Defining x86-64 Semantics in K

Sandeep Dasgupta

University of Illinois Urbana Champaign

December 17, 2017

Implemented

- X86-64 syntax & configuration
- Instruction Load Semantics
- Instruction fetch and execute semantics
- Instruction Semantics
  - 51 Base instructions (~3900 variants) and 11 pseuso-instructions (~316 variants).
    - Bit manipulation library built on top on MInt (or Bitvector).
    - Includes AVX2 instruction support.
    - Floating point arithmetic support.

Testing – Using GDB

```
.section .text
.globl _start

_start:
    addq $61438, %rax
    inforegisters
    addq $1, %rcx
    inforegisters
    adcb %cl,%al
    inforegisters
    adcb %cl,%al
    inforegisters
    nop
```

0_{48} 11101111 11111110

0_{48} 00000000 00000001

0_{48} 11101111 11111111

sf:1 pf:1

0_{48} 11101111 00000000

cf:1 af:1 zf:1 pf:1

```
129
158
159 4) reg  kstate  xstate
160 --------------------
161 rax     64'61184        61184
162 rbx     64'0    0
163 rcx     64'1    1
164 rdx     64'0    0
165 rsi     64'0    0
166 rdi     64'0    0
167 rsp     64'0    140737488347456
168 rbp     64'0    0
169 r8      64'0    0
170 r9      64'0    0
171 r10     64'0    0
172 r11     64'0    0
173 r12     64'0    0
174 r13     64'0    0
175 r14     64'0    0
176 r15     64'0    0
177 cf      1'1     1
178 pf      1'1     1
179 af      1'1     1
180 zf      1'1     1
181 sf      1'0     0
182 of      1'0     0
183 ymm0    256'0   00000000000000000000000000000000, 00000000000000000000000000000000
184 ymm1    256'0   00000000000000000000000000000000, 00000000000000000000000000000000
185 ...
186
```

Stratified Synthesis of “decb %bl”

decb %bl

```
movq $0xffffffffffffffff, %r9 # Storing -1 in %r9b
xorq   %rcx, %rcx             # carry flag cleared
adcb  %r9b, %bl               # %bl <- %bl – 1 + 0
```

- The synthesized program agrees with the target instruction only on the target’s write set.
  - Example: %r9 is used as scratch pad registers.
  - Can preserve the semantics by saving all the registers before executing the instructions and restoring only those not in the target’s write set.
- Stratified synthesis is not optimized for defining execution semantics.
  - We can use the formula generated by strata to generate the semantics.

%rbx : $\%rbx[63:8] \circ (0xff_9 + 0x0_1 \circ \%rbx[7:0])\ [7:0]$

%af : $(0xf_5 + 0x0_1 \circ \%rbx[3:0])[4:4] = 0x1_1$
%zf : $(0xff_9 + 0x0_1 \circ \%rbx[7:0])[7:0] = 0x0_8$
%sf : $(0xff_9 + 0x0_1 \circ \%rbx[7:0])[7:7] = 0x1_1$
%of : $(true \leftrightarrow \%rbx[7:7] = 0x1_1) \wedge\ !(true \leftrightarrow (0xff_9 + 0x0_1 \circ \%rbx[7:0])[7:7] = 0x1_1)$
…

To be implemented

- Instruction semantics using Strata (WIP)
  - Register and Immediate Variant.
- Semantics of memory instructions.
- Call instruction semantics.
- Library function support.

Related Work

- X86 modelling in Coq.
  - Coq: The world's best macro assembler?, PPDP '13
    - Non-FP, Non-SIMD subset of 32-X86, Total: 40/663 opcodes
  - Modular Development of Certified Program Verifiers with a Proof Assistant, ICFP '06
    - A subset of real X86-32.
- RockSalt
  - No Floating Point, SIMD, no I-64 Total: 219/663
- CompCert
  - Partial formal semantics of X86-32 in an RTL intermediate language.

Related Work

- ACL2 Developed by group led by Warren A. Hunt, Jr. in UT Austin
  - Supports ~300/663 opcodes.
  - Also can execute the semantics. Used for validating the model by co-simulation against actual X86 execution.
- Lifting Assembly to Intermediate Representation : A Novel Approach Leveraging Compilers, ASPLOS '16

Few Questions

- ACL2 vs K.
- With more rule, the kompile and krun are getting slower. Why krun!!