Московский физико-технический институт
Физтех-школа радиотехники и компьютерных технологий
Кафедра информатики и вычислительной техники

Запуск тестовых программ на языке C на RTL-модели ядра архитектуры Эльбрус

Подготовил: Федоров М.Д. Б01-903

Научный руководитель: Фролов П.В.

Москва, 2023

Введение

- Существуют генераторы синтетических программ на языке C и C++:

  rtg (разработан в МЦСТ)

  yarpgen (https://github.com/intel/yarpgen)

  csmith (https://github.com/csmith-project/csmith)

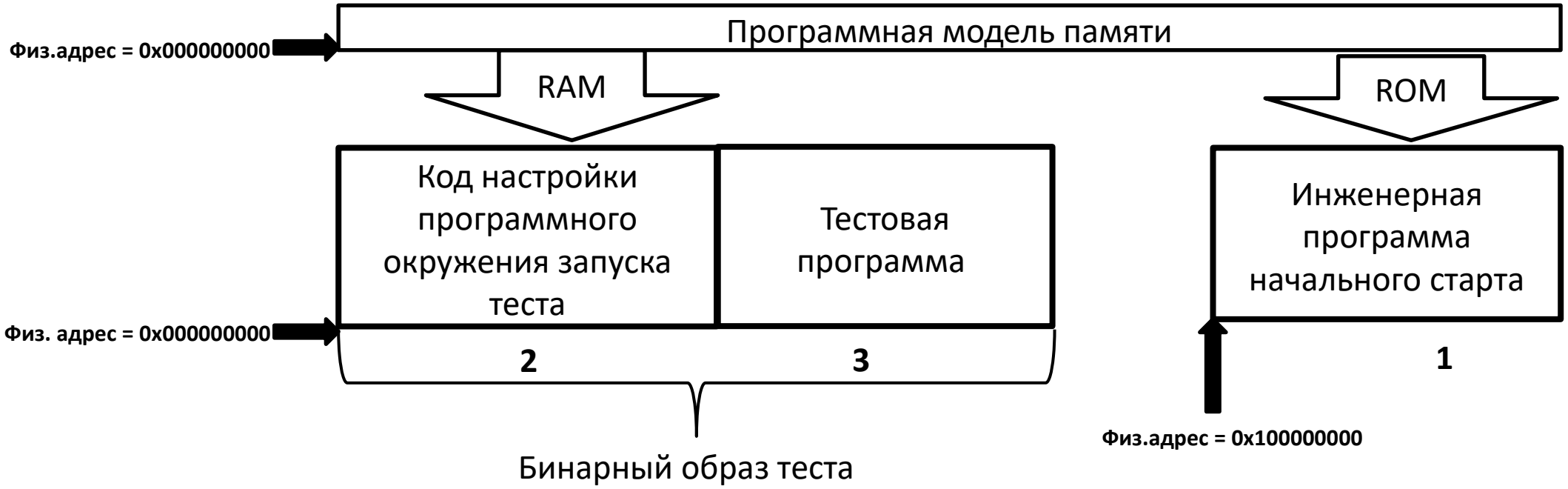
- Необходимо обеспечить использование генераторов для верификации RTL-модели микропроцессорного ядра семейства «Эльбрус».

- Для логической верификации RTL-модели ядра синтетические программы на языке C исполняют под управлением ОС на прототипах на основе ПЛИС.
- Некорректное завершение программы может означать наличие ошибки в RTL-описании
- Локализацию найденных в RTL-описании логических ошибок, их исправление и отладку удобно осуществлять при программном моделировании RTL-модели микропроцессора.
- Запуск ОС на прототипе может быть невозможен на ранних этапах разработки из-за логических ошибок в RTL.
- При программном моделировании необходим запуск без ОС.
- Ручная адаптация тестовых программ для исполнения без ОС является трудоёмкой.
- Нужен инструмент для запуска тестовых программ на языке C на RTL-модели верифицируемого микропроцессора.

Тестовое окружение для RTL-модели ядра

Тестовый стенд

- Перед запуском моделирования бинарные образы теста и инженерной программы начального старта размещаются в модели памяти.



Цель работы

- Разработать инструмент для запуска программ на языке С в тестовом окружении RTL-модели ядра архитектуры «Эльбрус» и применить для запуска синтетических тестовых программ.

Задачи

- Настройка программного окружения запуска тестовой программы.
- Определение успешности завершения тестовой программы.
- Формирование бинарного образа теста.
- Стандартизация программного интерфейса запуска сторонних генераторов тестовых программ на языке С.

Требования

- Параметризация опций компиляции, компилятора.
- Возможность запуска тестовых программ, собранных в 32-х и 64-х разрядных режима.
- Классификация кодов ошибок запуска теста.

Настройка программного окружения запуска тестовой программы

- Тестовая программа – это пользовательское приложение, предназначенное для запуска из-под ОС.
- При пользовательском запуске настройку программного окружения запуска осуществляет операционная система.
- Без операционной системы необходимо самому обеспечить настройку программное окружение.

Программное окружение запуска

- Системные регистры(стековые дескрипторы, регистр памяти глобальных данных, регистры регулирующие вещественные операции) – нужны для корректного указания граничных адресов и размеров стеков и памяти глобальных данных, правильного исполнение вычислительных инструкций.
- Таблица трансляции – необходима для того, чтобы тест работал в своем адресном пространстве.
- Таблица входов операционной системы – некоторые генераторы содержат системные вызовы. Во время моделирования тестовой ситуации могут возникать аппаратные прерывания. Системный вызов или прерывание обрабатываются в коде таблицы входов ОС.

Настройка программного окружения запуска тестовой программы

Настройка системных регистров

Дескрипторы фрагментов памяти

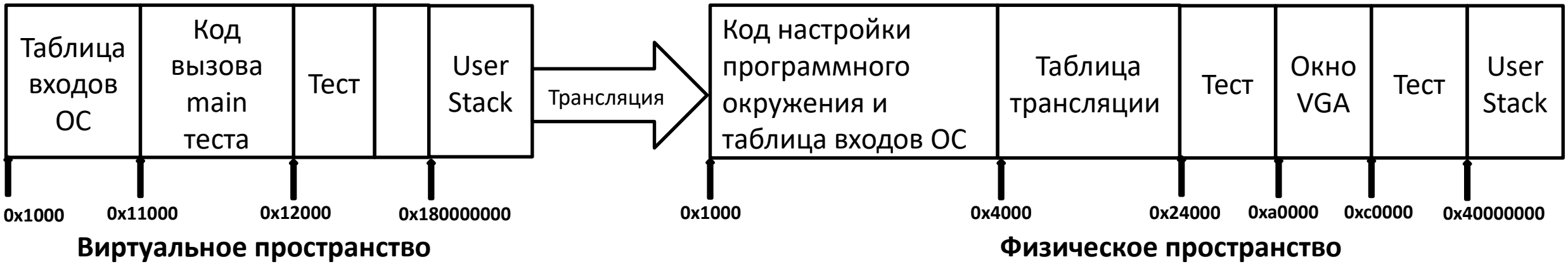
- Для пользовательского стека в регистре USD(User Stack Descriptor) выставляется размер стека либо дефолтный, либо переданный в качестве параметра при сборке. Граничные адреса прописываются достаточно большими для того, чтобы они не пересекались с адресами, используемыми в тесте. Аналогичные действия делаются для стека пользователя в регистре PSP(Procedure Stack Pointer).
- Таким же через дескриптор памяти глобальных данных(GD) для соответствующего фрагмента ставятся размер и граничные адреса так, чтобы не было коллизий с виртуальным адресным пространством тестовой программы.

Регистры вещественных операций

- Чтобы обеспечить при моделировании без ОС, такое же выполнение вещественных инструкций, как в пользовательских приложениях, значения регистра управления вещественными флагами для упакованных операций(PFPFR – Packed Floating Point Flag Register) и регистра управления вещественными(FPCR – Floating Point Control Register) ставятся такими же, как при запуске под операционной системой.

Настройка программного окружения запуска тестовой программы
Поддержка виртуальной адресации

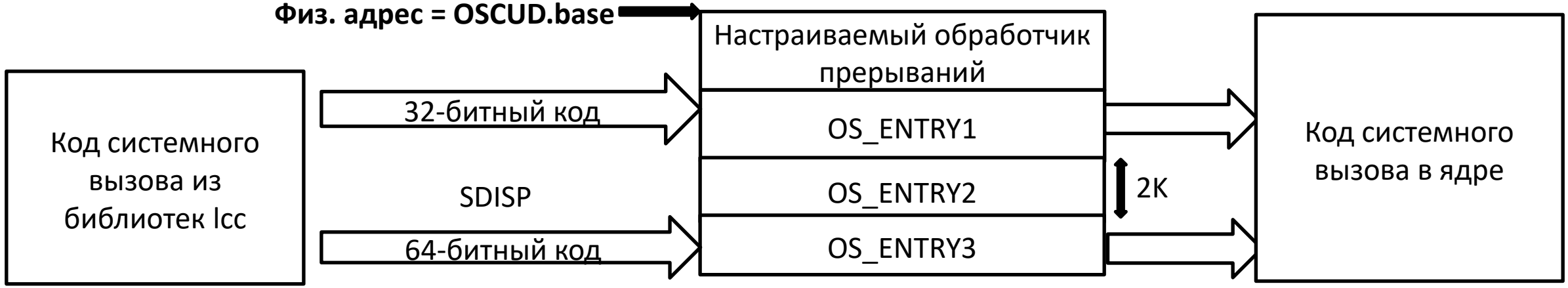
- Прежде всего через регистр режимов ядра(CORE_MODE) были включены режим совместимости с трансляции для 6-й версии системы команд и младше, а также единой таблицы страниц.
- Был выставлен флаг поддержки режима виртуальной трансляции для первичного(«Эльбрус») виртуального пространства в регистре MMU и адрес начала таблицы страниц в регистре U_PPTB.
- Инструмент формирует для стека пользователя PTE(Page Table Entry) только 2 уровня – память выделяется динамически страницами по 2 Мб, для бинарного образа теста только 3 уровня – память выделяется по 4Кб.
- Количество выделенной виртуальной памяти равно или немного больше размеров стека и образа теста.
- Для того, чтобы можно было использовать, стандартное тестовое окружение, в котором таблица входов операционной системы занимает виртуальное пространство от 0x1000 до 0x11000, виртуальный адрес теста должен начинаться с 0x12000.
- Физические адреса теста зависят не только от его размеров, но и от наличия окна видеопамяти в RTL.



Настройка программного окружения запуска тестовой программы
Обработка прерываний и системных вызовов

Прерывания

- В таблицу входов операционной системы в качестве нулевого входа вносится обработчик прерываний.
- В этом обработчике прерываний читается поле exc регистра прерываний TIR(Trap Info Register).
- После получения информации о прерывании формируется код ошибки и происходит завершение теста.
- **Системные вызовы**
- Системный вызов осуществляется при помощи инструкции SDISP. Затем происходит исполнение кода входа ОС. Там происходит разбор аргументов системного вызова и переход на код в ядре, который недоступен.
- В зависимости от битности кода в соответствующий вход ОС вносится обработчик системных вызовов в котором происходит определение номера системного вызова и передача параметров.
- В случае неподдерживаемого системного вызова происходит падение теста.



SCALL под управлением ОС

Определение успешности завершения тестовой программы

- Тест выпускается для проверки определенной ситуации.
- Необходимо быстро узнать класс причины ошибочного завершения теста.

Реализованная унификация кодов падения теста

- Код ошибки состоит из 24 битов. Старшие 4 бита [23:20] бит это тип ситуации, младшие 20 бит – диагностическая информация.

Причина падения теста	[23:20]	[19:0]
return !0	0x0	Возвращенное значение
exit(!0)	0x1	Аргумент exit()
Исключение	0x2	Номер младшего исключения в соответствии с порядком в СК
Неподдерживаемый scall	0x3	Номер системного вызова

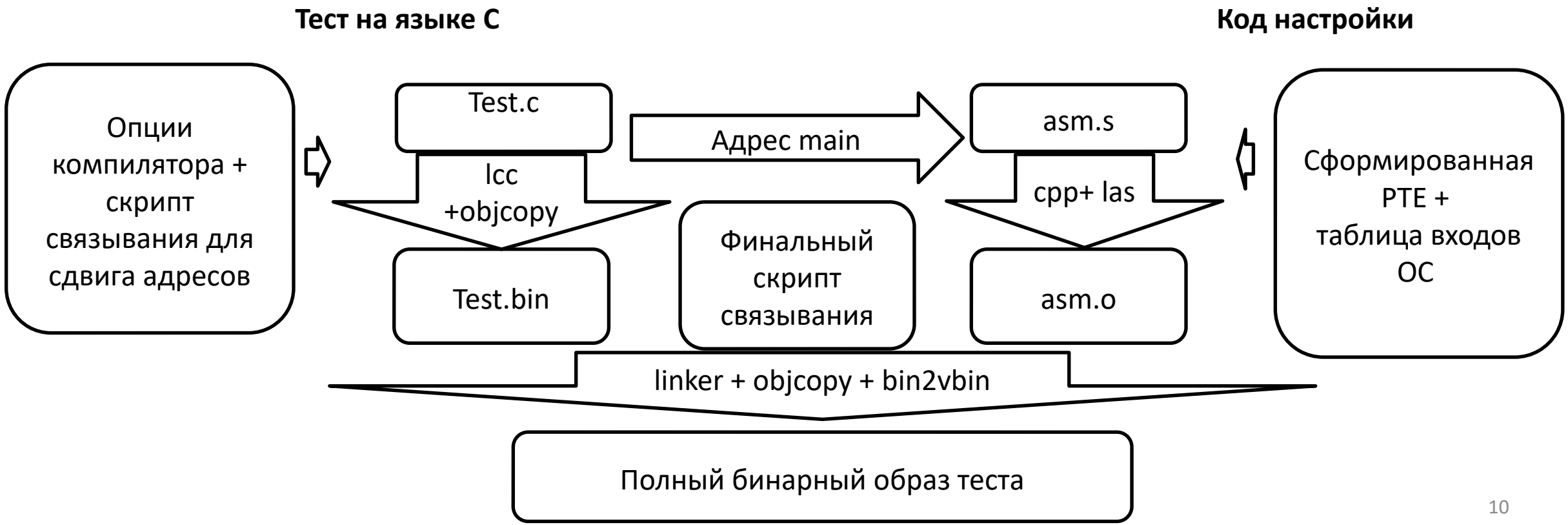
- В случае return 0 или exit(0) возвращаемое значение будет 0.

Реализованные методы обработки системного вызова exit()

- Генератор rtg, которые делают выход не через return, а через exit.
- Вызов exit обрабатывается либо через вход операционной системы, либо через замену его на аналогичный, но не системный вызов.

Формирование бинарного образа теста

- Программа настройки реализована 64 битной и не сможет быть связана по адресам с 32 битным кодом.
- Для решения данной проблемы тестовая программа при помощи директивы скрипта компоновки TARGET(binary) линкуется в виде байтов. Адрес main запоминается и подставляется в код настройки.
- При формировании полного бинарного образа теста в скриптах связывания учитываются такие факторы, как наличие окна видеопамяти и занятость в стандартном тестовом окружении виртуального адресного пространства от 0x1000 до 0x11000 под таблицу входов операционной системы.
- Присутствует возможность параметризации компилятора и опций компиляции.



Стандартизация программного интерфейса запуска сторонних генераторов

- Реализована скриптовая программа на языке python.
- На вход подается название конфигурационного файла и опции стандартного интерфейса командной строки, включающего в себя:
  - Зерно генерации.
  - Дополнительные отладочные опции.
- Разбор командной строки и конфигурационного файла происходит при помощи модуля genargparse (разработано в МЦСТ).
- Была реализована возможность получения бинарного образа теста сразу после его генерации.

Результаты работы

Разработан для запуска в тестовом окружении RTL-модели ядра архитектуры «Эльбрус» и применён для запуска синтетических тестовых программ на языке С инструмент, в котором реализованы:

- Настройка программного окружения запуска тестовой программы.
- Определение успешности завершения тестовой программы.
  - Реализована обработка вызова exit()
  - Классифицированы коды ошибок
- Формирование бинарного образа теста.
  - Есть возможность собирать 32-х разрядный код.
  - Доступна параметризация компилятора, размеров стека, бута.
- Стандартизация программного интерфейса запуска сторонних генераторов тестовых программ на языке С.