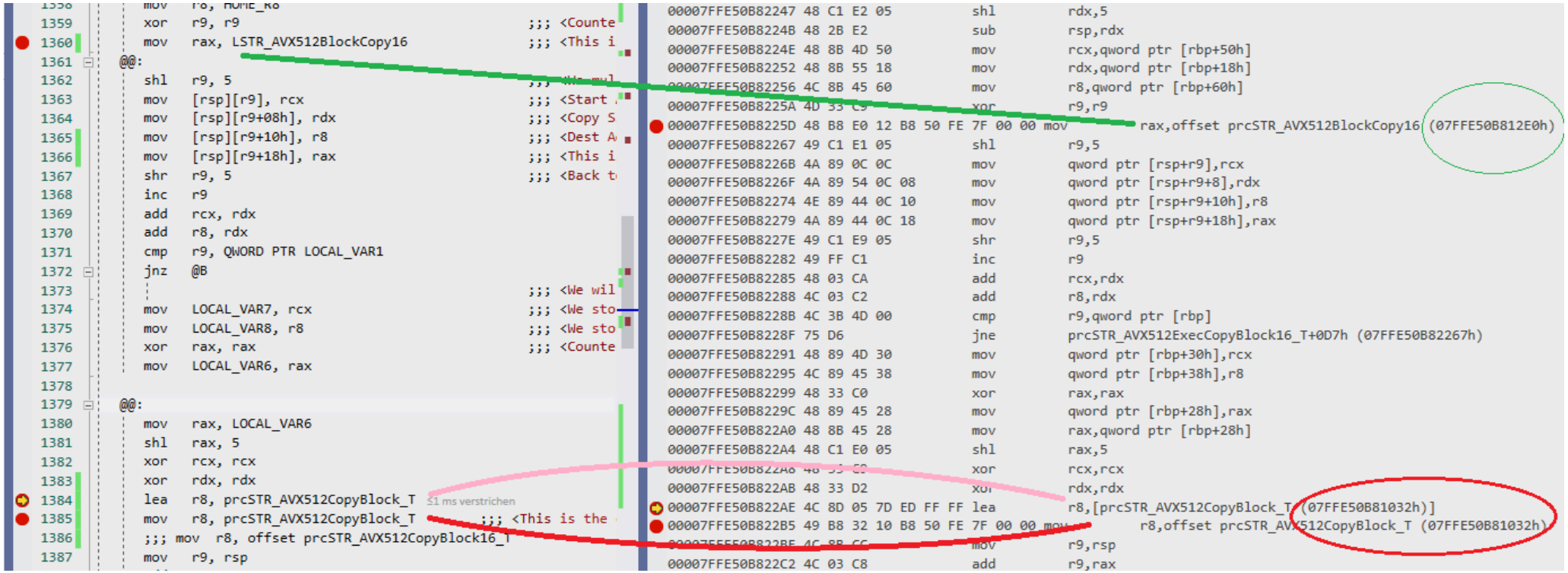
Hello,

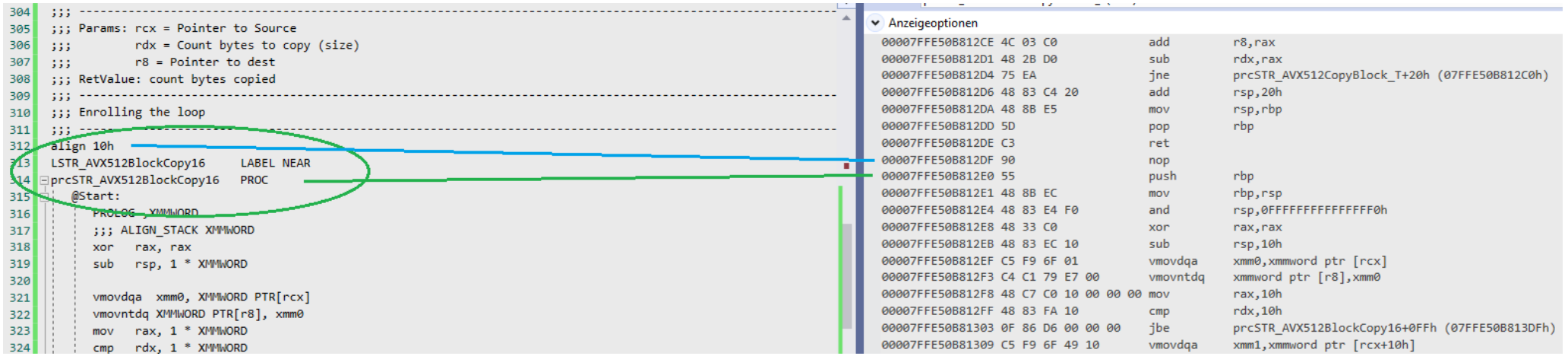
I think I found a BUG or may be I am wrong, and it's just something in the settings of MASM which I didn't use correctly. Below you will see a code snippet, where I try to load some pointer of procs which I have created in the same code seg (there is only one because of 64 Bit and non segements only 1 flat). Regarding to the recomended software optimization guide of intel and AMD I aligned nearly all procedures on 16 Byte boundaries. As you can see in the picture (1) below the first offset calc (GREEN) works fine. This is due to adding a LABEL directive before the PROCEDURE prcSTR_AVX512BlockCopy16 definition. (you will see this view pages later.) But the second one (RED) is wrong, as you can see on the address used for offset xxxxxx1032 which is not a natural 16 Byte boundary.

(Picture 1)



The following picture (2) shows you the defintion of the correctly measured pointer (GREEN), which is those one where I‘ve used the directive LABEL explicitly before the PROC definition. There is no bug on the align directive because as you can see the NOP which is included in the flow to start the PROC on 16 Byte boundary xxxxxxx12E0 .

(Picture 2)



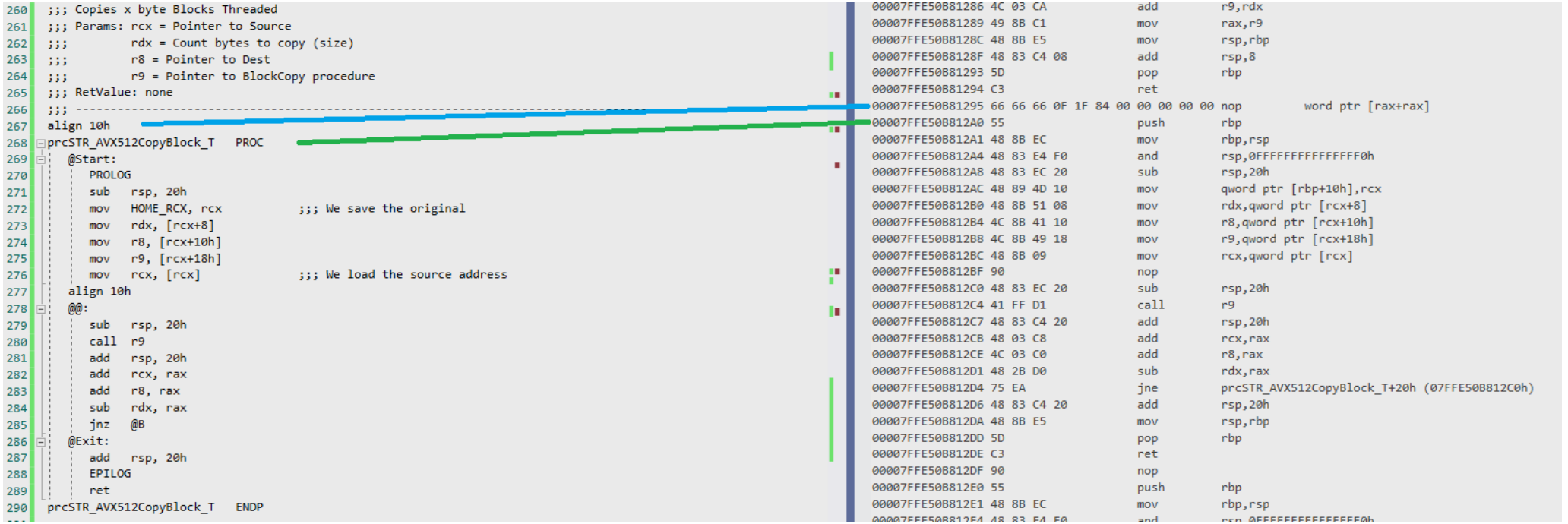
```
304  ;;; ----------------------------------------------------------------------------------------------------------------
305  ;;; Params: rcx = Pointer to Source
306  ;;;         rdx = Count bytes to copy (size)
307  ;;;         r8 = Pointer to dest
308  ;;; RetValue: count bytes copied
309  ;;; ----------------------------------------------------------------------------------------------------------------
310  ;;; Enrolling the loop
311  ;;; ----------------------------------------------------------------------------------------------------------------
312  align 10h
313  LSTR_AVX512BlockCopy16     LABEL NEAR
314  prcSTR_AVX512BlockCopy16   PROC
315    @Start:
316      PROLOG _XMMWORD
317      ;;; ALIGN_STACK XMMWORD
318      xor   rax, rax
319      sub   rsp, 1 * XMMWORD
320
321      vmovdqa  xmm0, XMMWORD PTR[rcx]
322      vmovntdq XMMWORD PTR[r8], xmm0
323      mov   rax, 1 * XMMWORD
324      cmp   rdx, 1 * XMMWORD
```

Anzeigeoptionen

```
00007FFE50B812CE 4C 03 C0            add         r8,rax
00007FFE50B812D1 48 2B D0            sub         rdx,rax
00007FFE50B812D4 75 EA               jne         prcSTR_AVX512CopyBlock_T+20h (07FFE50B812C0h)
00007FFE50B812D6 48 83 C4 20         add         rsp,20h
00007FFE50B812DA 48 8B E5            mov         rsp,rbp
00007FFE50B812DD 5D                  pop         rbp
00007FFE50B812DE C3                  ret
00007FFE50B812DF 90                  nop
00007FFE50B812E0 55                  push        rbp
00007FFE50B812E1 48 8B EC            mov         rbp,rsp
00007FFE50B812E4 48 83 E4 F0         and         rsp,0FFFFFFFFFFFFFFF0h
00007FFE50B812E8 48 33 C0            xor         rax,rax
00007FFE50B812EB 48 83 EC 10         sub         rsp,10h
00007FFE50B812EF C5 F9 6F 01         vmovdqa     xmm0,xmmword ptr [rcx]
00007FFE50B812F3 C4 C1 79 E7 00      vmovntdq    xmmword ptr [r8],xmm0
00007FFE50B812F8 48 C7 C0 10 00 00 00 mov        rax,10h
00007FFE50B812FF 48 83 FA 10         cmp         rdx,10h
00007FFE50B81303 0F 86 D6 00 00 00   jbe         prcSTR_AVX512BlockCopy16+0FFh (07FFE50B813DFh)
00007FFE50B81309 C5 F9 6F 49 10      vmovdqa     xmm1,xmmword ptr [rcx+10h]
```

And now in picture 3 I will show you the definition of the PROCEDURE which address is determined wrong. Again as you can see the align directive works fine (BLUE ; NOP etc..) . And therefore the PROC is correctly aligned. But if you will remember the first picture (1) this address is not used for address calc (see above pic. 1). The address which was determined is 07FFE50B81032h (this is wrong) it must be 07FFE50B812A0h (this ist he correct one)

(Picture 3)



Instead of using the correct address, I find out that the compiler creates something like a table where all the procedure are available (Picture 4). This looks like a dynamic created table of pointer's to all procs of the module. I didn't create this.

I hope there is only something missing or wrong defined in the settings of Visual Stuido or MASM.

(Picture 4)

```
00007FFE50B81005 E9 A6 0D 00 00       jmp         prcSTR_AVX512ProcessCopyBlock16 (07FFE50B81DB0h)
00007FFE50B8100A E9 11 14 00 00       jmp         prcSTR_AVX512CopyBlock16_T (07FFE50B82420h)
00007FFE50B8100F E9 DC 03 00 00       jmp         prcSTR_AVX512BlockCopy32 (07FFE50B813F0h)
00007FFE50B81014 E9 F7 0D 00 00       jmp         prcSTR_AVX512ExecCopyBlock64_T (07FFE50B81E10h)
00007FFE50B81019 E9 62 0C 00 00       jmp         prcSTR_AVX512CopyString (07FFE50B81C80h)
00007FFE50B8101E E9 AD 0F 00 00       jmp         prcSTR_AVX512ExecCopyBlock32_T (07FFE50B81FD0h)
00007FFE50B81023 E9 D0 00 00 00       jmp         prcSTR_AVX512GetLengthAligned16 (07FFE50B810F8h)
00007FFE50B81028 E9 23 0D 00 00       jmp         prcSTR_AVX512ProcessCopyBlock32 (07FFE50B81D50h)
00007FFE50B8102D E9 46 14 00 00       jmp         prcSTR_AVX512StringEqual (07FFE50B82478h)
00007FFE50B81032 E9 69 02 00 00       jmp         prcSTR_AVX512CopyBlock_T (07FFE50B812A0h)
00007FFE50B81037 E9 24 12 00 00       jmp         prcSTR_AVX512CopyBlock32_T (07FFE50B82260h)
00007FFE50B8103C E9 8F 00 00 00       jmp         prcLib_AVX512Initialize (07FFE50B810D0h)
00007FFE50B81041 E9 4A 11 00 00       jmp         prcSTR_AVX512ExecCopyBlock16_T (07FFE50B82190h)
00007FFE50B81046 E9 2D 01 00 00       jmp         prcSTR_AVX512GetLength16 (07FFE50B81178h)
00007FFE50B8104B E9 10 13 00 00       jmp         prcSTR_AVX512CopyBlock64_T (07FFE50B82360h)
00007FFE50B81050 E9 9B 0C 00 00       jmp         prcSTR_AVX512ProcessCopyBlock64 (07FFE50B81CF0h)
00007FFE50B81055 E9 86 02 00 00       jmp         prcSTR_AVX512BlockCopy16 (07FFE50B812E0h)
00007FFE50B8105A E9 D9 00 00 00       jmp         prcSTR_AVX512GetLengthAligned32 (07FFE50B81138h)
00007FFE50B8105F E9 A4 01 00 00       jmp         prcSTR_AVX512GetLength32 (07FFE50B81208h)
00007FFE50B81064 E9 37 06 00 00       jmp         prcSTR_AVX512BlockCopy64 (07FFE50B816A0h)
```