Tvorba vlastních API funkcí.

Jejich implementace do C# nebo C++ .

Zdeněk Pavlů

03.01.2018

Ukázky jak vytvářet vlastní API 32 z hlediska vstupů a výstupů z těchto funkcí.

1 - Co budeme potřebovat.

- Trpělivost.
- Literaturu a skripta :
  - Michal Brandejs - Mikroprocesory Intel – Pentium
    (Tento text byl vydán v nakladatelství Grada v roce 1994. Po vypršení platnosti nakladatelské smlouvy byl text autorem jako další vydání elektronicky zveřejněn dne 1. 9. 2010. Text lze šířit výhradně bezplatně a s uvedením autora a této copyrightové doložky. Text lze libovolně citovat, pokud je uveden odkaz na zdroj následovně: Brandejs, M. Mikroprocesory Intel – Pentium [online]. Brno : Fakulta informatiky, Masarykova univerzita, 2010.)
  - Assembler x86 - Ing. Petr Olivka, Ph.D.
  - Assembler ALES KEPRT
    (Tento text má za cíl posloužit jako učebnice programování v assembleru. Popisuje 32bitový assembler procesorů Intel (řady označované jako IA32 či x86) s cílem umožnit čtenářům či studentům základní pochopení principů, na kterých funguje assembler a potažmo mikroprocesory v současných počítačích. Ve výuce je assembler obvykle součástí cvičení některého z kurzů v řadě „Operační systémy“.)
  - Instrukce MMX - Martin Horák, Michal Zamazal, Lukáš Džbánek, Martin Švirák
    (MMX technologie je vytvořena firmou Intel pro urychlení multimediálních aplikací. Tato technologie obsahuje nové instrukce a datové typy, které zvyšují výkonnost procesoru. MMX technologie znamenala největší pokrok ve vývoji procesorů od doby, kdy se na trh dostaly 32-bit procesory Intel 386.)
  - SIMD instrukce využívané v moderních mikroprocesorech.
  - FPU, MMX, SSE, SSE2, SSE3 instrukce popis.
  - Popis instrukcí procesoru
  - Manuál pro MASM.
- Software:
  - Visual studio 2017 nebo nižší verzi.
    (Rozšířit a instalovat „**PInvoke.net“ , „Asm-Dude“)**
  - Nainstalovat MASM a WinASM studio instalace popsána na mém webu Manualy WinASM studio a MASM32.
  - Zpětný assembler W33Dasm (neumí dekódovat instrukce SSE a 64 bitové)
  - Zpětný assembler 32 a 64 bitů PEBrowse64 Professional složitější ale lepší.
- Omezení:
  - Možnost vytvářet pouze 32 bitové aplikace.
  - Každé volání Pivoke přináší zpomalení 20-30 cyklů procesoru z toho důvodu se snažíme každou funkci vytvořit tak, abychom tyto volání omezili.
  - Proto v co největší míře voláme hotové funkce API Windows v námi navržené funkci.
  - Naše API jsou uloženy v knihovně „dll“.
- **Možnosti WinAsm:**
  - Používané registry (AH AL AX BH BL BP BX CH CL CR0 CR2 CR3 CR4 CS CX DH DI DL DR0 DR1 DR2 DR3 DR6 DR7 DS DX EAX EBP EBX ECX EDI EDX ES ESI ESP FS GS MM0 MM1 MM2 MM3 MM4 MM5 MM6 MM7 SI SP SS ST XMM0 XMM1 XMM2 XMM3 XMM4 XMM5 XMM6 XMM7)
  - Nepoužívané registry(TR3 TR4 TR5 TR6 TR7)
  - Uživatelské instrukce(AAA AAD AAM AAS ADC ADD AND BSF BSR BSWAP BT BTC BTR BTS CALL CBW CDQ CLC CLD CMC CMOVA CMOVAE CMOVB CMOVBE CMOVC CMOVE

CMOVG CMOVGE CMOVL CMOVLE CMOVNA CMOVNAE CMOVNB CMOVNBE CMOVNC CMOVNE CMOVNG CMOVNGE CMOVNL CMOVNLE CMOVNO CMOVNP CMOVNS CMOVNZ CMOVO CMOVP CMOVPE CMOVPO CMOVS CMOVZ CMP CMPS CMPSB CMPSD CMPSW CMPXCHG CMPXCHG8B CPUID CWD CWDE DAA DAS DEC DIV ENTER IDIV IMUL INC JA JAE JB JBE JC JCXZ JE JECXZ JG JGE JL JLE JMP JNA JNAE JNB JNBE JNC JNE JNG JNGE JNL JNLE JNO JNP JNS JNZ JO JP JPE JPO JS JZ LAHF LEA LEAVE LODS LODSB LODSD LODSW LOOP LOOPD LOOPE LOOPED LOOPEW LOOPNE LOOPNED LOOPNEW LOOPNZ LOOPNZD LOOPNZW LOOPW LOOPZ LOOPZD LOOPZW MOV MOVS MOVSB MOVSD MOVSW MOVSX MOVZX MUL NEG NOP NOT OR POP POPA POPAD POPF POPFD PUSH PUSHA PUSHAD PUSHD PUSHF PUSHFD PUSHW RCL RCR REP REPE REPNE REPNZ REPZ RET RETF RETN ROL ROR SAHF SAL SAR SBB SCAS SCASB SCASD SCASW SETA SETAE SETB SETBE SETC SETE SETG SETGE SETL SETLE SETNA SETNAE SETNB SETNBE SETNC SETNE SETNG SETNGE SETNL SETNLE SETNO SETNP SETNS SETNZ SETO SETP SETPE SETPO SETS SETZ SHL SHLD SHR SHRD STC STD STOS STOSB STOSD STOSW SUB TEST UD2 XADD XCHG XLAT XLATB XOR)

- Systémové instrukce(ARPL BOUND CLI CLTS HLT IN INS INSB INSD INSW INT INTO INVD INVLPG IRET IRETD IRETDF IRETF LAR LDS LES LFS LGDT LGS LIDT LLDT LMSW LOCK LSL LSS LTR OUT OUTS OUTSB)
- FPU instrukce(F2XM1 FABS FADD FADDP FBLD FBSTP FCHS FCLEX FCMOVB FCMOVBE FCMOVE FCMOVNB FCMOVNBE FCMOVNE FCMOVNU FCMOVU FCOM FCOMI FCOMIP FCOMP FCOMPP FCOS FDECSTP FDIV FDIVP FDIVR FDIVRP FFREE FIADD FICOM FICOMP FIDIV FIDIVR FILD FIMUL FINCSTP FINIT FIST FISTP FISUB FISUBR FLD FLD1 FLDCW FLDENV FLDENVD FLDENVW FLDL2E FLDL2T FLDLG2 FLDLN2 FLDPI FLDZ FMUL FMULP FNCLEX FNINIT FNOP FNSAVE FNSAVED FNSAVEW FNSTCW FNSTENV FNSTENVD FNSTENVW FNSTSW FPATAN FPREM FPREM1 FPTAN FRNDINT FRSTOR FRSTORD FRSTORW FSAVE FSAVED FSAVEW FSCALE FSIN FSINCOS FSQRT FST FSTCW FSTENV FSTENVD FSTENVW FSTP FSTSW FSUB FSUBP FSUBR FSUBRP FTST FUCOM FUCOMI FUCOMIP FUCOMP FUCOMPP FWAIT FXAM FXCH FXTRACT FYL2X FYL2XP1 WAIT)
- Zastaralé FPU instrukce(ESC FENI FNENI FDISI FNDISI FSETPM)
- Multimediální instrukce(ADDPD ADDPS ADDSD ADDSS ANDNPD ANDNPS ANDPD ANDPS CLFLUSH CMPEQPS CMPEQSS CMPLEPS CMPLESS CMPLTPS CMPLTSS CMPNEQPS CMPNEQSS CMPNLEPS CMPNLESS CMPNLTPS CMPNLTSS CMPORDPS CMPORDSS CMPPD CMPPS CMPSD CMPSS CMPUNORDPS CMPUNORDSS COMISD COMISS CVTDQ2PD CVTDQ2PS CVTPD2DQ CVTPD2PI CVTPD2PS CVTPI2PD CVTPI2PS CVTPS2DQ CVTPS2PD CVTPS2PI CVTSD2SI CVTSD2SS CVTSI2SD CVTSI2SS CVTSS2SD CVTSS2SI CVTTPD2DQ CVTTPD2PI CVTTPS2DQ CVTTPS2PI CVTTSD2SI CVTTSS2SI DIVPD DIVPS DIVSD DIVSS EMMS FEMMS FXRSTOR FXSAVE LDMXCSR LFENCE MASKMOVDQU MASKMOVQ MAXPD MAXPS MAXSD MAXSS MFENCE MINPD MINPS MINSD MINSS MOVAPD MOVAPS MOVD MOVDQ2Q MOVDQA MOVDQU MOVHLPS MOVHPD MOVHPS MOVLHPS MOVLPD MOVLPS MOVMSKPD MOVMSKPS MOVNTDQ MOVNTI MOVNTPD MOVNTPS MOVNTQ MOVQ MOVQ2DQ MOVSD MOVSS MOVUPD MOVUPS MULPD MULPS MULSD MULSS ORPD ORPS PACKSSDW PACKSSWB PACKUSWB PADDB PADDD PADDQ PADDSB PADDSW PADDUSB PADDUSW PADDW PAND PANDN PAUSE PAVGB PAVGUSB PAVGW PCMPEQB PCMPEQD PCMPEQW PCMPGTB PCMPGTD PCMPGTW PEXTRW PF2FW PF2ID PF2IW PFACC PFADD PFCMPEQ PFCMPGE PFCMPGT PFMAX PFMIN PFMUL PFNACC PFPNACC PFRCP PFRCPIT1 PFRCPIT2 PFRSQIT1 PFRSQRT PFSUB PFSUBR PI2FD PINSRW PMADDWD PMAXSW PMAXUB PMINSW PMINUB PMOVMSKB

PMULHRW PMULHUW PMULHW PMULLW PMULUDQ POR PREFETCH PREFETCHNTA PREFETCHT0 PREFETCHT1 PREFETCHT2 PREFETCHW PSADBW PSHUFD PSHUFHW PSHUFLW PSHUFW PSLLD PSLLDQ PSLLQ PSLLW PSRAD PSRAW PSRLD PSRLDQ PSRLQ PSRLW PSUBB PSUBD PSUBQ PSUBSB PSUBSW PSUBUSB PSUBUSW PSUBW PSWAPD PUNPCKHBW PUNPCKHDQ PUNPCKHQDQ PUNPCKHWD PUNPCKLBW PUNPCKLDQ PUNPCKLQDQ PUNPCKLWD PXOR RCPPS RCPSS RSQRTPS RSQRTSS SFENCE SHUFPD SHUFPS SQRTPD SQRTPS SQRTSD SQRTSS STMXCSR SUBPD SUBPS SUBSD SUBSS UCOMISD UCOMISS UNPCKHPD UNPCKHPS UNPCKLPD UNPCKLPS XORPD XORPS)

- Klíčová slova vyšší úrovně pro vytváření kódu(.BREAK .CONTINUE .ELSE .ELSEIF .ENDIF .ENDW .IF .REPEAT .UNTIL .UNTILCXZ .WHILE INVOKE USES)
- WinAsm direktivy(? @@ @B @F ADDR BASIC BYTE C CARRY? CASEMAP DWORD FAR FAR16 FLAT FORTRAN FWORD NEAR NEAR16 NONE NULL OVERFLOW? OWORD PARITY? PASCAL QWORD REAL10 REAL4 REAL8 SBYTE SDWORD SIGN? STDCALL SWORD SYSCALL TBYTE VARARG WORD ZERO?)
- Klíčová slova WinAsm (produkce bez kódu) & makra (%OUT .186 .286 .286C .286P .287 .386 .386C .386P .387 .486 .486P .586 .686 .686P .8086 .8087 .ALPHA .CODE .CONST .CREF .DATA .DATA? .DOSSEG .ERR .ERR1 .ERR2 .ERRB .ERRDEF .ERRDIF .ERRDIFI .ERRE .ERRIDN .ERRIDNI .ERRNB .ERRNDEF .ERRNZ .EXIT .FARDATA .FARDATA? .K3D .LALL .LFCOND .LIST .LISTALL .LISTIF .LISTMACRO .LISTMACROALL .MMX .MODEL .MSFLOAT .NO87 .NOCREF .NOLIST .NOLISTIF .NOLISTMACRO .RADIX .SALL .SEQ .SFCOND .STACK .STARTUP .TFCOND .TYPE .XALL .XCREF .XLIST .XMM @CatStr @code @CodeSize @Cpu @CurSeg @data @DataSize @Date @Environ @fardata @fardata? @FileCur @FileName @InStr @Interface @Line @Model @SizeStr @stack @SubStr @Time @Version @WordSize ALIAS ALIGN ASSUME CATSTR COMM COMMENT+ DB DD DF DOSSEG DQ DT DUP DW ECHO ELSE ELSEIF ELSEIF1 ELSEIF2 ELSEIFB ELSEIFDEF ELSEIFDIF ELSEIFDIFI ELSEIFE ELSEIFIDN ELSEIFIDNI ELSEIFNB ELSEIFNDEF END ENDIF ENDM ENDP ENDS endw EQ EQU EVEN EXITM EXTERN EXTERNDEF EXTRN FOR FORC GE GOTO GROUP GT HIGH HIGHWORD IF IF1 IF2 IFB IFDEF IFDIF IFDIFI IFE IFIDN IFIDNI IFNB IFNDEF INCLUDE- INCLUDELIB- INSTR IRP IRPC LABEL LE LENGTH LENGTHOF LOCAL LOW LOWWORD LROFFSET LT MACRO MASK MOD NAME NE OFFSET OPATTR OPTION ORG PAGE POPCONTEXT PROC PROTO PTR PUBLIC PURGE PUSHCONTEXT RECORD REPEAT REPT SEG SEGMENT SHORT SIZE SIZEOF SIZESTR STRUC STRUCT SUBSTR SUBTITLE SUBTTL TEXTEQU THIS TITLE- TYPE TYPEDEF UNION WHILE WIDTH)
- Obecné(#define #include ACCELERATORS ALT AUTOCHECKBOX AUTORADIOBUTTON BEGIN BITMAP BLOCK CAPTION CLASS COMBOBOX CONTROL CTEXT CURSOR DIALOGEX DISCARDABLE EDITTEXT END EXSTYLE FILEFLAGS FILEFLAGSMASK FILEOS FILESUBTYPE FILETYPE FILEVERSION FONT GROUPBOX ICON LISTBOX LTEXT MENU MENUITEM NOINVERT NOT POPUP PRODUCTVERSION PUSHBUTTON SEPARATOR SHIFT STRINGTABLE STYLE VALUE VERSIONINFO VIRTKEY)
- Objekty(^BS_3STATE ^BS_AUTO3STATE ^BS_AUTOCHECKBOX ^BS_AUTORADIOBUTTON ^BS_BITMAP ^BS_BOTTOM ^BS_CENTER ^BS_CHECKBOX ^BS_DEFPUSHBUTTON ^BS_GROUPBOX ^BS_ICON ^BS_LEFT ^BS_LEFTTEXT ^BS_MULTILINE ^BS_NOTIFY ^BS_OWNERDRAW ^BS_PUSHBUTTON ^BS_PUSHLIKE ^BS_RADIOBUTTON ^BS_RIGHT ^BS_RIGHTBUTTON ^BS_TEXT ^BS_TOP ^BS_USERBUTTON ^BS_VCENTER ^CBS_AUTOHSCROLL ^CBS_DISABLENOSCROLL ^CBS_DROPDOWN ^CBS_DROPDOWNLIST ^CBS_HASSTRINGS ^CBS_LOWERCASE ^CBS_NOINTEGRALHEIGHT ^CBS_OEMCONVERT ^CBS_OWNERDRAWFIXED

^CBS_OWNERDRAWVARIABLE ^CBS_SIMPLE ^CBS_SORT ^CBS_UPPERCASE ^CCS_ADJUSTABLE ^CCS_NODIVIDER ^CCS_NOPARENTALIGN ^CCS_NORESIZE ^CCS_TOP ^DS_3DLOOK ^DS_ABSALIGN ^DS_CENTER ^DS_CENTERMOUSE ^DS_CONTEXTHELP ^DS_CONTROL ^DS_FIXEDSYS ^DS_LOCALEDIT ^DS_MODALFRAME ^DS_NOFAILCREATE ^DS_NOIDLEMSG ^DS_SETFONT ^DS_SETFOREGROUND ^DS_SYSMODAL ^ES_AUTOHSCROLL ^ES_AUTOVSCROLL ^ES_CENTER ^ES_DISABLENOSCROLL ^ES_LEFT ^ES_LOWERCASE ^ES_MULTILINE ^ES_NOHIDESEL ^ES_NUMBER ^ES_OEMCONVERT ^ES_PASSWORD ^ES_READONLY ^ES_RIGHT ^ES_SAVESEL ^ES_SELECTIONBAR ^ES_SUNKEN ^ES_UPPERCASE ^ES_VERTICAL ^ES_WANTRETURN ^LBS_DISABLENOSCROLL ^LBS_EXTENDEDSEL ^LBS_HASSTRINGS ^LBS_MULTICOLUMN ^LBS_MULTIPLESEL ^LBS_NODATA ^LBS_NOINTEGRALHEIGHT ^LBS_NOREDRAW ^LBS_NOTIFY ^LBS_OWNERDRAWFIXED ^LBS_OWNERDRAWVARIABLE ^LBS_SORT ^LBS_STANDARD ^LBS_USETABSTOPS ^LBS_WANTKEYBOARDINPUT ^LVS_ALIGNLEFT ^LVS_ALIGNMASK ^LVS_ALIGNTOP ^LVS_AUTOARRANGE ^LVS_EDITLABELS ^LVS_ICON ^LVS_LIST ^LVS_NOCOLUMNHEADER ^LVS_NOLABELWRAP ^LVS_NOSCROLL ^LVS_NOSORTHEADER ^LVS_OWNERDRAWFIXED ^LVS_REPORT ^LVS_SHAREIMAGELISTS ^LVS_SHOWSELALWAYS ^LVS_SINGLESEL ^LVS_SMALLICON ^LVS_SORTASCENDING ^LVS_SORTDESCENDING ^LVS_TYPEMASK ^LVS_TYPESTYLEMASK ^MDIS_ALLCHILDSTYLES ^RBS_BANDBORDERS ^RBS_VARHEIGHT ^SBS_BOTTOMALIGN ^SBS_HORZ ^SBS_LEFTALIGN ^SBS_RIGHTALIGN ^SBS_SIZEBOX ^SBS_SIZEBOXBOTTOMRIGHTALIGN ^SBS_SIZEBOXTOPLEFTALIGN ^SBS_SIZEGRIP ^SBS_TOPALIGN ^SBS_VERT ^SS_BITMAP ^SS_BLACKFRAME ^SS_BLACKRECT ^SS_CENTER ^SS_CENTERIMAGE ^SS_GRAYFRAME ^SS_GRAYRECT ^SS_ICON ^SS_LEFT ^SS_LEFTNOWORDWRAP ^SS_NOPREFIX ^SS_RIGHT ^SS_SIMPLE ^SS_SUNKEN ^SS_USERITEM ^SS_WHITEFRAME ^SS_WHITERECT ^TBSTYLE_FLAT ^TBSTYLE_TOOLTIPS ^TCS_BOTTOM ^TCS_FOCUSNEVER ^TVS_HASBUTTONS ^TVS_HASLINES ^TVS_LINESATROOT ^TVS_SHOWSELALWAYS ^WS_BORDER ^WS_CAPTION ^WS_CHILD ^WS_CHILDWINDOW ^WS_CLIPCHILDREN ^WS_CLIPSIBLINGS ^WS_DISABLED ^WS_DLGFRAME ^WS_GROUP ^WS_HSCROLL ^WS_ICONIC ^WS_MAXIMIZE ^WS_MAXIMIZEBOX ^WS_MINIMIZE ^WS_MINIMIZEBOX ^WS_OVERLAPPED ^WS_OVERLAPPEDWINDOW ^WS_POPUP ^WS_POPUPWINDOW ^WS_SIZEBOX ^WS_SYSMENU ^WS_TABSTOP ^WS_THICKFRAME ^WS_TILED ^WS_TILEDWINDOW ^WS_VISIBLE ^WS_VSCROLL)

- Otatní instrukce(@ECHO CLS DEL DIR ERRORLEVEL EXIST OFF PAUSE REM GOTO IF NOT)
- Api struktury, funkce a konstanty

ZAČÍNÁME.

Vývojové software:

- Operační systém Windows 10/64 procesor AMD 4 jádra
- Visual studio 2017, konzolová aplikace v C#
- WinAsm studio v. 5.1.1.0 + MASM32
- Zpětný assembler W32DSM89
- Zpětný assembler PEBrowsePro

Popis:

- Program.cs -> `Načtení knihovny ASM1.dll do paměti`
- `Třída1 -> uint Test1(){const uint K0 = 10; return K0;}`
- `Třída2 -> uint Test2(){uint i1 = 100; return i1;// hodnota promněné na adrese i1}`
- `Třída3 -> double Test3(){uint i1 = 100; double i2 = PI; return i1 + i2;// hodnota promněné na adrese i1 a i2}`
- `Třída4 -> uint Test4(uint x1, uint x2, uint x3){uint i1 = 100; return x1+x2+x3+i1;// hodnota promněné na adrese i1}`
- `Třída5 -> void Test5(out int x1){vrací hodnotu promněné na adrese i1}`
- `Třída5 -> void Test6(out int x1, out int x2){vrací hodnoty promněné na adrese i1 a i3}`
- `Třída6 -> double Test7((UInt64 i, UInt64 j){return i+j; využívá MMX a koprocesor}`
- `Třída7 -> void Test8(ref uint uSize){vrací velikost řetězce;}`
- `Třída8 -> uint Test9(MStruc struc){Načte data ze struktury do struktury dll, tyto data ze struktury sečte a vrátí výsledek ;}`
- `Třída9 -> Test10(out mRect.x0, out mRect.x1, out mRect.y0, out mRect.y1){Načte data z vnitřní struktury assembleru do struktury v C#;}`
- `Třída10 -> int Test11(int* pole, uint umsize){Načte data z pole C# do pole v assembleru, vrací poslední prvek pole;Využívá pointry}`
- `Třída10 -> double Test11(double* pole, uint umsize){Načte data z pole C# do pole v assembleru, vrací poslední prvek pole; Využívá pointry }`
- `Třída11 -> Test13(int* pole){načte data z pole assembleru do pole v C#; }`
- `Třída12 -> Test14(double* pole){načte data z pole assembleru do pole v C#}`

Ukázky jsem popsal jak ve Visual studiu tak i v dll knihovně v assembleru. Možnost knihovnu využívat beze změn v C++ .

Zdrojové texty:

Ukázka assembleru
Ukázka Visual studia