Проект «Инженерные кадры Зауралья»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра автоматизации производственных процессов

ИЗУЧЕНИЕ МНОГОЗАДАЧНОГО МОНИТОРА РЕАЛЬНОГО ВРЕМЕНИ MON-99

Методические указания
к лабораторной работе по дисциплине
«Программное обеспечение систем управления»
для студентов очной и заочной форм обучения направления 220700.62
«Автоматизация технологических процессов и производств»

Курган 2015

Кафедра: «Автоматизация производственных процессов»

Дисциплина: «Программное обеспечение систем управления»
(направление 220700.62).

Составили: доцент В.В. Тактаев, канд. техн. наук, доцент О.В. Дмитриева.

Утверждены на заседании кафедры 25 сентября 2014 г.

Рекомендованы методическим советом университета в рамках проекта «Инженерные кадры Зауралья» 20 декабря 2013 г.

Содержание

Введение

В большинстве случаев построения автоматизированных систем управления на базе программируемых контроллеров необходима специальная организация Программного обеспечения.

Для удовлетворения требований, предъявляемых к современной операционной системе (ОС), большое значение имеет ее структурное построение. Операционные системы прошли длительный путь развития от монолитных систем к хорошо структурированным модульным системам, способным к развитию, расширению и легкому переносу на новые платформы.

Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами.

Процесс (или по-другому, задача) - абстракция, описывающая выполняющуюся программу. Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов. Подсистема управления процессами планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Планирование процессов включает в себя решение следующих задач:

- определение момента времени для смены выполняемого процесса;
- выбор процесса на выполнение из очереди готовых процессов;
- переключение контекстов «старого» и «нового» процессов.

Первые две задачи решаются программными средствами, а последняя в значительной степени аппаратно.

Известны многочисленные варианты реализации диспетчеризации, описанные в рекомендованной литературе, от ОС QNX, TS до многозадачных мониторов реального времени MOST, TaskMon. На примере MON-99 изучаются основные принципы в лекционном материале и в выполняемых по программе курса лабораторных работах.

В многозадачной (многопроцессной) системе процесс может находиться в одном из трех **основных** состояний:

ВЫПОЛНЕНИЕ - активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;

ОЖИДАНИЕ - пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса (выделяют четыре типа ожидания Blocked-блокированный, Timing-ожидающий запуска по времени, Suspend-приостановленный, Delaying-задержанный).

ГОТОВНОСТЬ - также пассивное состояние процесса, но в этом случае процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.

В ходе жизненного цикла каждый процесс переходит из одного состояния в другое в соответствии с алгоритмом планирования процессов, реализуемым в данной операционной системе. Типичный граф состояний процесса показан на рисунке 1.

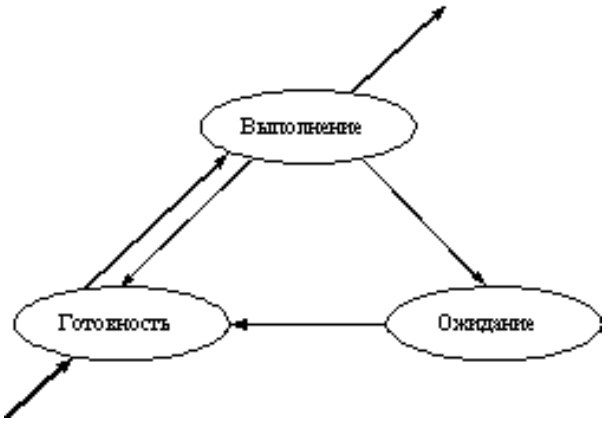


Рисунок 1 – Граф состояний процесса в многозадачной среде

В состоянии ВЫПОЛНЕНИЕ в однопроцессорной системе может находиться только один процесс, а в каждом из состояний ОЖИДАНИЕ и ГОТОВНОСТЬ - несколько процессов, эти процессы образуют очереди соответственно ожидающих и готовых процессов. Жизненный цикл процесса начинается с состояния ГОТОВНОСТЬ, когда процесс готов к выполнению и ждет своей очереди. При активизации процесс переходит в состояние ВЫПОЛНЕНИЕ и находится в нем до тех пор, пока либо он сам освободит процессор, перейдя в состояние ОЖИДАНИЯ какого-нибудь события, либо будет насильно "вытеснен" из процессора, например, вследствие исчерпания отведенного данному процессу кванта процессорного времени. В последнем случае процесс возвращается в состояние ГОТОВНОСТЬ. В это же состояние процесс переходит из состояния ОЖИДАНИЕ, после того, как ожидаемое событие произойдет. На протяжении существования процесса его выполнение может быть многократно прервано и продолжено. Для того, чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды отображается состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т.д. Эта информация называется *контекстом процесса*.

Кроме этого, операционной системе для реализации планирования процессов требуется дополнительная информация: идентификатор процесса, состояние процесса, данные о степени привилегированности процесса, место нахождения кодового сегмента и другая информация, информацию такого рода,

используемую ОС для планирования процессов, называют *дескриптором процесса*.

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки. Таким образом, каждый дескриптор, кроме всего прочего, содержит по крайней мере один указатель на другой дескриптор, соседствующий с ним в очереди.

Программный код только тогда начнет выполняться, когда для него операционной системой будет создан процесс. Создать процесс - это значит:

создать информационные структуры, описывающие данный процесс, то есть его дескриптор и контекст;

включить дескриптор нового процесса в очередь готовых процессов;

В соответствии с алгоритмами, основанными на квантовании, смена активного процесса происходит, если:

процесс завершился и покинул систему,

произошла ошибка,

процесс перешел в состояние ОЖИДАНИЕ,

исчерпан квант процессорного времени, отведенный данному процессу.

Процесс, который исчерпал свой квант, переводится в состояние ГОТОВНОСТЬ и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый процесс из очереди готовых. Таким образом, ни один процесс не занимает процессор надолго, поэтому квантование широко используется в системах разделения времени. Граф состояний процесса, изображенный на рисунке 1, соответствует алгоритму планирования, основанному на квантовании.

По разному может быть организована очередь готовых процессов: циклически, по правилу "первый пришел - первый обслужился" (FIFO) или по правилу "последний пришел - первый обслужился" (LIFO).

Другая группа алгоритмов использует понятие "приоритет" процесса. *Приоритет* - это число, характеризующее степень привилегированности процесса при использовании ресурсов вычислительной машины, в частности, процессорного времени: чем выше приоритет, тем выше привилегии.

Приоритет может выражаться целыми или дробными, положительным или отрицательным значением. Чем выше привилегии процесса, тем меньше времени он будет проводить в очередях. Приоритет может назначаться директивно администратором системы в зависимости от важности работы или внесенной платы, либо вычисляться самой ОС по определенным правилам, он может оставаться фиксированным на протяжении всей жизни процесса либо изменяться во времени в соответствии с некоторым законом. В последнем случае приоритеты называются динамическими.

Последовательность выполнения цикла двух работ.

- освоение информационной структуры, анализ по выбору одного из системных модулей монитора с заключением о достаточности функций, оптимальности текста или предложением по доработке;

- решение варианта прикладной системы: две и более задачи

Цель работы: изучение многозадачного монитора реального времени MON- на лабораторном комплексе с программируемым контроллером Кедр-2 и подготовка к выполнению аналогичной работы на стенде с контроллером Z-181.

Конфигурация технических средств для проведения лабораторной работы: для функционирования системы под управлением операционной системы MON-99 необходим контроллер КЕДР-2 и ПК.

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ MON-99

Программа MON-99 предназначена для организации параллельного выполнения в режиме реального времени группы взаимосвязанных прикладных задач. Задачи (процессы, нити) выполняют прикладные функции, кроме тела программы, состоят из дескриптора, состоящего из байта состояния, указателя стеков и таймаутов. Описатели необходимы и для других информационных структур (семафоры, буферы обмена межзадачными сообщениями и т.п.). Задачи реального времени взаимодействуют с монитором (диспетчером задач) посредством как минимум трех процедур:

- ожидание события (WAIT) - гибкая синхронизация;
- задержка на определенное время (DELAY) - жесткая синхронизация;
- ожидание события с контролем времени наступления события (WTIME).

Также задача может сообщить о завершении какого-либо события другой задаче, ожидающей этого события, применив процедуру POST.

Как правило, задачи имеют типовую структуру (рисунок 2).

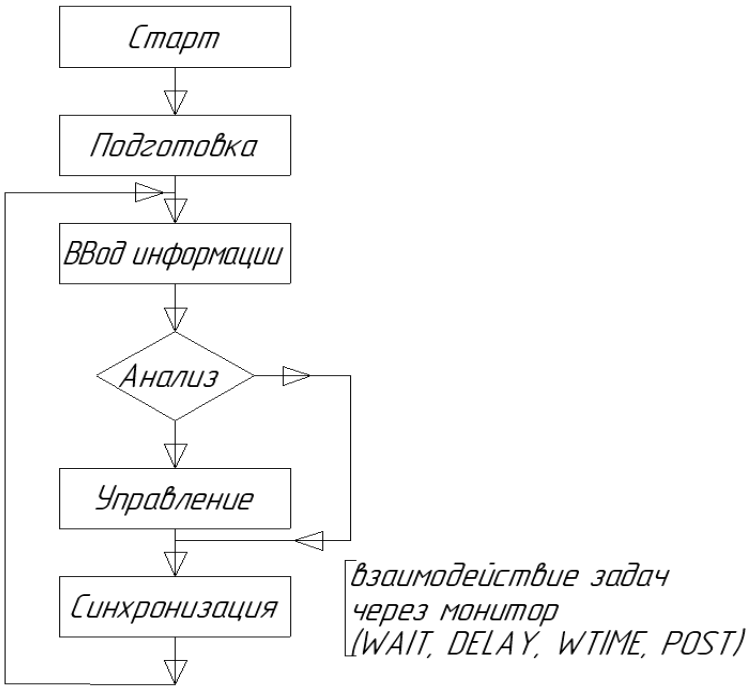


Рисунок 2 – Структура задачи

2. ФУНКЦИИ МОНИТОРА MON-99

Системные функции для организации взаимодействия задач выполняются операционной системой, в минимальном варианте - монитором MON-99

- формирование и инициализация дескрипторов информационных структур, необходимых для обеспечения мультизадачного режима;
- переключение процессора ЭВМ на выполнение той или иной задачи в соответствии с их приоритетами, временным регламентом, поступлением инициативных внешних сигналов и внутренней логикой взаимодействия задач;
- прием и обработка сигнала от таймера, на основании которого ведется служба времени, формирование «тиков» времени (в нашем случае 20 мсек, но возможно изменение от 2 до 65 мсек).

3. СТРУКТУРА И ПРОЦЕДУРЫ MON-99

Структура монитора реального времени MON-99 изображена на рисунке 3.

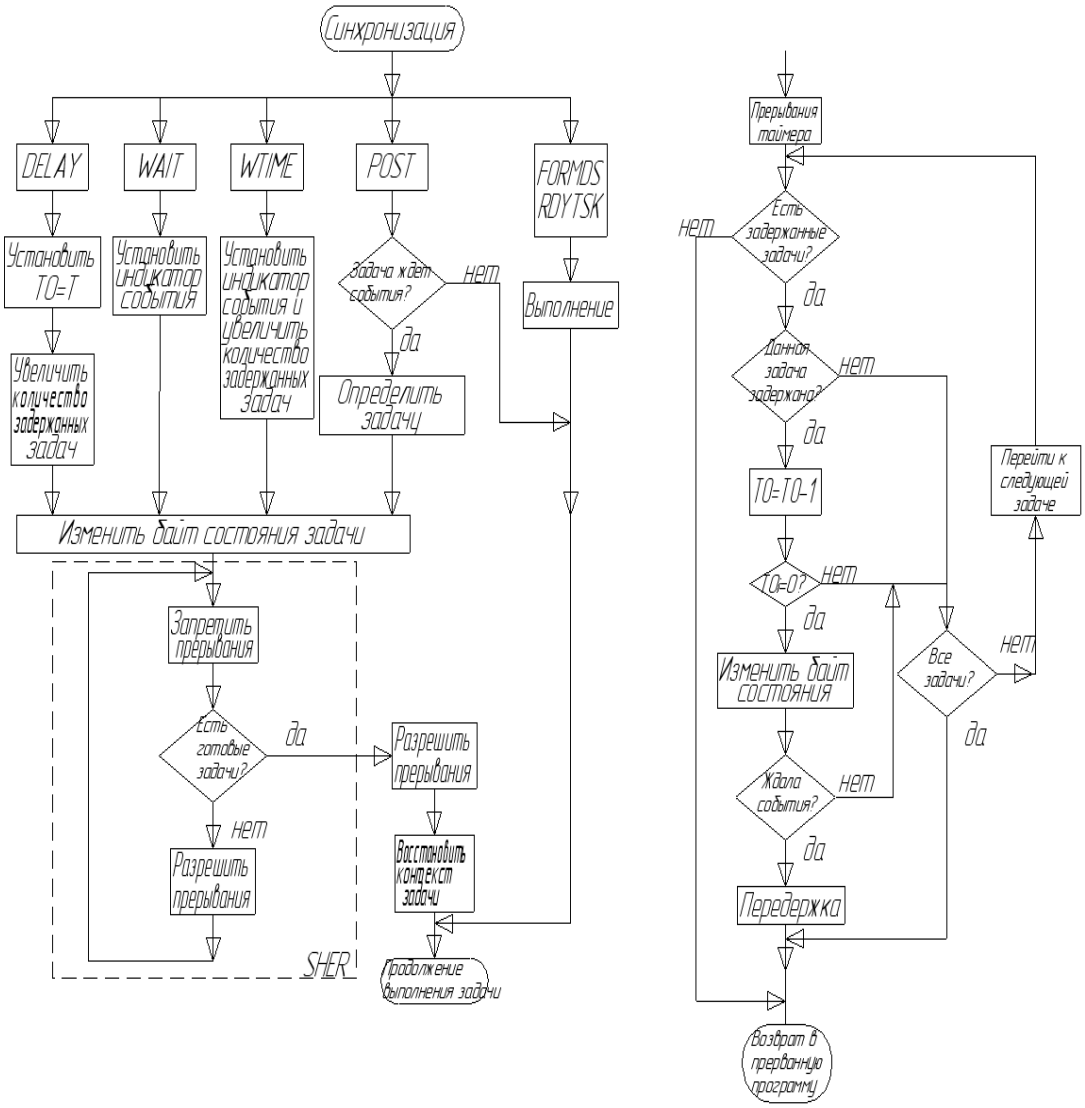


Рисунок 3 – Структура монитора MON-99

3.1. Процедура DELAY

Процедура предназначена для задержки выполнения задачи на определенное время (в тиках).

В паре H,L указывается число квантов времени, после истечения которых продолжается основная программа.

LXI H, <count>
CALL DELAY

3.2. Процедура WAIT

Применение процедуры вызывает блокировку задачи до тех пор, пока не придет сигнал. В паре B, C указывается адрес ключевого байта (т.е. адрес ожидаемого события).

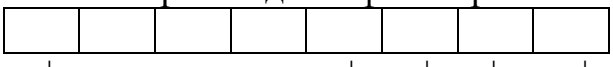
LXI B, <address>
CALL WAIT

3.3. Процедура WTIME

Применение процедуры вызывает блокировку задачи пока не придет сигнал или не истечет предельное время ожидания.

В паре H,L указывается число квантов времени. В паре B, C указывается идентификатор события (т.е. адрес байта ожидаемого события).

LXI H, <count>
LXI B, <address>
CALL WTIME
Формат идентификатора



признак ожидания номер задачи события

3.4. Процедура POST

Процедура выполняется задачей с максимальным быстродействием, следящей за изменением внешних дискретных и аналоговых сигналов. Типовое имя такой задачи - "ввод". Минимальная задержка задачи - 1 тик. При изменении сигнала вызывается процедура POST с идентификатором данного сигнала.

CALL POST

3.5. Процедура FORMDS

Процедура формирует дескриптор задач.

Перед вызовом процедуры в паре H,L указывается адрес таблицы, по которой формируется дескриптор и стек задачи.

```
LXI H, tableN
CALL FORMDS;  формировать дескриптор и стек задачи
```

Формат таблицы:

Номер задачи (байт)
Указатель стека
Стартовый адрес
Конец таблицы

Структура дескриптора задачи:

Status_ List	3 F80h	состояние задач (1 байт)
Delay_ List	3 F90h	таймауты (2 байта - младший, старший)
Stack_ List	3 FB0h	указатели стеков (2 байта - младший, старший)

Адреса состояния задачи:

Ready_ Count	3 FD0h	кол-во готовых задач;
Delay_ Count	3 FD1h	количество задержанных по времени задач;
CPU_t ask	3 FD2h	номер выполняемой задачи;

3.6. Процедура RDYTSK

Процедура переводит задачу в готовые.
В регистре A находится номер задачи.

```
MVI A, N
CALL RDYTSK;
```

Варианты заданий освоения процедур (параметр по номеру):

4.1	4.2	4.3	4.4	4.5	4.6	Прим еч.
T 25 s	ID 39 00h	T 35 s ID3A00h	ID 3B 00h	Task7 38 60h	Task 6	

4. КРАТКОЕ ОПИСАНИЕ ПРОГРАММЫ NEFT

Кедр-2 с MON-99 может взаимодействовать с различными системами верхнего уровня на базе программируемых контроллеров. Наиболее удобно использовать для этого систему верхнего уровня групповой замерной установки.

Программа позволяет устанавливать и контролировать параметры ГЗУ, фиксировать изменение параметров системы во времени, вести наладку контроллера. Доступ к функциям программы производится через главное меню.

Пункт "Наладка":

- "Регистрация" – для работы с программой нужно ввести пароль, т.е. зарегистрироваться как оператор.
- "Список операторов" – список тех пользователей, кому разрешена наладка контроллера.
- "Настройка параметров" – редактирование параметров установки.
- "Наладка контроллера" – позволяет читать и писать в ОЗУ, загружать файлы с программами в память и выполнять их.

Пункт "База данных":

- "Основная" – изменение параметров системы во времени.
- "Специальная" – то же, что и "Основная", но в расширенном виде.
- "График" – графическое представление основной базы данных.

Пункт "Протокол" – список, в котором отражены события, возникающие при работе программы.

Пункт "Выход" – завершение работы с программой.

Порядок работы с программой Neft.

1) Запускаем Neft.exe.
2) Выбираем требуемый COM – порт и его скорость (после этого должна появиться надпись "Связь").
3) Регистрируемся как оператор (Наладка/Регистрация).
4) Открываем окно наладки (Наладка/Наладка контроллера).

Слева расположено поле кодов, в нем отображаются данные, записываемые в ОЗУ или читаемые из него или из файла.

В поле "Начальный адрес" указывается адрес (в шестнадцатеричном виде, например 3850h).

В поле "Длина" в шестнадцатеричном виде указывается количество байт, записываемых/читаемых из ОЗУ или диска. При выполнении последующих действий байты в поле "Длина" должны изменить своё значение, в соответствии со значением реальной длины вашей программы (например, при записи 0090h, в последствии значение изменяется на 0079h).

Далее последовательно выполняются следующие действия:

"Читать ОЗУ"

"Писать ОЗУ"

"Читать файл"

"Писать в файл"
"Выполнить".
После завершения работы нажать кнопку "Выход".

5. ПОДГОТОВКА И ВЫПОЛНЕНИЕ ПРОГРАММ

5.1. Компилирование файлов

Программы для контроллера КЕДР-2 пишутся на ассемблере. Система команд совместима с КР580ИК80 (приложение А). Программы можно писать в любом текстовом редакторе.

Компиляция полученного файла (например, 11.asm) осуществляется при помощи ассемблера для CP/M – 80 и может проводится на любом компьютере имеющим файлы: 80cpm.exe (эмулятор CP/M), asm.com (сам ассемблер) и z80.exe (предварительный отладчик).

Формат командной строки:

80cpm.exe asm.com ll (без расширения “asm”)

В случае нормального завершения процесса компиляции получится два файла:

ll.prn – листинг программы,

ll.hex – откомпилированная программа.

Далее программу необходимо преобразовать в формат bin, при помощи эмулятора Z80.

1) Запускаем программу Z80.exe.
2) После запуска нажимаем ALT+D. В появившемся меню выбираем наш hex-файл (ll.hex), нажимаем ENTER.
3) В этом же меню нажимаем "W" и меняем расширение на bin.
4) В меню "Диапазон" указываем начальный адрес программы, нажимаем ENTER.
5) Выходим из программы (ESCAPE/Конец работы).

Файл готов для загрузки в КЕДР-2. (Загрузка производится при помощи программы Neft.exe).

5.2. Загрузка программы

1) В поле "Начальный адрес" указываем адрес начала программы в памяти (обычно совпадает с началом ОЗУ – 3850H).
2) В поле "Длина" ставим число, большее размера программы в байтах.
3) Нажимаем "Читать файл".
4) Выбираем нужный файл с программой.
5) Нажимаем "Писать ОЗУ". При необходимости выполняется контрольное чтение.
6) Нажимаем "Выполнить".

6. КОНТРОЛЛЕР КЕДР-2

Ниже приведены порты ввода-вывода контроллера Кедр-2 (таблица 1), а также адреса обработчиков прерывания и массива обмена с верхним уровнем.

Таблица 1 – Порты ввода-вывода

Speed_Sensor	69h	ввод с датчика скорости (бит 0);
Remote_Control	6Ah	ввод с пульта управления (8 бит);
Res	70h	сброс ввода (ввод/вывод);
Round_speed	78h	вывод кода скорости вращения ДПТ бит 0 - направление вращения (1 - по часовой стрелке, 0 - против);

Адреса обработчиков прерываний:
3F70h - служебный адрес (не используется);
3F72h - стартовый адрес обработки прерываний таймера (0150h);
3F74h - стартовый адрес обработки прерываний модема;
3F76h - стартовый адрес обработки прерываний АЦП;
3F78h - стартовый адрес обработки прерываний;
3F7Ah - стартовый адрес обработки прерываний ИРПС 3/4;
3F7Ch - стартовый адрес обработки прерываний ИРПС-2 (062Bh);
3F7Eh - стартовый адрес обработки прерываний ИРПС-1 (04ACh);

Адреса массива обмена с верхним уровнем:
3800h...3840h
383Dh - байт индикации:
биты 0,1,2 - номер позиции ПСМ
биты 3,4,5 - значение индикатора уровня жидкости (верхний, средний и нижний уровни)
бит 6 - сообщение "Охрана"
3840h, 3841h - показывает изменение давления dP;
3842h, 3843h - показывает уровень давления P;
3844h, 3845h - показывает температуру T внутри сепаратора;
3846h, 3847h - пульт ручного управления;
3848h, 3849h - позиции ПСМ;
384Ah, 384Bh - показывает что именно поступает в сепаратор - газ или жидкость (1 - жидкость, 2 - газ).

7. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Работа выполняется группой из 2 человек. Контроль подготовки к выполнению осуществляется преподавателем, возможно с уточнением задания для отдельного студента.

8. ОФОРМЛЕНИЕ ОТЧЕТА

Целью оформления отчёта является систематизация полученных данных при выполнении лабораторных работ знаний и практических навыков. Отчёт должен включать в себя: титульный лист (Приложение Б), варианты заданий, программы их реализации, комментарии, блок-схемы и выводы о проделанной лабораторной работе.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. – СПб: Питер, 2003, - 864с.
2. Р. Кертен. Введение в QNX Neutrino 2. М. Изд. Метрополис, 2001.
3. Монитор реального времени MON-99:учебное пособие для лабораторных работ, КГУ, 2012
4. Я. Белецкий. TopSpeed – расширенная версия языка Модула-2 для персональных ЭВМ. М. Мир. 1992
5. Фридман А.Л. Язык программирования С++. Курс лекций. М.: Интернет-университет, 2004. – 257с.
6. Холзнер С. MicrosoftVisual C++6 с самого начала – СПб: Питер 2005-2006. – 569с.

Приложение А

Система команд МП КР580ИК80А

Мнемоника	Описание команды	Длина, байт	Число тактов
MOV R1, R2	Передача из регистра R2 в регистр R1	1	5
MOV M, R	Передача из регистра в память	1	7
MOV R, M	Передача из памяти в регистр	1	7
MVI R	Передача байта в регистр	2	7
MVI M	Передача байта в память	2	10
LXI RP	Загрузка парных регистров B-C, D-E, H-L, SP	3	10
LDAX RP	Загрузка аккумулятора по адресу, указанному парой регистров B-C или D-E	1	7
STAX RP	Занесение содержимого аккумулятора по адресу, указанному парой регистров B-C или D-E	1	7
LDA	Загрузка аккумулятора по адресу, указанному командой	3	13
STA	Занесение содержимого аккумулятора по адресу, указанному в команде	3	13
LHLD	Загрузка регистров L, H из двух соседних ячеек, начиная с адреса, указанного в команде	3	16
SHLD	Занесение содержимого регистров L, H в две соседние ячейки, начиная с адреса, указанного в команде	3	16
XCHG	Обмен данными между парами регистров H-L и D-E	1	4
XTHL	Обмен данными между SP и H-L	1	18
SPHL	Занесение содержимого регистра H-L в SP	1	5
PUSH RP	Ввод содержимого регистров B-C, D-E или H-L в стек	1	11
PUSH PSW	Ввод PSW в стек	1	11
POP RP	Выдача данных из стека в регистры B-C, D-E, H-L	1	10

POP PSW	Выдача данных из стека в аккумулятор и регистр признаков	1	10
ADD R	Сложение содержимого регистра и аккумулятора	1	4
ADC R	То же, но с учетом переноса СУ	1	4
ADD M	Сложение содержимого ячейки памяти и аккумулятора	1	7
ADC M	То же, но с учетом переноса СУ	1	7
ADI	Сложение байта с содержимым аккумулятора	2	7
ACI	Сложение байта с содержимым аккумулятора с учетом СУ	2	7
DAD RP	Сложение содержимого пар B-C, D-E, H-L, SP с содержимым пары H-L	1	10
SUB R	Вычитание содержимого регистра из содержимого аккумулятора	1	4
SBB R	То же, но с заемом	1	4
SUB M	Вычитание содержимого памяти из содержимого аккумулятора	1	7
SBB M	То же, но с заемом	1	7
SUI	Вычитание байта из содержимого аккумулятора	2	7
SBI	То же, но с заемом	2	7
INR R	Увеличение содержимого регистра на единицу	1	5
INR M	Увеличение содержимого ячейки памяти на единицу	1	10
DCR R	Уменьшение содержимого регистра на единицу	1	5
DCR M	Уменьшение содержимого ячейки памяти на единицу	1	10
INX RP	Увеличение содержимого парных регистров B-C, D-E, H-L, SP на единицу	1	5
DCX RP	Уменьшение содержимого парных регистров B-C, D-E, H-L, SP на единицу	1	5
ANA R	Поразрядное логическое умножение содержимого регистра и аккумулятора	1	4
ANA M	Поразрядное логическое умножение содержимого ячейки памяти и аккумулятора	1	7
ANI	Поразрядное логическое умножение байта и содержимого аккумулятора	2	7
XRA R	Поразрядное исключающее ИЛИ над	1	4

	содержимым регистра и аккумулятора		
XRA M	Поразрядное исключающее ИЛИ над содержимым ячейки памяти и аккумулятора	1	7
XRI	Поразрядное исключающее ИЛИ над содержимым аккумулятора и байтом	2	7
ORA R	Поразрядное логическое сложение содержимого регистра и аккумулятора	1	4
ORA M	Поразрядное логическое сложение содержимого ячейки памяти и аккумулятора	1	7
ORI	Поразрядное логическое сложение содержимого аккумулятора и байта	2	7
CMP R	Сравнение содержимого регистра и аккумулятора	1	4
CMP M	Сравнение содержимого ячейки памяти и аккумулятора	1	7
CPI	Сравнение байта с содержимым аккумулятора	2	7
RLC	Циклический сдвиг содержимого аккумулятора влево	1	4
RRC	То же, но вправо	1	4
RAL	Циклический сдвиг содержимого аккумулятора влево через перенос	1	4
RAR	То же, но вправо	1	4
CMA	Инвертирование аккумулятора	1	4
STC	Установка флага переноса СУ в единицу	1	4
CMC	Инвертирование флага переноса	1	4
DAA	Двоично-десятичная коррекция содержимого аккумулятора	1	4
JMP	Безусловный переход	3	10
JC	Переход при наличии переноса	3	10
JNC	Переход при отсутствии переноса	3	10
JZ	Переход при нуле	3	10
JNZ	Переход при отсутствии нуля	3	10
JP	Переход при плюсе	3	10
JM	Переход при минусе	3	1

			0
JPE	Переход при четности	3	10
JPO	Переход при нечетности	3	10
PCHL	Занесение в счетчик команд содержимого регистра H-L	1	5
CALL	Вызов подпрограммы	3	17
CC	То же, но при переносе	3	11/17
CNC	То же, но при отсутствии переноса	3	11/17
CZ	Вызов подпрограммы при нуле	3	11/17
CNZ	То же, но при отсутствии нуля	3	11/17
CP	Вызов подпрограммы при плюсе	3	11/17
CM	То же, но при минусе	3	11/17
CPE	Вызов подпрограммы при четности	3	11/17
CPO	То же, но при нечетности	3	11/17
RET	Возврат	1	10
RC	Возврат при переносе	1	5/11
RNC	То же, но при отсутствии переноса	1	5/11
RZ	Возврат при нуле	1	5/11
RNZ	То же, но при отсутствии нуля	1	5/11
RP	Возврат при плюсе	1	5/11
RM	Возврат при минусе	1	5/11
RPE	Возврат при четности	1	5/11
RPO	Возврат при нечетности	1	5/11
RST	Повторный запуск	1	11

			1
IN	Ввод	2	10
OUT	Вывод	2	10
EI	Разрешить прерывание	1	4
DI	Запретить прерывание	1	4
NOP	Отсутствие операции	1	4
HLT	Останов	1	7

Примечания: 1. DDD, SSS – 3-разрядные поля в формате команды, адресующие один из регистров общего назначения или в качестве места назначения (D) или в качестве источника операнда (S); 2. RR – 2-разрядное поле в формате команды, адресующее один из парных регистров; 3. PSW – слово-состояние программы, первый байт которого равен содержимому A, второй – содержимому RS; 4. NNN – двоичное представление номера команды RST; 5. В знаменателе дроби указано число тактов при выполнении рассматриваемого в команде условия, в числителе – при не выполнении.

Формат регистра флагов

			C				

S – знак (0 – «+», 1 – «-»)

Z – признак нуля (1 – результат операции равен нулю)

AC – вспомогательный перенос из младшей тетрады в старшую

P – паритет (0 – чет, 1 – нечет)

C – перенос из разряда в разряд за границы разрядной сетки (из старшего разряда при сложении или из младшего - при сдвигах.)

Приложение Б

Министерство образования и науки Российской Федерации

Курганский государственный университет

Кафедра «АПП»

ОТЧЁТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по курсу
«Программное обеспечение систем управления»

Выполнил студент группы……

Принял
преподаватель……

Курган 2014

Тактаев Владимир Васильевич
Дмитриева Ольга Венедиктовна

ИЗУЧЕНИЕ МНОГОЗАДАЧНОГО МОНИТОРА РЕАЛЬНОГО ВРЕМЕНИ MON-99

Методические указания
к лабораторной работе по дисциплине
«Программное обеспечение систем управления»
для студентов очной и заочной форм обучения направления 220700.62
«Автоматизация технологических процессов и производств»

Авторская редакция

Подписано в печать 18.03.15	Формат 60x84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ.л. 1,5	Уч.-изд.л. 1,5
Заказ 60	Тираж 25	Не для продажи

РИЦ Курганского государственного университета.
640000, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.