-level bytecode
- Bytecode has no registries, it only works with stack-based operations
- Create a virtual machine that implements such bytecode into platform-specific assembly
- Virtual Program Counter (VPC) iterates over the bytecode and execute it
- Stack Pointer (SP) points to the top of the Stack



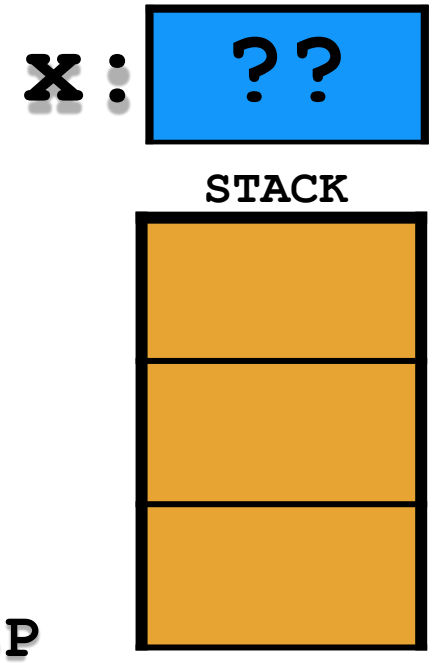
```
main()  {
    int x;
    x = 20 + 22;
}
```

Virtualize: Turn Code Into Data

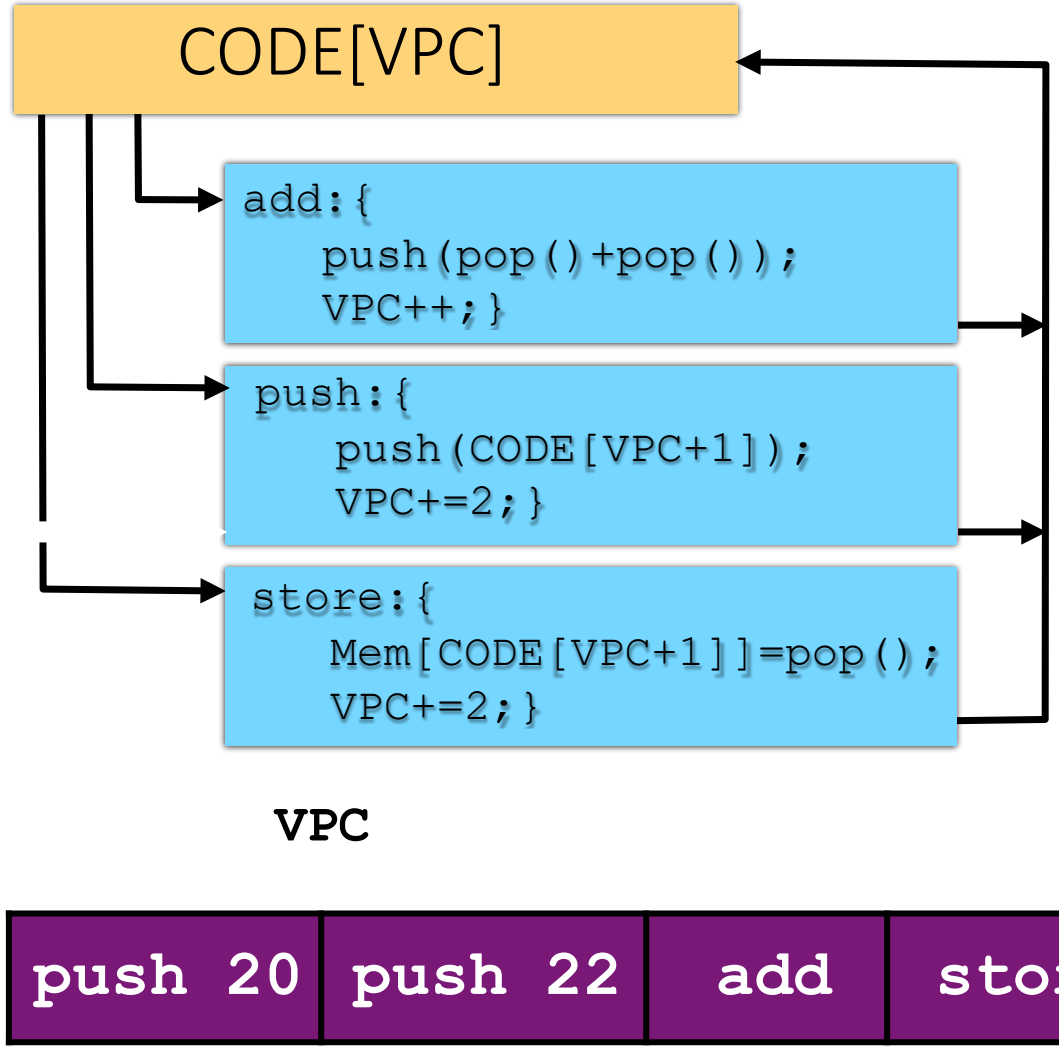
Ca' Foscari University of Venice



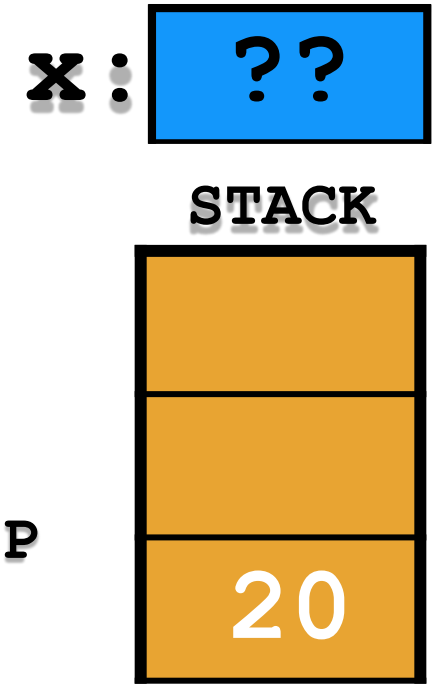

```
main() {
    int x;
    x = 20 + 22;
}
```



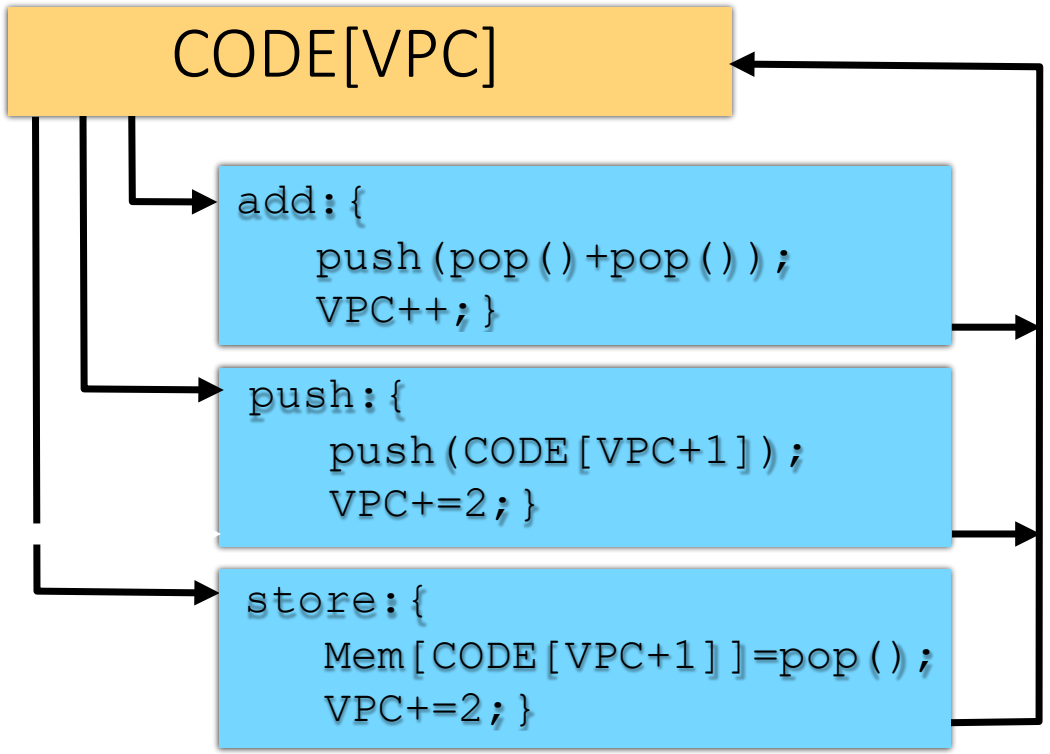
Example: Execute “Push 20”



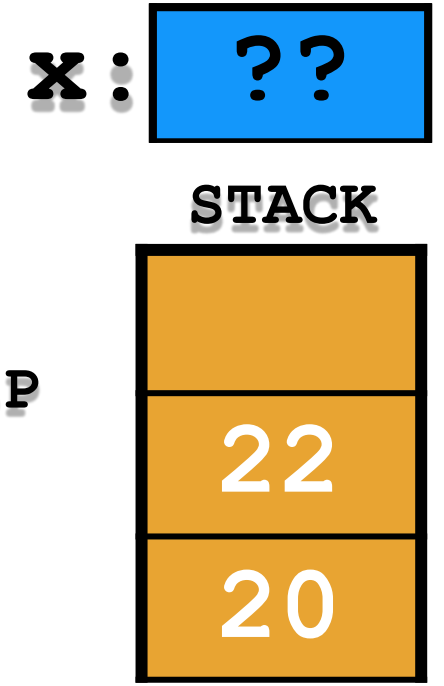

```
main() {
    int x;
    x = 20 + 22;
}
```



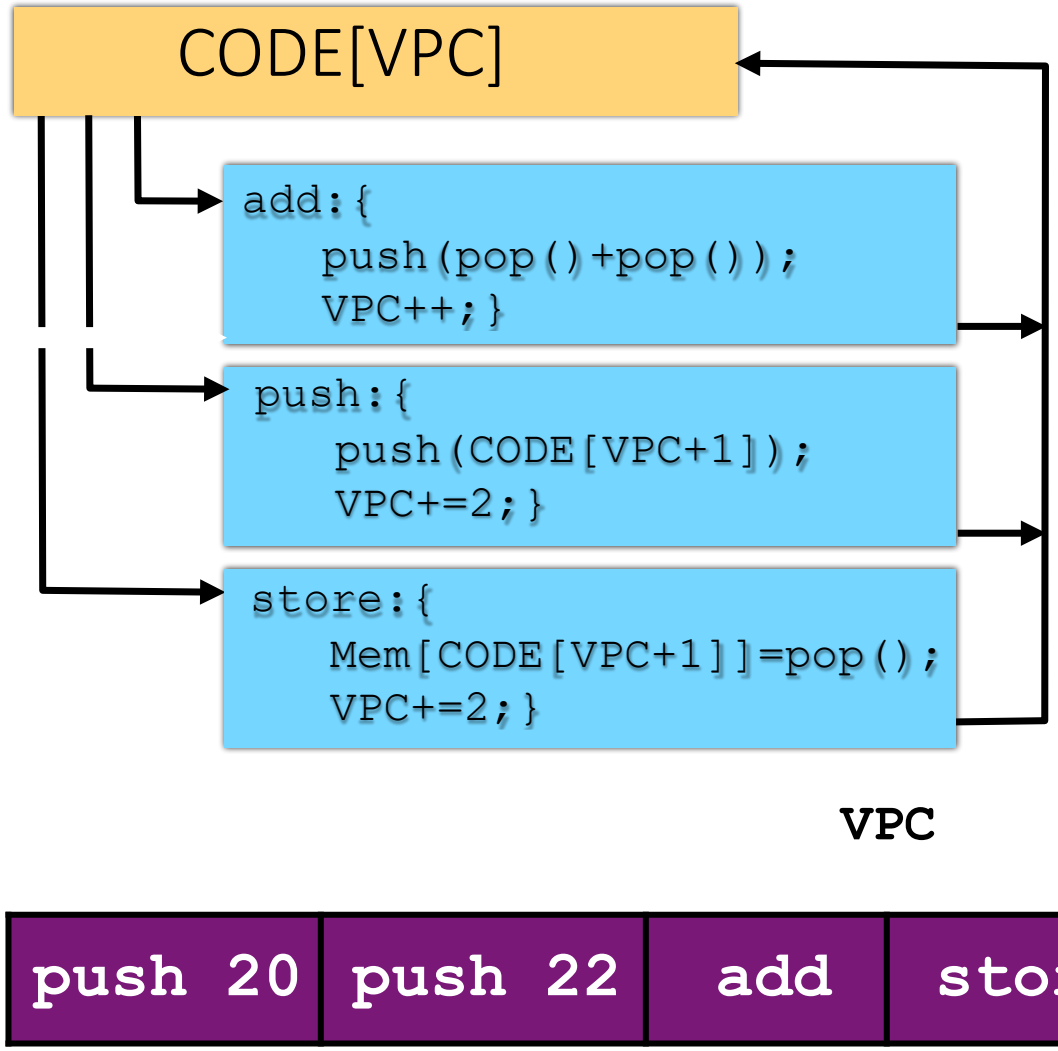
Example: Execute “Push 22”



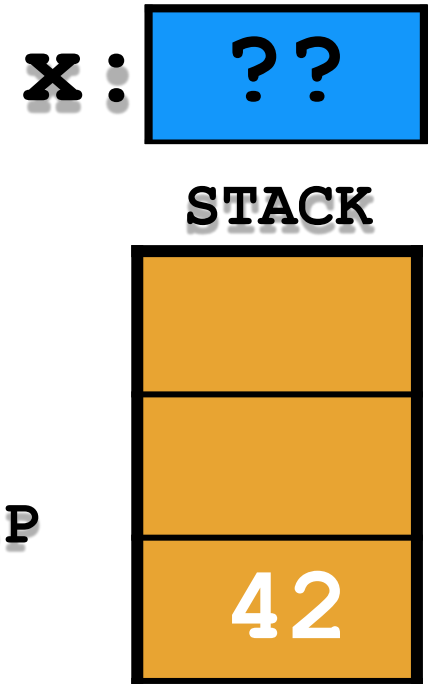

```
main() {
    int x;
    x = 20 + 22;
}
```



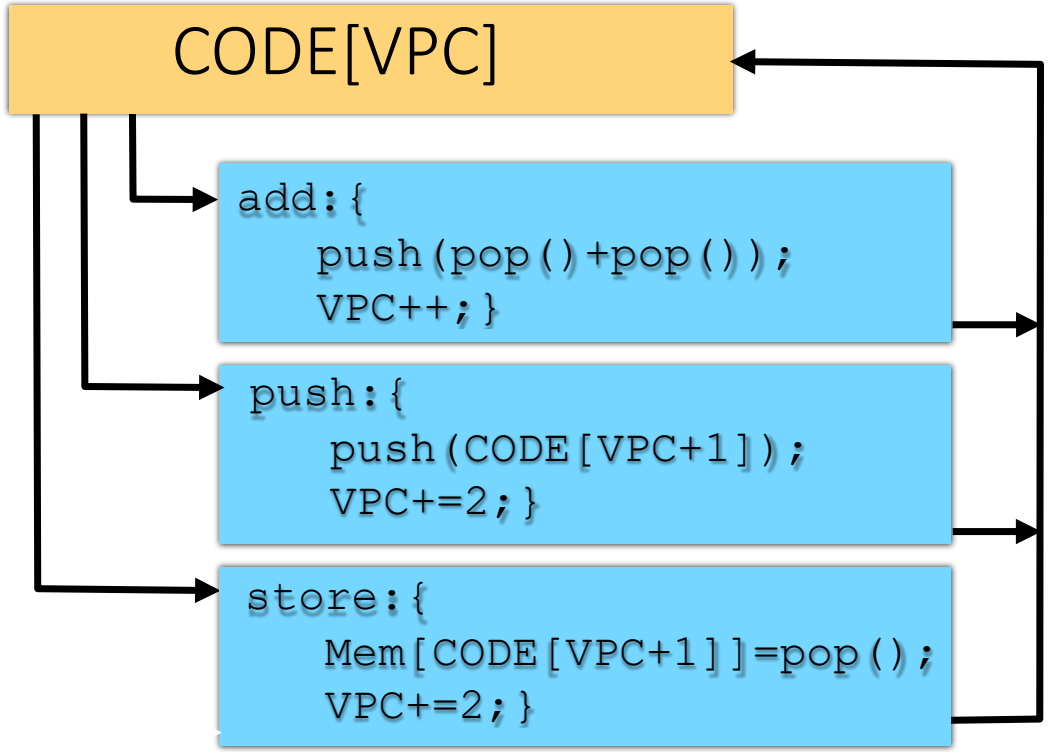
Example: add two numbers




```
main() {
    int x;
    x = 20 + 22;
}
```



Example: Store the result in memory




```
main() {
    int x;
    x = 20 + 22;
}
```

x: 42

STACK

VPC

SP

push 20	push 22	add	store &x

Liger VM

Ca' Foscari University of Venice

Here is a virtual machine that contains all the software tools you need for this course, developed by University of Arizona

- https://ligerlabs.org/software.html

- Download it this week, and follow the instructions
- Try Tigress obfuscator examples

Tigress Installation

Ca' Foscari University of Venice

- In case you want to install Tigress only:
  1. Make sure you have the following installed:
     - gcc (or a drop-in replacement such as clang)
     - perl
     - Bash
  2. Download Tigress from https://tigress.wtf/
  3. Unzip the zip file
  4. Set this environment variable:
     - export TIGRESS_HOME=PATH-TO-THIS-DIRECTORY/tigress/3.1
  5. Add this directory to your PATH:
     - export PATH= …:PATH-TO-THIS-DIRECTORY/tigress/3.1:…

Tigress Installation

Ca' Foscari University of Venice

Here's a simple test program you can use to try out Tigress:
/bin/cp PATH-TO-THIS-DIRECTORY/tigress/3.1/test1.c .

Here's the Tigress command:

```
tigress --Environment=ENV \
--Transform=Virtualize \
--Functions=main,fib,fac \
--out=result.c test1.c
```

ENV depends on your platform; it should be one of
x86_64:Linux:Gcc:4.6 x86_64:Darwin:Clang:5.1 armv7:Linux:Gcc:4.6 armv8:Linux:Gcc:4.6

This should give you an obfuscated program in result.c.

Exercises with Tigress

Ca' Foscari University of Venice

- Download these two files from our Moodle page or from here:
  - http://tigress.cs.arizona.edu/exercises.txt
  - http://tigress.cs.arizona.edu/fib.c
- The file **exercises.txt** contains Tigress commands I want you to run.
- The commands can be long and I don't want you to have to type them yourselves.
- The file **fib.c** contains the C file I want you to work on.

Example in exercises.txt

Ca' Foscari University of Venice

```
************************************************************
* 1) Opaque Predicates
************************************************************
tigress --Environment=x86_64:Linux:Gcc:4.6  --Seed=0 \
   --Transform=InitEntropy \
      --InitEntropyKinds=vars \
   --Transform=InitOpaque \
      --Functions=main\
      --InitOpaqueCount=2\
      --InitOpaqueStructs=list,array \
    --Transform=AddOpaque\
      --Functions=fib\
      --AddOpaqueKinds=question \
      --AddOpaqueCount=10 \
     --out=fib1.c fib.c
```