В тестировании были задействованы несколько машин, на одной из которых также проводились измерения в рамках работы [?]. Характеристики общей машины для курсовой работы 2009 года и данной приведены в таблице 0.1. Системные характеристики остальных тестовых машин приведены в таблицах 0.2, 0.3.

Таблица 0.1: Системные характеристики: ноутбук Pentium MMX

CPU	Intel® Pentium® Processor with MMX™ Technology 233 MHz, 66 MHz FSB, L2 Cache 512 KB
RAM	64 Mb DRAM
OS	Windows XP x86

Таблица 0.2: Системные характеристики тестовой машины №1 для тестирования

CPU	Intel i5 mobile, 2.4 GHz (up to 2.7GHz) 3 MB Smart Cache (L1 — 128KB, L2 — 512KB), 2/4 cores/threads
RAM	64 Mb DRAM
OS	Windows 10 x86-64

Таблица 0.3: Системные характеристики тестовой машины №2 для тестирования

CPU	Intel i5 3570k, 4.4 GHz 6 MB Smart Cache (L1 — 4x32KB, L2 — 4x256KB), 4/4 cores/threads
RAM	2x8GB, 1866 MHz, 9-9-9-27, CL9
OS	OpenSuse 13.2, x86-64 Linux kernel — 3.16.7-29-desktop

Тестирование проводилось на примере самоприменения компилятора Простого Рефала. Рассматривались режимы прямой компиляции исходных текстов и непосредственной генерации и выполнения интерпретируемого кода. Замеры происходили в виде нескольких выборок для каждой конфигурации тестовой машины и используемого компилятора. Исходный текст скриптов, выполняющих измерения приведён в приложении 3 на с. ??.

Рассмотрим результаты тестирования для машины №1. В качестве исходных данных использовались исходные тексты компилятора, по пять прогонов для каждого файла. Также, сделан девятикратный прогон каждого компилятора C++ с файлами в папке bootstrap. Таким образом, можно определить, на сколько быстрее целевой компилятор обрабатывает инициализацию массива кодов операций языка сборки по сравнению с построением в режиме компиляции. Используемые компиляторы:

- Visual C++ Express 2013, исходные тексты транслировались как на x86 и x86-64 архитектуры;
- Borland C++ Compiler 5.5;

- MinGW builds 4.8.1 rev 5
  - x86 с использованием потоков POSIX, механизм исключений — DWARF;
  - x86-64 с использованием потоков POSIX, механизм исключений — SEH;
- Open Watcom 1.9;
- Clang 3.7.0, исходные тексты транслировались как на x86 и x86-64 архитектуры, библиотеки в сборке MinGW builds 4.8.1 rev 5.

Результаты измерений приведены в таблице #. Данные приводятся в формате: общее время выполнения одного прохода, время сопоставления, время построения результата, время выполнения внешних функций, время выполнения рефал-функций. Размерность — секунды. Виды тестирования, приведённые в соответствующей таблице:

1. Среднее время самоприменения в режиме компиляции;
2. Среднее время самоприменения в режиме интерпретации;
3. Среднее время проверки целевого компилятора в режиме компиляции;
4. Среднее время проверки целевого компилятора в режиме интерпретации.

Таблица 0.4: Результаты тестовой машины №1

	Тестирование	
Конфигурация	*1	*2
Visual C++ 2013 x86	21.344/6.728/6.855/13.583/7.761	26.576/8.806/10.25/19.056/7.52
Visual C++ 2013 x86-64	21.032/6.88/7.045/13.925/7.107	25.534/8.156/10.321/18.478/7.056
Borland C++ 5.5	16.618/5.082/6.915/11.996/4.622	18.505/5.929/8.181/14.111/4.394
MinGW x86, DWARF	20.44/6.618/6.995/13.613/6.827	25.976/8.817/10.222/19.039/6.937
MinGW x86-64, SEH	20.648/6.579/7.54/14.119/6.529	26.45/8.779/11.086/19.865/6.585
Open Watcom	55.16/19.577/27.274/46.851/8.308	58.413/20.856/29.588/50.444/7.969
Clang 3.7.0 x86	23.832/7.941/8.841/16.781/7.05	29.357/10.275/11.903/22.178/7.179
Clang 3.7.0 x86-64	20.06/6.816/6.84/13.656/6.404	25.437/8.495/10.439/18.934/6.504

	Тестирование	
Конфигурация	*3	*4
Visual C++ 2013 x86	1.592/0.0/0.0/0.0/1.601	1.468/0.0/0.0/0.0/1.468
Visual C++ 2013 x86-64	0.821/0.0/0.0/0.0/0.819	1.565/0.0/0.0/0.0/1.565
Borland C++ 5.5	0.821/0.0/0.0/0.0/0.819	0.536/0.0/0.0/0.0/0.533
MinGW x86, DWARF	13.699/0.0/0.0/0.0/13.696	4.746/0.0/0.0/0.0/4.742
MinGW x86-64, SEH	13.472/0.0/0.0/0.0/13.47	4.608/0.0/0.0/0.0/4.602
Open Watcom	13.54/0.0/0.0/0.0/13.531	1.998/0.0/0.0/0.0/1.993
Clang 3.7.0 x86	11.471/0.0/0.0/0.0/11.471	5.619/0.0/0.0/0.0/5.617
Clang 3.7.0 x86-64	11.727/0.0/0.0/0.0/11.727	5.551/0.0/0.0/0.0/5.549

Таблица 0.5: Результаты ноутбука Pentium MMX

	Тестирование	
Конфигурация	*1	*2
MinGW x86	319.175/118.377/102.719/221.096/98.079	370.841/145.55/132.023/277.573/93.268
Borland C++ 5.5	235.086/79.515/86.176/165.691/69.395	258.958/95.062/101.382/196.444/62.514

	Тестирование	
Конфигурация	*3	*4
MinGW x86	320.637/0.0/0.0/0.0/320.577	96.859/0.0/0.0/0.0/96.795
Borland C++ 5.5	26.083/0.0/0.0/0.0/26.045	13.672/0.0/0.0/0.0/13.628

Таблица 0.6: Результаты тестовой машины №2

	Тестирование	
Конфигурация	*1	*2
GCC 4.8.3 x86	16.759/4.975/6.208/12.844/5.172	16.253/5.457/6.851/13.024/5.792
GCC 4.8.3 x86-64	16.021/4.752/5.999/12.207/4.801	16.615/4.923/6.274/12.731/5.012
Clang 3.5.0 x86	18.216/5.562/7.115/13.917/5.828	19.433/5.816/7.952/14.175/6.108
Clang 3.5.0 x86-64	18.983/5.807/7.632/14.510/6.129	19.794/6.649/8.577/14.601/6.892

	Тестирование	
Конфигурация	*3	*4
GCC 4.8.3 x86	9.272/0.0/0.0/0.0/9.320	3.111/0.0/0.0/0.0/3.209
GCC 4.8.3 x86-64	8.892/0.0/0.0/0.0/9.012	2.982/0.0/0.0/0.0/2.997
Clang 3.5.0 x86	10.781/0.0/0.0/0.0/10.892	3.687/0.0/0.0/0.0/3.619
Clang 3.5.0 x86-64	10.976/0.0/0.0/0.0/11.027	3.814/0.0/0.0/0.0/3.756

Необходимо отметить, что в силу ограниченной вычислительной мощности тесты на ноутбуке Pentium MMX проводились с меньшим числом выборки (а именно, 3 выборки для самоприменения и 5 выборок для тестирования целевых компиляторов). Также, в связи с архитектурой процессора (80486) были доступны лишь компиляторы MinGW x86 и Borland C++ 5.5.

На тестовой машине №2 проводилось тестирование на операционной системе Gnu Linux Opensuse 13.2 с выбором в качестве целевых компиляторов GCC 4.8.3 (g++ (SUSE Linux) 4.8.3 20140627 [gcc-4_8-branch revision 212064]) и clang version 3.5.0 (tags/RELEASE_350/final 216961). В качестве целевых платформ использовались платформы x86 и x86-64.

Как видно из результатов тестирования, сравнивать их достаточно сложно, так как не малое влияние оказывает время компиляции целевым компилятором и его выбор. По результатам анализа – самым быстрым компилятором оказался BCC 5.5, самым медленным — Watcom (на ос семейства Windows). В среде Gnu Linux разница между GCC и Clang не столь значительна, как в среде Windows. Среднее время на интерпретацию возрастало в пределах 7-29%.

Аналогичная ситуация произошла и при оценке размера получаемого исполнительного файла. Это вполне объясняется тем, что каждый компилятор имеет разные алгоритмы оптимизации и генерации кода. Если усреднить полученные значения и сравнивать их в рамках фиксированной конфигурации целевого компилятора и платформы, то разница между размером исполнимых файлов для режима интерпретации языка сборки от режима прямой компиляции составляет от 2 до 3 раз.