Subject: Re: BufferPainter::Clear() optimization
Posted by mirek on Wed, 20 May 2020 13:58:30 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 20 May 2020 12:41
- 3T3 is faster on MSBT19 with short transfers up to 50-60 dwords.

Interestingly, adding "inline" to it seems to fix the problem... For some reason, 32-bit MSC does not inline it unless you ask it to do so...

In fact, assembler for both inlined function is virtually the same, the only difference is different tail handling (IMO, mine is 1% esier on eye):

7a:

```
0017940D  cmp ecx,byte +0x4
00179410  jnl 0x17942f
00179412  test cl,0x2
00179415  jz 0x17941f
00179417  mov [eax+0x4],edi
0017941A  mov [eax],edi
0017941C  add eax,byte +0x8
0017941F  test cl,0x1
00179422  jz dword 0x1794b4
00179428  mov [eax],edi
0017942A  jmp dword 0x1794b4
0017942F  movd xmm0,edi
00179433  pshufd xmm0,xmm0,0x0
00179438  movups [eax+ecx*4-0x10],xmm0     <<= tail handling
0017943D  cmp ecx,byte +0x20
00179440  jl 0x179486
00179442  cmp ecx,0x100000
00179448  jl 0x179457
0017944A  push ecx
0017944B  push edi
0017944C  push eax
0017944D  call dword 0x14ff88
00179452  add esp,byte +0xc
00179455  jmp short 0x1794b4
00179457  lea edx,[ecx-0x20]
0017945A  lea edx,[eax+edx*4]
0017945D  nop dword [eax]
00179460  movups [eax],xmm0
00179463  movups [eax+0x10],xmm0
00179467  movups [eax+0x20],xmm0
0017946B  movups [eax+0x30],xmm0
```

```
0017946F  movups [eax+0x40],xmm0
00179473  movups [eax+0x50],xmm0
00179477  movups [eax+0x60],xmm0
0017947B  movups [eax+0x70],xmm0
0017947F  sub eax,byte -0x80
00179482  cmp eax,edx
00179484  jna 0x179460
00179486  test cl,0x10
00179489  jz 0x17949d
0017948B  movups [eax],xmm0
0017948E  movups [eax+0x10],xmm0
00179492  movups [eax+0x20],xmm0
00179496  movups [eax+0x30],xmm0
0017949A  add eax,byte +0x40
0017949D  test cl,0x8
001794A0  jz 0x1794ac
001794A2  movups [eax],xmm0
001794A5  movups [eax+0x10],xmm0
001794A9  add eax,byte +0x20
001794AC  test cl,0x4
001794AF  jz 0x1794b4
001794B1  movups [eax],xmm0
```

3T3

```
00179540  cmp eax,byte +0x4
00179543  jnl 0x179560
00179545  test al,0x1
00179547  jz 0x17954e
00179549  mov [edx],edi
0017954B  add edx,byte +0x4
0017954E  test al,0x2
00179550  jz dword 0x179607
00179556  mov [edx],edi
00179558  mov [edx+0x4],edi
0017955B  jmp dword 0x179607
00179560  movd xmm0,edi
00179564  mov ecx,edx
00179566  pshufd xmm0,xmm0,0x0
0017956B  cmp eax,byte +0x20
0017956E  jl 0x1795c6
00179570  cmp eax,0x100000
00179575  jng 0x179589
00179577  test dl,0x3
0017957A  jnz 0x179589
0017957C  push eax
0017957D  push edi
```

```
0017957E  push edx
0017957F  call dword 0x14ff88
00179584  add esp,byte +0xc
00179587  jmp short 0x179604
00179589  mov edi,eax
0017958B  sar edi,0x2
0017958E  sub edi,byte +0x7
00179591  shl edi,0x4
00179594  add edi,edx
00179596  mov eax,ecx
00179598  movups [eax],xmm0
0017959B  lea eax,[ecx+0x70]
0017959E  movups [ecx+0x10],xmm0
001795A2  movups [ecx+0x20],xmm0
001795A6  movups [ecx+0x30],xmm0
001795AA  movups [ecx+0x40],xmm0
001795AE  movups [ecx+0x50],xmm0
001795B2  movups [ecx+0x60],xmm0
001795B6  sub ecx,byte -0x80
001795B9  movups [eax],xmm0
001795BC  cmp ecx,edi
001795BE  jc 0x179596
001795C0  mov eax,[ebp-0x14]
001795C3  mov edi,[ebp-0x18]
001795C6  test al,0x10
001795C8  jz 0x1795e3
001795CA  mov eax,ecx
001795CC  movups [eax],xmm0
001795CF  lea eax,[ecx+0x30]
001795D2  movups [ecx+0x10],xmm0
001795D6  movups [ecx+0x20],xmm0
001795DA  add ecx,byte +0x40
001795DD  movups [eax],xmm0
001795E0  mov eax,[ebp-0x14]
001795E3  test al,0x8
001795E5  jz 0x1795f8
001795E7  mov eax,ecx
001795E9  movups [eax],xmm0
001795EC  lea eax,[ecx+0x10]
001795EF  add ecx,byte +0x20
001795F2  movups [eax],xmm0
001795F5  mov eax,[ebp-0x14]
001795F8  test al,0x4
001795FA  jz 0x1795ff
001795FC  movups [ecx],xmm0
001795FF  movups [edx+eax*4-0x10],xmm0     <= TAIL
```

EDIT: OK, now rechecking it, it looks like 3T3 has a bit more instructions doing weird things....