Jennifer Ediyanto

GNU Source Level Debugging

The H8/300 GNU Debugger (GDB) has source level debugging features which allow the users to debug their programs in C source level. This paper will provide a sample program and tutorial in source level debugging using GDB.

The following is a sample program (**Test.c**):

```
#include <stdio.h>

int sum, var1, var2;

main()
{
  sum = 0;
  var1 = 5;
  var2 = 10;

  process();

  iprintf("The value of var1 = %d\n", var1);
  iprintf("The value of var2 = %d\n", var2);
  iprintf("The value of sum = %d\n", sum);
}

process()
{
  var1 = var2 / var1;
  var2 = var1 * 3;
  sum = var1 + var2;
}
```

Compiling the Sample Program

The Test.c program has to be compiled with the following command line to produce the absolute file called **Test.x**:

C:\h8300\bin> **gcc -o test.x -g -O test.c**

where

gcc	name of the compiler
-o	compiler switch that specifies output file name
-g	compiler switch to include the debugging information
-O	compiler switch that optimizes the source code

The **g** switch is very important in order to be able to use the source level debugging.

Source Level Debugging

We can invoke the GDB Debugger by typing in GDB. The debugger will display the prompt **(gdb)**.

The following command starts and runs the debugger:

C:\h8300\bin> **gdb**

The command to connect the debugger to the simulator is as follows:

(gdb) **target sim**
Connected to the simulator

The following command loads the file called Test.x into the debugger:

(gdb) **load test.x**
*.text: 0x8000 .. 0xa45c ****
*.data: 0xa45c .. 0xa576 **
*.stack: 0xf000 .. 0xf014 **

The command to read the symbols from the absolute file is as follows:

(gdb) **file test.x**
Reading symbols from test.x ... done

The following command shows the first 10 lines of the source code:

(gdb) **list**

The command to see more source code (if the program is longer than 10 lines) is as follows:

(gdb) **<enter>**

The following command sets the breakpoint at line #7:

(gdb) **b 7**
breakpoint 1 at 0x807c: file test.c, line 7

The command to set the breakpoint at a function (i.e., process) is as follows:

(gdb) **b process**
breakpoint 2 at 0x80dc: file test.c, line 20

The following command gives the information on all breakpoints:

(gdb) **info break**

Num	*Type*	*Disp*	*Enb*	*Address*	*What*
1	*breakpoint*	*keep*	*Y*	*0x0000807c*	*in main at test.c:7*
2	*breakpoint*	*keep*	*Y*	*0x000080dc*	*in process at test.c:20*

The command to execute the progam is as follows:

(gdb) **run**
starting program: /h8300/bin/test.x
breakpoint 1, main() at test.c:7
7 sum=0;

The following command continues the program after the breakpoint:

(gdb) **c**
continuing
breakpoint 2, process() at test.c:20
20 var1 = var2 / var1;

The commands to single-step through the program are as follows:

(gdb) **s**
*21 var2 = var1 * 3;*
(gdb) **s**
22 sum = var1 + var2;
(gdb) **s**

23 }
(gdb) **s**
main() at test.c:13
13 iprintf("The value of var1 = %d\n", var1);

The following commands single-step over a function call: (We do not want to single step into function 'iprintf')

(gdb) **n**
The value of var1 = 2
main() at test.c:14
14 iprintf("The value of var2 = %d\n", var2);
(gdb) **n**
The value of var2 = 6
15 iprintf("The value of sum = %d\n",sum);
(gdb) **n**
The value of sum = 8
16 }
(gdb) **n**
0x802a in start()

If the single-stepping reaches the following message, then it marks the end of the program.

(gdb) **n**
Program received signal 1, Killed

The command to display information in all registers is as follows:

(gdb) **info reg**

r0	*0x15*	*21*
r1	*0x0*	*0*
r2	*0x0*	*0*
r3	*0xfffe*	*65534*
r4	*0x6*	*6*
r5	*0x21a*	*538*
r6	*0x0*	*0*
sp	*0xeffc*	*61436*
ccr	*0x4*	*4*
pc	*0x802c*	*32812*
cycles	*0x807c*	*32892*

The following commands display the address of a certain symbol:

(gdb) **info address main**
Symbol "main" is a function at address 0x8074
(gdb) **info address process**
Symbol "process" is a function at address 0x80d8

The command to disassemble the program from address 0x8074 to 0x80d8 is as follows:

(gdb) **disassem 0x8074 0x80d8**
Dump of assembler code from 0x8074 to 0x80d8:
...

The following command deletes the breakpoint at a certain function:

(gdb) **clear process**
Deleted breakpoint 2

The command to disable breakpoint #1 is as follows:

(gdb) **disable 1**

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

The following command gives the information of all breakpoints:

```
(gdb)  info  break
Num     Type           Disp    Enb    Address        What
1       breakpoint     keep    N      0x0000807c     in main at test.c:7
```

The command to exit from the debugger and return to the dos prompt is as follows:

```
(gdb)  q
C:\h8300\bin>
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300