Debugging Assembly Code with gdb

The GNU Debugger (GDB) is a portable debugger that runs on many Unix-like systems and works for many programming languages like C and C++, and for assembly code programs. It detects problems in a program while letting it run and allows users to examine different registers.

To use gdb with assembly programs you should assemble the .s files using the -g flag. This will include information in the object file to relate it back to the source file.

```
$ as -o program.o -g program.s --32

$ ld -melf_i386 -o program program.o
```

To invoke the debugger on the executable program, type in the terminal:

```
$ gdb ./program
```

This loads the executable program and brings up the gdb command line interpreter, which then waits for you to type commands. Program execution doesn't begin until you say so. Here are some useful commands. Many can be abbreviated, as shown.

`r[un]` *[args]*

Begin program execution. If the program normally takes command-line arguments you should specify them here.

`b[reak]` *[address]*

Set a breakpoint at the specified address (or at the current address if none specified). Addresses can be given symbolically or numerically. When execution reaches a breakpoint, you are thrown back into the gdb command line interpreter.

`c[ontinue]`

Continue execution after stopping at a breakpoint.

`i[nfo] r[egisters]` *[register]*

Print the value of a register (or, if none is specified, of all registers) in hex and decimal.

`s[tep]i`

Execute a single instruction and then return to the command line interpreter.

`q[uit]`

Exit from gdb.

https://en.wikipedia.org/wiki/GNU_Debugger