# Memory Model Testing

## Setup guide for tool suite

It is recommended to install the tool suite using opam which is a package manager for the OCaml programming language as the tool suite is developed using this language.

```
$ sudo apt install opam
$ opam init
$ opam update
$ eval $(opam env)
```

After installing the opam package manager install the required modules for diy7 tool suite

```
$ apt install libgmp-dev
$ opam install herdtools7
```

## Setup guide for tool suite

Alternative to install diy7 with opam :

```
$ opam install dune menhir zarith
$  make all
$  make install
```

Verify the installations.

Tools available in the suite :-

Litmus7 : a tool to run tests on hardware

diy7 : test generator tool

Herd7 :  memory model simulator

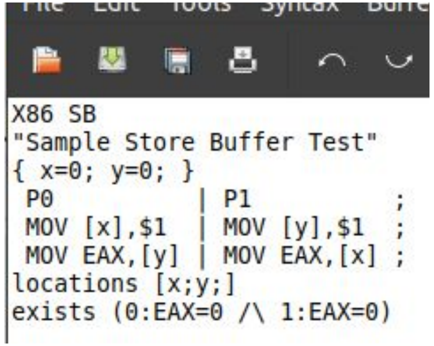
## How to run tests using litmus7 …..?

To run  tests we can use litmus7 tool using the below commands :

$ litmus7 <testname>

Example:

$ litmus7 sb.litmus



```
X86 SB
"Sample Store Buffer Test"
{ x=0; y=0; }
 P0          | P1          ;
 MOV [x],$1  | MOV [y],$1  ;
 MOV EAX,[y] | MOV EAX,[x] ;
locations [x;y;]
exists (0:EAX=0 /\ 1:EAX=0)
```

## How to run a test for a c file ...?

Run test for a c file :-

```
$ gcc -o test test.c -lpthread
$ ./test
```

## Generate c file for .litmus files using litmus7

```
$ litmus7 -o sb.tar sb.litmus
```

A tar file is generated which contains all the c files and a compiler script, Now run the compiler script which generates all the test files in c along with their .exe and .o files.

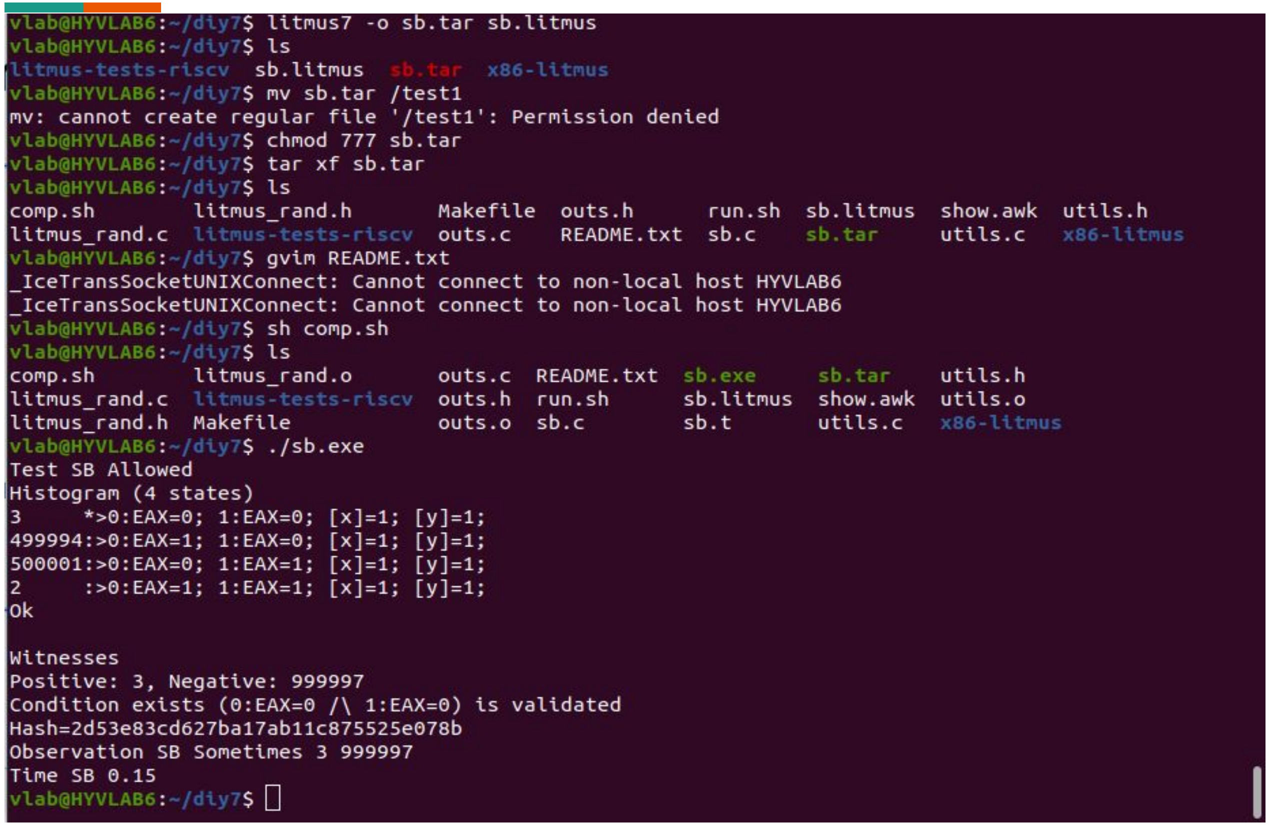
```
$ sh comp.sh
```

To run all the tests at a time :

```
$ sh run.sh
```

To run individual test (i.e sb test) :

```
$ ./sb.exe
```


```
vlab@HYVLAB6:~/diy7$ litmus7 -o sb.tar sb.litmus
vlab@HYVLAB6:~/diy7$ ls
litmus-tests-riscv  sb.litmus  sb.tar  x86-litmus
vlab@HYVLAB6:~/diy7$ mv sb.tar /test1
mv: cannot create regular file '/test1': Permission denied
vlab@HYVLAB6:~/diy7$ chmod 777 sb.tar
vlab@HYVLAB6:~/diy7$ tar xf sb.tar
vlab@HYVLAB6:~/diy7$ ls
comp.sh        litmus_rand.h       Makefile  outs.h      run.sh  sb.litmus  show.awk  utils.h
litmus_rand.c  litmus-tests-riscv  outs.c    README.txt  sb.c    sb.tar     utils.c   x86-litmus
vlab@HYVLAB6:~/diy7$ gvim README.txt
_IceTransSocketUNIXConnect: Cannot connect to non-local host HYVLAB6
_IceTransSocketUNIXConnect: Cannot connect to non-local host HYVLAB6
vlab@HYVLAB6:~/diy7$ sh comp.sh
vlab@HYVLAB6:~/diy7$ ls
comp.sh        litmus_rand.o       outs.c  README.txt  sb.exe     sb.tar    utils.h
litmus_rand.c  litmus-tests-riscv  outs.h  run.sh      sb.litmus  show.awk  utils.o
litmus_rand.h  Makefile            outs.o  sb.c        sb.t       utils.c   x86-litmus
vlab@HYVLAB6:~/diy7$ ./sb.exe
Test SB Allowed
Histogram (4 states)
3     *>0:EAX=0; 1:EAX=0; [x]=1; [y]=1;
499994:>0:EAX=1; 1:EAX=0; [x]=1; [y]=1;
500001:>0:EAX=0; 1:EAX=1; [x]=1; [y]=1;
2     :>0:EAX=1; 1:EAX=1; [x]=1; [y]=1;
Ok

Witnesses
Positive: 3, Negative: 999997
Condition exists (0:EAX=0 /\ 1:EAX=0) is validated
Hash=2d53e83cd627ba17ab11c875525e078b
Observation SB Sometimes 3 999997
Time SB 0.15
vlab@HYVLAB6:~/diy7$ 
```

## How to generate tests (.litmus files) using diy7 …?

To generate all tests for power pc:

```
$ diy7
```

To generate tests for a specific architecture use :

```
$ diy7 -conf <configuration file name>
```

Example  for generating tests for x86 architecture use x86.conf

```
$ diy7 -conf x86.conf
```

## Configuration for x86

Example .conf file for riscv architecture :

-arch X86

-name x86

-nprocs 3

-size 6

-mode default

## Configuration for Risc V

Example .conf file for riscv architecture :

```
-arch  RISCV
-name  Riscv
-nprocs 3
-size 6
-mode default
```

## Cloning Risc v tests from git :

Clone the risc v tests from git :

```
$ git clone https://github.com/litmus-tests/litmus-tests-riscv.git
```

The repository contains a make file which builds the hw-tests according to the specifications in the makefile, use GNU make to generate the test files.

```
$ make hw-tests CORES=2 GCC=riscv64-linux-gnu-gcc
```

The no of cores can be specified accordingly.

## Cross compilation for QEMU RISC V :

The extracted files contain run.sh and run.exe which are used to run the tests on the risc v machine.

We emulate the risc v processor architecture using QEMU system emulation: