Introduction

* Instruction set simulators (ISSs)用來幫助開發新的ISA及embedded software

ISS只要分為兩種 : dynamic binary translators(DBTs), interpreters

* Interpreter

依序逐行做fetch, decode並執行target instruction，實作上較簡單，但速度慢

* DBT

將整個程式轉成host instruction再執行，實作上較困難，但速度快

有instruction specific translation function去將target instruction轉成host instruction或是IR

* paper中提出一個新的框架 : 用interpreter的description去產生DBT

將特定行為的function丟給interpreter產生object file，再從object file中的template去產生DBT的translation function

* paper中實作C/C++的interpreter產生DBT

實驗證明，藉由這個框架使用足量的description產生的SH, MIPS64, ARM的執行速度會是原本interpreter的1.4到13.4倍

Related work

* JIT-CCS比一般的interpreter快，但還是比DBT慢
* 在QEMU中translation procedure是手寫的C code，因此將QEMU重新導向一個新的ISA的複雜程度比直接實作一個interpreter還高，這篇paper的框架能將這個複雜度變得跟直接實作interpreter差不多
* 部分的architecture description languages(ADLs)能產生ISS，但對開發人員來說他必須要額外去學這些語言才能產生ISS，但這個框架能讓開發人員從已經學會的語言去產生ISS (這裡是用C/C++)

Generated DBTs