минобрнауки россии

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет)

Кафедра

Институт информационных технологий

Математическое и программное обеспечение ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине

на тему

Программирование на ассемблере

Программирование на языке низкого уровня

Выполнил студент группы

1ПИб-02-2оп-23

*группа*

Направления подготовки (специальности)

09.03.04 Программная инженерия

*шифр, наименование*

Кринкин Олег Алексеевич

*фамилия, имя, отчество*

Руководитель

Виноградова Людмила Николаевна

*фамилия, имя, отчество*

Доцент, кандидат наук

*должность*

Дата представления работы

«       »     2024 г.

Заключение о допуске к защите

Оценка

Количество баллов

Подпись преподавателя

Череповец, 2024

*год*

Оглавление

[1. Введение 3](#_Toc186190002)

[2. Описание предметной области 4](#_Toc186190003)

[3. Постановка задачи 7](#_Toc186190004)

[4. Логическое проектирование 8](#_Toc186190005)

[5. Физическое проектирование 10](#_Toc186190006)

[6. Кодирование 11](#_Toc186190007)

[7. Тестирование 13](#_Toc186190008)

[8. Заключение 14](#_Toc186190009)

[Список литературы 15](#_Toc186190010)

[Приложение 1. Техническое задание 16](#_Toc186190011)

[Приложение 2. Руководство пользователя 23](#_Toc186190012)

[Приложение 3. Текст программы 26](#_Toc186190013)

# 1. Введение

В 1978 году компанией Intel был разработан 16-разрядный микропроцессор Intel 8086. С тех пор все последующие процессоры компании, а также совместимые с ними процессоры могут исполнять код, написанный для i8086. Всё это осуществимо благодаря тому, что современные процессоры поддерживают те машинные команды и набор регистров, что имел i8086 годами ранее. Поэтому для того, чтобы получить некоторые знания о работе этих процессоров, можно прибегнуть к изучению и разработке программ на языке ассемблера для Intel 8086. Эти программы всё также можно будет использовать на современных процессорах, но при этом они будут содержать лишь базовые машинные команды, что упростит понимание происходящих в процессоре процессов, т.к. машинные команды и устройство процессоров стали намного более комплексным спустя время.

Целью курсовой работы является разработка программы на языке ассемблера для процессора Intel 8086 с использованием функций операционной системы MS-DOS по работе с файлами. Необходимо чтобы программа обрабатывала исходный файл, а затем создавала новый файл, в котором будет находится содержимое исходного файла с поменяными местами соседними строчками.

.

# 2. Описание предметной области

Язык ассемблера – представление машинных команд процессора в виде, доступном для чтения человеком. Язык ассемблера считается языком программирования низкого уровня, в противовес высокоуровневым языкам, не привязанным к конкретной реализации вычислительной системы. Программы, написанные на языке ассемблера однозначным образом, переводятся в инструкции конкретного процессора и в большинстве случаев не могут быть перенесены без значительных изменений для запуска на машине с другой системой команд. Ассемблером называется программа, преобразующая код на языке ассемблера в машинный код. [2]

Функции MS-DOS для работы с файлами

Функции MS-DOS – это различные сервисные возможности, которые операционная система MS-DOS предоставляет пользовательским программам через прерывание DOS 21H и некоторые другие прерывания. Функции MS-DOS облегчают выполнение таких стандартных операций, как ввод с клавиатуры, вывод на экран, работа с файлами, операции с памятью и т. д. Каждая функция MS-DOS имеет свой номер и набор параметров. Для выполнения функции необходимо занести её номер в регистр AH, а в другие регистры занести необходимые параметры и вызвать прерывание DOS командой INT 21H. Некоторые функции возвращают результаты своего выполнения в различных регистрах или ячейках памяти. Некоторые функции имеют возможность возвращать состояние своего выполнения, т. е. выполнилась ли функция успешно или в процессе ее выполнения возникла какая-либо ошибка.

Функция DOS 3Ch

Назначение: создание нового файла в файловой системе FAT или перезапись старого пустым. В качестве аргументов функция принимает значение файлового аттрибута в регистре CX и имя файла в формате ASCIIZ строки в регистровой паре DS:DX. По завершению функция записывает идентификатор файла для последующей работы с ним в регистр AX или код ошибки если файл создать не удалось. [3]

Аттрибуты для файла могут быть следующими:

* 0 – только для чтения
* 1 – скрытый
* 2 – системный
* 3 – имя раздела

Функция DOS 3Dh

Назначение: открытие существующего файла. В качестве аргументов функция принимает путь к файлу в регистровой паре DS:DX и двоичный код режима доступа, в котором необходимо открыть файл. Возвращает идентификатор открытого файла в регистре AX для последующего использования. В случае возникновения ошибки возвращает её код в регистр AX.

Файл может иметь следующие режимы доступа:

* 000 – режим чтения
* 001 – режим записи
* 010 – режим чтения и записи

В MS-DOS 3.0 тажке были добавлены запрещающие режимы, имеющие двоичные номера с 4 до 6. [4]

Функция DOS 3Eh

Назначение: функция записывает все оставшиеся данные, связанные с указанным файлом на диск, закрывает файл и освобождает идентификатор файла для последующего использования. В качестве аргументов функция принимает идентификатор открытого файла в регистре BX. В случае ошибки возвращает код 06h в регистр AX. [5]

Функция DOS 3Fh

Назначение: чтение данных с файла или устройства. В качестве аргументов принимает идентификатор файла в регистре BX, количество байт, которое нужно прочесть в регистре CX и байтовый буфер в регистровой паре DS:DX. Записывает данные в байтовый буфер и возвращает количество прочитанных байт в регистр AX. В случае возникновения ошибки вернёт её код в регистр AX. [6]

Функция DOS 40h

Назначение: запись данных в файл или на устройство. В качестве параметров функция принимает идентификатор файла в регистре BX, количество байт, которое необходимо записать в регистре CX и байтовый буфер с данными в регистровой паре DS:DX. По завершению функция вносит в регистр AX количество записанных байт, а в случае возникновения ошибки вносит в него же код ошибки. [7]

# 3. Постановка задачи

Целью курсовой работы является разработка алгоритма обработки файла при помощи следующей последовательности функций операционной системы MS-DOS:

1. Открытие исходного файла (функция 3DH);
2. Создание нового файла, куда будут помещаться результаты обработки исходного файла (функция 3CH);
3. Чтение исходного файла (функция 3FH);
4. Обработка прочитанных данных;
5. Запись обработанных данных в созданный файл (функция 40H);
6. Закрытие файла с результатами обработки (функция 3EH);
7. Закрытие исходного файла (функция 3EH).

Алгоритм должен считать исходный файл и создать новый файл, поменяв местами соседние строчки исходного файла.

# 4. Логическое проектирование

Для решения поставленной задачи был разработан следующий алгоритм:

1. Программа, открывает исходный файл для чтения и вносит его идентификатор в переменную в памяти;
2. Создаётся и закрывается новый файл с именем output.txt;
3. Содержимое открытого исходного файла считывается в байтовый буфер размером 512 байт, после чего файл закрывается, а количество считанных байт вносится в переменную в памяти;
4. Созданный новый файл открывается для записи;
5. Из файлового буфера рассчитывается размер первой и второй строки, которые нужно записать. Размер строки равен количеству символов до символа конца строки и его самого;
6. Из файлового буфера в новый файл, учитывая ранее рассчитанные размеры записывается вторая строка, а затем первая, после чего программа возвращается к шагу 5, при этом перед переходом проверяется не закончился ли файл, а из буфера считываются следующие две строки;
7. Открытый для записи файл закрывается.

На рис. 1 представлена блок-схема описанного алгоритма:

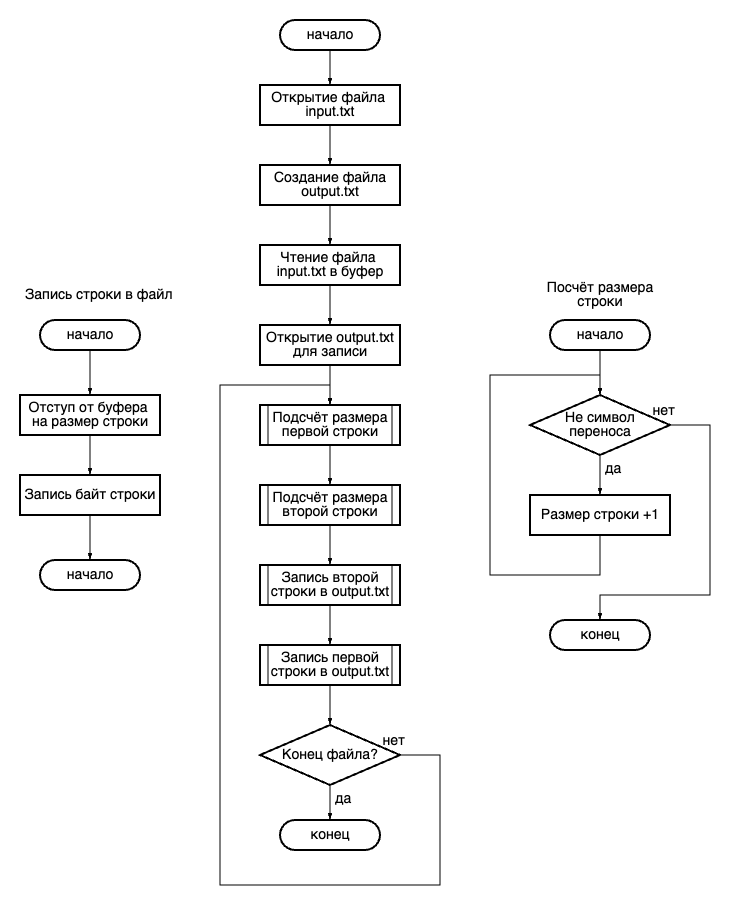


Рис. 1. Блок-схема описанного алгоритма

# 5. Физическое проектирование

Для реализации алгоритма в виде программы необходимо использование различных переменных и функций. Все использованние переменные описаны в таблице 1, а функции в таблице 2.

Таблица 1.

Использованние переменные

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Файловый буфер | fbuff | Byte |
| Количество данных в буфере | fbuff\_t | Word |
| Размер строки в файле | chunk\_t | Word |
| Счётчик цикла | loop\_i | Word |
| Идентификатор исходного файла | in\_id | Word |
| Номер текущей строки | l\_index | Byte |
| Имя исходного файла | input | Byte (ASCIIZ) |
| Имя нового файла | output | Byte (ASCIIZ) |

Таблица 2.

Использованные функции

|  |  |  |
| --- | --- | --- |
| Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| fnew | filename (DX) | Создание нового файла |
| fclose | file\_id (BX) | Закрытие файла |
| fopen | filename (DX), access (AL) | Открытие файла |
| fread | file\_id (BX), size (CX) | Чтение n байт из файла |
| fwrite | file\_id (BX), offset (SI), size (CX) | Запись n байт в файл |
| process |  | Запись прочитанных данных в файл с перестановкой строк |

# 6. Кодирование

Исходя из описанного алгоритма на языке ассемблера для процессора Intel 8086 был написана программа. Исходный текст программы содержит комментарии на английском языке по причине плохой поддержки кодировки Windows 1251 используемой средой разработки. Исходный текст написанной программы выглядит следующим образом:

; 3.7 (swap lines in file)

code SEGMENT

ORG 100h

JMP main

fnew: ;(DX: filename) -> AX: file\_id

MOV CX, 00h

MOV AH, 3Ch

INT 21h

RET

fclose: ;(BX: file\_id) -> None

MOV AH, 3Eh

INT 21h

RET

fopen: ;(DX: filename, AL: access) -> AX: file\_id

MOV AH, 3Dh

INT 21h

RET

fread: ;(BX: file\_id, CX: size) -> AX: readed bytes count

MOV AH, 3Fh

MOV DX, OFFSET fbuff

INT 21h

RET

fwrite: ;(BX: file\_id, SI: offset, CX: size) -> AX: err

MOV AH, 40h

MOV DX, SI

INT 21h

RET

process: ;() -> None

MOV SI, OFFSET fbuff

MOV sline\_t, 0

MOV fline\_t, 0

ADD SI, foffset

L1:MOV BX, [SI]

INC SI

INC fline\_t ;first line size

CMP BL, 0Ah

JE L2

JMP L1

L2:MOV BX, [SI]

CMP BL, 0

JE WRT

INC SI

INC sline\_t ;second line size

CMP BL, 0Ah

JE WRT

JMP L2

WRT:MOV BX, out\_id

MOV SI, OFFSET fbuff

ADD SI, foffset

ADD SI, fline\_t

MOV CX, sline\_t

CALL fwrite ;write second line

MOV SI, OFFSET fbuff

ADD SI, foffset

MOV CX, fline\_t

CALL fwrite ;write first line

ADD SI, fline\_t

ADD SI, sline\_t

SUB SI, OFFSET fbuff

MOV AX, fline\_t

ADD AX, sline\_t

ADD foffset, AX

CMP SI, fbuff\_t ;end of file?

JNE process

RET

main:

;opening input file

MOV DX, OFFSET input

MOV AL, 000

CALL fopen

MOV in\_id, AX

;creating output file

MOV DX, OFFSET output

CALL fnew

MOV BX, AX

CALL fclose

;read opened file

MOV BX, in\_id

MOV CX, 512

CALL fread

MOV fbuff\_t, AX

MOV BX, in\_id

CALL fclose

;open output file

MOV DX, OFFSET output

MOV AL, 001

CALL fopen

;write to output file

MOV out\_id, AX

CALL process

;close output file

CALL fclose

RET

fbuff DB 512 DUP (?) ;file buffer

fbuff\_t DW ? ;fbuffer usage

foffset DW ? ;process offset

fline\_t DW ? ;first line size

sline\_t DW ? ;second line size

in\_id DW ? ;input file id

out\_id DW ? ;output file id

input DB 'input.txt',0 ;input filename

output DB 'output.txt',0 ;output filename

code ENDS

END start

# 7. Тестирование

Результаты и этапы тестирования разработанной программы представлены в таблице 6.

Таблица 6.

Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Дата тестирования | Тестировщик | Описание теста | Результат теста |
| 18.12.24 | Кринкин О. А. | Создание файла | Файл не создан |
| 18.12.24 | Кринкин О. А. | Чтение файла | Отсутствие ошибок |
| 18.12.24 | Кринкин О. А. | Запись в файл | Отсутствие ошибок |
| 18.12.24 | Кринкин О. А. | Тестирование обработки | «Артефакты» при записи |
| 18.12.24 | Кринкин О. А. | Тестирование без пустой строки | Нулевой символ в строке |
| 26.12.24 | Кринкин О. А. | Нечётное кол-во строк в тексте | Бесконечное выполение |
| 26.12.24 | Кринкин О. А. | Комплексное тестирование | Отсутствие ошибок |

Тестирование проходило на следующем входном файле:

|  |
| --- |
| input.txt |
| Some funky line with number 1, because i am baka  Some funky line with number 2, because vbihebjkfvcekj  Some funky line with number 3, because something happens  Some funky line with number 4, because kto chto  Some funky line with number 5, because assembler zavtra  Some funky line with number 6, aaaaaaaaaaaah  Some funky line with number 7, uhh  Some funky line with number 8, because i am oleg  Some funky line with number 9, because 123456789009876543211234567890  Some funky line with number 0, because hihihiiihihihi |

Ожидаемый результат обработки данных выглядит следующим образом:

|  |
| --- |
| output.txt |
| Some funky line with number 2, because vbihebjkfvcekj  Some funky line with number 1, because i am baka  Some funky line with number 4, because kto chto  Some funky line with number 3, because something happens  Some funky line with number 6, aaaaaaaaaaaah  Some funky line with number 5, because assembler zavtra  Some funky line with number 8, because i am oleg  Some funky line with number 7, uhh  Some funky line with number 0, because hihihiiihi  Some funky line with number 9, because 123456789009876543211234567890 |

# 8. Заключение

В результате выполнения курсовой работы была написана программа на языке ассемблера для процессора Intel 8086 с использованием функций, предоставляемых операционной системой MS-DOS. Разработанная программа обрабатывает исходный файл и создаёт на его основе новый, поменяв соседние строки местами.

Также в ходе выполнения были изучены некоторые аспекты работы аппаратного и программного обеспечения Intel 8086 и MS-DOS, а также приобретён опыт написания комплексных программ используя язык ассемблера.

# Список литературы

1. 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.;
2. «Язык ассемблера» - Википедия [электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Язык\_ассемблера [свободный]. Дата обращения: 20.12.2024;
3. “Int 21h Function 3Ch“ – Assembly Language Help [электронный ресурс]. Режим доступа: https://fragglet.github.io/dos-help-files/alang.hlp/x\_at\_L82b9.html [открытый]. Дата обращения: 20.12.2024;
4. “Int 21h Function 3Dh” - Assembly Language Help [электронный ресурс]. Режим доступа: https://fragglet.github.io/dos-help-files/alang.hlp/x\_at\_L82bb.html [открытый]. Дата обращения: 20.12.2024;
5. “Int 21h Function 3Eh” - Assembly Language Help [электронный ресурс]. Режим доступа: https://fragglet.github.io/dos-help-files/alang.hlp/x\_at\_L82bd.html [открытый]. Дата обращения: 20.12.2024;
6. “Int 21h Function 3Fh” - Assembly Language Help [электронный ресурс]. Режим доступа: https://fragglet.github.io/dos-help-files/alang.hlp/x\_at\_L82bf.html [открытый]. Дата обращения: 20.12.2024;
7. “Int 21h Function 40h” - Assembly Language Help [электронный ресурс]. Режим доступа: https://fragglet.github.io/dos-help-files/alang.hlp/x\_at\_L82c1.html [открытый]. Дата обращения: 20.12.2024.

# Приложение 1.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего профессионального образования «ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

                                    Институт информационных технологий

*наименование института (факультета)*

                          Математическое и программное обеспечение ЭВМ

*наименование кафедры*

                                        Программирование на ассемблере

*наименование дисциплины в соответствии с учебным планом*

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ,

д. т. н., профессор                    Ершов Е. В.

«      »                       2024 г.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ НИЗКОГО УРОВНЯ

Техническое задание на курсовую работу

Листов 7

|  |  |
| --- | --- |
| Руководитель | Виноградова Л.Н. |
|  | *ФИО преподавателя* |
| Исполнитель |  |
| студент | 1ПИб-02-2оп-23 |
|  | *группа* |
|  | Кринкин О. А. |
|  | *Фамилия, имя, отчество* |

Введение

Курсовая работа направлена на разработку программы на языке программирования низкого уровня.

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: 14 октября 2024 года.

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

2. Назначение разработки

Основной задачей курсовой работы является проектирование и разработка программы, меняющей в исходном файле соседние строчки местами, а также изучение средств и методов работы с микропроцессором Intel 8086.

3. Требования к программе

3.1. Требования к функциональным характеристикам

Программа должна обрабатывать исходный текстовый файл с использованием функций DOS и формировать новый файл с результатами обработки исходного файла.

​Имя исходного и обработанного файлов задать в программе в виде ASCIIZ-строк.

​При обработке файла использовать следующую последовательность вызова функций DOS:

1. открытие исходного файла (функция 3DH);

2. создание нового файла, куда будут помещаться результаты обработки исходного файла (функция 3CH);

3. чтение исходного файла (функция 3FH);

4. замена прочитанных строк местами;

5. запись обработанных данных в созданный файл (функция 40H);

6. закрытие файла с результатами обработки (функция 3EH);

7. закрытие исходного файла (функция 3EH).

3.2. Требования к надежности

Для чтения данных из исходного файла может быть использован буфер, определенный в программе как массив байт или слов. Рекомендуемый размер буфера – 512 байт (256 слов).

Размер исходного файла для обработки должен быть меньшим или равным размеру буфера. Текст в файле не должен быть менее 5 строк, в строке не менее 10 слов, текст может быть как на русском языке, так и на английском.

3.3. Условия эксплуатации

Соответствуют условиям эксплуатации используемого пользователем технического средства.

3.4. Требования к составу и параметрам технических средств

Минимальные требования к техническим характеристикам компьютера для работы программы:

* процессор: Intel-совместимый, с тактовой частотой не менее 1 ГГц;
* ОЗУ: не менее 1 ГБ для 32-разрядной системы или 2 ГБ для 64-разрядной системы;
* место на жестком диске: не менее 1 ГБ свободного пространства;
* видеоадаптер: с поддержкой DirectX 9 с размером видеобуфера не менее 32 МБ;
* экран: с разрешающей способностью не менее 800 x 600 точек.

3.5. Требования к информационной и программной совместимости

Разработанная программа должна быть совместима с эмулятором процессора Intel 8086 – emu8086 и не вызывать ошибок в его работе.

3.6. Требования к маркировке и упаковке

Носитель информации или его упаковка должны иметь обозначение, идентифицирующее нахождение на носителе разработанной программы, а также её документации.

3.7. Требования к транспортированию и хранению

Исходный код и программа поставляются на лазерном диске, а программная документация доступна в электронном и печатном форматах. Требования по транспортировкее и хранению соответствуют требованиям хранения указанных для используемых носителей информации.

3.8. Специальные требования

Не предъявляются.

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;

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

Список литературы;

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. |

6. Стадии и этапы разработки

Стадии и этапы разработки представлены в табл. П1.2.

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результаты выполнения | Отметка о выполнении |
| Получение задания | 14.10.2024 | Полученное задание | Выполнено |
| Разработка технического задания | 21.10.2024 | Оформленное техническое задание | Выполнено |
| Разработка алгоритма | 18.12.2024 | Алгоритм программы | Выполнено |
| Написание программы | 18.12.2024 | Программа на языке ассемблера | Выполнено |
| Тестирование | 19.12.2024 | Успешное тестирование программы | Выполнено |
| Оформление сопроводительной документации | 20.12.2024 | Оформленная сопроводительная документация | Выполнено |

7. Порядок контроля и приемки

Порядок контроля и приема представлены в табл. П1.2.

Таблица П1.2

Порядок контроля и приема

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Сдача технического задания | 22.10.2024 | Согласованное техническое задание |  |
| Сдача расчетно-пояснительной записки | 20.12.2024 | Согласованная расчетно-пояснительная записка |  |
| Сдача курсовой работы | 27.12.2024 | Оценка за курсовую работу |  |

# Приложение 2.

# Руководство пользователя

1. Общие сведения о программе

Файл исходного текста программы имеет наименование “course.asm”. После компиляции имеет расширение .com, а при запуске считывает данные из файла input.txt рядом с исполняемым файлом, создаёт файл output.txt, в который заносится содержимое исходного файла, но при этом меняется положение соседних строчек местами.

2. Описание установки

Разработанная програма не требует установки, но требует на устройстве установленных эмулятора emu8086 или операционной системы MS-DOS.

3. Описание запуска

Для работы с программой в эмуляторе emu8086 необходимо открыть файл course.asm при помощи эмулятора. Для этого если файлы .asm ассоциированы с элулятором дважды щёлкнуть на файл или в главном окне эмулятора выбрать пункт “Open” и выбрать файл в открывшемся окне. В окне среды разработки эмулятора должен отобразится иссходный текст разработанной программы (рис. П2.1):

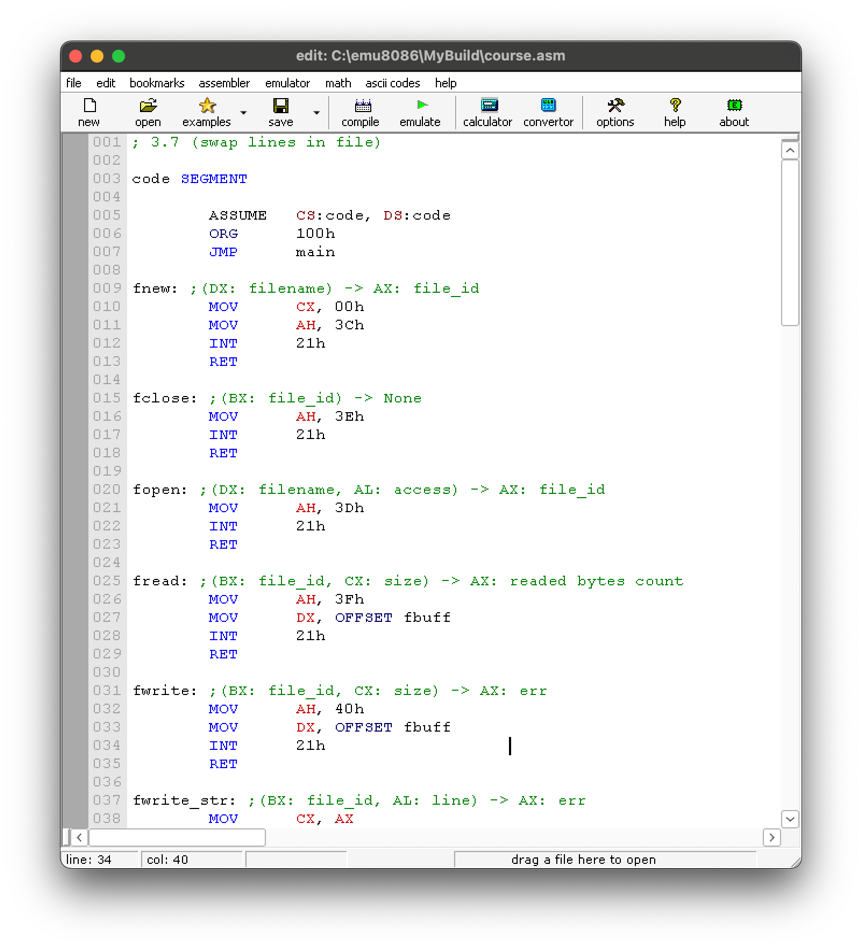


Рис. П2.2. Окно эмулятора с исходным текстом

Для компиляции и последующего запуска программы необходимо нажать кнопку «emulate» на верхней панели интерфейса эмулятора (рис. П2.2):

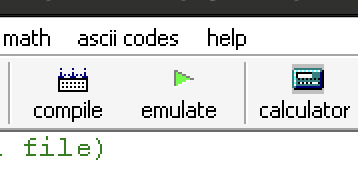


Рис. П2.3. Кнопка "emulate"

Среди открывшихся отладочных окон появится кнопка «run», которая отвечает за запуск цикла эмуляции и выполнения программы (рис. П2.3). Перез запуском по пути нахождения файла с исходным текстом или по пути компиляции эмулятора (C:\emu8086\MyBuild) необходимо разместить файл input.txt.



Рис. П2.4. Кнопка "run"

После нажатия на кнопку “run” будет запущен цикл эмуляции и выполнения программы, в процессе которого будет считан и обработан файл input.txt, создан файл output.txt в который будут записаны данные.

При помощи интерфейса эмулятора по необходимости можно приостановить или пошагово продолжать цикл эмуляции (рис. П2.4).

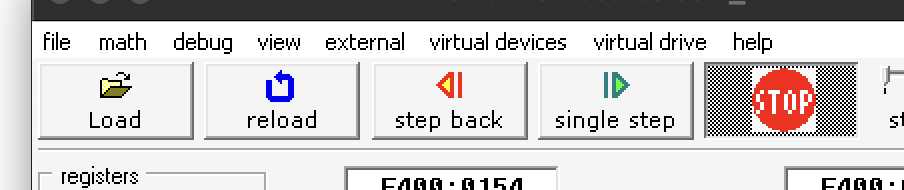


Рис. П2.5. Интерфейс управления эмуляцией

Для того чтобы изменить имя файлов имеется возможность изменить его в исходном тексте на строках 105 и 106 для исходного файла и создаваемого файла соответственно (рис. П2.5)

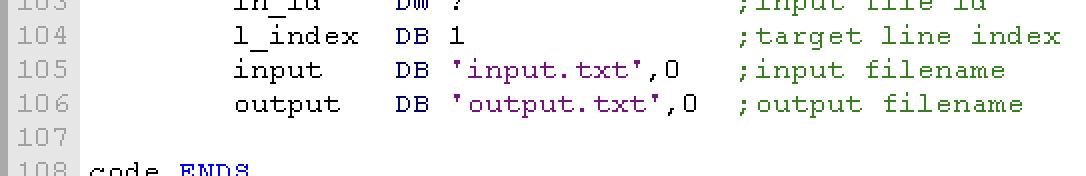


Рис. П2.6. Переменные с именами файлов

# Приложение 3.

# Текст программы

Файл «course.asm»

; 3.7 (swap lines in file)

code SEGMENT

ORG 100h

JMP main

fnew: ;(DX: filename) -> AX: file\_id

MOV CX, 00h

MOV AH, 3Ch

INT 21h

RET

fclose: ;(BX: file\_id) -> None

MOV AH, 3Eh

INT 21h

RET

fopen: ;(DX: filename, AL: access) -> AX: file\_id

MOV AH, 3Dh

INT 21h

RET

fread: ;(BX: file\_id, CX: size) -> AX: readed bytes count

MOV AH, 3Fh

MOV DX, OFFSET fbuff

INT 21h

RET

fwrite: ;(BX: file\_id, SI: offset, CX: size) -> AX: err

MOV AH, 40h

MOV DX, SI

INT 21h

RET

process: ;() -> None

MOV SI, OFFSET fbuff

MOV sline\_t, 0

MOV fline\_t, 0

ADD SI, foffset

L1:MOV BX, [SI]

INC SI

INC fline\_t ;first line size

CMP BL, 0Ah

JE L2

JMP L1

L2:MOV BX, [SI]

CMP BL, 0

JE WRT

INC SI

INC sline\_t ;second line size

CMP BL, 0Ah

JE WRT

JMP L2

WRT:MOV BX, out\_id

MOV SI, OFFSET fbuff

ADD SI, foffset

ADD SI, fline\_t

MOV CX, sline\_t

CALL fwrite ;write second line

MOV SI, OFFSET fbuff

ADD SI, foffset

MOV CX, fline\_t

CALL fwrite ;write first line

ADD SI, fline\_t

ADD SI, sline\_t

SUB SI, OFFSET fbuff

MOV AX, fline\_t

ADD AX, sline\_t

ADD foffset, AX

CMP SI, fbuff\_t ;end of file?

JNE process

RET

main:

;opening input file

MOV DX, OFFSET input

MOV AL, 000

CALL fopen

MOV in\_id, AX

;creating output file

MOV DX, OFFSET output

CALL fnew

MOV BX, AX

CALL fclose

;read opened file

MOV BX, in\_id

MOV CX, 512

CALL fread

MOV fbuff\_t, AX

MOV BX, in\_id

CALL fclose

;open output file

MOV DX, OFFSET output

MOV AL, 001

CALL fopen

;write to output file

MOV out\_id, AX

CALL process

;close output file

CALL fclose

RET

fbuff DB 512 DUP (?) ;file buffer

fbuff\_t DW ? ;fbuffer usage

foffset DW ? ;process offset

fline\_t DW ? ;first line size

sline\_t DW ? ;second line size

in\_id DW ? ;input file id

out\_id DW ? ;output file id

input DB 'input.txt',0 ;input filename

output DB 'output.txt',0 ;output filename

code ENDS

END start