The C11 addition to Litmus

Jacques-Pascal Deplaix - London

Introduction: What is Litmus ?

Litmus is compiler which takes a « litmus test » and produces an executable that tests memory models

A litmus test looks like this:

```
X86 MP
""


{}


 P0          | P1            ;
 MOV [x],$1 | MOV EAX,[x] ;
 MOV [y],$1 | MOV ECX,[y] ;

exists (1:EAX=1 /\ 1:ECX=0)
```

The litmus compilation model

The internal compilation model is the following:

```
                 +--------------+        +------------------------------+
                 |              |        |                              |
Litmus test ---->|   Parsing    |--------| Internal sauce (compilation) |--+
                 |              |        |                              |  |
                 +--------------+        +------------------------------+  |
+--------------------------------------------------------------------------+
|  +------------------+       +-----------+
|  |                  |       |           |
+--| Backend (C code) |-------|    GCC    |------> Executable
   |                  |       |           |
   +------------------+       +-----------+
```

The out: An example

```
static void *P0(void *_vb) {
  mbar();
  ...
  for (int _i = _size_of_test-1 ; _i >= 0 ; _i--) {

    barrier_wait(_th_id,_i,&barrier[_i]);

asm __volatile__ (
"\n"
"#START _litmus_P0\n"
"#_litmus_P0_0\n\t"

"movl $1,%[x]\n"
"#_litmus_P0_1\n\t"

"movl $1,%[y]\n"
"#END_litmus\n\t"
:[x] "=m" (_a->x[_i]),[y] "=m" (_a->y[_i])
:
:"cc","memory"
);
  }
  mbar();
  return NULL;
}
```

How to run litmus

If you have an adventurous soul, these steps can compile the exemple given before:

```
$ mkdir /tmp/test
$ litmus MP.litmus -o /tmp/test
$ cd /tmp/test
$ make
...
$ ./MP.exe
Test MP Allowed
Histogram (4 states)
  500020:>1:EAX=0; 1:ECX=0;
  24    *>1:EAX=1; 1:ECX=0;
  5     :>1:EAX=0; 1:ECX=1;
  499951:>1:EAX=1; 1:ECX=1;
Ok
Condition exists (1:EAX=1 /\ 1:ECX=0) is validated
Observation MP Sometimes 24 999976
```

For more informations, see: http://diy.inria.fr/doc/litmus.html

Handling C: Motivations

My work was to extend litmus with a new frontend: the C language.

Extra note: Worked for 6 months, 2 days per week for my third year (undergraduate) part-time

The motivations for the C frontend is the following:

- Handle multiple architectures with the same test
- Can be used to test the C compiler itself
- Be able to test the C model

As a side effect, it also allow us to test the new C11 feature: atomics

The C frontend: An example

We can give the following example that contains atomics:

```
C MP+poscscs
"PodWWScSc RfeScSc PodRRScSc FreScSc"
Prefetch=0:x=F,0:y=W,1:y=F,1:x=T
Com=Rf Fr

{}

P0 (atomic_int* y, atomic_int* x) {
    atomic_store(x,1);
    atomic_store(y,1);
}

P1 (atomic_int* y, atomic_int* x) {
    int r0 = atomic_load(y);
    int r1 = atomic_load(x);
}
```

The C frontend: The result on ARM

```
Generated assembler
 @START _litmus_P1
 dmb sy
 ldr r2, [r2, r1]
 dmb sy
 dmb sy
 ldr r3, [r3, r1]
 dmb sy
 @END _litmus_P1

 @START _litmus_P0
 dmb sy
 str lr, [r2, r1]
 dmb sy
 dmb sy
 str lr, [r3, r1]
 dmb sy
 @END _litmus_P0

Test MP+poscscs Allowed
Histogram (3 states)
380879:>1:r0=0; 1:r1=0;
1298853:>1:r0=0; 1:r1=1;
320268:>1:r0=1; 1:r1=1;
No
```

The C frontend: The result on X86

```
Generated assembler
 #START _litmus_P1

 movl (%rdx), %edx
 movl (%rax), %eax
 #END _litmus_P1

 #START _litmus_P0

 movl $1, (%rdx)
 mfence
 movl $1, (%rax)
 mfence
 #END _litmus_P0

Test MP+poscscs Allowed
Histogram (3 states)
2000001:>1:r0=0; 1:r1=0;
126   :>1:r0=0; 1:r1=1;
1999873:>1:r0=1; 1:r1=1;
No
```

The C frontend: The result on PPC

```
Generated assembler
 #START _litmus_P1
 sync
 lwz 9,0(3)
 cmpw 7,9,9
 bne- 7,$+4
 isync
 sync
 rldicl 10,9,0,32
 lwz 9,0(4)
 cmpw 7,9,9
 bne- 7,$+4
 isync
 rldicl 9,9,0,32
 #END _litmus_P1

 #START _litmus_P0
 sync
 li 9,1
 stw 9,0(4)
 sync
 stw 9,0(3)
 #END _litmus_P0
```

```
Test MP+poscscs Allowed
Histogram (3 states)
6747927:>1:r0=0; 1:r1=0;
5063533:>1:r0=0; 1:r1=1;
4188540:>1:r0=1; 1:r1=1;
No
```

Conclusion & Future

Current state:

- Test the C model: no results for now

Future work:

- Have a C frontend for Herd
- Which implies to parse the given C code

Questions ?

« C'est pas faux ! »