МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) Институт информационных технологий

Кафедра Математического и программного обеспечения ЭВМ

# КУРСОВАЯ РАБОТА

по дисциплине Программирование на ассемблере

на тему Программирование на языке низкого уровня

Выполнил студент группы
1ПИб-02-3оп-23
направление подготовки (специальности)
09.03.04., Программная инженерия
шифр, наименование
Богданов Ренат Алексеевич
фамилия, имя, отчество

Руководитель
Виноградова Людмила Николаевна
фамилия, имя, отчество
Доцент, кандидат технических наук
должность

Дата представления работы
« » _____ 2024 г.
Заключение о допуске к защите

Оценка _____, _____ количество баллов

Подпись преподавателя _____

Череповец, 2024
*год*

Аннотация

Курсовая работа посвящена теме «Программирование на языке низкого уровня». Выполнил: Богданов Ренат Алексеевич, студент группы 1ПИб-02-3оп-23 Института информационных технологий Череповецкого государственного университета.

В работе исследуются основные возможности языка ассемблера микропроцессора Intel 8086, в том числе принципы создания файлов и работы с ними на данном языке. Описывается процесс разработки программы, которая считывает данные из одного текстового файла, заменяет строчные буквы слов, заключённых в скобки, на прописные, и записывает результат обработки в другой текстовый файл.

Оглавление

# Введение

Язык ассемблера [1] – это низкоуровневый язык программирования, обеспечивающий прямое взаимодействие с аппаратными ресурсами компьютера. Программист взаимодействует с микропроцессором, используя его команды и регистры, что позволяет добиться высокой производительности и оптимизации кода. В данной курсовой работе рассматривается ассемблер для процессора Intel 8086, выпущенного в 1978 году и являющегося основой архитектуры x86.

Процессор Intel 8086 [2] представляет собой модернизированный вариант Intel 8080 с улучшенной производительностью. Он имеет 14 16-разрядных регистров: 4 регистра общего назначения (AX, BX, CX, DX), 2 индексных регистра (SI, DI), 2 указательных регистра (BP, SP), 4 сегментных регистра (CS, SS, DS, ES), программный счётчик команд (IP) и регистр флагов (FLAGS). Основные возможности включают поддержку 16-разрядной адресации, работу с операндами до 65535 и доступ к памяти до 1МБайт. Система команд включает более 100 операций, что позволяет программистам разрабатывать простые и сложные программы.

Цель курсовой работы – создание программы на языке ассемблера для процессора Intel 8086, которая заменяет строчные буквы слов в скобках на прописные и записывает результат в новый текстовый файл.

Разработка программы на языке ассемблера позволяет глубже понять принципы работы микропроцессора и использовать его возможности, что полезно как для программистов, так и для студентов.

# Основная часть

## 1. Описание предметной области

В рамках курсовой работы рассматривается работа с файлами в операционной системе DOS под архитектурой 8086 с использованием эмулятора emu8086. Эмулятор emu8086 позволяет разрабатывать и тестировать программы на ассемблере для процессора 8086 без необходимости работы с реальным оборудованием. Рассмотрим основные функции DOS для работы с файлами [3].

1) Создание файла (функция 3CH)

Регистр DX должен содержать адрес ASCIIZ-строки, а регистр CX - необходимый атрибут (для обычного файла значение атрибута равно 0).

При правильном открытии операция создает элемент оглавления с данным атрибутом, очищает флаг CF и устанавливает файловый номер в регистре AX. Если создаваемый файл уже существует, то длина этого файла устанавливается в 0 для перезаписи.

2) Запись в файл (функция 40H).

В регистре BX должен быть установлен файловый номер, в регистре CX - число записываемых байт, а в регистре DX - адрес области вывода.

Правильно выполненная операция записывает из памяти на диск все данные (256 байт), очищает флаг CF и устанавливает в регистре AX число действительно записанных байтов. Если диск переполнен, то число записанных байтов может отличаться от заданного числа.

3) Закрытие файла (функция 3EH).

В регистре BX должен находиться файловый номер. Эта операция записывает все оставшиеся еще данные из буфера на диск, корректирует оглавление и таблицу FAT.

В случае ошибки в регистре AX устанавливается код 06 (неправильный файловый номер).

4) Чтение дискового файла (функция 3DH).

Эта операция проверяет правильность имени файла и его наличие на диске. При открытии файла регистр DX должен содержать адрес необходимой ASCIIZ-строки, а регистр AL - код доступа.

Если файл с необходимым именем существует, то операция открытия устанавливает длину записи, равной 1, принимает существующий атрибут, сбрасывает флаг CF и заносит файловый номер в регистр AX.

Если файл отсутствует, то операция устанавливает флаг CF и заносит в регистр AX код ошибки: 02, 04, 05 или 12.

5) Чтение записей файла (функция 3FH).

В регистре BX нужно установить файловый номер, в регистре CX - число читаемых байтов и в регистре DX - адрес области ввода. В следующем примере происходит считывание записи длиной 512 байт.

Правильно выполненная операция считывает запись в память, сбрасывает флаг CF и устанавливает в регистре AX число действительно прочитанных байтов. Нулевое значение в регистре AX обозначает попытку чтения после конца файла. Ошибочная операция устанавливает флаг CF и возвращает в регистре AX код ошибки: 05 (нет доступа) или 06 (ошибка файлового номера).

6) Управление файловым указателем (функция 42H).

Система DOS имеет файловый указатель, который при открытии файла устанавливается в 0 и увеличивается на 1 при последовательных операциях записи или чтения. Для доступа к любым записям внутри файла можно менять файловый указатель с помощью этой функции, получая прямой доступ к требуемым записям файла.

Для установки файлового указателя необходимо поместить в регистр BX файловый номер и в регистровую пару CX:DX - требуемое смещение в байтах. Для смещений до 65535 в регистре CX устанавливается 0, а в DX - смещение. В регистре AL должен быть установлен один из кодов, который определяет точку отсчета смещения.

Правильно выполненная операция сбрасывает флаг CF и возвращает новое положение указателя в регистровой паре DX:AX. Неправильная операция

устанавливает флаг CF в 1 и возвращает в регистре AX код 01 (ошибка кода отсчета) или 06 (ошибка файлового номера).

## 2. Постановка задачи

Разрабатываемый программный продукт должен:

- получать на вход текстовый файл в кодировке ANSI для обработки;
- обрабатывать текст входного файла, заменяя строчные буквы русского алфавита, заключённые в круглые скобки, на прописные;
- создавать новый файл и записывать в него результаты обработки текста исходного файла или выводить изменённый текст на экран;
- выводить сообщения об ошибках или успешном выполнении программы на экран в случае возникновения исключительных ситуаций, таких как отсутствие исходного файла, некорректный формат данных в файле или пустой файл;
- работать на персональном компьютере с процессором не младше Intel 8086 под управлением ОС Windows XP и новее через эмулятор микропроцессора Intel 8086 – EMU8086 [4].

## 3. Выбор структур данных

В таблице 1 представлены структуры данных, используемые в программе.

Таблица 1

Структуры данных программы

| Обозначение | Наименование | Тип данных |
|---|---|---|
| 1 | 2 | 3 |
| inFileName | Путь к входному текстовому файлу input.txt | Байт |
| outFileName | Путь к выходному текстовому файлу output.txt | Байт |

Продолжение табл. 1

| 1 | 2 | 3 |
|---|---|---|
| buffer | Буфер для сохранения символов из текста входного файла | Байт |
| tempBuffer | Временный буфер для преобразования | Байт |
| bytesRead | Количество прочитанных байт | Слово |
| inFileHandle | Описатель входного файла input.txt | Слово |
| outFileHandle | Описатель выходного файла output.txt | Слово |
| insideParentheses | Флаг, показывающий, внутри ли скобок | Байт |
| errorMessage | Основное сообщение об ошибке | Байт |
| fileNotFoundMsg | Сообщение об ошибке: файл не найден | Байт |
| readErrorMsg | Сообщение об ошибке: ошибка чтения файла | Байт |
| writeErrorMsg | Сообщение об ошибке: ошибка записи файла | Байт |
| closeErrorMsg | Сообщение об ошибке: ошибка закрытия файла | Байт |
| unbalancedParenthesesMsg | Сообщение об ошибке: несбалансированные скобки | Байт |
| emptyFileMsg | Сообщение об ошибке: файл пуст | Байт |
| newline | Перенос строки | Байт |

4. Логическое проектирование

1) программа пытается открыть входной файл input.txt для чтения, либо выводит сообщение "Файл не найден" и завершает выполнение;

2) программа проверяет, не пуст ли файл, либо выводит сообщение "Файл пуст" и завершает выполнение;
3) программа создаёт выходной файл output.txt, либо выводит сообщение "Ошибка при записи файла" и завершает выполнение;
4) программа читает и выводит содержимое входного файла на экран блоками по 512 байт, либо выводит сообщение "Ошибка при чтении файла" и завершает выполнение;
5) программа обрабатывает текст исходного файла посимвольно:
   - если символ — открывающая скобка (, программа устанавливает флаг insideParentheses в 1;
   - если символ — закрывающая скобка ), программа проверяет, установлен ли флаг insideParentheses. Если флаг установлен, программа сбрасывает его в 0, иначе выводит сообщение "Несбалансированные скобки в файле" и завершает выполнение;
   - если символ строчная буква русского алфавита (от 'a' до 'z') и флаг insideParentheses установлен, программа преобразует его в прописную букву;
   - обработанный символ записывается во временный буфер tempBuffer;
6) программа записывает содержимое временного буфера tempBuffer в выходной файл, либо выводит сообщение "Ошибка при записи файла" и завершает выполнение;
7) после обработки всего содержимого входного файла программа проверяет, установлен ли флаг insideParentheses. Если флаг установлен, программа выводит сообщение "Несбалансированные скобки в файле" и завершает выполнение;
8) программа закрывает входной и выходной файлы, либо выводит сообщение "Ошибка при закрытии файла" и завершает выполнение;
9) программа открывает выходной файл для чтения, либо выводит сообщение о невозможности открыть файл и завершает выполнение;

10) программа читает и выводит содержимое выходного файла на экран блоками по 512 байт, либо выводит сообщение "Ошибка при чтении файла" и завершает выполнение;
11) программа завершает выполнение с кодом возврата 00h.

## 5. Физическое проектирование

В таблице 2 описаны модули, представленные в коде программы.

Таблица 2

Модули программы

| Имя модуля | Выполняемое действие |
|---|---|
| main | Открытие и закрытие файлов, чтение и запись данных, обработка данных, вывод сообщений об ошибках |
| fileNotFound | Вывод сообщения об ошибке File not found и завершение программы с ошибкой |
| readError | Вывод сообщения об ошибке Error reading file и завершение программы с ошибкой |
| writeError | Вывод сообщения об ошибке Error writing file и завершение программы с ошибкой |
| closeError | Вывод сообщения об ошибке Error closing file и завершение программы с ошибкой |
| unbalancedParentheses | Вывод сообщения об ошибке Unbalanced parentheses in file и завершение программы с ошибкой |
| emptyFile | Вывод сообщения об ошибке File is empty и завершение программы с ошибкой |
| showError | Вывод сообщения об ошибке и завершение программы с ошибкой |

6. Кодирование

1) Открытие исходного файла

Используется функция DOS 3Dh для открытия файла на чтение. Если файл не найден, программа выводит сообщение об ошибке и завершает выполнение.

```
1.    OPEN_INPUT:
2.    mov AH, 9h       ; функция вывода на экран
3.    mov DX, offset OPEN_MSG ; DX := адрес сообщения для вывода
      относительно начала сегмента
4.    int 21h          ; прерывание
5.    mov AL, 00h      ; открыть файл для чтения
6.    mov AH, 3Dh      ; функция открытия файла
7.    lea DX, inFileName ; DX := адрес файла input.txt
8.    int 21h          ; прерывание
9.    jc OPEN_ERROR    ; если CF = 1, то перейди к OPEN_ERROR
10.   mov inFileHandle, AX ; HANDLE_I := файловый указатель
      input.txt
11.   jmp GET_INPUT_SIZE
12.   OPEN_ERROR:
13.   mov AH, 9h       ; функция вывода на экран
14.   mov DX, offset fileNotFoundMsg ; DX := адрес сообщения ошибки
      относительно начала сегмента
15.   int 21h          ; прерывание
16.   jmp FINISH       ; переход к FINISH
```

2) Вычисление размера исходного файла

Используется функция DOS 42h для получения размера файла и установки указателя в начало файла.

```
1.    GET_INPUT_SIZE:
2.    mov AH, 42h      ; функция управления файловым указателем
3.    mov AL, 2        ; 2 – указатель на конец файла
4.    mov BX, inFileHandle ; BX := описатель файла input.txt
5.    xor CX, CX       ; CX:DX := смещение в файле в байтах
6.    xor DX, DX
7.    int 21h
8.    mov I_SIZE, AX   ; I_SIZE := размер файла input.txt
9.    mov AH, 42h      ; устанавливаем указатель обратно в начало
10.   mov AL, 0        ; 0 – указатель в начало файла
11.   mov BX, inFileHandle
12.   xor CX, CX
13.   xor DX, DX
14.   int 21h
```

3) Создание выходного файла

Используется функция DOS 3Ch для создания файла. Если создание файла не удалось, программа выводит сообщение об ошибке и завершает выполнение.

```
1.    CREATE_FILE:
2.    mov AH, 3Ch       ; функция создания файла
3.    mov CX, 00h       ; атрибуты файла
4.    lea DX, outFileName ; DX := адрес output.txt
5.    int 21h           ; прерывание
6.    jc CREATE_ERROR   ; если CF = 1, то перейди к CREATE_ERROR
7.    mov outFileHandle, AX ; HANDLE_O := описатель файла output.txt
8.    jmp READ_INPUT
9.
10.   CREATE_ERROR:
11.   mov AH, 9h        ; функция вывода на экран
12.   mov DX, offset writeErrorMsg ; DX := адрес сообщения ошибки
      относительно начала сегмента
13.   int 21h           ; прерывание
14.   jmp FINISH        ; переход к FINISH
```

4) Чтение исходного файла

Используется функция DOS 3Fh для чтения файла. Если чтение не удалось, программа выводит сообщение об ошибке и завершает выполнение.

```
1.    READ_INPUT:
2.    mov AH, 3Fh       ; функция чтения файла
3.    mov BX, inFileHandle ; BX := описатель файла input.txt
4.    mov CX, I_SIZE    ; CX := число байт для чтения (размер
      input.txt)
5.    lea DX, buffer    ; DX := адрес буфера, куда поместить текст
      файла
6.    int 21h
7.    jc READ_ERROR     ; если CF = 1, то перейди к READ_ERROR
8.    jmp PROCESS_TEXT
9.
10.   READ_ERROR:
11.   mov AH, 9h        ; функция вывода на экран строки
12.   mov DX, offset readErrorMsg ; DX := адрес сообщения ошибки
      относительно начала сегмента
13.   int 21h
14.   jmp FINISH        ; переход к FINISH
```

5) Посимвольная обработка исходного файла

Программа проходит по каждому символу в буфере. При обнаружении открывающей скобки ‘(’ устанавливается флаг “insideParentheses”. При обнаружении закрывающей скобки ‘)’ проверяется, установлен ли флаг. Если флаг установлен, он сбрасывается; иначе выводится сообщение об ошибке несбалансированных скобок.

Если символ является строчной буквой русского алфавита и флаг “insideParentheses” установлен, символ преобразуется в прописную букву.

```
1.    PROCESS_TEXT:
2.    xor SI, SI       ; индекс для буфера
3.    xor DI, DI       ; индекс для временного буфера
4.
5.    processLoop:
6.    cmp SI, bytesRead
7.    jge writeBuffer
8.    mov AL, buffer[SI]
9.    inc SI
10.
11.   ; Проверка на символы скобок
12.   cmp AL, '('
13.   je openParentheses
14.   cmp AL, ')'
15.   je closeParentheses
16.
17.   ; Преобразование букв в заглавные, если внутри скобок
18.   cmp insideParentheses, 1
19.   jne notInsideParentheses
20.   cmp AL, 'a'
21.   jb notLetter
22.   cmp AL, 'z'
23.   ja notLetter
24.   sub AL, 20h      ; Преобразование в заглавную букву
25.
26.   notLetter:
27.   jmp storeChar
28.
29.   openParentheses:
30.   mov insideParentheses, 1
31.   jmp storeChar
32.
33.   closeParentheses:
34.   cmp insideParentheses, 1
35.   jne unbalancedParentheses
36.   mov insideParentheses, 0
```

```
37.   jmp storeChar
38.
39.   notInsideParentheses:
40.   ; Ничего не меняем
41.
42.   storeChar:
43.   mov tempBuffer[DI], AL
44.   inc DI
45.   jmp processLoop
```

6) Запись обработанного текста в выходной файл

Обработанные символы записываются в выходной файл блоками по 512 байт, используя функцию DOS 40h.

```
1.    writeBuffer:
2.    ; Запись преобразованного буфера в выходной файл
3.    mov AH, 40h
4.    mov BX, outFileHandle
5.    mov CX, DI
6.    lea DX, tempBuffer
7.    int 21h
8.    jc writeError
9.
10.   jmp READ_INPUT
11.
12.   writeError:
13.   mov AH, 9h
14.   mov DX, offset writeErrorMsg
15.   jmp showError
```

7) Проверка на несбалансированные скобки

После обработки всех символов проверяется, установлен ли флаг “insideParentheses”. Если флаг установлен, выводится сообщение об ошибке несбалансированных скобок.

```
1.    closeInFileForProcessing:
2.    ; Проверка на несбалансированные скобки
3.    cmp insideParentheses, 1
4.    je unbalancedParentheses
```

8) Закрытие файлов

Входной и выходной файлы закрываются с помощью функции DOS 3Eh.

```
1.    ; Закрытие входного файла
2.    mov AH, 3Eh
3.    mov BX, inFileHandle
4.    int 21h
5.    jc closeError
6.
7.    ; Закрытие выходного файла
8.    mov AH, 3Eh
9.    mov BX, outFileHandle
10.   int 21h
11.   jc closeError
```

9) Вывод содержимого выходного файла на экран

Выходной файл открывается для чтения, и его содержимое выводится на экран блоками по 512 байт.

```
1.    ; Открытие выходного файла для чтения
2.    mov AH, 3Dh
3.    mov AL, 0
4.    lea DX, outFileName
5.    int 21h
6.    jc fileNotFound
7.    mov outFileHandle, ax
8.
9.    ; Чтение выходного файла и вывод на экран после обработки
10.   readOutFileLoop:
11.   mov ah, 3Fh
12.   mov bx, outFileHandle
13.   mov cx, 512
14.   lea dx, buffer
15.   int 21h
16.   jc readError
17.   mov bytesRead, ax
18.   cmp ax, 0
19.   je closeOutFile
20.
21.   ; Вывод прочитанного буфера на экран
22.   mov ah, 40h
23.   mov bx, 1 ; Дескриптор файла 1 - стандартный вывод
24.   mov cx, bytesRead
25.   lea dx, buffer
26.   int 21h
27.   jc writeError
```

```
28.
29.   jmp readOutFileLoop
30.
31.   closeOutFile:
32.   ; Закрытие выходного файла
33.   mov ah, 3Eh
34.   mov bx, outFileHandle
35.   int 21h
36.   jc closeError
```

Таким образом, процесс написания кода включает инициализацию сегментных регистров, открытие и закрытие файлов, чтение и запись данных, обработку символов и вывод результатов на экран с обработкой возможных ошибок.

7. Тестирование

Наборы тестовых данных представлены в таблице 3.

Таблица 3

Тестовые данные

| № | Исходные данные | Тестируемый модуль | Ожидаемый результат |
|---|---|---|---|
| 1 | (test) | Обработка скобок | (TEST) |
| 2 | (example) text (data) | Обработка скобок | (EXAMPLE) text (DATA) |
| 3 | Text (without) bracket | Обработка скобок | Text (WITHOUT) braces |
| 4 | (multiple) (brackets) | Обработка скобок | (MULTIPLE) (BRACKETS) |
| 5 | (nested (brackets)) | Обработка скобок | Ошибка: Несбалансированные скобки в файле |
| 6 | (unbalanced brackets | Обработка скобок | Ошибка: “Несбалансированные скобки в файле” |
| 7 | Пустой файл | Чтение файла | Ошибка: “Файл пуст” |
| 8 | Отсутствие файла | Чтение файла | Ошибка: “Файл не найден” |
| 9 | (TEXT) | Обработка скобок | (TEXT) |
| 10 | (lower) case (letters) (lower) case (letters) (lower) case (letters) (lower) case | Обработка скобок | (LOWER) case (LETTERS) (LOWER) case (LETTERS) (LOWER) case (LETTERS) (LOWER) case (LETTERS) (LOWER) case |

Результаты тестирования представлены в таблицы 4.

Таблица 4

Результаты тестирования

| Дата и время тестирования | Тестируемый модуль | Кто проводил тестирование | Описание теста | Результаты тестирования |
|---|---|---|---|---|
| 20.12.2024, 10:05 | Обработка скобок | Богданов Р.А. | Текст: “(test)” | Успех: “(TEST)” |
| 20.12.2024, 10:07 | Обработка скобок | Богданов Р.А. | Текст: “((example) text (data)” | Успех: “(EXAMPLE) text (DATA)” |
| 20.12.2024, 10:08 | Обработка скобок | Богданов Р.А. | Текст: “(Text (without) bracket” | Успех: “Text (WITHOUT) braces” |
| 20.12.2024, 10:11 | Обработка скобок | Богданов Р.А. | Текст: “((multiple) (brackets)” | Успех: “(MULTIPLE) (BRACKETS)” |
| 20.12.2024, 10:15 | Обработка скобок | Богданов Р.А. | Текст: “((nested (brackets))” | Успех: “Ошибка: Несбалансированные скобки в файле” |
| 20.12.2024, 10:17 | Обработка скобок | Богданов Р.А. | Текст: “((unbalanced brackets” | Успех: “Ошибка: “Несбалансированные скобки в файле”” |
| 20.12.2024, 10:26 | Чтение файла | Богданов Р.А. | Пустой файл | Успех: “Ошибка: “Файл пуст”” |
| 20.12.2024, 10:27 | Чтение файла | Богданов Р.А. | Отсутствие файла | Успех: “Ошибка: “Файл не найден”” |
| 20.12.2024, 10:33 | Обработка скобок | Богданов Р.А. | Текст: “((TEXT)” | Успех: “(TEXT)” |
| 20.12.2024, 10:35 | Обработка скобок | Богданов Р.А. | Текст: “((lower) case (letters) (lower) case (letters) (lower) case (letters) (lower) case” | Успех: “(LOWER) case (LETTERS) (LOWER) case (LETTERS) (LOWER) case (LETTERS) (LOWER) case (LETTERS) (LOWER) case” |

# Заключение

В ходе выполнения курсовой работы была разработана программа на языке ассемблера для процессора Intel 8086, предназначенная для обработки текстовых файлов. Программа успешно выполняет поставленную задачу: считывает данные из входного текстового файла, заменяет строчные буквы слов, заключённых в круглые скобки, на прописные, и записывает результат в выходной файл. В процессе разработки были использованы основные функции операционной системы DOS для работы с файлами, такие как открытие, чтение, запись и закрытие файлов. Кроме того, была введена обработка исключительных ситуаций, таких как отсутствие файла, несбалансированные скобки и пустой файл, что обеспечило устойчивость программы к ошибкам.

Разработка программы позволила глубже понять принципы работы микропроцессора Intel 8086 и использовать его возможности на практике. В результате работы была приобретена навык программирования на низком уровне, что является важным навыком для разработчиков программного обеспечения, работающих с аппаратными ресурсами компьютера.

Тестирование программы показало её корректную работу на различных наборах входных данных и подтвердило соответствие программного обеспечения требованиям технического задания. Полученные результаты позволяют заключить, что цель курсовой работы была достигнута, и разработанная программа может быть использована для автоматизации обработки текстовых файлов в соответствии с заданными требованиями.

Список источников

1. Язык ассемблера — ru.wikipedia.org — [Электронный ресурс] — https://ru.wikipedia.org/wiki/Язык_ассемблера — дата обращения: 18.12.2024.
2. Intel 8086 — ru.wikipedia.org — [Электронный ресурс] — https://ru.wikipedia.org/wiki/Intel_8086 — дата обращения: 18.12.2024.
3. Assembler — it.kgsu.ru — [Электронный ресурс] — https://it.kgsu.ru/Assembler/asm0063.html — дата обращения: 18.12.2024.
4. Emu8086 — emu8086.en.lo4d.com — [Электронный ресурс] — https://emu8086.en.lo4d.com/windows — дата обращения: 18.12.2024.
5. Ершов Е.В., д-р техн. наук, проф.; Виноградова Л.Н. и др. Методика и организация самостоятельной работы студентов – Коллектив авторов, ФГБОУ ВПО «Череповецкий государственный университет», 2012. –208 с.

Приложение 1

МИНОБРАНАУКИ РОССИИ
федеральное государственное бюджетное
образовательное учреждение высшего образования
«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий
наименование института (факультета)

Кафедра математического и программного обеспечения
Наименование кафедры

Программирование на ассемблере
Наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ
Зав. кафедрой МПО ЭВМ
д.т.н. _____ Ершов Е.В.
«____» _____2024 г.

# ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ НИЗКОГО УРОВНЯ
Техническое задание на курсовую работу
Листов 6


Руководитель Виноградова Людмила Николаевна
Ф.И.О. преподавателя

Исполнитель
студент 1ПИб-02-3оп-23
группа

Богданов Ренат Алексеевич
Фамилия, имя, отчество


2024 год

Введение

Данная курсовая работа посвящена разработке программы на языке низкого уровня для обработки текстовых файлов. Программа должна заменять буквы слов, стоящих в скобках, прописными.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Программирование на ассемблере», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 1 октября 2024 года.

Наименование темы разработки: «Программирование на языке низкого уровня».

2. Назначение разработки

Основной задачей курсовой работы является освоение на практике материала, полученного в ходе изучения дисциплины «Программирование на ассемблере», а также изучение средств и методов работы с микропроцессором Intel 8086.

3. Требования к программе

3.1. Требования к функциональным характеристикам

Программа должна выполнять следующие функции:

- чтение исходного текстового файла;
- поиск слов, стоящих в скобках;
- замена букв этих слов на прописные (заглавные);
- запись измененного текста обратно в файл или вывод измененного текста на экран.

3.2. Требования к надёжности

Программа должна быть устойчивой к ошибкам ввода данных, например, в случае отсутствия файлов или некорректного формата данных. Для этого необходимо реализовать проверку вводимых данных на корректность, а также предусмотреть обработку исключительных ситуаций, таких как:

- Отсутствие исходного файла.
- Неверный формат данных в файле (например, некорректное расположение скобок).
- Пустой файл или файл без слов в скобках.

При возникновении ошибок программа должна выводить на экран сообщение об ошибке.

3.3. Условия эксплуатации

Условия эксплуатации программного обеспечения должны соответствовать условиям эксплуатации ПК пользователя.

3.4. Требования к составу и параметрам технических средств

Минимальные системные требования:

- процессор с тактовой частотой не менее 1,2 GHz;
- оперативная память 512 Мб или больше;
- свободное место на жёстком диске от 50 Мб;
- процессор не младше, чем 8086;
- клавиатура, мышь и монитор.

3.5. Требования к информационной и программной совместимости.

Код программы написан на языке ассемблера для процессора Intel 8086 с использованием эмулятора микропроцессора EMU8086 на операционной системе Windows XP и новее.

3.6. Требования к маркировке и упаковке

Программа будет распространятся через копирование исходных файлов на USB-флеш-накопителях.

3.7. Требования к транспортированию и хранению

Файлы, требуемые для корректной работы программы, необходимо расположить на USB-флеш-накопителе, либо в внутренней памяти компьютера.

3.8. Специальные требования

Отсутствуют.

4. Требования к программной документации

4.1. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку (далее — РПЗ) с содержанием:

Титульный лист;

Оглавление;

Введение;

1. Описание предметной области;
2. Постановка задачи;
3. Логическое проектирование;
4. Физическое проектирование;
5. Кодирование;
6. Тестирование;
7. Заключение;

Литература;

Приложения; Техническое задание;

Руководство пользователя;

Текст программы.

4.2. Требования к оформлению

Требования к оформлению должны соблюдаться при выполнении работы на протяжении всего времени (в табл. П1.1).

Таблица П1.1

Требования к оформлению

| | |
|---|---|
| Документ | Печать на отдельных листах формата А4 (20х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. |
| Страницы | Ориентация — книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее — по 2 см, левое — 3 см, правое — 2 см. |
| Абзацы | Межстрочный интервал — 1,5, перед и после абзаца — 0. |
| Шрифты | Кегль — 14. В таблицах шрифт 12. Шрифт листинга — 8 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: «Рис.Х. Название В» приложениях: «Рис.П.3. Название» |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) — по центру. В теле таблицы (записи) текстовые значения — выравнены по левому краю, числа, даты — по правому. |

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице П1.2.

Таблица П1.2

Стадии и этапы разработки

| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка о выполнении |
|---|---|---|---|
| Получение задания | 1.10.2024 | Полученное задание | |
| Разработка технического задания | 20.10.2024-24.10.2024 | Оформленное техническое задание | |
| Разработка алгоритма | 28.10.2024 – 3.11.2024 | Готовый алгоритм | |
| Написание программы | 4.11.2024 – 23.12.2024 | Написанная программа | |
| Тестирование программы | 23.11.2024 – 25.12.2024 | Проверенная и отлаженная программа | |
| Написание РПЗ | 20.12.2024-25.12.2024 | Оформленное РПЗ | |

6. Порядок контроля и приёмки

Порядок контроля и приёма представлены в таблице П1.3.

Таблица П1.3

Порядок контроля и приёма

| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приёмке результата контрольного этапа |
|---|---|---|---|
| Технические задание | 21.10.2024-28.10.2024 | Оформленное техническое задание | |
| Теоретическая часть курсовой работы | 1.11.2024-4.11.2024 | Оформленная теоретическая часть | |
| Практическая часть курсовой работы | 3.12.2024-6.12.2024 | Программа | |
| Расчетно-пояснительная записка | 16.12.2024-18.12.2024 | Оформленная РПЗ | |
| Защита курсовой работы | 19.12.2024-23.12.2024 | Получение итоговой оценки за курсовую работу | |

Приложение 2

# Руководство пользователя

## 1. Общие сведения о программе

Программа на языке ассемблера для процессора Intel 8086, которая считывает данные из текстового файла, заменяет строчные буквы слов, заключённых в круглые скобки, на прописные, и записывает результат в другой текстовый файл. Программа также обрабатывает исключения, такие как отсутствие исходного файла, несбалансированные скобки и пустой файл.

## 2. Описание установки

1) Скачайте файл mycode.asm с диска.
2) Скачайте и установите эмулятор EMU8086 с официального сайта.
3) Создайте текстовый файл input.txt с необходимым содержимым.
4) Разместите файл input.txt в той же директории, где находится файл mycode.asm.

## 3. Описание запуска

1) Запустите эмулятор EMU8086.
2) Открой файл программы “mycode.asm”
3) Скомпилируйте программу.

Нажмите на кнопку “emulate” (см. рис. 1), а затем в открывшемся окне кнопку “run” (см. рис. 2) для запуска программы.

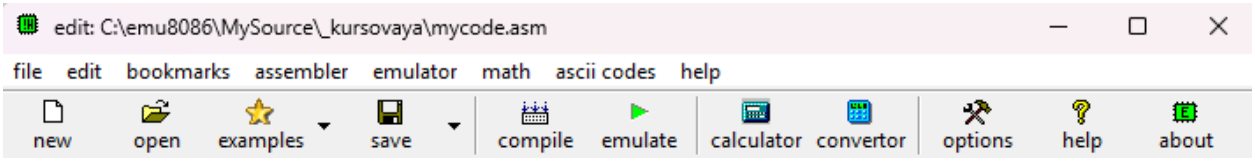


Рис. 1. Панель с кнопкой “emulate”

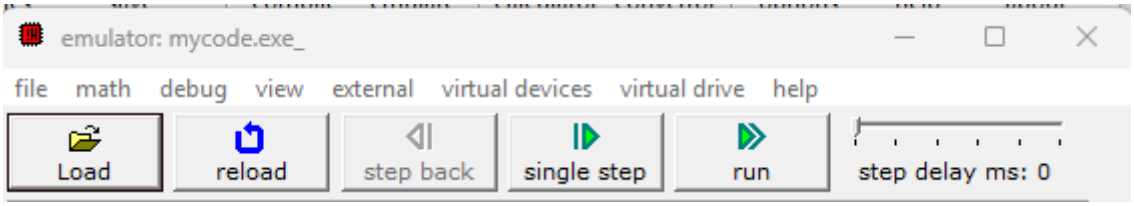


Рис. 2. Панель с кнопкой “run”

4. Инструкция по работе

1) Подготовка входного файла

Убедитесь, что файл “input.txt” находится в той же директории, что и файл “mycode.asm”. Напишите текст, который вы хотите обработать, в файле “input.txt”. Слова, которые необходимо отформатировать должны быть заключены в круглые скобки. Недопустимо заключать скобки внутри скобок и/или вводить не равное количество открывающих и закрывающих скобок. В таком случае программа выдаст ошибку и завершиться.

2) Запуск программы

После запуска программы она автоматически выполнит обработку текста в файле “input.txt”. Программа создаст новый файл “output.txt” с обработанным текстом.

3) Просмотр результата

После завершения работы программы вы увидите отформатированный текст или сообщение об ошибке (если таковая возникнет).

Откройте файл “output.txt” для просмотра изменённого текста.

4) Обработка ошибок

Если программа не сможет найти файл “input.txt”, она выведет сообщение “File not found”.

Если в файле присутствуют несбалансированные скобки, программа выведет сообщение “ Unbalanced parentheses in file”.

Если файл пуст, программа выведет сообщение “ File is empty”.

5. Пример работы

Задача: отформатировать текст “The (quick) (brown) fox jumps over the (lazy) dog”. Для этого поместим файл input.txt в директорию с файлом программы “mycode.asm” (см. рис. 3).

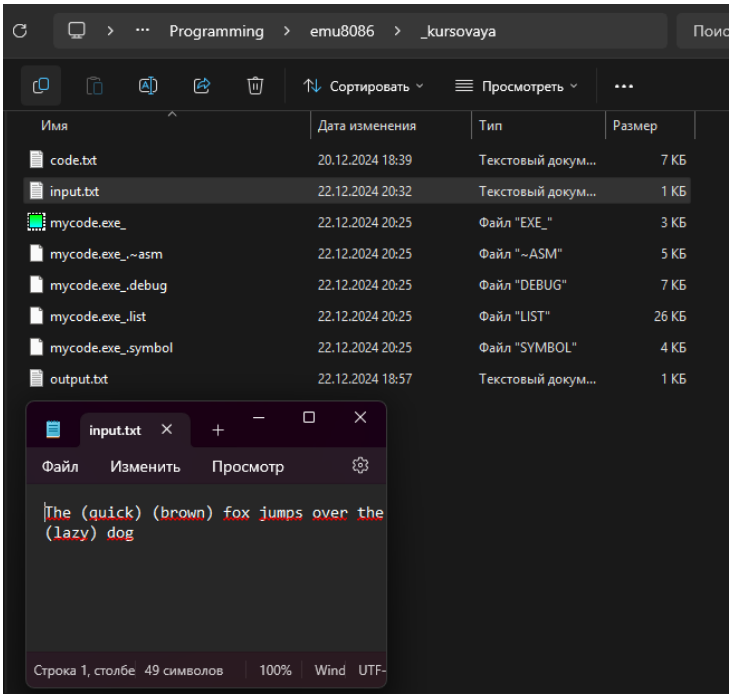


Рис. 3. Размещение файла с текстом в директории с программой

Открой программу EMU8086. В “приветственном” окне откроем файл (см. рис. 4). Для этого нажимаем кнопку “recent files”, затем выбираем “other” и в окне проводника находим нужный файл.

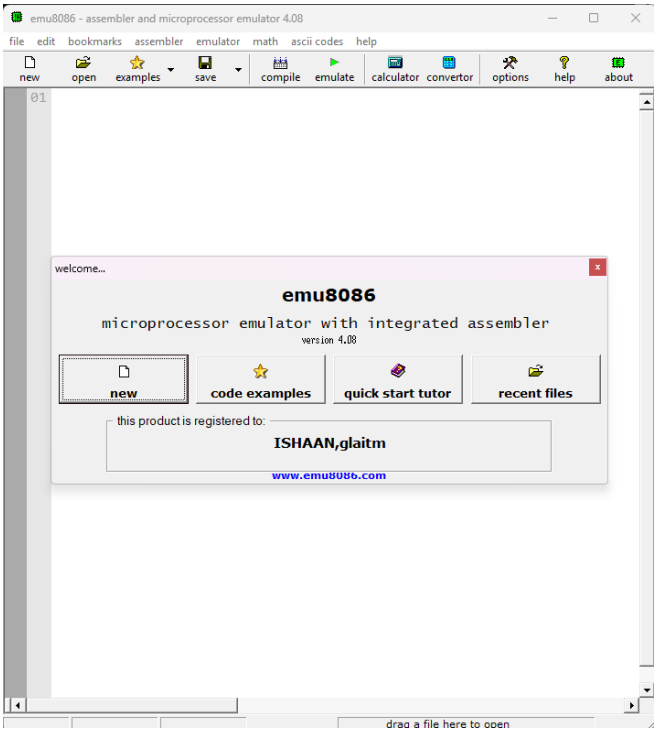


Рис. 4. Интерфейс программы EMU8086

На экране отобразится интерфейс программы. Нажмём кнопку “emulate”, затем “run”. Программа начнёт работу, выведет текст из файла “input.txt”, проведёт обработку текста, создаст файл “output.txt”, запишет в него отформатированный текст и выведет его на экран (см. рис. 5). После этого программа закончит свою работу. Отформатированный текст можно найти в файле “output.txt” в директории, где хранится “input.txt” и программа.

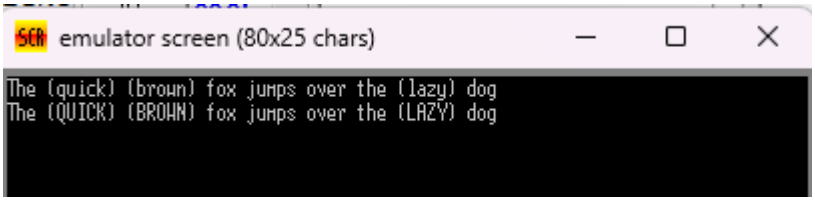


Рис. 5. Результат работы программы

Приложение 3

# Код программы

## Файл “mycode.asm”

```
.model small
.stack 100h
.data
inFileName db 'input.txt', 0
outFileName db 'output.txt', 0
buffer db 512 dup(0)
tempBuffer db 512 dup(0) ; Временный буфер для преобразования
bytesRead dw ?
inFileHandle dw ?
outFileHandle dw ?
insideParentheses db 0 ; Флаг, указывающий, внутри ли скобок
errorMessage db 'Error: $'
fileNotFoundMsg db 'Файл не найден$', 0
readErrorMsg db 'Ошибка при чтении файла$', 0
writeErrorMsg db 'Ошибка при записи файла$', 0
closeErrorMsg db 'Ошибка при закрытии файла$', 0
unbalancedParenthesesMsg db 'Несбалансированные скобки в файле$', 0
emptyFileMsg db 'Файл пуст$', 0
newline db 13, 10, '$' ; Перенос строки
.code
main proc
mov ax, @data
mov ds, ax

; Открытие входного файла
mov ah, 3Dh
mov al, 0
lea dx, inFileName
int 21h
jc fileNotFound
mov inFileHandle, ax

; Чтение входного файла для проверки на пустоту
mov ah, 3Fh
mov bx, inFileHandle
mov cx, 1 ; Читаем только один байт для проверки пустоты
lea dx, buffer
int 21h
jc readError
mov bytesRead, ax
cmp ax, 0
je emptyFile

; Сохраняем первый байт
mov al, buffer[0]

; Возвращаем указатель файла в начало
mov ah, 42h
mov bx, inFileHandle
mov al, 0 ; Устанавливаем указатель файла в начало
xor cx, cx
xor dx, dx
int 21h
jc readError

; Создание выходного файла
mov ah, 3Ch
mov cx, 0
lea dx, outFileName
int 21h
jc writeError
mov outFileHandle, ax

; Чтение входного файла и вывод на экран до конца
readInFileLoop:
mov ah, 3Fh
mov bx, inFileHandle
mov cx, 512
lea dx, buffer
int 21h
jc readError
mov bytesRead, ax
cmp ax, 0
je closeInFile

; Вывод прочитанного буфера на экран
mov ah, 40h
mov bx, 1 ; Дескриптор файла 1 -
```

```
    стандартный вывод
mov cx, bytesRead
lea dx, buffer
int 21h
jc writeError

jmp readInFileLoop

closeInFile:
; Закрытие входного файла
mov ah, 3Eh
mov bx, inFileHandle
int 21h
jc closeError

; Перенос строки
mov ah, 09h
lea dx, newline
int 21h

; Открытие входного файла снова для
    обработки
mov ah, 3Dh
mov al, 0
lea dx, inFileName
int 21h
jc fileNotFound
mov inFileHandle, ax

; Восстанавливаем первый байт
mov buffer[0], al

; Чтение входного файла для
    обработки
readProcessLoop:
mov ah, 3Fh
mov bx, inFileHandle
    mov cx, 512
    lea dx, buffer
    int 21h
    jc readError
    mov bytesRead, ax
    cmp ax, 0
    je closeInFileForProcessing

    ; Обработка символов
    xor si, si
    xor di, di
    processLoop:
    cmp si, bytesRead
    jge writeBuffer
    mov al, buffer[si]
    inc si

    ; Проверка на символ скобки
    cmp al, '('
    je openParentheses
    cmp al, ')'
    je closeParentheses

    ; Преобразование букв в верхний
    регистр, если внутри скобок
    cmp insideParentheses, 1
    jne notInsideParentheses
    cmp al, 'a'
    jb notLetter
    cmp al, 'z'
    ja notLetter
    sub al, 20h ; Преобразование в
    верхний регистр
    notLetter:
    jmp storeChar

    openParentheses:
    mov insideParentheses, 1
    jmp storeChar

    closeParentheses:
    cmp insideParentheses, 1
    jne unbalancedParentheses
    mov insideParentheses, 0
    jmp storeChar

    notInsideParentheses:
    ; Иначе ничего не делаем

    storeChar:
    mov tempBuffer[di], al
    inc di
    jmp processLoop

    writeBuffer:
    ; Запись преобразованного буфера
    в выходной файл
    mov ah, 40h
    mov bx, outFileHandle
    mov cx, di
    lea dx, tempBuffer
    int 21h
    jc writeError

    jmp readProcessLoop

    closeInFileForProcessing:
```

```
    ; Проверка на незакрытые скобки
    cmp insideParentheses, 1
    je unbalancedParentheses

    ; Закрытие входного файла
    mov ah, 3Eh
    mov bx, inFileHandle
    int 21h
    jc closeError

    ; Закрытие выходного файла
    mov ah, 3Eh
    mov bx, outFileHandle
    int 21h
    jc closeError

    ; Открытие выходного файла для чтения
    mov ah, 3Dh
    mov al, 0
    lea dx, outFileName
    int 21h
    jc fileNotFound
    mov outFileHandle, ax

    ; Чтение выходного файла и вывод на экран после обработки
    readOutFileLoop:
    mov ah, 3Fh
    mov bx, outFileHandle
    mov cx, 512
    lea dx, buffer
    int 21h
    jc readError
    mov bytesRead, ax
    cmp ax, 0
    je closeOutFile

    ; Вывод прочитанного буфера на экран
    mov ah, 40h
    mov bx, 1 ; Дескриптор файла 1 - стандартный вывод
    mov cx, bytesRead
    lea dx, buffer
    int 21h
    jc writeError

    jmp readOutFileLoop

    closeOutFile:
    ; Закрытие выходного файла
    mov ah, 3Eh
    mov bx, outFileHandle
    int 21h
    jc closeError

    ; Завершение программы
    mov ax, 4C00h
    int 21h

    fileNotFound:
    lea dx, fileNotFoundMsg
    jmp showError

    readError:
    lea dx, readErrorMsg
    jmp showError

    writeError:
    lea dx, writeErrorMsg
    jmp showError

    closeError:
    lea dx, closeErrorMsg
    jmp showError

    unbalancedParentheses:
    lea dx, unbalancedParenthesesMsg
    jmp showError

    emptyFile:
    lea dx, emptyFileMsg
    jmp showError

    showError:
    mov ah, 09h
    int 21h
    mov ax, 4C01h
    int 21h

    main endp
    end main
```