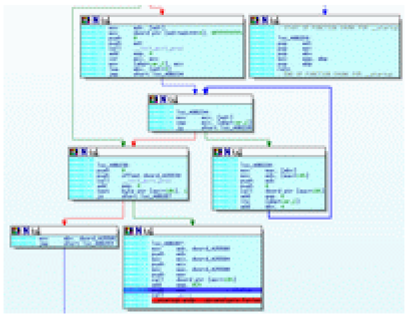
IDA Pro 5.3 Feature list



HIGHLIGHTS

- **New and improved debugger**

  The previous version of IDA Pro did not add anything to the debugger and we felt it is time for changes. We **reimplemented the debugger core** and improved the debugger modules.

  The new debugger is more efficient and has better support for multithreaded applications. Breakpoint handling is faster, more logical and less deadlocking. Exception handling is more user friendly.

  The **debugger servers are multithreaded**: they can handle multiple debug sessions, no need to kill a hung server or run multiple copies.

- **Debugger modules**

  We added two new debugger targets:

  - **iPhone** debugger. Click here for the details.
  - **Symbian OS** debugger. Click here for the details.

  We publish the source code of all debugger modules.

  The **Linux debugger** module has been improved to support multithreaded applications. We support NPTL based kernels.

- **Better analysis for PC and ARM**

  The most important improvements include support for PIC addressing modes, more jump tables and many other useful patterns. In practice this means that the output for **iPhone/iMac/Linux/Symbian** applications greatly improves. Please refer to the comparison page for more details.

- **New PDB plugin**

  The new plugin extracts all name and type information from a **PDB** file and imports it into the database. The difference is spectacular.

- **New TILIB utility**

  This small and nifty utility allows you to create **your own type libraries**. The Load C header command in IDA Pro could be used to load them in the past. The TILIB utility is easier to use and gives you more control. It also can import preprocessor symbol information.

- **Support for third party languages**

  Interested parties may register their own language interpreter **(perl/python/ruby - you name it)** to be used as the expression evaluator in IDA. This will allow you to use your favorite language everywhere in IDA.

- **Signatures**

  As usual, the new release comes with updated signatures, type libraries, ids files, etc. Namely, we updated them for the latest **Visual Studio, Intel, and Borland** compilers.

PROCESSOR MODULES

```
+ 6812: added support for HCS12X (thanks to Alex Bratovic)
+ ARM: 'mov' macro can consists of multiple (more than 2) instructions
(igor)
+ ARM: "stmfd/sub sp,sp" is considered as a typical code sequence; this
improved the listing
+ ARM: added support for signed byte element jump tables
+ ARM: better automatic arm/thumb mode switch
+ ARM: better detection of BL as sub or jump
+ ARM: DCQ means quadro word
+ ARM: ida knows that R7 is used as frame pointer in thumb mode
+ ARM: ida was leaving wrong targets of glue code intact, now it always
fixes them; this may eventually modify a user-defined offset but we are
certain that this is a good thing to do
+ ARM: more jump table variants are recognized
+ ARM: more glue code and thunk functions are detected
+ ARM: MOVL macro has been renamed as MOV to avoid confusion with MOVLS
(thumb mode MOV has always the S bit set); this renaming makes it
impossible to tell apart the basic MOV instruction and the MOV macro just
looking at the text. Please use the instruction sizes to tell them apart.
+ ARM: much better stack pointer tracing
+ ARM: one more .got addressing method is supported
+ ARM: one more pc-relative addressing method is supported
+ ARM: recognize table switches generated by Apple's compiler
+ ARM: reference into the middle of a macro instruction destroys it
(analysis improvement)
+ ARM: strip the low bit of thumb code references during offset analysis
+ ARM: thumb mode thunk targets are converted to functions
+ ARM: when the processor module is 100% certain that an offset must be
created, it may destroy old database information
+ AVR: added description of AT89C2051 (contribution of an ida user)
+ CLI: if the list of switch targets is too long, it is split into multiple
lines
+ CLI: better handling of obfuscated code
+ PC: added detection of check_security_cookie() function for object files
+ PC: added recognition of call+5/pop idiom for PIC code
+ PC: added support for the ud2 instruction
```

+ PC: added undocumented 3-byte nop instructions (0F 19..0F 1E)
+ PC: automatically recognize .got relative addressing for pic mode elf files
+ PC: better analysis of device drivers
+ PC: better handling of indirect calls by register
+ PC: ida knows that the "alloc_stack" function allocates stack
+ PC: inc/dec sp are taken into account for stack tracing (16-bit segments)
+ PC: indirect calls to noret functions stop the control flow
+ PC: more condition codes and the 'elf' register can be directly used in idc while the debugger is active
+ PC: **more gcc generated jump tables** are recognized
+ PC: third operand of imul instruction is never displayed as offset, stkvar or stroff
+ PC: user-specified callee address is used for all addressing modes (before is was used only for indirect register calls)
+ M32R: added support for undocumented form of the STH instruction (@R+ addressing mode)

FILE FORMATS

+ AR: added support for Apple/BSD ar libraries (Igor Skochnisky)
+ ELF: added more SPARC relocations
+ EPOC: added support for **Symbian S60 3d edition** SIS files
+ EPOC: ids files have been updated for Symbian SDK for S60 3d edition
+ MACH-O: the entry point of **packed executables** is visible even if it is in the HEADER segment
+ PDB: **new pdb plugin**: uses new DIA API and handled type information
+ PE: added support for data imports in GCC compiled binaries
+ PE: added support for **long segment names** (this and many other improvements thanks to Igor Skochinsky)
+ PE: added support for tiny PE files (thanks to Igor Skochinsky)
+ PIC: allow the user to choose the target device at the loading time; added pic18f2620 port definitions
+ environment variable IDA_LOADALL makes ida to load all segments of input file (pe,elf,coff)

KERNEL

+ added logic to avoid creation of too big multichunk functions
+ added an heuristic rule: switch targets can not be separate functions
+ added FPNUM_LENGTH and FPNUM_DIGITS ida.cfg parameters to set the desired floating point representation
+ added more noreturning functions to noret.cfg
+ added notion of enum element width: now enum types can be synchronized with the local type library without information loss; idc functions to handle the enum element width have been added
+ added **signatures for the latest VC8, VC9 and UnixInWindows**
+ added support for Visual Studio style enum size specification (e.g. enum name:int {...})
+ better handling of zero length bitfields
+ changed behaviour of the IDALOG_SILENT environment variable: it unconditionally suppresses all output to the message window
+ incorrect structure field types are ignored when building type string for the structure
+ new ida.cfg parameter: **WORKDIR** specifies the directory to create temprary database files; can be used to improve the speed of opening and closing huge databases
+ new idb event: area_cmt_changed; it is generated when a function or segment comment is changed
+ the plugin options specified by -O are accessible to PLUGIN_FIX plugins

+ preprocessor directives can be used in type declarations (e.g. #pragma pack)
+ stricter check of stkvars while guessing function types; this allows us to ignore corrupted stack frames
+ the "generate idc" command knows about patched bytes
+ the meaning of the -P command line switch has been changed: -P+: compress, -P: pack, -P-: unpack the database
+ updated **Intel compiler signatures** (added support for v10.1)
+ updated **Borland BDS signatures and added Delphi 2007** signatures (thanks to Peter Sawatzki)
+ gui: 'rename' command renames the structure field under the cursor if applied to an expression refering to global variable of a structure type; before is was renaming the global variable regardless of the cursor position
+ gui: added support for extra keyboard back/forward buttons
+ 'bool' is accepted in type declarations

IDC & SDK

+ IDC: added ChangeConfig() to modify ida.cfg settings on the fly
+ IDC: added CompileEx() to compile arbitrary IDC scripts from a string
+ IDC: added debugger option to specify **how exceptions are handled**. possible values: always, only for unknown exceptions, never display a dialog box upon continuation. The default is set to display the dialog box for all exceptions.
+ IDC: added exception defintion functions
+ IDC: added extended forms of AddStrucMember and SetMemberType
+ IDC: added GetEntryName() to get the name of an export outside of the address space of the program
+ IDC: added GetInputMD5()
+ IDC: added ResumeProcess() and WFNE_NOWAIT for GetDebuggerEvent()
+ IDC: added SetInputFilePath()
+ IDC: added Sleep()
+ IDC: SuspendThread/ResumeThread have been added
+ IDC: added Qword() function (64bit version of IDA)
+ SDK: added a plugin to specify switch idiom details (**uiswitch**)
+ SDK: added coagulate_dref event (occurs when the kernel analyzes a dref or coagulates data)
+ SDK: added more qstring member functions and more types based on qvector/qstring
+ SDK: added qsleep()
+ SDK: added qwstring class for unicode strings
+ SDK: added register_extlang() to register third party expression evaluators
+ SDK: added resolve_typedef2(), it returns the name of the resolved type
+ SDK: added SaveBase() function to save the current idb
+ SDK: added ui_preprocess and ui_postprocess events to intercept ui commands
+ SDK: added **xref creation/deletion events**
+ SDK: choose_local_type() to choose types from the local type library
+ SDK: choosers can be created without main menu and status bar
+ SDK: exported determine_rtl() and apply_startup_sig() functions
+ SDK: got rid of time_t in the header files because its size is compiler-dependent; we use qtime32_t instead
+ SDK: renamed processor_t::get_jump_target as next_exec_insn; this callback must return the address of the next executed instruction in all cases, not only for jump instructions
+ SDK: set_segm_start/end functions accept SEGMOD_... flags as the last parameter
+ SDK: added get_process_options()
+ SDK: added CH_NOBTNS to suppress all chooser buttons for modal windows

DEBUGGER

+ debugger: added commands to suspend/resume threads
+ debugger: added support for **multiple debug names per address**; ida will display only the first one in the listing though but other names can be used to refer to the location
+ debugger: **CPU window is sleeker**, occupies less space on the screen
+ debugger: debugger server kills the application if the server dies for some reason (SIGINT, SIGTERM, etc)
+ debugger: IDA does not steal the window focus when the debugger is controlled from a script or a plugin
+ debugger: if the remote debugger server becomes irresponsive, close the debug session gracefully
+ debugger: more detailed error message about debugger privileges
+ debugger: reimplemented the debugger core. the new core can handle multithreaded apps and is more intelligent with singlestep/breakpoints. it suspends some threads only if it really unavoidable (the previous core was suspending all threads for singlestepping)
+ debugger: the thread window has no main menu and occupies less screen space
+ debugger: we store **debugger desktops for different processors** separately
+ debugger: 32-bit and 64-bit versions store the default values in different registry keys

BUGFIXES

BUGFIX: 'open selectors window' command was always complaining about failure
BUGFIX: 'text search' would not find anything in user-defined graphs
BUGFIX: "bad declaration" error message could appear while loading some pdb files
BUGFIX: .net cli was incorrectly decoding conv.r4, conv.r8, and conv.r.un instructions
BUGFIX: 64-bit portion of Macho-O files could be proposed to be disassembled by default by 32-bit version of ida
BUGFIX: 64-bit: rebasing the program would leave the relocations in the incorrect sate because of a wrong loader file name
BUGFIX: abstract function prototype with the __spoils keyword could contain some garbage after the keyword
BUGFIX: anonymous structure types could crash ida
BUGFIX: arm: xrefs from byte operands with a displacement could be incorrect
BUGFIX: arrays of partial types (like _BYTE[5]) could not be declared
BUGFIX: binary search for too long string (>1024 bytes) would crash IDA
BUGFIX: calling get_colored_[demangled_]name with too small buffer would lead to fatal error
BUGFIX: closing a chooser window with a middle click on its tab would prevent ida from reopening it in the future
BUGFIX: could crash trying to demangle extremely long names
BUGFIX: could crash trying to refresh a graph view
BUGFIX: could crash when the debugger was launched
BUGFIX: could fail with "not enough memory" trying to open a huge database
BUGFIX: could hang trying to calculate the number of purged bytes
BUGFIX: could not display empty graphs
BUGFIX: could undefine some instructions upon the debugger start
BUGFIX: definition of iphdr structure was wrong in gnuunx.til
BUGFIX: duplicate field names in struct/union declarations were not reported
BUGFIX: envp in main() prototype was declared incorrectly
BUGFIX: epoc: exports of epoc files with versioning support were incorrectly parsed

BUGFIX: esp based stack variables were displayed incorrectly if the frame pointer delta was non-zero
BUGFIX: fatal error could occur at the end of the debugging session (interr:manage_debugger_segments)
BUGFIX: fixed a memory leak in idc interpreter
BUGFIX: functions with EH_prolog could have wrong stack trace
BUGFIX: get_process_qty() would fail if the debugger was not connected to a remote computer; now it automatically establishes connection if necessary
BUGFIX: graph overview window might lose its "topmost" attribute for some reason
BUGFIX: green arrow was displayed incorrectly in wince debugger
BUGFIX: gui: problems with window focus in mdi: right clicking on an inactive graph view would switch the focus to it but right clicking on the window which was active initially would not return focus to it
BUGFIX: gui: there could be garbage at the end of very long disassembly lines
BUGFIX: HEX loader would load garbage if user in the 'word addressing' mode for PIC processor
BUGFIX: huge basic blocks could generate endless "insuffucient resources" dialogboxes in the graph mode
BUGFIX: idc: exception codes and exit codes were signed extended in 64-bit ida
BUGFIX: if a plugin modified a standard struct or enum, the corresponding local type would stay unmodified and out of sync
BUGFIX: if a plugin would create a graph view and would not specify the zoom level, IDA would crash
BUGFIX: if the analysis indicator was disabled, ida would display garbage
BUGFIX: if the user specified java target for non-java input file, ida would quit without cleaning temporary files
BUGFIX: in amd64 elf files R_X86_64_PC32 relocation record could resolve incorrectly in some cases
BUGFIX: in some very rare cases ida could quit with an error message (trying to analyze a function with an unreachable loop that passes control to other basic blocks reachable from the function entry)
BUGFIX: it was impossible to use 'text search' in user-defined graphs
BUGFIX: linux: IDA could not display unicode strings if the LC/LC_CTYPE environment variables were missing; now it falls back to LANG
BUGFIX: list windows: pressing Ctrl-Enter staying at the last element would cause an access violation
BUGFIX: loading a corrupted til file could crash ida
BUGFIX: m32r: clrpsw/setpsw instructions would generate interr
BUGFIX: mac debugger: the error message about the "setgit procmod" requirement was always about mac_server. for local debugger, idal must be setgid procmod, not mac_server.
BUGFIX: macho files had empty 'imports' window
BUGFIX: mc68x16 lbra instruction stops the execution flow but ida was not aware of it
BUGFIX: mentioning debugger plugins as regular plugins in plugin.cfg could lead to a crash
BUGFIX: mips jalx instruction was toggling the mips16 bit at a wrong address
BUGFIX: mips: negative operands could not be converted to offsets
BUGFIX: MIPS16 jalx instruction was decoded incorrectly
BUGFIX: multiple copies of ida could run slowly on multicore cpus
BUGFIX: non-resursive implementation of gdl_graph_t::path because the recursive implementation was running out of stack in some special cases
BUGFIX: old segment name was unusable after a segment renaming
BUGFIX: pc elf files could have vc6win.til file loaded instead of gnuunx.til
BUGFIX: pc: feature bits of bswap instruction were wrong
BUGFIX: pc: some illegal instructions could be disassembled as 'mov' (opcodes C6 and C7)

BUGFIX: PIC: immediate operand of movlw and similar instructions was treated as a signed number
BUGFIX: PPC could not disassemble m[tf]ocrf instructions
BUGFIX: rebasing the database would not update some information (function prologs, etc) for x86 targets
BUGFIX: rebasing the program would not modify its imagebase in the database (no visible consequences, though)
BUGFIX: restarting the debugger could cause a crash if the stack trace window was opened by default
BUGFIX: SDK: intel.hpp, is_segreg() had a bug
BUGFIX: SDK: set_da() had a bug
BUGFIX: some EPOC6 SIS files could not be loaded
BUGFIX: some pic devices were placed in wrong cfg files
BUGFIX: some TMS470 ARM COFF files could not be loaded (the text segment would be skipped)
BUGFIX: some very old databases could not be upgraded
BUGFIX: sorted lists were not refreshed properly
BUGFIX: structure fields of the "structure offset" type were exported incorrectly to IDC file
BUGFIX: structures and enums that were created by importing local types had 'til type' flag which would prevent further synchronization from idb to local til
BUGFIX: switching between target processors in mc68xx was buggy and would lead disassembly problems (6805/6808)
BUGFIX: Symbian9 epoc import parsing was incorrect
BUGFIX: the check of address space limit was incorrect
BUGFIX: the current file offset was displayed incorrectly for processors with unusual byte size
BUGFIX: the cursor position was changing after a debug session
BUGFIX: the debugger was displaying a dialog box on exceptions with "don't stop" flag
BUGFIX: the default alignment was incorrectly set to 4 for 64bit programs (must be 8)
BUGFIX: the graph overview window would not be immediately displayed for user-defined graph views
BUGFIX: the second parameter of the create_struc_member event was wrong
BUGFIX: the stack analysis could fail with a fatal error for huge function with too many stack change points
BUGFIX: there could be some access violations if the Jump() function was repeatedly used from an IDC script
BUGFIX: there were discrepancies between 32-bit and 64-bit versions of IDA
BUGFIX: too long function names could crash ida (while displaying xref information)
BUGFIX: trace results in the file were too wide
BUGFIX: tree layout could crash on some cyclic graphs
BUGFIX: tricore module was not creating xrefs for offset expressions
BUGFIX: user-defined xrefs could be replaced by regular xrefs and then deleted by the kernel
BUGFIX: vmread/vmwrite instructions were decoded incorrectly in 64-bit mode
BUGFIX: when attaching to a process IDA would not properly switch to the debugger desktop
BUGFIX: if the graph layout algorithm failed, the graph would be left in an incorrect state (with temporary nodes)
BUGFIX: 64bit: it was impossible to edit a breakpoint at address > 0xFFFFFFFF
BUGFIX: IDA window title might display garbage after closing a mini database