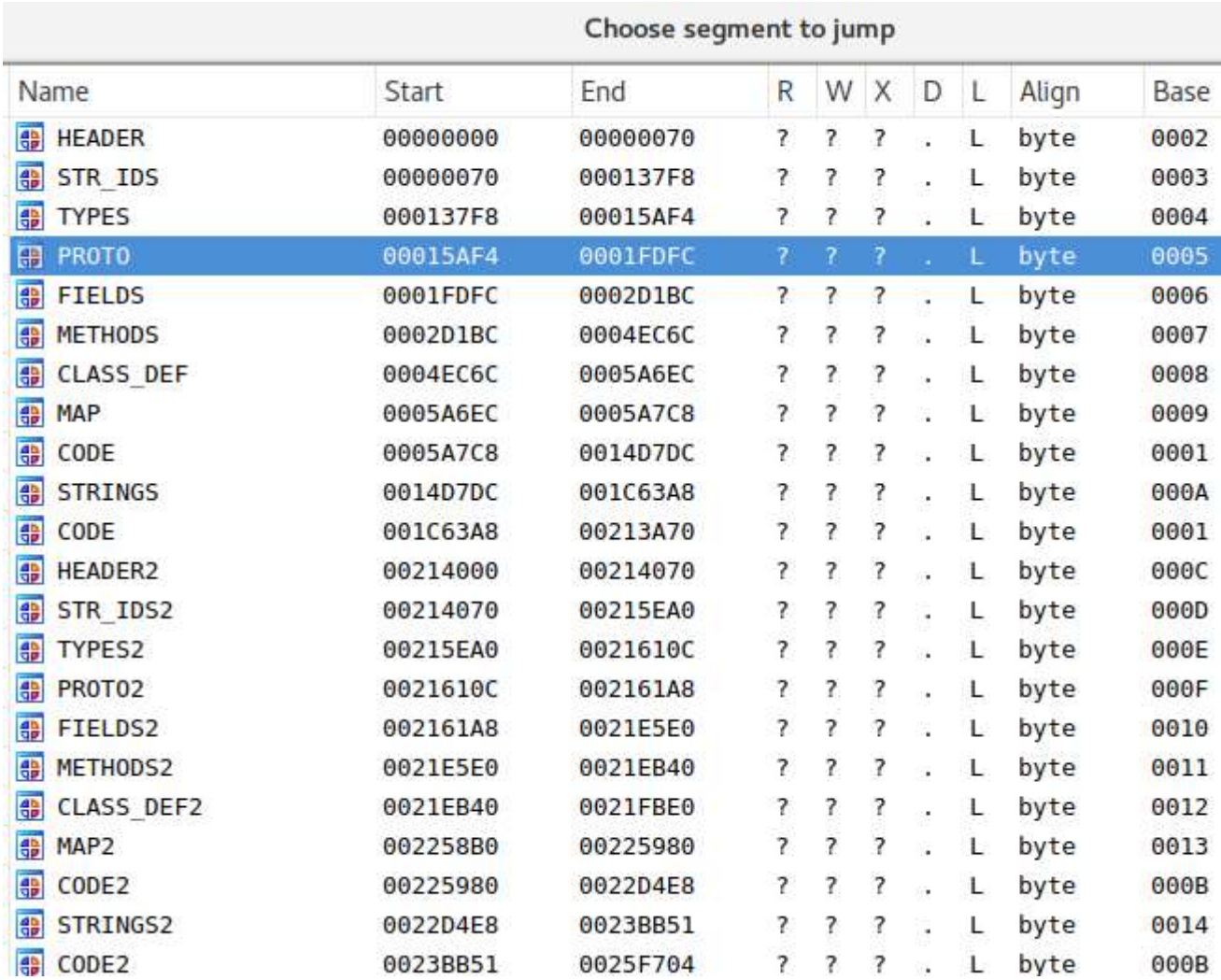
IDA 7.1 - What's new?

Highlights

- With this version of IDA we publish the decompiler intermediate language: the microcode. We were planning to do it since very long time but the microcode was constantly evolving, we could not do it. After ten years of evolution it looks mature and ready to be published. We believe that it will permit our users to implement much more powerful and higher level analysis algorithms than before. In the future we plan to use the microcode in IDA too: if the decompiler is present, the analysis will be improved automatically.

```
0. 0 ; STKD=C MINREF=0/END=24 ARGS: OFF=28/MINREF=28/END=128/SHADOW=0
0. 0 ; SAVEDREGS: ebp.4,ebx.4,esi.4,edi.4
0. 0 ; ????-BLOCK 0 PROP FAKE [START=4014E0 END=4014E0] MINREFS: STK=0/ARG=28, MAX
0. 0
1. 0 ; ????-BLOCK 1 PROP COMB [START=4014E0 END=4014FB] MINREFS: STK=0/ARG=28, MAX
1. 0 ; USE: esp.4,arg+0.8
1. 0 ; DEF: cf.1,zf.1,sf.1,of.1,pf.1,ebx.4,ebp.4,edi.4,sp+18.4
1. 0 ; DNU: cf.1,zf.1,sf.1,of.1,pf.1,ebx.4,ebp.4,edi.4
1. 0 mov    esp.4, ebp.4              ; 4014E1 u=esp.4      d=ebp.4
1. 1 mov    #0.1, cf.1                ; 4014E3 u=           d=cf.1
1. 2 mov    #0.1, zf.1                ; 4014E3 u=           d=zf.1
1. 3 mov    #0.1, of.1                ; 4014E3 u=           d=of.1
1. 4 und    sf.1                      ; 4014E3 u=           d=sf.1
1. 5 und    pf.1                      ; 4014E3 u=           d=pf.1
1. 6 mov    %argv.4, ebx.4            ; 4014E9 u=arg+4.4    d=ebx.4
1. 7 mov    %argc.4, edi.4            ; 4014EC u=arg+0.4    d=edi.4
1. 8 mov    &($dword_41D128).4, %suffix.4 ; 4014EF u=           d=sp+18.4
1. 9 goto   $loc_401587               ; 4014F6 u=
1. 9
2. 0 ; ????-BLOCK 2 PROP COMB [START=4014FB END=401506] MINREFS: STK=0/ARG=28, MAX
2. 0 ; USE: ebx.4,ds.2,(GLBLOW,LVARS,ARGS,GLBHIGH)
2. 0 ; DEF: cf.1,zf.1,sf.1,of.1,pf.1,ax.5
2. 0 ; DNU: cf.1,sf.1,of.1,pf.1
2. 0 ldx    ds.2, (ebx.4+#4.4), eax.4 ; 4014FB u=ebx.4,ds.2,(GLBLOW,LVARS,ARGS,GLB
2. 1 ldx    ds.2, (eax.4+#1.4), dl.1 ; 4014FE u=eax.4,ds.2,(GLBLOW,LVARS,ARGS,GLBH
2. 2 setb   dl.1, #0x61.1, cf.1       ; 401501 u=dl.1       d=cf.1
2. 3 seto   dl.1, #0x61.1, of.1       ; 401501 u=dl.1       d=of.1
2. 4 sub    dl.1, #0x61.1, dl.1       ; 401501 u=dl.1       d=dl.1
2. 5 setz   dl.1, #0.1, zf.1          ; 401501 u=dl.1       d=zf.1
2. 6 setp   dl.1, #0.1, pf.1          ; 401501 u=dl.1       d=pf.1
2. 7 sets   dl.1, sf.1                ; 401501 u=dl.1       d=sf.1
2. 8 jcnd   zf.1, $loc_401517         ; 401504 u=zf.1
2. 8
3. 0 ; ????-BLOCK 3 PROP COMB [START=401506 END=40150B] MINREFS: STK=0/ARG=28, MAX
3. 0 ; USE: dl.1
3. 0 ; DEF: cf.1,zf.1,sf.1,of.1,pf.1,dl.1
```

- Second, we improved the debugger module API in IDA v7.1. While the rest of the API was modernized in v7.0, we had no time to handle the debugger API. Now it is done, and we have new shiny calls (and got rid of the legacy definitions). While at it, we also improved the speed of the binary search and added support for named threads. Unfortunately, the change of the API means that third party debugger plugins need to be ported to IDA v7.1. We prepared a short porting guide for that, please see: here.
- We also improved our DEX loader to handle multidex android packages. They are becoming more and more popular, and this shortcoming was becoming really limiting.



Please pay attention to the segments HEADER2, TYPES2, METHODS2, etc.

- We removed two debugger modules with this release: **WinCE** and **Symbian**. We haven't heard any feedback about them since ages and nobody voiced against it when we polled our users on our forum, so we decided to let the obsolete stuff to disappear.
- There are numerous tiny improvements, both in IDA in the Decompiler. While we do not list all of them below, the output of both tools became clearer and easier to read in many cases.

Complete changelist

- **Processor Modules**
  - + ARM: add remaining ARMv8.1 instructions (SQRDMLAH and SQRDMLSH)
  - + ARM: combine MOV+MOVK sequences into one macro instruction with final immediate value (ARM64)
  - + ARM: use 'imagerel' operator for RVA offsets
  - + MIPS: convert lwl/lwr and similar sequences to the corresponding unaligned load/store macros (ulw/uld/etc.)
  - + PC: added decoding of UD0 and UD1 instructions
  - + PPC: always allow VLE code and offer the user to set all code to VLE when loading binary files

    - + DALVIK: added support for "invoke-polymorphic", "invoke-polymorphic/range", "invoke-custom", "invoke-custom/range" instructions
- **File Formats**
    - + COFF: default to Windows-1252 (or the local 8-bit encoding) for i386 files
    - + Mach-O: IDA now loads symbols present in possible, separate .dSYM file
    - + dex: added support for multidex android packages
    - + dex: added support for DEX file format 038
- **Debugger**
    - + debugger: OSX: introduce SYMBOL_PATH in dbg_macosx.cfg. This can significantly speed up symbol loading when using the Remote Mac OSX Debugger
    - + debugger: added support for thread names
    - + debugger: iOS: support remote process list for iOS 10 and later
    - + debugger: improved the speed of binary search in process memory (1000 times or more)
    - + debugger: use hardware breakpoints for tracing if "use hardware temporary breakpoints" is set
    - + ios_deploy: added "applist" phase
    - + ios_deploy: added "probe" phase
    - + ios_deploy: fix "kill" phase for iOS 10 and later
    - + ios_deploy: fix "proclist" phase for iOS 10 and later
    - + ios_deploy: improved "launch" and "kill" phases so that they work with any arbitrary application (even System apps)
    - + GDB: added support for GDB stubs reporting unavailable registers
    - + GDB: added support for QPassSignals (when an exception is set to pass to application, not suspend, and to be silent)
    - + GDB: improved calculation of maximum packet size supported by remote stub
    - + GDB: improved detection of architecture and setup of register sets
    - - removed WinCE and Symbian debuggers
- **Kernel/Misc**
    - + kernel: added support for 64-bit 'bytes' in ida64
    - + kernel: improved handling of noret functions
    - + kernel: improved heuristics of function detection
    - + kernel: improved switch recognition
    - + kernel: improved system eh region detection and analysis
    - + kernel: made create_generic_linput() and related functions thread-safe
    - + rtti: improved auto detection of 'complete object locator' as referrence before vtable
    - + FLIRT: ICL: added signatures for icl175 (Intel C++ 17.5)
    - + FLIRT: ICL: added signatures for icl180 (Intel C++ 18.0)
    - + FLIRT: ICL: added signatures for icl181 (Intel C++ 18.1)
    - + FLIRT: VC: added signatures for vc1412 (Visual Studio 2017.5)
    - + FLIRT: pelf: added support for x86_64 relocations 41 (R_X86_64_GOTPCRELX) and 42 (R_X86_64_REX_GOTPCRELX)
    - + FLIRT: vc/vc64: added signatures for ucrt 16299 (Windows 10 Fall Creators Update SDK)
    - + TIL: added a type library for Objective-C
    - + TIL: added prototypes for dispatch_sync(), dispatch_async() and other block-related functions to macosx.til
    - + TIL: added struct Block_layout and related _Block_xxx functions to macosx.til

    - + CFG: Added 'Chinese' culture file
    - + CFG: added config variable APPEND_IDB_EXT: append or replace input file extension by the database extension when constructing an IDB name
    - + CFG: removed CACHE_NODE_SIZE, changed config parameter XREF_CACHE_LIMIT to XREF_CACHE_COUNT, the maximum number of xref cache entries may be specified
- **User Interface**
    - + ui/qt: "Export data" will now propose a file name derived from the name of the first item that's being exported
    - + ui/qt: added API to clear, save and retrieve lines from the "Output window"
    - + ui/qt: choosers can now provide tooltips for their less descriptive column names
    - + ui/qt: during debugging, locals & watches can now be searched using Alt+T, Ctrl+T
    - + ui/qt: in addition to the font, it is now possible to specify top & bottom paddings around listing lines
    - + ui/qt: it's now possible to change the font for tabular data widgets
    - + ui/qt: script snippets: when creating a new snippet, inherit the language from the previous one
    - + ui/qt: searching for binary data now lets the user pick what encoding the string literal portions of the input text should be encoded into, in order to perform the search
    - + ui/qt: the "script snippets" editor now shows line numbers on the left
    - + ui/qt: when assigning a string literal to an address, it's now possible to first select the encoding, before selecting the string literal type
    - + ui/txt: mimic linux's behavior for non-ASCII input letters, so that they don't inadvertently maps to unrelated actions
    - + ui: ability to run the current script snippet from anywhere (using Ctrl+Shift+X)
    - + ui: added third set of kernel options to allow modification of AF_DORTTI amd AF_DOEH via UI
- **Scripts & SDK**
    - + SDK: modernized the debugger module api
    - + IDC: string comparison was stopping at the first zero character, now we take into account the real length of strings (which may have zeroes in the middle)
    - + Python,IDC: scripted loaders/procmods/plugins are now loaded in their own namespace by default, preventing namespace pollution (provided the script engine supports it.)
    - + SDK: added 'ui_screen_ea_changed' notification, letting plugins know when something changed the globally-available "current address"
    - + SDK: added format_charlit() to ua.hpp, a finer-grained version of print_charlit()
    - + SDK: deprecated get_min_spd_ea()
    - + SDK: tryblks: added find_syseh()
- **Decompilers**
    - + exported microcode api
    - + added UI action "Set call type..."
    - + improved handling of struct field reconstruction
    - + introduced KERNEL_NREGS configuration parameter
    - + introduced a rule to get rid of __OFSUB__ in some cases
    - + made numerous minor improvements to the optimization engine
    - + pc: added support for the xadd instruction

    - + recognize floating-point arithmetic aeabi helper functions
    - + recognize memory management helpers aeabi_memcpy*, aeabi_memset*, aeabi_memclr*
    - + simplified if-then-else construction with identical branches
    - + use a user-defined stroff instead of udt field references if there is a contradiction between them
    - + x64: added support for the syscall instruction (convert into linux system calls when possible)
- **BUGFIXES**
    - BUGFIX: pc: fixed interr 10148
    - BUGFIX: DWARF: The plugin could show a warning, when it couldn't make sense of duplicate typedefs
    - BUGFIX: Editing script snippets & closing their window while no IDB was loaded, could cause IDA to crash when the user decides to load an IDB
    - BUGFIX: En-masse application of some operations (e.g., set immediate radix to octal) would fail for operands beyond the second
    - BUGFIX: En-masse offset conversion could fail for values that pointed to addresses that had uninitialized data
    - BUGFIX: En-masse offset conversion could leave some valid values unconverted
    - BUGFIX: En-masse offset conversion would turn values that don't fall within the specified range into offsets nonetheless
    - BUGFIX: For structure members with a string literal type, certain string layouts (e.g., delphi with 4-bytes length header) wasn't respected
    - BUGFIX: IDA SDK 7.0's plugins/script_plg/procext.py was not ported to the IDA 7.0 processor modules API
    - BUGFIX: IDA could INTERR (40481) in case the user was switching back & forth between flat & graph view on a function that is hidden, and in the beginning of a segment which displays some additional information
    - BUGFIX: IDA could crash with out of memory error in some ARM64 files, especially if using the decompiler
    - BUGFIX: IDA's debugger memory cache could produce bad cache hits after a failed read at address 0
    - BUGFIX: IDAPython/bc695: action_ctx_base_t::form & action_ctx_base_t::form_title were missing
    - BUGFIX: IDAPython/bc695: many "inf" fields were not accessible anymore (e.g., 'mf')
    - BUGFIX: IDAPython: -1 couldn't be passed as 'len' to get_strlit_contents()
    - BUGFIX: IDAPython: Using execute_sync() with MFF_NOWAIT could cause IDA to abort()
    - BUGFIX: IDAPython: backward-compatible multi-choosers would lose their selection on double-click/enter
    - BUGFIX: IDAPython: deriving from IDP_Hooks, could cause error messages of the form "ValueError: invalid null reference in method 'IDP_Hooks_ev_adjust_argloc'" to be printed in the "Output window"
    - BUGFIX: IDAPython: failure to give a plugin a "wanted_name", could result in IDA crashing
    - BUGFIX: IDAPython: ida_diskio.get_ida_subdirs() was unusable
    - BUGFIX: IDAPython: ida_hexrays.cexpr_t()'s copy-constructor wouldn't create a duplicate of the underlying C tree

- BUGFIX: IDAPython: processor_t.is_sp_based() was not usable in case one followed the example in proctemplate.py
- BUGFIX: IDAPython: processor_t.set_func_start()/set_func_end() notifications were unusable
- BUGFIX: IDAPython: rebase_program() wouldn't accept a delta with its Most-Significant-Bit set to 1 (e.g., 0x80000000), making it very difficult to use
- BUGFIX: IDAPython: tab completion could fail to work, depending on the processor module (e.g., ARM)
- BUGFIX: In the 'Structures' window, structure members with string literal types whose encoding takes up more than 2 bytes per unit (e.g., UTF-32), would be represented as 'word's instead of 'dword's
- BUGFIX: Opening script snippets without a database opened, would result in the default snippet having an empty name
- BUGFIX: PDBs with a mix of unions and anonymous structures could cause interr 815
- BUGFIX: Retrieving the name of an address which is the last address of a function chunk that is itself placed right before the main function chunk, could provide erroneous results
- BUGFIX: TRICORE: register tracker-calculated offsets could override user-defined ones
- BUGFIX: Typing '0x23' in the calculator (i.e., hotkey '?') would result in truncated character display
- BUGFIX: gdb: IDA would not show a wait box for "Run until return"
- BUGFIX: loading a pdb file would add the current idb in the file history in the File menu
- BUGFIX: pdb plugin could crash the msdia library during remote debugging
- BUGFIX: qlist::swap() was broken
- BUGFIX: refreshing a chooser could cause IDA to hang in very specific situations
- BUGFIX: ui/qt: "Report a bug/issue" dialog's text edit had empty space to the right of the text area
- BUGFIX: ui/qt: after setting a structure member as a string literal, re-opening the dialog wouldn't show the right strlit type
- BUGFIX: ui/qt: in graph view, going to beginning/end of a long line in a long node, could result in the node moving out of the screen
- BUGFIX: ui/qt: navigating the preview listing with the keyboard in the "IDA Colors" configuration dialog was impossible
- BUGFIX: ui/qt: saving shortcuts from the shortcuts editor would fail for actions whose shortcut was cleared
- BUGFIX: ui/qt: setting field type in unions could end up truncating the union
- BUGFIX: ui/qt: when an automatic snapshot is taken, don't show the wait dialog, as it might interfere with the user's work
- BUGFIX: ui/txt: idat[64] could crash at exit-time when decompiler views were created
- BUGFIX: hexrays: ARM cinv instruction was decompiled incorrectly
- BUGFIX: hexrays: combining rule 24 (64bit negation) could produce incorrect code
- BUGFIX: hexrays: decompiler could crash after renaming a function argument on the stack if the previous user action was to change the type of a function argument type to a non-compatible type
- BUGFIX: hexrays: deleting user-defined label was impossible

- BUGFIX: hexrays: in some cases combination of 64bit operands could produce incorrect results
- BUGFIX: hexrays: in some rare cases combining rule for 64bit addition could produce wrong code
- BUGFIX: hexrays: it was impossible to rename the "result" variable
- BUGFIX: hexrays: renaming a function in the pseudocode window could cause a crash
- BUGFIX: hexrays: fixed many interrs
- BUGFIX: when debugging an iOS framework, the HEADER segment for the exe module would be missing