GDB Tutorial

Prepared for students learning to use the universal debugger called GDB: April 20, 2020.

View data values using the 'p' command

The format of the p command is **p/T <single-value argument>**

T is single letter designating the type of expected output. The choices for T are the form of the expected outputs are in the table below.

The argument may be any declared variable or any register from any component including GPR, FPU, SSE or AVX. See the special subsection on SSE (xmm) registers below.

The T type designator	Form of output
d	integer, decimal
t	integer, binary
s	string, ascii
a	integer, hex
f	IEEE format if parameter is float otherwise integer
u	integer, decimal
x	integer, hex

Examples: Let v be a declared variable in a C or C++ source program.

```
p/d   v
p/d   &v
p/f   v
p/f   &v
p/x   v
pix   &v
p/d   $rcx
p/x   $r8
p/s   $r14
p/t   $xmm3
p/f   $xmm12
and many others.
```

Registers of the SSE component

The SSE registers are named xmm0, xmm1, … through xmm15. Each xmm register holds 128 bits. There are various ways to place structure on those 128 bits. Here are some example of structuring those 128 bits.

v4_float: An array of 4 elements of 32 bits each; default output format is IEEE

v2_double: An array of 2 elements of 64 bits each; default output format is IEEE

v16_int8: An array of 16 elements of 8 bits each; default output format is twos complement

v8_int16: An array of 8 elements of 16 bits each; ditto

v4_int32: An array of 4 elements of 32 bits each; ditto

v2_int64: An array of 2 elements of 64 bits each; ditto

uint128: An array of 1 element of 128 bits; ditto

Default output format explained. When the “p” command is used to output an xmm register values in the first two structures named above will be assumed to floats conforming to the IEEE754 format. Values in the last 5 structures are assume to be twos complement integers. These defaults hold regardless of the number system used to express the values.

p/d $xmm2 outputs all values in all the structures using the 10 digits of the decimal system.

p/t $xmm2 outputs all values in all the structures using the 2 digits of the binary system

p/x $xmm2 outputs all values in all the structures using the 16 digits of the hex system.

View values in arrays using the p command

The format of the p command is **p/T <array argument>**

Let w be any array declared in C or C++. or X86. In all of the following examples the output will be all the elements of the array configured to adhere to the type specification

Examples:

```
p/x  w
p/d  w
p/t  w
p/f  w
and many others.
```

Be aware that the entire array will be outputted.

Watch how values are transformed to meet the output requirements of the type parameter. If the array w is a string but "p/d w" is used then integer values in base 10 are outputted because the type parameter is 'd'.

The X commands

The form is x + slash + parameter1 + parameter2 + parameter3 argument

parameter1 is a positive integer
parameter2 is the type of output (in each group)
parameter3 is the size of each output group

argument is an array (contiguous memory) declared in any source code: C, C++, X86.

parameter1 specifies how many groups of numbers to be outputted.
parameter2 is one of the type designators: d, t, s, a, x, etc.
parameter3 is specified by symbol: b=1 byte, h=2 bytes, w=4 bytes, g =8 bytes.

Suppose scores is an array that has been declared in either a high-level language program or a low-level language program. Where is is declared does not matter. The important characteristic is that it is a fixed number of contiguous bytes of memory.

Students of programming should test the following to thoroughly understand what each command produces. Familiarity with a technical tool leads to using that tool. It is important to use it. Reading about GDB is nice; but a student does not know gdb until he or she put his hands into it. [It is nice to read about swimming, but you can't pass a test in swimming until you do it.]

```
x/7dw   scores
x/8xw    scores
x.4xg     scores
```

```
x/6xw      scores
x/8xh      scores
x/10xb     scores
x/10d/w    scores
x/9sb      scores
x/7fw      scores
x/12tg     scores
x/8fg      scores
x/13uh     scores
x/20dg     $rsp
x/20/tg    $rip
x/11fg     $rcx
x/32uw     0x7fff23c33882
x/8fg      0x7fff23c55580
x/1tg      $rbp
```

To be continued.

Programmers: You may certainly test the commands in this tutorial by using your own programs. Some companion programs for use in learning gdb have been posted in the same folder where you obtained this tutorial document. You are welcome to learn gdb by experimenting with the posted sample programs, or use gdb with any existing program.

You don't need assembly to learn gdb. Gdb works on a dozen or more languages including the three we use in this course.