Микропроцессор 8080

Эдуард Пройдаков

Так сложилось, что микропроцессор Intel 8080, появившийся на свет божий 1 апреля 1974 г., сыграл очень большую роль на определенном этапе развития ВТ в СССР. Причин тому было несколько: в отличие от довольно жесткой архитектуры 16-разрядного микропроцессора LSI 11, которая шла от миникомпьютеров, 8-разрядный 8080 был удивительно прост. Студенты старших курсов Физтеха, проходившие практику у меня в лаборатории в Институте проблем управления (ИПУ), собирали на нем одноплатные компьютеры у себя в общежитии. С элементной базой в лаборатории всегда было неплохо, и мы даже поощряли подобные вещи - так студент быстрее начинал понимать, что к чему.

На этом процессоре, и его аналогах, совместимых с ним, были сделаны следующие микро-ЭВМ:

- ВЭФормика (НИИФЭФ);
- К-1715 (Роботрон, ГДР);
- СМ-1800 (ИНЭУМ, Москва);
- АПОС-80 (п/я 300, Е. Цибульский, С. Макеев, Э. Пройдаков и др.)

Единственный существенный недостаток - процессор требовал трех уровней напряжения питания.

Система команд 8080 насчитывала 79 инструкций (см. Справочник по системе команд микропроцессора Intel 8080).

Киевское НПО "Кристалл" в конце 70-х годов выпустило микропроцессорный комплект К580, в который входили следующие микросхемы:

- К580ВМ80 - процессор, совместимый с i8080;
- К580ВВ51 - адаптер последовательного интерфейса, аналог Intel 8151;
- К580ВВ53 - интервальный таймер;
- К580ВВ55А - адаптер параллельного интерфейса;
- К580ВТ57 - контроллер прямого доступа в память;
- К580ВН59А - контроллер прерываний;
- К580ВГ72 - контроллер накопителя на гибких дисках;
- К580ВГ75 - контроллер электронно-лучевой трубки;

- К580ВГ79 - контроллер клавиатуры и индикации;
- К580ВК91А - контроллер канала общего пользования;
- К580ВГ92А - контроллер приборного интерфейса.

К сожалению, у меня описания этих микросхем не сохранились, поэтому буду признателен всем, кто пришлет описание программирования указанных БИС.

Отмечу, что у корпорации Intel для 8080/85 выпускалась микросхема арифметического ускорителя. Однако так как 8080 составлял реальную угрозу продвигаемому МЭПом микропроцессору "Электроника-60", то эта микросхема в СССР никогда не воспроизводилась.

Архитектура процессора 8080

Регистровый файл

A	флаги
B	C
D	E
H	L
PC (счётчик команд)	
SP (указатель стека)	

Пара регистров A + флаги (первая строка таблицы) образуют 16 разрядный регистр, именуемый PSW, причём A занимает старший байт слова, а флаги — младший. Аналогично регистры B и C образуют 16-разрядный регистр BC, регистры D и E – регистр DE, регистры H и L – регистр HL.

Регистр флагов

D7							D0
S	Z	x	AC	x	P	x	C

S – знак
Z – нуль
x – значение не определено

AC – вспомогательный перенос

P – чётность

C – перенос

Назначение регистров

A	– аккумулятор. Все арифметические и логические операции производятся только между A и другими регистрами или между A и байтом непосредственных данных.
B, C, D, E, H и L	– 8-разрядные регистры общего назначения.
HL	– регистровая пара, состоящая из двух 8-разрядных регистров (H – старший регистр, L -- младший), используется для косвенно-регистровой адресации 64 Кбайт памяти.
DE	– часто используемая регистровая пара, поскольку имеется команда обмена содержимым между парами HL и DE .
PC	– счётчик команд, содержит адрес очередной исполняемой команды.
SP	– указатель стека автоматически инкрементируется на 2 при записи пары регистров в стек (отдельный 8-разрядный регистр в стек записать нельзя, только парами) и декрементируется при извлечении из регистровой пары из стека.
F	– регистр флагов. Непосредственно недоступен программисту, но его в составе PSW можно сохранить в стеке, а потом извлечь в другую регистровую пару, если нужно специально установить или проверить нужные флаги.

Так как адресное пространство всего 64 Кбайт, то полный адрес занимает 2 байта. Команды этого процессора бывают одно-, двух- и трёхбайтными. В первом байте всегда содержится код операции. Единственный существенный недостаток 8080 – процессор требовал трёх уровней напряжения питания.

Справочник по системе команд микропроцессора Intel 8080

Команды этого процессора бывают одно-, двух- и трехбайтными. В первом байте всегда содержится код операции.

Обозначения.

A, B, ..., L - названия 8-разрядных регистров.

BC, DE, HL - названия регистровых пар, образующих 16-разрядные регистры.

SP - 16-разрядный указатель стека.

PSW - слово состояния программы, содержит регистр флагов.

a16 - двухбайтовый адрес.

d8 - байт непосредственных данных.

d16 - два байта непосредственных данных.

pp - номер порта ввода-вывода.

Команда	Код	Описание
ADD A	87	A←(A) + (A)
ADD B	80	A←(B) + (A)
ADD C	81	A←(C) + (A)
ADD D	82	A←(D) + (A)
ADD E	83	A←(E) + (A)
ADD H	84	A←(H) + (A)
ADD L	85	A←(L) + (A)
ADD M	86	A←Loc(HL) + (A)
ADI d8	C6	A←d8 + (A)
ADC A	8F	A←(A) + (A) + CY
ADC B	88	A←(B) + (A) + CY
ADC C	89	A←(C) + (A) + CY
ADC D	8A	A←(D) + (A) + CY
ADC E	8B	A←(E) + (A) + CY
ADC H	8C	A←(H) + (A) + CY
ADC L	8D	A←(L) + (A) + CY
ADC M	8E	A←Loc(HL) + (A) + CY

ACI d8	CE	A←d8 + (A) + CF
ANA A	A7	Проверка A
ANA B	A0	Логическое И B с A
ANA C	A1	Логическое И C с A
ANA D	A2	Логическое И D с A
ANA E	A3	Логическое И E с A
ANA H	A4	Логическое И H с A
ANA L	A5	Логическое И L с A
ANA M	A6	Логическое И Loc(HL) с A
ANI d8	E6	Логическое И непосредственные данные с A
CALL a16	CD	Передать управление подпрограмме по адресу a16
CZ a16	CC	Вызвать подпрограмму по адресу a16, если нуль
CNZ a16	C4	То же, если не нуль
CP a16	F4	То же, если плюс
CM a16	FC	То же, если минус
CC a16	DC	То же, если перенос
CNC a16	D4	То же, если нет переноса
CPE a16	EC	То же, если чётно
CPO a16	E4	То же, если нечётно
CMA	2F	Инвертировать A
CMC	3F	Инвертировать перенос
CMP A	BF	Установить флаг FZ
CMP B	B8	Сравнить A с B
CMP C	B9	Сравнить A с C
CMP D	BA	Сравнить A с D

CMP E	BB	Сравнить A с E
CMP H	BC	Сравнить A с H
CMP L	BD	Сравнить A с L
CMP M	BE	Сравнить A с Loc(HL)
CPI d8	FE	Сравнить A с непосредственными данными, заданными в команде
DAA	27	Десятичная коррекция аккумулятора (совершенно бесполезная команда. Я так и ни разу ей и не воспользовался:)
DAD B	09	Сложить BC с HL
DAD D	19	Сложить DE с HL
DAD H	29	Сложить HL с HL (удвоение HL)
DAD SP	39	Сложить SP с HL
DCR A	3D	A←(A) - 1 (декремент A)
DCR B	05	B←(B) - 1
DCR C	0D	C←(C) - 1
DCR D	15	D←(D) - 1
DCR E	1D	E←(E) - 1
DCR H	25	H←(H) - 1
DCR L	2D	L←(L) - 1
DCR M	35	Loc (HL)←(Loc(HL)) -1
DCX B	0B	BC←(BC) - 1
DCX D	1B	DE←(DE) -1
DCX H	2B	HL←(HL) - 1
DCX SP	3B	SP←(SP) -1
DI	F3	Запретить прерывания

EI	FB	Разрешить прерывания
HLT	76	Останов процессора
IN pp	DB	Ввести данные из порта pp
INR A	3C	A←(A) + 1 (инкрементировать A)
INR B	04	Инкрементировать B
INR C	0C	Инкрементировать C
INR D	14	Инкрементировать D
INR E	1C	Инкрементировать E
INR H	24	Инкрементировать H
INR L	2C	Инкрементировать L
INR M	34	Инкрементировать содержимое Loc(HL)
INX B	03	Инкрементировать BC
INX D	13	Инкрементировать DE
INX H	23	Инкрементировать HL
INX SP	33	Инкрементировать SP
JMP a16	C3	Перейти по адресу a16
JZ a16	CA	То же, если нуль
JNZ a16	C2	То же, если не нуль
JP a16	F2	То же, если плюс
JM a16	FA	То же, если минус
JC a16	DA	То же, если перенос
JNC a16	D2	То же, если нет переноса
JPE a16	EA	Перейти по адресу a16, если паритет чётный
JPO a16	E2	Перейти по адресу a16, если паритет нечётный
LDA a16	3A	Загрузить A из ячейки с адресом a16

LDAX B	0A	Загрузить A из ячейки с адресом Loc(BC)
LDAX D	1A	Загрузить A из ячейки с адресом Loc(DE)
LHLD a16	2A	Загрузить в HL содержимое ячейки с адресом a16
LXI B,d16	01	Загрузить в BC непосредственные данные d16
LXI H,d16	21	Загрузить в HL непосредственные данные d16
LXI SP,d16	31	Загрузить в SP непосредственные данные d16
MOV A,A	7F	Переслать из A в A
MOV A,B	78	Переслать из B в A (B←(A))
MOV A,C	79	Переслать из C в A
MOV A,D	7A	Переслать из D в A
MOV A,E	7B	Переслать из E в A
MOV A,H	7C	Переслать из H в A
MOV A,L	7D	Переслать из L в A
MOV A,M	7E	Переслать из Loc(HL) в A
MOV B,A	47	Переслать из A в B
MOV B,B	40	Переслать из B в B (ещё одна странная команда)
MOV B,C	41	Переслать из C в B
MOV B,D	42	Переслать из D в B
MOV B,E	43	Переслать из E в B
MOV B,H	44	Переслать из H в B
MOV B,L	45	Переслать из L в B
MOV B,M	46	Переслать из Loc(HL) в B
MOV C,A	4F	Переслать из A в C
MOV C,B	48	Переслать из B в C
MOV C,C	49	Переслать из C в C

MOV C,D	4A	Переслать из D в C
MOV C,E	4B	Переслать из E в C
MOV C,H	4C	Переслать из H в C
MOV C,L	4D	Переслать из L в C
MOV C,M	4E	Переслать из Loc(HL) в C
MOV D,A	57	Переслать из A в D
MOV D,B	50	Переслать из B в D
MOV D,C	51	Переслать из C в D
MOV D,D	52	Переслать из D в D
MOV D,E	53	Переслать из E в D
MOV D,H	54	Переслать из H в D
MOV D,L	55	Переслать из L в D
MOV D,M	56	Переслать из Loc(HL) в D
MOV E,A	5F	Переслать из A в E
MOV E,B	58	Переслать из B в E
MOV E,C	59	Переслать из C в E
MOV E,D	5A	Переслать из D в E
MOV E,E	5B	Переслать из E в E
MOV E,H	5C	Переслать из H в E
MOV E,L	5D	Переслать из L в E
MOV E,M	5E	Переслать из Loc(HL) в E
MOV H,A	67	Переслать из A в H
MOV H,B	60	Переслать из B в H
MOV H,C	61	Переслать из C в H
MOV H,D	62	Переслать из D в H

MOV H,E	63	Переслать из E в H
MOV H,H	64	Переслать из H в H
MOV H,L	65	Переслать из L в H
MOV H,M	66	Переслать из Loc(HL) в H
MOV L,A	6F	Переслать из A в L
MOV L,B	68	Переслать из B в L
MOV L,C	69	Переслать из C в L
MOV L,D	6A	Переслать из D в L
MOV L,E	6B	Переслать из E в L
MOV L,H	6C	Переслать из H в L
MOV L,L	6D	Переслать из L в L
MOV L,M	6E	Переслать из Loc(HL) в L
MOV M,A	77	Переслать из A в M
MOV M,B	70	Переслать из B в M
MOV M,C	71	Переслать из C в M
MOV M,D	72	Переслать из D в M
MOV M,E	73	Переслать из E в M
MOV M,H	74	Переслать из H в M
MOV M,L	75	Переслать из L в M
MVI A,d8	3E	Переслать d8 в A
MVI B,d8	06	Переслать d8 в B
MVI C,d8	0E	Переслать d8 в C
MVI D,d8	16	Переслать d8 в D
MVI E,d8	1E	Переслать d8 в E
MVI H,d8	26	Переслать d8 в H

MVI L,d8	2E	Переслать d8 в L
MVI M,d8	36	Переслать d8 в Loc(HL)
NOP	00	Нет операции
ORA A	B7	Проверить A и сбросить перенос
ORA B	B0	Логическая операция A ИЛИ B
ORA C	B1	Логическая операция A ИЛИ C
ORA D	B2	Логическая операция A ИЛИ D
ORA E	B3	Логическая операция A ИЛИ E
ORA H	B4	Логическая операция A ИЛИ H
ORA L	B5	Логическая операция A ИЛИ L
ORA M	B6	Логическая операция A ИЛИ M
ORI d8	F6	Логическая операция A ИЛИ d8
OUT pp	D3	Записать A в порт pp
PCHL	E9	Передать управление по адресу в HL
POP B	C1	Извлечь слово из стека в BC
POP D	D1	Извлечь слово из стека в DE
POP H	E1	Извлечь слово из стека в HL
POP PSW	F1	Извлечь слово из стека в PSW
PUSH B	C5	Поместить в стек содержимое BC
PUSH D	D5	Поместить в стек содержимое DE
PUSH H	E5	Поместить в стек содержимое HL
PUSH PSW	F5	Поместить в стек содержимое PSW
RAL	17	Циклический сдвиг CY + A влево
RAR	1F	Циклический сдвиг CY + A вправо
RLC	07	Сдвинуть A влево на один разряд с переносом

RRC	0F	Сдвинуть A вправо на один разряд с переносом
RIM	20	Считать маску прерывания (только в 8085)
RET	C9	Возврат из подпрограммы
RZ	C8	Возврат из подпрограммы, если FZ=0
RNZ	C0	Возврат из подпрограммы, если FZ=1
RP	F0	Возврат из подпрограммы, если FP=1
RM	F8	Возврат из подпрограммы, если FP=0
RC	D8	Возврат из подпрограммы, если FC=1
RNC	D0	Возврат из подпрограммы, если FC=0
RPE	E8	Возврат из подпрограммы, если паритет чётный
RPO	E0	Возврат из подпрограммы, если паритет нечётный
RST 0	C7	Запуск программы с адреса 0
RST 1	CF	Запуск программы с адреса 8h
RST 2	D7	Запуск программы с адреса 10h
RST 3	DF	Запуск программы с адреса 18h
RST 4	E7	Запуск программы с адреса 20h
RST 5	EF	Запуск программы с адреса 28h
RST 6	F7	Запуск программы с адреса 30h
RST 7	FF	Запуск программы с адреса 38h
SIM	30	Установить маску прерывания (только в 8085)
SPHL	F9	Загрузить SP из HL
SHLD a16	22	Записать HL по адресу a16
STA a16	32	Записать A по адресу a16
STAX B	02	Записать A по адресу Loc(BC)
STAX D	12	Записать A по адресу Loc(DE)

STC	37	Установить флаг переноса (CF=1)
SUB A	97	Вычесть A из A (очистить A)
SUB B	90	Вычесть B из A
SUB C	91	Вычесть C из A
SUB D	92	Вычесть D из A
SUB E	93	Вычесть E из A
SUB H	94	Вычесть H из A
SUB L	95	Вычесть L из A
SUB M	96	Вычесть M из A
SUI d8	D6	Вычесть d8 из A
SBB A	9F	Вычесть A из A (очистить A)
SBB B	98	Вычесть с заёмом B из A
SBB C	99	Вычесть с заёмом C из A
SBB D	9A	Вычесть с заёмом D из A
SBB E	9B	Вычесть с заёмом E из A
SBB H	9C	Вычесть с заёмом H из A
SBB L	9D	Вычесть с заёмом L из A
SBB M	9E	Вычесть с заёмом M из A
SBI d8	DE	Вычесть с заемом d8 из A
XCHG	EB	Обмен содержимым DE и HL
XTHL	E3	Обмен содержимого вершины стека с содержимым HL
XRA A	AF	Исключающее ИЛИ A с A (очистка A)
XRA B	A8	Исключающее ИЛИ B с A
XRA C	A9	Исключающее ИЛИ C с A
XRA D	AA	Исключающее ИЛИ D с A

XRA E	AB	Исключающее ИЛИ E с A
XRA H	AC	Исключающее ИЛИ H с A
XRA L	AD	Исключающее ИЛИ L с A
XRA M	AE	Исключающее ИЛИ Loc(HL) с A
XRI d8	EE	Исключающее ИЛИ d8 с A

Команды с кодами 08, 10, 18, 38, CB , D 9, DD , ED и FD в системе команд МП 8080 отсутствуют.

О выполнении некоторых команд i8080

Команды пересылки данных между регистрами кодируются в одном байте (это типичный случай регистровой адресации) следующим образом:

01DDDSSS

где DDD - номер регистра назначения; SSS - номер регистра приёмника. Соответственно 01 - код операции пересылки. Никаких флагов команды пересылки не устанавливают. На выполнение команды тратится один машинный цикл.

Кодировка номера регистра (DDD и SSS)

Код	Регистр
000	B
001	C
010	D
011	E
100	H
101	L
110	M
111	A

M – содержимое ячейки памяти, адресуемое регистровой парой HL .

Пересылка из ячейки памяти в регистр и из регистра в память осуществляется с помощью косвенно регистровой адресации. Это означает, что адрес ячейки памяти загружается в регистровую пару HL, а в командах типа MOV A,M (такие команды кодируются как 01DDD110) в регистр A будет загружено содержимое ячейки памяти, адрес которой содержится в HL. Так как в таких командах требуется обращение к памяти, то на их выполнение нужно два машинных цикла. Система команд процессора очень экономична - она не рассчитана на поддержку языков высокого уровня. Все это появится в Intel-процессорах позже.

Замечу, что средняя длина команды в типичной программе равна двум байтам, а для программ более поздних 16-разрядных процессоров типа 8086 она равна 4,1. Поэтому на логических программах 8-разрядные процессоры не сильно уступали 16-разрядным.

Аналогично работают и команды записи в память.

Команды непосредственной пересылки двухбайтные. В первом байте кодируются код операции и регистр, а второй содержит байт пересылаемых данных:

00DDD110 XXXXXXXX

Очевидно, что для исполнения команды требуется два цикла.

Более интересна версия этой команды MVI M,d8, когда байт непосредственных данных пишется в память.

Она кодируется так:

00110110 XXXXXXXX

Исполнение занимает три цикла.

Очень полезна группа команд LXI непосредственной загрузки регистровых пар непосредственным значением. Она позволяет одной командой переместить сразу два байта данных и широко используется программистами как в операциях адресной арифметики, так и при выполнении целочисленных вычислений.

Команда кодируется так:

00RP0001 xxxxxxxx zzzzzzzz

где RP - регистровая пара; xxxxxxxx - младший байт данных, zzzzzzzzz - старший байт данных.

При исполнении команды, требующей трех машинных циклов, старший байт данных грузится в старший регистр регистровой пары, а младший байт - в младший регистр. Название старшего регистра стоит в названии пары первым.

Команда прямой загрузки аккумулятора LDA a16 позволяет загрузить в него данные, на которые указывает адрес, содержащийся в самой команде.

Длина команды три байта. Кодируется она так:

00111010 xxxxxxxx zzzzzzzz

где xxxxxxxx - младшая часть адреса; zzzzzzzz - старшая часть адреса.

Исполнение занимает машинных четыре цикла.

Симметричная по действию команда STA.

Команда LHLD addr (мнемоника расшифровывается Load H and L Direct) загружает в L содержимое ячейки памяти по адресу, кодируемому во втором и третьем байтах команды (т. е. адресация прямая). В H загружается байт из ячейки addr+1.

Команда выполняется за пять машинных циклов. Обратная ей по действию команда SHLD (Store H and L Direct).

Команда LDAX reg (мнемоника от Load accumulator indirect). Содержимое ячейки памяти, адресуемой регистровой парой BC или DE, за два цикла загружается в аккумулятор.

Обратная по действию команда STAX reg.

Очень полезная команда XCHG. (H)←>(D), (L)←>(E). Выполняется за один цикл.

Арифметические команды

Начнем со сложения. Сложить аккумулятор с содержимым регистра ADD reg

Кодировка: 10000RRR

Выполняется за один цикл. Обратите внимание, что все арифметические команды изменяют флаги: Z,S,P,CY,AC.

Для выполнения многобайтового сложения необходимо учитывать перенос. Поэтому младшие байты слагаемых складываются с помощью команды ADD, а все последующие в помощью команды ADC reg. Она учитывает при сложении содержимое флага переноса. Понятно, что в этой системе команд легко написать арифметику, которая будет работать с целыми числами длиной несколько килобайт (что и было сделано в системе muMATH-80).

Аналогично устроены команды вычитания.

Разновидностью команды сложения является команда инкремента регистра, необходимая для адресной арифметики и организации циклов.

INR reg

Кодировка: 00ККК100

Устанавливаются все флаги, за исключением CY.

Обратная по действию команда DCR reg.

Довольно редко используется команда DAA (мнемоника от Decimal Adjust Accumulator) выполняет следующие действия: если содержимое младшего полубайта (нибла) аккумулятора меньше 9 или установлен флаг CY, то (A) + 6, если значение старшего полубайта больше 9 или установлен флаг CY, то 6 добавляется к содержимому старшего полубайта.

При этом устанавливаются все флаги.

Логические команды

ANA reg

Флаг CY сбрасывается, а AC устанавливается (в 8085). В микропроцессоре 8080 на эти флаги влияет результат операции над третьими битами операндов.

В командах групп XRA и ORA флаги CY и AC сбрасываются.

CMP reg - сравнить регистр (Compare register), вычитает содержимое регистра из аккумулятора. Содержимое аккумулятора не изменяется. Флаги устанавливаются как при вычитании. Z=1, если (A) = (reg). CR = 1, если (A) < (reg).

Циклический сдвиг влево RCL работает следующим образом. Содержимое аккумулятора сдвигается на одну позицию влево, т. е. каждый старший бит получает значение стоящего рядом с ним младшего бита. Содержимое седьмого бита переходит в нулевой бит аккумулятора:

(CY) ←(b7).

RRC - сдвигает содержимое аккумулятора вправо. (CY)← (b0), (b7)←(b0).

RAL - (Rotate Left trough Carry). Сначала (CY) заносится в младший бит аккумулятора, а потом в CY записывается содержимое старшего его бита.

Формально: (b0)←(CY), (CY)←(b7).

Аналогично со сдвигом вправо: (Bn)←(Bn+1), (CY)←(b0), (b7)←(CY).

Команда CMA (мнемоника от Complement Accumulator) инвертирует каждый бит аккумулятора, т. е. 0 становится 1 и наоборот. Флаги не устанавливаются.

Команда CMC инвертирует содержимое флага переноса. Другие флаги не устанавливаются.

По командам управления стоит отметить выполнение команды вызова подпрограммы.

CALL addr - при ее выполнении в стек записывается адрес следующей за CALL команды, значение указателя стека дважды декрементируется, а управление передается по указанному адресу.

Команда возврата из подпрограммы записывает в счетчик команд адрес из вершины стека, увеличивает указатель стека на 2 и передает управление на новый адрес. Поэтому если модифицировать адрес возврата в стеке, то можно перейти совсем в другое место.

Для этого есть, впрочем, более удобная возможность - команда PCHL. Она позволяет передать управление по адресу в регистровой паре HL.

Иногда в системных программах используется команда останова HLT. Процессор останавливается, регистры и флаги не устанавливаются. Для запуска важно подать сигнал Reset.

IN port - данные из порта с указанным номером считываются в аккумулятор. Циклов 3. Флаги не изменяются.

OUT port - содержимое аккумулятора помещается на шину данных для записи в указанный в команде порт.

Как видим, система команд проста и незатейлива. Ее легко осваивали даже электронщики. Написано множество эмуляторов системы 8080, 8085 и Z-80 для PC, а также обучающих программ для студентов.

Конечно, через двадцать лет можно и Бейсик забыть. Поэтому, возможно, какие-то моменты я пропустил. Буду искренне благодарен всем приславшим на адрес chief@pcweek.ru свои дополнения и замечания, примеры учебных программ и т. п.

Литература

1. Озерецковский С. К 25-летию первого микропроцессора Intel. http://www.ritmpress.ru/it/press/cwm/41_96/list.htm.
2. Эмулятор компьютера Радио-РК. http://www.members.tripod.com/~barsuk.
3. Сайт Николая Жеведя, посвященный истории компьютера ОРИОН-128 (конструкция опубликована в первом номере журнале "Радио" за 1990 г.) http://orion128.nm.ru/
4. Справочник по микропроцессорам.http://www.microprocessor.sscc.ru/chiplist.
5. Левенталь Л., Сэйвилл У. Программирование на языке ассемблера для процессоров 8080 и 8085. –М.: Радио и связь, 1987. -448 с.