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-02-2оп-22
направление подготовки (специальности)
09.03.04 Программная инженерия
шифр, наименование
Зернов Владислав Александрович
фамилия, имя, отчество

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

Дата представления работы
«_____»_____20__ г.

Заключение о допуске к защите

Оценка _____, _____
количество баллов
Подпись преподавателя_____

Череповец, 2024 год

Аннотация

Курсовую работу по дисциплине «Программирование на ассемблере» на тему «Программирование на языке низкого уровня» выполнил студент группы 1ПИб-02-2оп-22 ИИТ Зернов В.А.

Тема: Программирование на языке низкого уровня.

Курсовая работа была создана на основе Технического задания, описанного в Приложении 1.

Курсовая работа посвящена программированию на языке низкого уровня Assembler. Продуктом курсовой работы будет являться программа, подсчитывающая количество структур, подходящих по условию.

Курсовая включает в себя разделы: аннотация; оглавление; введение; основная часть; заключение; список литературы. Также вместе с курсовой работой идут приложения: техническое задание; руководство пользователя; текст программы.

Оглавление

Введение

Данная курсовая работа фокусируется на создании программы для подсчета количества структур, подходящих под условие, что является актуальным в наше время.

Основной целью курсовой работы является разработка программы на низком языке программирования Assembler, которая находит самый старый год и номер выпуска этого журнала.

В данной работе будет проведен анализ языка программирования низкого уровня и его инструментов, используемых для обработки структур и массивов, с особым вниманием к алгоритмам подсчета количества структур, удовлетворяющих определенным условиям.

1. Изучение и описание предметной области

Предметной областью является набор команд для ассемблера x86, а именно команды для работы с массивами и структурами.

В этом языке программирования используются различные директивы для определения данных: DB (байт – 8 бит), DW (слово – 16 бит), DD (двойное слово – 32 бита), DP (слово в 6 байт – 48 бит), DQ (четверное слово – 64 бита) и DT (слово в 10 байт – 80 бит) [1].

Для перемещения данных из одной области в другую применяется инструкция MOV. Её синтаксис: MOV <пункт назначения>, <источник> [2].

Для управления набором данных различных типов можно использовать структуры. Структура – составной объект, занимающий несколько соседних ячеек памяти. Компоненты структуры называются полями, они могут быть разного типа (размера). Доступ к полям осуществляется по именам. Описание типа структуры:

<имя типа> STRUC

<имя поля>

……..

<имя поля>

<имя типа> ENDS

Например: структура DATE с полями Y – год, M - месяц, D – день
DATE STRUC

Y DW 2013

M DB 2

D DB ?

DATE ENDS

Описание типа структуры может размещаться в любом месте программы, но обязательно до описания переменных этого типа.

Имена полей не должны совпадать с именами других объектов программы. Также в ассемблере не допускается вложенность структур.

В ассемблере имя поля структуры относится к простейшим константным выражениям и его значением является смещение данного поля относительно начала структуры. При выделении памяти под переменную данного типа ассемблер размещает ее поля в соседних ячейках памяти.

После описания типа структуры можно описывать переменные этого типа. Такие переменные называются структурами и описываются так: имя_переменной имя_типа <начальное значение>

В ассемблере одной директивой можно описать сразу несколько структур, т.е. можно описать массив, элементами которого являются структуры. Для этого в директиве указывается несколько операндов и/или конструкция повторения DUP

Например: DTS DATE <2007,12,5>, 10 DUP (<>)

Здесь описывается массив из 11 структур типа DATE, причем поля первой из них будут иметь следующие начальные значения: 2007, 12 и 5, а остальные 10 структур получат одни и те же начальные значения, взятые по умолчанию: 2013, 2, ?. При этом имя DTS получит только первая структура, остальные остаются безымянными и доступ к ним осуществляется по адресным выражениям вида DTS+4, DTS+8 и т.д.

Чтобы сослаться на поле структуры используется конструкция: <имя переменной структуры>. <начальное поля>

Например: DTS.M

Такая конструкция обозначает ту ячейку памяти, которую занимает указанное поле данной структуры. Встречая эту конструкцию, ассемблер заменяет ее на адрес данной ячейки.

Для проверки условий применяются команды условных и безусловных переходов. Команды условных переходов позволяют передать управление процессору к указанной команде в зависимости от выполнения некоторых условий, определяемых состоянием флагов. Команда безусловного перехода передаёт управление всегда (без учёта условий). Синтаксис команд условного и безусловного переходов: <команда_перехода> <адресное_выражение> [5].

Данные команды представлены в таблице 1.

Таблица 1

Команды условных и безусловных переходов

Мнемоническое обозначение	Условие перехода	Проверяемые значения флагов
JMP	безусловный переход	Нет
JA / JNBE	больше (для беззнаковых чисел)	CF=0 и ZF=0
JAE / JNB	больше или равно (для беззнаковых чисел)	CF=0
JB / JNAE	меньше (для беззнаковых чисел)	CF=1
JBE / JNA	меньше или равно (для беззнаковых чисел)	CF=1 или AF=1
JC	перенос	CF=1
JCXZ	CX равно нулю	нет
JE / JZ	равно	ZF=1
JG / JNLE	больше (для знаковых чисел)	ZF=0 и SF и OF одинаковы (оба 0 или 1)
JGE / JNL	больше или равно (для знаковых чисел)	SF и OF одинаковы (оба 0 или 1)
JL / JNGE	меньше (для знаковых чисел)	SF и OF различны
JLE / JNG	меньше или равно (для знаковых чисел)	ZF=1 и SF и OF различны
JO	переполнение	OF=1
JNO	нет переполнения	OF=0
JP / JPE	паритет четный	PF=1
JNP / JPO	паритет нечетный	PF=0
JS	знак	SF=1
JNS	нет знака	SF=0
JNZ	не нуль	ZF=0

2. Постановка задачи

Необходимо написать программу, которая должна верно обрабатывать массив структур F и найти год самого старого журнала и записать это число в регистр BX, а в регистр BP номер этого журнала.

2.1 Исходные данные

В качестве исходных данных будет представлен массив структур F, который включает в себя информацию о журналах: год выпуска, название журнала, номер выпуска, цена.

2.2 Набор выполняемых функций

Программа будет иметь две основные задачи: выбор структур, соответствующих определенному условию, и сохранение количества выбранных структур в регистр.

3. Выбор структур данных для решения поставленной задачи

Для решения задачи будут использоваться структуры данных, представленные в табл. 2.

Таблица 2

Структуры данных

Наименование	Обозначение	Тип данных
Journal	Journal STRUC	Структура
F	F Journal <>	Массив структур
name	db 15 DUP (?)	Массив байтов
year	dw 1950	Двойное слово
numberJournal	dw 50	Двойное слово
price	dw 190	Двойное слово

Переменная Journal представляет структуру с данными о журнале.

F – это массив, предназначенный для хранения структур.

Переменные name, year, numberJournal и price хранят в себе информацию о журналах: название журнала, год выхода, номер журнала и цена.

4. Логическое проектирование

Дана структура Journal с полями: название журнала, год выпуска, номер журнала, цена, и массив F типа структуры, где собрана информация о 15 журналах. Найти год выпуска самого результата и вывести его и номер выпуска.

1. Инициализация массива структур:
   - объявление структуры Journal, содержащей информацию о журналах: название журнала, год выхода, номер журнала и цена;
   - задание характеристик каждого журнала в соответствии с этой структурой и помещение их в массив.
2. Подготовка к обходу через цикл:
   - установка счетчика таким образом, чтобы он был равен длине массива;
   - создание переменной для хранения самого старого года для сравнений в цикле;
   - создание переменной для хранения номера выпуска старого года.
3. Обход по массиву через цикл:
   - проверка на условие, что год выпуска структуры элемента массива меньше переменной, в которой хранится предыдущий старый год;
   - если условие истинно, то перезаписываем данные переменной хранения старого года и переменной для хранения номера старого года;
   - уменьшение счетчика.
4. Завершение программы:
   - выход из цикла, когда все элементы списка напитков проверены;
   - вывод самого старого года и номер этого выпуска.

Блок-схема программы представлена на рис. 1.

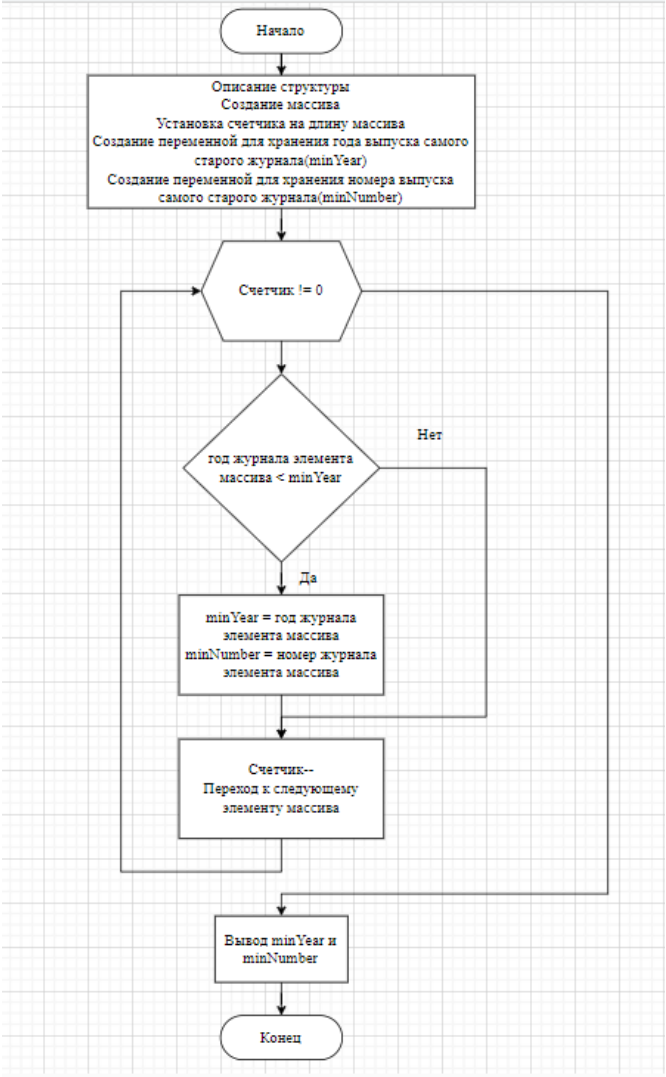

Рис. 1. Блок-схема программы

5. Физическое проектирование

Шаблон структуры состоит из последовательности следующего набора команд:

Journal STRUC

name db 15 dup(?)

year dw 1967

numberJournal dw 50

price dw 33

Journal ENDS

Где Journal – название структуры, name – поле, которое хранит

название журнала, year – поле, которое год выхода журнала, numberJournal – поле, которое хранит номер выпуска журнала, price – поле, которое хранит цену журнала.

Массив F представлен следующим видом: F Journal <…>. Где F – название массива, Journal – тип данных в массиве, а именно структура, <…> – инициализация элементов массива F, в котором записываются значения полей структуры через запятую.

Регистр CX используется как счётчик для цикла.

Регистр BX используется для навигации по массиву путем перехода к следующей структуре. Это достигается путем увеличения его значения на размер структуры с помощью операции ADD.

6. Проектирование интерфейса

Пользователь взаимодействует с любым текстовым редактором, предоставляя необходимые данные в кодовом файле. После компиляции программа обрабатывает эти данные, и результаты доступны в скомпилированной программе, конкретно в регистрах BX и BP.

Формой для ввода данных является код, где пользователь может изменять информацию о журналах по своему усмотрению.

Результаты выводятся в регистры BX и BP, которые содержат самый старый год журнала и его номер, заданным пользователем.

7. Кодирование

В данном разделе приведена последовательность команд, которые использовались для нахождения самого старого года выпуска журнала.

1. Объявление структуры Journal, инициализация массива F и инициализация двойного слова minYear и инициализация двойного слова minNumber:

```
.model small
```

```
.stack 100h

.data
; define struct "Journal"
; size of Journal = 21 bytes
Journal STRUC
name db 15 dup(?) ; name of journal
year dw 1967 ; release journal
numberJournal dw 50 ; number of journal
price dw 33 ; price journal
Journal ENDS
F Journal<'Lisa', 1995, 4, 33>, <'0 zatrat', 1995, 28, 25>, <'Winx', 2011, 54, 150>, <'top model', 2004, 784, 200>, <'Vogue', 1892, 1, 400>, <'Ejik', 1956, 46, 43>, <'ELLE', 1967, 23, 97>, <'Monster High', 2003, 47, 200>, <'17', 1986, 28, 313>, <'Burda', 2006, 31, 90>, <'Oops', 2015, 89, 170>, <'Igromania', 1999, 45, 50>, <'Tony Hawk', 1970, 34, 78>, <'Marvel', 1920, 789, 2>, <'DC', 1915, 654, 4>
minYear dw 21000
numJournal dw ?
```

2. Инициализация регистров для корректного доступа программы к данным, особенно к массиву структур F, содержащему информацию о журналах и переход к первой итерации цикла:

```
.code
ORG 100h
start:
MOV ax, @data ; set AX to data segment
MOV DS, AX ; set DS to point to data segment
MOV CX, 15 ; set CX to the amount of journal (15)
XOR BX, OFFSET F
CMP CX, 0
JA iteration
```

3. Этот участок кода проходит по массиву F. Для каждой структуры извлекается значение year, которое сравнивается с переменной minYear, и если значение меньше year, то записываем значение структуры в minYear и извлекаем значение numberJournal и записываем его в minNumber. Затем происходит переход к следующей структуре в массиве, и этот процесс повторяется, пока все структуры не будут просмотрены:

```
iteration:
XOR AX, AX ; null AX
DEC CX ; decrement CX
```

```
XOR DX, DX ; empty DX
MOV AX, minYear
MOV DX, F.year[BX]
CMP DX, AX ; DX <= AX
JLE check_number ; true => jump to number check
ADD BX, TYPE Journal ; shift address
CMP CX, 0 ; false => check CX != 0
JA iteration ; true => next iteration
JMP stop ; false => stop

check_number:
MOV minYear, DX
MOV AX, F.numberJournal[BX]
MOV numJournal, AX
ADD BX, TYPE Journal ; shift address
CMP CX, 0 ; check CX != 0
JA iteration ; true => next iteration
JMP stop ; false => stop

stop:
MOV BP, numJournal
MOV BX, minYear
MOV AX, 4C00h
INT 21h
END start
```

8. Тестирование

Было проведено тестирование для проверки работоспособности программы и выявления возможных ошибок. В результате было подтверждено, что программа функционирует корректно, и никаких проблем не было обнаружено.

Тестовые данные представлены в табл. 3.

Таблица 3

Тестовые данные

Исходные данные	Тестируемый модуль	Ожидаемые результаты
Объявление шаблона структуры	curs.asm	Корректное объявление шаблона структуры
Объявление и инициализация массива структур	curs.asm	Корректная инициализация массива структур.
Обработка массива, запись значения из структуры в регистр	curs.asm	Корректная обработка массива, корректная запись значения из структуры в регистр
Переход между структурами в массиве	curs.asm	Корректный переход между структурами в массиве
Установка меток для цикла	curs.asm	Корректный переход между метками
Подсчёт количества верных структур	curs.asm	Корректный подсчёт количества верных структур
Запись конечного числа в необходимый регистр	curs.asm	Корректная работа записи ответа в необходимый регистр

Результаты тестирования представлены в табл. 4.

Таблица 4

Результаты тестирования

Дата тестирования	Тестируемый модуль	Кто проводил тестирование	Описание теста	Результаты тестирования
23.12.2023	curs.asm	Зернов В.А.	Корректное объявление шаблона	Успех

			структуры	
23.12.2023	curs.asm	Зернов В.А.	Корректная инициализация массива структур	Успех
23.12.2023	curs.asm	Зернов В.А.	Корректная обработка массива, корректная запись значения из структуры в регистр	Успех
23.12.2023	curs.asm	Зернов В.А.	Корректный переход между структурами в массиве	Успех
23.12.2023	curs.asm	Зернов В.А.	Корректный переход между метками	Успех
23.12.2023	curs.asm	Зернов В.А.	Корректное перезаписывание переменной при сравнении	Успех
23.12.2023	curs.asm	Зернов В.А.	Корректный поиск самого старого года	Успех
23.12.2023	curs.asm	Зернов В.А.	Корректная работа записи ответа в необходимые регистры	Успех

Заключение

В результате выполнения курсовой работы была создана программа, предназначенная для поиска самого старого года выпуска журнала. Эта программа соответствует требованиям технического задания: она анализирует структуры и записывает окончательный результат в указанный регистр.

При выполнении курсовой работы были освоены навыки программирования на Assembler, что является ценным приобретением для дальнейшего обучения в программировании. Эти навыки позволяют лучше понимать работу высокоуровневых языков программирования и процессы программирования на уровне машины.

Список литературы

1. Директивы определения данных [Электронный ресурс]. Дата доступа: 25.12.2023. Ссылка: https://studfile.net/preview/16566533/page:7/

2. Руководство по ассемблеру x86 для начинающих [Электронный ресурс]. Дата доступа: 25.12.2023. Ссылка: https://habr.com/ru/articles/423077/

3. Ассемблер Intel x86-64 Структуры [Электронный ресурс]. Дата доступа: 25.12.2023. Ссылка: https://metanit.com/assembler/tutorial/3.11.php

4. Ассемблер Intel x86-64 массивы [Электронный ресурс]. Дата доступа: 25.12.2023. Ссылка: https://metanit.com/assembler/tutorial/3.9.php

5. Учебный курс. Часть 16. Условные и безусловные переходы [Электронный ресурс]. Дата доступа 25.12.2023. Ссылка: https://fasmworld.ru/uchebnyj-kurs/016-uslovnye-i-bezuslovnye-perexody/

6. Turbo Assembler - Wikipedia [Электронный ресурс]. Дата доступа 25.12.2023. Ссылка: https://en.m.wikipedia.org/wiki/Turbo_Assembler

7. DOSBox — Википедия [Электронный ресурс]. Дата доступа 25.12.2023. Ссылка: https://ru.wikipedia.org/wiki/DOSBox

8. Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских, О.Л. Селяничев «Методика и организация самостоятельной работы студентов» [Файл].

Техническое задание. Приложение 1

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

Институт информационных технологий
(наименование структурного подразделения)

Кафедра математического и программного 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.
«___» октября 2023 г.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ НИЗКОГО УРОВНЯ
Техническое задание на курсовую работу
Листов 7

Руководитель: Виноградова Л. Н.

Исполнитель: студент гр. 1ПИб-02-2оп-22
Зернов В.А.

2024 год

Введение

Цель данной курсовой работы заключается в изучении и применении концепции структур в языке программирования ассемблер. Программа будет исследовать создание и управление структурами, включая описание типов структур, создание переменных структурного типа и формирование массивов структур для оптимального использования памяти.

1. Основания для разработки

Разработка ведётся на основании задания на курсовую работу по дисциплине «Программирование на ассемблере», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 10 ноября 2023 года

Тема разработки: «Программирование на языке низкого уровня»

2. Назначение разработки

Основной задачей данной работы является практическое применение знаний, полученных при изучении дисциплины "Программирование на ассемблере", а также при освоении работы с программой-эмулятором процессора Intel 8086 "emu8086".

3. Требования к программе

3.1. Требования к функциональным характеристикам

Программа должна быть способна работать с структурой Journal, которая включает в себя информацию о журналах: название, год выпуска, номер выпуска, цена и массив F с данными о 15 журналах. Основная задача программы — найти самый старый год выпуска журнала в структуре Journal и

передать это число в регистр BX и номер этого выпуска передать в регистр BP для последующего использования.

3.2. Требования к надёжности

Программа должна обеспечивать надежную обработку структуры данных, сохраняя её целостность и создавая новые структуры с обработанными результатами. Необходимо гарантировать, что данные в структуре остаются неповрежденными и не подвергаются случайному удалению или изменению в процессе обработки. Важно, чтобы программа демонстрировала стабильную работу и способность корректно завершать свою работу даже при возможных сбоях или непредвиденных обстоятельствах.

3.3. Условия эксплуатации

Эксплуатационные условия программы должны включать в себя работу на совместимых средах, поддерживающих функциональность ассемблера. Программа должна корректно функционировать на подходящих для ассемблерной среды операционных системах. Также важно, чтобы входные данные соответствовали ожидаемому формату и были представлены для обработки в соответствии с требованиями программы.

3.4. Требования к составу и параметрам технических средств

Технические минимальные требования:

- оперативная память: не менее 128 Кб;
- процессор: архитектура x86-16 или совместимая;
- свободное место на диске: не менее 1 Мбайта;
- монитор для отображения программы;
- мышь и клавиатура для ввода данных и взаимодействия с программой.

3.5. Требования к информационной и программной совместимости

Для обеспечения совместимости программы требуется наличие операционной системы Windows 7, 8, 10, 11. Также необходимо установить ПО, такое как emu8086 или другой компилятор Assembler, чтобы разрабатывать и запускать программу на языке ассемблера.

3.6. Требования к маркировке и упаковке

Обычно требования к маркировке и упаковке не применимы к программе, поскольку она является цифровым продуктом, распространяемым в электронном формате.

3.7. Требования к транспортированию и хранению

Для правильной работы программы необходимо расположить соответствующие файлы на флеш-накопителе или в памяти компьютера. Рекомендуется сохранить программу на внешнем носителе, чтобы предотвратить потерю информации.

3.8. Специальные требования

Отсутствуют.

4. Требования к программной документации

4.1. Содержание расчётно-пояснительной записки:

- титульный лист;
- оглавление;
- введение;
- описание предметной области;
- описание разработки;
- описание созданной программы;
- заключение;
- источники;

- приложения.

4.2. Требования к оформлению

Требования к оформлению, установленные ГОСТ, должны быть выполнены на протяжении всей работы без каких-либо изменений (в табл. П1.1).

Таблица П1.1

Требования к оформлению

Документ	Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код. Листы и диск в конверте вложены в пластиковую папку скоросшивателя.
Страницы	Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код. Листы и диск в конверте вложены в пластиковую папку скоросшивателя.
Абзацы	Ориентация – книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см.
Шрифты	Межстрочный интервал – 1.5, перед и после абзаца – 0. Кегль – 14. В таблицах шрифт 12. Шрифт листинга – 10 (возможно в 2 колонки).
Рисунки	Подписывается под ним по центру: Рис.Х Название В приложениях: Рис.П1.3. 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 – выровнены 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
Определение темы курсовой работы	10.11.23	Утверждена тема для разработки	
Оформление технического задания	17.11.23	Выполнение технического задания	
Разработка алгоритма	01.12.2023	Готовый алгоритм	
Написание программы	20.12.2023	Написанная программа	
Тестирование программы	25.12.2023	Проверенная и отлаженная программа	
Оформление РПЗ	25.12.2023	Написание 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
Техническое задание	17.11.23	Оформленное техническое задание	
Теоретическая часть курсовой работы	06.12.2023	Оформленная теоретическая часть	
Практическая часть курсовой работы	07.12.2023	Программа	
Расчётно-пояснительная записка	23.12.2023	Оформленная РПЗ	
Защита курсовой работы	26.12.2023	Итоговая оценка за курсовую работу	

Руководство пользователя. Приложение 2

1. Общие сведения о программе

Эта программа анализирует информацию о журналах, выявляя самый старый год выпуска. Он использует массив структур, где каждая структура представляет данные об одном журнале.

2. Описание установки

Для запуска программы необходимо иметь доступ к файлу программы и использовать любой текстовый редактор. Также необходимо иметь программу DOSBox для компиляции кода. Установка программы не требуется.

3. Описание запуска

Необходимо извлечь программу из носителя на компьютер, используя дисковод. Для просмотра и редактирования кода необходимо щёлкнуть по файлу curs.asm два раза или воспользоваться контекстным меню с помощью нажатия правой кнопкой мыши по файлу (см. рис. П.2.1).

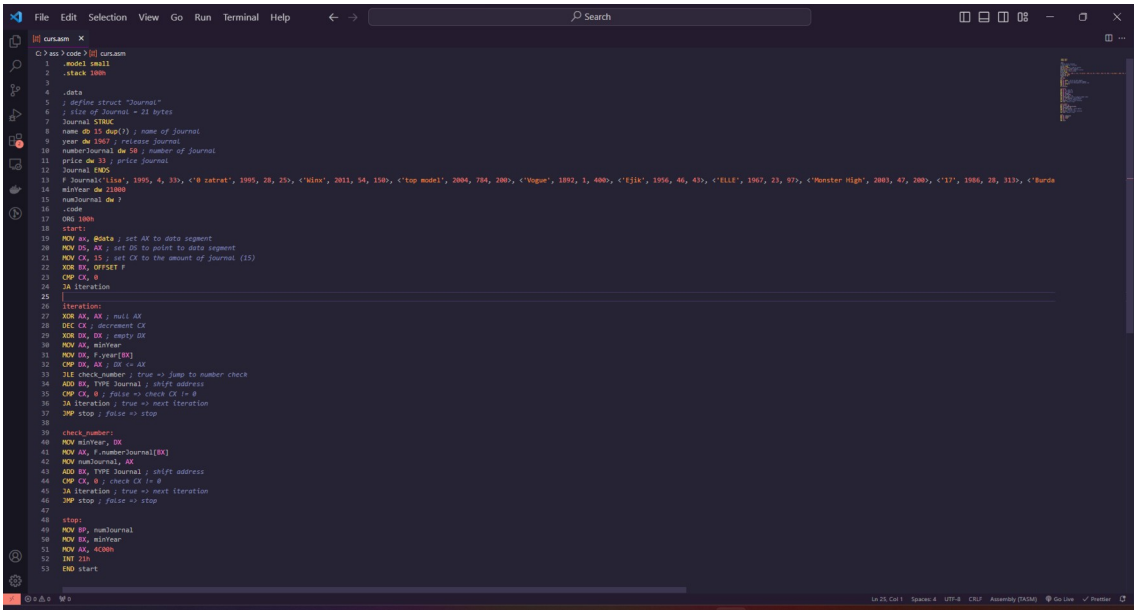

Рис. П.2.1. Программа, запущенная в Visual Studio Code

Для того чтобы запустить и скомпилировать код необходимо воспользоваться программой DOSBox. Для начала необходимо запустить DOSBox и перейти в директорию code с помощью команды cd code (см. рис. П.2.2).

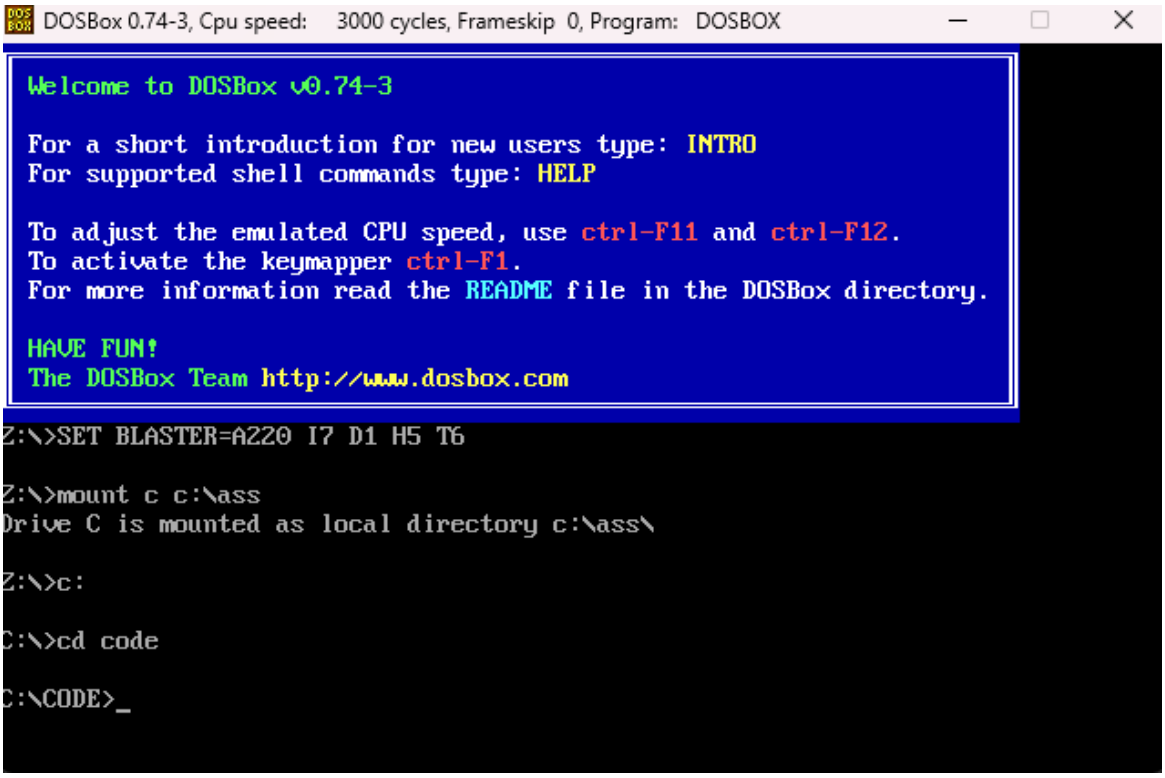

Рис. П.2.2. Запуск DOSBox и переход в директорию code

Затем нам необходимо ввести команду c:\tasm\bin\tasm.exe curs.asm. Через данную команду мы вызываем транслятор Turbo Assembler и он генерирует файл curs.obj, если в файле нет синтаксических ошибок (см. рис. П.2.3).

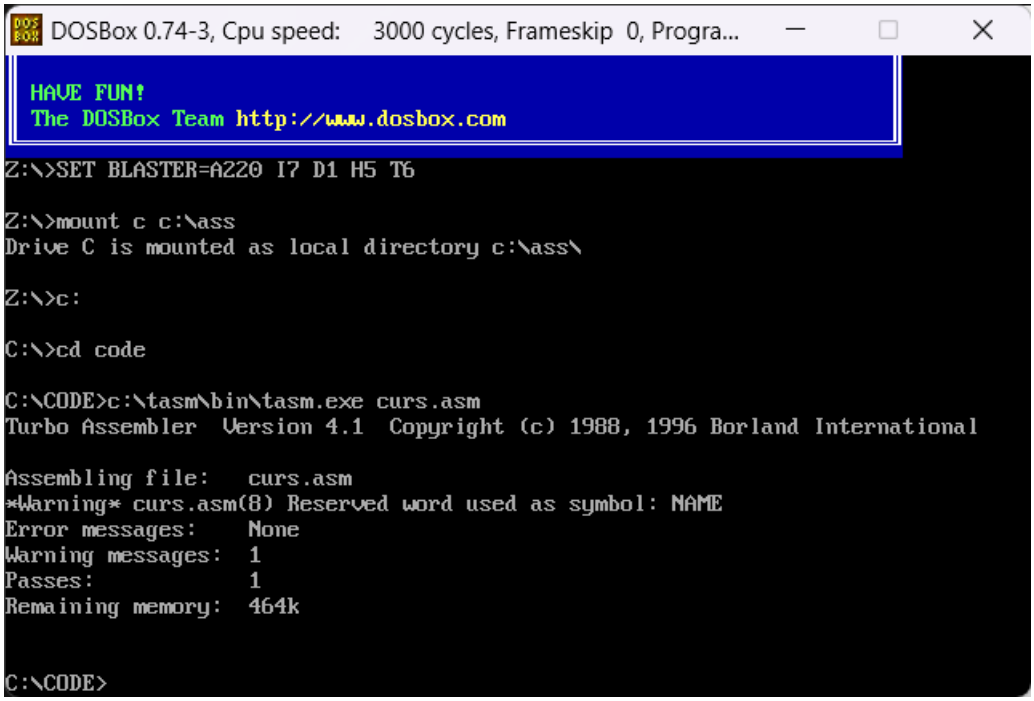

Рис. П.2.3. Ввод команды c:\tasm\bin\tasm.exe curs.asm

Далее мы вводим команду c:\tasm\bin\tlink.exe curs.obj. Данная команда вызывает компоновщик, который компонует curs.obj файл в curs.exe (см. рис. П.2.4).

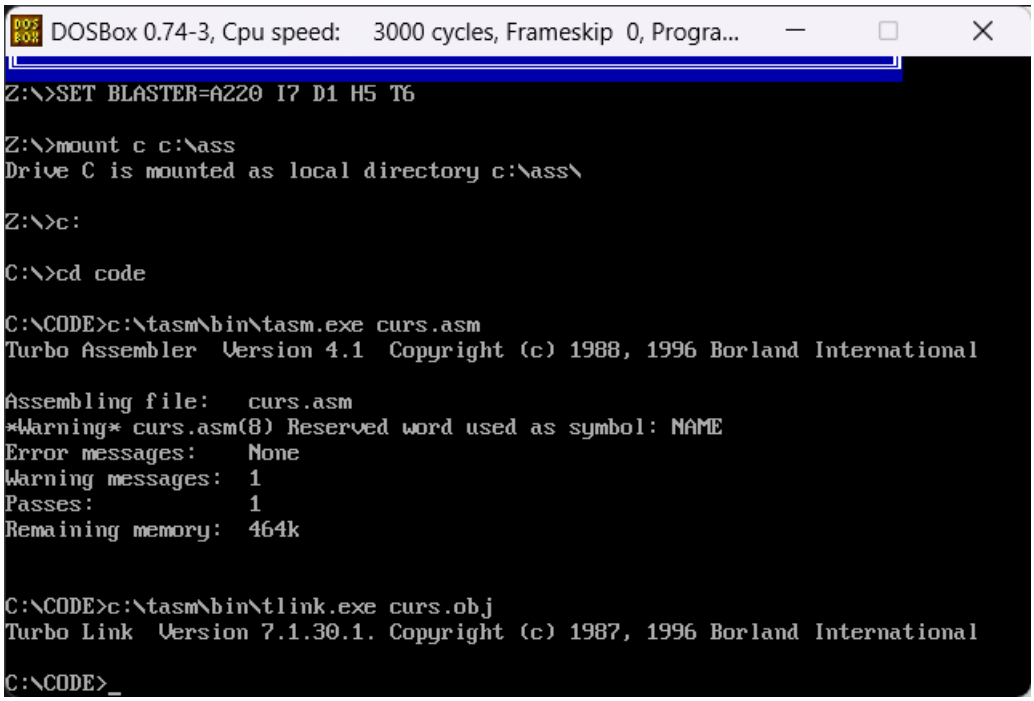

Рис. П.2.4. Ввод команды c:\tasm\bin\tlink.exe curs.obj

В конце мы вводим команду c:\tasm\bin\td.exe curs.exe. Данная команда вызывает отладчик в DOSBox (см. рис. П.2.5).

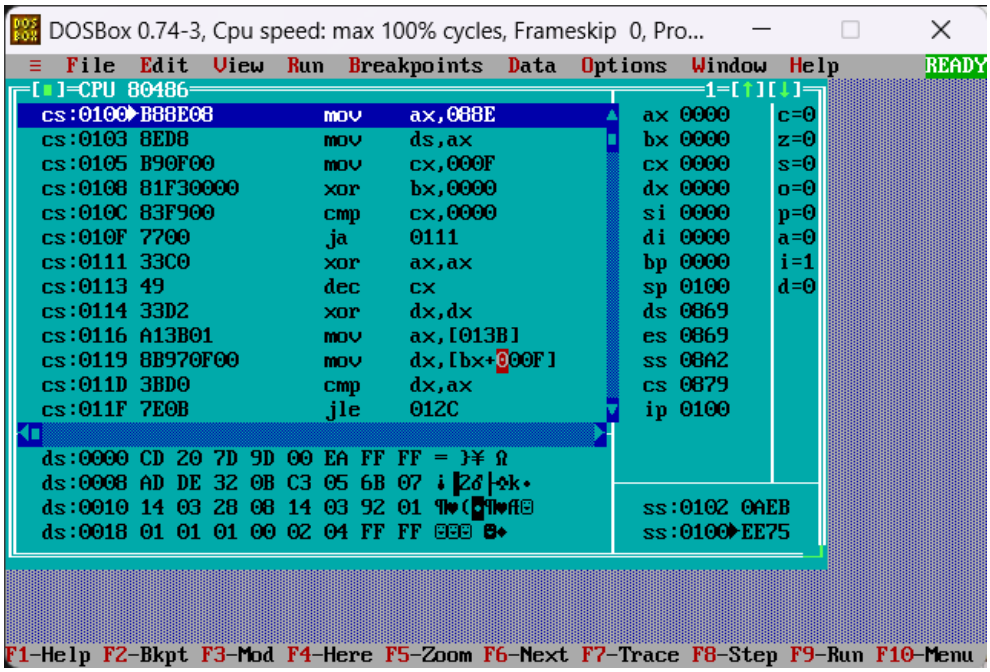

Рис. П.2.5. Вызов отладчика в DOSBox

При нажатии клавиши F8 будет выполнен шаг программы, при нажатии F9 будет выполнена вся программа. Результат можно увидеть в регистрах BX и BP (см. рис. П.2.6).

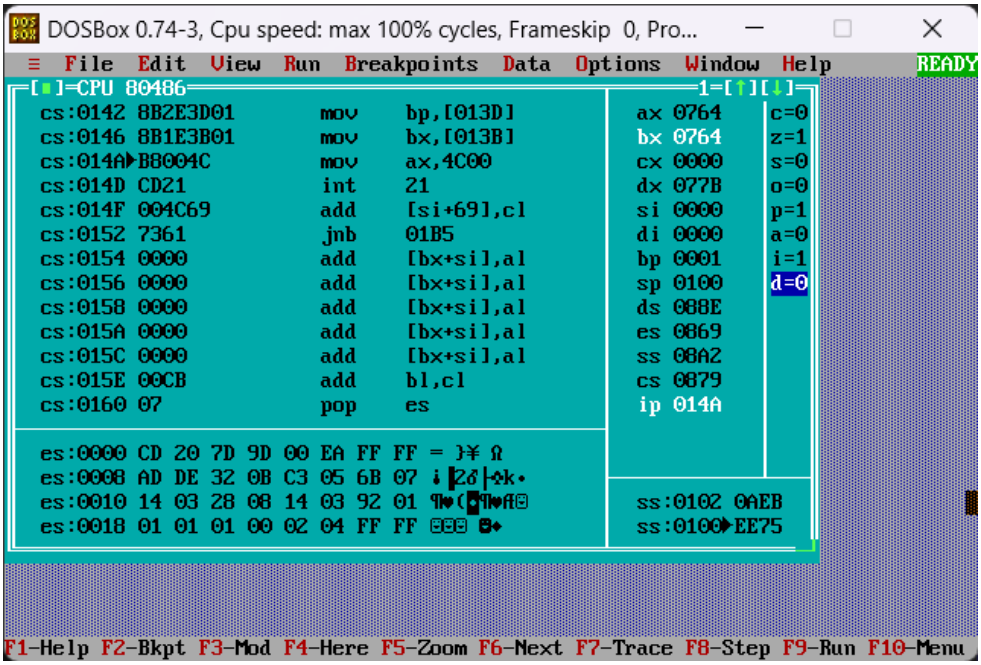

Рис. П.2.6. Результат выполнения программы

Для того чтобы отредактировать параметры журналов, необходимо поменять значения в элементах массива <”название журнала“, год выпуска, номер журнала, цена> в 13 строке (см. рис. П.2.7)

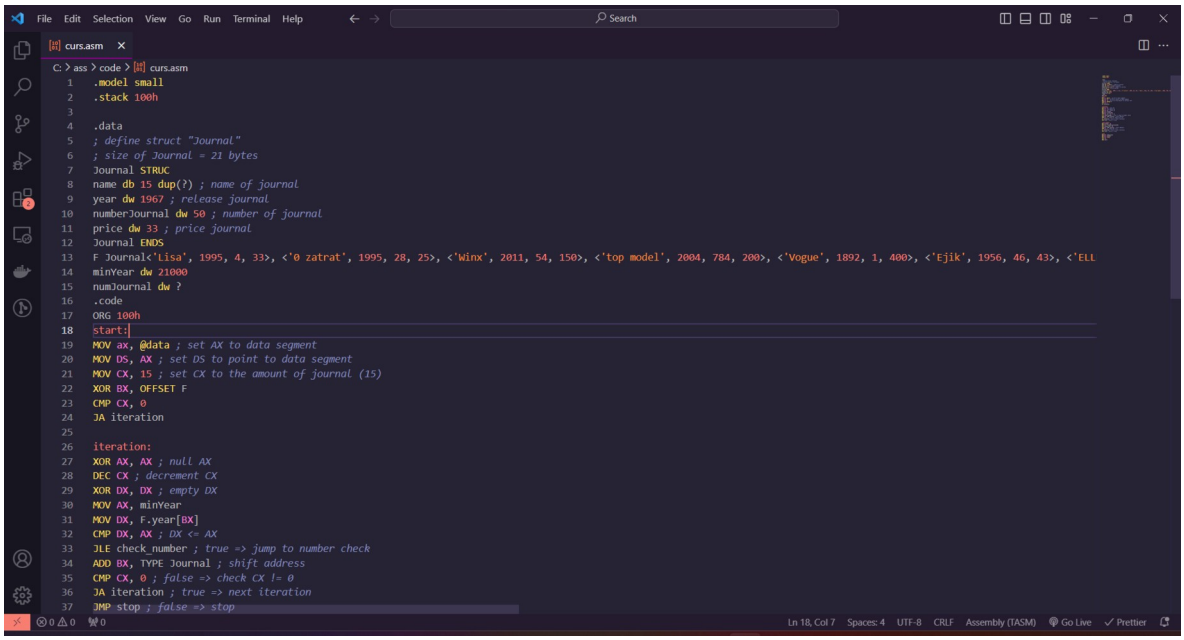

Рис. П.2.7. Фрагмент кода

Программный код. Приложение 3

Файл curs.asm:

```
.model small
.stack 100h
.data
; define struct "Journal"
; size of Journal = 21 bytes
Journal STRUC
name db 15 dup(?) ; name of journal
year dw 1967 ; release journal
numberJournal dw 50 ; number of journal
price dw 33 ; price journal
Journal ENDS
F Journal<'Lisa', 1995, 4, 33>, <'0 zatrat', 1995, 28, 25>, <'Winx', 2011, 54, 150>, <'top model', 2004, 784, 200>, <'Vogue', 1892, 1, 400>, <'Ejik', 1956, 46, 43>, <'ELLE', 1967, 23, 97>, <'Monster High', 2003, 47, 200>, <'17', 1986, 28, 313>, <'Burda', 2006, 31, 90>, <'Oops', 2015, 89, 170>, <'Igromania', 1999, 45, 50>, <'Tony Hawk', 1970, 34, 78>, <'Marvel', 1920, 789, 2>, <'DC', 1915, 654, 4>
minYear dw 21000
numJournal dw ?
.code
ORG 100h
start:
MOV ax, @data ; set AX to data segment
MOV DS, AX ; set DS to point to data segment
MOV CX, 15 ; set CX to the amount of journal (15)
XOR BX, OFFSET F
CMP CX, 0
JA iteration
iteration:
XOR AX, AX ; null AX
DEC CX ; decrement CX
XOR DX, DX ; empty DX
MOV AX, minYear
MOV DX, F.year[BX]
CMP DX, AX ; DX <= AX
JLE check_number ; true => jump to number check
ADD BX, TYPE Journal ; shift address
CMP CX, 0 ; false => check CX != 0
JA iteration ; true => next iteration
JMP stop ; false => stop
check_number:
```

```
MOV minYear, DX
MOV AX, F.numberJournal[BX]
MOV numJournal, AX
ADD BX, TYPE Journal ; shift address
CMP CX, 0 ; check CX != 0
JA iteration ; true => next iteration
JMP stop ; false => stop
stop:
MOV BP, numJournal
MOV BX, minYear
MOV AX, 4C00h
INT 21h
END start
```