

M6 - Cross-Optimization Testing

Guillermo Polito

ECI'23 - Universidad de Buenos Aires

Goals

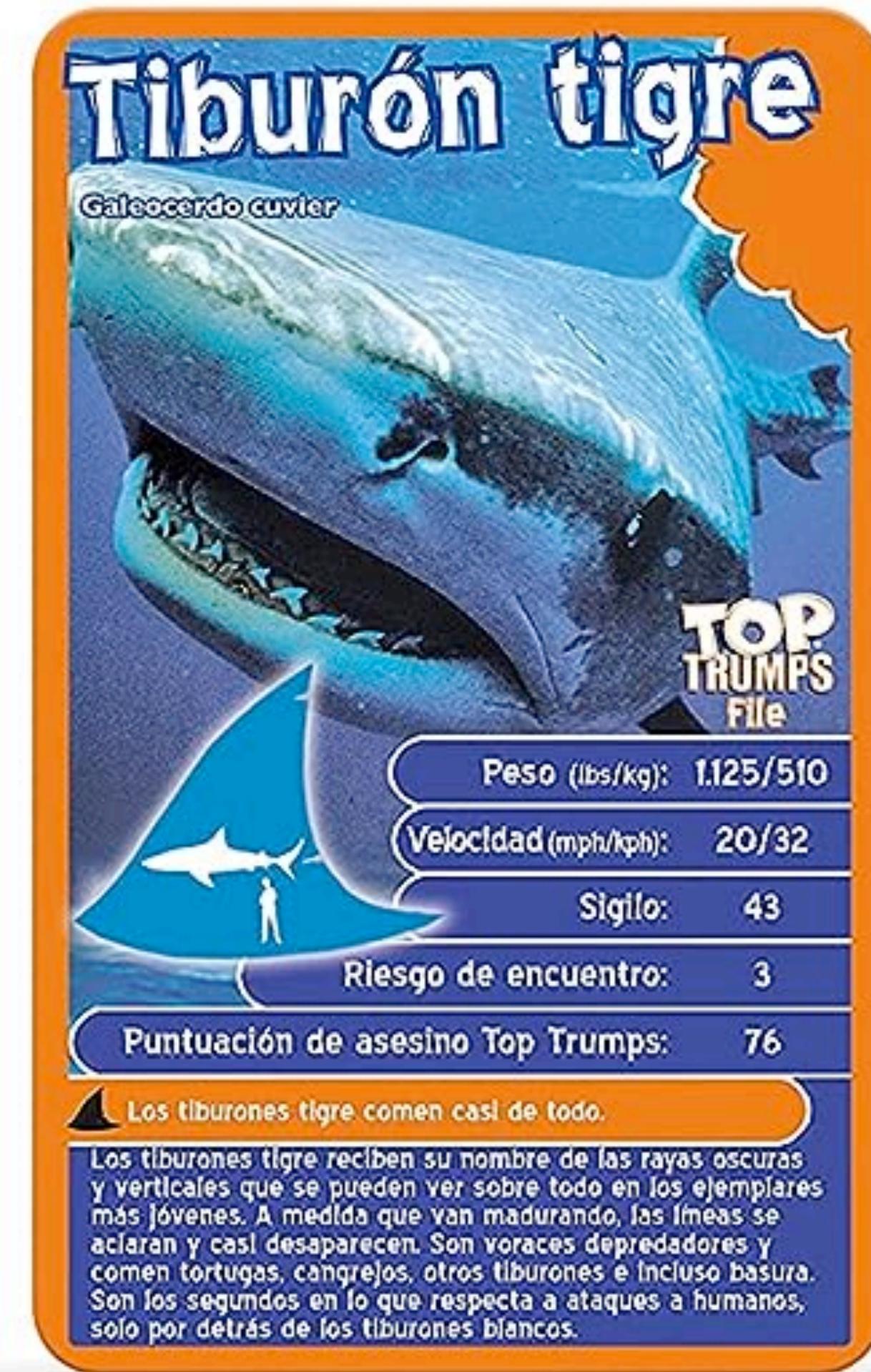
- Differential testing for compilers:
 - Across different compilers fro the same language
 - Across different variants of the same compiler
- Optimization invariants
 - Cross-optimization differential testing
 - Cross-architecture differential testing

Remember Differential Testing

80 km/h



32 km/h



What if we don't have two things to compare?

80 km/h



32 km/h



What if we don't have two things to compare?

After Lunch

80 km/h



Before Lunch

75 km/h



What if we don't have two things to compare?

In Nature

80 km/h



In Captivity

67 km/h



**How can we obtain
*variations of the same compiler?***

Optimization Invariants

- Insight: optimizations should not break the program
- Optimizations work in a *if proof then transform* fashion
- If no proof of correctness can be done, no transformation happens
- Thus, If optimizations break the program output we have a *bug*

Cross-Optimization Differential Testing

O2 level



00 level



Compiling with Optimizations

```
| program objectProgram result |
program := SmiParser parse: 'function main(){
    x := 7;
    return x + 1;
}'.

compiler := SmiCompiler new.

objectProgram := compiler compile: program.
result := self runProgram: objectProgram.
```

Compiling with Optimizations

```
| program objectProgram result |
program := SmiParser parse: 'function main(){
    x := 7;
    return x + 1;
}'.
```

```
compiler := SmiCompiler new.
compiler o1.
objectProgram := compiler compile: program.
result := self runProgram: objectProgram.
```

Smic optimization levels

- O0: no optimizations [DEFAULT]
- O1: stable optimizations
- O2: unstable optimizations

“Compiler starts in o0”

`compiler := SmiCompiler new.`

`compiler o1.`

`compiler o2.`

“Put it back to o0”

`compiler o0.`

Getting insights of a single optimization

- Testing a single optimization at a time
- Differential: single optimization vs no optimizations

“Compiler starts in o0”

```
compilerA := SmiCompiler new.
```

```
compilerB := SmiCompiler new.
```

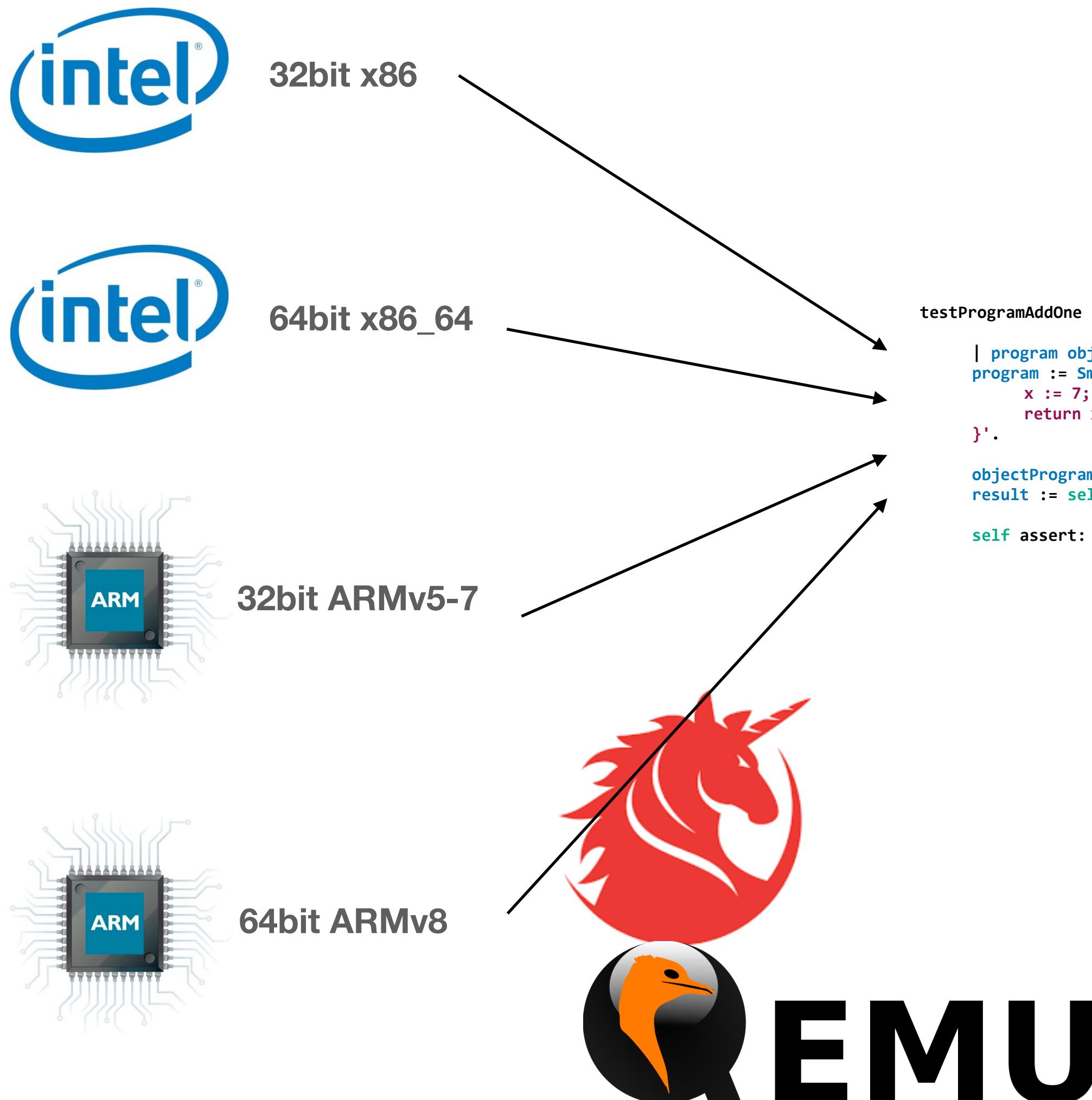
```
compilerB optimizations: { testedOptimization }
```

Obtaining all possible optimizations in Smic

SmiCompiler supportedOptimizations

```
>> anArray(  
  a DRBranchCollapse  
  a DRCopyPropagation  
  a DRPhiSimplification  
  a DROptimisationSequence  
  a DRCleanControlFlow  
  a DROptimisationSequence  
  a DRGlobalValueNumbering  
  a DRCogitCanonicaliser  
  a DROptimisationSequence)
```

Cross-architecture testing



```
testProgramAddOne
| program objectProgram result |
program := SmiParser parse: 'function main(){
    x := 7;
    return x + 1;
}'.

objectProgram := self compile: program.
result := self runProgram: objectProgram.

self assert: result equals: 8
```

**compilerA := SmiCompiler new.
compilerA architecture: #X64.**

**compilerB := SmiCompiler new.
compilerB architecture: #aarch64.**

Full Example

- Two compiler variants
- Differential runner

```
runnerA := PzBlockRunner on: [ :e | | p |
    p := compilerA compile: program.
    smiRunner run: p
].
```



```
runnerA := PzBlockRunner on: [ :e | | p |
    p := compilerB compile: program.
    smiRunner run: p
].
```



```
diffRunner := PzDifferentialRunner new
runnerA: runnerA;
runnerB: runnerB;
yourself.
```



```
diffRunner value: 'function main(){ ... }'
```

Possible Extensions and Next Steps

- Combine with grammar fuzzing!
- What should the runner compare?
 - The default differential runner compares exit statuses
 - Should they compare return values?
 - Exceptions?

Takeaways

- Optimizations keep program semantics
- Compilers could be modified to generate compiler versions
 - Cross-optimization
 - Cross-architecture
- Different observed behavior is evidence of bugs

Material

- An Empirical Comparison of Compiler Testing Techniques. ICSE'16
Junjie Chen et al.