МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ

Индекс по учебному плану: **Б1.В.05.01**
Направление подготовки: **09.03.01 Информатика и вычислительная техника**
Квалификация: **Бакалавр**
Профиль подготовки: **Автоматизированные системы обработки информации и управления**
Типы задач профессиональной деятельности: **проектный, производственно-технологический**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

Введение

Данные методические указания предназначены для проведения лабораторных работ по дисциплине «Электронные вычислительные машины»

Лабораторные работы ориентированы на изучение процесса выполнения команд ЭВМ как последовательности микрокоманд, реализующих элементарные действия над операционными элементами процессора.

Лабораторные работы выполняются с помощью программной модели учебной ЭВМ, которая достаточно проста для освоения базовых понятий архитектуры (система команд, командный цикл, способы адресации, уровни памяти, способы взаимодействия процессора с памятью и внешними устройствами) современной ЭВМ [1].

№ п/п	№ раздела	Наименование лабораторных работ	Трудоемкость (час.)
1	1	Выполнение команд операций	6
2	1	Выполнение команд безусловного перехода	4
3	1	Выполнение команд условного перехода	4
4	1	Выполнение команд вызова подпрограмм	6
5	1	Одноразрядный сумматор	6
6	1	Дешифратор	6

1 Лабораторная работа №1 «Выполнение команд операций»

1.1 Цель работы

Целью настоящей лабораторной работы является изучение процесса выполнения различных команд операций с помощью последовательности микрокоманд.

1.2 Общие положения

В лабораторной работе №1 задаются различные команды выполнения операций в виде формулы на языке Ассемблер.

В соответствии с заданной формулой команды необходимо выполнить команду ассемблера в виде последовательности микрокоманд и определить все этапы выполнения каждой микрокоманды.

1.3 Задание на лабораторную работу №1

1. Ознакомиться с построением и работой программной модели ЭВМ (Приложение 2, Приложение 3).
2. Написать программу и произвести ассемблирование программы.
3. Записать последовательность микрокоманд для заданного набора команд.
4. Зафиксировать результаты выполнения команд в виде копии экрана компьютера.
5. Оформить отчет по лабораторной работе в соответствии с правилами оформления текстовых документов (Приложение 5).
6. Титульный лист отчета оформить в соответствии с Приложением 1.
7. Отчет по лабораторной работе разместить на прилагаемом к отчетам лазерном диске типа CD-RW.

1.4 Ход работы

В процессе работы выполнить следующие действия:

1) включить компьютер;

2) загрузить программную модель ЭВМ;

3) выбрать выполняемую команду;

4) выполнить команду в виде последовательности микрокоманд;

5) зафиксировать результаты выполнения команд в виде копий экрана компьютера.

1.5 Выполнение команд операций

В программной модели учебной ЭВМ использован стандартный интерфейс Windows, реализованный в нескольких окнах.

1.5.1Окно «Текст программы»

Ввести последовательность команд в рабочее поле и произвести компиляцию (рис. 1).

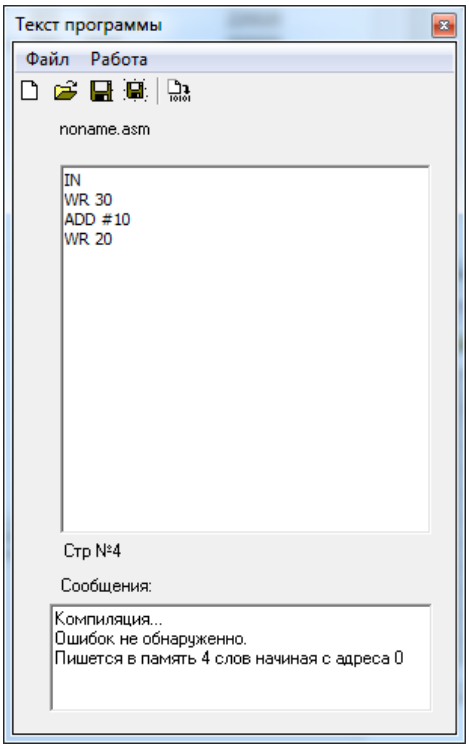


Рисунок 1 - Окно «Текст программы»

1.5.2 Окно «Микрокомандный уровень»

Окно «Микрокомандный уровень» (рис. 2) используется только в режиме микрокоманд. В это окно выводится мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

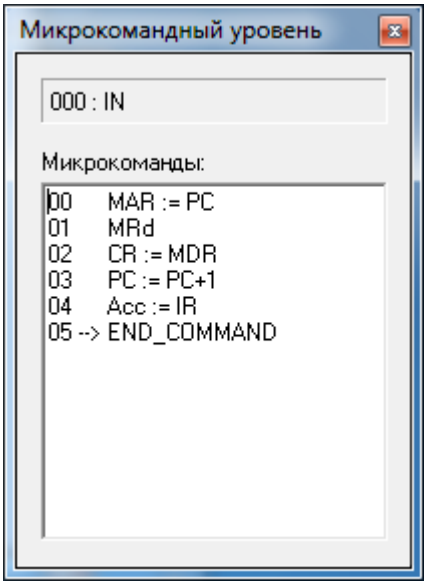


Рисунок 2 - Окно «Микрокомандный уровень»

1.5.3 Выполнение последовательности микрокоманд

Выполнить в режиме шаг введенную последовательность команд, фиксируя изменения значений объектов в Таблице 1.

Таблица 1 - Результаты выполнения последовательности микрокоманд

Адрес (PC)	Мнемокод	Микрокоманда	ОЗУ		CR			АЛУ	
			MAR	MDR	COP	TA	ADR	Acc	DR
000	IN	MAR:=PC	000	000000	00	0	000	000000	000000
		MRd	000						
		CR:=MDR		210030					
		PC:=PC+1			21	0	030		
001		MAR:=ADR							
		MDR := Acc	030						
		MWr		000000					
	WR 40	MAR := PC							
		MRd	001						
		CR := MDR		220040					

		PC := PC+1			22	0	040		
002		MAR :=ADR	040						
		MDR := Acc							
		MWr		000000					

1.6 Содержание отчета

Отчет по лабораторной работе №1 должен содержать:

1) оглавление;
2) описание цели работы;
3) описание хода работы в соответствии с данными методическими указаниями;
4) выводы по результатам выполнения работы;
5) список литературы;

Выводы

В результате выполнения данной лабораторной работы приобретены навыки выполнения различных команд операций с помощью последовательности микрокоманд.

Список литературы

1. Павлов, А.В. Архитектура вычислительных систем [Электронный ресурс] : учебн. пособие — Электрон. дан. — Санкт-Петербург: НИУ ИТМО, 2016. — 86 с. — Режим доступа: https://e.lanbook.com/book/91328. — Загл. с экрана.

2. Жмакин, А. П. Архитектура ЭВМ (+ CD-ROM) / А.П. Жмакин. - М.: БХВ-Петербург, 2010. - 352 с.

3. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник.-2-е изд., перераб. и доп./ Пятибратов А.П., Гудыно Л.П., Кириченко А.А.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2002. -512 с.

4. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем, - СПб.: Питер, 2009. - 720 с.

5. Брукшир Дж. Информатика и вычислительная техника. - СПб.: Питер, 2004. - 624 с.

6. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера /Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 с.

7. Лин, В. PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин. - М.: Радио и связь, 2015. - 320 с.

8. Пирогов, Владислав Ассемблер на примерах / Владислав Пирогов. - М.: БХВ-Петербург, 2013. - 416 с.

9. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е издание - СПб: Питер, 2013. - 816 с.

10. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К.Г. Финогенов. - М.: Горячая линия - Телеком, 2004. - 440 с.

11. Мир ПК. http://www.osp.ru

12. http://www.intuit.ru/department/hardware/archhard2/

2 Лабораторная работа №2

«Выполнение команд безусловного перехода»

2.1 Цель работы

Целью настоящей лабораторной работы является изучение процесса выполнения команд безусловного перехода с помощью последовательности микрокоманд.

2.2 Общие положения

В лабораторной работе №2 задаются различные команды безусловного перехода в виде формулы на языке Ассемблер.

В соответствии с заданной формулой команды необходимо выполнить команду ассемблера в виде последовательности микрокоманд и определить все этапы выполнения каждой микрокоманды.

2.3 Задание на лабораторную работу №2

1. Ознакомиться с построением и работой программной модели ЭВМ (Приложение 2, Приложение 3).
2. Написать программу и произвести ассемблирование программы.
3. Записать последовательность микрокоманд для заданного набора команд.
4. Зафиксировать результаты выполнения команд в виде копии экрана компьютера.
5. Оформить отчет по лабораторной работе в соответствии с правилами оформления текстовых документов (Приложение 5).
6. Титульный лист отчета оформить в соответствии с Приложением 1.
7. Отчет по лабораторной работе разместить на прилагаемом к отчетам лазерном диске типа CD-RW.

2.4 Ход работы

В процессе работы выполнить следующие действия:

1) включить компьютер;

2) загрузить программную модель ЭВМ;

3) выбрать выполняемую команду;

4) выполнить команду в виде последовательности микрокоманд;

5) зафиксировать результаты выполнения команд в виде копий экрана компьютера.

2.5 Выполнение команд безусловного перехода

В программной модели учебной ЭВМ использован стандартный интерфейс Windows, реализованный в нескольких окнах.

2.5.1Окно «Текст программы»

Ввести последовательность команд с использованием команды безусловного перехода в рабочее поле и произвести компиляцию (рис. 1).

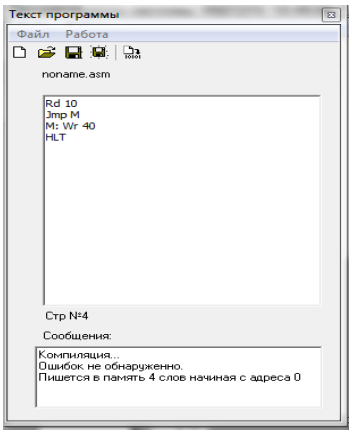


Рисунок 1 - Окно «Текст программы»

2.5.2 Окно «Микрокомандный уровень»

Выполнять последовательность команд в режиме «Шаг». При выполнении команды безусловного перехода перевести модель в режим «Микрокомандный уровень».

Окно «Микрокомандный уровень» (рис. 2) используется в режиме микрокоманд и показывает мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

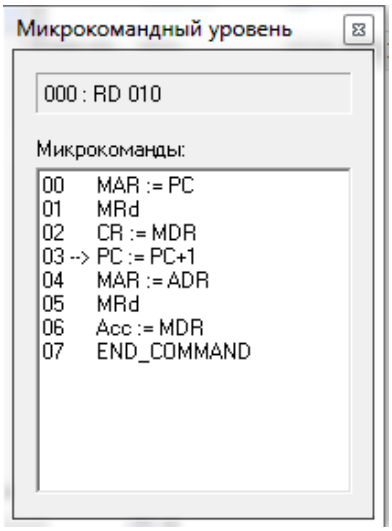


Рисунок 2 - Окно «Микрокомандный уровень»

2.5.3 Выполнение последовательности микрокоманд

Выполнить в режиме шаг введенную последовательность команд, фиксируя изменения значений объектов в Таблице 1.

Таблица 1 - Результаты выполнения последовательности микрокоманд

Адрес (PC)	Мнемокод	Микрокоманда	ОЗУ		CR			АЛУ	
			MAR	MDR	COP	TA	ADR	Acc	DR
000	RD 10	MAR := PC	000	000000	00	0	000	000000	000000
		MRd	000						
001		CR := MDR	010	210010	21	0	010	00005	
		PC := PC+1							
		MAR :=ADR							
		MRd							
		Acc := MDR		000005					
		END_COMMAND							
	JMP M	MAR := PC							
		MRd	001		10	0	002		
		CR := MDR		100002					
		PC := PC+1							
002		PC := ADR							
	WR 40	MAR := PC							

		MRd CR := MDR PC := PC+1 MAR :=ADR MDR := Acc	002 040	220040	22	0	040		
003									
		MWr		000000					

2. 6 Содержание отчета

Отчет по лабораторной работе №2 должен содержать:

1) оглавление;
2) описание цели работы;
3) описание хода работы в соответствии с данными методическими указаниями;
4) выводы по результатам выполнения работы;
5) список литературы;

Выводы

В результате выполнения данной лабораторной работы приобретены навыки выполнения команд безусловного перехода с помощью последовательности микрокоманд.

Список литературы

1. Павлов, А.В. Архитектура вычислительных систем [Электронный ресурс] : учебн. пособие — Электрон. дан. — Санкт-Петербург: НИУ ИТМО, 2016. — 86 с. — Режим доступа: https://e.lanbook.com/book/91328. — Загл. с экрана.

2. Жмакин, А. П. Архитектура ЭВМ (+ CD-ROM) / А.П. Жмакин. - М.: БХВ-Петербург, 2010. - 352 с.

3. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник.-2-е изд., перераб. и доп./ Пятибратов А.П., Гудыно Л.П., Кириченко А.А.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2002. -512 с.

4. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем, - СПб.: Питер, 2009. - 720 с.

5. Брукшир Дж. Информатика и вычислительная техника. - СПб.: Питер, 2004. - 624 с.

6. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера /Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 с.

7. Лин, В. PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин. - М.: Радио и связь, 2015. - 320 с.

8. Пирогов, Владислав Ассемблер на примерах / Владислав Пирогов. - М.: БХВ-Петербург, 2013. - 416 с.

9. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е издание - СПб: Питер, 2013. - 816 с.

10. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К.Г. Финогенов. - М.: Горячая линия - Телеком, 2004. - 440 с.

11. Мир ПК. http://www.osp.ru

12. http://www.intuit.ru/department/hardware/archhard2/

3 Лабораторная работа №3

«Выполнение команд условного перехода»

3.1 Цель работы

Целью настоящей лабораторной работы является изучение процесса выполнения команд условного перехода с помощью последовательности микрокоманд.

3.2 Общие положения

В лабораторной работе №3 задаются различные команды условного перехода в виде формулы на языке Ассемблер.

В соответствии с заданной формулой команды необходимо выполнить команду ассемблера в виде последовательности микрокоманд и определить все этапы выполнения каждой микрокоманды.

3.3 Задание на лабораторную работу №3

1. Ознакомиться с построением и работой программной модели ЭВМ (Приложение 2, Приложение 3).
2. Написать программу и произвести ассемблирование программы.
3. Записать последовательность микрокоманд для заданного набора команд.
4. Зафиксировать результаты выполнения команд в виде копии экрана компьютера.
5. Оформить отчет по лабораторной работе в соответствии с правилами оформления текстовых документов (Приложение 5).
6. Титульный лист отчета оформить в соответствии с Приложением 1.
7. Отчет по лабораторной работе разместить на прилагаемом к отчетам лазерном диске типа CD-RW.

3.4 Ход работы

В процессе работы выполнить следующие действия:

1) включить компьютер;

2) загрузить программную модель ЭВМ;

3) выбрать выполняемую команду;

4) выполнить команду в виде последовательности микрокоманд;

5) зафиксировать результаты выполнения команд в виде копий экрана компьютера.

3.5 Выполнение команд условного перехода

В программной модели учебной ЭВМ использован стандартный интерфейс Windows, реализованный в нескольких окнах.

3.5.1Окно «Текст программы»

Ввести последовательность команд с использованием команды условного перехода в рабочее поле и произвести компиляцию (рис. 1).

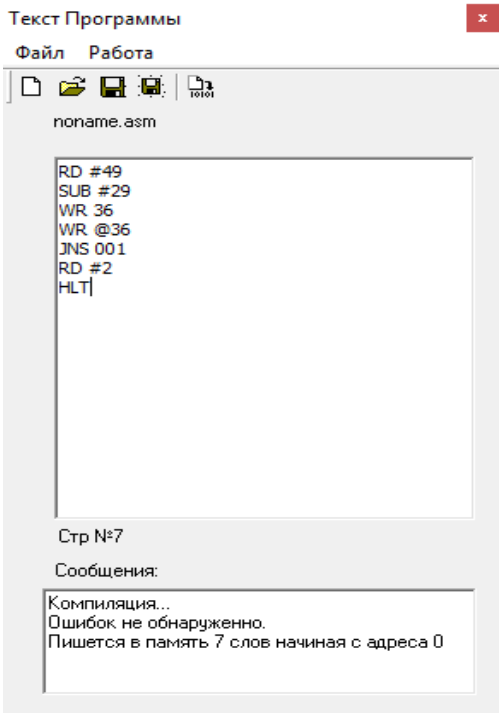


Рисунок 1 - Окно «Текст программы»

3.5.2 Окно «Микрокомандный уровень»

Выполнять последовательность команд в режиме «Шаг». При выполнении команды условного перехода перевести модель в режим «Микрокомандный уровень».

Окно «Микрокомандный уровень» (рис. 2) используется в режиме микрокоманд и показывает мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

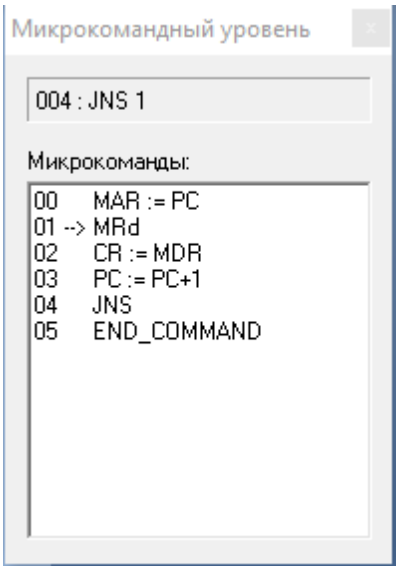


Рисунок 2 - Окно «Микрокомандный уровень»

3.5.3 Выполнение последовательности микрокоманд

Выполнить в режиме шаг введенную последовательность команд, фиксируя изменения значений объектов в Таблице 1.

Таблица 1 - Результаты выполнения последовательности микрокоманд

Адрес (PC)	Мнемокод	Микрокоманда	ОЗУ		CR			АЛУ			
			MAR	MDR	COP	TA	ADR	Acc	DR	RA	RB
000	RD #49	MAR := PC	000								
		MRd		211049							
		CR := MDR			21	1	049				
001		PC := PC+1									
		Acc := ADR						000049			
	SUB #29	MAR := PC	001								
		MRd		240029							
		CR := MDR			24	0	029				

002		PC := PC+1									
		DR :=ADR							000029		
		ALU< --COP						000049			
		StartALU						000020			
	JNS 001	MAR := PC	002								
		MRd		140001							
		CR := MDR			14	0	001				
003		PC := PC+1									
		JNS						100009			
	RD #002	MAR:=PC	003								
		MRd		211002							
		CR:=MDR			21	1	002				
004		PC:=PC+1									
		Acc:=ADR						002			

3. 6 Содержание отчета

Отчет по лабораторной работе №3 должен содержать:

1) оглавление;
2) описание цели работы;
3) описание хода работы в соответствии с данными методическими указаниями;
4) выводы по результатам выполнения работы;
5) список литературы;

Выводы

В результате выполнения данной лабораторной работы приобретены навыки выполнения команд условного перехода с помощью последовательности микрокоманд.

Список литературы

1. Павлов, А.В. Архитектура вычислительных систем [Электронный ресурс] : учебн. пособие — Электрон. дан. — Санкт-Петербург: НИУ ИТМО, 2016. — 86 с. — Режим доступа: https://e.lanbook.com/book/91328. — Загл. с экрана.

2. Жмакин, А. П. Архитектура ЭВМ (+ CD-ROM) / А.П. Жмакин. - М.: БХВ-Петербург, 2010. - 352 с.

3. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник.-2-е изд., перераб. и доп./ Пятибратов А.П., Гудыно Л.П., Кириченко А.А.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2002. -512 с.

4. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем, - СПб.: Питер, 2009. - 720 с.

5. Брукшир Дж. Информатика и вычислительная техника. - СПб.: Питер, 2004. - 624 с.

6. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера /Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 с.

7. Лин, В. PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин. - М.: Радио и связь, 2015. - 320 с.

8. Пирогов, Владислав Ассемблер на примерах / Владислав Пирогов. - М.: БХВ-Петербург, 2013. - 416 с.

9. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е издание - СПб: Питер, 2013. - 816 с.

10. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К.Г. Финогенов. - М.: Горячая линия - Телеком, 2004. - 440 с.

11. Мир ПК. http://www.osp.ru

12. http://www.intuit.ru/department/hardware/archhard2/

4 Лабораторная работа №4
«Выполнение команд вызова подпрограмм”

4.1 Цель работы

Целью настоящей лабораторной работы является изучение процесса выполнения команды вызова подпрограмм с помощью последовательности микрокоманд.

4.2 Общие положения

В лабораторной работе №4 задаются команды вызова подпрограмм в виде формулы на языке Ассемблер.

В соответствии с заданной формулой команды необходимо выполнить команду ассемблера в виде последовательности микрокоманд и определить все этапы выполнения каждой микрокоманды.

4.3 Задание на лабораторную работу №4

1. Ознакомиться с построением и работой программной модели ЭВМ (Приложение 2, Приложение 3).
2. Написать программу и произвести ассемблирование программы.
3. Записать последовательность микрокоманд для заданного набора команд.
4. Зафиксировать результаты выполнения команд в виде копии экрана компьютера.
5. Оформить отчет по лабораторной работе в соответствии с правилами оформления текстовых документов (Приложение 5).
6. Титульный лист отчета оформить в соответствии с Приложением 1.
7. Отчет по лабораторной работе разместить на прилагаемом к отчетам лазерном диске типа CD-RW.

4.4 Ход работы

В процессе работы выполнить следующие действия:

1) включить компьютер;

2) загрузить программную модель ЭВМ;

3) выбрать выполняемую команду;

4) выполнить команду в виде последовательности микрокоманд;

5) зафиксировать результаты выполнения команд в виде копий экрана компьютера.

4.5 Выполнение команд вызова подпрограмм

В программной модели учебной ЭВМ использован стандартный интерфейс Windows, реализованный в нескольких окнах.

4.5.1Окно «Текст программы»

Ввести последовательность команд с использованием команды вызова подпрограмм в рабочее поле и произвести компиляцию (рис. 1).

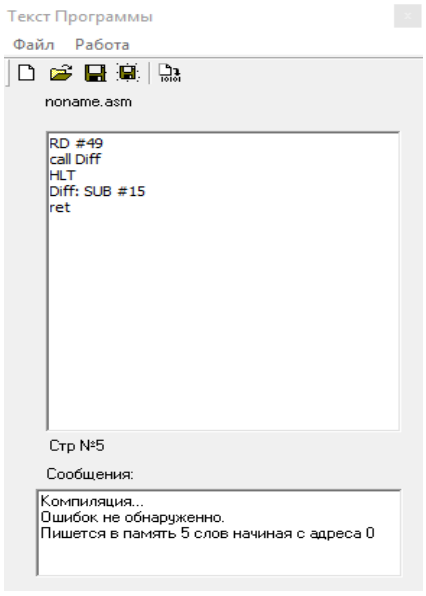


Рисунок 1 - Окно «Текст программы»

4.5.2 Окно «Микрокомандный уровень»

Выполнять последовательность команд в режиме «Шаг». При выполнении команды условного перехода перевести модель в режим «Микрокомандный уровень».

Окно «Микрокомандный уровень» (рис. 2) используется в режиме микрокоманд и показывает мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

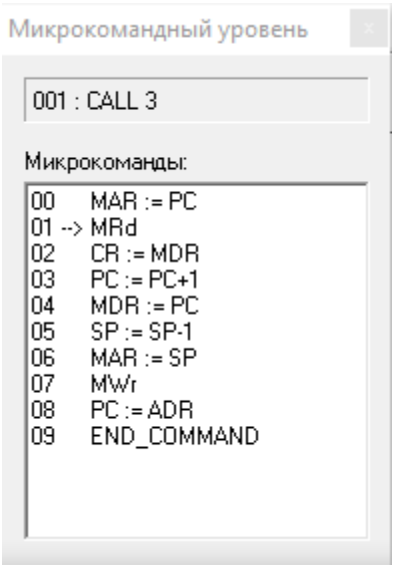


Рисунок 2 - Окно «Микрокомандный уровень»

4.5.3 Выполнение последовательности микрокоманд

Выполнить в режиме шаг введенную последовательность команд, фиксируя изменения значений объектов в Таблице 1.

Таблица 1 - Результаты выполнения последовательности микрокоманд

Адрес (PC)	Мнемокод	Микрокоманда	ОЗУ		CR			АЛУ			
			MAR	MDR	COP	TA	ADR	Acc	DR	RA	RB
000	RD #49	MAR := PC	000								
		MRd		211049							
		CR := MDR			21	1	049				
001		PC := PC+1									
		Acc := ADR						000049			
	CALL 3	MAR := PC	001								
		MRd		190003							
		CR := MDR			19	0	003				

002		PC := PC+1									
		MDR := PC		000002							
		SP:=SP-1									
		MAR :=SP	999								
		MWr									
003		PC:=ADR									
	SUB #15	MAR := PC	003								
		MRd		241015							
		CR := MDR			24	1	015				
004		PC:=PC+1									
		DR := ADR							015		
		ALU <-- COP									
		Start ALU						000034			
	RET	MAR := PC	004								
		MRd		080000							
		CR := MDR			08	0	000				
005		PC:=PC+1									
		MAR := SP	999								
		MRd		000002							
002		PC := MDR									
		SP := SP+1									

4. 6 Содержание отчета

Отчет по лабораторной работе №4 должен содержать:

1) оглавление;
2) описание цели работы;
3) описание хода работы в соответствии с данными методическими указаниями;
4) выводы по результатам выполнения работы;
5) список литературы;

Выводы

В результате выполнения данной лабораторной работы приобретены навыки выполнения команд вызова подпрограмм с помощью последовательности микрокоманд.

Список литературы

1. Павлов, А.В. Архитектура вычислительных систем [Электронный ресурс] : учебн. пособие — Электрон. дан. — Санкт-Петербург: НИУ ИТМО, 2016. — 86 с. — Режим доступа: https://e.lanbook.com/book/91328. — Загл. с экрана.

2. Жмакин, А. П. Архитектура ЭВМ (+ CD-ROM) / А.П. Жмакин. - М.: БХВ-Петербург, 2010. - 352 с.

3. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник.-2-е изд., перераб. и доп./ Пятибратов А.П., Гудыно Л.П., Кириченко А.А.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2002. -512 с.

4. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем, - СПб.: Питер, 2009. - 720 с.

5. Брукшир Дж. Информатика и вычислительная техника. - СПб.: Питер, 2004. - 624 с.

6. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера /Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 с.

7. Лин, В. PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин. - М.: Радио и связь, 2015. - 320 с.

8. Пирогов, Владислав Ассемблер на примерах / Владислав Пирогов. - М.: БХВ-Петербург, 2013. - 416 с.

9. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е издание - СПб: Питер, 2013. - 816 с.

10. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К.Г. Финогенов. - М.: Горячая линия - Телеком, 2004. - 440 с.

11. Мир ПК. http://www.osp.ru

12. http://www.intuit.ru/department/hardware/archhard2/

5 Лабораторная работа №5 «Одноразрядный сумматор ”

5.1 Цель работы

Целью лабораторной работы является изучение моделирование работы одноразрядного сумматора.

5.2 Общие положения

В лабораторной работе №5 в соответствии с таблицей истинности работы одноразрядного сумматора и формулой логической функции, задающей работу сумматора, строится электронная модель одноразрядного сумматора

5.3 Структура и принципы программы Workbench

Workbench — штатная графическая оболочка AmigaOS. Название оболочки является метафорой словосочетания «рабочий стол» (дословно — «верстак»), поэтому аналогия была продолжена и дальше: каталоги изображены ящиками рабочего стола, исполняемые файлы — инструментами, данные — проектами, а остальные составляющие GUI — разного рода приспособлениями. Большинство приложений AmigaOS используют всплывающие меню, традиционно начинающиеся со слова Project («Проект»), а не File («Файл»), как на других платформах. Строго говоря, термин Workbench относится только к основному файловому менеджеру AmigaOS, хотя часто используется для обозначения всей той её части, которая расположена вне ПЗУ.

5.4 Задание на лабораторную работу №5

1. Ознакомиться с работой программы моделирования электронных схем Workbench
2. Построить таблицу истинности работы одноразрядного сумматора
3. Написать формулу логической функции, задающей работу сумматора

4. Построить электронную модель одноразрядного сумматора с помощью программы Workbench.

5. Зафиксировать результаты работы схемы

5.5 Ход работы

В процессе работы выполнены следующие действия:

1) включить компьютер;
2) запускаем программу Workbench;
3) построить функциональную схему «Одноразрядный сумматор»;
4) зафиксировать результаты работы схемы.

5.6 Построение таблицы истинности

Построим таблицу (табл. 1) истинности для устройства реализующего арифметическую операцию сложения. Операция «+» бинарная, поэтому сумматор должен иметь два входа (A и B). В результате сложения двух одноразрядных двоичных чисел может получиться двухразрядное число (с переносом в следующий разряд). Значит, устройство должно иметь два выхода (P - перенос в следующий разряд, S - результат, остающийся в текущем разряде).

Таблица 1 – Таблица истинности

Входы		Выходы	
A	B	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

5.7 Построение формулы логической функции работы сумматора

По данной таблице истинности построим СДНФ:

1. Для переноса в старший разряд: P = A ∧ B

2. Для текущего разряда: S = ¬ A ∧ B ∨ A ∧ ¬ B

Преобразуем логическую формулу для S:

(¬ A • B) + (A • ¬ B) = (¬ A • A) + (¬ A • B) + (A • ¬ B) + (¬ B • B) =

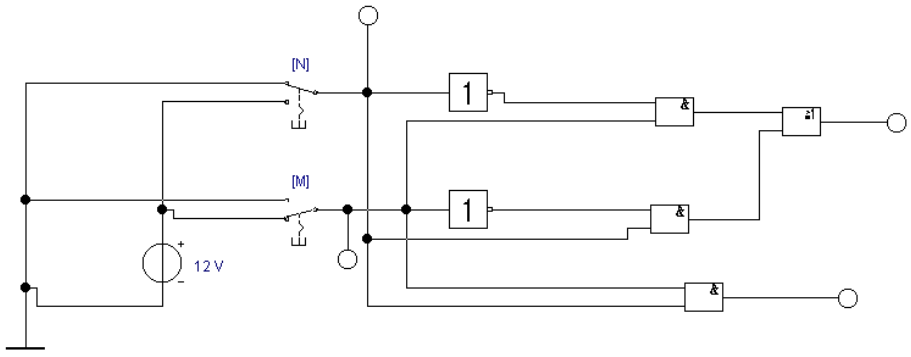
= ¬ A • (A + B) + ¬ B • (A + B) = (A + B) • ¬ (A • B)

С учетом формулы для переноса имеем:

S = (A + B) • ¬ (A • B) = (A + B) • ¬ P

5.8 Синтез функциональной схемы сумматора

Схема сумматора, приведена на рисунке 1. В данной схеме используется многовходовые логические элементы И и ИЛИ.



5.9 Содержание отчета

Отчет по лабораторной работе №5 должен содержать:

6) оглавление;
7) описание цели работы;
8) описание хода работы в соответствии с данными методическими указаниями;
9) выводы по результатам выполнения работы;
10) список литературы;

Выводы

В результате выполнения данной лабораторной работы приобретены навыки выполнения построения одноразрядного сумматора в программе Work-bench.

Список литературы

1. Павлов, А.В. Архитектура вычислительных систем [Электронный ресурс] : учебн. пособие — Электрон. дан. — Санкт-Петербург: НИУ ИТМО, 2016. — 86 с. — Режим доступа: https://e.lanbook.com/book/91328. — Загл. с экрана.

2. Жмакин, А. П. Архитектура ЭВМ (+ CD-ROM) / А.П. Жмакин. - М.: БХВ-Петербург, 2010. - 352 с.

3. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник.-2-е изд., перераб. и доп./ Пятибратов А.П., Гудыно Л.П., Кириченко А.А.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2002. -512 с.

4. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем, - СПб.: Питер, 2009. - 720 с.

5. Брукшир Дж. Информатика и вычислительная техника. - СПб.: Питер, 2004. - 624 с.

6. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера /Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 с.

7. Лин, В. PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин. - М.: Радио и связь, 2015. - 320 с.

8. Пирогов, Владислав Ассемблер на примерах / Владислав Пирогов. - М.: БХВ-Петербург, 2013. - 416 с.

9. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е издание - СПб: Питер, 2013. - 816 с.

10. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К.Г. Финогенов. - М.: Горячая линия - Телеком, 2004. - 440 с.

11. Мир ПК. http://www.osp.ru

12. http://www.intuit.ru/department/hardware/archhard2/

6 Лабораторная работа №6 «Дешифратор”

6.1 Цель работы

Целью лабораторной работы является изучение моделирование работы дешифратора.

6.2 Общие положения

В лабораторной работе №6 в соответствии с таблицей истинности работы дешифратора и формулой логической функции, задающей работу дешифратора, строится электронная модель дешифратора

6.3 Структура и принципы программы Workbench

Workbench — штатная графическая оболочка AmigaOS. Название оболочки является метафорой словосочетания «рабочий стол» (дословно — «верстак»), поэтому аналогия была продолжена и дальше: каталоги изображены ящиками рабочего стола, исполняемые файлы — инструментами, данные — проектами, а остальные составляющие GUI — разного рода приспособлениями. Большинство приложений AmigaOS используют всплывающие меню, традиционно начинающиеся со слова Project («Проект»), а не File («Файл»), как на других платформах. Строго говоря, термин Workbench относится только к основному файловому менеджеру AmigaOS, хотя часто используется для обозначения всей той её части, которая расположена вне ПЗУ.

6.4 Задание на лабораторную работу №6

1. Ознакомиться с работой программы моделирования электронных схем Workbench
2. Построить таблицу истинности работы дешифратора
3. Написать формулу логической функции, задающей работу дешифратора

4. Построить электронную модель дешифратора с помощью программы Workbench.

5. Зафиксировать результаты работы схемы

6.5 Ход работы

В процессе работы выполнены следующие действия:

1) включить компьютер;
2) запускаем программу Workbench;
3) построить функциональную схему дешифратора;
4) зафиксировать результаты работы схемы.

6.6 Построение таблицы истинности

Дешифратор - это логическая схема, преобразующая двоичный код в унарный, когда только на одном из всех выходов появляется активный сигнал

Номер этого активного выхода в десятичном коде совпадает с двоичным кодом, подаваемым на входные линии дешифратора

Таблица 1 – Таблица истинности

Входные сигналы			Выходные сигналы							
X_3	**X_2**	**X_1**	**$У_7$**	**$У_6$**	**$У_5$**	**$У_4$**	**$У_3$**	**$У_2$**	**$У_1$**	**$У_0$**
0	0	0	0	0	0	0	0	0	0	**1**
0	0	1	0	0	0	0	0	0	**1**	0
0	1	0	0	0	0	0	0	**1**	0	0
0	1	1	0	0	0	0	**1**	0	0	0
1	0	0	0	0	0	**1**	0	0	0	0
1	0	1	0	0	**1**	0	0	0	0	0
1	1	0	0	**1**	0	0	0	0	0	0
1	1	1	**1**	0	0	0	0	0	0	0

6.7 Построение формулы логической функции работы дешифратора

По данной таблице истинности построим СДНФ:

$$Y_0 = \overline{\overline{X}_2 \overline{X}_1} = \overline{X_2 \vee X_1}$$

$$Y_1 = \overline{\overline{X}_2 X_1} = \overline{X_2 \vee \overline{X}_1}$$

$$Y_2 = \overline{X_2 \overline{X}_1} = \overline{\overline{X}_2 \vee X_1}$$

$$Y_3 = \overline{X_2 X_1} = \overline{\overline{X}_2 \vee \overline{X}_1}$$

6.8 Синтез функциональной схемы дешифратора

Схема дешифратора приведена на рисунке 1. В данной схеме используется многовходовые логические элементы И.

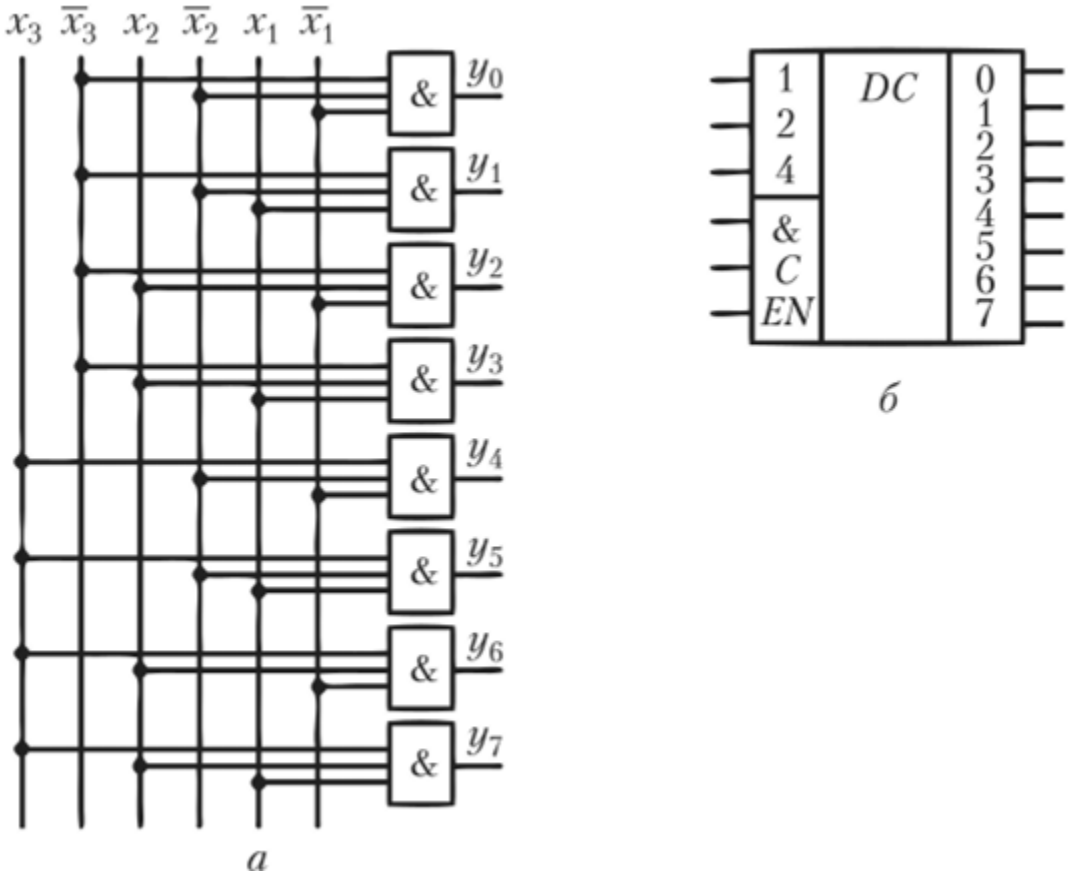


Рисунок - 1 Функциональная схема дешифратора

Электронная модель дешифратора приведена на рисунке 2

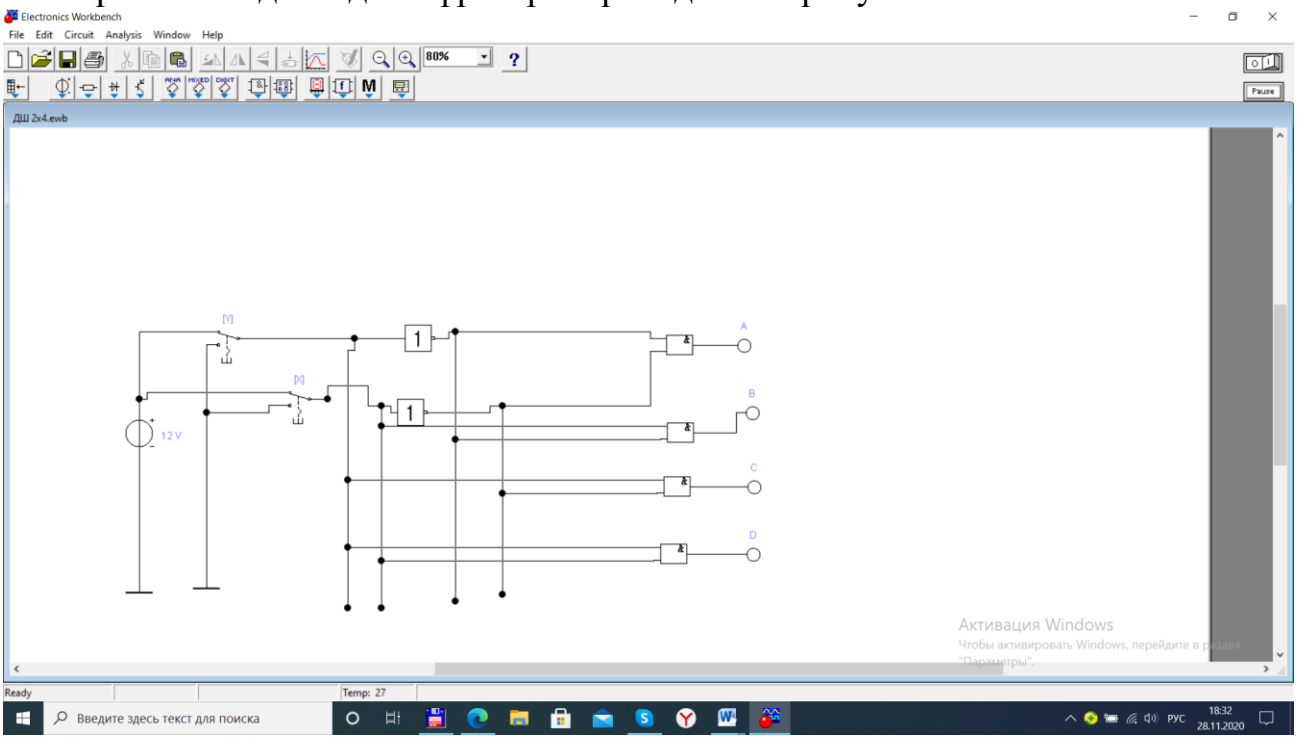


Рисунок 2 – Электронная модель дешифратора

5.9 Содержание отчета

Отчет по лабораторной работе №5 должен содержать:

1) оглавление;
2) описание цели работы;
3) описание хода работы в соответствии с данными методическими указаниями;
4) выводы по результатам выполнения работы;
5) список литературы;

Выводы

В результате выполнения данной лабораторной работы приобретены навыки выполнения построения дешифратора в программе Workbench.

Список литературы

1. Павлов, А.В. Архитектура вычислительных систем [Электронный ресурс] : учебн. пособие — Электрон. дан. — Санкт-Петербург: НИУ ИТМО, 2016. — 86 с. — Режим доступа: https://e.lanbook.com/book/91328. — Загл. с экрана.

2. Жмакин, А. П. Архитектура ЭВМ (+ CD-ROM) / А.П. Жмакин. - М.: БХВ-Петербург, 2010. - 352 с.

3. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник.-2-е изд., перераб. и доп./ Пятибратов А.П., Гудыно Л.П., Кириченко А.А.; Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2002. -512 с.

4. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем, - СПб.: Питер, 2009. - 720 с.

5. Брукшир Дж. Информатика и вычислительная техника. - СПб.: Питер, 2004. - 624 с.

6. Брэй, Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера /Барри Брэй. - М.: МК-Пресс, Корона-Век, 2008. - 576 с.

7. Лин, В. PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин. - М.: Радио и связь, 2015. - 320 с.

8. Пирогов, Владислав Ассемблер на примерах / Владислав Пирогов. - М.: БХВ-Петербург, 2013. - 416 с.

9. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е издание - СПб: Питер, 2013. - 816 с.

10. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К.Г. Финогенов. - М.: Горячая линия - Телеком, 2004. - 440 с.

11. Мир ПК. http://www.osp.ru

12. http://www.intuit.ru/department/hardware/archhard2/

Приложение 1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»
Кафедра компьютерных и телекоммуникационных систем

Отчет

по лабораторной работе № 1

по дисциплине «Электронные вычислительные машины»

Выполнение команд операций

Выполнил ст. группы 21302 Карпов Е. В.

Проверил к. т. н., доцент Белош В.В.

г. Чистополь

2023г.

Отчет

по лабораторной работе № 2

по дисциплине «Электронные вычислительные машины»

Выполнение команд безусловного перехода

Отчет

по лабораторной работе № 3

по дисциплине «Электронные вычислительные машины»

Выполнение команд условного перехода

Отчет

по лабораторной работе № 4

по дисциплине «Электронные вычислительные машины»

Выполнение команд вызова подпрограмм

Отчет

по лабораторной работе № 5

по дисциплине «Электронные вычислительные машины»

Одноразрядный сумматор

Отчет

по лабораторной работе № 6

по дисциплине «Электронные вычислительные машины»

Дешифратор

Приложение 2

Таблица команд учебной ЭВМ

Ст. / Мл.	0	1	2	3	4
0	NOP	JMP		MOV	
1	IN	JZ	RD	RD	RDI
2	OUT	JNZ	WR	WR	
3	IRET	JS	ADD	ADD	ADI
4	WRRB	JNS	SUB	SUB	SBI
5	WRSP	JO	MUL	MUL	MULI
6	PUSH	JNO	DIV	DIV	DIVI
7	POP	JRNZ		IN	
8	RET	INT	EI	OUT	
9	HLT	CALL	DI		

Типы адресации, их коды и обозначение

Обозначение	Код	Тип адресации	Пример команды
	0	Прямая (регистровая)	ADD 23 (ADD R3)
#	1	Непосредственная	ADD #33
@	2	Косвенная	ADD @33
[ ]	3	Относительная	ADD [33]
@R	4	Косвенно-регистровая	ADD @R3
@R+	5	Индексная с постинкрементом	ADD @R3+
-@R	6	Индексная с преддекрементом	ADD -@R3

Система команд учебной ЭВМ

КОП	Мнемо-код	Название	Действие
00	NOP	Пустая операция	Нет
01	IN	Ввод	Acc ← IR
02	OUT	Вывод	OR ← Acc
03	IRET	Возврат из прерывания	FLAGS.PC ←M(SP);INC (SP)
04	WRRB	Загрузка RB	RB ← CR [ADR]
05	WRSP	Загрузка SP	SP ← CR [ADR]
06	PUSH	Поместить в стек	DEC(SP); M(SP) ← R
07	POP	Извлечь из стека	R → M(SP); INC (SP)
08	RET	Возврат	PC → M(SP); INC (SP)
09	HLT	Стоп	Конец командных циклов
10	JMP	Безусловный переход	PC ← CR [ADR]
11	JZ	Переход, если 0	if Acc = 0 then PC ← CR [ADR]

12	JNZ	Переход, если не 0	if Acc ≠ 0 then PC ← CR [ADR]
13	JS	Переход, если отрицательно	if Acc < 0 then PC ← CR [ADR]
14	JNS	Переход, если положительно	if Acc≥ 0 then PC ← CR [ADR]
15	JO	Переход, если переполнение	if | Acc | >0 then PC ← CR [ADR]
16	JNO	Переход, если нет переполнения	if | Acc | ≤ 0 then PC ← CR [ADR]
17	JRNZ	Цикл	DEC(R); if R >0 then PC← CR [ADR]
18	INT	Программное прерывание	DEC(SP); M(SP) ← FLAGS.PC; PC ← M(V)
19	CALL	Вызов подпрограммы	DEC(SP); M(SP)← PC; PC ←CR(ADR)
20	Нет		
21	RD	Чтение	Acc ← DD
22	WR	Запись	M (*)← Acc
23	ADD	Сложение	Acc ← Acc + DD
24	SUB	Вычитание	Acc ← Acc - DD
25	MUL	Умножение	Acc ← Acc x DD
26	DIV	Деление	Acc ← Acc /DD
27	Нет		
28	EI	Разрешить прерывание	IF ← 1
29	DI	Запретить прерывание	IF ← 0
30	MOV	Пересылка	R1 ← R2
31	RD	Чтение	Acc ← R*
32	WR	Запись	R* ← Acc
33	ADD	Сложение	Acc ← Acc+ R*
34	SUB	Вычитание	Acc ← Acc – R*
35	MUL	Умножение	Acc ← Acc - R*
36	DIV	Деление	Acc ← Acc/R*
37	IN	Ввод	Acc ← BY (CR[ADR*])
38	OUT	Вывод	BY (CR[ADR*]) ← Acc
39	Нет		
40	Нет		
41	RDI	Чтение	Acc ← I
42	Нет		
43	JNS	Сложение	Acc ← Acc +I
44	JO	Вычитание	Acc ← Acc - I
45	JNO	Умножение	Acc ← Acc x I
46	JRNZ	Деление	Acc ← Acc/I

Обозначения:

DD— данные, формируемые командой в качестве (второго) операнда: прямо или косвенно адресуемая ячейка памяти или трехразрядный непосредственный операнд;

R* — содержимое регистра или косвенно адресуемая через регистр ячейка памяти;

ADR* — два младших разряда ADR поля регистра CR;
V — адрес памяти, соответствующий вектору прерывания;
M(*) — ячейка памяти, прямо или косвенно адресуемая в команде;
I — пятиразрядный непосредственный операнд со знаком.

Приложение 3

1 Описание архитектуры учебной ЭВМ

Современные процессоры и операционные системы — не слишком благоприятная среда для начального этапа изучения архитектуры ЭВМ.

Одним из решений этой проблемы может быть создание программных моделей учебных ЭВМ, которые, с одной стороны, достаточно просты, чтобы обучаемый мог освоить базовые понятия архитектуры (система команд, командный цикл, способы адресации, уровни памяти, способы взаимодействия процессора с памятью и внешними устройствами), с другой стороны — архитектурные особенности модели должны соответствовать тенденциям развития современных ЭВМ.

Программная модель позволяет реализовать доступ к различным элементам ЭВМ, обеспечивая удобство и наглядность. С другой стороны, модель позволяет игнорировать те особенности работы реальной ЭВМ, которые на данном уровне рассмотрения не являются существенными.

Далее приводится описание программной модели учебной ЭВМ, предназначенной для начальных этапов изучения архитектуры (в т. ч. на младших курсах вуза и даже в школе). Именно этим объясняется использование в модели десятичной системы счисления для кодирования команд и представления данных.

1.1 Структура ЭВМ

Моделируемая ЭВМ включает процессор, оперативную (ОЗУ) и сверхоперативную память, устройство ввода (УВв) и устройство вывода (УВыв). Процессор, в свою очередь, состоит из центрального устройства управления (УУ), арифметического устройства (АУ) и системных регистров (CR, PC, SP и др.). Структурная схема ЭВМ показана на рис. 8.1.

В ячейках ОЗУ хранятся команды и данные. Емкость ОЗУ составляет 1000 ячеек. По сигналу MWr выполняется запись содержимого регистра данных (MDR) в ячейку памяти с адресом, указанным в регистре адреса (MAR). По сигналу MRd происходит считывание — содержимое ячейки памяти с адресом, содержащимся в MAR, передается в MDR.

Сверхоперативная память с прямой адресацией содержит десять регистров общего назначения R0—R9. Доступ к ним осуществляется (аналогично доступу к ОЗУ) через регистры RAR и RDR.

АУ осуществляет выполнение одной из арифметических операций, определяемой кодом операции (COP), над содержимым аккумулятора (Acc) и регистра операнда (DR). Результат операции всегда помещается в Acc. При завершении выполнения операции АУ вырабатывает сигналы признаков результата: Z (равен 1, если результат равен нулю); S (равен 1, если результат отрицателен); OV (равен 1, если при выполнении операции произошло переполнение разрядной сетки). В случаях, когда эти условия не выполняются, соответствующие сигналы имеют нулевое значение.

В модели ЭВМ предусмотрены внешние устройства двух типов. Во-первых, это регистры IR и OR, которые могут обмениваться с аккумулятором с помощью безадресных команд IN (Асе := IR) и OUT (OR := Acc). Во-вторых, это набор моделей внешних устройств, которые могут подключаться к системе и взаимодействовать с ней в соответствии с заложенными в моделях алгоритмами. Каждое внешнее устройство имеет ряд программно-доступных регистров, может иметь собственный *обозреватель* (окно видимых элементов). Подробнее эти внешние устройства описаны в разд. 8.6.

УУ осуществляет выборку команд из ОЗУ в последовательности, определяемой естественным порядком выполнения команд (т. е. в порядке возрастания адресов команд в ОЗУ) или командами передачи управления; выборку из ОЗУ операндов, задаваемых адресами команды; инициирование выполнения операции, предписанной командой; останов или переход к выполнению следующей команды.

В качестве сверхоперативной памяти в модель включены регистры общего назначения (РОН), и может подключаться модель кэш-памяти.

В состав УУ ЭВМ входят:

PC — счетчик адреса команды, содержащий адрес текущей команды;

CR — регистр команды, содержащий код команды;

RB — регистр базового адреса, содержащий базовый адрес;

SP — указатель стека, содержащий адрес верхушки стека;

RA— регистр адреса, содержащий исполнительный адрес при косвенной адресации.

Регистры Acc, DR, IR, OR, CR и все ячейки ОЗУ и РОН имеют длину 6 десятичных разрядов, регистры PC, SP, RA и RB — 3 разряда.

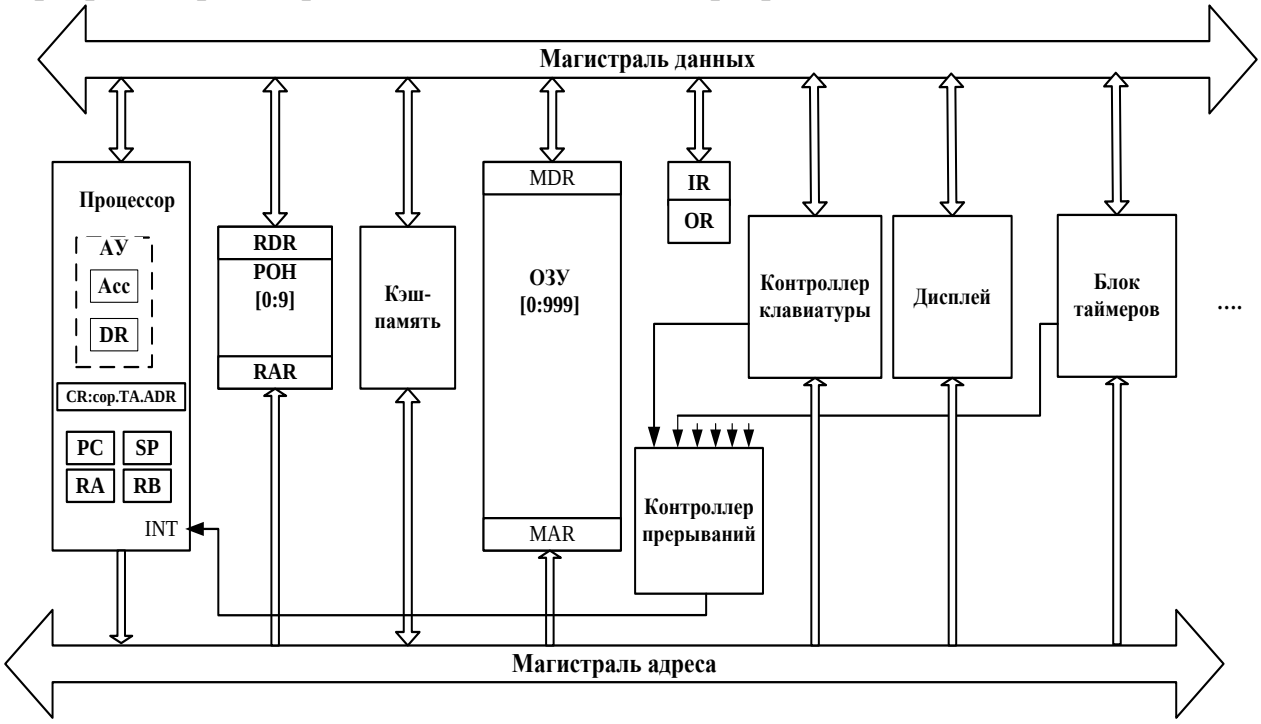


Рисунок 1 - Общая структура учебной ЭВМ

1.2 Представление данных в модели

Данные в ЭВМ представляются в формате, показанном на рис.2. Это целые десятичные числа, изменяющиеся в диапазоне "- 99 999...+ 99 999", содержащие знак и 5 десятичных цифр.

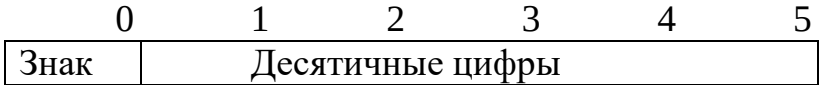


Рисунок 2 - Формат десятичных данных учебной ЭВМ

Старший разряд слова данных используется для кодирования знака: плюс (+) изображается как 0, минус (-) – как 1. Если результат арифметической операции выходит за пределы указанного диапазона, то говорят, что произошло переполнение разрядной сетки. АЛУ в этом случае вырабатывает сигнал переполнения OV = 1. Результатом операции деления является целая часть частного. Деление на ноль вызывает переполнение.

1.3 Система команд

При рассмотрении системы команд ЭВМ обычно анализируют три аспекта: форматы, способы адресации и систему операций.

1.3.1 Форматы команд

Большинство команд учебной ЭВМ являются одноадресными или безадресными, длиной в одно машинное слово (6 разрядов). Исключение составляют двухсловные команды с непосредственной адресацией и команда MOV, являющаяся двухадресной.

В форматах команд выделяется три поля:

- два старших разряда [0: 1] определяют код операции COP;
- разряд 2 может определять тип адресации (в одном случае (формат 5а) он определяет номер регистра);
- разряды [3:5] могут определять прямой или косвенный адрес памяти, номер регистра (в команде MOV номера двух регистров), адрес перехода или короткий непосредственный операнд. В двухсловных командах непосредственный операнд занимает поле [6:11].

Полный список форматов команд показан на рис.3, где приняты следующие обозначения:

COP — код операции;
ADR — адрес операнда в памяти;
ADC — адрес перехода;
I — непосредственный операнд;
R, Rl, R2 — номер регистра;
TA — тип адресации;
X — разряд не используется.

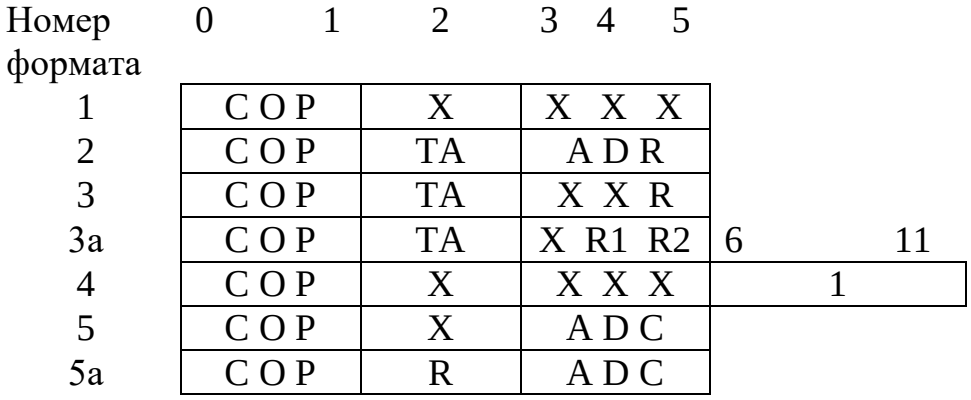


Рисунок 3 - Форматы команд учебной ЭВМ

1.3.2 Способы адресации

В ЭВМ принято различать пять основных способов адресации: *прямая, косвенная, непосредственная, относительная, безадресная.*

Каждый способ имеет разновидности. В модели учебной ЭВМ реализованы семь способов адресации, приведенные в табл. 1.

Таблица 1 -Адресация в командах учебной ЭВМ

Код ТА	Тип адресации	Исполнительный адрес
0	Прямая (регистровая)	ADR (R)
1	Непосредственная	-
2	Косвенная	ОЗУ (ADR) [3:5]
3	Относительная	ADR + RB
4	Косвенно-регистровая	РОН (R)[3:5]
5	Индексная с постинкрементом	РОН (R)[3:5], R:=R+1
6	Индексная с преддекрементом	R:=R-1, РОН(R)[3:5]

1.3.3 Система операций

Система команд учебной ЭВМ включает команды следующих классов:

- *арифметико-логические* и *специальные*: сложение, вычитание, умножение, деление;

- *пересылки* и *загрузки:* чтение, запись, пересылка (из регистра в регистр), помещение в стек, извлечение из стека, загрузка указателя стека, загрузка базового регистра;

- *ввода/вывода:* ввод, вывод;

- *передачи управления:* безусловный и шесть условных переходов, вызов подпрограммы, возврат из подпрограммы, цикл, программное прерывание, возврат из прерывания;

- *системные:* пустая операция, разрешить прерывание, запретить прерывание, стон.

Список команд учебной ЭВМ приведен в табл. 4 и в табл. 6.

1.4 Состояния и режимы работы ЭВМ

Ядром УУ ЭВМ является управляющий автомат (УА), вырабатывающий сигналы управления, которые инициируют работу АЛУ, РОН, ОЗУ и УВВ, передачу информации между регистрами устройств ЭВМ и действия над содержимым регистров УУ.

ЭВМ может находиться в одном из двух состояний: **Останов** и **Работа.**

В состояние **Работа** ЭВМ переходит по действию команд **Пуск** или **Шаг.** Команда **Пуск** запускает выполнение программы, представляющую собой последовательность команд, записанных в ОЗУ, в автоматическом режиме до команды HLT или точки останова. Программа выполняется по командам, начиная с ячейки ОЗУ, на которую указывает PC, причем изменение состояний объектов модели отображается в окнах обозревателей.

В состояние **Останов** ЭВМ переходит по действию команды **Стоп** или автоматически в зависимости от установленного режима работы.

Команда **Шаг,** в зависимости от установленного режима работы, запускает выполнение одной команды или одной микрокоманды (если установлен **Режим микрокоманд),** после чего переходит в состояние **Останов.**

В состоянии **Останов** допускается просмотр и модификация объектов модели: регистров процессора и РОН, ячеек ОЗУ, устройств ввода/вывода. В процессе модификации ячеек ОЗУ и РОН можно вводить данные для программы, в ячейки ОЗУ — программу в кодах. Кроме того, в режиме **Останов** можно менять параметры модели и режимы ее работы, вводить и/или редактировать программу в мнемокодах, ассемблировать мнемокоды, выполнять стандартные операции с файлами.

1.5 Интерфейс пользователя

В программной модели учебной ЭВМ использован стандартный интерфейс Windows, реализованный в нескольких окнах.

Основное окно модели **Модель учебной ЭВМ** содержит основное меню и кнопки на панели управления. В рабочее поле окна выводятся сообщения о функционировании системы в целом. Эти сообщения группируются в файле logfile.txt (по умолчанию), сохраняются на диске и могут быть проанализированы после завершения сеанса работы с моделью.

Меню содержит следующие пункты и команды:

Файл:

- неактивные команды;
- **Выход.**

Вид:

- **Показать все;**
- **Скрыть все;**
- **Процессор;**
- **Микрокомандный уровень;**
- **Память;**
- **Кэш-память;**
- **Программа;**
- **Текст программы.**

Внешние устройства:

- **Менеджер ВУ;**
- окна подключенных ВУ;

Работа:

- **Пуск;**
- **Стоп;**
- **Шаг;**
- **Режим микрокоманд;**
- **Кэш-память;**
- **Настройки.**

Команды меню **Вид** открывают окна соответствующих обозревателей, описанные далее. Менеджер внешних устройств позволяет подключать/отключать внешние устройства, предусмотренные в системе. Команда вызова менеджера внешних устройств выполняется при нажатии кнопки на панели инструментов. Подробнее о внешних устройствах и их обозревателях смотрите в разд. 1.6.

Команды меню **Работа** позволяют запустить программу в автоматическом (команда **Пуск)** или шаговом (команда **Шаг)** режиме, остановить выполнение программы в модели процессора (команда **Стоп).** Эти команды могут выполняться при нажатии соответствующих одноименных кнопок на панели инструментов основного окна.

Команда **Режим микрокоманд** включает/выключает микрокомандный режим работы процессора, а команда **Кэш-память** подключает/отключает в системе модель этого устройства.

Команда **Настройки** открывает диалоговое окно **Параметры системы,** позволяющее установить задержку реализации командного цикла (при выполнении программы в автоматическом режиме), а так же установить параметры файла logfile.txt, формируемого системой и записываемого на диск.

1.5.1 Окна основных обозревателей системы

Окно *Процессор*

Окно **Процессор** (рис. 4) обеспечивает доступ ко всем регистрам и флагам процессора.

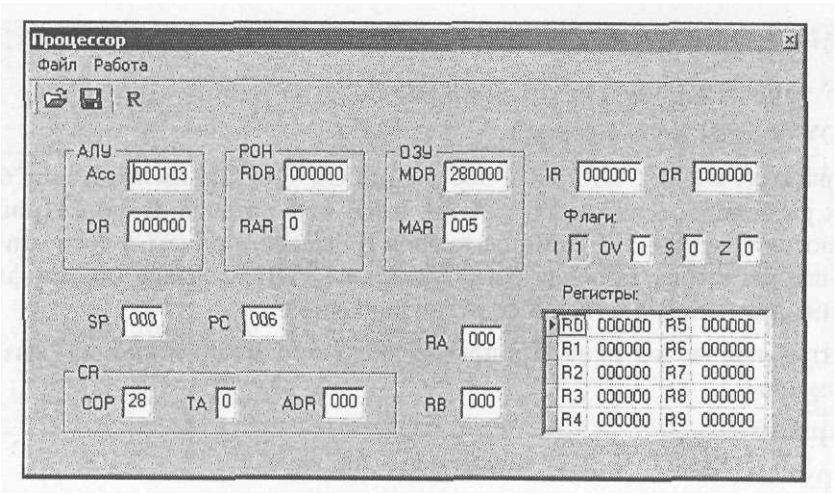


Рисунок 4 - Окно Процессор

Программно-доступные регистры и флаги:

- Acc — аккумулятор;
- PC — счетчик адреса команды, содержащий адрес текущей команды;
- SP — указатель стека, содержащий адрес верхушки стека;
- RB — регистр базового адреса, содержащий базовый адрес;
- RA — регистр адреса, содержащий исполнительный адрес при косвенной адресации;
- IR — входной регистр;
- OR — выходной регистр;
- I — флаг разрешения прерываний.

Системные регистры и флаги:

- DR — регистр данных АЛУ, содержащий второй операнд;
- MDR — регистр данных ОЗУ;
- MAR — регистр адреса ОЗУ;
- RDR — регистр данных блока РОН;

- RAR — регистр адреса блока РОН;
    - CR — регистр команд, содержащий поля:
      COP — код операции;
      TA — тип адресации;
      ADR — адрес или непосредственный операнд;
- Z — флаг нулевого значения Acc;
- S — флаг отрицательного значения Acc;
- OV — флаг переполнения.

Регистры Acc, DR, IR, OR, CR и все ячейки ОЗУ и РОН имеют длину 6 десятичных разрядов, регистры PC, SP, RA и RB — 3 разряда. В окне **Процессор** отражаются текущие значения регистров и флагов, причем в состоянии **Останов** все регистры, включая регистры блока РОН, и флаги (кроме флага I) доступны для непосредственного редактирования.

Элементы управления окна **Процессор** включают меню и кнопки, вызывающие команды:

- □ **Сохранить;**
- □ **Загрузить;**
- □ **Reset;**
- □ **Reset R0-R9** (только команда меню **Работа).**

Команды **Сохранить, Загрузить** позволяют сохранить текущее значение регистров и флагов процессора в файле и восстановить состояние процессора из файла. Команда Reset и кнопка R устанавливают все регистры (в т. ч. блок РОН) в начальное (нулевое) значение. Содержимое ячеек памяти при этом не меняется. Выполняемая лишь из меню **Работа** команда **Reset R0-R9** очищает только регистры блока РОН.

Окно *Память*

Окно **Память** (рис. 5) отражает текущее состояние ячеек ОЗУ. В этом окне допускается редактирование содержимого ячеек, кроме того, предусмотрена возможность выполнения (через меню или с помощью кнопок панели инструментов) пяти команд: **Сохранить, Загрузить, Перейти к, Вставить, Убрать.**

Команды **Сохранить, Загрузить** во всех окнах, где они предусмотрены, работают одинаково— сохраняют в файле текущее состояние объекта (в данном случае памяти) и восстанавливают это состояние из выбранного файла, причем файл в каждом окне записывается по умолчанию с характерным для этого окна расширением.

Команда **Перейти к** открывает диалоговое окно, позволяющее перейти на заданную ячейку ОЗУ.

Команда **Убрать** открывает диалог, в котором указывается диапазон ячеек с m по n. Содержимое ячеек в этом диапазоне теряется, а содержимое ячеек [(n+1): 999] перемещается в соседние ячейки с меньшими адресами. Освободившиеся ячейки с адресами 999, 998, ... заполняются нулями.

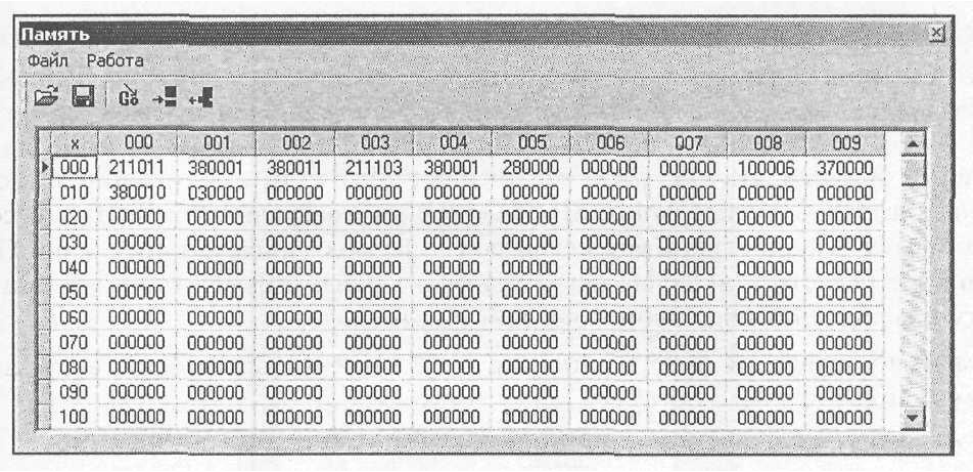


Рисунок 5 - Окно **Память**

Команда **Вставить,** позволяющая задать номера ячеек, перемещает содержимое всех ячеек, начиная от m-й на m-n позиций в направлении больших адресов, ячейки заданного диапазона [m:n] заполняются нулями, а содержимое последних ячеек памяти теряется.

Окно *Текст программы*

Окно **Текст программы** (рис. 6) содержит стандартное поле текстового редактора, в котором можно редактировать тексты, загружать в него текстовые файлы и сохранять подготовленный текст в виде файла.

Команды меню **Файл:**

Новая — открывает новый сеанс редактирования;

Загрузить — открывает стандартный диалог загрузки файла в окно редактора;

Сохранить — сохраняет файл под текущим именем;

Сохранить как — открывает стандартный диалог сохранения файла;

Вставить — позволяет вставить выбранный файл в позицию курсора.

Все перечисленные команды, кроме последней, дублированы кнопками на панели инструментов окна. На той же панели присутствует еще одна кнопка — **Компилировать,** которая запускает процедуру ассемблирования текста поле редактора.

Ту же процедуру можно запустить из меню Работа. Команда **Адрес вставки** позволяет задать адрес ячейки ОЗУ, начиная с которой программа будет размещаться в памяти. По умолчанию этот адрес принят равным 0.

Ниже области редактирования в строку состояния выводится позиция текущей строки редактора — номер строки, в которой находится курсор.

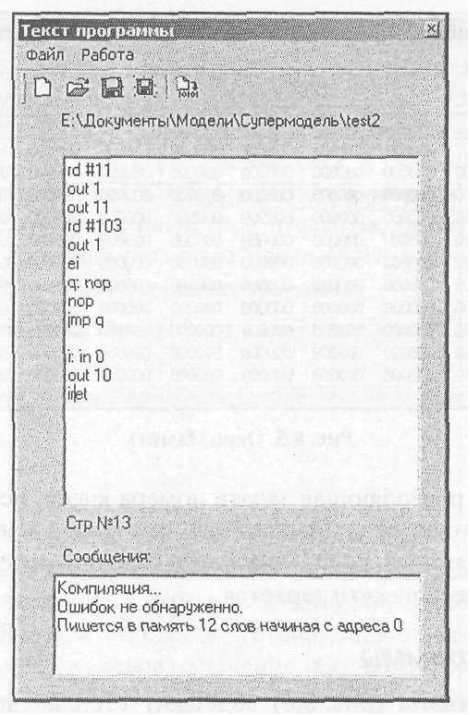


Рисунок 6 - Окно **Текст программы**

В случае обнаружения синтаксических ошибок в тексте программы диагностические сообщения процесса компиляции выводятся в окно сообщений и запись в память кодов (даже безошибочного начального фрагмента программы) не производится.

После исправления ошибок и повторной компиляции выдается сообщение об отсутствии ошибок, о расположении и размере области памяти, занятой под ассемблированную программу.

Набор текста программы производится по стандартным правилам языка ассемблера. В каждой строке может содержаться метка, одна команда и комментарий. Метка отделяется от команды двоеточием, символы после знака "точка с запятой" до конца строки игнорируются компилятором и могут рассматриваться как комментарии. Строка может начинаться с ; и, следовательно, содержать только комментарии.

Окно *Программа*

Окно **Программа** (рис. 7) отображает таблицу, имеющую 300 строк и 4 столбца. Каждая строка таблицы соответствует дизассемблированной ячейке ОЗУ. Второй столбец содержит адрес ячейки ОЗУ, третий — дизассемблированный мнемокод, четвертый - машинный код команды. В первом столбце может помещаться указатель → на текущую команду (текущее значение PC) и точка останова — красная заливка ячейки.

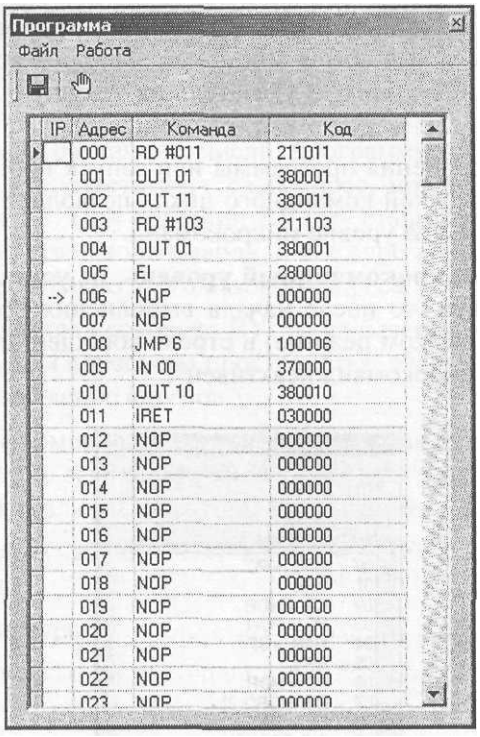


Рисунок 7 - Окно **Программа**

Окно **Программа** позволяет наблюдать процесс прохождения программы. В этом окне ничего нельзя редактировать. Органы управления окна позволяют сохранить содержимое окна в виде текстового файла, выбрать начальный адрес области ОЗУ, которая будет дизассемблироваться (размер области постоянный — 300 ячеек), а также установить/снять точку останова. Последнее можно проделать тремя способами: командой **Точка останова** из меню Работа, кнопкой на панели инструментов или двойным щелчком мыши в первой ячейке соответствующей строки. Характерно, что прочитать в это окно ничего нельзя. Сохраненный текстовый asm-файл можно загрузить в окно **Текст программы,** ассемблировать его и тогда дизассемблированное значение заданной области памяти автоматически появится в окне **Программа.** Такую процедуру удобно использовать, если программа изначально пишется или редактируется непосредственно в памяти в машинных кодах.

Начальный адрес области дизассемблирования задается в диалоге командой **Начальный адрес** меню **Работа.**

Окно *Микрокомандный уровень*

Окно **Микрокомандный уровень** (рис. 8) используется только в режиме микрокоманд, который устанавливается командой **Режим микрокоманд** меню **Работа. В** это окно выводится мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

Шаговый режим выполнения программы или запуск программы в автоматическом режиме с задержкой командного цикла позволяет наблюдать процесс выполнения программы на уровне микрокоманд.

Если открыть окно **Микрокомандный уровень,** не установив режим микрокоманд в меню **Работа,** то после начала выполнения программы в режиме **Шаг** (или в автоматическом режиме) в строке сообщений окна будет выдано сообщение "Режим микрокоманд неактивен".

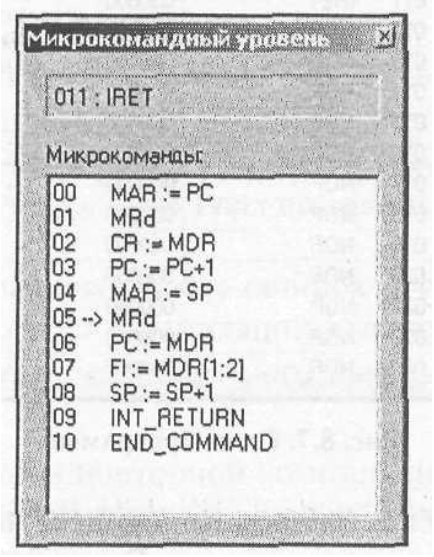


Рисунок 8 - Окно **Микрокомандный уровень**

Окно *Кэш-память*

Окно **Кэш-память** используется в режиме с подключенной кэш-памятью. Подробнее смотрите об этом режиме в разд. 1.8.

1.6. Внешние устройства

Модели внешних устройств (ВУ), используемые в описываемой системе, реализованы по единому принципу. С точки зрения процессора они представляют собой ряд программно-доступных регистров, лежащих в адресном пространстве ввода/вывода. Размер регистров ВУ совпадает с размером ячеек памяти и регистров данных процессора — шесть десятичных разрядов.

Доступ к регистрам ВУ осуществляется по командам IN aa, OUT aa, где aa — двухразрядный десятичный адрес регистра ВУ. Таким образом, общий объем адресного пространства ввода/вывода составляет 100 адресов. Следует помнить, что адресные пространства памяти и ввода/вывода в этой модели разделены.

Разные ВУ содержат различное число программно-доступных регистров, каждому из которых соответствует свой адрес, причем нумерация адресов всех ВУ начинается с 0. При создании ВУ ему ставится в соответствие *базовый адрес* в пространстве ввода/вывода, и все адреса его регистров становятся *смещениями* относительно этого базового адреса.

Если в системе создаются несколько ВУ, то их базовые адреса следует выбирать с учетом величины адресного пространства, занимаемого этими устройствами, исключая наложение адресов.

Если ВУ способно формировать запрос на прерывание, то при создании ему ставится в соответствие *вектор прерывания* — десятичное число. Разным ВУ должны назначаться различные векторы прерываний.

Программная модель учебной ЭВМ комплектуется набором внешних устройств, включающим:

- контроллер клавиатуры;
- дисплей;
- блок таймеров;
- тоногенератор,

которым по умолчанию присвоены параметры, перечисленные в табл. 2.

Таблица 2 - Параметры внешних устройств

Внешнее устройство	**Базовый адрес**	**Адрес регистров**	**Вектор прерывания**
Контроллер клавиатуры	0	0, 1, 2	0
Диспллей	10	0, 1, 2, 3	Нет
Блок таймеров	20	0, 1, 2, 3, 4, 5, 6	2
Тоногенератор	30	0, 1	Нет

При создании устройств пользователь может изменить назначенные по умолчанию базовый адрес и вектор прерывания.

В описываемой версии системы не предусмотрена возможность подключения в систему нескольких одинаковых устройств.

Большинство внешних устройств содержит регистры *управления* CR и *состояния* SR, причем обычно регистры CR доступны только по записи, а SR — по чтению.

Регистр CR содержит флаги и поля, определяющие режимы работы ВУ, а SR — флаги, отражающие текущее состояние ВУ. Флаги SR устанавливаются аппаратно, но сбрасываются программно (или по внешнему сигналу). Поля и флаги CR устанавливаются и сбрасываются программно при записи кода данных в регистр CR или специальными командами.

Контроллер ВУ интерпретирует код, записываемый по адресу CR как команду, если третий разряд этого кода равен 1, или как записываемые в CR данные, если третий разряд равен 0. В случае получения командного слова запись в регистр CR не производится, а пятый разряд слова рассматривается как код операции.

1.6.1 Контроллер клавиатуры

Контроллер клавиатуры (рис. 9) представляет собой модель внешнего устройства, принимающего ASCII-коды от клавиатуры ПЭВМ.

Символы помещаются последовательно в *буфер символов,* размер которого установлен равным 50 символам, и отображаются в окне обозревателя (рис.8. 10).

В состав контроллера клавиатуры входят три программно-доступных регистра:

DR (адрес 0) — регистр данных;

CR (адрес 1) — регистр управления, определяет режимы работы контроллера и содержит следующие флаги:

- E — флаг разрешения приема кодов в буфер;
- I — флаг разрешения прерывания;
- S — флаг режима посимвольного ввода.

SR (адрес 2) — регистр состояния, содержит два флага:

- Err — флаг ошибки;
- Rd — флаг готовности.

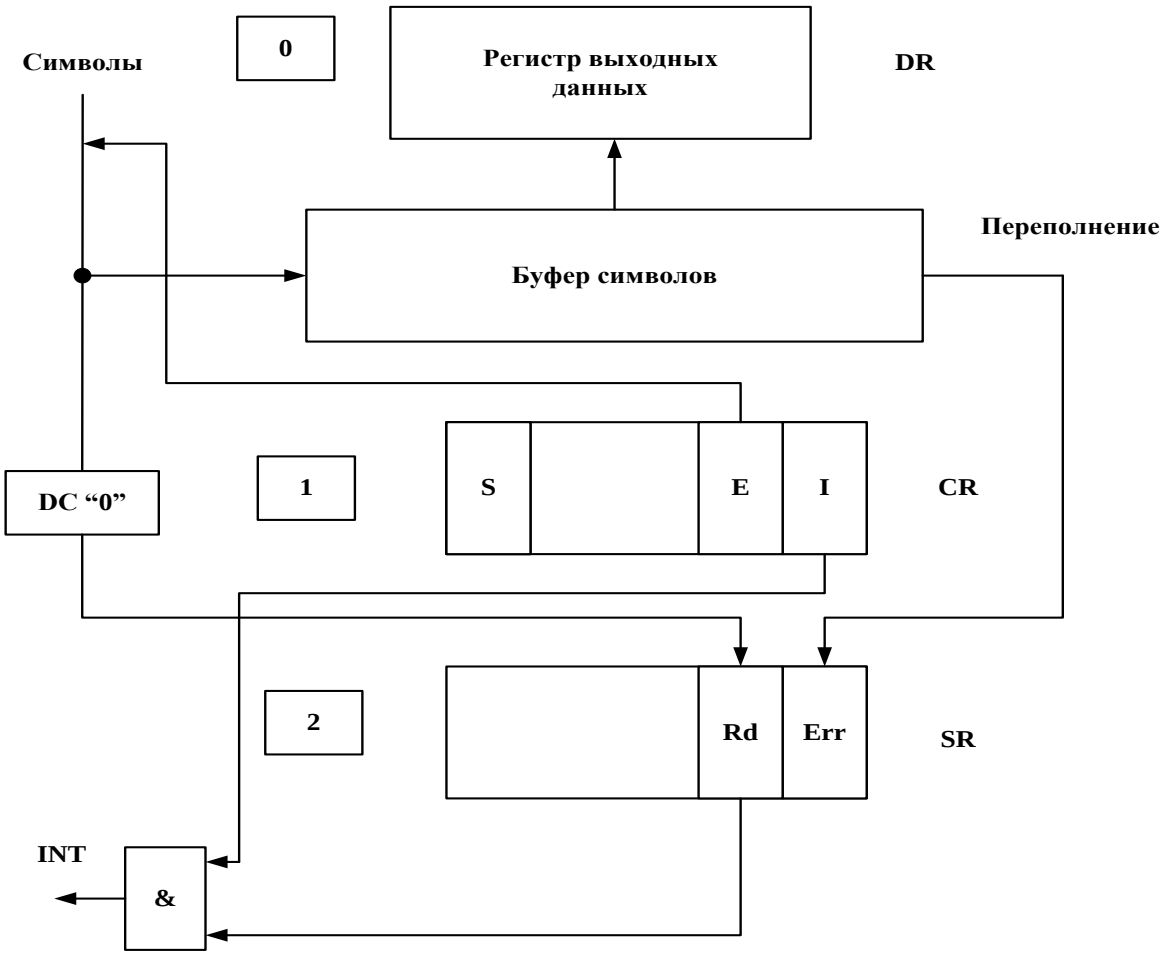


Рисунок 9 - Контроллер клавиатуры

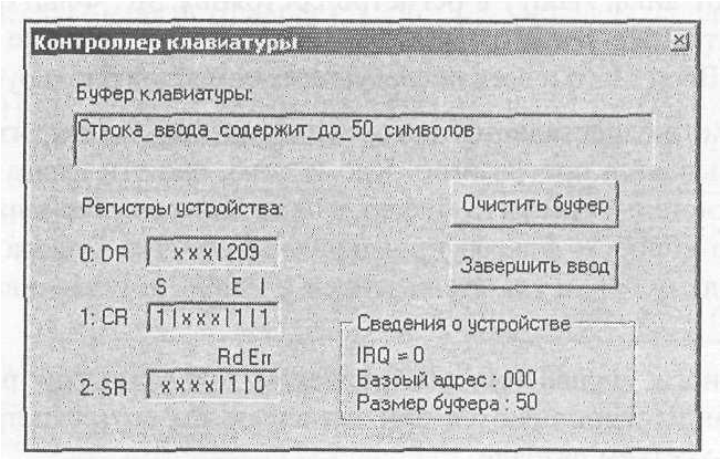


Рисунок 10 - Окно обозревателя контроллера клавиатуры

Регистр данных DR доступен только для чтения, через него считываются ASCII-коды из буфера, причем порядок чтения кодов из буфера соответствует порядку их записи в буфер — каждое чтение по адресу 0 автоматически перемещает указатель чтения буфера. В каждый момент времени DR содержит код символа по адресу указателя чтения буфера.

Флаги *регистра управления CR* устанавливаются и сбрасываются программно.

Флаг E, будучи установленным, разрешает прием кодов в буфер. При E = 0 контроллер игнорирует нажатие на клавиатуре, прием кодов в буфер не производится. На считывание кодов из буфера флаг E влияния не оказывает.

Флаг I, будучи установленным, разрешает при определенных условиях формирование контроллером запроса на прерывание. При I = 0 запрос на прерывание не формируется.

Флаг S = 1 устанавливает т. н. *режим посимвольного ввода,* иначе контроллер работает в обычном режиме. Флаг S устанавливается и сбрасывается программно, кроме того, S сбрасывается при нажатии кнопки **Очистить буфер** в окне **Контроллер клавиатуры.**

Условия формирования запроса на прерывание определяются, с одной стороны, значением флага разрешения прерывания I, с другой — режимом работы контроллера. В режиме посимвольного ввода запрос на прерывание формируется после ввода каждого символа (разумеется, при I= 1), в обычном режиме запрос будет сформирован по окончании набора строки.

Завершить набор строки можно, щелкнув по кнопке **Завершить ввод** в окне **Контроллер клавиатуры** (см. рис. 8.10). При этом устанавливается флаг готовности Rd (от англ. *ready)* в регистре состояния SR. Флаг ошибки Err (от англ. *error)* в том же регистре устанавливается при попытке ввода в буфер 51 -го символа. Ввод 51 -го и всех последующих символов блокируется.

Сброс флага Rd осуществляется автоматически при чтении из регистра DR, флаг Err сбрасывается программно. Кроме того, оба эти флага сбрасываются при нажатии кнопки **Очистить буфер** в окне **Контроллер клавиатуры;** одновременно

со сбросом флагов производится очистка буфера— весь буфер заполняется кодами 00h, и указатели записи и чтения устанавливаются на начало буфера.

Для программного управления контроллером предусмотрен ряд командных слов. Все команды выполняются при записи по адресу регистра управления CR кодов с 1 в третьем разряде.

Контроллер клавиатуры интерпретирует следующие командные слова:

xxx10l — очистить буфер (действие команды эквивалентно нажатию кнопки **Очистить буфер);**

xxx102 — сбросить флаг Err в регистре SR;

xxx103 — установить флаг S в регистре CR;

xxx104 — сбросить флаг S в регистре CR.

Если по адресу 1 произвести запись числа *xxx0nn,* то произойдет изменение 4-го и 5-го разрядов регистра CR по следующему правилу:

$$n = \begin{cases} 0 - \text{записать } 0; \\ 1 - \text{записать } 1; \\ 2,\ldots, 9 - \text{сохранить разряд без изменения.} \end{cases} \qquad (8.1)$$

8.6.2 Дисплей

Дисплей (рис. 11) представляет собой модель внешнего устройства, реализующую функции символьного дисплея. Дисплей может отображать символы, задаваемые ASCII-кодами, поступающими на его регистр данных. Дисплей включает:

видеопамять объемом 128 слов (ОЗУ дисплея);

символьный экран размером 8 строк по 16 символов в строке;

четыре программно-доступных регистра:

- DR (адрес 0) — регистр данных;
- CR (адрес 1) — регистр управления;
- SR (адрес 2) — регистр состояния;
- AR (адрес 3) — регистр адреса.

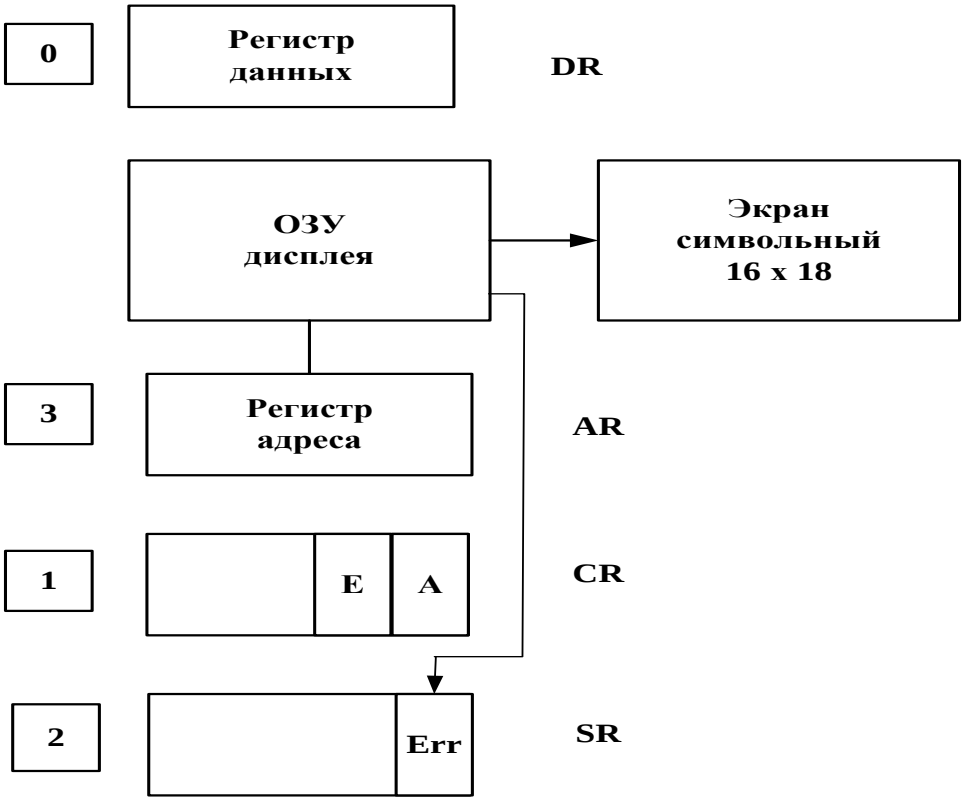


Рисунок 11 - Контроллер дисплея

Через *регистры адреса* AR и *данных* DR по записи и чтению осуществляется доступ к ячейкам видеопамяти. При обращении к регистру DR по записи содержимое аккумулятора записывается в DR и в ячейку видеопамяти, адрес которой установлен в регистре AR.

Регистр управления CR доступен только по записи и содержит в 4-м и 5-м разрядах соответственно два флага:

E — флаг разрешения работы дисплея; при E = 0 запись в регистры AR и DR блокируется;

A — флаг автоинкремента адреса; при A = 1 содержимое AR автоматически увеличивается на 1 после любого обращения к регистру DR — по записи или чтению.

Изменить значения этих флагов можно, если записать по адресу CR (по умолчанию — 11) код *xxx0nn,* при этом изменение 4-го и 5-го разрядов регистра CR произойдет согласно выражению (8.1).

Для программного управления дисплеем предусмотрены две команды, коды которых должны записываться по адресу регистра CR, причем в третьем разряде командных слов обязательно должна быть 1:

xxxl0l — очистить дисплей (действие команды эквивалентно нажатию кнопки **Очистить** в окне **Дисплей),** при этом очищается видеопамять (в каждую ячейку записывается код пробела— 032), устанавливается в 000 регистр адреса AR и сбрасываются флаги ошибки Err и автоинкремента A;

xxx102 — сбросить флаг ошибки Err.

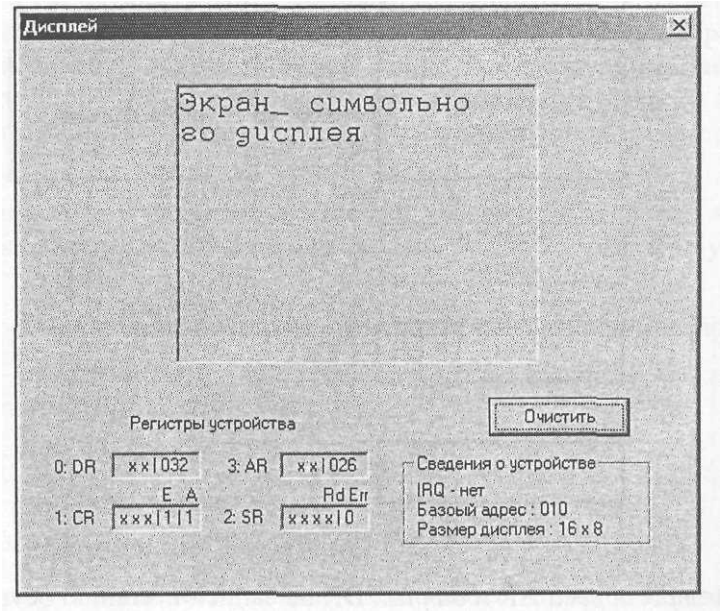


Рисунок 12 - Окно обозревателя контроллера дисплея

Регистр состояния SR доступен только по чтению и содержит единственный флаг (в пятом разряде) ошибки Err. Этот флаг устанавливается аппаратно при попытке записать в регистр адреса число, большее 127, причем как в режиме прямой записи в AR, так и в режиме автоинкремента после обращения по адресу 127. Сбрасывается флаг Err программно или при нажатии кнопки Очистить в окне **Дисплей** (рис. 12).

1.6.3 Блок таймеров

Блок таймеров (рис. 13) включает в себя три однотипных канала, каждый из которых содержит:

пятиразрядный десятичный реверсивный счетчик Т, на вход которого поступают метки времени (таймер);

программируемый предделитель D;

регистр управления таймером CTR;

флаг переполнения таймера FT.

Регистры таймеров Т доступны по записи и чтению (адреса 1, 3, 5 соответственно для Tl, Т2, ТЗ). Программа в любой момент может считать текущее содержимое таймера или записать в него новое значение.

На входы предделителей поступает общие для всех каналов метки времени CLK с периодом 1 мс. Предделители в каждом канале программируются независимо, поэтому таймеры могут работать с различной частотой.

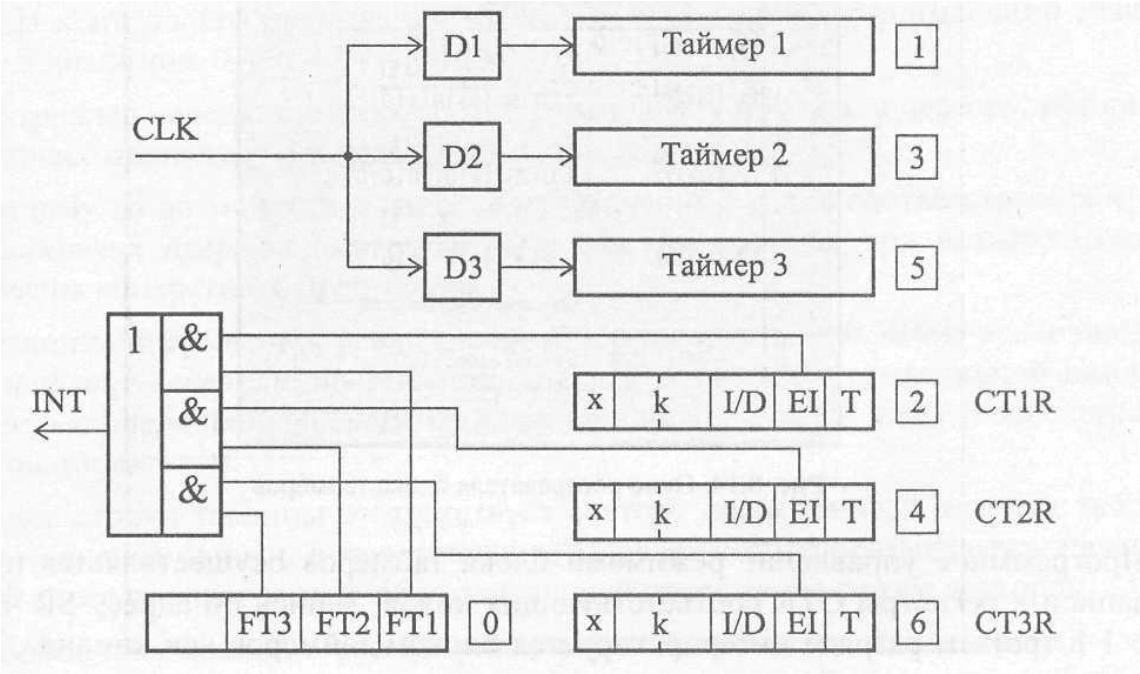


Рисунок 13 - Блок таймеров

Регистры управления CTR доступны по записи и чтению (адреса 2, 4, 6) и содержат следующие поля:

T (разряд 5) — флаг включения таймера;

EI (разряд 4) — флаг разрешения формирования запроса на прерывание при переполнении таймера;

I/D (разряд 3) — направление счета (инкремент/декремент), при I/D =0 таймер работает на сложение, при I/D =1 — на вычитание;

k (разряды [1:2]) — коэффициент деления предделителя (от 1 до 99).

Флаги переполнения таймеров собраны в один регистр — доступный только по чтению регистр состояния SR, имеющий адрес 0. Разряды регистра (5, 4 и 3 для Tl, T2, T3 соответственно) устанавливаются в 1 при переполнении соответствующего таймера. Для таймера, работающего на сложение, переполнение наступает при переходе его состояния из 99 999 в 0, для вычитающего таймера — переход из 0 в 99 999.

В окне обозревателя (рис. 8.14) предусмотрена кнопка **Сброс,** нажатие которой сбрасывает в 0 все регистры блока таймеров, кроме CTR, которые устанавливаются в состояние 001000. Таким образом, все три таймера обнуляются, переключаются в режим инкремента, прекращается счет, запрещаются прерывания, сбрасываются флаги переполнения и устанавливаются коэффициенты деления предделителей равными 01.

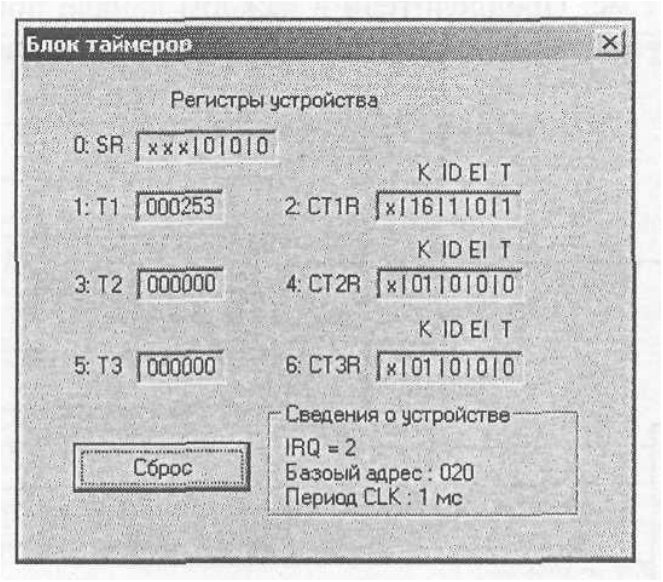


Рисунок 14 - Окно обозревателя блока таймеров

Программное управление режимами блока таймеров осуществляется путем записи в регистры CTR соответствующих кодов. Запись по адресу SR числа с 1 в третьем разряде интерпретируется блоком таймеров как команда, причем младшие разряды этого числа определяют код команды:

xxxl00 — общий сброс (эквивалентна нажатию кнопки Сброс в окне обозревателя);

xxxl0l — сброс флага переполнения таймера FT1;

xxx102 — сброс флага переполнения таймера FT2;

xxx103 — сброс флага переполнения таймера FT3.

1.6.4 Тоногенератор

Модель этого простого внешнего устройства не имеет собственного обозревателя, содержит всего два регистра, доступных только для записи:

FR (адрес 0) — регистр частоты звучания (Гц):

LR (адрес 1) — регистр длительности звучания (мс).

По умолчанию базовый адрес тоногенератора — 30. Сначала следует записать в FR требуемую частоту тона в герцах, затем в LR — длительность звучания в миллисекундах. Запись числа по адресу регистра LR одновременно является командой на начало звучания.

1.7 Подсистема прерываний

В модели учебной ЭВМ предусмотрен механизм векторных внешних прерываний. Внешние устройства формируют запросы на прерывания, которые поступают на входы *контроллера прерываний.* При подключении ВУ, способного формировать запрос на прерывание, ему ставится в соответствие номер входа

контроллера прерываний — вектор прерывания, принимающий значение в диапазоне 0—9.

Контроллер передает вектор, соответствующий запросу, процессору, который начинает процедуру обслуживания прерывания.

Каждому из возможных в системе прерываний должен соответствовать т. н. *обработчик прерывания*— подпрограмма, вызываемая при возникновении события конкретного прерывания.

Механизм прерываний, реализованный в модели учебной ЭВМ, поддерживает *таблицу векторов прерывании,* которая создается в оперативной памяти *моделью операционной системы* (если она используется) или непосредственно пользователем.

Номер строки таблицы соответствует вектору прерывания, а элемент таблицы — ячейка памяти, в трех младших разрядах которой размещается начальный адрес подпрограммы, обслуживающей прерывание с этим вектором.

Таблица прерываний в рассматриваемой модели жестко фиксирована— она занимает ячейки памяти с адресами 100—109. Таким образом, адрес обработчика с вектором 0 должен располагаться в ячейке 100, с вектором 2 — в ячейке 102. При работе с прерываниями не рекомендуется использовать ячейки 100—109 для других целей.

Процессор начинает обработку прерывания (если они разрешены), завершив текущую команду. При этом он:

- получает от контроллера вектор прерывания;
- формирует и помещает в верхушку стека слово, три младших разряда([3:5]) которого — текущее значение PC (адрес возврата из прерывания), а разряды [1:2] сохраняют десятичный эквивалент шестнадцатеричной цифры, определяющей значение вектора флагов (I, OV, S, Z). Например, если I=1, OV = 0, S = 1, Z = 1,то в разряды [1:2] запишется число $11_{10} = 1011_{2;}$
- сбрасывает в 0 флаг разрешения прерывания I;
- извлекает из таблицы векторов прерываний адрес обработчика, соответствующий обслуживаемому вектору, и помещает его в PC, осуществляя тем самым переход на подпрограмму обработчика прерывания.

Таким образом, вызов обработчика прерывания, в отличие от вызова подпрограммы, связан с помещением в стек не только адреса возврата, но и текущего значения вектора флагов. Поэтому последней командой подпрограммы обработчика должна быть команда IRET, которая не только возвращает в PC три младшие разряда ячейки — верхушки стека (как RET), НО И восстанавливает те значения флагов, которые были в момент перехода на обработку прерывания.

Не всякое событие, которое может вызвать прерывание, приводит к прерыванию текущей программы. В состав процессора входит программно-доступный флаг I разрешения прерывания. При I = 0 процессор не реагирует на запросы прерываний. После сброса процессора флаг I так же сброшен и все прерывания запрещены. Для того чтобы разрешить прерывания, следуете программе выполнить команду EI (от англ. *enable interrupt).*

Выше отмечалось, что при переходе на обработку прерывания флаг I автоматически сбрасывается, в этом случае прервать обслуживание одного прерывания

другим прерыванием нельзя. По команде IRET значение флагов восстанавливается, в т. ч. вновь устанавливается I = 1, следовательно, в основной программе прерывания опять разрешены.

Если требуется разрешить другие прерывания в обработчике прерывания, достаточно в нем выполнить команду EI. Контроллер прерываний и процессор на аппаратном уровне блокируют попытки запустить прерывание, если его обработчик начал, но не завершил работу.

Таким образом, флаг I разрешает или запрещает все прерывания системы. Если требуется выборочно разрешить некоторое подмножество прерываний, используются программно-доступные флаги разрешения прерываний непосредственно на внешних устройствах.

Как правило, каждое внешнее устройство, которое может вызвать прерывание, содержит в составе своих регистров разряд флага разрешения прерывания (см. формат регистров CR и CTR на рис. 9, 13), по умолчанию установленный в 0. Если оставить этот флаг в нуле, то внешнему устройству запрещается формировать запрос контроллеру прерываний.

Иногда бывает удобно (например, в режиме отладки) иметь возможность вызвать обработчик прерывания непосредственно из программы. Если использовать для этих целей команду CALL, которая помещает в стек только адрес возврата, то команда IRET, размещенная последней в обработчике, может исказить значения флагов (все они будут сброшены в 0, т. к. команда CALL формирует только три младшие разряда ячейки верхушки стека, оставляя остальные разряды в 000).

Поэтому в системах команд многих ЭВМ, в т. ч. и нашей модели, имеются команды вызова прерываний— INT n (в нашей модели $n \in \{0, 1,\ldots,9\}$), где n— вектор прерывания. Процессор, выполняя команду INT n, производит те же действия, что и при обработке прерывания с вектором n.

Характерно, что с помощью команды INT n можно вызвать обработчик прерывания даже в том случае, когда флаг разрешения прерывания I сброшен.

1.8 Программная модель кэш-памяти

К описанной в разд. 1.1 программной модели учебной ЭВМ может быть подключена программная модель кэш-памяти. Конкретная реализация кэш-памяти в описываемой программной модели показана на рис. 15.

Кэш-память содержит N ячеек (в модели N может выбираться из множества {4, 8, 16, 32}), каждая из которых включает трехразрядное поле тега (адреса ОЗУ), шестиразрядное поле данных и три однобитовых признака (флага):

Z — признак занятости ячейки;

U — признак использования;

W — признак записи в ячейку.

Таким образом, каждая ячейка кэш-памяти может дублировать одну любую ячейку ОЗУ, причем отмечается ее занятость (в начале работы модели все ячейки кэш-памяти свободны, $\forall Z_i = 0$), факт записи информации в ячейку во время пребывания ее в кэш-памяти, а также использование ячейки (т. е. любое обращение к ней).

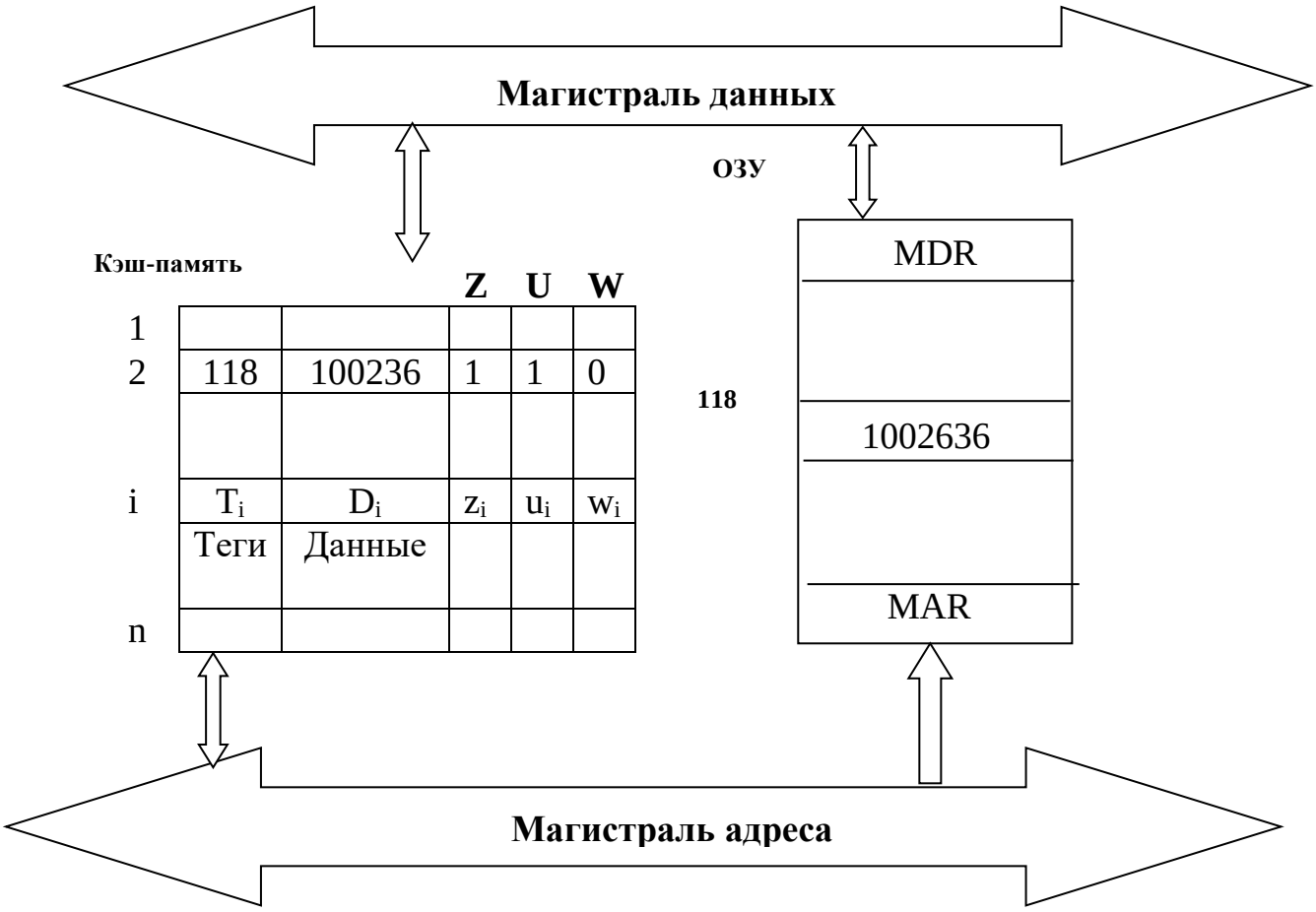


Рисунок 15 - Структура модели кэш-памяти

Текущее состояние кэш-памяти отображается на экране в отдельном окне в форме таблицы, причем количество строк соответствует выбранному числу ячеек

кэш. Столбцы таблицы определяют содержимое полей ячеек, например, так, как показано в табл. 3.

Таблица 3 - Пример текущего состояния кэш-памяти

	Теги	**Данные**	**Z**	**U**	**W**
1	012	220152	1	0	0
2	013	211003	1	1	0
3	050	000025	1	1	1
4	000	000000	0	0	0

Для настройки параметров кэш-памяти можно воспользоваться диалоговым окном **Кэш-память,** вызываемым командой **Вид Кэш-память,** а затем нажать первую кнопку на панели инструментов открытого окна. После этих действий появится диалоговое окно **Параметры кэш-памяти,** позволяющее выбрать *размер* кэш-памяти, *способ записи* в нее информации и *алгоритм замещения* ячеек.

Напомним, что при сквозной записи при кэш-попадании в процессорных циклах записи осуществляется запись как в ячейку кэш-памяти, так и в ячейку ОЗУ, а при обратной записи — только в ячейку кэш-памяти, причем эта ячейка отмечается битом записи (W_i :=1). При очистке ячеек, отмеченных битом записи, необходимо переписать измененное значение ноля данных в соответствующую ячейку ОЗУ.

При кэш-промахе следует поместить в кэш-память адресуемую процессором ячейку. При наличии свободных ячеек кэш-памяти требуемое слово помещается в одну из них (в порядке очереди). При отсутствии свободных ячеек следует отыскать ячейку кэш-памяти, содержимое которой можно удалить, записав на его место требуемые данные (команду). Поиск такой ячейки осуществляется с использованием *алгоритма замещения строк.*

В модели реализованы три различных алгоритма замещения строк:

случайное замещение, при реализации которого номер ячейки кэш-памяти выбирается случайным образом;

очередь, при которой выбор замещаемой ячейки определяется временем пребывания ее в кэш-памяти;

бит использования, случайный выбор осуществляется только из тех ячеек, которые имеют нулевое значение флага использования.

Напомним, что бит использования устанавливается в 1 при любом обращении к ячейке, однако, как только все биты U_i установятся в 1, все они тут же сбрасываются в 0, так что в кэш всегда ячейки разбиты на два непересекающихся подмножества по значению бита U — те, обращение к которым состоялось относительно недавно (после последнего сброса вектора **U**) имеют значение U = 1, иные — со значением U = 0 являются "кандидатами на удаление" при использовании алгоритма замещения *"бит использования".*

Если в параметрах кэш-памяти установлен флаг *"с учетом бита записи",* то все три алгоритма замещения осуществляют поиск "кандидата на удаление" прежде всего среди тех ячеек, признак записи которых не установлен, а при от-

сутствии таких ячеек (что крайне маловероятно) — среди всех ячеек кэш-памяти. При снятом флаге *"с учетом бита записи"* поиск осуществляется по всем ячейкам кэш-памяти без учета значения W.

Оценка эффективности работы системы с кэш-памятью определяется числом кэш-попаданий по отношению к общему числу обращений к памяти. Учитывая разницу в алгоритмах записи в режимах сквозной и обратной записи, эффективность использования кэш-памяти вычисляется по следующим выражениям (соответственно для сквозной и обратной записи):

$$K = \frac{S_к - S_{к_w}}{S_0}, \qquad (2)$$

$$K = \frac{S_к - S^i_{к_w}}{S_0} \qquad (3)$$

где:

K — коэффициент эффективности работы кэш-памяти;

S_o — общее число обращений к памяти;

S_K — число кэш-попаданий;

S_{Kw} — число сквозных записей при кэш-попадании (в режиме сквозной записи);

S^i_{Kw} — число обратных записей (в режиме обратной записи).