CS252 Projects

3rd Week

Jaein Jeong

Johnathon Jamison

Last week we've determined to implement an introspective architecture that corrects transient errors with additional checker. This week we worked on a simulator, SimpleScalar. SimpleScalar is a processor simulator which was developed in the University of Wisconsin to simulate lots of functions in a modern processor. Those functions are branch prediction, two level instruction and data cache, TLB, and out-of-order execution unit. In addition, it can be extended by modifying its source code. Some examples of extensions are data predictors and multiprocessor simulation. We believe we can implement our idea by modifying SimpleScalar.

Installation

SimpleScalar runs on a UNIX machine like Solaris and Linux. To install SimpleScalar on a UNIX machine, we need to download three files: simplesim.tar.gz (simulator sources), simpleutils.tar.gz (binary utilities), and simpletools.tar.gz (compiler, assembler, and libraries). As the SimpleScaler site had only version 2, we received version 3 from Mark Whitney. The source code needs to be built with options specifying the machine you will use when running the object code. One is the machine type and the other is the endianess – big endian or little endian. After extracting the archiving files, we can setup the binary utilities, simulators, and compiler. In the following installation commands, <HOST> identifies the type of host machine. For example, i486-unknown-gnu is the identifier for a Linux machine on x86 (486 or better), and sparc-sun-sunos4.1.3 is for a Solaris machine. <TARGET> identifies whether the machine is big-endian or little-endian. The target option is ssbig-na-sstrix for a big-endian machine (Solaris) and sslittle-na-sstrix for a little endian-machine (Linux on x86). We compiled the utilities and the simulator on a Linux machine.

These are commands to compile binary utilities, $IDIR is the root directory for the SimpleScalar:

```
cd $IDIR/binutils-2.5.2
configure --host=<HOST> --target=<TARGET> --with-gnu-as
            --with-gnu-ld --prefix=$IDIR
make
make install
```

When we were compiling the simulator codes, there was an error; an include directive was trying to include the wrong file. We changed #include <bsd/sgtty.h> to #include <sgtty.h>. These are commands to compile the simulator:

```
cd $IDIR/simplesim-3.0
make
```

While we were compiling gcc for the SimpleScalar architecture, there were a couple redefinitions of something in stdio.h. So we commented out cccp.c line 194 and gcc.c line 172. These are commands to compile the simulator:

```
cd $IDIR/gcc-2.6.3
configure --host=<HOST> --target=<TARGET> --with-gnu-as
            --with-gnu-ld --prefix=$IDIR
make LANGUAGES=c
../simplesim-3.0/sim-safe ./enquire -f >! float.h-cross
make install
```

The compiler generates binary codes to run on SimpleScalar. The path of the compiler is $IDIR/sslittle-na-sstrix/bin/gcc for the simulator on Linux, and $IDIR/sslittle-na-sstrix/bin/gcc. Then, programs for SimpleScalar can be compiled using the command line or a make file.

This is a test code for SimpleScalar, *test-jaein.c*:

```
#include <stdio.h>
void main(void)
{
  char str[] = "A test program by Jaein Jeong...";
  fprintf(stdout, "str = %s\n", str);
  exit(0);
}
```

It is compiled with the same make file for a host program.

```
CC=../../../sslittle-na-sstrix/bin/gcc
CFLAGS=-g -O3
all: test-jaein
test-jaein: test-jaein.c
        $(CC) $(CFLAGS) -o test-jaein test-jaein.c
test:   all
        ../simplesim-0.1/sim-safe test-jaein
```

```
distclean:
        -make clean
clean:
        rm -f test-jaein *.[oia] core *~
```

It is compiled with the make file like this:

```
[jaein@bene src]$ make
../../../sslittle-na-sstrix/bin/gcc -g -O3 -o test-jaein test-jaein.c
```

After being built, the program can be run like this:

```
[jaein@bene src]$ sim-fast test-jaein
...
sim: ** starting *fast* functional simulation **
warning: syscall: ioctl: ioctl code not supported d=1, req=1074164744
str = A test program by Jaein Jeong...

sim: ** simulation statistics **
sim_num_insn                  8080 # total number of instructions executed
sim_elapsed_time                 1 # total simulation time in seconds
sim_inst_rate            8080.0000 # simulation speed (in insts/sec)
...
```