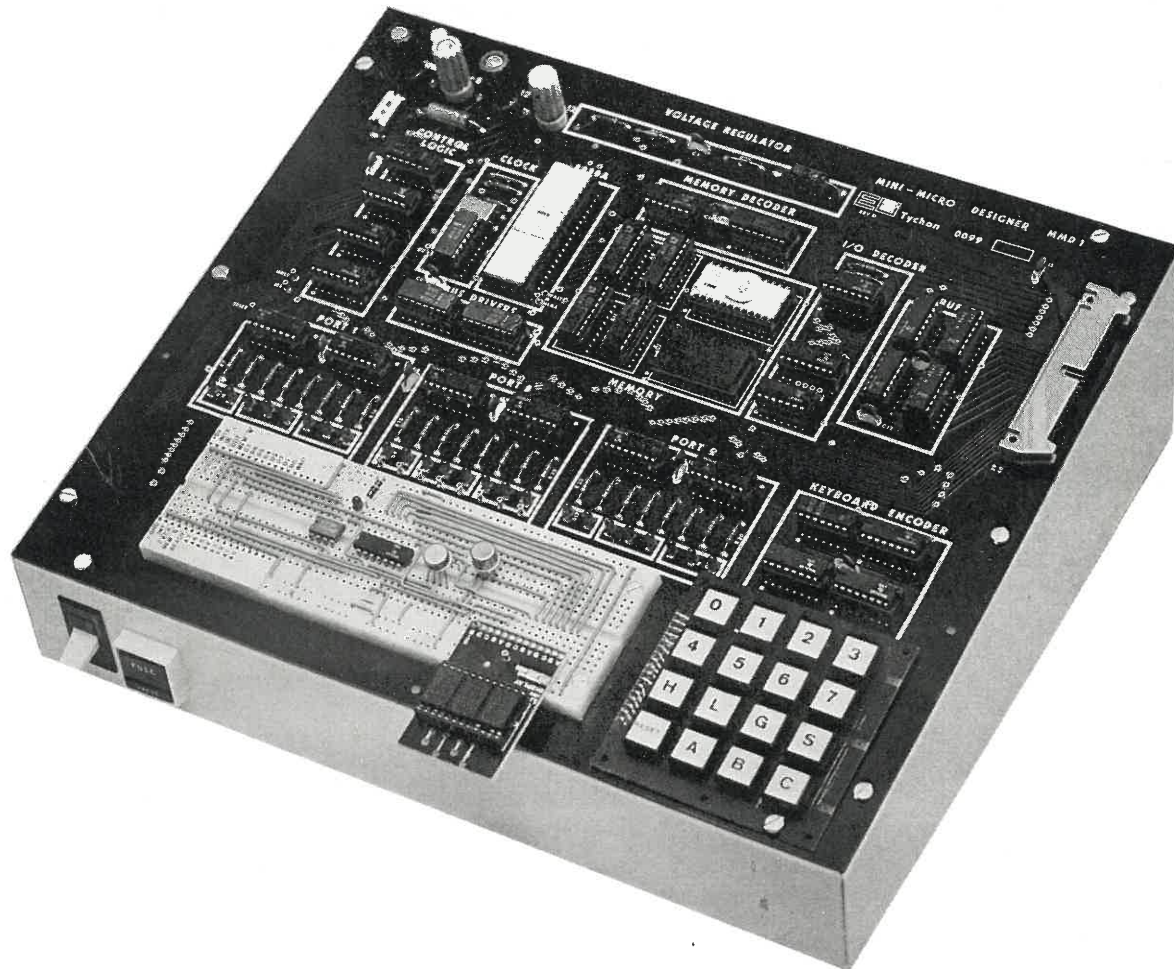


MMD-1 MINI-MICRO DESIGNER

EDUCATION AND DEVELOPMENT MICROCOMPUTER



- COMPLETE 8080A MICROPROCESSOR-BASED COMPUTER SUITABLE FOR EDUCATION AS WELL AS SOFTWARE AND SYSTEM DEVELOPMENT
- SIMPLE ENOUGH FOR SELF-INSTRUCTION AND YET COMPLETE ENOUGH FOR THE PROFESSIONAL SYSTEMS DESIGNER AND CLASSROOM TEACHING
- BASIC SYSTEM CONTAINS COMPLETE 8080A MPU SET INCLUDING MEMORIES WITH ADD-ON MEMORY AVAILABLE
- DIRECT KEYBOARD ENTRY OF DATA AND INSTRUCTIONS (NO DATA ENTRY TERMINAL NECESSARY)
- LIGHT EMITTING DIODE STATUS AND DATA INDICATORS
- INTEGRAL SOLDERLESS BREADBOARDING SOCKET WITH DIRECT BUFFERED ACCESS TO THE MICROPROCESSOR
- COMPLETE SELF CONTAINED POWER SUPPLY
- COMPLETE TUTORIAL DOCUMENTATION AND OPERATING MANUALS
- PROVISION FOR DIRECT TELEPRINTER OR CRT TERMINAL AND AUDIO CASSETTE INTERFACES



E & L INSTRUMENTS, INCORPORATED

MINI - MICRO DESIGNER

8080A BASED COMPUTER FOR TRAINING AND HARDWARE/ SOFTWARE DEVELOPMENT

E & L Instruments has introduced the first educational and engineering microcomputer that is simple and versatile enough to be used as a basic training system and yet has all of the features required for circuit design, interfacing experimentation and software development for the professional designer.

The Mini-Micro Designer (MMD-1) is a low cost, expandable system totally supported by the unique Bugbook educational materials. Well adapted for use as classroom texts or for self-instruction, the Bugbook concept is the ideal introduction to computers for beginners as well as those with considerable technical background and experience. The texts, which double as laboratory manuals, start with digital coding and microcomputer programming to involve the student rapidly in the practical aspects of microcomputer programming and interfacing.

DIRECT KEYBOARD ENTRY AND BREADBOARDING

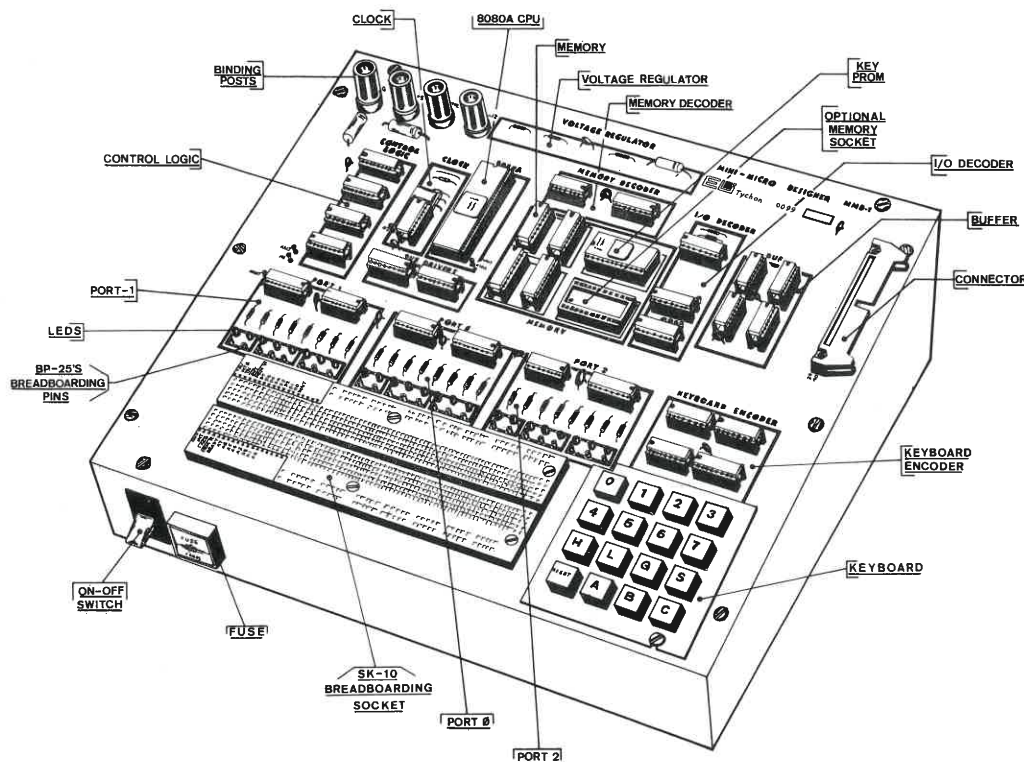
The MMD-1 features direct keyboard entry of data and instructions in easy to understand octal code. Immediate access to the 8080A central processor may be obtained with buffered input/output busses and the convenient breadboarding socket.

The breadboarding capability built directly into the MMD-1 system permits the student or experimenter to carry out over 60 experiments presented in the Bugbook text as well as many other circuit designs without the need for soldering or the construction of simple circuit functions.

Keyboard entry eliminates the need for a teleprinter or other costly data terminals. The ease of access to the busses of the MMD-1 make it an excellent software development tool. Additional modules are available to expand the memory of the MMD-1, permit interfacing with teleprinters, cathode ray tube terminals and an audio cassette recorder.

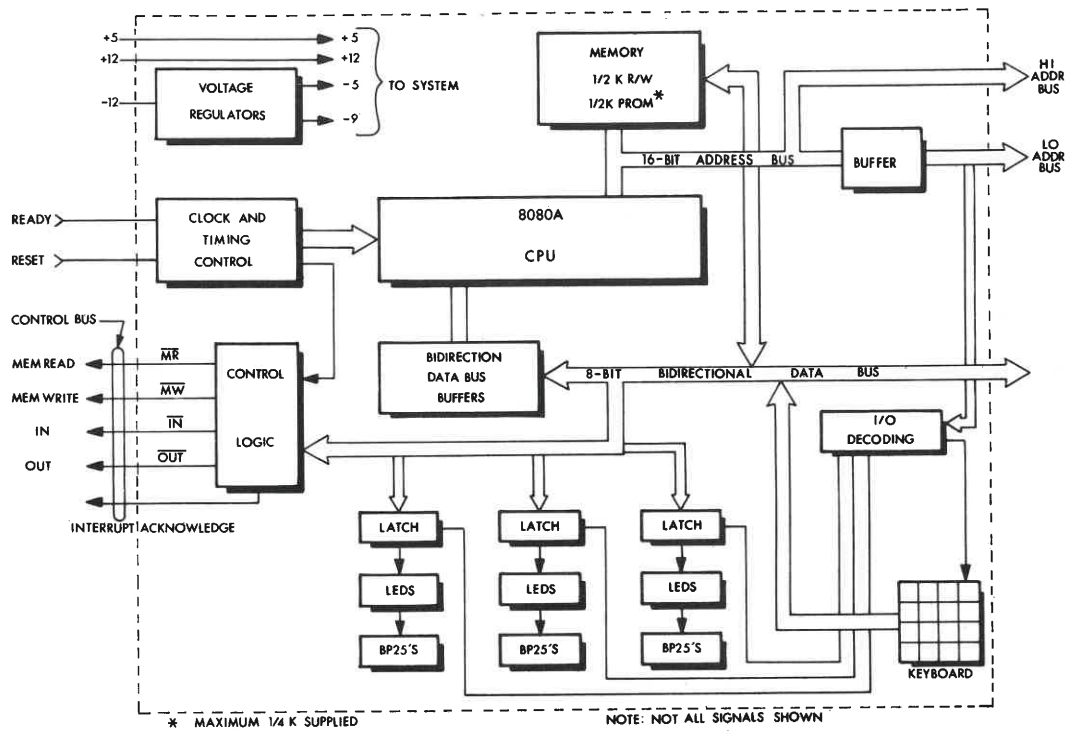
The MMD-1 is the only microcomputer system now on the market that permits the user to design his own interface in 10 to 20 minutes and implement the design with ordinary hookup wire and standard readily available 7400 series ICs. No special assembly tools are needed.

One of the unique features of the Mini-Micro Designer is the use of E & L's, standard, BP-25 Breadboarding Pins. These pins are directly linked to the three ports (LO, HI and DATA). They allow the user to output three distinct 8-bit bytes of latched data to peripheral circuitry.



CONTROLS, SOCKETS, CONNECTORS, AND CIRCUIT FUNCTIONS

FIGURE -1



BASIC MINI-MICRO DESIGNER (MMD-1)
FIGURE 2

BENCH-TOP CONVENIENCE

The essential features of the MMD-1 are shown in out-line drawing, (Figure 1). The complete basic unit is self contained within a slope-front case that measures 12" (30,5cm) long by 10" (25,4cm) wide by 3 1/2" (8,9cm) high.

The 8080A microprocessor and its associated circuits are located on the top center part of the printed circuit panel. The related chip set includes clock and timing control, two bidirectional data bus buffers, four static MOS random-access memory (read/write) chips for a total of 512 8 bit words and one 256 by 8 bit UV erasable PROM, pre-programmed to permit keyboard entry.

The keyboard permits direct entry of data and instructions. Immediate access to the 8080A is available with buffered input/output (I/O) busses and a convenient breadboarding socket.

Part of the socket is wired to the I/O bus and other critical control lines. Status and data are indicated with an array of light emitting diode (LED) lamps to permit monitoring the system.

The socket has been designed to accept either standard integrated circuits, and discrete components or pre-assembled functional plug-in modules available from E & L Instruments. The modules, called Outboards, are pre-designed, frequently used circuits that simplify interfacing and reduce the time required, for either the student or the professional designer, to breadboard the desired circuit. These Outboards are available either assembled or in kit form. (See General Catalogue for full line)

The panel mounted 40 pin connector makes the full computer bus structure available for external use.

The breadboarding socket will accommodate up to five 16-pin IC's and a wide range of discrete components and outboards. All power supply voltages are available at the binding posts on the panel. The +5Vdc and ground are also available on the SK-10 Breadboarding Socket.

The MMD-1 is available in three forms:

- *MMD-1/A: Factory assembled and ready for use, comes with operating manual and complete self teaching microcomputer training course (Bugbooks).
- *MMD-1/K: Kit form, requiring assembly and soldering of components to form a complete unit. Kit comes with operating manual and complete self-teaching microcomputer training course (Bugbooks).

MMD-1CBK: Circuit Board Kit that includes main and power supply P.C. boards, a premounted SK-10 interfaced breadboarding socket, the keyboard, operating manual, and complete self-teaching microcomputer training course (Bugbooks). Non-standard components necessary to build the unit are listed, with manufacturer's name and part number, in the Parts List of the operating manual.

Also available, for the CBK users, is the MMD-1/IC. This is the computer chip set (See Back Page).

* The Assembled or Kit Form may be purchased in either 115V dc or 230V dc operation.



INTEGRATED EDUCATIONAL SYSTEM

The Mini Micro Designer is designed to be an integral part of a complete educational system for the beginner as well as the experienced digital circuit designer. While the educational texts supplied with the basic MMD-1 emphasize the role of the microcomputer in machine and process control applications, it will nevertheless provide an excellent background for the understanding of all computer and data processing systems.

COMPLETE OPERATING MANUAL

Each MMD-1, complete or in kit form is supplied with a 34-page operating manual. This manual contains a general description, specifications and parts list as well as schematics. It also contains instructions for constructing and testing the main board as well as the power supply.

PROVEN MODULAR EDUCATIONAL TEXTS

The educational texts furnished with the MMD-1 are prepared for use by all students, regardless of educational background, for self-instruction or in a classroom situation. The text is completely coordinated with the MMD-1 hardware and it permits the student to move at his own pace without supplementary text books or references.

The educational approach employed is that of the famed E & L Instruments Bugbooks; paperback texts that emphasize the practical aspects of digital circuitry and integrated electronics. Unlike other texts, the student is not first required to master or review analog electronics before being introduced to digital electronics and the course material in non-mathematical.

Also, unlike other so-called educational texts, with the E & L Bugbooks, the student is not required to study material prepared by semiconductor manufacturers with experienced digital circuit designers in mind.

The standard text furnished with the MMD-1 can serve as home-study or classroom laboratory manuals. The series is collectively entitled, Introductory Experiments in Digital Electronics, and 8080A Microcomputer Programming and 8080A Microcomputer Interfacing.

The subjects of digital electronics, microcomputer interfacing and microcomputer programming are unified in a single course divided into chapters or units. The units begin with Digital Codes and An Introduction to Microcomputer Programming and progress through to the 8080A Instruction Set.

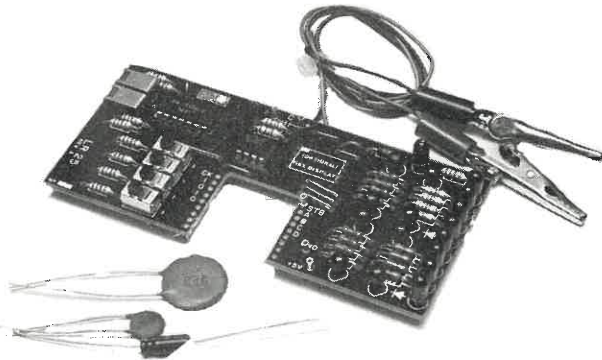
Among the topics covered are:

1. Digital Codes
2. An Introduction to Microcomputer Programming
3. Some 8080A Microcomputer Instructions
4. The MMD-1 Microcomputer
5. Some Simple 8080A Microcomputer Programs
6. Registers and Register Instructions
7. Logic Gates and Truth Tables
8. Logical Instructions
9. An introduction to Breadboarding
10. Integrated Circuit Chips
11. Flip-Flops and Latches
12. Decoders
13. Counters
14. Gating Digital Signals
15. Astable and Monostable Multivibrators
16. Device Select Pulses
17. The 8080A Instruction Set

VERSATILITY TO YOUR — DESIGNER

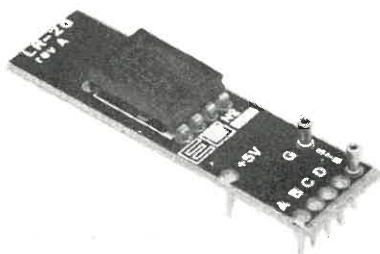
LR-25 BREADBOARDING STATION OUTBOARD

The LR-25 is an Outboard that provides the most commonly used digital circuit functions to aid in the breadboarding of interfaces and other circuits. Included on the Outboard are: a variable frequency clock, pulsers (bounce free switches), eight light-emitting diode (LED) indicators for data display, slide switches for programming or status changes and an optional hexadecimal display.



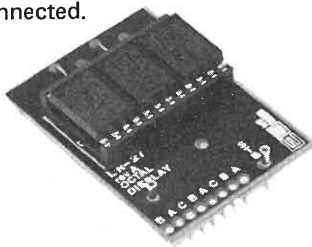
LR-26 DISPLAY OUTBOARD

The LR-26 uses a single hexadecimal display with built-in decoder, driver and latch. Decodes 0-9 and A-F.



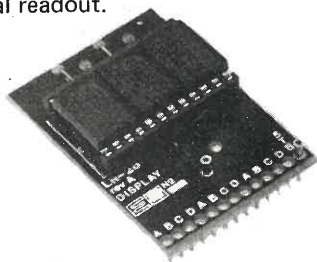
LR-27 OCTAL LATCH OUTBOARD

The LR-27 uses three hexadecimal displays. It is designed to directly accept hexadecimal code. The "D" pins are grounded together to provide Octal readout and all strobe inputs are interconnected.



LR-28 THREE DIGIT LATCH OUTBOARD

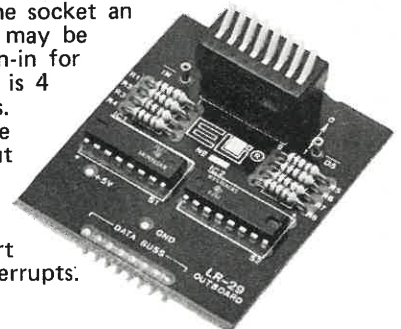
The LR-28 like the LR-27, uses three hexadecimal displays. Each display is independent of the other except that the strobe inputs are interconnected. This outboard provides a hexadecimal readout.



ALL OUTBOARDS, SHOWN HERE, MAY BE PURCHASED EITHER ASSEMBLED OR IN KIT FORM

LR-29 GENERAL INPUT PORT OUTBOARD

The LR-29 uses two tri-state buffer circuits to provide an 8-bit gated input buffer function. The buffer is enabled when both the IN and DS inputs are low (logic 0). The inputs to the gated buffer may be derived onboard by using the DIP switch to generate logic levels or the inputs may come from an external source. By removing the DIP switch from the socket an external logic level may be substituted. The fan-in for the external inputs is 4 standard TTL loads. Applications for the LR-29 include input of general data, status or sense switches and the generation of restart instructions for interrupts.



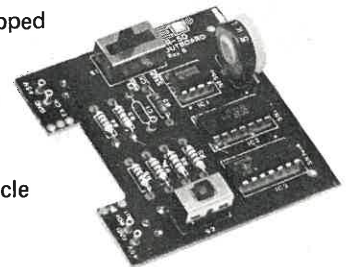
LR-50 SINGLE STEP OUTBOARD

The LR-50 is used to "single step" the Mini-Micro Designer. All of the electronics necessary to permit the user to step the computer one machine cycle at a time is contained on a single circuit board.

It may be used to monitor the data bus, address bus and status latch during each machine cycle. Input and output pulses, memory data, device addresses, as well as other signals may be observed by lamp monitors rather than expensive oscilloscopes or logic analyzers.

The LR-50 operates in three modes: Run, Clock and Step. These modes are selected by a three position slide switch. In the Run mode the MMD-1 will operate at its full clock speed; this is the normal operating mode. In the clock mode the computer is stepped automatically by an on board 555 timer contained on the module. The frequency of the timer and the computer execution speed may be easily varied. The EX CK (external clock) input mode is used when the computer is to be synchronized with an external clock.

The computer may be stepped manually when the three-position switch is in the Step mode. A second switch is provided on the Outboard in order to allow the user to step the computer one machine-cycle at a time.



KEX L/D PROM

An advanced version of KEX that performs data entry and assigns start address. Contains cassette load and dump program and linkage to Monitor Prom. An exchange policy with the standard KEX Prom supplied is available for a nominal reprogramming charge.

MONITOR PROM

A NEW pre-programmed PROM, for your MMD-1, that allows single instruction execution and examination of all registers including status flags. Requires KEX L/D PROM for operation and plugs into PROM location number 1 on MMD-1 panel.

COMING SOON

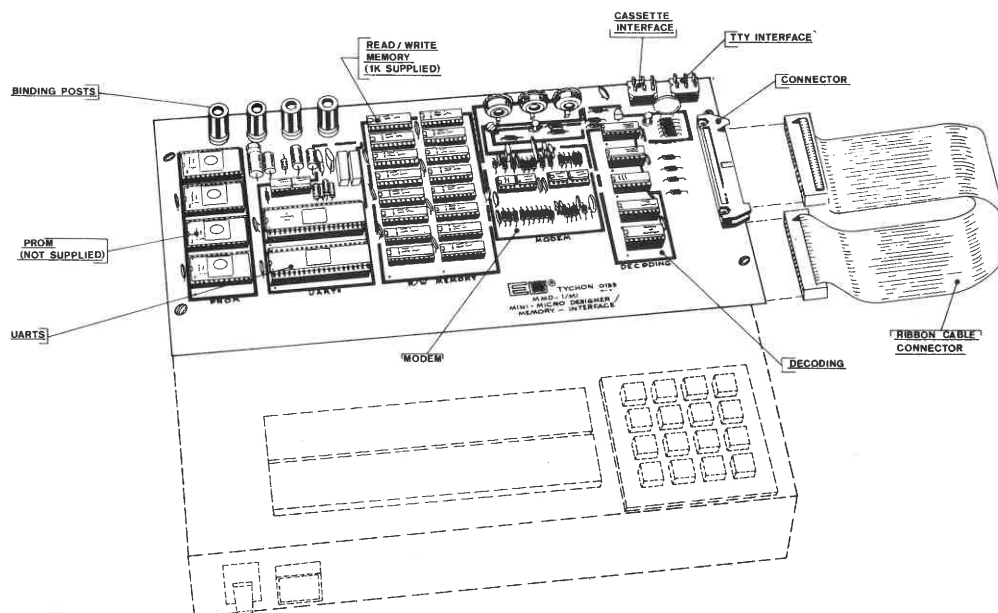
PROM PROGRAMMER

An inexpensive 1702A PROM PROGRAMMER, designed for use with your MMD-1, or other 8080A based systems. Available after Apr. 1, '77 from your local computer store.

MMD - 1/MI

MEMORY-INTERFACE BOARD

FOR AUDIO RECORDER AND TELEPRINTER



CONTROLS, CONNECTORS, AND CIRCUIT FUNCTIONS
FIGURE-3

The Memory and Interface Board (M/I) is an accessory designed to permit hardware and software development to be performed on the MMD-1 microcomputer system. The M/I board may be purchased factory assembled or in kit form.

The M/I board provides additional memory and teleprinter and audio cassette recorder interface. The use of an Audio Cassette Recorder allows for low cost storage of data in the course of software development.

The M/I board, illustrated in figure 3, is designed for simple assembly to the master board of the MMD-1 by means of screws and threaded standoffs. Electrical connections from the M/I board to the MMD-1 are made by means of ribbon cable and a 40-pin connector.

The M/I board, shown schematically in figure 4 provides up to 1024 words of programmable read-only memory (PROM) (none supplied) and 2048 words of read/write random access memory (RAM). (1024 words supplied)

The M/I board uses two universal asynchronous receiver transmitters (UART) for synchronization, formatting and parallel series conversions necessary for communications with teleprinters. The UARTS handle data in blocks of 8-bits. The American National Standard Code for Information Interchange (ASCII) is employed in the 8-bit data block.

The M/I board contains a relay driver circuit for use with a teleprinter paper tape reader relay. This option permits software control of all reader start/stop operations.

No restrictions are placed on the format for communications with the audio recorder as the recorder need not be directly interfaced with any other terminal for printing or display of readout.

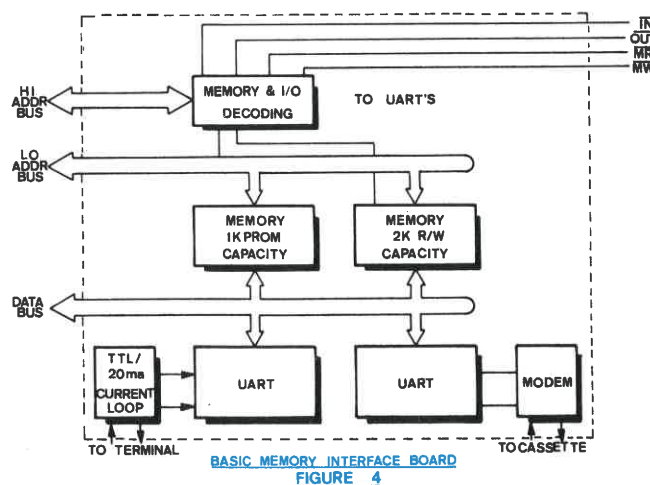
FIRMWARE ACCESSORIES

LOAD AND DUMP PROM (MMD-1 L/D PROM)

A single 256 word by 8-bit PROM contains the necessary program to load and dump data to and from an audio recorder. The PROM interacts with the MMD-1 keyboard and requires that the operator spend only about 2 seconds to set up the microcomputer for loading and dumping tapes.

D-BUG PROM SET (MMD-DEBUG PROM)

This set of four PROMs contains a convenient octal oriented program for debugging. It permits the operator to create and modify programs, establish start address, set built-in breakpoints and continue function, load and dump punched paper tapes, and interact with ASCII-coded teleprinter or cathode ray tube terminals.



BASIC MEMORY INTERFACE BOARD
FIGURE 4

SPECIFICATIONS

MMD-1

- **CENTRAL PROCESSOR**
8080A or equivalent
- **MEMORY**
Read/Write, Random Access (RAM); 512 words by 8 bits furnished on main MMD-1 circuit board.

Expandable up to a total of 65,536 words. (Additional power required over 2500 words.)

Programmable Read Only (PROM) programmed to permit keyboard entry of data in octal code; keyboard executive program (KEX) 256 words by 8 bits.

(Open socket on board permits an additional 256 words.)
- **DISPLAY**
Light-emitting diode lamps (LEDs); three groups of eight individually latched and addressable under software control.

LEDs display low address, high address and memory contents.
- **DATA ENTRY**
Keyboard; 16-switch with keys organized in octal code 0 to 7, Hi address (H), Lo address (L), go (G), reset (RESET), examine/deposit (S) and three optional keys (A, B, and C).
- **INTERFACE SOCKET**
E & L SK-10/IF 18. Direct interface connection includes $D_0 - D_7$; $A_0 - A_7$; In, Out, MEMW, +5, GND MEMR, INT, INTE, WAIT, READY, TACK.

Open area on socket will take up to five 16-pin ICs and discrete components with 20 to 26 AWG wire leads.
- **RIBBON CABLE CONNECTOR**
Dual, 20-pin wired the same as the SK-10 interface socket with A8 thru A15 added.
- **INTERNAL POWER SUPPLY**
Line Voltage - 115V or 230V ac, fused
- **OUTPUTS**
+ 5V dc @ 1.5A
+12V dc @ 150mA
- 12V dc @ 150mA
- **MICROPROCESSOR CHIP SET**

1	8224	Clock Generator and Driver
4	8111-2	1024-bit (256 by 4) static MOS RAM
2	8216	4-bit parallel bidirectional bus drivers.
1	1702	2048-bit (256 by 8) programmable read only memory (PROM) (Factory programmed as KEX PROM)
- **DIMENSIONS**
12" (30,5cm) by 10" (25,4cm) by 3½" (8.9cm)
- **WEIGHT**
6.3 pounds (2,86kg.)

MMD-1/MI

- **POWER CONSUMPTION** (with maximum memory installed)
+ 5V dc @ 1A
+12V dc @ 0.1A
- 12V dc @ 0.25A
- **MEMORY**
Read/Write (RAM) 2048 words capacity. (1024 words supplied)

Programmable ROM (PROM) 1048 word capacity (non supplied)
- **TELEPRINTER INTERFACE**
Full Duplex, 20 mA current loop
110 Baud
Reader Relay Control
- **AUDIO RECORDER INTERFACE**
Software controlled
Two-tone audio 2125 Hz/2975 Hz (FSK)

UART formatted
300 Baud Data Rate

Output Impedance: 47K ohms

Output Signal: 1 volt ac (2V P-to-P)

Input Impedance: 10K ohms

Input Signal: 1 volt ac (2V P-to-P)
- **RIBBON CABLE CONNECTOR**
Dual, 20-pin wired to full computer bus. (Interconnecting cable supplied - 6" long)

MMD-1/SS

In addition to the MMD-1 system, you might find it useful to purchase the MMD-1/SS. This package of electronic components and breadboarding equipment can be used when working with the Bugbook Series. The MMD-1/SS includes an LR-25 Universal Breadboarding Station, an SK-10 Breadboarding Socket, a 28 pin dual interconnect cable, to bring the computer bus to the blank socket, I.C.'s resistors, capacitors and 22 gage wire. The package will make it possible to do the basic experiments in the Bugbooks. You will find it necessary to purchase other components to do all the experiments in the books.



THE STUDENT STATION INCLUDES THE FOLLOWING:

One	-	28 Pin Dual Interconnect Cable (not shown)			
One	-	LR-25 Universal Breadboarding Socket			
One	-	Miscellaneous Wire Package			
One	-	Components package (resistors and capacitors)			
One	-	P8212 In/Out Port Chip			
Two	-	SN7400	One	-	SN7430
Two	-	SN7402	One	-	SN7432
Two	-	SN7404	One	-	SN74L42
One	-	SN74L04	Two	-	SN7474
Two	-	SN7408	Two	-	SN7475
One	-	SN7410	One	-	SN7476
One	-	SN7486	One	-	SN74154
One	-	SN7489	One	-	SN74LS155
Four	-	SN7490	One	-	SN74174
Two	-	SN7493	One	-	SN74193
One	-	SN74123	Two	-	DM8095
One	-	SN74148	One	-	NE555

ITEM DESCRIPTION AND LOCATION

E & L P/N	CAT. DESIG.	DESCRIPTION	PAGE
341-0050	MMD-1/A	Mini-Micro Designer - Factory Assembled Form	2 & 3
341-1200	MMD-1/K	Mini-Micro Designer - Kit Form	2 & 3
341-1000	MMD-1/CBK	Mini-Micro Designer - P.C. Boards, Socket & Keyboard	2 & 3
341-1300	MMD-1/IC	Mini-Micro Designer - Computer Chip Set	
341-2000	MMD-1/256 BIT PROM	Blank Prom	
341-3100	MMD-1/MI-A	Memory-Interface Board - Factory Assembled Form	6
341-3000	MMD-1/MI-K	Memory-Interface Board - Kit Form	6
341-3350	MMD-1/MI-RAM	Additional 1K of RAM for the M/I Board	
341-3400	MMD-1/MI-PROM	Blank PROM - includes I.C. Socket and Filter Cap	
341-3450	MMD-1/MI - L/D PROM	PROM - Preprogrammed for Loading and Dumping	6
335-1025	LR-25/A	Universal Breadboarding Station - Assembled Form	5
335-2025	LR-25/K	Universal Breadboarding Station - Kit Form	5
335-1026	LR-26/A	Display Outboard - Assembled Form	5
335-2026	LR-26/K	Display Outboard - Kit Form	5
335-1027	LR-27/A	Octal Latch Outboard - Assembled Form	5
335-2027	LR-27/K	Octal Latch Outboard - Kit Form	5
335-1028	LR-28/A	Three Digit Latch Outboard - Assembled Form	5
335-2028	LR-28/K	Three Digit Latch Outboard - Kit Form	5
335-1029	LR-29/A	General Input Port Outboard - Assembled Form	5
335-2029	LR-29/K	General Input Port Outboard - Kit Form	5
335-1050	LR-50/A	Single Step Outboard - Assembled Form	5
335-2050	LR-50/K	Single Step Outboard - Kit Form	5
341-4000	KEX L/D PROM	Advanced Version of the Standard KEX PROM	
341-4001	MONITOR PROM	Monitor PROM - Allows Single Instruction Execution	5
342-0500	PROM PROGRAMMER	1702A PROM PROGRAMMER - AVAILABLE APRIL 1, 1977	5
345-1014	IS-SW8	Complete Self-Teaching Micro-Computer Training Course	4
341-6530	MMD-DEBUG PROM	DEBUG PROM - Set of Four Pre-Programmed I.C.'s	6

ALL ITEMS
AVAILABLE
AT YOUR
LOCAL
COMPUTER
STORE

0144.... 1/77

E & L® E & L INSTRUMENTS, INCORPORATED
61 FIRST ST. — DERBY, CT. 06418

ALL SPECIFICATIONS ARE SUBJECT TO
CHANGE WITHOUT NOTICE.

Operation-How to Use the Mini-Micro

The Keyboard Executive software is the 'heart' of the system. It allows you to examine data or program steps and it allows you to change data or program steps stored in the R/W portion of the memory. We can also specify any address and start the program there. The keyswitches are labeled 0-7 for octal data, H for HI Address, L for LO Address, G for Go and S for See and Store. Three keys are not used by the KEX, A, B and C. The Reset key, R, will always reset the computer and restart the KEX if its PROM is in place. All manual data entry is through the keyboard in the basic Mini-Micro system and there are no binary toggle switches to worry about.

Whenever you want to start the system, depress R. This will reset the KEX and it will always output the first address in the R/W block of memory in section 3. This is HI=003 and LO=000. If you will only be using one block of R/W memory to get started, it MUST be in block 3. The KEX will not function without R/W memory in section 3, (see table 1).

To enter data, whether it will be used for new data or addresses, simply depress the numbered keys as you would on a calculator. Data will be entered into the three least significant (right-most) LEDs and it will shift to the left as more data is entered. If a mistake is made, simply reenter the data. Mistakes are shifted out the left side and lost. The Data Register LEDs will display the data just entered from the keyboard and this may be used as HI Address data by depressing H, or it may be used as LO Address data by depressing L. These keys will transfer the data to the proper LED display register and it will be used by the 8080A to address a new memory location.

Whenever a new H or L is specified by actuating one of these two keys, the KEX will always output the contents of the specified location on the Data Register LEDs. To examine the next location, depress the S key for its See function. By depressing S again and again, we can examine data in sequential memory locations. It should be noted that this See function follows increasing memory locations, not the sequential flow of a program.

To change the data in a R/W memory location, simply load the address using the data input key and the H and L keys. The old data presently in the location will immediately appear on the Data Register LEDs. Enter the new data into the Data register using the numeric keys and then enter it into the R/W location by depressing the S key for the Store function. After S is actuated, the new data is stored and the address is automatically incremented by one to go to the next location. The data from the next location is now displayed on the Data Register LEDs. The S key has two functions, both See and Store. How can we tell the difference? If the data has changed we will store it and see the next location. If the data hasn't changed, we will store it back where it came from and then see the next location. When we store old data back to the location from which we got it, we can't really see any change, but this is exactly what the KEX does. It inputs data from the specified location, allows us to make changes and then puts it back. If no changes are made, the old data is restored to its memory location.

Once a program is entered into the computer through the keyboard, we can start it by loading our starting address and actuating the G or Go key. This will transfer control from the KEX software to the software that we want to try. Starting addresses are loaded in the same way that we use them to examine and alter locations. Starting addresses don't have to be in R/W memory, but can just as easily be in PROM.

If your program starts at the first location in R/W memory, 003 000, you can simply depress R followed by G. We can do this because KEX always resets the address back to this first R/W location.

Keys labeled A, B and C are not used by KEX. It should be remembered that the three LED output ports and the keyboard input port are not hardwired for KEX alone. They are available for you to use in your programs. All fifteen keys may be used in any way you like, using software.

Keyboard Executive Software

The Keyboard Executive software is in a single 1702A type PROM which goes in PROM socket 0. This contains all the necessary software to operate the keyboard and the LED displays. This is our software controlled 'front panel', since the keys and LEDs perform functions determined by the KEX software.

How KEX Operates

Whenever the R key is depressed, the 8080A CPU chip will start to execute the program that starts at location 0. Looking at the software listing for KEX you will see that immediately after starting at location 0, the software instructions cause the computer to jump to location HI=000, LO=070 (HI=000 throughout KEX) where we START the program by pointing to the first R/W memory address in our minimum system, 003 000. The address and the data in that location are displayed on the three output ports. This is done between POINTA and POINTC. The software between POINTC and POINTD will do the necessary tasks to input new data from the keyboard and shift the data onto the LEDs. The shifting is done inside the 8080A with software instructions. Doing this in hardware would take many more chips, but it takes relatively few software steps. The software routines at POINTD, POINTE, POINTF and POINTG make up what is called a command decoder. The software decodes the key codes into real actions. Depressing H or L causes the data temporarily stored in the 8080A as numeric key inputs to be output to either the HI or LO set of LEDs. The S key code causes the current or new data to be put back into the memory location that we are looking at. Depressing G causes the computer to use the HI and LO address as the starting point for a new program.

The TIMOUT and KBRD software subroutines have specific tasks. TIMOUT will count its way through various loops for about 10 milliseconds, while the KBRD subroutine will input code from the keyboard. The KBRD subroutine has some unique features which illustrate an interesting hardware-software tradeoff. The keyswitches used in the Mini-Micro are not bounce free, so that when contacts are opened or closed, they can often re-make or re-break the contacts. This can be confusing to the computer since it can't distinguish between a real switch closure and a bounce. We don't want the computer to sense each bounce as a key closure so we would like some way to filter them out. Additional circuitry including latches, clocks and monostables could do this for us, but it complicates the system. We can also do the debouncing in software. The KBRD subroutine will recognize any key closure, but it will only input the key codes after being sure that the key is closed and not bouncing. It does this by waiting after sensing closure and then rechecking the switch to be sure it is still closed. It also checks when we release a key to be sure that it has stopped bouncing before it tries to sense another key being depressed by the user. We have traded some additional software steps for a great deal of hardware. Since we had plenty of PROM left, it was easy to include.

KEX PROGRAM LISTING (OCTAL)

ADDRESS		DATA	
HI	LOW	B7...B0	
			*000 000
000	000	303	JMP
000	001	070	START
000	002	000	0
/ JUMP UP 10 R/W MEMORY TO BE USED BY			
/ RESTARTS & VECTORED INTERRUPTS			
			*000 010
000	010	303	JMP
000	011	010	010
000	012	003	003
			*000 020
000	020	303	JMP
000	021	020	020
000	022	003	003
			*000 030
000	030	303	JMP
000	031	030	030
000	032	003	003
			*000 040
000	040	303	JMP
000	041	040	040
000	042	003	003
			*000 050
000	050	303	JMP
000	051	050	050
000	052	003	003
			*000 060
000	060	303	JMP
000	061	060	060
000	062	003	003
/ BEGINNING OF MAIN PROGRAM			
			*000 070
000	070	061	START, LXISP /SET STACK POINTER TO TOP OF R/W MEM.
000	071	000	000
000	072	004	004
000	073	041	LXIH /INITIAL VALUE OF H & L
000	074	000	000
000	075	003	003
000	076	116	POINTA, MOVCM /LOAD MEM DATA INTO TEMP DATA BUFFER
000	077	174	MOVAH /OUTPUT HI TO LED'S
000	100	323	OUT
000	101	001	001
000	102	175	MOVAL /OUTPUT LOW TO LED'S

KEX PROGRAM LISTING (CONTINUED)

ADDRESS		DATA	
HI	LOW	B7...B0	
000	103	323	OUT
000	104	000	000
000	105	171	POINTB, MOVAC / OUTPUT TEMP. DATA BUFFER DATA TO LED'S
000	106	323	OUT
000	107	002	002
000	110	315	POINTC, CALL / WAIT & INPUT NEXT KEY CLOSURE
000	111	315	KBRD
000	112	000	0
000	113	376	CPI
000	114	010	010
000	115	322	JNC / JUMP IF KEY WAS < 010
000	116	134	POINTD / (0-7 , OCTAL DIGIT)
000	117	000	0
000	120	107	MOVBA / SAVE KEY CODE
000	121	171	MOVAC / GET OLD VALUE
000	122	027	RAL / ROTATE 3 TIMES
000	123	027	RAL
000	124	027	RAL
000	125	346	ANI / MAK OUT LEAST SIG. OCTAL DIGIT
000	126	370	370
000	127	260	ORAB / OR IN NEW OCTAL DIGIT
000	130	117	MOVCA / PUT NEW DATA BACK INTO BUFFER
000	131	303	JMP
000	132	105	POINTB
000	133	000	0
000	134	376	POINTD, CPI
000	135	011	011 / "L" KEY
000	136	302	JNZ / JUMP IF NOT AN "L"
000	137	145	POINTE
000	140	000	0
000	141	151	MOVLCL / PUT BUFFER DATA IN L
000	142	303	JMP
000	143	076	POINTA
000	144	000	0
000	145	376	POINTE, CPI
000	146	010	010 / "H" KEY
000	147	302	JNZ / JUMP IF NOT AN "H"
000	150	156	POINTF
000	151	000	0
000	152	141	MOVHCL / PUT BUFFER DATA IN H
000	153	303	JMP
000	154	076	POINTA
000	155	000	0
000	156	376	POINTF, CPI
000	157	013	013 / "S" KEY
000	160	302	JNZ / JUMP IF NOT "S"
000	161	170	POINTG
000	162	000	0
000	163	161	MOVMC / PUT TEMP. DATA INTO MEMORY
000	164	043	INXH / INCERMENT H & L
000	165	303	JMP
000	166	076	POINTA
000	167	000	0
000	170	376	POINTG, CPI

KEX PROGRAM LISTING (CONTINUED)

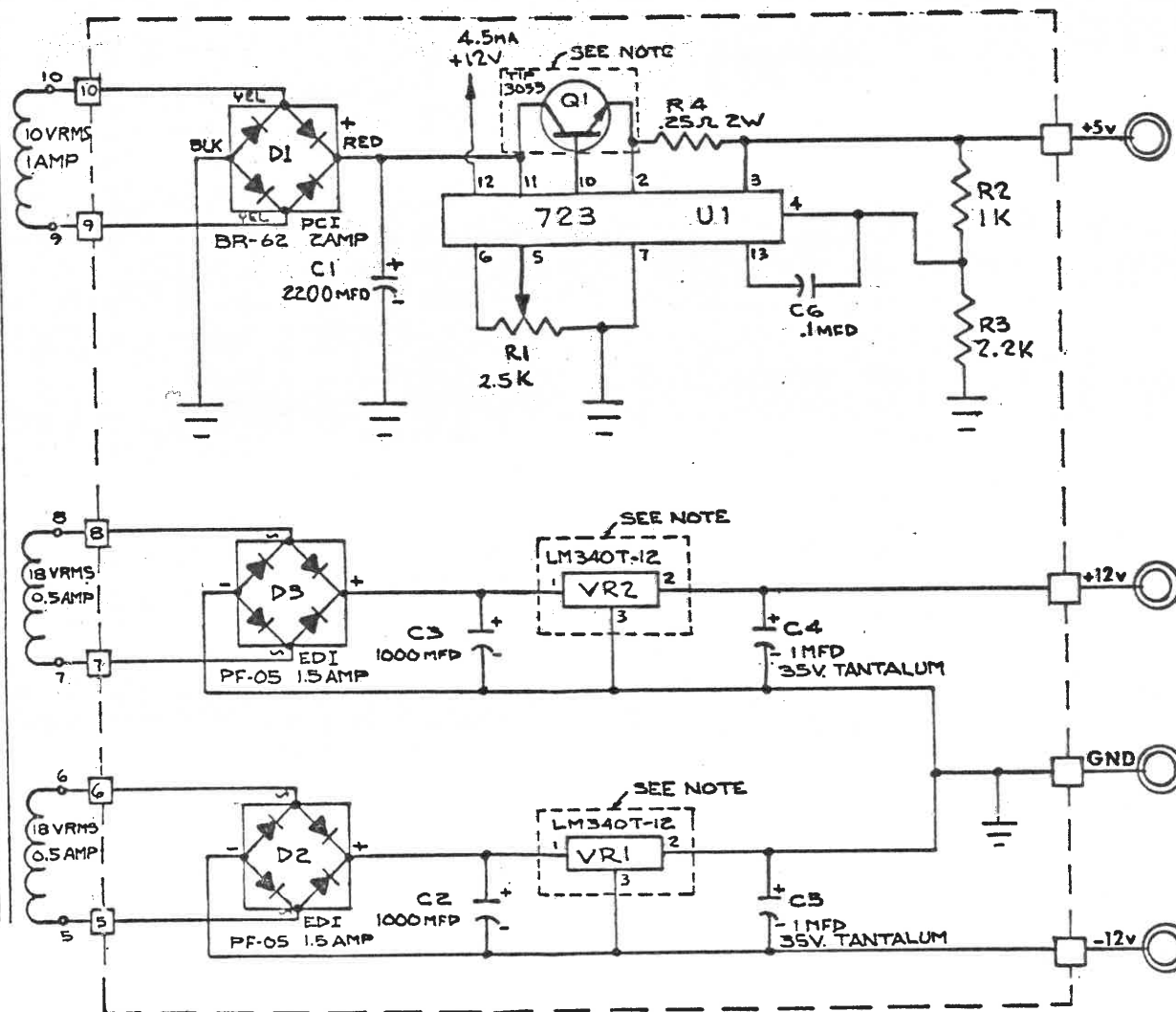
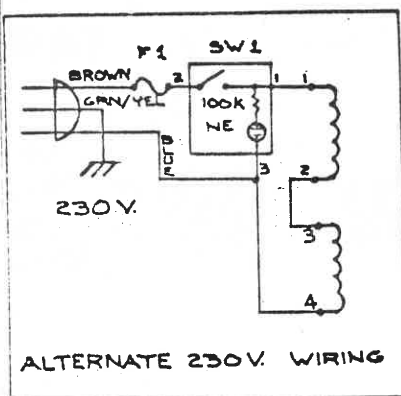
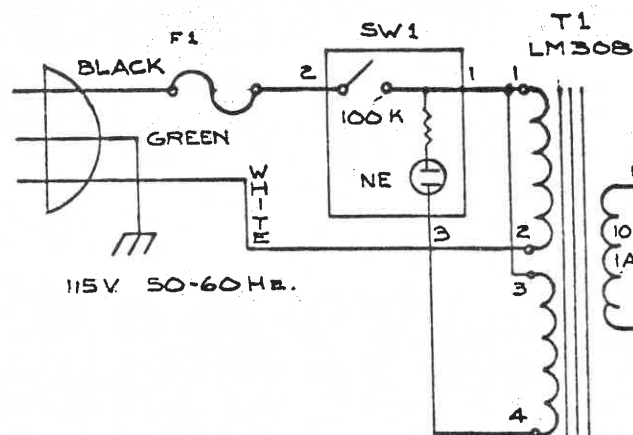
ADDRESS		DATA	
HI	LOW	B7...B0	
000	171	012	012 / "G" KEY
000	172	302	JNZ / JUMP IF NOT "G"
000	173	110	POINTC
000	174	000	0
000	175	351	PCHL / GO EXECUTE PGM POINTED TO BY H & L
/ THIS 10 MSEC DELAY DISTURBS NO REGISTERS OR FLAG			
*000 277			
000	277	365	TIMEOUT, PUSHPSW / SAVE REGISTERS
000	300	325	PUSHD
000	301	021	LXID / LOAD D & E WITH VALUE TO BE DECREMENTED
000	302	046	046 (E)
000	303	001	001 (D)
000	304	033	MORE, DCXD / JUMP IN THIS LOOP UNTIL
000	305	172	MOVAD / D & E ARE BOTH ZERO
000	306	263	ORAE
000	307	302	JNZ
000	310	304	MORE
000	311	000	0
000	312	321	POPD
000	313	361	POPPSW / RESTORE REGISTERS
000	314	311	RET
/ THE KBRD ROUTINE DEBOUNCES KEY CLOSURES			
/ AND TRANSLATES KEY CODES			
/ FLAGS AND REG A ARE CHANGED			
/ A0-A3= CODE ; A4-A7= 0000			
000	315	333	KBRD, IN / INPUT FROM KEYBOARD ENCODERS
000	316	000	000
000	317	267	ORAA / SET FLAGS
000	320	372	JM / JUMP BACK IF LAST KEY NOT RELEASED
000	321	315	KBRD
000	322	000	0
000	323	315	CALL / WAIT 10 MSEC
000	324	277	TIMOUT
000	325	000	0
000	326	333	FLAGCK, IN
000	327	000	000
000	330	267	ORAA
000	331	362	JP / JUMP BACK TO WAIT FOR A NEW
000	332	326	FLAGCK / KEY TO BE PRESSED
000	333	000	0
000	334	315	CALL / WAIT 10 MSEC FOR BOUNCING
000	335	277	TIMOUT
000	336	000	0
000	337	333	IN
000	340	000	000
000	341	267	ORAA
000	342	362	JP / JUMP BACK IF NEW KEY NOT STILL

KEX PROGRAM LISTING (CONTINUED)

ADDRESS		DATA	
HI	LOW	B ₇ ...B ₀	
000	343	326	FLAGCK / PRESSED (FALSE ALARM)
000	344	000	0
000	345	346	ANI / MASK OUT ALL BUT KEY CODE
000	346	017	017
000	347	345	PUSHH / SAVE H&L
000	350	046	MVIH / ZERO H REG
000	351	000	000
000	352	306	ADI / ADD THE ADDRESS OF THE BEGINNING OF
000	353	360	360 / THE TABLE TO THE KEY CODE
000	354	157	MOVLA /
000	355	176	MOVAM / FETCH NEW VALUE FROM TABLE
000	356	341	POPH / RESTORE H & L
000	357	311	RET

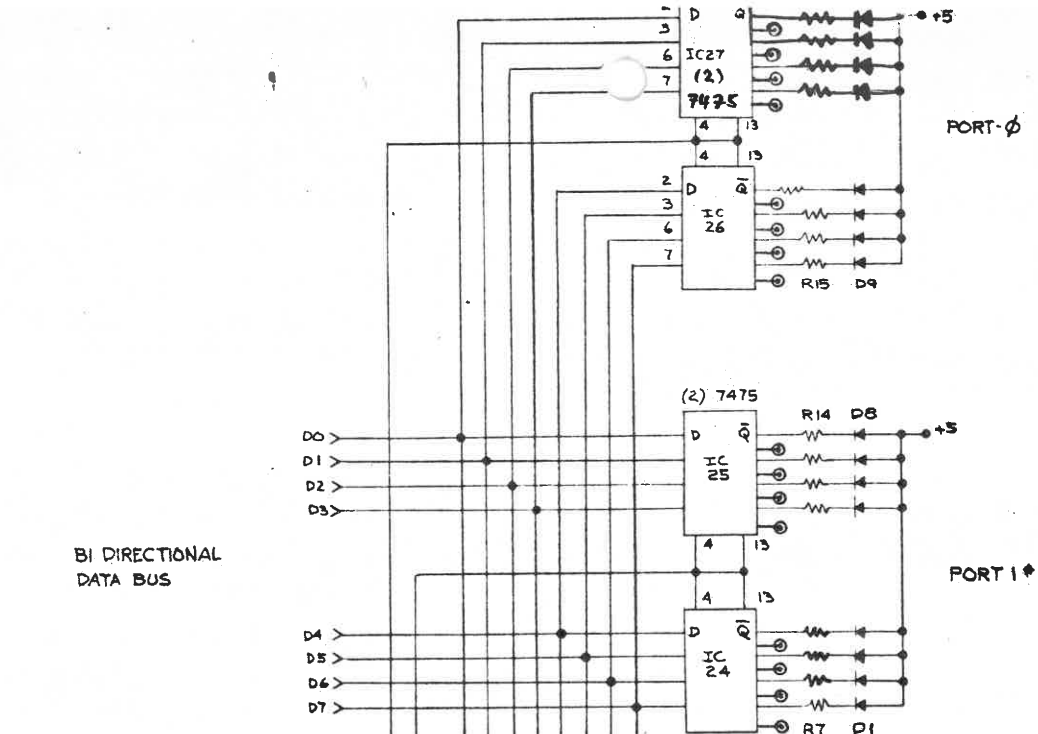
/ THIS TRANSLATION TABLE CONVERTS THE CODE
 / GENERATED BY KEY CLOSURES TO THE CODE
 / USED BY THE MAIN KEX PROGRAM

000	360	000	TABLE,	000
000	361	001		001
000	362	002		002
000	363	003		003
000	364	004		004
000	365	005		005
000	366	006		006
000	367	007		007
000	370	013		013 / S
000	371	000		000 / THIS CODE CAN'T BE GENERATED
000	372	017		017 / C
000	373	012		012 / G
000	374	010		010 / H
000	375	011		011 / L
000	376	015		015 / A
000	377	016		016 / B

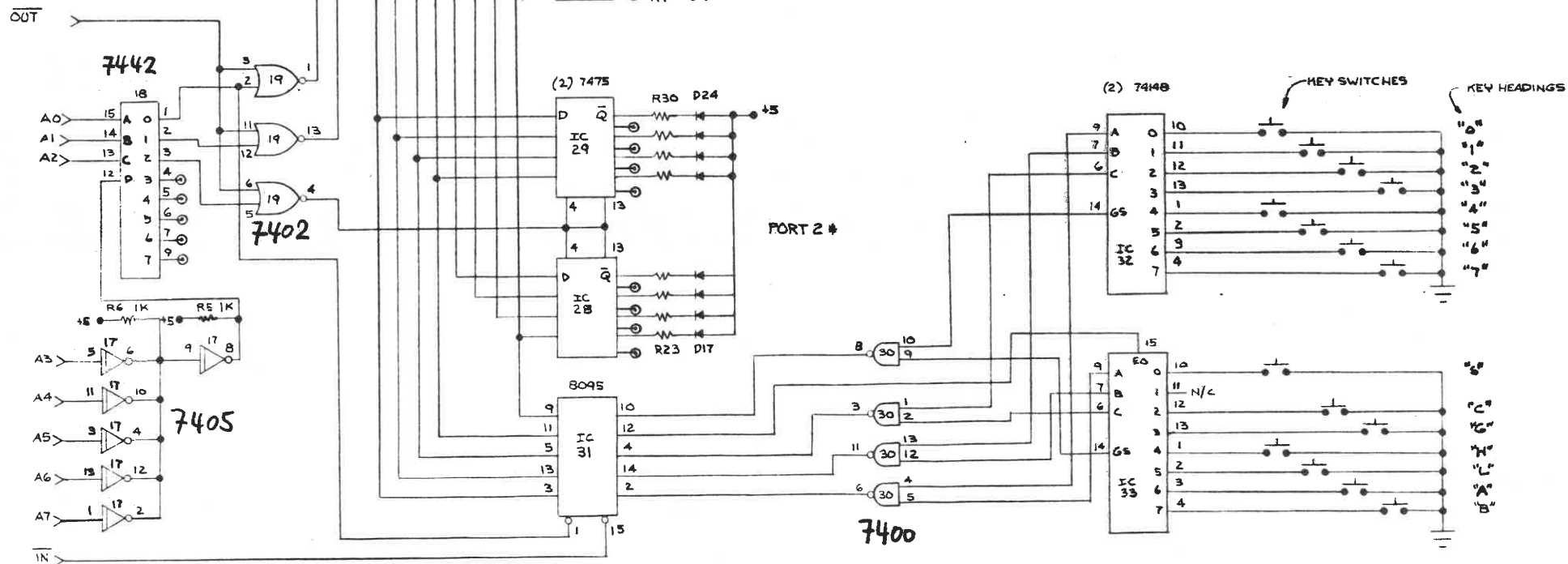
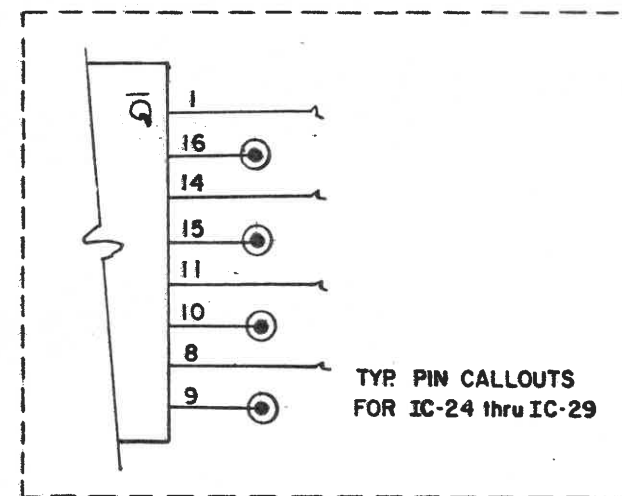




NOTE: Q1, VR1, VR2
MOUNTED ON REAR
PANEL OF CASE.

POWER SUPPLY SCHEMATIC



PUT/OUTPUT PORTS • LED DISPLAYS AND KEYBOARD
NOTES:
• WIRED AS PER OUTPUT PORT Q, EXCEPT,
PORT Q - LATCH Q OUTPUTS AVAILABLE ON
SOLDER PADS.



NOTE:  INDICATES EX-10 SOCKET CONNECTION
 INDICATES SP-25 BREADBOARDING PIN, AVAILABLE TO USER
 * INDICATES LOW-POWER SIGNAL

8085 INSTRUCTION SET*

A computer, no matter how sophisticated, can only do what it is "told" to do. One "tells" the computer what to do via a series of coded instructions referred to as a **Program**. The realm of the programmer is referred to as **Software**, in contrast to the **Hardware** that comprises the actual computer equipment. A computer's software refers to all of the programs that have been written for that computer.

When a computer is designed, the engineers provide the Central Processing Unit (CPU) with the ability to perform a particular set of operations. The CPU is designed such that a specific operation is performed when the CPU control logic decodes a particular instruction. Consequently, the operations that can be performed by a CPU define the computer's **Instruction Set**.

Each computer instruction allows the programmer to initiate the performance of a specific operation. All computers implement certain arithmetic operations in their instruction set, such as an instruction to add the contents of two registers. Often logical operations (e.g., OR the contents of two registers) and register operate instructions (e.g., increment a register) are included in the instruction set. A computer's instruction set will also have instructions that move data between registers, between a register and memory, and between a register and an I/O device. Most instruction sets also provide **Conditional Instructions**. A conditional instruction specifies an operation to be performed only if certain conditions have been met; for example, jump to a particular instruction if the result of the last operation was zero. Conditional instructions provide a program with a decision-making capability.

By logically organizing a sequence of instructions into a coherent program, the programmer can "tell" the computer to perform a very specific and useful function.

The computer, however, can only execute programs whose instructions are in a binary coded form (i.e., a series of 1's and 0's), that is called **Machine Code**. Because it would be extremely cumbersome to program in machine code, programming languages have been developed. There

are programs available which convert the programming language instructions into machine code that can be interpreted by the processor.

One type of programming language is **Assembly Language**. A unique assembly language mnemonic is assigned to each of the computer's instructions. The programmer can write a program (called the **Source Program**) using these mnemonics and certain operands; the source program is then converted into machine instructions (called the **Object Code**). Each assembly language instruction is converted into one machine code instruction (1 or more bytes) by an **Assembler** program. Assembly languages are usually machine dependent (i.e., they are usually able to run on only one type of computer).

THE 8085 INSTRUCTION SET

The 8085 instruction set includes five different types of instructions:

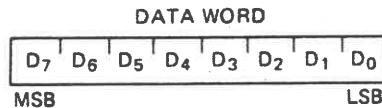
- **Data Transfer Group**—move data between registers or between memory and registers
- **Arithmetic Group**—add, subtract, increment or decrement data in registers or in memory
- **Logical Group**—AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory
- **Branch Group**—conditional and unconditional jump instructions, subroutine call instructions and return instructions
- **Stack, I/O and Machine Control Group**—includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

Instruction and Data Formats:

Memory for the 8085 is organized into 8-bit quantities, called **Bytes**. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

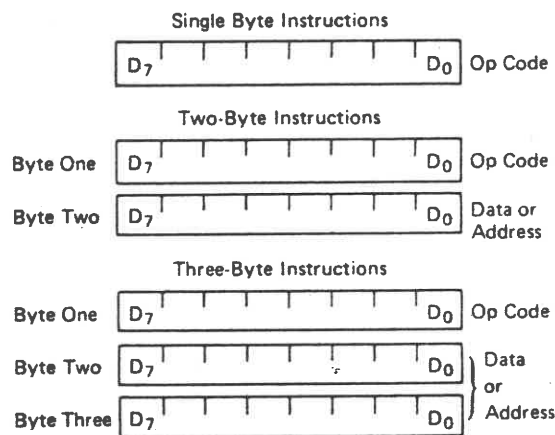
The 8085 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8085 is stored in the form of 8-bit binary integers:



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8085, BIT 0 is referred to as the Least Significant Bit (LSB), and BIT 7 (of an 8 bit number) is referred to as the Most Significant Bit (MSB).

The 8085 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.



Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8085 has four different modes for addressing data stored in memory or in registers:

- **Direct** – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- **Register** – The instruction specifies the register or register-pair in which the data is located.
- **Register Indirect** – The instruction specifies a register-pair which contains the memory

address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).

- **Immediate** – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- **Direct** – The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- **Register indirect** – The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

There are five condition flags associated with the execution of instructions on the 8085. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

- Zero:** If the result of an instruction has the value 0, this flag is set; otherwise it is reset.
- Sign:** If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.
- Parity:** If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).
- Carry:** If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

Symbols and Abbreviations:

The following symbols and abbreviations are used in the subsequent description of the 8085A instructions:

SYMBOLS MEANING

accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r,r1,r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD=destination, SSS=source):

DDD or SSS REGISTER NAME

111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp One of the register pairs:
B represents the B,C pair with B as the high-order register and C as the low-order register;
D represents the D,E pair with D as the high-order register and E as the low-order register;
H represents the H,L pair with H as the high-order register and L as the low-order register;
SP represents the 16-bit stack pointer register.

RP

The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh

The first (high-order) register of a designated register pair.

rl

The second (low-order) register of a designated register pair.

PC

16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).

SP

16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).

r_m

Bit m of the register r (bits are number 7 through 0 from left to right).

Z,S,P,CY,AC

The condition flags:

Zero,
Sign,
Parity,
Carry,
and Auxiliary Carry, respectively.

()

The contents of the memory location or registers enclosed in the parentheses.

←

"Is transferred to"

∧

Logical AND

⊕

Exclusive OR

∨

Inclusive OR

+

Addition

-

Two's complement subtraction

Multiplication

↔

"Is exchanged with"

—

The one's complement (e.g., (\bar{A}))

n

The restart number 0 through 7

NNN

The binary representation 000 through 111 for restart number 0 through 7 respectively.

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

Data Transfer Group:

This group of instructions transfers data to and from registers and memory. Condition flags are not affected by any instruction in this group.

MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1

States: 4 *Anz. Taktezyklen (500ns / 2 MHz)*

Addressing: register

Flags: none

MOV r, M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

0	1	D	D	D	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2

States: 7

Addressing: reg. indirect

Flags: none

MOV M, r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

0	1	1	1	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 2

States: 7

Addressing: reg. indirect

Flags: none

MVI r, data (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles: 2

States: 7

Addressing: immediate

Flags: none

MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles: 3

States: 10

Addressing: immed./reg. indirect

Flags: none

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

0	0	R	P	0	0	0	1
low-order data				high-order data			

Cycles: 3

States: 10

Addressing: immediate

Flags: none

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

LDA addr (Load Accumulator direct) $(A) \leftarrow ((\text{byte } 3)(\text{byte } 2))$

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

Cycles: 4

States: 13

Addressing: direct

Flags: none

STA addr (Store Accumulator direct) $((\text{byte } 3)(\text{byte } 2)) \leftarrow (A)$

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles: 4

States: 13

Addressing: direct

Flags: none

LHLD addr (Load H and L direct) $(L) \leftarrow ((\text{byte } 3)(\text{byte } 2))$ $(H) \leftarrow ((\text{byte } 3)(\text{byte } 2) + 1)$

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

0	0	1	0	1	0	1	0
low-order addr							
high-order addr							

Cycles: 5

States: 16

Addressing: direct

Flags: none

SHLD addr (Store H and L direct) $((\text{byte } 3)(\text{byte } 2)) \leftarrow (L)$ $((\text{byte } 3)(\text{byte } 2) + 1) \leftarrow (H)$

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

Cycles: 5

States: 16

Addressing: direct

Flags: none

LDAX rp (Load accumulator indirect) $(A) \leftarrow ((rp))$

The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

0	0	R	P	1	0	1	0
---	---	---	---	---	---	---	---

Cycles: 2

States: 7

Addressing: reg. indirect

Flags: none

STAX rp (Store accumulator indirect) $((rp)) \leftarrow (A)$

The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

0	0	R	P	0	0	1	0
---	---	---	---	---	---	---	---

Cycles: 2

States: 7

Addressing: reg. indirect

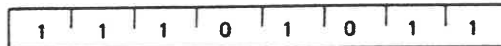
Flags: none

XCHG (Exchange H and L with D and E)

(H) \longleftrightarrow (D)

(L) \longleftrightarrow (E)

The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1

States: 4

Addressing: register

Flags: none

Arithmetic Group:

This group of instructions performs arithmetic operations on data in registers and memory.

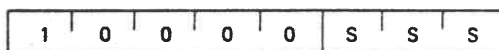
Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add Register)

(A) \leftarrow (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 1

States: 4

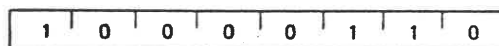
Addressing: register

Flags: Z,S,P,CY,AC

ADD M (Add memory)

(A) \leftarrow (A) + ((H) (L))

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 2

States: 7

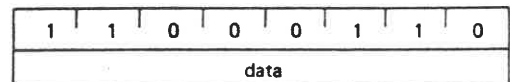
Addressing: reg. indirect

Flags: Z,S,P,CY,AC

ADI data (Add immediate)

(A) \leftarrow (A) + (byte 2)

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 2

States: 7

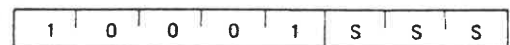
Addressing: immediate

Flags: Z,S,P,CY,AC

ADC r (Add Register with carry)

(A) \leftarrow (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 1

States: 4

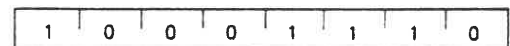
Addressing: register

Flags: Z,S,P,CY,AC

ADC M (Add memory with carry)

(A) \leftarrow (A) + ((H) (L)) + (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.



Cycles: 2

States: 7

Addressing: reg. indirect

Flags: Z,S,P,CY,AC

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

SEITE: 7

ACI data (Add immediate with carry)

$$(A) \leftarrow (A) + (\text{byte 2}) + (CY)$$

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

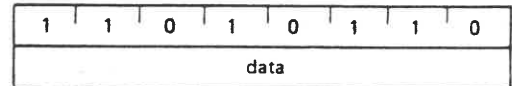


Cycles: 2
 States: 7
 Addressing: immediate
 Flags: Z,S,P,CY,AC

SUI data (Subtract immediate)

$$(A) \leftarrow (A) - (\text{byte 2})$$

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 2
 States: 7
 Addressing: immediate
 Flags: Z,S,P,CY,AC

SUB r (Subtract Register)

$$(A) \leftarrow (A) - (r)$$

The content of register *r* is subtracted from the content of the accumulator. The result is placed in the accumulator.

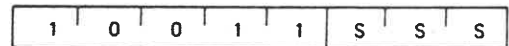


Cycles: 1
 States: 4
 Addressing: register
 Flags: Z,S,P,CY,AC

SBB r (Subtract Register with borrow)

$$(A) \leftarrow (A) - (r) - (CY)$$

The content of register *r* and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

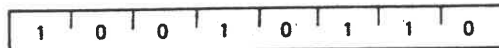


Cycles: 1
 States: 4
 Addressing: register
 Flags: Z,S,P,CY,AC

SUB M (Subtract memory)

$$(A) \leftarrow (A) - ((H) (L))$$

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

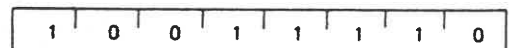


Cycles: 2
 States: 7
 Addressing: reg. indirect
 Flags: Z,S,P,CY,AC

SBB M (Subtract memory with borrow)

$$(A) \leftarrow (A) - ((H) (L)) - (CY)$$

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



Cycles: 2
 States: 7
 Addressing: reg. indirect
 Flags: Z,S,P,CY,AC

SBI data (Subtract immediate with borrow)

$(A) \leftarrow (A) - (\text{byte 2}) - (CY)$

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	1	0	1	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

INR r (Increment Register)

$(r) \leftarrow (r) + 1$

The content of register r is incremented by one.
Note: All condition flags except CY are affected.

0	0	D	D	D	1	0	0
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,AC

INR M (Increment memory)

$((H) (L)) \leftarrow ((H) (L)) + 1$

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags except CY are affected.

0	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

DCR r (Decrement Register)

$(r) \leftarrow (r) - 1$

The content of register r is decremented by one.
Note: All condition flags except CY are affected.

0	0	D	D	D	1	0	1
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,AC

DCR M (Decrement memory)

$((H) (L)) \leftarrow ((H) (L)) - 1$

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags except CY are affected.

0	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

INX rp (Increment register pair)

$(rh) (rl) \leftarrow (rh) (rl) + 1$

The content of the register pair rp is incremented by one. Note: No condition flags are affected.

0	0	R	P	0	0	1	1
---	---	---	---	---	---	---	---

Cycles: 1
States: 6
Addressing: register
Flags: none

DCX rp (Decrement register pair)

$(rh) (rl) \leftarrow (rh) (rl) - 1$

The content of the register pair rp is decremented by one. Note: No condition flags are affected.

0	0	R	P	1	0	1	1
---	---	---	---	---	---	---	---

Cycles: 1
States: 6
Addressing: register
Flags: none

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

DAD rp (Add register pair to H and L) $(H) (L) \leftarrow (H) (L) + (rh) (rl)$

The content of the register pair *rp* is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: Only the **CY flag** is affected. It is set if there is a carry out of the double precision add; otherwise it is reset.

0	0	R	P	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3
States: 10
Addressing: register
Flags: CY

DAA (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Flags: Z,S,P,CY,AC

Logical Group:

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

ANA r (AND Register) $(A) \leftarrow (A) \wedge (r)$

The content of register *r* is logically anded with the content of the accumulator. The result is placed in the accumulator. The **CY flag** is cleared and **AC** is set.

1	0	1	0	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ANA M (AND memory) $(A) \leftarrow (A) \wedge ((H) (L))$

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. The **CY flag** is cleared and **AC** is set.

1	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ANI data (AND immediate) $(A) \leftarrow (A) \wedge (\text{byte } 2)$

The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. The **CY flag** is cleared and **AC** is set.

1	1	1	0	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA r (Exclusive OR Register) $(A) \leftarrow (A) \vee (r)$

The content of register *r* is exclusive-or'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.

1	0	1	0	1	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

XRA M (Exclusive OR Memory) $(A) \leftarrow (A) \vee ((H) (L))$

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XRI data (Exclusive OR immediate) $(A) \leftarrow (A) \vee (\text{byte 2})$

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	1	1	0	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ORA r (OR Register) $(A) \leftarrow (A) \vee (r)$

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	1	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ORA M (OR memory) $(A) \leftarrow (A) \vee ((H) (L))$

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	1	0	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ORI data (OR Immediate) $(A) \leftarrow (A) \vee (\text{byte 2})$

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	1	1	1	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

CMP r (Compare Register) $(A) - (r)$

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if $(A) = (r)$. The CY flag is set to 1 if $(A) < (r)$.

1	0	1	1	1	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

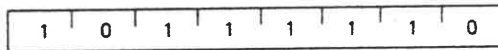
MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

SEITE: 11

CMP M (Compare memory) $(A) - ((H) (L))$

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if $(A) = ((H) (L))$. The CY flag is set to 1 if $(A) < ((H) (L))$.



Cycles: 2

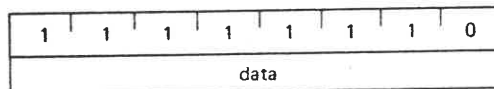
States: 7

Addressing: reg. indirect

Flags: Z,S,P,CY,AC

CPI data (Compare immediate) $(A) - (\text{byte 2})$

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if $(A) = (\text{byte 2})$. The CY flag is set to 1 if $(A) < (\text{byte 2})$.



Cycles: 2

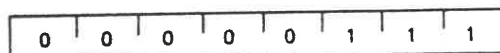
States: 7

Addressing: immediate

Flags: Z,S,P,CY,AC

RLC (Rotate left) $(A_{n+1}) \leftarrow (A_n) ; (A_0) \leftarrow (A_7)$ $(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. Only the CY flag is affected.



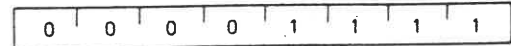
Cycles: 1

States: 4

Flags: CY

RRC (Rotate right) $(A_n) \leftarrow (A_{n+1}) ; (A_7) \leftarrow (A_0)$ $(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. Only the CY flag is affected.



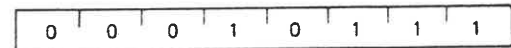
Cycles: 1

States: 4

Flags: CY

RAL (Rotate left through carry) $(A_{n+1}) \leftarrow (A_n) ; (CY) \leftarrow (A_7)$ $(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. Only the CY flag is affected.



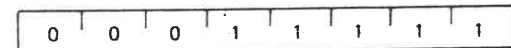
Cycles: 1

States: 4

Flags: CY

RAR (Rotate right through carry) $(A_n) \leftarrow (A_{n+1}) ; (CY) \leftarrow (A_0)$ $(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. Only the CY flag is affected.



Cycles: 1

States: 4

Flags: CY

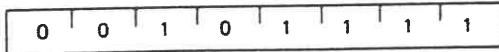
MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

SEITE: 12

CMA (Complement accumulator)(A) \leftarrow (A)

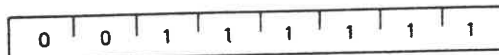
The contents of the accumulator are complemented (zero bits become 1, one bits become 0). No flags are affected.



Cycles: 1
States: 4
Flags: none

CMC (Complement carry)(CY) \leftarrow (CY)

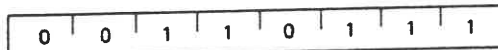
The CY flag is complemented. No other flags are affected.



Cycles: 1
States: 4
Flags: CY

STC (Set carry)(CY) \leftarrow 1

The CY flag is set to 1. No other flags are affected.



Cycles: 1
States: 4
Flags: CY

Branch Group:

This group of instructions alter normal sequential program flow.

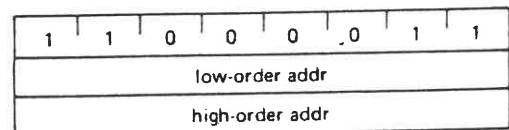
Condition flags are not affected by any instruction in this group.

The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION		CCC
NZ	— not zero (Z = 0)	000
Z	— zero (Z = 1)	001
NC	— no carry (CY = 0)	010
C	— carry (CY = 1)	011
PO	— parity odd (P = 0)	100
PE	— parity even (P = 1)	101
P	— plus (S = 0)	110
M	— minus (S = 1)	111

JMP addr (Jump)(PC) \leftarrow (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 3
States: 10
Addressing: immediate
Flags: none

Jcondition addr (Conditional jump)

If (CCC),

(PC) \leftarrow (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

1	1	C	C	C	0	1	0
low-order addr							
high-order addr							

Cycles: 2/3

States: 7/10

Addressing: immediate

Flags: none

CALL addr (Call)

((SP) - 1) \leftarrow (PCH)

((SP) - 2) \leftarrow (PCL)

(SP) \leftarrow (SP) - 2

(PC) \leftarrow (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cycles: 5

States: 18

Addressing: immediate/reg. indirect

Flags: none

Ccondition addr (Condition call)

If (CCC),

((SP) - 1) \leftarrow (PCH)

((SP) - 2) \leftarrow (PCL)

(SP) \leftarrow (SP) - 2

(PC) \leftarrow (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	1	0	0
low-order addr							
high-order addr							

Cycles: 2/5

States: 9/18

Addressing: immediate/reg. indirect

Flags: none

RET (Return)

(PCL) \leftarrow ((SP));

(PCH) \leftarrow ((SP) + 1);

(SP) \leftarrow (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

1	1	0	0	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3

States: 10

Addressing: reg. indirect

Flags: none

Rcondition (Conditional return)

If (CCC),

(PCL) \leftarrow ((SP));

(PCH) \leftarrow ((SP) + 1);

(SP) \leftarrow (SP) + 2;

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	0	0	0
---	---	---	---	---	---	---	---

Cycles: 1/3

States: 6/12

Addressing: reg. indirect

Flags: none

RST n (Restart)

$((SP) - 1) \leftarrow (PCH)$
 $((SP) - 2) \leftarrow (PCL)$
 $(SP) \leftarrow (SP) - 2$
 $(PC) \leftarrow 8 * (NNN)$

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

1	1	N	N	N	1	1	1
---	---	---	---	---	---	---	---

Cycles: 3
 States: 12
 Addressing: reg. indirect
 Flags: none

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Program Counter After Restart

PCHL (Jump H and L indirect - move H and L to PC)

$(PCH) \leftarrow (H)$
 $(PCL) \leftarrow (L)$

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

1	1	1	0	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 1
 States: 6
 Addressing: register
 Flags: none

Stack, I/O, and Machine Control Group:

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, condition flags are not affected by any instructions in this group.

PUSH rp (Push)

$((SP) - 1) \leftarrow (rh)$
 $((SP) - 2) \leftarrow (rl)$
 $(SP) \leftarrow (SP) - 2$

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Note: Register pair $rp = SP$ may not be specified.

1	1	R	P	0	1	0	1
---	---	---	---	---	---	---	---

Cycles: 3
 States: 12
 Addressing: reg. indirect
 Flags: none

PUSH PSW (Push processor status word)

$((SP) - 1) \leftarrow (A)$
 $((SP) - 2)_0 \leftarrow (CY), ((SP) - 2)_1 \leftarrow X$
 $((SP) - 2)_2 \leftarrow (P), ((SP) - 2)_3 \leftarrow X$
 $((SP) - 2)_4 \leftarrow (AC), ((SP) - 2)_5 \leftarrow X$
 $((SP) - 2)_6 \leftarrow (Z), ((SP) - 2)_7 \leftarrow (S)$
 $(SP) \leftarrow (SP) - 2$ X: Undefined.

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.

1	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Cycles: 3
 States: 12
 Addressing: reg. indirect
 Flags: none

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehls - Satz

FLAG WORD

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
S	Z	X	AC	X	P	X	CY

X: undefined

POP rp (Pop)

(rl) ← ((SP))
 (rh) ← ((SP) + 1)
 (SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. Note: Register pair rp = SP may not be specified.

1	1	R	P	0	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3
 States: 10
 Addressing: reg. indirect
 Flags: none

POP PSW (Pop processor status word)

(CY) ← ((SP))₀
 (P) ← ((SP))₂
 (AC) ← ((SP))₄
 (Z) ← ((SP))₆
 (S) ← ((SP))₇
 (A) ← ((SP) + 1)
 (SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

1	1	1	1	0	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3
 States: 10
 Addressing: reg. indirect
 Flags: Z,S,P,CY,AC

XTHL (Exchange stack top with H and L)

(L) ↔ ((SP))
 (H) ↔ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

1	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Cycles: 5
 States: 16
 Addressing: reg. indirect
 Flags: none

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

1	1	1	1	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 1
 States: 6
 Addressing: register
 Flags: none

IN port (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

1	1	0	1	1	0	1	1
port							

Cycles: 3
 States: 10
 Addressing: direct
 Flags: none

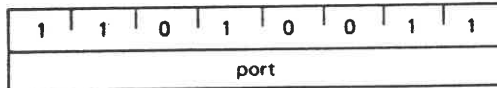
MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehlssatz

OUT port (Output)

(data) ← (A)

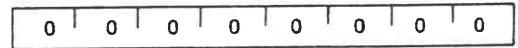
The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.



Cycles: 3
States: 10
Addressing: direct
Flags: none

NOP (No op)

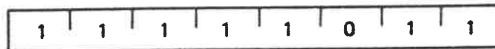
No operation is performed. The registers and flags are unaffected.



Cycles: 1
States: 4
Flags: none

EI (Enable interrupts)

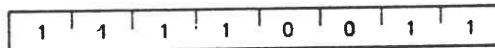
The interrupt system is enabled following the execution of the next instruction.



Cycles: 1
States: 4
Flags: none

DI (Disable interrupts)

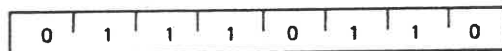
The interrupt system is disabled immediately following the execution of the DI instruction.



Cycles: 1
States: 4
Flags: none

HLT (Halt)

The processor is stopped. The registers and flags are unaffected.



Cycles: 1
States: 5
Flags: none

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehlssatz

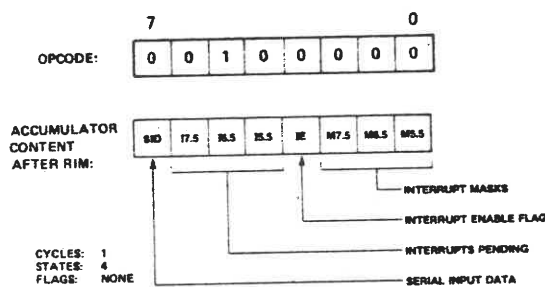
8085A

RIM (Read Interrupt Mask)

After the execution of the RIM instruction, the accumulator is loaded with the restart interrupt masks, the Interrupt Enable Flag, any pending interrupts, and the contents of the serial input data line (SID).

When the first RIM instruction is executed following a TRAP interrupt, the IE bit in the accumulator reflects the *previous* interrupt enable status before the TRAP occurred. The first RIM after a TRAP also has the action of releasing the IE status bit such that all subsequent RIM's provide current interrupt enable status.

IMPORTANT: The RIM instruction must be executed following a TRAP interrupt in order to restore the correct Interrupt Enable Status.



SIM (Set Interrupt Masks)

During execution of the SIM instruction, the contents of the accumulator will be used in programming the restart interrupt masks. Bits 0-2 will set/reset the mask bit for RST 5.5, 6.5, 7.5 of the interrupt mask register, if bit 3 is 1 ("set"). Bit 3 is a "Mask Set Enable" control.

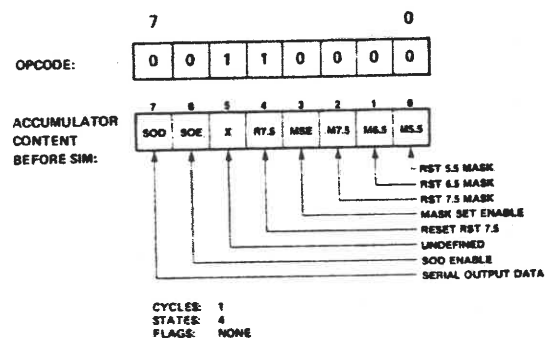
Setting the mask (i.e. mask bit = 1) disables the corresponding interrupt.

	Set	Reset
RST 5.5 MASK	if bit 0 = 1	if bit 0 = 0
RST 6.5 MASK	bit 1 = 1	bit 1 = 0
RST 7.5 MASK	bit 2 = 1	bit 2 = 0

The RST 7.5 (edge triggered) internal request flip flop will be reset if bit 4 of the accumulator = 1; regardless of whether RST 7.5 is masked or not.

A hardware RESET of the 8085A will set all RST MASKs, and reset/disable all interrupts.

SIM can also load the SOD output latch. Accumulator bit 7 is loaded into the SOD latch if bit 6 is set. The latch is unaffected if bit 6 is a zero. RESET IN input sets the SOD latch to zero.



MIKROCOMPUTER - GRUNDKURS A. 8080 - Befehlssatz

KAPITEL: A

SEITE: 18

Copyright © 1979 by DIGICOMP AG ZÜRICH
Dieses Blatt ist zweifarbig. Haben Sie eine eintarige Kopie vor sich, so wurde diese widerrechtlich hergestellt und muss sofort vernichtet werden.
Zusätzlich bestellerbezogen geschützt durch COPYGUARD®.

Mnemonic	Description	Instruction Code(I)								Clock(Z)
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
MOVE, LOAD, AND STORE										
MOV r,r	Move register to register	0	1	0	0	0	S	S	S	4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	0	0	0	1	1	0	7
MVI r	Move immediate register	0	0	0	0	0	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register	0	0	0	0	0	0	0	1	10
	Pair B & C									
LXI D	Load immediate register	0	0	0	1	0	0	0	1	10
	Pair D & E									
LXI H	Load immediate register	0	0	1	0	0	0	0	1	10
	Pair H & L									
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	12
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18

Mnemonic	Description	Instruction Code(I)								Clocks(Z)
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RETURN										
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	6/12
RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
RNZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	12
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	0	0	0	1	0	0	4
DCR r	Decrement register	0	0	0	0	1	0	1	0	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	0	1	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	4

DIGICOMP AG
Werdstrasse 36 8004 Zürich 01 241 79 09

COMPUTEACH®

MIKROCOMPUTER - GRUNDKURS

A. 8080 - Befehlssatz

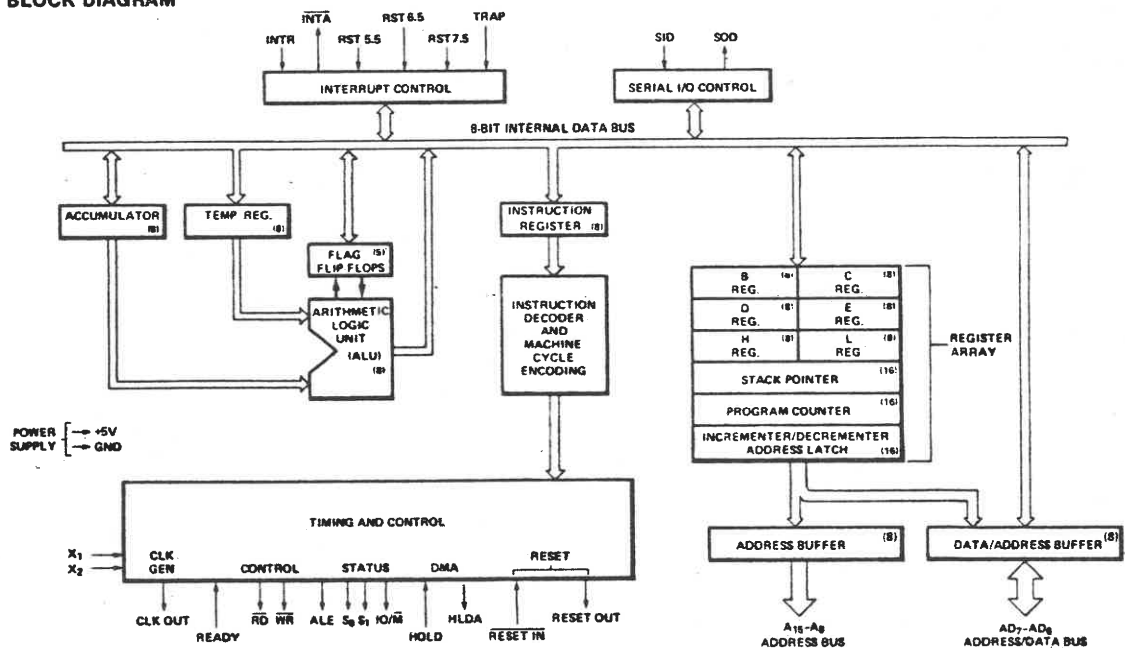
8085A INSTRUCTION SET SUMMARY (Cont.)

Mnemonic	Description	Instruction Code(1)								Clock(2) Cycles
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
CONTROL										
EI	Enable interrupts	1	1	1	1	1	0	0	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	5
NEW 8085A INSTRUCTIONS										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDD or SSS: B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.
2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

*All mnemonics copyright ©Intel Corporation 1977

8085A, 8085A-2 CPU FUNCTIONAL BLOCK DIAGRAM



DIGICOMP AG
Werdstrasse 36 8004 Zürich 01 241 79 09

COMPUTEACH®