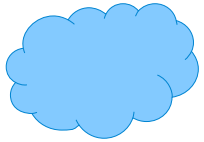
Stack machines unchained: code emulation with *ESIL*

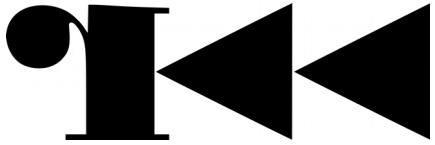
Arnau Gàmez i Montolio | *@arnaugamez*

April 27, 2019
Overdrive Conference - Girona

Who am I

- Student - *Maths* & *CS* @ UB
- President - *@HackingLliure*
- Collaborator - *#r2con*





Outline

1. Emulation
2. Intermediate languages & ESIL
3. ESIL operation
4. Demos

What is emulation?

- Simulate the execution of code of the **same or different CPU**

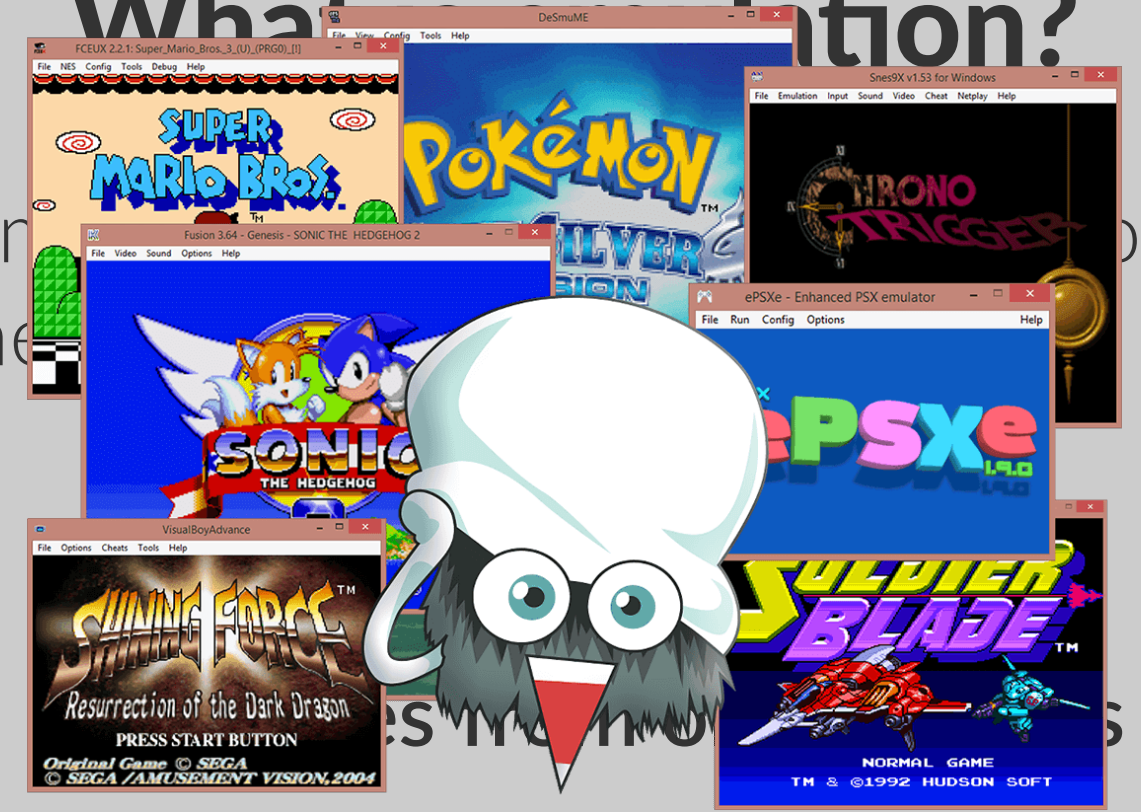
What is emulation?

- Simulate the execution of code of the **same or different CPU**

Run **games from old consoles**

What is emulation?

- Sim… of the…



Why emulation?

- **Understand** specific snippet of code
- **Avoid risks** of native code execution
- Help **debugging** and **code analysis**
- Explore **non-native executables**

Outline

1. Emulation
2. **Intermediate languages & ESIL**
3. ESIL operation
4. Demos

Intermediate languages

*"Language of an **abstract machine** designed to aid in the analysis of computer programs" -- wikipedia*

Vital for (de)compilation

What is ESIL?

- **E**valuable **S**trings **I**ntermediate **L**anguage
- Small set of instructions
- Based on reverse polish notation (stack)
- Designed with **emulation and evaluation in mind**, not human-friendly reading

What is ESIL?

- Infinite memory and set of registers
- “Native” register aliases
- Ability to implement **custom ops** and call external functions

Why ESIL?

- Need for emulation on r2land
- Easy to generate, parse and modify
- Extensibility
- Why not?

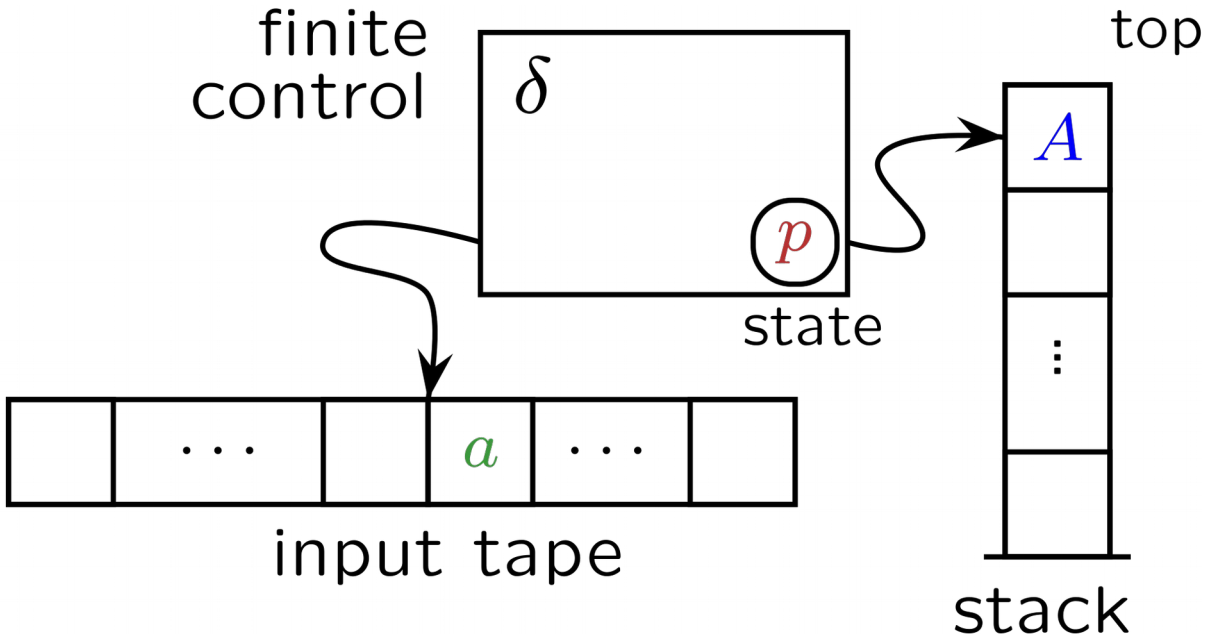
Outline

1. Emulation
2. Intermediate languages & ESIL
3. **ESIL operation**
4. Demos

ESIL

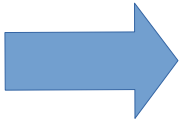
Stack machine on steroids

Stack machines / PDA's

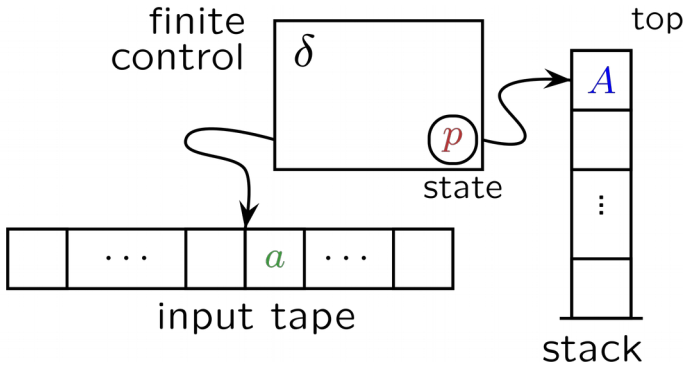


Stack machines / PDA's

- input symbol
- current state
- stack symbol



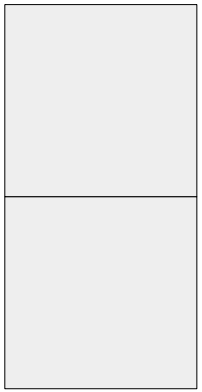
- state transition
- manipulate stack (push/pop)



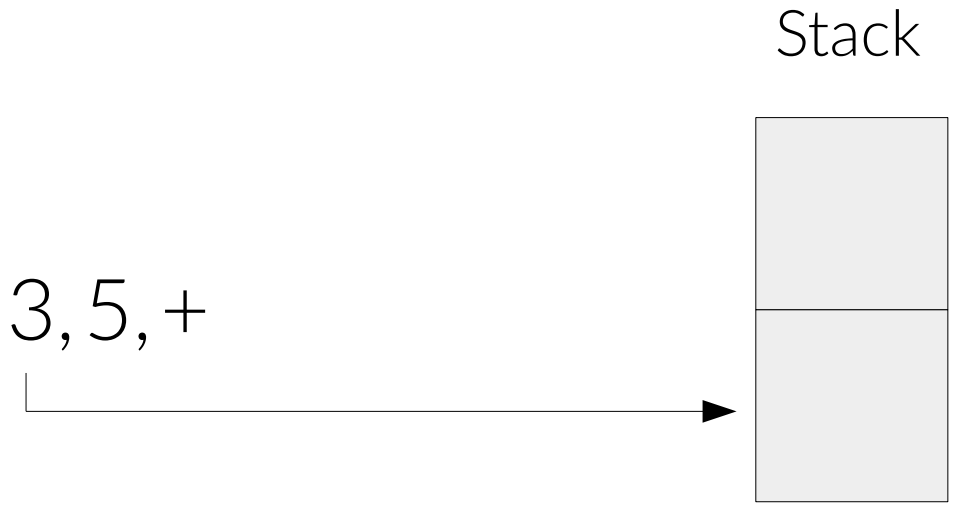
Visual animation

3, 5, +

Stack



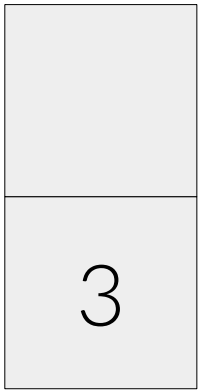
Visual animation



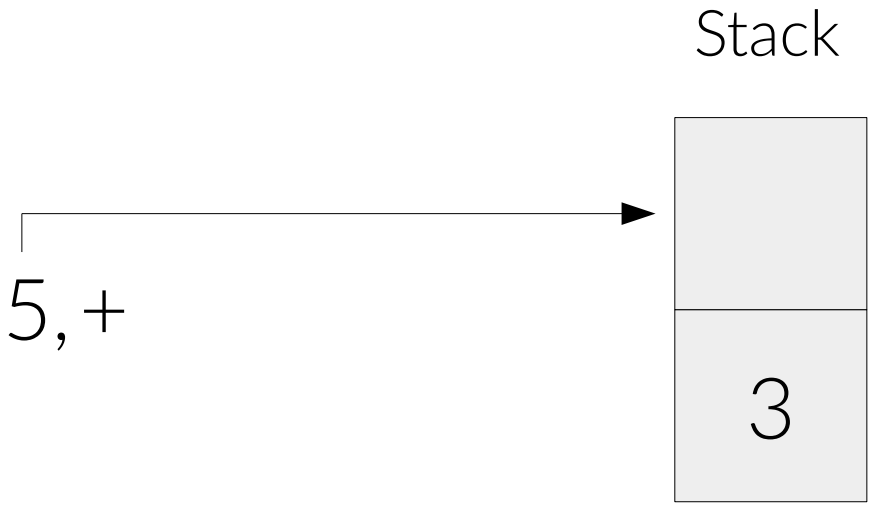
Visual animation

5, +

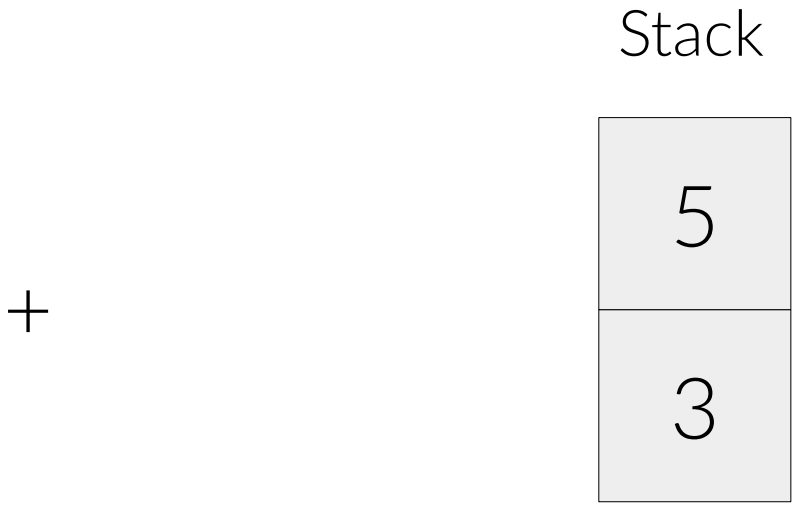
Stack



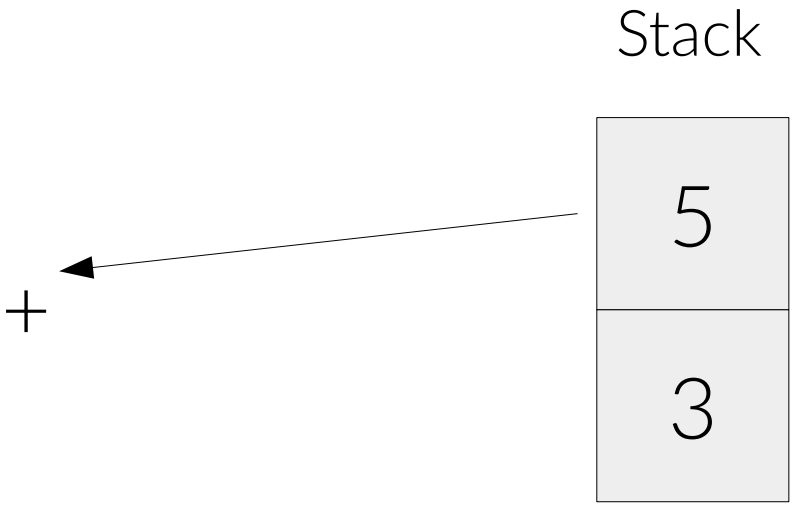
Visual animation



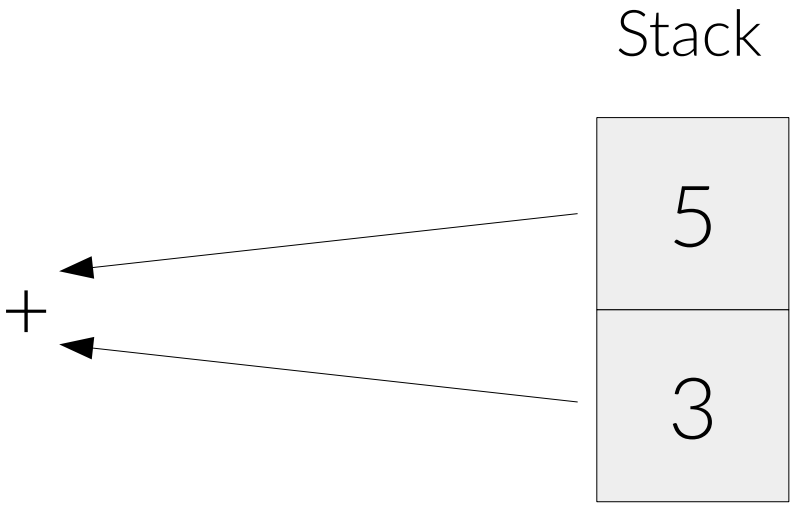
Visual animation



Visual animation

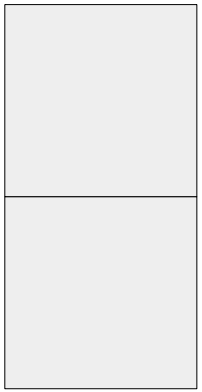


Visual animation



Visual animation

Stack



Digression

radare2 live crash crouse

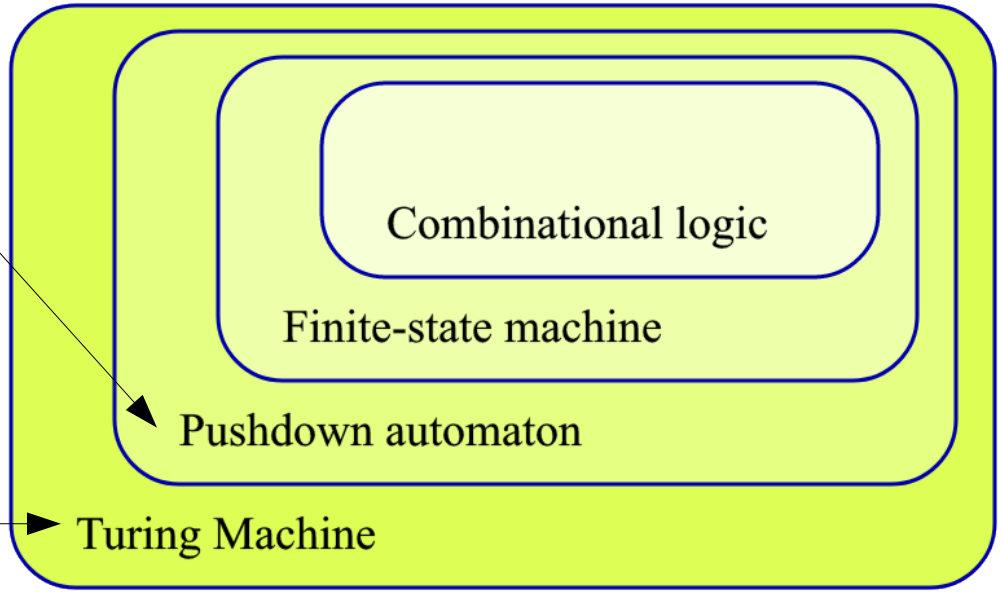
Example

ae 3,5,+

Expanding stack machines

We are here

We want to be here



cc @condr3t

HOW?

HOW?

STEROIDS

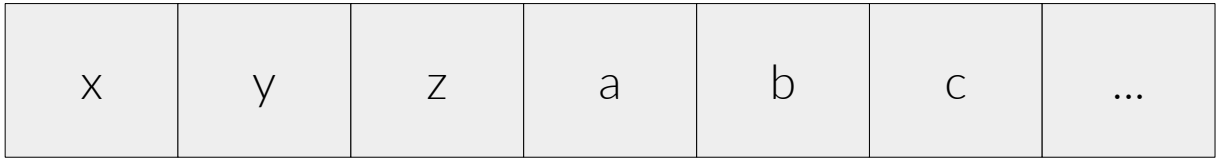
(aka cheating)

Steroids x1

- Add **random access** operations
- Add **control flow** operations

Steroids x2

- **Register** access
- Add "**extra tape**" with random access (virtual memory, VM stack)



Basic practical usage

ESIL options are under **ae** (**a**nalysis **e**sil) subcommands

- ae***i*** – ***i***nit
- ae***im*** – ***i***nit ***m***emory
- ae***ip*** – ***i***nst. ***p***ointer
- ae***s*** – ***s***tep
- ae***su*** – ***s***tep ***u***ntil
- ae***so*** – ***s***tep ***o***ver
- ae***ss*** – ***s***tep ***s***kip
- ae***r*** – ***r***egisters

ESIL operands

Check ***ae??*** on a radare2 shell

(description and examples)

ESIL internal vars (flags)

Prefixed with **$** | read-only

- $z – zero flag
- $cx – carry flag from bit x
- ...

Updated on each operation. Used to set flags for particular arch.

Outline

1. Emulation
2. Intermediate languages & ESIL
3. ESIL operation
4. **Demos**

Demo

Defeat simple crackme

cc @pof @jvoisin

Demo

Deobfuscate encrypted code

cc @superponible

Thanks

- Pancake (*@trufae*)
- Xvilka (*@akochkov*)
- Condret (*@condr3t*)
- Skuater (*@sanguinawer*)