Mosa Project Documentation

Feb 14, 2020

Introduction

MOSA is an open source software project that natively executes .NET applications within a virtual hypervisor or on bare metal hardware!

The MOSA project consists of:

- Compiler - a high quality, multithreaded, cross-platform, optimizing .NET compiler
- Kernel - a small kernel operating system
- Device Drivers Framework - a modular, device drivers framework and device drivers
- Debugger - QEMU-based debugger

CHAPTER 1

Current Status

The target platforms are:

- Intel X86/32-bit (stable)
- Intel X64 (in development)
- ARM v6 (in early development)

The MOSA compiler supports nearly all and object oriented non-object oriented code, including:

- Generic Code (example: List<T>)
- Delegates (static and non-static) and with optional parameters
- Exception Handling (try, finally, and catch code blocks)

The MOSA compiler seeks to provide high quality code generation using the following optimizations:

- Constant Folding and Propagation
- Strength Reduction optimization
- Dead Code Elimination
- Single Static Assignment (SSA)
- Global Value Numbering / Common Subexpession Elimination
- Sparse Conditional Constant Propagation
- Inlined Expansion
- Loop-Invariant Code Motion
- Block Reordering
- Greedy Register Allocation

CHAPTER 2

Getting Started

2.1 Download

The MOSA project is available as a zip download or via git:

```
git clone https://github.com/mosa/MOSA-Project.git
```

2.2 Prerequisites

You will also need the following prerequisites:

2.2.1 Windows

Install any Visual Studio version 2018 or newer. All editions are supported including the fully-featured free Community Edition.

Note: The MOSA source code repository includes Qemu virtual emulator for Windows.

The CodeMaid Visual Studio Extension is strongly recommended for MOSA contributors.

2.2.2 Linux

Install Mono and Qemu.

The minimum supported version of Mono is 5.16.

If using the APT package manager you can use the following command to quickly set up QEMU and Mono

```
sudo apt-get -y install mono-devel qemu
```

2.2.3 Mac

Install Mono and Qemu.

2.3 Running on Windows

Double click on the "Compile.bat" script in the root directory to compile all the tools, sample kernels, and demos.

Next double click on the "Launcher.bat" script, which will bring up the MOSA Launcher tool (screenshot below) that can:

- Compile the operating system
- Create a virtual disk image, with the compiled binary and boot loader
- Launch a virtual machine instance (QEMU by default)

By default, the CoolWorld operating system demo is pre-selected. Click the "Compile and Run" button to compile and launch the demo.

CHAPTER 3

Join the Discussion

Join us on Gitter chat. This is the most interactive way to connect to MOSA's development team.

CHAPTER 4

License

MOSA is licensed under the New BSD License.

CHAPTER 5

Documentation

5.1 Frequently Asked Questions (FAQs)

These are questions we're frequently answering in our IRC channel, the mailing list or which come up during conversations.

5.1.1 What does MOSA stand for?

Managed Operation System Alliance. It was an alliance with the SharpOS project to create a .NET based operating system and toolsets. As operating systems share a lot of common groundwork MOSA aims to standardize interfaces in order to foster portability and to provide both projects with basic implementations of these interfaces.

5.1.2 Who can join?

Anyone who is interested in operating system or .NET development can join. You have to keep in mind our *New BSD License* for your contributions though.

5.1.3 What kind of .NET runtime will be available?

We are developing an entirely new runtime as part of the MOSA effort. This runtime is developed along the CIL specifications published by ECMA. We are building our own runtime with pluggable algorithms in order to be very flexible and usable for research.

5.1.4 How is the Cosmos project different than MOSA?

Cosmos is designed to be an operating system toolkit plugin for Visual Studio. The Cosmos toolkit, once installed, integrates with Visual Studio in two significant ways. First, the toolkit introduces a new Cosmos project type that can launch and control the build process. Second, the toolkit integrates with Visual Studio's debugger and provides break

points and watches. The toolkit requires Microsoft's implementation of the .NET framework to compile a Cosmos operating system.

In comparison, MOSA has no dependencies on any Microsoft's applications including Visual Studio, the .NET framework or Windows operation system. MOSA can run on Windows, Linux or the Apple's OSX operating systems.

Another important difference is Cosmos compiles to Assembly ASM and uses Netwide Assembler, NASM, to finally compile to binary. MOSA compiles directly to binary and has its own linker implementation.

5.1.5 Are Cosmos and MOSA working together?

No; Cosmos and MOSA are seperate and independent projects.

5.2 Compiler Design

The MOSA Compiler framework is designed around two pipelines each with multiple stages.

5.2.1 Compiler Pipeline

The **Compiler Pipeline** is the primary pipeline that executes the steps necessary to compile an application, such as building the type system, compiling each method, linking, and emitting the final object file.

One stage executes the **Method Compiler Pipeline** for each method in the application.

5.2.2 Method Compiler Pipeline

The **Method Compiler Pipeline** is used to compile a single method by progressively lowers the high level instruction representation to the final opcode instructions of the target platform. This pipeline is executed at least once for all methods in the application.

All the stages can be grouped into one or more of these categories:

Decoding Stage Creates an instruction stream for a method from the source representation, such as a method from a .NET application

Transformation Stages Transforms the instruction stream between various representations, usually from a higher level to a lower level representations

Optimization Stages Transforms instructions intended to optimize the code to execute faster

Register Allocation Stage Allocates the virtual registers in the instruction stream to platform specific physical registers

Platform Specific Transformation Stages Transforms the instructions in the stream into specific platform opcodes

5.2.3 Intermediate Representations

The compiler framework uses a linear intermediate representation (vs an expression tree) to transform the source application into binary machine code. There are several levels of intermediate representations before code generation. These are:

- Common Intermediate Language (CIL)
- High-Level Intermediate Representation (IR)

- Machine specific Intermediate Representation (MIR)

During compilation of an CIL method the instructions are represented by CIL instruction classes and moving forward, the linear instruction stream is modified to use instructions from the intermediate representation. In some cases an instruction from the intermediate representation can not be emitted directly to machine code and it is replaced by a sequence of machine specific instructions. The machine specific instruction classes are provided by the appropriate platform.

5.3 Compiler Optimizations

5.3.1 Optimizations

Basic Optimizations Basic optimizations are a family of common types of optimizations, such as Constant Folding, Strength Reduction, Simplification, and many others.

Bit Tracker Bit Tracker tracks the known state of bits and value ranges thru the various operations. This enables various optimization and shortcuts.

Long Expansion Expands 64-bit instructions into 32-bit components for platforms without native 64-bit instructions.

Inline Expansion Inline Expansion replaces a methods call site with the body of the called method. This improves the performance, because calls can be expensive (storing the registers, placing the arguments onto stack, jumping to another location).

Static Single Assignment Form Transforms instructions to Single Static Assignment (SSA) form. In SSA form, all virtual registers may only have one definition. This is not an optimization by itself, but this form enables other optimization opportunities in other types of optimizations.

Sparse Conditional Constant Propagation Sparse conditional constant propagation is an optimization applied after conversion to static single assignment form (SSA). It simultaneously removes dead code and propagates constants. It can finds more constant values, and thus more opportunities for improvement, than other optimizations.

Value Numbering Value numbering is a technique of determining when two computations in a program are equivalent and eliminating one of them with while preserving the same semantics.

Two Pass Optimizations This options causes the optimization stages to be executed again, possibility unlocked additional optimizations.

5.4 Settings Options

Here are the setting options for the compiler tools:

5.4.1 Compiler Settings

Settings	Description
Compiler.Platform	Platform x86, x64, ARMv8A32
Compiler.BaseAddress	Base address of the compiled application
Compiler.TraceLevel	Trace level for debugging
Compiler.MethodScanner	If true, enable the experimental method scanner
Compiler.Multithreading	If true, enables multithreading during compiling process
Compiler.MultithreadingMaxThreads	Maximum number of threads used by the compiler
Compiler.Binary	If true, emits object file, otherwise no object file is created
Compiler.EmitInline	If true, emits all inlined methods into the object file
Compiler.OutputFile	Filename of the object file
Compiler.SourceFiles	Filename(s) of the source files

5.4.2 Compiler Optimizations Settings

Settings	Description
Optimizations.Basic	If true, enables prebuilt transformation optimizations, like constant folding and strength reduction
Optimizations.SSA	If true, transforms instructions to Single Static Assignment (SSA) form
Optimizations.SCCP	If true, enables Sparse Conditional Constant Propagation (SCCP) optimizations
Optimizations.ValueNumbering	If true, enables the Value Numbering (VN) optimizations
Optimizations.LongExpansion	If true, transforms some 64-bit instructions into 32-bit instructions prior to platform transformations
Optimizations.BitTracker	If true, enables the bit tracker optimizations
Optimizations.LoopInvariantCodeMotion	If true, enables the loop invariant code motion optimizations
Optimizations.Devirtualization	If true, transforms virtual methods calls into static method calls
Optimizations.Platform	If true, enable platform specific optimizations
Optimizations.Inline	If true, small methods can be inlined
Optimizations.Inline.Maximum	Maximum number of instructions that can be inlined within a method
Optimizations.Inline.AggressiveMaximum	Maximum number of instructions that can be inlined when a method is explicited marked to be inlined
Optimizations.Inline.ExplicitOnly	If true, only explicitly marked methods are inlined
Optimizations.TwoPass	If true, some optimization stages are executed twice
Optimizations.Inline.Aggressive	Methods to be aggressively inline
Optimizations.Inline.Exclude	Methods that may not be inlined

5.4.3 Linker Settings

Settings	Description
Linker.Format	Type of ELF object file elf32 or elf64
Linker.Symbols	If true, emits the symbols into the object file
Linker.StaticRelocations	If true, emits static relocation information into the object file
Linker.Dwarf	If true, emits DWARF debug information into the object file
Linker.ShortSymbolNames	If true, emits short symbol names into the object file
Linker.CustomSections.{Name}.SectionName	Name of linker section with this section name
Linker.CustomSections.{Name}.SourceFile	Name of linker section using the specific file
Linker.CustomSections.{Name}.Address	Name of linker section with this address

5.4.4 Common Settings

Settings	Description
SearchPaths	Folder to search for files
DefaultFolder	Default folder to output files
TemporaryFolder	Specifies a temporary folder

5.4.5 Compiler Debug Settings

Settings	Description
CompilerDebug.Statistics	If true, enables statistics gathering
CompilerDebug.DebugFile	Filename to emit a MOSA specific debug information
CompilerDebug.MapFile	Filename to emit a map of all symbols
CompilerDebug.CompileTimesFile	Filename to emit compile times for each method
CompilerDebug.AsmFile	Filename to emit ASM disassembly
CompilerDebug.NasmFile	Filename to emit disassembly using the NASM tool
CompilerDebug.InlinedFile	Filename to emit a list of all methods that were inlined
CompilerDebug.PreLinkHashFile	Filename to emit a list of all methods with their hash value prior to linking
CompilerDebug.PostLinkHashFile	Filename to emit a list of all methods with their hash value after linking

5.4.6 Compiler X86 Settings

Settings	Description
X86.InterruptMethodName	Name of the method that handles interrupts

5.4.7 Explorer Settings

Settings	Description
Explorer.Filter	Specifies the default method filter name when Explorer is launched

5.4.8 Launcher Settings

Settings	Description
Launcher.Start	If true, immediately start the compiler upon launch
Launcher.Launch	If true, launch a virtual machine after compiling the application and generating the virtual machine image
Launcher.Exit	If true, exit immediately after launch
Launcher.PlugKorlib	If true, automatically include the plugs for CoreLib
Launcher.HuntForCorLib	If true, search for CoreLib in various directories
Launcher.LaunchGDB	If true, launch the GNU GDB application after VM launch
Launcher.LaunchDebugger	If true, launch the MOSA debugger application after VM launch
Launcher.Test	If true, monitors VM serial for success or failure messages

5.4.9 Image Settings

Settings	Description
Image.Format	Format of the vritual image file BIN, IMG, VHD, VDI, ISO, VMDK
Image.FileSystem	File system of the primary partition in the image file FAT12, FAT16, FAT32, ISO????
Image.BootLoader	Type of bootloader grub0.97, grub2.00, syslinux6.03, syslinux3.72
Image.Destination	Destination directory of the image file
Image.ImageFile	Filename of the image file

5.4.10 Emulator Settings

Settings	Description
Emulator	Type of Emulator Qemu, VMware, Bochs
Emulator.Memory	Amount of memory for the virtual machine in MB
Emulator.Display	If true, show the video display
Emulator.GDB	If true, enables GDB within emulator
Emulator.Serial	Serial Emulation type None, Pipe, TCPServer, TCPClient
Emulator.Serial.Host	Serial Host Name or IP address
Emulator.Serial.Port	Serial Port
Emulator.Serial.Pipe	Serial Pipename

5.4.11 GDB Settings

Settings	Description
GDB.Host	Host IP or Name for GDB
GDB.Port	Port Number for GDB

5.4.12 Multiboot Settings

Settings	Description
Multiboot.Version	Multiboot version none, v1, v2
Multiboot.Video	If true, enable VGA BIOS Extension (VBE)
Multiboot.Video.Width	Video Width
Multiboot.Video.Height	Video Height
Multiboot.Video.Depth	Video Depth

5.4.13 Debugger Settings

Settings	Description
Debugger.WatchFile	Filename of the watch file
Debugger.BreakpointFile	Filename of the breakpoint file

5.4.14 Application Location Settings

Settings	Description
AppLocation.Bochs	Location of the BOCHS application
AppLocation.Qemu	Location of the QEMU application
AppLocation.QemuBIOS	Location of the QEMU BIOS
AppLocation.QemuImg	Location of the QEMUImg application
AppLocation.VmwarePlayer	Location of the VMPlayer application
AppLocation.Ndisasm	Location of the Ndisasm application
AppLocation.Mkisofs	Location of the Mkisofs application
AppLocation.GDB	Location of the QEMU application

5.4.15 Import Settings

Settings	Description
Import	Filename of another settings file to import

5.5 Command Line Arguments

The command line arguments serve as shortcuts to the common set of *Settings Options* used by the MOSA tools.

Tip: Specific settings may also be specified on the command line using the `-setting` or `-s` arguments. For example to set the `Compiler.OutputFile` settings with `Mosa.HelloWorld.x86.bin`, pass the following two arguments `-setting Compiler.OutputFile=Mosa.HelloWorld.x86.bin` on the command line.

Below are the command line arguments available:

Argument	Setting	Value Set
Compiler:		
{none}	Compiler.SourceFiles	{value}
-settings	Settings	{value}
-s	Settings	{value}
-o	Compiler.OutputFile	{value}
-threading	Compiler.Multithreading	true
-threading-off	Compiler.Multithreading	false
-base	Compiler.BaseAddress	{value}
-scanner	Compiler.MethodScanner	true
-no-code	Compiler.Binary	false
-path	SearchPaths	
-inline	Optimizations.Inline	true
-inline-off	Optimizations.Inline	false
-ssa	Optimizations.SSA	true
-ssa-off	Optimizations.SSA	false
-sccp	Optimizations.SCCP	true
-sccp-off	Optimizations.SCCP	false
-basic-optimizations	Optimizations.Basic	true

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
-basic-optimizations-off	Optimizations.Basic	false
-inline-explicit	Optimizations.Inline.ExplicitOnly	true
-inline-explicit-off	Optimizations.Inline.ExplicitOnly	false
-long-expansion	Optimizations.LongExpansion	true
-long-expansion-off	Optimizations.LongExpansion	false
-two-pass	Optimizations.TwoPass	true
-two-pass-off	Optimizations.TwoPass	true
-value-numbering	Optimizations.ValueNumbering	true
-value-numbering-off	Optimizations.ValueNumbering	false
-loop-invariant-code-motion	Optimizations.LoopInvariantCodeMotion	true
-loop-invariant-code-motion-off	Optimizations.LoopInvariantCodeMotion	false
-platform-optimizations	Optimizations.Platform	true
-platform-optimizations-off	Optimizations.Platform	false
-bit-tracker	Optimizations.BitTracker	true
-bit-tracker-off	Optimizations.BitTracker	false
-devirtualization	Optimizations.Devirtualization	true
-devirtualization-off	Optimizations.Devirtualization	false
-inline-level	Optimizations.Inline.Maximum	{value}
-platform	Compiler.Platform	{value}
-x86	Compiler.Platform	x86
-x64	Compiler.Platform	x64
-armv8a32	Compiler.Platform	armv8a32
Compiler - Debug Output Information:		
-output-nasm	CompilerDebug.NasmFile	%DEFAULT%
-output-asm	CompilerDebug.AsmFile	%DEFAULT%
-output-map	CompilerDebug.MapFile	%DEFAULT%
-output-time	CompilerDebug.CompilerTimeFile	%DEFAULT%
-output-debug	CompilerDebug.DebugFile	%DEFAULT%
-output-inlined	CompilerDebug.InlinedFile	%DEFAULT%
-output-hash	CompilerDebug.PreLinkHashFile	%DEFAULT%
-output-hash	CompilerDebug.PostLinkHashFile	%DEFAULT%
Compiler - X86:		
-interrupt-method	X86.InterruptMethodName	{value}
Linker:		
-emit-all-symbols	Linker.Symbols	true
-emit-all-symbols-off	Linker.Symbols	false
-emit-relocations	Linker.StaticRelocations	true
-emit-relocations-off	Linker.StaticRelocations	false
-emit-static-relocations	Linker.StaticRelocations	true
-emit-drawf	Linker.Drawf	true
-emit-drawf-off	Linker.Drawf	false
-drawf	Linker.Drawf	true
Explorer:		
-filter	Explorer.Filter	{value}

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
Launcher:		
-autoexit	Launcher.Exit	true
-autoexit-off	Launcher.Exit	false
-autostart	Launcher.Start	true
-autostart-off	Launcher.Start	false
-autolaunch	Launcher.Launch	true
-autolaunch-off	Launcher.Launch	false
-launch	Launcher.Launch	true
-launch-off	Launcher.Launch	false
Launcher - Emulator:		
-emulator	Emulator	
-qemu	Emulator	qemu
-vmware	Emulator	vmware
-bochs	Emulator	bochs
-display	Emulator.Display	on
-display-off	Emulator.Display	off
-memory	Emulator.Memory	
-qemu-gdb	Emulator.GDB	false
Launcher - Image:		
-image	Image.ImageFile	{value}
-destination	Image.Folder	{value}
-dest	Image.Folder	{value}
-vhd	Image.Format	vhd
-img	Image.Format	img
-vdi	Image.Format	vdi
-iso	Image.Format	iso
-vmdk	Image.Format	vmdk
-blocks	Image.DiskBlocks	
-volume-label	Image.VolumeLabel	
-mbr	Image.MasterBootRecordFile	
-boot	Image.BootBlockFile	
Launcher - Boot:		
-multiboot-v1	Multiboot.Version	v1
-multiboot-v2	Multiboot.Version	v2
-multiboot-none	Multiboot.Version	
-multiboot	Multiboot.Version	{value}
Launcher - Serial:		
-serial-connection	Emulator.Serial	
-serial-pipe	Emulator.Serial	pipe
-serial-tcpclient	Emulator.Serial	tcpclient
-serial-tcpserver	Emulator.Serial	tcpserver
-serial-connection-port	Emulator.Serial.Port	{value}
-serial-connection-host	Emulator.Serial.Host	{value}

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
Launcher - Video BIOS Extension (VBE):		
-video	Multiboot.Video	true
-video-width	Multiboot.Video.Width	{value}
-video-height	Multiboot.Video.Height	{value}
-video-depth	Multiboot.Video.Depth	{value}
Launcher - GDB:		
-launch-gdb-debugger	Launcher.LaunchDebugger	true
Launcher - Boot Loader:		
-bootloader	Image.BootLoader	{value}
-grub	Image.BootLoader	grub0.97
-grub0.97	Image.BootLoader	grub0.97
-grub2.00	Image.BootLoader	grub2.00
-syslinux	Image.BootLoader	syslinux_v3.72
-syslinux3.72	Image.BootLoader	syslinux3.72
-syslinux6.0	Image.BootLoader	syslinux6.03
Launcher - Advance:		
-hunt-corlib	Launcher.HuntForCorLib	true
-plug-korlib	Launcher.PlugKorlib	true
Launcher - GDB		
-gdb	Launcher.LaunchDebugger	true
Launcher & Debugger - GDB		
-gdb-port	GDB.Port	{value}
-gdb-host	GDB.Host	{value}
Debugger:		
-breakpoints	Debugger.BreakpointFile	{value}
-watch	Debugger.WatchFile	{value}
Optimization Levels:		
-o0	Optimizations.Basic	false
-o0	Optimizations.SSA	false
-o0	Optimizations.ValueNumbering	false
-o0	Optimizations.SCCP	false
-o0	Optimizations.Devirtualization	false
-o0	Optimizations.LongExpansion	false
-o0	Optimizations.Platform	false
-o0	Optimizations.Inline	false
-o0	Optimizations.LoopInvariantCodeMotion	false
-o0	Optimizations.BitTracker	false
-o0	Optimizations.TwoPass	false
-o0	Optimizations.Inline.Maximum	0
-o1	Optimizations.Basic	true
-o1	Optimizations.SSA	false

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
-o1	Optimizations.ValueNumbering	false
-o1	Optimizations.SCCP	false
-o1	Optimizations.Devirtualization	true
-o1	Optimizations.LongExpansion	false
-o1	Optimizations.Platform	false
-o1	Optimizations.Inline	false
-o1	Optimizations.LoopInvariantCodeMotion	false
-o1	Optimizations.BitTracker	false
-o1	Optimizations.TwoPass	false
-o1	Optimizations.Inline.Maximum	0
-o2	Optimizations.Basic	true
-o2	Optimizations.SSA	true
-o2	Optimizations.ValueNumbering	true
-o2	Optimizations.SCCP	false
-o2	Optimizations.Devirtualization	true
-o2	Optimizations.LongExpansion	false
-o2	Optimizations.Platform	false
-o2	Optimizations.Inline	false
-o2	Optimizations.LoopInvariantCodeMotion	false
-o2	Optimizations.BitTracker	false
-o2	Optimizations.TwoPass	false
-o2	Optimizations.Inline.Maximum	0
-o3	Optimizations.Basic	true
-o3	Optimizations.SSA	true
-o3	Optimizations.ValueNumbering	true
-o3	Optimizations.SCCP	true
-o3	Optimizations.Devirtualization	true
-o3	Optimizations.LongExpansion	false
-o3	Optimizations.Platform	false
-o3	Optimizations.Inline	false
-o3	Optimizations.LoopInvariantCodeMotion	false
-o3	Optimizations.BitTracker	false
-o3	Optimizations.TwoPass	false
-o3	Optimizations.Inline.Maximum	0
-o4	Optimizations.Basic	true
-o4	Optimizations.SSA	true
-o4	Optimizations.ValueNumbering	true
-o4	Optimizations.SCCP	true
-o4	Optimizations.Devirtualization	true
-o4	Optimizations.LongExpansion	true
-o4	Optimizations.Platform	false
-o4	Optimizations.Inline	false
-o4	Optimizations.LoopInvariantCodeMotion	false
-o4	Optimizations.BitTracker	false
-o4	Optimizations.TwoPass	false
-o4	Optimizations.Inline.Maximum	0

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
-o5	Optimizations.Basic	true
-o5	Optimizations.SSA	true
-o5	Optimizations.ValueNumbering	true
-o5	Optimizations.SCCP	true
-o5	Optimizations.Devirtualization	true
-o5	Optimizations.LongExpansion	true
-o5	Optimizations.Platform	true
-o5	Optimizations.Inline	false
-o5	Optimizations.LoopInvariantCodeMotion	false
-o5	Optimizations.BitTracker	false
-o5	Optimizations.TwoPass	false
-o5	Optimizations.Inline.Maximum	0
-o6	Optimizations.Basic	true
-o6	Optimizations.SSA	true
-o6	Optimizations.ValueNumbering	true
-o6	Optimizations.SCCP	true
-o6	Optimizations.Devirtualization	true
-o6	Optimizations.LongExpansion	true
-o6	Optimizations.Platform	true
-o6	Optimizations.Inline	true
-o6	Optimizations.LoopInvariantCodeMotion	false
-o6	Optimizations.BitTracker	false
-o6	Optimizations.TwoPass	false
-o6	Optimizations.Inline.Maximum	5
-o7	Optimizations.Basic	true
-o7	Optimizations.SSA	true
-o7	Optimizations.ValueNumbering	true
-o7	Optimizations.SCCP	true
-o7	Optimizations.Devirtualization	true
-o7	Optimizations.LongExpansion	true
-o7	Optimizations.Platform	true
-o7	Optimizations.Inline	false
-o7	Optimizations.LoopInvariantCodeMotion	true
-o7	Optimizations.BitTracker	false
-o7	Optimizations.TwoPass	false
-o7	Optimizations.Inline.Maximum	10
-o8	Optimizations.Basic	true
-o8	Optimizations.SSA	true
-o8	Optimizations.ValueNumbering	true
-o8	Optimizations.SCCP	true
-o8	Optimizations.Devirtualization	true
-o8	Optimizations.LongExpansion	true
-o8	Optimizations.Platform	true
-o8	Optimizations.Inline	true
-o8	Optimizations.LoopInvariantCodeMotion	true

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
-o8	Optimizations.BitTracker	true
-o8	Optimizations.TwoPass	true
-o8	Optimizations.Inline.Maximum	10
-o9	Optimizations.Basic	true
-o9	Optimizations.SSA	true
-o9	Optimizations.ValueNumbering	true
-o9	Optimizations.SCCP	true
-o9	Optimizations.Devirtualization	true
-o9	Optimizations.LongExpansion	true
-o9	Optimizations.Platform	true
-o9	Optimizations.Inline	true
-o9	Optimizations.LoopInvariantCodeMotion	true
-o9	Optimizations.BitTracker	true
-o9	Optimizations.TwoPass	true
-o9	Optimizations.Inline.Maximum	15
-oNone	Optimizations.Basic	false
-oNone	Optimizations.SSA	false
-oNone	Optimizations.ValueNumbering	false
-oNone	Optimizations.SCCP	false
-oNone	Optimizations.Devirtualization	false
-oNone	Optimizations.LongExpansion	false
-oNone	Optimizations.Platform	false
-oNone	Optimizations.Inline	false
-oNone	Optimizations.LoopInvariantCodeMotion	false
-oNone	Optimizations.BitTracker	false
-oNone	Optimizations.TwoPass	false
-oNone	Optimizations.Inline.Maximum	0
-oMax	Optimizations.Basic	true
-oMax	Optimizations.SSA	true
-oMax	Optimizations.ValueNumbering	true
-oMax	Optimizations.SCCP	true
-oMax	Optimizations.Devirtualization	true
-oMax	Optimizations.LongExpansion	true
-oMax	Optimizations.Platform	true
-oMax	Optimizations.Inline	true
-oMax	Optimizations.LoopInvariantCodeMotion	true
-oMax	Optimizations.BitTracker	true
-oMax	Optimizations.TwoPass	true
-oMax	Optimizations.Inline.Maximum	15
-oSize	Optimizations.Basic	true
-oSize	Optimizations.SSA	true
-oSize	Optimizations.ValueNumbering	true
-oSize	Optimizations.SCCP	true
-oSize	Optimizations.Devirtualization	true
-oSize	Optimizations.LongExpansion	true

Continued on next page

Table 1 – continued from previous page

Argument	Setting	Value Set
-oSize	Optimizations.Platform	true
-oSize	Optimizations.Inline	true
-oSize	Optimizations.LoopInvariantCodeMotion	true
-oSize	Optimizations.BitTracker	true
-oSize	Optimizations.TwoPass	true
-oSize	Optimizations.Inline.Maximum	3
-oFast	Optimizations.Basic	true
-oFast	Optimizations.SSA	true
-oFast	Optimizations.ValueNumbering	true
-oFast	Optimizations.SCCP	false
-oFast	Optimizations.Devirtualization	true
-oFast	Optimizations.LongExpansion	false
-oFast	Optimizations.Platform	false
-oFast	Optimizations.Inline	false
-oFast	Optimizations.LoopInvariantCodeMotion	false
-oFast	Optimizations.BitTracker	false
-oFast	Optimizations.TwoPass	false
-oFast	Optimizations.Inline.Maximum	0

Note: `{value}` is the next argument

5.6 MOSA Tools

The MOSA project is developing a set of tools along with its core operating system components. This page provides quick links to descriptions of all MOSA tools.

- *MOSA Compiler*
- *MOSA Launcher*
- *MOSA Explorer*
- *MOSA Debugger*
- *MOSA Boot Image Tool*

5.7 MOSA Compiler

The **MOSA Compiler** is a console application used to compile a .NET application to a binary object file.

The compiler is invoked via Command Line:

```
Mosa.Tool.Compiler.exe -o Mosa.HelloWorld.x86.bin Mosa.HelloWorld.x86.exe
```

Output:

```
X:\MOSA-Project\bin>Mosa.Tool.Compiler.exe -o Mosa.HelloWorld.x86.bin Mosa.HelloWorld.
↪x86.exe
MOSA Compiler, Version 2.0.0.0.
Copyright 2020 by the MOSA Project. Licensed under the New BSD License.

Parsing options...
 > Output file: _main.exe
 > Input file(s): Mosa.HelloWorld.x86.exe
 > Platform: x86

Compiling ...

0.56 [0] Compile Started
0.59 [0] Setup Started
0.60 [0] Setup Completed
0.60 [0] Compiling Methods
2.97 [0] Compiling Methods Completed
2.97 [0] Finalization Started
3.06 [0] Linking Started
3.08 [0] Linking Completed
3.10 [0] Finalization Completed
3.10 [0] Compile Completed
```

5.7.1 Command Line Options

[TODO]

5.8 MOSA Launcher

The **MOSA Launcher** is GUI application used to select various compiler and build options. Once options are select, the tool automates the entire build process including launching of a virtual machine.

5.8.1 Command Line Options

[TODO]

5.9 MOSA Launcher Console

The **MOSA Launcher Console** is a console application that automates the entire build process including launching of a virtual machine.

Tip: See *MOSA Launch* for a graphic user interface version of this console application.

5.9.1 Example

A quick example that compiles `Mosa.TestWorld.x86.exe` demo with `-oMax` (all optimization enabled) and launches it using `Qemu`:

```
Mosa.Tool.Launcher.Console.exe -oMax Mosa.TestWorld.x86.exe
```

```
Current Directory: X:\MOSA-Project-tgiphil\bin
Compiling: 0.89 secs: Compile Started
Compiling: 0.94 secs: Compiling Methods
Compiling: 4.32 secs: Compiling Methods Completed
Compiling: 4.56 secs: Linking Started
Compiling: 4.59 secs: Linking Completed
Compiling: 4.66 secs: Compile Completed
Generating Image: img
Launching Application: ..\Tools\QEMU\qemu-system-i386.exe
Arguments:  -L "..\Tools\QEMU" -cpu qemu32,+sse4.1 -serial null -hda
↪"C:\Users\phil\AppData\Local\Temp\MOSA\Mosa.TestWorld.x86.img"
```

5.9.2 Command Line Options

See the *command line arguments* for a list of available options. Here are the most common options available:

5.10 MOSA Explorer

The **MOSA Explorer** is GUI application used to visualize the compiler transformation of a method from the highest respresentation to the lowest, lowest level. The instruction stream at each stage can be viewed. In addition, specific stage logs is also available. in some cases, these logs describe why certain transforms were made.

5.10.1 Command Line Options

[TODO]

5.11 MOSA Debugger

The **MOSA Debugger** is GUI application used to debug a MOSA compiled application using QEMU Virtual Machine Emulator.

5.11.1 Command Line Options

[TODO]

5.12 MOSA Boot Image Tool

The Boot Image Tool is a command line application that can be used to create bootable images containing a MOSA operating system application.

Warning: This tool has been superceded by the MOSA Launcher Tool which can automate the entire build chain functionality.

The tool has several command line options. Example:

```
Mosa.Tool.CreateBootImage.exe -o bin/Mosa.HelloWorld.x86.img --mbr Tools/syslinux/3.
↪72/mbr.bin --boot Tools/syslinux/3.72/ldlinux.bin --syslinux --volume-label␣
↪MOSABOOT --blocks 25000 --filesystem fat16 --format img Tools/syslinux/3.72/ldlinux.
↪sys Tools/syslinux/3.72/mboot.c32 Demos/unix/syslinux.cfg bin/Mosa.HelloWorld.x86.
↪bin,main.exe
```

The following options are supported:

Table 2: Arguments

Option	Arguments	Description
–volume	Volume Name	Set the volume name for the first partition
–blocks	# of Blocks	Set the number of 512-byte blocks
–filesystem	fat12/fat16/fat32	File System type
–format	img/vhd/vdi/vmdk/img	Disk Image Format
–syslinux		Patch disk image for syslinux
–mbr	Filename	Use file for Master Boot Record
–boot	Filename	Use file for Boot Record
	Filename[,Destination]	Include file in file system. Optional Destination will rename the file

The tool can create disk images for the following emulators:

Table 3: File formats

Emulator	File format
Virtual PC 2004/2007	.VHD
Virtual Server	.VHD
VMware	.VHD
VirtualBox	.VDI
QEMU	.IMG
Raw Image	.IMG

5.13 Unit Tests

Execute the script `RunAllUnitTestsWithPause.bat` in the `/Tests` subdirectory.

The unit tests will take a few minutes to execute on modern PC. The results of all tests will be automatically displayed on the screen. The last line shows the total number of tests and failed tests, and the total time. Similar to the following:

```
Total Elapsed: 95.3 secs

Unit Test Results:
   Passed:   68164
   Skipped:  4
   Failures: 0
   Total:    68168

All unit tests passed successfully!
```

5.14 USB Flash Drive Installation

While most of the development and testing of MOSA is done using virtualization software, MOSA does indeed boot on real hardware too.

Below are the steps for deploying a MOSA disk image to a USB flash drive:

> **Warning:** These instructions may vary slightly depending on your installation.

1. Create a MOSA disk image using the MOSA Launcher Tool.
2. Download the dd utility for Windows.
3. Copy the `dd.exe` executable to the build directory (usually a sub-folder under temp):

```
%TEMP%\MOSA
```

4. Open a command prompt window and change directory to the build directory.

```
cd %TEMP%\MOSA
```

5. Connect the USB key you wish to ERASE and install the MOSA image onto.

> **Danger:** Data on the USB flash drive will be lost!

6. Determine the device path for the USB flash drive.

Get a list all the block devices on your system by typing the command below. Find the one for the USB flash drive you just connected. Be careful, if you select or mistype the wrong device, you can corrupt your hard drive or other storage devices. Unless you understand these steps completely, do not proceed.

```
dd -list
```

7. Type the following and substitute the of= parameter with the device path found in the previous step.

```
dd of=\\?\Device\HarddiskX\PartitionX if=bootimage.img bs=512 -progress
```

8. Wait until all the blocks are written to the USB key before disconnecting it.
9. Now boot a PC or laptop with the USB flash drive connected!

5.15 Get Involved

We need your help!

5.15.1 Contribution process

If you are interested in supporting this project, join the project itself. Start downloading the code, look at the documentation provided and join us on Gitter chat. This is the most interactive way to connect to MOSA's development team.

5.15.2 Jobs

We need help to achieve our goals. You can either help us with documentation, community, by fixing bugs in our code base, fixing reported issues or taking on one of the open jobs. If you want to contribute, please make yourself heard on our IRC channel.

5.15.3 Rules

We are open to contributions from all areas. There are some exceptions though, which mean that we can not accept your contributions if you:

- have inspected proprietary code with Reflector, ildasm or similar tools and you plan on providing MOSA an implementation of that.
- have access to proprietary code related to operating system development, managed code or runtime implementations for managed environments.

Please be careful with any contribution you make regarding to patents, other open source licenses and any potential restrictions. We can only accept contributions compatible with the New BSD License. MIT license is a compatible license, while GNU licenses are incomparable. Read more about our License policy.

5.16 Authors

The MOSA project is a team effort and we want to recognize everyone's contribution. We list all the contributors to MOSA in our source code repository.

View the Credits.txt file.

If you have contributed to MOSA and your name is not on the list, please add your name and submit a pull request.

5.17 New BSD License

MOSA is licensed under the New BSD License:

```
Copyright (c) 2008, MOSA-Project
All rights reserved.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

* Redistributions of source code must retain the
  above copyright notice, this list of conditions
  and the following disclaimer.

* Redistributions in binary form must reproduce the
  above copyright notice, this list of conditions and
  the following disclaimer in the documentation and/or
  other materials provided with the distribution.

* Neither the name of Managed Operating System Alliance (MOSA)
  nor the names of its contributors may be used to endorse
  or promote products derived from this software without
```

(continues on next page)

(continued from previous page)

```
  specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.
```

We will accept contributions submitted under the New BSD License or the compatible MIT/X11 License. Note: GNU licenses are not compatible with the BSD license.