МультиОберон-1

Дагаев Дмитрий Викторович,
Главный Эксперт,
АО «Русатом Автоматизированные системы управления»
Консультант проекта Информатика-21

Вышла версия 1.0

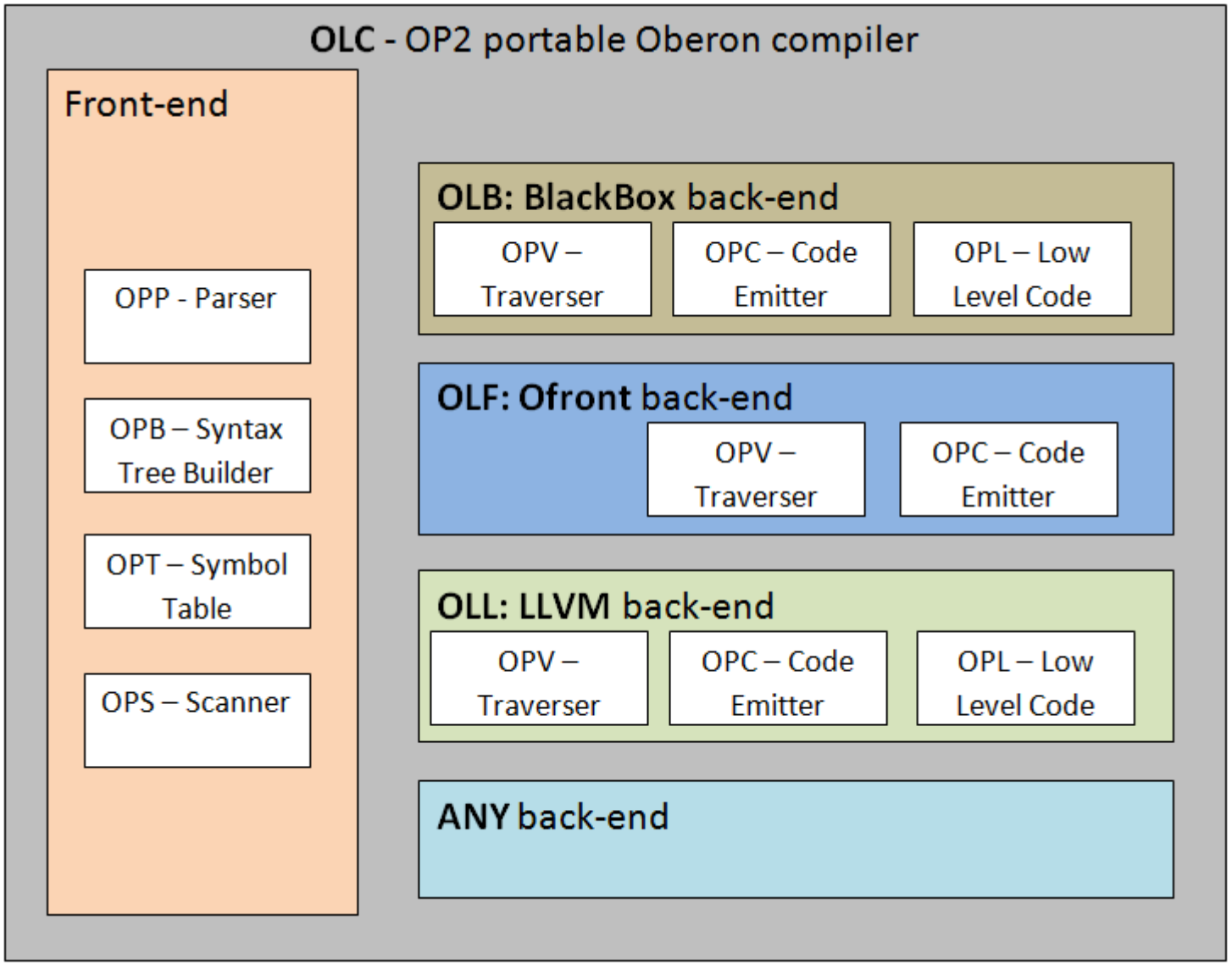
Спасибо всем, кто принимал участие в обсуждении и высказывал критические замечания!

МультиОберон по концепции

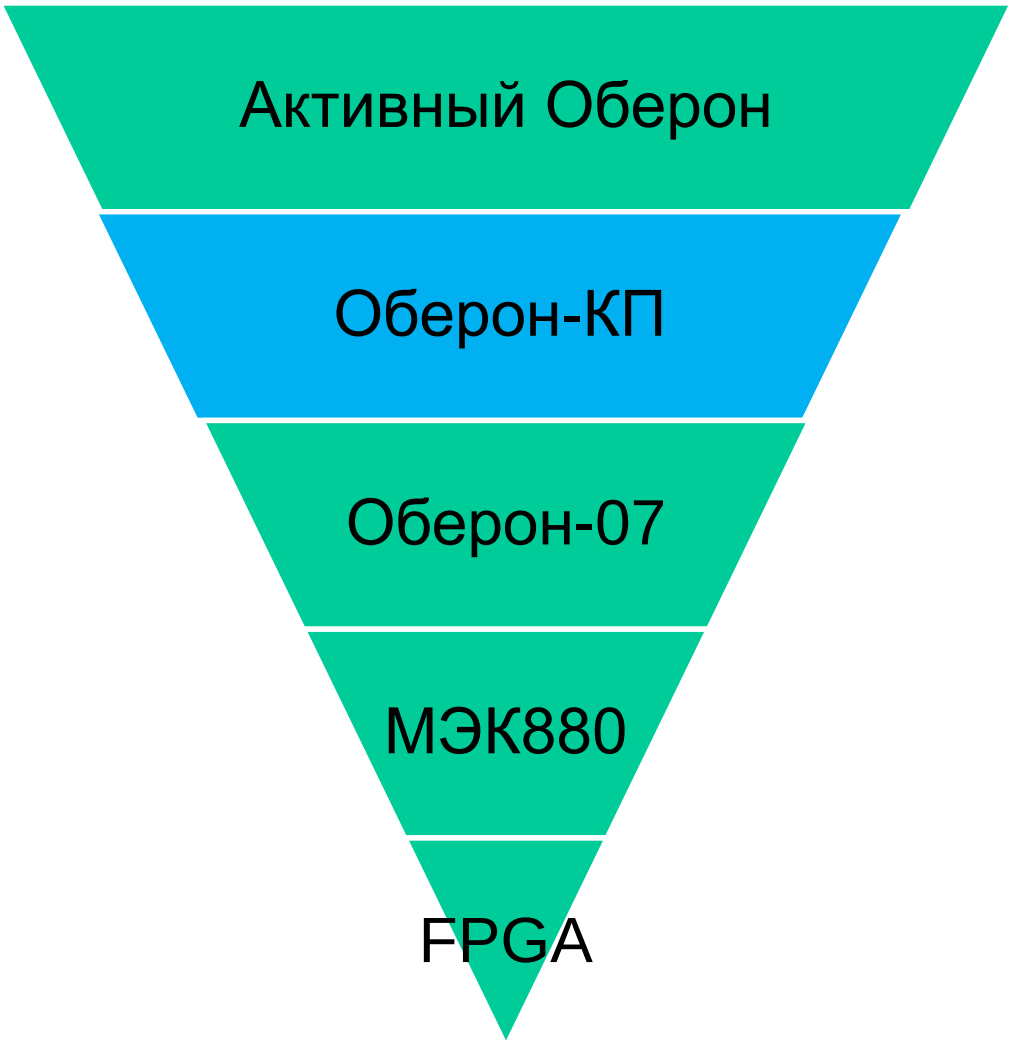
Исправление (ректификация) имен (чжэнмин, кит. 正名 *zhèngmíng)*, МультиОберон вместо CP.

OP2 компилятор (Regis Crelier) со сменными бэкендами:

- BlackBox back-end;
- Ofront back-end;
- LLVM back-end;
- Расширения в виде back-end.



Двустороннее масштабирование



Очень существенным представляется демасштабирование, при котором при сокращении объема функциональности увеличиваются показатели надежности и защищенности используемых решений.

Ограничение ключевых слов

МЭК880 -9 удалить, *4 изменить, Oberon07 -5 удалить *3 изменить,

Oberon-КП эталонный набор слов, Active Oberon +4 добавить.

ABSTRACT	EXTENSIBLE	POINTER
ARRAY	FOR	PROCEDURE
BEGIN	IF	RECORD
BY	IMPORT	REPEAT
CASE	IN	RETURN
CLOSE	IS	THEN
CONST	LIMITED	TO
DIV	LOOP	TYPE
DO	MOD	UNTIL
ELSE	MODULE	VAR
ELSIF	NEW	WHILE
USE	NIL	WITH
EMPTY	OF	
END	OR	
EXIT	OUT	

ABSTRACT	EXTENSIBLE	POINTER
ARRAY	FOR	PROCEDURE
BEGIN	IF	RECORD
BY	IMPORT	REPEAT
CASE	IN	RETURN
CLOSE	IS	THEN
CONST	LIMITED	TO
DIV	LOOP	TYPE
DO	MOD	UNTIL
ELSE	MODULE	VAR
ELSIF	NEW	WHILE
USE	NIL	WITH
EMPTY	OF	
END	OR	
EXIT	OUT	

ABSTRACT	EXTENSIBLE	POINTER
ARRAY	FOR	PROCEDURE
BEGIN	IF	RECORD
BY	IMPORT	REPEAT
CASE	IN	RETURN
CLOSE	IS	THEN
CONST	LIMITED	TO
DIV	LOOP	TYPE
DO	MOD	UNTIL
ELSE	MODULE	VAR
ELSIF	NEW	WHILE
USE	NIL	WITH
EMPTY	OF	
END	OR	
EXIT	OUT	

ABSTRACT	EXTENSIBLE	POINTER
ARRAY	FOR	PROCEDURE
BEGIN	IF	RECORD
BY	IMPORT	REPEAT
CASE	IN	RETURN
CLOSE	IS	THEN
CONST	LIMITED	TO
DIV	LOOP	TYPE
DO	MOD	UNTIL
ELSE	MODULE	VAR
ELSIF	NEW	WHILE
USE	NIL	WITH
EMPTY	OF	AWAIT
END	OR	ACTIVITY
EXIT	OUT	[safe]
		[exclusive]

Требуется какая-то декомпозиция сложности проекта. Древовидная структура проекта, как я думаю, помогла бы

И.А.Денисов

Списки импорта и дерево проекта

Дерево можно рекурсивно распечатать для каждого модуля (OmtestHelloWorld):

```
c:\suok5\dsu>Binwe\ombc trav OmtestHelloWorld
OmtestHelloWorld
  Runner
    SYSTEM
    ?c:\suok5\dsu/System/Sbwe/SYSTEM.ouf     (? - нет списка импорта)
    Kernel
    ?c:\suok5\dsu/System/Sbwe/Kernel.ouf
    Ostrings                                 (есть список импорта System/Code/Ostrings.ouf)
      Strings
      ?c:\suok5\dsu/System/Sbwe/Strings.ouf
      Math
      ?c:\suok5\dsu/System/Sbwe/Math.ouf
    -OStrings
    OLog
      Log
      ?c:\suok5\dsu/System/Sbwe/Log.ouf
    -OLog
    Api
    -Api [KERNEL32.dll]
  -Runner
-OmtestHelloWorld
```

Управление сборкой

При сборке BlackBox необходимо указывать список всех импортируемых модулей.

Опция –r означает рекурсивный пробег по спискам модулей:

```
Bfwe\omfsh co -odc OmtestHelloWorld

Bfwe\omfsh build -r OmtestHelloWorld

Bfwe\omfsh link -r OmtestHelloWorld
```

Команда build не нужна для BlackBox, т.к. .ocf объектные файлы уже созданы компилятором.

Платформо-зависимые модули

Инсталляция платформо-зависимых модулей:

```
Binwe\omfc co -64 -h -odc Api_fwr
  Math_fw OStrings_fw OLog_fw
  Kernel_fwr17 Runner_fwr17 Files_17
  Times Testing HostApi_fwr
  HostConLog_fw HostTimes_fwr
  HostFiles_fwr17
```

Расширения имен файлов модулей означают платформо-зависимость. Например, Kernel_fwr17 означает Kernel версии от 1.7 для Omf, Windows, X64. Он компилится вместо Code, Sym в Cfwr, Sfwr.

Исполнение биткода – это у Вас времянка или нет? Зачем Вы используете JIT для динамической загрузки?

А.Е.Недоря

Варианты динамической загрузки

	*.ocf	*.o COFF Windows	*.o ELF Linux	*.bc JIT
Omb BlackBox	+	-	-	-
Omf OFront	-	+	+	-
Oml LLVM	-	+	+	+

Добавлена динамическая загрузка для Oml и Omf объектных файлов форматов COFF и ELF с привязкой всех имен загружаемого модуля.

Используются базовый модуль Baseloader_17 и загрузчики OmcOcfLoader OmcObjLoader_Coff OmcObjLoader_Elf и OmlBcLoader для прогона биткода.

Загрузчики выполнены для архитектур X86, X64.

Динамически загружаемые модули опознаются и поддерживаются ядром рантайма Kernel.

Решения редактора связей

Необходимо динамическое связывание позиций релокации с именами в загруженных модулях. Реализовано для архитектур X86 и X64.

Disassembly Raw Hex (zero bytes in bold): **488B0D00000000**

String Literal: **"\x48\x8B\x0D\x00\x00\x00\x00"**

Array Literal: **{ 0x48, 0x8B, 0x0D, 0x00, 0x00, 0x00, 0x00 }**

Disassembly: **0: 48 8b 0d 00 00 00 00 mov rcx,QWORD PTR [rip+0x0] # 0x7**

Необходима привязка к 64 битным адресам для X64. Данный код (например, для X64) дописывается в виде байтов в конец каждого загружаемого модуля для каждой вызываемой функции, расположенной вне интервала 32-битной адресации.

```
push    LL LL LL LL       # low adr 4 bytes
mov DWORD PTR [rsp+0x4],  HH HH HH HH     # push high adr 4 bytes
ret                       # call LL LL LL LL HH HH HH HH
ret                       # ret
```

Времена прогона тестового модуля

	32 bit	64 bit	32 JIT	64 JIT
Omb BlackBox	0.19	-	-	-
Omf OFront	0.21	0.33	-	-
Oml LLVM	0.21	0.25	23.0	1.9

Результаты для Windows 7 Pro 64 bit.

Времена загрузки через JIT существенно выше, чем загрузка объектных файлов.

Разница между загрузкой объектников LLVM и загрузкой ocf BlackBox минимальна.

Omf Ofront должен загружать еще и SYSTEM.dll

* Работа с JIT требует подключения динамической библиотеки LLVMT размером от 31 Мб.

О вариантах загружаемых программных модулей

Размер HelloWorld, КБ для Windows 7 Pro 64 bit.

	Исходный	Промежуточн	Объектный	Исполняемый
Omb	1.636	-	0.466	128.5
Omf32	1.636	1.918 (.c)	4.507	700.9
Omf64	1.636	1.918 (.c)	6.126	880.1
Oml32	1.636	3.464 (.bc)	2.555	182.8
Oml64	1.636	3.472 (.bc)	3.268	259.6

Как известно, модульность Go реализуется только через исполняемые exe-файлы, минимальный из которых HelloWorld занимает 2 МБ.

Загрузка объектных (и .ocf) файлов существенно экономичнее.

Создание исполняемых и dll/so файлов требует разрешения всех зависимостей.

Но пока Oml не быстрее Omb. А мне кажется, некоторые ожидают, что LLVM автоматически даст прирост производительности.

И.А.Дегтяренко (Trurl)

Методика измерения временных характеристик (benchmarking)

Вызов оцениваемой процедуры:

```
PROCEDURE IsPalindrome* (IN str: ARRAY
OF CHAR): BOOLEAN;

BEGIN …

END IsPalindrome;
```

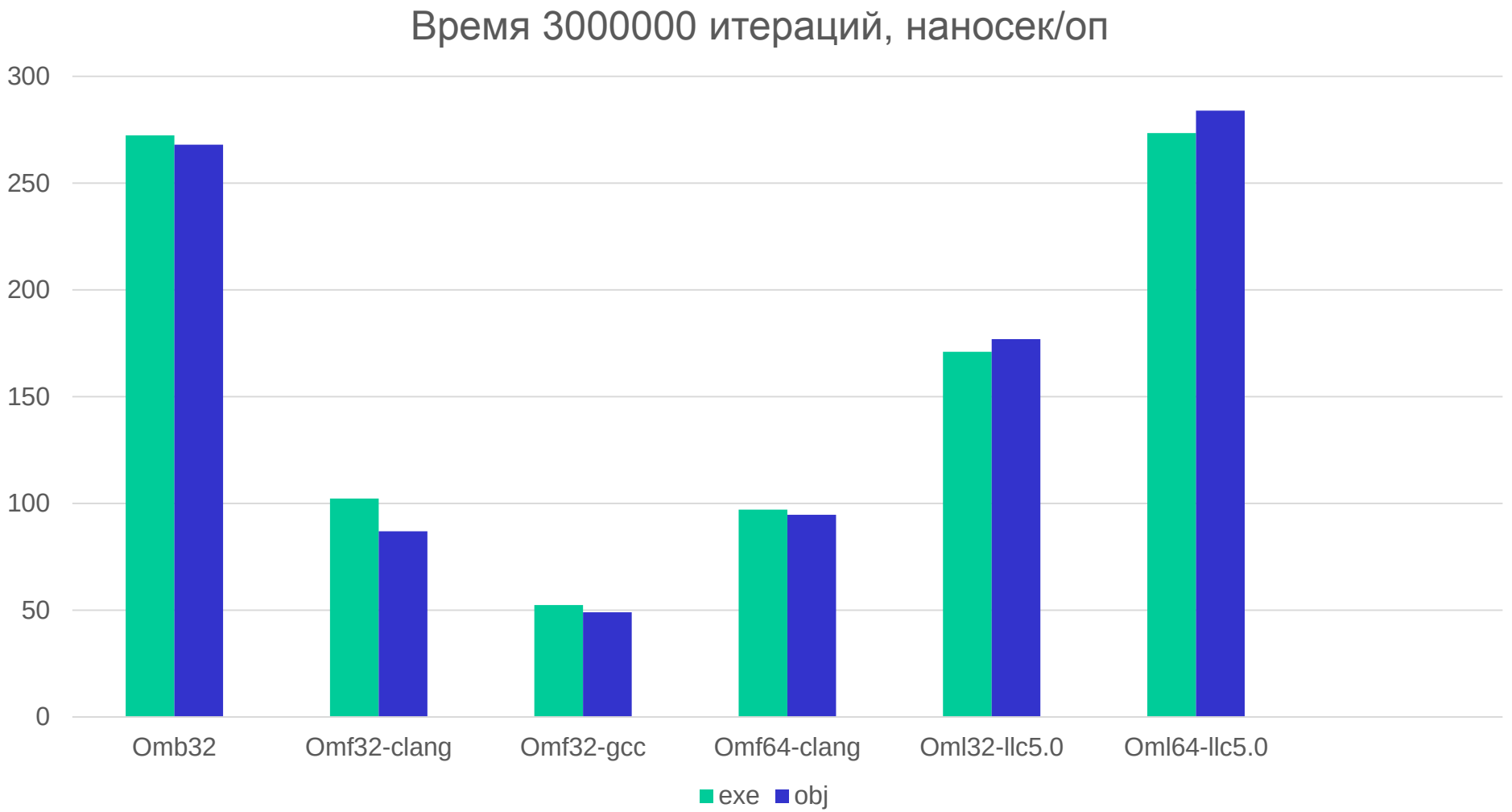
Вызов теста с bench.num повторов:

```
PROCEDURE BenchmarkPalindrome* (IN bench:
  T.Bench; VAR num_done: INTEGER);
```

Вызов для каждого варианта num повторов:

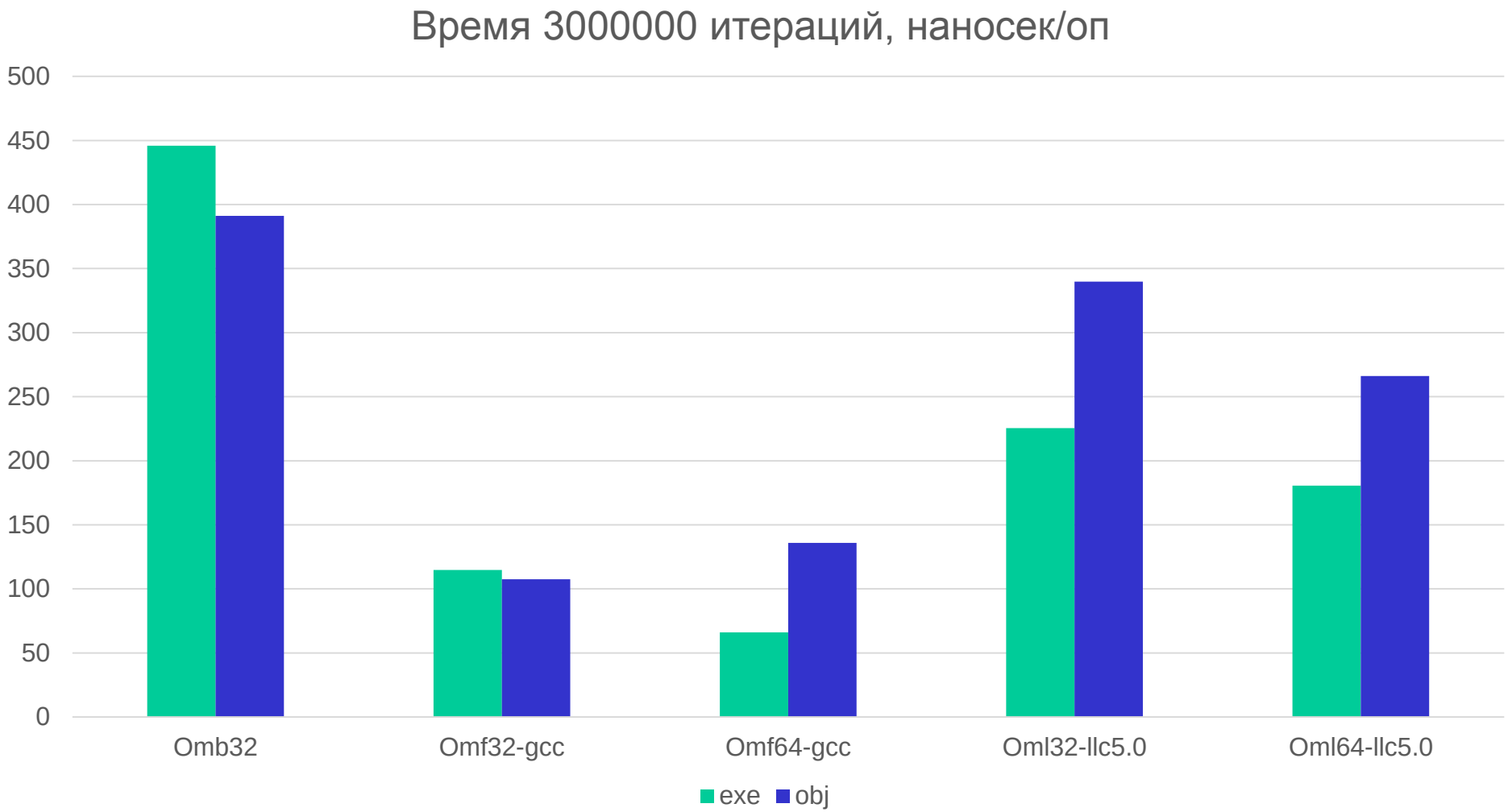
```
Omtest\C*\OmtestBenchSet.exe -num 3000000
```

Замеры производительности для Windows 7 Pro 64 bit



Оптимизация для Omf - снижение до 32% (gcc 18%), оптимизация Oml32 - до 63%, Oml64 не работает для LLVM5.0 Windows

Замеры производительности для Linux 4.9.0 x86_64



Оптимизация для Omf дает снижение до 15%, оптимизация Oml снижает до 41%, модульность *.o ухудшает производительность.

Замеры времени само-компиляции

Данную метрику использовал Вирт для оценки компиляторов. Для каждого компилятора реализована средствами самого компилятора.

Сравнение для Windows 7.

	Omb	Omf32	Omf64	Oml32	Oml64
Windows	11.6	53.9	53.0	575.2	20.3

Сравнение для Linux 4.9.0 x86_64 (другие, более мощные технические средства).

	Omb	Omf32	Omf64	Oml32	Oml64
Linux	0.5	15.3	14.7	10.2	9.4

Выводы и направления улучшения производительности

Возможность улучшения производительности в 6 раз.

Изменение алгоритма обработки ошибок с учетом FP и без.

Переход от LLVM 5.0 к более поздним версиям (сопровождается изменением интерфейса) для Oml.

Предпочтительное использование gcc.

Оптимизация базовых функций работы (строки), определенные в рантайме.

(Опубликоавать тесты) не только можно, а даже нужно.

О.Н.Чередниченко (Zorko)

Тесты компилятора и тесты модулей

Вызов теста:

```
Binwe\ombc test OmtestOmcSystemTest
```

Приводит к вызову тестера:

```
OmcTester.Run(st, "OmtestOmcSystemTest")
```

Для каждой функции tf=Test* вызывается:

```
tf(rec); где   rec: Testing.Rec;
```

Результаты выводятся в лог:

```
[OmtestOmcSystemTest.Test1Addr.028] ?LONG li_res= 12
li_req= 16 :Proper position of ptr after long

[OmtestOmcSystemTest.Test1Addr.029] ?LONG li_res= 16
li_req= 24 :SIZE of Rec with LONGINT aligned by 8

[ALL] ========== Total 46 tests, 2 bad, result=
95.65%
```

Тесты и вопросы выравнивания

Непрошедший тест на выравнивание LONGINT в структурах (Omb выравнивание = 4) может привести к отказу сборщика мусора:

```
SysPtrAlign8 = RECORD
   b: BYTE;
   li: LONGINT;
   p: SYSTEM.PTR;  (* 12 или 16??? *)
END;
```

Для кроссплатформ-Omf потребовалось менять алгоритм заполнения:

```
struct OmtestOmcSystemTest_SysPtrAlign8 {
    _BYTE b;
    char _prvt1[7];   (* выравнивание на 8 *)
    LONGINT li;
    SYSTEM_PTR p;
    char _prvt0[4];    (* выравнивание размера структур *)
} OmtestOmcSystemTest_SysPtrAlign8;
```

Трассировка ошибок

….makes implementation of Kernel.InstallExcp and Kernel.RemoveExcp impossible, at least using SEH.

X512

Обработчик Omb – список по FP

BlackBox, получив состояние указателя фрейма - регистра FP, может отследить стек вызываемых процедур. Доступ к регистрам!

```
*** trap # 23
F Addr=28F34C 28F368
 Addr=28F350 4D03D5 proc   2
 Addr=28F354 0
 Addr=28F358 2
 Addr=28F35C -1
 Addr=28F360 42C132
 Addr=28F364 28F3DE
F Addr=28F368 28F480
 Addr=28F36C 4D0508 proc   3        1

F Addr=28F480 0
 (pc=4D0278, sp=28F34C, fp=28F34C) in .4D0278OmtestMkTraps.Halt.4D03D5
<OmtestMkTraps.RunOpt.4D0508<OmtestMkTraps.MAIN
Register End OmtestMkTraps Address= 4C0000 opts= {0, 2..6, 8, 16, 17}
Module OmtestMkTraps 4C0000 loading resLoad= 0 resString='' importingNames='' importedNames=''
objectName=''
```

Каждый адрес FP содержит адрес следующего FP (буква F синего цвета), а последующая ячейка содержит адрес вызываемой процедуры.

Первый адрес (1) – это значение регистра PC, содержит ссылку на функцию OmtestMkTraps.Halt;

Второй адрес (2 красным) – лежит в последующей ячейки и содержит ссылку на функцию OmtestMkTraps.RunOpt.

Обработчик OFront – макросы DLINK

Макросы Ofront'а выстраивают динамически цепочку вызова:

```
#define __ENTER(name)   SYSTEM_DLINK __dl = {SYSTEM_dlink,
   name}; SYSTEM_dlink = &__dl

#define __EXIT          SYSTEM_dlink = __dl.next
```

Данная цепочка обновляется при вызове процедуры:

```
__ENTER("OmtestOmcSystemTest.Test0Conv");

…

__EXIT;
```

Использование данного механизма ухудшает производительность (приводит к дополнительным операциям при вызове процедуры) и может динамически отключаться опцией –dl для бэкендов Omf, Oml.

Обработчик содержимого стека

Причины появления:

1. Оптимизирующие компиляторы стремятся исключить «излишние» операции, такие, как присваивание указателя фрейма FP. При высоких уровнях оптимизации FP уже нет.
2. Нет общего кроссплатформенного решения получения состояния регистров даже для Ofront'a. В модели LLVM конкретика по регистрам не определяется.
3. Макросы DLINK ухудшают производительность системы.
4. Современные отладчики черпают всю информацию из содержимого стека.

Предложенное решение – подключаемый обработчик при задании:

```
Runner.SetHandlerType(Runner.HANDLER_TYPE_STACK);
```

Недостаток – неочевидный алгоритм извлечения информации из стека. Например, адрес возврата процедуры или адрес процедуры из предыдущего состояния стека, не обнуленного при следующем вызове (никто не обещал, что стек будет обнуляться).

Интересно посмотреть на результат выполнения Div(-2147483647 - 1, -2) в МультиОбероне с разными настройками

Константин (Comdiv)

Реализация теста с переполнениями

	INTEGER Windows	LONGINT Windows	INTEGER Linux	LONGINT Linux
Omb32	+	+	+	+
Omf32	+	-461168601842738790 4611686018427387904	+	+
Omf64	+	-461168601842738790 4611686018427387904	+	+
Oml32	+	+	+	+
Oml64	+	+	+	+

Реализация тестов:

- Тесты для 4 байт OmtestOmcSimpleTest.Test8Overflows.000 – INTEGER;
- Тесты для 8 байт OmtestOmcSimpleTest.Test8Overflows.001 – LONGINT.

Для INTEGER:

Div(-2147483647 - 1, -2)

Для LONGINT:

Div(- 7FFFFFFFFFFFFFFFL - 1, -2)

Можно ли повлиять на компилятор gcc для Windows …

Некоторые мои решения могут быть взяты в МультиОберон практически один-в-один

О.Н.Чередниченко (Zorko)

Интеграция выработанного лабораторией ADRINT

Концепция RESTRICT определяет механизм расширения МультиОберона. Добавляем ADRINT – целое размером адреса. Модуль профиля RestrictAdrint экспортирует +ADRINT во все загружающие его модули (экспорт – ADRINT*).

```
MODULE RestrictAdrint;

   RESTRICT +ADRINT*;

END RestrictAdrint.
```

В тесте OmtestOmcRestrictTest импортируем профиль RestrictAdrint или указываем явно, как +LARGEINT:

```
IMPORT SYSTEM, RestrictAdrint;

RESTRICT +LARGEINT;

..

ADRINT ai;

ai := SYSTEM.VAR(rec);
```

Использование ADRINT вместо INTEGER в BlackBox Meta

Для компиляции Meta в 64бит или использовать ADRINT, или делать 2 версии INTEGER и LONGINT.

```
PROCEDURE TypOf (struct: Kernel.Type): ADRINT;
BEGIN
 IF SYSTEM.VAL(ADRINT, struct) DIV 256 = 0 THEN
    RETURN SYSTEM.VAL(ADRINT, struct)
 ELSE
    RETURN 16 + struct.id MOD 4
 END
END TypOf;
```

В МультиОбероне SYSTEM.ADR() возвращает целое INTEGER для 32 бит и LONGINT для 64, что по сути является типом ADRINT.

Вопросы по докладу …

Дагаев Дмитрий Викторович,
Главный Эксперт,
АО «Русатом Автоматизированные системы управления»
Консультант проекта Информатика-21
forum.oberoncore.ru
www.inr.ac.ru/~info21/

dvdagaev@oberon.org