Reverse Engineering and Code Emulation with Ghidra

Karl 'Rosie' Sickendick - @kc0bfv

October 2019

Outline

Section 0

Who Am I?

Who Am I?

- Karl 'Rosie' Sickendick - @kc0bfv
- Active duty Major in the Air Force
- Stationed at Idaho National Labs as part of an Air Force fellowship program
- BS in EE, MS in CS
- Experience with: cyber capability and tactics development
- Very happy to be here!

Section 1

Introduction to Ghidra

Ghidra Intro

- Comrehensive software reverse engineering tool
  - NSA-produced, open source
  - Your tax dollars at work!
- Features a decompiler that supports many architectures
  - Sleigh: understands the architecture, bytes -> PCode
  - PCode: architecture-agnostic intermediate language
  - Decompiler: does data-flow analysis on the PCode
- "Easily" extensible to new architectures
  - Build a Sleigh module, map bytes -> instructions -> PCode
  - Get the decompiler for free

Ghidra Intro

- Setup
  - Install OpenJDK 11
  - Download zip: `https://ghidra-sre.org/`
  - Unzip somewhere you can write
  - Make sure Java is in your PATH...
  - 'Dual monitors strongly suggested'...
  - `GHIDRA_INSTALL/ghidra_version/ghidraRun`

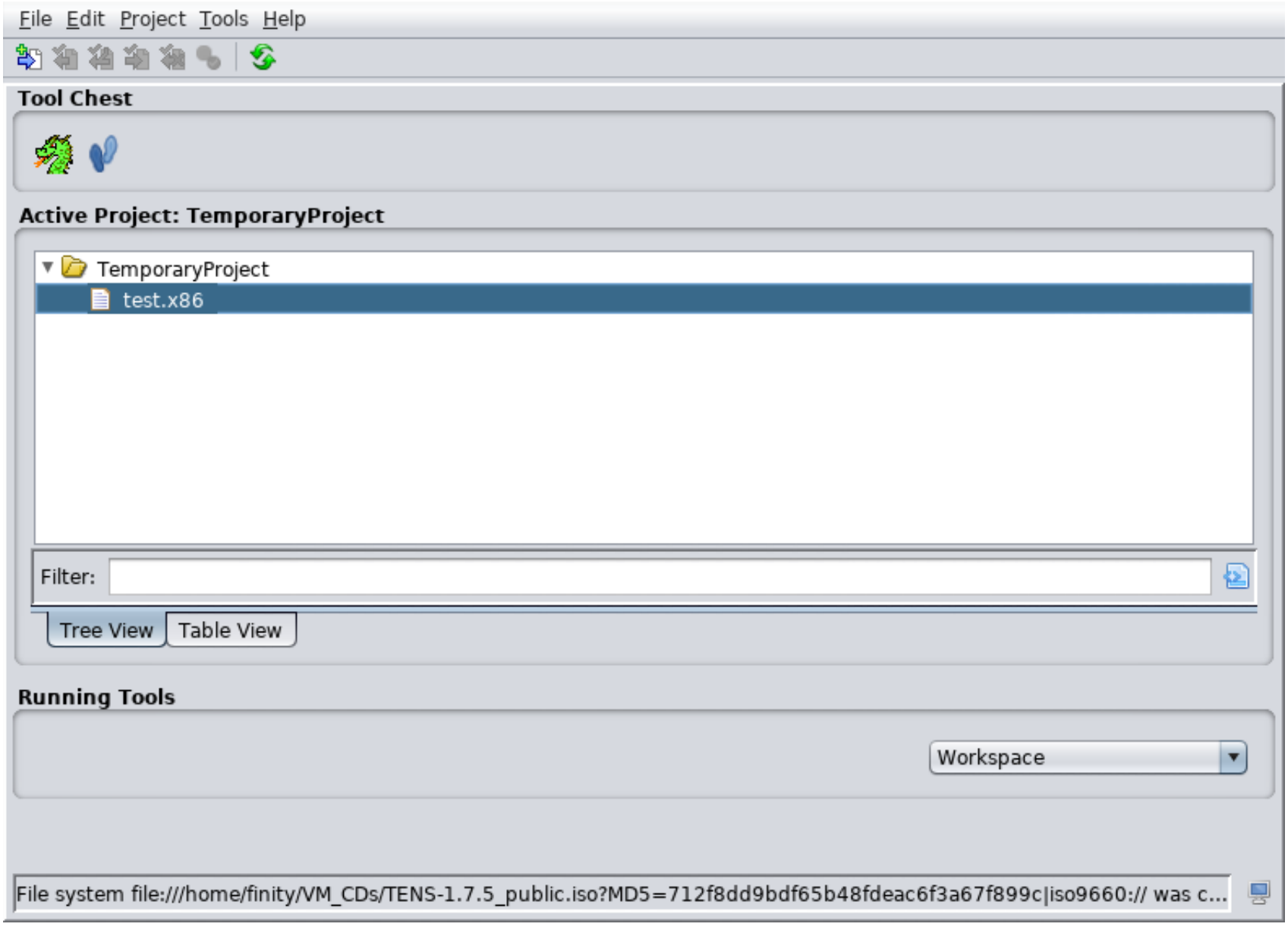
Ghidra Organization

- Local
  - Non-Shared Projects
    - Folders
    - Executables
    - Libraries
    - History
    - Version Control
    - Data Types/Structures/Classes
- Server
  - Shared Projects
    - ... Same as Non-Shared ...
    - Check-out/Check-in
    - Change Merging
- File Systems
  - File Systems
    - ...
  - Import Files

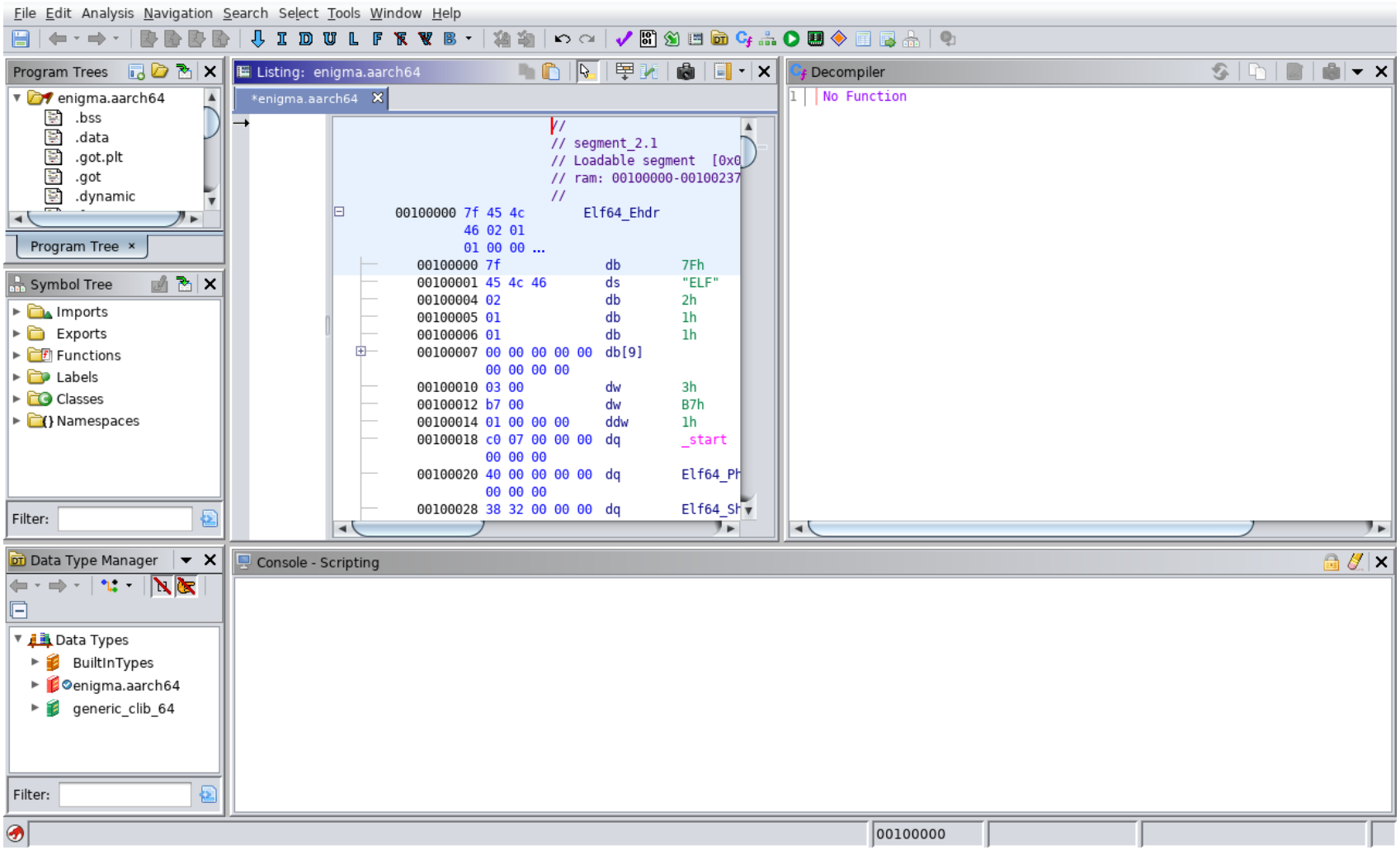
Importing Files & Supported Types

- 6502 (NES? Atari?)
- Motorolla: 6805, 68000
- Intel: 8048, 8051, 8085, 80251
- ARM & AARCH64 {big,little} endian
- AVR{8,32}
- JVM
- MIPS {32,64} bit {big,little} endian
- PIC-{12,16,17,18,24}
- PowerPC {32,64} bit {big,little} endian
- Sparc
- TI_MSP430
- x86 {16,32,64} bit
- Z{80,180}

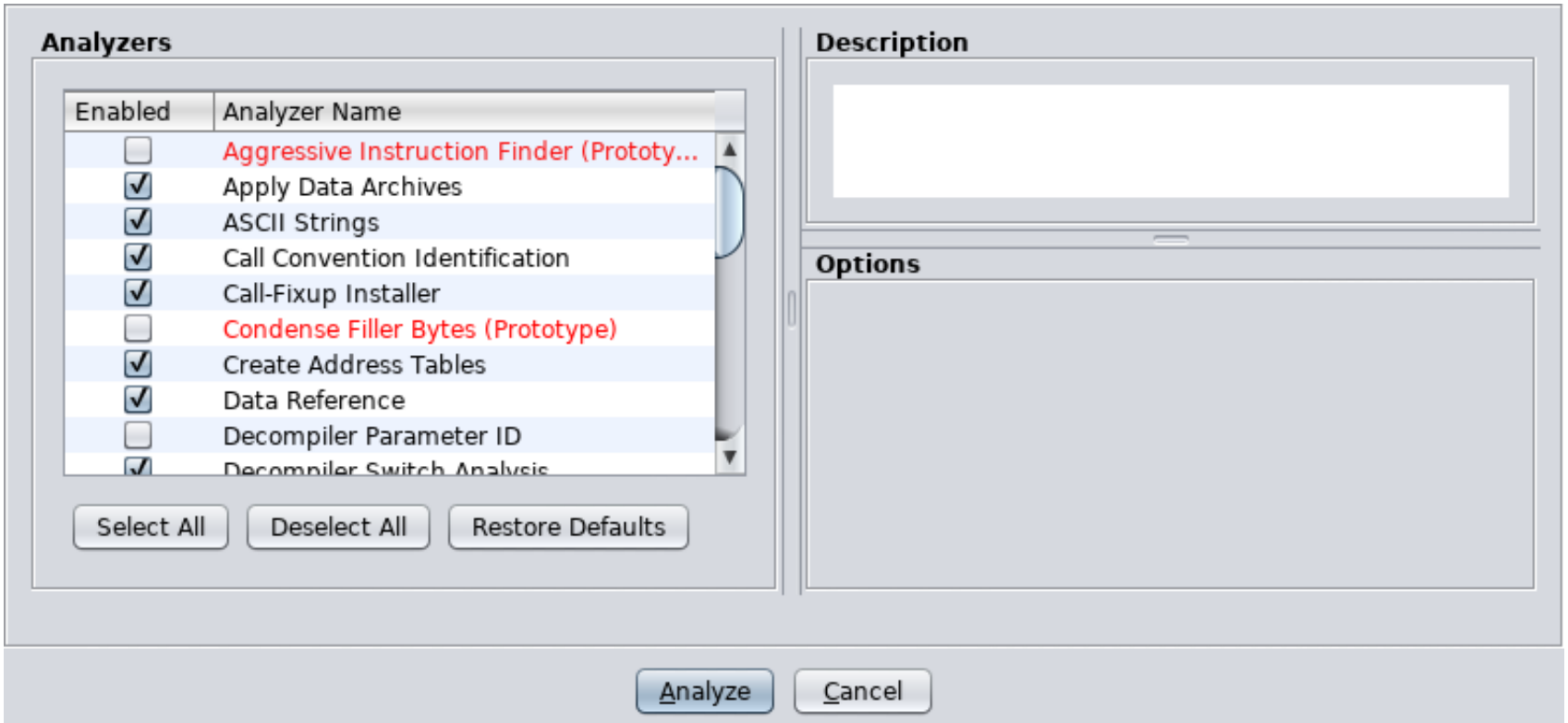
Importing Files & Supported Types



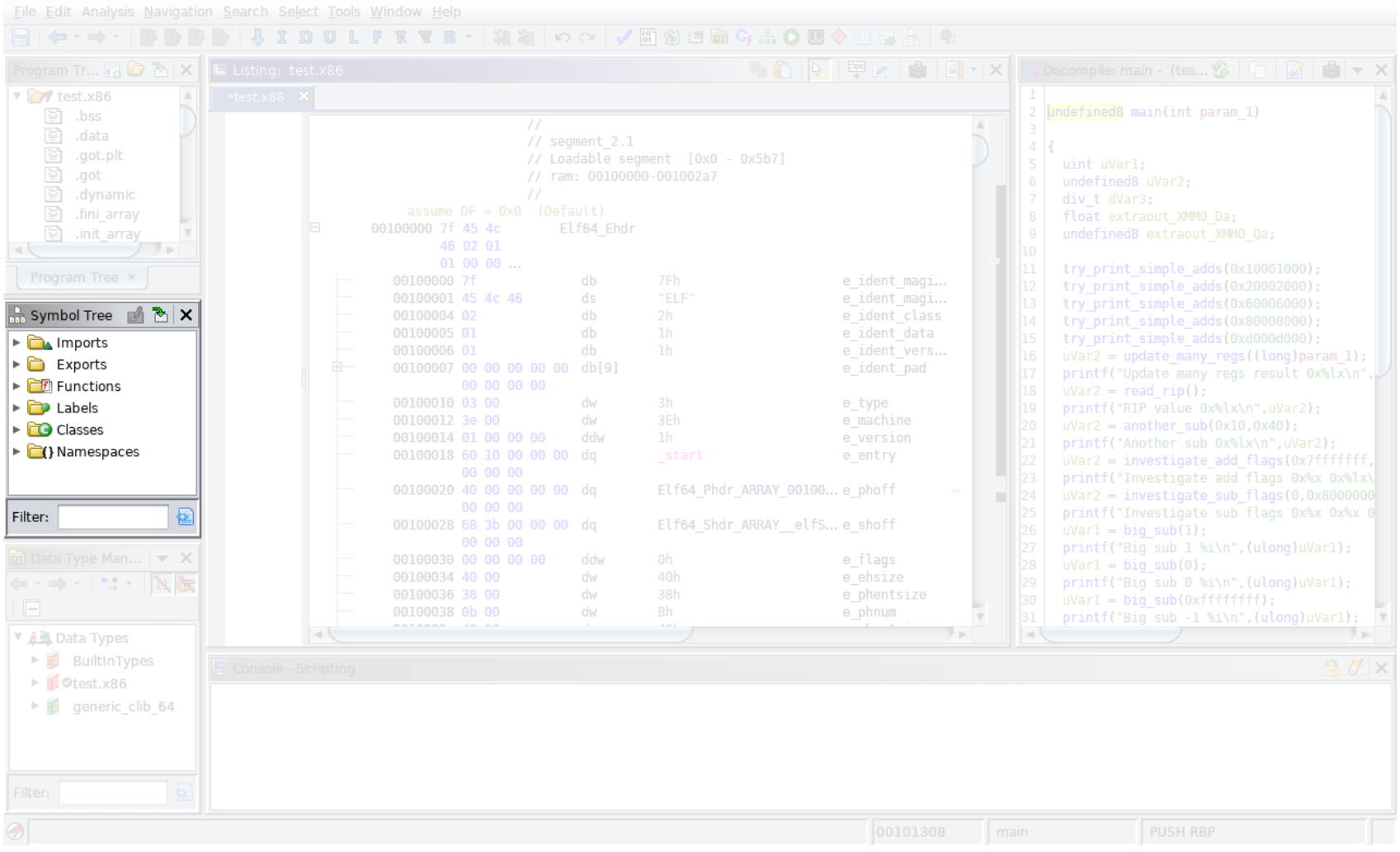
Starting Analysis



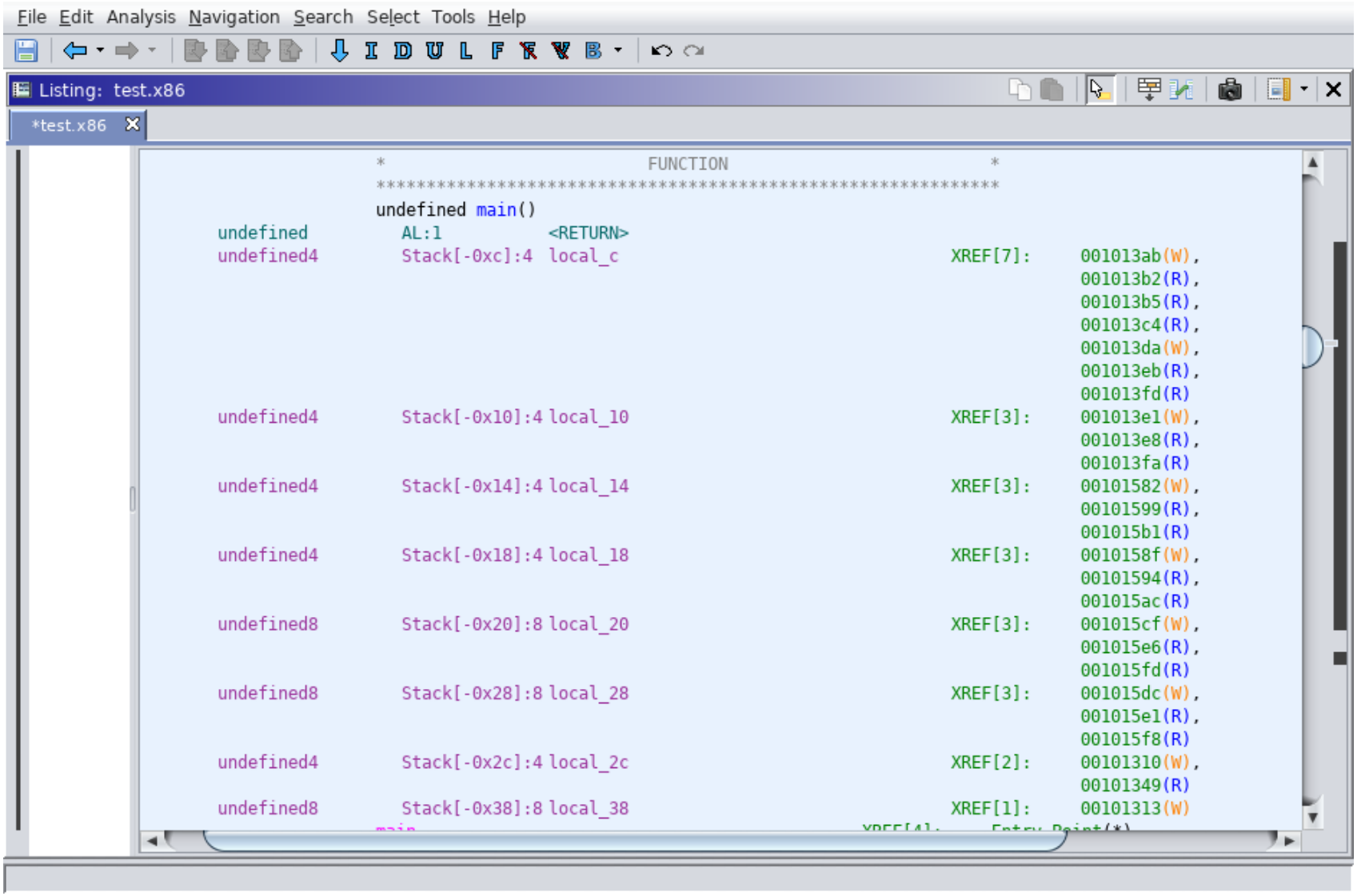
Starting Analysis



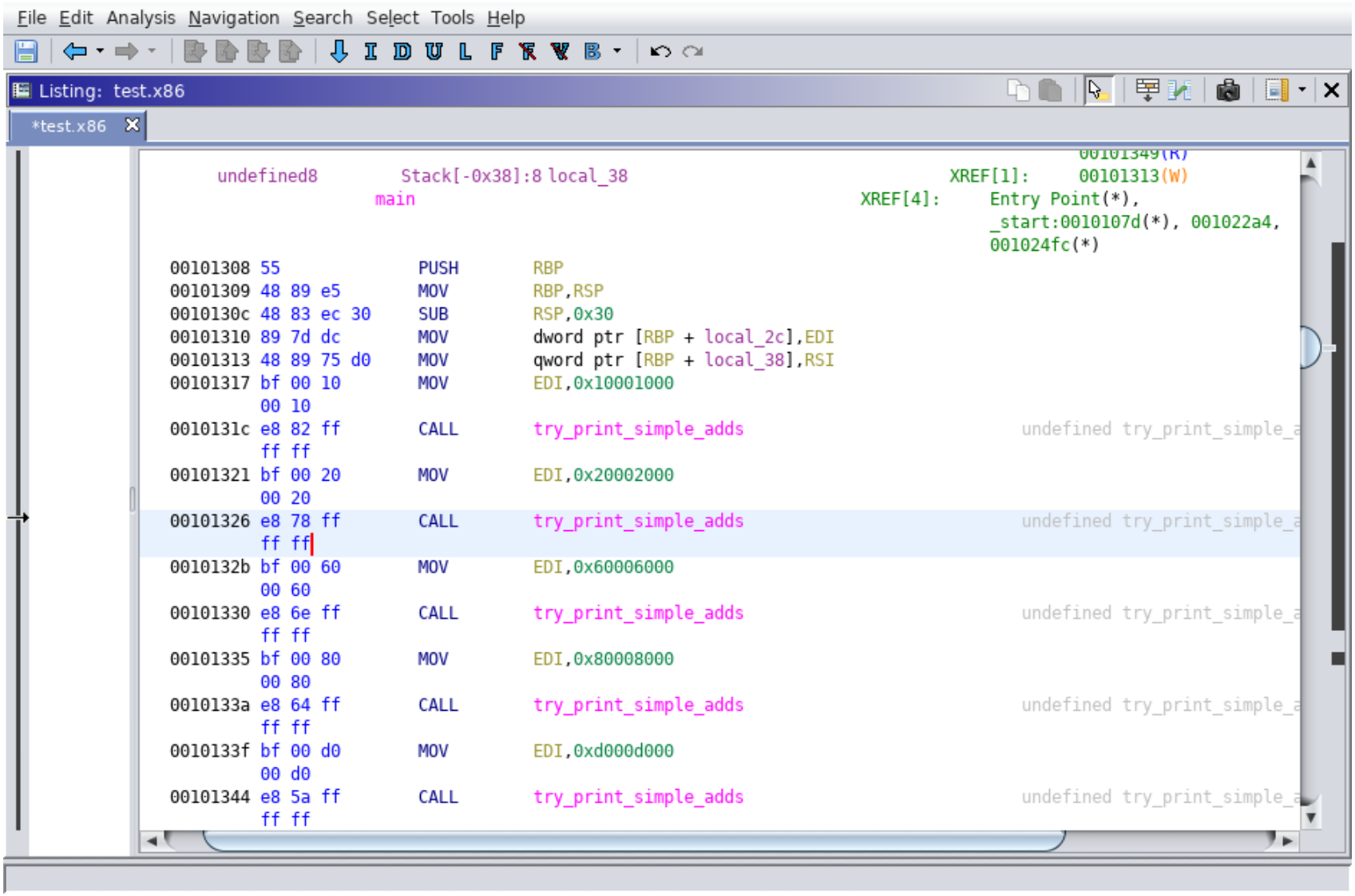
Symbol Tree


File Edit Analysis Navigation Search Select Tools Window Help
Program Tr...
test.x86
.bss
.data
.got.plt
.got
.dynamic
.fini_array
.init_array
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Man...
Data Types
BuiltInTypes
test.x86
generic_clib_64
Filter:
Listing: test.x86
*test.x86
// segment_2.1
// Loadable segment [0x0 - 0x5b7]
// ram: 00100000-001002a7
assume DF = 0x0 (Default)
00100000 7f 45 4c Elf64_Ehdr
46 02 01
01 00 00 ...
00100000 7f db 7Fh e_ident_magi...
00100001 45 4c 46 ds "ELF" e_ident_magi...
00100004 02 db 2h e_ident_class
00100005 01 db 1h e_ident_data
00100006 01 db 1h e_ident_vers...
00100007 00 00 00 00 00 db[9] e_ident_pad
00 00 00 00
00100010 03 00 dw 3h e_type
00100012 3e 00 dw 3Eh e_machine
00100014 01 00 00 00 ddw 1h e_version
00100018 60 10 00 00 00 dq _start e_entry
00 00 00
00100020 40 00 00 00 00 dq Elf64_Phdr_ARRAY_00100... e_phoff
00 00 00
00100028 68 3b 00 00 00 dq Elf64_Shdr_ARRAY__elfS... e_shoff
00 00 00
00100030 00 00 00 00 ddw 0h e_flags
00100034 40 00 dw 40h e_ehsize
00100036 38 00 dw 38h e_phentsize
00100038 0b 00 dw 8h e_phnum
Decompile: main - (tes...
undefined8 main(int param_1)
{
uint uVar1;
undefined8 uVar2;
div_t dVar3;
float extraout_XMM0_Da;
undefined8 extraout_XMM0_Qa;
try_print_simple_adds(0x10001000);
try_print_simple_adds(0x20002000);
try_print_simple_adds(0x60006000);
try_print_simple_adds(0x80008000);
try_print_simple_adds(0xd000d000);
uVar2 = update_many_regs((long)param_1);
printf("Update many regs result 0x%lx\n",
uVar2 = read_rip();
printf("RIP value 0x%lx\n",uVar2);
uVar2 = another_sub(0x10,0x40);
printf("Another sub 0x%lx\n",uVar2);
uVar2 = investigate_add_flags(0x7fffffff,
printf("Investigate add flags 0x%x 0x%lx\
uVar2 = investigate_sub_flags(0,0x8000000
printf("Investigate sub flags 0x%x 0x%x 0
uVar1 = big_sub(1);
printf("Big sub 1 %i\n",(ulong)uVar1);
uVar1 = big_sub(0);
printf("Big sub 0 %i\n",(ulong)uVar1);
uVar1 = big_sub(0xffffffff);
printf("Big sub -1 %i\n",(ulong)uVar1);
Console - Scripting
00101308
main
PUSH RBP

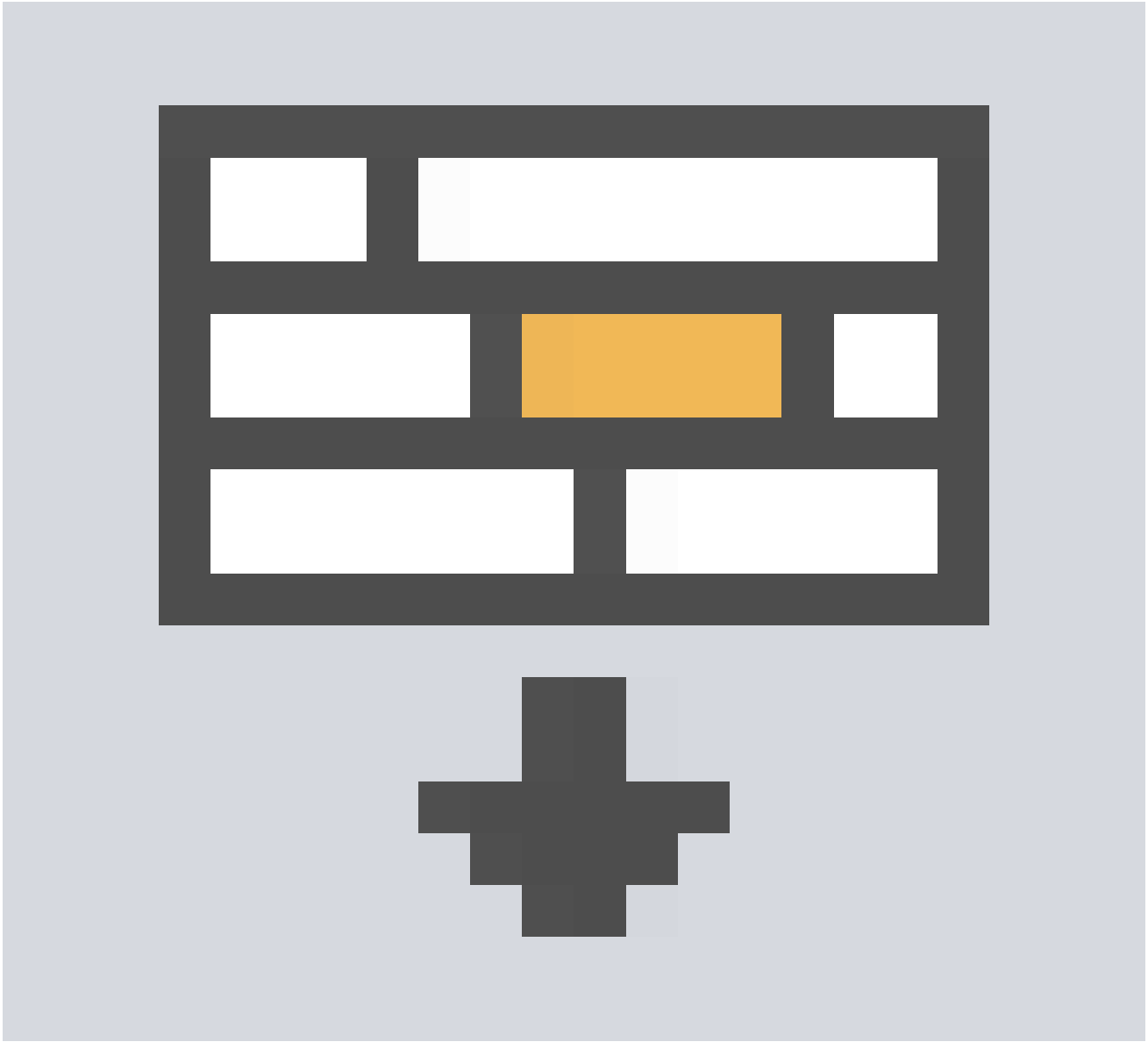
Program Listing



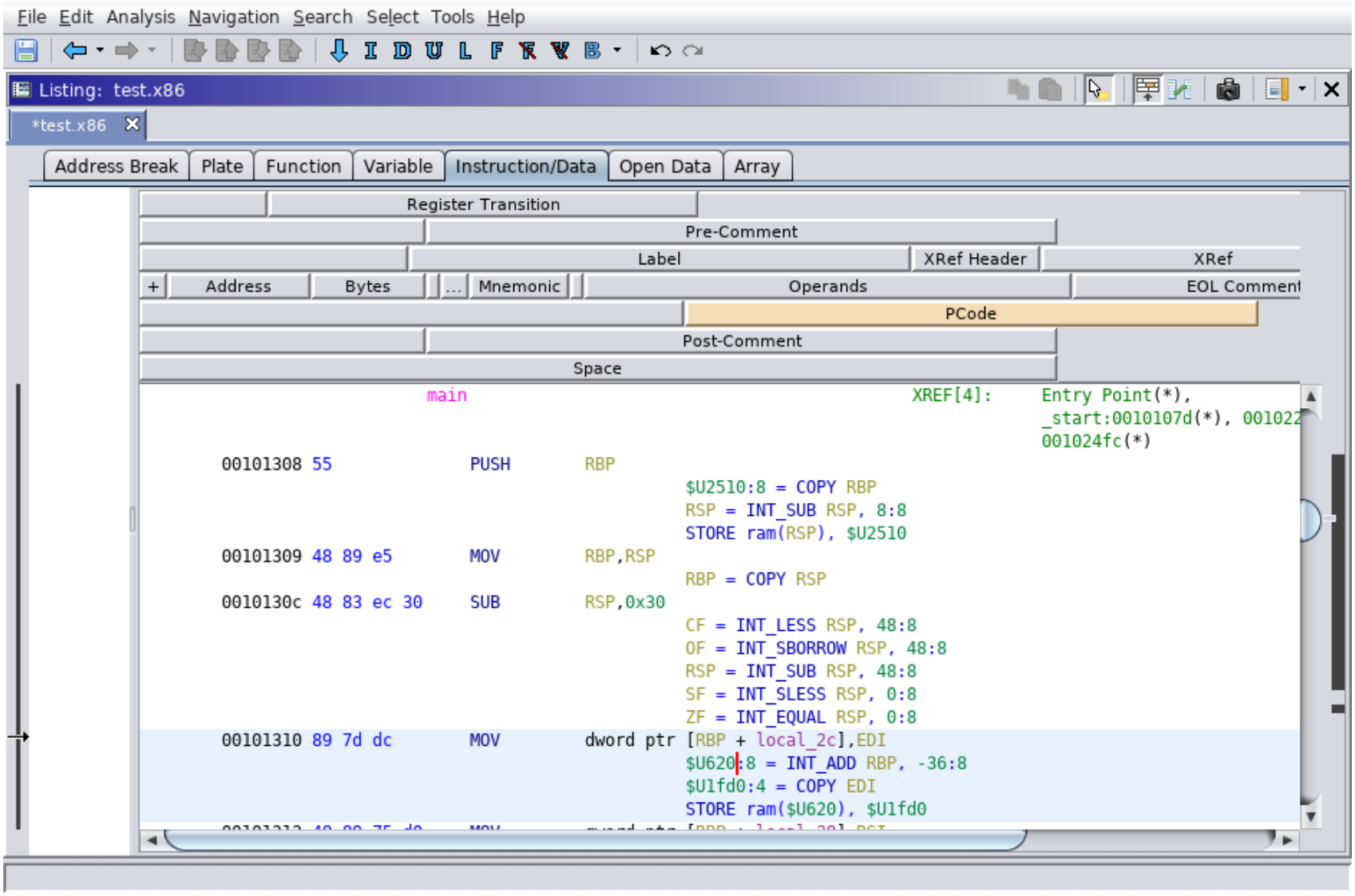
Program Listing



Showing PCode



Showing PCode

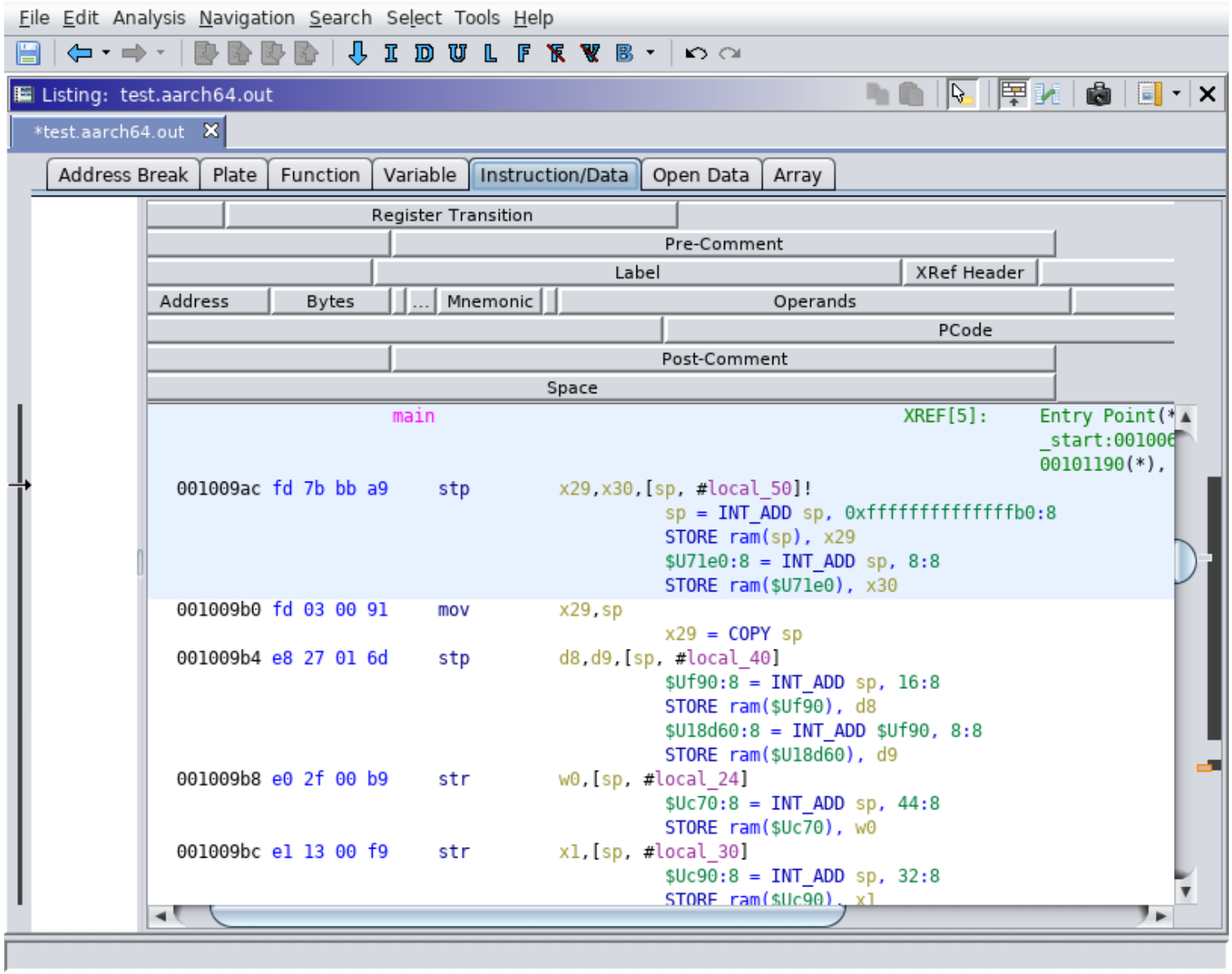


PCode Explanation

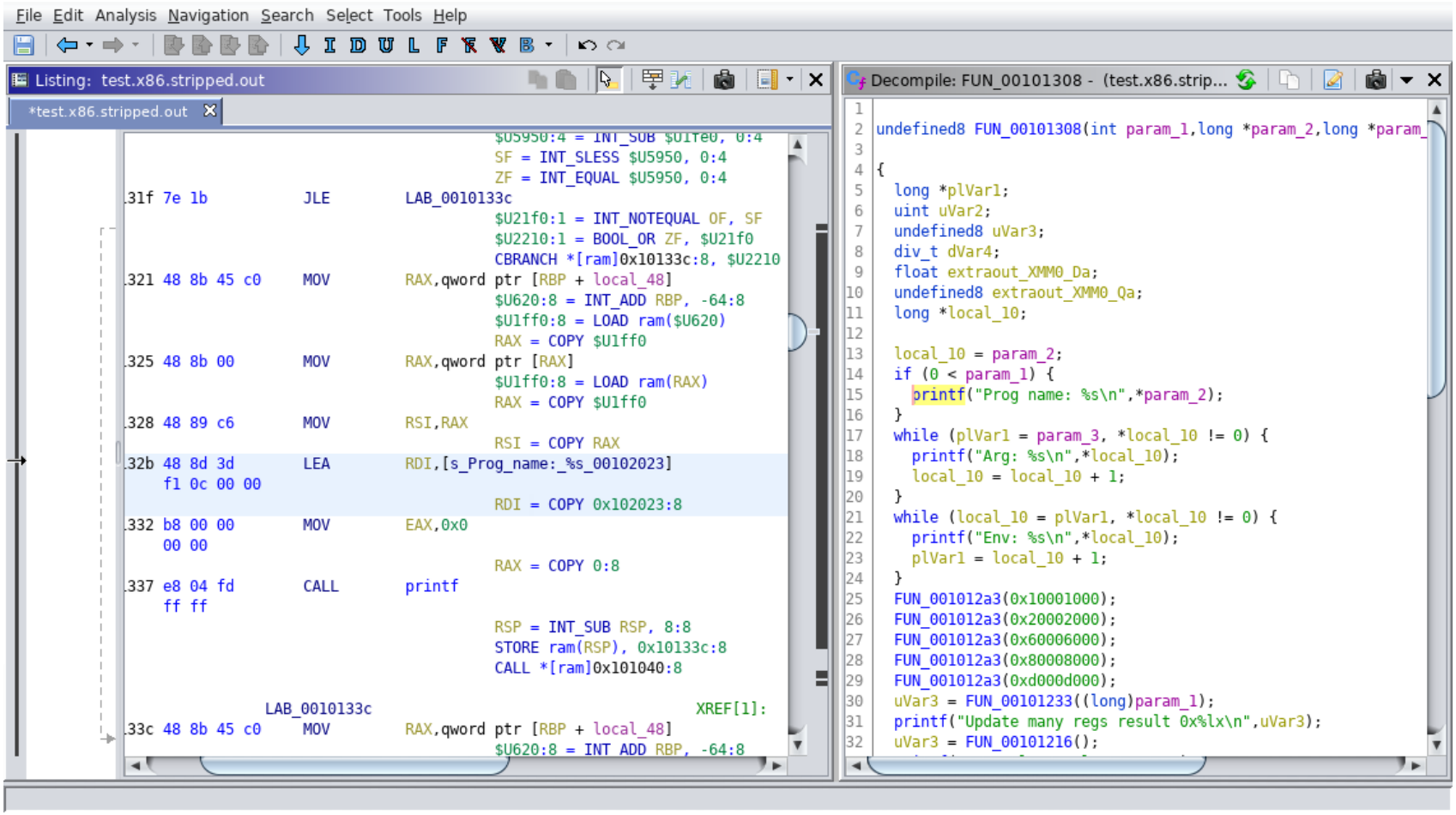
```
PUSH        RBP
```

```
$U2510:8 = COPY RBP
RSP = INT_SUB RSP, 8:8
STORE ram(RSP), $U2510
```

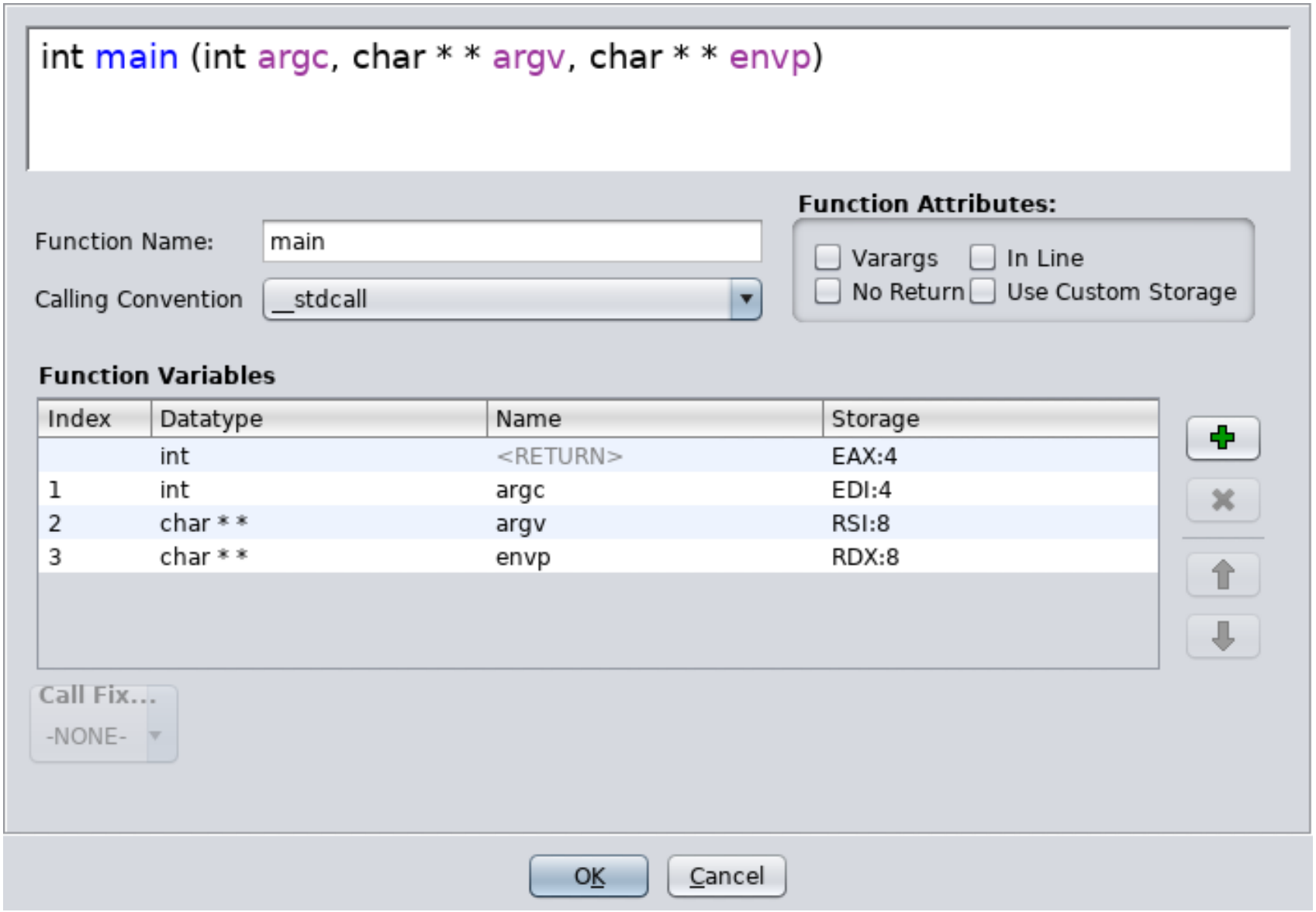
PCode Explanation



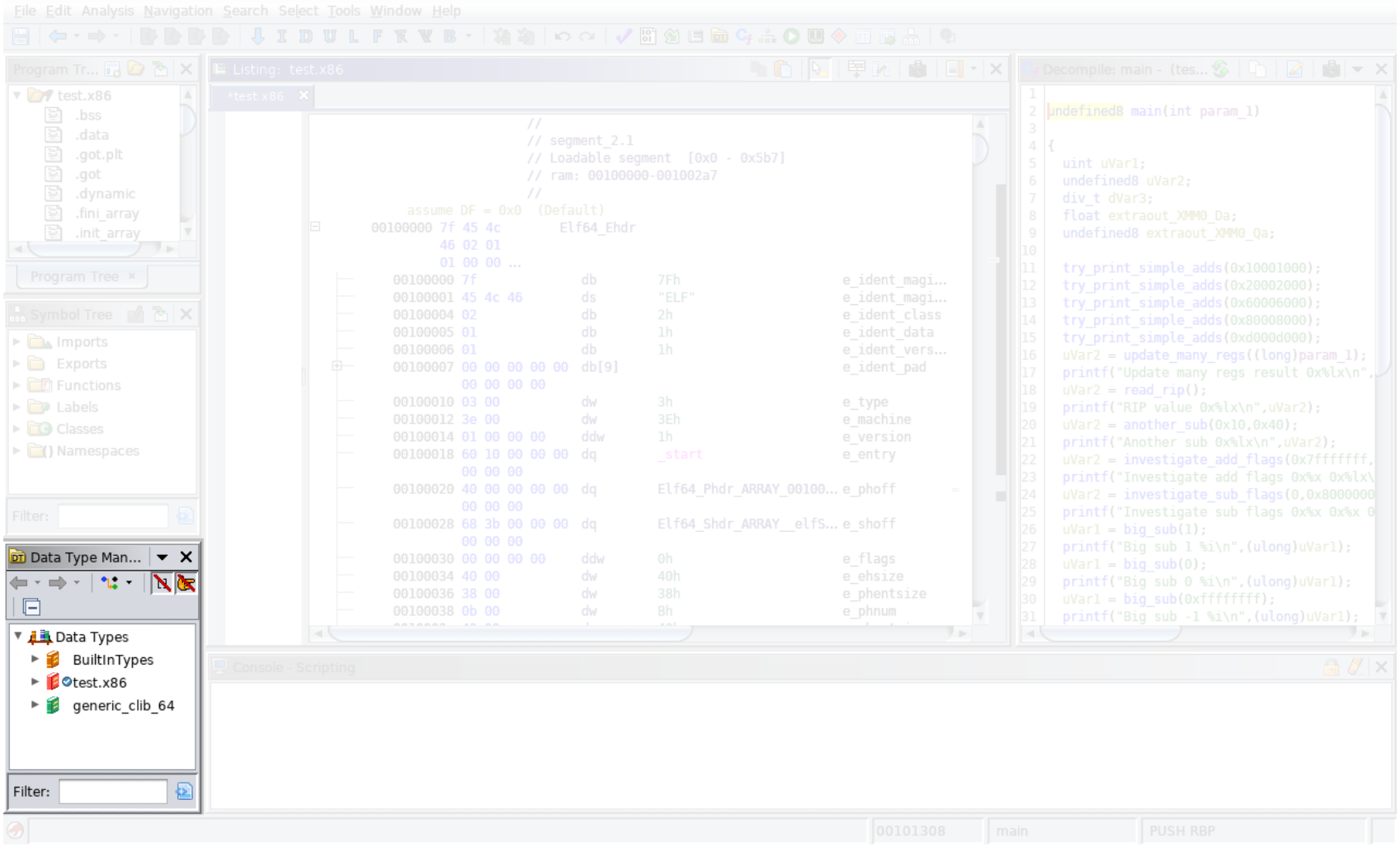
Decompilation



Improving Decompilation - Function Signatures



Improving Decompilation - Structures

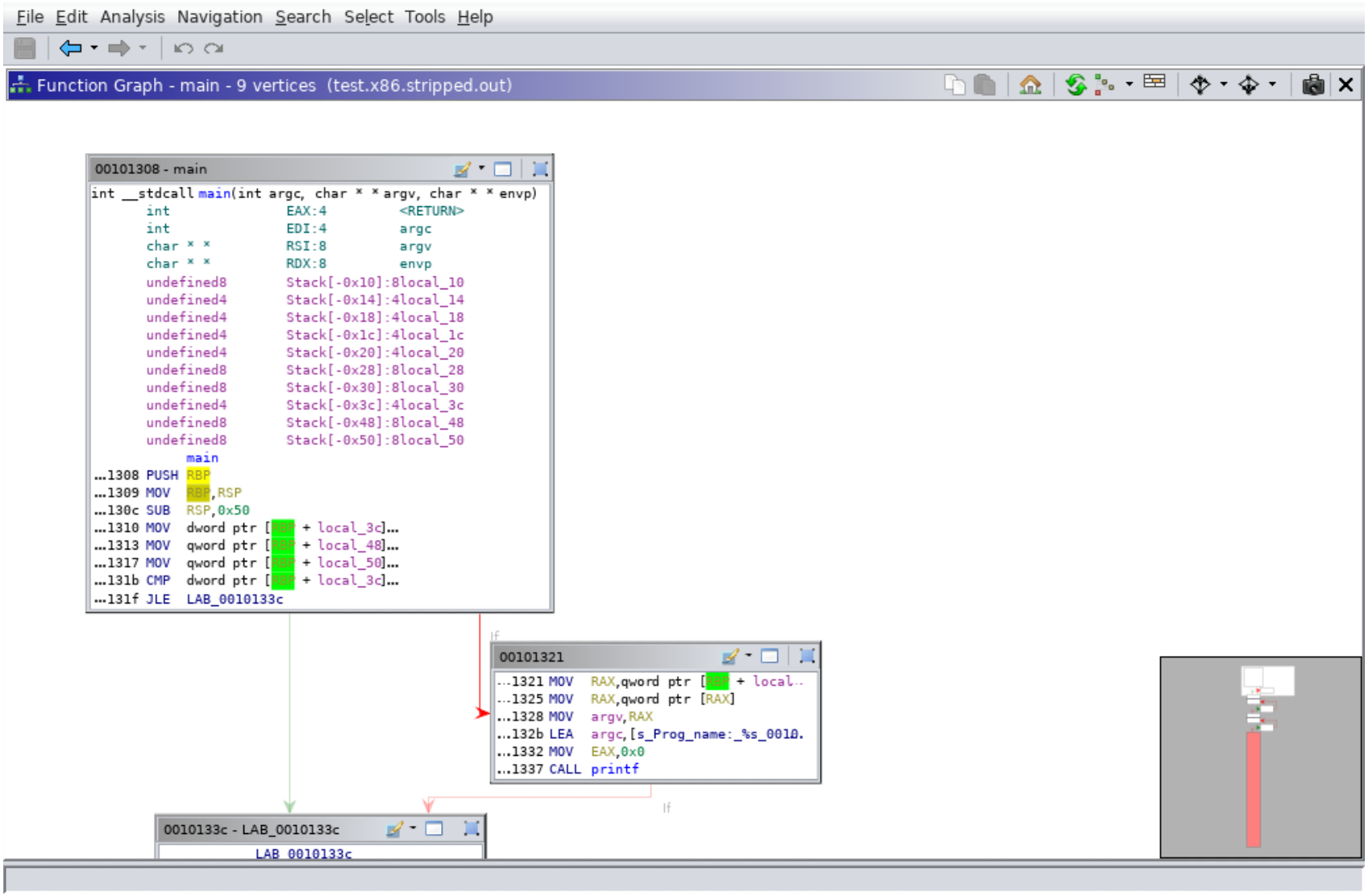


Improving Decompilation - Structures

Decompile: FUN_00101318 - (test.x86.stripped.out W...

```
1
2  ulong FUN_00101318(__time_t param_1,long param_2)
3
4  {
5    timespec local_38;
6    timespec local_28;
7    uint local_10;
8    int local_c;
9
10   local_38.tv_sec = 0;
11   local_38.tv_nsec = 0;
12   local_c = -1;
13   local_10 = 0;
14   local_28.tv_sec = param_1;
15   local_28.tv_nsec = param_2;
16   while (local_c != 0) {
17     local_10 = local_10 + 1;
18     local_c = nanosleep(&local_28,&local_38);
19     local_28.tv_sec = local_38.tv_sec;
20     local_28.tv_nsec = local_38.tv_nsec;
21   }
22   return (ulong)local_10;
23 }
24
```

Function Graphs

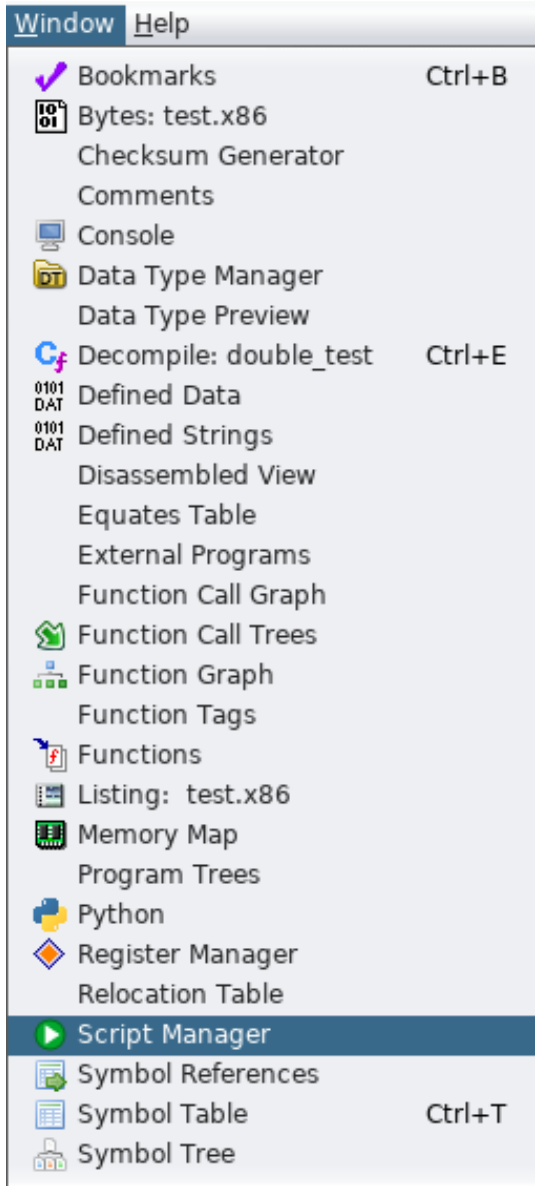


Other Features

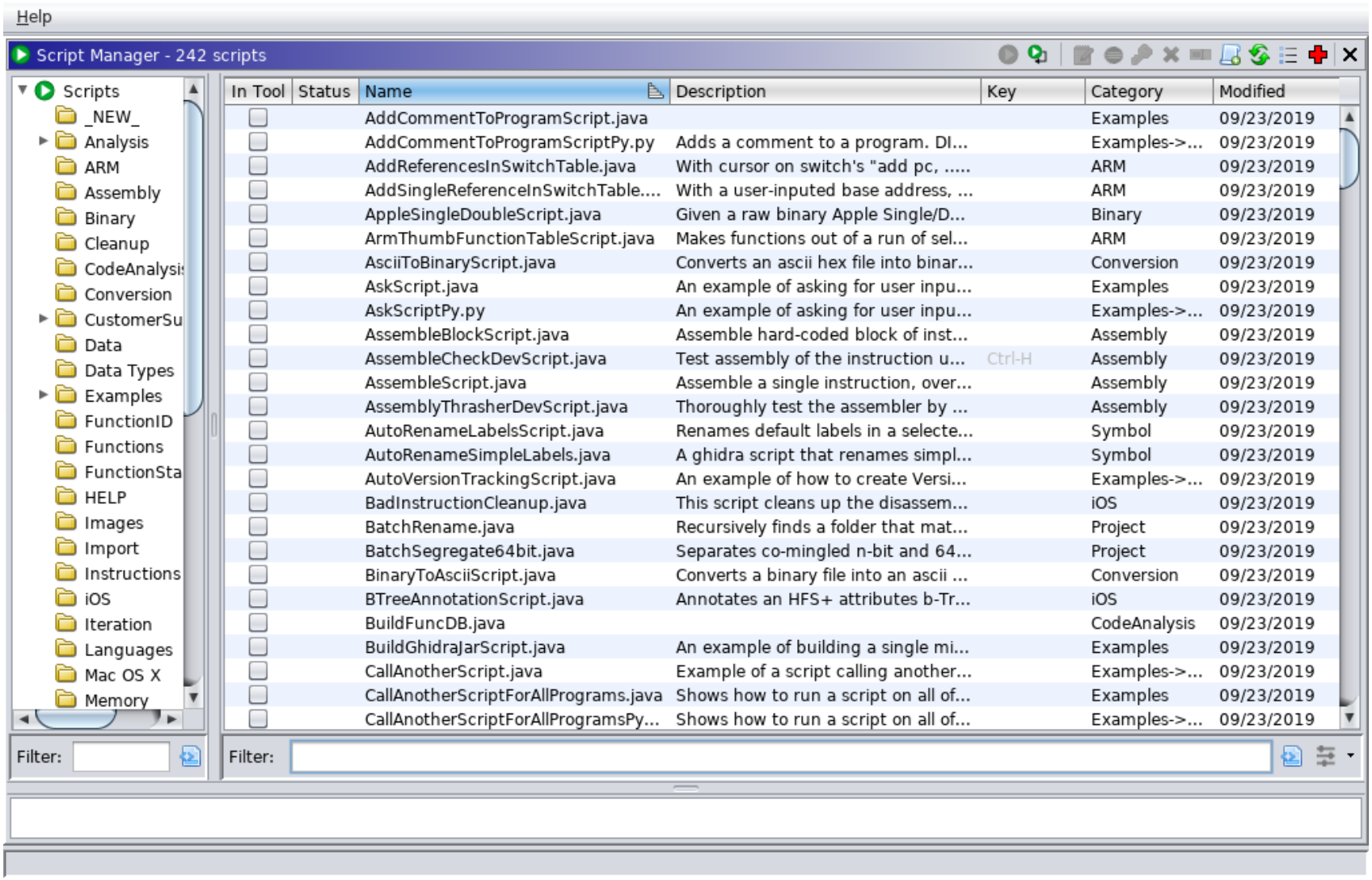
- Undo!
- Middle clicking an entity to highlight all instances
- Binary view
- Highlight interdependence as forward and backward slices
- Function Call Trees
- C++ classes
- Linking to other binaries
- Server mode
- Scripts

UNDO!

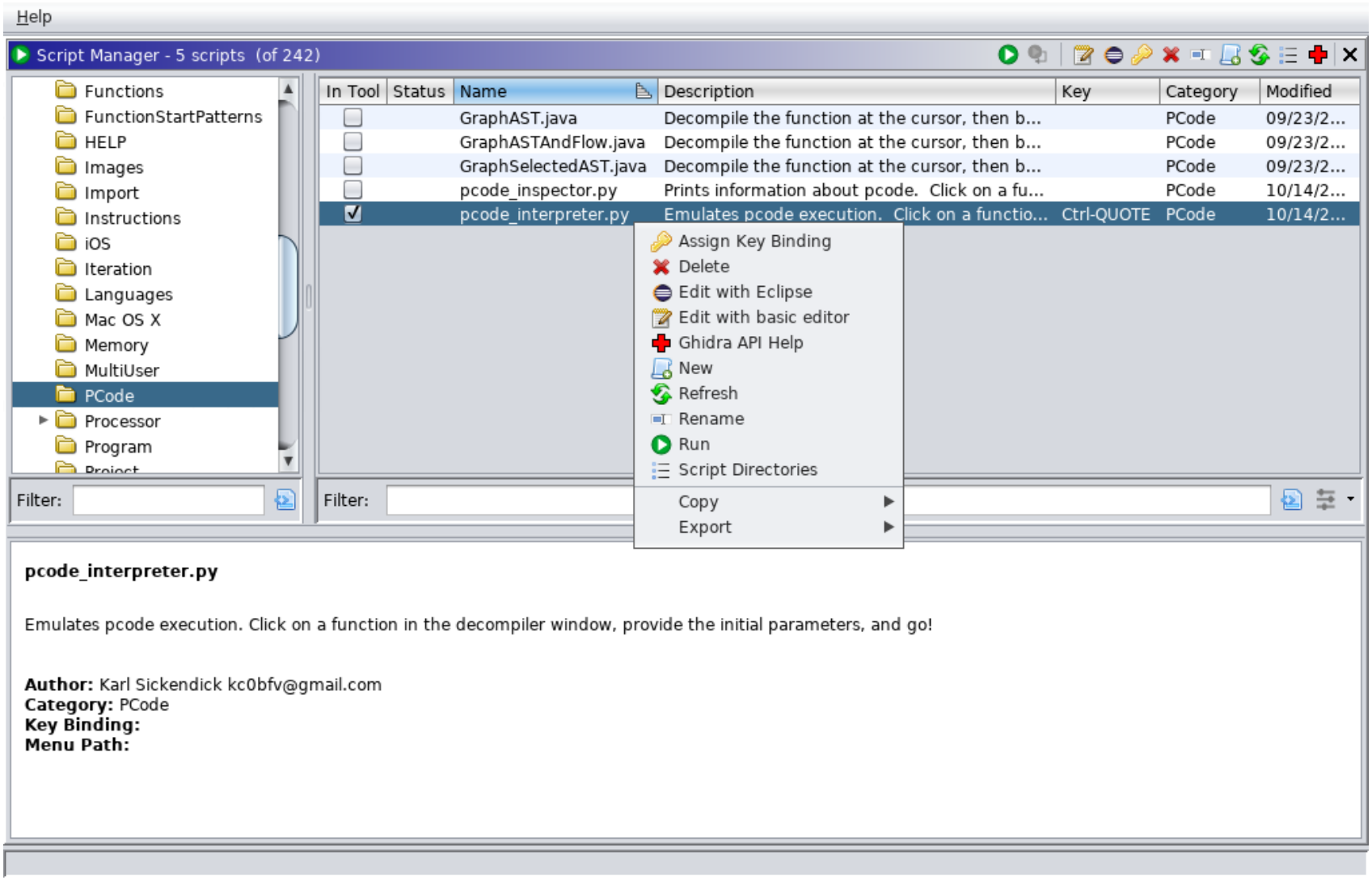
Scripts


Window
Help
Bookmarks Ctrl+B
Bytes: test.x86
Checksum Generator
Comments
Console
Data Type Manager
Data Type Preview
Decompile: double_test Ctrl+E
Defined Data
Defined Strings
Disassembled View
Equates Table
External Programs
Function Call Graph
Function Call Trees
Function Graph
Function Tags
Functions
Listing: test.x86
Memory Map
Program Trees
Python
Register Manager
Relocation Table
Script Manager
Symbol References
Symbol Table Ctrl+T
Symbol Tree

Scripts


Help
Script Manager - 242 scripts
Scripts
NEW
Analysis
ARM
Assembly
Binary
Cleanup
CodeAnalysis
Conversion
CustomerSu
Data
Data Types
Examples
FunctionID
Functions
FunctionSta
HELP
Images
Import
Instructions
iOS
Iteration
Languages
Mac OS X
Memory
Filter:
In Tool | Status | Name | Description | Key | Category | Modified
AddCommentToProgramScript.java | Examples | 09/23/2019
AddCommentToProgramScriptPy.py | Adds a comment to a program. DI... | Examples->... | 09/23/2019
AddReferencesInSwitchTable.java | With cursor on switch's "add pc, ... | ARM | 09/23/2019
AddSingleReferenceInSwitchTable.... | With a user-inputed base address, ... | ARM | 09/23/2019
AppleSingleDoubleScript.java | Given a raw binary Apple Single/D... | Binary | 09/23/2019
ArmThumbFunctionTableScript.java | Makes functions out of a run of sel... | ARM | 09/23/2019
AsciiToBinaryScript.java | Converts an ascii hex file into binar... | Conversion | 09/23/2019
AskScript.java | An example of asking for user inpu... | Examples | 09/23/2019
AskScriptPy.py | An example of asking for user inpu... | Examples->... | 09/23/2019
AssembleBlockScript.java | Assemble hard-coded block of inst... | Assembly | 09/23/2019
AssembleCheckDevScript.java | Test assembly of the instruction u... | Ctrl-H | Assembly | 09/23/2019
AssembleScript.java | Assemble a single instruction, over... | Assembly | 09/23/2019
AssemblyThrasherDevScript.java | Thoroughly test the assembler by ... | Assembly | 09/23/2019
AutoRenameLabelsScript.java | Renames default labels in a selecte... | Symbol | 09/23/2019
AutoRenameSimpleLabels.java | A ghidra script that renames simpl... | Symbol | 09/23/2019
AutoVersionTrackingScript.java | An example of how to create Versi... | Examples->... | 09/23/2019
BadInstructionCleanup.java | This script cleans up the disassem... | iOS | 09/23/2019
BatchRename.java | Recursively finds a folder that mat... | Project | 09/23/2019
BatchSegregate64bit.java | Separates co-mingled n-bit and 64... | Project | 09/23/2019
BinaryToAsciiScript.java | Converts a binary file into an ascii ... | Conversion | 09/23/2019
BTreeAnnotationScript.java | Annotates an HFS+ attributes b-Tr... | iOS | 09/23/2019
BuildFuncDB.java | CodeAnalysis | 09/23/2019
BuildGhidraJarScript.java | An example of building a single mi... | Examples | 09/23/2019
CallAnotherScript.java | Example of a script calling another... | Examples->... | 09/23/2019
CallAnotherScriptForAllPrograms.java | Shows how to run a script on all of... | Examples | 09/23/2019
CallAnotherScriptForAllProgramsPy... | Shows how to run a script on all of... | Examples->... | 09/23/2019
Filter:

Scripts


Help
Script Manager - 5 scripts (of 242)
Functions
FunctionStartPatterns
HELP
Images
Import
Instructions
iOS
Iteration
Languages
Mac OS X
Memory
MultiUser
PCode
Processor
Program
In Tool
Status
Name
Description
Key
Category
Modified
GraphAST.java
Decompile the function at the cursor, then b...
PCode
09/23/2...
GraphASTAndFlow.java
Decompile the function at the cursor, then b...
PCode
09/23/2...
GraphSelectedAST.java
Decompile the function at the cursor, then b...
PCode
09/23/2...
pcode_inspector.py
Prints information about pcode. Click on a fu...
PCode
10/14/2...
pcode_interpreter.py
Emulates pcode execution. Click on a functio...
Ctrl-QUOTE
PCode
10/14/2...
Assign Key Binding
Delete
Edit with Eclipse
Edit with basic editor
Ghidra API Help
New
Refresh
Rename
Run
Script Directories
Copy
Export
Filter:
Filter:
pcode_interpreter.py
Emulates pcode execution. Click on a function in the decompiler window, provide the initial parameters, and go!
Author: Karl Sickendick kc0bfv@gmail.com
Category: PCode
Key Binding:
Menu Path:

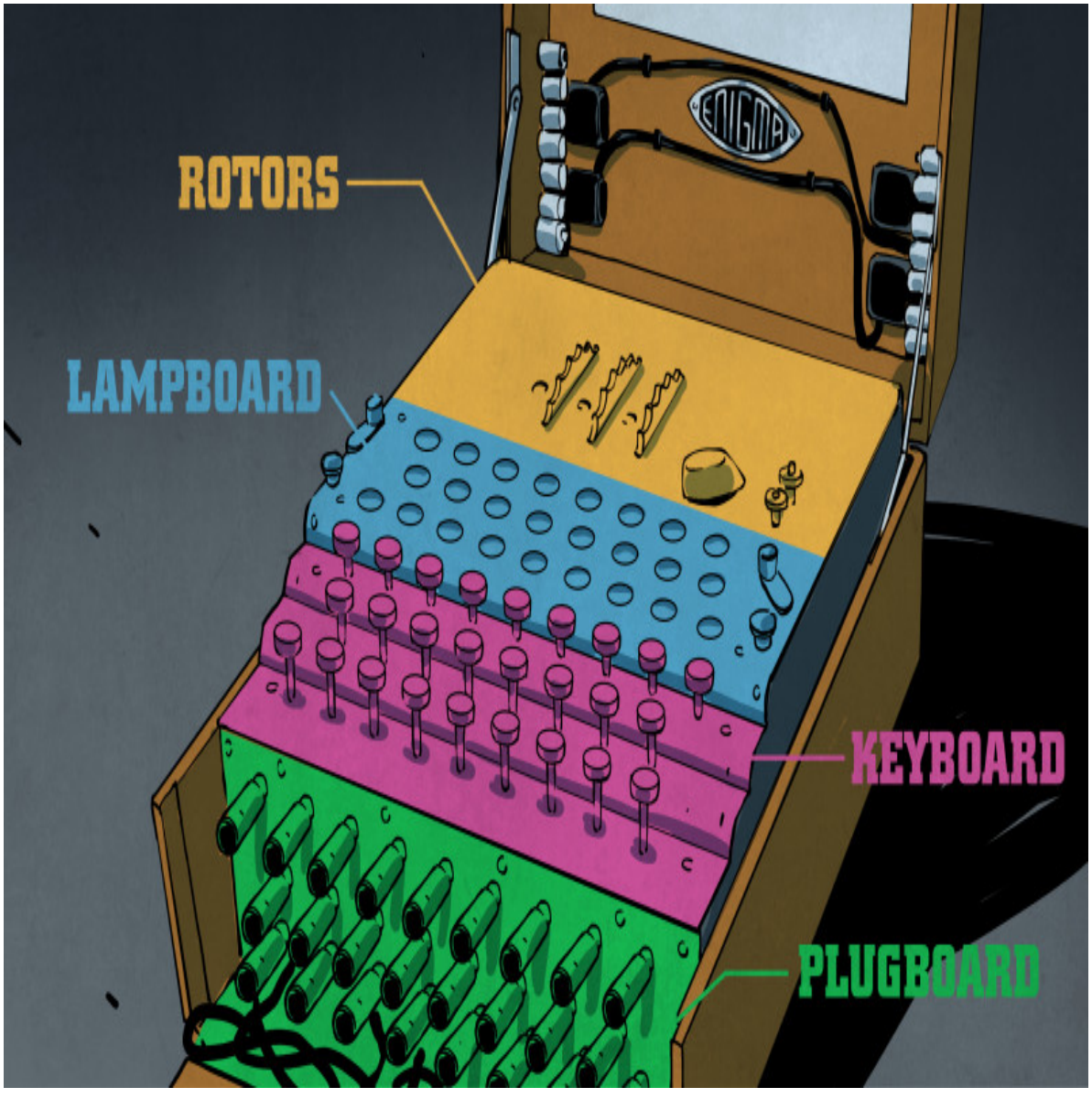
Section 2

PCode Emulation with Python

PCode Emulation with Python - Use Cases

- General Benefits
  - Middle ground between static & dynamic analysis
  - Simplifies analysis requiring hardware breakpoints
  - Customize environment for dynamic-ish analysis
- Obvious Use Cases
  - Examining malicious code behavior
  - Reversing crypto algorithms
  - Reversing code from an unusual architecture

An Enigma



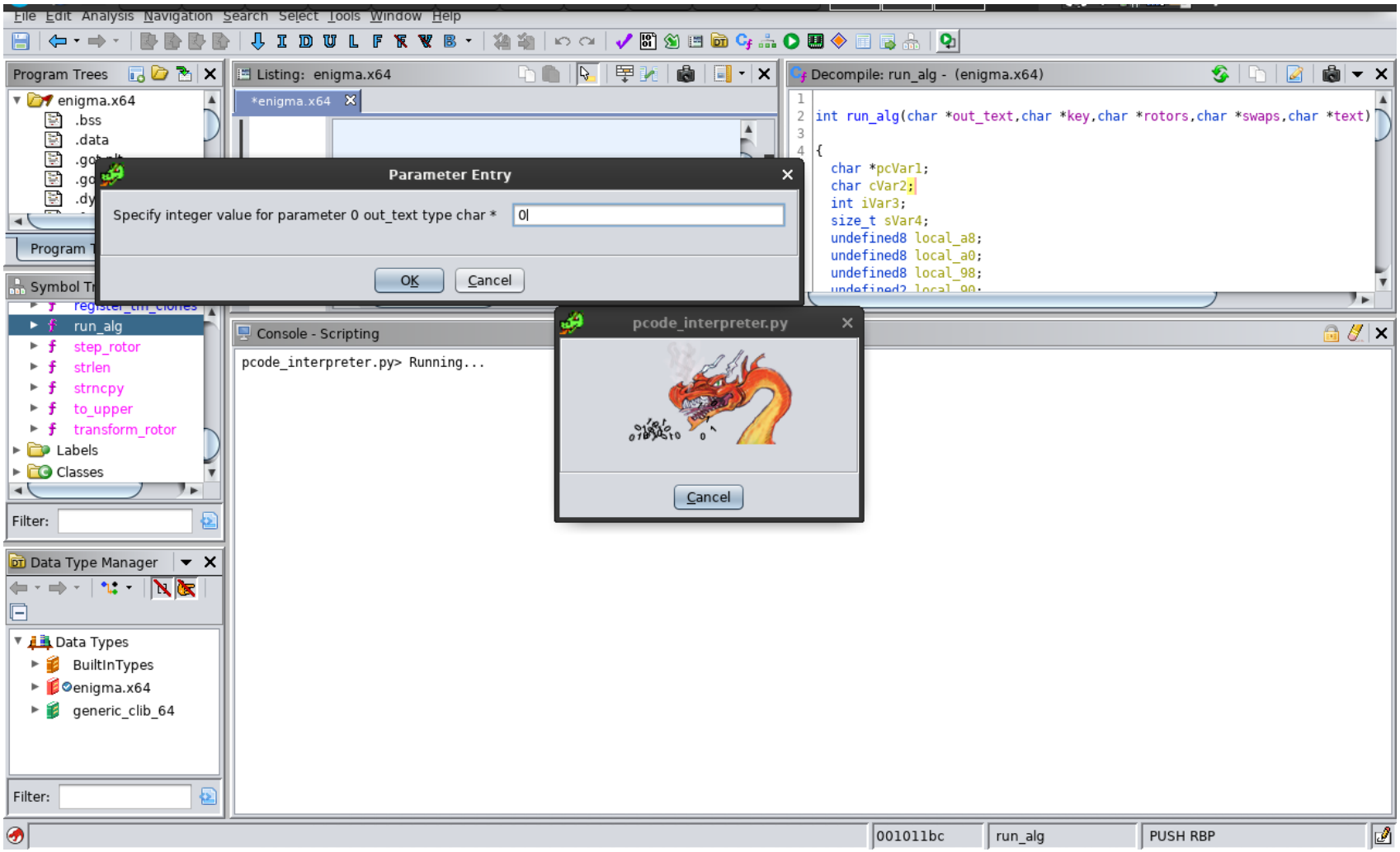
PCode Emulation - Initial RAM Environment Setup

- Output @ 0x0: "\x00\x00\x00\x00\x00\x00\x00\x00"
- Key / initial rotor states @ 0x8: "AAA"
- Rotors selected @ 0x10: "123"
- Swaps @ 0x18 : 0x00
- Input text @ 0x20: "AAAAAAA\x00"

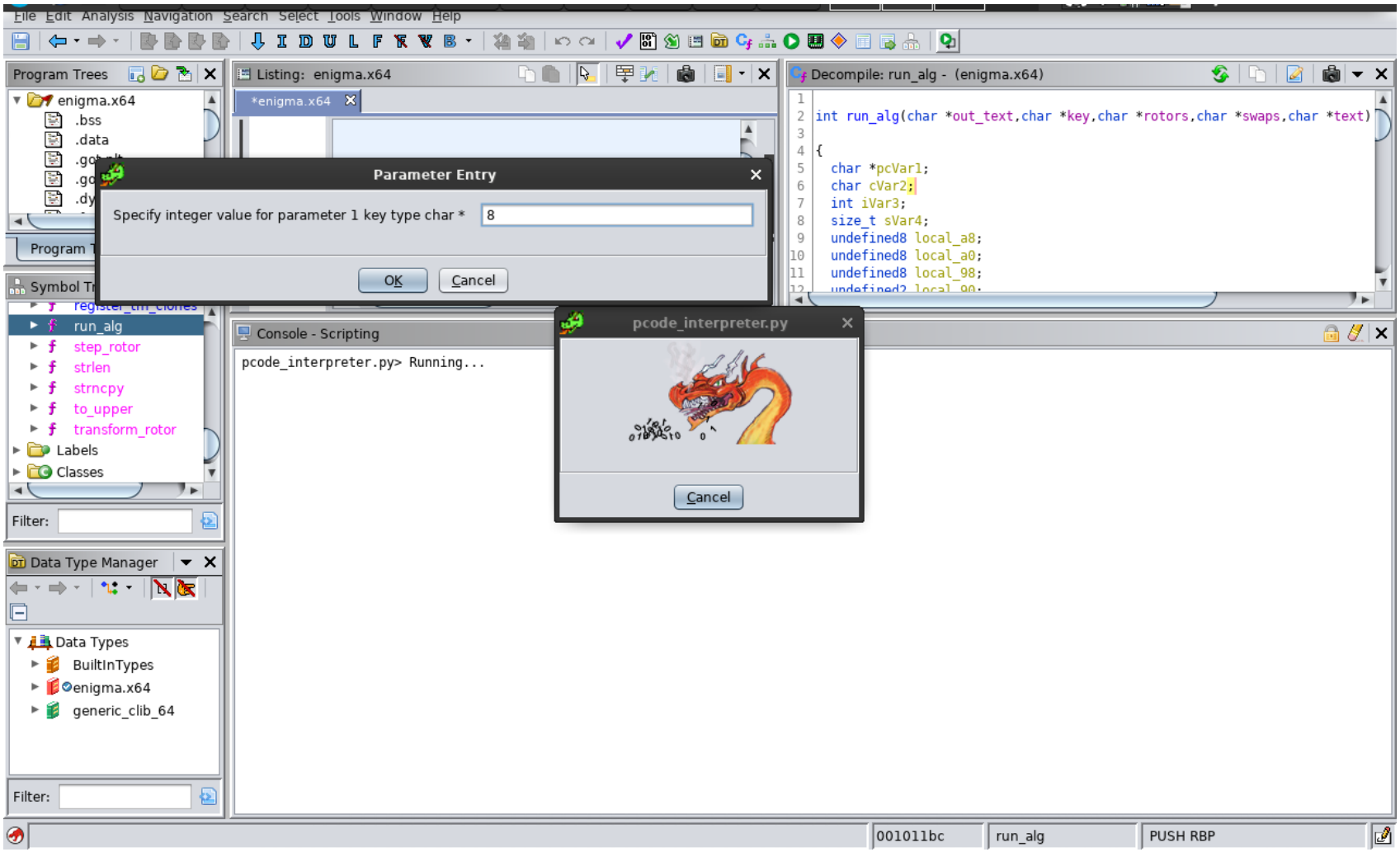
```
00000000:  0000 0000 0000 0000 4141 4100 0000 0000
00000010:  3132 3300 0000 0000 0000 0000 0000 0000
00000020:  4141 4141 4141 4100 0000 0000 0000 0000
```

```
state.ram.store(0x08, 8, 0x0000000000414141)
state.ram.store(0x10, 8, 0x0000000000333231)
state.ram.store(0x18, 8, 0x0000000000000000)
state.ram.store(0x20, 8, 0x0041414141414141)
```

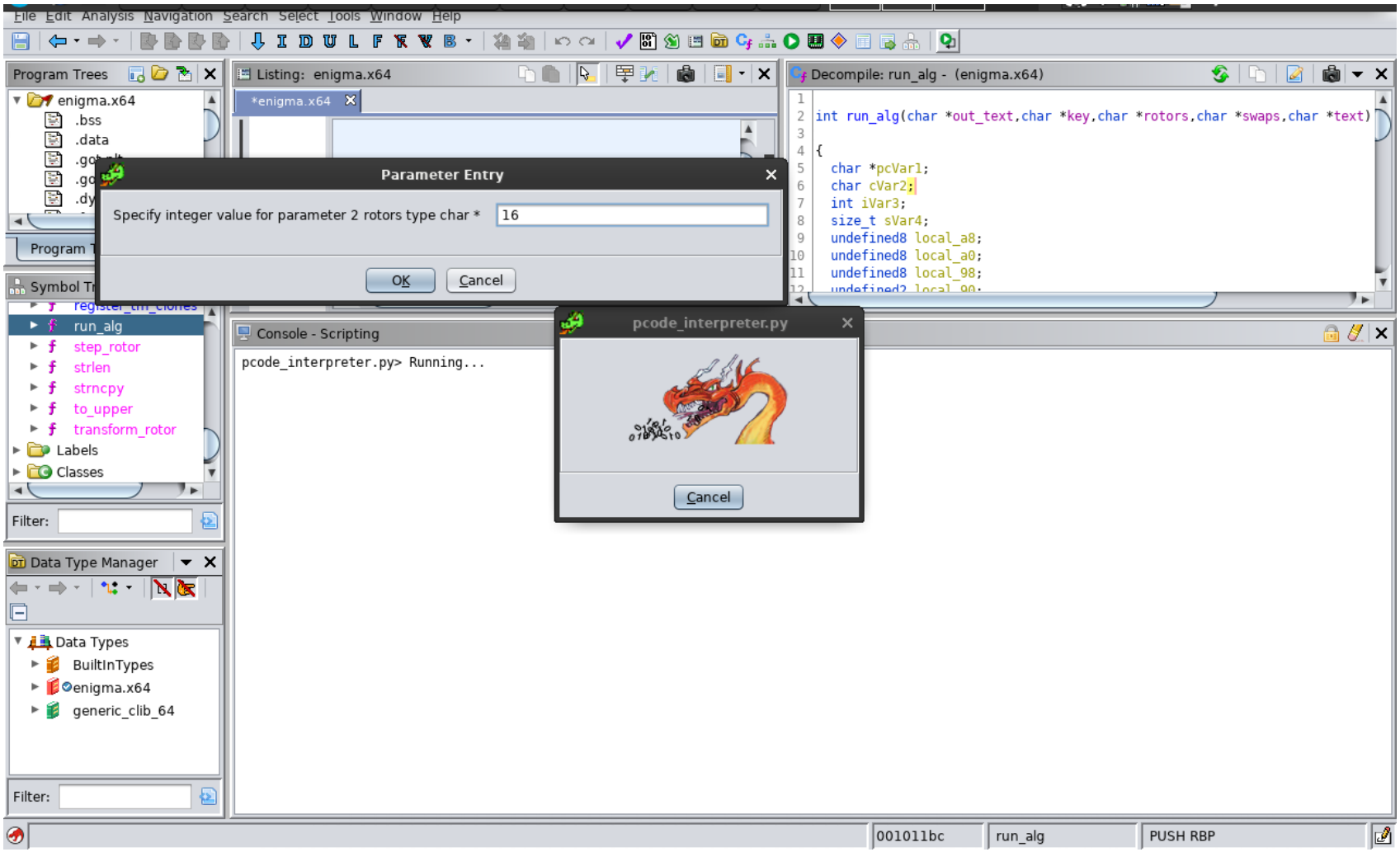
PCode Emulation - Parameter 0



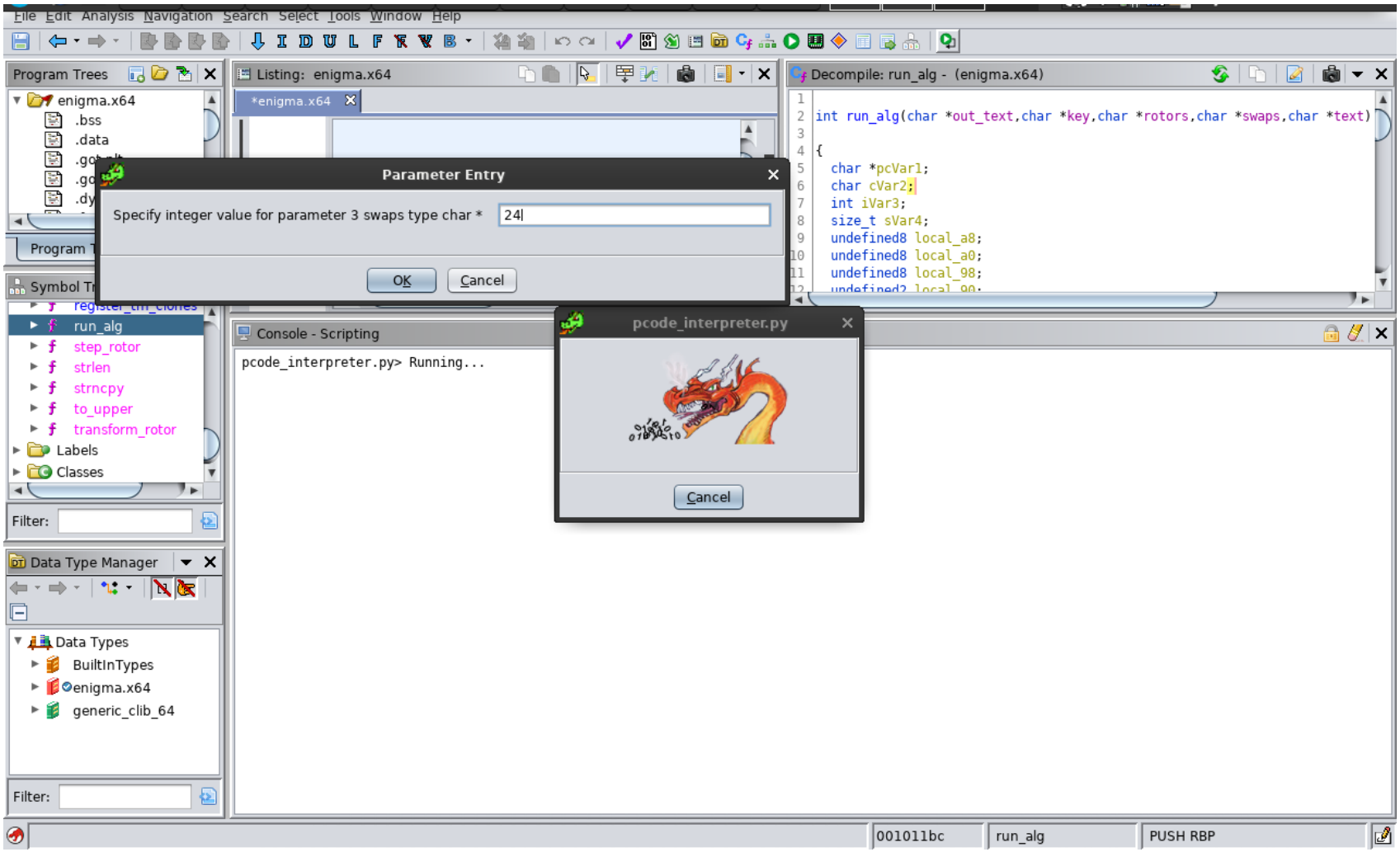
PCode Emulation - Parameter 1



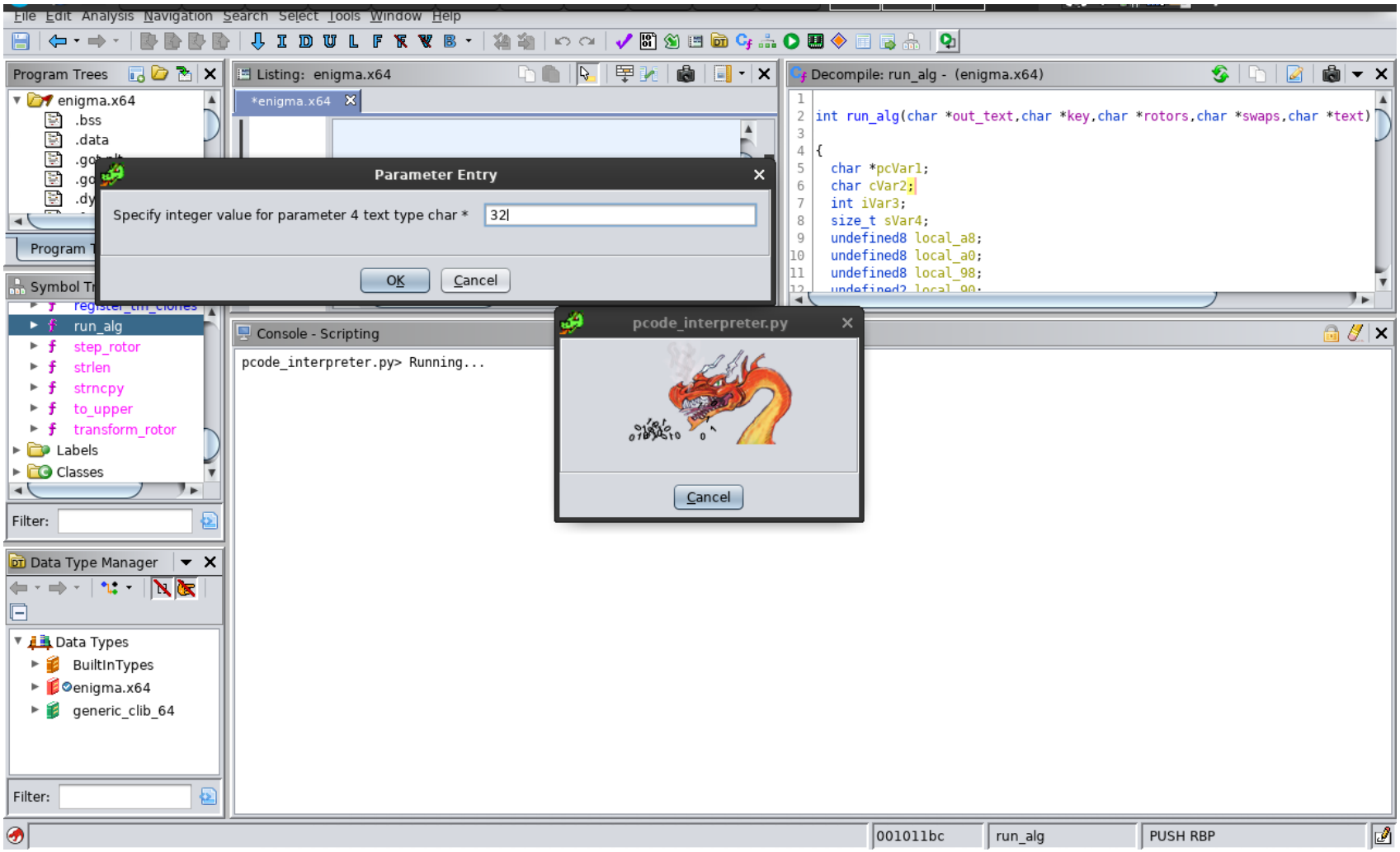
PCode Emulation - Parameter 2



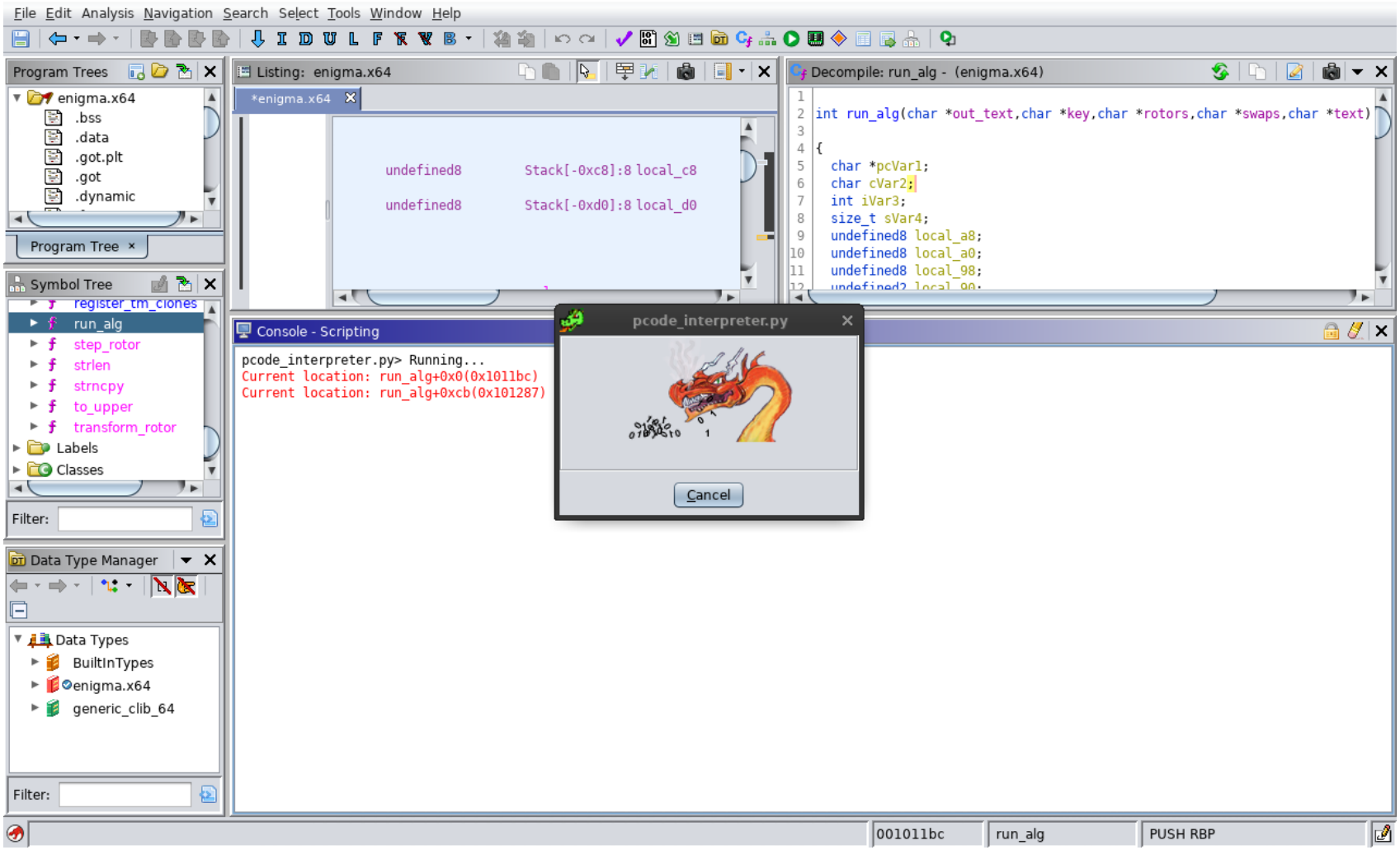
PCode Emulation - Parameter 3



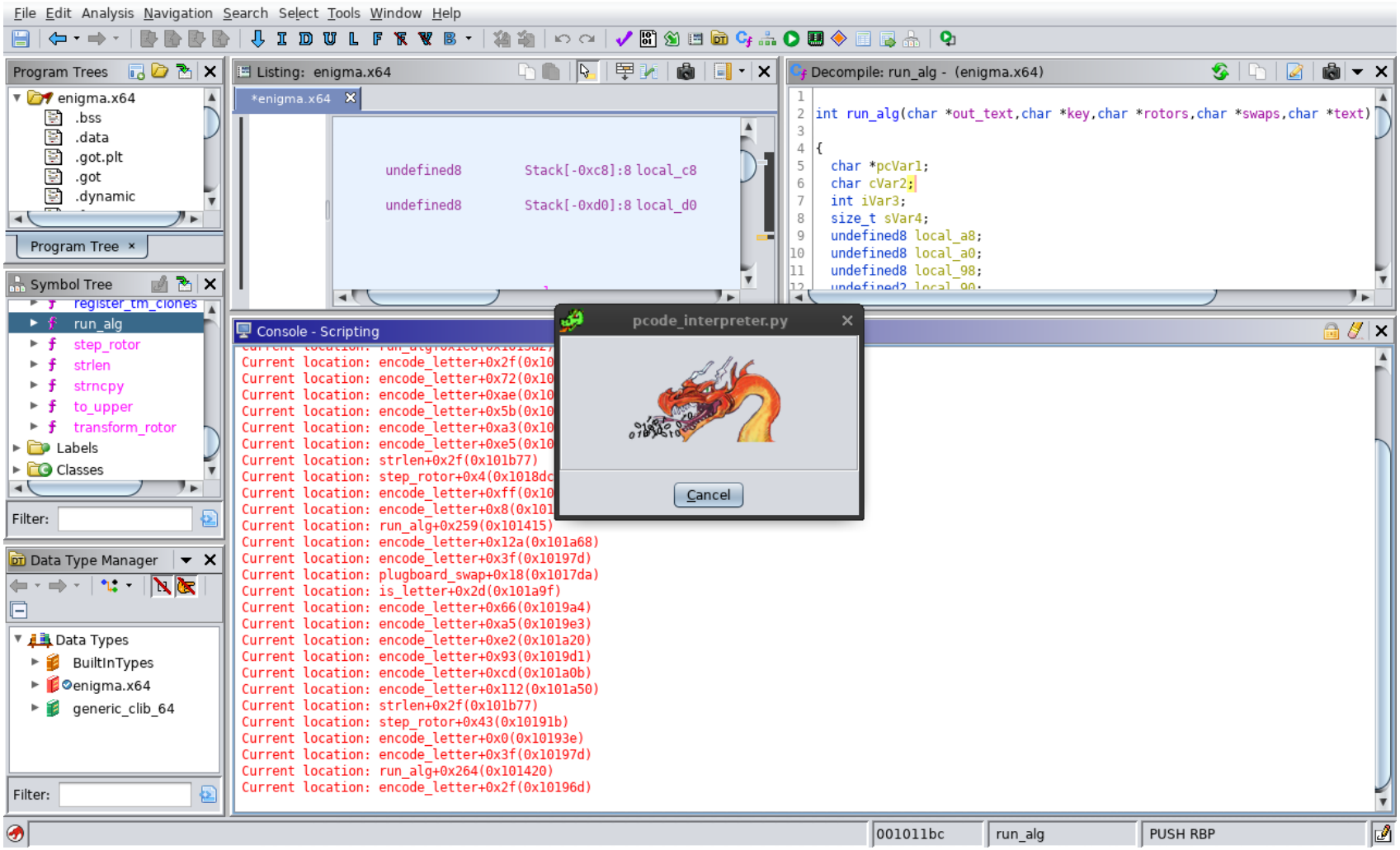
PCode Emulation - Parameter 4



PCode Emulation - Running...

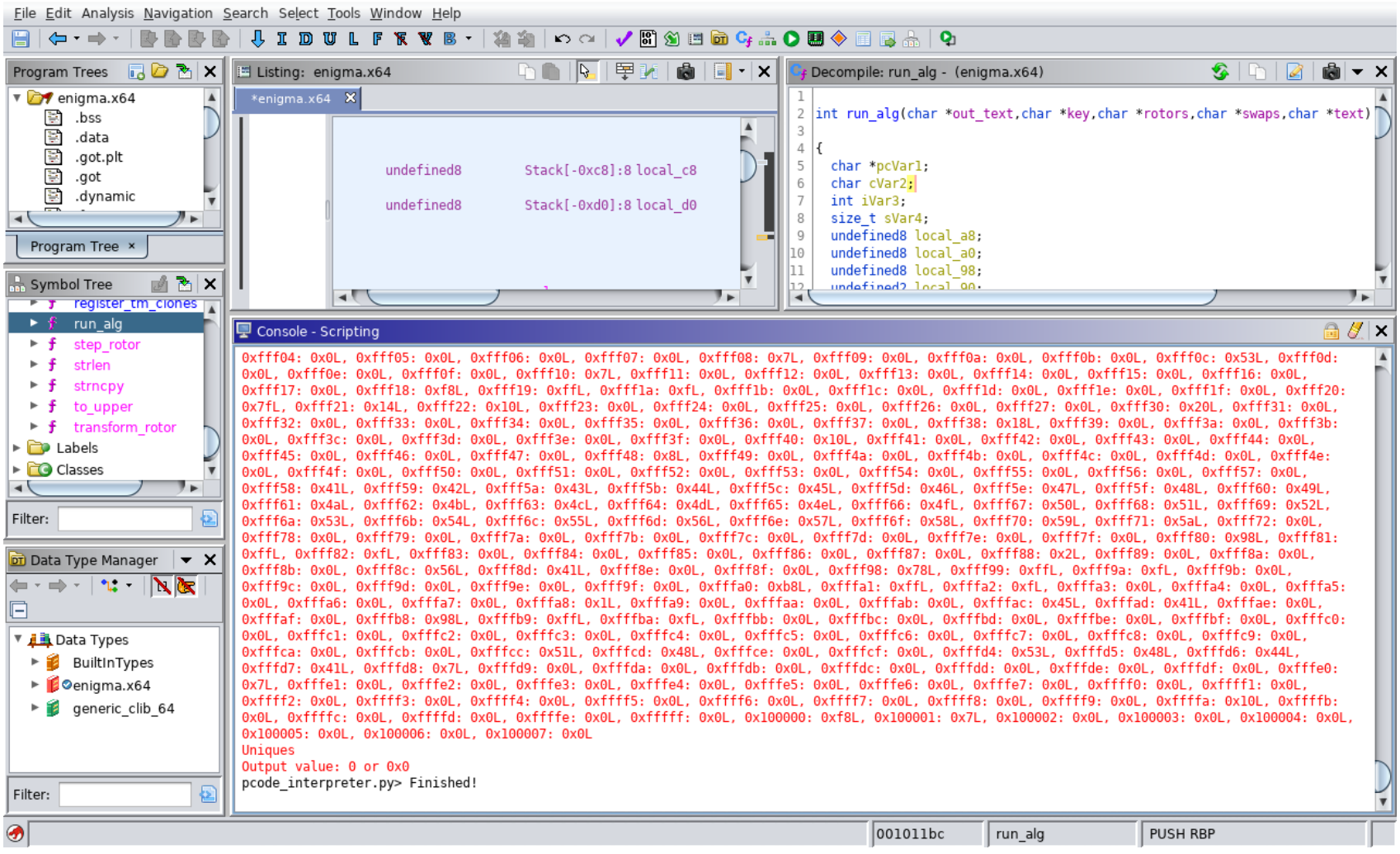


PCode Emulation - Running...

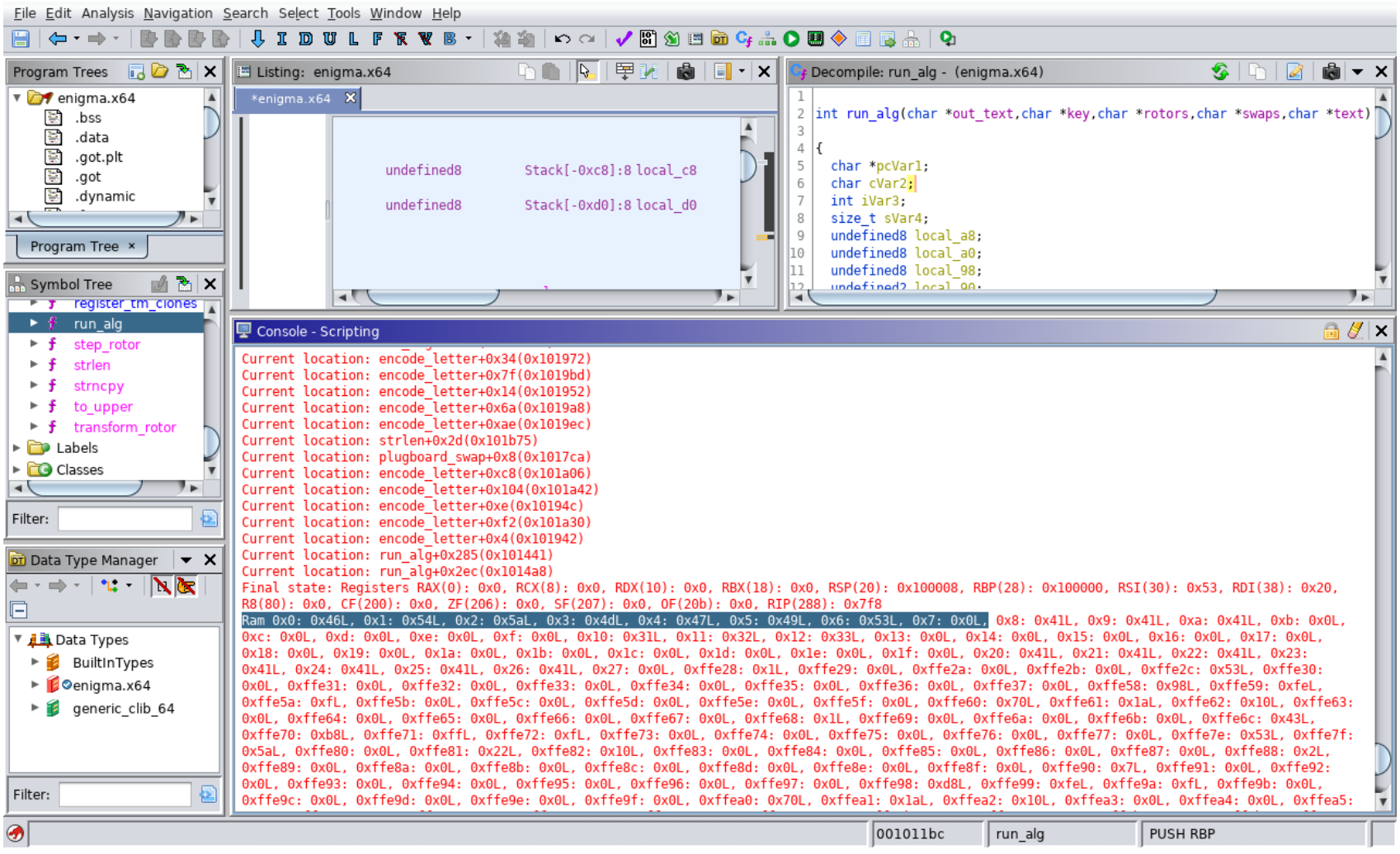


About 2.25 minutes later...

PCode Emulation - Complete!



PCode Emulation - Complete!



PCode Emulation - Checking Work

- Emulation output:
  - `'\x46\x54\x5a\x4d\x47\x49\x53' == 'FTZMGIS'`

PCode Emulation - Checking Work

- Emulation output:
  - `'\x46\x54\x5a\x4d\x47\x49\x53' == 'FTZMGIS'`
- Actual code output:
  - `'./enigma AAA 123 "" AAAAAAA' == 'FTZMGIS'`

PCode Emulation with Python - Shortcomings & Next Steps

- Shortcomings
  - Interface is rough
  - Problems with headless mode
  - Syscall and library handling...
- Improvements
  - Improve those shortcomings
  - More analysis methods than 'execute a function'
  - Breakpoints!
  - Simplify environment input and output

Section 3

Ghidra Scripts via Python

Ghidra Scripts

- Why use Python?
  - The alternative is Java
  - The standard library is massive
  - Tons of open source projects to import (Angr?)
  - Simplified prototyping (The console?)
- How does Ghidra employ Python?
  - Jython - Java Python
  - Python 2.7.1

Minimal Python Script Example

```
#Documentation here
#@author Author name and email
#@category PCode
#@keybinding
#@menupath
#@toolbar


from __future__ import print_function

import logging

logger = logging.getLogger(__name__)
logging.basicConfig(level=logging.INFO)


def func_here():
    logger.info("Hello World!")

if __name__ == "__main__":
    func_here()
```

Help Documentation

Help Documentation

Modifier and Type	Method	Description
void	**analyzeAll**(**Program** program)	Starts auto-analysis on the specified program and performs complete analysis of the entire program.
void	**analyzeChanges**(**Program** program)	Starts auto-analysis if not started and waits for pending analysis to complete.
Address	**askAddress**(java.lang.String title, java.lang.String message)	Returns an Address, using the String parameters for guidance.
byte[]	**askBytes**(java.lang.String title, java.lang.String message)	Returns a byte array, using the String parameters for guidance.
<T> T	**askChoice**(java.lang.String title, java.lang.String message, java.util.List<T> choices, T defaultValue)	Returns an object that represents one of the choices in the given list.
<T> java.util.List<T>	**askChoices**(java.lang.String title, java.lang.String message, java.util.List<T> choices)	Returns an array of Objects representing one or more choices from the given list.
<T> java.util.List<T>	**askChoices**(java.lang.String title, java.lang.String message, java.util.List<T> choices, java.util.List<java.lang.String> choiceLabels)	Returns an array of Objects representing one or more choices from the given list.
java.io.File	**askDirectory**(java.lang.String title, java.lang.String approveButtonText)	Returns a directory File object, using the String parameters for guidance.
DomainFile	**askDomainFile**(java.lang.String title)	Returns a DomainFile, using the title parameter for guidance.
double	**askDouble**(java.lang.String title, java.lang.String message)	Returns a double, using the String parameters for guidance.

Useful API

- Request input from the user
  - Normal data types
    - askBytes(...) -> bytes
    - askInt(...) -> int
    - askString(...) -> string
  - Ghidra-specific types
    - askAddress(...) -> GenericAddress
    - askLanguage(...) -> LanguageCompilerSpecPair
    - askProgram(...) -> ProgramDB
- Build Ghidra-specific types
  - toAddr(int) -> GenericAddress
  - parseAddress(...) -> GenericAddress
  - parseLanguageCompileSpecPair(...) -> ...
  - parseProjectFolder(...) -> ...

Useful API

- Introspect the open program
  - currentProgram -> ProgramDB
  - currentAddress -> GenericAddress
  - currentProgram.getLanguage() -> SleighLanguage
  - getFirstFunction() & getLastFunction() -> FunctionDB
  - getFunctionContaining(Address) -> FunctionDB
  - getDataContaining(Address) -> ...
  - getInstructionContaining(Address) -> ...
  - getPreComment(Address) -> string
  - setCurentLocation(...)

Useful API

- currentProgram.getLanguage()
  - .getLanguageDescription() -> LanguageDescription
  - .getProgramCounter() -> Register (includes name like ‘RIP’)
  - .isBigEndian() -> bool
- currentProgram.getLanguage().getLanguageDescription()
  - .getProcessor() -> Processor (includes name like ‘x86’)
  - .getSize() -> int

Useful API

- currentProgram.getCompilerSpec()
  - .getStackPointer() -> Register
  - .stackGrowsNegative() -> bool
- currentProgram.getLanguage().getProgramCounter()
  - .getOffset() -> int
  - .getMinimumByteSize() -> int
  - .getBitLength() -> int

Section 4

Resources

Ghidra Resources

- Official Ghidra Page: `https://ghidra-sre.org/`
- Great walkthrough: `https://github.com/0xAlexei/INFILTRATE2019`
- PCode and Sleigh reference documentation
  - ${GHIDRA_INSTALL}/docs/languages/html/{pcoderef,sleigh}.html
  - Also found at - `https://ghidra.re/courses/languages/html/`
- Ghidra Tutorials
  - ${GHIDRA_INSTALL}/GhidraClass
  - Also found at - `https://ghidra.re/courses/GhidraClass/`

Questions?