Subject: Re: "better" version of Iscale functions
Posted by mdelfede on Sun, 06 Apr 2008 17:36:21 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 06 April 2008 04:47mdelfede wrote on Wed, 02 April 2008 10:48
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.

Well, this is what MSC does seem to do to divide these numbers:

```
0041A920  push edi
0041A921  push esi
0041A922  push ebx
0041A923  xor edi,edi
0041A925  mov eax,[esp+0x14]
0041A929  or eax,eax
0041A92B  jnl 0x41a941
0041A92D  inc edi
0041A92E  mov edx,[esp+0x10]
0041A932  neg eax
0041A934  neg edx
0041A936  sbb eax,byte +0x0
0041A939  mov [esp+0x14],eax
0041A93D  mov [esp+0x10],edx
0041A941  mov eax,[esp+0x1c]
0041A945  or eax,eax
0041A947  jnl 0x41a95d
0041A949  inc edi
0041A94A  mov edx,[esp+0x18]
0041A94E  neg eax
0041A950  neg edx
0041A952  sbb eax,byte +0x0
0041A955  mov [esp+0x1c],eax
0041A959  mov [esp+0x18],edx
0041A95D  or eax,eax
0041A95F  jnz 0x41a979
0041A961  mov ecx,[esp+0x18]
0041A965  mov eax,[esp+0x14]
0041A969  xor edx,edx
0041A96B  div ecx
0041A96D  mov ebx,eax
```

```
0041A96F  mov eax,[esp+0x10]
0041A973  div ecx
0041A975  mov edx,ebx
0041A977  jmp short 0x41a9ba
0041A979  mov ebx,eax
0041A97B  mov ecx,[esp+0x18]
0041A97F  mov edx,[esp+0x14]
0041A983  mov eax,[esp+0x10]
0041A987  shr ebx,1
0041A989  rcr ecx,1
0041A98B  shr edx,1
0041A98D  rcr eax,1
0041A98F  or ebx,ebx
0041A991  jnz 0x41a987
0041A993  div ecx
0041A995  mov esi,eax
0041A997  mul dword [esp+0x1c]
0041A99B  mov ecx,eax
0041A99D  mov eax,[esp+0x18]
0041A9A1  mul esi
0041A9A3  add edx,ecx
0041A9A5  jc 0x41a9b5
0041A9A7  cmp edx,[esp+0x14]
0041A9AB  ja 0x41a9b5
0041A9AD  jc 0x41a9b6
0041A9AF  cmp eax,[esp+0x10]
0041A9B3  jna 0x41a9b6
0041A9B5  dec esi
0041A9B6  xor edx,edx
0041A9B8  mov eax,esi
0041A9BA  dec edi
0041A9BB  jnz 0x41a9c4
0041A9BD  neg edx
0041A9BF  neg eax
0041A9C1  sbb edx,byte +0x0
0041A9C4  pop ebx
0041A9C5  pop esi
0041A9C6  pop edi
0041A9C7  ret 0x10
```

I would say we should better learn GCC assembly syntax

Mirek

Wow, that's what I call "optimizing compiler".....

Max