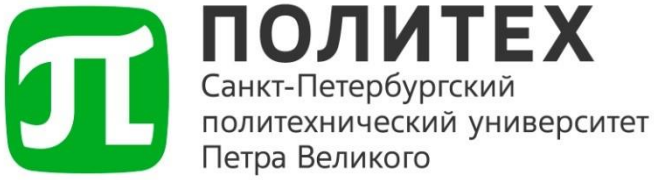
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
ВШ ПРОГРАММНОЙ ИНЖЕНЕРИИ



Индивидуальная работа
по дисциплине «Архитектура ЭВМ»

Выполнил
Студент группы № 13537/1 Григоренко С. А.

Руководитель
Профессор д. т. н. Молодяков С. А.

«11» декабря 2018 г

Санкт-Петербург
2018 г.

Содержание

Введение

Ассемблер — транслятор исходного текста программы, написанной на языке ассемблера, в программу на машинном языке.

Как и сам язык, ассемблеры, как правило, специфичны для конкретной архитектуры, операционной системы и варианта синтаксиса языка. Вместе с тем существуют мультиплатформенные или вовсе универсальные (точнее, ограниченно-универсальные, потому что на языке низкого уровня нельзя написать аппаратно-независимые программы) ассемблеры, которые могут работать на разных платформах и операционных системах. Среди последних можно также выделить группу кросс-ассемблеров, способных собирать машинный код и исполняемые модули (файлы) для других архитектур и операционных систем.

Ассемблирование может быть не первым и не последним этапом на пути получения исполнимого модуля программы. Так, многие компиляторы с языков программирования высокого уровня выдают результат в виде программы на языке ассемблера, которую в дальнейшем обрабатывает ассемблер. Также результатом ассемблирования может быть не исполняемый, а объектный модуль, содержащий разрозненные блоки машинного кода и данных программы, из которого (или из нескольких объектных модулей) в дальнейшем с помощью редактора связей может быть получен исполнимый файл.

Условие задачи

Дано арифметическое выражение. Разработать программу проверки правильности по следующим критериям:

- Допустимые знаки операций (“+”, “-“, “*”,”/”);
- Допустимые константы (целые без знака);
- Допустимые переменные (до пяти латинских букв);
- Скобки (только круглые, они должны быть парными).

Продумать диагностику по различным типам ошибок

Описание решения

Пользователь вводит последовательность символов, одновременно с этим программа проверяет вводимые символы на соответствие синтаксису выражения. Выражение определено с помощью формы Бэкуса-Наура(БНФ) – формальной системы описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории. Мной была реализована следующая БНФ-конструкция:

<выражение> ::= <терм> | <терм> + <выражение> | <терм> – <выражение>
<терм> ::= [( ] <множитель> [ )] | [( ] <множитель> * <терм> [ )] | [( ] <множитель> / <терм> [ )]
<множитель> ::= <целое без знака> | <идентификатор>
<целое без знака> ::= <цифра> | <цифра> <целое без знака>
<идентификатор> ::= <буква>
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<буква> ::= a | b | c | d | e | f

Стоит заметить, что парность скобок и то, что соответствующие открывающие скобки вводятся перед закрывающими проверяется отдельно. В случае не соответствия выражения введенного пользователем БНФ-конструкции или ошибок со скобками выводится сообщение о неправильности выражения, в противном случае выводится сообщение о корректности выражения.

Список используемых обработчиков прерываний

В программе используется обработчик прерываний INT 21H и следующие его функции:

Функция	**Назначение**
01h	Ввод символа с клавиатуры
01h	Вывод символа на экран
09h	Вывод строки символов на экран

Список используемых макросов

В программе используется макрос gg. Он выводит сообщение с просьбой ввести строку.

```
gg macro
   mov ah, 09h ;Просим пользователя ввести строку
   lea dx, prompt
   int 21h
   lea dx, newline
   int 21h
endm
```

Текст программы с комментариями

```
.MODEL SMALL
.STACK 100h
gg macro
   mov ah, 09h ;Просим пользователя ввести строку
   lea dx, prompt
   int 21h
   lea dx, newline
   int 21h
endm
.DATA
   char DB 1
   openBrackets DW 0
   closedBrackets DW 0
   prompt DB 'Enter a string: ','$'
   newline DB 0Dh, 0Ah, '$'
   messTrue DB 'This is an expression','$'
   messFalse DB 'This is NOT an expression','$'
.CODE
main proc
   mov ax, @data
   mov ds, ax

   gg ;вызываем макрос

   call GetChar ;Получаем первый символ ввода
   call IsExpression  ;Начинаем проверку

   cmp al, 1 ;Если строка не прошла проверку, то это - не выражение
   jne notAnExpression
```

```
    lea si, openBrackets
    lea di, closedBrackets
    mov dx, [si]
    mov bx, [di]

    cmp dx, bx ; Если в строке не одинаковое количество ‘(’ и  ‘)’, то это – не выражение
    jne notAnExpression; To chto ( nashodyatsa pered ) proveryaetsa v isTerm

    lea si, char
    mov bl, [si]
    cmp bl, 13    ;13 – asci код возврата каретки
    ; Если после выражения строка не заканчивается, то это – не выражение
    jne notAnExpression

        mov ah, 09h   ;Вывод сообщения о том, что строка - выражение
        lea dx, newline
        int 21h
        lea dx, messTrue
        int 21h

    jmp exit
    notAnExpression:  ;Вывод сообщения о том, что строка – не выражение
        mov ah, 09h
        lea dx, newline
        int 21h
        lea dx, messFalse
        int 21h
    exit:
main endp

GetChar proc ;Сохраняет следующий символ ввода в переменную char
    push si
    push ax

    mov ah,1h
    int 21h

    lea si, char
    mov [si], al
    pop ax
    pop si
ret
GetChar endp

IsExpression proc ;Проверяет определение выражения по БНФ-конструкции
    call IsTerm
    cmp ax,1
    jne IsExpressionFalse1
        push si
        lea si, char
```

```
        mov al, [si]
        pop si

        cmp al,43
        je IsExpressionTrue2
        cmp al,45
        jne IsExpressionFalse2
        IsExpressionTrue2:
            call GetChar
            call IsExpression
            ret
        IsExpressionFalse2:
            mov ax,1
            ret
    IsExpressionFalse1:
        mov ax, 0
        ret
IsExpression endp

IsTerm proc ; Проверяет определение терма  по БНФ-конструкции
    call IsOpenBracket

    call IsMultiplier
    cmp ax,1
    jne IsTermFalse1
        call IsClosedBracket

        push si
        push di
        lea si, openBrackets
        lea di, closedBrackets
        mov ax, [si]
        mov bx, [di]
        pop di
        pop si

        cmp ax, bx
        jl IsTermFalse1

        push si
        lea si, char
        mov al, [si]
        pop si

        cmp al,42
        je IsTermTrue2
        cmp al,47
        jne IsTermFalse2
        IsTermTrue2:
            call GetChar
            call IsTerm
```

```
        ret
      IsTermFalse2:
        mov ax,1
        ret
   IsTermFalse1:
      mov ax, 0
      ret
IsTerm endp

IsMultiplier proc   ; ; Проверяет определение множителя  по БНФ-конструкции
   call IsLetter
   cmp ax, 1
   jne IsMultiplierFalse
      mov ax, 1
      ret
   IsMultiplierFalse:
      call IsUnsignedInt
      ret
IsMultiplier endp

IsUnsignedInt proc; Проверяет определение  целого без знака по БНФ-конструкции
   call IsDigit
   cmp ax, 1
   jne IsUnsignedIntFalse
      call IsUnsignedInt
      mov ax, 1
      ret
   IsUnsignedIntFalse:
      mov ax, 0
      ret
IsUnsignedInt endp

IsDigit proc ; Проверяет определение цифры  по БНФ-конструкции
   push si
   lea si, char
   mov al, [si]
   pop si

   cmp al, 57
   jg IsDigitFalse
   cmp al, 48
   jl IsDigitFalse
      call GetChar
       mov ax, 1
      ret
   isDigitFalse:
      mov ax, 0
      ret
IsDigit endp

IsLetter proc ; Проверяет определение буквы  по БНФ-конструкции
```

```
    push si
    lea si, char
    mov al, [si]
    pop si

    cmp al, 102
    jg IsLetterFalse
    cmp al, 97
    jl IsLetterFalse
        call GetChar
        mov ax, 1
        ret
    IsLetterFalse:
        mov ax, 0
        ret
IsLetter endp

IsOpenBracket proc ; Обрабатывает открывающую скобку
    push ax
    push si
    lea si, char
    mov al, [si]
    pop si

    cmp al, 40
    jne IsOpenBracketFalse
        call GetChar

        push si
        lea si, openBrackets
        mov ax, [si]
        inc ax
        mov [si], ax
        pop si

        call IsOpenBracket
    IsOpenBracketFalse:
        pop ax
        ret
IsOpenBracket endp

IsClosedBracket proc ; Обрабатывает закрывающую скобку
    push ax
    push si
    lea si, char
    mov al, [si]
    pop si

    cmp al, 41
    jne IsClosedBracketFalse
        call GetChar
```

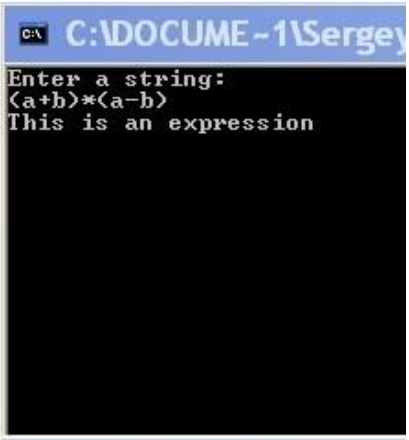
```
        push si
        lea si, closedBrackets
        mov ax, [si]
        inc ax
        mov [si], ax
        pop si

        call IsClosedBracket
    IsClosedBracketFalse:
        pop ax
        ret
IsClosedBracket endp
end main
```

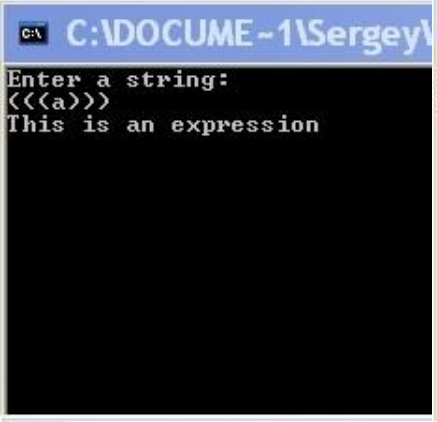
Примеры работы программы


C:\DOCUME~1\Ser
Enter a string:
a+b+12/c
This is an expression


C:\DOCUME~1\Sergey\
Enter a string:
11-22
This is an expression

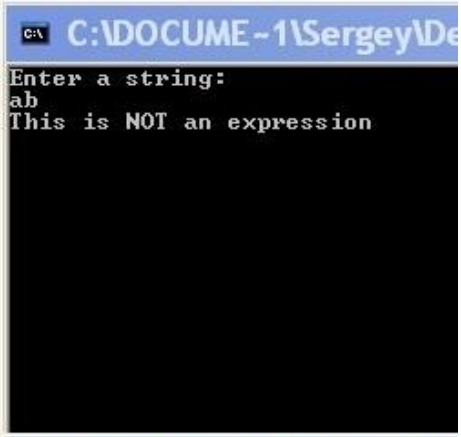

C:\DOCUME~1\Sergey
Enter a string:
(a+b)*(a-b)
This is an expression


C:\DOCUME~1\Ser
Enter a string:
(a*a+2*a*b+b*b)
This is an expression

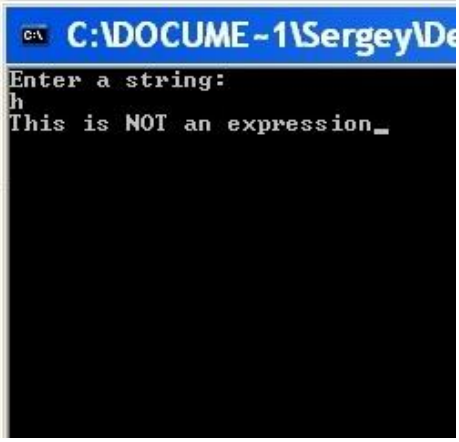

C:\DOCUME~1\Sergey\
Enter a string:
(((a)))
This is an expression

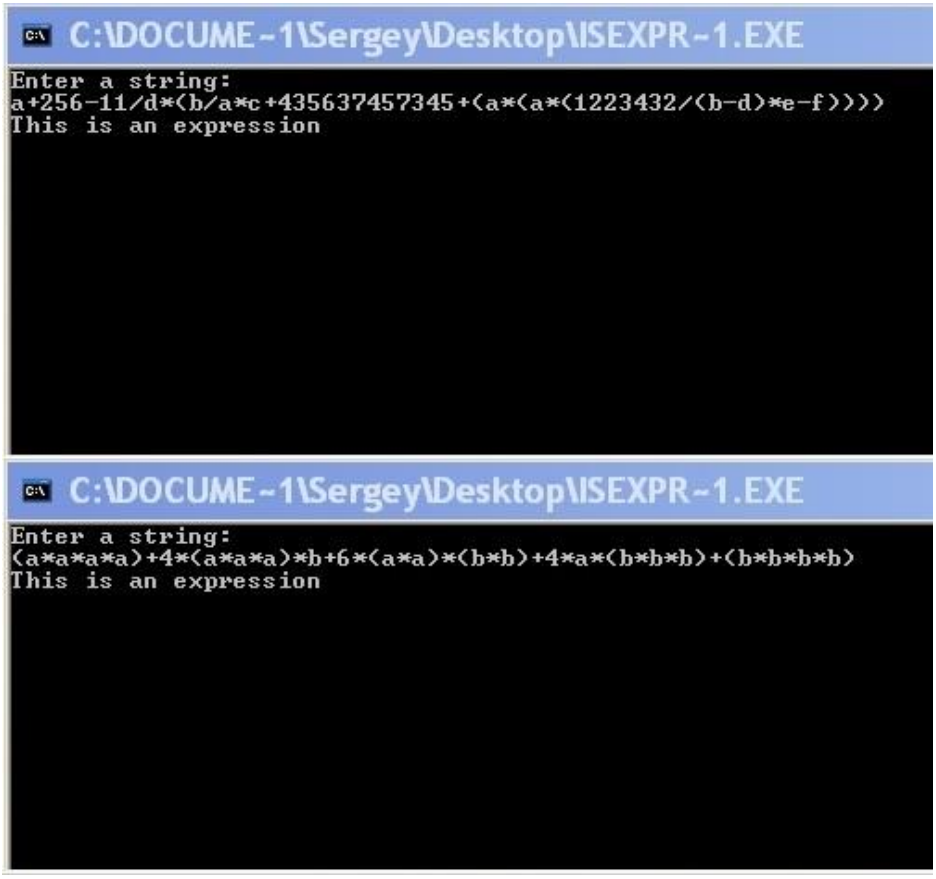

C:\DOCUME~1\Sergey
Enter a string:
)
This is NOT an expression


C:\DOCUME~1\Sergey
Enter a string:
(a*a*)
This is NOT an expression


C:\DOCUME~1\Sergey\De
Enter a string:
ab
This is NOT an expression


C:\DOCUME~1\Sergey
Enter a string:
128+(256*(d+4)
This is NOT an expression


C:\DOCUME~1\Sergey\De
Enter a string:
h
This is NOT an expression_



Список используемой литературы

1. Дао Л. Микропрограммирование процессора 8088. М.: Мир, 1988
2. Молодяков С. А. Архитектура ЭВМ. Программирование переферийных устройств. Лабораторный практикум. СПб.: СПбГПУ, 2014
3. https://ru.wikipedia.org/wiki/Ассемблер