Subject: What does SSE2 usage enhance?
Posted by crydev on Wed, 10 Apr 2013 11:31:12 GMT
View Forum Message <> Reply to Message

A question got up to me. What exactly are the pro's of using SSE2 as flag in your build? What kind of things can it speed up if the processor supports it?

Subject: Re: What does SSE2 usage enhance?
Posted by mirek on Mon, 15 Apr 2013 16:34:45 GMT
View Forum Message <> Reply to Message

crydev wrote on Wed, 10 April 2013 07:31A question got up to me. What exactly are the pro's of using SSE2 as flag in your build? What kind of things can it speed up if the processor supports it?

SSE2 tells compiler to use SSE2 for FP arithmetics in 32-bit SSE2 x86 builds. (In 64-bit builds, it is alsways on).

Cons: Executable will not work on CPUs without SSE2 - SSE2 is supported since PentiumIV and Athlon 64, basically for more than 10 years now...

Mirek

Subject: Re: What does SSE2 usage enhance?
Posted by crydev on Wed, 17 Apr 2013 08:54:28 GMT
View Forum Message <> Reply to Message

Thanks Mirek for your reply. Does that mean a SSE enabled version of functions as memcpy() can be used in U++? Or can I assume that by enabling SSE2 in the compiler flags automatically enables a SSE2 enabled version of memcpy? I think the Windows stock function is fairly slow.

p.s. My situation is copying blocks of 1~8 bytes very fast and very frequently in a loop. I have a chunk of memory which I loop though. Say my loop index currently is in the middle of my memory chunk, than it copies 4 bytes from the current index.

Subject: Re: What does SSE2 usage enhance?
Posted by mirek on Wed, 17 Apr 2013 09:17:29 GMT
View Forum Message <> Reply to Message

crydev wrote on Wed, 17 April 2013 04:54Thanks Mirek for your reply. Does that mean a SSE enabled version of functions as memcpy() can be used in U++? Or can I assume that by enabling SSE2 in the compiler flags automatically enables a SSE2 enabled version of memcpy? I think the Windows stock function is fairly slow.

SSE2 flag is likely to have no impact on memcpy. If it is emitted as function call (not intrinsics), it is likely that the function is SSE2 optimized anyway. For intrinsics, SSE2 code is way too complicated.

The main difference is in code using FP arithmetics: without SSE2, it is using x87 FP stack, with SSE2 it is using XMM register file, which is potentially faster.

Mirek

Subject: Re: What does SSE2 usage enhance?
Posted by mirek on Wed, 17 Apr 2013 09:20:17 GMT
View Forum Message <> Reply to Message

crydev wrote on Wed, 17 April 2013 04:54
p.s. My situation is copying blocks of 1~8 bytes very fast and very frequently in a loop. I have a chunk of memory which I loop though. Say my loop index currently is in the middle of my memory chunk, than it copies 4 bytes from the current index.

BTW, you might have a look at SVO_MEMCPY.

Subject: Re: What does SSE2 usage enhance?
Posted by crydev on Wed, 17 Apr 2013 20:04:15 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 17 April 2013 11:20crydev wrote on Wed, 17 April 2013 04:54
p.s. My situation is copying blocks of 1~8 bytes very fast and very frequently in a loop. I have a chunk of memory which I loop though. Say my loop index currently is in the middle of my memory chunk, than it copies 4 bytes from the current index.

BTW, you might have a look at SVO_MEMCPY.

Thanks Mirek for your suggestion. I tried it but that one is actually a lot slower than the conventional memcpy. I stepped through the ASM though and it seems that the default memcpy is already partially optimzed for SSE2.

Subject: Re: What does SSE2 usage enhance?
Posted by mirek on Thu, 18 Apr 2013 04:34:16 GMT
View Forum Message <> Reply to Message

crydev wrote on Wed, 17 April 2013 16:04mirek wrote on Wed, 17 April 2013 11:20crydev wrote on Wed, 17 April 2013 04:54
p.s. My situation is copying blocks of 1~8 bytes very fast and very frequently in a loop. I have a chunk of memory which I loop though. Say my loop index currently is in the middle of my memory

chunk, than it copies 4 bytes from the current index.

BTW, you might have a look at SVO_MEMCPY.

Thanks Mirek for your suggestion. I tried it but that one is actually a lot slower than the conventional memcpy. I stepped through the ASM though and it seems that the default memcpy is already partially optimzed for SSE2.

Are you really copying just 1-8 bytes? For such small amounts and unaligned data, SSE2 is IMO meaningless.

Mirek

Subject: Re: What does SSE2 usage enhance?
Posted by crydev on Thu, 18 Apr 2013 06:33:06 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 18 April 2013 06:34crydev wrote on Wed, 17 April 2013 16:04mirek wrote on Wed, 17 April 2013 11:20crydev wrote on Wed, 17 April 2013 04:54
p.s. My situation is copying blocks of 1~8 bytes very fast and very frequently in a loop. I have a chunk of memory which I loop though. Say my loop index currently is in the middle of my memory chunk, than it copies 4 bytes from the current index.

BTW, you might have a look at SVO_MEMCPY.

Thanks Mirek for your suggestion. I tried it but that one is actually a lot slower than the conventional memcpy. I stepped through the ASM though and it seems that the default memcpy is already partially optimzed for SSE2.

Are you really copying just 1-8 bytes? For such small amounts and unaligned data, SSE2 is IMO meaningless.

Mirek

Yes it is only about small blocks of 1 ~ 8 bytes. I'll keep it at memcpy. It is good enough. Thanks

---

Subject: Re: What does SSE2 usage enhance?
Posted by mirek on Thu, 18 Apr 2013 08:38:36 GMT
View Forum Message <> Reply to Message

Well, I am asking because in that case SVO_MEMCPY has a problem (it should be optimization over memcpy for small blocks).

Can you show me a small code snippet?

Mirek

Subject: Re: What does SSE2 usage enhance?
Posted by crydev on Thu, 18 Apr 2013 09:43:22 GMT
View Forum Message <> Reply to Message

Here is a small code snippet. If you see anything that is strange or wrong I would appreciate feedback.

The memcpy line is the line where I copy, in this case, 4 bytes from the address of i in array buffer to float variable tempStore.

```
template<>
void MemoryScanner::ScanWorker(const MemoryRegion& region, const float& value)
{
 Byte *buffer = (Byte*)MemoryAlloc(region.MemorySize);
 if (!ReadProcessMemory(this->mOpenedProcessHandle, (void*)region.BaseAddress, buffer
  , region.MemorySize, NULL))
 {
  MemoryFree(buffer);
  return;
 }

 Vector<MemoryBlockBase*> localResults;

 for (int i = 0; i < region.MemorySize; i++)
 {
  float tempStore;
  memcpy(&tempStore, &(buffer[i]), sizeof(float));

  if (TemplateCompare(tempStore, value)) // WRITES TO FREED BLOCKS DETECTED
  {
   MemoryBlock<float>* mb = new MemoryBlock<float>();
   mb->Address = static_cast<unsigned int>(region.BaseAddress + i);
   mb->Size = sizeof(float);
   mb->Buffer = tempStore;
   localResults.Add(mb);
  }
 }

 MemoryFree(buffer);

 AtomicInc(this->ThreadFinishCount);

 if (localResults.GetCount() > 0)
 {
  this->AddThreadSpecificSearchResults(localResults);
```

```
 }

 this->UpdateScanningProgress(this->ThreadFinishCount);
}
```

Subject: Re: What does SSE2 usage enhance?
Posted by mirek on Thu, 18 Apr 2013 09:58:49 GMT
View Forum Message <> Reply to Message

I see, I have checked the assembly and memcpy gets inlined with unaligned load, which is still faster than loading 4 bytes separated like SVO_MOVE does.

So the actual code for this memcpy is

```
00401744  mov ecx,[eax]
00401746  mov [esp+0x4],ecx
```

(SVO_MOVE is designed for small variable size).

Mirek
