*ObjAsm **Tools***

OA_Tools

Version 2.0

1. Introduction

ObjAsm Tools (OA_Tools) is collection of tools to verify your source code, which must be in plain ANSI text format.

ObjAsm methods, regular **MASM®** procedures and macros can be scanned to detect possible code problems like unpreserved registers or unnecessary register preservation and unused argument or local values.

Unpreserved registers (see Microsoft Windows ABI) often lead to unexpected crashes of your application and unused argument or local variables wastes the stack space.

The results of the tests are displayed in separate child windows of the main application for each analysed file.

To avoid unexpected results, the scanned code should be well formed. That means, that it should compile before this tool is used. Only **Intel®** syntax is supported.

The usage of annotations decreases “false positive” matches. They should be used whenever possible.

To extend the functionality of the application, you can add your favourite helper tools by simply adding them to the setup “tool” panel.

2. Contents

3. EULA

END-USER SOFTWARE LICENSE AGREEMENT FOR OBJASM

I**MPORTANT! READ CAREFULLY**: This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and me for the **ObjAsm** software product accompanying this EULA, which includes computer software and may include "online" or electronic documentation, associated media, and printed materials ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the SOFTWARE PRODUCT.

GRANT OF LICENSE

1. Subjected to the terms and provisions of this Agreement, I grant to you the non-exclusive, limited and royalty-free right to use this Software. The term "Software" includes all elements of the Software such as data files and screen displays. You are not receiving any ownership or proprietary right, title or interest in or to the Software or the copyright, trademarks, or other rights related thereto.

2. You are completely free to distribute your own developed source, object or executable code for commercial purposes, but NOT the source code of the SOFTWARE PRODUCT.

LIMITED WARRANTY

NO WARRANTIES. I expressly disclaim any warranty for the Software Product. THE SOFTWARE PRODUCT AND ANY RELATED DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT REMAINS WITH YOU.

NO LIABILITY FOR DAMAGES. In no event shall I be liable for any damages whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if it has been advised of the possibility of such damages.

4. Acknowledgements

I'd like to express my very great appreciation to all authors mentioned in the source code, whose valuable and constructive contributions made this work possible.

Thank you!

Corrections, comments, suggestions, contributions, etc. may be sent to the MASM32 Forum, or directly mailed to:

ObjAsm@gmx.net

G. Friedrich,

March, 2019

5. Usage

The application is designed to perform the following verifications on the source code:

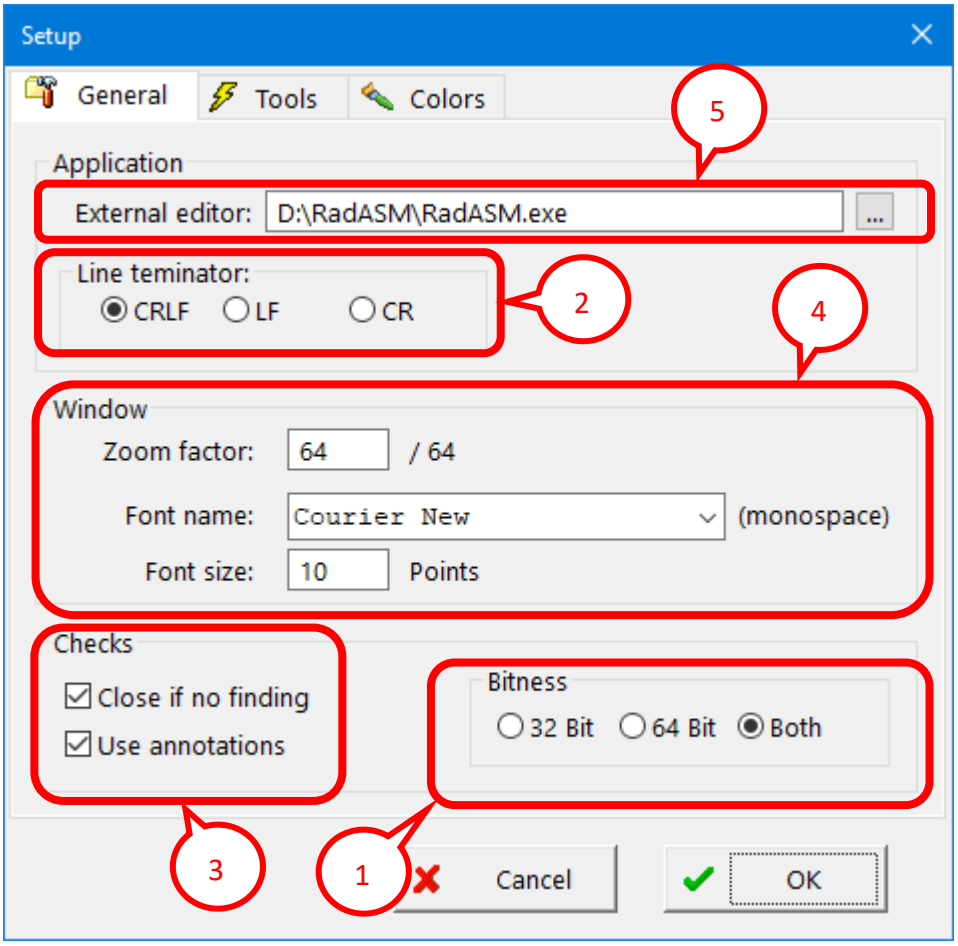
1. Register usage and preservation verification.
   It follows the Microsoft Windows® ABI rules for 32 bit and 64 bit. Preserved registers are identified at the beginning of the code block after the “uses” keyword.

2. Local variables usage verification.
   Unused local variables are detected to reduce stack usage.
   Local variables are those following the ”local” keyword.

3. Arguments usage verification.
   Unused arguments are detected to reduce stack usage and improve speed.

Setup

The setup dialog has three tabs. The first one, called “General” is to set administrative parameters of the application. It looks like:



Bitness

The verification algorithms are designed to operate on 32 bit, 64 bit and mixed source code. This setting is done on the setup dialog (Field 1).

Selecting “32 Bit”, only those register with this wordsize are analysed. Likewise, when “64 Bit” is selected. Choosing “Both”, the variable wordsize registers (i.e. xax, xcx, etc.) are included in the analysis.

Line terminator

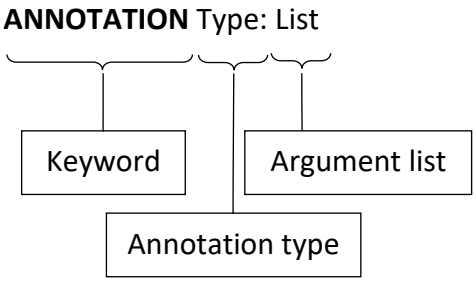
The line terminator can vary depending on the editor used (Field 2). Windows typically used “CRLF” combination to mark a line end. On Linux distributions, “LF” is used, while Apple OSes use “CR” only.

Close if no finding

This setting allows closing analysis windows that shows no findings. This reduces notably an overcrowded output (Field 3).

Use annotations

Source code annotations are useful to declare registers or variables that are used in a macro and not visible in the code block of a method or procedure. This helps to avoid false positives. The syntax of an annotation looks like:



Examples:

ANNOTATION prv: xdi xsi

prv: coder preserves the following list of registers e.g. using a sequence of push and pop or storing the non-volatile registers in local variables.

ANNOTATION use: foo

use: the following list of arguments are used but hidden behind e.g. macros or equates. They can be register or variable names.

Note: ANNOTATION is a macro that accepts any parameter and does nothing.

Output window settings

Some formatting of the output windows can be performed on field 4. Initial font settings like Face and Size can be adjusted. Likewise, the initial zooming factor can be set.

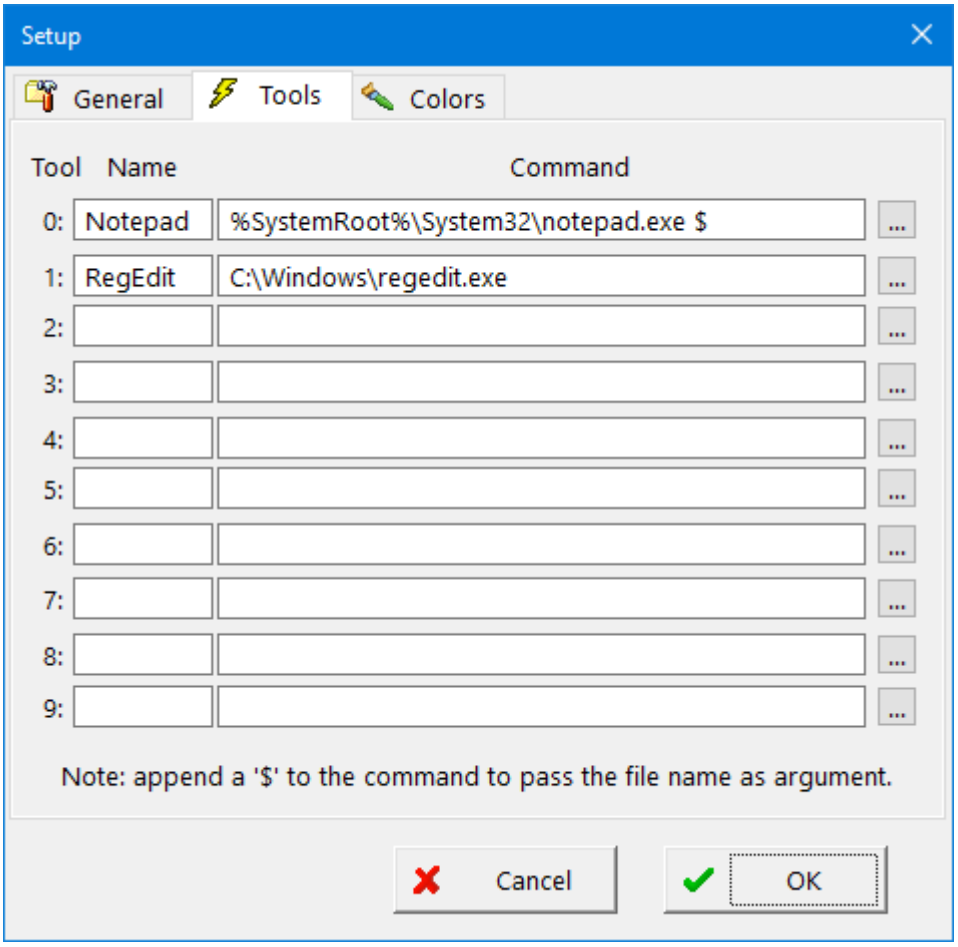
External editor

Once a finding is detected, the source code can be quickly reached using the “external editor”. The setting is done on field 5.

Tools

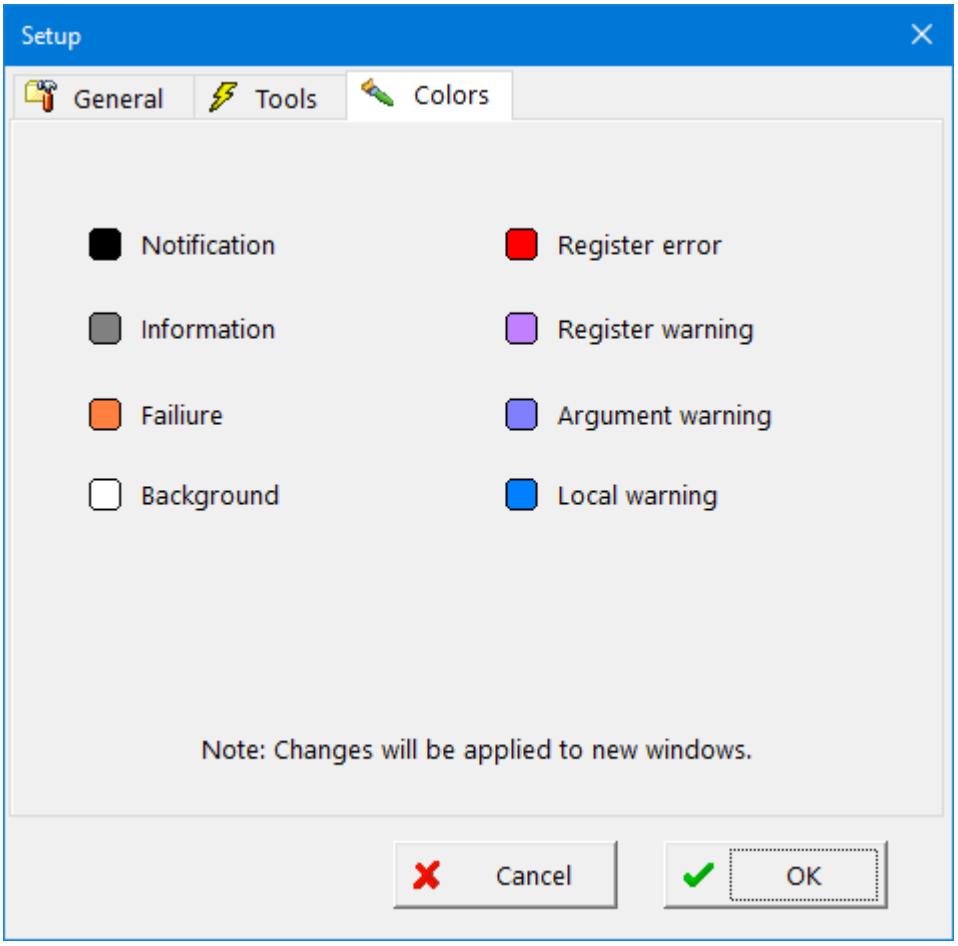
The second tab is called "Tools". Here are general tools of your choice configured. It's very handy to have your favourite tools always at hand. The use them, select one from the “Tools” main menu.

Note: to launch the tool with the file name of the selected child window as first parameter, place a $ Symbol after the command string, separated by a space.



Colors

The third tab is called "Colors". Output colors can be setup here. Default values need to be changed if high contrast display settings are enabled.



Analysis

Selecting the proper verification from the “Checks” menu will trigger the file selection dialog. Many files can be selected at once. For each of them, the verification will be done and the output will be displayed on the corresponding child window. If the option “Close if no finding” is activated and there is no finding, the child window will be closed automatically.

During the analysis, the application does not respond to any input. This situation is indicated by an orange menu bar. Once the colour is grey again, the application is ready to accept new commands.

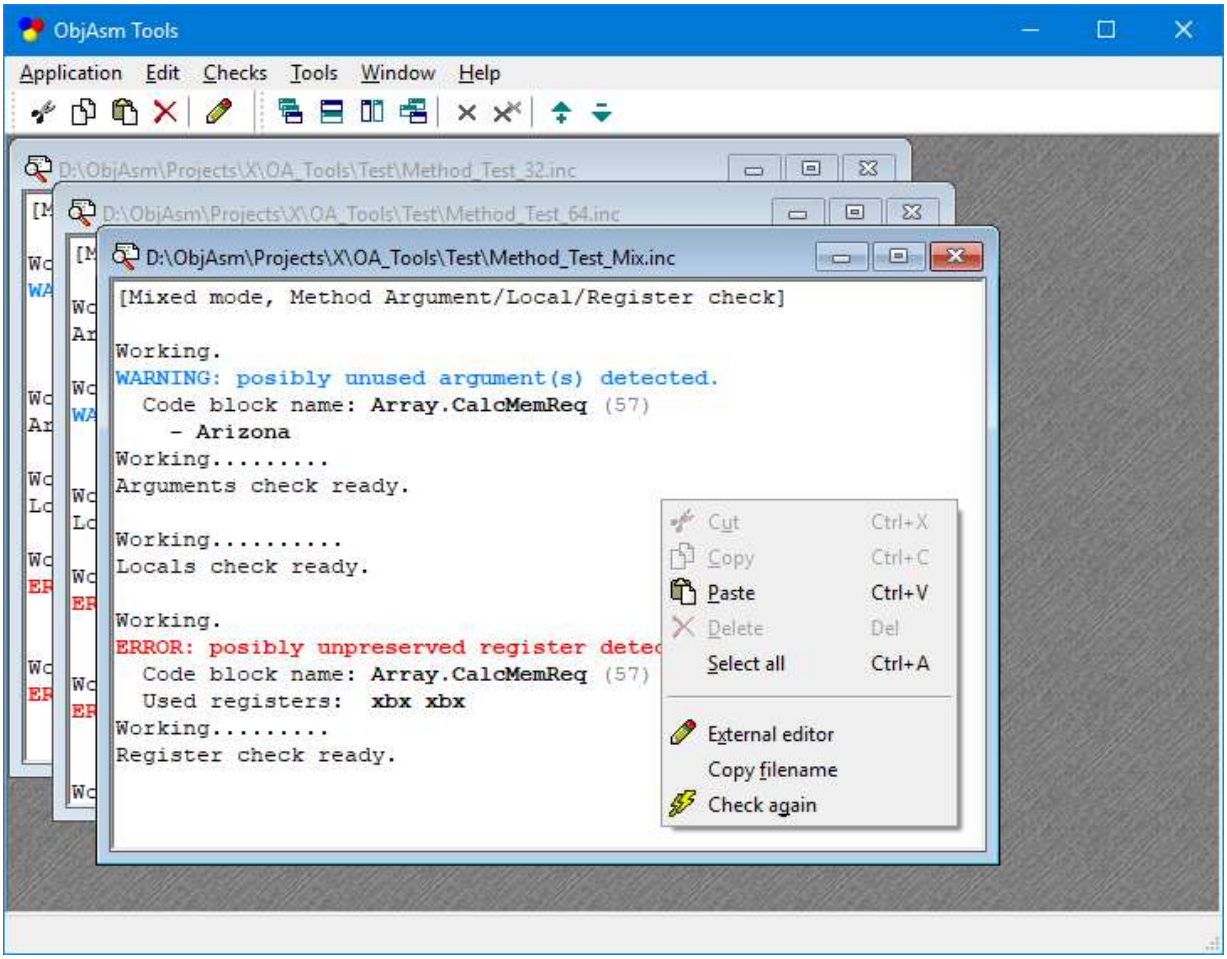
The source code files must be formatted as plain ANSI text. Comments introduced by an “;” character or by the COMMENT keyword are ignored by the analysis algorithms.

Each child window has a context menu. Using it, you will be able to perform additional actions like:

- Cut, Copy, Paste, delete and Select All
- Launch an external editor for the analysed file
- Copy the file name to the clipboard
- Reanalyse the file

In some situations, it is convenient to simply have the file name. If you select the "Copy File Name" menu item, it is copied to the clipboard.

If something has changed on the code, the file's previous checks will restart by the “Check again” menu item.


ObjAsm Tools
Application Edit Checks Tools Window Help
D:\ObjAsm\Projects\X\OA_Tools\Test\Method_Test_32.inc
D:\ObjAsm\Projects\X\OA_Tools\Test\Method_Test_64.inc
D:\ObjAsm\Projects\X\OA_Tools\Test\Method_Test_Mix.inc
[Mixed mode, Method Argument/Local/Register check]
Working.
WARNING: posibly unused argument(s) detected.
Code block name: Array.CalcMemReq (57)
- Arizona
Working.....
Arguments check ready.
Working.....
Locals check ready.
Working.
ERROR: posibly unpreserved register detec
Code block name: Array.CalcMemReq (57)
Used registers: xbx xbx
Working.....
Register check ready.
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Select all Ctrl+A
External editor
Copy filename
Check again

Accelerators

The application provides the following accelerators (keystroke combinations):

F1	Help
F2	Show the setup dialog
F5	Perform all method checks
Ctrl + F5	Perform all procedure checks
Alt + F5	Perform all macro checks
Shift + F4	Tile windows vertically
Shift + F5	Cascade windows
Ctrl + Shift + F6	Move to previous window
Ctrl + F6	Move to next window
Ctrl + F4	Close active window
Ctrl + Shift + F4	Close all windows
Ctrl + A	Select all
Ctrl + X	Cut selection
Ctrl + C	Copy selection
Ctrl + V	Insert
Ctrl + I	Zoom in
Ctrl + O	Zoom out
Del	Delete selection