Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение высшего образования
«Севастопольский государственный университет»

ИССЛЕДОВАНИЕ АРХИТЕКТУРЫ И СИСТЕМЫ КОМАНД ВОСЬМИРАЗРЯДНОГО МИКРОПРОЦЕССОРА

Методические указания

к выполнению лабораторных работ
для студентов, обучающихся по направлению
09.03.02 “Информационные системы и технологии”
дневной и заочной формы обучения

Севастополь

2022

УДК 004.732

Исследование архитектуры и системы команд восьмиразрядного микропроцессора. Методические указания к лабораторным занятиям по дисциплине "Технические средства информационных систем" / Сост. Чернега В.С., Дрозин А.Ю. — Севастополь: Изд-во СевГУ, 2022 — 31 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине “Технические средства информационных систем“. Целью методических указаний является помощь студентом в выполнении лабораторных работ. Излагаются теоретические и практические сведения о 8-разрядных микропроцессорах, необходимые для выполнения лабораторной работы, а также программа работы и требования к содержанию отчета.

Методические указания рассмотрены и утверждены на методическом семинаре и заседании кафедры информационных систем
(протокол № 1 от 30 августа 2022 г.)

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

Рецензент: Кротов К.В., канд. техн. наук, доцент кафедры ИС

Содержание

Лабораторная работа

Исследование архитектуры универсального 8-разрядного микропроцессора

1. Цель работы

Исследовать архитектуру и основные блоки 8-разрядного процессора. Исследовать взаимодействие основных блоков процессора при выполнении команд разных типов. Приобрести навыки написания и отладки ассемблерных программ в эмуляторе KP580 Emulator.

2. Краткие теоретические сведения

2.1. Структурная схема 8-разрядного микропроцессора

Структурная схема 8-разрядного микропроцессора типа 8080, назначение функциональных блоков и его функционирование подробно описано в [1 - 4], а также изображена на рисунке 2.1.

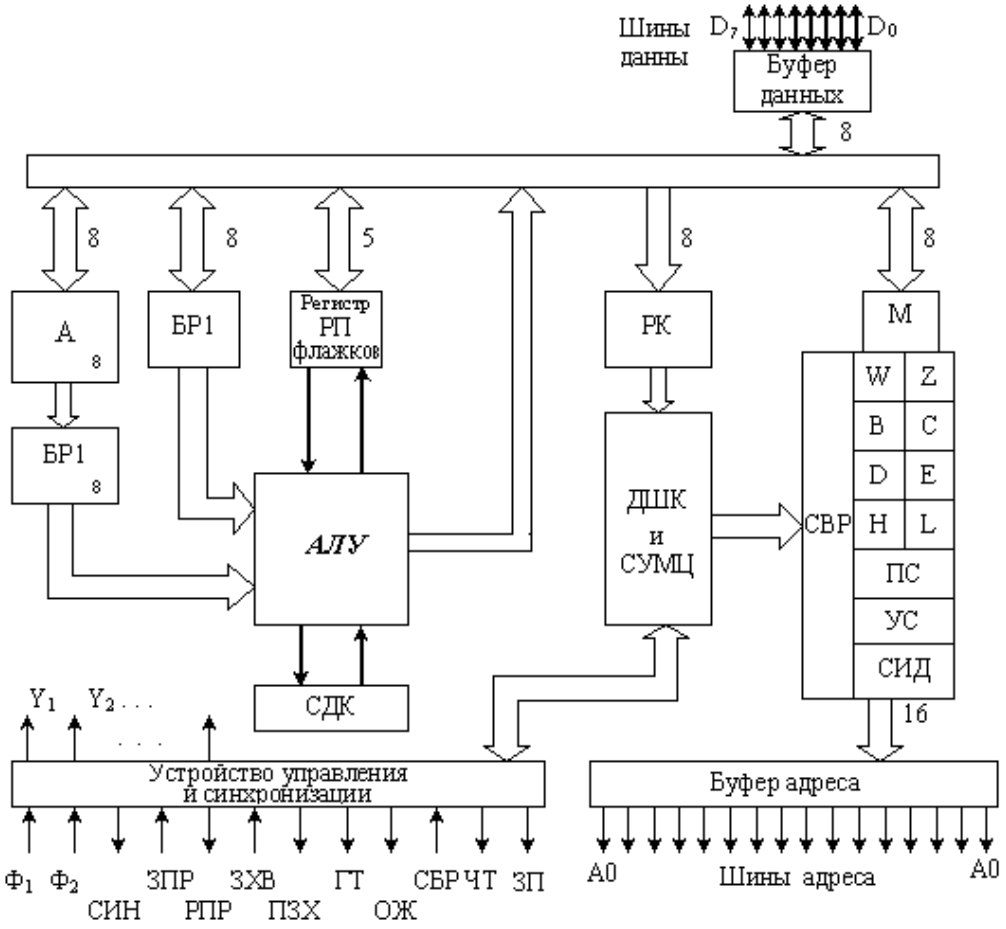

Рисунок 2.1 – Структурная схема 8-разрядного микропроцессора

Формат команды 8080 содержит от одного до трех байтов. Время, затрачиваемое на извлечение 1 байта информации или выполнения команды, определяемой одним машинным словом, называют ***машинным циклом*** *(М).* Каждая команда требует для выборки и выполнения от одного до пяти машинных циклов. Машинные циклы именуются M_1, M_2, M_3, M_4, M_5.

Выполнение каждой команды в МП происходит в строгой последовательности, определяемой кодом команды, и синхронизируется сигналами $Ф_1$ и $Ф_2$ тактового генератора. Период синхросигналов $Ф_1$ или $Ф_2$ называется ***машинным тактом*** (Т) Любой машинный цикл (М) включает от трех до пяти тактов: T_1, T_2, T_3, T_4, T_5. Каждый такт длится в течение одного периода синхросигнала (длительность такта при частоте 2 МГц = 0,5 мкс). Имеется три состояния, которые могут длиться неограниченное число тактов: WAIT (Ожидание), HOLD (Захват), HALT (Останов).

В течение такта T_1 содержимое программного счетчика ПС выдается на адресную шину, а на выходах СИН вырабатывается высокий потенциал. на шину данных подается 8-разрядный код, характеризующий выполняемый цикл. На первом такте каждого машинного цикла МП указывает тип выполняемого цикла с помощью 8-разрядного слова состояния цикла, выдаваемого на шины данных. Слово состояния выдается на шины данных лишь во время импульса СИНХР (такты T_1 и T_2), а используется на протяжении всего машинного цикла. Поэтому его необходимо записывать в специальный регистр слова состояния РгСС. Запись его осуществляется в момент совпадения сигналов СИНХР и $Ф_1$ на втором такте (рис.4.6). Слово состояния в последующем используется для формирования сигналов раздельного обращения к памяти и внешним устройствам, так как в процессоре такие сигналы отсутствуют (например, Чт относится как к памяти, так и к внешним устройствам).

За T_1 всегда следует такт T_2, в течение которого проверяется наличие сигналов подтверждения ГТ и ЗАХВАТ, а также проверяется не находится ли МП в состоянии останова HALT. Если на входе READY имеется сигнал готовности (высокий уровень), то МП переходит к такту T_3, в противном случае – в состояние ОЖИДАНИЕ (такт T_w) и находится в нем до тех пор, пока не появится сигнал готовности. Таким образом, сигнал ГОТ позволяет синхронизировать МП с памятью с любым временем доступа или с любым внешним устройством. Более того, сигнал ГОТ позволяет осуществить пошаговое выполнение программы.

Вовремя T_2 слово состояния цикла (PSWC) записывается в регистр состояния. Передним фронтом $Ф_2$ заканчивается формирование сигнала СИН, и вырабатывается единичный сигнал *Прием*, позволяющий поступить байту на вход МП через ШФ. В этом же такте T_2 из сигнала *Прием* и D_7 *PSWC* формируется сигнал *Чт Память*, позволяющий поступать данным из памяти на ШД микропроцессора. Изменения данных в этом такте восприниматься не будут так как их запись в МП осуществляется в фиксированные моменты времени в такте T_3.

В такте T_3 во время заднего фронта $Ф_1$ производится запись данных во внутренний регистр кода команды. Положительным фронтом $Ф_2$ оканчивается сигнал ПРИЕМ на выходе МП и сигнал ЧТ Память. Импульс на выходе Прием формируется в машинных циклах: чтение команды, ЧТ данных из памяти, прерывания, чтение из стека или внешнего устройства.

На основании декодирования команды ДШК схема управления формируют сигналы управления и синхронизации для внутренних пересылок данных, а также соответствующие дешифрируемой команде машинные циклы.

На последующих тактах T_4 и T_5 ДШК расшифровывает код команды, определяет количество байтов в команде, формирует команды на внутренние пересылки данных и подготавливает МП к выполнению следующих машинных циклов.

В конце последнего машинного цикла выполнения каждой команды анализируется наличие запроса прерывания на входе ЗПР. Если запрос присутствует и прерывания разрешены (команда *EI*), то МП входит в специальный цикл M_1, во время которого содержимое ПС не изменяется, формируется признак начала обработки прерывания INTA, а прерывающее устройство посылает в МП код команды RST с адресом прерывающей программы.

Самые *простые команды*, не требующие обращения к памяти, выполняются в течении одного машинного цикла *за четыре такта*, т.е. за 2 мкс, *самые длинные* – в течении 5 машинных циклов – за *18 тактов*, т.е. 9 мкс.

Выборка команд длиной 2 и 3 байта производится соответственно за два или три машинных цикла, при этом *первый байт* команды заносится *в РК*, второй в регистр *W*, а третий – в регистр *Z*.

С точки зрения программиста процессор представляет собой ряд программно-доступных регистров общего назначения, арифметико-логическое устройство, выполняющее операции сложения и вычитания двоичных 8-разрядных чисел, логические операции, операции сдвига и некоторые другие действия. Для выполнения умножения и деления операндов требуется составлять отдельные программы.

К регистрам общего назначения относятся аккумулятор A и регистры B,C,D,E,H и L. Имеется также регистр признаков – регистр флагов F. 8-разрядный аккумулятор A используется в большинстве команд арифметической и логической обработки. Обычно он адресуется неявно и является как источником, так и приёмником операндов и результата;

Признаки результата операции фиксируются во флаговом регистре F. Пять флагов C, P, AC, Z и M упакованы в байт, три разряда которого не используются. Флаги имеют следующее функциональное назначение:

C (carry) – признак переноса из старшего разряда АЛУ;

P (parity) – признак четного числа единиц в результате операции;

AC (auxiliary carry) – признак дополнительного переноса из младших четырех разрядов (младшей тетрады) АЛУ. Используется наиболее часто при сложении чисел в двоично-десятичной форме;

Z (zero) – признак нулевого результата;

S (sign) – знак результата.

Значение флага указывает на результат выполнения какой-либо операции. Флаги всегда устанавливаются или сбрасываются автоматически после выполнения очередной команды, влияющей на флаги, в зависимости от результата операции. При этом флаг считается установленным, если флаговый разряд принимает значение 1, и сброшенным, если значение разряда равно 0.

Регистры общего назначения (РОН), кроме аккумулятора могут объединяться в пары (B-C,D-E и H-L) и использоваться как 16-битовые регистры. Особенностью регистровой пары H-L является то, что она может неявно применяться для косвенной адресации памяти.

2.2. Система команд микропроцессора

2.2.1 Классификация команд

Команды восьмиразрядного процессора можно классифицировать по нескольким признакам. По виду выполняемых операций все команды МП можно разделить на следующие группы:

1) Передачи данных;
2) Арифметических операций;
3) Логических операций
4) Сдвига;
5) Регистровых операций;
6) Передачи управления;
7) Работа со стеком;
8) Ввода/вывода;
9) Управление процессором.

Эти команды занимают в памяти 1 байт, а при использовании непосредственного операнда - 2 байта. Команды можно классифицировать в соответствии с *адресом*, содержащимся в команде на следующие:

1) *Команды обращения к памяти.* Операция, указанная в команде, относится к содержимому, хранящемуся в памяти ЗУ по определенному адресу, т.е. команда задает адреса ячейки памяти ЗУ. Например, команда ADD 200 означает: выбрать число в качестве второго операнда для сложения с числом, хранимым в аккумуляторе и являющимся первым операндом.

2) *Команды обращения к регистру*. Для выполняемой операции не требуется адресация оперативной памяти. Операция выполняется, как правило, над одним операндом, хранящимся в аккумуляторе. Например, CLEAR (Очистить) - означает обнулить аккумулятор.

3) *Команды обращения к устройствам ввода-вывода*. Эти команды обеспечивают передачу данных между МП и периферийным оборудованием.

Кроме этого, команды классифицируют на группы по типу операций, которые должны выполняться.

2.2.2. Команды передачи данных

С помощью таких команд можно осуществить передачу данных от одного из регистров A, B, C, D, E, H и L к одному из регистров A, B, C, D, E, H и L. Кроме того, можно выбрать ячейку памяти M (1байт) микропроцессора, адресуясь к ней, как к одному из регистров. При этом для адресации памяти используется содержимое регистровой пары HL.

Есть еще одна возможность задать содержимое регистров: загрузить непосредственный операнд в выбранный регистр. Эти команды, однако, не позволяют выполнять запоминание в обратном направлении (не существует команд, по которым содержимое регистра могло бы быть помещено в поле операндов команды).

Формат команд:

(Метка:) MOV r_1, r_2 ; Комментарий r_1 = A,B,C,D,E,H,L,M r_2 = A,B,C,D,E,H,L,M.	Move data – передать данные

При этом r_1 определяет регистр или ячейку памяти, в которую производится запись, а r_2 – регистр или ячейку памяти, из которых извлекается число или непосредственный операнд.

Если $r_1=r_2$ то команда является пустой. Вместо этой команды можно использовать команду NOP. Пустая команда ни при каких обстоятельствах не изменяет содержимого регистров, памяти и состояния МП.

(Метка:) MVI r,число ; Комментари) r = A,B,C,D,E,H,L,M	Move immediate data - Передать непосредственный операнд.

По этой команде число, расположенное в поле операндов, заносится в регистр r.

Примеры:

M1: MOV A,B; <B>→<A>

M2: MOV M,A; <A>→<M> или <A>→<<HL>>

MOV B,M; <M>→<B> или <<HL>>→< B >.

MVI M, 12Q; 12Q→<M> или 12Q→ <<HL>>.

Q – обозначает, что число в восьмеричной системе.

Пример: передать содержимое ячейки 9B 73H в регистр A; содержимое регистров H и L не определено.

Выполнение:

```
MVI     H,9BH       ; <9B>→ H
MVI     L,73H       ; <73>→L
MOV     A, M        ; Выборка из памяти <HL>.
```

2.2.3. Команды регистровых операций

Команды регистровых операций занимают в памяти один байт. Они используются для того, чтобы содержимое того или иного регистра увеличить или уменьшить на единицу. Это относится к регистрам:

r = B, C, D, E, H, L.

Данные команды изменяют Z-, P- и S–биты состояния, хотя регистр A не участвует в операциях. Благодаря этому можно использовать названные регистры для организации программных циклов. Формат команд имеет следующий вид:

```
(Метка:)  INR  r          ; Увеличить содержимое регистра на единицу
                          ;  Increment register
```

Пример:

```
        INR C             ; <C> + 1 → <C>
        DCR r             ; Уменьшить содержимое регистра на 1
                          ; Decrement register
```

Для загрузки аккумулятора A содержимым ячейки памяти используются команды LDAX B и LDAX D.

```
        LDAX  B           ; Загрузить Акк. Содержимым ячейки памяти, адрес
                          ; которой
                          ; находится в регистровой паре B,C
                          ; Load Akkumulator
        LDAX  D           ; <<D,E>> → A. Обе эти команды однобайтные.
```

Команда LDA адрес загружает в Акк. Содержимое ячейки памяти, адресуемой вторым и третьим байтами команды.

[<B3> <B2>] → A.

Команда запоминания STA адрес производит противоположную передачу:

A → [<B3> <B2>] .

2.2.4. Команды работы со стеком

PUSH rp (rp = B, C; D, E; H, L, PSW) -*Запоминание значения регистров в стеке.* При выполнении команды содержимое пары регистров запоминается в стеке. Одной из пар регистров является слово состояния процессора *PSW*, которое состоит из содержимого *аккумулятора* (старший байт) и регистра состояния (младший байт). Нет такой команды, при выполнении которой запомнится в стеке только один регистр.

Пример:

PUSH B. При выполнении этой команды <B,C>, запоминается в вершине стека, а указатель стека *уменьшается* на 2. <B> запоминается первым, потом <C> заканчивает стек в его вершине.

POP rp (rp = B, C; D, E; H, L, PSW)- *Загрузка регистров из стека.* По этой команде пара регистров *rp* загружается содержимым вершины стека.

Пример: POP D. Эта команда загружает регистры D и E из вершины стека и *увеличивает* указатель стека на 2. Регистр E загружается первым.

Стек имеет следующие особенности:

- указатель стека содержит адрес ячейки, которая была занята самой последней (младший занятый адрес). Стек может быть расположен в любом месте памяти;
- данные запоминаются в стеке с использованием *предуменьшения,* то есть команды уменьшают указатель стека на 1 перед запоминанием каждого байта;
- данные загружаются из стека с использованием *послеувеличения,* то есть команды увеличивают указатель стека 1 после загрузки каждого байта;
- отсутствуют указатели выхода за границы стека в ту или иную сторону, что типично для *микропроцессоров* всех типов.

SPHL - Загрузка указателя стека (sp) – однобайтная команда.

<H> $\rightarrow$ <<SP> +1>; <L> $\rightarrow$ <<SP>>.

PCHL - Загрузка программного счетчика PC .

<H,L> $\rightarrow$ <PC>.

2.2.5. Команды логических и арифметических операций

В состав **команд логических операций** входят команды выполнения операций И, ИЛИ, исключающее ИЛИ (сумма по модулю 2), а также операция сравнения.

Список команд МП:

ANA *r* (M) <A> and <r>$\rightarrow$ <A>

XRA *r* (M) <A> xor <r>$\rightarrow$ <A>

ORA *r* (M) <A> or <r>→ <A>

CMP *r* (M) если <A> < <r>, то <C бит> = 1

если <A> = <r>, то <Z бит> = 1

если <A> > <r>, то <C бит> = 0

Аналогично с командами арифметических операций, существуют команды логических операций, когда вместо *r* задается число.

ANI число;

XRI число;

ORI число;

CPI число;

При выполнении команд сравнения содержимое аккумулятора сравнивается либо с содержимым регистра (CMP r), либо со значением числа (CPI число). При этом содержимое аккумулятора не изменяется. Биты состояния устанавливаются в зависимости от результата вычитания операнда из содержимого аккумулятора.

Команда исключающего ИЛИ

XRA r: (A)⊕(r)→A

производит поразрядное сложение операндов по mod 2. Команда XRA применяется для инвертирования определенных битов слова с помощью слова-маски на основе тождества

$$1 \oplus x = \overline{x}.$$

Другое применение XRA связано со сравнением слов на абсолютное равенство. В единственном случае, когда операнды поразрядно совпадают, результат операции содержит нули во всех разрядах (согласно тождеству X ⊕ X = 0), о чем сигнализирует флажок (бит состояние) Z=1.

Команды арифметических операций, в зависимости от типа применяемого операнда, занимают в памяти 1, 2 или 3 байта. По командам арифметических операций содержимое одного из регистров B, C D, E, H, L или содержимое ячейки памяти M (адресуемой регистровой парой H) или непосредственный операнд команды прибавляется (вычитается) к (из) содержимому регистра A, причем в операции может быть учтено значение C-бита. Значения битов состояния (C, S, Z, P) устанавливаются в зависимости от результата выполнения операции: C - переполнение; S - знак результата (минус S=1); P=1 – четное число единиц в аккумуляторе.

Примеры команд:

ADD r ; Сложить содержимое регистра (ячейки памяти) и аккумулятора

ADD M ;

ADC r (M) ; Сложить содержимое регистра (ячейки памяти) и аккумулятора

; с учетом бита переноса

SUB r (M)

SBB r (M)
ADI число ; Сложить непосредственный операнд и аккумулятор
ACI число

Аналогичные команды для вычитания SUI и SBI .

DAD rp - сложить содержимое регистровой пары rp (B или D) и аккумулятора, в роли которого выступает регистровая пара H,L.

Команды сдвига.

Команды данной группы требуют 1 байт памяти. По этим командам содержимое аккумулятора сдвигается влево или вправо на один разряд. В этой операции принимает и С-бит. Значение С-бита может в результате выполнения команды изменяться, а значения остальных флагов остается неизменным.

Формат команды имеет вид:

(Метка:) RLC ; Сдвинуть циклически содержимое аккумулятора влево
; Rotate accumulator left
RRC ; Сдвинуть циклически содержимое аккумулятора вправо

RAL ; Сдвинуть циклически содержимое аккумулятора влево
; через бит переноса

RAR ; Сдвинуть циклически содержимое аккумулятора вправо
; через бит переноса

Если обозначить разряды в А от A[0] до A[7], то выполняемую операцию можно описать следующим образом:

RLC: $A[7] \rightarrow < C_б >$; $A[m] \rightarrow A[m+1]$, $m = 0,...,6$; $A[7] \rightarrow A[0]$.
RAL: $A[7] \rightarrow < C_б >$; $A[m] \rightarrow A[m+1]$, $m = 0,...,6$; $C_б \rightarrow A[0]$.

2.2.6. Команды передачи управления

Эти команды часто называют командами перехода. Позволяют выполнять различные действия в соответствии со значением внешних сигналов или выработанных в процессе выполнения операций условий. Все типы команды делятся на команды безусловного и условного перехода. К безусловным командам относятся:

JUMP адрес ; Обеспечивается переход в программе по адресу,
; указанному в команде. При выполнении этой команды
; адрес перехода загружается в программный счетчик PC,

; причем текущее значение PC теряется.

SKIP	; Пропускается следующая команда программы
CALL имя	; Команда вызова подпрограммы. Осуществляется переход к ; подпрограмме с указанным именем
RET №	; Возврат из подпрограммы ; Осуществляется повторный запуск с адреса: 8 × №

При выполнении команды вызова CALL временно запоминается текущее содержимое программного счетчика (т.е. адрес команды, следующий за командой CALL и называемый адресом возврата), и загружается адрес перехода команды CALL в PC, а после выполнения программы адрес возврата передается в PC. Последняя функция реализуется специальной однобайтной командой RET. Для запоминания адресов возврата используется стековая память.

Имеется группа команд условного перехода, выполняемых в зависимости от значения одного из четырех флагов состояния (C, Z, S, P). Если условие перехода выполнено, то осуществляется переход по адресу, указанному в команде. В противном случае выполняется следующая команда. Группу команд условного перехода образуют 8 команд.

Метка:	JC	Адрес	; Перейти, если C $_{бит}$ = 1	*Jump if carry*
	JNC	Адрес	; Перейти, если C $_{бит}$ = 0	*Jump if no carry*
	JZ	Адрес	; Перейти, если Z $_{бит}$ = 1	*Jump if zero*
	JNZ	Адрес	; Перейти, если Z $_{бит}$ = 0	*Jump if not zero*
	JP	Адрес	; Перейти, если S $_{бит}$ = 0	*Jump if positive*
	JM	Адрес	; Перейти, если S $_{бит}$ = 1	*Jump if minus*
	JPE	Адрес	; Перейти, если P $_{бит}$ = 1	*Jump if parity even*
	JPO	Адрес	; Перейти, если P $_{бит}$ = 0	*Jump if parity odd*

2.2.7. Команды ввода/вывода

Имеется две команды*: IN* и *OUT* . Эти команды используются для того, чтобы передать (считать) байт данных из канала ввода в аккумулятор, или для того, чтобы содержимое Akk передать (записать) в выбранный канал вывода. Номера канала при этом задаются операндом.

Номера от *0* до *7* соответствуют каналам ввода, а номера от 8 до 31 – каналам вывода. Поэтому данные команды требуют всегда одного операнда. Команды ввода/вывода не влияют на биты состояния.

Формат команды:

(Метка:) *IN канал* ; Комментарий

Пример: *m1: IN 6* *;* считать один байт из 6-го канала

m2: OUT 29 *;* передать 1 байт из Akk в 29-й канал

Команда останова *HLT* . Она занимает в памяти 1 байт и используется для того, чтобы остановить выполнения команд в *микропроцессоре*. Содержимое регистров и памяти остается неизменным. Повторный запуск микропроцессора возможен только по сигналу прерывания.

3. Описание лабораторной установки

В качестве лабораторной установки используется персональный компьютер с установленным эмулятором процессоров Intel 8080 (программный модуль kp580_new). При запуске эмулятора появляется рабочее окно. в котором набирается текст ассемблерной программы (рисунок 3.1).

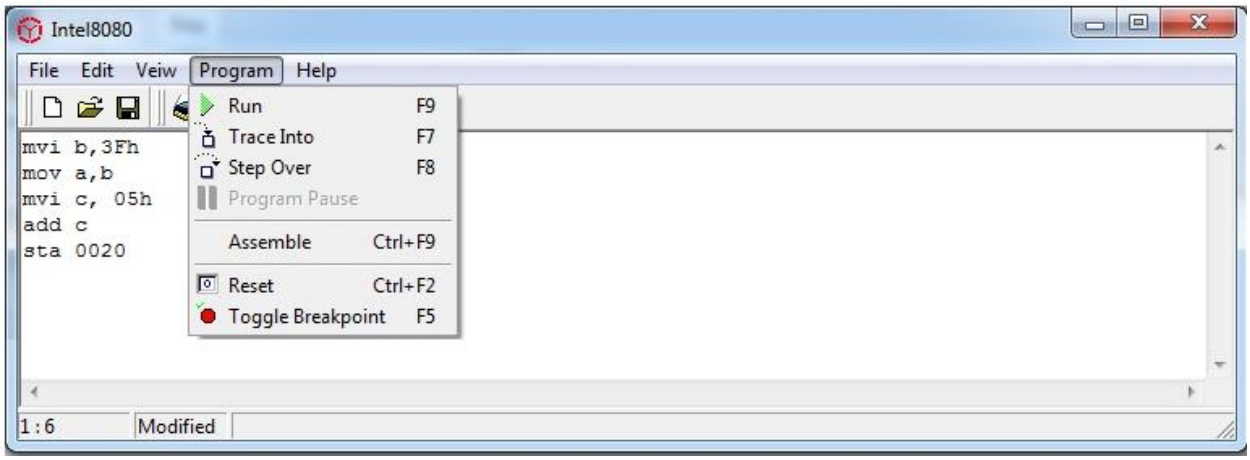

Рисунок 3.1 – Рабочее окно эмулятора Intel 8080

Для ассемблирования программы необходимо в командной линейке выбрать Program и нажать клавишу Assemble. Откроется окно (рисунок 3.2).

Экран отладчика разделен на пять окон. В основном окне отображается отлаживаемая программа в ассемблерной мнемонике и в машинных кодах, представленных в 16-ричной системе. Здесь же выводятся адреса ячеек памяти, в которых размещается отлаживаемая программа. Имеется два окна для отображения содержимого ОЗУ, а также окно с регистрами общего

назначения, программным счетчиком и указателем стека. Крайнее правое окно служит для отображения содержимого регистров (портов) ввода/вывода. Для пошаговой отладки следует последовательно нажимать клавишу F8.

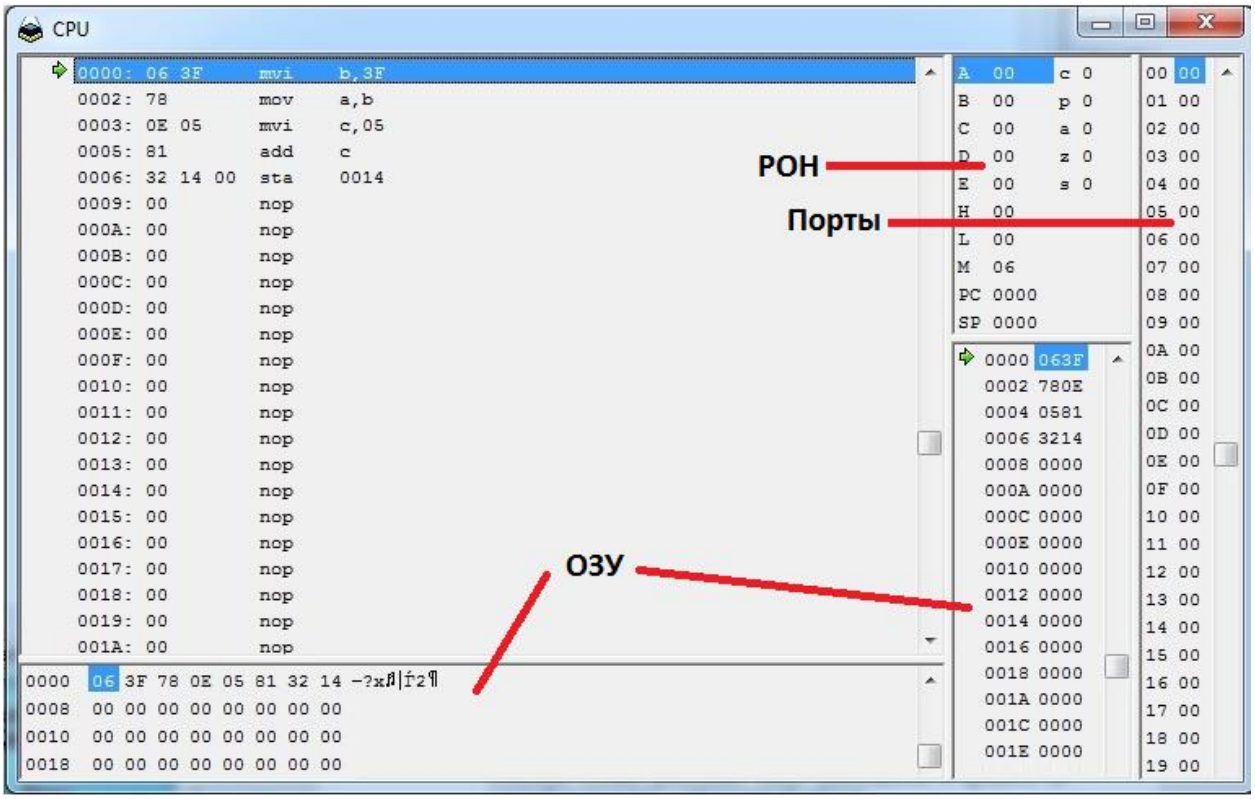

Рисунок 3.2 – Окно экранного отладчика

4 Программа лабораторной работы

4.1. Изучить архитектуру МП КР580ВМ80 (выполняется в процессе домашней подготовки к лабораторной работе).

4.2. Изучить основные команды МП КР580ВМ80 (выполняется в процессе домашней подготовки к лабораторной работе).

4.3. Изучить возможности эмулятора и экранного отладчика kp580_new. Исследовать изменение в основных блоках процессора в ходе выполнения команд различных типов (выполняется в процессе домашней подготовки к лабораторной работе).

4.4. Составить блок-схему алгоритма функционирования программы в соответствии с заданным вариантом (Приложение А).

4.5. Реализовать ассемблерную программу в соответствии с заданным вариантом.

4.6. Рассчитать длительности выполнения полученных программ в зависимости от используемых команд.

4.7. Сделать выводы по результатам проведенных исследований и расчетов.

Перечень заданий приведен в приложении А.

5 Содержание отчета

5.1. Цель и программа работы.
5.2. Структурная схема МП КР580ВМ80.
5.3. Описание взаимодействия блоков микропроцессора при выполнении команд различной длины и различных типов.
5.4. Результаты проведенных исследований и расчетов времени выполнения команд.
5.5. Выводы по работе с анализом результатов выполненных исследований и расчетов.

6. Контрольные вопросы

6.1. Расскажите об основных блоках процессора 8-разрядного микропроцессора и их назначении.
6.2. Объясните понятие машинного цикла. Перечислите виды машинных циклов МП КР580ВМ80.
6.3. Проанализируйте подробно с привлечением временных диаграмм работу процессора при различных режимах работы: программно-управляемом, обслуживания прерываний, прямого доступа в память.
6.4. Перечислите основные внешние выходы МП КР580ВМ80, расскажите об их назначении.
6.5. С какой целью процессор вначале каждого машинного цикла выдает слово состояния цикла?
6.6. Для чего служат регистры общего назначения и каковы особенности их применения?
6.7. Объясните назначение регистра признаков и как используется значение флагов.
6.8. Каково назначение регистров W и Z?
6.9. Расскажите о роли счетчика команд в организации выполнения программы.
6.10. Расскажите о роли указателя стека в организации выполнения программы.
6.11. Проведите классификацию команд ассемблера микропроцессора КР580ВМ80.
6.12. Расскажите о регистре флагов процессора КР580ВМ80. Какие команда влияют на состояние данного регистра, какие нет?

6.13. Приведите примеры команд логических операций ассемблера процессора КР580ВМ80 и назовите случаи их применения.
6.14. Расскажите о командах арифметических операций ассемблера процессора КР580ВМ80. Приведите примеры таких команд.
6.15. Какова роль счетчика команд в организации выполнения программы? Можно ли нарушить порядок изменения состояний программного счетчика?
6.16. Расскажите о командах ветвления ассемблера процессора КР580ВМ80. Приведите примеры таких команд с различными условиями.
6.17. Расскажите о принципах организации ветвлений в ассемблере процессора КР580ВМ80. Приведите примеры организации циклов с различными условиями останова.
6.18. Расскажите о командах логического и арифметического сдвигов, объясните разницу между ними. Приведите примеры выполнения сдвигов.
6.19. В чем заключается схожесть, а в чем отличие программного счетчика и указателя стека?

Список рекомендованной литературы

1. Майоров В.Г. Практический курс программирования микропроцессорных систем / В.Г. Майоров, А.В. Гаврилов — М.: Машиностроение, 1989. — 272 с.

2. Новиков Ю.В. Основы микропроцессорной техники: Учебное пособие/Ю.В. Новиков, П.К. Скоробогатов. — М.: Интернет-университет информационных технологий; БИНОМ, 2006. — 359 с.

3. Федотова Д.Э. Архитектура ЭВМ и систем [Электронный ресурс]: лабораторная работа. Учебное пособие/ Федотова Д.Э.— Электрон. текстовые данные. — М.: Российский новый университет, 2009.— 124 с.— Режим доступа: http://www.iprbookshop.ru/21263

4. Чернега В.С. Технические средства информационных систем. Конспект лекций / В.С. Чернега. – Севастополь: Изд-во СевГУ, 2022 – 160 с.

ПРИЛОЖЕНИЕ А

1. Заполнить вручную N ячеек памяти произвольными однобайтными натуральными числами. Размер N задается преподавателем.
2. Осуществить сортировку созданного массива натуральных чисел, используя алгоритм сортировки подсчетом. Направление сортировки задается преподавателем.
3. Вычислить сумму элементов массива, расположенные на позициях кратных трем.
4. Найти минимальный и максимальный элементы в массиве.

Заказ № _____ от «___» _____ 20__г. Тираж _____ экз.
Изд-во СевГУ