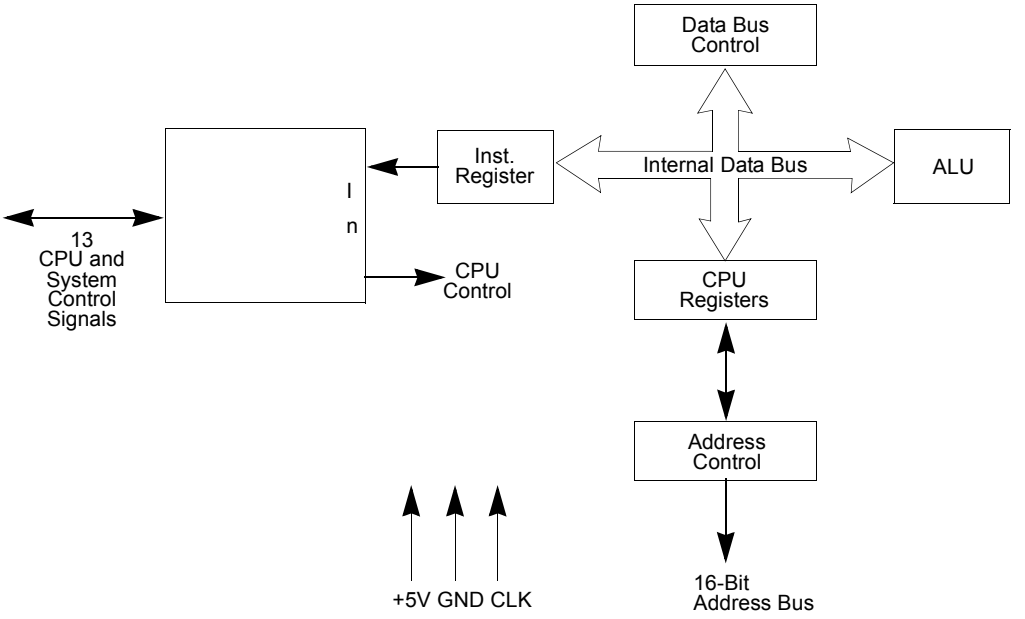


Figure 1. Z80 CPU Block Diagram

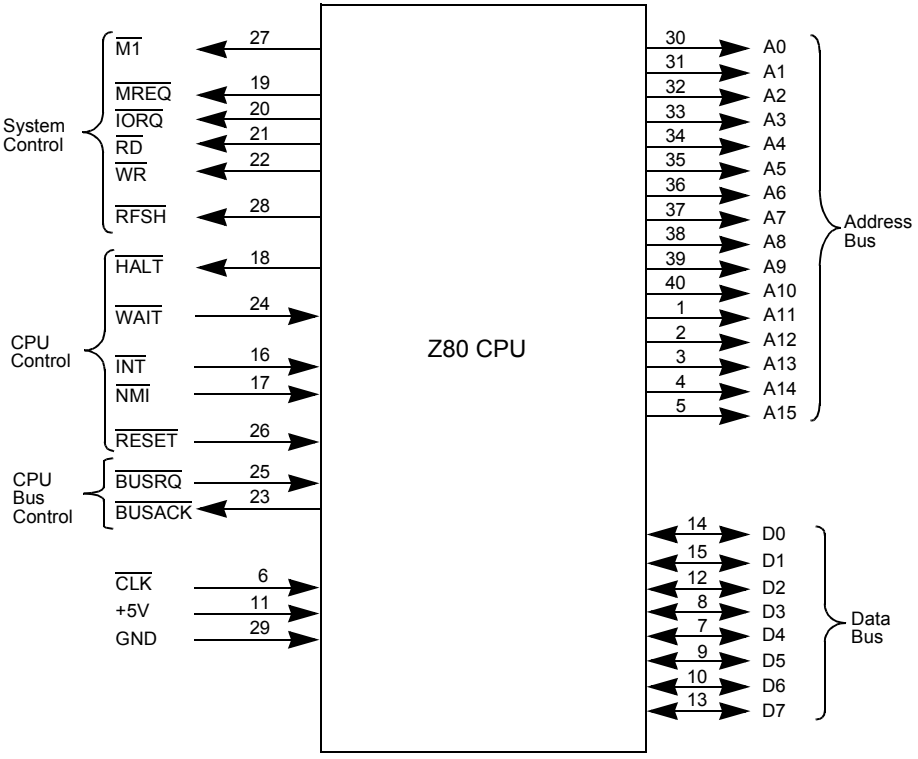


Figure 3. Z80 I/O Pin Configuration

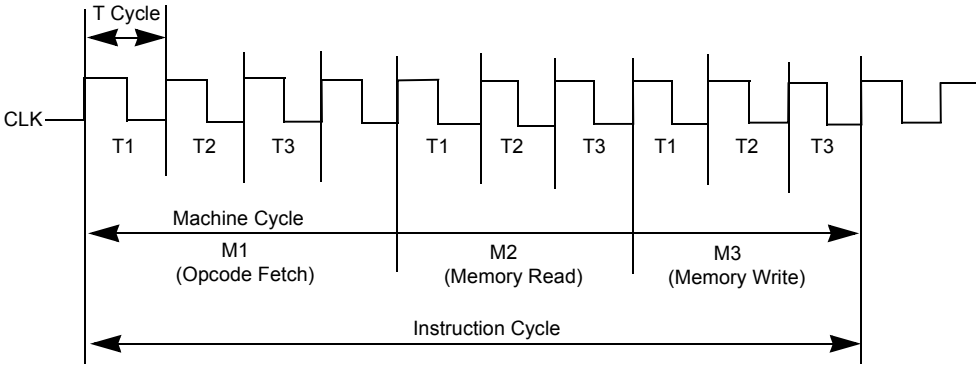


Figure 4. Basic CPU Timing Example

Instruction Fetch

Figure 5 depicts the timing during an M1 (opcode fetch) cycle. The PC is placed on the address bus at the beginning of the M1 cycle. One half clock cycle later the $\overline{MREQ}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{MREQ}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{RD}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{RD}$ and $\overline{MREQ}$ signals. Thus, the data has already been sampled by the CPU before the $\overline{RD}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time.

During T3 and T4, the lower seven bits of the address bus contain a memory refresh address and the $\overline{RFSH}$ signal becomes active tindicating that a refresh read of all dynamic memories must be accomplished. An $\overline{RD}$ signal is not generated during refresh time to prevent data from different memory

segments from being gated onto the data bus. The $\overline{MREQ}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself because the refresh address is only guaranteed to be stable during $\overline{MREQ}$ time.

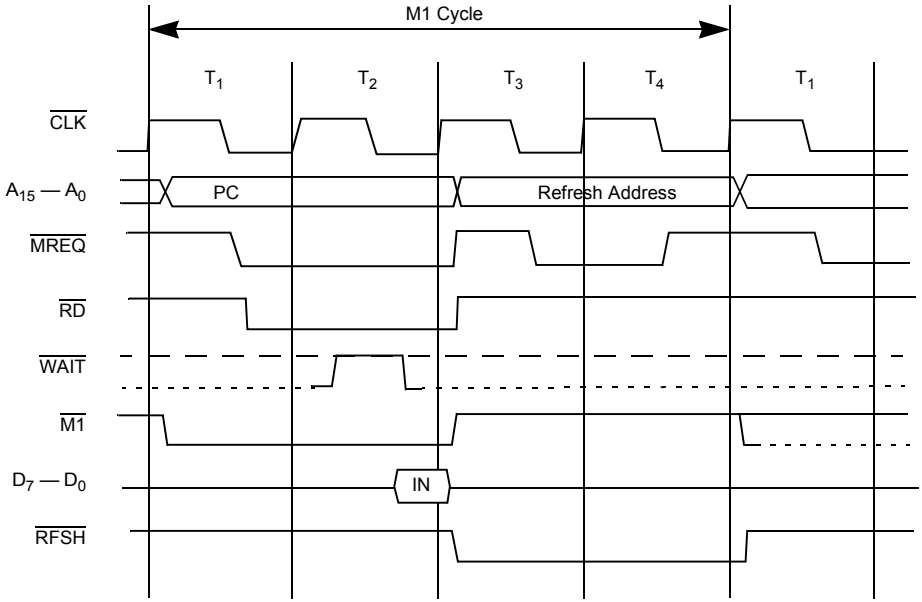


Figure 5. Instruction Op Code Fetch

Memory Read Or Write

Figure 6 illustrates the timing of memory read or write cycles other than an Op Code fetch cycle. These cycles are generally three clock periods long unless wait states are requested by the memory through the $\overline{WAIT}$ signal. The $\overline{MREQ}$ signal and the $\overline{RD}$ signal are used the same as in the fetch cycle. In a memory write cycle, the $\overline{MREQ}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{WR}$ line is active when data on the data bus is stable so that

it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore, the $\overline{WR}$ signal goes inactive one-half T state before the address and data bus contents are changed so that the overlap requirements for almost any type of semiconductor memory type is met.

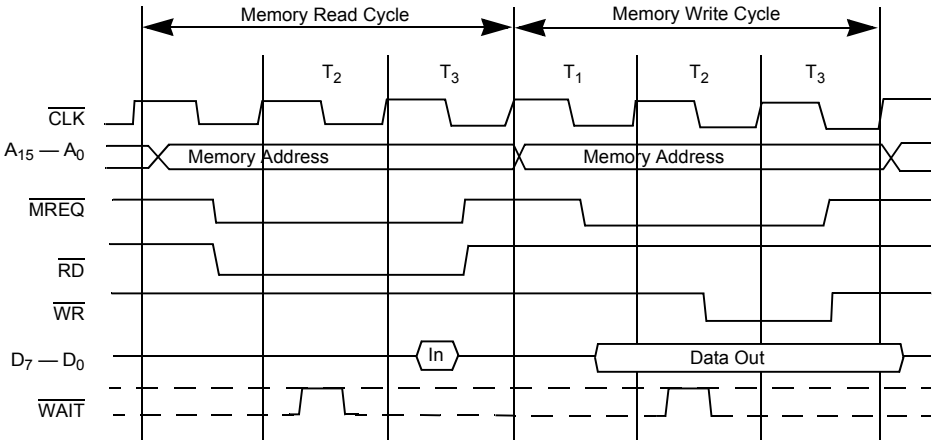


Figure 6. Memory Read or Write Cycle

Input or Output Cycles

Figure 7 illustrates an I/O read or I/O write operation. During I/O operations a single wait state is automatically inserted. The reason is that during I/O operations, the time from when the $\overline{IORQ}$ signal goes active until the CPU must sample the $\overline{WAIT}$ line is very short. Without this extra state, sufficient time does not exist for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required. Also, without this wait state, it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time, the $\overline{WAIT}$ request signal is sampled.

During a read I/O operation, the $\overline{RD}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{WR}$ line is used as a clock to the I/O port.

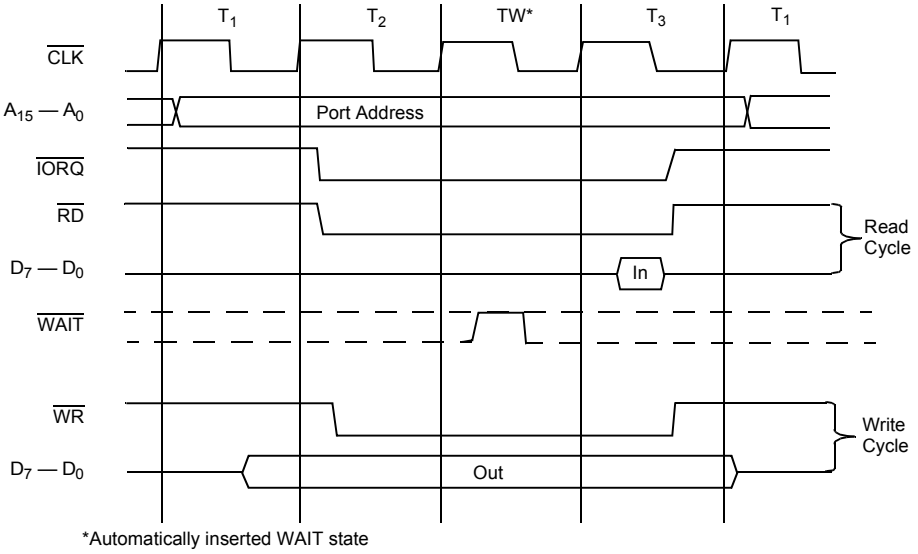


Figure 7. Input or Output Cycles

Bus Request/Acknowledge Cycle

Figure 8 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{\text{BUSREQ}}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{\text{BUSREQ}}$ signal is active, the CPU sets its address, data, and tristate control signals to the high-impedance state with the rising edge of the next clock pulse. At that time, any external device can control the buses to transfer data between memory and I/O devices. (This operation is generally known as Direct Memory Access [DMA] using cycle stealing.) The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is required. If very long DMA cycles are used, and dynamic memories are used, the external controller also performs the refresh function. This situation only occurs if very large blocks of data

are transferred under DMA control. During a bus request cycle, the CPU cannot be interrupted by either an $\overline{NMI}$ or an $\overline{INT}$ signal.

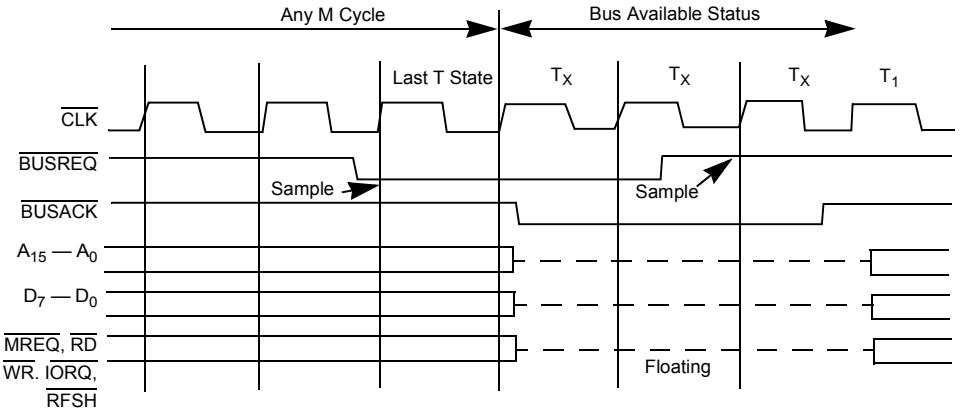


Figure 8. Bus Request/Acknowledge Cycle

Interrupt Request/Acknowledge Cycle

Figure 9 illustrates the timing associated with an interrupt cycle. The CPU samples the interrupt signal ($\overline{INT}$) with the rising edge of the last clock at the end of any instruction. The signal is not accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSREQ}$ signal is active. When the signal is accepted, a special M1 cycle is generated. During this special M1 cycle, the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which
I/O device must insert the response vector. Refer to Chapter 6 for details on how the interrupt response vector is utilized by the CPU.

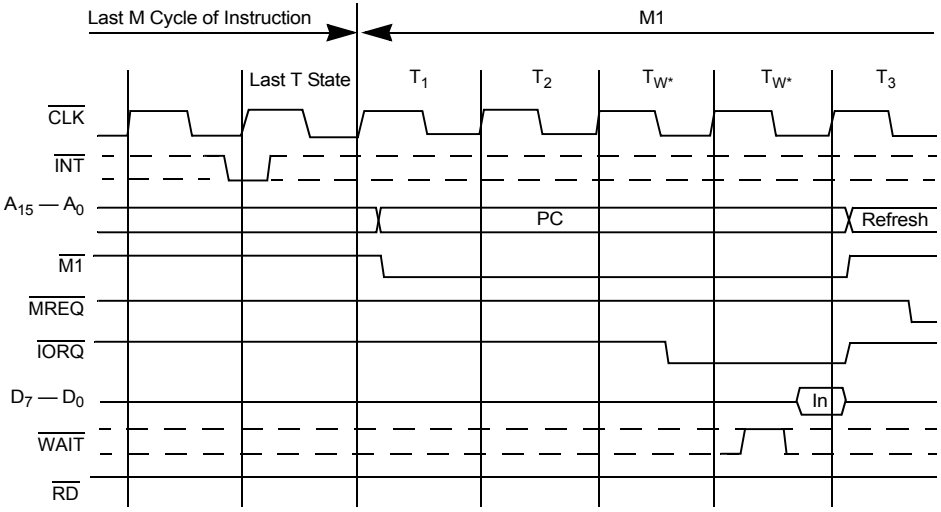


Figure 9. Interrupt Request/Acknowledge Cycle

Non-Maskable Interrupt Response

Figure 10 illustrates the request/acknowledge cycle for the non-maskable interrupt. This signal is sampled at the same time as the interrupt line, but this line takes priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non-maskable interrupt is similar to a normal memory read operation. The only difference is that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location `0066H`. The service routine for the non-maskable interrupt must begin at this location if this interrupt is used.

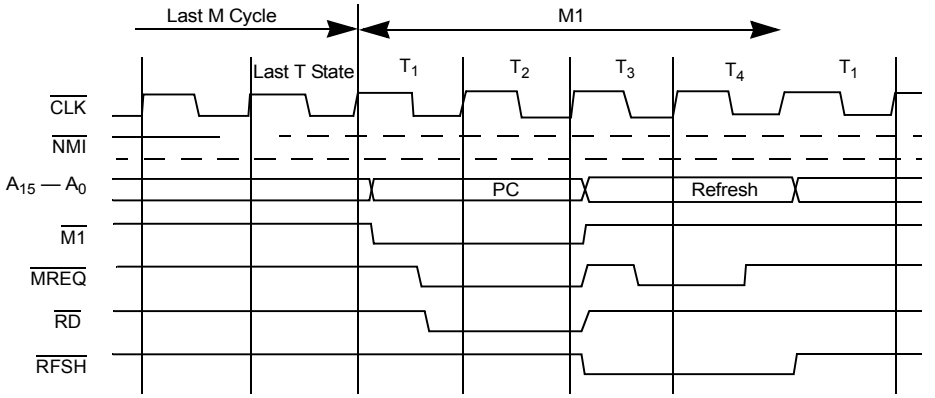


Figure 10. Non-Maskable Interrupt Request Operation

HALT Exit

Whenever a software HALT instruction is executed, the CPU executes NOPs until an interrupt is received (either a non-maskable or a maskable interrupt while the interrupt flip-flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as depicted in Figure 11. If a non-maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the HALT state is exited on the next rising clock edge. The following cycle is an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non-maskable one is acknowledged since it has highest priority. The purpose of executing NOP instructions while in the HALT state is to keep the memory refresh signals active. Each cycle in the HALT state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The HALT acknowledge signal is active during this time indicating that the processor is in the HALT state.

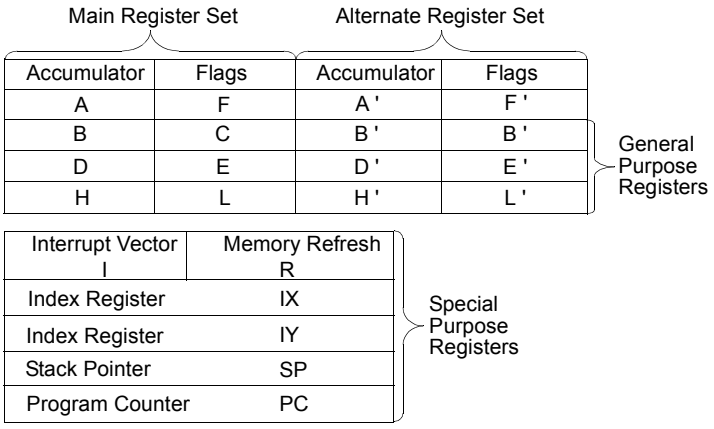


Figure 2. Z80 CPU Register Configuration

Z80 Status Indicator Flags

The flag registers (F and F') supply information to the user about the status of the Z80 at any given time. The bit positions for each flag is listed below:

7	6	5	4	3	2	1	0
S	Z	X	N	X	P/V	N	C

Symbol	Field Name
C	Carry Flag
N	Add/Subtract
P/V	Parity/Overflow Flag
H	Half Carry Flag
Z	Zero Flag
S	Sign Flag
X	Not Used

Each of the two flag registers contains 6 bits of status information that are set or cleared by CPU operations. (Bits 3 and 5 are not used.) Four of these bits (C, P/V, Z, and S) may be tested for use with conditional `JUMP`, `CALL`, or `RETURN` instructions. Two flags may not be tested (H, N) and are used for BCD arithmetic.

Carry Flag

The Carry Flag (C) is set or cleared depending on the operation performed. For `ADD` instructions that generate a Carry, and `SUB` instructions that generate a Borrow, the Carry Flag sets. The Carry Flag is reset by an `ADD` instruction that does not generate a Carry, and by a `SUB` instruction that does not generate a Borrow. This saved Carry facilitates software routines

Table 2. 8-Bit Load Group LD □□□t□□our□□

		Source															
		Implied		Register							Reg Indirect			11indexed		Ext Addr.	Imme.
Destination		I	R	A	B	C	D	E	F	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	Inn)	n
Register	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	FD 7E d	DD 7E d	FD 3A nn	FD 2E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		DD D5 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		DD DE n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		DD 1B n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		DD 1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		DD 2B n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		DD 36 n
Reg Indirect	(HL)			77	70	71	72	73	74	75							DD 78 D
	(BC)			02													
	(DE)			12													
INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXT, ADDR	(nn)			32 n n				.									
IMPLIED	I			ED 47													
	R			ED 4F													

Table 3. 16-Bit Load Group LD □□□t□□our□□, PUSH op and POP op

			Source									
			Register							Imm. Ext.	Ext. Addr.	Reg. Indir.
	Register		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
		AF										P1
		BC								01 n n	ED 4B n n	C1
		DE								11 n n	ED 5B n n	D1
		HL								21 n n	2A n n	E1
		SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
		IX								DD 21 n n	DD 2A n n	DD E1
		IY								FD 21 n n	FD 2A n n	FD E1
	EXT ADDR.	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH Instructions →	REG. IND.	(SP)	F6	C6	D6	E6		DD E6	FD E6			

NOTE: The Push & Pop instruction adjust the SP after every execution.

POP Instructions

Table 5. Block Transfer Group

Destination		Source	
Reg. Indir.	(DE)	Reg. Indir.	
		(HL)	
		(ED) A0	LDI - Load (DE) → (HL) Inc HL and DE, Dec BC
		(ED) B0	LDIR, - Load (DE) →(HL) Inc HL and DE, Dec BC, Repeat until BC = 0
		(ED) A8	LDD - Load (DE) → (HL) Inc HL and DE, Dec BC
		(ED) B8	LDDR - Load (DE) → (HL) Dec HL and DE, Dec BC, Repeat until BC = 0

Note:
Reg HL points to source
Reg DE points to destination
Reg BC is byte counter

Table 6. Block Search Group

Search Location	
Reg. Indir.	
(HL)	
(ED) A1	CPI Inc HL, Dec BC
(ED) B1	CPRI. Inc HL, Dec BC Repeat until) BC = 0 or find match
(ED) A9	WD Dec HL and BC
(ED) B9	CPDR Dec HL and BC Repeat until BC = 0 or find match

Note: HL points to location in memory to be compared with accumulator contents
BC Is byte counter

Arithmetic and Logical

Table 7 lists all the 8-bit arithmetic operations that can be performed with the accumulator, also listed are the increment (`INC`) and decrement (`DEC`)

Five general-purpose arithmetic instructions operate on the accumulator or carry flag. These five are listed in Table 8. The decimal adjust instruction can adjust for subtraction as well as addition, making BCD arithmetic operations simple. Note that to allow for this operation the flag N is used. This flag is set if the last arithmetic operation was a subtract. The negate accumulator (NEG) instruction forms the two's complement of the number in the accumulator. Finally, notice that a reset carry instruction is not included in the Z80 because this operation can be easily achieved through other instructions such as a logical AND of the accumulator with itself.

Table 9 lists all the 16-bit arithmetic operations between 16-bit registers. There are five groups of instructions including add with carry and subtract with carry. ADC and SBC affect all the flags. These two groups simplify address calculation operations or other 16-bit arithmetic operations.

Table 7. 8-Bit Arithmetic and Logic

	Source										
	Register Addressing							Reg Indir.	Indexed		Immed.
	A	B	C	D	E	F	L	(HL)	(IX+d)	(IY+d)	n
ADD	87	80	81	82	83	84	85	88	DD 86 d	FD 86 d	C6 n
ADD W CARRY ADC	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT SUB	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARR SBC	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
AND	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n

Table 7. 8-Bit Arithmetic and Logic

	Source										
	Register Addressing							Reg Indir.	Indexed		Immed.
OR	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE CP	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT INC	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT DEC	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Table 8. General-Purpose AF Operation

Decimal Adjust Acc, DAA	27
Complement Acc, CPL	2F
Negate Acc, NEG (2's complement	ED 44
Complement Carry Flag, CCF	3F
Set Carry Flag, SCF	37

Table 9. 16-Bit Arithmetic

Source					
BC	DE	HL	SP	IX	IY

Table 9. 16-Bit Arithmetic

			Source					
Destination		HL	09	19	29	39		
	ADD	IX	DD 09	DD 19		DD 39	DD 29	
		IY	FD 09	FD 19		FD 39		FD 29
	ADD with carry and set flags ADC	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	SUB with carry and set flags SBC	HL	ED 42	ED 52	ED 62	ED 72		
	Increment INC		03	13	23	33	DD 23	FD 23
	Decrement DEC		DB	1B	2B	3B	DD 2B	FD 2B

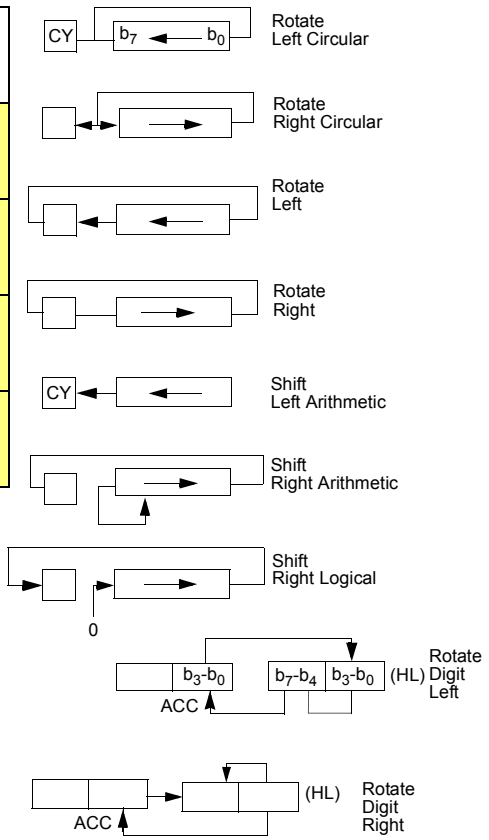
Rotate and Shift

A major feature of the Z80 is to rotate or shift data in the accumulator, any general-purpose register, or any memory location. All the rotate and shift Op Codes are depicted in Figure 10. Also included in the Z80 are arithmetic and logical shift operations. These operations are useful in a wide range of applications including integer multiplication and division. Two BCD digit rotate instructions (RRD and RLD) allow a digit in the accumulator to be rotated with the two digits in a memory location pointed to by register pair HL (See Figure 10). These instructions allow for efficient BCD arithmetic.

Table 10. Rotates and Shifts

Type of Rotate Shift	Source A	B	C	D	E	F	L	(HL)	(IX+d)	(IY+d)
RCL	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 06	CB 0E	DD CB d 06	FD CB d 06
RRC	CB 0F	CB 08	CB 09	CB 0A	CB 06	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
								ED 6F		
								ED 67		

	A
RLCA	D7
RRCA	0F
RLA	17
RRA	1F



Bit Manipulation

The ability to set, reset, and test individual bits in a register or memory location is needed in almost every program. These bits may be flags in a general-purpose software routine, indications of external control

Table 11. Bit Manipulation Group (Continued)

		Register Addressing							Reg. Indir.	Indexed	
Test Bit	0	C8 47	C8 40	C8 41	C8 42	C8 43	C8 44	C8 45	C8 46	C8 d	C8 d
	1	C8 4F	C8 48	C8 49	C8 4A	C8 48	C8 4C	C8 4D	C8 4E	46 DD C8 d 4E	46 FD C8 d 4E
	2	C8 57	C8 50	C8 51	C8 52	C8 53	C8 54	C8 55	C8 56	DD C8 d 56	FD C8 d 56
	3	C8 5F	C8 58	C8 59	C8 5A	C8 5B	C8 5C	C8 5D	C8 5E	DD C8 d 46	FD C8 d 46
	4	C8 67	C8 60	C8 61	C8 62	C8 63	C8 64	C8 65	C8 66	DD C8 d 66	FD C8 d 66
	5	C8 6F	C8 68	C8 69	C8 6A	C8 68	C8 6C	C8 6D	C8 6E	DD C8 d 6E	FD C8 d 6E
	6	C8 77	C8 70	C8 71	C8 72	C8 73	C8 74	C8 75	C8 76	DD C8 d 76	FD C8 d 76
	7	C8 7F	C8 78	C8 79	C8 7A	C8 78	C8 7C	CS 7D	C8 7E	DD C8 d 46	DD C8 d 46

Table 11. Bit Manipulation Group (Continued)

		Register Addressing							Reg. Indir.	Indexed	
Rest Bit RES	0	C8 87	C8 80	C8 81	C8 82	C8 83	C8 84	C8 85	C8 86	DD C8 d 86	FD C8 d 86
	1	C8 8F	C8 88	C8 89	C8 8A	C8 88	C8 8C	C8 8D	C8 8E	DD C8 d 8E	FD C8 d 8E
	2	C8 97	C8 90	CS 91	C8 92	C8 93	C8 94	C8 95	C8 96	DD C8 d 96	FD C8 d 96
	3	C8 9F	C8 98	C8 99	C8 9A	CS 98	C8 90	C8 90	C8 9E	DD C8 d 9E	FD C8 d 9E
	4	C8 A7	C8 AO	C8 AI	C8 A2	C6 A3	C8 A4	C8 A5	C8 A6	DD C8 d A6	FD C8 d A6
	5	C8 AF	C8 A8	C8 A9	C8 AA	08 AB	C8 AC	C8 AD	C8 AE	DD C8 d AE	FD C8 d AE
	6	C8 B7	C8 B0	C8 B1	C8 82	C8 B3	C8 B4	C8 B5	C8 B6	DD C8 d B6	FD C8 d B6
	7	C8 BF	C8 B8	C8 89	C8 8A	C8 B8	C8 8C	C8 BD	C8 9E	DD C8 d BE	DD C8 d BE

Table 11. Bit Manipulation Group (Continued)

		Register Addressing							Reg. Indir.	Indexed	
Set Bit SET	0	C8 C7	C8 C0	C8 C1	C8 C2	C8 C3	C8 C4	C8 C5	C8 C6	DD C8 d C6	FD C8 d C6
	1	C8 CF	C8 C8	C8 C9	C8 CA	C8 C8	C8 CC	C8 CD	C8 CE	DD C8 d CE	FD C8 d CE
	2	C8 D7	C8 DO	C8 D1	C8 D2	C8 D3	C8 D4	C8 DS	C8 D6	DD C8 d D6	FD C8 d D6
	3	C8 DF	C8 D8	C8 09	C8 DA	C8 DS	C8 DC	C8 DD	C8 DE	DD C8 d DE	FD C8 d DE
	4	C8 E7	C8 E0	C8 E1	C8 E2	C8 E3	C8 E4	C8 E5	C8 E6	DD C8 d E6	FD C8 d E6
	5	C8 EF	C8 E8	C8 E9	C8 EA	C8 EB	C8 EC	C8 ED	C8 EE	DD C8 d EE	FD C8 d EE
	6	C8 F7	C8 FO	C8 F1	C8 F2	C8 F3	C8 F4	C8 FS	C8 F6	DD C8 d F6	FD C8 d F6
	7	C8 FF	C8 F8	C8 F9	C8 FA	C8 FB	C8 FC	C8 FD	C8 FE	DD C8 d FE	FD C8 d FE

Table 12. Jump, Call, and Return Group

			Condition									
			Un-Cond.	Carry	Non Carry	Zero	Non Zero	Parity Even	Parity Odd	Sign Neg	Sign Pos	Reg B≠0
JUMP JP	IMMED. EXT.	nn	C3 n n	D8 n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP JR	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP JP	Register INDIR.	(HL)	EB									
		(IX)	DD E9									
		(IY)	FD E9									
CALL	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
Decrement B, Jump If Non Zero DJNZ	RELATIVE	PC+e										10 e-2
Return RE	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
Return From INT RETI			ED 4D									
Return From Non Maskable INT RETN			ED 45									

The instruction `DJNZ` is used to facilitate program loop control. This two byte, relative jump instruction decrements the B register and the jump occurs if the B register has not been decremented to zero. The relative displacement is expressed as a signed two's complement number. A simple example of its use is:

Address	Instruction	Comments
N, N+1	LD B, 7	: set B register to count of 7
N+2 to N+9	(Perform a sequence of instructions)	: loop to be performed 7 times
N+10,N+11	DJNZ -8	: to jump from N+12 to N+2
N + 12	(Next Instruction)	

Table 13 lists the eight Op Codes for the restart instruction. This instruction is a single byte call to any of the eight addresses listed. The simple mnemonic for these eight calls is also listed. This instruction is useful for frequently-used routines because memory consumption is minimized.

Table 13. Restart Group

		Op Code	
CALL Address	0000H	C7	RST 0
	0008H	CF	RST 8
	0010H	D7	RST 16
	0018H	DF	RST 24
	0020H	E7	RST 32
	0028H	EF	RST 40
	0030H	F7	RST 48
	0038H	FF	RST 56

Input/Output

The Z80 has an extensive set of input and output instructions as shown in Table 14 and Table 15. The addressing of the input or output device can be either absolute or register indirect, using the C register. In the register indirect addressing mode, data can be transferred between the I/O devices and any of the internal registers. In addition, eight block transfer instructions have been implemented. These instructions are similar to the memory block transfers except that they use register pair HL for a pointer to the memory source (output commands) or destination (input commands) while register B is used as a byte counter. Register C holds the address of the port for which the input or output command is required. Because register B is eight bits in length, the I/O block transfer command handles up to 256 bytes.

In the instructions `IN A`, and `OUT n, A`, the I/O device address n appears in the lower half of the address bus (A7-A0) while the accumulator content

Table 14. Input Group

				Immed.	Register Indir.	
				(n)	(c)	
Input Destination	Input IN	Register Address	A	DB n	ED 7B	
			B		ED 40	
			C		ED 48	
			D		ED 50	
			E		ED 58	
			H		ED 60	
			L		ED 68	
	INI - input & inc HL, Dec B	Register Indir	(HL)		ED A2	Block Input Commands
	INIR - INP, Inc HL, Dec B, repeat IF B≠0				ED B2	
	IND - input & Inc Dec HL, Dec B				ED AA	
	INDR - input, Dec HL, Dec B, repeat IF B≠0				ED BA	

Table 15. Output group

			Source								
			Register							Register Indir.	
			A	B	C	D	E	H	L	(HL)	
OUT	Immed.	(n)	D3 n								
	Reg Ind.	(c)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69		
OUT - output inc HL, dec B										ED A3	Block Output Command
OUT - output dec B, repeat if B≠0										ED B3	
OUT - output dec HL and B										ED AB	
OUTDR - output, dec HL and B, repeat IF B≠0										ED BB	
	Port Destination Address										

Table 16. Miscellaneous CPU Control

NOP	00	
HALT	76	
Disable INT (EI)	F3	
Enable INT (EI)	FB	
Set INT mode 0 IM0	ED 46	8080A mode
Set INT mode 1 IM1	ED 56	Call to location 0038H
Set INT mode 2 IM2	ED 5E	indirect call using register I and B bits from INTER device as a pointer