

1 Floating point numbers

Consider the following program:

```
1      section .data
2          x DD 3.14
3          y DD 2.1
4
5      section .text
6          MOVSS xmm0, [x]
7          MOVSS xmm1, [y]
8
9          ADDSS xmm0, xmm1
10
11         MOV eax, 1
12         INT 80h
```

See about **xmm** registers [here](#).

How many **xmm** registers?.

MOVSS instruction is used to move the floating point values to the **xmm** registers. **SS** means **S**caler **S**ingle precision.

We will execute this program with GDB.

```
(gdb) info registers xmm0
xmm0      {v8_bfloat16 = {0xf5c3, 0x4048, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v8_half = {0xf5c3, 0x4048, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_float = {0x4048f5c3, 0x0, 0x0, 0x0}, v2_double = {0x4048f5c3, 0x0}, v16_int8 = {0xc3, 0xf5, 0x48, 0x40, 0x0 <repeats 12 times>}, v8_int16 = {0xf5c3, 0x4048, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_int32 = {0x4048f5c3, 0x0, 0x0, 0x0}, v2_int64 = {0x4048f5c3, 0x0}, uint128 = 0x4048f5c3}
(gdb)
```

Figure 1: Values in **xmm**

We can see a bunch of arrays(I guess) which have a bunch of values in them. To get the value stored in **xmm0** we particularly use the **v4_float** array.

```
(gdb) p $xmm0.v4_float[0]
$1 = 3.1400001
(gdb)
```

Figure 2: **v4_float** array's 0th element(**xmm0**)

Notice that we assigned the value 3.14 to **xmm0** but in this image it's showing 3.1400001.

Similar behaviour can be seen in **xmm1** as well.

```
(gdb) p $xmm1.v4_float[0]
$3 = 2.09999999
(gdb) █
```

Figure 3: **v4_float** array's 0^{th} element(**xmm1**)

The reason why these values are not stored exactly as they were defined is that we use a special notation called [IEEE](#) floating-point notations.

From the image below:

```
(gdb) p $xmm0.v4_float[0]
$4 = 5.23999977
(gdb) █
```

Figure 4: Sum of **xmm0** and **xmm1**

We can see that the sum is also not exact.