Санкт-Петербургский Государственный Электротехнический Университет им. В. И. Ульянова (Ленина)

Открытый факультет

Кафедра вычислительной техники

**Пояснительная записка к курсовому проекту по дисциплине**

**«Архитектура современных ЭВМ»**

Выполнил: Макаров С.Э.

Руководитель: Валов А.А.

Санкт-Петербург

2013

**Оглавление**

Оглавление

[Введение 3](#_Toc370993388)

[1. Техническое задание 3](#_Toc370993389)

[1.1. Область применения 3](#_Toc370993390)

[1.2. Индивидуальное задание и литературные источники 3](#_Toc370993391)

[1.3. Метод конвейеризации 3](#_Toc370993392)

[1.4. Технические требования 4](#_Toc370993393)

[1.4.1. Перечень команд 4](#_Toc370993394)

[1.4.2. Диапазон и максимальная абсолютная погрешность данных 6](#_Toc370993395)

[1.4.3. Форматы представления команд и данных 6](#_Toc370993396)

[1.4.4. Используемый способ организации виртуальной памяти 7](#_Toc370993397)

[1.4.5. Способы организации ввода/вывода информации 8](#_Toc370993398)

[1.4.6. Способ оценки производительности ЭВМ 8](#_Toc370993399)

[1.4.7. Требуемое значение приращения производительности ЭВМ, которое должно быть получено в результате модификации первоначально созданной ЭВМ 9](#_Toc370993400)

[1.5. Способ тестирования создаваемой ЭВМ 9](#_Toc370993401)

Введение 3

1. Техническое задание 3

1.1. Область применения 3

1.2. Индивидуальное задание и литературные источники 3

1.3. Метод конвейеризации, используемый при модификации классической модели ЭВМ 4

1.4. Технические требования 4

1.4.1. Перечень команд 4

1.4.2. Диапазон и максимальная абсолютная погрешность 6

1.4.3. Форматы представления команд и данных 6

1.4.4. Используемый способ организации виртуальной памяти 7

1.4.5. Способы организации ввода/вывода информации 9

1.4.6. Способы оценки производительности ЭВМ 9

1.4.7. Требуемое значение приращения производительности ЭВМ 10

1.5. Способ тестирования создаваемой ЭВМ 10

1.6. Способ взаимодействия программ пользователя с внешними устройствами 11

1.7. Демонстрация возможностей взаимодействия CPU с внешними устройствами 11

1.8. Интерфейс Win32 APS 11

2. Структура и алгоритм функционирования классической модели 12

2.1. Вводные замечания 12

2.2. Регистровая структура ЭВМ 14

2.3. Алгоритм функционирования ЭВМ 14

2.4. Средства управления прерываниями ввода/выводаCPU 18

2.5. Организация виртуальной памяти 19

2.6. Оценка производительности созданной ЭВМ 19

3. Модификация созданной ЭВМ 21

3.1. Пути повышения производительности ЭВМ 21

3.2. Организация конвейерной обработки команд 21

3.3. Расчет прироста производительности 24

4. Описание тестовой задачи 24

4.1. Описание программы (исходной и объектной) 24

4.2. Моделирование процесса выполнения программы 25

5. Программирование ввода и вывода 27

5.1. Пример использования для ввода/вывода системных функций 27

5.2. Общие сведенья об интерфейсе Win32 API.

Заключение 28

# Введение

Целью курсового проекта является освоение способов организации ЭВМ и правил ее функционирования, а также приобретение навыков по составлению архитектурных, структурных, алгоритмических и программных описаний ЭВМ.

Цель курсового проекта достигается разработкой архитектуры классической модели ЭВМ с заданными техническими параметрами, которая затем модифицируется. Модификация проекта заключается в реализации конвейерного метода обработки команд программы.

# 1. Техническое задание

## 1.1. Область применения

Создаваемая ЭВМ используется для управления кондиционером воздуха на основании датчика температуры. Указанная машина относится к специализированным ЭВМ.

## 1.2. Индивидуальное задание и литературные источники

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер варианта | Объем виртуальной страницы Кбайт | Способ  адресации | | | | Основная  память | |
| Н | О | П | К | Объем  Мбайт | ШД |
| 4 | 2 | + | - | + | + | 64 | 64 |

**Литературные источники:**

* + Конспект лекций по дисциплине «Архитектура ЭВМ» Валов А.А.
  + «Архитектура компьютера» Э.Таниенбаум. 4-е издание, Изд. «Питер» 2003г

## 1.3. Метод конвейеризации

Для модернизации классической ЭВМ в проекте используется простой способ конвейеризации, позволяющий одновременно выполнять текущую команду и производить выборку следующей команды.

## 1.4. Технические требования

### 1.4.1. Перечень команд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мнемокод | Код | Операнд | Назначение и действия | Флаги |
| ADD | 01100 | Rr | Сложение с фиксированной точкой  Акк 🡨Акк + Rr | OF, ZF, SF |
| FADD | 01010 | Rr | Сложение с плавающей точкой  Акк 🡨Акк + Rr | OF, ZF, SF |
| WR | 00011 | Rd | Запись в память из аккумулятора  RAM(Rd) 🡨Акк |  |
| READ | 00100 | Rr | Чтение из памяти в аккумулятор  Акк🡨RAM(Rr) |  |
| JMP | 00101 | K | Безусловный переход  IP🡨K + IP |  |
| JS | 00001 | K | Условный переход  If (S==1) then IP🡨IP+K |  |
| JZ | 00010 | K | Условный переход  If (Z==1) then IP+K |  |
| HLT | 00000 |  | Останов  Тп 🡨0 |  |
| CLI | 00110 |  | Сброс флага IF  I 🡨0 | IF |
| STI | 00111 |  | Установка IF  I 🡨1 | IF |
| CMP | 01000 | Rr | Вычитание без записи результата, установка флагов  Aкк-Rr | OF, ZF, SF |
| SUB | 01011 | Rr | Вычитание  Aкк 🡨 Aкк-Rr | OF, ZF, SF |
| PUSH | 10001 | Rr | Запись в стек  SP 🡨 SP-4  RAM(SP) 🡨 Rr |  |
| POP | 01101 | Rr | Чтение из стека  Rr 🡨 RAM(SP)  SP 🡨 SP+4 | OF,ZF,SF,IF |
| INT | 01110 | I | Вызов прерывания  SP 🡨 SP-4  RAM(SP) 🡨 IP  SP 🡨 SP-4  RAM(SP) 🡨 IF  IF 🡨 0  IP 🡨 RAM(I\*4) | IF |
| IRET | 01111 |  | Возврат из прерывания  IF🡨RAM(SP)  SP🡨SP+4  IP🡨RAM(SP)  SP🡨SP+4 | OF,ZF,SF,IF |
| AND | 01001 | Rr | Побитовое логическое И  Aкк 🡨Aкк&Rr | ZF |
| OR | 10010 | Rr | Побитовое логическое ИЛИ  Aкк🡨Aкк|Rr | ZF |
| NOT | 10100 |  | Логическое отрицание  Aкк 🡨 Aкк | ZF |
| IN | 11010 | N | Загрузка в Акк байта порта I/O  Акк 🡨 PORT(N) |  |
| OUT | 11011 | N | Вывод значения из регистра в порт  PORT(N) 🡨Aкк |  |
| CONTSAVE | 11100 | Rd | Сохранение контекста процесса  RAM(Rd)🡨IP  RAM(Rd+4)🡨FLAGS  RAM(Rd+8)🡨A  RAM(Rd+12)🡨R1  RAM(Rd+16)🡨PDBR1  RAM(Rd+20)🡨PDSR1  RAM(Rd+24)🡨SP |  |
| CONTREST | 01100 | Rr | Восстановление контекста процесса  IP🡨RAM(Rr)  FLAGS🡨RAM(Rr+4)  A🡨RAM(Rr+8)  R1🡨RAM(Rr+12)  PDBR1🡨RAM(Rr+16)  PDSR1🡨RAM(Rr+20)  SP🡨RAM(Rr+24) | OF, ZF, SF, IF |
| MOVREG | 01010 | Rr | Загружает Rr в регистр косвенной адресации REG←RAM(Rr) |  |
| CMP | 00011 | Rd | Сравнить содержимое регистра Rd с Акк. Установить флаг ZF – если равны, SF – если Rd < Акк | ZF,SF |
| CMD | 00100 | Rd | Сравнить содержимое регистра Rd с Акк. Установить флаг ZF – если равны, SF – если Rd > Акк | ZF,SF |
| UVE | 11101 | Rr | Увеличение на 1  RAM(Rr) 🡨 Rr+1 |  |
| UME | 11100 | Rr | Уменьшение на 1  RAM(Rr) 🡨 Rr-1 |  |

Акк – (аккумулятор) источник данных или приемник;

К – адресная константа, для задания которой выделены n+1 разрядов. Диапазон значений адресной константы: -2\*\*n≤K≤+(2\*\*n-1).

Rr – источник данных;

Rd – приемник данных.

Формат состояния процессора

### 1.4.2. Диапазон и максимальная абсолютная погрешность данных

Диапазон для чисел с фиксированной точкой:

Диапазон для чисел с плавающей точкой:

Максимальная абсолютная погрешность для чисел с фиксированной точкой:

Максимальная абсолютная погрешность для чисел с плавающей точкой:

### 1.4.3. Форматы представления команд и данных

СА - способ адресации

**Адресация:**

**1. Непосредственная (СА=01):**

23 0

|  |
| --- |
| операнд |

Операнд хранится непосредственно в команде.

**2. Прямая (СА=00, СА=11):**

23 0

|  |
| --- |
| Адрес |

В команде указывается адрес ячейки, в которой хранится информация (операнд).

**3. Косвенная (СА=10):**

Адрес операнда хранится в специальном регистре (**INDirectAdressingRegistry**).

### 1.4.4. Используемый способ организации виртуальной памяти

### 1.4.5. Способы организации ввода/вывода информации

Существует 3 способа организации системы ввода/вывода.

1. Программной проверки готовности.

Программа опрашивает порт ввода/вывода, когда порт принимает значение «готово» - производится обмен с внешним устройством.

2. Прямой доступ к памяти.

Используется канал прямого доступа к памяти, по которому массивы данных передаются непосредственно между периферийным устройством и ОЗУ, минуя микропроцессор. Это позволяет достичь наибольшей скорости передачи, но требует специального контроллера прямого доступа к памяти.

3. Использование прерываний.

Устройство сообщает о своей готовности с помощью прерывания. Обработчик прерывания осуществляет ввод/вывод и возвращает управление основной программе.

В машине используется 2 способа: способ программной проверки готовности и использование прерываний.

### 1.4.6. Способ оценки производительности ЭВМ

Для вычисления величины времени выполнения обобщенной команды используется формула, определяющая величину взвешенного арифметического среднего, которая позволяет учесть частоту использования каждой команды в заданной смеси команд.

X= 

Xi – время (такты) выполнения команды i-го типа в заданной смеси команд;

Ci – дробная величина (доля). Характеризует частоту использования команды i-го типа в заданной смеси команд.

### 1.4.7. Требуемое значение приращения производительности ЭВМ, которое должно быть получено в результате модификации первоначально созданной ЭВМ

В результате модификации первоначально созданной ЭВМ ожидается приращение производительности на…

## 1.5. Способ тестирования создаваемой ЭВМ

Тестирование создаваемой ЭВМ будет осуществлено методом моделирования процесса выполнения программы. Моделирование заключается в описании последовательности микроопераций, выполняемых функциональными блоками ЭВМ при интерпретации машинных инструкций, составляющих объектную программу и результатов, получаемых с помощью микроопераций. А также в описании последовательности проверок признаков, характеризующих результат микроопераций и позволяющих осуществлять выбор очередной микрооперации.