

Re: Floating point load–store behaviour.

## Re: Floating point load–store behaviour.

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2006-08/msg04390.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2006-08/msg04390.html)

---

- *From:* Keith Thompson <kst-u@xxxxxxx>
  - *Date:* Wed, 23 Aug 2006 19:34:53 GMT
- 

thisismyidentity@xxxxxxxx writes:

I am trying to predict the behaviour of floating point load and store operations on integer locations. I ve written a small piece of code having some inline assembly code which I am describing here.

Why?

In standard C, attempting to read a floating–point value from an integer object invokes undefined behavior. Anything can happen. If you want to know what happens on your particular system, this isn't the place to ask about it.

```
=====
#include<stdio.h>
#include<stdlib.h>

void fp_int_mem_ops(unsigned long);
void print_byte_vals(unsigned long);

static int unmatched_count = 0;

main ()
```

This should be "int main(void)".

```
{
  unsigned long foo;
  int i, seed;
  for(i = 1000 ; i < 500000; i++){// Just to generate some random
  ints.
```

Please avoid "/\*" comments in code posted to this newsgroup. They're legal in C99, but not in C90, and as you can see they can create

## Re: Floating point load–store behaviour.

problems with line–wrapping. If a `"/* ... */` comment is wrapped onto a second line, it won't create a syntax error.

```
seed = rand();
srand(seed);
foo = rand();
fp_int_mem_ops(foo);
}
```

This is a *really* bad way to generate random numbers. By repeatedly calling `srand()`, you're interfering with the generator. Calling `srand()` and `rand()` in some arbitrarily complicated way gives you poorer random numbers, not better ones.

You should call `srand()` exactly once, before any call to `rand()`. One common way to do this is:

```
srand(time(NULL));
```

which sets a seed based on the current time. Then call `rand()` once for each random number you need.

If your system's `rand()` isn't good enough (it's often mediocre, and should not be depended on for cryptographically secure random numbers), use some system–specific random number generator; playing tricks with `rand()` will only make things worse.

The only reason to call `srand()` more than once is to repeat the same sequence, but I don't think that's what you need here.

```
printf ("\n%d mismatches\n", unmatched_count);
return 0;

}

void fp_int_mem_ops(unsigned long location)
{
    unsigned long fp, fp1;

    printf ("\n===\n");
    printf ("%x\n", location);

    __asm__ __volatile__ (
        "flds %1;"
        "fstps %0;"
        : "=m"(fp)
        : "m"(location), "m" (fp1)
        );
}
```

Re: Floating point load-store behaviour.

`__asm__` is non-standard, and we can't help you with it here. If you have questions about this, try a newsgroup that's specific to your system.

I don't understand your assembly code, but if `location` is supposed to be a memory location, you should declare it as a pointer, not as an integer.

```
printf("%x\n", fp);

if(location != fp)
{
printf("NOT MATCHED\n");
unmatched_count++;
}
```

```
return 0;
```

```
}
=====
```

[snip]

—  
Keith Thompson (The\_Other\_Keith) kst-u@xxxxxxxx <<http://www.ghoti.net/~kst>>  
San Diego Supercomputer Center <\*> <<http://users.sdsc.edu/~kst>>  
We must do something. This is something. Therefore, we must do this.