Contents

[1. State of the art 2](#_Toc333393408)

[1.1. Introduction 2](#_Toc333393409)

[1.2. Translators 2](#_Toc333393410)

[1.2.1. Mandreel 2](#_Toc333393411)

[1.2.2. JSIL 3](#_Toc333393412)

[1.2.3. XMLVM 3](#_Toc333393413)

[1.2.4. Tangible software solutions 4](#_Toc333393414)

[1.2.5. TonicVM 5](#_Toc333393415)

[1.2.6. Sharpen 6](#_Toc333393416)

[1.2.7. Minor projects 6](#_Toc333393417)

[1.3. Decompilers and assembly browsers 7](#_Toc333393418)

[1.3.1. ILSpy 7](#_Toc333393419)

[1.3.2. Other Decompilers 8](#_Toc333393420)

[1.4. Memory management in C++ 9](#_Toc333393421)

[1.4.1. A Method for Automatic Optimization of Dynamic Memory Management in C++ 9](#_Toc333393422)

[1.4.2. Comparison of Garbage Collector Prototypes for C++ Applications 11](#_Toc333393423)

[1.5. Libraries and code infrastructures 12](#_Toc333393424)

[1.5.1. LLVM 12](#_Toc333393425)

[1.5.2. Mono.CECIL 12](#_Toc333393426)

[1.5.3. OpenC++ and OpenJava 12](#_Toc333393427)

[1.6. Research projects 13](#_Toc333393428)

[1.6.1. Alchemy 13](#_Toc333393429)

[1.6.2. Emscripten 13](#_Toc333393430)

[1.6.3. Jangaroo 14](#_Toc333393431)

[1.6.4. IKVM.NET 14](#_Toc333393432)

[1.6.5. Garbage collector projects 14](#_Toc333393433)

[1.7. Aporte 15](#_Toc333393434)

[2. Interesting links 16](#_Toc333393435)

[3. References 17](#_Toc333393436)

# State of the art

## Introduction

Nowadays the technology world is converging into a totally distributed system and into the world of Internet of Things. All kind of devices with all kind of firmwares and operating systems are interconnected creating a super-system able to do complex tasks.

For maintaining the interoperability between devices it is necessary to compile applications written in the specific language that accepts the processor. One option is to deploy Virtual Machines in all the devices benefiting of the portability of the code, but the performance it is not ideal. For this reason some code translators have taken into account for accom

After that, several assembly browsers and decompilers are explained which are able to extract source code from executables or assemblied files. Also is discussed different related projects and memory management solutions for optimizing the application.

## Translators

The translator is a tool for converting the code compiled for one system to the same application but compiled for other system. The translation can be done in the source code level (change text strings) or in other levels like byte code level (change instructions).

The translation provides the user with a tool which for developing the application in a comfortable language (high-level language) benefiting of all the high-level features and gaining time in the programming stage.

When the code is finished, the translator is able to port this application developed in a high-level language to a low-level language programmed application, benefitting of the speed and performance of these languages.

In this section the more relevant considered translators are explained.

### Mandreel

Mandreel Framework(see [1]) is a set of tools for converting OpenGL based applications (in C++ or Objective-C) to web applications. It is fully integrated with Microsoft Visual Studio and Apple XCode environments and it is optimized for JavaScript and Action Script 3 based applications or Adobe Flash / AIR based applications.

One of the most important features of this framework is that the input languages (C++ and Objective-C) are the languages of Android and iOS, thus, Mandreel framework provides tools for porting mobile based applications to web applications.

There are two Mandreel SDK packages:

* Mandreel SDK for JavaScript: Build C++ applications for JavaScript based platforms i.e. html5 applications. Allows 2D and 3D hardware accelerated applications and standard APIs like OpenGL and OpenAL.
* Mandreel SDK for Action Script 3: Build C++ applications for the Adboe Flash/AIR platforms. Provides the same features than the previous case: 2D/3D applications and OpenGL/OpenAL standard APIs.

Mandreel framework is ideal for gaming and there are different enterprises (i.e. GameLoft, glu) that are using this framework for providing applications over Google Chrome platform for the social networg Google Plus (based on html5).

### JSIL

JSIL (see [2]) is a compiler that transforms .NET based applications into JavaScript applications. JSIL transforms the CIL bytecode (native format of Microsoft .NET applications) into standards-compliant, cross-browser JavaScript.

Figure 1.1 JSIL process

This translator is ideal for games or graphical applications in web browsers. However this translator does not search performance (because of the JavaScript language) but is looking for compatibility in all the browsers.

### XMLVM

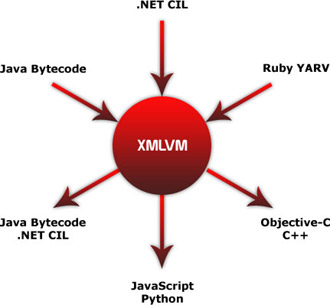


Figure 1.2 XMLVM

XMLVM (see [3]) aims to provide a flexible and extensible cross-compiler toolchain. The main feature of this project is that instead of cross-compiling the source, XMLVM cross-compiles at the byte-code instructions which is easier for transform than the source code with high-level languages.

XMLVM supports Sun Microsystems virtual machine (Java) and Microsoft Common Language Runtime (.NET). As XMLVM works with byte codes, it supports several languages based on the same virtual machine.

The process consists on transforming the byte code in an XML representation, then this XML is transformed into another XML (with the target language format) and then return the source code in the target language.

Figure 1.3 XMLVM process

The main advantage of this system is the architecture. With this architecture, it can be added differnt input or output languages only change the 3 first boxes for the input language or the two last boxes for the output language (and compute the transformation from the input XML to the output XML with XSLT method).

### Tangible software solutions

Tangible software solutions (see [4]) defines itself as a software development and consulting firm specializing in source code conversion tools and source code conversion projects.

The conversion utilities they offer are:

* Instant C# (VB to C# converter)
* Instant VB (C# to VB Converter)
* C++ to C# Converter
* C++ to VB Converter
* C++ to Java Converter
* C# to Java Converter
* VB to Java Converter
* Java to C# Converter
* Java to VB Converter
* C# to C++ Converter
* VB to C++ Converter
* Java to C++ Converter

Tangible software solutions offer several translations. The translations are good but they do not support some features (i.e. namespaces not supported when translating from C# to C++). Another disadvantage is that is not a freeware solution.

### TonicVM

TonicVM (To .NET Intermediate Code Virtual Machine) (see [5]) is an experimental software tool from InnoveWare Solutions.

TonicVM is a source-to-source compiler and a Virtual Machine. In the first stage is used a front-end ( compiler) which is made of a lexer and a parser and extracts an Abstract Syntax Tree representation of the source code. In some languages this front-end also contains a preprocessor (i.e. C/C++).

The syntax tree is translated to common intermediate code using a program called back-end. The back-end is made of a code optimizer (run transformations in the syntax tree in order to be target language compliant) and a code generator (exports the transformed syntax tree to source code).

Finally the target source code is executed over a Virtual Machine.

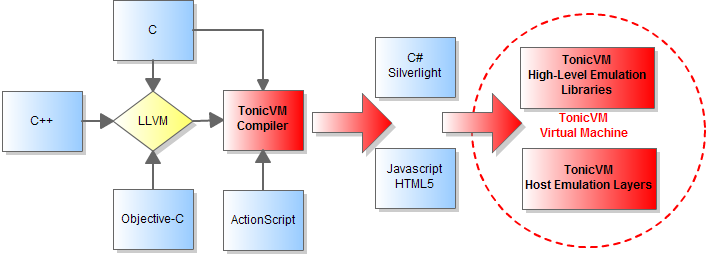


Figure 1.4 TonicVM scheme

The general scheme is shown in Figure 1.4. However the overall scheme in a simpler way is described below in Figure 1.5 represented as different processes of the program.

Figure 1.5 TonnicVM general scheme

The architecture followed by TonicVM is similar to the architecture this work will follow. It is important to notice the use of ASTs (Abstract Syntax Tree) for representing the code, optimize the code and generate the code. The box "Code Optimizer" applies transformations to the tree generated in the box "AST" and the box "Code generator" converts the tree in plain text to be compiled.

### Sharpen

Sharpen (see [6]) is an Eclipse plug-in for multi-platform development from one codebase. Sharpen currently compiles Java to C#, and the difference between the other projects are:

* Native support for db4o: a open-source object-oriented database that enables Java and .NET developers to store and retrieve any application object in a simply way.
* .NET naming conventions: Only for convention
* Customization: namespaces, methods and properties...

CONCL

### Minor projects

#### Cxx2cs

Direct C++ to C# translator (see [7]). This project has academic aim only and offers the translation from C++ to C# without any compilation progress.

#### Code Translator (online)

This project is an online service for translating code. From now it only supports from VB.NET to C# and from C# to VB.NET (see [8]).

#### Source Code Translation from excelsoftware

ExcelSoftware proposes a tool and a method to improve the code translation work (see [9]). This translation method requires human editing, but eliminates much of the work and avoid a lot of possible errors.

The process is divided in three stages:

* Create design of old code: A graphic representation of the actual code depending on the language. The diagram (or diagrams) shows the static structure of the program.
* Modify design for new code: Once the class diagram (design) is done (and also a dictionary which relates the class with its properties and references), using a tool provided by ExcelSoftware, the design is changed in order to fit the target language rules. Can be made changes in names, references, properties etc.
* Translate function logic: This tool provides an option to simplify the language translation process in specific languages like Java and Objective-C.

Figure 1.6 ExcelSoftware translator

There are more minor projects but these three are representative enough of all the projects and works in progress in the community.

## Decompilers and assembly browsers

For this work it is important to study different tools for extract code from applications or assemblies. This feature allows extracting information of the code and program structure from a file which is previously compiled.

In this section will be studied different decompilers and assembly browsers. All of them have approximately the same features, the only difference is the languages supported (input and output languages).

### ILSpy

ILSpy (see [10]) is a .NET assembly browser and decompiler. It is open-source and its development started after the announcement that the free version of NET Reflector would cease to exist by end of February 2011.

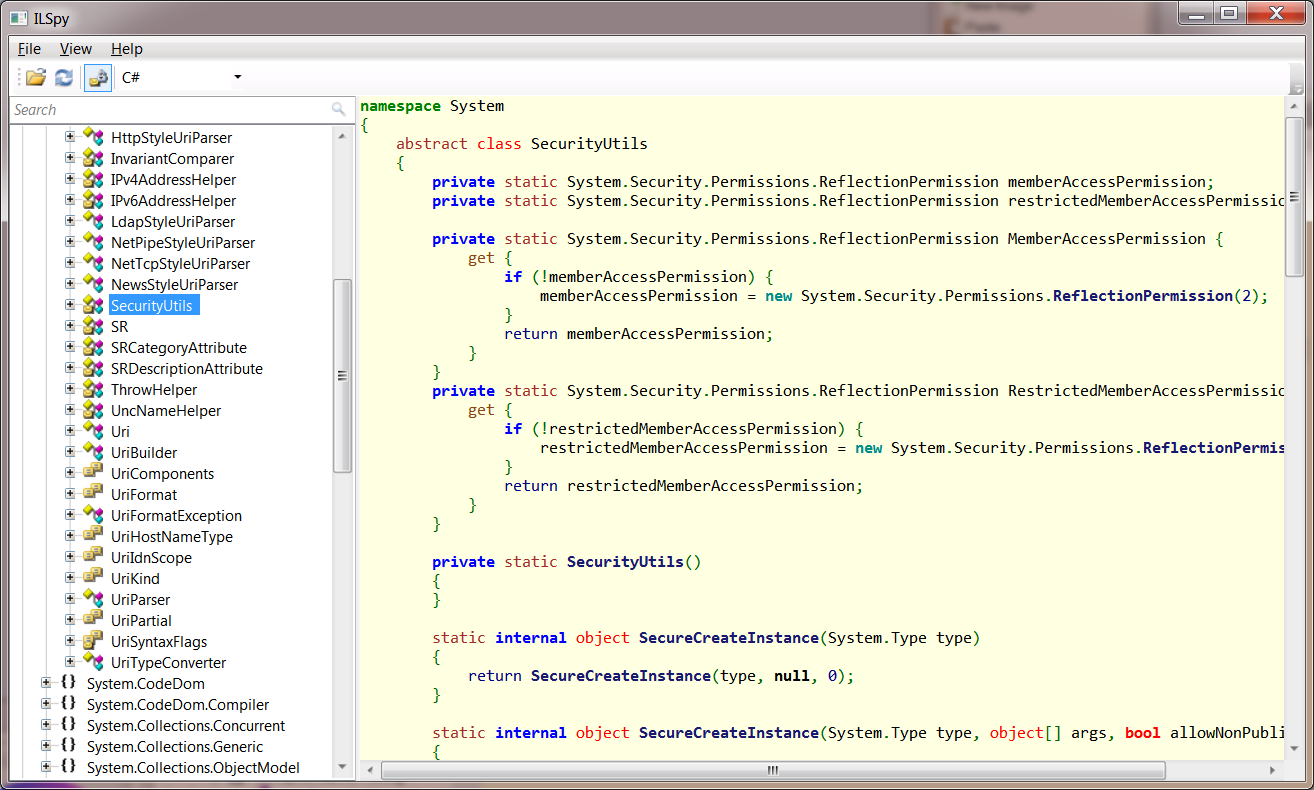


Figure 1.7 ILSpy

ILSPy loads the assembly with the Mono.CECIL library (explained after in this chapter). This library inspects programs and libraries in CIL format. With the CECIL library the ILSPy extracts all the types definitions, methods, fields etc. Through an AST Builder, the program generates an Abstract Syntax Tree representing the source code in an abstract way. The final stage is to convert this Abstract Syntax Tree in text (source code). This program defines different templates, each one for each language. The supported languages are:

* IL
* C#
* VB

Figure 1.8 ILSPy process

CONC

### Other Decompilers

There are several different .NET Decompilers with almost the same features of ILSPy. Also almost all the .NET Decompilers have the same look and feel. The most important are .NET Reflector [11] and Spices .NET Decompiler [12].

Also there are decompilers for other languages or frameworks like DJ Java Decompiler [15] for Java language, VB Decompiler [16] for Visual Basic language and Sothink SWF Decompiler [17] for Flash animations.

## Memory management in C++

### A Method for Automatic Optimization of Dynamic Memory Management in C++

This paper (see [19]) explains the implemented pre-processor based method named Amplify. This method optimizes C++ applications to exploit the temporal locality in dynamic memory usage.

Amplify is based on Structure pools:

*"[...]The concept of object pools, also known as memory pools, is a well-known memory management technique. An object pool acts as a layer between the application and the dynamic memory management subsystem. When allocating memory for an object, a call to an object pool is made instead of making direct calls to the memory manager, i.e. malloc ( ) . An object pool holds a free list containing objects of a specific type. When an object is requested an object is extracted from the free list and returned to the caller. Thus, no call to the memory manager is needed, instead an already allocated, but not currently used, object is reused. Further, when deallocating memory a call to the pool is made instead of directly calling the memory manager, i.e. free ( ) . The pool will then insert the object into its free list for later reuse. This kind of strategy reduces the number of calls made to the memory management subsystem. However, in a multithreaded environment there is still a need for mutual exclusion during operations on the pool’s free list.[...]"*

The objects in the pool contains all the references to other objects. If an object is needed, when this object is extracted from the pool already contains all its fields and pointers to other elements.

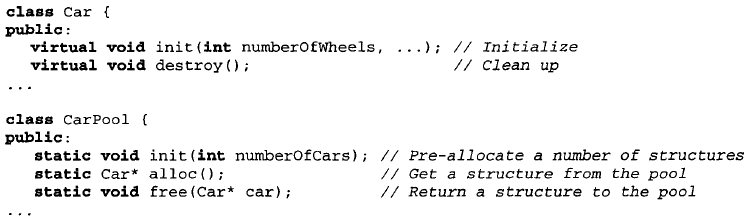


Figure 1.13 Pool example

The use of this structure is tested in three test cases:

* Test case 1: The worst case. A very simple case (with shallow structures) only for test if the Amplify overhead makes worst the overall performance
* Test case 2: A normal case of application
* Test case 3: The best case. Very deep structures in which the Amplify method can reuse objects and optimize the memory management.

The results are shown in the following figures. The figures are extracted from **¡Error! No se encuentra el origen de la referencia.**. In these tests the Amplify method is compared with ptmalloc [20] and Hoard [21].

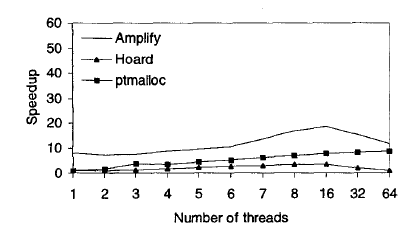


Figure 1.14 Test case 1

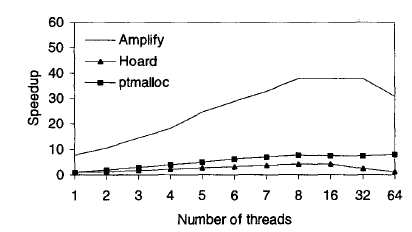


Figure 1.15 Test case 2

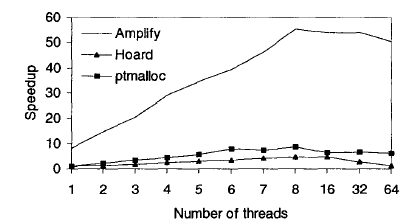


Figure 1.16 Test case 3

### Comparison of Garbage Collector Prototypes for C++ Applications

A Garbage Collector (GC) is a critical memory management process in programming languages such as C++. The programmers need to be aware about memory management and cannot be relieved from this task. They have to implement explicitly this task whatever the size of C++ applications. Consequently, this leads to errors and bugs (i.e. dangling pointers, allocation failures, etc.). Therefore, supporting implicit memory management based of aspect-oriented programming can provide a number of benefits such as freeing the programmer of memory management task.

This paper (see [22]) uses Aspect-Oriented Programming to manage implicity the allocationd and release of memory. Two prototypes of memory management are described in this project. These methods implement reference counting and mark and sweep techniques.

The authors conclude that implementing these aspect-oriented programming (AOP) techniques, the user is relieved from memory management tasks and the bugs are reduced. The futur research lines of this work is to add a garbage collector in C++ using AOP.

## Libraries and code infrastructures

### LLVM

LLVM (see [23], [24]) is a research project at the Univerity of Illinois. The goal of LLVM is to provide a modern, SSA-based compilation strategy capable of supporting both static and dynamic compilation or arbitrary programming languages. Since this point, LLVM has been an unmbrella project consisting of different subprojects.

LLVM Project can be defined as a collection of modular and reusable compiler and toolchain technologies.

Some sample projects based on LLVM are:

* Clang: C/C++ compiler
* libc++ and libc++ ABI: High-performance implementation of C++ standard library
* vmkit: Java and .NET Virtual Machines

LLVM is important in the science community and it is growing in the last years and there are research projects which combine other research lines with LLVM, an example is showed in [25] which is "An LLVM-based decoder for MPEG reconfigurable video coding". This project combines the active research of LLVM and MPEG in order to take benefit of the portability and scalability of LLVM in MPEG decoders systems.

LLVM is taking so importance than there is comparisons between LLVM and important compilers like GCC. The paper [26] studies this comparison and concludes that LLVM is better for the actual systems because of the portability and scalability that a Virtual Machine offers. LLVM has to improve its performance in some cases because the optimization of GCC is better, but improving this point, LLVM can be considered very near of GCC in a global point of view.

### Mono.CECIL

Mono.Cecil (see [27]) is a library written by Jb Evain to generate and inspect programs and libraries in the ECMA CIL format. It has full support for generics, and support some debugging symbol format.

In simple English, with Cecil, you can load existing managed assemblies, browse all the contained types, modify them on the fly and save back to the disk the modified assembly.

Today it is used by the Mono Debugger, the bug-finding and compliance checking tool Gendarme, MoMA, DB4O, as well as many other tools.

### OpenC++ and OpenJava

OpenC++ (see [28]) is a version of C++ with a Metaobject Protocol. In other words, it is a tool of source-code translation for C++. Programmers can easily implement various kinds of translation so that they can define new syntax, new annotation, and new object behavior.

OpenJava (see [29]) follows the same idea of OpenC++ but for Java language. It also uses MOP in order to provide customization to the programmers.

## Research projects

### Alchemy

Alchemy (see [30]) is a research project that allows users to compile C and C++ code that is targeted to run on the open source ActionScript Virtual Machine (AVM2). The purpose of this preview is to assess the level of community interest in reusing existing C and C++ libraries in Web applications that run on Adobe® Flash® Player and Adobe AIR®.

With Alchemy, Web application developers can now reuse hundreds of millions of lines of existing open source C and C++ client or server-side code on the Flash Platform. Alchemy brings the power of high performance C and C++ libraries to Web applications with minimal degradation on AVM2. The C/C++ code is compiled to ActionScript 3.0 as a SWF or SWC that runs on Adobe Flash Player 10 or Adobe AIR 1.5.

Alchemy is primarily intended to be used with C/C++ libraries that have few operating system dependencies. Ideally suited for computation-intensive use cases, such as audio/video transcoding, data manipulation, XML parsing, cryptographic functions or physics simulation, performance can be considerably faster than ActionScript 3.0 and anywhere from 2-10x slower than native C/C++ code. Alchemy is not intended for general development of SWF applications using C/C++.

With Alchemy, it is easy bridge between C/C++ and ActionScript 3.0 to expand the capabilities of applications on the Flash Platform, while ensuring that the generated SWCs and SWFs cannot bypass existing Flash Player security protections.

Adobe is providing some example libraries, and developers are encouraged to share their ported libraries.

The Alchemy preview is prerelease software that is not supported by Adobe and may contain bugs. It is therefore advised that Alchemy not be used to generate code for use in production.

We are making this software available to gauge community interest in Alchemy and welcome your feedback. Please use the feedback link below to request features, make comments and report problems. Please also note that this is a research project and there is no assurance that there will be a shipping version of Alchemy.

### Emscripten

Emscripten (see [31]) is an LLVM-to-JavaScript compiler. It takes LLVM bitcode (which can be generated from C/C++ using Clang, or any other language that can be converted into LLVM bitcode) and compiles that into JavaScript, which can be run on the web (or anywhere else JavaScript can run).

Using Emscripten, you can

* Compile C and C++ code into JavaScript and run that on the web
* Run code in languages like Python as well, by compiling CPython from C to JavaScript and interpreting code in that on the web

### Jangaroo

Jangaroo (see [32]) is an Open Source project building developer tools that adopt the power of ActionScript 3 to create high-quality JavaScript frameworks and applications. Jangaroo is released under the Apache License, Version 2.0.

The main tool is an ActionScript-3-to-JavaScript compiler called jooc. It takes source code written in a subset of ActionScript 3 and translates it into JavaScript 1.x that is understood by current browsers (with the aid of a light-weight runtime).

There are several applications using this converter. The major part of this applications are games and graphical applications.

### IKVM.NET

IKVM.NET (see [33]) is an implementation of Java for Mono and the Microsoft .NET Framework. It includes the following components:

* A Java Virtual Machine implemented in .NET
* A .NET implementation of the Java class libraries
* Tools that enable Java and .NET interoperability

IKVM.NET includes ikvmc, a Java bytecode to .NET IL translator. If you have a Java library that you would like to use in a .NET application, with ikvmc you can create a .dll library from a .jar library.

### Garbage collector projects

Garbage collector is an automatic memory management which frees memory occupied by objects that are no longer in use by the program. Was invented by John McCarthy in 1959.

There are programming languages which have integrated an implementation of garbage collector (i.e. .NET environment, Java ) but low level languages like C/C++ did not have garbage collector, the programmer is in charge of the memory management (this is a manual memory management).

There are different projects for implementing automatic memory management in manual manual memory management environments (such as C/C++).

#### Boehm-Demers-Weiser conservative garbage collector: A garbage collector for C and C++ REF

This project (see [34]) is an implementation of a garbage collector for C and C++ languages. This garbage collector allows allocating memory without explicity deallocating this memory after use it.

#### A garbage collection framework for C++ REF

This project (see [35]) is a very simple framework for use garbage collector philosophy in C++ which does not have native garbage collector support.

## Aporte

* Traducción a nivel de código
* Soprtar features de C# (máximas posibles)
* Proceso transparente: de .exe a .exe
* Gestión de memoria decente en C++
* Servicio cloud

# Interesting links

<http://www.gccxml.org/HTML/Index.html>

<http://digitocero.com/en/blog/exporting-and-visualizing-gccs-abstract-syntax-tree-ast>

<http://www.semanticdesigns.com/Products/FrontEnds/CppFrontEnd.html>

<http://www.csg.is.titech.ac.jp/~chiba/javassist/index.html>

<http://www.csg.is.titech.ac.jp/openjava/>

<http://labs.adobe.com/technologies/alchemy/>

<http://www.jangaroo.net/home/>

<http://code.google.com/p/nativeclient/>

<https://github.com/kripken/emscripten/wiki>

<http://www.hpl.hp.com/personal/Hans_Boehm/gc/>

<http://www.codeproject.com/Articles/912/A-garbage-collection-framework-for-C>

# References

1. http://www.mandreel.com/
2. http://jsil.org
3. http://xmlvm.org/overview/
4. http://tangiblesoftwaresolutions.com/index.htm
5. http://www.innoveware.com/tonicvm.html
6. http://community.versant.com/Projects/html/projectspaces/db4o\_product\_design/sharpen.html
7. http://sourceforge.net/projects/cxx2cs/
8. http://www.carlosag.net/tools/codetranslator/
9. http://www.excelsoftware.com/whitepapers.html -> Source Code Translation
10. http://wiki.sharpdevelop.net/ILSpy.ashx
11. http://www.reflector.net/
12. http://www.9rays.net/Category/54-spicesnet-decompiler.aspx
13. http://www.jetbrains.com/decompiler/
14. http://www.telerik.com/products/decompiler.aspx
15. http://www.neshkov.com/dj.html
16. http://www.vb-decompiler.org/
17. http://www.sothink.com/product/flashdecompiler/
18. http://www.backerstreet.com/rec/rec.htm
19. D. Häggander, P. Lidén and LArs Lundberg, "A Method for Automatic Optimization of Dynamic Memory Management in C++", at Dep. of Software Engineering and Computer Science, Blekinge Institute of Technology, Ronneby, Sweden.
20. E.Berger K McKinley, R. Blumofe, and P.Wilson, "Hoard: A Scalable Memory Allocator for Multithreaded Applications", in Proc. of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX), Cambridge, MA, Novemebre 2000
21. W. Gloger, "Dynamic memory allocator implementations in Linux", http://www.dent.med. uni-muenchen.de/~wmglo/malloc-slides.html
22. H. Mcheick, A.Sioud, "Comparison of Garbage Collector Prototypes for C++ Applications", at University of Quebec at Chicoutimi, UQAC, Canada
23. C. Lattner, V. Adve,"LLVM: A compilation framework for Lifelong Program Analysis & Transformation", University of Illinois at Urbana-Champaign
24. http://llvm.org/
25. J. Gorin, M.Wipliez, J.Piat F.Prêteux, M. Raulet, "An LLVM-based decoder for MPEG reconfigurable video coding", ARTEMIS, Institut Télécom SudParis, Evry, France and IETR, INSA Rennes, France.
26. J. Kim, S. Lee, S. Moon, S. Kim,"Comparison of LLVM and GCC on the ARM Platform", Seoul National University, Korea Institute of Science and Technology, Seoul, Korea
27. http://www.mono-project.com/Cecil
28. http://opencxx.sourceforge.net/
29. http://www.csg.is.titech.ac.jp/openjava/
30. http://labs.adobe.com/technologies/alchemy/
31. https://github.com/kripken/emscripten
32. http://www.jangaroo.net/home/
33. http://www.ikvm.net/index.html
34. http://www.hpl.hp.com/personal/Hans\_Boehm/gc/
35. http://www.codeproject.com/Articles/912/A-garbage-collection-framework-for-C