8080 OPERATING SYSTEM for the Digital Group 8080 CPU Card

General Design

This programming system provides five key programs and many supporting subroutines. The user is able to enter his programming, check out his programming, and finally run his programming under the control of these five included programs.

The first program is a cassette reading program, almost completely contained on the Eraseable Read Only Memory (EROM). A frequency shifting data cassette is converted from serial data to parallel data and loaded into memory. The default data rate is 1100 BPS, and the default start and stop addresses are 001 000 and 007 377 respectively.

The next program is a cassette writing program which allows storing the contents of memory on a low-cost audio cassette recorder. The default data rate and addresses are the same as for cassette reading.

A storage dump program uses the CRT readout board and a TV set to display several items necessary to ease programming. The six internal registers of the 8080 CPU are interpreted and displayed exactly as they were immediately prior to calling the TV storage dump program. The internal 8080 status flags are also dumped and interpreted as are the stack pointer address and return address. The return address is only valid should the TV storage dump occur during a subroutine. The full contents of memory are then displayed, 96 bytes at a time, except for every 3rd display which culminates a page boundary. The initial address for each line is displayed at the left of each line, and six sequential bytes are displayed to the right.

A keyboard programming capability allows entering octal code directly from the system keyboard. The default address is 006 000. Programming may be entered at any available address, but programming below 006 000 runs the risk of destroying key portions of the operating system.

The final programming section is an operations monitor. The TV displays a list of up to ten options available to the user. The user then enters the number of the desired operation, and a table lookup selection performs a branch to the desired program.

Using the Digital Group 8080 Operating System

Initial Cassette Read:

After turning on the microprocessor, the message "READ 8080 INITIALIZE Cassette" will appear on the screen. Start the cassette recorder reading the cassette, and when the low tone begins, push the reset button and release. When data begins after the short tone leader, the TV will display the least significant digit of the octal page being currently loaded, byte by byte. Memory is checked byte by byte, and missing or defective memory addresses are indicated by a "." being printed instead of the page. When the tone stops, the operatior

monitor assumes control, and the program loops awaiting a keyboard entry of the desired selection.

Storage Dump:

The typical first entry will be a request to view storage to find some free area where some additional user supplied programming may be placed. Pressing a "3" will result in a display of the register, flag, and stack data.

Successively pressing the "Space" key will page through memory, 96 bytes at a time. To set storage immediately to a desired page rather than having to successively page up to it, enter an "S" (either upper or lower case is fine on entries) and the three digits (Most Significant, Middle, then Least Significant) which make up the desired page.

Entering a "P" will cause a branch to the keyboard programming routine

Entering an "R" will Return the control to the operations monitor.

Keyboard Program:

Once available locations in storage have been found, the user can manually enter programming from the keyboard by typing a "4" if in the operations monitor, or a "P" if in a TV dump. A title will be displayed along with the default address of the tape shipped.

Programming may be entered by merely typing in the desired octal code, MSB through LSB. The results will be displayed on the TV along with some past bytes to insure proper sequencing as well as aid short term entry error detection.

The page (high) and/or byte (low) address may be preset by entering an "H" and MSB through LSB of the octal address and/or an "L" and the MSB through LSB. The current address is displayed on the TV after entry. Memory is changed only following the third entry of the data byte.

Use care when entering code below 006 000. Since this is system area any code or operations can result in an inoperative system with no means of recovery other than re-reading the cassette.

Enter an "R" to return to the Operations Monitor.

Type an "S" to go to the Storage Dump directly from programming. Actual programming typically sees considerable "S" and "P" as entries are made, then viewed.

Cassette Write:

Once the desired programming has been entered, the user may wish to save it for later usage. The user is also advised to save all programming on cassette prior to initial execution to avoid potential programming self-destruction. If self-destruction upon execution

occurs, the program may be reloaded and suitable corrections made.

Insert a cassette and start the recorder in record mode. After making sure that the leader on the cassette has passed by the record head, enter a "2" while in the Operations Monitor. The TV will display the message "Cassette being written" until the cassette recording operation is finished about 1/2 minute later, then return to the Operations Monitor. Turn off the recorder, and you have the system and the added programming on the cassette.

Cassette Read:

Cassettes may be read by pressing "1" while in the Operations Monitor, or they can be read when power is applied.

Panic Button:

Pressing the reset button will always return the user to the initial cassette load, or Operations Monitor.

Fine Points of the 8080 Operating System:

Memory Extent:

The 8080 Operating System is designed to occupy the lower 1.5K of the 8080 CPU system. The default read and write high address is preset to 2K. However, the cassettes may be any length up to 64K, but at the read/write speed of 100 bytes per second, the cassette should be no longer than required.

If you have greater than 2K of memory on your system, modify the data at 001 205(byte) and 001 210 (page) to reflect the memory extent desired on the cassette. Example: You have 10K of 8080 system, and you wish to write 4K worth of programming. Since the octal equivalent of 4K is 017377, enter 377 at 001 205 and 017 at 001210. The default address is now set to 017 377. The cassette read programming will be automatically modified by the cassette. Cassettes of varying lengths may be interchangeably read with no operator intervention eg. 2K, 32K, 13K, 20K, etc.

Data Rate:

RAM address 001 027 contains the timing loop constant which controls the resultant cassette baud rate. The normal constant is 041, which resaults in 1100 baud. By making the constant larger, the timing loop is increased, and a baud rate of 165 baud is possible with a constant of 377, for example.

So what? Well, by using these lower data rates, a modem may be attached for inter-hobbyist telephone data transmissions at some standardized rate.

Address 001 027 will have to be preset prior to both Read and Write for proper operation.

Storage Dump:

The initial page of the TV dump which displays and interprets the registers, flags, and stack pointers can be the most useful part of the whole system when faced with a confusing software problem. Insert an unconditional branch to 003 000 in place of the byte following the point in question. This will display and interpret the registers and flags, generally giving a much better picture of what is happening in that "insolvable problem". Another technique is to use the "Restart 7" as a branch. This then involves inserting a single "377" byte. The "Restart 7" must then be vectored forward to 003 000. The software Operating System cassette included has this feature included, so you may get a storage dump by merely inserting a "377" in your programming.

Interrupts/Restarts 1-7:

The 8080 has eight restart or interrupt addresses at the low end of storage normally occupied by a ROM to give a power on and go capability. The EROM provided in the Digital Group kits vectors Restarts 1-7 through the EROM to the beginning of page 001 as shown in the software listings. The user may now vector forward these interrupt/restarts as desired, but interrupt level programming is best left to the experts. Restart 7 has the lowest level priority on the 8080 CPU board system.

Interrupt/Restart Ø:

The Reset function on the 8080 will force programming to begin at address 000 000, as does restart Ø ("307"). The Reset is used to control the Operations Monitor and the initial cassette read operation. It also has the highest priority of the eight interrupts. The EROM has control of Reset/Restart Ø and finally branches it forward to address 005 000 where the Operations Monitor resides.

Operations Monitor:

Page 005 of the 8080 Operating System is dedicated to aiding the user to make his program selections. The title area uses bytes 005 124 through 005 377. Up to 10 (0 - 9) different program start locations may be specified by putting the high and low addresses at the proper place between 005 100 and 005 123.

The user can title his program by inserting the ASCII characters desired in the format required. Here is the secret: A special subroutine called TV Editor controls the messages displayed on the TV screen. This subroutine is entered from the Operations Monitor to put the message on the TV. Address 005 272 - 005 377 can be used to enter a set of titles in a special machine code. "377" = Erase the screen, "376" - "200" are ASCII characters, "177" - "001" are the octal number of spaces, and "000" means the end of the message.

Example: You wish to add "5 Go" to the Operations Monitor message.

Address	Data	Explanation
005 271	016	Insert 14 blanks from last character
005 272	265	"5"
005 273	240	Space
005 274	307	"G"
005 275	357	"o"
005 276	000	End of message.

The program routing portion of the Operations Monitor is located between 005 100 and 005 123 as shown by the listings. The byte portion of the branch address is placed on the even address boundary, and the page portion on the odd address.

Example: You have designed the above program "Go" to execute from address 006 000. Since you also wish to branch to "Go" from a "5" entry when in the Operations Monitor, place an "000" at address 005 112 and an "006" at address 005 113.

Typing a "5" when in the Operations Monitor will now result in execution of "Go".

Subroutines You may wish to call for your own programming:

Subroutine	Address	Operation and Comments
TV	000 372	Prints a character on the TV through the Digital Group CRT readout attached to Port Ø. Load Accumulator with character prior to calling. Accumulator returned cleared to "000".
SPACE	000 370	Prints a space (blank position) on the TV. Accumulator need not be preset. Accumulator will return cleared.
Home Erase	002 021	Prints 512 spaces on the TV, with the cursor set so that the next character entry will appear at the upper left of the screen. Accumulator, B, and C are cleared at end.
TV Editor	002 045	Previously described during Operations Monitor Operation. Preset H & L regs to address of initial byte of the message prior to calling. Accumulator, B, C, E, H&L are cleared or changed when subroutine ends.
Keyboard	002 000	This subroutine loops until an MSB keypresse strobe bit goes high. The program another loop until the MSB returns to low level. Th Accumulator will have the input character.

Subroutine	Address	Operation and Comments
Seconds	001 153	Preset Accumulator to the number of seconds to elapse before returning. Accumulator, C, and D registers are cleared.

Some suggested practice programs for those new to an 8080 microprocessor:

1. Clear the screen and write an "A":

Address	Data	Explanation
006 000	315	Call the subroutine "Home Erase"
006 001	021	
006 002	002	
006 003	076	Load the Accumulator with the ASCII code for "A"
006 004	301	
006 005	315	Print the "A" on the screen
006 006	372	
006 007	000	
006 010	166	Halt and rejoice!
005 112	000	Modify the Operations Monitor to excute the above
005 113	006	program at 006 000.

Push "Reset" and then typing a "5" should run the program. Push "Reset" to return to the Operations Monitor after execution.

2. Modify the above program to print an "a".

3. Print your name.

4. Print your name in the middle of the screen using "TV Editor".

5. Print your name in the middle of the screen, flashing on and off. (Hint - Use two "Seconds" subroutines and an unconditional branch to program beginning to loop.)

6. Print only the 128 possible characters on the screen and stop, using less than 20 bytes (Hint - Load Accum, Save, Print, Restore and Modify, loop not end.)

Score : Over 100 bytes = HA!
Over 30 bytes = Fair
20-25 bytes = Good
15-19 bytes = Giant

(Can be done in 14 bytes)

PROGRAM: 8080 OPERATING SYSTEM - EROM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	000 000	303		Jump unconditional
	000 001	100		
	000 002	000		
	000 003	322		R
	000 004	345		e
	000 005	341		a
	000 006	344		d
	000 007	240		
	000 010	303		Jump unconditional
	000 011	002		
	000 012	001		
	000 013	270		8
	000 014	260		0
	000 015	270		8
	000 016	260		0
	000 017	240		
	000 020	303		Jump unconditional
	000 021	005		
	000 022	001		
	000 023	311		I
	000 024	316		N
	000 025	311		I
	000 026	324		T
	000 027	311		I
	000 030	303		Jump unconditional
	000 031	010		
	000 032	001		
	000 033	301		A
	000 034	314		L
	000 035	311		I
	000 036	332		Z
	000 037	305		E
	000 040	303		Jump unconditional
	000 041	013		
	000 042	001		
	000 043	240		
	000 044	303		C
	000 045	341		a
	000 046	363		s
	000 047	363		s
	000 050	303		Jump unconditional
	000 051	016		
	000 052	001		
	000 053	345		e
	000 054	364		t
	000 055	364		t
	000 056	345		e
	000 057	240		

PROGRAM: 8080 OPERATING SYSTEM - EROM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	000 060	303		Jump unconditional
	000 061	021		
	000 062	001		
	000 063			
	000 064			
	000 065			
	000 066			
	000 067			
	000 070	303		Jump unconditional
	000 071	024		
	000 072	001		
	000 073			
	000 074			
	000 075			
	000 076			
	000 077			
	000 100	041		Load L & H (Constant Area)
	000 101	000		
	000 102	001		
	000 103	176		Load Accum with Memory
	000 104	376		Compare A with 123
	000 105	123		
	000 106	302		Branch not equal
	000 107	120		
	000 110	000		
	000 111	054		Increment L
	000 112	176		Load Accum with Memory
	000 113	376		Compare A with 123
	000 114	123		
	000 115	312		Branch if equal
	000 116	000		
	000 117	005		
	000 120	061		Set Stack Pointer to top of Page 001
	000 121	000		
	000 122	002		
Clear screen	000 123	315		Call (Clear TV)
	000 124	343		
	000 125	000		
Write Mess.	000 126	041		Load L & H (Message Area)
	000 127	003		
	000 130	000		
	000 131	006		Load B with 006 (6 Blocks)
	000 132	006		
	000 133	016		Load C with 005 (5 Char in Block)
	000 134	005		
	000 135	176		Load Accum with Memory
	000 136	315		Call (Write Char)
	000 137	372		

PROGRAM: 8080 OPERATING SYSTEM - EROM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	000 140	000		
	000 141	054		Increment L
	000 142	015		Decrement C
	000 143	302		Jump not zero
	000 144	135		
	000 145	000		
	000 146	054		Increment L
	000 147	054		Increment L
	000 150	054		Increment L
	000 151	005		Decrement B
	000 152	302		Jump not zero
	000 153	133		
	000 154	000		
Set Read	000 155	041		Load L & H (set Read/Constant)
Speed	000 156	027		
	000 157	001		
	000 160	066		Store 041 at 001 027
*	000 161	041		(Read Speed Constant)
Disable	000 162	363		Disable interrupts
Int.	000 163	041		Load L & H (set Start Address)
Start	000 164	030		
Addr.	000 165	001		
	000 166	066		Store 000 at 1030
	000 167	000		
	000 170	054		Increment L
	000 171	066		Store 001 at 1031
	000 172	001		
Ending	000 173	054		Increment L
Addr.	000 174	066		Store 377 at 1032
	000 175	377		
	000 176	054		Increment L
	000 177	066		Store 007 at 1033
	000 200	007		
	000 201	052		Load L & H with Start Address
	000 202	030		
	000 203	001		
	000 204	315		Call (Cassette Byte Read)
	000 205	264		
	000 206	000		
	000 207	162		Load Memory from D (Store the Byte)
	000 210	176		Load A from Memory (Retrieve the Byte)
	000 211	272		Compare A with D
	000 212	312		Jump if equal (1)
	000 213	222		
	000 214	000		
	000 215	076		Load A with "."
	000 216	256		(error indicator)
	000 217	303		Jump unconditional (2) (Write TV)

PROGRAM: 8080 OPERATING SYSTEM - EROM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	000 220	227		
	000 221	000		
(1)	000 222	174		Load A with H
	000 223	346		AND A with 007
	000 224	007		
	000 225	366		OR A with 260
	000 226	260		
(2)	000 227	315		Call (TV)
	000 230	372		
	000 231	000		
	000 232	353		Exchange D & E with H & L
	000 233	052		Load H & L with Ending Address
	000 234	032		
	000 235	001		
	000 236	175		Load A with L
	000 237	273		Compare A with E
	000 240	302		Jump not equal
	000 241	250		(not end of page yet)
	000 242	000		
	000 243	174		Load A with H
	000 244	272		Compare A with D
	000 245	312		Jump if equal
	000 246	000		(go to operations monitor)
	000 247	005		
	000 250	353		Exchange D & E with H & L
	000 251	043		Increment L & H
	000 252	303		Jump unconditional
	000 253	204		(Read Another Byte)
	000 254	000		
	000 255	000		
	000 256	000		
	000 257	000		
	000 260	000		
	000 261	000		
	000 262	000		
	000 263	000		
Byte	000 264	021		Load E & D
Read	000 265	010		(Initialize for byte read)
	000 266	000		
	000 267	333		In 1
	000 270	001		
	000 271	346		AND A with 001
	000 272	001		
	000 273	302		Jump not zero
	000 274	267		(Loop until Start Bit)
	000 275	000		
	000 276	016		Load C with 003
	000 277	003		(Time delay to middle of 1st bit)

PROGRAM: 8080 OPERATING SYSTEM - EROM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	000 300	315		Call (Delay Loop)
	000 301	324		
	000 302	000		
	000 303	333		In 1
	000 304	001		
	000 305	346		AND Accumulator with 001
	000 306	001		
	000 307	202		Add D to Accum
	000 310	017		Rotate Right through Carry
	000 311	127		Load D with A
	000 312	016		Load C with 002
	000 313	002		(Time delay between bits)
	000 314	315		Call (Delay Loop)
	000 315	324		
	000 316	000		
	000 317	035		Decrement E
	000 320	302		Jump not zero
	000 321	303		
	000 322	000		
	000 323	311		Return
Delay	000 324	345		Push H & L on Stack
Loop	000 325	041		Load L & H with Read constant
	000 326	027		
	000 327	001		
	000 330	106		Load B with Memory
	000 331	341		Pop H & L off Stack
	000 332	005		Decrement B
	000 333	302		Jump not zero
	000 334	332		
	000 335	000		
	000 336	015		Decrement C
	000 337	302		Jump not zero
	000 340	324		
	000 341	000		
	000 342	311		Return
Clear	000 343	076		Load A with 377
TV	000 344	377		
	000 345	315		Call (Write Char)
	000 346	372		
	000 347	000		
	000 350	006		Load B with 000
	000 351	000		
Even	000 352	016		Load C with 004
Erase	000 353	004		
Odd	000 354	315		Call (Erase Char)
Erase	000 355	370		
(Preset	000 356	000		
B&C)	000 357	015		Decrement C

PROGRAM: 8080 OPERATING SYSTEM - EROM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	000 360	302		Jump not zero
	000 361	354		
	000 362	000		
	000 363	005		Decrement B
	000 364	302		Jump not zero
	000 365	352		
	000 366	000		
	000 367	311		Return
Erase	000 370	076		Load A with 240
Char	000 371	240		
Write	000 372	323		Out Ø
Char	000 373	000		
	000 374	257		Clear A
	000 375	323		Out Ø
	000 376	000		
	000 377	311		Return

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	001 000	123		Coded Characters to indicate Cass Read
	001 001	123		"
	001 002	303		Ø10 Restart
	001 003	*		
	001 004	*		
	001 005	303		Ø20 Restart
	001 006	*		
	001 007	*		
	001 010	303		030 Restart
	001 011	*		
	001 012	*		
	001 013	303		040 Restart
	001 014	*		
	001 015	*		
	001 016	303		050 Restart
	001 017	*		
	001 020	*		
	001 021	303		060 Restart
	001 022	*		
	001 023	*		
	001 024	303		070 Restart
	001 025	000		(Storage Dump)
	001 026	003		
	001 027	041	*	Read Speed Constant
	001 030	000	*	Low Starting Address
	001 031	001	*	High
	001 032	377	*	Low Ending Address
	001 033	007	*	High
	001 034			Reserved for Title Writer
	001 035			
	001 036			
	001 037			
Write Cassette	001 040	076		Load Accum with 001
	001 041	001		
	001 042	323		Out 1
	001 043	001		
	001 044	076		Load Accum with 005
	001 045	005		
	001 046	315		Call (Seconds)
	001 047	153		
	001 050	001		
	001 051	052		Load L&H with Start & Address
	001 052	030		
	001 053	001		
	001 054	315		Call (Cassette byte write)
	001 055	102		
	001 056	001		
	001 057	353		Exchange D&E with H&L

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	001 060	052		Load L & H with Ending Address
	001 061	032		
	001 062	001		
	001 063	175		Load A with L
	001 064	273		Compare A with E
	001 065	302		Jump not equal (Not end yet)
	001 066	075		
	001 067	001		
	001 070	174		Load A with H
	001 071	272		Compare A with D
	001 072	312		Jump if equal (end of ~~reading~~ writing)
	001 073	143		
	001 074	001		
Not end	001 075	353		Exchange D&E with H&L
yet	001 076	043		Increment H&L
	001 077	303		Jump unconditional
	001 100	054		
	001 101	001		
Byte	001 102	036		Load E with 011
Write	001 103	011		
	001 104	257		Clear Accumulator
	001 105	176		Load Accum with Memory
	001 106	027		Rotate Accum left
	001 107	323		Out 1
	001 110	001		(Bits)
	001 111	016		Load C with 002
	001 112	002		(Time delay between bits)
	001 113	315		Call (Time Delay)
	001 114	324		
	001 115	000		
EROM	001 116	006		Load B with 003
Delay	001 117	003		
Null	001 120	005		Decrement B
	001 121	302		Jump not zero
	001 122	120		
	001 123	001		
	001 124	037		Rotate Accum Right
	001 125	035		Decrement E
	001 126	302		Jump not zero
	001 127	107		
	001 130	001		
	001 131	076		Load Accum with 001
	001 132	001		
	001 133	323		Out 1
	001 134	001		(Stop Bit)
	001 135	016		Load C with 004
	001 136	004		(Time delay for stop bit)
	001 137	315		Call (Time Delay)

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	001 140	324		
	001 141	000		
	001 142	311		Return
Ending	001 143	076		Load A with 005
Tone	001 144	005		
	001 145	315		Call (Seconds)
	001 146	153		
	001 147	001		
	001 150	303		Jump unconditional
	001 151	000		(Operations Monitor)
	001 152	005		
Seconds	001 153	026		Load D with 040
	001 154	040		
	001 155	016		Load C with 100
	001 156	100		
	001 157	315		Call (Time Delay)
	001 160	324		
	001 161	000		
	001 162	025		Decrement D
	001 163	302		Jump not zero
	001 164	155		
	001 165	001		
	001 166	075		Decrement A
	001 167	302		Jump not zero
	001 170	153		
	001 171	001		
	001 172	311		Return
2K Write	001 173	041		Load L&H (Set Start Address)
	001 174	030		
	001 175	001		
	001 176	066		Store 000 at 001030
	001 177	000		
	001 200	054		Increment L&H
	001 201	066		Load 001 at 1031
	001 202	001		
	001 203	054		Increment L&H
	001 204	066		Load 377 at 1032
	001 205	377		
	001 206	054		Increment L&H
	001 207	066		Load 007 at 1033
	001 210	007		
	001 211	041		Load L&H with
	001 212	222		
	001 213	001		
	001 214	315		Call (TV Editor)
	001 215	045		
	001 216	002		
	001 217	303		Jump unconditional (Write Cassette)

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	001 220	040		
	001 221	001		
	001 222	377		(Home Erase)
	001 223	144		(Spacer)
	001 224	303		C
	001 225	341		a
	001 226	363		s
	001 227	363		s
	001 230	345		e
	001 231	364		t
	001 232	364		t
	001 233	345		e
	001 234	240		
	001 235	342		b
	001 236	345		e
	001 237	351		i
	001 240	356		n
	001 241	347		g
	001 242	240		
	001 243	367		w
	001 244	362		r
	001 245	351		i
	001 246	364		t
	001 247	364		t
	001 250	345		e
	001 251	356		n
	001 252	000		(Return)
Program-	001 253	315		Call (Erase)
mer	001 254	343		
Patch	001 255	000		
	001 256	303		Jump unconditional
	001 257	101		
	001 260	004		
	001 261	000		
	001 262	000		
	001 263	000		
	001 264	000		
	001 265	000		
	001 266	000		
	001 267	000		
	001 270	000		
	001 271	000		
	001 272	000		
	001 273	000		
	001 274	000		
	001 275	000		
	001 276	000		
	001 277	000		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	001 300	000		
	001 301	000		
	001 302	000		
	001 303	000		
	001 304	000		
	001 305	000		
	001 306	000		
	001 307	000		
	001 310	000		
	001 311	000		
	001 312	000		
	001 313	000		
	001 314	000		
	001 315	000		
	001 316	000		
	001 317	000		
	001 320			
	001 321			
	001 322			
	001 323			
	001 324			
	001 325			
	001 326			
	001 327			
	001 330			
	001 331			
	001 332			
	001 333			
	001 334			
	001 335			
	001 336			
	001 337			
	001 340			
	001 341			
	001 342			
	001 343			
	001 344			
	001 345			
	001 346			
	001 347			
	001 350			
	001 351			
	001 352			
	001 353			Reserved for Stack
	001 354			
	001 355			
	001 356			
	001 357			

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	001 360			
	001 361			
	001 362			
	001 363			
	001 364			
	001 365			Reserved for Stack
	001 366			
	001 367			
	001 370			
	001 371			
	001 372			
	001 373			
	001 374			
	001 375			
	001 376			
	001 377			
Keyboard	002 000	333		In Ø
input	002 001	000		
	002 002	376		Compare Accum with 200
	002 003	200		
	002 004	372		Jump if Minus
	002 005	000		
	002 006	002		
	002 007	365		Push A
	002 010	333		In Ø
	002 011	000		Compare Accum with 200
	002 012	376		
	002 013	200		
	002 014	362		Jump if not minus
	002 015	010		
	002 016	002		
	002 017	361		Pop A
	002 020	311		Return
512 Erase	002 021	076		Load A with 377
	002 022	377		
	002 023	315		Call (TV Writer)
	002 024	372		
	002 025	000		
	002 026	001		Load C&B with 000, 002
	002 027	000		
	002 030	002		
Eraser	002 031	315		Call (Erase Char)
	002 032	370		
	002 033	000		
	002 034	015		Decrement C
	002 035	302		Jump not zero
	002 036	031		
	002 037	002		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	002 040	005		Decrement B
	002 041	302		Jump not zero
	002 042	031		
	002 043	002		
	002 044	311		Return
TV Editor	002 045	176		Load Accum with Mem
	002 046	376		Compare Accum with 377
	002 047	377		
	002 050	302		Branch not equal (1)
	002 051	061		
	002 052	002		
	002 053	315		Call (512 Erase)
	002 054	021		
	002 055	002		
	002 056	303		Branch Uncondx (2)
	002 057	110		
	002 060	002		
	002 061	346		AND Accum with 377
	002 062	377		
	002 063	362		Jump if MSB is Ø
	002 064	074		
	002 065	002		
	002 066	315		Call (TV Writer)
	002 067	372		
	002 070	000		
	002 071	303		Jump Uncondx (2)
	002 072	110		
	002 073	002		
	002 074	376		Compare Accum with 000
	002 075	000		
	002 076	310		Return if equal
	002 077	365		Push A
	002 100	315		Call (Erase)
	002 101	370		
	002 102	000		
	002 103	361		Pop A
	002 104	075		Decrement A
	002 105	302		Jump not Zero
	002 106	077		
	002 107	002		
	002 110	043		Increment L&H
	002 111	303		Jump Uncondx
	002 112	045		
	002 113	002		
TV Writer	002 114	176		Load A with Mem
	002 115	315		Call (TV)
	002 116	372		
	002 117	000		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	002 120	043		Increment L&H
	002 121	035		Decrement E
	002 122	302		Jump not zero
	002 123	114		
	002 124	002		
	002 125	311		Return
TV Spacer	002 126	315		Call (Space)
	002 127	370		
	002 130	000		
	002 131	035		Decrement E
	002 132	302		Jump not zero
	002 133	126		
	002 134	002		
	002 135	311		Return
Octal	002 136	173		Load A with E
character	002 137	346		AND A with 300
	002 140	300		(Mask All but MSB)
	002 141	007		Rotate Left
	002 142	007		Rotate Left
	002 143	366		OR A with 260
	002 144	260		(Convert to ASCII)
	002 145	315		Call (TV)
	002 146	372		
	002 147	000		
	002 150	173		Load A with E
	002 151	346		AND A with 070
	002 152	070		(Mask all but Middle digit)
	002 153	017		Rotate Right
	002 154	017		"
	002 155	017		"
	002 156	366		OR A with 260
	002 157	260		(Convert to ASCII)
	002 160	315		Call (TV)
	002 161	372		
	002 162	000		
	002 163	173		Load A with E
	002 164	346		AND A with 007
	002 165	007		
	002 166	366		OR A with 260
	002 167	260		(Convert to ASCII)
	002 170	315		Call (TV)
	002 171	372		
	002 172	000		
	002 173	311		Return
	002 174			
	002 175			
	002 176			
	002 177			

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	002 200	377		(Erase Screen)
	002 201	010		(Space)
	002 202	324	T	
	002 203	326	V	
	002 204	240		
	002 205	323	S	
	002 206	324	T	
	002 207	317	O	
	002 210	322	R	
	002 211	301	A	
	002 212	307	G	
	002 213	305	E	
	002 214	240		
	002 215	304	D	
	002 216	325	U	
	002 217	315	M	
	002 220	320	P	
	002 221	051		
	002 222	322	R	
	002 223	345	E	
	002 224	347	G	
	002 225	351	I	
	002 226	363	S	
	002 227	364	T	
	002 230	345	E	
	002 231	362	R	
	002 232	363	S	
	002 233	272	:	
	002 234	027		
	002 235	301	A	
	002 236	003		
	002 237	302	B	
	002 240	003		
	002 241	303	C	
	002 242	003		
	002 243	304	D	
	002 244	003		
	002 245	305	E	
	002 246	003		
	002 247	310	H	
	002 250	003		
	002 251	314	L	
	002 252	006		
	002 253	000		(Return)
	002 254	045		(Space)
	002 255	306	F	
	002 256	354	l	
	002 257	341	a	

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	002 260	347	g	
	002 261	363	s	
	002 262	272	:	
	002 263	032		
	002 264	323	S	
	002 265	351	i	
	002 266	347	g	
	002 267	356	n	
	002 270	002		
	002 271	332	Z	
	002 272	345	e	
	002 273	362	r	
	002 274	357	o	
	002 275	002		
	002 276	302	B	
	002 277	303	C	
	002 300	304	D	
	002 301	002		
	002 302	320	P	
	002 303	341	a	
	002 304	362	r	
	002 305	351	i	
	002 306	364	t	
	002 307	371	y	
	002 310	002		
	002 311	303	C	
	002 312	341	a	
	002 313	362	r	
	002 314	362	r	
	002 315	371	y	
	002 316	004		
	002 317	000		(Return)
ack Edit	002 320	100		(Space)
	002 321	323	S	
	002 322	364	t	
	002 323	341	a	
	002 324	343	c	
	002 325	353	k	
	002 326	240		
	002 327	320	P	
	002 330	357	o	
	002 331	351	i	
	002 332	356	n	
	002 333	364	t	
	002 334	345	e	
	002 335	362	r	
	002 336	240		
	002 337	301	A	

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	002 340	344	d	
	002 341	344	d	
	002 342	362	r	
	002 343	345	e	
	002 344	363	s	
	002 345	363	s	
	002 346	272	:	
	002 347	002		(Space)
	002 350	000		(Return)
	002 351	042		(Space)
	002 352	322	R	
	002 353	345	e	
	002 354	364	t	
	002 355	365	u	
	002 356	362	r	
	002 357	356	n	
	002 360	240		
	002 361	301	A	
	002 362	344	d	
	002 363	344	d	
	002 364	362	r	
	002 365	345	e	
	002 366	363	s	
	002 367	363	s	
	002 370	277	?	
	002 371	002		(Space)
	002 372	000		(Return)
	002 373	320	P	
	002 374	341	a	
	002 375	347	g	
	002 376	345	e	
	002 377	000		(Return)
	003 000	365		Push A & Flags
	003 001	305		Push B&C
	003 002	325		Push D&E
	003 003	345		Push H&L
	003 004	041		Clear H&L
	003 005	000		
	003 006	000		
	003 007	071		Add Stack Printer to H&L*
	003 010	006		Load B with 012
	003 011	012		
	003 012	043		Increment L&H
	003 013	005		Decrement B
	003 014	302		Jump not zero
	003 015	012		
	003 016	003		
	003 017	345		Push H&L

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	003 020	116		Load C with Memory
	003 021	043		Increment H&L
	003 022	106		Load B with Memory
	003 023	305		Push B&C
	003 024	053		Decrement L&H
	003 025	053		"
	003 026	053		"
	003 027	053		"
	003 030	353		Exchange D&E with H&L
	003 031	041		Load L&H
	003 032	200		
	003 033	002		
	003 034	315		Call (TV Editor)
	003 035	045		
	003 036	002		
	003 037	353		Exchange D&E with H&L
	003 040	136		Load E with Memory
	003 041	315		Call ~~(clear)~~ WRITE A TO TV
	003 042	136		
	003 043	002		
	003 044	053		Decrement L&H
	003 045	026		Load D with 006
	003 046	006		
	003 047	315		Call (Space)
	003 050	370		
	003 051	000		
	003 052	053		Decrement L&H
	003 053	136		Load E with Memory
	003 054	315		Call (Char)
	003 055	136		
	003 056	002		
	003 057	025		Decrement D
	003 060	302		Jump not zero
	003 061	047		
	003 062	003		
	003 063	353		Exchange D&E with H&L
	003 064	041		Load L&H
	003 065	254		(Flags Edit)
	003 066	002		
	003 067	315		Call (TV Editor)
	003 070	045		
	003 071	002		
	003 072	006		Load B with 006
	003 073	006		
	003 074	023		Increment D&E
	003 075	005		Decrement B
	003 076	302		Jump not zero
	003 077	074		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	003 100	003		
	003 101	032		Load A with Mem, using D&E addr.
	003 102	315		Call (Flag Writer Short)
	003 103	176		
	003 104	003		
	003 105	315		Call (Flag Writer Short)
	003 106	176		
	003 107	003		
	003 110	315		Call (Flag Writer Long)
	003 111	175		
	003 112	003		
	003 113	315		Call (Flag Writer Long)
	003 114	175		
	003 115	003		
	003 116	315		Call (Flag Writer Long)
	003 117	175		
	003 120	003		
	003 121	041		Load L&H
	003 122	320		(Stack Pointer Edit)
	003 123	002		
	003 124	315		Call (TV Editor)
	003 125	045		
	003 126	002		
	003 127	006		Load B with 007
	003 130	007		
	003 131	033		Decrement D&E
	003 132	005		Decrement B
	003 133	302		Jump not zero
	003 134	131		
	003 135	003		
	003 136	353		Exchange D&E with H&L
	003 137	136		Load E with mem
	003 140	315		Call (Character)
	003 141	136		
	003 142	002		
	003 143	053		Decrement H&L
	003 144	136		Load E with Mem
	003 145	315		Call (Character)
	003 146	136		
	003 147	002		
	003 150	053		Decrement H&L
	003 151	353		Exchange D&E with H&L
	003 152	041		Load L&H
	003 153	351		(Return Edit Addr)
	003 154	002		
	003 155	315		Call (TV Editor)
	003 156	045		
	003 157	002		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	003 160	353		Exchange D&E with H&L
	003 161	136		Load E with Mem
	003 162	315		Call (Character)
	003 163	136		
	003 164	002		
	003 165	053		Decrement H&L
	003 166	136		Load E with Mem
	003 167	315		Call (Character)
	003 170	136		
	003 171	002		
	003 172	303		Jump Uncondx
	003 173	222		(Begin Storage Dumping)
	003 174	003		
Long Flag	003 175	007		Rotate Left
Short	003 176	007		Rotate Left
Flag	003 177	107		Load B with A
	003 200	346		AND A with 00k
	003 201	001		
	003 202	366		OR A with 260
	003 203	260		(Convert to ASCII)
	003 204	315		Call (TV)
	003 205	372		
	003 206	000		
	003 207	016		Load C with 005
	003 210	005		
	003 211	315		Call (Space)
	003 212	370		
	003 213	000		
	003 214	015		Decrement C
	003 215	302		Jump not zero
	003 216	211		
	003 217	003		
	003 220	170		Load A with B
	003 221	311		Return
Dump	003 222	061		Load Stack Pointer
	003 223	000		
	003 224	002		
	003 225	041		Load L&H with 000
	003 226	000		
	003 227	000		
	003 230	315		Call (Keyboard)
	003 231	000		
	003 232	002		
	003 233	346		AND A with 337
	003 234	337		
	003 235	376		Compare A with 200
	003 236	200		(look for space key)
	003 237	312		Jump if equal

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	003 240	312		
	003 241	003		
	003 242	376		Compare A with "S"
	003 243	323		(Preset Storage)
	003 244	312		Jump if equal
	003 245	264		
	003 246	003		
	003 247	376		Compare A with "P"
	003 250	320		(Program)
	003 251	312		Jump if equal
	003 252	065		
	003 253	004		
	003 254	376		Compare A with "R"
	003 255	322		(Run Reset to monitor)
	003 256	312		Jump if equal
	003 257	000		
	003 260	000		
	003 261	303		Jump Uncondx
	003 262	230		
	003 263	003		
Page Preset	003 264	315		Call (Home Erase)
	003 265	021		
	003 266	002		
	003 267	000		NOP
	002 270	041		Load H&L
	003 271	373		("Page" Edit Address)
	003 272	002		
	003 273	315		Call (TV Editor)
	003 274	045		
	003 275	002		
	003 276	041		Load H&L
	003 277	360		("Address" Edit Address)
	003 300	002		
	003 301	315		Call (TV Editor)
	003 302	045		
	003 303	002		
	003 304	315		Call (ASCII Conv)
	003 305	001		
	003 306	004		
	003 307	147		Load H with L
	003 310	056		Load L with 000
	003 311	000		
Dump	003 312	315		Call (Home Erase)
	003 313	021		
	003 314	002		
	003 315	134		Load E with H
	003 316	315		Call (Character)
	003 317	136		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	003 320	002		
	003 321	135		Load E with L
	003 322	315		Call (Character)
	003 323	136		
	003 324	002		
	003 325	036		Load E with 002
	003 326	002		
	003 327	315		Call (Spacer)
	003 330	126		
	003 331	002		
	003 332	315		Call (Line Writer)
	003 333	362		
	003 334	003		
	003 335	175		Load A with L
	003 336	376		Compare A with 140
	003 337	140		(Check for end of 1st page)
	003 340	312		Jump if equal
	003 341	230		(new page)
	003 342	003		
	003 343	376		Compare A with 300
	003 344	300		(End of 2nd page)
	003 345	312		Jump if equal
	003 346	230		(new page)
	003 347	003		
	003 350	376		Compare A with 002
	003 351	002		(End of 3rd page)
	003 352	302		Jump not equal
	003 353	315		(another line dump)
	003 354	003		
	003 355	056		Load L with 000
	003 356	000		(Reset to Page Boundary)
	003 357	303		Jump Uncondx
	003 360	230		(new page)
	003 361	003		
Line Writer	003 362	006		Load B with 006
	003 363	006		
	003 364	315		Call (Space)
	003 365	370		
	003 366	000		
	003 367	136		Load E with Mem
	003 370	315		Call (Character)
	003 371	136		
	003 372	002		
	003 373	043		Increment L&H
	003 374	005		Decrement B
	003 375	302		Jump not zero
	003 376	364		
	003 377	003		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	004 000	311		Return
ASCII Conv	004 001	315		Call (Keyboard)
	004 002	000		
	004 003	002		
ASCII Short	004 004	107		Load B with A (Save A)
	004 005	315		Call (TV)
	004 006	372		
	004 007	000		
	004 010	170		Load A with B (Restore A)
	004 011	017		Rotate Right
	004 012	017		Rotate Right
	004 013	346		AND A with 300
	004 014	300		(Mask off to MS Digit)
	004 015	117		Load C with A
	004 016	315		Call (Keyboard)
	004 017	000		
	004 020	002		
	004 021	107		Load B with A (Save A)
	004 022	315		Call (TV)
	004 023	372		
	004 024	000		
	004 025	170		Load A with B (Restore A)
	004 026	007		Rotate Left
	004 027	007		
	004 030	007		
	004 031	346		AND A with 070
	004 032	070		(Mask off to Middle digit)
	004 033	201		ADD C to A
	004 034	117		Load C with A
	004 035	315		Call (Keyboard)
	004 036	000		
	004 037	002		
	004 040	107		Load B with A (Save A)
	004 041	315		Call (TV)
	004 042	372		
	004 043	000		
	004 044	170		Load A with B (Restore A)
	004 045	346		AND A with 007
	004 046	007		(Mask to LS Digit)
	004 047	201		ADD C to A
	004 050	016		Load C with 000
	004 051	000		
	004 052	006		Load B with 200
	004 053	200		
	004 054	005		Decrement B
	004 055	302		Jump not zero
	004 056	054		
	004 057	004		

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	004 060	015		Decrement C
	004 061	302		Jump not zero
	004 062	052		
	004 063	004		
	004 064	311		Return
Keyboard Prog	004 065	061		Load Stack Pointer
	004 066	000		
	004 067	002		
	004 070	041		Load L&H
	004 071	337		(Program Edit Message)
	004 072	004		
	004 073	315		Call (TV Editor)
	004 074	045		
	004 075	002		
	004 076	041		Load L&H
	004 077	000		(Default Program Start Loc)
	004 100	006		
	004 101	134		Load E with H
	004 102	315		Call (Character)
	004 103	136		
	004 104	002		
	004 105	135		Load E with L
	004 106	315		Call (Character)
	004 107	136		
	004 110	002		
	004 111	315		Call (Keyboard)
	004 112	000		
	004 113	002		
	004 114	346		AND A with 337
	004 115	337		
	004 116	376		Compare A with "H"
	004 117	310		(Preset H address)
	004 120	302		Jump not equal
	004 121	142		
	004 122	004		
High Set	004 123	315		Call (Home Erase)
	004 124	021		
	004 125	002		
	004 126	076		Load A with "H"
	004 127	310		
	004 130	315		Call (TV)
	004 131	372		
	004 132	000		
	004 133	315		Call (ASCII Conv)
	004 134	001		
	004 135	004		
	004 136	147		Load H with A (Preset page addr)
	004 137	303		Jump Uncondx

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	004 140	253		
	004 141	001		
	004 142	376		Compare A with "L"
	004 143	314		(Preset L address)
	004 144	302		Jump not equal
	004 145	166		(next check)
	004 146	004		
Low Set	004 147	315		Call (Home Erase)
	004 150	021		
	004 151	002		
	004 152	076		Load A with "L"
	004 153	314		
	004 154	315		Call (TV)
	004 155	372		
	004 156	000		
	004 157	315		Call (ASCII Conv)
	004 160	001		
	004 161	004		
	004 162	157		Load L with A (Preset byte addr)
	004 163	303		Jump Uncondx
	004 164	253		
	004 165	001		
	004 166	376		Compare A with "S"
	004 167	323		(Go to Storage Dump?)
	004 170	312		Jump if equal
	004 171	000		(Storage Dump)
	004 172	003		
	004 173	376		Compare A with "R"
	004 174	322		(Go to Run-Reset?)
	004 075	312		Jump if equal
	004 176	000		(Run-Reset)
	004 177	000		
Make #'s only	004 200	376		Compare A with 230
	004 201	230		
	004 202	322		Jump if not less
	004 203	111		
	004 204	004		
	004 205	366		OR A with 040
	004 206	040		
	004 207	000		NOP
	004 210	000		NOP
	004 211	000		NOP
Mini Dump	004 212	365		Push A (Save A)
	004 213	006		Load B with 010
	004 214	010		
	004 215	053		Decrement L&H
	004 216	005		Decrement B
	004 217	302		Jump not zero

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	004 220	215		
	004 221	004		
	004 222	315		Call (Home Erase)
	004 223	021		
	004 224	002		
	004 225	006		Load B with 010
	004 226	010		
	004 227	134		Load E with H
	004 230	315		Call (Character)
	004 231	136		
	004 232	002		
	004 233	135		Load E with L
	004 234	315		Call (Character)
	004 235	136		
	004 236	002		
	004 237	036		Load E with 003
	004 240	003		(3 spaces)
	004 241	315		Call (Spacer)
	004 242	126		
	004 243	002		
	004 244	136		Load E with Mem
	004 245	315		Call (Character)
	004 246	136		
	004 247	002		
	004 250	036		Load E with 024
	004 251	024		
	004 252	315		Call (Spacer)
	004 253	126		
	004 254	002		
	004 255	043		Increment L&H
	004 256	005		Decrement B
	004 257	302		Jump not zero
	004 260	227		(another line)
	004 261	004		
	004 262	134		Load E with H
	004 263	315		Call (Character)
	004 264	136		
	004 265	002		
	004 266	135		Load E with L
	004 267	315		Call (Character)
	004 270	136		
	004 271	002		
	004 272	036		Load E with 003
	004 273	003		
	004 274	315		Call (Spacer)
	004 275	126		
	004 276	002		
	004 277	361		Pop A (Restore A)

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	004 300	315		Call (ASCII short)
	004 301	004		
	004 302	004		
	004 303	167		Load Mem with A
	004 304	036		Load E with 024
	004 305	024		
	004 306	315		Call (Spacer)
	004 307	126		
	004 310	002		
	004 311	043		Increment L&H
	004 312	134		Load E with H
	004 313	315		Call (Character)
	004 314	136		
	004 315	002		
	004 316	135		Load E with L
	004 317	315		Call (Character)
	004 320	136		
	004 321	002		
	004 322	036		Load E with 004
	004 323	004		
	004 324	315		Call (Spacer)
	004 325	126		
	004 326	002		
	004 327	076		Load A with "?"
	004 330	277		
	004 331	315		Call (TV)
	004 332	372		
	004 333	000		
	004 334	303		Jump Uncondx
	004 335	111		
	004 336	004		
Edit	004 337	377		Home Erase
	004 340	046		(Space)
	004 341	313	K	
	004 342	305	E	
	004 343	331	Y	
	004 344	302	B	
	004 345	317	O	
	004 346	301	A	
	004 347	322	R	
	004 350	304	D	
	004 351	240		
	004 352	320	P	
	004 353	322	R	
	004 354	317	O	
	004 355	307	G	
	004 356	322	R	
	004 357	301	A	

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	004 360	315	M	
	004 361	315	M	
	004 362	305	E	
	004 363	322	R	
	004 364	147		(Space)
	004 365	301	A	
	004 366	344	d	
	004 367	344	d	
	004 370	362	r	
	004 371	345	e	
	004 372	363	s	
	004 373	363	s	
	004 374	272	:	
	004 375	002		(Space)
	004 376	000		(Return)
	004 377			
	005 000	061		Load Stack Pointer
	005 001	000		
	005 002	002		
	005 003	373		Enable Interrupts
	005 004	041		Load L&H
	005 005	124		
	005 006	005		
	005 007	315		Call (TV Editor)
	005 010	045		
	005 011	002		
	005 012	315		Call (Keyboard)
	005 013	000		
	005 014	002		
	005 015	376		Compare A with 272
	005 016	272		
	005 017	322		Jump if not less
	005 020	012		
	005 021	005		
	005 022	376		Compare A with 260
	005 023	260		
	005 024	332		Jump if less
	005 025	012		
	005 026	005		
	005 027	365		Push A
	005 030	315		Call (Home Erase)
	005 031	021		
	005 032	002		
	005 033	361		Pop A
	005 034	007		Shift Left
	005 035	346		AND A with 136
	005 036	136		(Produce the L index)
	005 037	157		Load L with A

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	005 040	046		Load H with 005
	005 041	005		
	005 042	176		Load A with Mem
	005 043	062		Load Mem with A directly
	005 044	054		
	005 045	005		
	005 046	043		Increment L&H
	005 047	176		Load A with Mem
	005 050	062		
	005 051	055		
	005 052	005		
	005 053	303		
	005 054	*		
	005 055	*		
	005 056			
	005 057			
	005 060			
	005 061			
	005 062			
	005 063			
	005 064			
	005 065			
	005 066			
	005 067			
	005 070			
	005 071			
	005 072			
	005 073			
	005 074			
	005 075			
	005 076			
	005 077			
	005 100	000	L	Ø select address
	005 101	000	H	
	005 102	201	L	1 "
	005 103	000	H	
	005 104	173	L	2 "
	005 105	001	H	
	005 106	000	L	3 "
	005 107	003	H	
	005 110	065	L	4 "
	005 111	004	H	
	005 112	000	L	5 "
	005 113	000	H	
	005 114	000	L	6 "
	005 115	000	H	
	005 116	000	L	7 "
	005 117	000	H	

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	005 120	000	L	8 "
	005 121	000	H	
	005 122	000	L	9 "
	005 123	000	H	
	005 124	377		(Home Erase)
	005 125	011		(Spaces)
	005 126	270	8	
	005 127	260	0	
	005 130	270	8	
	005 131	260	0	
	005 132	240		
	005 133	317	O	
	005 134	320	P	
	005 135	240		
	005 136	323	S	
	005 137	331	Y	
	005 140	323	S	
	005 141	324	T	
	005 142	305	E	
	005 143	315	M	
	005 144	011		(Space)
	005 145	323	S	
	005 146	345	e	
	005 147	354	l	
	005 150	345	e	
	005 151	343	c	
	005 152	364	t	
	005 153	240		
	005 154	317	O	
	005 155	360	p	
	005 156	364	t	
	005 157	351	i	
	005 160	357	o	
	005 161	356	n	
	005 162	272	:	
	005 163	062		(Space)
	005 164	261	1	
	005 165	240		
	005 166	322	R	
	005 167	305	E	
	005 070	301	A	
	005 171	304	D	
	005 172	240		
	005 173	303	C	
	005 174	341	a	
	005 175	363	s	
	005 176	363	s	
	005 177	345	e	

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	005 200	364	t	
	005 201	364	t	
	005 202	345	e	
	005 203	021		(Space)
	005 204	262	2	
	005 205	240		
	005 206	327	W	
	005 207	322	R	
	005 210	311	I	
	005 211	324	T	
	005 212	305	E	
	005 213	240		
	005 214	303	C	
	005 215	341	a	
	005 216	363	s	
	005 217	363	s	
	005 220	345	e	
	005 221	364	t	
	005 222	364	t	
	005 223	345	e	
	005 224	020		(Space)
	005 225	263	3	
	005 226	240		
	005 227	324	T	
	005 230	326	V	
	005 231	240		
	005 232	323	S	
	005 233	364	t	
	005 234	357	o	
	005 235	362	r	
	005 236	341	a	
	005 237	347	g	
	005 240	345	e	
	005 241	240		
	005 242	304	D	
	005 243	365	u	
	005 244	355	m	
	005 245	360	p	
	005 246	017		(Space)
	005 247	264	4	
	005 250	240		
	005 251	313	K	
	005 252	345	e	
	005 253	371	y	
	005 254	342	b	
	005 255	357	o	
	005 256	341	a	
	005 257	362	r	

PROGRAM: 8080 OPERATING SYSTEM

LABEL	OCTAL ADDRESS	OCTAL CODE	MNEMONIC	COMMENTS
	005 260	344	d	
	005 261	240		
	005 262	320	P	
	005 263	362	r	
	005 264	357	o	
	005 265	347	g	
	005 266	362	r	
	005 267	341	a	
	005 270	355	m	
	005 271	016		(Space)
	005 272	000		(Return)
	005 273			
	005 274			
	005 275			
	005 276			
	005 277			
	.			
	.			(Operations Edit Area)
	.			
	005 377			
	006 000			
	.			
	.			(User's Area
	.			
	ff			

EDIT CHARACTERS for TV

	CHAR	OCTAL	BINARY*	CHAR	OCTAL	BINARY*	CHAR	OCTAL	BINARY*
	α	200	10 000 000	ø	260	10 110 000	`	340	11 100 000
	β	201	10 000 001	1	261	10 110 001	a	341	11 100 001
	γ	202	10 000 010	2	262	10 110 010	b	342	11 100 010
	δ	203	10 000 011	3	263	10 110 011	c	343	11 100 011
	ε	204	10 000 100	4	264	10 110 100	d	344	11 100 100
	ζ	205	10 000 101	5	265	10 110 101	e	345	11 100 101
	η	206	10 000 110	6	266	10 110 110	f	346	11 100 110
	θ	207	10 000 111	7	267	10 110 111	g	347	11 100 111
	ι	210	10 001 000	8	270	10 111 000	h	350	11 101 000
	κ	211	10 001 001	9	271	10 111 001	i	351	11 101 001
Linefeed	λ	212	10 001 010	:	272	10 111 010	j	352	11 101 010
	μ	213	10 001 011	;	273	10 111 011	k	353	11 101 011
	ν	214	10 001 100	<	274	10 111 100	l	354	11 101 100
Return	ξ	215	10 001 101	=	275	10 111 101	m	355	11 101 101
	ο	216	10 001 110	>	276	10 111 110	n	356	11 101 110
	π	217	10 001 111	?	277	10 111 111	o	357	11 101 111
	ρ	220	10 010 000	@	300	11 000 000	p	360	11 110 000
CTRL Q	σ	221	10 010 001	A	301	11 000 001	q	361	11 110 001
	τ	222	10 010 010	B	302	11 000 010	r	362	11 110 010
	υ	223	10 010 011	C	303	11 000 011	s	363	11 110 011
	φ	224	10 010 100	D	304	11 000 100	t	364	11 110 100
	χ	225	10 010 101	E	305	11 000 101	u	365	11 110 101
	ψ	226	10 010 110	F	306	11 000 110	v	366	11 110 110
	ω	227	10 010 111	G	307	11 000 111	w	367	11 110 111
	Ω	230	10 011 000	H	310	11 001 000	x	370	11 111 000
	√	231	10 011 001	I	311	11 001 001	y	371	11 111 001
	→	232	10 011 010	J	312	11 001 010	z	372	11 111 010
Escape	←	233	10 011 011	K	313	11 001 011	{	373	11 111 011
	↑	234	10 011 100	L	314	11 001 100	¦	374	11 111 100
	↓	235	10 011 101	M	315	11 001 101	}	375	11 111 101
	∫	236	10 011 110	N	316	11 001 110	~	376	11 111 110
	≠	237	10 011 111	O	317	11 001 111	solid	377	11 111 111
space	blank	240	10 100 000	P	320	11 010 000			
	!	241	10 100 001	Q	321	11 010 001			
	"	242	10 100 010	R	322	11 010 010			
	#	243	10 100 011	S	323	11 010 011			
	$	244	10 100 100	T	324	11 010 100			
	%	245	10 100 101	U	325	11 010 101			
	&	246	10 100 110	V	326	11 010 110			
	'	247	10 100 111	W	327	11 010 111			
	(	250	10 101 000	X	330	11 011 000			
	)	251	10 101 001	Y	331	11 011 001			
	*	252	10 101 010	Z	332	11 011 010			
	+	253	10 101 011	[	333	11 011 011			
	,	254	10 101 100	\	334	11 011 100			
	-	255	10 101 101	]	335	11 011 101			
	.	256	10 101 110	‾	336	11 011 110			
	/	257	10 101 111	_	337	11 011 111			

* Binary digits are coded 87654321 - 8th bit is always on and is used as a strobe bit.

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 8080 INSTRUCTION SET

The 8080 instruction set includes five different types of instructions:

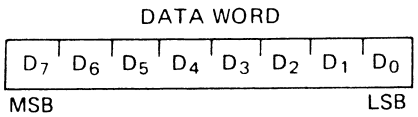
- **Data Transfer Group**—move data between registers or between memory and registers
- **Arithmetic Group** – add, subtract, increment or decrement data in registers or in memory
- **Logical Group** – AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory
- **Branch Group** – 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 8080 is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

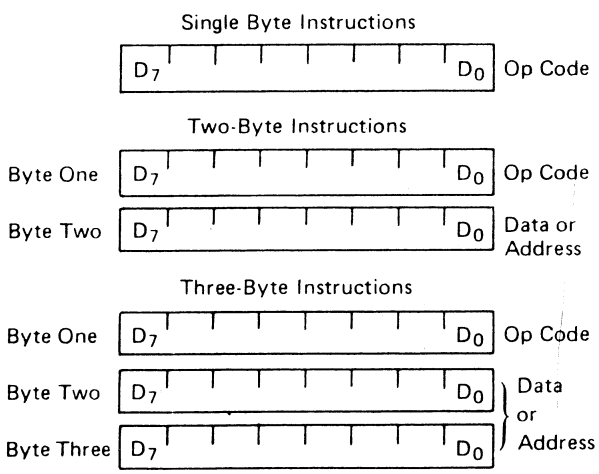
The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8080 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 8080, 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 8080 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.



Addressing Modes:

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 8080 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.

Condition Flags:

There are five condition flags associated with the execution of instructions on the 8080. 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 8080 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"
$\wedge$	Logical AND
$\forall$	Exclusive OR
$\vee$	Inclusive OR
+	Addition
−	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
$\overline{\quad}$	The one's complement (e.g., $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

Description Format:

The following pages provide a detailed description of the instruction set of the 8080. Each instruction is described in the following manner:

1. The MAC 80 assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the left side of the first line.
2. The name of the instruction is enclosed in parenthesis on the right side of the first line.
3. The next line(s) contain a symbolic description of the operation of the instruction.
4. This is followed by a narative description of the operation of the instruction.
5. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

6. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a Conditional Jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page 4-2) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

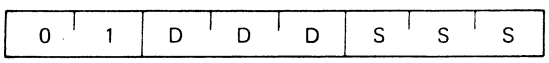
Data Transfer Group:

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.

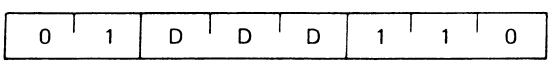


Cycles: 1
States: 5
Addressing: register
Flags: none

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.

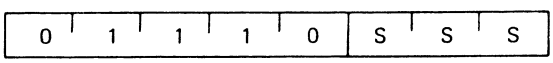


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

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.

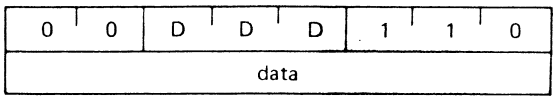


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MVI r, data (Move Immediate)

(r) ← (byte 2)

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

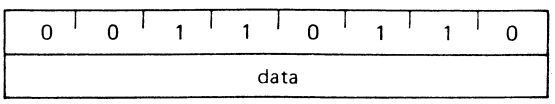


Cycles: 2
States: 7
Addressing: immediate
Flags: none

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.



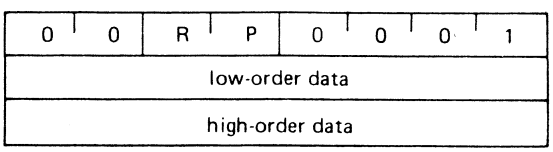
Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

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.

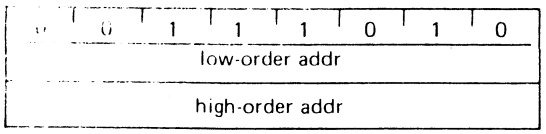


Cycles: 3
States: 10
Addressing: immediate
Flags: none

LDA addr (Load Accumulator direct)

(A) ← ((byte 3)(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.

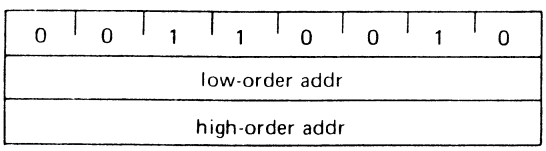


Cycles: 4
States: 13
Addressing: direct
Flags: none

STA addr (Store Accumulator direct)

((byte 3)(byte 2)) ← (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.



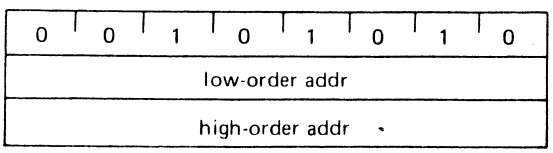
Cycles: 4
States: 13
Addressing: direct
Flags: none

LHLD addr (Load H and L direct)

(L) ← ((byte 3)(byte 2))

(H) ← ((byte 3)(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.



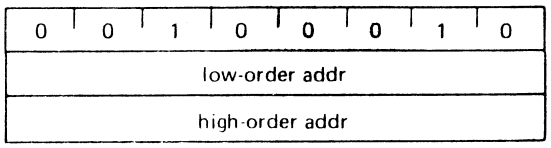
Cycles: 5
States: 16
Addressing: direct
Flags: none

SHLD addr (Store H and L direct)

((byte 3)(byte 2)) ← (L)

((byte 3)(byte 2) + 1) ← (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.

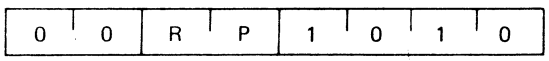


Cycles: 5
States: 16
Addressing: direct
Flags: none

LDAX rp (Load accumulator indirect)

(A) ← ((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.

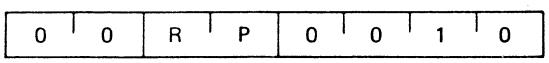


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

STAX rp (Store accumulator indirect)

((rp)) ← (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.



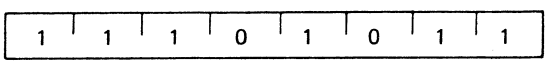
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

XCHG (Exchange H and L with D and E)

(H) ↔ (D)

(L) ↔ (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.

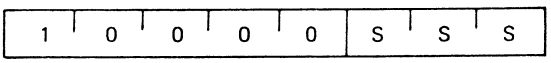
Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

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) ← (A) + (r)

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

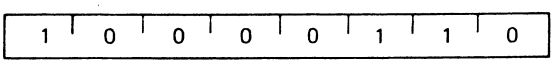


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADD M (Add memory)

(A) ← (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.

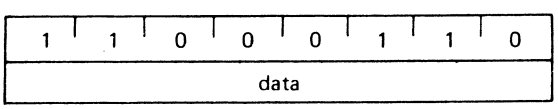


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ADI data (Add immediate)

(A) ← (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.

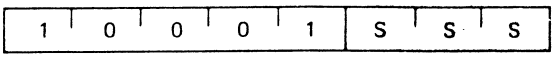


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ADC r (Add Register with carry)

(A) ← (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.

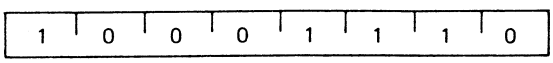


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADC M (Add memory with carry)

(A) ← (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.

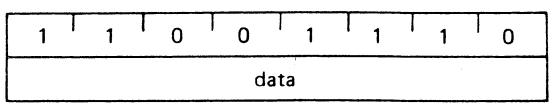


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ACI data (Add immediate with carry)

(A) ← (A) + (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.

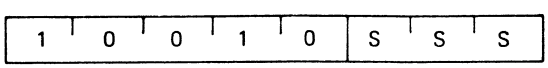


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SUB r (Subtract Register)

(A) ← (A) − (r)

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

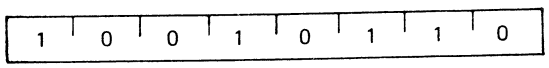


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SUB M (Subtract memory)

(A) ← (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.

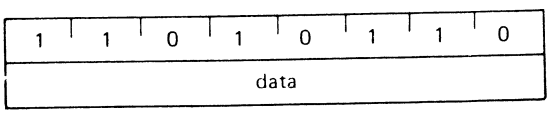


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SUI data (Subtract immediate)

(A) ← (A) – (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.

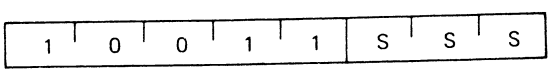


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SBB r (Subtract Register with borrow)

(A) ← (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.

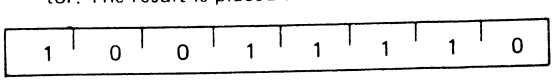


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SBB M (Subtract memory with borrow)

(A) ← (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.

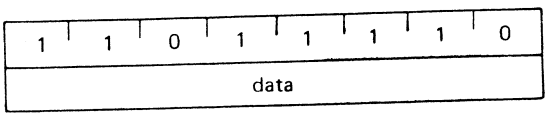


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SBI data (Subtract immediate with borrow)

(A) ← (A) – (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.

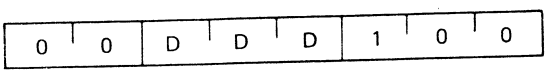


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

INR r (Increment Register)

(r) ← (r) + 1

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

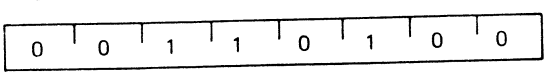


Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

INR M (Increment memory)

((H) (L)) ← ((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.

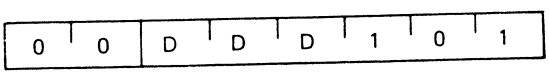


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

DCR r (Decrement Register)

(r) ← (r) – 1

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

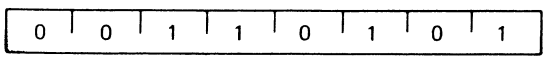


Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

DCR M (Decrement memory)

((H) (L)) ← ((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.

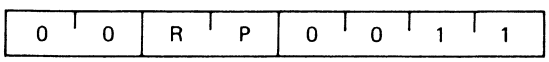


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

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

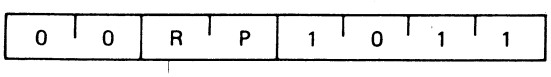


Cycles: 1
States: 5
Addressing: register
Flags: none

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) − 1

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

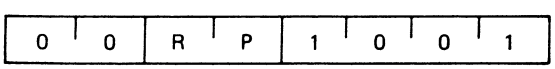


Cycles: 1
States: 5
Addressing: register
Flags: none

DAD rp (Add register pair to H and L)

(H) (L) ← (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.



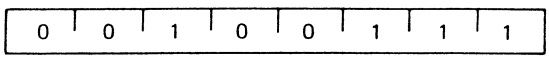
Cycles: 3
States: 10
Addressing: register
Flags: CY

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.



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

Logical Group:

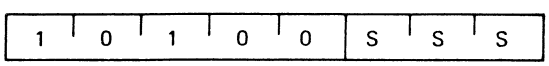
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

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) ← (A) ∧ (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.**

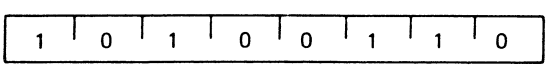


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ANA M (AND memory)

(A) ← (A) ∧ ((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.**

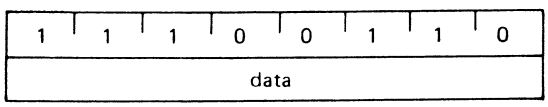


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ANI data (AND immediate)

(A) ← (A) ∧ (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 and AC flags are cleared.**

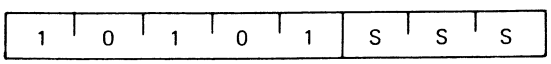


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA r (Exclusive OR Register)

(A) ← (A) ∀ (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.**

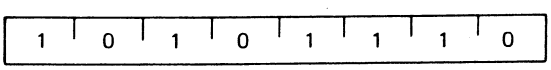


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

XRA M (Exclusive OR Memory)

(A) ← (A) ∀ ((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.**

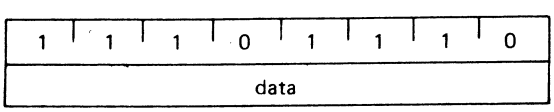


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XRI data (Exclusive OR immediate)

(A) ← (A) ∀ (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.**

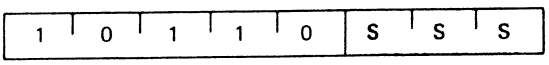


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ORA r (OR Register)

(A) ← (A) V (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.**

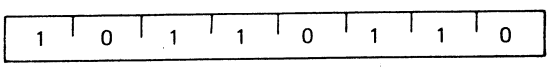


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ORA M (OR memory)

(A) ← (A) V ((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.**

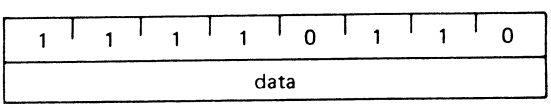


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ORI data (OR Immediate)

(A) ← (A) V (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.**

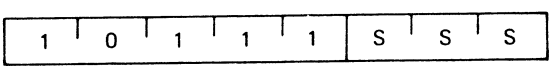


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

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).**

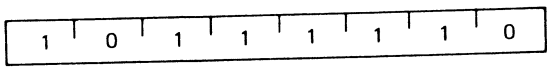


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

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)).

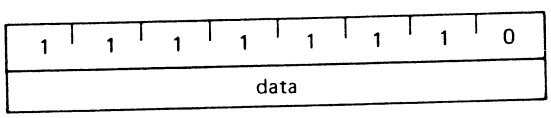


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

CPI data (Compare immediate)

(A) – (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) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).



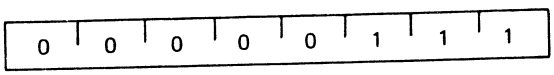
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

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.**



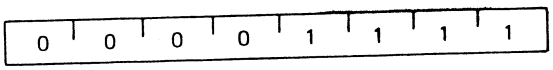
Cycles: 1
States: 4
Flags: CY

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.**



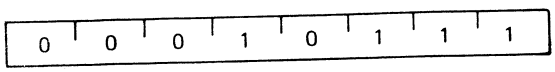
Cycles: 1
States: 4
Flags: CY

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.**



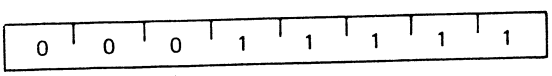
Cycles: 1
States: 4
Flags: CY

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.**

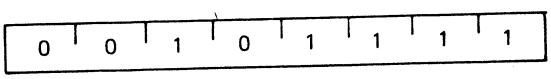


Cycles: 1
States: 4
Flags: CY

CMA (Complement accumulator)

$(A) \leftarrow (\overline{A})$

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) ← $(\overline{CY})$

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

0	0	1	1	1	1	1	1

Cycles: 1
States: 4
Flags: CY

STC (Set carry)

(CY) ← 1

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

0	0	1	1	0	1	1	1

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) ← (byte 3) (byte 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	0	0	1	1
low-order addr							
high-order addr							

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

Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (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: 3
States: 10
Addressing: immediate
Flags: none

CALL addr (Call)

((SP) – 1) ← (PCH)

((SP) – 2) ← (PCL)

(SP) ← (SP) – 2

(PC) ← (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.

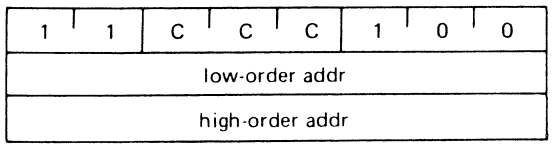
1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cycles: 5
States: 17
Addressing: immediate/reg. indirect
Flags: none

Ccondition addr (Condition call)

If (CCC),
((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← (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.

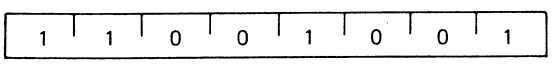


Cycles: 3/5
States: 11/17
Addressing: immediate/reg. indirect
Flags: none

RET (Return)

(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (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.

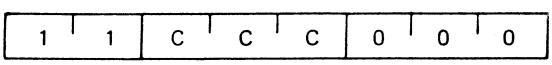


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

Rcondition (Conditional return)

If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

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

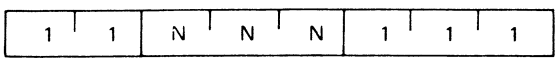


Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

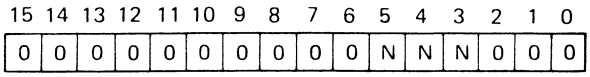
RST n (Restart)

((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← 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.



Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

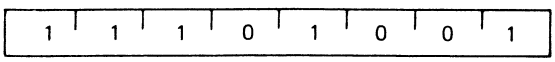


Program Counter After Restart

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

(PCH) ← (H)
(PCL) ← (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.



Cycles: 1
States: 5
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.**

FLAG WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	0	AC	0	P	1	CY

PUSH rp (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (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: 11
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

((SP) − 1) ← (A)
$((SP) - 2)_0$ ← (CY), $((SP) - 2)_1$ ← 1
$((SP) - 2)_2$ ← (P), $((SP) - 2)_3$ ← 0
$((SP) - 2)_4$ ← (AC), $((SP) - 2)_5$ ← 0
$((SP) - 2)_6$ ← (Z), $((SP) - 2)_7$ ← (S)
(SP) ← (SP) − 2

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: 11
Addressing: reg. indirect
Flags: none

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: 18
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: 5
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

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.

1	1	0	1	0	0	1	1
port							

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

EI (Enable interrupts)

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

1	1	1	1	1	0	1	1

Cycles: 1
States: 4
Flags: none

DI (Disable interrupts)

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

1	1	1	1	0	0	1	1

Cycles: 1
States: 4
Flags: none

HLT (Halt)

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

0	1	1	1	0	1	1	0

Cycles: 1
States: 7
Flags: none

NOP (No op)

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

0	0	0	0	0	0	0	0

Cycles: 1
States: 4
Flags: none

INSTRUCTION SET

Summary of Processor Instructions
By Alphabetical Order

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
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
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
ANA r	And register with A	1	0	1	0	0	S	S	S	4
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CMA	Compliment A	0	0	1	0	1	1	1	1	4
CMC	Compliment carry	0	0	1	1	1	1	1	1	4
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
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
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
HLT	Halt	0	1	1	1	0	1	1	0	7
IN	Input	1	1	0	1	1	0	1	1	10
INR M	Increment memory	0	0	1	1	0	1	0	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
LDA	Load A direct	0	0	1	1	1	0	1	0	13
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
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
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	D	D	D	1	1	0	7
MOV $_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
NOP	No-operation	0	0	0	0	0	0	0	0	4
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
OUT	Output	1	1	0	1	0	0	1	1	10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
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
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RET	Return	1	1	0	0	1	0	0	1	10
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RST	Restart	1	1	A	A	A	1	1	1	11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
STA	Store A direct	0	0	1	1	0	0	1	0	13
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
STC	Set carry	0	0	1	1	0	1	1	1	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011E – 100H – 101L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
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	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
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
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
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
ANA r	And register with A	1	0	1	0	0	S	S	S	4
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
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
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
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
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
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
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
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
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
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
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
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
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
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
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
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
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
EI	Enable Interrupts	1	1	1	1	1	0	1	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

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

8080 Programming

MOVE/LOAD/STORE Immediate

Octal	Hex	Mnemonic	Comment
076 XXX	3E XX	MVI A,XXX	Load A with XXX
006 XXX	06 XX	B,XXX	B XXX
016 XXX	0E XX	C,XXX	C XXX
026 XXX	16 XX	D,XXX	D XXX
036 XXX	1E XX	E,XXX	E XXX
046 XXX	26 XX	H,XXX	H XXX
056 XXX	2E XX	L,XXX	L XXX
066 XXX	36 XX	M,XXX	Mem XXX
001 CCC BBB	01 CC BB	LXI B,BBBCCC	Load C and B with CCC, BBB
021 EEE DDD	11 EE BB	D,DDDEEE	E D EEE, DDD
041 LLL HHH	21 LL HH	H,HHHLLL	L H LLL, HHH
061 LLL HHH	31 LL HH	LXI SP,HHHLLL	Load Stack Pointer Immediate LLL,HHH
371	F9	SPHL	Load Stack Pointer with H & L
351	E9	PCHL	Load Program Counter with H&L
072 LLL HHH	3A LL HH	LDA HHHLLL	Load A with Memory Direct LLL, HHH
012	0A	LDAX B	Load A with Memory using B&C for addr
032	1A	LDAX D	Load A with Memory using D&E for addr
062 LLL HHH	32 LL HH	STA HHHLLL	Store Memory with A Direct LLL, HHH
002	02	STAX B	Store Mem with A using B&C for addr.
022	12	STAX D	Store Mem with A using D&E for addr.
052 LLL HHH	2A LL HH	LHLD HHHLLL	Load H&L with Mem direct LLL,HHH
042 LLL HHH	22 LL HH	SHLD HHHLLL	Store Mem from H&L direct LLL,HHH
353	EB	XCHG	Exchange D&E with H&L
343	E3	XTHL	Exchange Top of Stack with H&L

8080 Programming

LOGIC OPERATIONS

Octal	Hex	Mnemonic	Comment
247	A7	ANA A	AND A with A
240	A0	B	B
241	A1	C	C
242	A2	D	D
243	A3	E	E
244	A4	H	H
245	A5	L	L
246	A6	M	Memory
346 XXX	E6 XX	ANI XXX	XXX
267	B7	ORA A	OR A with A
260	B0	B	B
261	B1	C	C
262	B2	D	D
263	B3	E	E
264	B4	H	H
265	B5	L	L
266	B6	M	Memory
366 XXX	F6 XX	ORI XXX	XXX
257	AF	XRA A	XOR A with A
250	A8	B	B
251	A9	C	C
252	AA	D	D
253	AB	E	E
254	AC	H	H
255	AD	L	L
256	AE	M	Memory
356 XXX	EE XX	XRI XXX	XXX
277	BF	CMP A	COMPARE A with A
270	B8	B	B
271	B9	C	C
272	BA	D	D
273	BB	E	E
274	BC	H	H
275	BD	L	L
276	BE	M	Memory
376 XXX	FE XX	CPI XXX	XXX

Oct	Hex	Mnemonic	Comment	Oct	Hex	Mnemonic	Comment
074	3C	INR A	Increment A	075	3D	DCR A	Decrement A
004	04	B	B	005	05	B	B
014	0C	C	C	015	0D	C	C
024	14	D	D	025	15	D	D
034	1C	E	E	035	1D	E	E
044	24	H	H	045	25	H	H
054	2C	L	L	055	2D	L	L
064	34	M	Mem	065	35	M	Mem
003	03	INX B	B&C	013	0B	DCX B	B&C
023	13	INX D	D&E	033	1B	DCX D	D&E
043	23	INX H	H&L	053	2B	DCX H	H&L
063	33	INX SP	Stk P	073	3B	DCX SP	Stk Ptr

ARITHMETIC OPERATIONS

Octal	Hex	Mnemonic	Comment
207	87	ADD A	ADD A to A
200	80	B	B
201	81	C	C
202	82	D	D
203	83	E	E
204	84	H	H
205	85	L	L
206	86	M	Memory
306 XXX	C6 XX	ADI XXX	XXX
217	8F	ADC A	ADD A & Carry to A
210	88	B	B
211	89	C	C
212	8A	D	D
213	8B	E	E
214	8C	H	H
215	8D	L	L
216	8E	M	Memory
316 XXX	CE XX	ACI XXX	XXX
227	97	SUB A	SUBTRACT A from A
220	90	B	B
221	91	C	C
222	92	D	D
223	93	E	E
224	94	H	H
225	95	L	L
226	96	M	Memory
326 XXX	D6 XX	SUI XXX	XXX
237	9F	SBB A	SUBTRACT A & Borrow from A
230	98	B	B
231	99	C	C
232	9A	D	D
233	9B	E	E
234	9C	H	H
235	9D	L	L
236	9E	M	Memory
336 XXX	DE XX	SBI XXX	XXX
011	09	DAD B	ADD B&C to H&L
031	19	DAD D	ADD D&E to H&L
051	29	DAD H	ADD H&L to H&L
071	39	DAD SP	ADD Stack Pointer to H&L

8080 Programming

BRANCHING AND SUBROUTINING

Octal	Hex	Mnemonic	Comment
303 LLL HHH	C3 LL HH	JMP HHHLLL	Jump unconditionally to LLL,HHH
312 LLL HHH	CA LL HH	JZ HHHLLL	Jump if zero/equal to LLL,HHH
332 LLL HHH	DA LL HH	JC HHHLLL	Jump if underflow/overflow or less
372 LLL HHH	FA LL HH	JM HHHLLL	Jump if MSB is 1 to LLL, HHH
352 LLL HHH	EA LL HH	JPE HHHLLL	Jump if parity is even to LLL,HHH
302 LLL HHH	C2 LL HH	JNZ HHHLLL	Jump if not zero/equal to LLL,HHH
322 LLL HHH	D2 LL HH	JNC HHHLLL	Jump if no under/overflow/not less
362 LLL HHH	F2 LL HH	JP HHHLLL	Jump if MSB is not 1 to LLL,HHH
342 LLL HHH	E2 LL HH	JPO HHHLLL	Jump if parity is not even to LLL,HHH
315 LLL HHH	CD LL HH	CALL HHHLLL	Call unconditionally to LLL,HHH
314 LLL HHH	CC LL HH	CZ HHHLLL	Call if zero/equal to LLL,HHH
334 LLL HHH	DC LL HH	CC HHHLLL	Call if under/overflow/or less
374 LLL HHH	FC LL HH	CM HHHLLL	Call if MSB is 1 to LLL,HHH
354 LLL HHH	EC LL HH	CPE HHHLLL	Call if parity is even to LLL,HHH
304 LLL HHH	C4 LL HH	CNZ HHHLLL	Call if not zero/equal to LLL,HHH
324 LLL HHH	D4 LL HH	CNC HHHLLL	Call if no under/overflow/or not less
364 LLL HHH	F4 LL HH	CP HHHLLL	Call if MSB is not 1 to LLL,HHH
344 LLL HHH	E4 LL HH	CPO HHHLLL	Call if parity is not even to LLL,HHH
311	C9	RET	Return unconditional
310	C8	RZ	Return if zero/equal
330	D8	RC	Return if under/overflow/or less
370	F8	RM	Return if MSB is 1
350	E8	RPE	Return if parity is even
300	CØ	RNZ	Return if not zero or equal
320	DØ	RNC	Return if no under/overflow/or not less
360	FØ	RP	Return if MSB is not 1
340	EØ	RPO	Return if parity is not even

RESTART, I/O, MISC

Octal	Hex	Mnemonic	Comment
307	C7	RST 000	Restart at 000 000
317	CF	010	000 010
327	D7	020	000 020
337	DF	030	000 030
347	E7	040	000 040
357	EF	050	000 050
367	F7	060	000 060
377	FF	070	000 070
007	07	RLC	Rotate Left
017	0F	RRC	Rotate Right
027	17	RAL	Rotate Left through Carry
037	1F	RAR	Rotate Right through Carry
365	F5	PUSH PSW	Push A and Flags on Stack
305	C5	B	B&C
325	D5	D	D&E
345	E5	H	H&L
361	F1	POP PSW	Pop A&Flags from Stack
301	C1	B	B&C
321	D1	D	D&E
341	E1	H	H&L
333 XXX	DB XX	IN XXX	Input from Port XXX
323 XXX	D3 XX	OUT XXX	Output to Port XXX
166	76	HLT	Halt
000	00	NOP	NOP
373	FB	EI	Enable Interrupts
363	F3	DI	Disable Interrupts
057	2F	CMA	Complement A
077	3F	CMC	Complement Carry
067	37	STC	Set Carry
047	27	DAA	Decimal Adjust