;) ASTRA-256

БИНАРНЫЙ КОМПЬЮТЕР ПРОГРАММИРУЕМЫЙ НА АССЕМБЛЕРЕ

СОДЕРЖАНИЕ

НАЗНАЧЕНИЕ ИНДИКАТОРОВ

Индикатор	Назначение
🟢 **AD7** – 🟢 **AD0**	Регистр адреса (аналог состояния шины адреса). Отображает текущий адрес ячейки памяти, содержимое которой можно изменить вручную. В процессе выполнения программы отображает содержимое счетчика команд.
🟢 **DI7** – 🟢 **DI0**	в режиме "DI view" отображает состояние регистра ввода данных DI (загорается индикатор 🟢 **DI**). Если индикатор ● **DI** не горит, то на 🟢 **DI0** – 🟢 **DI7** отображается состояние ячейки памяти, адресуемой 🟢 **AD0** – 🟢 **AD7**.
🟢 **DO7** – 🟢 **DO0**	отображает состояние регистра DO. Используется для вывода результата вычислений. В режиме отладчика "Debug" 🟢 **DO0** – 🟢 **DO7** могут быть использованы для отображения содержимого аккумулятора. (В этом случае включен индикатор 🟡 **ACC**).
🟢 **RUN**	показывает, что устройство находится в процессе выполнения программы. Зажигается после нажатия клавиши **RUN/STOP**.
🟡 **ACC**	сигнал о том, что 🟢 **DO0** – 🟢 **DO7** отображают состояние аккумулятора. Используется только в режиме "Debug", при условии нажатия кнопки **DO/A/<--**.
🟡 **DI**	сигнал о том, что на 🟢 **DI0** – 🟢 **DI7** отображается состояние регистра DI. DI обычно используется для ввода данных.
🔴 **HALT**	загорается красным цветом при выполнении команды HLT. Команда HLT временно приостанавливает работу программы до нажатия клавиши **ENTER**. (Поскольку машина находится при этом в состоянии выполнения программы, также включен индикатор 🟢 **RUN**). Загорается желтым цветом при выполнении команды INKBD.
🔵 **DEBUG**	светится, если машина находится в состоянии отладки "Debug".
🔵 **BIN**	светится, если машина находится в состоянии ввода данных в бинарном режиме.

НАЗНАЧЕНИЕ КЛАВИШ

Клавиша	Назначение
0 – 7	используются для ввода адреса в регистр адреса или данных в ячейку памяти, которую адресует регистр адреса, или регистр входных данных DI. Если установлен режим "Бинарный ввод" (при этом горит индикатор ● **BIN**), то эти кнопки меняют состояние регистров ● **AD0** – ● **AD7** или ● **DI0** – ● **DI7**, находящихся над ними (в зависимости от того, какая кнопка была перед этим нажата SLCT ADR / SAVE DATA / SAVE DI). При нажатии одной их этих кнопок инвертируется значение находящегося над кнопкой бита адреса ● **AD0** – ● **AD7** или бит данных ● **DI0** – ● **DI7**. Эти кнопки являются аналогом тумблеров, которые использовали "настоящие программисты" для ввода в память данных и программ. Если нажатием клавиши BIN/HEX установлен режим "16-ричный ввод" (индикатор ● **BIN** не горит), то клавиши 0 – 7 выполняют роль первых 8 клавиш ввода значения байта в 16-ричном формате. Во многих современных компьютерах, включая серию PC, для записи чисел в двоичной системе принят порядок записи разрядов - слева на право, от старших битов к младшим. Поэтому в нашем компьютере порядок следования клавиш от 0 до 7 на клавиатуре нетрадиционен. Это сделано для того, чтобы каждая из них была расположена непосредственно под тем битом-индикатором, значение которого она изменяет.
0 – F	предназначены для ввода значений в 16-ричном формате. Если установлен режим "16-ричный ввод" (при этом индикатор ● **BIN** не горит). Первое нажатие одной из этих кнопок устанавливает старшие четыре бита, второе нажатие устанавливает значение младших четырех бит вводимых данных.
A – Z	Клавиатура. Служит для ввода команд на Assembler. Кроме того, с этими клавишами работают команды которые возвращают код нажатых клавиш (см. Команды ввода / вывода).
BIN/HEX	"бинарный/16-ричный ввод". Если установлен режим "Бинарный ввод" (при этом горит индикатор ● **BIN**), то состояние регистров ● **AD0** – ● **AD7** или ● **DI0** – ● **DI7** меняется при нажатии кнопок 0 – 7, находящихся под каждой из них (в зависимости от того, какая кнопка перед была этим нажата SLCT ADR / SAVE DATA / SAVE DI). Если установлен режим "16-ричный ввод" (при этом не горит индикатор ● **BIN**), то ввод значения производится в 16-ричном формате клавишами 0 – F.

DO/A/<--	Клавиша "Backspace" стирает предыдущий введенный символ в режиме редактирования команд Ассемблера (**ASM** --> **SAVE DATA**). В остальных случаях нажатие этой клавиши переводит индикаторы ● **DO0** – ● **DO7** в режим отображения содержимого аккумулятора АС. При этом загорается индикатор ● **ACC**. Повторное нажатие этой клавиши возвращает ● **DO0** – ● **DO7** в режим отображения регистра DO и гасит ● **ACC**.
DI VIEW	режим отображения на индикаторах ● **DI0** – ● **DI7** содержимого регистра входных данных DI. (При этом загорается индикатор ● **DI**). Если этот режим выключен, то на ● **DI0** – ● **DI7** всегда отображается содержимое ячейки памяти, адресуемой (шиной адреса) ● **AD0** – ● **AD7**.
SLCT ADR	при нажатии на эту кнопку компьютер переходит в режим ввода адреса. (Задействованы индикаторы ● **AD0** – ● **AD7**). Адрес можно: • Ввести с **0** – **7** в режиме "Bin". • Ввести с **0** – **F** в режиме "Hex". • Увеличить на 1, нажав на клавишу **→**. • Уменьшить на 1, нажав на клавишу **←**. При этом, если выключен режим "Di View" (индикатор ● **DI** не горит), то на индикаторах ● **DI0** – ● **DI7** отображается состояние ячейки памяти, адресуемой ● **AD0** – ● **AD7**. Таким образом, у вас всегда есть возможность просмотреть содержимое ячейки памяти по заданному адресу.
ENTER	завершение ввода адреса или данных, (в зависимости от ранее нажатых **SLCT ADR** / **SAVE DATA** / **SAVE DI**), с записью этого значения в ранее выбранный регистр. Например, если была нажата клавиша **SLCT ADR**, то производится запись в регистр адреса.
SAVE DATA	В режиме “MEM”: при нажатии на эту кнопку компьютер переходит в режим ввода данных в адресуемую ячейку (задействованы индикаторы ● **DI0** – ● **DI7**). Адрес можно: • Ввести с **0** – **7** в режиме "Bin". • Ввести с **0** – **F** в режиме "Hex". В этом случае при нажатии на клавишу **ENTER** значение, отображаемое на ● **DI0** – ● **DI7**, записывается в ячейку памяти, которая адресуется ● **AD0** – ● **AD7**. Внимание! По окончании работы программы **SAVE DATA** значение регистра адреса автоматически увеличивается на 1 для удобства ввода следующей команды. Выход из режима **SAVE DATA** зависит от установки параметра "ВЫХОД ИЗ SAVE DATA ПО ENTER" в разделе "Настройки" и может производиться либо

	по нажатию клавиши **ENTER** либо при повторном нажатии **SAVE DATA**. В режиме “ASM”: при нажатии на эту кнопку компьютер переходит в режим ввода команд на языке Ассемблер. Переход к редактированию следующей команды осуществляется по нажатию клавиши **ENTER**. При этом значение регистра адреса автоматически увеличивается на количество байт введенной команды. Выход из режима ввода команд производится по повторному нажатию клавиши **SAVE DATA**.
SAVE DI	запись данных в регистр DI. Отличается от **SAVE DATA** тем, что при нажатии на кнопку **ENTER** данные записываются в регистр входных данных DI. По окончании работы программы **SAVE DI** значение регистра адреса не изменяется.
SAVE PROG	запись программы (по сути – дампа памяти) на виртуальный носитель. При нажатии на эту клавишу производится ввод имени программы. Запись программы в файл с указанным именем производится нажатием **ENTER**.
LOAD PROG	чтение программы (по сути – дампа памяти) с виртуального носителя в память. При нажатии на эту клавишу производится ввод имени программы. Чтение программы из файла с указанным именем производится нажатием **ENTER**.
RUN/STOP	запуск / остановка программы. Запуск начиная с текущего адреса AD. Если в момент запуска машина находилась в режиме отладки "Debug", то машина выходит из этого режима. При запуске зажигается индикатор ● **RUN**. Если программа была предварительно запущена, нажатием кнопки производится остановка программы.
DBG	перевод компьютера в режим "отладки" – пошаговое выполнение команд. При этом загорается индикатор ● **DEBUG**. В этом режиме команды выполняются пошагово. Функцию команды "шаг вперед" выполняют клавиша ␣. Выход из режима производится нажатием клавиши **DBG** или **RUN / STOP**.
MEM	отображение на экране состояния ячеек памяти устройства. Этот режим позволяет наблюдать, как изменяются значения ячеек памяти в процессе выполнения программы.

HELP VIEW	вывод на экран справочника команд. При нажатии на клавиши **A** – **Z** выводится список команд, которые начинаются с этой буквы, с их кратким описанием. Выход из режима производится нажатием любой функциональной клавиши.
ASM	Переход в режим отображения последовательности команд Assembler. При этом в строке последовательно отображается: • Адрес, по которому находится команда (он виден на предыдущем рисунке слева); • Символьное название команды; • Второй, третий и четвертый байты команды, если команда 2-х, 3-х или 4-х байтная соответственно. В этом режиме, при нажатии на клавишу **SAVE DATA** мы можем редактировать команду, записывая ее название с клавиатуры.
MENU	Вход в меню приложения. Выбор режимов: работы, настройки или справки.
EXIT	Выход из приложения. **Внимание!** Перед выходом из приложения сохраните свою работу в файл с помощью кнопки **SAVE PROG**. (После подтверждения выхода **данные автоматически не сохраняются**.)

СТРУКТУРА ВИРТУАЛЬНОЙ МАШИНЫ

Виртуальная машина представляет собой 8-битный двоичный компьютер, имеющий архитектуру фон Неймана, с набором из 75 команд.

В распоряжении программиста имеется отдельный 8-разрядный регистр – аккумулятор AC, с содержимым которого можно производить арифметические и логические операции, и оперативная память.

Объем оперативной памяти 256 байт.

В старших адресах оперативной памяти имеется также набор специальных 8-разрядных регистров:

- указатель стека SP;
- регистр флагов FR;
- регистр ввода данных DI;
- счетчик команд IP;
- регистр вывода данных DO.

РЕГИСТРЫ

Аккумулятор AC. Предназначен для выполнения арифметических и логических операций над содержимым. Содержимое этого регистра может быть выведено на индикаторы ● **DO0** – ● **DO7**, если в режиме "Debug" была нажата кнопка **DO/A/<--**.

Регистры виртуальной машины отображены на память и размещены в верхней области памяти с адреса 251 по 255.

Регистр - указатель стека SP (находится в памяти по адресу 251 (11111011)). В этом регистре содержится текущий адрес вершины стека.

- При записи байта в стек, значение SP уменьшается на 1 и байт записывается в ячейку памяти, адресуемую SP.
- При извлечении байта из стека (операция производится в обратном порядке), байт считывается из ячейки памяти, адресуемой SP, после чего сам SP увеличивается на 1.

Регистр флагов FR (находится в памяти по адресу 252 (11111100)). Отдельные биты FR имеют свое назначение. Назначение битов регистра флагов FR:

- **ZF** - флаг нуля (= 1, если предыдущая инструкция в аккумуляторе дала ответ = 0);
- **CF** - флаг переноса (= 1, если предыдущая инструкция в аккумуляторе дала бит переноса, иначе = 0);
- **TF** - флаг переполнения счетчика адреса (= 0, если работает в штатном режиме. = 1, если случилось переполнение счетчика адреса);

Регистр ввода данных DI (находится в памяти по адресу 253 (11111101)). В DI хранятся данные, которые "настоящие программисты" вводили "с тумблеров". Содержимое этого регистра отображается на индикаторах ● **DI0** – ● **DI7**.

Регистр счетчика команд IP (находится в памяти по адресу 254 (11111110)). Содержимое этого регистра можно прочитать или записать "с тумблеров". В регистре находится текущий адрес, из которого выбрана команда для выполнения. После выполнения текущей команды, содержимое регистра автоматически изменяется (на количество байт, которое занимает команда), указывая на адрес следующей команды. Содержимое этого регистра отображается на индикаторах ● **AD7** – ● **AD0**.

Регистр вывода данных DO (находится в памяти по адресу 255 (11111111)). В этот регистр, как правило, выводится результат, или какой-то отдельный байт результата вычислений. Содержимое этого регистра выводится на индикаторы ● **DO0** – ● **DO7**.

СИСТЕМА КОМАНД

Команды могут быть одно-байтные, 2-байтные, или 3-байтные (есть три 4-байтных команды). Т.е., при выполнении таких команд, счетчик команд IP будет увеличиваться на 1, 2 , 3 или 4 соответственно, для выбора следующей команды.

Команды управления выполнением программы
NOP | **00h** | 00000000b | 1 byte Нет операции (ничего не делает, "пустая" команда).
SPEED | **02h** | 00000010b | 2 bytes Установить скорость, с которой процессор будет проходить через каждую инструкцию после запуска программы. 2-й байт - параметр торможения скорости выполнения программы.
ADDRIP | **03h** | 00000011b | 2 bytes | TF Сложить содержимое указанной ячейки памяти с содержимым счетчика команд IP. Оставить результат в IP. 2-й байт - адрес ячейки памяти, содержимое которой нужно сложить с IP. Если при выполнении команды происходит переполнение регистра счетчика команд IP, тогда флаг TF = 1, иначе TF = 0.
HLT | **0Eh** | 00001110b | 1 byte Приостановить выполнение инструкций. Включает индикатор ● **HALT**, который горит красным цветом. Продолжить выполнение при нажатии клавиши **ENTER** или **RUN/STOP** со следующей команды. Обычно используется для ожидания ввода значения в регистр DI с клавиатуры.
STOP | **0Fh** | 00001111b | 1 byte Останавливает выполнение программы до нажатия клавиши **RUN/STOP**.
Команды копирования в/из аккумулятора
MOVLA | **10h** | 00010000b | 2 bytes | ZF Занести байт в аккумулятор.

2-й байт - данные, которые будут занесены в аккумулятор.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

MOVRA | **11h** | 00010001b | 2 bytes | ZF

Скопировать в аккумулятор содержимое ячейки памяти.
2-й байт - адрес ячейки памяти, данные из которой будут записаны в аккумулятор.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

MOVAR | **12h** | 00010010b | 2 bytes | ZF

Скопировать содержимое аккумулятора в указанную ячейку памяти.
2-й байт - адрес ячейки памяти.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

MOVIRA | **13h** | 00010011b | 2 bytes | ZF

Копирует в Аккумулятор содержимое ячейки памяти, адрес которой содержится в другой ячейке памяти.
2-й байт - адрес ячейки памяти, в которой содержится адрес другой ячейки памяти.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

MOVIAR | **14h** | 00010100b | 2 bytes

Копирует содержимое Аккумулятора в ячейку памяти, адрес которой содержится в другой ячейке памяти.
2-й байт - адрес ячейки памяти, в которой содержится адрес другой ячейки памяти.

MOVILR | **15h** | 00010101b | 3 bytes

Записывает буквальное значение в ячейку памяти, адрес которой содержится в другой ячейке.
2-й байт - значение, которое должно быть занесено в ячейку памяти.
3-й байт - адрес ячейки памяти, в которой содержится адрес другой ячейки памяти, в которую должно быть скопировано значение.

MOVAL | **16h** | 00010110b | 2 bytes | ZF

Помещает содержимое Аккумулятора во второй байт этой же команды.
2-й байт - байт, в который будет записано содержимое Аккумулятора. Изначально может иметь любое значение. Оно изменится в процессе выполнения команды.
Если в Аккумуляторе 00h, тогда флаг ZF = 1, иначе ZF = 0.

LOIRA | **17h** | 00010111b | 2 bytes | ZF

Заносит в Аккумулятор число из ячейки, адрес которой размещен в другой ячейке. После этого, в зависимости от значения флага CF, увеличивает или уменьшает на 1 значение адреса, содержащегося в указанной ячейке. Если CF=0, адрес увеличивается на 1, если CF=1, адрес уменьшается на 1.
2-й байт - адрес ячейки, в которой содержится адрес для выполнения косвенной адресации. После выполнения операции, содержимое этой ячейки увеличивается или уменьшается на 1.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

CLEARA | **E4h** | 11100100b | 2 bytes | ZF

Переносит содержимое Аккумулятора по указанному адресу, после чего обнуляет Аккумулятор.
2-й байт - адрес ячейки памяти, в которую будет записано содержимое Аккумулятора.
Всегда устанавливает ZF = 1.

Команды копирования в память

MOVLR | **20h** | 00100000b | 3 bytes

Занести байт в память по указанному адресу.
2-й байт - данные, которые будут занесены в ячейку памяти.
3-й байт - адрес ячейки памяти, в которую будут занесены данные.

MOVRR | **21h** | 00100001b | 3 bytes

Скопировать значение из одной ячейки памяти в другую.
2-й байт - адрес ячейки памяти, из которой будут прочитаны данные.
3-й байт - адрес ячейки памяти, в которую будут записаны данные.

MOVIRR | **22h** | 00100010b | 3 bytes

Копирует содержимое одной ячейки памяти в другую ячейку, при косвенной адресации обеих ячеек памяти.
2-й байт - адрес ячейки памяти с адресом другой ячейки памяти, из которой копируется значение.
3-й байт - адрес ячейки памяти, с адресом другой ячейки памяти, в которую должно быть скопировано значение.

CLEARR | **E5h** | 11100101b | 2 bytes

Помещает ноль в адресуемую ячейку памяти.
2-й байт - адрес ячейки памяти, в которую будет записан 00h.

Команды обмена

XCHGRA | **30h** | 00110000b | 2 bytes | ZF

Меняет местами содержимое аккумулятора и содержимое указанной ячейки памяти.
2-й байт - адрес ячейки памяти, с которой производится обмен значениями.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

XCHGRR | **31h** | 00110001b | 3 bytes

Меняет местами содержимое пары адресуемых ячеек памяти.
2-й байт - адрес ячейки памяти, с которой производится обмен значениями.
3-й байт - адрес ячейки памяти, с которой производится обмен значениями.

Команды арифметических-логических операций с аккумулятором

ADDLA | **40h** | 01000000b | 2 bytes | ZF, CF

Складывает содержимое аккумулятора с указанным значением. Помещает результат в аккумулятор.
2-й байт - значение, которое нужно сложить с содержимым аккумулятора.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

ADDRA | **41h** | 01000001b | 2 bytes | ZF, CF

Складывает содержимое аккумулятора со значением в адресуемой ячейке памяти. Помещает результат в аккумулятор.
2-й байт - адрес ячейки памяти, содержимое которой нужно сложить с содержимым аккумулятора.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

SUBLA | **42h** | 01000010b | 2 bytes | ZF, CF

Вычесть из аккумулятора указанное значение и оставить результат в аккумуляторе.
2-й байт - значение, которое нужно вычесть из содержимого аккумулятора.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0
Если в результате операции аккумуляторе получаем отрицательное число, тогда флаг CF = 1, иначе CF = 0.

SUBRA | **43h** | 01000011b | 2 bytes | ZF, CF

Вычесть из аккумулятора значение из указанной ячейки памяти и оставить результат в аккумуляторе.
2-й байт - адрес ячейки памяти, значение из которой нужно вычесть из содержимого аккумулятора.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0
Если в результате операции аккумуляторе получаем отрицательное число, тогда флаг CF = 1, иначе CF = 0.

ANDLA | **44h** | 01000100b | 2 bytes | ZF

Выполнить логическое "AND" с содержимым аккумулятора и указанным значением. Результат поместить в аккумулятор.
2 байт - значение, с которым нужно сделать логическое "AND".

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

ANDRA | **45h** | 01000101b | 2 bytes | ZF

Выполнить логическое "AND" содержимого аккумулятора со значением из указанной ячейки памяти. Результат поместить в аккумулятор.
2-й байт - адрес ячейки памяти, с содержимым которой нужно сделать операцию "AND".

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

ORLA | **46h** | 01000110b | 2 bytes | ZF

Выполнить логическое "OR" содержимого аккумулятора и указанного значения.

2-й байт - значение, с которым нужно сделать логическое "OR".

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

ORRA | **47h** | 01000111b | 2 bytes | ZF

Выполнить логическое "OR" содержимого аккумулятора со значением из указанной ячейки памяти.
2-й байт - адрес ячейки памяти, с содержимым которой нужно сделать логическое "OR".

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

XORLA | **48h** | 01001000b | 2 bytes | ZF

Выполняет операцию логическое "XOR" между содержимым аккумулятора и указанным значением. Результат помещается в аккумулятор.
2-й байт - значение, с которым нужно сделать логическое "XOR".

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

XORRA | **49h** | 01001001b | 2 bytes | ZF

Выполнить логическое "XOR" содержимого аккумулятора со значением из указанной ячейки памяти. Результат помещается в аккумулятор.
2-й байт - адрес ячейки памяти, со значением в которой нужно сделать логическое "XOR".

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

DECA | **4Ah** | 01001010b | 1 byte | ZF, CF

Уменьшает на 1 значения в аккумуляторе AC.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции аккумуляторе получаем отрицательное число, тогда флаг CF = 1, иначе CF = 0.

INCA | **4Bh** | 01001011b | 1 byte | ZF, CF

Увеличивает на 1 значение в аккумуляторе AC.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

DAA | **4Ch** | 01001100b | 1 byte | ZF, CF

Десятичная коррекция аккумулятора после сложения двоично-десятичных чисел.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

DAS | **4Dh** | 01001101b | 1 byte | ZF, CF

Десятичная коррекция аккумулятора после вычитания двоично-десятичных чисел.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

NOTA | **4Eh** | 01001110b | 1 byte | ZF

Инверсия содержимого аккумулятора.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

AAD | **3Eh** | 00111110b | 1 bytes | ZF

Преобразует число в Аккумуляторе из двоично-десятичного вида в двоичный вид.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

AAA | **3Fh** | 00111111b | 1 bytes | ZF, CF

Преобразует число в Аккумуляторе из двоичного вида в двоично-десятичный вид.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в Аккумуляторе получаем переполнение (значение, которое превышает FFh), тогда флаг CF = 1, иначе CF = 0.

ADDLACF | **88h** | 10001000b | 2 bytes | ZF, CF

Складывает содержимое Аккумулятора с буквальным значением и значением бита CF.
Помещает результат в Аккумулятор.
2-й байт - значение, которое нужно сложить с содержимым Аккумулятора.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в Аккумуляторе получаем переполнение (значение, которое превышает FFh), то флаг CF = 1, иначе CF = 0.

ADDRACF | **89h** | 10001001b | 2 bytes | ZF, CF

Складывает содержимое Аккумулятора со значением в адресуемой ячейке памяти и значением бита CF. Помещает результат в Аккумулятор.
2-й байт - адрес ячейки памяти, содержимое которой нужно сложить с содержимым Аккумулятора.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции в Аккумуляторе получаем переполнение (значение, которое превышает FFh), то флаг CF = 1, иначе CF = 0.

SUBLACF | **8Ah** | 10001010b | 2 bytes | ZF, CF

Вычесть из содержимого Аккумулятора буквальное значение и бит CF, оставить результат в Аккумуляторе.
2-й байт - значение, которое нужно вычесть из содержимого Аккумулятора.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0
Если в результате операции в Аккумуляторе получаем отрицательное число, то флаг CF = 1, иначе CF = 0.

SUBRACF | **8Bh** | 10001011b | 2 bytes | ZF, CF

Вычесть из Аккумулятора значение из указанной ячейки памяти и бит CF, оставить результат в Аккумуляторе.
2-й байт - адрес ячейки памяти, значение из которой нужно вычесть из содержимого Аккумулятора.
Если в результате операции в Аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0
Если в результате операции в Аккумуляторе получаем отрицательное число, то флаг CF = 1, иначе CF = 0.

Команды арифметических-логических операций с памятью

DECR | **50h** | 01010000b | 2 bytes

Уменьшает на 1 значения в указанной ячейке памяти.
2-й байт - адрес ячейки памяти, с которой производится операция.

INCR | **51h** | 01010001b | 2 bytes

Увеличивает на 1 значения в указанной ячейке памяти.
2-й байт - адрес ячейки памяти, содержимое которой увеличивается на единицу.

Команды сдвига аккумулятора

SHIFTLA | **60h** | 01100000b | 1 byte | ZF, CF

Сдвиг влево данных в аккумуляторе. Крайний левый бит из аккумулятора переходит во флаг переноса CF.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

SHIFTRA | **61h** | 01100001b | 1 byte | ZF, CF

Сдвиг вправо данных в аккумуляторе. Крайний правый бит из аккумулятора переходит во флаг переноса CF.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

ROLACF | **62h** | 01100010b | 1 byte | ZF, CF

Сдвиг влево данных в аккумуляторе через бит флага переноса CF. Крайний левый бит из аккумулятора переходит во флаг переноса CF. В младший бит помещается предыдущее значение флага переноса.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

RORACF | **63h** | 01100011b | 1 byte | ZF, CF

Сдвиг вправо данных в аккумуляторе через бит флага переноса CF. Крайний правый бит из аккумулятора отправляется во флаг переноса CF. В старший бит помещается предыдущее значение флага переноса.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

Команды сдвига в ячейках памяти

SHIFTLR | **70h** | 01110000b | 2 bytes

Сдвиг влево данных, в адресуемой ячейке памяти. Крайний левый бит из адресуемой ячейки теряется.
2-й байт - номер ячейки памяти, с которой производится операция.

SHIFTRR | **71h** | 01110001b | 2 bytes

Сдвиг вправо данных, в адресуемой ячейке памяти. Крайний правый бит из адресуемой ячейки теряется.
2-й байт - номер ячейки памяти, с которой производится операция.

Команды работы с битами аккумулятора и флагами
CBA | **80h** | 10000000b | 2 bytes | ZF Установить в 0 указанный бит в аккумуляторе. 2-й байт, номер бита в аккумуляторе, который сбрасывается в 0. Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
SBA | **81h** | 10000001b | 2 bytes Установить в 1 указанный бит в в аккумуляторе AC. 2-й байт - номер бита, который устанавливается в 1.
XCHGAA | **82h** | 10000010b | 1 byte | ZF Меняет местами старшие и младшие полбайта в аккумуляторе. Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
CLRCF | **83h** | 10000011b | 1 byte Сбрасывает флаг переноса CF в 0
CLRTF | **84h** | 10000100b | 1 byte Сбрасывает флаг TF в 0
MOVCFA | **86h** | 10000110b | 2 bytes | ZF Скопировать содержимое флага CF в указанный бит аккумулятора. 2-й байт - номер бита в аккумуляторе, которому присваивается значение бита CF. Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
MOVACF | **87h** | 10000111b | 2 bytes | ZF, CF Скопировать содержимое указанного бита аккумулятора в бит флага CF. 2-й байт - номер бита в аккумуляторе, значение которого записывается в бит CF. Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

Команды работы с битами в памяти
CBR | **90h** | 10010000b | 3 bytes Установить в 0 указанный бит в указанной ячейке памяти. 2-й байт, номер бита, который сбрасывается в 0. 3-й байт, адрес ячейки памяти, с которой производится операция.
SBR | **91h** | 10010001b | 3 bytes Установить в 1 указанный бит в адресуемой ячейке памяти. 2-й байт - номер бита, который устанавливается в 1. 3-й байт - адрес ячейки памяти, с которой производится операция.
MOVCFR | **92h** | 10010010b | 3 bytes Скопировать содержимое флага CF в указанный бит ячейки памяти. 2-й байт, номер бита, в который копируется значение флага CF. 3-й байт - адрес байта, с которым производится операция.
MOVRCF | **93h** | 10010011b | 3 bytes | CF Скопировать содержимое указанного бита ячейки памяти в бит флага CF. 2-й байт, номер бита, значение которого копируется во флаг CF. 3-й байт - адрес байта, с которым производится операция.
Команды работы со стеком
PUSHA | **A0h** | 10100000b | 1 byte | ZF Запись в стек содержимого аккумулятора. Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.
PUSHR | **A1h** | 10100001b | 2 bytes Запись в стек значения указанной ячейки памяти. 2-й байт - адрес ячейки памяти, значение из которой записывается в стек.
PUSHL | **A2h** | 10100010b | 2 bytes

Запись байта в стек.
2-й байт - значение, которое будет занесено в стек.

POPA | **A3h** | 10100011b | 1 byte | ZF

Переместить значение из стека в аккумулятор AC.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

POPR | **A4h** | 10100100b | 2 bytes

Переместить значение из стека в указанную ячейку памяти.
2-й байт - адрес ячейки памяти, в которую записываются данные.

MOVSPA | **A5h** | 10100101b | 1 byte | ZF

Занести содержимое указателя стека SP в аккумулятор.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

MOVASP | **A6h** | 10100110b | 1 byte | ZF

Занести содержимое аккумулятора в указатель стека SP.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

SETSP | **A7h** | 10100111b | 2 bytes

Записать байта в указатель стека SP.
2-й байт - данные, которые будут записаны в указатель стека.

INITSP | **A8h** | 10101000b | 1 byte

Записывает фиксированное начальное значение 251 (11111011b) в указатель стека SP.

Команды работы с подпрограммами и безусловный переход

CALL | **B0h** | 10110000b | 2 bytes

Вызывает выполнение подпрограммы, до тех пор, пока не встретит команду **RETURN** в подпрограмме. После чего возвращается из подпрограммы и начинает выполнять следующую команду.
2-й байт, адрес входа в подпрограмму.

RETURN | **B1h** | 10110001b | 1 byte

Команда возврата из подпрограммы, вызванной командой **CALL**.

JMP | **B2h** | 10110010b | 2 bytes

Безусловный переход. Перейти к выполнению команды, начиная с указанного адреса.
2-й байт - адрес перехода, с которого будет продолжено выполнение программы.

Команды условных переходов

LOOP | **C0h** | 11000000b | 3 bytes

Условный переход со счетчиком. Уменьшает содержимое указанной ячейки памяти на единицу. Если результат не равен нулю, переходит по адресу, указанному в 3-м байте команды. Иначе выполняет следующую команду.

2-й байт - адрес ячейки памяти, содержимое которой уменьшается на единицу.
3-й байт - адрес перехода, если после вычитания результат не равен нулю.

LOOPI | **C1h** | 11000001b | 3 bytes

Условный переход со счетчиком. Увеличивает содержимое указанной ячейки памяти на единицу. Если результат не равен нулю, переходит по адресу, указанному в 3-м байте команды. Иначе выполняет следующую команду.

2-й байт - адрес ячейки памяти, содержимое которой увеличивается на единицу.
3-й байт - адрес перехода, если после сложения результат не равен нулю.

JRBNZ | **C2h** | 11000010b | 4 bytes

Проверить указанный бит по указанному адресу. Если этот бит = 0, то начать выполнять следующую команду. Если он = 1, перейти по адресу, который указан в 4-м байте команды.

2-й байт - номер бита, который проверяется.
3-й байт - адрес байта, в котором проверяется бит.
4-й байт - адрес перехода, если проверяемый бит = 1.

JRBZ | **C3h** | 11000011b | 4 bytes

Проверить указанный бит по указанному адресу. Если этот бит = 1, то начать выполнять следующую команду. Если он = 0, перейти по адресу, который указан в 4-м байте команды.

2-й байт - номер бита, который проверяется.
3-й байт - адрес байта, в котором проверяется бит.
4-й байт - адрес перехода, если проверяемый бит = 0.

JZFNZ | **C4h** | 11000100b | 2 bytes

Проверить бит флага нуля ZF. Если ZF = 1, перейти по указанному адресу перехода. Если ZF = 0, выполнить следующую команду.
2-й байт - адрес перехода, если флаг ZF = 1.

JZFZ | **C5h** | 11000101b | 2 bytes

Проверить бит флага нуля ZF. Если ZF = 0, перейти по указанному адресу перехода. Если ZF = 1, выполнить следующую команду.
2-й байт - адрес перехода, если флаг ZF = 0.

JCFNZ | **C6h** | 11000110b | 2 bytes

Проверить бит флага переноса CF. Если CF = 1, перейти по указанному адресу. Если CF = 0, выполнить следующую команду.
2-й байт - адрес перехода, если флаг CF = 1.

JCFZ | **C7h** | 11000111b | 2 bytes

Проверить бит флага переноса CF. Если CF = 0, перейти по указанному адресу. Если CF = 1, выполнить следующую команду.
2-й байт - адрес перехода, если флаг CF = 0.

JTFNZ | **C8h** | 11001000b | 2 bytes

Проверить бит флага TF. Если TF = 1, перейти по указанному адресу перехода. Если TF = 0, выполнить следующую команду.
2-й байт - адрес перехода, если флаг TF = 1.

JTFZ | **C9h** | 11001001b | 2 bytes

Проверить бит флага TF. Если TF = 0, перейти по указанному адресу перехода. Если TF = 1, выполнить следующую команду.
2-й байт - адрес перехода, если флаг TF = 0.

JALR | **B7h** | 10110111b | 3 bytes

Сравнить содержимое Аккумулятора с содержимым адресуемой ячейки памяти. Если значение в Аккумуляторе меньше значения в адресуемой ячейке памяти, то перейти по адресу, указанному в третьем байте команды.
2-й байт - адрес ячейки памяти, в которой находится сравниваемое число.
3-й байт - адрес перехода.

JALL | **B8h** | 10111000b | 3 bytes

Сравнить содержимое Аккумулятора с буквальным значением. Если значение в Аккумуляторе меньше заданного значения, то перейти по адресу, указанному в третьем байте команды.
2-й байт - сравниваемое значение.
3-й байт - адрес перехода.

JAER | **B9h** | 10111001b | 3 bytes

Сравнить содержимое Аккумулятора с содержимым адресуемой ячейки памяти. Если значение в Аккумуляторе равно значению в адресуемой ячейке памяти, то перейти по адресу, указанному в третьем байте команды.
2-й байт - адрес ячейки памяти, в которой находится сравниваемое число.
3-й байт - адрес перехода.

JAEL | **BAh** | 10111010b | 3 bytes

Сравнить содержимое Аккумулятора с буквальным значением. Если значение в Аккумуляторе равно заданному значению, то перейти по адресу, указанному в третьем байте команды.
2-й байт - сравниваемое значение.
3-й байт - адрес перехода.

JAGR | **BBh** | 10111011b | 3 bytes

Сравнить содержимое Аккумулятора с содержимым адресуемой ячейки памяти. Если значение в Аккумуляторе больше значения в адресуемой ячейке, то перейти по адресу, указанному в третьем байте команды.
2-й байт - адрес ячейки памяти, в которой находится сравниваемое число.
3-й байт - адрес перехода.

JAGL | **BCh** | 10111100b | 3 bytes

Сравнить содержимое Аккумулятора с буквальным значением. Если значение в

Аккумуляторе больше заданного значения, то перейти по адресу, указанному в третьем байте команды.
2-й байт - сравниваемое значение.
3-й байт - адрес перехода.

JRLR | **BDh** | 10111101b | 3 bytes

Сравнить содержимое одной адресуемой ячейки памяти с содержимым другой адресуемой ячейки памяти. Если значение в первой ячейке меньше значения во второй ячейке, то перейти по адресу, указанному в четвертом байте команды.
2-й байт - адрес ячейки памяти, в которой находится первое сравниваемое число.
3-й байт - адрес ячейки памяти, в которой находится второе сравниваемое число.
4-й байт - адрес перехода.

JRER | **BEh** | 10111110b | 4 bytes

Сравнить содержимое одной адресуемой ячейки памяти с содержимым другой адресуемой ячейки памяти. Если значение в первой ячейке равно значению во второй ячейке памяти, то перейти по адресу, который указан в четвертом байте команды.
2-й байт - адрес ячейки памяти, в которой находится первое сравниваемое число.
3-й байт - адрес ячейки памяти, в которой находится второе сравниваемое число.
4-й байт - адрес перехода.

JRGER | **BFh** | 10111111b | 4 bytes

Сравнить содержимое одной адресуемой ячейки памяти с содержимым другой адресуемой ячейки памяти. Если значение в первой ячейке больше или равно значению во второй ячейке памяти, то перейти по адресу, указанному в четвертом байте команды.
2-й байт - адрес ячейки памяти, в которой находится первое сравниваемое число.
3-й байт - адрес ячейки памяти, в которой находится второе сравниваемое число.
4-й байт - адрес перехода.

Команды ввода/вывода

OUTDO | **D0h** | 11010000b | 1 byte

Записывает значение из аккумулятора AC в регистр DO (ячейка памяти 255).

INDI | **D1h** | 11010001b | 1 byte | ZF

Записывает значение из регистра входных данных DI (ячейка по адресу 253) в аккумулятор. Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

INKBD | **D2h** | 11010010b | 1 byte | ZF

Записывает код последней нажатой клавиши в аккумулятор. Приостанавливает выполнение программы и включает желтый цвет индикатора ● **HALT** до нажатия любой клавиши, код которой помещает в AC.
Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

OUTKBD | **D3h** | 11010011b | 1 byte | ZF

Записывает байт аккумулятора в порт управления цветом клавиши. Младшие 6 бит определяют адрес клавиши, старшие 2 бита управляют цветом клавиши.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

OUTCLRKBD | **D4h** | 11010100b | 1 bytes

Гасит подсветку всех клавиш (цифры + алфавит), кроме функциональных.

INCOLKBD | **D5h** | 11010101b | 1 bytes

Возвращает в Аккумулятор цвет клавиши. Перед вызовом команды код клавиши помещается в 6 младших бит Аккумулятора. После выполнения команды код цвета возвращается в два старших бита Аккумулятора.

Расширенные команды (цепочные операции и умножение)

MOVSTR | **E0h** | 11100000b | 4 bytes | TF

Копирует заданное количество байт из ячеек памяти, начиная с "адреса-источника", в ячейки памяти, начиная с "адреса-получателя".
2-й байт - количество копируемых ячеек памяти.
3-й байт - адрес первой ячейки памяти в массиве, из которого будут прочитаны данные.
4-й байт - адрес первой ячейки памяти в массиве, в который будут записаны данные.

Если в результате выполнения команды значение адреса ячейки памяти выходит за пределы адресного пространства, тогда флаг TF = 1, иначе TF = 0.

MULRA | **E1h** | 11100001b | 3 bytes

Умножить значение в аккумуляторе на значение в указанной ячейке памяти. Третий байт - адрес ячейки памяти, в которой будет размещен младший байт результата. Старший байт результата размещается по адресу, на 1 большему от адреса младшего байта.

2-й байт - адрес ячейки памяти, в которой находится второй множитель.
3-й байт - адрес ячейки памяти, в которой будет размещен младший байт результата.

Если в результате операции в аккумуляторе получаем 00h, тогда флаг ZF = 1, иначе ZF = 0.

DIVRA | **E2h** | 11100010b | 3 bytes | ZF | CF

Делить значение в указанной ячейке памяти на значение в аккумуляторе.
2-й байт - адрес ячейки памяти, где находится младший байт 16-битного делимого числа. Старший байт идет за ним.
3-й байт - адрес ячейки памяти, в которой будет находится младший байт результата.

Старший байт результата размещается по адресу, на 1 большему от адреса младшего байта. Дробная часть (остаток от деления) будет находится по адресу, на 2 большему от младшего байта результата.

При делении на ноль или переполнении 4-й бит регистра флагов FR устанавливается в 1. Иначе он = 0.
Если в аккумуляторе находится значение 00h, тогда флаг ZF = 1, иначе ZF = 0.
Если в результате операции получаем переполнение (значение, которое превышает FFFFh), тогда флаг CF = 1, иначе CF = 0.

Рекомендуется вместо **DIVRA** использовать команду умножения **MULRA** на число, обратное делителю.

RETAD | **E3h** | 11100011b | 2 bytes | TF

Предполагается, что за этой командой всегда следует двухбайтная команда JMP. Запоминает в указанной ячейке памяти так называемый "адрес возврата." Для вычисления значения "адреса возврата" команда "узнает" адрес расположения своего собственного первого байта, и прибавляет к нему 4.
2-й байт - адрес ячейки памяти, в которую будет записан "адрес возврата".
Если при выполнении команды адрес возврата больше FFh, то флаг TF = 1, иначе TF = 0.

X | **E6h** | 11100110b | 2 bytes

Выполняет одну инструкцию, адрес которой задан вторым байтом команды, после чего продолжает выполнение программы.
2-й байт - адрес первого байта инструкции, которая должна быть вызвана.
Внимание! При вызове этой командой инструкций условных и безусловных переходов, а также самой себя, корректная работа этих команд не гарантируется.