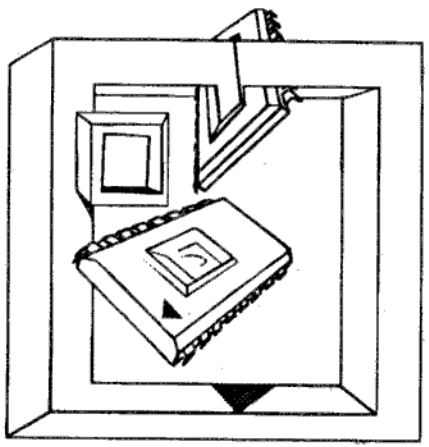
В помощь разработчику
микропроцессорной техники

В.Ф.КОРОЛЕВ

МИКРОПРОЦЕССОР Zilog Z*80



1992
АРГУС—МАСТЕР
НТЦ «ЭЛИС»

ББК 32.973.2
УДК 681.322
К 68

К.т.н. Королев Владимир Фаддеевич
Микропроцессор Zilog Z*80

Издательство «АРГУС»
при участии издательства «МАСТЕР»
по заказу НТЦ «ЭЛИС»

Подписано в печать с готовых диапозитивов 27.08.92. Формат 60×90 1/16. Гарнитура «Тип-таймс». Печать офсетная. Усл. печ. л. 4,5. Тираж 10 000 экз. Заказ 1298.

Издательство «Аргус» при участии издательства «Мастер».

Типография акционерного общества «Молодая гвардия». Адрес АО: 103030, Москва, Сущевская, 21.

К $\frac{2405000000-3}{Г89(03)-92}$ без объявл.

ISBN 5-87896-003-6

ВВЕДЕНИЕ

Появившись в конце 70-х годов как усовершенствованный последователь архитектурной линии Intel 8080, этот микропроцессор стал, пожалуй, наиболее популярным 8-разрядным устройством в мире, достигнув к середине 80-х годов масштабов выпуска в 26 миллионов штук в год при цене менее 1 доллара и захватив до 35% мирового рынка микросхем этого класса. Он с одинаковым успехом применялся (и применяется) как в качестве контроллера в наиболее массовых периферийных устройствах персональных ЭВМ (в первую очередь — матричных принтерах), так и в выпускавшихся громадными тиражами бытовых, учебных и персональных ЭВМ, особенно популярных в Японии.

Для вычислительных машин, построенных на его основе, разработаны объемы системного, инструментального и прикладного программного обспечения, сравнимые с лидером в этой области — семейством машин IBM PC. Имеются несколько дисковых операционных систем, наиболее популярными из которых являются CP/M — стандартная де-факто операционная система для 8-разрядных ЭВМ в мире, и система MSX, совместимая по файловой структуре и форматам записи на диск с MS DOS. За разработку ОС MSX по заказу конгламерата японских компаний ее автор — фирма MICROSOFT была подвергнута санкциям со стороны всемогущей IBM, которая поняла опасность этой системы для своего рынка персональных ЭВМ.

В настоящее время новые разработки персональных ЭВМ на основе этого процессора уже не ведутся, но он продолжает использоваться в контроллерах различных автоматических устройств, промышленной и медицинской электронике, бытовой технике. Кроме того, активно эксплуатируется и продолжает пополняться многомиллионный парк бытовых и учебных ЭВМ на его базе.

В нашей стране по необъяснимым причинам эта линия развития микропроцессорной техники до последнего времени не получала никакого развития, 8-разрядная архитектура продолжает представляться аналогами устройств intel 8080A и 8085, практически исчезнувших с мирового рынка к моменту начала их действительно массового производства в СССР.

Однако большая популярность процессора Z80, который с 1991 г как будто бы начинает выпускаться и отечественной (если иметь в виду бывший СССР) промышленностью, а также вызванная долгожданным для любого грамотного разработчика прорывом зарубежных электронных компонентов на наш внутренний рынок, делает совершенно необходимым знакомство с этим изделием, безусловно, выдающимся для своего времени, и видимо на долгий еще период эталонным для 8-разрядных устройств.

Настоящая книга ставит своей целью познакомить разработч
ков микропроцессорной техники, широкий круг радиолюбителей
программистов с микропроцессором Z80 в объеме, достаточном д
квалифицированной разработки аппаратных средств на его основ
и, кроме того, обеспечить их минимально необходимой, но дост
точной информацией для перехода к программированию этого пр
цессора специалистами и любителями, знакомыми с программ
рованием процессоров 8080, 8085.

Книга в основном написана на основе материалов раздела «Zilo
Z80» из работы Адама Осборна (Adam Osborne) «An Introduction
Microcomputers. Vol. II. Some Real Products» издательства SYBEX, о
нако приведенные в ней данные значительно переработаны, исп
равлены и дополнены материалами автора и более новыми сведе
ниями фирм-изготовителей.

1. МИКРОПРОЦЕССОР Z80.

1.1. ОБЩИЕ СВЕДЕНИЯ.

Микропроцессор Zilog Z80 разрабатывался как усовершенство
ванная модель по отношению к процессору Intel 8080A теми же раз
работчиками, уволившимися из фирмы Intel и образовавшим
фирму Zilog.

Система команд Z80 включает полную систему команд 8080A
как подмножество*. Хуже обстоит дело с физической совместимо
стью — цоколевки процессоров абсолютно различны.

Совместимость ограничена системой команд и фукциональны
ми возможностями. Программы, написанные для 8080A, всегда бу
дут выполняться на Z80, разумеется при условии, что внешние ус
тройства, к которым обращается (если обращается) программа
одинаковы и подключены одинаковым образом.

Логика подключения периферийных микросхем обеих процес
соров, особенно, если иметь в виду систему на 8080A с контрол
лером типа 8228 (КР580ВК28), достаточно близка, поэтому систе
му, построенную для 8080A достаточно легко модифицировать для
использования Z80 вместо него.

1.2. ОСНОВНЫЕ РАЗЛИЧИЯ МЕЖДУ 8080A И Z80.

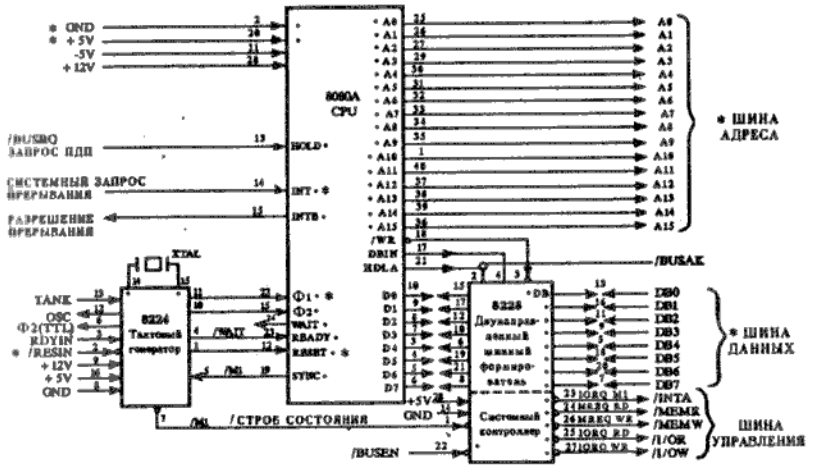
Постараемся подчеркнуть основные различия между Z80 и
8080A, прежде, чем описывать их детально. Если читатель хорошо
знаком с процессором 8080A, есть смысл прочесть этот раздел сра

* Примечание. Полным аналогом микропроцессора 8080A является КР580ВМ80А. В дальнейшем отечественные аналоги зарубежных микросхем будут приводиться в скобках сразу вслед за ее упоминанием в тексте.

ру, если нет — пропустить его и вернуться после знакомства с под
робным описанием Z80, приведенным далее, начиная с раздела 1.3.

Для программиста Z80 предоставляет больше регистров, спосо
бов адресации, а также намного более обширную систему команд.

Важным отличием для разработчика является один источник пи
тания (+5 в), одна системная тактовая последовательность, наличие
дополнительного прерывания и встроенной логики для регенерации
динамической памяти.



Примечание:

/M1 — сигналы Z80, эквивалентные показанным, или новые сигналы.

* — сигналы, которые воспроизведены Z80.

• — сигналы, которые нуждаются в обработке аппаратными средствами замещающей схемы.

Рис. 1. Система 8080A и эквиваленты сигналов Z80.

При использовании 8080A центральное процессорное устройство фактически включает в себя три микросхемы (рис. 1): собственно микропроцессор 8080A (КР580ВМ80А), тактовый генератор 8224 (КР580ГФ24) и контроллер системной шины 8228 (КР580ВК28). Z80 в той или иной степени выполняет функции всех трех из них.

На рис. 1 знаком точки показаны сигналы, которые должно использовать единое устройство, которое могло бы заменить показанную структуру из трех микросхем. Звездочками выделены сигналы, которые использованы в Z80.

Сравнение показывает, что многие (но не все) функции, требуемые от центрального процессорного устройства разработчики Z80 уместили в стандартный 40-выводный корпус.

Очевидно также, что в конфигурации центральных процессорны устройств существует значительное сходство.

Итак, просуммируем аппаратные различия:

- Z80 требует одного источника питания +5в вместо трех;
- Логика тактирования находится внутри Z80;
- Сложная двойная тактовая последовательность 8080A зам няется одним тактовым сигналом;
- Добавлены внутренние логические цепи регенерации дин мической памяти;
- Изменилась структура сигналов управления чтением и з писью во внешние устройства. Система на 8080A обычн использует четыре сигнала управления — чтение памяти запись в память, чтение порта, запись в порт. Z80 выраба тывает общие сигналы чтения и записи, дополненные сиг налами выбора памяти и выбора порта. Это означает, чт при замене 8080A на Z80 потребуется дополнительная логик Ее можно построить так, чтобы сформировать те же сигналы что и у 8080A, либо изменить логику выбора и формировани записи/чтения у каждой периферийной микросхемы. Боле детально способы решения этой проблемы будут описаны ниже;
- Изменилась временная диаграмма перевода шин адресов и данных в третье состояние при операциях прямого доступа в память. 8080A освобождает эти шины в начале третьего или четвертого тактов машинного цикла, в котором приходит требование ПДП, переходя в специальное состояние захвата шины. Z80 действует более прямолинейно, освобождая шины в начале нового машинного цикла. Это сопровождается выдачей сигнала подтверждения захвата;
- В Z80 введен дополнительный запрос прерывания. В дополнение ко входам сброса и прерывания, аналогичных 8080A, Z80 имеет вход немаскируемого прерывания (NMI), которое обычно используется для запуска короткой программы обработки отказа питания при его обнаружении.

Теперь рассмотрим внутреннюю организацию Z80 в терминах совместимости и расширения системы команд.

Как показано в таблице 4, набор команд 8080A является подмножеством набора Z80. К несчастью, мнемонические обозначения команд Z80 полностью изменились, поэтому команды аналогичные 8080A трудно выделить сразу.

В наборе команд 8080A существует очень мало неиспользуемых объектных кодов. Z80 определяет эти неиспользованные коды и принимает следующий за таким кодом байт за второй байт объектного кода своей расширенной системы команд, например:

11011101 — объектный код, не использованный в 8080A
xxxxxxxx — значит, здесь 2-й байт объектного кода команды Z80.

Это приводит к тому, что новые команды Z80 имеют двухбайтовый объектный код, при этом к системе команд может быть добавлено очень большое число новых.

При расширении возможностей 8080A его регистры и флаги состояния должны быть подмножеством новой расширенной структуры для сохранения совместимости.

Вот следующие основные дополнения, сделанные во внутренней архитектуре Z80:

- Стандартные регистры общего назначения (РОН) и флаги дублированы. Это очень упрощает обработку одноуровневых прерываний, поскольку нет необходимости сохранять регистры и аккумулятор в стеке, вместо этого программа просто переключается на альтернативный набор регистров;
- Добавлены два индексных регистра. Это означает, что новые команды Z80 могут использовать индексную адресацию памяти;
- Регистр вектора прерывания позволяет дополнить однобайтовый вектор, вырабатываемый внешней логикой подтверждения прерывания, до двухбайтового вектора, обеспечивающего переход на программу обработки прерывания, размещенную в любом месте памяти;
- Команда пересылки блока позволяет одной командой пере-

Z80		8080A	
команды	такты	команды	такты
LD R,(IX+d)	19	LXI H,d DAD RP MOV R,M	10 10 7 27

LD RP,ADDR	20	LHLD ADDR
		MOV C,L
		MOV B,H
			16
			5
			5
			26

SET B,(HL)	15	MOV A,M
		ORI MASK
		MOV M,A
			7
			7
			7
			21

Таблица 1. Сравнение циклов выполнения команд процессоров Z80 и 8080A.

слать непрерывную последовательность байтов любой длины между одной областью памяти и одним портом ввода-вывода. Можно также осуществить поиск заданного байта внутри блока памяти командой сравнения с блоком.

Хотя при поверхностном взгляде система команд Z80 кажется очень мощной, надо заметить, что наборы команд весьма субъективны, и трудно определить, что на самом деле плохо, а что — хорошо.

Во-первых, преимущество в скорости выполнения, предоставляемое новыми командами Z80 уменьшается из-за того, что они требуют загрузки двух байт объектного кода.

Некоторые примеры команд Z80 и эквивалентных им последовательностей команд 8080A приведены в табл.1.

Во-вторых, вновь программирующий на Z80 может найти его набор команд ужасающе сложным. В то время, как большинство потенциальных пользователей микрокомпьютеров запуганы уже простыми наборами команд на языке ассемблера, возможно, что эти пользователи отрицательно отреагируют на набор команд, чья сложность (если не мощность) оказывается больше, чем у многих миникомпьютеров.

1.3. ПРОГРАММНО — ДОСТУПНЫЕ РЕГИСТРЫ.

С этого раздела начинается подробное описание микропроцессора Z80.

Z80 имеет два набора 8-битных программно-доступных регистров и два слова состояния программы (PSW). Каждый из наборов полностью аналогичен регистрам A — L процессора 8080A. В каждый момент времени лишь один из них и одно слово состояния активны и доступны программам.

Кроме того, Z80 имеет 16-битовый счетчик адреса программы (PC), 16-битовый указатель стека (SP), два 16-битовых индексных регистра IX и IY, 8-битовый счетчик регенерации (R).

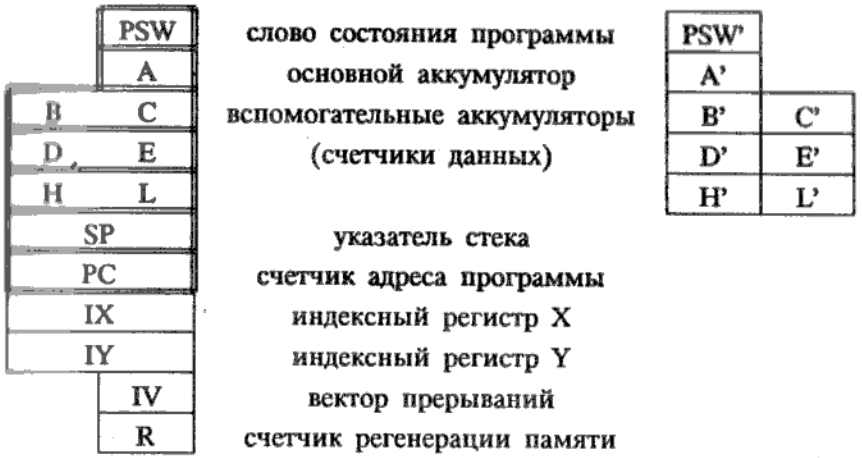
Рис.2 показывает регистры Z80. На этом рисунке подмножество регистров 8080A выделено двойной рамкой.

Z80 использует слово состояния программы, регистры A, B, C, D, E, H, L и указатель стека SP точно так же, как и 8080A, поэтому дополнительного обсуждения этих регистров не требуется.

Слово состояния программы и регистры A, B, C, D, E, H, L дублированы.

Одной командой Z80 можно перейти с одного набора на другой, или обменять содержание выбранных регистров. В каждый момент времени доступен один или другой набор регистров, но не оба.

Имеются два 16-битных индексных регистра, обозначаемых IX и IY. Более корректно рассматривать их как регистры базового адреса, что станет понятнее при рассмотрении способов адресации Z80 ниже.



В двойной рамке — подмножество регистров 8080A

Рис. 2. Структура регистров Z80.

Регистр вектора прерываний IV исполняет роль, подобную байту 2 команды прерывания, выдаваемой программируемым контроллером прерываний 8259 (КР580ВН59). Логика подтверждения прерывания Z80 обеспечивает дополнительную возможность запуска подпрограммы обслуживания прерывания по команде вызова, старший байт адреса которого берется из этого регистра.

Счетчик регенерации памяти осуществляет функцию микрокомпьютерных систем, которую просмотрели все разработчики 8-разрядных микропроцессоров, кроме фирм Fairchild и Zilog. Динамическая память не может сохранять свое содержимое слишком долго вне зависимости от того, подано ли на нее питание. Для сохранения данных необходимо осуществлять доступ в нее с интервалом в миллисекунды. Платой за этот недостаток является ее дешевизна, как и простота устройств, необходимых для выполнения процесса регенерации.

Почти единственным логическим устройством, обеспечивающим процесс динамической регенерации является счетчик, перебирающий адреса динамической памяти — в этом и состоит назначение счетчика регенерации памяти R. Z80 вырабатывает также специальный управляющий сигнал доступа в память для регенерации, предоставляя таким образом все средства, необходимые для этой цели.

1.4. СПОСОБЫ АДРЕСАЦИИ.

Команды Z80 используют все способы адресации 8080A и, кроме того, имеют следующие два дополнительных:

— Ряд команд обращения к памяти используют регистры IX и IY для индексной, или относительной адресации к базовому адресу;

— Имеются несколько команд перехода по относительному адресу.

Команды обращения к памяти, которые используют регистры IX и IY, содержат однобайтовое значение смещения. Это 8-разрядное значение, заключенное в объектном коде команды, складывается с 16-разрядным значением из указанного индексного регистра, образуя исполнительный адрес ячейки памяти:

содержимое IX или IY	+	байт смещения из объектного кода команды (двоичное число со знаком)	=	исполнительный адрес

Эта процедура является стандартной для микрокомпьютеров с индексной адресацией.

Двухбайтовая команда перехода по относительному адресу также имеет обычный вид для команд такого рода.

8-битное значение смещения содержится в виде байта в объектном коде команды, это значение складывается с содержимым счетчика адреса программ как двоичное целое со знаком — после того, как последний инкрементируется — чтобы показать на адрес следующей команды программы:

содержимое PC на момент окончания команды	+	байт смещения из объектного кода команды (двоичное число со знаком)	=	адрес перехода

Код следующей команды будет загружен с адреса памяти, равного pppq + 2 + DD, где p, q и D — любые шестнадцатеричные цифры, DD рассматривается как 8-битное двоичное целое со знаком.

Усовершенствование способов адресации в Z80 имеет существенное значение в сравнении с 8080A. Наличие регистров IX и IY позволяет опытному программисту использовать пространство регистров процессора более эффективно. Регистры IX и IY можно рассматривать как средство адресации памяти, в качестве которого в 8080A приходится использовать пары HL и DE. За счет освобождения регистровых пар для манипуляций с данными можно значительно уменьшить количество команд обращений к памяти, выполняемых Z80.

Двухбайтовая команда перехода по относительному адресу полезна, поскольку в большинстве программ до 80% команд перехода осуществляются в пределах 128 байт вперед или назад. Экономия времени выполнения и уменьшение длины программы по сравнению с использованием трехбайтовых команд переходов по абсолютному адресу в ряде случаев могут оказаться весьма существенными.

1.5. ФЛАГИ СОСТОЯНИЯ.

Z80, как и 8080A, использует слово состояния программы PSW для хранения флагов состояния. Эти флаги и их обозначения для Z80 следующие:

Перенос	CARRY	C
Нуль	ZERO	Z
Знак	SIGN	S
Четность/переполнение	PARITY/OVERFLOW	P/O
Дополнительный перенос	AUXILIARY CARRY	Ac
Разность	SUBSTRACT	N

Статусные флаги записаны в PSW в Z80 в следующем порядке (в сравнении с 8080A):

7	6	5	4	3	2	1	0	номер бита
S	Z	X	Ac	X	P/O	N	C	PSW Z80
S	Z	X	Ac	X	P	X	C	PSW 8080A

Здесь X — неиспользованные биты PSW.

Флаги четности/переполнения и разности отличаются от 8080A. Остальные в точности такие же, как у него.

8080A имеет флаг четности, но не переполнения. Z80 использует один и тот же флаг для обеих операций. Флаг переполнения Z80 абсолютно стандартен, и имеет значение лишь при выполнении двоичной арифметики со знаком, т.е. когда значение четности бессмысленно.

В Z80 этот флаг используется арифметическими командами как флаг переполнения, а остальными — как флаг четности.

Флаг разности используется командой DAA (десятичная настройка аккумулятора) для операций с двоично-десятичным кодом, чтобы отличить десятичное сложение от вычитания. Флаги разности и дополнительного переноса не могут использоваться командами условных переходов, вызовов подпрограмм и возвратов.

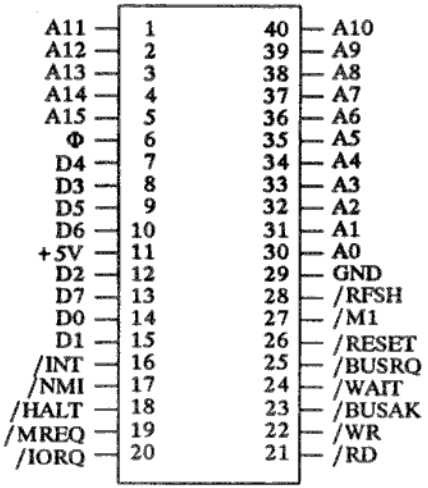
1.6. ВЫВОДЫ И СИГНАЛЫ МИКРОПРОЦЕССОРА.

Расположение выводов и соответствующие им сигналы Z80 показаны на рис.3.

Рисунок 1 показывает в непосредственном сравнении Z80 и стандартную для 8080A трехкристальную систему (8080A, 8224, 8228).

Рассмотрем вначале шины адреса и данных.

По 16 адресным линиям A0-A15 выдаются адреса памяти и портов ввода/вывода. Адресные линии имеют три состояния, и могут



Вывод	Описание	Тип
A0–A15	Шина адресов	Выход с 3-мя сост.
D0–D7	Шина данных	Двунапр. с 3-мя сост.
/M1	Определяет цикл загрузки команды	Выход
/MREQ	Запрос памяти. Определяет цикл доступа в память	Выход
/IORQ	Запрос порта ввода/вывода, определяет цикл обращения к порту ввода/вывода	Выход
/RD	Процессор читает данные из памяти или порта	Выход с 3-мя сост.
/WR	Процессор пишет данные в память или порт	Выход с 3-мя сост.
/RFSH	Регенерация динамической памяти	Выход
/HALT	Выполнен останов процессора	Выход
/WAIT	Запрос состояния ожидания	Вход
/INT	Запрос прерывания	Вход
/NMI	Запрос немаскируемого прерывания	Вход
/RESET	Сброс и инициализация процессора	Вход
/BUSRQ	Запрос на захват управления шинами адреса, данных и управления	Вход
/BUSAK	Подтверждение захвата шин	Выход
ф	Тактовая частота	Вход
+5V,GND	Питание и земля	Вход

Наименования сигналов, имеющих активный низкий уровень, начинаются со знака "/".

Рис.3 Сигналы и назначение выводов процессора Z80.

переходить в плавающее состояние по сигналу процессора, передавая управление адресной шиной внешней логике. Между адресными шинами 8080A и Z80 нет никакой разницы.

По линиям шины данных D0 — D7 двунаправленно передаются данные в Z80 или из него. Как и у адресной шины, линии шины данных имеют три состояния. Шина данных Z80 отличается от шины процессора 8080A. У последнего она мультиплексирована — по ней, кроме данных, выдается байт состояния процессора на такте T2 каждого машинного цикла. Байт стробируется сигналом SYNC. Z80 не мультиплексирует шину данных подобным образом.

Перейдем теперь к сигналам управления. Они подразделяются на системные сигналы, сигналы управления процессора и сигналы управления шинами. Рассмотрим сначала системные сигналы управления:

— /M1 идентифицирует машинный цикл загрузки кода команды. Его функция подобна, но не идентична импульсу SYNC процессора 8080A.

— /MREQ сопровождает любую операцию доступа в память, линия имеет третье состояние.

— /IORQ сопровождает любую операцию обращения к портам ввода-вывода. Когда IORQ имеет низкий уровень, на линии AO-A7 выставлен адрес порта ввода/вывода. Этот сигнал используется также для подтверждения прерывания, и в этом (и только в этом) случае выдается совместно с M1. В остальных ситуациях M1, естественно, выдается только в случае чтения из памяти, где находится объектный код программы.

— /RD — это сигнал с тремя состояниями, который показывает, что процессор хочет прочитать данные из памяти или порта ввода-вывода, выбор же между ними определяется по состояниям /MREQ и /IORQ.

— /WR также имеет три состояния и показывает, что процессор хочет записать данные. Выбор между памятью и портами ввода определяется, как и у предыдущего сигнала.

— /RFSH — сигнал управления, используемый для регенерации динамической памяти. Когда он имеет низкий уровень, текущий сигнал MREQ означает обращение к памяти с целью регенерации, и текущий адрес регенерации выдается по линиям AO-A6 шины адреса.

Теперь рассмотрим сигналы управления процессора:

— /HALT становится низким после выполнения команды останова процессора HALT. Процессор при этом переходит в состояние останова, в котором беспрерывно повторяет выполнение пустой команды NOP, чтобы обеспечить возможность регенерации памяти. Вывести процессор из этого состояния можно лишь прерыванием или сбросом.

— /WAIT эквивалентен входу READY микропроцессора 8080A. Внешняя логика, которая не успевает выполнить запрос процессора за нормальное время выполнения машинного цикла может расширить отведенный ей временной интервал, установив низкий уровень на этом входе. В ответ на это процессор переходит в состояние ожидания, занимающее целое число следующих непрерывно периодов тактовой частоты.

— /INT, /NMI — два входа запросов прерываний. Разница между ними в том, что NMI имеет больший приоритет и не может быть запрещено.

— /RESET — это обычный сигнал управления сбросом. Когда Z80 сброшен, он делает следующее:

а) счетчик адреса программ PC, регистры IX и IY сбрасываются в нуль;

б) запросы прерывания через INT запрещаются;

в) все линии с тремя состояниями освобождаются.

— /BUSRQ и /BUSAK — сигналы запроса и подтверждения захвата шины. Для выполнения любой операции прямого доступа в память внешняя логика должна перехватить управление системной шиной. Это осуществлется выдачей низкого уровня на вход /BUSRQ. По завершении каждого машинного цикла процессор освободит все тристабильные линии и подтвердит запрос шины установкой в низкий уровень выхода /BUSAK.

1.7. СООТВЕТСТВИЕ СИГНАЛОВ Z80 и 8080A.

Если Вы разрабатываете новое изделие на микропроцессоре Z80, тогда вопросы совместимости сигналов Z80 и 8080A для Вас не важны.

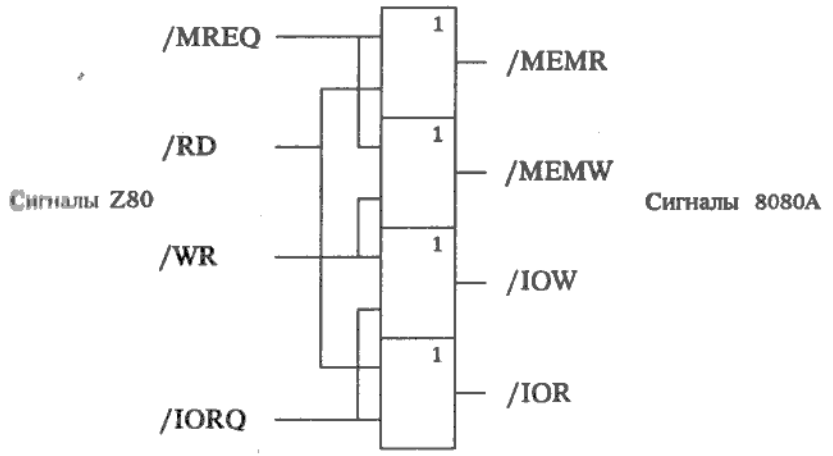
Если же Вы заменяете 8080A и Z80 в уже готовой системе, полезной бы оказалась некоторая таблица соответствия сигналов этих процессоров. К сожалению, такую таблицу составить нельзя. Как мы уже говорили, Z80, полностью заменяя 8080A, содержит и ряд элементов, реализованных в микросхемах 8224 (тактовый генератор) и 8228 (контроллер системной шины).

Выделим основные направления в решении проблемы совместимости.

Возможно, наиболее важная концептуальная разница между Z80 и 8080A заключается в сигналах управления чтением и записью. Системный контроллер 8228 вырабатывает четыре различных сигнала управления для чтения в память, записи в память, чтения порта, записи в порт (/MEMR, /MEMW, /IOR, /IOW соответственно, см рис.1).

Z80 имеет общие сигналы чтения и записи, а также сигналы выбора памяти и выбора портов ввода/вывода.

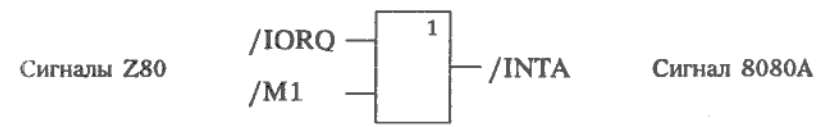
Добавив логику, достаточно легко имитировать четыре сигнала 8228 из сигналов Z80. Вот одна элементарная возможность:



Если Ваш проект позволяет, будет гораздо рациональней распространить принцип организации управления записью/чтением, принятый в Z80 на все устройства, окружающие процессор. Каждое из таких устройств требует отдельных логических сигналов выбора микросхемы (или устройства) и строба доступа в устройство для записи или чтения, например:

В схемотехнике систем на 8080A логический сигнал выбора устройства формируется декодированием шины адреса, а в качестве стробов прямо или косвенно используются четыре вышеупомянутых сигнала /MEMR — /IOW. При построении схемы по принципам, принятым в Z80, сигнал выбора формируется из сигналов шины плюс сигналы /MREQ и /IORQ, а стробы — из сигналов /WR и /RD.

Z80 не имеет сигнала подтверждения прерывания. Для его получения можно скомбинировать сигналы /IORQ и /M1 следующим образом:



Надо отметить, что данное решение может быть отнесено лишь к устройствам, вырабатывающим однобайтную команду подтверждения прерывания или однобайтный вектор, например, 8214 (К589ИК14). Для устройств, выводящих трехбайтовую команду CALL, таких как 8259 (КР580ВН59) необходима более сложная схема, которая будет прокомментирована ниже.

Сигналы HOLD и HLDA процессора 8080A функционально замещаются сигналами /BUSRQ и /BUSAK.

Сигнал SYNC процессора 8080A не имеет прямого эквивалента в Z80. Сигнал /M1 последнего принимает низкий уровень в течении циклов загрузки команды и подтверждения прерывания, но не вырабатывается во втором и последующих машинных циклах выполнения команды. Часто инверсия /M1 может быть использована взамен сигнала SYNC для управления устройствами семейства 8080A, которые требуют этого сигнала.

У Z80 нет сигналов, эквивалентных INTE, WAIT и Ф2 процессора 8080A, а также сигнала, эквивалентного BUSEN контроллера 8228.

Если по каким-либо причинам внешняя логика должна знать, когда прерывания запрещаются внутри процессора, применение Z80 невозможно. С другой стороны, хотя 8080A и сообщает внешней логике о том, когда прерывания запрещены, сигналом INTE, эта логика ничего не может поделать с фактом запрещения прерывания, поэтому польза от наличия такого сигнала сомнительна. Другое дело, что наличие у процессора независимого вывода, состояние которого может управляться программно, не оказывая влияния на другие компоненты системы, и не требующее для реализации этого управления никаких внешних устройств, бывает весьма полезно при написании программ диагностики отказов аппаратных средств. С этой точки зрения отсутствие такого вывода у Z80 является заметной потерей.

Вход /WAIT процессора Z80 выполняет ту же функцию, что и вход READY у 8080A. Независимо от того, когда состояние ожидания запрошено, такты ожидания вставляются между тактами T2 и T3 машинного цикла.

Более того, как будет видно вскоре, некоторые команды искусственно вставляют такт ожидания между этими тактами не ожидая внешнего запроса.

Для того, чтобы выработать эквивалент выходного сигнала WAIT процессора 8080A, в системе на Z80 потребуется довольно сложная схема, декодирующая объектный код команды, тактовый сигнал и состояние входа WAIT. По всей вероятности, будет проще разработать альтернативную схему, не требующую наличия сигнала, эквивалентного выходу WAIT процессора 8080A.

Z80 просто не имеет второго тактового сигнала, аналогичного такту Ф2 процессора 8080A, и любое устройство, для которого нужен этот сигнал, не может работать в системе на Z80.

Вход BUSEN контроллера 8228 используется внешней логикой чтобы освободить системную шину. В системе на Z80 сам процессор освобождает системную шину, а при наличии на шине буферных схем эквивалентный сигнал подается на эти буфера.

С другой стороны, 8080A не имеет сигналов /RFSH, /HALT и /NMI.

/RFSH относится лишь к применению динамической памяти, и вне этого контекста его сравнение с сигналами 8080A бессмысленно.

/NMI, будучи немаскируемым прерыванием, также просто не имеет сопоставимого эквивалента в 8080A.

Сигнал /HALT процессора Z80 требует некоторого обсуждения. Один из наиболее обескураживающих аспектов в применении 8080A состоит во взаимодействии состояний ожидания, останова и захвата шины. Давайте рассмотрим эти состояния, сравнивая конфигурации Z80 и 8080A и из этого рассмотрения будет видно назначение выхода /HALT процессора Z80.

Назначением состояния ожидания является удлинение машинного цикла обращения к памяти или порту ввода-вывода при использовании медленных внешних устройств. Состояние ожидания включает один или более тактов ожидания, включаемых между тактами T2 и T3 машинного цикла. Формирование тактов ожидания в Z80 и 8080A в точности одинаково кроме отсутствия у Z80 сигнала подтверждения ожидания и того факта, что при некоторых обстоятельствах он вставляет такты ожидания автоматически, без внешнего запроса.

Условия для состояния захвата шины (HOLD) существуют, когда внешняя логика затребовала управление системной шиной, чтобы выполнить процедуру прямого доступа в память. Опять же, Z80 и 8080A имеют очень похожие состояния захвата. Единственным существенным различием является то, что Z80 входит в состояние захвата по окончании машинного цикла, в то время, как 8080A входит в это состояние на тактах T3 или T4.

Контроллеру системной шины 8228 также необходим высокий уровень на входе /BUSEN, чтобы перевести шины данных и управления в третье состояние. В системе на Z80 такой необходимости нет.

Большая разница между Z80 и 8080A возникает в состоянии останова (HALT). Когда 8080A выполняет команду останова, он переходит в состояние останова, которое отличается от состояния захвата. В системе на 8080A существует ряд сложных взаимовлияний между состояниями захвата, останова, ожидания и прерываниями. Этих затруднений нет в системах на Z80, поскольку последний не имеет состояния останова.

После выполнения команды HALT Z80 переводит выход HALT в состояние низкого уровня, а затем продолжает непрерывно вы-

полнять команду NOP. Это позволяет логике регенерации памят продолжать функционировать.

Если Вы заменяете процессор 8080A на Z80 в конкретной си стеме, необходимо обратить особое внимание на состояние ост нова. Это единственная область, где могут возникнуть неожиданны эффекты несовместимости.

1.8. ТАКТИРОВАНИЕ И ВЫПОЛНЕНИЕ КОМАНД.

Временные диаграммы работы Z80 концептуально похожи, н проще, чем у 8080A. Как и в последнем, выполнение команд Z8 подразделяется на машинные циклы и такты.

Однако, все машинные циклы Z80 состоят из трех или четыре тактов. Некоторые команды всегда включают такты ожидания, и этом случае машинный цикл может занимать три, четыре или пят тактов.

8080A может потребовать от одного до пяти машинных циклов чтобы выполнить команду. Команды Z80 включают от одного д шести машинных циклов. Если выделить необязательные машин ные циклы и такты команды Z80 и 8080A, их обобщенные вре менные диаграммы могут быть представлены в сравнении как по казано ниже:

Цикл выполнения команды процессором 8080A

MC1					mc2					mc3					mc4					mc5				
T1	T2	T3	T4	t5	T1	T2	T3	t4	t5	T1	T2	T3	t4	t5	T1	T2	T3	t4	t5	T1	T2	T3	t4	t5

Цикл выполнения команды процессором Z80

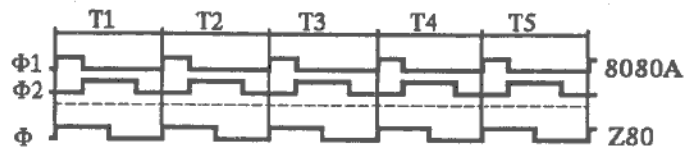
MC1						mc2						mc3						mc4						mc5						mc6			
T1	T2	tw	tw	T3	T4	T1	T2	tw	T3	t4		T1	T2	tw	T3	t4		T1	T2	tw	T3	T4		T1	T2	tw	T3	t4		T1	T2	T3	t4

Обязательные машинные циклы и обязательные такты в машин ных циклах обозначены заглавными буквами.

У процессора Z80 такты tw в цикле MC1 включаются только при подтверждении прерывания, а в остальных циклах — при обраще нии к портам ввода/вывода.

Тактирование Z80 также гораздо проще, чем 8080A. В то время как 8080A имеет два тактовых сигнала, Z80 использует один.

Временные диаграммы тактовых сигналов показаны в сравнении ниже:



1.9. МАШИННЫЙ ЦИКЛ ЗАГРУЗКИ КОДА КОМАНДЫ.

В сравнении с 8080A, машинный цикл загрузки кода команды Z80 значительно проще. Отсутствует импульс SYNC и выдача слова состояния на шину данных. Вся временная диаграмма выполнения команды упрощается до цикла загрузки ее кода, за которым могут следовать циклы доступа в память или порты ввода-вывода. Добавим сюда возможные такты ожидания, подтверждение прерывания, захват шины — и это все возможные варианты.

Начнем с рассмотрения машинного цикла загрузки команды. Временная диаграмма этого цикла показана на рис.4.

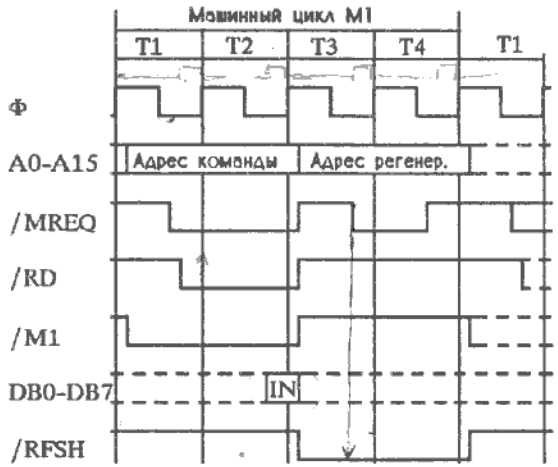


Рис. 4. Цикл загрузки кода команды Z80.

Из рис.4 следует, что цикл загрузки кода команды идентифицируется установкой выхода M1 в состояние низкого уровня в течение тактов T1 и T2. Поскольку на шину данных не выводится слово состояния, содержимое счетчика адреса команд выдается на шину адреса в начале цикла и остается стабильным в течение T1 и T2.

Физически загрузка кода команды — это цикл чтения из памяти. Поэтому оба управляющих сигнала /RD и /MREQ переводятся в состояние низкого уровня. Это происходит на середине T1, когда состояние адресной шины уже стабилизировалось. Таким образом, задающие фронты /MREQ и /RD могут быть использованы для выбора устройств памяти и стробирования выдачи ими данных. Процессор считывает данные с их шины по переднему фронту тактовой частоты на такте T3.

Такты T3 и T4 цикла загрузки команды используются процессором для внутренних операций. Эти такты также используются для

регенерации памяти. Как только содержимое счетчика адреса
манд удаляется с шины адреса, на линии A0-A6 этой шины
дается содержимое счетчика регистра регенерации. Этот адрес
тается на адресной шине до окончания T4.

Поскольку регенерация является операцией доступа в памя
сигнал /MREQ опять переводится в состояние низкого уровня,
нако вместо сигнала /RD он сопровождается сигналом /RFSH.
ким образом схема управления памятью предупреждается от
пытки чтения памяти в цикле регенерации.

1.10. ЦИКЛ ЧТЕНИЯ ДАННЫХ ИЗ ПАМЯТИ.

Схема интерфейса памяти реагирует на цикл загрузки кода
манды и на цикл чтения данных из памяти совершенно одинако
Однако, между этими циклами есть небольшая разница. Времен
диаграмма чтения данных из памяти приведена на рис.5.

Рис. 5. Цикл чтения данных из памяти Z80.

Принципиальная разница с циклом загрузки кода команды,
торую надо отметить, заключается в том, что в цикле чтения данн
они читаются процессором с шины по заднему фронту тактов
частоты на такте T3, т.е. в середине T3, в то время, как в цик
загрузки — по переднему фронту тактовой частоты в том же так
Кроме того, обычный цикл чтения данных состоит из трех, а
четырех тактов. Вспомним также, что цикл загрузки кода коман
сопровождается управляющим сигналом M1 в течение первых дв
тактов.

1.11. ЦИКЛ ЗАПИСИ ДАННЫХ В ПАМЯТЬ.

Этот цикл представлен на рис.6. Разница между временными д
аграммами записи и чтения очевидна: в цикле записи состоя
низкого уровня вместо сигнала /RD принимает сигнал /WR. Э

сигнал схемой интерфейса памяти может использоваться в качестве
ала записи данных с их шины в устройство памяти.

Рис. 6. Цикл записи данных в память Z80.

Как и 8080A, процессор Z80 позволяет установить состояние
идания (WAIT) между тактами T2 и T3 машинного цикла. Со-
ояние ожидания освобождает внешние схемы от необходимости
ботать со скоростью процессора.

Процессор проверяет состояние входа /WAIT по заднему фронту
актовой частоты на такте T2. Если в этот момент времени на ука-
анном входе низкий уровень, он немедленно переходит в состояние
идания. Число тактов ожидания зависит от того, насколько долго
игнал /WAIT будет находиться в состоянии низкого уровня. Как
олько процессор обнаружит высокое состояние этого сигнала по
ледующему заднему фронту тактовой частоты, он начнет такт T3
ее следующего переднего фронта.

Заметим, что единственный сигнал /WAIT у Z80 заменяет сиг-
алы READY и WAIT процессора 8080A. В результате не существует
игнала, сообщающего внешним устройствам о том, что процессор
аходится в состоянии ожидания.

В случае, когда внешней схеме необходимо знать о том, нахо-
ится ли процессор в состоянии ожидания, приведем список пра-
ил, следуя которым можно установить это:

- Процессор опрашивает сигнал /WAIT по заднему фронту тактовой частоты на такте T2;
- Если сигнал /WAIT имеет низкий уровень, процессор будет его опрашивать по каждому следующему заднему фронту тактовой частоты.
- Процессор не опрашивает вход /WAIT ни на каком периоде тактовой частоты, кроме тактов T2 и тактов ожидания.

Важно также, что на протяжении состояния ожидания процесс
регенерации памяти приостанавливается. Поэтому использование

Рис. 7. Такты ожидания в цикле чтения Z80.

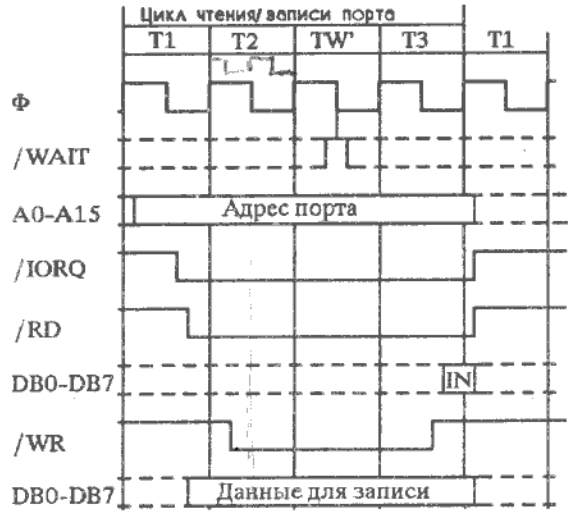
этого состояния, например, для организации пошагового выпо нения программ с целью отладки в системах с динамической п мятью вызывает затруднения. Вообще, при использовании дин мической памяти и наличии схем, использующих принудительн состояние ожидания, необходим анализ процесса регенерации д определения соответствия его периода техническим условиям микросхемы динамической памяти. Напомним, что одно обращ ние к памяти с целью регенерации производится при выполнени каждой команды, цикл регенерации требует 128 таких обращени максимальная длительность команды Z80 без дополнительных та тов ожидания составляет 23 такта, а период регенерации некоторы микросхем динамических ОЗУ не должен быть более 1 мсек.

Организация регенерации в Z80 имеет еще и тот недостаток, ч процесс производится перебором 128 адресов. Такая регенерац обеспечивает работу с микросхемами ОЗУ, имеющими организ цию 16К слов, в то время, как многие микросхемы, имеющие объе 64К слов и более, в т.ч. отечественные 565РУ5, требуют регенерац по 256 адресам. Временная диаграмма состояния ожидания пр ведена на рис.7.

1.13. ЦИКЛЫ ВВОДА — ВЫВОДА.

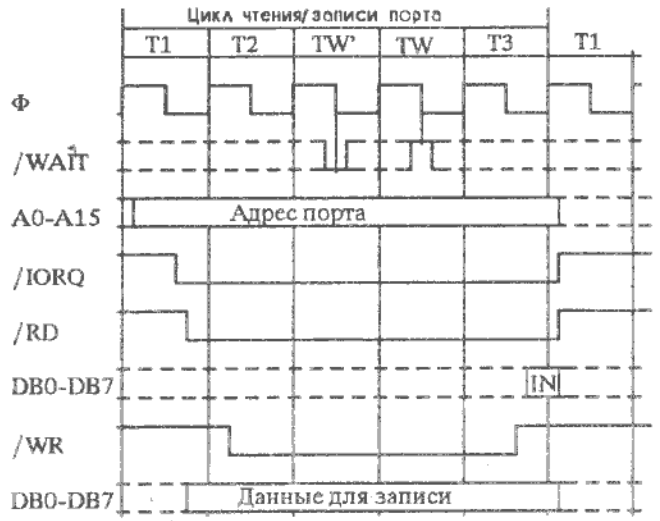
Временные диаграммы Z80 при чтении и записи в порты вв да/вывода показаны на рис.8 и рис.9 соответственно.

Важным обстоятельством является то, что фирма Zilog учла, ч большинство периферийных кристаллов, разработанных в одно врем с процессором Z80 не могло функционировать со скоростью проце сора. Поэтому один такт ожидания автоматически вводится между Т и Т3 в операциях обращения к портам. Это обстоятельство весьма бл гоприятно для разработчика, поскольку хотя быстродействие перифе рийных кристаллов и несколько выросло за время, прошедшее с р



TW' — такт ожидания, вставленный процессором независимо от состояния сигнала /WAIT

Рис. 8. Циклы обращения к портам ввода/вывода



TW' — такт ожидания, вставленный процессором независимо от состояния сигнала /WAIT

Рис. 9. Циклы обращения к портам с дополнительным тактом ожидания

работки процессора, но соответственно (и даже более) выросла и максимально допустимая тактовая частота работы Z80.

В остальном временные диаграммы отличаются от циклов чтения/записи данных в память лишь переводом в низкий уровень сигнала /IORQ вместо /MREQ.

Заметим, что абсолютно ничто не препятствует адресоваться к устройствам ввода/вывода в адресном пространстве памяти. Но если выбрана такая стратегия, помните, что устройства ввода/вывода должны успевать работать со скоростью процессора, иначе вводите внешние состояния ожидания.

1.14. ЦИКЛ ЗАХВАТА ШИНЫ.

Логика операции захвата шины у 8080A и Z80 эквивалентна. Z80 освобождает шины адреса, данных и тристабильные линии шины управления, когда опознает низкий уровень на входе /BUSRQ. Этот вход опрашивается процессором на переднем фронте тактовой частоты в начале последнего такта каждого машинного цикла. Если обнаруживается низкий уровень сигнала /BUSRQ, начиная со следующего такта процессор освобождает системную шину и выдает сигнал подтверждения /BUSAK низким уровнем.

Далее по каждому переднему фронту тактовой частоты продолжает проверяться состояние входа /BUSRQ, и как только будет обнаружен его высокий уровень, низкий уровень /BUSAK снимается, и со следующего такта начинает выполняться новый машинный цикл.

Временная диаграмма захвата и освобождения шины показана на рис. 10.

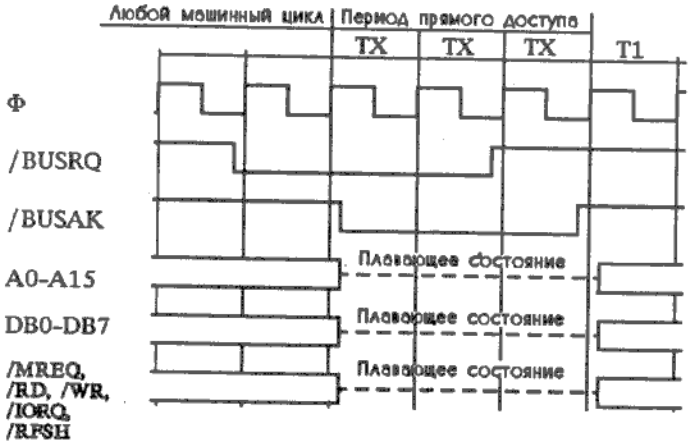


Рис.10. Захват и освобождение шины Z80.

Между процедурами захвата шины у 8080A и Z80 существует одно заметное различие.

В то время, как Z80 освобождает шину между машинными циклами, 8080A делает это на тактах T3 или T4 в зависимости от конкретной выполняемой команды. Следовательно, возможны ситуации, когда Z80 освободит шину на три такта позже, чем 8080A в тех же условиях. В то же время максимальное время ожидания захвата шины у Z80 меньше на один такт.

Заметим также, что при захвате шины, как и в состоянии ожидания, процесс регенерации памяти прерывается, поэтому в этой части справедливы те же замечания, что и для состояния ожидания.

Для нормального функционирования регенерации выгоднее использовать захваты шины из нескольких коротких процедур, чем одну длинную.

1.15. ВНЕШНИЕ ПРЕРЫВАНИЯ.

В отличие от 8080A, процессор Z80 имеет два входа запросов прерывания, один из которых не может быть запрещен.

Временная диаграмма подтверждения запроса прерывания нижнего уровня (маскируемого прерывания) существенно изменилась по сравнению с 8080A, и показана на рис. 11.

Сигнал запроса прерывания INT опрашивается процессором на переднем фронте тактовой частоты в последнем такте каждого цикла выполнения команды (а не машинного цикла).

Запрос прерывания будет игнорирован, если маскируемое прерывание запрещено программой, или сигнал BUSRQ имеет низкий уровень. Таким образом, запрос шины имеет более высокий приоритет, чем маскируемое прерывание.

Если сигнал /INT имеет низкий уровень, и маскирующие его условия не выполнены, процессор подтверждает прерывание, выставив низкий уровень на выходы /M1 и /IORQ. Это происходит в специальном машинном цикле подтверждения прерывания, как показано на рис.11.

Заметим, что машинный цикл имеет два вставленных автоматически такта ожидания, так что даже медленная логика, построенная, например, по схеме приоритетной дейзи-цепочки будет иметь время, необходимое для срабатывания.

Когда сигнал IORQ при низком уровне сигнала M1 такте принимает низкий уровень, это сочетание должно интерпретироваться внешней схемой подтверждения прерывания как требование поместить вектор прерывания на шину данных.

Вектор прерывания может иметь одну из трех форм, в зависимости от трех режимов обработки прерывания, задаваемых программно.

В режиме 0 вектор прерывания воспринимается как однобайт-

ный объектный код команды, которая должна быть выполнена после цикла подтверждения прерывания. Это эквивалентно стандартному ответу командой RST, используемому процессором 8080A. Когда Вы заменяете 8080A на Z80, следовательно, надо использовать режим 0.

При использовании внешних контроллеров прерываний, формирующих однобайтовую команду RST, с подтверждением прерывания не возникает никаких проблем. Иначе обстоит дело при использовании контроллера прерываний 8259, вырабатывающего трехбайтовый код команды CALL.

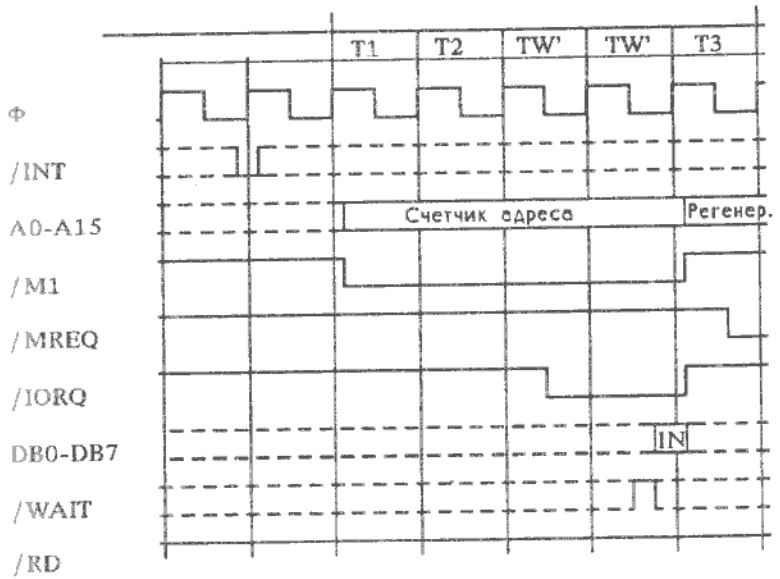
Для работы последнего необходима передача ему трех импульсов подтверждения прерывания /INTA, по одному в каждом из трех машинных циклов выполнения команды. Во временной диаграмме каждого цикла эти импульсы должны замещать стробы /MEMR, /IOR. Вырабатывает сигналы /INTA системный контроллер 8228 (8238).

Процессор Z80 не реализует данную функцию, поэтому для подключения микросхемы 8259 необходима достаточно сложная внешняя логика. Приведем основные сведения, необходимые для ее разработки:

- При выполнении команды CALL, инициированной в цикле подтверждения прерывания, два заключительных машинных цикла этой команды являются циклами чтения из памяти. Следующий за ними машинный цикл всегда является циклом загрузки кода команды. Таким образом, цикл выполнения команды подтверждения прерывания CALL может быть выделен RS-триггером, взводимым по совпадению низких уровней сигналов /M1 и /IORQ и сбрасываемым по совпадению низких уровней /M1 и /MREQ;
- В машинном цикле подтверждения прерывания сигнал /RD имеет высокий уровень, в двух последующих машинных циклах выполнения команды CALL он принимает низкий уровень, разрешая внешним устройствам выход на шину процессора. Необходимо принять меры, чтобы в процессе выполнения трехбайтовой команды подтверждения прерывания был запрещен выход на шину данных процессора для всех устройств, кроме 8259;
- Сигнал /INTA необходимо сформировать как из сигнала /IORQ в первом машинном цикле команды подтверждения прерывания, так и из сигнала /MREQ (или /RD) в двух последующих машинных циклах. При этом, поскольку последние являются циклами обращения к памяти, не имеющими вставленных процессором тактов ожидания, скорость реакции контроллера в них должна быть достаточно высокой.

Суммируя возможности применения контроллера 8259 в систе-

мах на базе Z80, можно сказать, что архитектурные концепции этих устройств отличаются достаточно сильно, и применение данного контроллера хотя и возможно, но не является оптимальным вариантом. В то же время, в отечественных условиях оно достаточно оправдано, поскольку распространенность периферийных микросхем, специализированных для процессора Z80 и имеющих оптимальную для него систему прерываний на основе дейзи — цепочки, остается весьма узкой.



TW' — такты ожидания, вводимые процессором независимо от состояния сигнала /WAIT.

Рис. 11. Подтверждение запроса маскируемого прерывания.

Логика подтверждения прерывания в режиме 1 считает, что следующей после подтверждения прерывания командой будет команда рестарта на адрес 0056H. Если используется режим 1, вектор прерывания не нужен вообще. Это похоже на использование процессора 8080A с контроллером 8228 без контроллера прерывания, когда команду рестарта (но на другую ячейку!) выдает сам контроллер.

Режим 2 подтверждения прерывания не имеет аналога в системах на 8080A. В режиме 2 Вы можете построить таблицу 16-разрядных векторов адресов обработки прерываний, примерно как в системах с использоваем программируемого контроллера прерываний 8259 (КР580ВН59), но, разумеется, без использования этого устройства.

Адреса обработки прерываний могут указывать на любое место

в памяти. Эти 16-разрядные адреса определяют первую команду в подпрограммах обработки прерываний. Когда прерывание подтверждается в режиме 2, внешняя схема должна поместить на шину данных вектор подтверждения прерывания. Процессор объединит содержимое регистра IV с вектором подтверждения прерывания, образовав 16-битный адрес, который укажет на соответствующий вектор адреса в таблице адресов обработки прерываний. Поскольку 16-битовые адреса могут лежать в памяти лишь в словах с четными адресами, только семь из восьми бит вектора подтверждения прерывания используются процессором для формирования адреса в таблице, младший бит адреса устанавливается в 0. Таким образом, процедура доступа в таблицу получается следующей:

Содержимое IV (как старший байт)	+	Вектор с шины данных (как младший байт с нулевым младшим битом)	=	Двухбайтовый адрес в таблице адресов обработки прерываний

Процессор Z80 выполнит команду вызова подпрограммы (CALL) по адресу, выбранному из таблицы адресов обработки прерываний.

Приведем простой пример. Пусть имеется 64 возможных внешних прерывания, и каждое имеет собственную подпрограмму обработки. Тогда 64 адреса подпрограмм обработки прерываний можно разместить в 128-байтной таблице. Предположим, что эти 128 байт расположены в области памяти, начиная с адреса 0F00H и до 0F7FH. Тогда для использования режима 2 надо записать в регистр прерываний IV значение 0FH. Теперь пусть возникло внешнее прерывание и цепи формирования вектора подтверждения прерывания выставили на шину данных вектор 2EH.

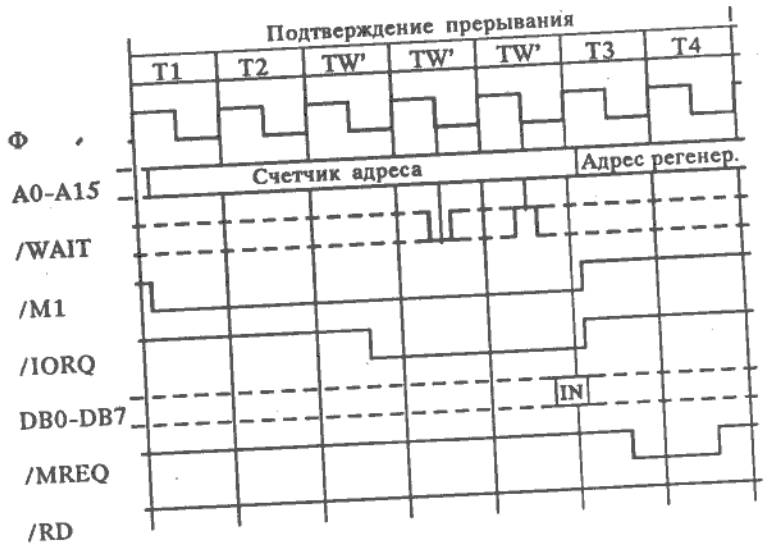
Процесс передачи управления тогда пойдет следующим образом. Прежде всего процессор сформирует адрес слова в таблице, образованный из содержимого IV и вектора подтверждения. Получится значение адреса, равное 0F2EH. Затем процессор прочитает с этого адреса двухбайтовое слово, равное, например, 0078H и вызовет подпрограмму, используя считанное слово в качестве адреса вызова, т.е. выполнит команду CALL 0078H.

Если двух тактов ожидания недостаточно внешним цепям для того, чтобы разобраться с приоритетами прерываний и вовремя выставить вектор подтверждения на шину данных, могут быть введены дополнительные такты ожидания путем использования сигнала WAIT. Соответствующие временные диаграммы показаны на рис. 12.

Немаскируемое прерывание отличается от маскируемого двумя важными обстоятельствами.

Во первых, оно имеет приоритет, более высокий, чем у маскируемого прерывания и захвата шины.

Во вторых, немаскируемое прерывание функционирует только в



TW' — такты ожидания, вводимые процессором независимо от состояния сигнала /WAIT.

Рис. 12. Такты ожидания при подтверждении маскируемого прерывания.

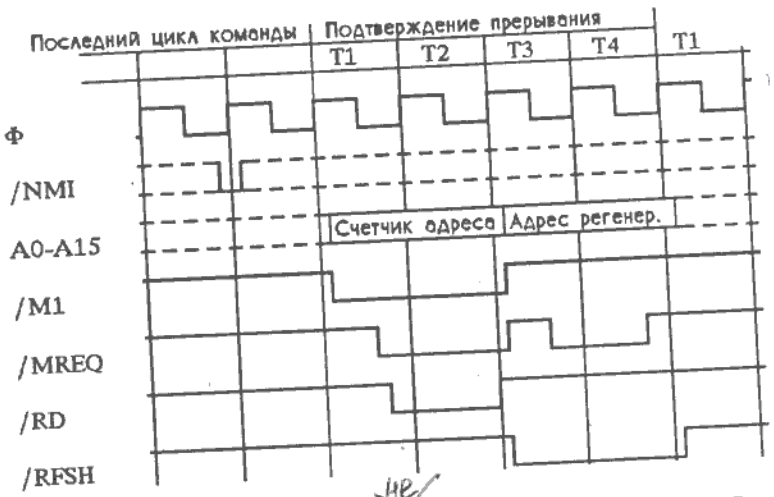


Рис. 13. Реакция на запрос немаскируемого прерывания.

Во первых, оно имеет приоритет, более высокий, чем у маскируемого прерывания и захвата шины.

Во вторых, немаскируемое прерывание функционирует только в режиме 1. Вслед за подтверждением прерывания всегда выполняется команда рестарта (RST) с передачей управления на адрес 0066H. Никакая другая команда по этому прерыванию не выполняется, а вектор подтверждения, даже если он выставлен на шину данных, игнорируется.

Временная диаграмма немаскируемого прерывания показана на рис. 13.

1.16. КОМАНДА ОСТАНОВА.

После выполнения команды останова процессор Z80 выполняет последовательность пустых (NOP) команд, пока не получит запроса прерывания или сброса.

Запросы как маскируемого, так и немаскируемого прерываний анализируются по переднему фронту тактовой частоты Ф в такте T4 каждого машинного цикла команды NOP.

Состояние останова прекратится, когда будет обнаружен запрос любого прерывания, после чего начнется соответствующи цикл подтверждения прерывания (рис. 11-13).

Заметим, что в результате выполнения последовательности команд NOP в течение состояния останова регенерация динамической памяти продолжает выполняться, поэтому данное состояние может продолжаться сколь угодно долго без опасности потери данных в динамической памяти.

Временная диаграмма состояния останова приведена на рис. 14.

Рис. 14. Выполнение команды останова.

2. НАБОР КОМАНД Z80.

2.1. ОБЩИЕ СВЕДЕНИЯ.

Система команд Z80 здесь будет рассматриваться как расширение системы 8080A.

Таблицы 2 и 3 содержат краткие сведения по каждой команде Z80 и устанавливают соответствующие между кодами и мнемоническими обозначениями команд.

К сожалению, тот факт, что система команд 8080A является подмножеством Z80, из таблицы 2 неочевиден, поскольку одинаковые команды этих процессоров имеют существенно разные мнемоники. Поэтому в таблице 4 приведены эквиваленты командам процессора 8080A. Немногие случаи несовместимости специально отмечены.

2.2. КОМАНДЫ ВВОДА/ВЫВОДА.

Существует несколько типов команд ввода/вывода процессора Z80:

— Простые команды IN и OUT, аналогичные 8080A, где во втором байте объективного кода команды содержится адрес порта ввода/вывода, и который выдается на линии AO-A7 шины адреса;

— Команды ввода/.вывода с косвенной регистровой адресацией порта. Эти команды передают данные между регистрами A, B, C, D, E. H или L и портом ввода/вывода, адрес которого указан в регистре C. Например, команды:

```
LD C,PORTN; загрузка адреса порта в регистр C;
    .
    .
    .
    .
IN D,(C) ; ввод данных с порта PORTN в регистр D;
```

эквивалентны следующим:

```
IN A,PORTN
LD D,A
```

Адрес порта, содержащийся в C, как обычно, выдается на линии A0-A7.

— Команды блочного обмена с портом ввода/вывода. Эти команды передают блок данных между портом ввода/вывода, указанным в регистре C и областью памяти, адресованной содержимым HL. Регистр B используется как счетчик пересылаемых байтов. Пока блок пересылается, после каждого байта содержимое B уменьшается на единицу. В команде блочного обмена с портом можно указать, увеличивать или уменьшать при этом содержимое пары HL.

Приведем пример программы с ее эквивалентом для 8080A:

```
        Z80                          8080A
LD B,COUNT              MVI B,COUNT
LD C,PORTN              LXI H,START
LD HL,START     LOOP:   IN PORTN
OUTIR                   MOV M,A
                        INX H
                        DCR B
                        JNZ LOOP
```

Эти программы пересылают количество байт, равное COUNT из порта с адресом PORTN в буфер, начинающийся с адреса памяти START в пордке возрастания адресов. COUNT и PORTN — символы, представляющие 8-битное целое без знака, START — метка адреса. Команда блочного обмена с портом окончит ра боту, когда содержимое регистра B станет равным 0.

При использовании команд блочного обмена следует учитывать два обстоятельства.

Во-первых, на все время пересылки блока прерывается процесс регенерации динамической памяти, поэтому при ана лизе программы, использующей блочные пересылки следует проверять выполнение условий регенерации, как указывалось выше. Во-вторых, на это время исключается также и прием запросов прерывания, так что время реакции на прерывание может оказаться на несколько порядков больше, чем в обыч ных условиях. Это надо учитывать при написании программ реального времени.

К счастью, процедуры прямого доступа в память этими командами не замедляются.

— Пошаговые команды блочного обмена с портами ввода/вывода. Они аналогичны командам блочного обмена, описанным выше в категории 3 за исключением того, что выполнение команды заканчивается после пересылки каждого байта. Обращаясь к примеру команды OUTIR, если эту команду заменить на OUTI, то после ее выполнения один байт будет передан из порта PORTN по адресу START, содержимое B будет декрементировано, пары HL — инкрементировано, а содержимое C останется без изменения.

Пошаговые команды блочного обмена позволяют организовать блочные пересылки, не нарушающие процесса регенерации динамической памяти и не удлиняющие время реакции на прерывание. Однако, из-за использования трехбайтовой команды условного перехода, такие пересылки лишь немногим более эффективны как в смысле быстродействия, так и экономии длины объектного кода, чем написанные в системе команд 8080A.

При использовании процессора Z80, в особенности, если применяются команды блочного обмена с портами, следует учитывать одну программно-аппаратную тонкость. Если у процессора 8080A (хотя это и не указано в спецификации первичного разработчика) команды ввода/вывода выдают один и тот же адрес порта как на старшую, так и на младшую половину шины адреса, и это часто позволяет использовать общий дешифратор для устройств памяти и ввода/вывода, то у Z80 дело обстоит иначе. Если при обычных командах IN и OUT у процессоров Z80 всех изготовителей, с которыми имел дело автор, адрес порта выдается аналогично 8080A, то в командах блочных пересылок процессора Z80 по спецификации первичного разработчика, а значит и у всех изготовителей реальных микросхем — по старшей половине шины выдается содержимое регистра B, т.е. оставшееся количество байт блочной пересылки. Это, конечно, дает для периферийных устройств возможность знать объем пересылки, но безусловно, вызывает у разработчика лишнюю головную боль.

Теперь рассмотрим преимущества, определяемые наличием новых команд ввода/вывода у процессора Z80.

Наличие косвенной регистровой адресации портов создает возможность программе, записанной в ПЗУ, обращаться к любому порту. Если адрес порта меняется, это изменение определется новой загрузкой регистра C. При выполнении программы из ОЗУ их наличие позволяет разработчику программы отказаться от использования динамического изменения объектного кода программы — одной из самых неприятных процедур при написании и отладке программ.

Команды блочного обмена с портами не вызывают большого энтузиазма как по причинам, изложенным выше, так и по следующим дополнительным обстоятельствам:

Исполнив единственный объектный код команды, блочная пересылка осуществляет передачу до 256 байт между портом и памятью. Эта передача осуществляется со скоростью процессора, а значит, периферийное устройство должно передавать или принимать данные с той же самой скоростью. Если же (как обычно и бывает) его скорость меньше, чем у процессора, единственным выходом остается использование состояния ожидания. Это еще увеличивает время выполнения длинной команды пересылки что, как указывалось выше, создает трудности с регенерацией памяти и реагированием на прерывания. Кроме того, введение состояний ожидания требует использования дополнительной схемотехники. Поэтому использование обычной логики пересылок с опросом слова состояния периферийного устройства в подавляющем большинстве случаев проще и эффективнее.

В заключение заметим, что все новые команды ввода/вывода имеют двухбайтный объектный код.

2.3. КОМАНДЫ ПЕРЕСЫЛОК С ПАМЯТЬЮ.

Команды, которые здесь классифицированы как пересылки с памятью, являются подмножеством команд загрузки (Load) по классификации фирмы Zilog. В этой группе команд Z80 предлагает одно нововведение: адресацию относительно базы (точнее базового адреса), хотя фирма-разработчик и считает ее индексной адресацией.

Команда, которая пересылает данные между регистром и байтовой ячейкой памяти может определить адрес памяти как содержимое индексного регистра плюс 8-битное смещение, задаваемое объектным кодом команды. Приведем пример фрагмента программы для Z80 и 8080A:

Z80	8080A
LD IX,BASE	LXI H,BASE
LD C,(IX+DISP)	LXI D,DISP
	DAD D
	MOV C,M

Заметим, что программа для Z80 не использует регистров процессора — за исключением индексного регистра. На 8080A требуется использование двух регистровых пар — HL и DE. Это пример реальной ценности использования индексного регистра. Z80 может использовать пары DE и HL для хранения промежуточных результатов, 8080A не может этого делать, ему пришлось бы хранить такие данные в памяти.

Главное приемущество, которое возникает при использовании индексной адресации состоит в том, что хорошо написанная программа для Z80 содержит намного меньше обращений к памяти, чем эквивалентная программа для 8080A. В результате растет быстродействие и уменьшается объем объектного кода.

Другие команды пересылок с памятью, отсутствующие в 8080A, включают команды загрузки индексных регистров из памяти и их запоминания в ней. Поскольку 8080A не имеет таких регистров, не было и соответствующих команд. Z80 имеет также команды пересылки 16-битного слова данных между любой регистровой парой, кроме AF, и памятью. Напомним, что 8080A умеет делать такие пересылки только с парой HL.

2.4. КОМАНДЫ ПЕРЕСЫЛОК БЛОКОВ И ПОИСКА В МАССИВЕ.

Команды пересылки блоков позволют переслать до 64 Кбайт данных из одной произвольной области памяти в другую. Пара HL

указывает на адрес буфера — источника, DE — буфера — приемника, а в паре BC задается длина пересылаемого блока.

После пересылки каждого байта пара BC декрементируетсяя, выполнение команды продолжается, пока в этой паре не образуется нуль. Код команды позволяет указать, инкрементировать или декрементировать пары HL и DE после пересылки каждого байта. Таким образом, данные могут пересылаться, начиная с младших или старших адресов. Ниже приведен пример программы блочной пересылки и ее эквивалент для 8080A:

Z80		8080A
LD BC,COUNT		LXI B,COUNT
LD DE,DEST		LXI D,DEST
LD HL,SRCE		LXI H,SRCE
LDIR	LOOP:	MOV A,M
		INX H
		STAX D
		INX D
		DCX B
		MOV A,B
		ORA C
		JNZ LOOP

Две последовательности команд, показанные выше, передают блок данных длиной COUNT байт из буфера, начинающегося с адреса SRCE в буфер с адреса DEST. SRCE и DEST- 16-битовые адресные метки, COUNT — символ, представляющий 16-битовое двоичное число.

Приведенное выше сравнение заставляет 8080A выглядеть особенно плохо, показывая слабость системы команд последнего. Адреса памяти инкрементируются разными командами, статусные флаги после декрементирования регистровой пары не устанавливаются, поэтому содержимое BC проверяется двумя дополнительными командами: загрузкой B в A и логическим сложением с C.

Однако на практике эти преимущества удается реализовать далеко не всегда. Вспомним, что говорилось выше по поводу команд блочного обмена с портами в части влияния на процесс регенерации памяти и время реакции на прерывание. В случае блочных пересылок ситуация может ухудшиться еще ровно в 256 раз и время пересылки может не только превосходить максимально допустимый период регенерации, но и быть сравнимым с периодом системных прерываний от таймера (тиков). В случае маскирования обработки последних возможны самые разные эффекты — от срыва изображения в игровых программах до нарушений хода часов реального времени в контроллерах.

Поэтому использование команд блочных пересылок требует до-

статочно глубоких знаний архитектуры вычислительной системы и, иногда, проведения довольно объемных расчетов их максимально возможной длины. В результате их можно рекомендовать к использованию лишь системным программистам.

В особенности опасно их применение в инструментальных программах и библиотеках, рассчитанных на применение в самых различных вычислительных структурах на базе процессора Z80. В данном случае в состав документации на такой программный продукт должно быть в обязательном порядке включено максимальное возможное значение времени выполнения одиночной команды в машинных тактах.

Команда поиска в блоке просматривает блок данных в памяти, отыскивая байт, совпадающий с содержимым аккумулятора. Пара HL содержит адрес в памяти, в то время, как пара BC, как и ранее, служит счетчиком байтов. Когда найден совпадающий с содержимым аккумулятора байт, выполнение команды останавливается. В процессе выполнения команды после каждого сравнения пара BC декрементируется, а пара HL — инкрементируется или декрементируется в зависимости от вида команды. Таким образом, поиск можно вести как с начала, так и с конца блока.

Результат каждого сравнения устанавливает флаги Z и P/O. Если совпадающий байт найден, Z устанавливается в 1, иначе Z=0.

Когда пара BC становится равной 0, флаг P/O устанавливается в 0, иначе он равен 1.

Ниже приведен пример, использующий команду поиска в блоке для процессора Z80 и его эквивалент для 8080A.

```
          Z80
          LD A,REFC
          LD BC,COUNT
          LD HL,SRCE
          CPDR
          JR Z,FOUND
        ;БАЙТ НЕ НАЙДЕН
          —
          —
          —
        ;БАЙТ НАЙДЕН
FOUND:    —
          —
          —
```

```
          8080A
          LXI BC,COUNT
          LXI HL,SOURCE
LOOP:     MVI A,REFC
          CMP M
          JZ FOUND
          DCX H
          DCX B
          MOV A,B
          ORA C
          JNZ LOOP
        ;БАЙТ НЕ НАЙДЕН
          —
          —
          —
        ;БАЙТ НАЙДЕН
FOUND:    —
          —
```


Каждый из показанных примеров ищет байт, совпадающий с символом REFC, в буфере памяти. Старший адрес буфера равен SRCE, а его длина COUNT.

В приведенном выше примере поиск осуществляется от старшего адреса к младшему.

FOUND — метка в программе, на которую происходит переход, когда байт найден. Если байт не найден, т.е. пара BC досчитала до нуля, выполнение программы продолжается в естественном порядке.

Команда поиска в блоке особенно полезна при просмотре буферов большой длины для поиска часто встречающихся байтов. Предположим, в некотором буфере находится текст в коде ASCII, в который добавлены управляющие символы.

Пусть все эти коды имеют двухбайтную длину, где первый байт — код 1BH, а второй — определяет сам управляющий код. Вы можете использовать один набор регистров для поиска в буфере, а другой — для обработки этого буфера после того, как этот управляющий код обнаружен. Выполняется такая смена наборов командой EXX.

Разумеется, при использовании команд поиска в блоке справедливы все замечания, сделанные выше относительно операций с длительным и непредсказуемым временем выполнения, поэтому и ими надо пользоваться с большой осторожностью.

Видимо, понимая сложности работы с командами непредсказуемой длительности выполнения, разработчики Z80 снабдили каждую команду блочных пересылок и поиска одношаговым аналогом так же, как и команды блочного ввода-вывода.

Использование этих команд хотя и требует организации цикла, но снимает все приведенные выше затруднения, и является вполне разумным компромиссом, особенно при разработке программ, обладающих свойством переносимости между различными аппаратными конфигурациями систем на процессоре Z80.

2.5. КОМАНДЫ ОБРАБОТКИ ДАННЫХ ПАМЯТИ.

Эта группа команд, включающая арифметические и логические операции над ячейками памяти, является частью совокупности арифметических и логических команд по классификации фирмы «Zilog».

Внутри группы команд, оперирующих с памятью, Z80 предлагает одно, но очень ценное усовершенствование, заключающееся во введении эквивалентов всех таких команд 8080A, но использующих адресацию относительно базового адреса (индексную адресацию).

Мы уже рассматривали это усовершенствование на примере группы команд пересылок с памятью. Ниже приведен пример программы с ее эквивалентом для 8080A:

Z80	8080A
LD IX,BASE ADD (IX + DISP)	LXI H,BASE LXI D,DISP DAD D ADD M

Те же самые комментарии, которые мы дали, рассматривая примеры использования индексной адресации в командах пересылок с памятью, справедливы и в данном случае.

2.6. КОМАНДЫ ЗАГРУЗКИ НЕПОСРЕДСТВЕННЫМИ ДАННЫМИ.

В этой группе команд процессор Z80 обладает двумя расширениями:

— введены команды, позволяющие загрузить непосредственные данные в дополнительные регистры процессора (IX, IY);
— предусмотрены команды для загрузки байта непосредственных данных в память с использованием адресации относительно базового адреса.

2.7. КОМАНДЫ ПЕРЕХОДОВ.

В дополнение к трехбайтовой команде безусловного перехода JUMP, имеющейся в системе команд 8080A, Z80 имеет двухбайтовую команду безусловного перехода, и две команды, которые позволяют осуществить безусловный переход на адрес, записанный в индексных регистрах.

Две команды перехода по индексным регистрам переписывают содержимое указанного в них индексного регистра в счетчик адреса команды.

Двухбайтовая команда безусловного перехода использует второй байт объекта кода команды, как 8-битное двоичное число со знаком, которое складывается с содержимым счетчика адреса команды после того, как он инкременируется для того, чтобы указать на следующую команду. Это обычная процедура перехода по относительному адресу.

Заметим, что Z80 использует многие из свободных кодов команд 8080A именно для реализации двухбайтовых команд безусловного и условных ветвлений. Это резонно, поскольку использование двухбайтового кода команды плюс третий байт величины перехода не создал бы никаких преимуществ перед имеющимися у 8080A трехбайтовыми командами переходов, которые, естественно, имеются и в системе команд Z80.

2.8. КОМАНДЫ ВЫЗОВА ПОДПРОГРАММЫ И ВОЗВРАТА.

В этой части системы команд Z80 и 8080A совершенно аналогичны и включают как команды безусловного вызова и возврата, так и целый ряд условных, что является одной из наиболее сильных сторон системы команд 8080A.

2.9. КОМАНДЫ, ОБРАБАТЫВАЮЩИЕ НЕПОСРЕДСТВЕННЫЕ ДАННЫЕ.

Эта группа команд также совершенно аналогична командам 8080A.

2.10. КОМАНДЫ УСЛОВНЫХ ПЕРЕХОДОВ.

Процессор Z80 предлагает два существенных усовершенствования в этой группе команд:

— введены двухбайтовые эквиваленты для четырех наиболее употребительных команд условных переходов 8080A. Функционируют они точно также, как и описанная выше двухбайтовая команда безусловного перехода;
— введена команда декрементирования регистра и перехода по его ненулевому содержимому, которая обычно бывает полезна при создании любых итеративных циклов.

Когда эта команда выполняется, содержимое регистра B уменьшается на единицу, и если после этого становится равным нулю, выполняется следующая в памяти команда. Если же равенство нулю не достигнуто, выполняется переход на адрес, определяемый смещением, записанным во втором байте кода команды.

Команда двухбайтовая. Ее второй байт интерпретируется как двоичное число со знаком.

Ниже приведен пример использования команды DJNZ и эквивалент этой программы для 8080A.

	Z80		8080A
	AND A		ANA A
	LD IX,VALA		LXI H,VALA
	LD IX,VALB		LXI D,VALB
	LD B,CNT		MVI B,CNT
LOOP:	LD A,(IX)	LOOP:	LDAX D
	ADC A,(IY)		ADC M
	LD (IX),A		MOV M,A
	INC IX		INX H
	INC IY		INX D
	DJNZ LOOP		DCR R
			JNZ LOOP

Две последовательности команд, приведенных выше, совершают простое многобайтовое двоичное сложение. Содержимое двух буферов, начинающихся с VALA и VALB, суммируется, результат записывается в буфер VALA.

Первая команда в каждой последовательности введена, чтобы очистить флаг переноса. Как и в 8080A, в Z80 не существует команды, устанавливающей этот флаг в 0, не совершая никаких других действий.

2.11. КОМАНДЫ МЕЖРЕГИСТРОВЫХ ПЕРЕСЫЛОК.

Эта группа команд включает подмножество команд загрузки (LOAD), и команд обмена (EXCHANGE), за исключением команд обмена с верхушкой стека.

Расширения системы команд в этой группе прямо связаны с введением в Z80 дополнительных регистров.

Другими словами, для регистров, отсутствующих в 8080A, введены и команды пересылок данных с этими регистрами.

Команды, которые осуществляют обмен между основной и дублирующей группой регистров, требуют комментария. Заметьте, что можно осуществить обмен со всей дублирующей группой регистров, но можно и между отдельными регистровыми парами. Если использовать эти команды вслед за подтверждением прерывания, нет необходимости сохранять содержимое регисторов в стеке. Конечно, это будет работать лишь при одном уровне прерываний. Имеются также случаи, когда дублирующий набор регистров может эффективно использоваться и в обычной программной логике, как было показано при рассмотрении команд блочных пересылок.

2.12. КОМАНДЫ МЕЖРЕГИСТРОВОЙ ОБРАБОТКИ ДАННЫХ.

Имеются несколько новых команд межрегистровой обработки данных, которые выполняют следущее:

- прибавление (без учета флага переноса) содержимого регистровой пары к индексному регистру;
- прибавление к содержимому регистровой пары HL другой регистровой пары с учетом флага переноса;
- вычитание из содержимого регистровой пары HL другой регистровой пары с учетом флага переноса.

2.13. КОМАНДЫ ОБРАБОТКИ ДАННЫХ В РЕГИСТРЕ.

В этой группе команд Z80 вводит три дополнения:

- можно инкрементировать и декрементировать содержимое индексных регистров;
- добавлено большое количество команд линейного и кольцевого сдвига. Эти команды показаны в таблице 2. В частности, отметим команды RLD и RRD, которые очень полезны при левом или правом сдвигах двоично-десятичных чисел.

2.14. КОМАНДЫ ОБРАБОТКИ БИТОВ.

Эта группа команд совершенно отсутствует в 8080A, но в то же время имеет важное значение в применениях микропроцессора.

Особенно они важны в обработке сигналов. Одиночный сигнал — это одноразрядная двоичная величина, а не часть 8-битного набора. Один из важнейших просчетов разработчиков микропроцессоров — игнорирование команд обработки битов.

Z80 имеет команды установки битов в единицу (SET), сброса в нуль (RES) или контроля состояния (BIT) индивидуальных битов в памяти и регистрах общего назначения.

Результат проверки отображается в состоянии флага Z.

Приведем пример команды Z80 с его эквивалентом для 8080A:

Z80	8080A
BIT 4,A	MOV B,A ANI 10H

8080A проверяет бит аккумулятора, одновременно разрушая все остальные биты, из-за этого приходится сохранять содержимое аккумулятора перед проверкой.

Продолжим пример, чтобы подчеркнуть мощность команд обработки бит Z80:

Z80	8080A
LD IY,BASE SET 2,(IY+DISP)	LXI H,BASE LXI D,DISP DAD D MVI A,4 ORA M

Заметим опять, что 8080A требует использования регистров A, D, E, H и L.

Битовые команды Z80 оперируют только с памятью и регистрами общего назначения, но в большинстве приложений микропроцессоров устанавливать, сбрасывать и проверять состояния необходимо главным образом у портов ввода-вывода. Z80 не предоставляет таких команд, и эти возможности появляются лишь при выборе адресации портов как ячеек памяти. Заметим также, что команды оперирования с битами могут быть достаточно гро-

моздки как в смысле длины кода (до 4-х байт) так и особенно по длительности выполнения (до 23 тактов).

2.15. КОМАНДЫ РАБОТЫ СО СТЕКОМ.

Дополнительные команды работы со стеком Z80 позволяют записывать в стек, извлекать из него и обменивать с верхушкой стека оба индексных регистра.

2.16. КОМАНДЫ УПРАВЛЕНИЯ ПРЕРЫВАНИЯМИ.

В дополнение к трем командам управления прерываниями 8080A (EI, DI и RST), Z80 имеет две команды возврата из прерываний. Для возврата из подпрограмм обработки, соответственно, маскируемого и немаскируемого прерываний, используются команды RETI и RETN.

Обе они вляются двухбайтовыми командами. В самом процессоре эти команды разрешают прерывания соответствующего вида, и кроме того, действуют точно так же, как команда возврата из прерывания (RET). Необходимость в специальных командах вызвана более сложной, чем у 8080A, системой обработки сигналов прерываний из-за наличия двух их видов.

3. ЭЛЕКТРИЧЕСКИЕ ХАРАКТЕРИСТИКИ.

3.1. СТАТИЧЕСКИЕ ХАРАКТЕРИСТИКИ.

Все напряжения даны относительно U_{ss}=0V.

Параметр	Обозн.	Ед. изм.	Мин. значение	Макс. значение	Условия измерений
Рабочее напряжение	U_{cc}	В	4.75	5.25	T_a=0–70 ° C
Входное напряжение	U_{il} U_{ih}	В	-0.3 2	0.8 U_{cc}	T_a=0–70 ° C
Напряжение тактового сигнала	U_{ihc}	В	U_{cc} - 0.6*)	U_{cc} + 0.6	
Выходное напряжение	U_{ol} U_{oh}	В	— 2.4	0.4 —	I_{ol}=1.8мА T_a=0–70 ° C I_{oh}=0.25мА T_a=0–70 ° C

Параметр	Обозн.	Ед. изм.	Мин. значение	Макс. значение	Условия измерений
Ток потребления	I_{cc}	мА	—	150/200**)	U_{cc}=5В+5% T_a=0–70 ° C
Входной ток утечки	I_{li}	мкА	—	10	U_{in}=0 - U_{cc}
Ток утечки тристабильного выхода в плавающем состоянии	I_{loh} I_{lo}	мкА мкА	—	10 -10	U_{out}=2.4В - U_{cc} U_{out}=0.4В
Ток утечки шины данных при вводе	I_{ld}	мкА	—	10	U_x=0 - U_{cc}
Входная емкость тактового входа	C_c	пФ	—	20	T_a=20 ° C и f=1МГц
Емкость входа	C_i	пФ	—	5	
Емкость выхода	C_o	пФ	—	10	

*) Допускается использование открытого TTL-выхода с нагрузочным регистром не более 330 ом, соединенным с U_{cc}.
**) В числителе—для Z80, а в знаменателе—для Z80A.

3.2. ДИНАМИЧЕСКИЕ ХАРАКТЕРИСТИКИ Z80

При U_{cc} = 5В ± 5%, Cl = 50пФ и T_a = 0–70 ° C.

Время в наносекундах.

Параметр	Обозначение	Мин. значение	Макс. значение
Период тактовых импульсов	T_c	400	*)
Длительность низкого уровня тактового сигнала	T_w(ФL)	180	2000
Длительность высокого уровня тактового сигнала	T_w(ФL)	180	**)

Параметр	Обозначение	Мин. значение	Макс. значение
Длительность переднего/заднего фронта тактового сигнала	T_r, T_f	—	30
Установка сигнала /WAIT до H-L перехода такта	T_b (WT)	70	—
Установка сигнала /RESET до L-H перехода такта	T_b (RS)	90	—
Установка сигнала /INT до L-H перехода такта	T_b (IT)	80	—
Установка сигнала /BUSRQ до L-H перехода такта	T_b (BQ)	80	—
Установка данных до L-H перехода такта в цикле M1	T_{bc} (D)	50	—
Установка данных до H-L перехода такта в циклах M2–M5	T_{bc} (D)	60	—
Задержка сигналов на шинах	T_h	0	—
Ширина импульса низкого уровня сигнала /NMI	T_w (NMI)	80	—

*) $T_c = T_w(CL) + T_w(CH) + T_r + T_f$

**) Не имеет фиксированного значения, т.е. при высоком уровне тактового сигнала МП Z80 может находиться в устойчивом состоянии сколь угодно долго.

Времена задержек

При U_{cc} = 5В ± 5%, C_l = 50пФ и T_a = 0–70° C

Задержка	Обозначение	Макс. значение, нс
от H-L перехода такта до /M1=L	T_{dl} (m1)	130
от H-L перехода такта до /M1=H	T_{dm} (m1)	130
от H-L перехода такта до /MREQ=H	T_{dhc} (mr)	100
от L-H перехода такта до /MREQ=H	T_{dhc} (mr)	100
от H-L перехода такта до /MREQ=L	T_{dlc} (mr)	100
от L-H перехода такта до /IORQ=L	T_{dlc} (jr)	90
от H-L перехода такта до /IORQ=L	T_{dlc} (jr)	110
от L-H перехода такта до /IORQ=H	T_{dhc} (jr)	100
от H-L перехода такта до /IORQ=H	T_{dhc} (jr)	110

Задержка	Обозначение	Макс. значение, пс
от L-H перехода такта до /RD=L	T_{dlc} (rd)	100
от H-L перехода такта до /RD=L	T_{dlc} (rd)	130
от L-H перехода такта до /RD=H	T_{dhc} (rd)	100
от H-L перехода такта до /RD=H	T_{dhc} (rd)	110
от L-H перехода такта до /WR=L	T_{dlc} (wr)	80
от H-L перехода такта до /WR=L	T_{dlc} (wr)	90
от L-H перехода такта до /WR=H	T_{dhc} (wr)	100
от L-H перехода такта до /RFSH=H	T_{dh} (rf)	150
от L-H перехода такта до /RFSH=L	T_{dl} (rf)	180
от H-L перехода такта до /HALT=L	T_d (ht)	300
от L-H перехода такта до /BUSAK=L	T_{dl} (da)	120
от H-L перехода такта до /BUSAK=H	T_{dh} (da)	100
вывода адреса	T_d (ad)	145
адреса до перехода к третьему состоянию	T (ad)	110
вывода данных	T_d (d)	230
данных до перехода к третьему состоянию в цикле записи	T (d)	90
сигналов /MREQ, /IORQ, /WR до перехода к третьему состоянию	T (c)	100

Время задержки увеличивается на 10нс при возрастании емкости нагрузки на каждые 50пФ до максимально 200 пФ для шины данных и 100пФ для шин адреса и управления.

Дополнительные данные о временах.

Вывод адреса до активизации /MREQ в циклах обращения к памяти:

$T_{acm} = T_w(CH) + T_r - 75$нс.

Вывод адреса до активизации /IORQ, /RD, /WR в циклах ввода/вывода:

$T_{ac1} = T_c - 80$нс.

Задержка адреса после снятия /RD или /WR:

$T_{ca} = T_w(CL) + T_r - 40$нс.

Задержка адреса после снятия /RD или /WR при переходе в третье состояние:

$T_{caf} = T_w(CL) + T_f - 60$нс.

Вывод данных до активизации /WR в циклах обращения к памяти:

$T_{dcm} = T_c - 210$нс.

Вывод данных до активизации /WR в циклах ввода-вывода:
$T_{dc1} = T_w(CL) + T_r - 210$ нс.

Задержка данных после снятия /WR:
$T_{cdf} = T_w(CL) + T_r - 80$нс.

Ширина импульса низкого уровня /MREQ:
$T_w(MRL) = T_c - 40$нс.

Ширина импульса высокого уровня /MREQ:
$T_w(MRH) = T_w(CH) + T_r - 30$нс.

Ширина импульса низкого уровня /WR:
$T_w(WRL) = T_c - 40$нс.

Вывод /M1 до активизации /IORQ в цикле подтверждения прерывания:
$T_{m1} = 2T_c + T_w(CH) + T_r - 80$ нс.

3.3. ДИНАМИЧЕСКИЕ ХАРАКТЕРИСТИКИ Z80A.

При U_{cc}=5В ± 5%, C_l=50пФ и V_a=0–70 ° С.

Параметр	Обозначение	Мин. значение	Макс. значение
Период тактовых импульсов	T_c	250	*)
Длительность низкого уровня тактового сигнала	T_w(ФL)	110	2000
Длительность высокого уровня тактового сигнала	T_w(ФL)	110	**)
Длительность переднего/заднего фронта тактового сигнала	T_r, T_f	—	30
Установка сигнала /WAIT до H-L перехода такта	T_b (WT)	70	—
Установка сигнала /RESET до L-H перехода такта	T_b (RS)	60	—
Установка сигнала /INT до L-H перехода такта	T_b (IT)	80	—
Установка сигнала /BUSRQ до L-H перехода такта	T_b (BQ)	50	—
Установка данных до L-H перехода такта в цикле M1	T_{bc} (D)	35	—
Установка данных до H-L перехода такта в циклах M2–M5	T_{bc} (D)	50	—

Параметр	Обозначение	Мин. значение	Макс. значение
Задержка сигналов на шинах	T_h	0	—
Ширина импульса низкого уровня сигнала /NMI	T_w (NMI)	80	—

*) $T_c = T_w(CL) + T_w(CH) + T_r + T_r$

**) Не имеет фиксированного значения, т.е. при высоком уровне тактового сигнала МП Z80A может находиться в устойчивом состоянии сколь угодно долго.

Времена задержек.

При U_{cc} = 5В ± 5%, C_l = 50пФ и V_a = 0–7С.

Задержка	Обозначение	Макс. значение, нс
от H-L перехода такта до /M1=L	T_{dl} (m1)	100
от H-L перехода такта до /M1=H	T_{dm} (m1)	100
от H-L перехода такта до /MREQ=H	T_{dhc} (mr)	85
от L-H перехода такта до /MREQ=H	T_{dhc} (mr)	85
от H-L перехода такта до /MREQ=L	T_{dlc} (mr)	85
от L-H перехода такта до /IORQ=L	T_{dlc} (jr)	75
от H-L перехода такта до /IORQ=L	T_{dlc} (jr)	85
от L-H перехода такта до /IORQ=H	T_{dhc} (jr)	85
от H-L перехода такта до /IORQ=H	T_{dhc} (jr)	85
от L-H перехода такта до /RD=L	T_{dlc} (rd)	85
от H-L перехода такта до /RD=L	T_{dlc} (rd)	95
от L-H перехода такта до /RD=H	T_{dhc} (rd)	85
от H-L перехода такта до /RD=H	T_{dhc} (rd)	85
от L-H перехода такта до /WR=L	T_{dlc} (wr)	65
от H-L перехода такта до /WR=L	T_{dlc} (wr)	80
от L-H перехода такта до /WR=H	T_{dhc} (wr)	80
от L-H перехода такта до /RFSH=H	T_{dh} (rf)	120
от L-H перехода такта до /RFSH=L	T_{dl} (rf)	130
от H-L перехода такта до /HALT=L	T_d (ht)	300
от L-H перехода такта до /BUSAK=L	T_{dl} (da)	100
от H-L перехода такта до /BUSAK=H	T_{dh} (da)	100
вывода адреса	T_d (ad)	110
адреса до перехода к третьему состоянию	T (ad)	90

Задержка	Обозначение	Макс. значение, нс
вывода данных	T_d (d)	150
данных до перехода к треьему состоянию в цикле записи	T (d)	90
сигналов /MREQ, /IORQ, /WR до перехода к третьему состоянию	T (c)	80

Время задержки увеличивается на 10нс при возрастании емкости нагрузки на каждые 50пФ до максимально 200 пФ для шины данных и 100пФ для шин адреса и управления.

Дополнительные данные о времени.

Вывод адреса до активизации /MREQ в циклах обращения к памяти:
$T_{acm} = T_w(CH) + T_r - 65$нс.

Вывод адреса до активизации /IORQ, /RD, /WR в циклах ввода/вывода:
$T_{ac1} = T_c - 70$нс.

Задержка адреса после снятия /RD или /WR:
$T_{ca} = T_w(CL) + T_r - 50$нс.

Задержка адреса после снятия /RD или /WR при переходе в третье состояние:
$T_{caf} = T_w(CL) + T_r - 45$нс.

Вывод данных до активизации /WR в циклах обращения к памяти:
$T_{dcm} = T_c - 170$нс.

Вывод данных до активизации /WR в циклах ввода-вывода:
$T_{dc1} = T_w(CL) + T_r - 170$ нс.

Задержка данных после снятия /WR:
$T_{cdf} = T_w(CL) + T_r - 70$нс.

Ширина импульса низкого уровня /MREQ:
$T_w(MRL) = T_c - 30$нс.

Ширина импульса высокого уровня /MREQ:
$T_w(MRH) = T_w(CH) + T_r - 20$нс.

Ширина импульса низкого уровня /WR:
$T_w(WRL) = T_c - 30$нс.

Вывод /M1 до активизации /IORQ в цикле подтверждения прерывания:
$T_{m1} = 2T_c + T_w(CH) + T_r - 65$ нс.

3.4. ПРЕДЕЛЬНЫЕ ЗНАЧЕНИЯ.

Предельные значения даны при $T_a = 70°$ C.

Параметр	Обозначение	Ед. измер.	Мин. значение	Макс. значение
Рабочее напряжение	U_{cc}	В	-0.3	7
Входное напряжение	U_{in}	В	-0.3	7
Диапазон рабочих температур	T_a	С	0	70
Диапазон температур хранения	T_{st}	С	-65	150
Рассеиваемая мощность	P	Вт	—	1.5

На корпусе дополнительно может быть указано исполнение:
C—керамический корпус
P—пластмассовый корпус
S—обычные условия эксплуатации (5В ± 5%, 0-70° C)
E—расширенные условия эксплуатации (5В ± 5%, -40 +8° C)
M—военное исполнение (5В ± 10%, -55 +12° C)

3.5. ПОКАЗАТЕЛИ НАДЕЖНОСТИ.

Интенсивность отказов L_{po} менее пяти на десять в минус восьмой степени в час.

При средней электрической нагрузке (U_{cc} от 4,75 до 5,25 В и температуре окружающей среды менее 50С), нормальных климатических и механических воздействиях, средняя наработка на отказ составляет 200 тыс. часов.

Таблица 2. Мнемоники и команды Z80 в алфавитном порядке

Мнемоника	Команда	CZPSNH	Длн	Ткт	Комментарий
ADC A, r	A: = A + r + CY	**V*0*	1	4	r = A, B, C, D, E, H, L
ADC A, (HL)	A: = A + (HL) + CY		1	7	
ADC A, n	A: = A + n + CY		2	7	n - байт (0..FF)
ADC A, (ii + n)	A: = A + (ii + n) + CY		3	19	ii = IX, IY
ADC HL, rr	HL: = HL + rr + CY	**V*0x	2	15	rr = BC, DE, HL, SP
ADD A, r	A: = A + r	**V*0*	1	4	
ADD A, (HL)	A: = A + (HL)		1	7	
ADD A, n	A: = A + n		2	7	
ADD A, (ii + n)	A: = A + (ii + n)		3	19	
ADD HL, rr	HL: = HL + rr	*...0x	1	11	
ADD IX, ry	IX: = IX + px		2	15	ry = BC, DE, SP, IY
ADD IY, rx	IY: = IY + py		2	15	rx = BC, DE, SP, IX
AND r	A: = A and r	0*P*01	1	4	
AND (HL)	A: = A and (HL)		1	7	
AND n	A: = A and n		2	7	
AND (ii + n)	A: = A and (ii + n)		3	19	
BIT b, r	Z: = not r < bit № b	.*xx01	2	8	b - номер бита (0..7)
BIT b, (HL)	Z: = not (HL) < bit № b		2	12	x < bit №b - бит с но-
BIT b, (ii + n)	Z: = not (ii + n) < bit № b		4	20	мером b из объекта x
CALL nn	PUSH PC;PC: = nn	.....	3	17	nn - слово (0..FFFF)
CALL cc, nn	если cc то CALL nn		3	17	cc = C, NC, Z, NZ, M, P, PE,
	иначе продолжать			10	PO
CCF	CY: = not CY	*...0x	1	4	
CP r	A-r	**V*1*	1	4	
CP (HL)	A-(HL)		1	7	
CP n	A-n		2	7	
CP (ii + n)	A-(ii + n)		3	19	
CPD	A-(HL);dec HL;dec BC	.***1*	2	16	PV = 0 если BC = 0, иначе PV = 1
CPDR	Повтор CPD до Z = 1	.***1*	2	21	
	или BC = 0			16	
CPI	A-(HL);inc HL;dec BC	.***1*	2	16	PV = 0 если BC = 0, иначе PV = 1
CPIR	Повтор CPI	.***1*	2	21	
	до Z = 1 или BC = 0			16	

Таблица 2. Продолжение.

Мнемоника	Команда	CZPSNH	Длн	Ткт	Комментарий
CPL	A: = A xor 255	11	1	4	
DAA	Десятич.настр.аккум.	**P*.*	1	4	
DEC r	r: = r-1	.*V*1*	1	4	
DEC (HL)	(HL): = (HL)-1		1	11	
DEC (ii + n)	(ii + n): = (ii + n)-1		3	23	
DEC rr	rr: = rr-1	.....	1	6	
DEC ii	ii: = ii-1		2	10	
DI	IFF: = 0	.....	1	4	
DJNZ e	dec B;если B = /0 JR e	.....	2	13	e - относительный
	если B = 0 продолжать			8	адрес
EI	IFF: = 1	.....	1	4	
EX AF, AF'	AFAF'	.....	1	4	
EX DE, HL	DEHL		1	4	
EX (SP), HL	(SP)HL		1	4	
EX (SP), ii	(SP)ii		2	23	
EXX	BCBC';DEDE';	.....	1	4	
	HLHL'				
HALT	Останов	.....	1	4	
IM 1	Режим прерываний 1	.....	2	8	
IM 2	Режим прерываний 2				
IM 3	Режим прерываний 3				
IN A, (n)	A: = порт(n)	.....	2	11	
IN r, (C)	r: = порт(C)	.*P*0*	2	12	
IN ?, (C)	устанавливает флаги	.*P*0*	2	12	не может быть вве-
	аналогично IN r, (C)				дена мнемоникой; код ED 70
INC r	r: = r + 1	.*V*0*	1	4	
INC (HL)	(HL): = (HL) + 1		1	11	
INC (ii + n)	(ii + n): = (ii + n) + 1		3	23	
INC rr	rr: = rr + 1	.....	1	6	
INC ii	ii: = ii + 1		2	10	
IND	(HL): = порт(C);	x*xx1x	2	16	Z = 1 если B = 0, иначе
	dec HL;dec B				Z = 0
INDR	Повтор IND до B = 0	x1xx1x	2	21	
				16	

Таблица 2. Продолжение.

Мнемоника	Команда	CZPSNH	Длн	Ткт	Комментарий
INI	(HL): = порт(C); inc HL;dec B	x*xx1x	2	16	Z = 1 если B = 0, иначе Z = 0
INIR	Повтор INI до B = 0	x1xx1x	2	21 16	
JP nn	PC: = nn	.....	3	10	
JP cc, nn	если cc то JP nn	.....	3	10	
JP (HL)	PC: = HL		1	4	
JP (ii)	PC: = ii		2	8	
JR e	PC: = PC + e	.....	2	12	
JR cond, e	Если cond то JR e иначе NOP		2	12 7	cond = C, NC, Z, NZ
LD r, r	r: = r	.....	1	4	
LD r, (HL)	r: = (HL)		1	7	
LD r, n	r: = n		2	7	
LD r, (ii + n)	r: = (ii + n)		3	19	
LD (HL), r	(HL): = r		1	7	
LD (ii + n), r	(ii + n): = r		3	19	
LD (HL), n	(HL): = n		2	10	
LD (ii + n), n	(ii + n): = n		4	19	
LD A, (BC)	A: = (BC)		1	7	
LD A, (DE)	A: = (DE)		1	7	
LD A, (nn)	A: = (nn)		3	13	
LD (BC), A	(BC): = A		1	7	
LD (DE), A	(DE): = A		1	7	
LD (nn), A	(nn): = A		3	13	
LD A, I	A: = I	***00	2	9	PV = IFF
LD A, R	A: = R		2	9	PV = IFF
LD I, A	I: = A	.....	2	9	
LD R, A	R: = A		2	9	
LD rr, nn	rr: = nn		3	10	
LD ii, nn	ii: = nn		4	14	
LD HL, (nn)	HL: = (nn)		3	16	
LD rr, (nn)	rr: = (nn)		4	20	

Таблица 2. Продолжение.

Мнемоника	Команда	CZPSNH	Длн	Ткт	Комментарий
LD ii, (nn)	ii: = (nn)		4	20	
LD (nn), HL	(nn): = HL		3	16	
LD (nn), rr	(nn): = rr		4	20	
LD (nn), ii	(nn): = ii		4	20	
LD SP, HL	SP: = HL		1	6	
LD SP, ii	SP: = ii		1	10	
LDD	(DE): = (HL); dec DE, HL, BC	..*.00	2	16	PV = 0 если BC = 0, иначе PV = 1
LDDR	Повтор LDD до Z = 1 или BC = 0	..0.00	2	21 16	
LDI	(DE): = (HL); inc DE, HL;dec BC	..*.00	2	16	PV = 0 если BC = 0, иначе PV = 1
LDIR	Повтор LDI до Z = 1 или BC = 0	..0.00	2	21 16	
NEG	A: = 0-A	**V*1*	2	8	
NOP	Пустая операция	.....	1	4	
OR r	A: = A or r	0*P*00	1	4	
OR (HL)	A: = A or (HL)		1	7	
OR n	A: = A or n		2	7	
OR (ii + n)	A: = A or (ii + n)		3	19	
OTDR	Повтор OUTD до B = 0	x1xx1x	2	21 16	
OTIR	Повтор OUTI до B = 0	x1xx1x	2	21 16	
OUT (n), A	порт(n): = A	.....	2	11	
OUT (C), r	порт(C): = r		2	12	
OUTD	порт(C): = (HL); dec HL;dec B	x*xx1x	2	16	Z = 1 если B = 0, иначе Z = 0
OUTI	порт(C): = (HL); inc HL;dec B	x*xx1x	2	16	Z = 1 если B = 0, иначе Z = 0
POP qq	qq: = (SP);SP: = SP + 2	.....	1	10	qq = AF, BC, DE, HL
POP ii	ii: = (SP);SP: = SP + 2		2	14	

Таблица 2. Продолжение.

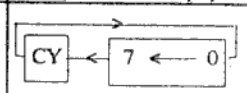
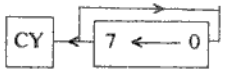
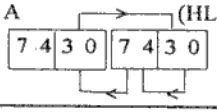
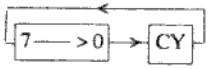
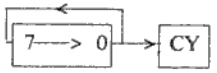
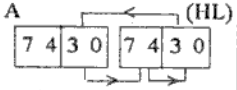
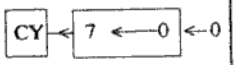
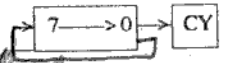
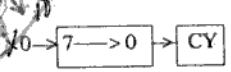
Мнемоника	Команда	CZPSNH	Длн	Ткт	Комментарий
PUSH qq	SP: = SP-2;(SP): = qq	.....	1	11	
PUSH ii	SP: = SP-2;(SP): = ii		2	15	
RES b, r	r < 2B-S0: = 0	.....	2	8	
RES b, (HL)	(HL) < 2B-B0: = 0		2	15	
RES b, (ii + n)	(ii + n) < 2B-B0: = 0		4	23	
RET	POP PC	.....	1	10	
RET cc	Если cc то RET	.....	1	11	
	иначе NOP			5	
RETI	Возврат из прерыв.	.....	2	14	
RETN	Возвр.из немас.прер.	.....	2	14	
RL r		**P*00	2	8	
RL (HL)			2	15	
RL (ii + n)			4	23	
RLA		*...00	1	4	
RLC r		**P*00	2	8	
RLC (HL)			2	15	
RLC (ii + n)			4	23	
RLCA		*...00	1	4	
RLD		*P*00	2	18	
RR r		**P*00	2	8	
RR (HL)			2	15	
RR (ii + n)			4	23	
RRA		*...00	1	4	
RRC r		**P*00	2	8	
RRC (HL)			2	15	
RRC (ii + n)			4	23	
RRCA		*...00	1	4	
RRD		.*P*00	2	18	
RST adr	CALL adr	.....	1	11	adr - байт (000xxx00b)
SBC A, r	A: = A-r-CY	**V*1*	1	4	

Таблица 2. Продолжение.

Мнемоника	Команда	CZPSNH	Длн	Ткт	Комментарий
SBC A, (HL)	A: = A-(HL)-CY		1	7	
SBC A, n	A: = A-n-CY		2	7	
SBC A, (ii + n)	A: = A-(ii + n)-CY		3	19	
SBC HL, rr	HL: = HL-rr-CY	**V*1x	2	15	
SCF	CY: = 1	1...00	1	4	
SET b, r	r2B-S0: = 1	.....	2	8	
SET b, (HL)	(HL)2B-S0: = 1		2	15	
SET b, (ii + n)	(ii + n)2B-S0: = 1		4	23	
SLA r		**P*00	2	8	
SLA (HL)			2	15	
SLA (ii + n)			4	23	
SRA r		**P*00	2	8	
SRA (HL)			2	15	
SRA (ii + n)			4	23	
SRL r		**P*00	2	8	
SRL (HL)			2	15	
SRL (ii + n)			4	23	
SUB r	A: = A-r	**V*0*	1	4	
SUB (HL)	A: = A-(HL)		1	7	
SUB n	A: = A-n		2	7	
SUB (ii + n)	A: = A-(ii + n)		3	19	
XOR r	A: = A xor r	0*P*00	1	4	
XOR (HL)	A: = A xor (HL)		1	7	
XOR n	A: = A xor n		2	7	
XOR (ii + n)	A: = A xor (ii + n)		3	19	

SYSTEM.SYS

Таблица 3. Коды команд Z80. Команды без префикса.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NOP	LD bc, nn	LD (bc), a	INC bc	INC b	DEC b	LD b, n	RLCA	EX af, af'	ADD hl, bc	LD a, (bc)	DEC bc	INC c	DEC c	LD c, n	RRCA
10	DJNZ dis	LD de, nn	LD (de), a	INC de	INC d	DEC d	LD d, n	RLA	JR dis	ADD hl, de	LD a, (de)	DEC de	INC e	DEC e	LD e, n	RRA
20	JR nz, dis	LD hl, nn	LD (nn)hl	INC hl	INC h	DEC h	LD h, n	DAA	JR z, dis	ADD hl, hl	LD hl(nn)	DEC hl	INC l	DEC l	LD l, n	CPL
30	JR nc, dis	LD sp, nn	LD (nn), a	INC sp	INC (hl)	DEC (hl)	LD (hl), n	SCF	JR c, dis	ADD hl, sp	LD a, (nn)	DEC sp	INC a	DEC a	LD a, n	CCF
40	LD b, b	LD b, c	LD b, d	LD b, e	LD b, h	LD b, l	LD b, (hl)	LD b, a	LD c, b	LD c, c	LD c, d	LD c, e	LD c, h	LD c, l	LD c, (hl)	LD c, a
50	LD d, b	LD d, c	LD d, d	LD d, e	LD d, h	LD d, l	LD d, (hl)	LD d, a	LD e, b	LD e, c	LD e, d	LD e, e	LD e, h	LD e, l	LD e, (hl)	LD e, a
60	LD h, b	LD h, c	LD h, d	LD h, e	LD h, h	LD h, l	LD h, (hl)	LD h, a	LD l, b	LD l, c	LD l, d	LD l, e	LD l, h	LD l, l	LD l, (hl)	LD l, a
70	LD (hl), b	LD (hl), c	LD (hl), d	LD (hl), e	LD (hl), h	LD (hl), l	HALT	LD (hl), a	LD a, b	LD a, c	LD a, d	LD a, e	LD a, h	LD a, l	LD a, (hl)	LD a, a

80	ADD a, b	ADD a, c	ADD a, d	ADD a, e	ADD a, h	ADD a, l	ADD a, (hl)	ADD a, a	ADC a, b	ADC a, c	ADC a, d	ADC a, e	ADC a, h	ADC a, l	ADC a, (hl)	ADC a, a
90	SUB b	SUB c	SUB d	SUB e	SUB h	SUB l	SUB (hl)	SUB a	SBC b	SBC c	SBC d	SBC e	SBC h	SBC l	SBC (hl)	SBC a
A0	AND b	AND c	AND d	AND e	AND h	AND l	AND (hl)	AND a	XOR b	XOR c	XOR d	XOR e	XOR h	XOR l	XOR (hl)	XOR a
B0	OR b	OR c	OR d	OR e	OR h	OR l	OR (hl)	OR a	CP b	CP c	CP d	CP e	CP h	CP l	CP (hl)	CP a
C0	RET nz	POP bc	JP nz, adr	JP adr	CALL nz, adr	PUSH bc	ADD a, n	RST 0	RET z	RET	JP z, adr	см табл.2	CALL z, adr	CALL adr	ADC a, n	RST 8
D0	RET nc	POP de	JP nc, adr	OUT (n), a	CALL nc, adr	PUSH de	SUB n	RST 10	RET c	EXX	JP c, adr	IN (n), a	CALL c, adr	см табл.4	SBC a, n	RST 18
E0	RET po	POP hl	JP po, adr	EX (sp)hl	CALL po, adr	PUSH hl	ADD n	RST 20	RET pe	JP (hl)	JP pe, adr	EX de, hl	CALL pe, adr	см табл.3	XOR n	RST 28
F0	RET pe	POP af	JP p, adr	DI	CALL p, adr	PUSH af	OR n	RST 30	RET m	LD sp, hl	JP m, adr	EI	CALL m, adr	см табл.4	CP n	RST 38

Таблица 3. Продолжение. Команды с префиксом CB.

CB	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	RLC b	RLC c	RLC d	RLC e	RLC h	RLC l	RLC (hl)	RLC a	RRC b	RRC c	RRC d	RRC e	RRC h	RRC l	RRC (hl)	RRC a
10	RL b	RL c	RL d	RL e	RL h	RL l	RL (hl)	RL a	RR b	RR c	RR d	RR e	RR h	RR l	RR (hl)	RR a
20	SLA b	SLA c	SLA d	SLA e	SLA h	SLA l	SLA (hl)	SLA a	SRA b	SRA c	SRA d	SRA e	SRA h	SRA l	SRA (hl)	SRA a
30							SLI (HL)	SLI a	SRL b	SRL c	SRL d	SRL e	SRL h	SRL l	SRL (hl)	SRL a
40	BIT 0, b	BIT 0, c	BIT 0, d	BIT 0, e	BIT 0, h	BIT 0, l	BIT 0, (hl)	BIT 0, a	BIT 1, b	BIT 1, c	BIT 1, d	BIT 1, e	BIT 1, h	BIT 1, l	BIT 1, (hl)	BIT 1, a
50	BIT 2, b	BIT 2, c	BIT 2, d	BIT 2, e	BIT 2, h	BIT 2, l	BIT 2, (hl)	BIT 2, a	BIT 3, b	BIT 3, c	BIT 3, d	BIT 3, e	BIT 3, h	BIT 3, l	BIT 3, (hl)	BIT 3, a
60	BIT 4, b	BIT 4, c	BIT 4, d	BIT 4, e	BIT 4, h	BIT 4, l	BIT 4, (hl)	BIT 4, a	BIT 5, b	BIT 5, c	BIT 5, d	BIT 5, e	BIT 5, h	BIT 5, l	BIT 5, (hl)	BIT 5, a
70	BIT 6, b	BIT 6, c	BIT 6, d	BIT 6, e	BIT 6, h	BIT 6, l	BIT 6, (hl)	BIT 6, a	BIT 7, b	BIT 7, c	BIT 7, d	BIT 7, e	BIT 7, h	BIT 7, l	BIT 7, (hl)	BIT 7, a

80	SET 0, b	SET 0, c	SET 0, d	SET 0, e	SET 0, h	SET 0, l	SET 0, (hl)	SET 0, a	SET 1, b	SET 1, c	SET 1, d	SET 1, e	SET 1, h	SET 1, l	SET 1, (hl)	SET 1, a
90	SET 2, b	SET 2, c	SET 2, d	SET 2, e	SET 2, h	SET 2, l	SET 2, (hl)	SET 2, a	SET 3, b	SET 3, c	SET 3, d	SET 3, e	SET 3, h	SET 3, l	SET 3, (hl)	SET 3, a
A0	SET 4, b	SET 4, c	SET 4, d	SET 4, e	SET 4, h	SET 4, l	SET 4, (hl)	SET 4, a	SET 5, b	SET 5, c	SET 5, d	SET 5, e	SET 5, h	SET 5, l	SET 5, (hl)	SET 5, a
B0	SET 6, b	SET 6, c	SET 6, d	SET 6. e	SET 6, h	SET 6, l	SET 6, (hl)	SET 6, a	SET 7, b	SET 7, c	SET 7, d	SET 7, e	SET 7, h	SET 7, l	SET 7, (hl)	SET 7, a
C0	RES 0, b	RES 0, c	RES 0, d	RES 0, e	RES 0, h	RES 0, l	RES 0, (hl)	RES 0, a	RES 1, b	RES 1, c	RES 1, d	RES 1, e	RES 1, h	RES 1, l	RES 1, (hl)	RES 1, a
D0	RES 2, b	RES 2, c	RES 2, d	RES 2, e	RES 2, h	RES 2, l	RES 2, (hl)	RES 2, a	RES 3, b	RES 3, c	RES 3, d	RES 3, e	RES 3, h	RES 3, l	RES 3, (hl)	RES 3, a
E0	RES 4, b	RES 4, c	RES 4, d	RES 4, e	RES 4, h	RES 4, l	RES 4, (hl)	RES 4, a	RES 5, b	RES 5, c	RES 5, d	RES 5, e	RES 5, h	RES 5, l	RES 5, (hl)	RES 5, a
F0	RES 6, b	RES 6, c	RES 6, d	RES 6, e	RES 6, h	RES 6, l	RES 6, (hl)	RES 6, a	RES 7, b	RES 7, c	RES 7, d	RES 7, e	RES 7, h	RES 7, l	RES 7, (hl)	RES 7, a

Таблица 3. Продолжение. Команды с префиксом ED.

ED	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00																
10																
20																
30																
40	IN b, (c)	OUT (c), b	SBC hl, bc	LD (nn)bc	NEG	RETN	IM 0	LD i, a	IN c, (c)	OUT (c), c	ADC hl, bc	LD bc(nn)		RETI		LD r, a
50	IN d, (c)	OUT (c), d	SBC hl, de	LD (nn)de			IM 1		IN e, (c)	OUT (c), e	ADC hl, de	LD de(nn)			IM 2	
60	IN h, (c)	OUT (c), h	SBC hl, hl					RRD	IN l, (c)	OUT (c), l	ADC hl, hl			RLD		
70			SBC hl, sp	LD (nn)sp					IN a, (c)	OUT (c), a	ADC hl, sp	LD sp(nn)				

80																
90																
A0	LDI	CPI	INI	OUTI					LDD	CPD	IND	OUTD				
B0	LDIR	CPIR	INIR	OTIR					LDDR	CPDR	INDR	OTDR				
C0																
D0																
E0																
F0																

В таблицах команд без префикса и с префиксом CB допускается замена операнда HL на операнды IX либо IY. В этом случае возможны варианты:

1. Использование косвенной адресации в основной таблице. Операнд (HL) преобразуется в (IX + смещение) или (IY + смещение). Код:

Префикс DD / FD
Код команды
Смещение

2. Непосредственное использование регитстра HL. Операнд HL преобразуется с IX или IY. Код:

Префикс DD / FD
Код команды

3. Использование косвенной адресации в таблице команд с префиксом CB. Операнд (HL) преобразуется в (IX + смещение) или (IY + смещение). Код:

Префикс DD / FD
CB
Смещение
Код команды

Таблица 4. Мнемоники и коды команд Z80 и эквивалентные команды 8080A в алфавитном порядке.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
ADC DATA	CE YY	2	7	ACI DATA	7
ADC (HL)	8E	1	7	ADC M	7
ADC HL, RP	ED 01xx1010	2	15		
ADC (IX + DISP)	DD 8E YY	3	19		
ADC (IY + DISP)	FD 8E YY	3	19		
ADC REG	10001xxx	1	4	ADC REG	4
ADD ~~DATA~~ A, DATA	C6 YY	2	7	ADI DATA	7
ADD (HL)	86	1	7	ADD M	7
ADD HL, RP	00xx1001	1	11	*DAD RP	10
ADD (IX + DISP)	DD 86 YY	3	19		
ADD IX, RP	DD 00xx1001	2	15		
ADD (IY + DISP)	FD 86 YY	3	19		
ADD IY, RP	FD00xx1001	2	15		
ADD REG A, REG	10000xxx	1	4	ADD REG	4
AND DATA	E6 YY	2	7	ANI DATA	7
AND (HL)	A6	1	7	ANA M	7
AND (IX + DISP)	DD A6 YY	3	19		
AND (IY + DISP)	FD A6 YY	3	19		
AND REG	10100xxx	1	4	ANA REG	4
BIT B, (HL)	CB 01bbb110	2	12		
BIT B, (IX + DISP)	DD CB YY kod	4	20		
BIT B, (IY + DISP)	FD CB YY kod	4	20		
BIT B, REG	CB 01bbbxxx	2	8		
CALL LABEL	CD ppqq	3	17	CALL LABEL	17
CALL C, LABEL	DC ppqq	3	0/17	*CC LABEL	11/17
CALL M, LABEL	FC ppqq	3	10/17	*CM LABEL	11/17
CALL NC, LABEL	D4 ppqq	3	10/17	*CNC LABEL	11/17
CALL NZ, LABEL	C4 ppqq	3	10/17	*CNZ LABEL	11/17
CALL P, LABEL	F4 ppqq	3	10/17	*CP LABEL	11/17
CALL PE, LABEL	EC ppqq	3	10/17	*CPE LABEL	11/17
CALL PO, LABEL	E4 ppqq	3	10/17	*CPO LABEL	11/17
CALL Z, LABEL	CC ppqq	3	10/17	*CZLABEL	11/17
CP DATA	FEYY	2	7	CPI DATA	7
CP (HL)	BE	1	7	CMP M	7
CP (IX + DISP)	DD BE YY	3	19		

Таблица 4. Продолжение.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
CP (IY + DISP)	FD BE YY	3	19		
CP REG	10111xxx	1	4	CMP REG	4
CPDR	ED B9	2	21/16		
CPI	ED A1	2	16		
CPIR	ED B1	2	21/16		
CPL	2F	1	4	CMA	4
DAA	27	1	4	DAA	4
DEC (HL)	35	1	11	*DCR M	10
DEC IX	DD 2B	2	10		
DEC (1X + DISP)	DD 35 YY	3	23		
DEC IY	FD 2B	2	10		
DEC (IY + DISP)	FD 35 YY	3	23		
DEC RP	00xx1011	1	6	*DCX RP	5
DEC REG	00xx1011	1	4	*DCR REG	5
DI	F3	1	4	DI	4
DJNZ DISP	10 YY	2	8/13		
EI	FB	1	4	EI	4
EX AF, AF'	08	1	4		
EX DE, HL	EB	1	4	XCHG	4
EX (SP), HL	E3	1	19	*XTHL	18
EX (SP), IX	DD E3	2	23		
EX (SP), IY	FD E3	2	23		
EXX	D9	1	4		
HALT	76	1	4	HLT	4
IM 0	ED 46	2	8		
IM 1	ED 56	2	8		
IM 2	ED 5E	2	8		
IN A, PORT	DB YY	2	11	IN PORT	10
IN REG, (C)	ED 01ddd000	2	12		
INC (HL)	34	1	11	*INR M	10
INC IX	DD 23	2	10		
INC (IX + DISP)	DD 34 YY	3	23		
INC IY	FD 23	2	10		
INC (IY + DISP)	FD 34 YY	3	23		
INC RP	00xx0011	1	6	*INX RP	5
INC REG	00xxx100	1	4	*INR REG	5

Таблица 4. Продолжение.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
IND	ED AA	2	15		
INDR	ED BA	2	20/15		
INI	ED A2	2	15		
INIR	ED B2	2	20/15		
JP LABEL	C3 ppqq	3	10	JMP LABEL	10
JP C, LABEL	DA ppqq	3	10	JC LABEL	10
JP (HL)	E9	1	4	*PCHL	5
JP (IX)	DD E9	2	8		
JP (IY)	FD E9	2	8		
JP M, LABEL	FA ppqq	3	10	JM LABEL	10
JP NC, LABEL	D2 ppqq	3	10	JNC LABEL	10
JP NZ, LABEL	C2 ppqq	3	10	JNZ LABEL	10
JP P, LABEL	F2 ppqq	3	10	JP LABEL	10
JP PE, LABEL	EA ppqq	3	10	JPE LABEL	10
JP PO, LABEL	E2 ppqq	3	10	JPO LABEL	10
JP Z, LABEL	CA ppqq	3	10	JZ LABEL	10
JR C, DISP	38 YY	2	7/12		
JR DISP	18 YY	2	12		
JR NC, DISP	30 YY	2	7/12		
JR NZ, DISP	20 YY	2	7/12		
JR Z, DISP	28 YY	2	7/12		
LD A, (ADDR)	3A ppqq	3	13	LDA ADDR	13
LD A, (BC)	0A	1	7	LDAX B	7
LD A, (DE)	1A	1	7	LDAX D	7
LD A, I	ED 57	2	9		
LD A, R	ED 5F	2	9		
LD (ADDR), A	32 ppqq	3	13	STA ADDR	13
LD (ADDR), HL	22 ppqq	3	16	SHLD ADDR	16
LD (ADDR), IX	ED 22 ppqq	4	20		
LD (ADDR), IY	FD 22 ppqq	4	20		
LD (ADDR), SP	ED 53 ppqq	4	20		
LD (BC), A	02	1	7	STAX B	7
LD (DE), A	12	1	7	STAX B	7
LD HL, (ADDR)	2A ppqq	3	16	LHLD ADDR	16
LD (HL), DATA	36 YY	2	10	MVI M, DATA	10
LD (HL), REG	01110sss	1	7	MOV M, REG	7

Таблица 4. Продолжение.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
LD I, A	ED 47	2	9		
LD IX, (ADDR)	DD 2A ppqq	4	20		
LD IX, DATA16	DD 21 YYYY	4	14		
LD (IX + DISP), DATA	DD 36 YY YY	4	19		
LD (IX + DISP), REG	DD 01110sss YY	3	19		
LD IY, (ADDR)	FD 2A ppqq	4	20		
LD IY, DATA16	FD 21 YYYY	4	14		
LD (IY + DISP), DATA	FD 36 YYYY	4	19		
LD (IY + DISP), REG	FD 01110sss YY	3	19		
LD R, A	ED 4F	2	9		
LD REG, DATA	00ddd110 YY	2	7	MVI REG, DATA	7
LD REG, (HL)	01ddd110	1	7	MOV REG, M	7
LD REG, (IX + DISP)	DD 01ddd110 YY	3	19		
LD REG, (IY + DISP)	FD 01ddd110 YY	3	19		
LD REG, REG	00ddd110	1	4	*MOV REG, REG	5
LD RP, (ADDR)	ED 01xx1011 ppqq	4	20		
LD RP, DATA16	00xx0001 YYYY	3	10	LXI RP, DATA16	10
LD SP, HL	F9	1	6	*SPHL	5
LD SP, IX	DD F9	2	10		
LD SP, IY	FD F9	2	10		
LDD	ED A8	2	16		
LDDR	ED B8	2	21/16		
LDI	ED A0	2	16		
LDIR	ED B0	2	21/16		
NEG	ED 44	2	8		
NOP	00	1	4	NOP	4
OR DATA	F6 YY	2	7	ORI DATA	7

Таблица 4. Продолжение.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
OR (HL)	B6	1	7	ORA M	7
OR (IX + DISP)	DD B6 YY	3	19		
OR (IY + DISP)	FD B6 YY	3	19		
OR REG	10110xxx	1	4	*ORA REG	5
OUT (C), REG	ED 01sss001	2	12		
OUT PORT, A	D3 YY	2	11	*OUT PORT	10
OUTD	ED A8	2	15		
OUTDR	ED BB	2	20/15		
OUTI	ED A3	2	15		
OUTIR	ED B3	2	20/15		
POP IX	DD E1	2	14		
POP IY	FD E1	2	14		
POP RP	11xx000	1	10	POP RP	10
PUSH IX	DD E5	2	15		
PUSH IY	FD E5	2	15		
PUSH RP	11xx0101	1	11	PUSH RP	11
RES B, (HL)	CB 10bbb110	2	15		
RES B, (IX + DISP)	DD CB YY 10bbb110	4	23		
RES B, (IY + DISP)	FD CB YY 10bbb110	4	23		
RES B, REG	CB 10bbbxxx	2	8		
RET	C9	1	10	RET	10
RET C	D9	1	5/11	RC	5/11
RET M	F8	1	5/11	RM	5/11
RET NC	D0	1	5/11	RNC	5/11
RET NZ	C0	1	5/11	RNZ	5/11
RET P	F0	1	5/11	RP	5/11
RET PE	E8	1	5/11	RPE	5/11
RET PO	E0	1	5/11	RPO	5/11
RET Z	C8	1	5/11	RZ	5/11
RETI	ED 4D	2	14		
RETN	ED 45	2	14		
RL (HL)	CB 16	2	15		
RL (IX + DISP)	DD CB YY 16	4	23		
RL (IY + DISP)	FD CB YY 16	4	23		

Таблица 4. Продолжение.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
RL REG	CB 00010xxx	2	8		
RLA	17	1	4	RAL	4
RLC (HL)	CB 06	2	15		
RLC (IX + DISP)	DD CB YY 06	4	23		
RLC (IY + DISP)	FD CB YY 06	4	23		
RLC REG	CB 00000xxx	2	8		
RLCA	07	1	4	RLC	4
RLD	ED 6F	2	18		
RR (HL)	CB 1E	2	15		
RR (IX + DISP)	DD CB YY 1E	4	23		
RR (1Y + DISP)	FD CB YY 1E	4	23		
RR REG	CB 00011xxx	2	8		
RRA	1F	1	4	RAR	4
RRC (HL)	CB 0E	2	15		
RRC (IX + DISP)	DD CB YY 0E	4	23		
RRC (IY + DISP)	FD CB YY 0E	4	23		
RRC REG	CB 00001xxx	2	8		
RRCA	0F	1	4	RRC	4
RRD	ED 67	2	18		
RST N	11xxx111	1	11	RST N	11
SBC DATA	DE YY	2	7	SBI DATA	7
SBC (HL)	9E	1	7	SBB M	7
SBC HL, RP	ED 01xx0010	2	15		
SBC (IX + DISP)	DD 9E YY	3	19		
SBC (IY + DISP)	FD 9E YY	3	19		
SBC REG	10011xxx	1	4	SBB REG	4
SCF	37	1	4	STC	4
SET B, (HL)	CB 11bbb110	2	15		
SET B, (IX + DISP)	DD CB YY 11bbb110	4	23		
SET B, (IY + DISP)	FD CB YY 11bbb110	4	23		
SET B, REG	CB 11bbbxxx	2	8		
SLA (HL)	CB26	2	15		
SLA (IX + DISP)	DD CB YY 26	4	23		
SLA (IY + DISP)	FD CB YY 26	4	23		

Таблица 4. Продолжение.

Z80				8080A	
Мнемоника	Код	Длн	Ткт	Мнемоника	Ткт
SLA REG	CB 00100xxx	2	8		
SRA (HL)	CB 2E	2	15		
SRA (IX + DISP)	DD CB YY 2E	4	23		
SRA (IY + DISP)	FD CB YY 2E	4	23		
SRA REG	CB 00101xxx	2	8		
SRL (HL)	CB 3E	2	15		
SRL (IX + DISP)	DD CB YY 3E	4	23		
SRL (IY + DISP)	FD CB YY 3E	4	23		
SRL REG	CB 00111xxx	2	8		
SUB DATA	D6 YY	2	7	SUI DATA	7
SUB (HL)	96	1	7	SUB M	7
SUB (IX + DISP)	DD 96 YY	3	19		
SUB (IY + DISP)	FD 96 YY	3	19		
SUB REG	10010xxx	1	4	SUB REG	4
XOR DATA	EE YY	2	7	XRI DATA	7
XOR (HL)	AE	1	7	XRA M	7
XOR (IX + DISP)	DD AE YY	3	19		
XOR (IY + DISP)	FD AE YY	3	19		
XOR REG	10101xxx	1	4	XRA REG	4

Обозначения в таблице 4:

x - двоичная цифра;
bbb - двоичные цифры, определяющие позицию бита в регистре или байте памяти;
ddd - двоичные цифры, определяющие регистр результата;
sss - двоичные цифры, определяющие регистр источника;
pppq - шестнадцатиричные цифры, определяющие адрес в памяти;
YY - две произвольные шестнадцатиричные цифры;
YYYY - четыре произвольные шестнадцатиричные цифры.

Там, где показаны два возможных количества тактов выполнения (например, 5/11), длительность выполнения зависит от состояния флагов.

Звездочками отмечены команды, эквивалентные по результату, но отличающиеся длительностью выполнения.

Для групповых команд указана длительность одной итерации.

ОГЛАВЛЕНИЕ

В СЕРИИ
«В помощь разработчику микропроцессорной техники»
готовится следующий выпуск:

«Микросхемы и модули запоминающих устройств для персональных ЭВМ»

В выпуске рассмотрены характеристики, цоколевки, временные диаграммы и условия применения микросхем ПЗУ с ультрафиолетовым стиранием, динамических ОЗУ, SIP и SIMM модулей, применяемых в персональных ЭВМ зарубежного производства. Объем приведенных данных достаточен для самостоятельной разработки устройств памяти на их основе.

С предварительными заказами обращайтесь по адресу:
125057, Москва, ул. Новопесчаная, 8 корп. 3, МГНПП ЭЛИС.