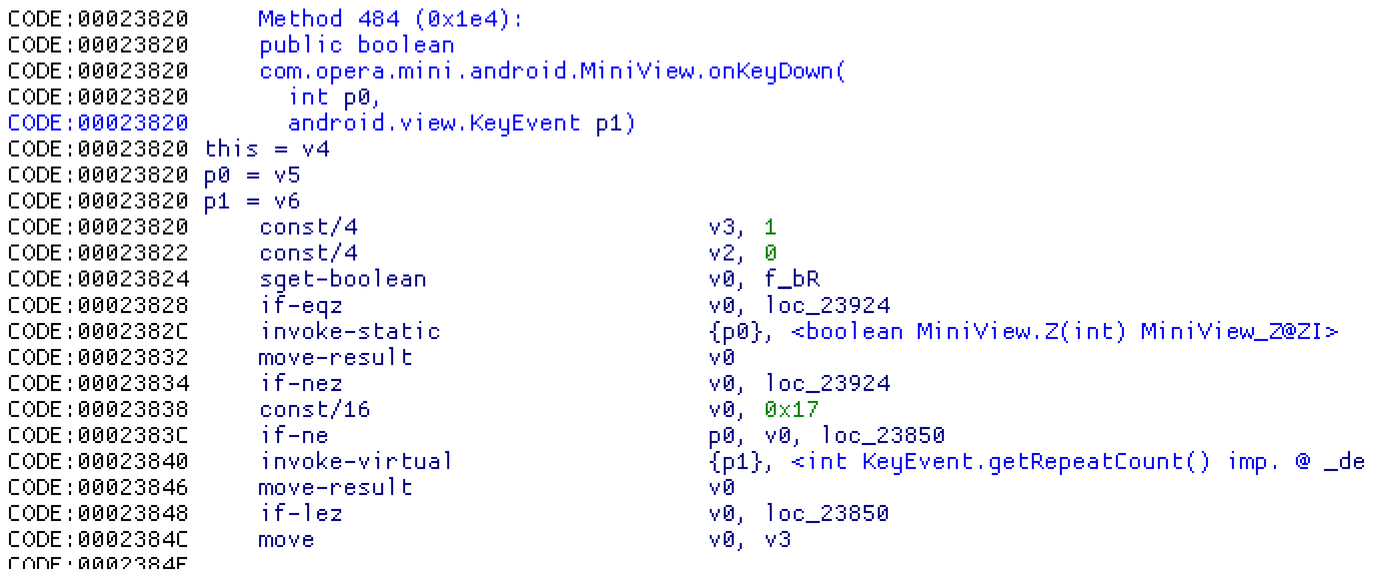
IDA Pro 6.1

What's new?

HIGHLIGHTS

- **Support for Android**

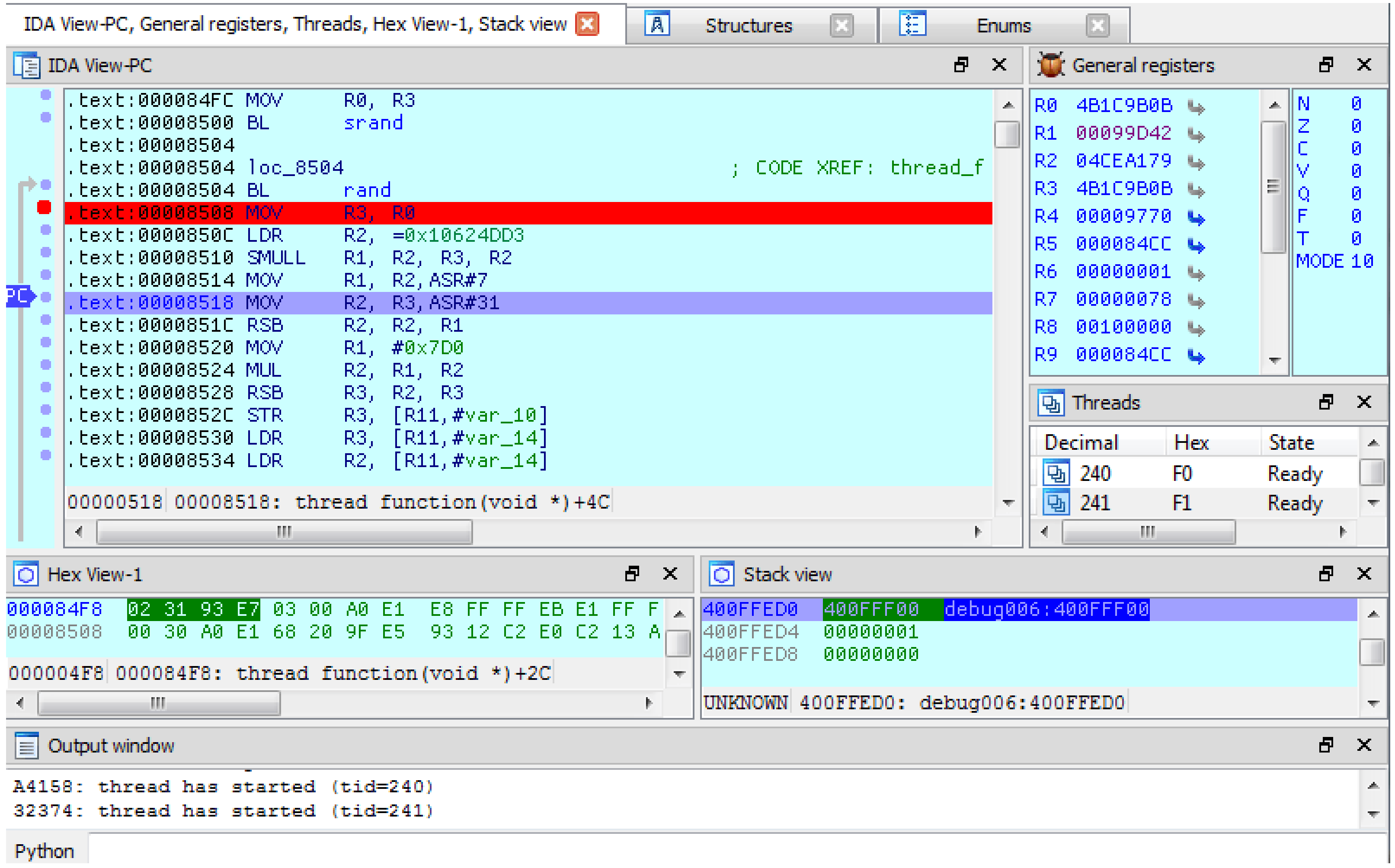
The long awaited Android support in IDA is ready!
The new version can disassemble Android bytecode (Dalvik).
An IDA user kindly contributed the processor module and file loader (thank you!)
A screenshot for your pleasure:



```
CODE:00023820     Method 484 (0x1e4):
CODE:00023820     public boolean
CODE:00023820     com.opera.mini.android.MiniView.onKeyDown(
CODE:00023820       int p0,
CODE:00023820       android.view.KeyEvent p1)
CODE:00023820 this = v4
CODE:00023820 p0 = v5
CODE:00023820 p1 = v6
CODE:00023820     const/4                         v3, 1
CODE:00023822     const/4                         v2, 0
CODE:00023824     sget-boolean                    v0, f_bR
CODE:00023828     if-eqz                          v0, loc_23924
CODE:0002382C     invoke-static                   {p0}, <boolean MiniView.Z(int) MiniView_Z@ZI>
CODE:00023832     move-result                     v0
CODE:00023834     if-nez                          v0, loc_23924
CODE:00023838     const/16                        v0, 0x17
CODE:0002383C     if-ne                           p0, v0, loc_23850
CODE:00023840     invoke-virtual                  {p1}, <int KeyEvent.getRepeatCount() imp. @ _de
CODE:00023846     move-result                     v0
CODE:00023848     if-lez                          v0, loc_23850
CODE:0002384C     move                            v0, v3
```

Dalvik disassembler is available in the Advanced Edition.

Native ARM code can be debugged too.
IDA Pro supports mixed ARM/Thumb code and can handle multithreaded applications:



```
.text:000084FC MOV     R0, R3
.text:00008500 BL      srand
.text:00008504
.text:00008504 loc_8504                          ; CODE XREF: thread_f
.text:00008504 BL      rand
.text:00008508 MOV     R3, R0
.text:0000850C LDR     R2, =0x10624DD3
.text:00008510 SMULL   R1, R2, R3, R2
.text:00008514 MOV     R1, R2,ASR#7
.text:00008518 MOV     R2, R3,ASR#31
.text:0000851C RSB     R2, R2, R1
.text:00008520 MOV     R1, #0x7D0
.text:00008524 MUL     R2, R1, R2
.text:00008528 RSB     R3, R2, R3
.text:0000852C STR     R3, [R11,#var_10]
.text:00008530 LDR     R3, [R11,#var_14]
.text:00008534 LDR     R2, [R11,#var_14]
00000518 00008518: thread function(void *)+4C
```

```
R0 4B1C9B0B    N    0
R1 00099D42    Z    0
R2 04CEA179    C    0
R3 4B1C9B0B    V    0
R4 00009770    Q    0
R5 000084CC    F    0
R6 00000001    T    0
R7 00000078    MODE 10
R8 00100000
R9 000084CC
```

Decimal	Hex	State
240	F0	Ready
241	F1	Ready

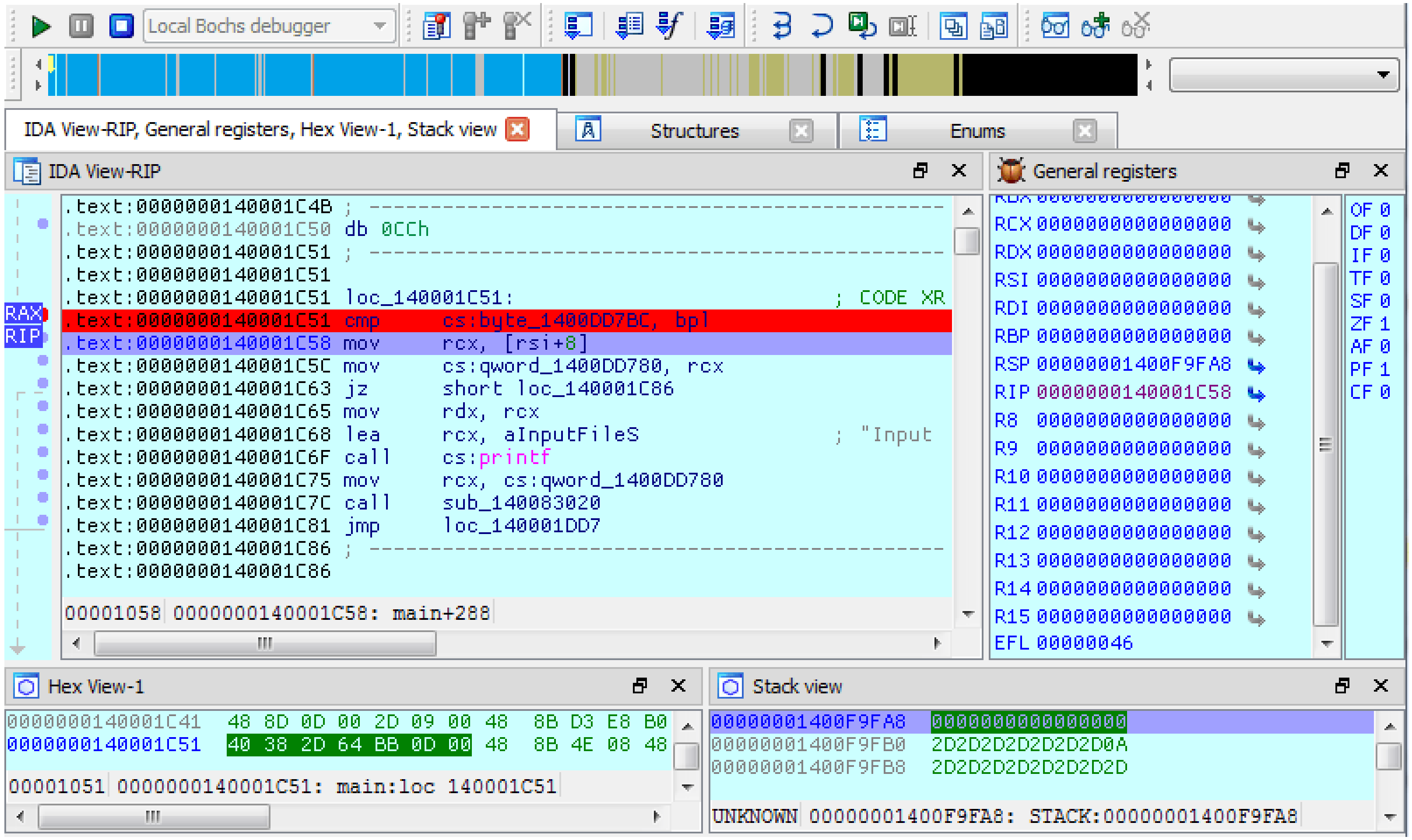
```
000084F8  02 31 93 E7 03 00 A0 E1  E8 FF FF EB E1 FF F
00008508  00 30 A0 E1 68 20 9F E5  93 12 C2 E0 C2 13 A
000004F8 000084F8: thread function(void *)+2C
```

```
400FFED0  400FFF00  debug006:400FFF00
400FFED4  00000001
400FFED8  00000000
UNKNOWN 400FFED0: debug006:400FFED0
```

```
A4158: thread has started (tid=240)
32374: thread has started (tid=241)
```

- **64-bit support for Bochs/GDB debuggers**

  The Bochs emulating debugger is very handy for small snippets of code. Before we could handle only 32-bit code but the new version adds 64-bit support.



```
.text:000000014001C4B ; ---------------------------------------------------
.text:000000014001C50 db 0CCh
.text:000000014001C51 ; ---------------------------------------------------
.text:000000014001C51
.text:000000014001C51 loc_140001C51:                          ; CODE XR
.text:000000014001C51 cmp     cs:byte_1400DD7BC, bpl
.text:000000014001C58 mov     rcx, [rsi+8]
.text:000000014001C5C mov     cs:qword_1400DD780, rcx
.text:000000014001C63 jz      short loc_140001C86
.text:000000014001C65 mov     rdx, rcx
.text:000000014001C68 lea     rcx, aInputFileS              ; "Input
.text:000000014001C6F call    cs:printf
.text:000000014001C75 mov     rcx, cs:qword_1400DD780
.text:000000014001C7C call    sub_140083020
.text:000000014001C81 jmp     loc_140001DD7
.text:000000014001C86 ; ---------------------------------------------------
.text:000000014001C86
00001058 0000000140001C58: main+288
```

```
RCX 0000000000000000     OF 0
RDX 0000000000000000     DF 0
RSI 0000000000000000     IF 0
RDI 0000000000000000     TF 0
RBP 0000000000000000     SF 0
RSP 00000001400F9FA8     ZF 1
RIP 0000000140001C58     AF 0
R8  0000000000000000     PF 1
R9  0000000000000000     CF 0
R10 0000000000000000
R11 0000000000000000
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000
EFL 00000046
```

```
0000000140001C41  48 8D 0D 00 2D 09 00 48  8B D3 E8 B0
0000000140001C51  40 38 2D 64 BB 0D 00 48  8B 4E 08 48
00001051 0000000140001C51: main:loc_140001C51
```

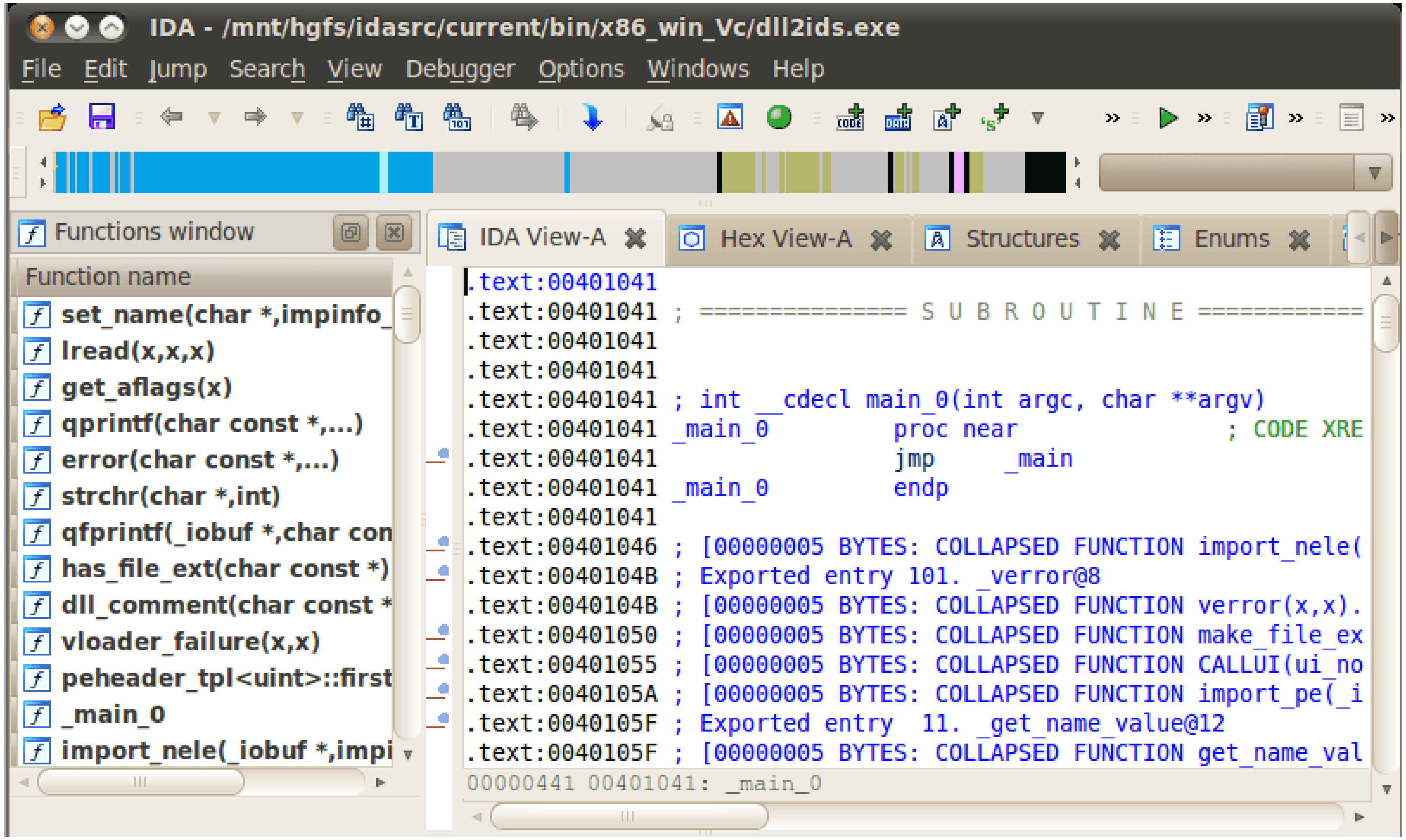
```
00000001400F9FA8  0000000000000000
00000001400F9FB0  2D2D2D2D2D2D0A
00000001400F9FB8  2D2D2D2D2D2D2D2D
UNKNOWN 00000001400F9FA8: STACK:00000001400F9FA8
```

Currently only the IDB mode is supported, later we plan to add PE+ support as well.

The GDBServer module adds x64 support and works with the latest VMWare versions.

- **Loading PDB files under Linux/MacOSX**

  Another long awaited feature is loading of PDB files under Linux and Mac OS X. Lack of this feature was a blocking factor for many Unix users. It is available now. Below is a screenshot made immediately after loading a PE file with PDB info on Linux:


IDA - /mnt/hgfs/idasrc/current/bin/x86_win_Vc/dll2ids.exe
File Edit Jump Search View Debugger Options Windows Help
Functions window
IDA View-A
Hex View-A
Structures
Enums

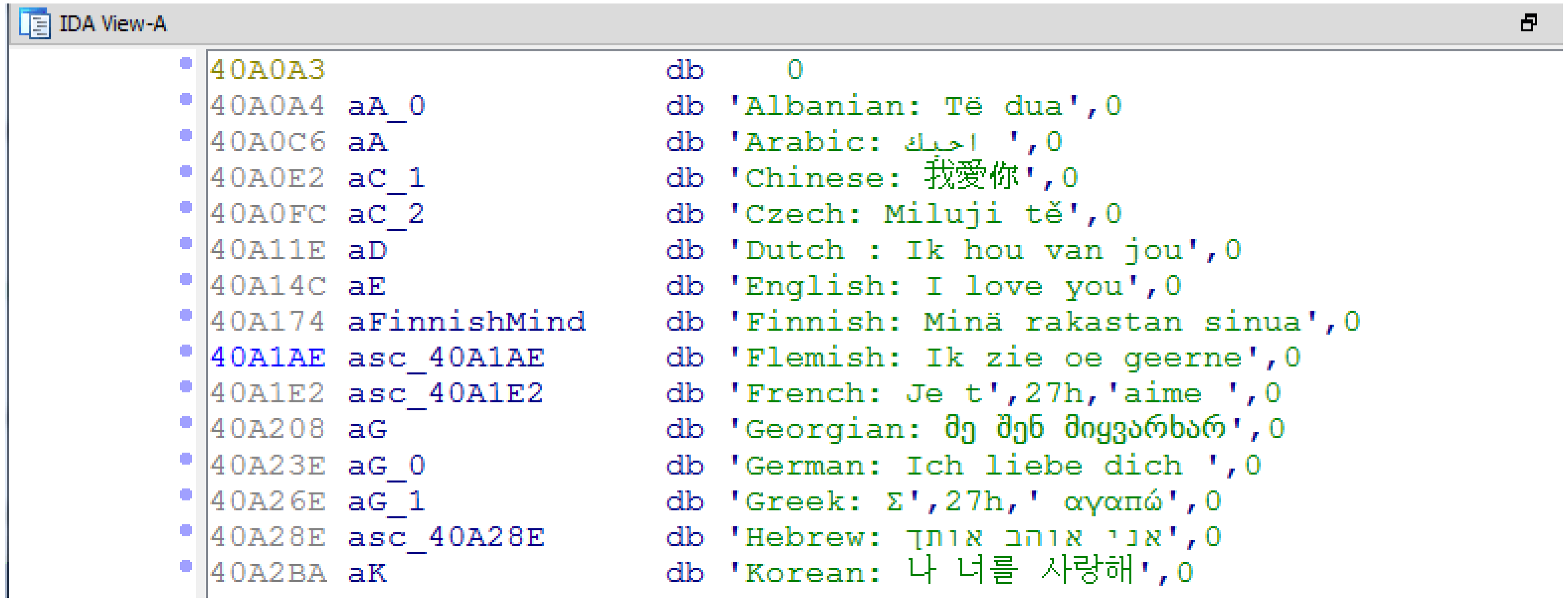
```
Function name
set_name(char *,impinfo_
lread(x,x,x)
get_aflags(x)
qprintf(char const *,...)
error(char const *,...)
strchr(char *,int)
qfprintf(_iobuf *,char con
has_file_ext(char const *)
dll_comment(char const *
vloader_failure(x,x)
peheader_tpl<uint>::first
_main_0
import_nele(_iobuf *,impi
```

```
.text:00401041
.text:00401041 ; =============== S U B R O U T I N E ===========
.text:00401041
.text:00401041
.text:00401041 ; int __cdecl main_0(int argc, char **argv)
.text:00401041 _main_0         proc near               ; CODE XRE
.text:00401041                 jmp     _main
.text:00401041 _main_0         endp
.text:00401041
.text:00401046 ; [00000005 BYTES: COLLAPSED FUNCTION import_nele(
.text:0040104B ; Exported entry 101. _verror@8
.text:0040104B ; [00000005 BYTES: COLLAPSED FUNCTION verror(x,x).
.text:00401050 ; [00000005 BYTES: COLLAPSED FUNCTION make_file_ex
.text:00401055 ; [00000005 BYTES: COLLAPSED FUNCTION CALLUI(ui_no
.text:0040105A ; [00000005 BYTES: COLLAPSED FUNCTION import_pe(_i
.text:0040105F ; Exported entry  11. _get_name_value@12
.text:0040105F ; [00000005 BYTES: COLLAPSED FUNCTION get_name_val
00000441 00401041: _main_0
```

We added PDB support to the win32 debugger server. The Unix version of IDA connects to a remote MS Windows computer (or local Wine session) and retrieves PDB information from it.

- **String encodings**

Not only Unicode, but other character encodings can be displayed in the disassembly listing. It is even possible to specify the encoding of individual strings:

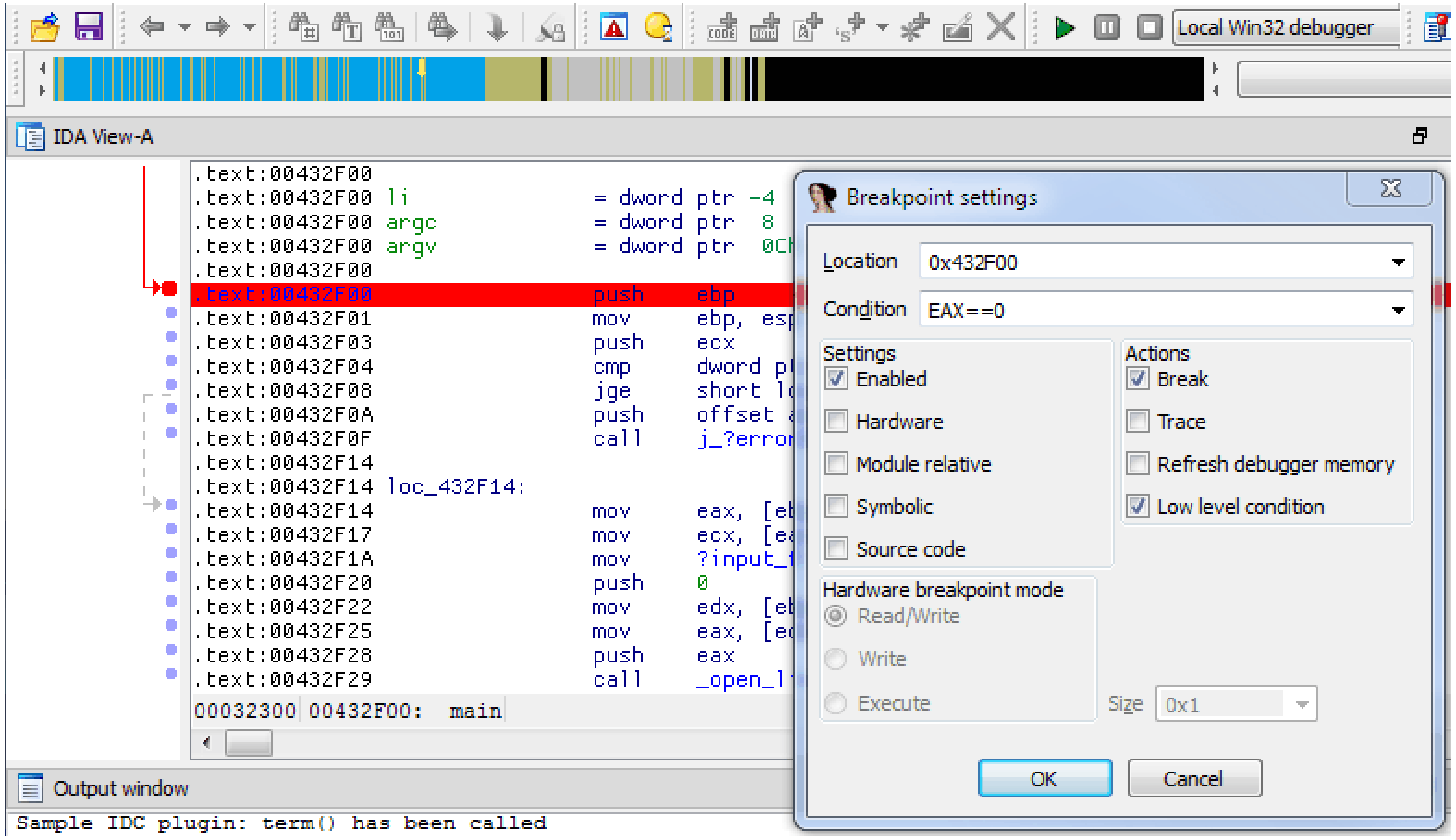


IDA View-A

```
40A0A3                    db    0
40A0A4 aA_0               db 'Albanian: Të dua',0
40A0C6 aA                 db 'Arabic: احبك ',0
40A0E2 aC_1               db 'Chinese: 我愛你',0
40A0FC aC_2               db 'Czech: Miluji tě',0
40A11E aD                 db 'Dutch : Ik hou van jou',0
40A14C aE                 db 'English: I love you',0
40A174 aFinnishMind       db 'Finnish: Minä rakastan sinua',0
40A1AE asc_40A1AE         db 'Flemish: Ik zie oe geerne',0
40A1E2 asc_40A1E2         db 'French: Je t',27h,'aime ',0
40A208 aG                 db 'Georgian: მე შენ მიყვარხარ',0
40A23E aG_0               db 'German: Ich liebe dich ',0
40A26E aG_1               db 'Greek: Σ',27h,' αγαπώ',0
40A28E asc_40A28E         db 'Hebrew: אני אוהב אותך',0
40A2BA aK                 db 'Korean: 나 너를 사랑해',0
```

- **Low level conditional breakpoints**

Conditional breakpoints can be very slow, especially during remote debugging. We addressed this problem by creating server side low level conditional breakpoints. They speed up the debugger tremendously. In our tests breakpoints were handled more than 20 times faster, even when running the remote server on the same computer as IDA Pro. Low level breakpoints are beneifical even for local debugging, so they are available for local debuggers too:


```
.text:00432F00
.text:00432F00 li              = dword ptr -4
.text:00432F00 argc            = dword ptr  8
.text:00432F00 argv            = dword ptr  0Ch
.text:00432F00
.text:00432F00                 push    ebp
.text:00432F01                 mov     ebp, esp
.text:00432F03                 push    ecx
.text:00432F04                 cmp     dword pt
.text:00432F08                 jge     short lo
.text:00432F0A                 push    offset a
.text:00432F0F                 call    j_?error
.text:00432F14
.text:00432F14 loc_432F14:
.text:00432F14                 mov     eax, [eb
.text:00432F17                 mov     ecx, [ea
.text:00432F1A                 mov     ?input_f
.text:00432F20                 push    0
.text:00432F22                 mov     edx, [eb
.text:00432F25                 mov     eax, [ec
.text:00432F28                 push    eax
.text:00432F29                 call    _open_l
00032300 00432F00: main
```

```
Sample IDC plugin: term() has been called
```

Breakpoint settings

Location: 0x432F00

Condition: EAX==0

Settings: [✓ Enabled] Hardware, Module relative, Symbolic, Source code

Actions: [✓ Break] Trace, Refresh debugger memory, [✓ Low level condition]

Hardware breakpoint mode: [✓ Read/Write] Write, Execute

Size: 0x1

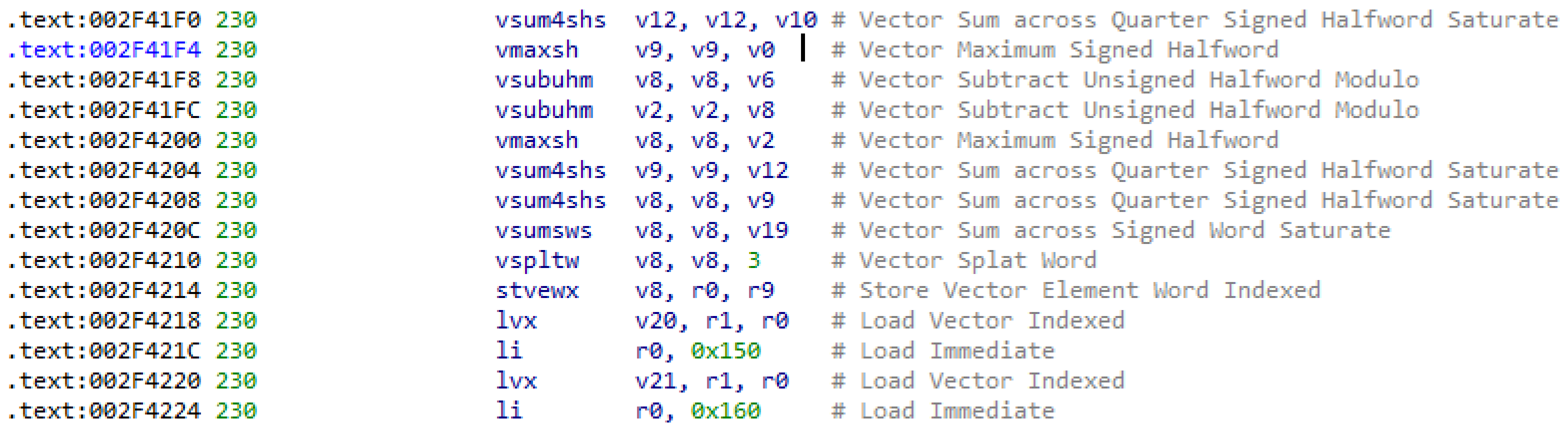
By the way, the screenshot shows other new breakpoint features: module relative, symbolic, and source code breakpoints. Unfortunately we had no time to finish source level debugging, so source level breakpoints are disabled for the moment.

- **Multithreaded debugger**

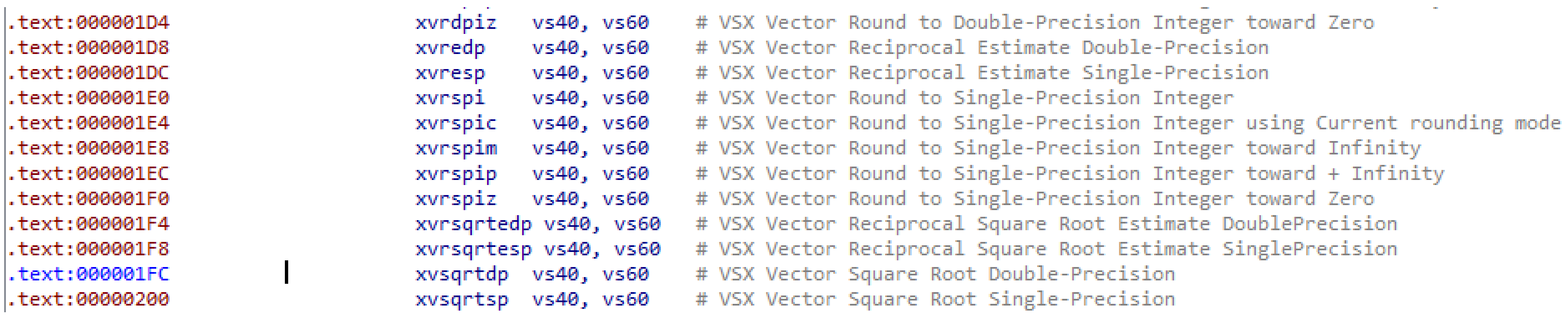
  Another measure to speed up the debugger: we made the debugger itself multithreaded. While this feature is not visible, it makes IDA Pro more responsive and enjoyable to use. Also we introduced multithread support in the IDA kernel. The kernel is still single threaded but it is much more friendly towards multithreaded plugins.

- **Power PC improvements**

  Many things were improved in the Power PC module. All the latest instructions defined by Power ISA were added, including Altivec and VSX extensions.

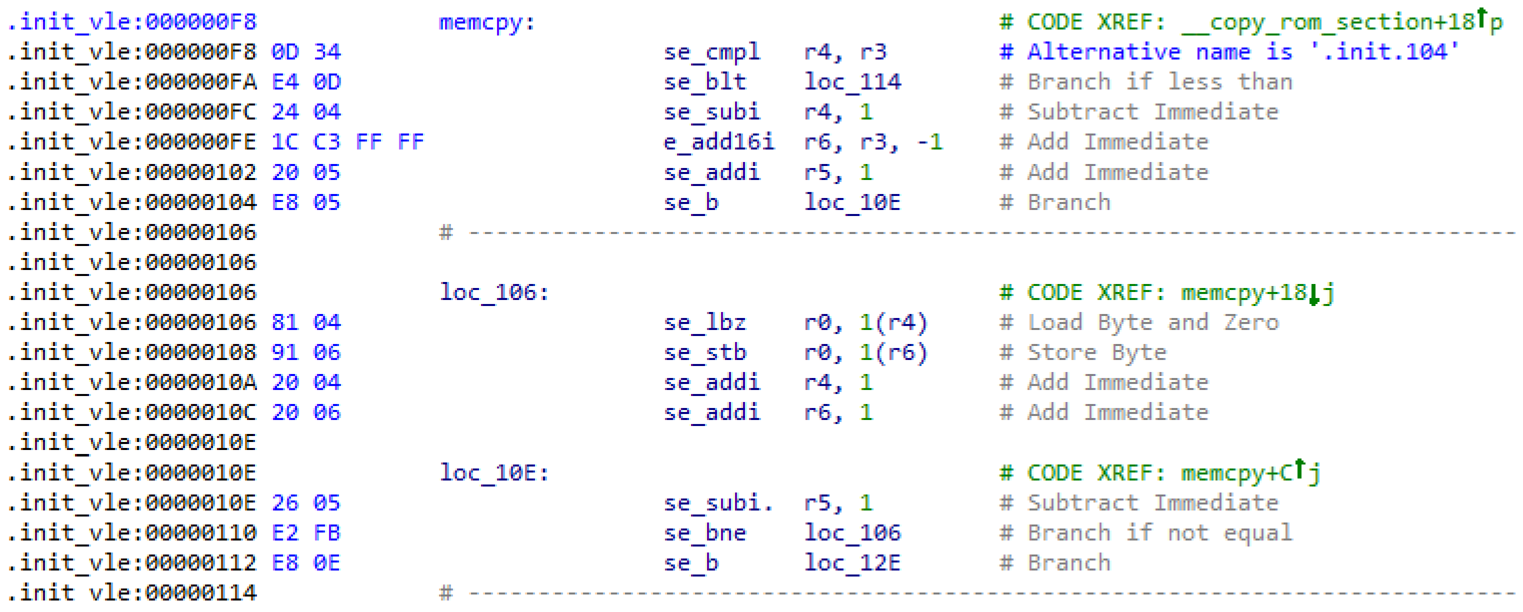


```
.text:002F41F0 230          vsum4shs  v12, v12, v10 # Vector Sum across Quarter Signed Halfword Saturate
.text:002F41F4 230          vmaxsh    v9, v9, v0    # Vector Maximum Signed Halfword
.text:002F41F8 230          vsubuhm   v8, v8, v6    # Vector Subtract Unsigned Halfword Modulo
.text:002F41FC 230          vsubuhm   v2, v2, v8    # Vector Subtract Unsigned Halfword Modulo
.text:002F4200 230          vmaxsh    v8, v8, v2    # Vector Maximum Signed Halfword
.text:002F4204 230          vsum4shs  v9, v9, v12   # Vector Sum across Quarter Signed Halfword Saturate
.text:002F4208 230          vsum4shs  v8, v8, v9    # Vector Sum across Quarter Signed Halfword Saturate
.text:002F420C 230          vsumsws   v8, v8, v19   # Vector Sum across Signed Word Saturate
.text:002F4210 230          vspltw    v8, v8, 3     # Vector Splat Word
.text:002F4214 230          stvewx    v8, r0, r9    # Store Vector Element Word Indexed
.text:002F4218 230          lvx       v20, r1, r0   # Load Vector Indexed
.text:002F421C 230          li        r0, 0x150     # Load Immediate
.text:002F4220 230          lvx       v21, r1, r0   # Load Vector Indexed
.text:002F4224 230          li        r0, 0x160     # Load Immediate
```



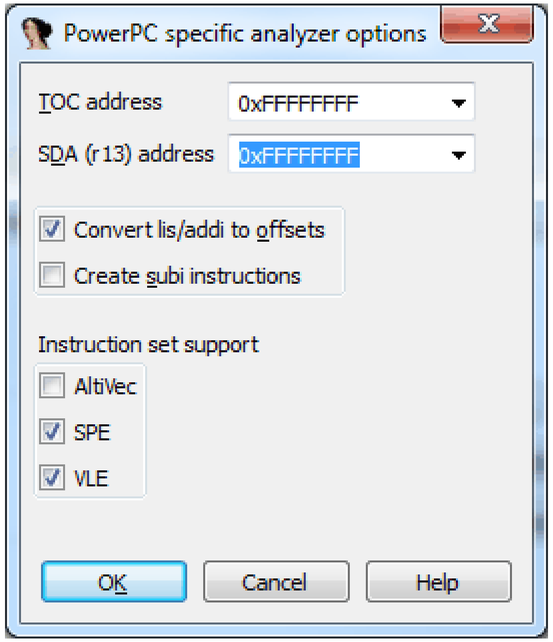
```
.text:000001D4              xvrdpiz    vs40, vs60    # VSX Vector Round to Double-Precision Integer toward Zero
.text:000001D8              xvredp     vs40, vs60    # VSX Vector Reciprocal Estimate Double-Precision
.text:000001DC              xvresp     vs40, vs60    # VSX Vector Reciprocal Estimate Single-Precision
.text:000001E0              xvrspi     vs40, vs60    # VSX Vector Round to Single-Precision Integer
.text:000001E4              xvrspic    vs40, vs60    # VSX Vector Round to Single-Precision Integer using Current rounding mode
.text:000001E8              xvrspim    vs40, vs60    # VSX Vector Round to Single-Precision Integer toward Infinity
.text:000001EC              xvrspip    vs40, vs60    # VSX Vector Round to Single-Precision Integer toward + Infinity
.text:000001F0              xvrspiz    vs40, vs60    # VSX Vector Round to Single-Precision Integer toward Zero
.text:000001F4              xvrsqrtedp vs40, vs60    # VSX Vector Reciprocal Square Root Estimate DoublePrecision
.text:000001F8              xvrsqrtesp vs40, vs60    # VSX Vector Reciprocal Square Root Estimate SinglePrecision
.text:000001FC              xvsqrtdp   vs40, vs60    # VSX Vector Square Root Double-Precision
.text:00000200              xvsqrtsp   vs40, vs60    # VSX Vector Square Root Single-Precision
```

Another addition is the VLE (Variable Length Encoding) instruction set, used in many embedded PPC processors.



```
.init_vle:000000F8             memcpy:                             # CODE XREF: __copy_rom_section+18↑p
.init_vle:000000F8 0D 34                      se_cmpl   r4, r3      # Alternative name is '.init.104'
.init_vle:000000FA E4 0D                      se_blt    loc_114     # Branch if less than
.init_vle:000000FC 24 04                      se_subi   r4, 1       # Subtract Immediate
.init_vle:000000FE 1C C3 FF FF                e_add16i  r6, r3, -1  # Add Immediate
.init_vle:00000102 20 05                      se_addi   r5, 1       # Add Immediate
.init_vle:00000104 E8 05                      se_b      loc_10E     # Branch
.init_vle:00000106             # ---------------------------------------------------------------------------
.init_vle:00000106
.init_vle:00000106             loc_106:                            # CODE XREF: memcpy+18↓j
.init_vle:00000106 81 04                      se_lbz    r0, 1(r4)   # Load Byte and Zero
.init_vle:00000108 91 06                      se_stb    r0, 1(r6)   # Store Byte
.init_vle:0000010A 20 04                      se_addi   r4, 1       # Add Immediate
.init_vle:0000010C 20 06                      se_addi   r6, 1       # Add Immediate
.init_vle:0000010E
.init_vle:0000010E             loc_10E:                            # CODE XREF: memcpy+C↑j
.init_vle:0000010E 26 05                      se_subi.  r5, 1       # Subtract Immediate
.init_vle:00000110 E2 FB                      se_bne    loc_106     # Branch if not equal
.init_vle:00000112 E8 0E                      se_b      loc_12E     # Branch
.init_vle:00000114             # ---------------------------------------------------------------------------
```

Also useful for embedded PPC is the new option to set a fixed value for the r13 register, commonly used as base for the small data area.



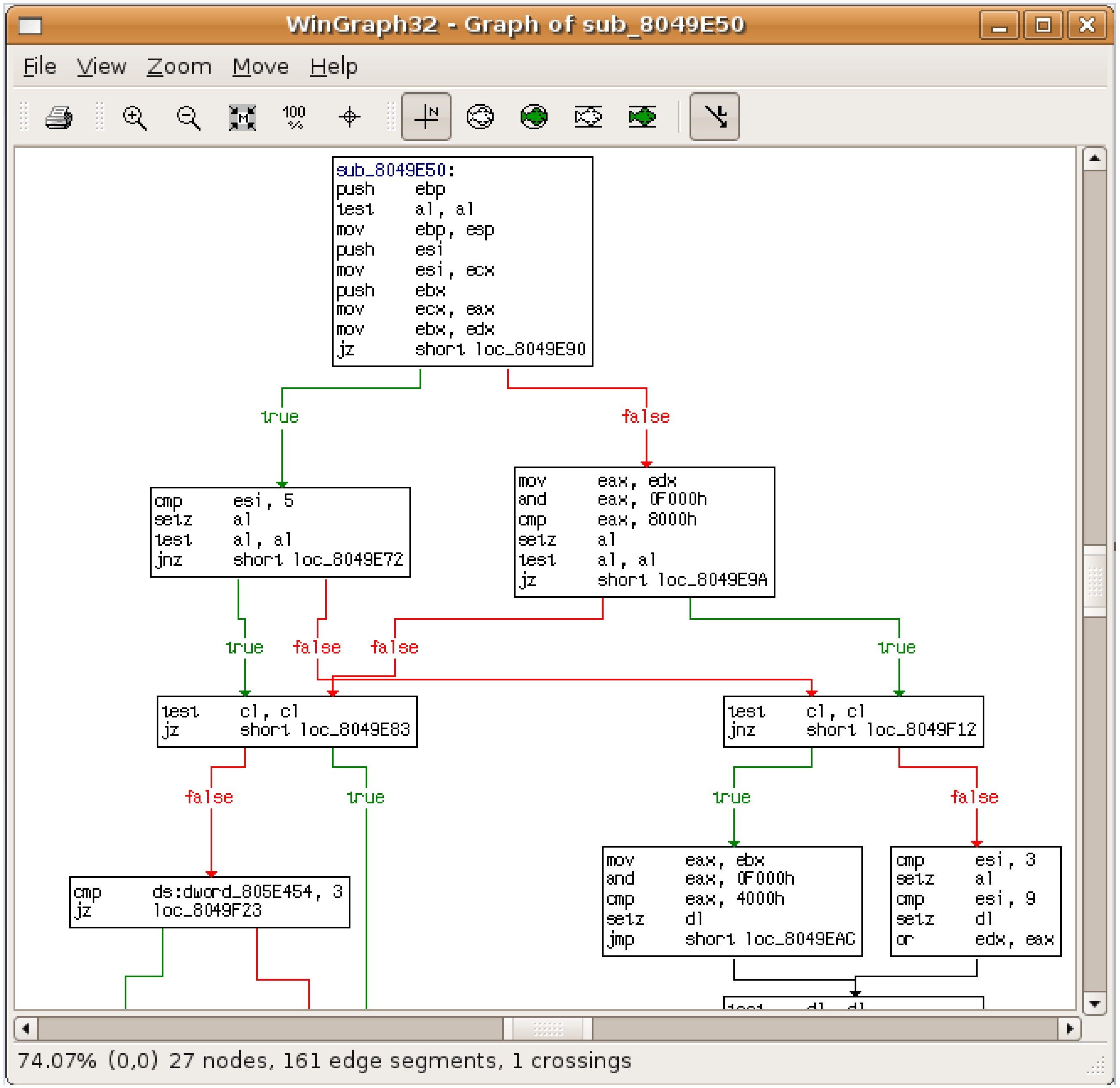
TOC address 0xFFFFFFFF
SDA (r13) address 0xFFFFFFFF

[✓] Convert lis/addi to offsets
[ ] Create subi instructions

Instruction set support
[ ] AltiVec
[✓] SPE
[✓] VLE

- **Wingraph is back!**

Chris Eagle has ported Wingraph32 to Qt framework (thanks!), and now we include it with all platforms, not just Windows.



```
sub_8049E50:
push    ebp
test    al, al
mov     ebp, esp
push    esi
mov     esi, ecx
push    ebx
mov     ecx, eax
mov     ebx, edx
jz      short loc_8049E90
```

```
cmp     esi, 5
setz    al
test    al, al
jnz     short loc_8049E72
```

```
mov     eax, edx
and     eax, 0F000h
cmp     eax, 8000h
setz    al
test    al, al
jz      short loc_8049E9A
```

```
test    cl, cl
jz      short loc_8049E83
```

```
test    cl, cl
jnz     short loc_8049F12
```

```
cmp     ds:dword_805E454, 3
jz      loc_8049F23
```

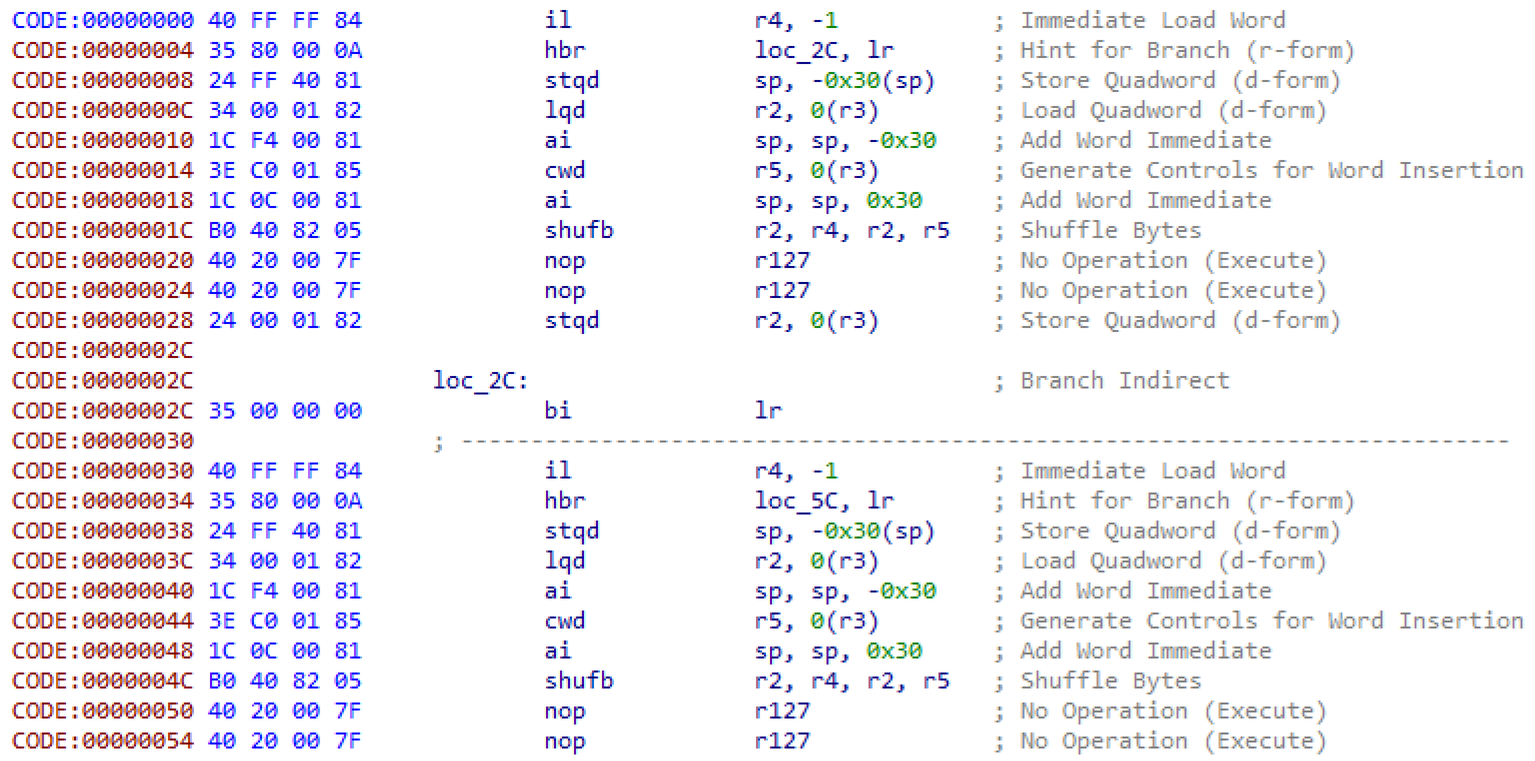
```
mov     eax, ebx
and     eax, 0F000h
cmp     eax, 4000h
setz    dl
jmp     short loc_8049EAC
```

```
cmp     esi, 3
setz    al
cmp     esi, 9
setz    dl
or      edx, eax
```

```
74.07% (0,0) 27 nodes, 161 edge segments, 1 crossings
```

- **SPU**

In addition to Dalvik, there is another new processor module in 6.1. It is the SPU (aka Synergistic Processing Unit) of the Cell BE processor, used in Sony PS3 console. This processor module is available in the Advanced Edition.



```
CODE:00000000 40 FF FF 84         il          r4, -1          ; Immediate Load Word
CODE:00000004 35 80 00 0A         hbr         loc_2C, lr      ; Hint for Branch (r-form)
CODE:00000008 24 FF 40 81         stqd        sp, -0x30(sp)   ; Store Quadword (d-form)
CODE:0000000C 34 00 01 82         lqd         r2, 0(r3)       ; Load Quadword (d-form)
CODE:00000010 1C F4 00 81         ai          sp, sp, -0x30   ; Add Word Immediate
CODE:00000014 3E C0 01 85         cwd         r5, 0(r3)       ; Generate Controls for Word Insertion
CODE:00000018 1C 0C 00 81         ai          sp, sp, 0x30    ; Add Word Immediate
CODE:0000001C B0 40 82 05         shufb       r2, r4, r2, r5  ; Shuffle Bytes
CODE:00000020 40 20 00 7F         nop         r127            ; No Operation (Execute)
CODE:00000024 40 20 00 7F         nop         r127            ; No Operation (Execute)
CODE:00000028 24 00 01 82         stqd        r2, 0(r3)       ; Store Quadword (d-form)
CODE:0000002C
CODE:0000002C                 loc_2C:                         ; Branch Indirect
CODE:0000002C 35 00 00 00         bi          lr
CODE:00000030                 ; ---------------------------------------------------------------------------
CODE:00000030 40 FF FF 84         il          r4, -1          ; Immediate Load Word
CODE:00000034 35 80 00 0A         hbr         loc_5C, lr      ; Hint for Branch (r-form)
CODE:00000038 24 FF 40 81         stqd        sp, -0x30(sp)   ; Store Quadword (d-form)
CODE:0000003C 34 00 01 82         lqd         r2, 0(r3)       ; Load Quadword (d-form)
CODE:00000040 1C F4 00 81         ai          sp, sp, -0x30   ; Add Word Immediate
CODE:00000044 3E C0 01 85         cwd         r5, 0(r3)       ; Generate Controls for Word Insertion
CODE:00000048 1C 0C 00 81         ai          sp, sp, 0x30    ; Add Word Immediate
CODE:0000004C B0 40 82 05         shufb       r2, r4, r2, r5  ; Shuffle Bytes
CODE:00000050 40 20 00 7F         nop         r127            ; No Operation (Execute)
CODE:00000054 40 20 00 7F         nop         r127            ; No Operation (Execute)
```

The detailed changelist is below:

PROCESSOR MODULES

+ DALVIK: new processor module (**Android Dalvik VM**)
+ SPU: new processor module (**Cell Broadband Engine Synergistic Processor Unit**); contributed by Felix Domke
+ ARM: turned on BL-as-jump analysis for ARM code. Before it was enabled only for Thumb code
+ AVR: added XMega instructions DES, LAC, LAS, LAT, XCH
+ AVR: decode eijmp and eicall instructions
+ C166: allow double-word and floating-point items in the disassembly
+ EBC: discover and comment function thunks
+ EBC: implemented instruction auto comments
+ EBC: made disassembly syntax closer to the one used in UEFI specification
+ EBC: trace stack pointer and create stack variables
+ MIPS: added support for Cavium Networks (**Octeon**) instructions
+ MIPS: added support for **MIPS64r2** instructions (doubleword bit manipulation)
+ MIPS: added support for Sony PSP (**Allegrex**) instructions
+ MIPS: added type system support (parameter identification and tracking)
+ MSP430: added support for MSP430X (20-bit) instructions
+ MSP430: resolve PC-relative (aka symbolic) addresses
+ PC: recognize prologs of VB6 applications (substantially speeds up analysis in some cases)
+ PC: show Intel conditional branch hints (prefixes 2E/3E)
+ PC: disassemble retn/retf opcodes with operand size override

+ PC: disassemble undocumented bswap ax instruction
+ PIC: automatically track changes to the PA0 status bit (bank selector) for 12-bit PIC processors
+ PIC: track values of BANK and PCLATH registers through the code flow - this improves disassembly of code that resides in multiple banks
+ PPC: added support for **AltiVec** instructions (including Cell BE extensions)
+ PPC: added support for **VLE (Variable Length Encoding)** instructions
+ PPC: it is now possible to specify a fixed base for the r13 register (small data area, often used in **embedded PPC** processors) and automatically convert all references to it
+ PPC: recognize switches used in 64-bit code with 32-bit addressing
+ PPC: updated GNU register names to reflect current conventions
+ SuperH: added option to disable immediates substitution (pre-6.0 behavior)
+ SuperH: it is now possible to use zero-offset structure fields in indirect register operands

FILE FORMATS

+ DEX: new loader for **Dalvik Executable** files
+ COFF: added support for TI MSP430 files
+ COFF: handle Xbox 360 files (PPCBE). Also small improvements for ARM and MIPS files
+ DOS: added support of loading of CodeView debug info for DOS .exe files
+ ELF: added support for **Cell SPU** files (no relocations supported yet)
+ ELF: added support for PPC64 relocations
+ ELF: added support for R_*_IRELATIVE relocations
+ ELF: Android prelinked files are detected and loaded at the correct address
+ ELF: handle files produced by Tasking C166/ST10 compiler
+ ELF: if data at entry point is not present in the section list, use program headers to load the missing code.
+ ELF: implemented some workarounds to load **Cisco IOS** files
+ ELF: PPC: handle files with VLE code sections and mark them as such
+ ELF: PPC: handle VLE relocations
+ ELF: support PSP PRX files
+ NE: support self-loading NE files
+ PE: added support for ARMv7 files

KERNEL

+ added support for arbitrarily big types in the type parser
+ added support for custom data formats inside structures
+ improved PIT (parameter identification and tracking) to better handle compex functions
+ improved the speed of rebasing the program
+ IDS: added ceddk.ids for Windows CE

FLIRT & TILS

+ FLIRT: added autodetection of the programs written in the D language
+ FLIRT: added **Digital Mars** FLIRT signatures
+ FLIRT: added FLIRT signatures for the Intel Compose XE 2011 ICL compiler
+ FLIRT: pcf: handle ARMv7 COFF files
+ FLIRT: pcf: handle PowerPC BE (Xbox 360) COFF files
+ FLIRT: pelf: i386 TLS related relocations require special handing because the linker modifies instructions
+ FLIRT: pelf: added support for SuperH files

+ prepared new mssdk til files based on the Windows SDK 7.0a

SCRIPTS & SDK

+ IDAPython: added PluginForm class which adds the possibility to extend the UI with **PyQt or PySide**
+ IDAPython: Python statement execution and script timeout are configurable
+ IDAPython: added AskUsingForm() with embedded choosers support
+ IDAPython: added idautils.DecodePreviousInstruction() / DecodePrecedingInstruction()
+ IDAPython: added idc.BeginTypeUpdating() / EndTypeUpdating() for fast batch type update operations
+ IDAPython: added more IDP callbacks
+ IDAPython: added UI_Hooks with a few notification events
+ IDAPython: added process_ui_action()
+ IDAPython: better handling of ea_t in 32/64bit
+ IDAPython: Added netnode.index() method
+ IDC: added DbgRead/DbgWrite functions to access the debuggee memory directly
+ IDC: added highlevel breakpoint management class
+ IDC: added get_nsec_stamp()
+ IDC: added SetBptCndEx(), unlink(), rename(), mkdir() functions
+ IDC: added ProcessUiAction()
+ IDC: added sp register change points functions
+ SDK: added begin_type_updating() / end_type_updating() functions to allow faster updates to the types
+ SDK: added get_strmem2()
+ SDK: added support for asynchronious execute_sync() calls (MFF_NOWAIT)
+ SDK: added system-independent functions to work with pipes
+ SDK: added process_ui_command()
+ SDK: IDC engine is thread safe. However, multiple threads should not access/modify the same IDC variables, this is not supported
+ SDK: implemented choosers embeddable in forms
+ SDK: introduced get_full_data_elsize(), useful for wide-byte processors
+ SDK: introduced qisspace and similar functions to avoid problems with signed chars
+ SDK: introduced thread-local functions to handle error codes (set_qerrno/get_qerrno)
+ SDK: renamed init_process() to launch_process()
+ SDK: trim() removes all whitespace at the string end (before it was removing only spaces and tabs)

USER INTERFACE

+ **wingraph** for Qt, kindly shared by Chris Eagle
+ graph: respect the selection priority when displaying nodes and clicking on them
+ added "New instance" menu entry
+ added "Produce header file from local types" menu entry
+ added 'Unsort' command in choosers
+ added Select All/Deselect All context menu items to the structure offset dialog
+ allow to open any file by drag&dropping on IDA icon (previously only .idb files could be opened this way)
+ allow multiple selection in the recent scripts window
+ enabled multi-selection in the Strings List
+ improved 'rename register' dialog box
+ improved the rebase dialog
+ it is now possible to set a string's encoding from "Setup ASCII types" dialog (Alt-A)
+ pressing Ctrl+K will always jump to the stack variable under the cursor (even if stack window is already open)
+ qt: implemented functions to load/free custom icons to be used in contexts like the chooser

+ qt: improved scroll speed
+ qt: improved the windows list dialog (**Ctrl-Tab**)
+ qt: improved wait dialog speed
+ txt: implemented the Load Binary dialog
+ gui: this is the last release of VCL based idag.exe

DEBUGGER

+ added support for server-side **low-level breakpoint conditions**. Such conditions are evaluated on the remote computer, without causing any network traffic
+ added support for **Android** debugger target (native ARM only)
+ Bochs: added debugging support for **64bit code snippets**
+ Bochs: path to Bochs can now only be specified through IDA.CFG or PATH environment variable
+ GDB: added support for **debugging x64 code**
+ GDB: enabled "Run external program" option for Linux and OS X
+ GDB: handle read/write memory breakpoints if the stub supports them (e.g. VMWare)
+ GDB: improved debugging of MIPS16 code
+ Windbg: added support for the 'reconnect' option
+ Windbg: the debugging tools path can now only be specified through IDA.CFG or PATH environment variable

BUGFIXES
----

all bugfixes since the initial release of IDA 6.0:

BUGFIX: 'open file' dialog in idal was not sorting directories to the end of the list
BUGFIX: "copy structure" and "create structure from data" commands should copy the type information
BUGFIX: "Produce HTML file" functionality was susceptible to Javascript injection vulnerability
BUGFIX: .NET: opcode "constrained." was decoded incorrectly
BUGFIX: a variable name was accepted and ignored in "enum : int mystupidvarname"
BUGFIX: Adding an enum or struct from an already parsed typeinfo that does not correspond to an enum or struct would cause IDA to crash
BUGFIX: AIF: a specially crafted file could trigger arbitrary code execution
BUGFIX: appcall was failing on high addresses
BUGFIX: arm debuggers could lose control after stepping over pop {pc} insn (the target address was calculated incorrectly)
BUGFIX: ARM: ARM processor module was ignoring the "Mark typical code sequences as code" autonalysis setting
BUGFIX: ARM: in rare cases, bogus data interpreted as code could crash IDA with a stack overflow
BUGFIX: ARM: TBB/THB switch constructs were marked up incorrectly, leading to incorrect decompilation in Hex-Rays
BUGFIX: Bochs debugger plugin was hanging if bochsdbg was terminated due to a crash or VM OS shutdown
BUGFIX: Bochs debugger run menu item was not present in the list when no database is opened
BUGFIX: change_storage_type() was creating sparse flags very inefficiently in some cases
BUGFIX: coff/psx/geos loaders had an integer overflow bug in memory allocation
BUGFIX: COFF: a specially crafted file could trigger a heap overflow
BUGFIX: COFF: relocation REL_ARM_SECREL was not handled
BUGFIX: convert_codepage() was prone to buffer overflow exploits
BUGFIX: debugger / stack view address size was incorrect when debugging without an initial database
BUGFIX: debugger options were not restored if the database had no segments
BUGFIX: demangler: for Borland names do not unmangle procedure/template name when it contains >= 36 arguments
BUGFIX: EBC: indirect register operands without index were disassembled incorrectly

BUGFIX: ELF: import list for ELF files was attaching one of the linked .so files to all imports. Since ELF imports use global namespace, don't attach a library name to them.
BUGFIX: ELF: some SuperH files marked as "sh2a-or-sh3" were loaded incorrectly
BUGFIX: ELF: symbols were not loaded from some ELF files with non-standard section names
BUGFIX: enums with custom size were printed incorrectly and thus their names were lost after editing in "Local Types" list
BUGFIX: EPOC: a specially crafted file could cause a heap overflow
BUGFIX: Executing a script with File/Script file could add a wrong file name to the recent scripts list in some cases
BUGFIX: exiting IDA at the very start of debugging would lead to an internal error
BUGFIX: EXPLOAD: a specially crafted file could trigger a heap overflow
BUGFIX: fixed a longstanding 'nrect(..)' internal error that was occurring in rare cases
BUGFIX: fixed a very rare btree error (there was no logic to handle a double page overflow during a key deletion; only single page overflows were handled)
BUGFIX: fixed DLL hijacking exploit for windmp, windbg and pdb plugins
BUGFIX: Fixed multiple execution of the same sync request for blocking operations like launching modal dialog as the chooser.
BUGFIX: fixed occasional crash when opening the breakpoint list
BUGFIX: GDB: for big-endian ARM targets, PSR register value was sent in wrong byte order
BUGFIX: get_flags_novalue() could fail in some rare circumstances (when the debugger is running and a previously defined memory area disappears it could return garbage)
BUGFIX: header() callback was not working in scripted processor modules
BUGFIX: HEX files for wide-byte processors (e.g. AVR) were loaded at a wrong address if a start address record was present
BUGFIX: hardware breakpoints were not deleted correctly on OSX
BUGFIX: hppa: delay slots were calculated wrongly while applying type information to function calls
BUGFIX: IDA could interr when parsing a C header with the same type name as in a loaded standard type library.
BUGFIX: IDA would crash on Mac / Linux when exiting after the user has attached to a process without an initial database
BUGFIX: IDA could fail to detect some address space overflows (when too many big segments were created)
BUGFIX: idag -S switch was not working properly for file names with spaces
BUGFIX: IDC: open_loader_input() would leak linput_t handles
BUGFIX: IDC: SetSegmentAttr() could crash if passed wrong segment address
BUGFIX: implemented the "CLOSED_BY_ESC" configuration parameter for idaq
BUGFIX: in some cases, trying to focus the recent scripts window with Alt-F9 after having added a new script may not work properly unless the window is closed and reopened
BUGFIX: in some cases, when the cursor was on a structure member, IDA was proposing to rename the whole structure instead of the member
BUGFIX: integer overflow was possible in qcalloc()
BUGFIX: get_chooser_object() was broken in the text UI
BUGFIX: it was impossible to launch idaq64 with command line arguments on OS X
BUGFIX: it was impossible to remotely debug 32-bit programs from IDA64
BUGFIX: it was not possible to rename stack variables from the listing at the start of the function in PowerPC files
BUGFIX: it was possible to rename a register to a name with a space
BUGFIX: it was possible to specify malicious plugins to be autorun at the database opening time; introduced an option to enable/disable autorun plugins and set it to 'off' by default
BUGFIX: kernel: on big-endian processors, float values in collapsed (terse) structures were displayed wrong
BUGFIX: OS X debugger could fail if a hardware breakpoint and software breakpoint occurred at the same address simultaneously
BUGFIX: Mach-O: buffer overflow when loading Mach-O files with corrupted export information
BUGFIX: Mach-O: some corrupted files could cause IDA to crash with out-of-memory exception
BUGFIX: MSP430: sub and subc instructions were swapped

BUGFIX: on very rare occasions the graph overview window would process a paint event after having closed a file and access invalid memory
BUGFIX: opcode bytes were not always printed along with the insruction for TMS320C6
BUGFIX: PatchByte() and similar functions were not refreshing the disassembly view
BUGFIX: PC: pushfq and some other 64-bit stack operating instructions were not handled during stack pointer tracing
BUGFIX: PC: some memory references were displayed incorrectly in TASM Ideal mode (for example: [name[eax*4], note the second bracket)
BUGFIX: PC: some switch constructs were marked up incorrectly by IDA leading to wrong decompilation in Hex-Rays
BUGFIX: PC: the wait instruction could be printed with erroneous prefix byte which belonged to the following non-FPU instruction
BUGFIX: PDB plugin would crash on certain input files
BUGFIX: PEF: a specially crafted file could trigger heap overflow
BUGFIX: PPC: immediate operands for some binary instructions (ori, xori, etc.) were incorrectly displayed as signed values
BUGFIX: pressing Esc in a form with Yes/No/Cancel buttons would return 0 (must return -1)
BUGFIX: qt: added graphs toolbar and implemented prev/next toolbar menu
BUGFIX: qt: adding items to the top-level Edit/Jump/Search menus of enum and struct views would fail
BUGFIX: qt: adding menu items to the Edit menu could fail if it was invisible
BUGFIX: qt: after executing custom menu items from the menu by keyboard on Windows the current focus might be lost
BUGFIX: qt: breakpoint dialog was missing the "Refresh debugger memory" option
BUGFIX: qt: call the sizer() callback in the chooser only for refresh events
BUGFIX: qt: calling msg() from chooser's sizer() and getl() callbacks would crash idaq
BUGFIX: qt: correctly associate the idb extension on Windows
BUGFIX: qt: correctly restore arrows width in disassembly when loading a saved database
BUGFIX: qt: correctly restore focus on Windows after having executed an action in the menu (make sure the focus doesn't remain on the menu)
BUGFIX: qt: correctly restore focus with floating docks under Linux
BUGFIX: qt: correctly restore row selection in a sorted list in a chooser after an edit action
BUGFIX: qt: correctly update navigation history when clicking on an edge in graph mode
BUGFIX: qt: could crash when calling Exit() or idaapi.qexit() from scripts
BUGFIX: qt: could sometimes crash when renaming structure members from the disassembly
BUGFIX: qt: couldn't close dock tabs with the middle mouse button
BUGFIX: qt: debug actions were not updated when an instant debugging session ended
BUGFIX: qt: docking the graph overview in a tab view would lead to problems
BUGFIX: qt: don't ask twice in the Save File dialog to overwrite an existing file
BUGFIX: qt: don't show the Sync submenu in a stackview.
BUGFIX: qt: fixed -t command line switch behavior
BUGFIX: qt: fixed a problem with the shortcut system on mac
BUGFIX: qt: fixed case insensitive completer for input fields in forms.
BUGFIX: qt: fixed incremental search in choosers
BUGFIX: qt: fixed some minor graph rendering glitches
BUGFIX: qt: fixed specific group box frame drawing issue in forms
BUGFIX: qt: fixed the not working Follow in Dump command in the hex editor
BUGFIX: qt: fixed the setting of the initial focus in forms
BUGFIX: qt: fixed wait dialog problems on Linux
BUGFIX: qt: fixed wrong behavior of the numpad Enter
BUGFIX: qt: implemented alternative key to Ins on OS X
BUGFIX: qt: implemented blinking arrows in graph view when debugging
BUGFIX: qt: implemented HELP/ENDHELP in custom forms
BUGFIX: qt: implemented external help support for Windows
BUGFIX: qt: implemented FORM_PERSIST flag in open_tform
BUGFIX: qt: implemented auto-indentation in comment/script dialog

BUGFIX: qt: implemented set_dock_pos()
BUGFIX: qt: improved quality of graph rendering in zoom mode
BUGFIX: qt: improved shortcuts behavior on OS X
BUGFIX: qt: input fields in forms were not generating change events
BUGFIX: qt: it was not possible to open Struct window if a function stack window was open before
BUGFIX: qt: it was not possible to overwrite menu label shortcuts with user created shortcuts
BUGFIX: qt: mac: fixed minor glitch in drawing the cursor
BUGFIX: qt: make sure that after closing an idb all actions are refreshed.
BUGFIX: qt: message box shortcuts now work without pressing Alt
BUGFIX: qt: Produce HTML file was using wrong font
BUGFIX: qt: remember the position of the cursor in the struct view when saving database
BUGFIX: qt: reset desktop was not working properly sometimes on mac
BUGFIX: qt: restore focus after a dock drag operation
BUGFIX: qt: select current thread in debug mode
BUGFIX: qt: set_custom_viewer_popup and add_custom_viewer_popup work now even on non-TCustomViewer IDA memos
BUGFIX: qt: set_focused_field in forms would fail at initialization time
BUGFIX: qt: shortcuts for custom data types were not set correctly
BUGFIX: qt: show lock status on the Highlight toolbar button
BUGFIX: qt: show text cursor in the output window
BUGFIX: qt: some entries of the quick open dialog may fail because of wrong context
BUGFIX: qt: the '.' shortcut now activates the command line when the current focus is in the output window already
BUGFIX: qt: the Cancel button in forms was not returning -1
BUGFIX: qt: the chooser now accepts Home and End even from the numpad and acts the same when Ctrl is pressed. Also, the fast search is cleared when pressing these keys
BUGFIX: qt: the Del shortcut in the watchlist was not always working
BUGFIX: qt: the jump to neighbor node shortcuts were working only on mac
BUGFIX: qt: the main window would not show when starting to debug from the command line
BUGFIX: qt: UI would hang if typing a non-matching letter at the last item of a chooser
BUGFIX: qt: was eating too much cpu time when idle
BUGFIX: qt: was not using system locale to convert text data, so localized comments, file paths, etc. were not displayed properly
BUGFIX: qt: would hang if trying to incrementally search for an item in a chooser without having a selection first
BUGFIX: qt: would not revert to default stack variable name if the name was cleared
BUGFIX: text: chooser was leaking memory on destruction
BUGFIX: right click menu was not listing structures with unions and unions as creatable variable types
BUGFIX: rebase_program() was not updating the xref cache, so cross-references could be wrong immediately after rebasing
BUGFIX: Recent scripts window displays blank script file names if no database was open
BUGFIX: result of custom_ana notification was not handled properly, breaking some processor extension plugins.
BUGFIX: IDC: Qword() was not returning 64bit values in IDA32
BUGFIX: SBN: a specially crafted input file could lead to buffer overflow
BUGFIX: SDK: get_default_reftype() was not working correctly for processors with wide bytes
BUGFIX: The IDC engine was failing on __get/setattr__ functions for IDC objects if those functions were registered from the SDK via set_idc_getattr()/set_idc_setattr()
BUGFIX: SDK: launch_process(formerly init_process) function did not handle properly quoted command-line arguments on Linux and OS X
BUGFIX: SDK: OutMnem() did not work properly for values of 'width' different from default
BUGFIX: set_auto_plugins() was allowing arbitrary plugin path (including UNC) thus leading to malicious code execution
BUGFIX: shortcuts for custom graph actions were not working
BUGFIX: some win32 OEM keys were incorrectly converted to qt codes
BUGFIX: SPARC: R_SPARC_JMP_SLOT relocation was not processed properly in 64-bit files
BUGFIX: SPARC: some WR instructions were decoded incorrectly in V8 mode
BUGFIX: stack view was always using 64-bit addressing in IDA64, even for 32-bit programs

BUGFIX: Symbian debugger was not clearing the old process list before retrieving a new one.
BUGFIX: text version: in the 'create array' dialog box, it was impossible to switch back from binary indexes to any other number base
BUGFIX: The "OK" button in the Choose Structure window was not being enabled when a struct is selected for the first time
BUGFIX: The debugger popup menu to open a register class window was not working
BUGFIX: type parser: type definitions without the terminating ; were silently ignored at the end of the input file (or line)
BUGFIX: ui: a byte with value 0xFF was not printed as a character, even if it was in the AsciiStringChars list.
BUGFIX: ui: avoid duplicate upper/lower-case history entries on Windows
BUGFIX: ui: binary search was searching for wrong pattern if a too long number was entered
BUGFIX: ui: buffer overflow could happen when trying to display a very long string
BUGFIX: ui: Calculator (Shift-/ key) was picking up wrong value from disassembly on OSX and Linux
BUGFIX: ui: fill the Edit->Plugins menu with PLUGIN_FIX plugins when no IDB is open
BUGFIX: ui: IDA could hang while trying to display a hint in some rare situations
BUGFIX: ui: IDA could lock up for some time while trying to display a hint.
BUGFIX: ui: in the 'User Offset' dialog, set initial focus to the 'Base address' field
BUGFIX: ui: the cross reference list would show empty if already open for the same target
BUGFIX: unix: unicode strings were not handled correctly for some locales
BUGFIX: while undecorating names try to preserve the suffix after '@'. remove it only in some special cases
BUGFIX: Windbg debugging mode option was not saved in instant debugging mode
BUGFIX: zero values were always represented as "0" in terse structure representations, even if they should be replaced by offsets or enums or something else