


МИКРОПРОЦЕССОРЫ
КОМПЛЕКТ
Z80


книга

Z80 CENTRAL
PROCESSOR
UNIT

Справочное пособие



**PDF version by Deny (Денисенко Д.А.)
e-mail: DenyDA@mail.ru
2007**



ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР Z80CPU

1

Краткая характеристика МП Z80

2

Архитектура ЦП

3

Описание выводов

4

Временные диаграммы машинных циклов

5

Система команд

6

Система прерываний

7

Технические характеристики

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ	4
ВВЕДЕНИЕ	5
1. КРАТКАЯ ХАРАКТЕРИСТИКА МИКРОПРОЦЕССОРА Z80	6
2. АРХИТЕКТУРА ЦП	7
2.1. Регистры	7
2.2. Арифметическо-логическое устройство (АЛУ)	10
2.3. Регистр команд и устройство управление процессором	10
2.4. Управление шинами адреса и данных	10
3. ОПИСАНИЕ ВЫВОДОВ	11
4. ВРЕМЕННЫЕ ДИАГРАММЫ МАШИННЫХ ЦИКЛОВ	14
4.1. Извлечение кода операции	14
4.2. Цикл чтения памяти и цикл записи в память	15
4.3. Циклы ввода/вывода	16
4.4. Цикл предоставления доступа к шине	17
4.5. Цикл подтверждения маскируемого прерывания	18
4.6. Цикл подтверждения немаскируемого прерывания	20
4.7. Выполнение команды останова	20
5. СИСТЕМА КОМАНД	21
5.1. Методы адресации	21
5.2. Группы команд	23
5.3. Флаги признаков	35
5.4. Очередность выполнения по циклам	37
6. СИСТЕМА ПРЕРЫВАНИЙ	41
6.1. Разрешение и запрещение прерываний	41
6.2. Приём запросов в ЦП	43
6.3. Обработка прерываний	45
6.3.1. Обработка немаскируемого прерывания	45
6.3.2. Маскируемое прерывание. Режим 0	47
6.3.3. Маскируемое прерывание. Режим 1	49
6.3.4. Маскируемое прерывание. Режим 2	51
7. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ	55
7.1. Схемы входных и выходных каскадов	55
7.2. Электрические и временные параметры	56
7.2.1. Статические параметры Z80, Z80A	56
7.2.2. Динамические параметры Z80	57
7.2.3. Динамические параметры Z80A	58
7.2.4. Предельные значения	62
7.3. Надежность	62
ПРИЛОЖЕНИЕ А. КОДОВЫЕ ТАБЛИЦЫ КОМАНД	63
ПРИЛОЖЕНИЕ Б. СООТВЕТСТВИЕ МНЕМОНИК АССЕМБЛЕРА Z80 И I8080	70
ЛИТЕРАТУРА	73

ПРЕДИСЛОВИЕ

Предлагаемая книга является первой из серии "Микропроцессорный комплект Z80". Она посвящена главной БИС комплекта - центральному процессору Z80CPU и представляет собой справочник с расширенной описательной частью.

Книга содержит 7 глав.

Первая - это краткая характеристика МП Z80. В главе 2 рассматривается общая архитектура МП и назначение её составных частей. Особое внимание уделено программно-доступным регистрам. Глава 3 содержит описание выводов МП. В главе 4 с помощью временных диаграмм анализируются процессы на шинах при выполнении всех машинных циклов. Глава 5 посвящена системе команд. Для лучшего понимания назначения команд они разбиты по функциональному признаку. Рассматривается порядок выполнения команд по циклам и влияние их на состояние флагов. В шестой главе рассмотрены прерывания МП Z80. Подробно отражены вопросы разрешения/запрещения прерываний, процедура приёма запросов в ЦП, работа МП в различных режимах прерываний. Временные и электрические параметры БИС даны в главе 7. Для удобства программирования в приложениях книги приведена система команд в виде кодовых таблиц. Таблица соответствия мнемочкодов Ассемблера для идентичных команд микропроцессоров Z80 и I8080 поможет тем, кто имеет опыт программирования для КР580ВМ80.

Цель написания данной книги - дать разработчикам и пользователям вычислительной техники исчерпывающую информацию о микропроцессоре Z80. Знакомство с ней, мы надеемся, будет полезно как опытным инженерам в проектировании, ремонте и наладке оборудования, так и любителям, постигающим компьютерную технику и азы программирования с помощью популярных сейчас бытовых компьютеров.

Выражаем признательность кандидату технических наук А.Н. Цырульникову за внимательное изучение рукописи и полезные, предложения которые способствовали улучшению материала книги и формы его изложения. Будем благодарны читателям за все отзывы, пожелания и предложения, которые следует присылать по адресу: 220008 Минск, а/я-103.

ВВЕДЕНИЕ

Микропроцессор Z80 был разработан в 1976 году фирмой Zilog (США), учрежденной специалистами, принимавшими ранее самое непосредственное участие в разработках пионерских микропроцессоров фирмы Intel. Оптимальное сочетание аппаратных и программных достижений того времени предопределило его широкое распространение. Теперь уже можно сказать, что Z80 - это вершина восьмиразрядных микропроцессоров.

Впоследствии фирма Zilog разработала целый комплект БИС для построения микрокомпьютерных систем на базе своего МП. В комплект входят:

Z80CPU Central Processor Unit - центральный процессор;

Z80PIO Parallel Input. Output - интерфейс параллельного ввода/вывода;

Z80CTC Counter Timer Circuit - счетчик/таймер;

Z80SIO Serial Input Output - интерфейс последовательного ввода/вывода;

Z80DMA Direct Memory Access - контроллер прямого доступа к памяти;

Z80DART Dual Asynchronous Receiver/Transmitter двухканальный асинхронный приемопередатчик.

На базе этого комплекта созданы микрокомпьютерные системы для управления широким классом технологического оборудования: от станков с ЧПУ до химических установок, встроенные системы управления (от автомобилей до бытовых приборов), медицинская аппаратура и, конечно, персональные компьютеры, и периферийные устройства к ним (принтеры, графопостроители и пр.).

К настоящему времени многими фирмами разработаны аппаратно-программные средства поддержки проектирования и отладки систем, построенных на базе Z80.

Существует несколько вариантов микропроцессора: Z80, Z80A, Z80B и Z80H (high speed), которые имеют максимальную тактовую частоту 2,5, 4, 6 и 8 МГц соответственно. Помимо обозначения Z80CPU, характеризующего фирменную принадлежность и функциональное назначение БИС, на её корпусе также указывается стандартная маркировка Z8400.

Микропроцессор Z80L (low power) предназначен для использования в системах с аккумуляторным питанием. Он характеризуется пониженным потреблением мощности и имеет две разновидности: Z8300-1 - 1МГц, 15мА и Z8300-3 - 2.6МГц 25мА.

Материал данной брошюры в равной мере касается всех типов микропроцессора, за исключением гл. 7. В ней приведены технические параметры двух наиболее часто используемых МП Z80 и Z80A.

1. Краткая характеристика микропроцессора Z80

МП Z80 представляет собой БИС с 8500 транзисторами на Кристаллической пластине площадью 4,6*4,9 мм² и выпускается в DIP корпусе с 40 выводами. БИС выполнена по n-канальной МОП технологии с кремниевыми затворами и работает от одного источника питания +5В. Все входы и выходы микросхемы ТТЛ-совместимы

МП Z80 предназначен для работы с памятью (постоянной и оперативной) с общей емкостью до 64К. Память имеет байтовую структуру - возможна адресация в памяти любого байта. Ширина выборки из памяти - 1 байт. При обращении к памяти используются 16-разрядные (двухбайтные) адреса.

Организация МП Z80 отмечена следующими основными особенностями:

- трехшинной структурой с шинами адреса, данных и управления;
- наличием регистровой памяти, образованной программно доступными и общими и специализированными регистрами, а также регистрами временного хранения;
- наличием двух (главного и вспомогательного) аккумуляторов. Флаговых регистров и наборов РОН;
- магнетральным принципом связей, реализованным в виде связывающей основные узлы МП двунаправленной шины данных, имеющей ширину, равную длине слов, обрабатываемых микропроцессором (8 разрядов);
- наличием 16-разрядной шины адреса, обеспечивавшей возможность прямой адресации любого байта в памяти емкостью 64 Кбайт;
- наличием 10 способов адресации: непосредственная, регистровая, косвенная, абсолютная, модифицированная нуль-страничная, относительная, индексная, битовая, встроенная и смешиваемая;
- расширенным набором команд (158 базовых команд для работы с 16-, 8-, 4- и однобитными данными);
- наличием четырех форматов команд (1-, 2-, 3- и 4-байтного);
- наличием средств для работы с подпрограммами: команды вызова и возврата, с тон числе условного;
- наличием средств организации стековой памяти (регистр - указатель стека, схемы дополнения операций инкремента декремента, специальные команды стековых операций);
- наличием эффективных средств обработки массивов данных: пересылки, сравнения и ввода/вывода блоков;
- развитой системой прерываний: возможна реализация векторных многоуровневых приоритетных прерываний без подключения БИС контроллера прерываний. Имеются 3 программно выбираемых режима маскируемого прерывания, а также немаскируемое прерывание;
- возможностью реализации в МП режима прямого доступа к памяти путем подключения специальной БИС (контроллера ПДП);
- упрощенными схемами интерфейса - отпадает необходимость в дополнительных БИС, таких как, например, генератор тактовых импульсов и системный контроллер для МП I8080;
- наличием встроенной схемы регенерации динамического ОЗУ.

Программное обеспечение МП совместимо с программной частью МП Intel 8080. Набор команд Z80, по существу, является расширенным набором команд I8080, поэтому МП Z80 может выполнять программы, написанные для I8080.

2. Архитектура ЦП

Архитектуре МП Z80 является типичной для 8-разрядных микропроцессоров. В ней можно выделить следующие основные части:

- блок регистров,
- арифметическо-логическое устройство,
- регистр команд,
- дешифратор команд и устройство управления,
- схемы управления шинами адреса и данных.

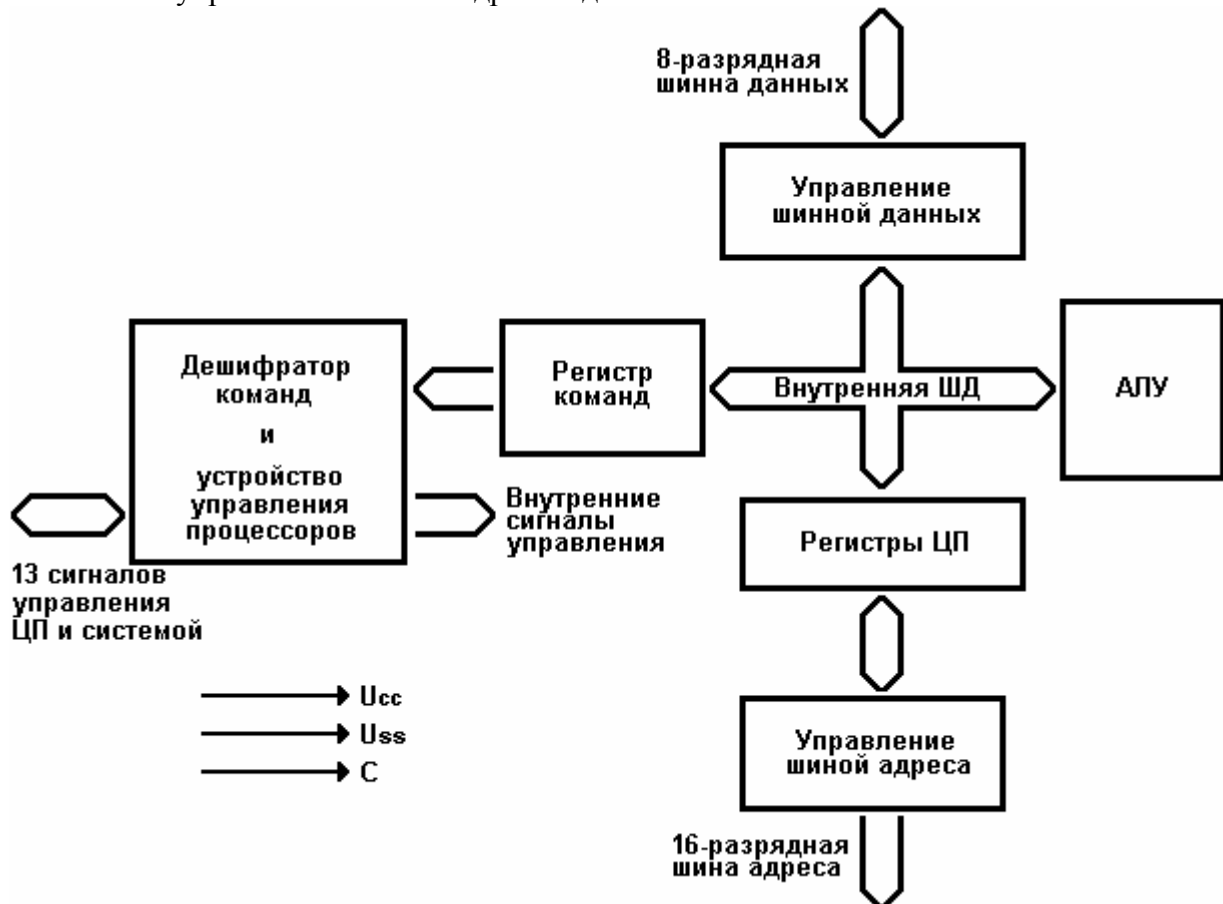


Рис. 2.1. Блок-схема ЦП Z80

2.1. Регистры

Программно доступными средствами ЦП являются 22 внутренних восьми- и шестнадцатиразрядных регистра. Они образуют три блока: 2 альтернативных блока (главный и вспомогательный) - по шесть 8-разрядных регистров, аккумулятору и регистру флагов в каждом, и блок специальных регистров.



Рис. 2.2. Регистры ЦП Z80

Специальные регистры

PC Program Counter - Счётчик команд

Счётчик команд является программно доступным регистром, и используется для приёма, преобразования и выдачи на шину адреса текущего 16-разрядного адреса команды. Содержимое счётчика команд автоматически инкрементируется после выборки каждого байта команды. В случае перехода в программе, новый адрес автоматически заносится в счётчик команд.

SP Btack Pointer - Указатель стека

Указатель содержит 16-разрядный адрес ячейки стека, к которой было последнее обращение. Содержимое SP декрементируется, когда данные загружаются в стек и инкрементируется при чтении. Стек организуется во внешнем ОЗУ по принципу LIFO (Last In - First Out). Обмен данными между стеком и ЦП может быть автоматическим (как в случае обработки подпрограмм), либо командами PUSH и POP. Стек позволяет простую реализацию многоуровневых прерываний, практически неограниченное вложение подпрограмм и упрощение при многих видах обработки данных.

IX, IY Indexregisters - Индексные регистры

Каждый из этих двух регистров может содержать 16-разрядный базовый адрес, используемый при индексной адресации. Базовый адрес складывается со смещением, которое указано в команде в дополнительном коде. Их сумма образует действительный адрес ячейки памяти, содержащей данные. Этот вид адресации удобен при обработке таблиц и массивов.

I Interruptveotor register - Регистр вектора прерывания

Это 8-разрядный программно доступный регистр, используемый в режиме прерывания 2 (IM2). Он содержит старший байт вектора прерывания. Младший байт принимается от внешнего устройства. Путём их объединения формируется полный вектор прерывания. (Подробнее см. гл. 6).

R Refreshregister - Регистр регенерации памяти

Это 8-разрядный программно доступный регистр, обеспечивавший возможность использования динамических ОЗУ без внешних схем регенерации. Содержимое его младших 7 разрядов автоматически увеличивается на единицу после каждой выборки команды. При этом восьмой бит сохраняет значение, полученное при выполнении команды загрузки этого регистра. Т.о. формируется адрес регенерации, который подается в младшую часть адресной шины во время декодирования и выполнения команды в ЦП (в старшую часть - содержимое регистра I). Регенерация данного вида называется "прозрачной" и не снижает быстродействия процессора.

Главный и вспомогательный блоки регистров

ЦП Z80 содержит два альтернативных блока регистров: главный (A-L) и вспомогательный (A'-L'). С точки зрения программиста оба блока абсолютно равноправны, но в данный момент можно работать лишь с одним из них. Переключение этих блоков производится командами EXX и EX AF.AF'. Такая организация удобна тем, что позволяет быстро сохранить содержимое регистровых блоков при вызове подпрограмм или при возникновении прерывания. Однако следует помнить: отсутствует средство подтверждения, какой из блоков (главный или вспомогательный) используется в данный момент.

A, A' Accumulator - Аккумулятор

Каждый из двух блоков регистров содержит по одному 8-разрядному аккумулятору. При арифметических и логических операциях он служит источником одного из операндов и, как правило, приёмником результата. Второй операнд берётся из другого регистра, либо из памяти.

Диапазон представления целых чисел без знака в аккумуляторе от 0 до 255, со знаком от -128 до +127.

F, F' Flagregister - Флаговый регистр

В составе каждого блока регистров имеется свой флаговый регистр. Флаговый регистр (называемый также регистром условий) содержит набор одноразрядных признаков, которые устанавливаются по результату операции. Флаговые биты 7, 6, 2, 0 служат для реализации условных переходов и условных вызовов подпрограмм или возвратов; биты 4 и 1 служат для реализации двоично-десятичной арифметики.



Рис. 2.3. Регистр флагов

Флаги устанавливаются в следующих условиях:

S=1, если результат операции отрицателен.

Z=1, если результат операции равен нулю.

H=1, если при арифметической операции был перенос между битами 3 и 4.

P/V=1, а). если при логических операциях и командах сдвига количество установленных в единицу битов чётно (функция четности P); б). если результат арифметической операции находится вне диапазона представления чисел со знаком, т.е. меньше -128, либо больше +127 (функция переполнения V).

N=1, если в предыдущей команде выполнялось вычитание (команды типа SUB, DEC, CMP).

C=1, если при сложении возникает перенос из 7 бита аккумулятора, либо при вычитании - заём. В командах сдвига состояние флага C однозначно соответствует сдвигаемому в него биту.

B, C, D, E, H, L Регистры общего назначения (РОН)

B', C', D', E', H', L'

Могут быть использованы как накопители данных или указатели адресов операндов. РОНЫ могут использоваться как самостоятельные 8-разрядные регистры, либо как 16-разрядные попарно: BC, DE, HL и BC', DE', HL'.

2.2. Арифметическо-логическое устройство (АЛУ)

В АЛУ выполняются арифметические и логические действия над 8-разрядными операндами. Внутренне АЛУ связано с регистрами и через внутреннюю шину данных с внешней шиной. В АЛУ выполняются следующие операции:

- сложение;
- вычитание
- логическое И;
- логическое ИЛИ;
- логическое исключающее ИЛИ;
- сравнение;
- увеличение на единицу;
- уменьшение на единицу;
- установка и сброс бита;
- анализ значения бита;
- сдвиг влево и вправо (арифметический и логический);
- вращение влево и вправо (циклический сдвиг).

2.3. Регистр команд и устройство управление процессором

После извлечения команды из памяти, она загружается в регистр команд. Дешифратор команд, входящий в устройство управления, преобразует код команды в управляющие сигналы:

- внутренние, необходимые для считывания/записи данных в регистры и управления АЛУ,

- внешние, подаваемые на шину управления.

Кроме того, устройство управления реагирует на внешние управляющие сигналы.

2.4. Управление шиной адреса и данных

Блок управления ША состоит из регистра адреса и буфера адреса. Буфер адреса представляет собой выходные формирователи с тремя устойчивыми состояниями. Он предназначен для выдачи 16-разрядного адреса из регистра адреса на шину.

Блок управления ШД представляет собой бинаправленную трёхстабильную схему, применяемую для обмена информацией ЦП с внешними устройствами. При выводе информации содержимое внутренней ШД запоминается в 8-разрядном регистре и через выходные формирователи выдается на внешнюю шину данных.

3. Описание выводов

Микросхема Z80 выпускается в стандартном 40 выводном корпусе с двухрядным расположением выводов типа DIP.

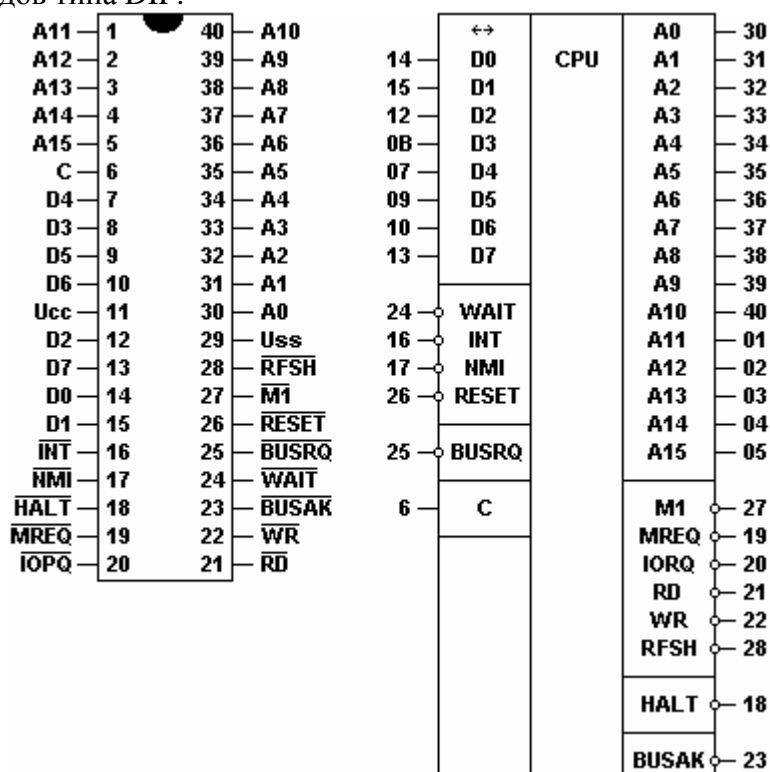


Рис. 3.1. Распределение выводов и условное графическое изображение

A0...A15 Address Bus - Адресная шина

Трёхстабильный выход. Активный уровень - высокий. A0-A15 образуют 16-разрядную адресную шину, которая выдает адреса для обмена данными с памятью (64К максимум) и с устройствами ввода-вывода (65536 каналов максимум). A0 является самым младшим адресным битом. Во время регенерации ОЗУ 7 младших битов содержат действительный адрес регенерации.

D0...D7 Data Bus - Шина данных

Трёхстабильный вход-выход. Активный уровень - высокий. D0-D7 образуют 8-разрядную двунаправленную шину данных, по которой осуществляется обмен данными между ЦП и памятью, либо между ЦП и устройствами ввода-вывода.

M1 Machine Cycle 1 - Машинный цикл 1

Трёхстабильный выход. Активный уровень - низкий. /M1 указывает, что в текущей машинном цикле происходит чтение кода операции из памяти. При считывании кода операции вида CB, ED, DD, FD вырабатывается ещё один цикл M1 для считывания второго байта кода операции, т.е. сигнал /M1 активизируется дважды.

/M1 также активизируется вместе с сигналом /IORQ в цикле подтверждения прерывания.

MREQ Memory Request - Запрос памяти

Трёхстабильный выход. Активный уровень - низкий. Сигнал запроса памяти указывает системе, что на адресной шине установлен адрес для операции чтения памяти или записи в память.

IORQ Input/Output Request - Запрос ввода-вывода

Трёхстабильный выход. Активный уровень - низкий. Сигнал /IORQ указывает, что пика адреса содержит адрес внешнего устройства для операции ввода или вывода. Кроме того, сигнал IORQ генерируется также совместно с сигналом /M1 в цикле подтверждения прерывания. Тем самым устройству, запросившему прерывание, указывается, что вектор прерывания может быть помещен на шину данных.

- RD** Reed - Чтение
Трехстабильный выход. Активный уровень - низкий. Сигнал /RD указывает, что ЦП выполняет цикл чтения данных из памяти или устройства ввода-вывода. Адресованное устройство ввода-вывода или память должны использовать этот сигнал для стробирования подачи данных на шину данных.
- WR** Write - Запись
Трехстабильный выход. Активный уровень – низкий. Сигнал /WR указывает, что процессор выдает на ШД данные, предназначенные для записи в адресованную ячейку памяти или устройство вывода.
- RFSH** Refresh - Регенерация
Выход. Активный уровень - низкий. Сигнал /RFSH указывает, что младшие 7 разрядов шины адреса содержат адрес регенерации для динамической памяти и текущий сигнал /MREQ может использоваться для восстановления информации. Примеры использования сигнала /RFSH см. в книге 7 "Построение систем".
- HALT** Halt State - Состояние останова
Выход. Активный уровень - низкий. Сигнал /HALT указывает, что ЦП выполняет команду останова программы и ожидает маскируемое либо немаскируемое прерывание, чтобы завершить эту команду и начать обработку подпрограммы прерывания. В состоянии останова ЦП выполняет холостые команды для обеспечения процесса регенерации памяти.
- WAIT** Halt - Запрос ожидания
Вход. Активный уровень - низкий. Сигнал /WAIT указывает ЦП, что адресованная ячейка памяти или устройство ввода-вывода ещё не готово к передаче данных. ЦП генерирует состояние ожидания (холостые такты, в которых не происходит никаких изменений с ЦП) до тех пор, пока активен этот сигнал. С помощью этого сигнала с ЦП могут синхронизироваться ЗУ и устройства ввода-вывода практически любого быстродействия. /WAIT также может использоваться при отладке для реализации пошагового режима.
- INT** Interrupt Request - Запрос прерывания
Вход. Активный уровень - низкий. Сигнал /INT, формируемый устройством ввода-вывода, анализируется в конце выполнения текущей команды. Запрос учитывается, если триггер прерываний (IFF1), управляемый программно, установлен в состояние "разрешить прерывания", и не активен сигнал /BUSRQ
- NMI** Non Maskable Interrupt - Немаскируемый запрос прерывания
Вход, запускаемый отрицательным фронтом. Фронт запуска активизирует внутренний триггер NMI. Линия /NMI имеет более высокий приоритет, чем /INT и всегда распознается в конце выполнения текущей команды, независимо от состояния триггера разрешения прерываний. /NMI автоматически производит перезапуск (рестарт) ЦП с адрес 66H. Содержимое счётчика команд (адрес возврата) автоматически сохраняется во внешнем стеке. Т.о. пользователь может возвратиться к прерванной программе.
- RESET** Reset - Сброс
Вход. Активный уровень – низкий. Сигнал /RESET имеет самый высокий приоритет и приводит ЦП в начальное состояние:
 • сброс счетчика команд PC=0000H;
 • сброс триггера разрешения прерываний
 • очистка регистров I и R;
 • установка режима прерываний IM0.
 Для корректного сброса сигнал /RESET должен быть активен не менее 3-х периодов тактовой частоты. В это время адресная шина и шина данных находятся в высокоомном состоянии, а все выходы сигналов управления неактивны.
- BUSRQ** Bus Request - Запрос доступа к шине
Вход. Активный уровень - низкий. Сигнал /BUSRQ имеет более высокий приоритет, чем /NMI и анализируется в конце каждого машинного цикла. Он делает запрос ЦП на перевод всех его шин в высокоомное состояние для того, чтобы другие устройства смогли управлять этими шинами (например, при прямом доступе к памяти). Если активизирован

сигнал /BUSRQ то ЦП переводит шины в высокоомное состояние как только завершен текущий машинный цикл.

BUSAK Bus Acknowledge - Предоставление доступа к шине

Выход. Активный уровень - низкий. Если был активизирован сигнал BUSRQ, то ЦП переводит свои шины в высокоомное состояние, как только завершен текущий машинный цикл. После этого ЦП активизирует сигнал /BUSAK, который сообщает запрашивающему устройству, что шины адреса и данных, а также трехстабильные сигналы управления находятся в высокоомном состоянии, и внешнее устройство может ими управлять.

C Clock - Такт

Вход для однофазной тактовой синхронизации. При управлении от ТТЛ-схемы вход C дополнительно подключается к линии +5В через внешнее сопротивление 330 Ом.

U_{CC} Плюс источника питания

U_{SS} Потенциал "земли"

4. Временные диаграммы машинных циклов

Обработка команд программы микропроцессором Z80 представляет собой поэтапное выполнение следующих машинных циклов:

- извлечение кода операции (цикл M1);
- цикл чтения/записи памяти;
- цикл ввода/вывода;
- цикл предоставления доступа к шине;
- цикл подтверждения маскируемого прерывания;
- цикл подтверждения немаскируемого прерывания;
- выполнение команды останова.

Все команды состоят из последовательности машинных циклов. Каждый из этих машинных циклов продолжается от 3 до 6 тактов и может быть удлинён путём введения дополнительных тактов T_w (время ожидания), если скорость ЦП ограничивается быстродействием внешнего устройства.

Рис. 4.1 показывает, что команды, как правило, состоят из 3 машинных циклов. Первый машинный цикл каждой команды - это цикл извлечения кода операции, который длится 4, 5 или 6 тактов синхронизации (если он не продлевается сигналом /WAIT), в цикле M1 из памяти извлекается код команды, которая потом выполняется. В последующих машинных циклах осуществляется передача данных между ЦП и памятью или устройством ввода-вывода. Эти циклы продолжаются от 3 до 6 тактов и также могут быть продлены сигналом /WAIT, если ЦП должен синхронизироваться с внешним устройством.

Далее рассматриваются временные диаграммы базовых машинных циклов. Очередность их выполнения, а также точное время выполнения команд (в тактах) приведены в 5.4.

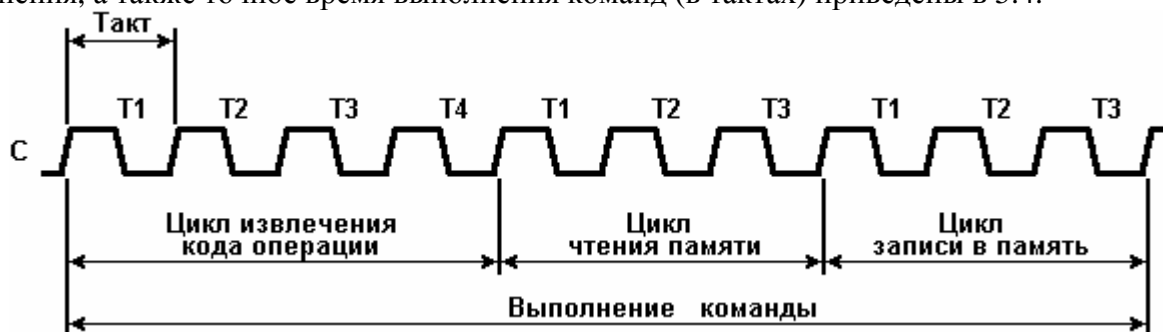


Рис. 4.1. Пример выполнения команды

4.1. Извлечение кода операции

На рис.4.2 отражены временные процессы цикла M1. Содержимое счётчика команд PC (адрес кода операции в памяти) подается на шину адреса непосредственно в начале машинного цикла. Через полтакта (когда адрес памяти стабилизируется на шине) активизируется сигнал /MREQ. Его спадающий фронт прямо используется для выбора микросхемы запоминающего устройства. Одновременно с этим включается сигнал /RD и позволяет передать данные из памяти на шину данных ЦП. По нарастающему фронту такта T3 данные вводятся в ЦП. Этот же фронт используется и для выключения сигналов /MREQ и /RD

Во время тактов T3 и T4 происходит дешифрация и выполнение извлеченной команды внутри МП. Одновременно с этим производится регенерация динамической памяти: на 7 младших битов шины адреса подается адрес регенерации, и активизируется сигнал /RFSH указывая на то, что ША содержит адрес регенерации. Само восстановление происходит по сигналу /MREQ; сигнал RFSH нельзя использовать для этого, т.к. Устойчивость адреса регенерации обеспечивается только ко времени включения /MREQ. Во время регенерации сигнал /RD не формируется, чтобы избежать передачи данных из различных областей памяти на шину данных.

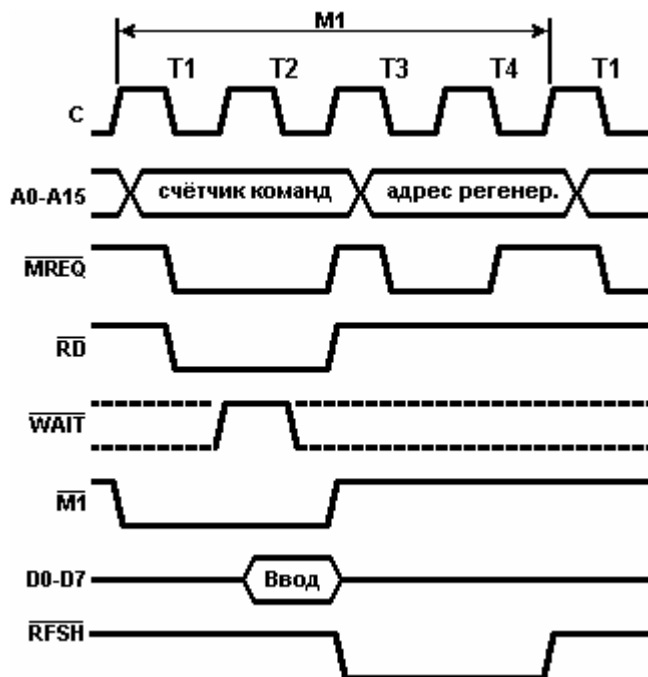


Рис. 4.2. Извлечение кода операции (цикл M1)

На рис.4.3 показано, как продлевается цикл извлечения кода операции, если память активизирует линию /WAIT. Во время спадающего фронта такта T2 и каждого последующего такта T_w ЦП анализирует линию /WAIT. Если она активна, то микропроцессор вырабатывает дополнительный такт ожидания T_w . Т.о. цикл считывания продолжается сколь угодно долго и приводится в соответствие со временем доступа к любой памяти.

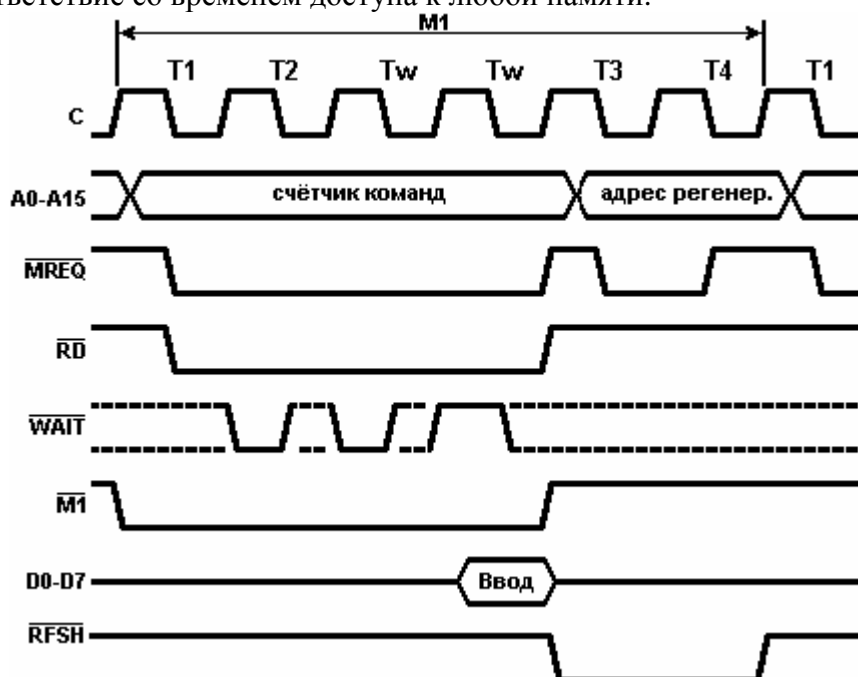


Рис. 4.3. Извлечение кода операции (цикл M1) с тактами ожидания

4.2. Цикл чтения памяти и цикл записи в память

На рис. 4.4 показаны временные процессы циклов чтения из памяти и записи в память. Длительность этих циклов 3 такта, если не активизирована линия /WAIT. Сигналы /MREQ и /RD используются так же, как в цикле M1. В обоих циклах сигнал /MREQ активизируется, когда уровни сигналов на шине адреса стабилизировались, поэтому его спадающий фронт используется для выбора микросхемы ЗУ. В цикле записи линия /WR активизируется, когда данные на шине данных уже стабилизировались, т.о. этот сигнал непосредственно используется как импульс записи для всех типов полупроводниковой памяти. Он выключается на полтакта до изменения содержимого ША и ШД, что удовлетворяет параметрам всех типов полупроводниковых ОЗУ.

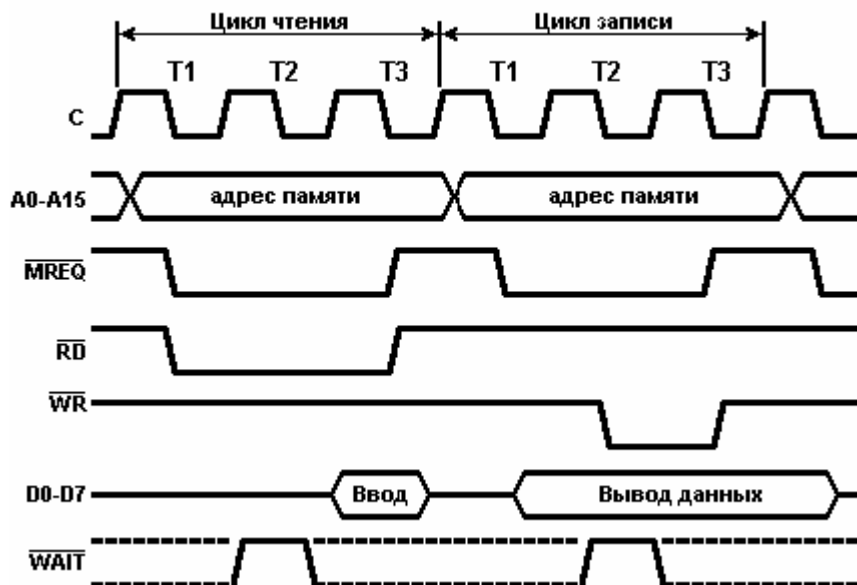


Рис. 4.4. Цикл чтения из памяти и цикл записи в память

На рис.4.5 показано, как запрос /WAIT продлевает операцию чтения или записи в память. Это происходит так же, как в цикле M1.

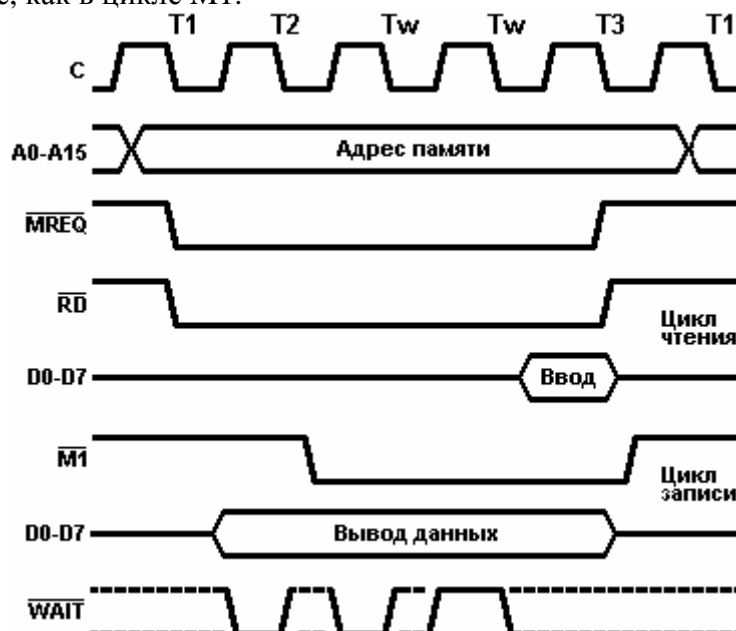


Рис. 4.6. Цикл чтения из памяти и цикл записи в память с тактами ожидания

4.3. Циклы ввода/вывода

На рис. 4.6 изображены временные диаграммы циклов ввода-вывода.

При вводе/выводе содержимое шины адреса различно для двух случаев.

1). Команды IN A,(n) и OUT (n),A:

A0-A7 - содержит адрес канала (n).

A8-A15 - содержимое аккумулятора.

2) Команды IN r,(C), INI, INIR, IND, INDR и OUT (C),r, OUTI, OTIR, OUTD, OTDR:

A0-A7 - Содержимое регистра C.

A8-A15 - содержимое регистра B¹.

Важно отметить, что в операциях ввода-вывода автоматически вводится такт ожидания T_w^* , потому что время от включения сигнала /IORQ до момента опроса линии ожидания центральным процессором недостаточно для декодирования адреса устройства ввода-вывода и активизации им линии /WAIT. Во время этого такта также опрашивается линия ожидания, что

¹ 1. Т.е. МП Z80 может адресовать 64К устройств ввода/вывода (в отличие от I8080, где A8-A15 дублируют A0-A7).

дает возможность согласовать работу ЦП с работой любых медленно действующих устройств. В операции ввода сигнал /RD используется для передачи данных адресованного канала на шину данных ЦП, как и при чтении памяти. В операциях вывода сигнал /WR используется как строб записи. Он выключается за полтакта до изменения состояния ША и ШД, что обеспечивает надежность записи в канал.

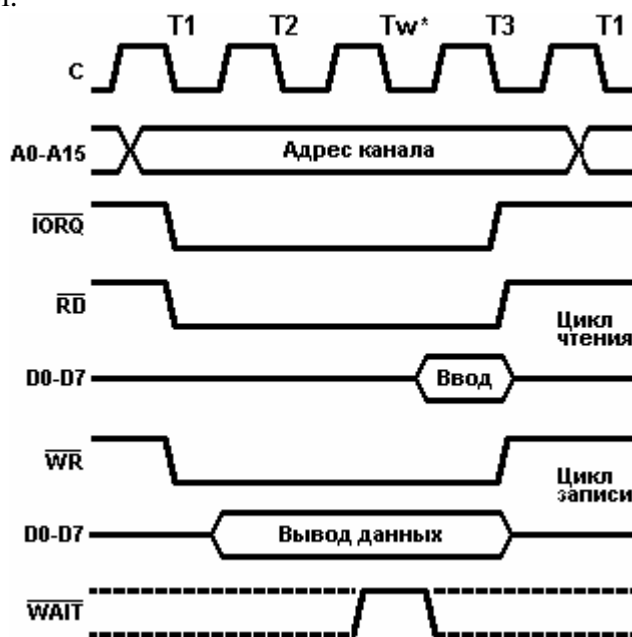


Рис. 4.6. Цикл ввода и цикл вывода

На рис. 4.7 изображены циклы ввода/вывода с дополнительными тактами ожидания.

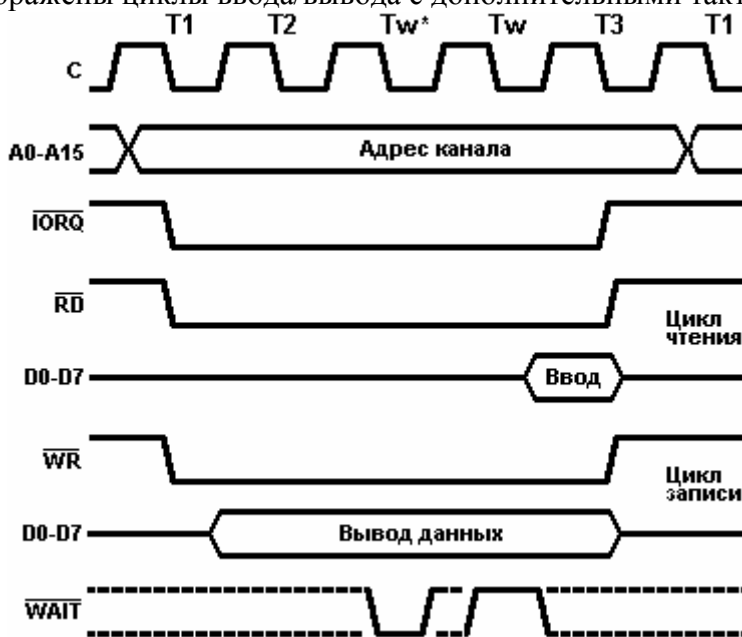


Рис. 4.7. Цикл ввода и цикл вывода с тактами ожидания

4.4. Цикл предоставления доступа к шине

Рис.4.8 отражает временные диаграммы запроса на доступ к шине и цикла подтверждения запроса. Сигнал /BUSRQ воспринимается ЦП во время нарастающего фронта последнего такта каждого машинного цикла. Если он имеет активный низкий уровень, то во время нарастающего фронта следующего тактового импульса (т.е. по окончании цикла) ЦП переводит шины адреса и данных, а также трехстабильные Сигналы управления в высокоомное состояние и активизирует сигнал подтверждения /BUSAK. Т.о. максимальное время до предоставления шины равно длительности машинного цикла (если запрос поступил в начале этого- цикла).

Теперь внешнее устройство, запросившее доступ, может управлять шинами и передавать данные между памятью и УВВ. Этот режим называется прямым доступом к памяти (ПДП - DMA - Direct Memory Access). Его нельзя прервать ни сигналом /INT, ни /NMI.

Внешнее устройство поддерживает сигнал /BUSRQ в активном состоянии столько времени, сколько необходимо ему для прямого доступа. Состояние /BUSRQ анализируется процессором по нарастающему фронту каждого тактового импульса. Как только зафиксирован неактивный уровень /BUSRQ, со следующего такта управление шинами возвращается ЦП, и он возобновляет нормальную обработку команд.

Следует помнить, что во время длительных циклов ПДП (например, в случае пересылки больших блоков данных) и при использовании динамического ЗУ регенерацию должно осуществлять внешнее устройство.

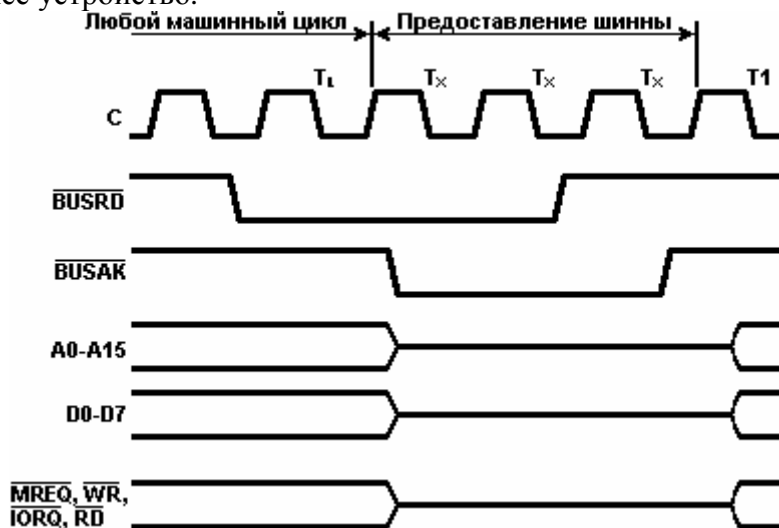


Рис. 4.6. Цикл предоставления доступа к шине

4.5. Цикл подтверждения маскируемого прерывания

На рис.4.9 показаны временные процессы при запросе и подтверждении прерывания. Сигнал прерывания /INT опрашивается ЦП во время нарастающего фронта последнего такта в конце каждой команды. Запрос игнорируется, если триггер разрешения прерывания сброшен или активен сигнал /BUSRQ. Если запрос воспринимается, то ЦП вырабатывает специальный цикл M1, в котором активизируется сигнал /IORQ (вместо /MREQ). Низкий уровень сигнала /IORQ указывает устройству, запросившему прерывание, что оно может подать на шину данных 8 разрядный вектор прерывания. В этот цикл автоматически включаются два такта ожидания T_w^* (это позволяет сравнительно легко реализовать схему приоритетных прерываний), которые дают необходимое время для определения запросившего устройства и стабилизации вектора прерывания на шине данных.

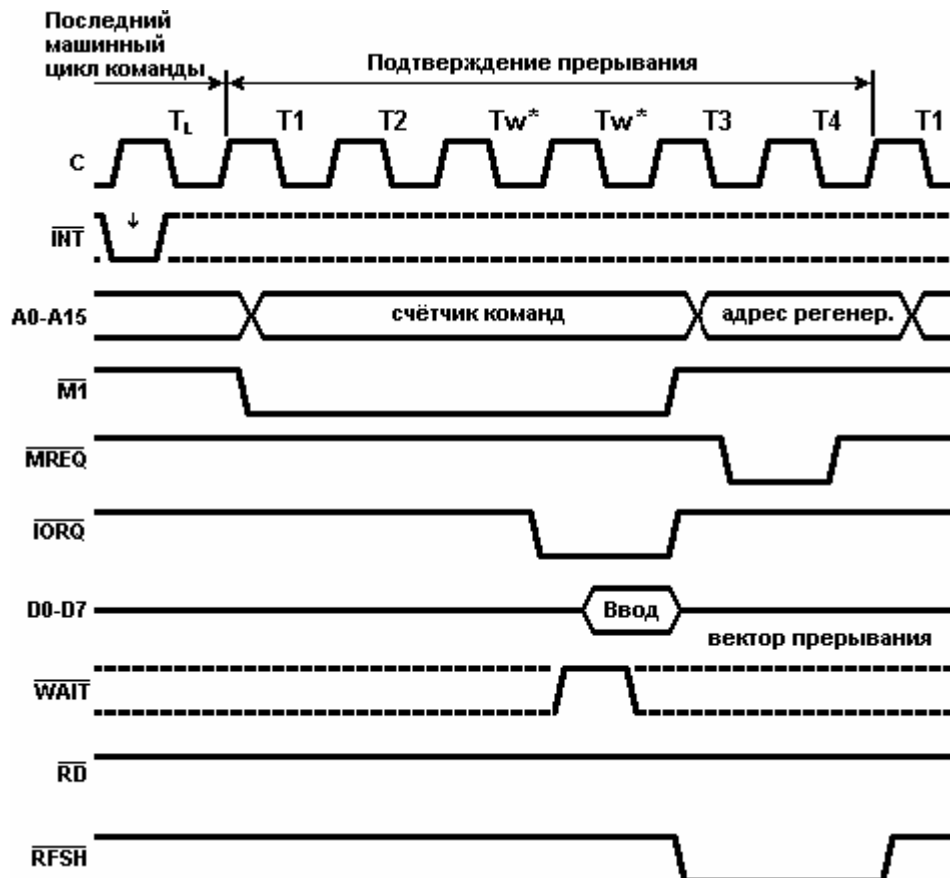


Рис. 4.9. Цикл подтверждения маскируемого прерывания

Кроме двух обязательных тактов ожидания активным сигналом /WAIT могут быть введены дополнительные такты ожидания, что отражено на рис. 4.10.

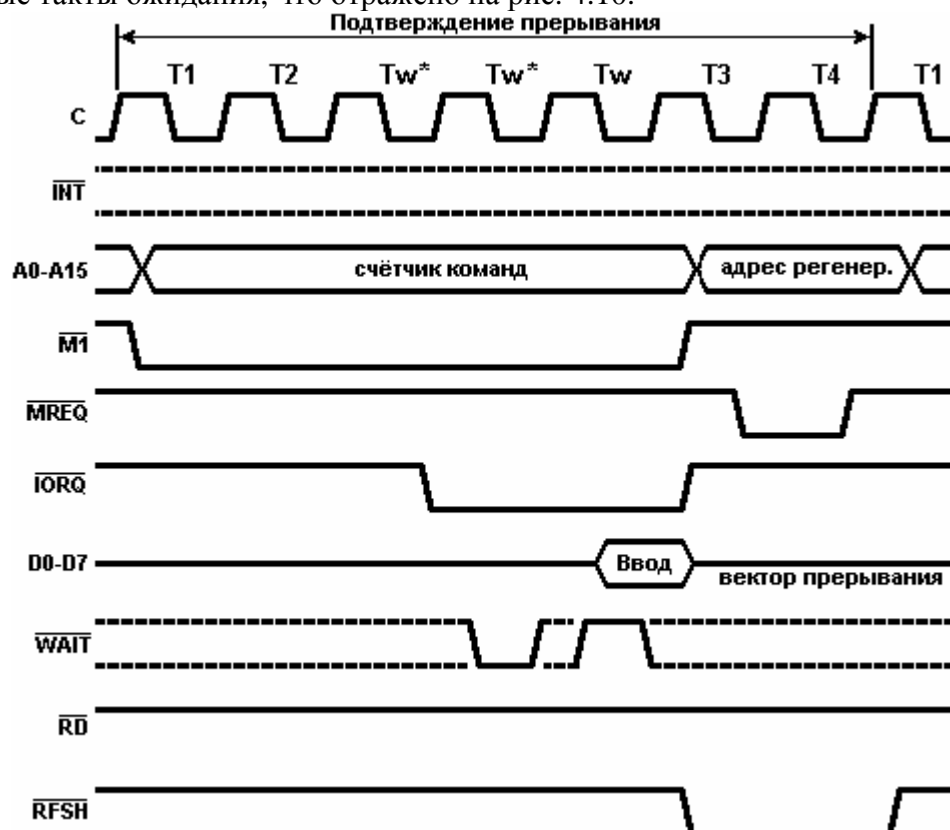


Рис. 4.10. Цикл подтверждения маскируемого прерывания с тактами ожидания

4.6. Цикл подтверждения немаскируемого прерывания

На рис. 4.11 показан цикл запроса/подтверждения немаскируемого прерывания. Информация о запросе /NMI анализируется одновременно с сигналом /INT (подробнее см. 6.2), но /NMI имеет более высокий приоритет и его нельзя запретить программно. Назначение /NMI - немедленное реагирование ЦП на внешние события (например, отказ питания). Ответ ЦП сходен с обычным циклом M1, с той лишь разницей, что шина данных игнорируется, т.к. при NMI ЦП выполняет повторный запуск с адреса 66H и вектор прерывания не нужен.

В цикле подтверждения NMI сигнал /WAIT не воспринимается.

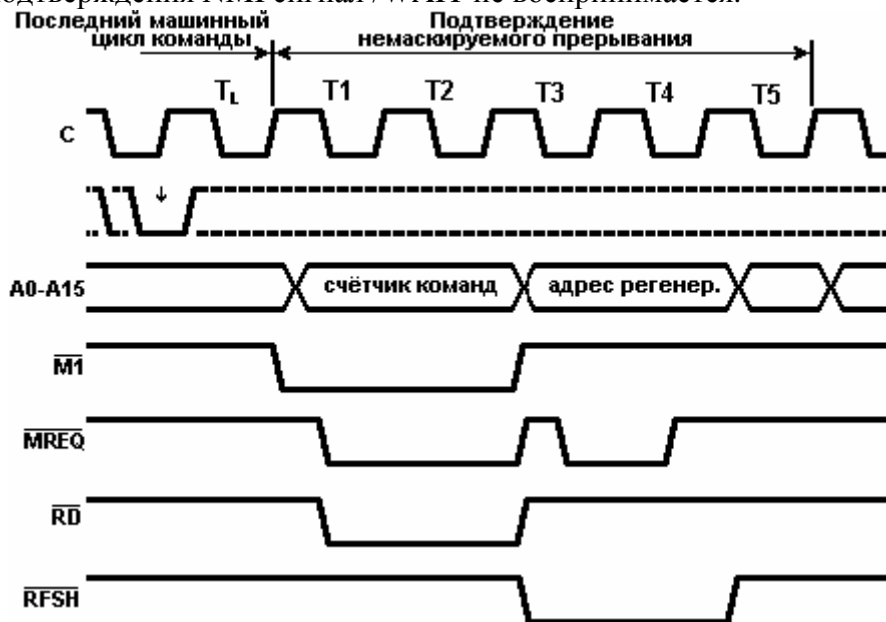


Рис. 4.11. Цикл подтверждения немаскируемого прерывания

4.7. Выполнение команды останова

Команда HALT вводит МП в состояние останова. Каждый цикл в состоянии останова представляет собой обычный цикл M1 с той лишь разницей, что извлекаемые из памяти данные игнорируются, и внутренне процессор формирует команду NOP. Холостые команды выполняются в целях полдержания процесса регенерации.

Сигнал /HALT своим активным уровнем сообщает, что ЦП находится в состоянии останова. Выход из этого состояния возможен только по прерыванию (маскируемому, если оно разрешено, или немаскируемому). Обе линии прерывания опрашиваются во время нарастающего фронта такта T4, как показано на рис. 4.12. Если прерывание воспринято, то следующий цикл будет циклом подтверждения прерывания.

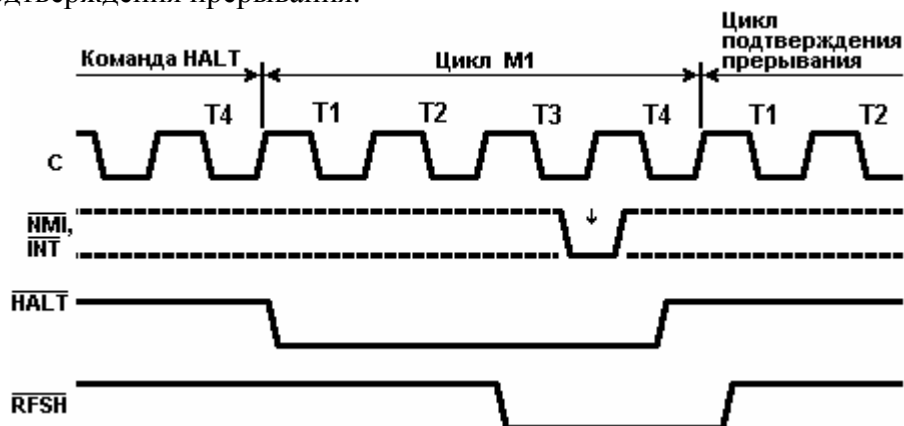


Рис. 4.12. Выполнение команды останова

5. Система команд

5.1. Методы адресации

Одним из важнейших показателей мощности системы команд любого микропроцессора является количество методов адресации. Под методом адресации понимается способ обращения команды к обрабатываемому операнду. Различные методы адресации необходимы, с одной стороны, для удобства программирования, с другой, для эффективной работы программ.

Далее рассмотрены методы адресации, используемые в системе команд МП Z80.

Непосредственная адресация.

При таком методе адресации операнд находится в памяти непосредственно за кодом операции команды. Он может быть 8 или 16-битной константой. Формат команд в этой случае:

Код операции								1 или 2 байта
d7	d6	d5	d4	d3	d2	d1	d0	Операнд

Код операции								1 или 2 байта
d7	d6	d5	d4	d3	d2	d1	d0	Младший байт операнда
d7	d6	d5	d4	d3	d2	d1	d0	Старший байт операнда

Регистровая адресация.

В этом случае операнд находится в регистре либо в паре регистров ЦП. Регистр адресуется тремя битами, входящими в код операции, пара регистров - двумя.

Косвенная адресация.

При этой методе адресации операнд находится в ячейке памяти, адрес которой содержится в одной из регистровых пар BC, DE или HL.

Абсолютная адресация.

За кодом операции в этом случае следуют два байта (16 бит), которые являются адресом. Это может быть адрес данных в памяти, адрес перехода или адрес подпрограммы.

Код операции								1 или 2 байта
d7	d6	d5	d4	d3	d2	d1	d0	Младший байт операнда
d7	d6	d5	d4	d3	d2	d1	d0	Старший байт операнда

Модифицированная нуль-страничная адресация.

Адресное пространство микропроцессора условно можно разбить на страницы по 4К. Тогда к нулевой странице будут относиться адреса 0000H...1000H. МП Z80 имеет специфический вид адресации, когда в коде операции 3 бита, задают модифицированный адрес рестарта. Эти адреса располагаются на нулевой странице области памяти МП. Команда состоит из одного байта:

1	1	t5	t4	t3	1	1	1	Код операции RST p
---	---	----	----	----	---	---	---	--------------------

А эффективный адрес рестарта будет:

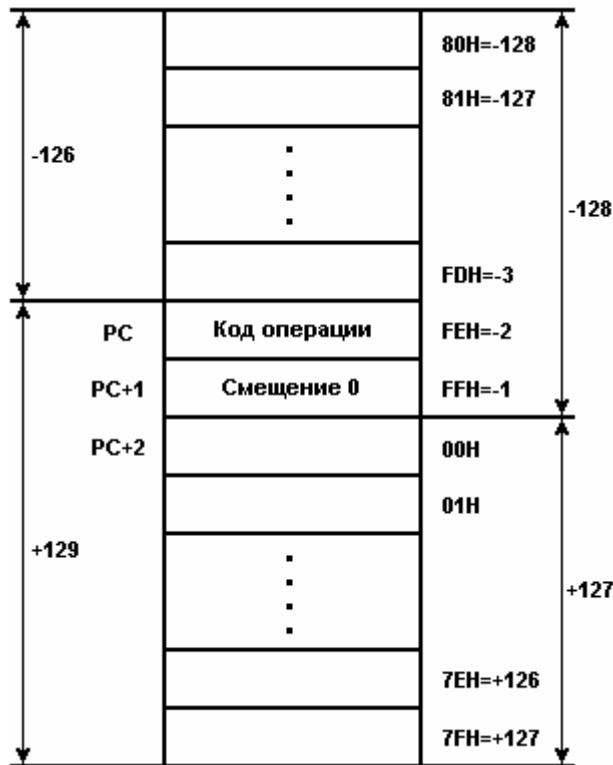
0 0 0 0 0 0 0 0 0 t5 t4 t3 0 0 0 В,

или 00H, 08h, 10H, 18H, 20h, 28H, 30H и 38H.

Следующие три вида адресации МП Z80 существенно расширяют его возможности по сравнению с I8080.

Относительная адресация.

Этот вид адресации может использоваться командами условного и безусловного переходов. В каком случае они состоят из двух байтов: первый байт содержит код операции, а второй - смещение в дополнительном коде. Действительный адрес получается прибавлением смещения к текущему значению счетчика команд.



Смещение как число со знаком, может задаваться от -128 до +127. Но т.к. оно складывается с (PC+2), то фактически можно, адресоваться к ячейкам памяти, расположенным на расстоянии от -126 до +129 байт от адреса кода операции команды перехода.

Преимущества относительной адресации перед абсолютной:

- команда занимает в памяти на один байт меньше места;
- программа становится перемещаемой, т.е. не зависит от своего места расположения в памяти.

Индексная адресация.

Сущность индексной адресации аналогична относительной адресации, только в данном случае базой выступает не счётчик команд PC, а один из индексных регистров IX или IY. Данный метод адресации удобен при обработке массивов данных. Смещение d, непосредственно представленное в команде, складывается с содержимым индексного регистра и образует адрес памяти, по которому находятся данные. Т.о. можно адресоваться к данным находящимся в памяти на расстоянии от -128 до +127 байт от содержимого индексного регистра. Структура команд в данном случае:

Управляющее слово DD или FD
Код операции
Смещение d

Битовая адресация.

Этот вид адресация используется в группе команд для работы с отдельными битами. Три разряда кода операции являются номером бита операнда, над которым будут производиться действия (тестирование, сброс или установка).

Встроенная адресация.

Некоторые команды, состоящие только из кода операции, однозначно определяют, где находится операнд. Например, команды работы с аккумулятором NEG и CPL.

Смешанная адресация.

Если в команде операнды адресованы по-разному, то имеет место смешанная адресация, например LD (IX+d),r

5.2. Группы команд

Систему команд МП можно разделить на группы команд в соответствии с их функциональным назначением. Группы команд представлены в таблицах 5.1.1 ... 5.1.12.

Первая колонка в этих таблицах содержит символические обозначения (мнемокоды), используемые при написании программ на языке Ассемблера МП Z80. В основе этих символических обозначений, как правило, лежит аббревиатура полной записи команды на английском языке.

Во второй - время выполнения в тактах.

В третьей колонке приведены формат и двоичный кол команды.

Действия, происходящие при выполнении команды, показаны в четвертой колонке. Следует отметить, что здесь не отражен порядок выполнения этих действий; этот вопрос подробно рассмотрен в 5.4.

Последняя колонка отражает состояние флагов после выполнения команды.

В дальнейшем описании и таблицах приняты следующие обозначения:

r	8-битный регистр: A=111; B=000; C=001; D=010; E=011; H=100; L=101
dd qq pp rr	16-битная регистровая пара: dd BC=00; DE=01; HL=10; SP=11 qq BC=00; DE=01; HL=10; AF=11 pp BC=00; DE=01; IX=10; SP=11 rr BC=00; DE=01; IY=10; SP=11
p	8-битный операнд
pn	16-битный операнд
(nn)	содержимое ячейки nn (8-битный операнд) или содержимое ячеек nn и nn+1 (16-битный операнд)
t	r, (HL), (IX+d) или (IY+d)
e	t или p
d	8-битное смещение (-128 ... +127)
e	8-битное смещение (-126 ... +129)
b	позиционный номер бита: D0=000; D1=001; D2=010; D3=011; D4=100; D5=101; D6=110; D7=111
cc	условие перехода: NZ=000 (не ноль); Z=001 (ноль); NC=010 (нет переноса); C=011 (перенос); PO=100 (нечетность); PE=101 (четность); P=110 (знак плюс); M=111 (знак минус)
p	адрес рестарта: p=00H; p=08H; p=10H; p=18H; p=20H; p=28H; p=30H; p=38H
a	номер рестарта: a=000; a=001; a=010; a=011; a=100; a=101; a=110; a=111
	Состояние флагов: 0 флаг сброшен 1 флаг установлен - неизменное X неопределенное ? изменяется в соответствии с результатом V функции переполнения флага P/V P Функция четности флага P/V IF состояние IFF2

T число тактов за которое выполняется команда

8-битовые команды загрузки. Мнемоническое сокращение языка Ассемблера для всех команд загрузки одинаково: LD (load - загрузить). Далее следует указание приёмника, затем - указание источника пересылаемых данных.

LD приемник, источник

Таблица 5.1.1 содержит все команды 8-битной загрузки. Большую часть кодов в ней представляет команда LD r1,r2. В качестве r1 и r2 могут использоваться семь регистров: A, B, C, D, E, H и L, каждый из которых может служить как источником, так и приемником. Следовательно, эта команда дает в результате 49 кодов операции. В их числе семь команд с r1=r2,

которые по действию аналогичны NOP, т.к. фактического переноса данных не происходит, и флаги не изменяются.

Источником переносимых данных может быть константа, непосредственно представленная в команде сразу за кодом операции. Команда в этом случае выглядит так:

LD r,n

Как источником, так и приёмником может служить память. Это даёт три типа команд:

LD r,(HL)

LD (HL),r

LD (HL),n

Память не может быть одновременно и источником, и приёмником пересылаемых данных; пересылка всегда происходит через ЦП.

С помощью управляющего слова DD либо FD предоставляется возможность использовать команды с индексным способом адресации. В этих командах за кодом операции следует смещение d.

LD r,(IX+d)

LD r,(IY+d)

LD (IX+d),r

LD (IY+d),r

LD (IX+d),n

LD (IY+d),n

Эти команды не влияют на состояние флагов.

В следующих командах данные пересылаются между аккумулятором и ячейкой памяти, адресуемой абсолютно – (nn), либо косвенно через регистровую пару:

LD A,(BC)

LD A,(DE)

LD A,(nn)

LD (BC),A

LD (DE),A

LD (nn),A

Флаги также остаются без изменений.

Регистры I и R могут использоваться как источник или приёмник в сочетании с аккумулятором:

LD A,I

LD A,R

LD I,A

LD R,A

Команды LD A,I и LD A,R в отличие от всех остальных команд загрузки оказывают влияние на флаги:

$H \leftarrow 0$

$N \leftarrow 0$

$P/V \leftarrow \text{IFF2}$ (триггер разрешения /запрещения прерываний)

В сумке группа 8-битных команд загрузки содержит 104 различных кода.

16 битовые команды загрузки. Команды этой группы представлены в таблице 5.1.2. В них возможна регистровая адресация и косвенная через указатель стека. Сюда же входят команды пересылки в стек и извлечения из стека, их мнемонические обозначения PUSH и POP соответственно. Отличие их от обычных команд загрузки состоит в том, что указатель стека автоматически уменьшается или увеличивается, когда байт "заталкивается" в стек или извлекается из стека. По действию команды PUSH и POP взаимно обратные. Это в равной степени откосится и к последовательности обработки байтов (см. таблицу 5.3): при PUSH сначала обрабатывается старший байт, при POP - младший.

При всех 16-битовых командах загрузки, за исключением POP AF не происходит изменения флагов

16-битовые команды обмена. Эта группа содержит 6 команд, которые изображены в таблице 5.1.8. Мнемоническое обозначение EX происходит от слова Exchange (обмен).

Команда EX AF,AF' дает возможность программным путем переключать пары регистров A, F и A', F'. А команда EXX позволяет переключаться на вспомогательный набор РОНов. Эти команды длиной всего один байт, и время, необходимое для переключения, сведено к минимальному. Т.о. реализовано очень короткое время отклика на прерывание.

Как и 16-битовые команды загрузки, команды обмена не влияют на состояние флагов.

Команды обработки блоков. Команды этой группы (см. таблицу 5.1.3) выполняют две функции пересылка блоков и поиск байта в блоке данных.

Команды пересылки имеют мнемоники:

LDI	загрузка с инкрементом
LDIR	загрузка с инкрементом до BC=0
LDD	загрузка с декрементом
LDDR	загрузка с декрементом до BC=0

Команда LDI (LDD) пересылает содержимое области памяти, начиная с адреса, указанного в HL в область памяти с начальным адресом в DE. Затем обе регистровые пары инкрементируются (декрементируются), и счётчик байтов BC уменьшается на 1.

Команда LDIR (LDDR) выполняется, так же как и LDI (LDD), только обработка сразу "зацикливается". Действие команды подобно пересылке блока данных при ПДП. Как для адресации, так и для счётчика байтов отведено по 16 бит, т.о. могут обрабатываться блоки любой длины и в любом месте памяти. Блоки данных могут также перекрываться. При HL=DE команда теряет смысл т.к. источник и приёмник идентичны. Следует обратить внимание, что при обработке команды сначала пересылаются данные, только затем декрементируется BC, и на основании этого решается вопрос о цикличности команды, выполнять дальнейшую пересылку данных либо прекратить команду. После выполнения команды BC=0, что соответствует длине блока в $2^{16}=65536$ байт.

Выполнение этих команд влияет на флаги:

$H \leftarrow 0$

$N \leftarrow 0$

$P/V \leftarrow 1$, при циклическом выполнении команды до пересылки последнего байта

$P/V \leftarrow 0$, когда результат декремента BC достигнет 0.

Поиск данных в блоке выполняют четыре команды:

CPI	сравнение и инкремент
CPIR	сравнение и инкремент до BC=0 или A=(HL)
CPD	сравнение и декремент
CPDR	сравнение и декремент до BC=0 или A=(HL)

При выполнении команды CPI (CPD) из содержимого аккумулятора вычитается содержимое ячейки памяти с адресом в HL, затем декрементируется счётчик байтов BC, и инкрементируется (декрементируется) HL. В соответствии с результатом сравнения устанавливаются флаги.

Команда CPIR (CPDR) является циклическим повторением команды CPI (CPD) до тех пор, пока не выполнится одно из условий. Это либо нахождение идентичного байта, либо окончание просматриваемого блока. Соответственно этому устанавливаются флаги:

$S \leftarrow 1$, когда результат сравнения отрицательный, иначе $S \leftarrow 0$

$Z \leftarrow 1$, когда $A=(HL)$, иначе $Z \leftarrow 0$

$H \leftarrow 1$, когда был перенос в 4-й разряд, иначе $H \leftarrow 0$

$P/V \leftarrow 1$, при $BC \neq 0$

$P/V \leftarrow 0$, когда результат декремента BC достигнет 0

$N \leftarrow 1$

При каждом выполнении, до $P/V=0$, счётчик команд дважды декрементируется, что означает вновь обращение к этой команде.

8-битные арифметические и логические команды. Команды этой группы отражены в таблице 5.1.4. Они всегда используют аккумулятор в качестве приёмника результата. Следовательно, в команде указывается только источник. В совокупности в этой группе 108 различных кодов, которые имеют следующие мнемоники:

ADD s	сложение
ADC s	сложение с учетом переноса
SUB s	вычитание
SBC s	вычитание с учетом переноса
AND s	логическое "И"
OR s	логическое "ИЛИ"
XOR s	"исключающее ИЛИ"
CP s	сравнение с аккумулятором
INC t	инкремент
DEC t	декремент

Флаги C, Z, S и H устанавливаются соответственно результату операции. Флаги P/V действует в качестве флага переполнения V. Флаг N соответствует биту 4 кода выполняемой операции и указывает команде DAA (см. ниже), какая из арифметических операций выполнялась - сложение или вычитание.

Команда CP s выполняется так же, как и SUB s, с той разницей, что по результату операции лишь устанавливаются флаги, а содержимое аккумулятора не изменяется.

Отдельно можно выделить команды, в которых арифметические и логические операции производятся только над содержимым аккумулятора:

ADD A	удвоение аккумулятора
ADC A	как ADD A с учётом переноса
SUB A	очистка аккумулятора ($A \leftarrow 0$), флаги: $S \leftarrow 0$, $Z \leftarrow 1$, $H \leftarrow 0$, $P/V \leftarrow 0$; $N \leftarrow 1$, $C \leftarrow 0$.
SBC A	как SUB A при $C=0$; когда $C=1$, содержимое аккумулятора сбрасывается в 0FFH. Флаги: $S \leftarrow 1$, $Z \leftarrow 0$, $H \leftarrow 1$, $P/V \leftarrow 0$; $N \leftarrow 1$, $C \leftarrow 1$.
AND A	аккумулятор остается без изменений; в соответствии с его содержимым устанавливаются флаги, $S \leftarrow$ старший бит; $Z \leftarrow 1$ при $A=0$, иначе $Z \leftarrow 0$; $P/V \leftarrow 1$ при чётном количестве единиц, иначе $P/V \leftarrow 0$; $N \leftarrow 0$; $C \leftarrow 0$.
OR A	как при AND A.
XOR A	очистка аккумулятора ($A \leftarrow 00$); флаги - как при AND A.
CP A	аккумулятор без изменений; флаги устанавливаются как при SUB A.

Общие операции с аккумулятором и флагами. К этой группе (таблица 5.1.4) относятся следующие 5 команд:

DAA	десятичная коррекция аккумулятора
CPL	дополнение аккумулятора
NEG	инверсия аккумулятора
CCF	инверсии флага переноса
SCF	установка Флага переноса

Десятичная коррекция аккумулятора используется после сложения либо вычитания чисел в BCD (двоично-десятичной) формате. В этом формате 8-битный операнд представляет собой двузначное десятичное число, в котором младшей десятичной цифре соответствует младшая тетрада, а старшей – старшая. Для цифр BCD формата допустимы значения от 0 до 9. Однако, результат сложения или вычитания может превысить 9. В этом случае необходима десятичная коррекция. После команд сложения она действует следующим образом:

1) если содержимое младшей тетрады аккумулятора больше 9 или $H=1$, то к аккумулятору добавляется число 6.

2) если содержимое старшей тетрады аккумулятора стало после этого больше 9 или $C=1$, то число 6 добавляется и к старшей тетраде аккумулятора.

После вычитания:

1) если $H=1$, то от аккумулятора вычитается число 6.

2) если $C=1$, то число 6 вычитается и из старшей тетрады аккумулятора.

Команда CPL побитно инвертирует содержимое аккумулятора (дополнение до 1) Команда NEG изменяет знак содержимого аккумулятора на противоположный (дополнение до 2), она соответствует вычитанию содержимого аккумулятора из нуля. С помощью команды SCF флаг переноса устанавливается в 1, с помощью команды CCF его можно инвертировать.

16-битовые арифметические команды. Команды этой группы сведены в таблицу 5.1.5. Их мнемоники соответствуют принятым для 8 битных арифметических команд, только дополнительно указывается приемник результата. 16-битовые арифметические команды позволяют использовать регистровые пары и 16-разрядные регистры.

В этой группе не предусмотрена команда SUB, поэтому при потребности в ней следует применять команду SBC, предварительно сбросив флаг переноса, например, командой AND A.

Команды вращения и сдвига. Эта группа команд объединена в таблице 5.1.11.

Эти команды позволяют адресоваться к любому регистру или ячейке памяти. Они особенно удобны при выполнении умножения и деления. Две команды (RLD и RRD) вращают тетрады аккумулятора и ячейки памяти, адресуемой парой HL. Их целесообразно применять в двоично-десятичной (BCD) арифметике. Конкретный вид вращения или сдвига ясен из символических пояснений к каждой команде. Эта группа содержит 76 команд.

Команды для работы с битами. Почти в каждой программе требуется возможность устанавливать, сбрасывать и проверять состояние, какого либо бита одного из регистров либо ячейки памяти. Такие биты в самом общем случае выполняют роль флагов, которые указывают программе какие предпринять действия. МП Z80 имеет команды, которые производят тестирование, установку и сброс любого бита регистра или ячейки памяти. Они сведены в таблице 5.1.7. При регистровой адресации действия производятся над аккумулятором или над одним из РОНов. При косвенной и индексной адресации действия производятся над памятью. При тестировании бита устанавливается флаг Z, если бит, равен нулю. Эта группа содержит 240 команд.

Команды Переходов собраны в таблице 5.1.6. Переход - это ветвление программы, при котором в счетчик команд PC загружается новый адрес. В команде он представлен абсолютно, относительно или косвенно. Следует помнить, что условный переход происходит только при выполнении условия, заданного в команде. Если условие не выполняется, то программа переходит к команде, следующей за командой перехода. При абсолютной адресации команда содержит адрес перехода. Такая команда состоит из трех байтов: кода операции, младшего байта адреса и старшего байта адреса перехода.

При относительной адресации команда состоит из двух байтов. Второй байт - это расстояние между адресом перехода и текущим состоянием счетчика команд (т.е. смещение). Смещение может быть от +127 до -126 относительно адреса кода операции команды, следующей за командой перехода.

При косвенной адресации в командах перехода содержится регистровая пара HL или одно из индексных регистров (IX или IY) прямо загружается в счетчик команд.

Для организации циклов в программе очень удобна команда DJNZ e. Эти двухбайтная команда перехода уменьшает содержимое регистра B на единицу, и если оно не стало равно нулю, то происходит переход на смещение e, заданное в команде.

Группа команд перехода не влияет на состояние флагов.

Команды вызова и возврата. Эта группа объединена в таблице 5.1.10.

Команда вызова подпрограммы CALL - это специфическая команда перехода, при которой МП запоминает адрес команды, следующей за командой CALL. Команда возврата RET противоположна команде CALL. В ней адрес из стека прямо загружается в счётчик команд, и осуществляется возврат к прерванной программе. МП имеет две специальные команды возврата: RETI - возврат из маскируемого прерывания и RETN - возврат из немаскируемого прерывания. Команда RETI является единственной командой, которая распознается периферийными микросхемами комплекта Z80. Это необходимо для организации приоритетных прерываний и нормального возврата из прерываний, поступивших от периферийных схем. Об этом подробно написано в книге 7 "Построение систем. Программирование. Отладка".

Команды вызова подпрограмм состоят из 3 байтов. Однако, специфические команды рестарта RST r состоят из одного байта. Фиксированный адрес перехода встроен в код операции этих команд (см. методы адресации).

Группа команд вызова и возврата не влияет на состояние флагов.

Команды ввода-вывода. Набор команд ввода-вывода представлен в таблице 5.1.9. Адресация в этих командах может быть непосредственной либо косвенной, но регистру C.

Команды с непосредственной адресацией IN A,(n) и OUT (n),A аналогичны командам ввода-вывода МП I8080. Но в добавление к адресу порта n, выставляемому в младшую часть шины адреса, в старшую часть ША подается содержимое аккумулятора. Это позволяет, предварительно загрузив аккумулятор нужным значением, использовать 16-разрядный адрес ввода-вывода. На флаги эти две команды не влияют.

В командах с косвенной адресацией данные пересылаются между портом адрес которого загружен в регистр C, и одним из РОНов.

IN r,(C)
OUT (C), r

При этом в старшую часть адреса подается содержимое регистра B, т.е. также может использоваться 16-разрядный адрес ввода-вывода.

Две команды IN C,(C) и OUT (C),C имеют особенности. В команде вывода регистр C содержит адрес канала и его же в качестве данных посылает в порт. В команде ввода содержимое регистра C после ввода из порта замещается принятыми данными.

Команды OUT (C),r не влияют на флаги. Команды ввода IN r,(C) устанавливают флаги Z, S, P, H в соответствии с принятыми данными и сбрасывают флаг N. Следует уделить команду INF, которая, воздействуя на регистр флагов, не изменяет состояния РОН.

Косвенная адресация используется также командами ввода-вывода блоков. Они пересылают данные между портом, адресуемым регистром C, и памятью, адресуемой регистровой парой HL.

INI	ввод и инкремент HL
INIR	ввод и инкремент HL до B=0
IND	ввод и декремент HL
INDR	ввод и декремент HL по B=0
OUTI	вывод и инкремент HL
OTIR	вывод и инкремент HL до B=0
OUTD	вывод и декремент HL
OTDR	вывод и декремент HL до B=0

Регистр B здесь используется как двоичный счётчик - он автоматически декрементируется после пересылки каждого байта (максимально 256). Вместе с тем, содержимое B выдается в старшую часть ША. Это может быть использовано в случае, когда необходимо выводить данные в несколько портов с идущими подряд (в старшем байте) 16-разрядными адресами.

Команды ввода-вывода блоков устанавливают флаги следующим образом:

S	неопределён
Z←1,	если регистр B=0 (выполнение команд заканчивается), иначе Z←0
H	неопределён
P/V	неопределён
N←1	
C	не изменяется

Команды управления микропроцессором. Эта группа (таблица 5.1.12) содержит шесть команд управления.

Команда NOP - это пустая команда, при которой не выполняется никаких действий. Команда HALT останавливает работу МП до тех пор, пока не будет принят запрос на прерывание. После выполнения команды HALT МП внутренне генерирует команды NOP, чтобы производить регенерацию динамической памяти. Для работы с прерываниями служат команды DI и EI, которые запрещают и разрешают прерывания соответственно. МП может работать в одном из трёх режимов, прерывания, которые устанавливаются командами IM 0, IM 1 и IM 2 (подробнее см. главу 6).

Таблица 5.1.1. 8-битовые команды загрузки

Команда	T	Код			Пояснения	C	Z	P	S	N	H
LD r1,r2	4	01	r1	r2	r1 ← r2						
LD r,(HL)	7	01	r	110	r ← (HL)						
LD (HL),r	7	01	110	r	(HL) ← r						
LD r,n	7	00	r	110	r ← n						
		-	n	-							
LD (HL),n	10	00	110	110	(HL) ← n						
		-	n	-							
LD r,(IX+d)	19	11	011	101	r ← (IX+d)						
		01	r	110							
		-	d	-							
LD r,(IY+d)	19	11	111	101	r ← (IY+d)						
		01	r	110							
		-	d	-							
LD {IX+d},r	19	11	011	101	(IX+d) ← r						
		01	110	r							
		-	d	-							
LD (IY+d),r	19	11	111	101	(IY+d) ← r						
		01	110	r							
		-	d	-							
LD (IX+d),n	19	11	011	101	(IX+d) ← n						
		00	110	110							
		-	d	-							
		-	n	-							
LD (IY+d),n	19	11	111	101	(IY+d) ← n						
		00	110	110							
		-	d	-							
		-	n	-							
LD A,(BC)	7	00	001	010	A ← (BC)						
LD A,(DE)	7	00	011	010	A ← (DE)						
LD A,(nn)	13	00	111	010	A ← (nn)						
		-	n	-							
		-	n	-							
LD (BC),A	7	00	000	010	(BC) ← A						
LD (DE),A	7	00	010	010	(DE) ← A						
LD (nn),A	13	00	110	010	(nn) ← A						
		-	n	-							
		-	n	-							
LD A,I	9	11	101	101	A ← I		?	IF	?	0	0
		01	010	111							
LD A,R	9	11	101	101	A ← R		?	IF	?	0	0
		01	011	111							
LD I,A	9	11	101	101	I ← A						
		01	000	111							
LD R,A		11	101	101	R ← A						
		01	001	111							

Таблица 5.1.2. 16-битовые команды загрузки

Команда	T	Код			Пояснения	C	Z	P	S	N	H
LD dd,nn	10	00	dd0	001	dd ← nn						
		-	n	-							
		-	n	-							
LD IX,nn	14	11	011	101	IX ← nn						
		00	100	001							
		-	n	-							
		-	n	-							
LD IY,nn	14	11	111	101	IY ← nn						
		00	100	001							
		-	n	-							
		-	n	-							
LD HL,(nn)	16	00	101	010	H ← (nn+1) L ← (nn)						
		-	n	-							
		-	n	-							

Команда	T	Код			Пояснения	C	Z	P	S	N	H
LD dd,(nn)	20	11	101	101	dd _H ← (nn+1)						
		01	dd1	011	dd _L ← (nn)						
		-	n	-							
		-	n	-							
LD IX,(nn)	20	11	011	101	IX _H ← (nn+1)						
		00	101	010	IX _L ← (nn)						
		-	n	-							
		-	n	-							
LD IY,(nn)	20	11	111	101	IY _H ← (nn+1)						
		00	101	010	IY _L ← (nn)						
		-	n	-							
		-	n	-							
LD (nn),HL	16	00	100	010	(nn+1) ← H (nn) ← L						
		-	n	-							
		-	n	-							
LD (nn),dd	20	11	101	101	(nn+1) ← dd _H						
		01	dd0	011	(nn) ← dd _L						
		-	n	-							
		-	n	-							
LD (nn),IX	20	11	011	101	(nn+1) ← IX _H						
		00	100	010	(nn) ← IX _L						
		-	n	-							
		-	n	-							
LD (nn),IY	20	11	111	101	(nn+1) ← IY _H						
		00	100	010	(nn) ← IY _L						
		-	n	-							
		-	n	-							
LD SP,HL	6	11	111	001	SP ← HL						
LD SP,IX	10	11	011	001	SP ← IX						
		11	111	001							
LD SP,IY	10	11	111	001	SP ← IY						
		11	111	001							
PUSH qq	11	11	qq0	101	(SP-2) ← qq _L						
					(SP-1) ← qq _H						
PUSH IX	15	11	011	101	(SP-2) ← IX _L						
		11	100	101	(SP-1) ← IX _H						
PUSH IY	15	11	111	101	(SP-2) ← IY _L						
		11	100	101	(SP-1) ← IY _H						
POP qq	10	11	qq0	001	qq _H ← (SP+1)						
					qq _L ← (SP)						
POP IX	14	11	011	101	IX _H ← (SP+1)						
		11	100	001	IX _L ← (SP)						
POP IY	14	11	111	101	IY _H ← (SP+1)						
		11	100	001	IY _L ← (SP)						

Таблица 5.1.3. Команды обработки блоков

Команда	T	Код			Пояснения	C	Z	P	S	N	H
LDI	16	11	101	101	(DE) ← (HL)			?		0	0
		10	100	000	затем DE ← DE+1			P=0, если BC-1=0			
					HL ← HL+1			P=1, если BC-1≠0			
					BC ← BC-1						
LDIR	21	11	101	101	(DE) ← (HL)			0		0	0
	(16)	10	110	000	затем DE ← DE+1						
					HL ← HL+1						
					BC ← BC-1						
					и повтор до BC=0						
LDD	16	11	101	101	(DE) ← (HL)			?		0	0
		10	101	000	затем DE ← DE-1			P=0, если BC-1=0			
					HL ← HL-1			P=1, если BC-1≠0			
					BC ← BC-1						

Команда	T	Код			Пояснения	C	Z	P	S	N	H
LDDR	21 (16)	11 10	101 111	101 000	(DE) ← (HL) затем DE ← DE-1 HL ← HL-1 BC ← BC-1 и повтор до BC=0		0			0	0
CPI	16	11	101	101	A-(HL)=? затем HL ← HL+1 BC←BC-1		?	?	?	1	?
CPIR	21 (16)	11 10	101 110	101 001	A-(HL)=? затем HL ← HL+1 BC←BC-1 и повтор по BC=0 или A=(HL)		?	?	?	1	?
CPD	16	11 10	101 101	101 001	A-(HL)=? затем HL ← HL-1 BC←BC-1		?	?	?	1	?
CPDR	21 (16)	11 10	101 111	101 001	A-(HL)=? затем HL ← HL-1 BC←BC-1 и повтор по BC=0 или A=(HL)		?	?	?	1	?

Таблица 5.1.4. 8-битовые арифметические и логические команды

Команда	T	Код			Пояснения	C	Z	P	S	N	H
ADD A,r	4	10	<u>000</u>	r	A ← A+r	?	?	V	?	0	?
ADD A,n	7	11	-	110	A ← A+n	?	?	V	?	0	?
		-	n	-							
ADD A,(HL)	7	10	<u>000</u>	110	A ← A+(HL)	?	?	V	?	0	?
ADD A,(IX+d)	19	11	<u>011</u>	101	A ← A+(IX+d)	?	?	V	?	0	?
		10	<u>000</u>	110							
		-	d	-							
ADD A,(IY+d)	19	11	<u>111</u>	101	A ← A+(IY+d)	?	?	V	?	0	?
		10	<u>000</u>	110							
		-	d	-							
ADC A,s			001		A ← A+s+C	?	?	V	?	0	?
SUB s			010		A ← A-s	?	?	V	?	1	?
SBC A,s			011		A ← A-s-C	?	?	V	?	1	?
AND s			100		A ← A AND s	0	?	P	?	0	1
OR s			110		A ← A OR a	0	?	P	?	0	0
XOR s			101		A ← A XOR a	0	?	P	?	0	0
CP s			111		A-s=? Код подставляется вместо <u>000</u> в командах	?	?	V	?	1	?
INC r	4	00	r	<u>100</u>	r ← r+1		?	V	?	0	?
INC (HL)	11	00	110	<u>100</u>	(HL) ← (HL) + 1		?	V	?	0	?
INC (IX+d)	23	11	011	101	(IX+d) ← (IX+d)+1		?	V	?	0	?
		00	110	<u>100</u>							
		-	d	-							
INC (IY+d)	23	11	111	101	(IY+d) ← (IY+d)+1		?	V	?	0	?
		00	110	<u>100</u>							
		-	d	-							
DEC t			101		t ← t-1 Код подставляется вместо <u>100</u> в командах INC		?	V	?	1	?

Команда	T	Код			Пояснения	C	Z	P	S	N	H
DAA	4	00	100	111	Двоично-десятичная коррекция A	?	?	P	?		?
CPL	4	00	101	111	A ← инверсия A					1	1
NEG	8	11	101	101	A ← -A	?	?	V	?	1	?
		01	000	100							
CCF	4	00	111	111	C ← инверсия C	?				0	X
SCF	4	00	110	111	C ← 1	1				0	0

Таблица 5.1.5. 16 битовые арифметические команды

Команда	T	Код			Пояснения	C	Z	P	S	N	H
ADD HL,dd	11	00	dd1	001	HL ← HL+dd	?				0	X
ADC HL,dd	15	11	101	101	HL ← HL+dd+C	?	?	V	?	0	X
		01	dd1	010							
SBC HL,dd	15	11	101	101	HL ← HL-dd-C	?	?	V	?	1	X
		01	dd0	010							
ADD IX,pp	15	11	011	101	IX ← IX+pp	?				0	X
		00	pp1	001							
ADD IY,rr	15	11	111	101	IY ← IY+rr	?				0	X
		00	rr1	001							
INC dd	6	00	dd0	011	dd ← dd+1						
INC IX	10	11	011	101	IX ← IX+1						
		00	100	011							
INC IY	10	11	111	101	IY ← IY+1						
		00	100	011							
DEC dd	6	00	dd1	011	dd ← dd-1						
DEC IX	10	11	011	101	IX ← IX-1						
		00	101	011							
DEC IY	10	11	111	101	IY ← IY-1						
		00	101	011							

Таблица 5.1.6. Команды переходов

Команда	T	Код			Пояснения	C	Z	P	S	N	H
JP nn	10	11	000	011	PC ← nn						
		-	n	-							
		-	n	-							
JP cc,nn	10	11	cc	010	PC ← nn, если условие выполняется						
		-	n	-	PC ← PC+3, если нет						
		-	n	-							
JR e	12	00	011	000	PC ← PC+e						
		-	e-2	-							
JR C,e	12	00	111	000	PC ← PC+e, если C=1						
	(7)	-	e-2	-	PC ← PC+2, если C=0						
JR NC,e	12	00	110	000	PC ← PC+e, если C=0						
	(7)	-	e-2	-	PC ← PC+2, если C=1						
JR Z,e	12	00	101	000	PC ← PC+e, если Z=1						
	(7)	-	e-2	-	PC ← PC+2, если Z=0						
JR NZ,e	12	00	100	000	PC ← PC+e, если Z=0						
	(7)	-	e-2	-	PC ← PC+2, если Z=1						
JP (HL)	4	11	101	001	PC ← HL						
JP (IX)	8	11	011	101	PC ← IX						
		11	101	001	PC ← IY						
JP (IY)	8	11	111	101							
		11	101	001							

Команда	T	Код			Пояснения	C	Z	P	S	N	H
DJNZ e	13 (8)	00 -	010 e-2	000 -	B ← B-1, при B=0 PC ← PC+2 при B≠0 PC ← PC+e						

Таблица 5.1.7 Команды для работы с битами

Команда	T	Код			Пояснения	C	Z	P	S	N	H
BIT b,r	8	11 01	001 b	011 r	Z ← инверсия r _b		?	X	X	0	1
BIT b,(HL)	12	11 01	001 b	011 110	Z ← инверсия (HL) _b		?	X	X	0	1
BIT b,(IX+d)	20	11 11	011 001	101 011	Z ← инверсия (IX+d) _b		?	X	X	0	1
		-	d	-							
BIT b,(IY+d)	20	01 11	b 111	110 101	Z ← инверсия (IY+d) _b		?	X	X	0	1
		11	001	011							
		-	d	-							
		01	b	110							
SET b,r	8	11	001	011	r _b ← 1						
		<u>11</u>	b	r							
SET b,(HL)	15	11	001	011	(HL) _b ← 1						
		<u>11</u>	b	110							
SET b,(IX+d)	23	11	011	101	(IX+d) _b ← 1						
		11	001	011							
		-	d	-							
SET b,(IY+d)	23	11	111	101	(IY+d) _b ← 1						
		11	001	011							
		-	d	-							
		11	b	110							
RES b,t		10				t _b ← 0. Код 10 подставляется вместо 11 в командах SET					

Таблица 5.1.8 Команды обмена

Команда	T	Код			Пояснения	C	Z	P	S	N	H
EX DE,HL	4	11	101	011	DE ↔ HL						
EX AF,AF'	4	00	001	000	AF ↔ AF'						
EXX	4	11	011	001	BC ↔ BC' DE ↔ DE' HL ↔ HL'						
EX (SP),HL	19	11	100	011	H ↔ (SP+1) L ↔ (SP)						
EX (SP),IX	23	11	011	101	IX _H ↔ (SP+1)						
		11	100	011	IX _L ↔ (SP)						
EX (SP),IY	23	11	111	101	IY _H ↔ (SP+1)						
		11	100	011	IY _L ↔ (SP)						

Таблица 5.1.9. Команды ввода вывода

Команда	T	Код			Пояснения	C	Z	P	S	N	H
IN A,(n)	11	11	011	011	A ← (n) ADR ← A/n						
		-	n	-							
IN r,(C)	12	11 01	101 r	101 000	r ← (C) ADR ← B/C		?	P	?	0	?
INF	12	11 01	101 110	101 000	F ← (C) ADR ← B/C F-регистр флагов		?	P	?	0	?
INI	16	11 10	101 100	101 010	(HL)←(C) ADR←B/C затем B ← B-1 HL ← HL+1		?	X	X	1	X
					Z=1, если B-1=0 Z=0, если B-1≠0						
INIR	21 (16)	11 10	101 110	101 010	(HL)←(C) ADR←B/C затем B ← B-1 HL ← HL+1		1	X	X	1	X
					и повтор до B=0						
IND	16	11 10	101 101	101 010	(HL)←(C) ADR←B/C затем B ← B-1 HL ← HL-1		?	X	X	1	X
					Z=1, если B-1=0 Z=0, если B-1≠0						

Команда	T	Код			Пояснения	C	Z	P	S	N	H
INDR	21 (16)	11 10	101 111	101 010	(HL)←(C) ADR←B/C затем B ← B-1 HL ← HL-1 и повтор до B=0		1	X	X	1	X
OUT (n),A	11	11	010	011	(n) ← A ADR ← A/n						
OUT (C),r	12	11	101	101	(C) ← r ADR ← B/C						
OUTI	16	11	101	101	(C)←(HL) ADR←B/C		?	X	X	1	X
		10	100	011	затем B ← B-1 HL ← HL+1		Z=1, если B-1=0 Z=0, если B-1≠0				
OTIR	21 (16)	11 10	101 110	101 011	(C)←(HL) ADR←B/C затем B ← B-1 HL ← HL+1 и повтор до B=0		1	X	X	1	X
OUTD	16	11	101	101	(C)←(HL) ADR←B/C		?	X	X	1	X
		10	101	011	затем B ← B-1 HL ← HL-1		Z=1, если B-1=0 Z=0, если B-1≠0				
OTDR	21 (16)	11 10	101 111	101 011	(C)←(HL) ADR←B/C затем B ← B-1 HL ← HL-1 и повтор до B=0	1	X	X	1	X	1

Таблица 5.1.10. Команды вызова и возврата

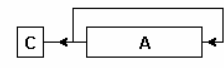
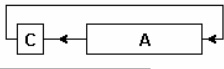
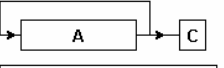

Команда	T	Код			Пояснения	C	Z	P	S	N	H
CALL nn	17	11	001	101	(SP-1) ← PC _H (SP-2) ← PC _L PC ← nn						
CALL cc,nn	17 (10)	11	cc	100	(SP-2) ← PC _L PC←nn, если cc верно PC←PC+3, если нет						
RET	10	11	001	001	PC _L ← (SP) PC _H ← (SP+1)						
RET cc	11 (6)	11	cc	000	PC _L ← (SP) PC _H ← (SP+1) PC ← PC+1, если cc не выполняется						
RETI	14	11	101	101	Возврат из INT						
RETN	14	11	101	101	Возврат из NMI						
		01	001	101							
		01	000	101							
RST p	11	11	a	111	(SP-1) ← PC _H (SP-2) ← PC _L PC ← 0 PC _L ← p						

Для некоторых команд в колонке T указано два значения - без скобок и в скобках:

- для условных команд JR, CALL, RET в скобках указано количество тактов выполнения команды при невыполнении условия;

- для "циклических" команд DJNZ, LDIR, LDDR, CPIR, CPDR, INIR, INDR, OTIR. OTDR в скобках указано количество тактов, за которое выполняется команда в последний (перед завершением) раз.

Таблица 5.1.11. Команды сдвига и вращения

Команда	T	Код			Пояснения	C	Z	P	S	N	H
RLCA	4	00	000	111		?				0	0
RLA	4	00	010	111		?				0	0
RRCA	4	00	001	111		?				0	0
RRA	4	00	011	111		?				0	0

Команда	T	Код			Пояснения	C	Z	P	S	N	H	
RLC r	8	11	001	011		?	?	P	?	0	0	
		00	<u>000</u>	r								
RLC (HL)	15	11	001	011			?	?	P	?	0	0
		00	<u>000</u>	110								
RLC (IX+d)	23	11	011	101			?	?	P	?	0	0
		11	001	011								
		-	d	-								
RLC (IY+d)	23	00	<u>000</u>	110		?	?	P	?	0	0	
		11	111	101								
		11	001	011								
		-	d	-								
		00	<u>000</u>	110								
RL t			010			?	?	P	?	0	0	
RRC t			001			?	?	P	?	0	0	
RR t			011			?	?	P	?	0	0	
SLA t			100			?	?	P	?	0	0	
SRA t			101			?	?	P	?	0	0	
SRL t			111			?	?	P	?	0	0	
RLD	18	11	101	101			?	P	?	0	0	
		01	101	111								
RRD	18	11	101	101			?	P	?	0	0	
		01	100	111								

Таблица 5.1.12. Команды управления микропроцессором

Команда	T	Код			Пояснения	C	Z	P	S	N	H
NOP	4	00	000	000	Нет операции						
HALT	4	01	110	110	Останов						
DI	4	11	110	011	Прерывания запрещены						
EI	4	11	111	011	Прерывания разрешены						
IM 0	8	11	101	101	Режим прерывания 0						
		01	000	110							
IM 1	8	11	101	101	Режим прерывания 1						
		01	010	110							
IM 2	8	11	101	101	Режим прерываний 2						
		01	011	110							

5.3. Флаги признаков

Каждый из двух флаговых регистров МП Z80 содержит по шесть информационных битов, состояние которых устанавливается в соответствии с результатом операции. Флаги S, Z, C и P/V используются в командах условного перехода, условного вызова подпрограмм и условного возврата из подпрограмм. Флаги H и N непосредственно программно не анализируются. Они используются в двоично-десятичной арифметике.

S (F7) Sign - Флаг знака

Этот Флаг предназначен для обработки чисел со знаком. Флаг устанавливается, когда результат операции отрицателен. Т.к. бит 7 является знаковым (любое отрицательное число содержит единицу в 7 бите), то этот флаг копирует состояние 7 бита аккумулятора.

При программировании состояние этого флага условно кодируется как P (Plus, знак плюс, S=0) и M (Minus, знак минус, S=1).

Z (F6) Zero - Флаг нуля

Флаг Z устанавливается или обнуляется в результате выполнения следующих команд:

1. При 8-битных арифметических и логических операциях флаг устанавливается в 1, если результат выполнения команды равен нулю. Если результат не равен нулю, флаг Z обнуляется.

2. При командах поиска и сравнения флаг Z устанавливается в 1, когда содержимое ячейки памяти адресуемой регистровой парой HL, совпадает с содержимым аккумулятора.

3. После выполнения команды тестирования отдельного бита во флаг Z записывается инвертированное значение проверенного бита.

4. Если при выполнении команд INI, IND, OUTI, OUTD результат декремента (B-1) не равен нулю, то флаг Z=0. Когда результат (B-1)=0, флаг Z устанавливается.

5. Флаг Z устанавливается в 1, когда в результате выполнения команд IN r,(C) содержимое регистра r обнуляется.

При программировании этот флаг следует условно кодировать как Z (Zero, результат равен нулю, Z=1) и NZ (No Zero, результат не равен нулю, Z=0)

C (F0) Carry - Флаг переноса

На состояние этого флага влияют следующие команды:

1. Команды сложения устанавливают флаг C, когда есть Перенос от бита 7 к биту 8 результата сложения однобайтных чисел, или от бита 15 к биту 16 при сложении двухбайтных чисел. Иначе C=0.

2. Команды вычитания устанавливают флаг, если есть заём, и сбрасывают, когда его нет.

3. Команды RLA, RRA, RL t и RR t используют флаг C как бит связи между 7 и 0 битами.

4. После выполнения команд RLCA, RLC t, SLA t флаг C копирует значение бита 7 операнда.

5. После выполнения команд RRCA, RRC t, SRA t, SRL t флаг C копирует значение бита 0 операнда.

6. Логические команды AND s, OR s и XOR s обнуляют флаг C.

7. Команда SCF устанавливает флаг C, а команда CCF его инвертирует.

При программировании флаг кодируется как C (Carry, был перенос, C=1) и NC (No Carry, не было переноса, C=0).

P/V (F2) Parity/Overflow - Флаг чётности/переполнения

Этот флаг имеет несколько функций:

1. При арифметических операциях над числами со знаком он действует как флаг переполнения V. т.к. диапазон представления чисел со знаком от -128 до +127, то при операциях над ними возможен выход результата за эти пределы - переполнение. В этом случае флаг, устанавливается в 1, сигнализируя об ошибочности результата. Логика установки флага V в 1 такова:

а) если был перенос из 6-го бита в 7-ой, и не было переноса из 7-го бита во флаг C.

б) не было переноса из 6-го бита в 7-ой, но есть перенос из 7-го бита во флаг C.

Примеры:

1. Сложение	+76=	01001100
	-114=	<u>10001110</u>
C=0		11011010 = -38 верно
V=0		
2. Сложение	-1=	11111111
	-1=	<u>11111111</u>
C=1		11111110 = -2 верно
V=0		
3. Сложение	+82=	01010010
	+94=	<u>01011110</u>
C=0		10110000 = -80 неверно!
V=1		

В последнем примере произошла установка флага V. Это означает, что ошибка должна быть программно учтена при анализе результата.

2. При выполнении логических операций, команд вращения и сдвига, а также команды IN r,(C) флаговый бит 2 действует как флаг чётности P. Он устанавливается, когда регистр содержит четное количество единиц, и обнуляется при нечетном количестве единиц.

3. При выполнении команд LD A,I и LD A,R флаг P/V копирует состояние триггера прерываний IFF2, что позволяет в любое время узнать состояние процессора.

4. Во время выполнения команд поиска CPI, CPIR, CPD и CPDR, а также команд пересылки блоков LDI, LPIR, LDD и LDDR флаг P/V=1, пока содержимое счётчика байтов (регистр BC) отлично от нуля. Когда BC=0, флаг P/V сбрасывается.

При программировании этот флаг условно кодируется как PO (Parity Odd, нечётно, P/V=0) и PE (Parity Even, чётно, P/V=1).

H (F4) Halfcarry - Флаг полупереноса

Команды сложения, вычитания, инкремента, декремента и сравнения 8-битных операндов устанавливают флаг H когда был перенос из третьего бита результата в четвертый, или был заём от четвертого бита к третьему. Этим он указывает команде DAA требует ли результат десятичной коррекции.

N (F1) Addition/Subtraction - Флаг сложения/вычитания

Команды сложение, инкремента и логические команды обнуляют флаг N, а команды вычитания, декремента, сравнения и инвертирования его устанавливают.

Состояние флага N используется командой DAA. Т.к. алгоритм корректировки результата двоично-десятичных операций различен при сложении и вычитании, этот флаг показывает, какое именно действие происходило перед этим.

В таблице 5.2 показано как выполнение команд влияет на регистр флагов. Команды, которые здесь не представлены, не влияют ни на один флаг.

Таблица 5.2. Флаги признаков.

Группа команд	Регистр Флагов								Пояснения
	F7 S	F6 Z	F5	F4 H	F3	F2 P/V	F1 N	F0 C	
LD A,I; LD A,R	?	?	X	0	X	IF	0		P/V ← IFF2
LDI; LDD	X	X	X	0	X	?	0		Если BC≠0, то P/V=1
LDIR; LDDR	X	X	X	0	X	0	0		Если BC=0, то P/V=0
CPI; CPD; CPIR; CPDR	X	?	X	X	X	?	1		Если BC≠0, то P/V=1 Если BC=0, то P/V=0 Если A≠(HL), то Z=0 Если A=(HL), то Z=1
ADD s; ADC s	?	?	X	?	X	V	0	?	8-битовые арифметические команды
SUB s; SBC s; CP s; NEG	?	?	X	?	X	V	1	?	
AND s	?	?	X	1	X	P	0	0	Логические команды
OR s; XOR s	?	?	X	0	X	P	0	0	
INC t	?	?	X	?	X	V	0		Только 8-битовые команды
DEC t	?	?	X	?	X	V	1		
ADD HL,dd; ADD IX,pp; ADD IY,rr			X	X	X		0	?	16-битовые – арифметические команды
ADC HL,dd	?	?	X	X	X	V	0	?	
SBC HL,dd	?	?	X	X	X	V	1	?	
DAA	?	?	X	?	X	P		?	
CPL			X	1	X		1		
CCF			X	X	X		0	?	
SCF			X	0	X		0	1	
RLCA; RLA; RRCA; RRA			X	0	X		0	?	Вращение аккумулятора
RLC t; RRC t; RL t; RR t; SLA t; SRA t; SRL t	?	?	X	0	X	P	0	?	Группа команд вращения и сдвига
RLD; RRD	?	?	X	0	X	P	0		Перестановка тетрад Z ← инверсия s _b
BIT b,t	X	?	X	1	X	X	1		
IN r,(C); INF	?	?	X	?	X	P	0		
INI; IND; OUTI; OUTD	X	?	X	X	X	X	1		Блочные команды ввода-вывода: Если B≠0, то Z=0 Если B=0, то Z=1
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1		

5.4. Очередность выполнения по циклам

Для обозначения действий микропроцессора в каждой машинном цикле в таблице 5.3 приняты следующие сокращения:

- OCR (operation code read) - чтение кода операции;
 CBR (control byte read) - чтение управляющего байта кода операции (ED, DD, FD, CB),
 IOP (internal CPU-operation) - внутренние операции ЦП, выходные сигналы управления неактивны;
 MRD (memory read) - чтение байта из косвенно адресованной ячейки памяти;
 MRH (memory read, high byte) - чтение старшего байта 16-разрядного слова из косвенно адресованной ячейки памяти;
 MRL (memory read, low byte) - чтение младшего байта 16-разрядного слова из косвенно адресованной ячейки памяти;
 MWR (memory write) - запись байта в косвенно адресованную ячейку памяти;
 MWH (memory write, high byte) - запись старшего байта 16-разрядного слова в косвенно адресованную ячейку памяти,
 MWL (memory write, low byte) - запись младшего байта 16-разрядного слова в косвенно адресованную ячейку памяти;
 ORD (operand read) - чтение операнда-байта, непосредственно представленного в команде;
 ORH (operand read, high byte) - чтение старшего байта 16-разрядного слова, непосредственно представленного в команде;
 ORL (operand read, low byte) - чтение младшего байта 16-разрядного слова, непосредственно представленного в команде;
 PRD (port read) - ввод данных из порта;
 PWR (port write) - вывод данных в порт;
 SRH (stack read, high byte) - чтение из стека старшего байта;
 SRL (stack read, low byte) - чтение из стека младшего байта;
 SWH (stack write, high byte) - запись в стек старшего байта;
 SWL (stack write, low byte) - запись в стек младшего байта;
 INTA (interrupt acknowledge) - подтверждение прерывания микропроцессором;
 SPI (stack pointer increment) - инкремент указателя стека в конце цикла;
 SPD (stack pointer decrement) - декремент указателя стека в конце цикла;
 (n) - продолжительность цикла в периодах тактовой частоты.

Таблица 6.3. очередность выполнения по циклам

Тип команды	Байт	Цикл M1	Цикл M2	Цикл M3	Цикл M4	Цикл M5	Пояснение
LD r1,r2	1	OCR(4)					
LD r,n	2	OCR(4)	ORD(3)				
LD r,(HL)	1	OCR(4)	MRD(3)				
LD (HL),r	1	OCR(4)	MWR(3)				
LD r,(ii+d)	3	CBR(4); OCR(4)	ORD(3)	IOP(5)	MRD(3)		ii: IX или IY
LD (ii+d),r	3	CBR(4); OCR(4)	ORD(3)	IOP(5)	MWR(3)		
LD (HL),n	2	OCR(4)	ORD(3)	MWR(3)			
LD A,(BC); LD A,(DE)	1	OCR(4)	MRD(3)				
LD (BC),A; LD (DE),A	1	OCR(4)	MWR(3)				
LD A,(nn)	3	OCR(4)	ORL(3)	ORH(3)	MRD(3)		
LD (nn),a	3	OCR(4)	ORL(3)	ORH(3)	MWR(3)		
LD A,i	2	CBR(4); OCR(5)					i: I или R
LD i,A	2	CBR(4); OCR(5)					i: I или R
LD dd,nn	3	OCR(4)	ORL(3)	ORH(3)			
LD ii,nn	4	CBR(4); OCR(4)	ORL(3)	ORH(3)			ii: IX или IY
LD HL,(nn)	3	OCR(4)	ORL(3)	ORH(3)	MRL(3)	MRH(3)	
LD (nn),HL	3	OCR(4)	ORL(3)	ORH(3)	MWL(3)	MWH(3)	
LD dd,(nn)	4	CBR(4); OCR(4)	ORL(3)	ORH(3)	MRL(3)	MRH(3)	
LD (nn),dd	4	CBR(4); OCR(4)	ORL(3)	ORH(3)	MWL(3)	MWH(3)	
LD ii,(nn)	4	CBR(4); OCR(4)	ORL(3)	ORH(3)	MRL(3)	MRH(3)	ii: IX или IY
LD (nn),ii	4	CBR(4); OCR(4)	ORL(3)	ORH(3)	MWL(3)	MWH(3)	ii: IX или IY

Тип команды	Байт	Цикл М1	Цикл М2	Цикл М3	Цикл М4	Цикл М5	Пояснение
LD SP,HL LD SP,ii	1 2	OCR(6) OCR(6)					ii: IX или IY
PUSH qq PUSH ii POP qq POP ii	1 2 1 2	OCR(5); SPD CBR(4); OCR(5); SPD OCR(4) CBR(4); OCR(4)	SWH(3); SPD SWH(3); SPD SRL(3); SPI SRL(3); SPI	SWL(3) SWL(3) SRH(3); SPI SRH(3); SPI			ii: IX или IY ii: IX или IY
EX DE,HL EX AF,AF'; EXX EX (SP),HL EX (SP),ii	1 1 1 1	OCR(4) OCR(4) OCR(4) CBR(4); OCR(4)					ii: IX или IY
LDI; LDD; CPI; CPD LDIR; LDDR; CPIR; CPDR	2 2	CBR(4); OCR(4) CBR(4); OCR(4)	MRD(3) MRD(3)	MWR(3) MWR(3)			IOP(5) в М4 только когда ВС≠0
ALU r ALU n ALU (HL) ALU (ii+d)	1 2 1 3	OCR(4) OCR(4) OCR(4) CBR(4); OCR(4)					ALU означает: ADD, ADC, SUB, SEC, AND, OR, XOR или CP
INC r; DEC r INC (HL); DEC (HL) INC (ii+d); DEC (ii+d)	1 1 3	OCR(4) OCR(4) CBR(4); OCR(4)	MRD(3) MRD(3) ORD(3)	MWR(3) MWR(3) IOP(5)			ii: IX или IY
DAA CPL NEG CCF; SCF	1 1 2 1	OCR(4) OCR(4) CBR(4); OCR(4) OCR(4)					
NOP; HALT	1	OCR(4)					
DI; EI IM0; IM1; IM2	1 2	OCR(4) CBR(4); OCR(4)					
ADD HL,dd ADC HL,dd; SBC HL,dd; ADD ii,pp	1 2	OCR(4) CBR(4); OCR(4)	IOP(4) IOP(4)	IOP(3) IOP(3)			ADD ii,pp – это ADD IX,pp или ADD IY,rr
INC dd; DEC dd INC ii; DEC ii	1 2	OCR(6) CBR(4); OCR(6)					ii: IX или IY
RLCA; RRCA; RLA; RRA ROT r ROT (HL) ROT (ii+d) RLD; RRD	1 2 2 4 2	OCR(4) CBR(4); OCR(4) CBR(4); OCR(4) CBR(4); CBR(4) CBR(4); OCR(4)					ROT означает: RLC, RL, RRC, RR, SLA, SRA или SRL.
BIT b,r; SET b,r RES b,r BIT b,(HL) SET b,(HL) RES b,(HL) BIT b,(ii+d) SET b,(ii+d); RES b,(ii+d)	2 2 2 4 4	CBR(4); OCR(4) CBR(4); OCR(4) CBR(4); OCR(4) CBR(4); CBR(4) CBR(4); CBR(4)	MRD(4) MRD(4) MRD(4) ORD(3) ORD(3)	MWR(3) MWR(3) MWR(3) OCR(5) OCR(5)			ii: IX или IY
JP nn; JP cc,nn JR e JR C,e; JR Z,e; JR NC,e; JR NZ,e JP (HL) JP (ii) DJNZ e	3 2 2 1 2 2	OCR(4) OCR(4) OCR(4) OCR(4) CBR(4); OCR(4) OCR(5)	ORL(3) ORD(3) ORD(3) ORD(3)	ORH(3) IOP(5) IOP(5) IOP(5)			IOP(5) – при невьполнении условия ii: IX или IY IOP(5), если B≠0
CALL nn CALL cc,nn CALL cc,nn	3 3 3	OCR(4) OCR(4) OCR(4)	ORL(3) ORL(3) ORL(3)	ORH(4); SPD ORH(4); SPD ORH(3)	SWH(3); SPD SWH(3); SPD	SWL(3) SWL(3)	cc cc не выполняется
RET RET cc	1 1	OCR(4) OCR(4)	SRL(3); SPI SRL(3); SPI	SRH(3); SPI SRH(3); SPI			cc

Тип команды	Байт	Цикл М1	Цикл М2	Цикл М3	Цикл М4	Цикл М5	Пояснение
RET сс	1	OCR(4)					сс не выполняется
RETI; RETN	2	CBR(4); OCR(4)	SRL(3); SPI	SRH(3); SPI			
RST p	1	OCR(5); SPD	SWH(3); SPD	SWL(3)			
IN A,(n)	2	OCR(4)	ORD(4)	PRD(4)			
IN r,(C); INF	2	CBR(4); OCR(4)	PRD(4)				
INI; IND	2	CBR(4); OCR(5)	PRD(4)	MWR(3)			
INIR; INDR	2	CBR(4); OCR(5)	PRD(4)	MWR(3)	IOP(5)		IOP(5), если B≠0
OUT (n),A	2	OCR(4)	ORD(4)	PWR(4)			
OUT (C),r	2	CBR(4); OCR(4)	PWR(4)				
OUTI; OUTD	2	CBR(4); OCR(5)	MRD(3)	PWR(4)			
OTIR; OTDR	2	CBR(4); OCR(5)	MRD(3)	PWR(4)	IOP(5)		IOP(5), если B≠0
Подтверждение прерывания:							
NMI	-	OCR(5); SPD	SWH(3); SPD	SWL(3)			См. рис. 6.4
INT, Режим 0	-	INTA(6); SPD	SWH(3); SPD	SWL(3)			При команде RST p
	-	INTA(6)	ORL(3)	ORH(4); SPD	SWH(3); SPD	SWL(3)	При команде CALL
INT, Режим 1	-	INTA(7); SPD	SWH(3); SPD	SWL(3)			RST 38H
INT, Режим 2	-	INTA(7); SPD	SWH(3); SPD	SWL(3)	MRL(3)	MRH(3)	См. рис. 6.13

Принятые обозначения отражают программную суть происходящих в данном цикле действий. С точки зрения процессов на шинах все они сводятся к машинным циклам, рассмотренным в главе 4:

OCR, CBR - цикл извлечения кода операции (рис. 4.2)

MRD, MRH, MRL, - цикл чтения из памяти (рис. 4.4)

ORD, ORH, ORL,

SRH, SRL

MWR, MWH, MWL - цикл записи в память (рис. 4.4)

SWH, SWL

PRD - цикл ввода (рис. 4.6)

PWR - цикл вывода (рис. 4.6)

INTA - цикл подтверждения маскируемого прерывания (рис 4.9).

6. Система прерываний

Прерывания служат для приостановки прямого выполнения программы, с тем, чтобы процессор смог отреагировать на определенный запрос, сформированный периферийным устройством в зависимости от какого-либо обстоятельства. Реакция ЦП на запрос выражается в переходе к выполнению некоторой программы, которая называется программой обработки прерывания (Interrupt- Service Routine - ISR). После её окончания продолжается выполнение прерванной программы². Путём присвоения приоритетов запросам можно добиться, чтобы ЦП реагировал, прежде всего, на наиболее важное событие, игнорируя запросы либо прерывая ISR менее важных.

В отличие от циклического опроса, при котором программно проверяется выполнение определенного условия, и для которого требуется время выполнения и место в памяти, прерывание обеспечивает практически мгновенную реакцию, т.к. не зависит от периода опроса, отсутствие программы опроса, как таковой, выражается в экономии памяти.

Различные варианты применения прерываний являются эффективным средством повышения производительности микропроцессорной системы.

6.1. Разрешение и запрещение прерываний

Для приёма запросов на прерывания микропроцессор Z80 имеет два входа: INT и NMI. Различие между ними - в приоритете и маскируемости.

INT (маскируемое прерывание) может быть программно запрещено или разрешено. Необходимость в запрещении (маскировании) прерываний возникает, например, когда условия работы в реальном масштабе времени делают нежелательным прерывание данного участка программы.

Состояние "запретить прерывания" либо "разрешить прерывания" запоминается программно-доступный триггером IFF1 (Interrupt Flip Flop) внутри ЦП. Посредством команды разрешения прерываний EI (Enable Interrupt) либо запрещения прерываний DI (Disable Interrupt) этот триггер соответственно устанавливается либо сбрасывается. В некоторых случаях для сохранения его текущего состояния требуется промежуточная память, в качестве которой предусмотрен триггер IFF2.

При сбросе ЦП сигналом /RESET оба триггера сбрасываются, блокируя требования на маскируемые прерывания. Посредством команды EI их можно разблокировать. Когда прерывание принимается, IFF1 и IFF2 также автоматически сбрасываются, чтобы предотвратить дальнейшие прерывания до тех пор, пока они не будут разрешены новой командой EI.

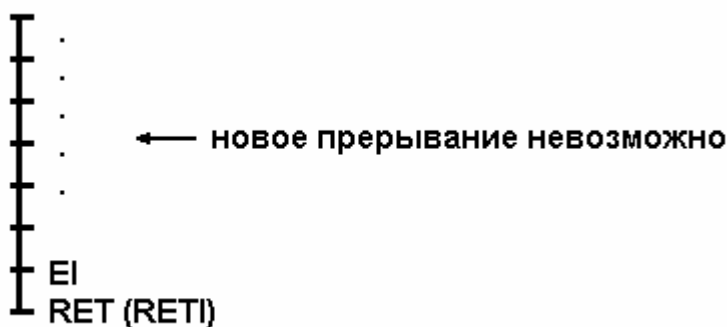
Применение EI внутри ISR дает следующие две возможности:

² Для корректного возврата в прерванную программу предусмотрено следующее:

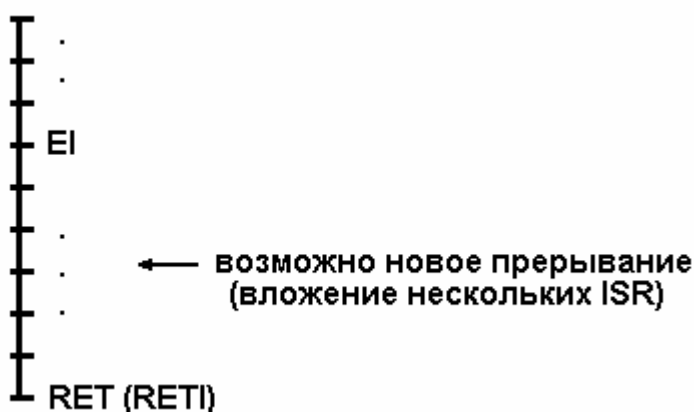
- при входе в ISR адрес возврата автоматически сохраняет я в стеке;
- внутри ISR программно предпринимаются меры для сохранения содержимого регистров ЦП двумя способами:

- а) смена блока регистров командами EXX, EX AF, AF'.
- б) пересылка в стек командой PUSH.

а) Начало ISR



б) Начало ISR



Важное обстоятельство: при выполнении команды EI поступивший в это время запрос на прерывание будет блокирован до тех пор, пока не выполнится следующая за EI команда. Цель такой задержки состоит в том, чтобы гарантировать (после команды EI) возможность выполнения команды возврата из ISR (RET или RETI).

NMI (немаскируемое прерывание) имеет более высокий по отношению к INT приоритет и не может быть запрещено программным способом. Таким образом, если оно затребовано каким-либо периферийным устройством, то ЦП, безусловно, прерывает текущую программу. Этот вид прерываний предназначен, как правило, для очень важных событий (отказ, спад напряжения питания и др.).

В отличие от INT, когда состояния IFF1 и IFF2 совпадают, при подтверждении немаскируемого прерывания IFF1 сбрасывается, а IFF2 остается без изменений. Это делается для того, чтобы на время обработки NMI сохранить состояние IFF1, которое имелось до приема /NMI. После окончания NMI-SR по команде возврата RETN состояние IFF1 восстанавливается из IFF2.

Команды LD A,I и LD A,R предоставляют возможность тестирования IFF2: они пересылают содержимое IFF2 в P/V-флаг. Т. о значение IFF2 может использоваться для ветвления программы.

В таблице 6.1 в компактном виде представлены все действия, влияющие на состояние триггеров разрешения прерываний.

Таблица 6.1. Состояние триггеров разрешения прерываний

Операция	IFF1	IFF2	Примечание
Сброс ЦП сигналом /RESET	0	0	Запрещение INT при пуске системы
Команда EI	1	1	Разрешение INT (задерживается на одну команду)
Команда DI	0	0	Запрещение INT
Прием INT	0	0	Прием INT и обработка прерывания
Команда RETI	■	■	Выход из ISR
Прием NMI	0	■	Прием NMI и обработка прерывания
Команда RETN	IFF1 ← IFF2	■	Выход из NMI-SR
Команда LD A,I	■	■	P/V ← IFF2
Команда LD A,R	■	■	P/V ← IFF2

■ состояние триггера не изменяется

6.2. Приём запросов в ЦП

Реакция микропроцессора на поступивший запрос прерывания зависит от точного времени поступления этого запроса, а также запросов с более высокими приоритетами.

Кроме двух входов прерывания у микропроцессора Z80 есть ещё вход запроса шины для ПДП (/BUSRQ), который имеет более высокий приоритет и, следовательно, тоже влияет на разрешение/запрещение прерывания.

На рис.6.1 в упрощенном виде показаны основные аппаратные средства микропроцессора, предназначенные для приёма запросов /BUSRQ, /NMI, /INT. Каждая линия запроса снабжена соответствующим триггером приёма (Flip-Flop): BUSRQ-FF, NMI-FF, INT-FF, в которые по нарастающему фронту последнего такта (TL) машинного цикла вводятся информация о запросах. Установка какого-либо из этих триггеров означает приём соответствующего запроса в ЦП. Далее принятые запросы обрабатываются устройством управления, где в соответствии с указанными приоритетами подтверждается в первую очередь тот или иной запрос, и вырабатываются соответствующие внутренние и внешние сигналы управления.

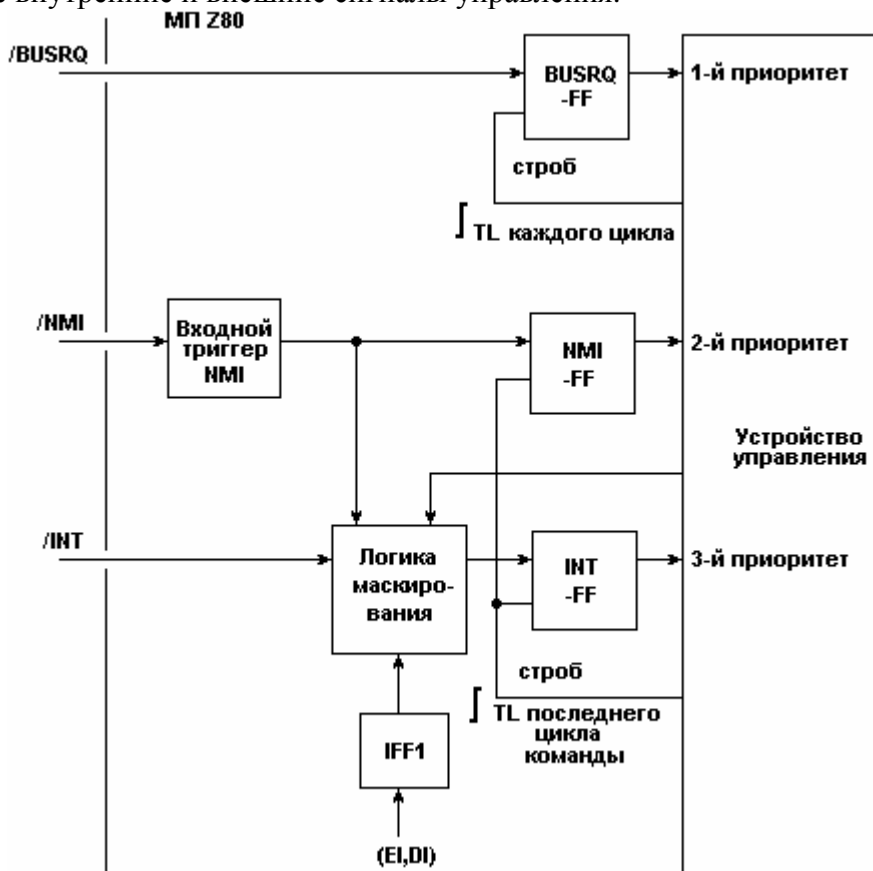


Рис. 6.1. Прием запросов /BUSRQ, /NMI, /INT

Линии приема запросов имеют существенные различия.

Триггером BUSRQ-FF анализируется уровень сигнала /BUSRQ по нарастающему фронту последнего такта (TL) каждого машинного цикла³.

Активный уровень сигнала /NMI сразу фиксируется входным триггером независимо от времени его поступления. Его Состояние (а не уровень /NMI) анализируется триггером NMI-FF по нарастающему фронту TL последнего цикла команды.

Уровень сигнала /INT до приёма в INT-FF анализируется схемой логики маскирования, где он может быть заблокирован поступающими сюда же сигналами:

- от триггера разрешения/запрещения IFF1,
- от входного триггера NMI,
- от устройства управления, который сигнализирует об обработке запросов с высшими

³ Временные диаграммы процессов при запросе шины изображены на рисунке 4.8. Описание режима ПДП для полной конфигурации системы содержится в книге 6 "Контроллер ПДП Z80DMA" и книге 7 "Z80/Построение систем. Программирование. Отладка".

приоритетами. Уровень результирующего сигнала анализируется триггером INT-FF по нарастающему фронту TL последнего цикла команды.

Приоритет /BUSRQ по отношению к /NMI и /INT состоит в следующем. При одновременном поступлении запроса шины и одного из запросов прерывания подтверждается запрос /BUSRQ, и шина предоставляется для ПДП. Запрос шины может прервать обработку NMI-SR или ISR на любой цикле. Во время ПДП прерывания от входов /NMI и /INT не обслуживаются.

Приоритет /NMI по отношению к /INT состоит в том, что если оба запроса поступили до нарастающего фронта такта TL последнего цикла команды, то /INT не воспринимается внутренним триггером INT-F/F и, следовательно, игнорируется до конца NMI-SR. Если же в течение одной команды запрос INT поступил до нарастающего фронта TL, а NMI - после, то выполняется первая команда программы обработки INT, и только потом - переход к программе обработки NMI.

При подтверждении одного из запросов прерывания (NMI либо INT) процессор вырабатывает соответствующий цикл подтверждения (см. в конце таблицы 5.3).

Все реакция ЦП на сигналы управления /BUSRQ, /NMI и /INT с учётом граничных условий представлены в таблице 6.2, а также в виде алгоритма на рисунке 6.2.

Таблица 6.2. Реакция ЦП на запросы /BUSRQ, /NMI, /INT

Сигнал	Поступление сигнала	Приём сигнала	Действие	Примечание
/BUSRQ	До нарастающего фронта такта TL	При нарастающем фронте такта TL любого машинного цикла	Подтверждение (/BUSAK) в следующем машинном цикле	/BUSRQ имеет более высокий приоритет, чем /NMI и /INT
/NMI	Внутри цикла до нарастающего фронта такта TL	Сразу фиксируется во входном триггере NMI (мин. ширина импульса 80 мс)	Непосредственно после обработки этой команды – переход к программе обработки NMI (так же непосредственно после EI)	Для повторного выполнения программы обработки NMI сигнала /NMI должен быть выключен на короткое время
-/-	Внутри цикла после нарастающего фронта TL	--/--	После обработки этой команды выполняется следующая, потом переход к программе обработки NMI	--/--
/INT	До нарастающего фронта такта TL	Только при нарастающем фронте последнего такта команды (TL)	После обработки команды, если прерывания были разрешены, переход к ISR. Если прерывания запрещены - /INT игнорируется	При выполнении команды EI /INT не прерывается, независимо от состояния IFF1 и IFF2. Он может быть принят при выполнении следующей за EI команды (если будет активен до нарастающего фронта TL этой команды)
-/-	После нарастающего фронта такта TL	--/--	Блокируется независимо от состояния IFF1 и IFF2	Может быть принят при выполнении следующей команды, если будет активен до нарастающего фронта её последнего такта TL

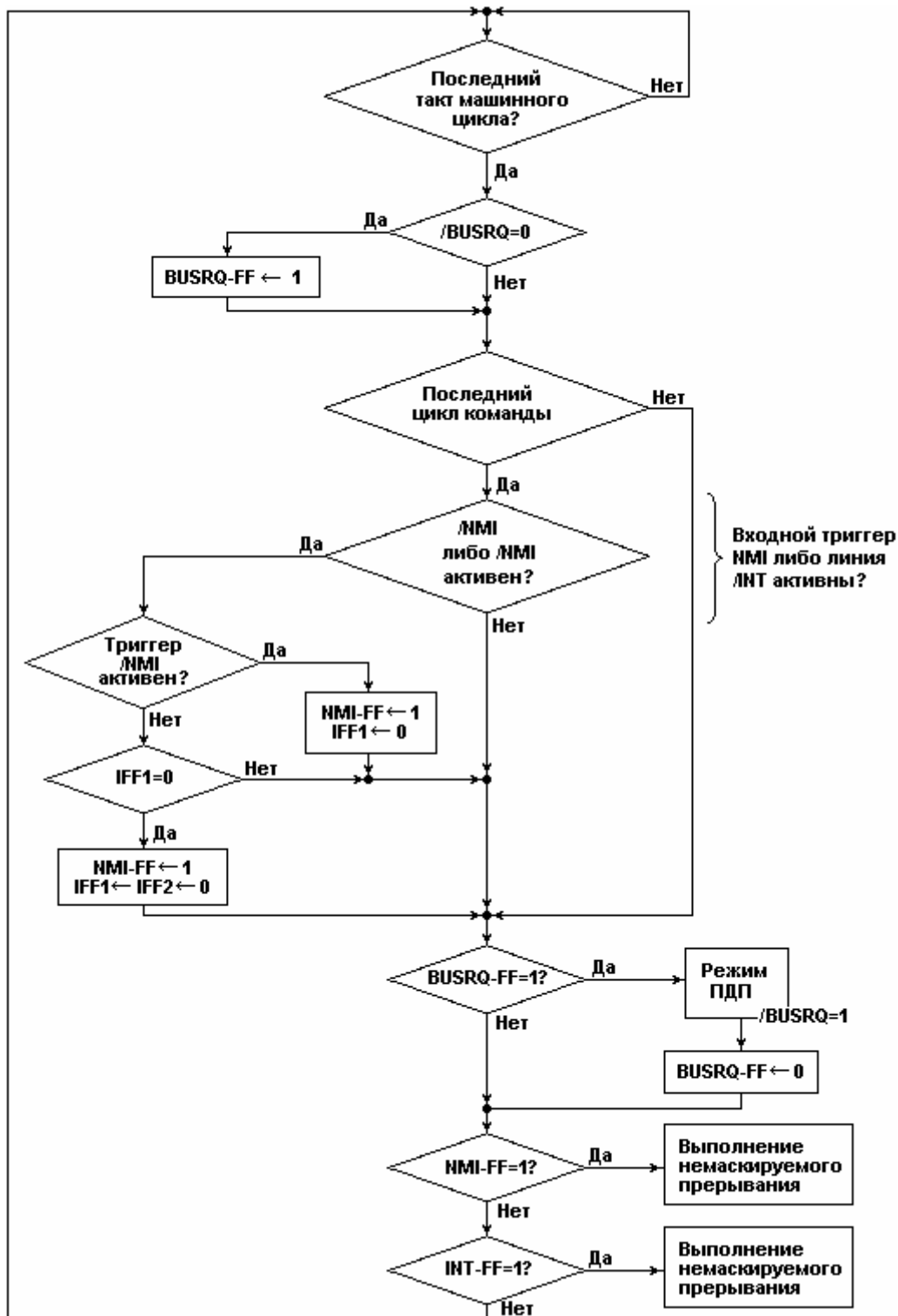


Рис. 6.2. Алгоритм обработки запросов /BUSRQ, /NMI, /INT

6.3. Обработка прерываний

6.3.1. Обработка немаскируемого прерывания

Если принят запрос NMI, то по завершении текущей команды состояние счётчика команд PC автоматически сохраняется в стеке, и в PC заносится адрес 0066H. Т.о. происходит перезапуск процессора с адреса 66H. С этого места в памяти должна начинаться подпрограмма обработки NMI; завершаться она должна командой возврата RETN. Алгоритм обработки NMI изображен на рисунке 6.3, временные диаграммы вызова подпрограммы и возврата из неё - на рисунке 6.4 и рисунке 6.5 соответственно.

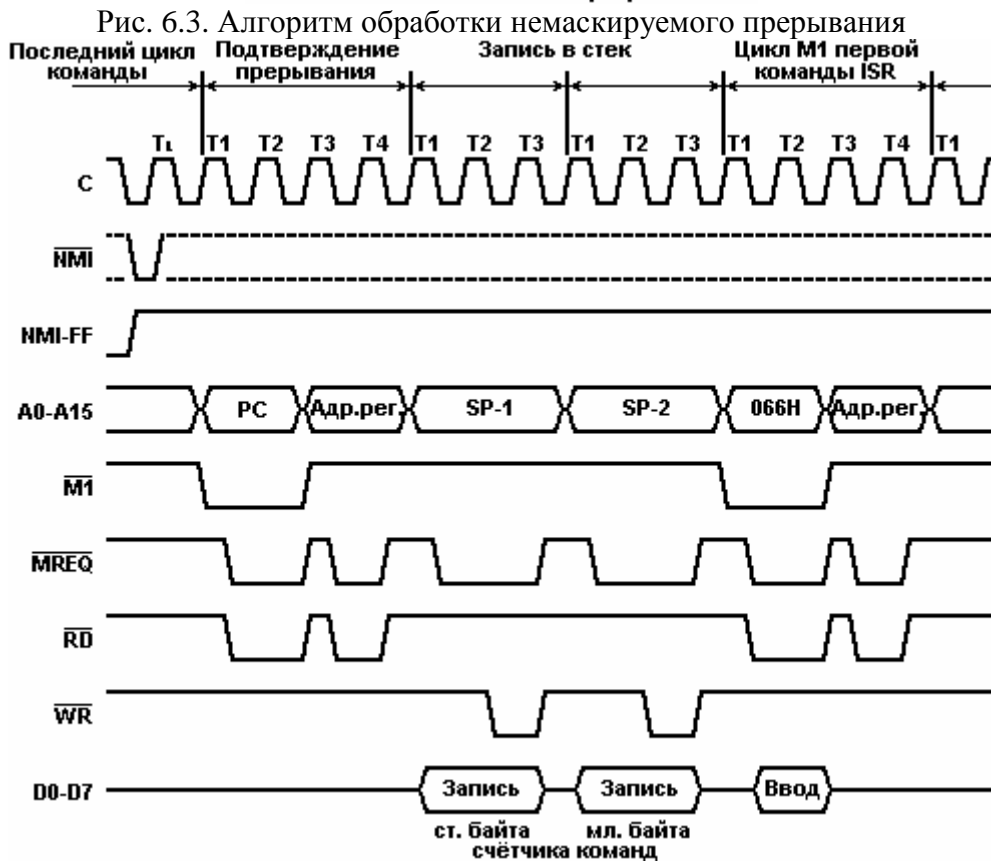
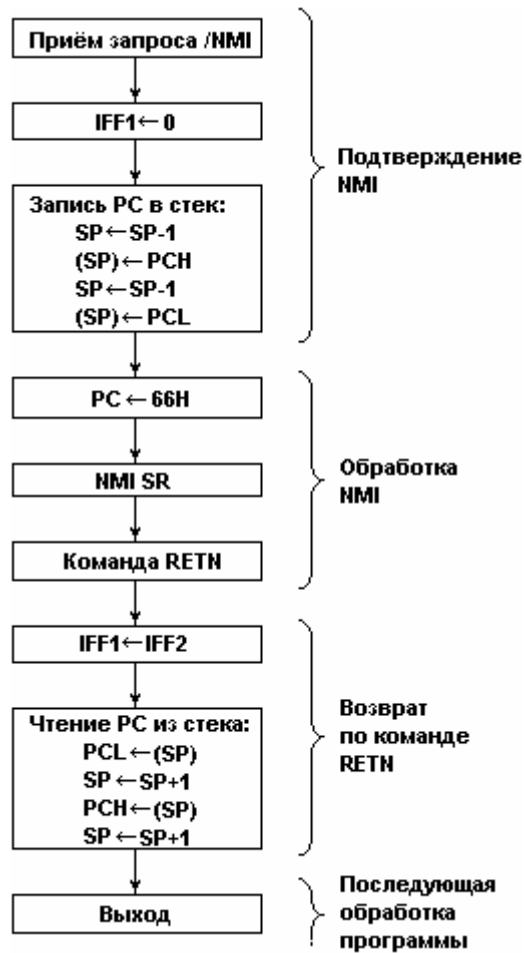


Рис. 6.4. Вызов программы обработки немаскируемого прерывания

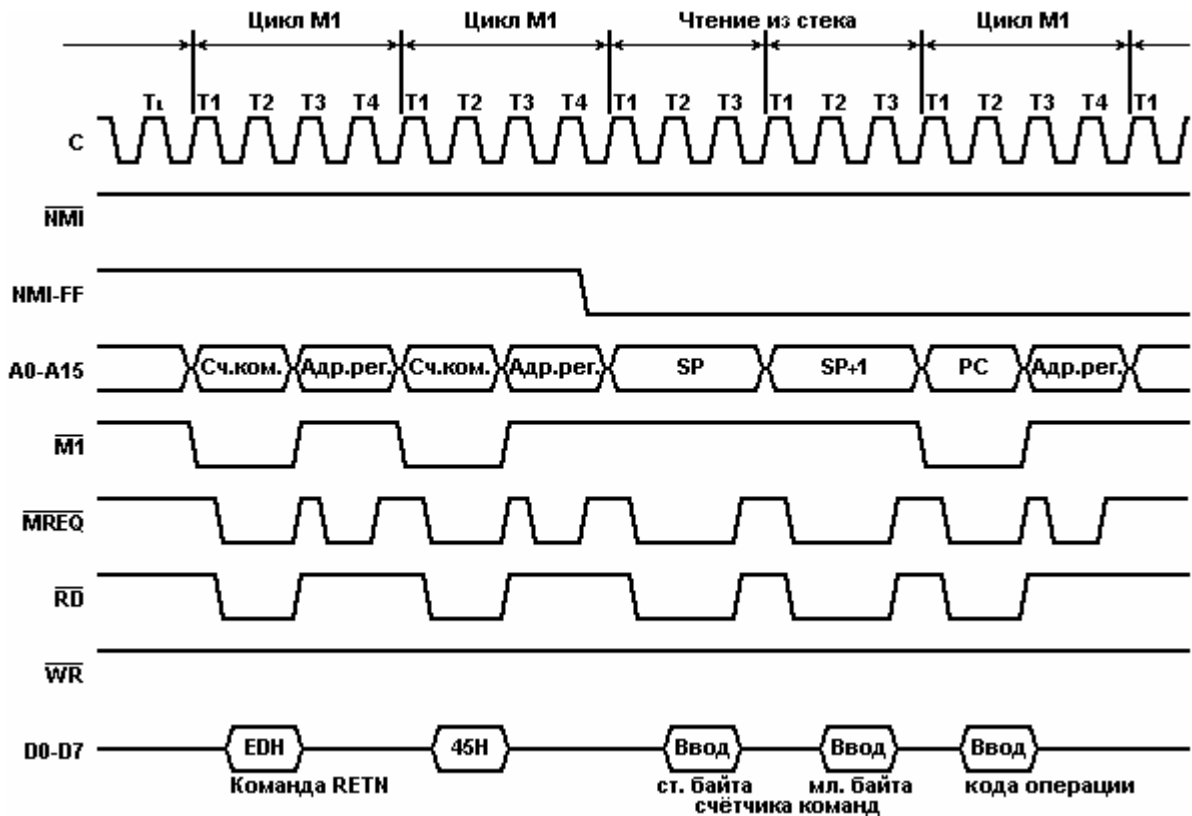


Рис. 6.5. Возврат из подпрограммы обработки немаскируемого прерывания

6.3.2. Маскируемое прерывание. Режим 0

Этот режим аналогичен реализованному в МП Intel 8080. Для него необходима внешняя вспомогательная логика, которая в случае прерывания подает команду на шину данных ЦП. Передача этой команды происходит в цикле подтверждения INTA, когда одновременно активизируются сигналы /M1 и /IORQ (см. рисунки 4.9 и 6.6).



Рис. 6.6. Подача запроса на прерывание и формирование сигнала подтверждения прерывания

Т.о. происходит чтение кода операции, но не из памяти, а из прерывающего устройства. Это может быть любая команда, однако, наиболее эффективно в этом случае использование одной из восьми команд повторного запуска RST p, где p - адрес рестарта. Для использования этих команд на ШД должны быть поданы следующие комбинации битов:

Команда	D7	D0
RST 00H	11	<u>000</u> 111
RST 08H	11	<u>001</u> 111
RST 10H	11	<u>010</u> 111
RST 18H	11	<u>011</u> 111
RST 20H	11	<u>100</u> 111
RST 28H	11	<u>101</u> 111
RST 30H	11	<u>110</u> 111
RST 38H	11	<u>111</u> 111

Алгоритм обработки изображен на рис.6.7. Если в подпрограмме присутствует команда EI, то после выполнения следующей за ней команды происходит установка IFF1, IFF2. Эти блоки отмечены пунктиром.

Как и для МП I8080, в этом режиме возможно использование контроллера прерываний 8259 (580BH59). Этот вариант подробно рассмотрен в книге 7.

Режим 0 автоматически устанавливается при выполнении общего сброса, а также соответствующей командой IM0 (Interrupt Mode 0).

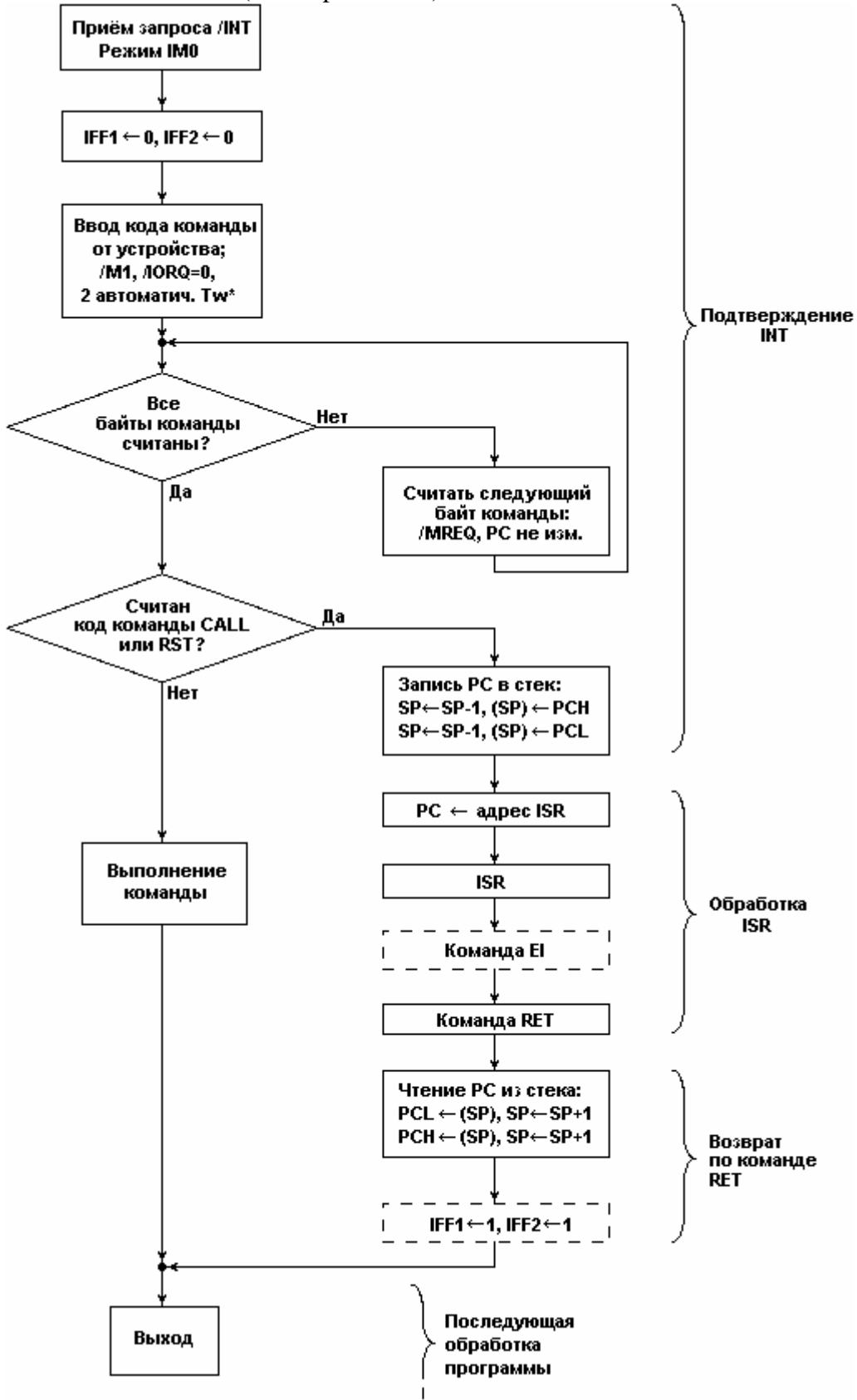


Рис. 6.7. Алгоритм обработки маскируемого прерывания. Режим 0.

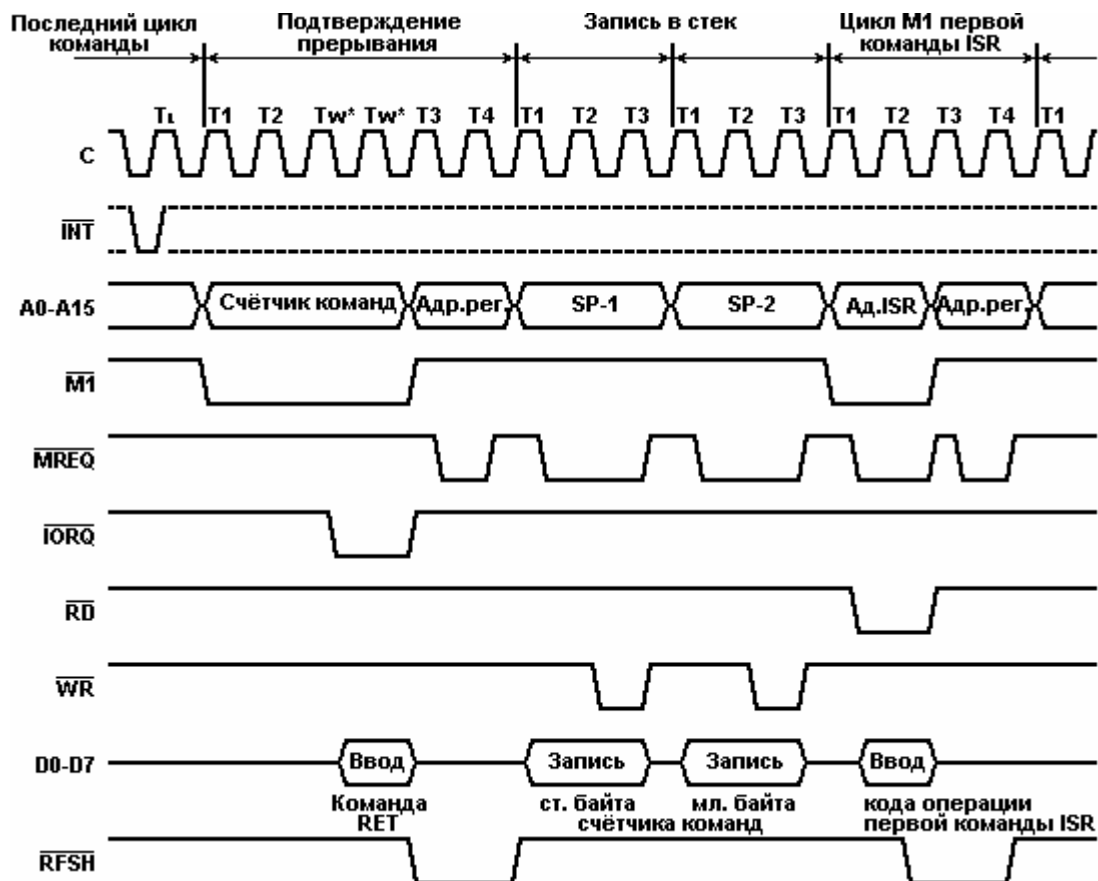


Рис. 6.6. Вызов программы обработки маскируемого прерывания ISR в режиме 0 (IM 0)

6.3.3. Маскируемое прерывание. Режим 1

Режим прерываний 1 устанавливается командой IM 1. Он применяется в системах с минимальной конфигурацией, где нежелательно наращивание аппаратной части введением вспомогательной логики. Обработка сходка с NMI, за исключением того, что вместо адреса 0066H происходит рестарт к адресу 0038H и цикл подтверждения продлевается на два такта ожидания. Возврат - по команде RET.

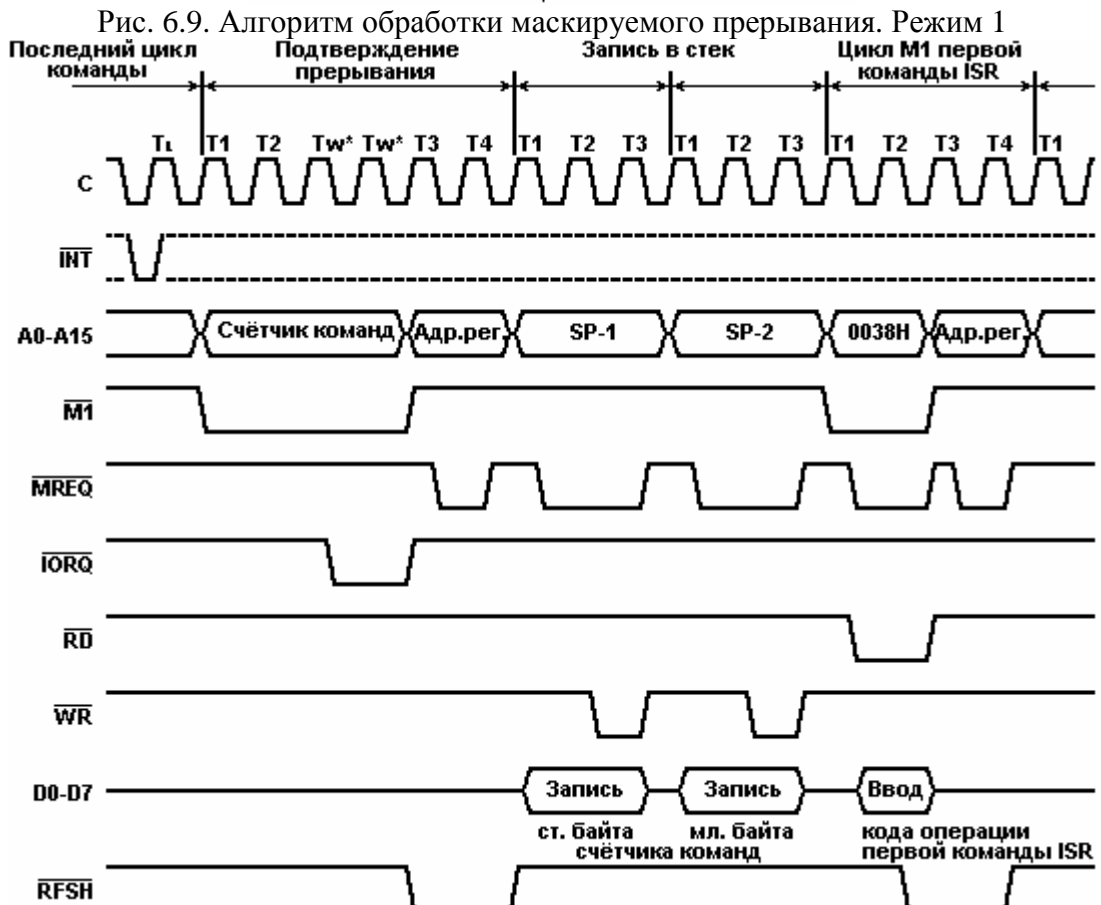
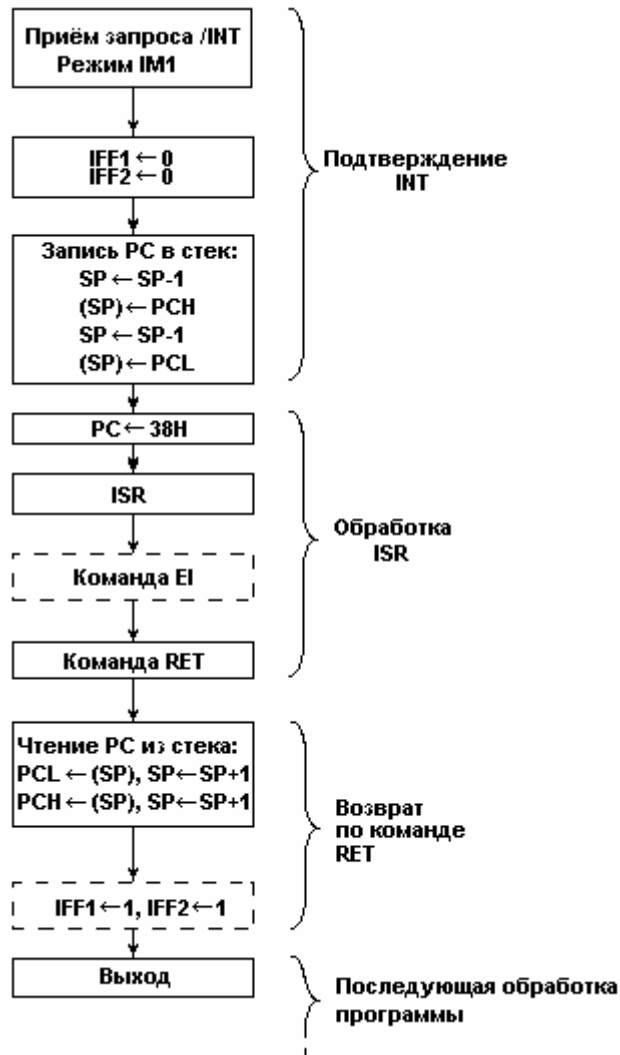


Рис. 6.10. Вызов программы обработки маскируемого прерывания ISR в режиме 1 (IM 1)

6.3.4. Маскируемое прерывание. Режим 2

Режим прерываний 2 устанавливается командой IM 2. Это самый производительный режим прерываний микропроцессора Z80. Он применяется в сложных системах с развитой сетью периферийных элементов. При этом в памяти программируется таблица стартовых адресов каждой ISR (см. рис. 6.11). Таблица может быть размещена в любой области памяти с соблюдением лишь правила записи стартовых адресов: младший байт заносится в ячейку с чётным адресом ($A0=0$), старший байт в следующую (по возрастанию) ячейку.

В случае приёма прерывания ЦП формирует 16-разрядный указатель IP (Interrupt Pointer) для выборки стартового адреса нужной программы ISR из таблицы. Для этого прерывавшим устройством в виде 8 разрядного вектора поставляется младший байт, а старший извлекается из регистра I микропроцессора. Сформированный таким образом IP указывает на ячейку памяти (в таблице), где хранится стартовый адрес ISR. Считав этот адрес из двух смежных ячеек, процессор переходит к обработке программы ISR. Возврат из неё должен происходить по команде RETI. В целом алгоритм обработки изображен на рис 6.12.

В микропроцессорном комплекте Z80 предусмотрено, что периферийные БИС могут выдавать в качестве вектора прерывания только чётные байты ($D0=0$) - для однозначности адресации ячеек таблицы. Значит, даже при неизменном состоянии регистра I центрального процессора возможно обслуживание 128-ми устройств в режиме прерываний 2, что во многом превышает потребности микропроцессорной системы среднего класса.

Для перехода к ISR в режиме 2 требуется 19 тактовых периодов (см. временные диаграммы на рис. 6.13): 7 периодов для ввода 8 разрядного вектора от прерывающего устройства, следующие 6 - для сохранения в стеке текущего состояния PC (адреса возврата), и ещё 6 - для считывания стартового адреса ISR.

Подробное описание прерываний в системе с полной конфигурацией, особенно, в отношении периферийных элементов комплекта Z80, а также организации приоритетного обслуживания см. в книге 7 "Z80/Построение систем. Программирование. Отладка".

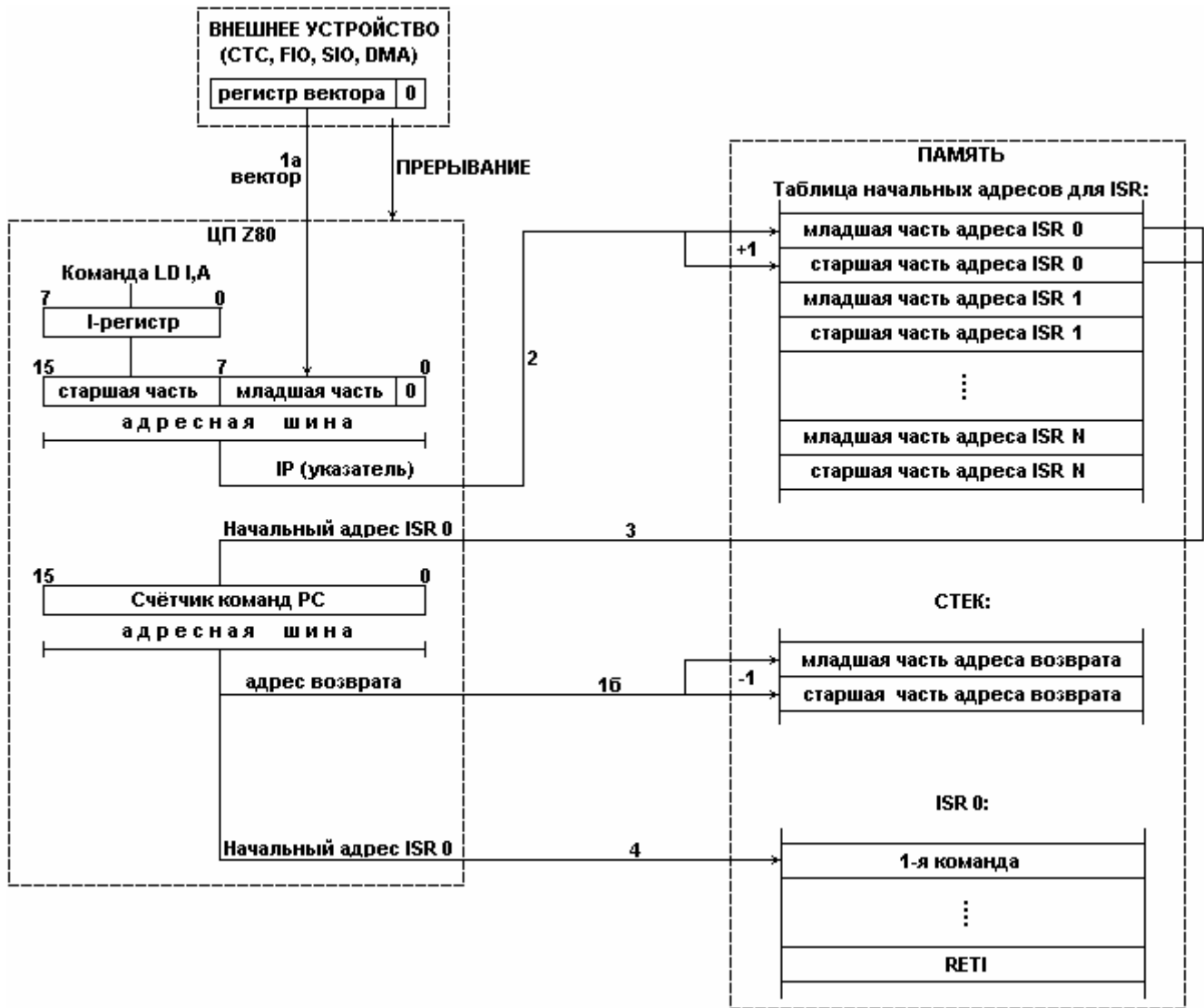


Рис. 6.11. Переход к ISR в режиме прерывания 2:

- 1а - приём вектора прерывания устройства в ЦП,
- 1б - адрес возврата сохраняется в стеке,
- 2 - сформированный 16-разрядный IP указывает на начальный адрес ISR,
- 3 - начальный адрес ISR загружается в PC,
- 4 - считывается 1-я команда программы ISR.

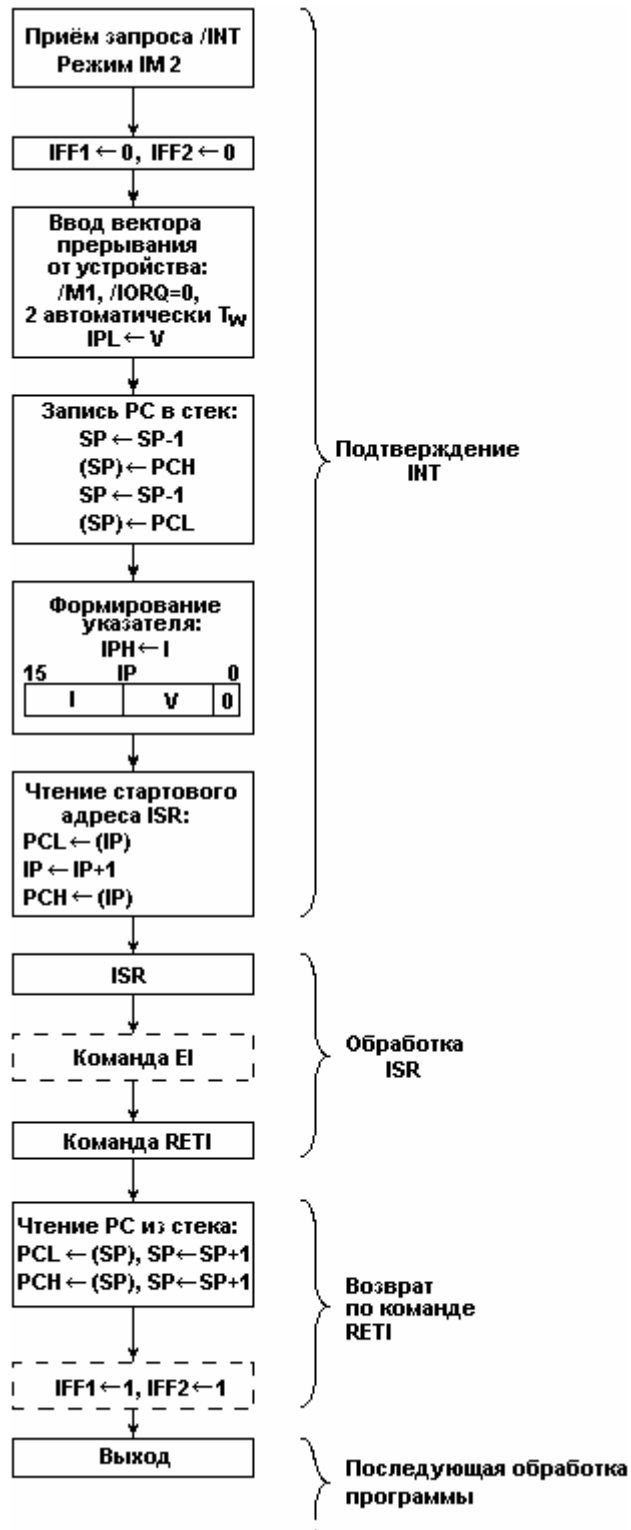


Рис. 6.12. Алгоритм обработки маскируемого прерывания. Режим 2.

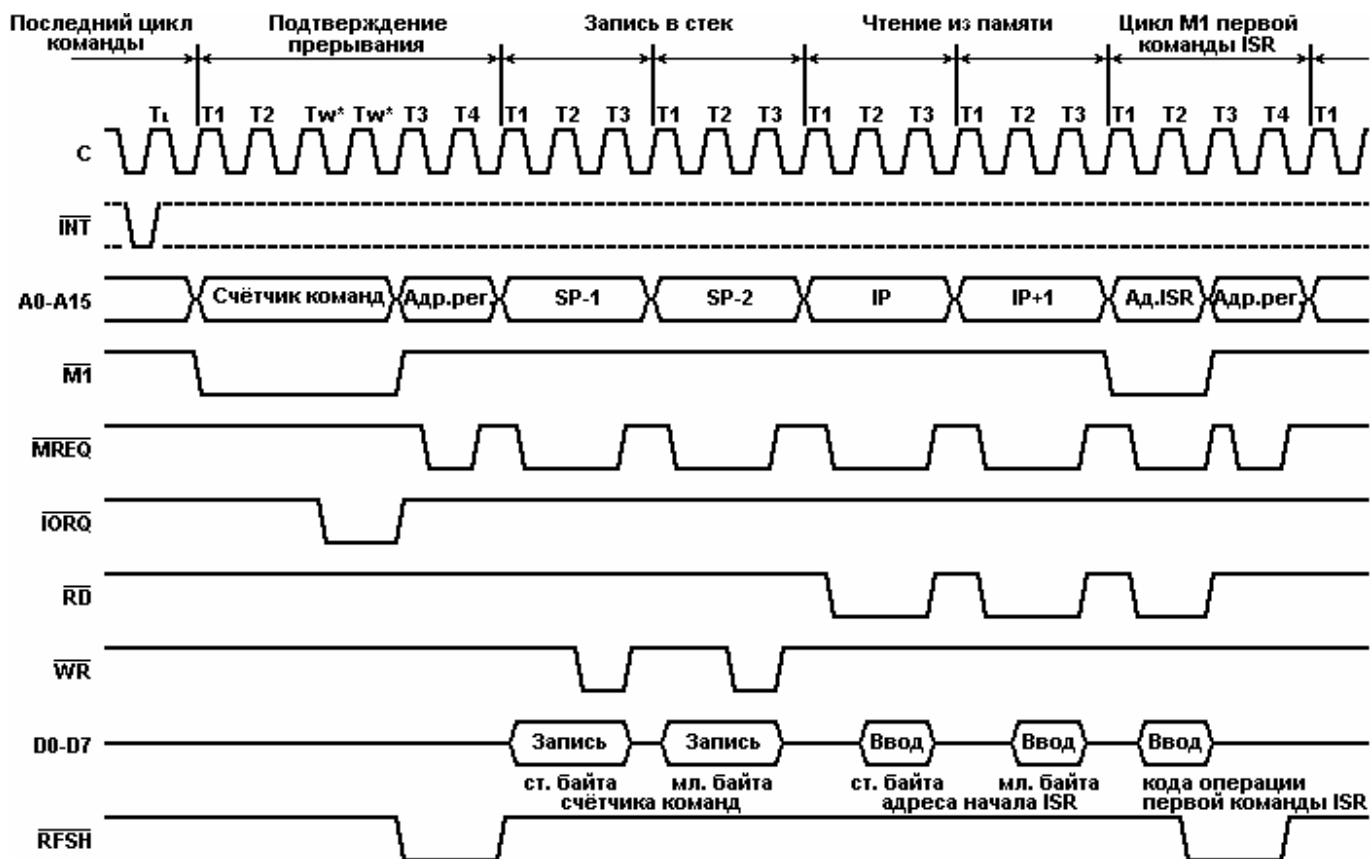


Рис. 6.13. Вызов программы обработки маскируемого прерывания ISR в режиме 2 (IM 2)

7. Технические характеристики

7.1. Схемы входных и выходных каскадов

На приведенных ниже схемах приняты следующие обозначения:

- I (INPUT) - внешний вход;
- O (OUTPUT) - внешний выход;
- i (input) - внутренний вход;
- o (output) - внутренний выход;
- I/O (INPUT/OUTPUT) - внешний вход/выход;
- HOLD - внутренний сигнал перевода в третье состояние

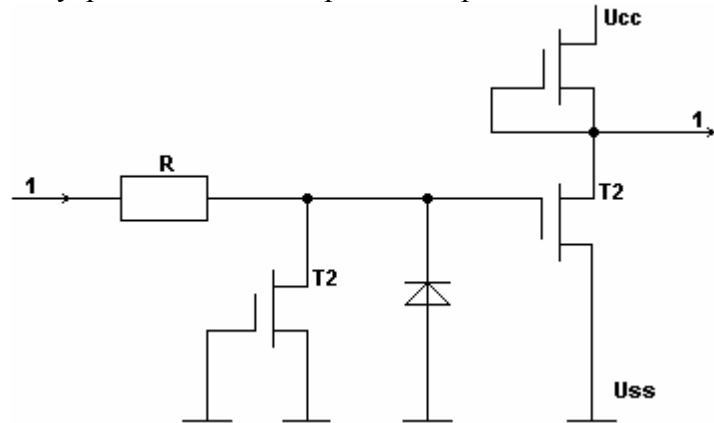


Рис. 7.1. Схема входов /WAIT, /INT, /NMI, /RESET, /BUSRQ и С.

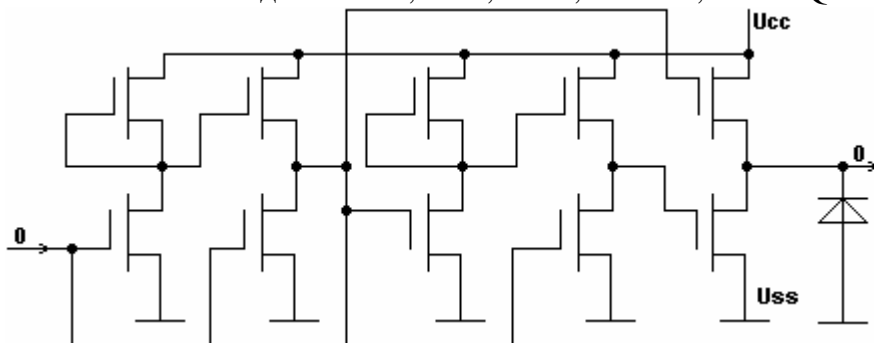


Рис. 7.2. Схема выходов /M1, /RFSH, /HALT и /BUSAK.

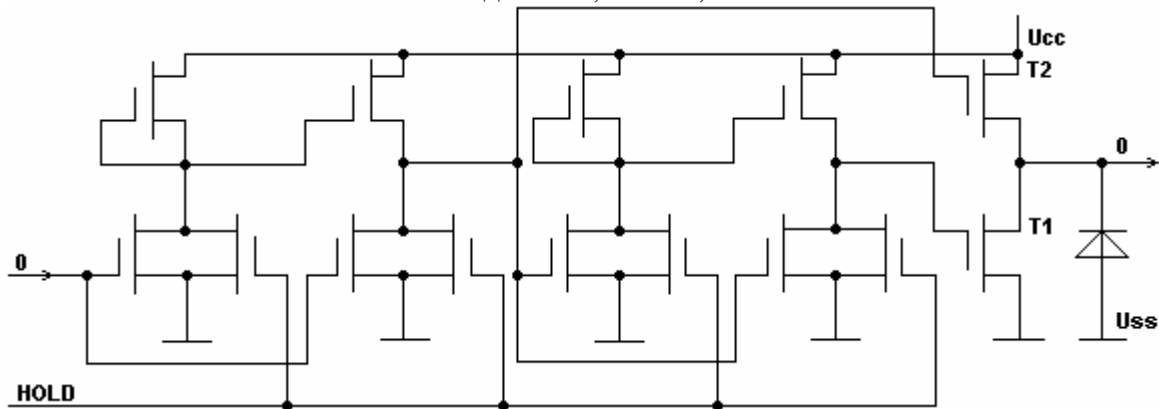


Рис. 7.3. Схема тристабильных выходов A0-A15, /MREQ, /IORQ, /RD и /WR.

Параметр	Обозначение	Единица измерения	Минимальное значение	Максимальное значение	Пояснения
Выходная ёмкость	C_o	пФ	-	10	

* в числителе - для Z80, в знаменателе - для Z80A.

7.2.2. Динамические характеристики Z80

При $U_{CC}=5V\pm 5\%$, $C_L=50$ пФ и $V_m=0\div 70^\circ C$.

Параметр	Обозначение	Минимальное значение нс	Максимальное значение нс
Период тактовых импульсов	t_C	400	*)
Длительность низкого уровня тактового сигнала	$t_{W(CL)}$	180	2000
Длительность высокого уровня тактового сигнала	$t_{W(CH)}$	180	**)
Длительность переднего/заднего фронта тактового сигнала	t_r, t_f	-	30
Установка сигнала /WAIT до H-L перехода такта	$t_{a(WT)}$	70	-
Установка сигнала /RESET до L-H перехода такта	$t_{a(RS)}$	90	-
Установка сигнала /INT до L-H перехода такта	$t_{a(IT)}$	60	-
Установка сигнала /BUSRQ до L-H перехода такта	$t_{a(BQ)}$	80	-
Установка данных до L-H перехода такта в цикле M1	$t_{a(D)}$	50	-
Установка данных до H-L перехода такта в циклах M2-M5	$t_{a\bar{c}(D)}$	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 $U_{CC}=5V\pm 5\%$, $C_L=50$ пФ и $V_m=0\div 70^\circ C$.

Задержка	Обозначение	Максимальное значение, нс
от H-L перехода такта до /M1=L	$t_{DL(M1)}$	130
от H-L перехода такта до /M1=H	$t_{DH(M1)}$	130
от H-L перехода такта до /MREQ=H	$t_{DHC(MR)}$	100
от L-H перехода такта до /MREQ=H	$t_{DH\bar{C}(MR)}$	100
от H-L перехода такта до /MREQ=L	$t_{DH\bar{C}(MR)}$	100
от L-H перехода такта до /IORQ=L	$t_{DLC(IR)}$	90
от H-L перехода такта до /IORQ=L	$t_{DL\bar{C}(IR)}$	110
от L-H перехода такта до /IORQ=H	$t_{DHC(IR)}$	100
от H-L перехода такта до /IORQ=H	$t_{DH\bar{C}(IR)}$	110
от L-H перехода такта до /RD=L	$t_{DLC(RD)}$	100
от H-L перехода такта до /RD=L	$t_{DL\bar{C}(RD)}$	130
от L-H перехода такта до /RD=H	$t_{DHC(RD)}$	100
от H-L перехода такта до /RD=H	$t_{DH\bar{C}(RD)}$	110
от L-H перехода такта до /WR=L	$t_{DLC(WR)}$	80
от H-L перехода такта до /WR=L	$t_{DL\bar{C}(WR)}$	90
от L-H перехода такта до /WR=H	$t_{DH\bar{C}(WR)}$	100

Задержка	Обозначение	Максимальное значение, нс
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	$t_{DL(BA)}$	120
от H-L перехода такта до /BUSAK=H	$t_{DH(BA)}$	110
вывода адреса	$t_{D(AD)}$	145
адреса до перехода к третьему состоянию	$t_{F(AD)}$	110
вывода данных	$t_{D(D)}$	230
данных до перехода к третьему состоянию в цикле записи	$t_{F(D)}$	90
сигналов /MREQ, /IORQ, /RD, /WR до перехода к третьему состоянию	$t_{F(C)}$	100

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

Дополнительные данные о времени.

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

$$t_{acm} = t_{w(CH)} + t_F - 75 \text{ нс}$$

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

$$t_{ac1} = t_C - 80 \text{ нс}$$

Задержка адреса после снятия /RD или WR:

$$t_{ca} = t_{w(CL)} + t_r - 40 \text{ нс}$$

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

$$t_{caf} = t_{w(CL)} + t_r - 60 \text{ нс}$$

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

$$t_{dcm} = t_C - 210 \text{ нс}$$

Вывод данных до активизации /WR в циклах ввода-вывода:

$$t_{ac1} = t_{w(CL)} + t_r - 210 \text{ нс}$$

Задержка данных после снятия /WR:

$$t_{cdf} = t_{w(CL)} + t_r - 60 \text{ нс}$$

Ширина импульса низкого уровня /MREQ:

$$t_{w(MRL)} = t_C - 40 \text{ нс}$$

Ширина импульса высокого уровня /MREQ:

$$t_{w(MRH)} = t_{w(CH)} + t_F - 30 \text{ нс}$$

Ширина импульса низкого уровня /WR:

$$t_{w(WRL)} = t_C - 40 \text{ нс}$$

Вывод /M1 до активизации /IORQ в цикле подтверждения прерывания:

$$t_{M1} = 2t_C + t_{w(CH)} + t_F - 80 \text{ нс}$$

7.2.3. Динамические характеристики Z80A

При $U_{CC}=5V \pm 5\%$, $C_L=50\text{пФ}$ и $V_m=0 \div 70^\circ\text{C}$.

Параметр	Обозначение	Минимальное значение нс	Максимальное значение нс
Период тактовых импульсов	t_C	250	*)
Длительность низкого уровня тактового сигнала	$t_{w(CL)}$	110	2000
Длительность высокого уровня тактового сигнала	$t_{w(CH)}$	110	**)
Длительность переднего/заднего фронта тактового сигнала	t_r, t_f	-	30

Параметр	Обозначение	Минимальное значение нс	Максимальное значение нс
Установка сигнала /WAIT до H-L перехода такта	$t_{a(WT)}$	70	-
Установка сигнала /RESET до L-H перехода такта	$t_{a(RS)}$	60	-
Установка сигнала /INT до L-H перехода такта	$t_{a(IT)}$	80	-
Установка сигнала /BUSRQ до L-H перехода такта	$t_{a(BQ)}$	50	-
Установка данных до L-H перехода такта в цикле M1	$t_{ac(D)}$	35	-
Установка данных до H-L перехода такта в циклах M2-M5	$t_{ac\bar{c}(D)}$	50	-
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 Z80A может находиться в устойчивом состоянии сколь угодно долго.

Времена задержек

При $U_{CC} = 5V \pm 5\%$, $C_L = 50$ пФ и $V_m = 0 \div 70^\circ C$.

Задержка	Обозначение	Максимальное значение, нс
от H-L перехода такта до /M1=L	$t_{DL(M1)}$	100
от H-L перехода такта до /M1=H	$t_{DH(M1)}$	100
от H-L перехода такта до /MREQ=H	$t_{DHC(MR)}$	85
от L-H перехода такта до /MREQ=H	$t_{DH\bar{C}(MR)}$	85
от H-L перехода такта до /MREQ=L	$t_{DH\bar{C}(MR)}$	85
от L-H перехода такта до /IORQ=L	$t_{DLC(IR)}$	75
от H-L перехода такта до /IORQ=L	$t_{DL\bar{C}(IR)}$	85
от L-H перехода такта до /IORQ=H	$t_{DHC(IR)}$	85
от H-L перехода такта до /IORQ=H	$t_{DH\bar{C}(IR)}$	85
от L-H перехода такта до /RD=L	$t_{DLC(RD)}$	85
от H-L перехода такта до /RD=L	$t_{DL\bar{C}(RD)}$	95
от L-H перехода такта до /RD=H	$t_{DHC(RD)}$	85
от H-L перехода такта до /RD=H	$t_{DH\bar{C}(RD)}$	85
от L-H перехода такта до /WR=L	$t_{DLC(WR)}$	65
от H-L перехода такта до /WR=L	$t_{DL\bar{C}(WR)}$	80
от L-H перехода такта до /WR=H	$t_{DH\bar{C}(WR)}$	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	$t_{DL(BA)}$	100
от H-L перехода такта до /BUSAK=H	$t_{DH(BA)}$	100
вывода адреса	$t_{D(AD)}$	110
адреса до перехода к третьему состоянию	$t_{F(AD)}$	90
вывода данных	$t_{D(D)}$	150
данных до перехода к третьему состоянию в цикле записи	$t_{F(D)}$	90
сигналов /MREQ, /IORQ, /RD, /WR до перехода к третьему состоянию	$t_{F(C)}$	80

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

Дополнительные данные о времени

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

$$t_{acm} = t_{w(CH)} + t_F - 65 \text{ нс}$$

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

$$t_{ac1} = t_C - 70 \text{ нс}$$

Задержка адреса после снятия /RD или WR:

$$t_{ca} = t_{w(CL)} + t_r - 50 \text{ нс}$$

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

$$t_{caf} = t_{w(CL)} + t_r - 45 \text{ нс}$$

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

$$t_{dcm} = t_C - 170 \text{ нс}$$

Вывод данных до активизации /WR в циклах ввода-вывода:

$$t_{ac1} = t_{w(CL)} + t_r - 170 \text{ нс}$$

Задержка данных после снятия /WR:

$$t_{cdf} = t_{w(CL)} + t_r - 70 \text{ нс}$$

Ширина импульса низкого уровня /MREQ:

$$t_{w(MRL)} = t_C - 30 \text{ нс}$$

Ширина импульса высокого уровня /MREQ:

$$t_{w(MRH)} = t_{w(CH)} + t_F - 20 \text{ нс}$$

Ширина импульса низкого уровня /WR:

$$t_{w(WRL)} = t_C - 30 \text{ нс}$$

Вывод /M1 до активизации /IORQ в цикле подтверждения прерывания:

$$t_{M1} = 2t_C + t_{w(CH)} + t_F - 65 \text{ нс}$$

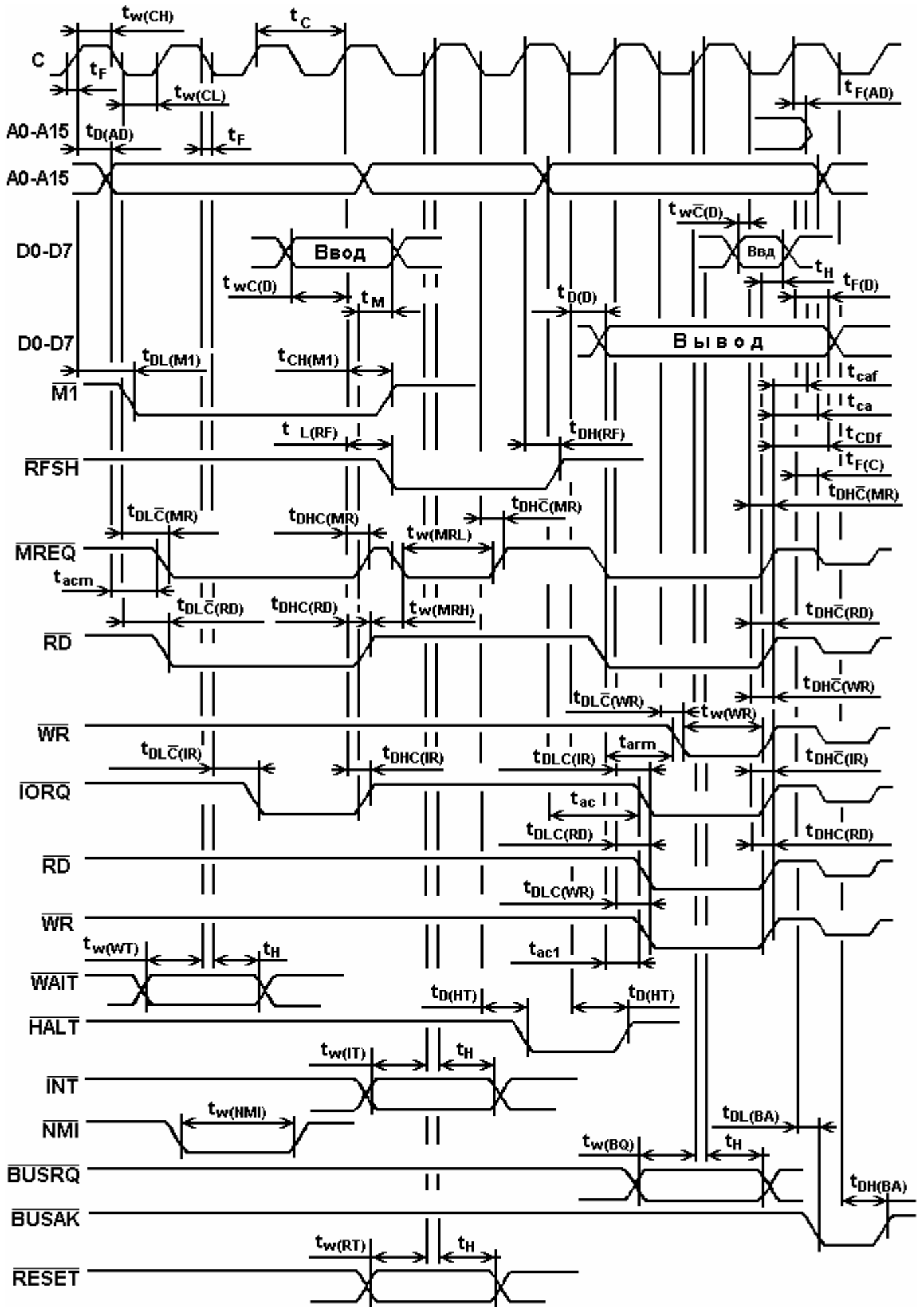


Рис. 7.5. Временные характеристики МП Z80/Z80A (к 7.2.2 и 7.2.3)

7.2.4. Предельные значения

Предельные значения даны при $V_a=0\div 70^\circ\text{C}$.

Параметр	Обозначение	Единица измерения	Минимальное значение	Максимальное значение
Рабочее напряжение	U_{CC}	В	-0,3	7
Входное напряжение	U_I	В	-0,3	7
Диапазон рабочей температуры	V_a	$^\circ\text{C}$	0	70
Диапазон температуры хранения	V_{atu}	$^\circ\text{C}$	-65	150
Мощность рассеяния	P	Вт	-	1,5

Существует несколько исполнений (указывается на корпусе)

С - керамический корпус;

Р - пластмассовый корпус;

S - стандартные условия эксплуатации ($5\text{В}\pm 5\%$, $0\div 70^\circ\text{C}$);

E - расширенные условия эксплуатации ($5\text{В}\pm 5\%$, $-40\div 85^\circ\text{C}$);

M - военное исполнение ($5\text{В}\pm 10\%$, $-55\div 125^\circ\text{C}$).

7.3. Надежность

Интенсивность отказов:

$$L_{PO,B} < 5 \cdot 10^{-5} \text{ ч}^{-1}$$

При средней электрической нагрузке (рабочее напряжение U_{CC} от 4,75 В до 5,25 В и температуре окружающей среды $V_a < 50^\circ\text{C}$, нормальной климатической и механической нагрузке наработка на отказ 2 000 000 часов.

ПРИЛОЖЕНИЯ

Приложение А. Кодовые таблицы команд

Основной набор команд
старшая тетрада

hex	0	1	2	3	4	5	6	7
0	NOP	DJNZ e	JR NZ,e	JR NC,e	LD B,B	LD D,B	LD H,B	LD (HL),B
1	LD BC,nn	LD DE,nn	LD HL,nn	LD SP,nn	LD B,C	LD D,C	LD H,C	LD (HL),C
2	LD (BC),A	LD (DE),A	LD (nn),HL	LD (nn),A	LD B,D	LD D,D	LD H,D	LD (HL),D
3	INC BC	INC DE	INC HL	INC SP	LD B,E	LD D,E	LD H,E	LD (HL),E
4	INC B	INC D	INC H	INC (HL)	LD B,H	LD D,H	LD H,H	LD (HL),H
5	DEC B	DEC D	DEC H	DEC (HL)	LD B,L	LD D,L	LD H,L	LD (HL),L
6	LD B,n	LD D,n	LD H,n	LD (HL),n	LD B,(HL)	LD D,(HL)	LD H,(HL)	HALT
7	RLCA	RLA	DAA	SCF	LD B,A	LD D,A	LD H,A	LD (HL),A
8	EX AF,AF'	JR e	JR Z,e	JR C,e	LD C,B	LD E,B	LD L,B	LD A,B
9	ADD HL,BC	ADD HL,DE	ADD HL,HL	ADD HL,SP	LD C,C	LD E,C	LD L,C	LD A,C
A	LD A,(BC)	LD A,(DE)	LD HL,(nn)	LD A,(nn)	LD C,D	LD E,D	LD L,D	LD A,D
B	DEC BC	DEC DE	DEC HL	DEC SP	LD C,E	LD E,E	LD L,E	LD A,E
C	INC C	INC E	INC L	INC A	LD C,H	LD E,H	LD L,H	LD A,H
D	DEC C	DEC E	DEC L	DEC A	LD C,L	LD E,L	LD L,L	LD A,L
E	LD C,n	LD E,n	LD L,n	LD A,n	LD C,(HL)	LD E,(HL)	LD L,(HL)	LD A,(HL)
F	RRCA	RRA	CPL	CCF	LD C,A	LD E,A	LD L,A	LD A,A

м
л
а
д
ш
а
я
т
е
т
р
а
д
а

старшая тетрада

8	9	A	B	C	D	E	F	hex
ADD A,B	SUB B	AND B	OR B	RET NZ	RET NC	RET PO	RET P	0
ADD A,C	SUB C	AND C	OR C	POP BC	POP DE	POP HL	POP AF	1
ADD A,D	SUB D	AND D	OR D	JP NZ,nn	JP NC,nn	JP PO,nn	JP P,nn	2
ADD A,E	SUB E	AND E	OR E	JP nn	OUT (n),A	EX (SP),HL	DI	3
ADD A,H	SUB H	AND H	OR H	CALL NZ,nn	CALL NC,nn	CALL PO,nn	CALL P,nn	4
ADD A,L	SUB L	AND L	OR L	PUSH NZ,nn	PUSH NC,nn	PUSH PO,nn	PUSH P,nn	5
ADD A,(HL)	SUB (HL)	AND (HL)	OR (HL)	ADD A,n	SUB n	AND n	OR n	6
ADD A,A	SUB A	AND A	OR A	RST 00H	RST 10H	RST 20H	RST 30H	7
ADC A,B	SBC A,B	XOR B	CP B	RET Z	RET C	RET PE	RET M	8
ADC A,C	SBC A,C	XOR C	CP C	RET	EXX	JP (HL)	LD SP,HL	9
ADC A,D	SBC A,D	XOR D	CP D	JP Z,nn	JP C,nn	JP PE,nn	JP M,nn	A
ADC A,E	SBC A,E	XOR E	CP E	Таблица CB	IN A,(n)	EX DE,HL	EI	B
ADC A,H	SBC A,H	XOR H	CP H	CALL Z,nn	CALL C,nn	CALL PE,nn	CALL M,nn	C
ADC A,L	SBC A,L	XOR L	CP L	CALL nn	Таблица DD	Таблица ED	Таблица FD	D
ADC A,(HL)	SBC A,(HL)	XOR (HL)	CP (HL)	ADC A,n	SBC A,n	XOR n	CP n	E
ADC A,A	SBC A	XOR A	CP A	RST 08H	RST 18H	RST 28H	RST 38H	F

м
л
д
ш
а
я
т
е
т
р
а
д
а

Команда CB

старшая тетрада

hex	0	1	2	3	4	5	6	7
0	RLC B	RL B	SLA B		BIT 0,B	BIT 2,B	BIT 4,B	BIT 6,B
1	RLC C	RL C	SLA C		BIT 0,C	BIT 2,C	BIT 4,C	BIT 6,C
2	RLC D	RL D	SLA D		BIT 0,D	BIT 2,D	BIT 4,D	BIT 6,D
3	RLC E	RL E	SLA E		BIT 0,E	BIT 2,E	BIT 4,E	BIT 6,E
4	RLC H	RL H	SLA H		BIT 0,H	BIT 2,H	BIT 4,H	BIT 6,H
5	RLC L	RL L	SLA L		BIT 0,L	BIT 2,L	BIT 4,L	BIT 6,L
6	RLC (HL)	RL (HL)	SLA (HL)		BIT 0,(HL)	BIT 2,(HL)	BIT 4,(HL)	BIT 6,(HL)
7	RLC A	RL A	SLA A		BIT 0,A	BIT 2,A	BIT 4,A	BIT 6,A
8	RRC B	RR B	SRA B	SRL B	BIT 1,B	BIT 3,B	BIT 5,B	BIT 7,B
9	RRC C	RR C	SRA C	SRL C	BIT 1,C	BIT 3,C	BIT 5,C	BIT 7,C
A	RRC D	RR D	SRA D	SRL D	BIT 1,0	BIT 3,D	BIT 5,D	BIT 7,D
B	RRC E	RR E	SRA E	SRL E	BIT 1,E	BIT 3,E	BIT 5,E	BIT 7,E
C	RRC H	RR H	SRA H	SRL H	BIT 1,H	BIT 3,H	BIT 5,H	BIT 7,H
D	RRC L	RR L	SRA L	SRL L	BIT 1,L	BIT 3,L	BIT 3,L	BIT 7,L
E	RRC (HL)	RR (HL)	SRA (HL)	SRL (HL)	BIT 1,(HL)	BIT 3,(HL)	BIT 5,(HL)	BIT 7,(HL)
F	RRC A	RR A	SRA A	SRL A	BIT 1,A	BIT 3,A	BIT 5,A	BIT 7,A

м
л
а
д
ш
а
я
т
е
т
р
а
д
а

старшая тетрада

8	9	A	B	C	D	E	F	hex	м л д ш а я т е т р а д а
RES 0,B	RES 2,B	RES 4,B	RES 6,B	SET 0,B	SET 2,B	SET 4,B	SET 6,B	0	
RES 0,C	RES 2,C	RES 4,C	RES 6,C	SET 0,C	SET 2,C	SET 4,C	SET 6,C	1	
RES 0,D	RES 2,D	RES 4,D	RES 6,D	SET 0,D	SET 2,D	SET 4,D	SET 6,D	2	
RES 0,E	RES 2,E	RES 4,E	RES 6,E	SET 0,E	SET 2,E	SET 4,E	SET 6,E	3	
RES 0,H	RES 2,H	RES 4,H	RES 6,H	SET 0,H	SET 2,H	SET 4,H	SET 6,H	4	
RES 0,L	RES 2,L	RES 4,L	RES 6,L	SET 0,L	SET 2,L	SET 4,L	SET 6,L	5	
RES 0,(HL)	RES 2,(HL)	RES 4,(HL)	RES 6,(HL)	SET 0,(HL)	SET 2,(HL)	SET 4,(HL)	SET 6,(HL)	6	
RES 0,A	RES 2,A	RES 4,A	RES 6,A	SET 0,A	SET 2,A	SET 4,A	SET 6,A	7	
RES 1,B	RES 3,B	RES 5,B	RES 7,B	SET 1,B	SET 3,B	SET 5,B	SET 7,B	8	
RES 1,C	RES 3,C	RES 5,C	RES 7,C	SET 1,C	SET 3,C	SET 5,C	SET 7,C	9	
RES 1,D	RES 3,D	RES 5,D	RES 7,D	SET 1,D	SET 3,D	SET 5,D	SET 7,D	A	
RES 1,E	RES 3,E	RES 5,E	RES 7,E	SET 1,E	SET 3,E	SET 5,E	SET 7,E	B	
RES 1,H	RES 3,H	RES 5,H	RES 7,H	SET 1,H	SET 3,H	SET 5,H	SET 7,H	C	
RES 1,L	RES 3,L	RES 5,L	RES 7,L	SET 1,L	SET 3,L	SET 5,L	SET 7,L	D	
RES 1,(HL)	RES 3,(HL)	RES 5,(HL)	RES 7,(HL)	SET 1,(HL)	SET 3,(HL)	SET 5,(HL)	SET 7,(HL)	E	
RES 1,A	RES 3,A	RES 5,A	RES 7,A	SET 1,A	SET 3,A	SET 5,A	SET 7,A	F	

Команда ED

старшая тетрада

м
л
а
д
ш
а
я
т
е
т
р
а
д
а

hex	4	5	6	7	A	B
0	IN B,(C)	IN D,(C)	IN H,(C)	INF	LDI	LDIR
1	OUT (C),B	OUT (C),D	OUT (C),H		CPI	CPIR
2	SBC HL,BC	SBC HL,DE	SBC HL,HL	SBC HL,SP	INI	INIR
3	LD (nn),BC	LD (nn),DE		LD (nn),SP	OUTI	OTIR
4	NEG					
5	RETN					
6	IM 0	IM 1				
7	LD I,A	LD A,I	RRD			
8	IN C,(C)	IN E,(C)	IN L,(C)	IN A,(C)	LDD	LDDR
9	OUT (C),C	OUT (C),E	OUT (C),L	OUT (C),A	CPD	CPDR
A	ADC HL,BC	ADC HL,DE	ADC HL,HL	ADC HL,SP	IND	INDR
B	LD BC,(nn)	LD DE,(nn)		LD SP,(nn)	OUTD	OTDR
C						
D	RETI					
E		IM 2				
F	LD R,A	LD A,R	RLD			

Команды DD/FD

Первый байт:

DD: ii=IX

FD: ii=IY

Второй байт	Команда
09	ADD ii,BC
19	ADD ii,DE
21	LD ii,nn
22	LD (nn),ii
23	INC ii
29	ADD ii,ii
2A	LD ii,(nn)
2B	DEC ii
34	INC (ii+d)
35	DEC (ii+d)
36	LD (ii+d),n
39	ADD ii,SP
46	LD B,(ii+d)
4E	LD C,(ii+d)
56	LD D,(ii+d)
5E	LD E,(ii+d)
66	LD H,(ii+d)
6E	LD L,(ii+d)

Второй байт	Команда
70	LD (ii+d),B
71	LD (ii+d),C
72	LD (ii+d),D
73	LD (ii+d),E
74	LD (ii+d),H
75	LD (ii+d),L
77	LD (ii+d),A
7E	LD A,(ii+d)
B6	ADD A,(ii+d)
BE	ADC A,(ii+d)
96	SUB (ii+d)
9E	SBC A,(ii+d)
A6	AND (ii+d)
AE	XOR (ii+d)
B6	OR (ii+d)
BE	CP (ii+d)
CB	Таблица DD/CB Таблица FD/CB
E1	POP ii
E3	EX (SP),ii
E5	PUSH ii
E9	JP (ii)
F9	LD SP,ii

Команды DD CB / FD CB

Первый байт	Второй байт	Третий байт	Четвёртый байт	Команда
DD: ii=IX FD: ii=IY	CB	Смещение d	06	RLC (ii+d)
			0E	RRC (ii+d)
			16	RL (ii+d)
			1E	RR (ii+d)
			26	SLA (ii+d)
			2E	SRA (ii+d)
			3E	SRL (ii+d)
			46	BIT 0,(ii+d)
			4E	BIT 1,(ii+d)
			56	BIT 2,(ii+d)
			5E	BIT 3,(ii+d)
			66	BIT 4,(ii+d)
			6E	BIT 5,(ii+d)
			76	BIT 6,(ii+d)
			7E	BIT 7,(ii+d)
			86	RES 0,(ii+d)
			8E	RES 1,(ii+d)
			96	RES 2,(ii+d)
			9E	RES 3,(ii+d)
			A6	RES 4,(ii+d)
			AE	RES 5,(ii+d)
			B6	RES 6,(ii+d)
			BE	RES 7,(ii+d)
			C6	SET 0,(ii+d)
			CE	SET 1,(ii+d)
			D6	SET 2,(ii+d)
			DE	SET 3,(ii+d)
			E6	SET 4,(ii+d)
			EE	SET 5,(ii+d)
			F6	SET 6,(ii+d)
FE	SET 7,(ii+d)			

Приложение Б. Соответствие мнемоник ассемблера Z80 и I8080

hex	Z80	I8080
00	NOP	NOP
01	LD BC,nn	LXI B,nn
02	LD (BC),A	STAX B
03	INC BC	INX B
04	INC B	INR B
05	DEC B	DCR B
06	LD B,n	MVI B,n
07	RLCA	RLC
08	EX AF,AF'	-
09	ADD HL,BC	DAD B
0A	LD A,(BC)	LDAX B
0B	DEC BC	DCX B
0C	INC C	INR C
0D	DEC C	OCR C
0E	LD C,n	MVI C,n
0F	RRCA	RRC
10	DJNZ e	-
11	LD DE,nn	LXI D,nn
12	LD (DE),A	STAX D
13	INC DE	INX D
14	INC D	INR D
15	DEC D	DCR R
16	LD D,n	MVI D,n
17	RLA	RAL
18	JR e	-
19	ADD HL,DE	DAD D
1A	LD A,(DE)	LDAX D
1B	DEC DE	DCX D
1C	INC E	INR E
1D	DEC E	OCR E
1E	LD E,n	MVI E,n
1F	RRA	RAR
20	JR NZ,e	-
21	LD HL,nn	LXI H,nn
22	LD (nn),HL	SHLD n
23	INC HL	INX H
24	INC H	INR H
25	DEC H	DCR H
26	LD H,n	MVI H,n
27	DAA	DAA
28	JR Z,e	-
29	ADD HL,HL	DAD H
2A	LD HL,(nn)	LHLD n
2B	DEC HL	DCX H
2C	INC L	INR L
2D	DEC L	DCR L
2E	LD L,n	MVI L,n

hex	Z80	I8080
2F	CPL	CMA
30	JR NC,e	-
31	LD SP,nn	LXI SP,nn
32	LD (nn),A	STA nn
33	INC SP	INX SP
34	INC (HL)	INR H
35	DEC (HL)	DCR H
36	LD (HL),n	MVI H,n
37	SCF	STC
38	JR C,e	-
39	ADD HL,SP	DAD SP
3A	LD A,(nn)	LDA nn
3B	DEC SP	DCX SP
3C	INC A	INR A
3D	DEC A	DCR A
3E	LD A,N	MVI A,n
3F	CCF	CAC
40	LD B,B	MOV B,B
41	LD B,C	MOV B,C
42	LD B,D	MOV B,D
43	LD B,E	MOV B,E
44	LD B,H	MOV B,H
45	LD B,L	MOV B,L
46	LD B,(HL)	MOV B,N
47	LD B,A	MOV B,A
48	LD C,B	MOV C,B
49	LD C,C	MOV C,C
4A	LD C,D	MOV C,D
4B	LD C,E	MOV C,E
4C	LD C,H	MOV C,H
4D	LD C,L	MOV C,L
4E	LD C,(HL)	MOV C,N
4F	LD C,A	MOV C,A
50	LD D,B	MOV D,B
51	LD D,C	MOV D,C
52	LD D,D	MOV D,D
53	LD D,E	MOV D,E
54	LD D,H	MOV D,H
55	LD D,L	MOV D,L
56	LD D,(HL)	MOV D,N
57	LD D,A	MOV D,A
58	LD E,B	MOV E,B
59	LD E,C	MOV E,C
5A	LD E,D	MOV E,D
5B	LD E,E	MOV E,E
5C	LD E,H	MOV E,H
5D	LD E,L	MOV E,L

hex	Z80	I8080
5E	LD E,(HL)	MOV E,N
5F	LD E,A	MOV E,A
60	LD H,B	MOV H,B
61	LD H,C	MOV H,C
62	LD H,D	MOV H,D
63	LD H,E	MOV H,E
64	LD H,H	MOV H,H
65	LD H,L	MOV H,L
66	LD H,(HL)	MOV H,N
67	LD H,A	MOV H,A
68	LD L,B	MOV L,B
69	LD L,C	MOV L,C
6A	LD L,D	MOV L,D
6B	LD L,E	MOV L,E
6C	LD L,H	MOV L,H
6D	LD L,L	MOV L,L
6E	LD L,(HL)	MOV L,N
6F	LD L,A	MOV L,A
70	LD (HL),B	MOV N,B
71	LD (HL),C	MOV N,C
72	LD (HL),D	MOV N,D
73	LD (HL),E	MOV N,E
74	LD (HL),H	MOV N,H
75	LD (HL),L	MOV N,L
76	HALT	HLE
77	LD (HL),A	MOV N,A
78	LD A,B	MOV A,B
79	LD A,C	MOV A,C
7A	LD A,D	MOV A,D
7B	LD A,E	MOV A,E
7C	LD A,H	MOV A,H
7D	LD A,L	MOV A,L
7E	LD A,(HL)	MOV A,N
7F	LD A,A	MOV A,A
80	ADD A,B	ADD B
81	ADD A,C	ADD C
82	ADD A,D	ADD D
83	ADD A,E	ADD E
84	ADD A,H	ADD H
85	ADD A,L	ADD L
86	ADD A,(HL)	ADD N
87	ADD A,A	ADD A
88	ADC A,B	ADC B
89	ADC A,C	ADC C
8A	ADC A,D	ADC D
8B	ADC A,E	ADC E
8C	ADC A,H	ADC H
8D	ADC A,L	ADC L
8E	ADC A,(HL)	ADC N
8F	ADC A,A	ADC A
90	SUB B	SUB B

hex	Z80	I8080
91	SUB C	SUB C
92	SUB D	SUB D
93	SUB E	SUB E
94	SUB H	SUB H
95	SUB L	SUB L
96	SUB (HL)	SUB N
97	SUB A	SUB A
98	SBC A,B	SBB B
99	SBC A,C	SBB C
9A	SBC A,D	SBB D
9B	SBC A,E	SBB E
9C	SBC A,H	SBB H
9D	SBC A,L	SBB L
9E	SBC A,(HL)	SBB N
9F	SBC A,A	SBB A
A0	AND B	AMA B
A1	AND C	AMA C
A2	AND D	AMA D
A3	AND E	AMA E
A4	AND H	AMA H
A5	AND L	AMA L
A6	AND (HL)	AMA N
A7	AND A	AMA A
A8	XOR B	XRA B
A9	XOR C	XRA C
AA	XOR D	XRA D
AB	XOR E	XRA E
AC	XOR H	XRA H
AD	XOR L	XRA L
AE	XOR (HL)	XRA N
AF	XOR A	XRA A
B0	OR B	ORA B
B1	OR C	ORA C
B2	OR D	ORA D
B3	OR E	ORA E
B4	OR H	ORA H
B5	OR L	ORA L
B6	OR (HL)	ORA N
B7	OR A	ORA A
B8	CP B	CMP B
B9	CP C	CMP C
BA	CP D	CMP D
BB	CP E	CMP E
BC	CP H	CMP H
BD	CP L	CMP L
BE	CP (HL)	CMP N
BF	CP A	CMP A
C0	RET NZ	RNZ
C1	POP BC	POP B
C2	JP NZ,nn	JNZ nn
C3	JP nn	JMP nn

hex	Z80	I8080
C4	CALL NZ,nn	CNZ nn
C5	PUSH BC	PUSH B
C6	ADD A,n	ADI n
C7	RST 00H	RST 0
C8	RET Z	RZ
C9	RET	RET
CA	JP Z,nn	JZ nn
CB	Таблица CB	-
CC	CALL Z,nn	CZ nn
CD	CALL nn	CALL nn
CE	ADC A,n	ACI n
CF	RST 08H	RST 1
D0	RET NC	RNC
D1	POP DE	POP D
D2	JP NC,nn	JNC nn
D3	OUT (N),A	OUT n
D4	CALL NC,nn	CNC nn
D5	PUSH DE	PUSH D
D6	SUB n	SUI n
D7	RST 10H	RST 2
D8	RET C	RC
D9	EXX	-
DA	JP C,nn	JC nn
DB	IN A,(n)	IN n
DC	CALL C,nn	CC nn
DD	Таблица DD	-
DE	SBC A,n	SBI n
DF	RST 18H	RST 3
E0	RET PO	RPO
E1	POP HL	POP H
E2	JP PO,nn	JPO nn
E3	EX (SP),HL	XTHL
E4	CALL PO,nn	CPO nn
E5	PUSH HL	PUSH H
E6	AND n	AMI n
E7	RST 20H	RST 4
E8	RET PE	RPE
E9	JP (HL)	PCHL
EA	JP PE,nn	JPE nn
EB	EX DE,HL	XCHG
EC	CALL PE,nn	CPE nn
ED	Таблица ED	-
EE	XOR n	XRI n
EF	RST 28H	RST 5
F0	RET P	RP
F1	POP AF	POP PSW
F2	JP P,nn	JP nn
F3	DI	DI
F4	CALL P,nn	CP nn
F5	PUSH AF	PUSH PSW
F6	OR n	ORI n

hex	Z80	I8080
F7	RST 30H	RST 6
F8	RET M	RM
F9	LD SP,HL	SPHL
FA	JP M,nn	JM nn
FB	EI	EI
FC	CALL M,nn	CM nn
FD	Таблица FD	-
FE	CP n	CPI n
FF	RST 38H	RST 7

Литература

1. Z80 - Technical Manual, Cupertino (USA): Zilog Inc. 1976.
2. Zilog. Z80CPU, Z80A CPU technical Manual, Cupertino, CA, 1977.
3. Zilog Data Book. Zilog Inc. Cupertino (Cal.) S.a., 1978.
4. Component Products - Zilog Inc. - Compbell (Cal.), 1984.
5. Zilog Inc., Components Data Book, USA, 1985.
6. Barden W, "The Z80 microcomputer handbook" Indianapolis: Howard W. Sams & Co. Jr.1978.
7. Zaks. Rodney. "Programming the Z80" Berkeley, Sybex, 3-Edition, 1982.
8. Leventhal L.A. "Z80 Assembly Language Programming" - Berkeley, Calif.: Osborne/McGrow-Hill, 1979.
9. Classen L. "Programmierung des Mikroprozessorsystems U880 - K1520" VEB Verlag Technik, Berlin. 1981.
10. Hasselberg M. "422 neue Z80-Befehle". MC. 1/1982.
11. Kieser B., Meder M. "Mikroprozessortechnik. Aufbau und Anwendung des Mikroprozessorsystems U880", 4 Auflage, VEB Verlag Technik, Berlin, 1986.
12. Classen L., Oefler U. "Technische Infomatik. Wissenspeicher Mikrorechner-programmierung" VEB Verlag Technik, Berlin, 1987.
13. Коффон Дж. "Технические средства микропроцессорных систем" - М., Мир, 1983.
14. Микрокомпьютерные медицинские системы: проектирование и применения". Под ред. У. Томпкинса, Дж. Уэбстера; М., Мир, 1983.
15. Морисита Ивао. "Аппаратные средства микроЭВМ" М, Мир,1988.
16. Рафикузаман М. "Микропроцессоры и машинное проектирование микропроцессорных систем" в 2-х кн. М, Мир, 1988.