:) ASTRA-256

ASSEMBLY

BINARY

COMPUTER

TABLE OF CONTENTS

LEDs PURPOSE

LED	Purpose
● **AD7** – ● **AD0**	Address register. Displays a current memory address of the content that can be changed (is analogous to the status of the address bus). During program execution displays the contents of the Instruction Pointer IP.
● **DI7** – ● **DI0**	display the status of the Data Input register DI in the 'DI view' mode (the ● **DI** LED lights up). If the ● **DI** is off, ● **DI0** – ● **DI7** show the status of a memory cell addressed by ● **AD0** – ● **AD7**.
● **DO7** – ● **DO0**	display the status of the Data Output register DO. Used to display the calculation results. In the 'Debug' mode ● **DO0** – ● **DO7** can display the contents of the Accumulator. (In this case, the ● **ACC** LED is turned on).
● **RUN**	indicates that the device is executing a program. Lights up after pressing the **RUN/STOP** key.
● **ACC**	indicates that ● **DO0** – ● **DO7** LEDs are in the Accumulator status mode. It's used only in the 'Debug' mode when the **DO/A/<--** key is pressed.
● **DI**	indicates that ● **DI0** – ● **DI7** LEDs are displaying the status of the DI register. The DI is usually used for data input.
● **HALT**	lights up red when the 'HALT' instruction is executed. HALT temporarily pauses the program execution until the **ENTER** key is pressed. (Since the computer is in the program execution process, the ● **RUN** is also on). Lights up yellow when the INKBD instruction is executed.
● **DEBUG**	lights up when the machine is in the 'Debug' mode.
● **BIN**	lights up if the machine is in the Binary Data Input mode.

KEYS PURPOSE

Key	Purpose
0 – 7	used to enter data into the address register, or into the memory cell addressed by the AD register, or into the Data Input register DI. If the 'Binary Input' mode is set (the ● **BIN** LED is on), these keys affect the status of the ● **AD0** – ● **AD7** or ● **DI0** – ● **DI7** registers located above them, (depending on key pressed before: SLCT ADR / SAVE DATA / SAVE DI). When one of the keys is pressed, the value within the corresponding address bit ● **AD0** – ● **AD7** or data bit ● **DI0** – ● **DI7**. is inverted. These keys are analogous to the toggle switches that 'real programmers' used to enter data, programs to computer memory. If by pressing the BIN/HEX key the Hexadecimal Input mode is set (the ● **BIN** is off), the keys 0 – 7 are used as the first 8 keys to enter the value in hexadecimal format. In many modern computers, including the PC series, the conventional order of writing numbers in the binary system is from left to right, from high bits to low bits. Thus in our Computer, the order of keys from 0 to 7 on the keyboard is not conventional. This is done so that each key is located directly under the LED the value it changes.
0 – F	used to enter data in hexadecimal format if the 'Hexadecimal Input' mode is set (the ● **BIN** LED is off). The first press of one of the keys sets the upper four bits, the second press sets the value of the lower four bits of the data.
A – Z	Keyboard. Used to input instructions in Assembly language. In addition, these keys return the code of the pressed keys. (See Input / Output Instructions).
BIN/HEX	'Binary / Hexadecimal input'. If the 'Binary Input' mode is set (the ● **BIN** is on), the keys 0 – 7 change the status of the registers ● **AD0** – ● **AD7** or ● **DI0** – ● **DI7** located above them (depending on the key pressed before: SLCT ADR / SAVE DATA / SAVE DI). If the 'Hexadecimal Input' mode is set (the ● **BIN** LED is off), keys 0 – F are used to enter a value in 16-hex format.
DO/A/<--	The 'Backspace' key deletes a previous character entered in the Assembly instructions editing mode (ASM --> SAVE DATA). In other cases, pressing this key switches the ● **DO0** – ● **DO7** LEDs to the Accumulator contents display mode. (● **ACC** LED turns on). Pressing the key again returns ● **DO0** – ● **DO7** to the DO register display mode (● **ACC** turns

	off).
DI VIEW	If the mode is on (● **DI** LED lights up) ● **DI0** – ● **DI7** display the contents of the Data Input register. If the mode is off, ● **DI0** – ● **DI7** LEDs always display the contents of the memory cells addressed by ● **AD0** – ● **AD7**.
SLCT ADR	by pressing this key, the Computer switches to the Address Input mode. (● **AD0** – ● **AD7** LEDs are involved). Address can be: • Entered from **0** – **7** keys in the 'Bin' mode. • Entered from **0** – **F** keys in the 'Hex' mode. • Incremented by 1 by pressing → key. • Decremented by 1 by pressing ← key. If the 'Di View' mode is off (● **DI** LED is off), then ● **DI0** – ● **DI7** LEDs display the status of a memory cell addressed by ● **AD0** – ● **AD7**. Thus, you can always view the contents of a memory cell at a given address.
ENTER	end of data input or an address input (depending on the previously pressed **SLCT ADR** / **SAVE DATA** / **SAVE DI**), writing a value to a specified register. For instance, if **SLCT ADR** key was previously pressed, a value is written to the adress register.
SAVE DATA	In the 'MEM' mode: by pressing this key Computer switches to the mode of entering data to addressed memory cell (● **DI0** – ● **DI7** LEDs are involved). Address can be: • Entered from **0** – **7** keys in the 'Bin' mode. • Entered from **0** – **F** keys in the 'Hex' mode. In this case, by pressing the **ENTER** key a value displayed by ● **DI0** – ● **DI7**, LEDs is written to the memory cell addressed by ● **AD0** – ● **AD7**. Note! For convenience of entering the next instruction, at the end of saving data a value of the address register automatically increments by 1. Exiting the **SAVE DATA** mode depends on the 'EXIT SAVE DATA BY ENTER' option set in the Settings. It can be done either by the **ENTER** or by pressing the **SAVE DATA** again. In the 'ASM' mode: by pressing this key Computer switches to the Assembly instructions editing mode. To go to the next instruction press the **ENTER** key. In this case, a value of the address register automatically increments by the number of bytes of the entered instruction.

	To exit the mode press **SAVE DATA** again.
SAVE DI	writing data to the DI register. Pressing the **ENTER** key writes the data to the Data Input register DI. At the end of the **SAVE DI** program a value of the address register does not change.
SAVE PROG	writing a program (essentially a memory dump) to a virtual medium. After pressing this key the program name is entered. By pressing the **ENTER** a program is written to a file.
LOAD PROG	reading a program (essentially a memory dump) from a virtual medium into memory. After pressing the key, a program name is entered. By pressing the **ENTER** a program is reading from a file.
RUN/STOP	program start / stop. Run a program starting from the current address AD. If at the time of start the machine was in the 'Debug' mode, it exits the mode. When starting, the ● **RUN** LED lights up. Stops the program if it was previously launched.
DBG	switch Computer to the 'Debug' mode - step-by-step execution of instructions. At the time, the ● **DEBUG** LED lights up. In this mode, the instructions are executed step by step. To step forward use the **␣** key. To exit the mode press the **DBG** key or the **RUN / STOP** key.
MEM	display on the screen the memory cell status. The mode allows to observe how a values within memory cells are changing during program execution.
HELP VIEW	display the Instruction Reference on the screen. Pressing the **A** – **Z** keys displays an instruction list starting with a corresponding letter, and a brief description of an instructions. To exit the mode, press any function key.
ASM	Switch to the Assembly instructions display mode. In this case, in a row the following is displayed: • A memory address of instruction; • A symbolic name of the instruction; • The second, third and fourth bytes of the instruction, if the instruction is 2, 3 or 4-byte, respectively.

	In this mode, you can edit the instruction by writing its name from the keyboard after pressing the **SAVE DATA** key.
MENU	Enter the App Menu. Selecting modes: operation, settings, or reference.
EXIT	Exit the App. **Caution!** Before exiting, save your work to a file using the **SAVE PROG** key. (After confirming the exit, **the data is not saved automatically**.)

THE STRUCTURE OF VIRTUAL MACHINE

The software represents a virtual 8 bit binary computer with a von Neumann architecture. The instruction set includes 75 instructions.

At the programmer's disposal are a separate 8-bit register as Accumulator (AC) to perform arithmetic and logical operations with its values, RAM.

RAM capacity is 256 bytes.

There is also a set of 8-bit registers in the high addresses of RAM which are set aside for specific uses:

- Stack Pointer SP;
- Flags Register FR;
- Data Input register DI;
- Instruction Pointer IP;
- Data Output register DO.

REGISTERS

Accumulator AC. Designed to perform arithmetic and logical operations with the contents. If in the Debug mode the **DO/A/<--** key was pressed, the contents of this register can be shown on the ● **DO0** – ● **DO7** LEDs.

Virtual machine registers are mapped into memory and located in the upper memory space from address 251 to 255.

Stack Pointer SP (memory address 251 (binary 11111011)). The register contains the address value of the top of the stack (TOS).

- A byte is pushed onto the stack by decrementing SP by 1 and writing a byte at the new TOS.
- When a byte is popped off the stack (operation is performed in reverse order). A byte is read from a memory cell addressed by SP, then SP increments by 1.

Flags Register FR (memory address 252 (binary 11111100)). Discrete bits within FR register are as follows:

- **ZF** - Zero Flag (if = 1, the previous instruction in the Accumulator resulted in a zero answer);
- **CF** - Carry Flag (if = 1, the previous instruction in the Accumulator resulted in a Carry Flag, otherwise = 0);
- **TF** - Trap Flag (= 0 if it works in the normal mode. = 1 if overflow of the address counter occurred);

Data Input register DI (memory address 253 (binary 11111101)). DI stores data that 'old programmers' entered 'from toggle switches'. The contents of the register are displayed by LEDs ● **DI0** – ● **DI7**.

Instruction Pointer IP (memory address 254 (binary 11111110)). The contents of the register can be written to or read 'from toggle switches'. The register contains the address of instruction that is currently executing. When the instruction executed the contents of register are automatically changed (by the number of bytes the instruction occupies), indicating the address of the next instruction. The contents of this register are displayed by the LEDs ● **AD7** – ● **AD0**.

Data Output register DO (memory address 255 (binary 11111111)). As a rule, the register contains the result or a discreet byte of the calculation. The contents of this register are displayed by the ● **DO0** – ● **DO7** LEDs.

INSTRUCTION SET

Instructions can be a single-byte, a 2-byte, a 3-byte, (also there are three 4-byte instructions). Therefore when instruction executed, to indicate the next instruction the instruction pointer IP will increment by 1, 2, 3 or 4, respectively.

Processor Control Instructions
NOP | **00h** | 00000000b | 1 byte No operation (Doesn't perform any operation, 'empty' instruction).
SPEED | **02h** | 00000010b | 2 bytes Set the speed that the CPU steps through each instruction after starting the program. 2nd byte - speed reduction parameter
ADDRIP | **03h** | 00000011b | 2 bytes | TF Add the contents of a specific memory cell to the contents of the instruction pointer IP. Store the result in IP. 2nd byte - memory address of value to add to instruction pointer. If the Instruction Pointer IP overflow occurred, then the Trap Flag TF = 1, otherwise TF = 0.
HLT | **0Eh** | 00001110b | 1 byte Pause executing instructions. The ● **HALT** LED lights up red. Pressing the **ENTER** or the **RUN/STOP** key will continue the execution with the next instruction. Usually used to wait for a value from the keyboard to be entered into the DI register.
STOP | **0Fh** | 00001111b | 1 byte Stops program execution until the **RUN/STOP** key is pressed.
Data Transfer to / from Accumulator
MOVLA | **10h** | 00010000b | 2 bytes | ZF Write a literal value to the Accumulator. 2nd byte - data to write. If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Copy a memory cell contents to the Accumulator.
2nd byte - memory address of value to copy.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Copy the Accumulator contents to a specific memory cell.
2nd byte - memory address to write to.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Copy to Accumulator the contents of a memory location pointed to by the contents of the specified memory cell.
2nd byte - an intermediate location containing an effective address of data.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Copy the contents of Accumulator to a memory location pointed to by the contents of specified memory cell.
2nd byte - an intermediate location containing an effective address of data.

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

Write a literal value to a memory location pointed to by the contents of specified memory cell.
2nd byte - a literal value to write
3d byte - an intermediate location containing an effective address to write the data.

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

Write the contents of Accumulator to the 2nd byte of the instruction itself.
2nd byte - a byte to write to. It can have any initial value that will change during the execution of the instruction.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Write to Accumulator the contents of a memory location pointed to by the contents of the specified memory cell. Increment or decrement by 1 the contents of an intermediate location (an effective address). If CF=0, address increments by 1, if CF=1, address decrements by 1.

2nd byte - an intermediate location containing an effective address of data.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Copy the contents of Accumulator to a specific memory address, then set the Accumulator to zero.
2nd byte - a memory address to write to.

Always sets ZF = 1.

Data Transfer to Memory

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

Write a literal value to specified memory address.
2nd byte - data to write.
3rd byte - memory address to write to.

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

Copy the contents from one memory cell to another.
2nd byte - memory address to read from.
3rd byte - memory address to write to.

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

Copy the contents of one memory cell to another with indirect addressing of both memory cells.
2nd byte - an intermediate location containing an effective address to copy from.
3rd byte - an intermediate location containing an effective address to copy to.

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

Write a zero to a specific memory cell.
2nd byte - a memory address to write to.

Exchange Instructions

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

Swaps the contents of the Accumulator and a specific memory cell.
2nd byte - memory address of the contents to swap.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Swaps the contents between a pair of memory cells.
2nd byte - memory address of contents to swap.
3rd byte - memory address of contents to swap.

Arithmetic and Logical Instructions (Accumulator)

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

Add a literal value to the contents of the Accumulator. Result stored in the Accumulator.
2nd byte - the value to add.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Add a value from specific memory cell to the contents of the Accumulator and store the result in the Accumulator.

2nd byte - memory address to add value from.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Subtract a literal value from the contents of the Accumulator and store the result in the Accumulator.

2nd byte - a value to subtract.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Subtract the contents of a specific memory cell from the Accumulator, store the result in the Accumulator. If result is a negative number, it is represented with a sign bit in the Carry Flag CF.
2nd byte - memory address of value to subtract.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Perform a logical AND of the contents of the Accumulator and a literal value. Store the result in the Accumulator.
2 bytes - a value to 'AND' with Accumulator.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Perform a logical 'AND' of the contents of the Accumulator and the contents of specific memory cell. Store the result in the Accumulator.
2nd byte - memory address of value to 'AND'.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Perform a logical 'OR' of the contents of the Accumulator and a literal value.
2nd byte - a value to 'OR' with the Accumulator.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Perform a logical 'OR' of the contents of the Accumulator and the contents of a specific memory cell.
2nd byte - memory address within value to 'OR'.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Perform a logical 'XOR' of the contents of the Accumulator and a literal value. Store the result in the Accumulator.
2nd byte - a value to 'XOR' with the Accumulator.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Perform a logical 'XOR' the contents of the Accumulator and a value within specific memory cell. Store the result in the Accumulator.
2nd byte - memory address of value to 'XOR'.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Decrement the value in the Accumulator by 1.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Increment the value in the Accumulator by 1.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Decimal correction of the Accumulator after adding binary decimal numbers.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Decimal correction of the Accumulator after subtracting binary decimal numbers.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Inversion of the contents of the Accumulator.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Convert a number in the Accumulator from binary-decimal to binary.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Convert a number in the Accumulator from binary to binary-decimal.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Add a literal value and a value of the CF bit to the contents of Accumulator. Store the result in the Accumulator.
2nd byte - a literal value to add.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Add a value within addressed memory cell and a value of the CF bit to the contents of Accumulator. Store the result in the Accumulator.
2nd byte - memory address to add value from.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is overflow (a value exceeds FFh), Carry Flag CF = 1, otherwise CF = 0.

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

Subtract a literal value and a value of the CF bit from the contents of Accumulator. Store the result in the Accumulator.
2nd byte - a literal value to subtract.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

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

Subtract a value within addressed memory cell and a value of the CF bit from the contents of Accumulator. Store the result in the Accumulator.
2nd byte - a memory address of value to subtract.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If result in the Accumulator is a negative number, the Carry Flag CF = 1, otherwise CF = 0.

Arithmetic and Logical Instructions (Memory)

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

Decrement a value in a specific memory cell by one.
2nd byte - memory address of value to decrement.

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

Increment a value in a specific memory cell by one.
2nd byte - memory address of value to increment.

Accumulator Shift Instructions

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

Shift the contents of the Accumulator left by one. The leftmost bit from the Accumulator goes into Carry Flag CF.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Shift the contents of the Accumulator right by one. The rightmost bit from the Accumulator goes into Carry Flag CF.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Shift the contents of the Accumulator left through the Carry Flag. The leftmost bit from the Accumulator goes into Carry Flag. The previous value from the Carry Flag writes to the low bit.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Shift the contents of the Accumulator right through the Carry Flag. The rightmost bit from the Accumulator goes into Carry Flag. The previous value from the Carry Flag writes to the low bit.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

Memory Shift Instructions

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

Shift the data within memory cell left by one. The leftmost bit from the memory cell is lost.
2nd byte - memory address of data to shift.

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

Shift the data within memory cell right by one. The rightmost bit from the memory cell is lost.
2nd byte - memory address of data to shift.

Bit Manipulation Instructions (Accumulator and Flags)

CBA | **80h** | 10000000b | 2 bytes | ZF

Set a specific bit in the Accumulator to 0.
2nd byte - number of a bit within Accumulator to reset.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

SBA | **81h** | 10000001b | 2 bytes

Set specific bit in the Accumulator to 1.
2nd byte - number of a bit within Accumulator to set.

XCHGAA | **82h** | 10000010b | 1 byte | ZF

Swap the high and low half-bytes within the Accumulator.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

CLRCF | **83h** | 10000011b | 1 byte

Resets the Carry Flag to 0.

CLRTF | **84h** | 10000100b | 1 byte

Resets the Trap Flag TF to 0.

MOVCFA | **86h** | 10000110b | 2 bytes | ZF

Copy the contents of the Carry Flag CF to a specific Accumulator bit.
2nd byte - number of a bit within Accumulator to store a value.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

MOVACF | **87h** | 10000111b | 2 bytes | ZF, CF

Copy the contents of a specific Accumulator bit to the Carry Flag.
2nd byte - number of a bit within Accumulator to copy.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0

Memory Bit Manipulation Instructions

CBR | **90h** | 10010000b | 3 bytes

Set a specific bit within the specified memory cell to 0.
2nd byte - specified bit which to reset.
3rd byte - specified memory address of data.

SBR | **91h** | 10010001b | 3 bytes

Set a specific bit within the specified memory cell to 1.
2nd byte - specified bit which to reset.
3rd byte - specified memory address of data.

MOVCFR | **92h** | 10010010b | 3 bytes

Copy the contents of the Carry Flag to a specific bit within a specified memory cell.
2nd byte - specified bit with which to store the value.
3rd byte - specified memory address of data.

MOVRCF | **93h** | 10010011b | 3 bytes | CF

Copy the contents of a specific bit within a specified memory cell to the Carry Flag CF.
2nd byte - specified bit to copy value from.
3rd byte - specified memory address of data.

Stack Manipulation Instructions

PUSHA | **A0h** | 10100000b | 1 byte | ZF

Write the contents of the Accumulator to the stack
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

PUSHR | **A1h** | 10100001b | 2 bytes

Write the value from a specific memory cell to the stack.
2nd byte - memory address of value to read from.

PUSHL | **A2h** | 10100010b | 2 bytes

Write a literal value to the stack.
2nd byte - the value to write.

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

Move the contents from the stack to Accumulator AC.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Move the contents from the stack to a specific memory cell.
2nd byte - memory address to write to.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Write the contents of the Stack Pointer SP to the Accumulator.
If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Write the contents of the Accumulator to the Stack Pointer SP.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Write a byte to the Stack Pointer SP.
2nd byte - data to write.

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

Write a fixed initial value of 251 (11111011b) to the Stack Pointer SP.

Subroutines Instructions and Unconditional Transfer

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

Initialize the execution of a subroutine until **RETURN** instruction is executed. Then return to the very next memory location after the **CALL** instruction.
2nd byte - subroutine entry address.

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

Return from a subroutine initialized by the **CALL** instruction.

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

Unconditional Transfer. Go to the specific memory address and continue executing instructions from there.
2nd byte - Instruction address to jump to.

Conditional Transfer Instructions

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

Conditional transfer with a counter. Decrement the contents of a specific memory cell by one. If the result is non-zero, go to the address specified in the 3rd byte of the instruction. Otherwise execute the next instruction.
2nd byte - memory address within value to decrement.
3rd byte - transition address, if the result is non-zero.

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

Conditional transfer with a counter. Increment the contents of a specific memory cell by one. If the

result is non-zero, go to the address specified in the 3rd byte of the instruction. Otherwise execute the next instruction.
2nd byte - memory address within value to increment.
3rd byte - transition address, if the result is non-zero.

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

Check a specific bit within a specified address. If it is ZERO, go to the next instruction. If it = 1, go to the address specified in the 4th byte of the instruction.

2nd byte - bit to check.
3rd byte - specified memory address of a bit.
4th byte - transition address if the bit = 1.

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

Check a specific bit within a specified address. If it is ONE, go to the next instruction. If it = 0, go to the address specified in the 4th byte of the instruction.

2nd byte - bit to check.
3rd byte - specified memory address.
4th byte - transition address, if the bit = 0.

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

Check the Zero Flag. If ZF = 1, go to a specified transition address. If ZF = 0, execute the next instruction.
2nd byte - transition address, if ZF = 1.

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

Check the Zero Flag. If ZF = 0, go to the specified transition address. If ZF = 1, execute the next instruction.
2nd byte - transition address, if the flag ZF = 0.

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

Check the Carry Flag. If CF = 1, go to a specified transition address. If CF = 0, execute the next instruction.
2nd byte - transition address, if the flag CF = 1.

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

Check the Carry Flag. If CF = 0, go to a specified transition address. If CF = 1, execute the next instruction.
2nd byte - transition address, if the flag CF = 0.

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

Check the Trap Flag. If TF = 1, go to a specified transition address. If TF = 0, execute the next instruction.
2nd byte - transition address if the TF flag = 1.

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

Check the Trap Flag. If TF = 0, go to a specified transition address. If TF = 1, execute the next instruction.
2nd byte - transition address if the TF = 0.

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

Compare the contents of Accumulator with the contents of addressed memory cell. If a value within Accumulator is less than a value within addressed memory cell, then go to the address specified in the 3rd byte of the instruction.
2nd byte - a memory address of value to compare.
3nd byte - a memory address to jump to.

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

Compare the contents of Accumulator with a literal value. If a value within Accumulator is less than a literal value, then go to the address specified in the 3rd byte of the instruction.

2nd byte - a literal value to compare.
3nd byte - a memory address to jump to.

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

Compare the contents of Accumulator with the contents of addressed memory cell. If a value within Accumulator is equal to a value within addressed memory cell, then go to the address specified in the 3rd byte of the instruction.

2nd byte - a memory address of value to compare.
3nd byte - a memory address to jump to.

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

Compare the contents of Accumulator with a literal value. If a value within Accumulator is equal to a literal value, then go to the address specified in the 3rd byte of the instruction.

2nd byte - a memory address of value to compare.
3nd byte - a memory address to jump to.

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

Compare the contents of Accumulator with the contents of addressed memory cell. If a value within Accumulator is greater than a value within addressed memory cell, then go to the address specified in the 3rd byte of the instruction.

2nd byte - a memory address of value to compare.
3nd byte - a memory address to jump to.

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

Compare the contents of Accumulator with a literal value. If a value within Accumulator is greater than a literal value, then go to the address specified in the 3rd byte of the instruction.

2nd byte - a literal value to compare.
3nd byte - a memory address to jump to.

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

Compare the contents of one memory cell with the contents of another memory cell. If a value within the first memory cell is less than a value within the second one, then go to the address specified in the 4th byte of the instruction.

2nd byte - a memory address of the first value to compare.
3nd byte - a memory address of the second value to compare.
4th byte - a memory address to jump to.

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

Compare the contents of one memory cell with the contents of another memory cell. If a value within the first memory cell is equal to a value within the second one, then go to the address specified in the 4th byte of the instruction.

2nd byte - a memory address of the first value to compare.

3nd byte - a memory address of the second value to compare.
4th byte - a memory address to jump to.

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

Compare the contents of one memory cell with the contents of another memory cell. If a value within the first memory cell is equal or greater than a value within the second one, then go to the address specified in the 4th byte of the instruction.

2nd byte - a memory address of the first value to compare.
3nd byte - a memory address of the second value to compare.
4th byte - a memory address to jump to.

Input / Output Instructions

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

Write a value from the Accumulator to the Data Output register (memory cell 255).

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

Write a value from the Data Input register DI (memory cell 253) to the Accumulator.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Write a code of the last key pressed to the Accumulator. Pauses program execution, the ● **HALT** LED lights up yellow until any key is pressed. Stores key code in the Accumulator.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Write the contents of the Accumulator to the key color management port. The lower 6 bits determine the address of the key, the higher 2 bits determine the key color.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Turn off the backlight of all keys (numbers and alphabet), except for the functional keys.

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

Return the key color to the Accumulator. An address of the key is placed in the lower 6 bits of Accumulator before calling the instruction. After the instruction is executed a color code is returned to the higher 2 bits of the Accumulator.

Miscellaneous Instructions (Chain Operations and Multiplication)

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

Copy the specified number of bytes from the array of memory cells starting from the "source address" to the array of memory cells starting from the "destination address".

2nd byte - the number of cells to copy.
3rd byte - the first memory address in the array to read from.
4th byte - the first memory address in the array to write to.
If a resulting value of a memory address goes beyond the address space, then the Trap Flag TF = 1, otherwise TF = 0.

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

Multiply a value from the Accumulator by a value within a specific memory cell. The third byte specified is a memory address within the low byte of the result stored. The high byte of the result will be stored within the address greater by 1.
2nd byte - memory address of a second factor.
3rd byte - memory address of the low byte of the result to store to.

If result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.

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

Divide a value within a specified memory cell by a value in the Accumulator.
2nd byte - memory address of the low byte of the 16-bit dividend. The high byte location follows.
3rd byte - memory address of the low byte of the result.

The high byte of the result is stored within address larger by 1. The fractional part (a division remainder) will be stored within address larger by 2.

If overflow occurred, the 4th bit of the Flag Register is set to 1. Otherwise = 0.
If a result in the Accumulator is 00h, Zero Flag ZF = 1, otherwise ZF = 0.
If a result is overflow (a value exceeds FFFFh), Carry Flag CF = 1, otherwise CF = 0.

Instead of **DIVRA** it is recommended to use the **MULRA** instruction to multiply by the number inverse of the divisor.

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

It is assumed that this instruction is always followed by a two-byte JMP instruction. Memorize the so-called "return address." The instruction "finds out" the location of its own first byte, and adds 4 to it.

2nd byte - a memory address to write a return address to.
If a return address is greater than FFh, then the Trap Flag TF = 1, otherwise TF = 0.

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

Execute one instruction at the address specified in the 2nd byte and then continue the program.
2nd byte - a memory address of the first byte of the instruction to be called.

Note! When calling a conditional or unconditional jump instructions, as well as itself, the correct operation of these instructions is not guaranteed.