Subject: Re: "better" version of Iscale functions
Posted by mdelfede on Wed, 02 Apr 2008 14:48:36 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 02 April 2008 15:11mdelfede wrote on Tue, 01 April 2008 17:10
BTW, we could completely drop the assembly code, as-is it's not portable between compilers with greater integer width.

Yes, but int64 does not come cheap on non-64 architecture. Maybe even that FP computation could be faster. Of course, as long as FP is performed by HW. For ARM this new iscale can be good.

I thought you were in holydays

Back to Iscale, I don't know about modern processors that does have an hardware ftp and doesn't have 32x32->64 bit mul and 64/32 ->32 bit div core instructions... but I can be wrong.
Yet, I don't remember if intel ones works just with unsigned or signed or both integers...
BTW, I noticed that my iscale needs to work with 32 bit result; if not it'll use full 64 bit math for the multiply (in iscalefloor and iscaleceil) which can be slow.
I guess that using 32x32 multiply and 64/32 division, GCC translates it directly in DIV and MUL, but I've not checked yet.

Quote:
My version is also *not* portable on compilers with 64 bit wide integers, but can be made ok just changing function prototype :

int32_t iscale(int32_t x, int32_t y, int32_t z)

Leaving so to the compiler the integer width check and warnings.

IMO, that really is not that bug trouble, as any serious portable code should work with 32-bit int.

Mirek[/quote]

I can agree, but I think more and more that the lack of width specs in C++ is really a nasty stuff.
Now it's too late, but if I'd have to write a framework from scratch, I'd use some typedef'd int8, int16, int32 and so on stuffs.

BTW, our Iscale is much better than micro$oft's one, MulDiv, which returns -1 on 0 divisor.... so on both

MulDiv(1,-1,1)
MulDiv(a, b,0)

you get -1 as result.

Another possibility would be to use GCC built in asm, which is much different in syntax from Intel one (MS), but it's quite complicated, even if much more powerful.

Max