Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

ОТЧЁТ ПО
Домашнему заданию

Удалова Виктория ИУ5-35Б

Парадигмы и конструкции языков программирования

Москва
2023

Задание

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Реферат на тему "Принципы работы с памятью в языке программирования ассемблера"

Ассемблер — это язык программирования, который предназначен для написания низкоуровневого кода, который напрямую исполняется процессором. Он использует мнемоники для команд процессора и позволяет программисту иметь полный контроль над аппаратными ресурсами компьютера.

Парадигма языка ассемблера включает в себя низкоуровневое программирование, где каждая инструкция соответствует определенной команде процессора.

Основные конструкции языка ассемблера включают в себя:

1. Мнемоники команд: в языке ассемблера каждая команда процессора представлена мнемоническим символьным именем, которое упрощает написание кода и его понимание. Например, команда MOV используется для перемещения данных между регистрами или памятью.

2. Регистры: ассемблер использует регистры процессора для хранения и обработки данных. Регистры обычно имеют ограниченный размер и отличаются по назначению (например, регистр EAX может использоваться для хранения данных, а регистр ECX для счетчика цикла).

3. Директивы: ассемблер содержит директивы, которые позволяют задавать различные параметры сборки программы, например, указание секции, инициализация данных, выделение памяти и другие.

4. Переходы и циклы: для управления потоком выполнения программы в ассемблере используются команды перехода (например, JMP, JE, JG) и циклы (например, LOOP).

5. Обработка данных: язык ассемблера позволяет производить арифметические операции (сложение, вычитание, умножение, деление) над данными, а также операции сравнения и логические операции.

6. Ввод/вывод: ассемблер содержит команды для взаимодействия с вводом/выводом данных, например, для чтения данных с клавиатуры или вывода на экран.

Основные принципы работы с памятью в языке программирования ассемблера:

1. Адресация:

В ассемблере доступ к данным в памяти осуществляется посредством адресации. Это означает, что для обращения к определенным ячейкам памяти необходимо указать их адрес. Существуют различные типы адресации, включая прямую (когда адрес указан явно), косвенную (когда адрес указан в регистре), индексированную (когда используется смещение от базового адреса) и базово-индексированную (когда совмещаются базовый адрес и индекс).

2. Регистры:

Регистры — это маленькие, быстрые области памяти внутри процессора, используемые для временного хранения данных. Работа с регистрами важна при работе с памятью, так как они позволяют быстро и эффективно обрабатывать данные без необходимости постоянного обращения к памяти.

3. Работа с данными:

Для чтения, записи и обработки данных в памяти на языке ассемблера используются специальные команды. Например, команды загрузки данных в регистры (mov), чтения и записи в память (load, store), арифметические операции (add, sub, mul, div) и операции сравнения (cmp, test).

4. Стек:

Стек — это область памяти, используемая для управления вызовами функций, передачи параметров, хранения локальных переменных и возврата адресов вызова. Операции работы со стеком включают push (помещение данных на вершину стека) и pop (извлечение данных с вершины стека).

5. Выделение и освобождение памяти:

В ассемблере программист самостоятельно отвечает за выделение и освобождение памяти. Для выделения памяти используются специальные системные вызовы, а для освобождения - соответствующие команды.

Пример кода на ассемблере:

```
section .data
    msg db 'Hello, world!', 0    ; Строка для вывода

section .text
global _start

_start:
    ; Пример работы с данными
    mov eax, 4                ; Системный вызов для вывода
    mov ebx, 1                ; Файловый дескриптор stdout
    mov ecx, msg              ; Адрес строки для вывода
    mov edx, 13               ; Длина строки
    int 0x80                  ; Вызов системного вызова

    ; Пример работы с регистрами
    mov ecx, 10               ; Загружаем число 10 в регистр ecx
    mov edx, 5                ; Загружаем число 5 в регистр edx
    add ecx, edx              ; Складываем два числа
    sub ecx, 2                ; Вычитаем 2
    mov eax, ecx              ; Результат сохраняем в регистр eax

    ; Пример работы со стеком
    push eax                  ; Помещаем значение eax на вершину стека
    pop ebx                   ; Извлекаем значение eax с вершины стека в регистр ebx

    ; Пример выделения и освобождения памяти
    mov eax, 4                ; Системный вызов для выделения памяти
    mov ebx, 0                ; Размер памяти для выделения
    int 0x80                  ; Вызов системного вызова

    mov eax, 1                ; Системный вызов для освобождения памяти
```

```
    mov ebx, ebx            ; Адрес памяти, которую нужно освободить
    int 0x80                ; Вызов системного вызова

    ; Выход из программы
    mov eax, 1              ; Системный вызов для завершения программы
    xor ebx, ebx            ; Код возврата 0
    int 0x80                ; Вызов системного вызова
```

Этот пример показывает основные аспекты языка программирования ассемблера, такие как адресация, работа с данными, регистрами, стеком и выделение/освобождение памяти.

Для работы с кодом на языке ассемблера в современных операционных системах, таких как Windows или Linux, можно использовать эмуляторы, такие как DOSBox. DOSBox позволяет запускать программы, написанные на ассемблере для DOS, на современных компьютерах. Для этого необходимо скомпилировать исходный код на ассемблере в исполняемый файл, который затем можно запустить в DOSBox.