МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

«Использование операторного языка для реализации операций, обеспечивающих специализированные исследовательские системы»

Студент 2-го курса Магистратуры:
Рубан Андрей Алексеевич
Студент:
Майдан Алексей Эдгарович
Научный руководитель: Балакирев Н.Е.

ЦЕЛЬ РАБОТЫ

Создание и демонстрация применения операторного языка для реализации алгоритмов в прикладных областях исследования, позволяющего оптимизировать используемые ресурсы, обеспечить наглядность и понимание их содержания, а также облегчить их разработку за счет создания библиотек макро операций в терминах предметной области.

ВВЕДЕНИЕ

В рамках работ по исследованию и распознаванию информационного содержания волн создание программных средств ведется как на языках высокого уровня, так и на ассемблере. Использование макроязыка позволило уйти от трудоемкости написания программ в терминах команд компьютера и разработать специализированный язык операторов. Постепенное расширение набора операторов привело к созданию целой системы инструментов, которые по своему функционалу стали близки к языкам высокого уровня и, кроме того, обеспечили гибкость в именовании и оформлении операторов. Таким образом, появилась возможность конструировать специализированные алгоритмы в терминах предметной области и в то же время определять «смысл» производимых действий на содержательном уровне в терминах этой предметной области. При разработке языка операторов был выбран язык FASM, который обеспечил возможность создавать любой необходимый набор операторов, в том числе и возможность именования операторов на русском языке.

ЯЗЫК ПОЗВОЛИЛ РЕШИТЬ СЛЕДУЮЩИЕ ЗАДАЧИ

Создание такого **языка** в рамках исследований в области извлечения информационного содержания волн **позволило нам решать следующие задачи**:

- Формализовать специализированные операции работы с данными через набор функциональных операторов.
- Оформить такие специализированные операции, опирающиеся на формализованные операторы, в виде синтаксических конструкций, понятных по содержанию и удобных в применении. Семантическая «нагрузка» на такие синтаксические конструкции может быть переопределена через синонимию таких конструкций.
- Обеспечить систематизацию используемых системных библиотек и оптимизацию используемых ресурсов при написании программ.
- Предоставить возможность упрощения представления основных системных директив при оформлении программ (Например, Начало_Программы или же Конец_Программы).
- Разрешить проблемы совместного использования различных кодировок при оформлении программ.

ПУТЬ ДЛЯ ДОСТИЖЕНИЯ ПОСТАВЛЕННЫХ ЗАДАЧ

Решение поставленных задач достигается через:

- Множество форм представления операторов.
- Предоставление сервиса для систематизации и оптимизации используемых ресурсов.
- Упрощение представления основных системных директив при оформлении программ.
- Обеспечению сервиса по совместному использованию различных кодировок при оформлении программ.

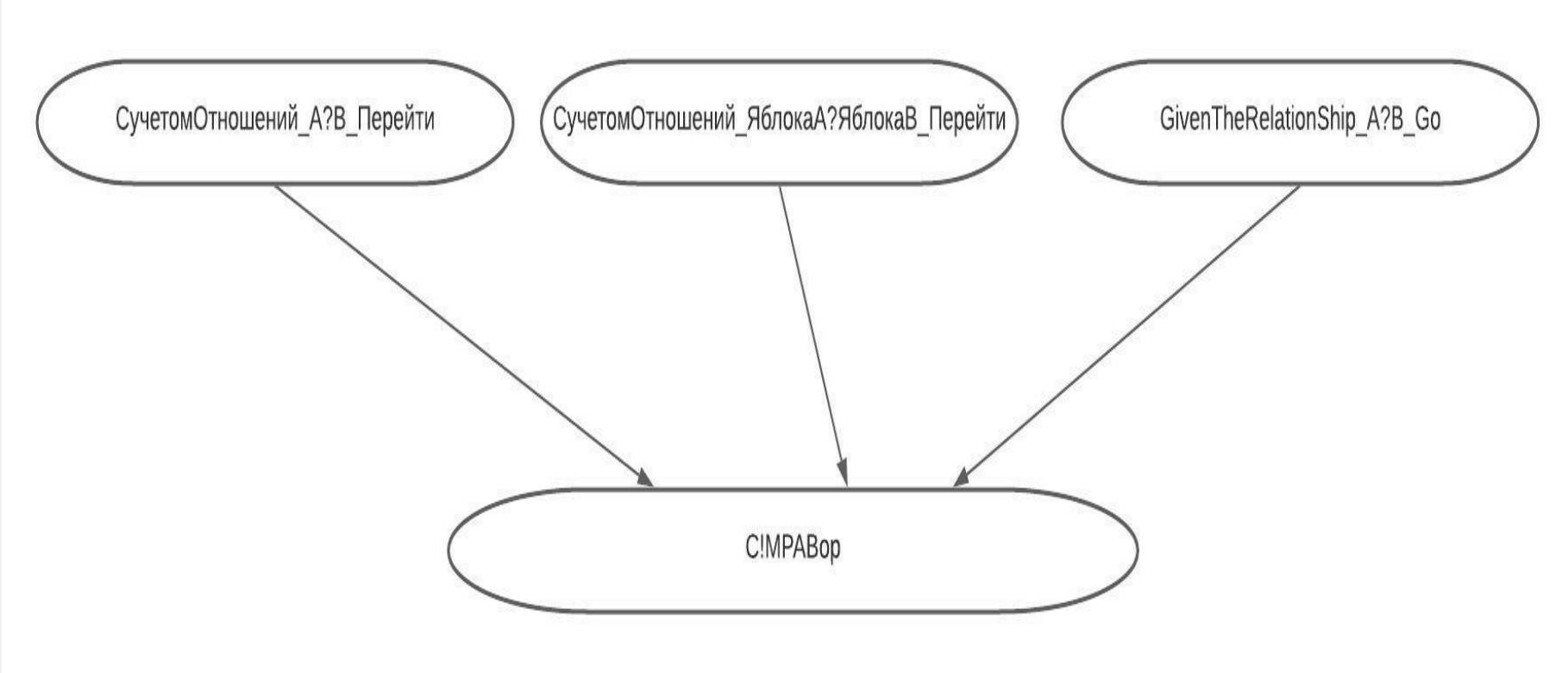
МНОЖЕСТВО ФОРМ ПРЕДСТАВЛЕНИЯ ОПЕРАТОРОВ

Все созданные операторы в таком языке могут иметь две взаимосвязанные формы:

- **Функциональная форма** предполагает оператор, который ориентирован на оптимальную реализацию алгоритма и **по виду** подобен вызову процедуры с параметрами**. По содержанию** функциональный оператор представляет из себя последовательность команд, которая появится после развертывания макроса с учетом входных параметров
- **Семантическая форма** предполагает оператор, который ориентирован на удобство применения и содержательность в представлении. **По виду** это некоторое формальное выражение с удобным способом задания параметров. **По содержанию** это алгоритм разборки синтаксиса оператора для представления его в функциональной форме.

СИНОНИМЫ СЕМАНТИЧЕСКИХ ОПЕРАТОРОВ

Механизм макрообработки позволяет относительно одного **семантического оператора** иметь множество **синонимов**, таким образом расширяя лингвистический функционал операторов с учетом предметных областей.



ПРИМЕР ИСПОЛЬЗОВАНИЯ ОПЕРАТОРОВ

Пусть имеется следующая **задача**:

Необходимо установить отношение между тремя беззнаковыми переменными и перейти с учетом установленного отношения в одну из 13 точек программы для дальнейшей обработки. Значения, между которыми требуется установить отношение, будут находится по адресам AX, BX, CX. А 13 входов для точек перехода будут именоваться P1, P2,…., P13. Тогда под именем UC!ABCJmp мы создаем функциональную форму оператора, которое будет реализовывать это действие через определенную последовательность команд.

Функциональная форма обращения

```
UC!ABCJmp    AX,BX,CX,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,ExiPloxo
```

Трудности использования только лишь функционального оператора

При использовании только лишь функциональной формы такого оператора возникают трудности, такие как:

- Слабая наглядность передаваемых параметров оператору при обращении.
- При большом количестве возможных адресов для передачи управления в программе, возникает потребность все время держать в голове или где-то смотреть при каком условии произойдет передача управления по тому или иному адресу.

ПРЕДЛАГАЕМАЯ КАСКАДНАЯ СЕМАНТИЧЕСКАЯ ФОРМА ОПЕРАТОРА

Предлагаемая многокаскадная семантическая форма, которая будет обращаться к функциональной форме.

```
1  УчитываяОтношенияWПерейти\
2          A?B?C?\
3                   $<\>Real=>                  P1\
4                    $</>Real=>                 P2\
5                     $<-->Real=>               P3\
6                      $</``>Real=>             P4\
7                       $<_/>Real=>             P5\
8                        $<``\>Real=>           P6\
9                         $<\_>Real=>           P7\
10                         $</\>Real=>          P8\
11                          $</\.>Real=>        P9\
12                           $<./\>Real=>       P10\
13                            $<\/>Real=>       P11\
14                             $<`\/>Real=>     P12\
15                              $<\/`>Real=>    P13\
16                               $bad=>         ПлохойВыход
```

ПАССИВНОЕ ВНЕСЕНИЕ КОММЕНТАРИЯ ДЛЯ УДОБСТВА ВОСПРИЯТИЯ СИНТАКСИЧЕСКОЙ ФОРМЫ.

Для контекстной ориентации по направлениям переходов были реализованы пассивные комментарии. Последовательность $<\> => Pk означает следующее

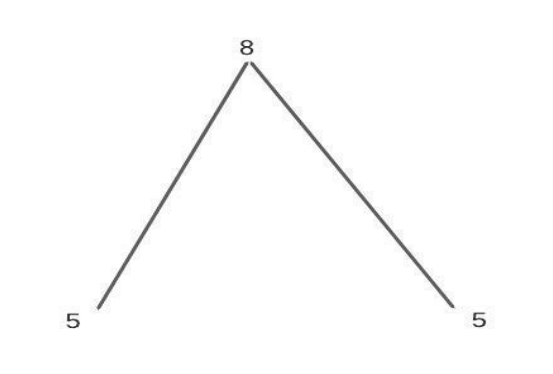
$ - означает начало комментария;

< > скобки для графического представления фигуры, образуемой тремя точками;

=> стрелка указывающая на точку входа в блок;

Pk метка входа в блок обработки

ГРАФИЧЕСКОЕ ОБОЗНАЧЕНИЕ ВВОДИМЫХ КОММЕНТАРИЕВ



ИСПОЛЬЗОВАНИЕ ЯЗЫКА ОПЕРАТОРОВ ДЛЯ ОПТИМИЗАЦИИ ИСПОЛЬЗУЕМЫХ РЕСУРСОВ

В противовес тупому присоединению различных библиотек реализован набор операторов, который позволяет производить систематизацию библиотек и минимизировать используемые ресурсы при трансляции программы за счет указания одного символа, который позволяет подключать или отключать группы операторов или отдельные операторы.

Фрагмент списка операторов используемых и не используемых в программе

```
;*************************************************************************
! УпрятаliРегистры       1+      'C:\FASM\BIBLMCN\PUSHe.inc'
! ВосстановилиРегистры   2*      'C:\FASM\BIBLMCN\PUSHe.inc'
! УпрятатьРегистры       3*      'C:\FASM\BIBLMCN\PUSHe.inc'
! ВостановитьРегистры    4*      'C:\FASM\BIBLMCN\PUSHe.inc'
! CrPUSH                 5*      'C:\FASM\BIBLMCN\PUSHe.inc'
! CrPOP                  6*      'C:\FASM\BIBLMCN\PUSHe.inc'
;*************************************************************************
! PUSHe                  7-      'C:\FASM\BIBLMCN\PUSHREG\CrPUSHs.inc'
! POPe                   8-      'C:\FASM\BIBLMCN\PUSHREG\CrPUSHs.inc'
! CrPUSHs                9-      'C:\FASM\BIBLMCN\PUSHREG\CrPUSHs.inc'
! CrPOPs                 10-     'C:\FASM\BIBLMCN\PUSHREG\CrPUSHs.inc'
! CrPUSHe                11-     'C:\FASM\BIBLMCN\PUSHREG\CrPUSHs.inc'
! CrPOPe                 12-     'C:\FASM\BIBLMCN\PUSHREG\CrPUSHs.inc'
;*************************************************************************
```

УПРОЩЕНИЕ ПРЕДСТАВЛЕНИЯ ОСНОВНЫХ СИСТЕМНЫХ ДИРЕКТИВ В ЯЗЫКЕ ОПЕРАТОРОВ

Одна из главных идей создания такого языка операторов – приближение именования используемых операторов при написании текста программы к предметной области, для которой пишется текст программы, а, соответственно, и приближение описания программы к более человечному понимаю, что в свою очередь дает программисту возможность выразить интерпретацию алгоритма человеческим языком.

Немало важным моментом для воплощения в жизнь данной идеи является упрощение представления основных системных директив, которые задают трафарет программы, или же можно сказать «шапку» программы.

Использования языка FASM позволило нам реализовать системные операторы, направленные на облегчение использование некоторых системных директив, например, начала программы или объявления сегмента команд и сегмента данных. В реализованных операторах, любой параметр, с которым мы обращаемы к шаблону, сопровождается ключевым словом, которому присваивается конкретное значение

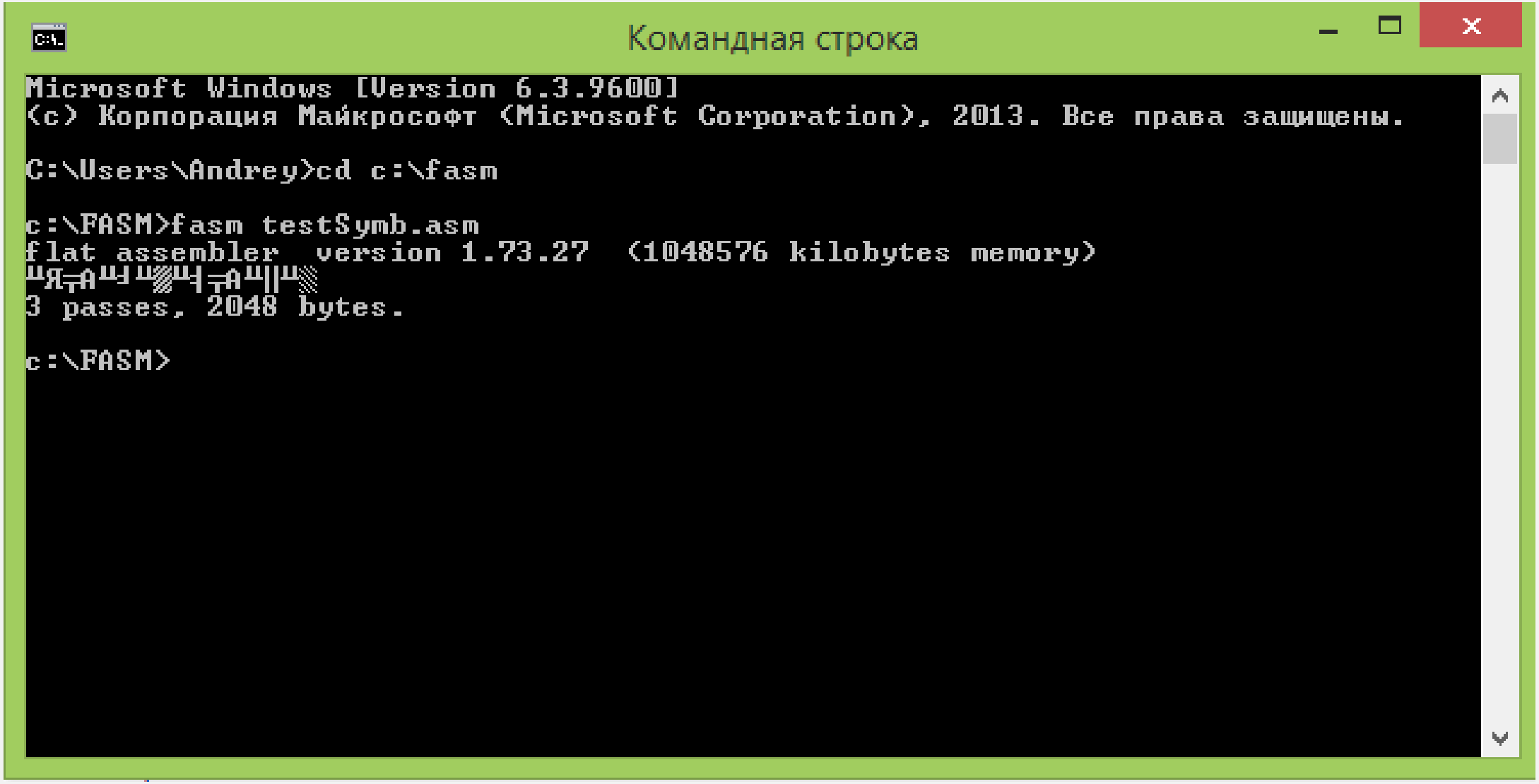
```
1  include 'C:\FASM\MZERO.inc'
2
3  НАЧАЛО_Программы Форма=PE,Вывод=Console,Имя_Входа=start
4  СЕКЦИЯ_ДАННЫХ Имя=data,Чтение=+,Запись=+
5
6  ; Данные пишутся здесь
7
8  СЕКЦИЯ_КОМАНД
9
10 ; Текст программы пишется здесь
11
12 СЕГМЕНТ_ИМПОРТА
```

РЕШЕНИЕ ПРОБЛЕМЫ ИСПОЛЬЗОВАНИЯ РУССКИХ СИМВОЛОВ

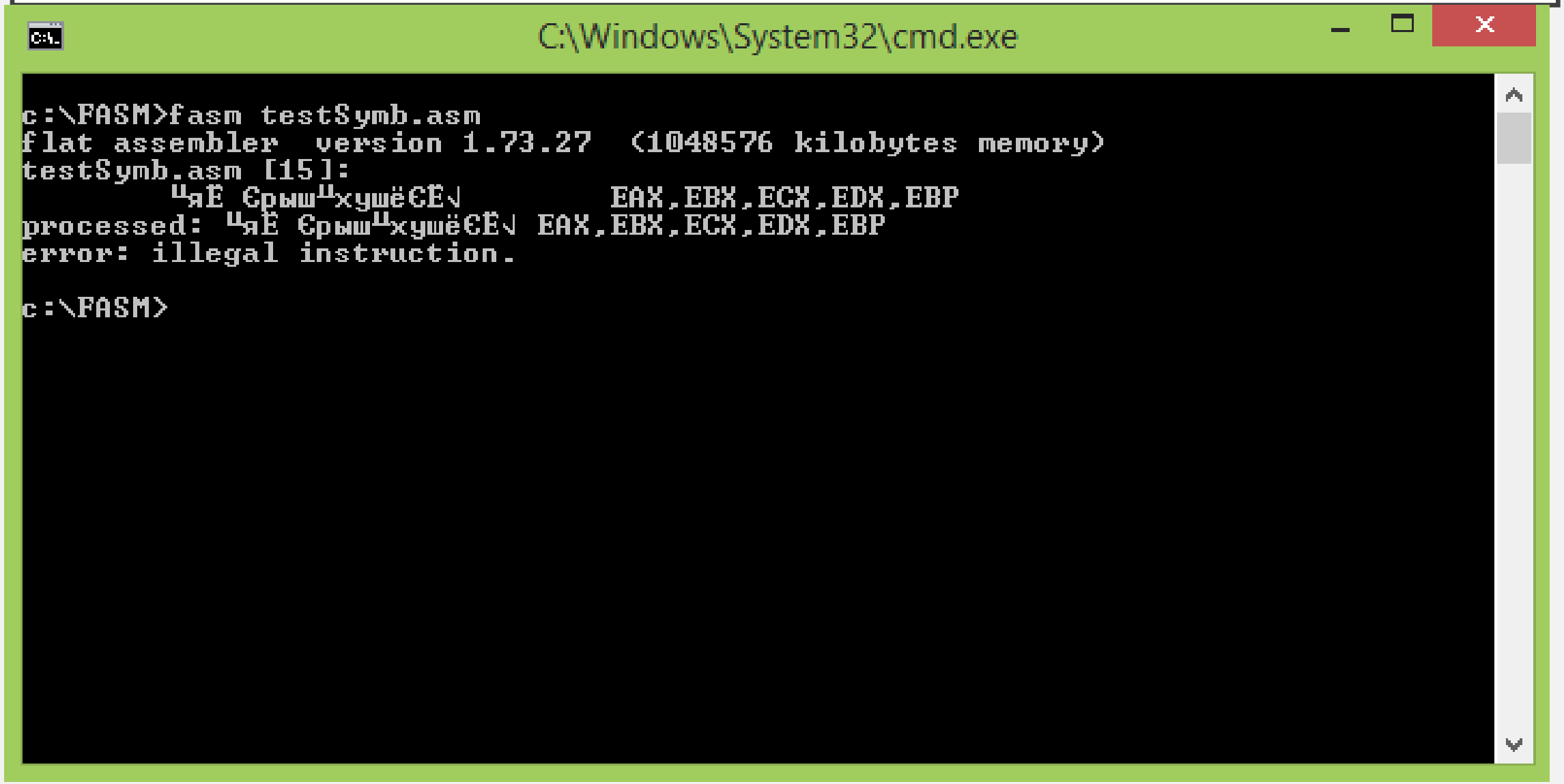
В ходе оформления программ и их исполнении, когда на русском языке используются имена **операторов** (макроопределений, именуемых русскими символами), диагностические распечатки в макроопределениях, а также задаются текстовые константы - наблюдается коллизия между кодировками, используемыми в редакторах, таких как БЛОКНОТ и Notepad++ и кодировкой используемой ОС при выдаче на экран. Всё это **проявляется в** виде появления «**некорректных**» **символов** при распечатке или же в **несогласованности** имен **макроопределений** в момент их макровызова.

В редакторе «БЛОКНОТ» используется только кодировка «windows-*1251»,* а в редакторе «Notepad++» можно использовать целый спектр кодировок, в том числе и кодировку ANSI (кодировка windows) или «*windows-1251»*, а для распечатки на экран из программы нужна кодировка «OEM 866». Но, если изменить кодировку файла на «OEM 866» и при этом использовать русские имена макроопределений, то они будут неправильно интерпретироваться транслятором ассемблера FASM.

ПРИМЕР НЕКОРРЕКТНОГО ОТОБРАЖЕНИЯ СИМВОЛОВ ПРИ ПЕЧАТИ



ПРИМЕР НЕСОГЛАСОВАННОСТИ ИМЕН МАКРООПРЕДЕЛЕНИЙ В МОМЕНТ ИХ МАКРОВЫЗОВА



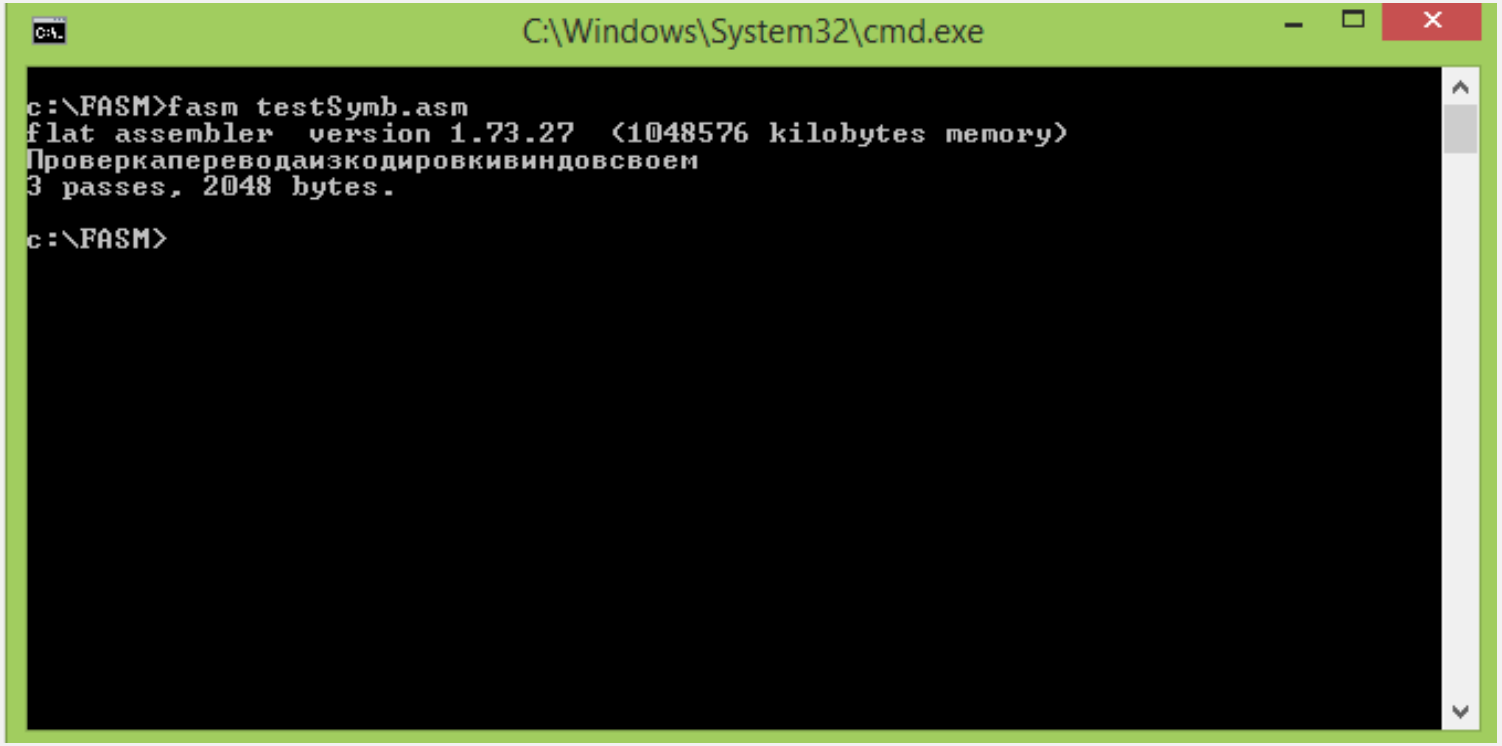
РЕШЕНИЕ ПРОБЛЕМЫ КОЛЛИЗИИ КОДИРОВОК

Созданы операторы автоматического «перевода» на лету из кодировки «*windows-1251»* в кодировку «OEM 866» для «БЛОКНОТ» и из кодировки «UTF-8» опять же в кодировку «OEM 866» для «Notepad++» для вышеперечисленных случаев.

Были написаны 4 макроопределения: для редактора Notepad++ «dbr» - для задания русских текстовых констант и Print!Rfall- для представления распечаток во время трансляции, а в редакторе БЛОКНОТ «dbb» и Print!Bfall, соответственно.

ПРИМЕР ИСПОЛЬЗОВАНИЯ

Применение макроопределений не вызывает затруднений, необходимо знание только имени оператора при макровызове. Например, вместо задания текстовой константы на Ассемблере в обычной ситуации надо писать ***db " Проверка перевода из кодировки виндовс в оем"***, тогда как для случая использования макроопределения — это будет ***dbb "Проверка перевода из кодировки виндовс в оем".***



ЗАКЛЮЧЕНИЕ

В результате работы продемонстрированы ключевые преимущества языка операторов, а так же проведено его расширение с учетом поставленных задач

СПАСИБО ЗА ВНИМАНИЕ