IDA Pro 5.2 feature preview

HIGHLIGHTS

Improved iPhone support

IDA Pro 5.2 handles iPhone executables out of the box.

Much improved ARM and PowerPC support

The updated ARM module supports **200 new instructions**. This module started with mere 38 instructions: the ARM was a small and sleek processor; now it boasts 3 different instruction encodings [arm, thumb, thumb32], dsp, floating point, vector, simd, and even java instructions!

Much improved PowerPC module

The PowerPC module has ~40 new instructions.

Easy debugger scripts in IDC

The debugger is finally available from IDC. There is no need to write complex plugins, install event handlers and express the logic in a finite state machine form. Simple and natural functions that allow to wait for the next debugger event and continue the script without yielding control to the ida kernel have been introduced.

Improved type support

The type system has been improved to support types of abitrary length and complexity (there was a limitation of 1024 bytes per type descriptor). The user interface offers a new window to display and manipulate local types. This allows for easy migration of types from one database to another. You can even export all local types in a compiler readable form!

Detailed list of changes

PROCESSOR MODULES

+ 6811: the output is more conforming to Motorola freeware assemblers (thanks for Alex Bratovic)
+ 68xx: CodeWarrior and GNU output support have been added by Alex Bratovic
+ ARM: add/sub instructions are better emulated
+ ARM: added bxj insn (Igor Skochinsky)
+ ARM: added one more switch pattern (Igor Skochnisky)
+ ARM: added RealView v3.1 low-endian signatures
+ ARM: added recognition of rt_switch8() function (Igor Skochinsky)
+ ARM: added support for v6ZK instructions
+ ARM: added support for VFP (vector floating point) instructions
+ ARM: added thumb32 encodings and v7 instructions
+ ARM: analysis of some pc-relative addressing modes has been improved (notable difference for iphone executables)
+ ARM: armv6 instructions are supported
+ ARM: better propagation of thumb-bit; more glue code patterns are recognized
+ ARM: better recognition of thunk functions and flow detection
+ ARM: better register tracking in the thumb mode
+ ARM: BX LR is considered as a return instruction

+ ARM: more intelligent handling of immediate values in instructions; they are converted to offsets only in the second analysis pass and only if the kernel option permits it
+ ARM: more jump tables are recognized
+ ARM: more return instruction forms are recognized
+ ARM: more thumb-mode macro instructions are recognized (this and many other arm improvements thanks to Igor Skochnisky)
+ ARM: much better handling of glue code
+ ARM: noreturn-analysis has been turned on for ARM
+ ARM: some pc-relative addressing modes were not supported
+ ARM: thumb glue handling has been improved
+ ARM: "push lr" is considered as a function start
+ ARM FMXR/FMRX: added VFP system registers FPINST, FPINST2, MVFR1, MVFR0 (by Igor)
+ ARM: better tracing of the stack pointer
+ ARM: more jumptables are recognized (Igor)
+ ARM: skip irrelevant instruction during function frame analysis
+ M32R: added floating point and bit instructions
+ MIPS: automatically determine $gp value for pic programs (heuristic rule)
+ PC: call insn with the sole purpose of obfuscating the code are recognized
+ PC: prolog analysis has been improved
+ PC: thunk functions have priority over function chunks
+ PC: XlatAsciiOutput translation is applied to the hex view and unexplored byte comments
+ PPC: added support for ~40 new instructions
+ SH3: added support for i/o port config files; sign extend byte/word immediate transfers
+ TMS320C55: added support for 6-byte instructions and some undocumented parallel instructions (thanks to Roman Vasiliev & Ivan Litvin)

FILE FORMATS

+ ELF: added handling of the R_X86_64_RELATIVE relocation type
+ ELF: arm: some elf files have low bit 1 in the thumb function addresses; ignore it
+ ELF: improved processing of .got section in elf files
+ ELF: pc: added support for 64bit gotpcrel and plt relocation types
+ added support for iPhone Mach-O files (no relocations yet)
+ MACHO: entry point of arm executables is detected
+ MACHO: 64bit macho files are supported; better reloc handling; cfstring handling (all by Igor Skochinsky)
+ MACHO: added handling of arm relocations; thumb-exports; (thanks to Igor Skochinsky)
+ support for SunOS a.out file format has been added (an ida user contribution)
+ TDS: OPTVAR32 records are handled

KERNEL

+ Added an option not to truncate functions upon code deletion
+ added noret.cfg with the names of non-returning functions
+ added support for new visual studio mangling schemes (thanks to Yury Haron)
+ added support for the #include_next directive in the c header parser
+ anomalious situations when the same block was repeatedly converted to code and back to data are better handled
+ FLAIR: added support for reaview libraries
+ do not automatically create non-zero page unicode strings (usually it is wrong)
+ new analysis option: ignore control flow to pure data segments. it is off by default except for mach-o files
+ new keyword in function prototypes: __spoils<reglist>. It specifies the list of spoiled registers for non-standard functions

+ new switch -L can be used to specify the name of the log file
+ non-zero page unicode strings are displayed in a more readable form
+ propagate function names from export thunks to function implementations
+ updated gnuunx.til
+ do not create data array in the stack segment if it contains the entry point
+ references to local types are made by their ordinal numbers
+ structure/enum modifications are automatically synchronized into local til

IDC & SDK

+ added more debugger related IDC functions and reimplemented the core functionality of uunp in IDC
+ c parser: accept preprocessor macros in include directives
+ IDC: Added Eval(), IsString(), IsLong(), IsFloat() functions
+ IDC: added GenFuncGdl() and GenCallGdl() functions to generate GDL files
+ IDC: added high level debugger functions to idc; now it is possible to write scripts to control the debugger
+ IDC: added functions to manipulate local types
+ IDC: string variables can hold strings > 1023 characters
+ SDK: add_chooser_command() to add user-defined actions to chooser windows
+ SDK: added a callback for mouse click events in custom viewers; graph viewers are custom viewers too and all custom viewer functions can be used with them
+ SDK: added pc_module_t::find_reg_value callback so that plugins can find register values if necessary
+ SDK: added readsel2() to get more info about the current selection
+ SDK: added reserve(), capacity(), swap() functions to the qvector template class
+ SDK: added set_custom_viewer_range()
+ SDK: added support for negative operand values in offsets; the kernel will use a negative value if the OOF_SIGNED bit is used in outflags; PPC and M32R listings are much better
+ SDK: added wait_for_next_event() and get_debug_event() functions
+ SDK: almost all type-related functions have til_t as the first parameter; this will allow us to introduce local type libraries and local types in the future
+ SDK: more efficient (but more memory hungry) implementation of qvector::resize()
+ SDK: new event ui_ready_to_run: occurs when the user interface is fully initialized. this event can be used to run automatic actions from plugins
+ SDK: new function compact_til() should be called before storing til file to the disk; otherwise store_til() will compact the til anyway
+ SDK: new function: parse_reg_name()
+ SDK: ph.calc_arglocs has been superseded by ph.calc_arglocs2 (optimization)
+ SDK: prototypes of type-related functions have been modified to support unlimited type strings; old functions are still available but their use is strongly discouraged
+ SDK: qeprintf() function to print on stderr has been exported
+ SDK: qsplitpath() function has been replaced by qdirname()
+ SDK: replaced value_t with idc_value_t, a class which allocated/frees memory itself and does not require manual handling
+ SDK: show_wait_box() can display dialogs with the "cancel" button. For that, pass the wait messager prefixed with "HIDECANCEL\n"
+ SDK: added export_type() to copy struct/enum definitions to til improved local til view
+ SDK: added new functions: for_all_types(), replace_subtypes(), create_numbered_type_name()
+ SDK: added print_type_to_qstring(), a generic type printing function
+ SDK: added ui_open_builtin callback and corresponding functions
+ SDK: introduced compile time constant: IDA_SDK_VERSION
+ SDK: introduced request_refresh() for more efficient window refreshing; this function allows the programmer to specify exactly what windows must be refreshed

USER INTERFACE

+ ui: remember input field history in the registry (gui)
+ ui: added menu items to produce GDL files (flow chart and call graph)
+ ui: breakpoint commands are available in the "occurrences of..." lists
+ ui: if a saved desktop is incompatible with the current screen resolution, inform the user in the message window
+ ui: local type definitions can be manipulated in the new "local types" window
+ ui: local type library can be synchronized to the current idb and vice versa
+ ui: text search results have one more column: function name
+ ui: added "export local types to header file" command (right click in the local types window)
+ ui: Shift-F1 opens the local types window

BUGFIXES

BUGFIX: -o switch would cause get_input_file_path() to return wrong value
BUGFIX: "can't assign to segment register" was displayed on some object files
BUGFIX: 16-bit PIC instruction that skip the next instruction were not taken into account during the analysis
BUGFIX: 64bit delayed import tables were parsed incorrectly; 32-bit import tables could have an extra empty slot at the end
BUGFIX: 68k: 2-byte floating point constants in instructions were decoded incorrectly
BUGFIX: ad218x: CF_JUMP bit was set for wrong insns
BUGFIX: add_regvar was incorrectly handling the situation when a register is redefined in a smaller range
BUGFIX: adding xrefs from the user interface was adding only code xrefs
BUGFIX: an access violation could occur during the analysis of huge and complex programs with fragmented functions
BUGFIX: ar2idt was broken
BUGFIX: arm $-auxiliary symbols in elf files were not always handled
BUGFIX: arm: bx thumb_code was creating wrong xrefs if the target address has the low 2 bits set
BUGFIX: build_funcarg_arrays() would fail on typedefed function types
BUGFIX: build_funcarg_info() could return >= 0 for a non-function type
BUGFIX: c parser: enum members starting with an underscore could not be used in other enum declarations
BUGFIX: Cbuilder 5.1 has a bug in realloc(), added a workaround
BUGFIX: closing forms with customized "No" button by clicking on the X window close button would return "true" as if the user pressed OK
BUGFIX: complex type names could be intermixed during get_type_size()
BUGFIX: corrupted database would cause an access violation if an unexisting function tail was removed
BUGFIX: could display an internal error on old database with partial jump table information attached to indirect jumps
BUGFIX: could not debug new PocketPC devices because the processor was detected as xscale for some reason (thanks to Alexander Alkhovik for the tests)
BUGFIX: curloc() destructor could spoil database (in theory)
BUGFIX: demangler mode was set incorrectly
BUGFIX: epoc file without import table could not be loaded
BUGFIX: epoc idata section was not created in the text version
BUGFIX: epoc loader could crash trying to load incorrect input files
BUGFIX: EPOCV9 target processor type was not detected correctly
BUGFIX: extern "C" { extern... could not be parsed
BUGFIX: F12 (generate graph for wingrap32) was creating too many outgoing edges for some nodes
BUGFIX: FLAIR signature files with names 1023 chars or longer could not be loaded
BUGFIX: function names in the xrefs window were aligned to the right
BUGFIX: gcc v3 static names that are local to a function were not demangled
BUGFIX: get_idp_type() was using static buffers to return its return value; there was not

enough of them to handle deeply nested structures
BUGFIX: get_short_name() group was not demangling gcc v3.x generated names
BUGFIX: guess_type() could recurse indefinitely and crash
BUGFIX: help file was mentioning IDC FindProc() function which did not exist anymore
BUGFIX: IDA was accepting "struct DWORD" as a valid type which is illegal because DWORD is declared as a scalar type, not a struct
BUGFIX: IDA could crash if the file history in the registry had holes in the numbering
BUGFIX: IDA was not reanalyzing the stack if the purged bytes of a called function became determined
BUGFIX: IDC: is was impossible to take segment snapshot with SetSegmentAttr()
BUGFIX: if a function with graph node group information was truncated, the graph view could not be rendered in some cases
BUGFIX: if lazy jumps were on, some jump targets could not be reached (the ones just below the visible part of the listing)
BUGFIX: if the function end was changed after sp-analysis had been planned but not processed, the sp-analysis would not be performed at all
BUGFIX: in some cases function tails were not reanalyzed
BUGFIX: indirect calls were not using the type information in memory dumps (because memory dumps had dangling pointers to system dlls and the callee was determined to be outside of the program)
BUGFIX: It was impossible to open the structure window if the enum window was the active window
BUGFIX: m32r: incorrect encoding of the mvtc instruction could cause an internal error
BUGFIX: m32r: it was impossible to change the sign of immediate operands
BUGFIX: mac: debugged application arguments were malformed
BUGFIX: mac: if idal was not in path, it could not be launched using the relative path
BUGFIX: memory corruption might occur in idag.exe at the very start
BUGFIX: mips: non-returning functions were terminating the execution flow too early, without taking into account the delay slots
BUGFIX: modifying the function end would destroy detailed prolog info
BUGFIX: modifying the function start would destroy frame info
BUGFIX: moving a segment while staying at address 0 would hang ida
BUGFIX: omf import libraries were displayed incorrectly in the listing (superfluous 'db' directirves were present)
BUGFIX: only the first 2 operands of instructions were handled with complex struct offset dialog box
BUGFIX: pc: "xchg rax, r8" was not disassembled correctly
BUGFIX: pc: 16bit indirect call targets were guessed incorrectly (the current segment base was not taken into account)
BUGFIX: pc: 16bit: the segment part of a far jump to an invalid address would be displayed using the selector value if that segment happened to have a corresponding selector. this could confuse the user
BUGFIX: pc: in some rare cases ida could miss the end of a basic block terminating with an indirect jump
BUGFIX: pc: sahf/lahf feature bits were wrong
BUGFIX: pc: the target of an indirect call with a register phrase and immediate displacement which had been converted to a user-defined offset was not determined correctly
BUGFIX: pc64: A1 opcode with the REX prefix was decoded incorrectly
BUGFIX: pic18 instructions comments were missing from ida.int
BUGFIX: PPC: unpredictable branch hints could not be decoded
BUGFIX: preprocessor token glue idiom (##) with empty tokens was not handled properly
BUGFIX: qvector.resize() had an integer overlow
BUGFIX: SDK: calc_names_cmts() had wrong calling convention for the callback
BUGFIX: SDK: dbg_exception event parameters were wrong
BUGFIX: SelStart()/SelEnd() IDC functions were working incorrectly for hexviews
BUGFIX: some functions would be incorrectly marked as "thunk"

BUGFIX: some intentionally malformed PE files could not be loaded correctly
BUGFIX: sparc flow charts were not optimal, call instructions were dividing nodes
BUGFIX: sparc high offsets were broken
BUGFIX: Symbian ROM EKA2 images were loaded incorrectly (in fact we could not tell apart EKA2 and EKA1, now we added an heuristic rule)
BUGFIX: the 'copy to clipboard' command was not available unless we are in a eaview
BUGFIX: the argument names of idb structure fields typed as "pointer to function" were incorrectly translated to type system field names
BUGFIX: the code to refresh the navigation band could spoil exception records
BUGFIX: the debugger server could accept a specially crafted incoming connection without a password
BUGFIX: the dialog box displayed by choose_enum() was not positioning on the default enum
BUGFIX: the font attribute was wrong in the output html files
BUGFIX: the pushinfo of 1byte functions could delete the pushinfo of the next function; it is not saved in the database anymore
BUGFIX: THREAD_EXIT debugging event was not handled properly
BUGFIX: tracing was not skipping library and debuger segments
BUGFIX: truncating a function with renamed registers could corrupt memory
BUGFIX: when a new mdi window was opened, ida would forget about the last disassembly view pointer
BUGFIX: when the graph overview window was active, displaying a modal form would cause an error message about focus and window visibility
BUGFIX: ARM: LDC/STC: some addressing modes were disassembled incorrectly
BUGFIX: ARM: SRS was missing the first argument (SP)
BUGFIX: ARM: XScale MRA and MAR instructions were swapped
BUGFIX: could not parse a header file if it contained redefinitions of standard structure types using typedefs
BUGFIX: IDA would crash at the exit time if the cpu registers window was open but the debugger never run
BUGFIX: it was impossible to find GetIdbPath using the search function of the help
BUGFIX: notepad could ignore the Enter key after displaying a modal window
BUGFIX: set_visible_func() called with a pointer to function tail could crash
BUGFIX: some switch idioms were recognized as having too many cases
BUGFIX: structure types with gap at the very beginning could not be used correctly in function prototypes
BUGFIX: structure/enum window would appear to be empty after a mass type deletion from a script even if not all types were deleted
BUGFIX: switch_info_ex_t structures were not deleted upon the 'undefine' command
BUGFIX: the 'create stroff' command was not displaying the sizeof(struct) choice for structures with embedded unions if the union was the first field of the structure
BUGFIX: the string list would continue to use the old program boundaries after segment creation/deletion
BUGFIX: debugger: DEP exceptions are correctly reported as "execution failure"