In the original code, the real number that had to be shown, had been previously rounded and then multiplied by 100000000000000000, and is into the st0 FPU register just ready to be store in memory in 18 digits BCD (Binary Coded Decimal) format.
With the new code we avoid the initial rounded code (frndint noped) and then obtain the decimal part and store it in memory just before do the same with the integer part to get an extra precision of the real number. Then we avoid again other rounded made in the original code (in normal asm). Indeed the memory space used for the last rounded is used to insert the new code.

Bytes:
**D9 C0 D9 C0 9B D9 7D E6 66 81 65 E6 FF F3 66 81 4D E6 00 0C D9 6D E6 D9 FC DE E9 66 B8 64 00 66 89 45 E6 DE 4D E6 DF 75 E8 DF 75 E9 9B DB E3**

```
fld st(0),st(0)                      //store value in st0 also in st1
fld st(0),st(0)                      //now also in st2
fwait                                //Check pending unmasked floating-point exceptions
fnstcw word ptr ss:[ebp-1A]          //save FPU register control word in memory
and word ptr ss:[ebp-1A],F3FF        //clears only the RC (Rounding Control) bits in FPU register control word, leaving all other bits unchanged
or word ptr ss:[ebp-1A],C00          //set the RC bits to the FPU register control word to cero (00) -> truncate
fldcw word ptr ss:[ebp-1A]           //load the  FPU register control word with the new set bits
frndint                              //with these setting frndint make a truncate of the st0 register value
fsubp st(1),st(0)                    //subtract st1 minus st0
mov ax,64                            //store $64 (decimal 100) in ax
mov word ptr ss:[ebp-1A],ax          //move ax to memory
fimul st(0),word ptr ss:[ebp-1A]     //multiply value in st0 by decimal 100
fbstp tword ptr ss:[ebp-18],st(0)    //store integer part of st0 value into 18 digits BCD (Binary Coded Decimal) format ans store it in memory and
                                     //pop st0
fbstp tword ptr ss:[ebp-17],st(0)
fwait                                //Check pending unmasked floating-point exceptions
fninit                               //Initialize FPU without checking for pending unmasked floating-point exceptions
```