Московский авиационный институт
(Государственный технический университет)

А. В. Бруханский

ЛАБОРАТОРНАЯ РАБОТА

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ НА БАЗЕ МП К580ВМ80 (INTEL 8080)

Методическое пособие по курсу
«Вычислительные системы и микропроцессоры».

Москва - 2008

Цель работы – изучение основных принципов разработки программного обеспечения микропроцессорных устройств обработки сигналов и средств автоматизации программирования, реализованных на универсальных ЭВМ.

1. Общая характеристика программных средств, используемых при разработке микропроцессорных систем.

Требование сокращения сроков создания микропроцессорных систем обработки радиолокационной и радионавигационной информации определяет необходимость автоматизации этого процесса. Существенно ускорить разработку подобных систем позволяют как специальные аппаратно-программные отладочные комплексы, так и программные средства, реализованные на ЭВМ различных типов.

Программные средства, подготовленные для использования в проектируемой микропроцессорной системе, а также уже эксплуатируемые на базе ее аппаратуры и облегчающие подготовку новых программ, называются резидентным программным обеспечением. Программные средства, реализованные и эксплуатируемые на внешней ЭВМ и облегчающие разработку программ и аппаратной части микропроцессорной системы, носят название кросспрограммного обеспечения.

Универсальные ЭВМ, обладающие развитой операционной системой, широким набором сервисных программ, значительным объемом оперативной и внешней памяти, представляют большие возможности разработчикам микропроцессорных систем. Основными компонентами кросспрограммного обеспечения являются следующие программы: трансляторы, интерпретаторы, редакторы связей (компоновщики), загрузчики, отладчики, редакторы текстов, эмуляторы.

Трансляторы обеспечивают перевод программы, написанной на языке ассемблера или языках высокого уровня, например, Фортране, PL/M, Паскале, Бейсике, в последовательность кодированных команд конкретного микропроцессора. Результатом работы транслятора является объектный модуль – программа в кодах микропроцессора, содержащая обычно некоторую служебную информацию об обращениях к отсутствующим в ней подпрограммам, о точках входа в программу и т.п. Транслятор с машинно-ориентированного языка ассемблера называют ассемблером, а трансляторы с языков высокого уровня – компиляторами. Хотя языки высокого уровня существенно упрощают процесс программирования, однако приводят к менее эффективным программам. Эти программы занимают на 15-200% больше памяти и требуют для выполнения на 15-300% больше времени, чем программы, написанные на ассемблере опытным программистом [2]. В микропроцессорных системах обработки сигналов резерв времени на выполнение необходимых операций, как правило, невелик, а объем памяти ограничен, что является решающим обстоятельством в пользу составления программ на ассемблере. К тому же ассемблер является единственным языком, обеспечивающим полный доступ ко всем программно-управляемым компонентам микропроцессорных систем.

Интерпретаторы позволяют выполнить программу, написанную на алгоритмическом языке, минуя этап ее трансляции – путем ее построчного ввода, различения операторов исходного языка и их последовательного выполнения. Использование интерпретаторов налагает определенные ограничения на конструкции и запись операторов исходного языка. Наиболее удобен для интерпретации язык Бейсик. Объем программы интерпретатора обычно невелик, а его использование в интерактивном (диалоговом) режиме отладки программы весьма удобно, что способствует его распространению на микро-ЭВМ. Однако многократное выполнение программы, что требуется в системах обработки сигналов, средствами ее

интерпретации нецелесообразно, ввиду невысокой скорости этого процесса. В составе кросспрограммного обеспечения интерпретаторы позволяют выполнить программу, подготовленную для микропроцессорной системы, на ЭВМ с другим набором команд.

Редакторы связей или компоновщики объединяют несколько объектных модулей, созданных транслятором, в загрузочный модуль, полностью готовый к выполнению. В некоторых случаях, особенно при программировании на ассемблере, когда программа не содержит обращений к подпрограммам, транслируемым отдельно, использование редактора связей может быть не обязательным.

Загрузчики обеспечивают занесение загрузочного модуля, хранящегося обычно на внешнем накопителе, в оперативную память ЭВМ или микропроцессорной системы. Если загрузочный модуль подготовлен в абсолютном формате, т.е. с указанием конкретных адресов ячеек памяти, то загрузчик помещает его в строго определенное место. Если же загрузочный модуль является перемещаемым, то загрузчик помещает его в произвольную свободную область памяти, предварительно настроив адреса, указанные в модуле по выбранному месту памяти.

Использование отладчиков ускоряет процесс поиска и устранения ошибок в программе. С их помощью можно приостановить выполнение программы в любой точке, вывести на печать содержимое регистров микропроцессора и произвольных ячеек памяти, организовать автоматическое слежение за изменением содержимого определенных областей памяти и порядком выполнения операторов программы. Отладчики позволяют также выполнять программу в пошаговом режиме.

Редакторы текста предназначены для ввода текста программы с клавиатуры дисплея и внесения в нее изменений. Перемещая «окно» редактора, через которое «виден» отображаемый фрагмент вдоль текста, легко найти нужное место программы, следить за правильностью вносимых изменений. Редакторы текстов позволяют вставлять, удалять и изменять отдельные символы и строки программы, переносить фрагменты внутри программы. С их помощью можно автоматически найти фрагмент текста, содержащий требуемое слово или сочетание символов.

Программа эмулятор моделирует выполнение команд микропроцессора на внешней ЭВМ с другим набором команд. С помощью эмулятора можно отлаживать программы микропроцессорных систем до их воплощения в аппаратуре в условиях, близких к реальным. Программно эмулировать (воспроизводить) можно не только работу микропроцессора, но и других элементов микропроцессорной системы – памяти, интерфейсов ввода/вывода информации, контроллеров и других устройств, т.е. моделировать выполнение программы в микропроцессорной системе вполне определенной структуры с учетом всех внутренних связей.

Требования к управляющим программам микропроцессорных устройств обработки сигналов и методы их оптимизации

Микропроцессорные устройства обнаружения, оценивания и фильтрации сигналов являются специализированными управляющими и вычислительными устройствами, аппаратурные и программные средства которых жестко определяются требованиями конкретной задачи. Вследствие этого объем их оперативной памяти обычно невелик, программные средства исчерпываются набором управляющих программ, занесенных в постоянное запоминающее устройство (ПЗУ). В конфигурации внешних устройств таких систем, как правило, отсутствуют накопители большой емкости на магнитных носителях. Эти обстоятельства делают очень трудным или вообще невозможным разработку и отладку их программного обеспечения без использования кросс - программных средств.

К разработке управляющей программы МП - устройства обработки сигналов приступают после того, как пройден этап формализации задачи и записаны алгоритмы работы устройства. К началу составления программы должен быть выбран тип микропроцессора, решен вопрос о способе обмена информацией со всеми внешними устройствами (программный ввод-вывод, по прерываниям, прямой доступ к памяти), а также осуществлено предварительное разделение всех выполняемых устройством функций на выполняемые аппаратурно (специализированными устройствами) и выполняемые программно (процессором).

Основными исходными данными к составлению управляющих программ МП – устройств обработки сигналов являются: темп поступления входных данных, их разрядность и формат представления, максимальная задержка с выдачей очередного результата обработки (управляющей информации) или интервал времени, отведенный на обработку всего массива данных, точность (разрядность) результата расчетов, способ обмена информацией с каждым из внешних устройств, число источников прерываний и их приоритеты. Дополнительными условиями, непосредственно сказывающимся на разрабатываемой программе, является максимально допустимый объем постоянного и оперативного ЗУ, возможность применения средств аппаратурной поддержки вычислений (матричных перемножителей, функциональных преобразователей), возможность параллельного выполнения операций в многопроцессорном устройстве.

Ряд исходных данных может быть уточнен в ходе составления программы, например, выбор способа обмена с внешними устройствами, разделение функций на программно и аппаратурно выполняемые, в зависимости от соотношения требуемого и фактического времени выполнения разработанной программы. В целом нужно стремиться к максимальной загрузке процессора за счет отказа от дополнительных аппаратурных средств, а также к сокращению времени выполнения программы и объема занимаемой памяти. Эти характеристики управляющей программы определяют ее эффективность.

Стремление повысить эффективность программы диктует использование ассемблера вместо языков высокого уровня, оптимизацию режима обмена с внешними устройствами, замену сложных в вычислительном отношении операций на более простые. Например, округление постоянных коэффициентов (множителей) до значений, кратных целой степени числа 2, позволяет заменить умножение и деление на полноразрядное число совокупностью небольшого числа арифметических сдвигов и сложений; операция объединения квадратурных составляющих $z=\sqrt{x^2+y^2}$ может быть приближенно заменена выражением $z\approx\max(|x|,|y|)+0.5\min(|x|,|y|)$ или еще более простым алгоритмом $z\approx|x|+|y|$ и т.д.

Сокращение объема программы достигается оформлением совокупности повторяющихся операторов в виде подпрограммы, однако это ведет к некоторому увеличению времени ее выполнения.

Поскольку команды микропроцессора, связанные с обращением к памяти, выполняются дольше команд, использующих только его рабочие регистры, то существенного сокращения времени выполнения программы можно добиться за счет минимизации числа обращений к памяти и рационального использования регистров. Так команда прямой загрузки регистра-аккумулятора микропроцессора серии КР580 содержит ячейки памяти LDA (см. Приложение 1) требует для выполнения 13 тактов и 4-х обращений к памяти, команда косвенной загрузки регистра MOV R,M – 7 тактов и 2-х обращений к памяти, а команда загрузки в пару рабочих регистров двух байт из области стека POP – 5,5 тактов и 1,5 обращений к памяти на 1 загружаемый байт. Использование однобайтных команд с косвенной адресацией и команд работы со стеком одновременно сокращает и объем программы.

Наконец, радикальным средством сокращения объема программы и времени ее выполнения является применение другого микропроцессора – более быстродействующего и с большим числом разрядов. Увеличение разрядности слов, обрабатываемых микропроцессором за одну операцию, позволяет отказаться от программного наращивания разрядности, т.е. последовательной обработки разрядов (байт) многоразрядного (многобайтного) слова – одной из главных причин снижения эффективности программы при высоких требованиях к точности вычислений. В случае использования разрядно-модульных микропроцессоров увеличение разрядности слова достигается добавлением необходимого количества процессорных секций и расширением шины данных. Здесь особенно просто обмениваются объем программы и время ее выполнения на аппаратурные затраты.

Правила записи программы на языке ассемблера

Язык ассемблера(ассемблер) микропроцессора КР580, реализованный в кросс-системе программирования на ПЭВМ соответствует общепринятой версии языка [2].

Строка записи программы во входном файле состоит их 4-х полей:

поле метки;

поле команды (псевдокоманды);

поле операндов;

поле комментария.

Использование перечисленных полей другим образом не допускается, за исключением случая, когда поле команды пустое: при этом все остальные позиции строки с первой по седьмую и с тринадцатой по сорок восьмую считаются комментарием. Метка может иметь произвольную длину и начинаться с буквы латинского алфавита. Значимыми считаются 8 первых символов метки.

Метка отделяется от обозначения команды (псевдокоманды) двоеточием.

Мнемоническое обозначение команды микропроцессора (см.Приложение 1) и псевдокоманды (директивы) ассемблера (см.ниже), как и метка, могут занимать любые позиции внутри отведенных им полей. Обозначения команд переводятся кросс-ассемблером в объективный код, а псевдокоманды служат для управления процессом трансляции и размещения в памяти требуемых констант. Перед мнемоникой псевдокоманды ставится точка.

Программа кросс-ассемблер принимает следующие псевдокоманды:

.org (адрес) – псевдокоманда установки начального адреса программы. Если программа начинается с этой псевдокоманды, то отсчет адресов команд в листинге трансляции начинается с адреса (адрес). При этом в файл объектного модуля заносится информация для загрузчика о начальном адресе загрузки программы в память микропроцессорной системы.

.equ (идентификатор),(шестнадцатеричная константа) – псевдокоманда определения значения метки. Метка, определенная с помощью этой псевдокоманды, может использоваться в трехбайтовых командах микропроцессора в качестве символического адреса программы или непосредственного операнда, и в двухбайтовых командах – в качестве адреса устройства ввода/вывода или непосредственного операнда.

(метка) .db (байт данных(- псевдокоманда определения байта данных. По этой псевдокоманде транслятор заносит в объектный модуль указанный байт данных. Метка псевдокоманды не является обязательной.

(метка) .dw (символический адрес) – псевдокоманда определения двух байт данных, в соответствии которой транслятор заносит в объектный модуль числовое

значение символического адреса, представляющего собой метку команды или псевдокоманды. Метка перед DW не обязательна.

(метка) .rs (десятичная константа) – псевдокоманда резервирования области памяти. Выполняя эту псевдокоманду, транслятор пропускает в объектном модуле число байт,заданное десятичной константой. Данная область используется во время выполнения программы для записи выходной информации или хранения результатов промежуточных операций. Метка не обязательна.

.end – псевдокоманда, указывающая транслятору конец текста программы. Она всегда завершает программу на ассемблере.

В качестве операндов в командах микропроцессора могут указываться обозначения регистров A, B, C, D, У, H, L, регистровых пар B, D, H, SP, где SP – обозначение указателя стека (SP), ячейки памяти M. В двух и трехбайтовых командах операндом может быть однобайтовая константа в шестнадцатеричной форме, символическое обозначение одно – и двухбайтовой константы, метка команды или псевдокоманды. Все константы в программе задаются в десятичной, шестнадцатеричной, восьмеричной или двоичной форме с префиксами d', h' или 0x , 0 или o' и b' соответственно. В качестве операндов разрешается использовать арифметические и логические выражения.

Использование кросс-программных средств для ввода и трансляции программы

Для ввода исходного текста программы можно использовать программы Блокнот, WordPad или пользоваться встроенным редактором текста любого файлового менеджера.

Трансляция программы выполняется с помощью ассемблера PseudoSam 85 компании PseudoCode – исполняемый файл A85.com. Вызов транслятора осуществляется из командной строки:

A85.com имя_файла_программы

Результатом трансляции является файл с объектным кодом имя_файла_программы.obj и листинг трансляции имя_файла_программы.lst, содержащий диагностику ошибок трансляции.

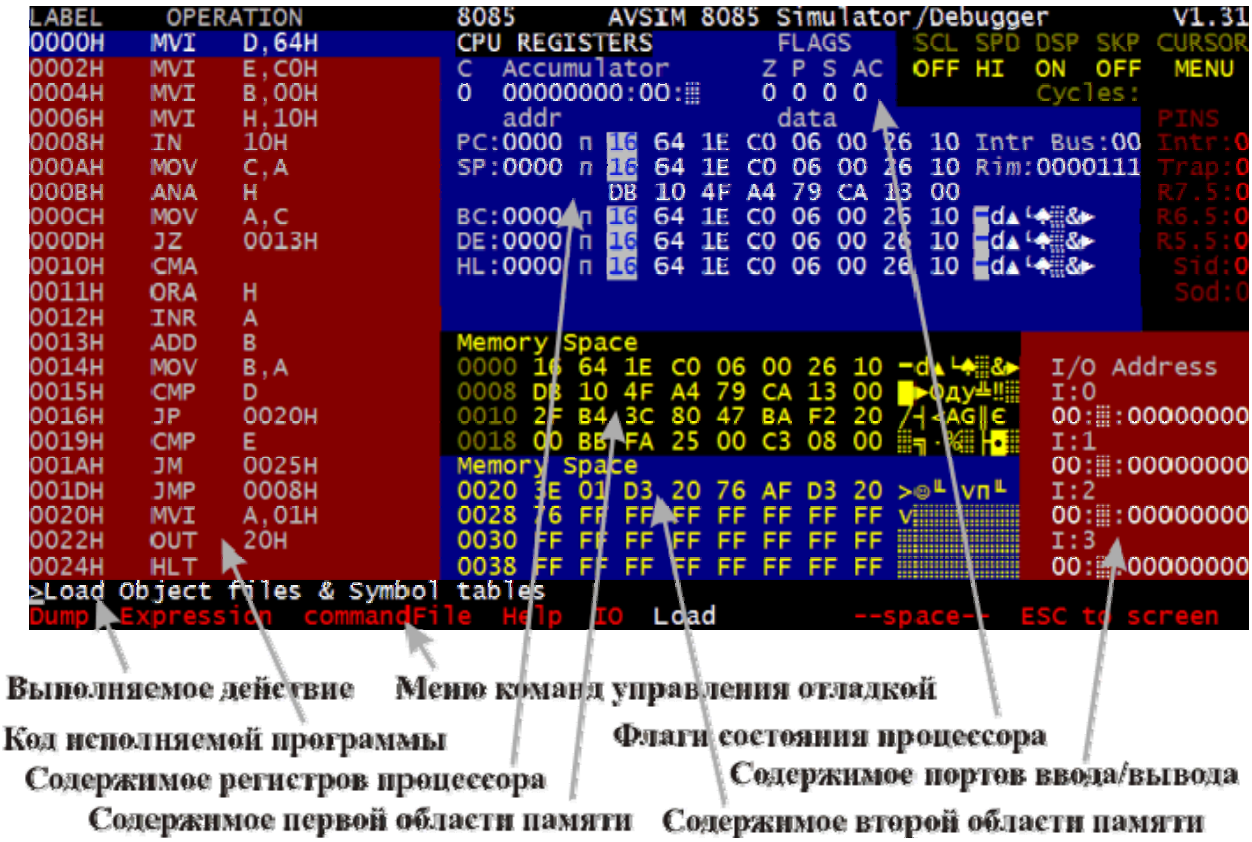
Полученный объектный код загружается в программу эмулятора и отладчика AVSIM 85 (исполняемый файл avsim85.exe). Запуск эмулятора осуществляется также из командной строки с параметром –c1:

avsim85.exe –c1

Загрузка объектного файла программы в эмулятор выполняется командой load program из командного меню. Эмулятор отображает объектный код программы в виде ассемблерного листинга и позволяет выполнить программу, в том числе, в пошаговом режиме. В процессе выполнения можно ввести числовые данные из портов ввода, посмотреть содержимое регистров процессора и оперативной памяти после любой выполненной команды эмулируемого процессора. При необходимости в тексте программы задаются точки останова (прерывания выполнения).

Вид окна эмулятора AVSIM 85 с загруженной программой показан ниже.

Перечень команд управления эмуляцией и отладкой приведен в приложении 2.



Пример разработки программы цифрового фильтра

Пусть требуется разработать программу микропроцессорной системы, осуществляющей обработку последовательности входных отсчетов $x_1, x_2, \ldots, x_{20}$ (пачки квантованных на 64 уровня импульсов) в соответствии с алгоритмом двукратной чересперiодной компенсации:

$$y_к = x_к – 2x_{к-1} + x_{к-2}$$

Предполагается, что массив входных отсчетов к моменту начала работы программы записан в оперативной памяти устройства, начиная с адреса BX=12B0, (адрес задается шестнадцатиричным числом) по отсчету в байт. Место для массива для входных отсчетов необходимо отвести внутри самой программы.

В приведенном задании не требуется обеспечить ввод/вывод информации на внешние устройства, поэтому задача выбора способа обмена не стоит. Разрядность входных отсчетов не превышает 6. Если они положительны (фильтрация осуществляется после амплитудного детектора), то при 8 разрядном представлении результата фильтрации и промежуточных операций, переполнение разрядной сетки в процессе вычислений исключено.

В задании не указано предельное время на выполнение фильтрации. Это означает, что имеется либо значительный резерв времени, либо возможность распараллеливать процесс вычислений, причем число параллельно работающих микропроцессоров находится исходя из времени выполнения составленной программы. В данном примере, если устройство является многоканальным, то число каналов, которое может обслужить один микропроцессор, определяется как $N=D_п/T_{вып.}$ где $D_п$ – длительность пачки, $T_{вып.}$ – время выполнения программы.

Алгоритм двукратной ЧПК прост в реализации, однако при его использовании в случае обработки пачки с фиксированным началом (не в «скользящем окне») необходимо правильно задать начальные условия. Положить $X_0=X_{-1}=0$ нецелесообразно, так как это приведет к резким выбросам выходного

процесса в моменты k = 1 и k = 2. Один отсчет не дает возможности осуществить фильтрацию, при двух отсчетах X_1 и X_2 уже можно использовать алгоритм однократной ЧПК $У_1 = X1 - X_2$ для режекции низкочастотной помехи. При трех и более входных отсчетах используется алгоритм двукратной ЧПК. Отметим, что программно менять вид и параметры алгоритма обработки является достоинством микропроцессорных устройств, таким образом, при пачке объемом 20 импульсов, как определенно в задании, выходной массив данных будет содержать 19 отсчетов, причем первый выходной отсчет $У_1 = X_1 - X_2$.

Структурную схему программы составим применительно к уровню операций с регистрами микропроцессора : A – аккумулятор, B, C, D, E, H , L – регистрами общего назначения, SP – регистром указателем стека. Операции загрузки регистра содержим ячейки памяти будем обозначать R = X, где R – один из регистров, X – содержимое ячейки. Операцию занесения в память обозначим X= R, где X – содержимое регистра, заносимое в память, R – обозначение регистра. Операции над содержимым регистров записываем как над переменными, например : A=B – пересылка содержимого регистра B в аккумулятор, A=A+C - сложение содержимого регистров A и C с занесением суммы в A и т.д.

При разработке программы стремимся время на обращение к памяти и максимально использовать регистры микропроцессора. Примем следующее распределение регистров для хранения информации:

регистр A – отсчет $X_{к-2}$
регистр B – отсчет $X_{к-1}$
регистр C – число оставшихся отсчетов
регистр D – отсчет $X_{к+1}$
регистр E – отсчет $X_к$
регистры H-L – адрес очередного записываемого в память выходного отсчета
регистр SP – адрес очередного загружаемого в регистры входного отсчета.

Такое распределение регистров позволяет за одно обращение к стеку, который назначается в области памяти, содержащей входные отсчеты, загрузить в рабочие регистры для входных отсчета и получить два выходных отсчета.

Структурная схема программы приведена на рис. 1. В блоках 1, 2, 5 осуществляется начальная загрузка регистров в соответствии с принятым распределением. Блок 3 – реализация алгоритма ЧПК – 1 для первых двух отсчетов. Блок 4 – запись $У_1$ в память. Цикл по числу отсчетов начинается с блока 6 – загрузки в пару регистров D- E очередной пары входных отсчетов $X_к$ и $X_{к+1}$. Блоки 7 и 12 – реализация алгоритма ЧПК-2, причем умножение на 2 заменено двукратным вычитанием, а блоки 8 и 13 – занесение в память выходных отсчетов $У_к$ и $У_{к+1}$. В блоках 9 и 15 осуществляется декрементация (уменьшение на 1) содержимого счетчика числа оставшихся отсчетов, а в блоках 10 и 16 – проверка на равенство его нулю. Нулевое содержимое счетчика (регистра C) означает окончание программы. Поскольку число отсчетов четно, то первую проверку (блок 10) можно изъять из программы. Ее включение делает программу более универсальной. В блоке 14 осуществляется пересылка содержимого регистров в соответствии с принятым соглашением, подготавливающая следующий цикл.

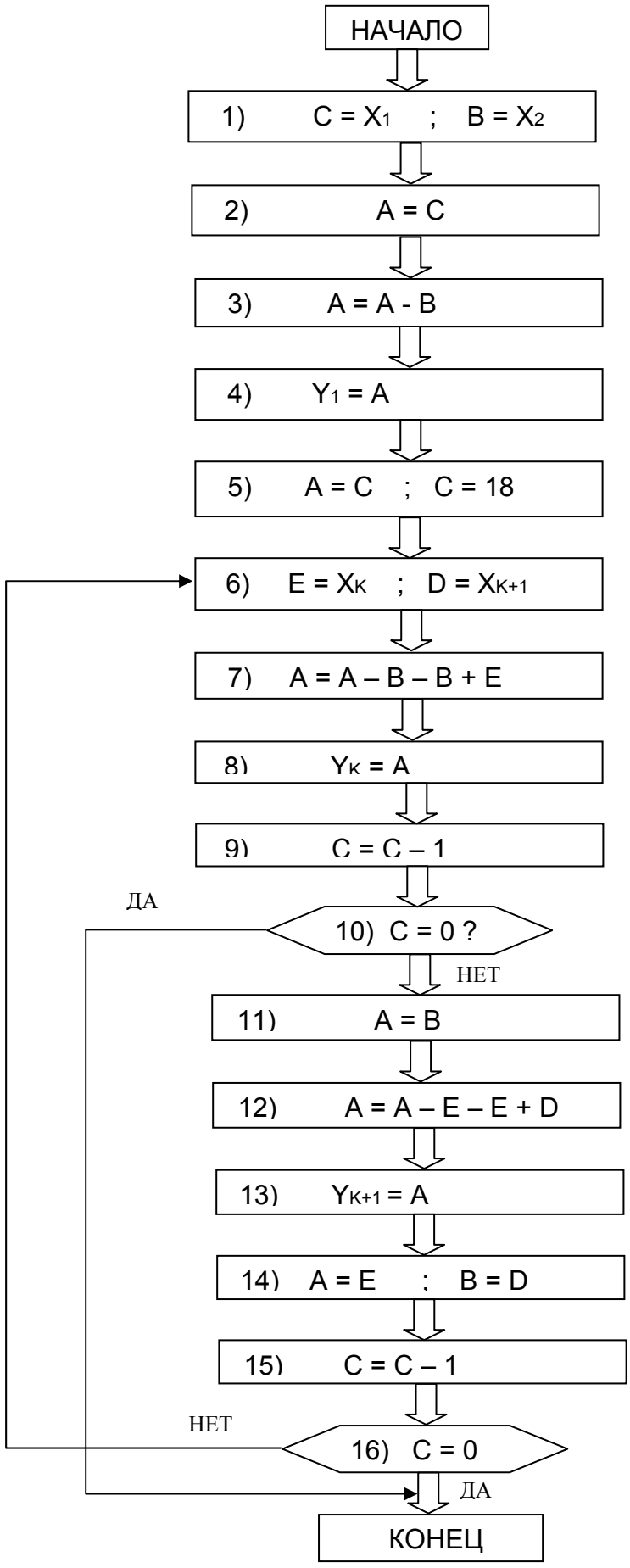

НАЧАЛО
1) C = X1 ; B = X2
2) A = C
3) A = A - B
4) Y1 = A
5) A = C ; C = 18
6) E = XK ; D = XK+1
7) A = A – B – B + E
8) YK = A
9) C = C – 1
ДА
10) C = 0 ?
НЕТ
11) A = B
12) A = A – E – E + D
13) YK+1 = A
14) A = E ; B = D
15) C = C – 1
НЕТ
16) C = 0
ДА
КОНЕЦ

Рис. 1

Структурная схема рис.1 не содержит указаний на способ адресации ячеек памяти, однако из принятых соглашений о регистрах, ясно, что входные и выходные массивы будут адресоваться косвенно – первый как стек, второй – через пару регистров H –L. Во втором случае нужно своевременно инкрементировать (увеличить на 1) содержимое H-L перед записью очередного отсчета в память.

Текст программы на языке ассемблера представлен на рис.2. Псевдокомандой (директивой) ассемблера EQU определяется значение адреса массива входных отсчетов BX, а псевдокомандой DS – резервируется 19 байт под массив выходных отсчетов. Все числовые значения (адреса и непосредственные операнды) могут записываться в программе либо шестнадцатеричными символами с префиксом h', либо десятичными – без префикса или двоичными с префиксом – b'. Так число 18_{10} записывается как h'12, Метки требуется отделять от обозначения команды двоеточием, а комментарии начинать символом «;». Блок 6 структурной схемы в программе помечен меткой CIKL.

ПРОГРАММА ОБРАБОТКИ ПАЧКИ ИМПУЛЬСОВ С ПОМОЩЬЮ ДВУКРАТНОЙ СИСТЕМЫ ЧПК

```
        .EQU   BX, h'12B0;     ЗАДАНИЕ СИМВОЛ. АДРЕСА BX
        LXI    H, BY;          ЗАГР. АДРЕСА  МАССИВА  BY В  H-L
        LXI    SP, BX;         ЗАГР. АДРЕСА МАССИВА BX  В SP
        POP    B;              ЗАГР.  B <= X (2),  C <= X (1)
        MOV    A, C;           ЗАГР.  A  < =  X (1)
        SUB    B;              (A) = X (1)  - X (2) = Y (1)
        MOV    M, A;           ЗАНЕСЕНИЕ Y (1) В ПАМЯТЬ
        MOV    A, C;           ЗАГРУЗКА X (1)  В АККУМУЛЯТОР
        MVI    C, 12;          ЗАГР. C  - СЧЕТЧИКА ОТСЧЕТОВ
CIKL:   POP    D;              ЗАГР.  D  < =  X (К+1) , E  < = X (К)
        SUB    B;              (A) = (A) – X (К-1)
        SUB    B;              (A) = (A) – X (К-1)
        ADD    E;              (A) = (A) + X (К) = Y(К)
        INX    H;              ИНКРЕМЕНТ АДРЕСА В H-L
        MOV    M, A;           ЗАНЕСЕНИЕ  Y (К) В  ПАМЯТЬ
        DCR    C;              ДЕКРЕМЕНТ СЧЕТЧИКА ЦИКЛОВ
        JZ     KON;            ВЫХОД ИЗ ЦИКЛА , ЕСЛИ (C) =0
        MOV    A, B;           ЗАНЕСЕНИЕ  В АККУМУЛЯТОР  X (К-1)
        SUB    E;              (A)  =  (A)  - X  (К)
        SUB    E;              (A)  =  (A)  - X  (К)
        ADD    D;              (A)  =  (A)  - X  (К+1)  = Y (К+1)
        INX    H;              ИНКРЕМЕНТ АДРЕСА  В H-L
        MOV    M, A;           ЗАНЕСЕНИЕ  Y (К+1) В ПАМЯТЬ
        MOV    A, E;           (A)  =  X (К)
        MOV    B, D;           (B)  = X (К+1)
        DCR    C;              ДЕКРЕМЕНТ СЧЕТЧИКА ЦИКЛОВ
        JNZ    CIKL;           ВОЗВРАТ НА CIKL, ЕСЛИ  (C)  >   0
KON:    HLT;                   ОСТАНОВ ПРОЦЕССОРА
BY:     .DS    19;             РЕЗЕРВ.  19 БАЙТ ПОД МАССИВ BY
        .END;                  КОНЕЦ ПРОГРАММЫ
```

Рис. 2

На рис.3 приведен листинг трансляции исходной программы кросс-ассемблером PseudoSam 85 (исполняемый файл A85.com). Первая колонка каждой строки листинга – номер строки исходного текста, вторая колонка – адрес или константа, определенная директивой EQU, четвертая – машинный код процессора (шестнадцатеричный результат трансляции), пятая – мнемоника команды, шестая – операнды команды, седьмая – комментарий.

```
000001 12B0             .EQU   BX, h'12B0;   ЗАДАНИЕ СИМВОЛ. АДРЕСА BX
000002 0000 212400      LXI    H, BY;        ЗАГР. АДРЕСА  МАССИВА  BY В  H-L
000003 0003 31B012      LXI    SP, BX;       ЗАГР. АДРЕСА МАССИВА BX  В SP
000004 0006 C1          POP    B;            ЗАГР.  B <= X (2),  C <= X (1)
000005 0007 79          MOV    A, C;         ЗАГР.  A  < =  X (1)
000006 0008 90          SUB    B;            (A) = X (1)  - X (2) = Y (1)
000007 0009 77          MOV    M, A;         ЗАНЕСЕНИЕ Y (1) В ПАМЯТЬ
000008 000A 79          MOV    A, C;         ЗАГРУЗКА X (1)  В АККУМУЛЯТОР
000009 000B 0E0C        MVI    C, 12;        ЗАГР. C  - СЧЕТЧИКА ОТСЧЕТОВ
000010 000D D1          CIKL:  POP    D;            ЗАГР.  D  < =  X (К+1) , E  < = X (К)
000011 000E 90          SUB    B;            (A) = (A) – X (К-1)
000012 000F 90          SUB    B;            (A) = (A) – X (К-1)
000013 0010 83          ADD    E;            (A) = (A) + X (К) = Y(К)
000014 0011 23          INX    H;            ИНКРЕМЕНТ АДРЕСА В H-L
000015 0012 77          MOV    M, A;         ЗАНЕСЕНИЕ  Y (К) В  ПАМЯТЬ
000016 0013 0D          DCR    C;            ДЕКРЕМЕНТ СЧЕТЧИКА ЦИКЛОВ
000017 0014 CA2300      JZ     KON;          ВЫХОД ИЗ ЦИКЛА , ЕСЛИ (C) =0
000018 0017 78          MOV    A, B;         ЗАНЕСЕНИЕ  В АККУМУЛЯТОР  X (К-1)
000019 0018 93          SUB    E;            (A)  =  (A)  - X  (К)
000020 0019 93          SUB    E;            (A)  =  (A)  - X  (К)
000021 001A 82          ADD    D;            (A)  =  (A)  - X  (К+1)  = Y (К+1)
000022 001B 23          INX    H;            ИНКРЕМЕНТ АДРЕСА  В H-L
000023 001C 77          MOV    M, A;         ЗАНЕСЕНИЕ  Y (К+1) В ПАМЯТЬ
000024 001D 7B          MOV    A, E;         (A)  =  X (К)
000025 001E 42          MOV    B, D;         (B)  = X (К+1)
000026 001F 0D          DCR    C;            ДЕКРЕМЕНТ СЧЕТЧИКА ЦИКЛОВ
000027 0020 C20D00      JNZ    CIKL;         ВОЗВРАТ НА CIKL, ЕСЛИ  (C)  >   0
000028 0023 76          KON:   HLT;          ОСТАНОВ ПРОЦЕССОРА
000029 0024             BY:    .DS     19;   РЕЗЕРВ.  19 БАЙТ ПОД МАССИВ BY
000030 0024             .END;                КОНЕЦ ПРОГРАММЫ

Константы, определенные ассемблером:
BX      =12B0
BY      =0024
CIKL    =000D
KON     =0023
```

Рис. 3

Одновременно с листингом программы кросс-ассемблером формируется файл объектного модуля, содержащий только управляющие коды микропроцессора КР580. Файл создается на той же папке, в которой записана исходная программа. При необходимости его можно просмотреть на экране дисплея или распечатать с помощью принтера.

Порядок выполнения работы

1. Получить у преподавателя задание на разработку программы.
2. Составить структурную схему программы и утвердить ее у преподавателя.
3. Подготовить программу на языке ассемблера.
4. Записать программу в файл на магнитном диске под именем «имяпрограммы.ASM», пользуясь экранным редактором текстов.
5. Выполнить трансляцию программы кросс-ассемблером А85
6. При наличии синтаксических ошибок в программе просмотреть файл листинга «имяпрограммы.lst». С помощью редактора текстов, исправить очевидные ошибки, вызвав на редактирование файл «имяпрограммы.ASM», и провести повторную трансляцию программы.
7. Вывести файл листинга «имяпрограммы.lst» на печать.

8. Проанализировать оставшиеся ошибки, а также особенности трансляции двухбайтовых и трехбайтовых команд, представления абсолютных адресов в листинге программы.

9. Подсчитать время выполнения программы в машинных тактах, пользуясь Приложением 1.

10. Запустить эмулятор выполнения программ процессора i8085 – файл AVSIM85.EXE

11. Выполнить программу в пошаговом режиме с помощью эмулятора и отладчика AVSIM85.

12. Убедиться в правильности выполнения кода программы.

Содержание отчета

1. Задание на разработку программы.
2. Структурная схема программы.
3. Листинг трансляции программы.
4. Пояснения ошибок в листинге и результат подсчета времени выполнения программы.

Контрольные вопросы

1. Поясните различия резидентного и кросс-программного обеспечения.
2. Каково назначение трансляторов, интерпретаторов, редакторов связей, компоновщиков, загрузчиков, отладчиков, редакторов текста, эмуляторов?
3. В чем преимущества языка ассемблера перед языками высокого уровня при составлении управляющих программ устройств обработки сигналов?
4. Какие этапы обработки проходит программа управления микропроцессорной системой, написанная на алгоритмическом языке внешней ЭВМ до момента ее записи в ПЗУ микропроцессорной системы?
5. Какие требования предъявляются к управляющим программам микропроцессорных устройств обработки сигналов?
6. Укажите пути оптимизации управляющих программ по объему требуемой памяти и быстродействию.

Литература

1. Бруханский А.В., Жуков С.Г., Карташкин А.С. Микропроцессорные устройства обработки радиолокационной и радионавигационной информации. – М., МАИ, 1987
2. Алексеенко А,Г, Галицын А.А., Иванников А.Д. Проектирование радиоэлектронной аппаратуры на микропроцессорах. - М.,Радио и связь, 1984.
3. Материалы веб-сайта кафедры 403 по 8-разрядным микропроцессорам (http://kaf403.rloc.ru/).

ПРИЛОЖЕНИЕ 1. ПЕРЕЧЕНЬ КОМАНД МП СЕРИИ КР580 (i8080)

Обозначения:

r или r1 или r2 – один из регистров микропроцессора A, B, C, D, E, H, L;
A -аккумулятор ;
(r), (r1), (r2), (A) - содержимое регистров ;
rp – одна из регистровых пар B (B-C), D (D-E), H (H-L) или указатель стека SP ;
data8 - однобайтовая константа (непосредственный операнд);
data16 – двухбайтовая константа (непосредственный операнд) ;
addr – двухбайтовый адрес основной памяти или метки программы;
port – однобайтовый адрес внешнего устройства или его символическое обозначение.

Операнд, обозначаемый в команде символом M, указывает ячейку памяти, адрес которой находится в паре регистров H- L.

Коды регистров источника (SSS) и приемника (DDD) указаны в таблице 2.

Признаки результата (S, Z, P, C и AC) устанавливаются лишь при выполнении большинства арифметических и логических команд, а также команды POP PSW. В табл. 1 для каждой из команд дан перечень устанавливаемых признаков (в командах INX, DCX и CMA признаки не устанавливаются).

В командах условного перехода, условного обращения к подпрограмме и условного возврата из подпрограммы используются коды условий (CCC) из табл. 3. Мнемонические обозначения этих команд составляются из символов J, C или R и соответствующих символов (cc) из табл. 3 (например, JNC, CP, RM).

Число тактов, необходимых для исполнения команд условного обращения к подпрограмме и условного возврата из подпрограммы, зависит от того, выполнено (знаменатель дроби) или не выполнено (числитель дроби) условие, указанное в команде.

Табл. 1. Команды процессора КР580ИК80А (i8080)

Список команд пересылки МП КР580ИК80А

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание
				Пересылка
MOV r1, r2	01DDDSSS	а	5	(r2) ⇒ (r1)
MOV r, M	01DDD110	а	7	(M) ⇒ (r)
MOV M, r	01110SSS	а	7	(r) ⇒ (M)
MVI r, data 8	00DDD110	з	7	data⇒ (r)
MVI M, data 8	00110110	з	10	data⇒ (M)
				Загрузка
LXI rp, data 16	00RP0001	л	10	data⇒ (rp)
LDA addr	00111010	к	13	(addr) ⇒ (A)
LHLD addr	00101010	к	16	(addr) ⇒ (L), (addr + 1) ⇒ (H)
LDAX rp	00RP1010	д	7	(rp) ⇒ (A)
				Обмен
XCHG	11101011	г	4	(H) ⇔ (D), (L) ⇔ (E)
				Запись
STA addr	00110010	к	13	(A) ⇒ (addr)
SHLD addr	00100010	к	16	(L) ⇒ (addr), (H) ⇒ (addr + 1)
STAX rp	00RP0010	д	7	(A) ⇒ (rp)

Список арифметических и логических команд МП КР580ИК80А

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание	Устанавливаемые признаки
				Сложение	
ADD r	10000SSS	б	4	(A) + (r) ⇒ (A)	
ADD M	10000110	б	7	(A) + (M) ⇒ (A)	S, Z, P,
ADI data 8	11000110	ж	7	(A) + data 8 ⇒ (A)	C, AC
				Сложение с переносом	
ADC r	10001SSS	б	4	(A) + (r) + (C) ⇒ (A)	
ADC M	10001110	б	7	(A) + (M) + (C) ⇒ (A)	S, Z, P,
ACI data 8	11001110	ж	7	(A) + data 8 + (C) ⇒ (A)	C, AC
				Вычитание	
SUB r	10010SSS	б	4	(A) — (r) ⇒ (A)	
SUB M	10010110	б	7	(A) — (M) ⇒ (A)	S, Z, P,
SUI data 8	11010110	ж	7	(A) — data 8 ⇒ (A)	C, AC
				Вычитание с заемом	
SBB r	10011SSS	б	4	(A) — (r) — (C) ⇒ (A)	
SBB M	10011110	б	7	(A) — (M) — (C) ⇒ (A)	S, Z, P,
SBI data	11011110	ж	7	(A) — data 8 — (C) ⇒ (A)	C, AC
				Логическое умножение	
ANA r	10100SSS	б	4	(A) AND (r) ⇒ (A)	
ANA M	10100110	б	7	(A) AND (M) ⇒ (A)	S, Z, P,
ANI data 8	11100110	ж	7	(A) AND data 8 ⇒ (A)	C, AC
				Исключающее ИЛИ	
XRA r	10101SSS	б	4	(A) XOR (r) ⇒ (A)	
XRA M	10101110	б	7	(A) XOR (M) ⇒ (A)	S, Z, P,
XRI data	11101110	ж	7	(A) XOR data 8 ⇒ (A)	C, AC
				ИЛИ	
ORA r	10110SSS	б	4	(A) OR (r) ⇒ (A)	S, Z, P,
ORA M	10110110	б	7	(A) OR (M) ⇒ (A)	C = 0,
ORI data 8	11110110	ж	7	(A) OR data 8 ⇒ (A)	AC = 0
				Сравнение	
CMP r	10111SSS	б	4	(A) — (r)	
CMP M	10111110	б	7	(A) — (M)	S, Z, P,
CPI data 8	11111110	б	7	(A) — data 8	C, AC
				Инкремент	
INR r	00DDD100	в	5	(r) + 1 ⇒ (r)	S, Z, P,
INR M	00110100	в	10	(M) + 1 ⇒ (M)	AC
				Декремент	
INX rp	00RP0011	д	5	(rp) + 1 ⇒ (rp)	—
DCR r	OODDD101	в	5	(r) — 1 ⇒ (r)	S, Z, P,
DCR M	00110101	в	10	(M) — 1 ⇒ (M)	AC
DCX rp	00RP1011	д	5	(rp) — 1 ⇒ (rp)	—
				Сложение содержимых регистровых пар	
DAD rp	00RP1001	д	10	(H, L) + (rp) ⇒ (H, L)	C
				Циклический сдвиг влево	
RLC	00000111	г	4	Все биты A смещаются на один разряд влево. Старший разряд A переходит в его нулевой разряд и регистр признака C	C
				Циклический сдвиг вправо	
RRC	00001111	г	4	Выполняется аналогично RLC. Старший разряд A и C приобретают значение младшего разряда A	C
				Арифметический сдвиг влево через перенос	
RAL	00010111	г	4	Все биты A сдвигаются на один разряд влево; старший разряд A переходит в C, а C — в младший разряд A	C
				Арифметический сдвиг вправо через перенос	
RAR	00011111	г	4	Выполняется аналогично RAL, но сдвиг вправо	C
				Десятичная коррекция аккумулятора	
DAA	00100111	г	4	Преобразование содержимого A в двоично-десятичный код	S, Z, P, C, AC
				Инвертирование A	
CMA	00101111	г	4	Получение обратного кода (инверсии) содержимого A	—
				Установка переноса в 1	
STC	00110111	г	4	1 ⇒ C	C = 1
				Инвертирование переноса	
CMC	00111111	г	4	Инвертирование содержимого регистра C	C

Список команд управления, ввода-вывода и работы со стеком

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание
JMP addr	11000011	к	10	Безусловный переход к команде, адрес которой определяется вторым и третьим байтами данной команды
PCHL	11101001	г	5	Косвенный переход по адресу, указанному в регистрах H, L
Jcc addr	11CCC010	и	10	Условные переходы: если условие (cc из табл. 2) истинно, то переход к команде, адрес которой определяется вторым и третьим байтами данной команды; иначе выполняется команда, расположенная вслед за данной
CALL addr	11001101	к	17	Обращение к подпрограмме: содержимое счетчика команд заносится в стек по адресу, на который указывает SP; содержимое SP уменьшается на 2 и выполняется переход к команде, адрес которой определяется вторым и третьим байтами данной команды
Ccc addr	11CCC100	и	11/17	Условное обращение к подпрограмме: если условие (cc из табл. 2) истинно, то выполняется команда CALL; иначе выполняется команда, расположенная вслед за данной
RET	11001001	г	10	Возврат из подпрограммы: переход к команде, адрес которой записан в верхней паре ячеек стека, и увеличение содержимого SP на 2
Rcc	11CCC000	в	5/11	Условный возврат из подпрограммы: если условие (cc из табл. 2) истинно, то выполняется команда RET; иначе выполняется команда, расположенная вслед за данной
RST n	11NNN111	в	11	Повторный запуск МП с адреса 8·NNN (0, 8, 16,…,56)
RUSH rp	11RP0101	д	11	Запись в стек содержимого пары регистров (rp)
POP rp	11RP0001	д	10	Считывание данных из стека в пару регистров (rp); при выдаче PSW происходит установка S, Z, P, C и AC
XTHL	11100011	г	18	Обмен содержимым верхней пары ячеек стека и пары регистров H, L
SPHL	11111001	г	5	Пересылка содержимого пары регистров H, L в указатель стека SP
IN port	11011011	е	10	Ввод в аккумулятор данных из адресуемого порта
OUT port	11010011	е	10	Вывод данных из аккумулятора в адресуемый порт
EI	11111011	г	4	Разрешение прерывания
DI	11110011	г	4	Запрещение прерывания
HLT	01110110	г	7	Останов
NOP	00000000	г	4	Нет операции

Табл. 2. Коды регистров и пар регистров, используемые в командах МП

Регистры				Пары регистров			
Код	Имя (r)	Код	Имя (r)	Код (RP)	Имя пары (rp)	Регистры пары	
						старший	младший
000	B	100	H	00	B	B	C
001	C	101	L	01	D	D	E
010	D	110	M (память)	10	H	H	L
011	E	111	A (аккумулятор)	11	PSW	A	PSW

Табл. 3. Коды условий, используемые в командах условных переходов

Код (CCC)	Мнемоника (cc)	Условие	Код (CCC)	Мнемоника (cc)	Условие
000	NZ	Не нуль (Z=0)	001	Z	Нуль (Z = l)
010	NC	Нет переноса (C = 0)	011	C	Перенос (C = 1)
100	PO	Нечетность (P = 0)	101	PE	Четность (P = 1)
110	P	Плюс (S = 0)	111	M	Минус (S = l)

Приложение 2. Команды эмулятора AVSIM85

Управление эмуляцией

<F1> Пуск/Стоп – старт процесса выполнения программы и его приостановка
<F10> Шаг - эмуляция выполнения одной команды процессора
<F9> Отменить – Возврат на одну команду назад
<F2> Переместить указатель точки прерывания на шаг вверх
<F4> Переместить указатель точки прерывания на шаг вниз
<F3> Установка ТП - Установить динамическую точку прерывания в месте положения указателя
<F5> Скорость – установка скорости процесса эмуляции

Управление переключателями

<Ctrl>+<Page Up> - переключение режима прокрутки областей отображения памяти Memory Space, содержимого регистров и областей памяти, отображаемых справа от содержимого регистров процессора
<Alt>+<F5> Переключение "Метка/Адрес": Адреса и операнды отображаются либо символически (Label), либо в шестнадцатеричном формате (ADDR)
<F6> Включение/Выключения обновления экрана после выполнения каждой инструкции в режиме автоматического выполнения программы
ON: Экран обновляется после каждой выполненной команды
OFF: Обновляется только окно трассировки до момента прерывания
<Alt>+<F6> Переключение режима обновления
ON: Окно обновляется даже когда переключатель обновления трассировки экрана в состоянии OFF
OFF: Окно обновляется только если переключатель обновления экрана – в состоянии ON
<F7> Тип курсора - Hex / ASCII / Binary (Шестнадцатер./Символьный/Двоичный). Курсор перемещается к месту соответствующего отображению выбранного объекта, если тот отображается в разных формах
<F8> Переключение обхода вызовов подпрограмм
SKIP ON – выполнение подпрограммы по команде CALL за один шаг, без ее пошагового выполнения
SKIP OFF – трассировка подпрограммы, вызванной командой CALL.

Меню команд управления

Большинство команд и подкоманд, перечисленных в нижней строке окна эмулятора, вызывается при нажатии клавиши с начальной буквой команды.

Esc – переключает положение курсора из меню команд на экран отображения состояния микропроцессора и памяти и обратно. Клавиши управления процессом эмуляции и клавиша Esc продолжают работать все время.

Ctl-C – позволяет выйти из любой команды на меню верхнего уровня.

Quit – выход из программы эмулятора. Для подтверждения выхода необходимо выбрать подкоманду **Exit** или нажать клавишу «**E**».

Load – Загрузка программы или блока данных из объектного файла в шестнадцатеричном формате. При выборе подкоманды **Program** код программы загружается в эмулируемую постоянную память (ROM), при выборе Data – код или данные загружаются в эмулятор ОЗУ (RAM). Выбор подкоманды **Symbol-table** позволяет загрузить таблицу символов (символическое отображение меток и адресов).

Help – позволяет вывести краткую справку по командам управления (подкоманда **Commands**), режимам отображения (**Display**), командам управления выполнением программы (**Simulation**).

Reset – установка в нулевое состояние программного счетчика и флагов процессора (подкоманда **Cpu**), фиксация счетчика тактов микропроцессора (**Cycles**), сброс установленных точек прерывания и т.п.

Set – установка точек прерывания (BREAKPOINTS), карты распределения памяти (**Memory-map**), позволяющей выделить программе дополнительное пространство под оперативную (RAM) или постоянную (ROM) память, запуск счетчика тактов микропроцессора (**Cycles**) и т.п.

Memory – очистка области памяти в заданном диапазоне(**Clear**), заполнение области памяти произвольным символом (**Fill**), перемещение данных из одной области памяти в другую (**Move**), поиск символа в заданной области памяти (**Search**).

Execute – непосредственное выполнение команды процессора, введенной в мнемонической ассемблерной форме. При вводе операндов команды и адресов допустимы выражения, использующие следующие операторы:
+, -, @, четыре уровня скобок(), $ - текущее значение программного счетчика PC.
Указатели системы счисления:
Binary: %111 or 111B, 'x' or "x" символьные константы
Octal: @377 or 377Q,
Hex: $FF or FFH,
Система счисления по умолчанию - десятичная
Формат команды: {метка:} мнемоника_операции {операнды} {; коментарий}

Клавиши редактирования

→ курсор направо	Enter ввод команды
← курсор налево	Delete – удаление символа
← возврат на шаг	Insert – вставка пробела
+/- инкремент/декремент byte/word/flag	

Клавиши быстрого доступа

ctl-A	Accumulator
ctl-P	Program Counter
ctl-S	Stack Pointer
ctl-BDH	REGISTER:BC/DE/HL
ctl-C	Carry Flag
alt-ZPSC	FLAGS:Z/P/S/AC
ctl-R	RIM: Int Status/Masks
ctl-IT	PINS: Int/Trap
alt-I	Sid Pin
alt-B	Interrupt Bus Data
alt-AB	Dump Areas ½
alt-1..9	I/O Address 0-9
alt-Y	Cycles Counter
↑	move up one row
↓	move down one row
HOME	first character of window
END	last character of window
PgUp	scroll up one window
PgDn	scroll down one window

Enter	возврат на предыдущую позицию
Ctl-→	перемещение на одно окно направо
Ctl-←	перемещение на одно окно налево
Insert	переключение byte/nibble/bit