

NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF INDUSTRIAL TECHNOLOGY

DEPARTMENT OF ELECTRONIC ENGINEERING

BACHELOR OF ENGINEERING (HONS) DEGREE

Supplementary Examination July-August 2011

TEE 2112

MICROPROCESSORS

Duration of Examination – 3 Hours

INSTRUCTIONS TO CANDIDATES

1. This question paper consists of 3 printed papers and 7 questions.
 2. Answer any FIVE questions only.
 3. Each question carries 20 marks.
 4. Show your steps clearly in any calculation.
 5. Start the answers for each question on a fresh page.
 6. All programming questions refer to the Z80 microprocessor.
-

Question 1

- a) Explain the difference between a jump and call instruction? [4]
- b) Explain what you think the following mnemonic and expressions mean.
 - i. ADD A,n
 - ii. ADC A,(IX + n)
 - iii. SUB A,r
 - iv. SBC A,m
 - v. INC (HL)
 - vi. DEC BC[12]
- c) What are the basic flags incorporated into a Z80 microprocessor. State the purpose of each flag. [8]

Question 2

- a) Identify the addressing modes used in the following instructions with explanation.
- i. LD A,B
 - ii. LD A,(BC)
 - iii. ADD A,38H
 - iv. CPL
 - v. DEC IX
- [10]
- b) Describe the three maskable interrupt modes for the Z80 CPU
- [10]

Question 3

Design a Z80 program using assembly mnemonics and write the code which simulate a traffic light intersection with equal times for red and green of five seconds and two seconds for amber. Show block diagram, flowchart and program code.

[20]

Question 4

- a) Describe the following and state their roles in a microprocessor.
- i. Arithmetic logic unit.
 - ii. General purpose registers.
 - iii. Program counter.
 - iv. Flag/Status register.
 - v. Microprocessor control logic.
 - vi. Accumulator.
 - vii. Stack Pointer.
- [7]
- b) Discuss the advantage of having alternate register sets in a microprocessor.
- [3]
- c) A microprocessor uses a 16 bit address bus and an 8-bit data bus.
- i. What is the maximum size of memory that can be accessed by the microprocessor? [3]
 - ii. What is the length of a data word? [2]
- d) Using two's compliment, carry out the mathematical process: $35 - 67$ [5]

Question 5

- a) The following program segment is executing in a microprocessor based system. Draw a table showing the contents of the memory locations 0070H, 0071H, 0072H and 0073H for each step of the program. Assume the locations contain zeros at the start.

```
LD    A, 3FH
LD    (0070H), A
LD    IX, 0070H
LD    B, (0070H)
LD    (IX+03H), B
INC   IX
LD    (0072H), IX
```

[12]

- b)i. What is an interrupt and for what purposes can it be used in a microprocessor? [4]
- ii. What are the steps which occur when an interrupt happens? Briefly explain what function each one performs. [4]

Question 6

Design a program to add two blocks of data element by element and store the results in another block. Each block has 8 bytes stored in consecutive memory locations. Suggest starting addresses for each block. Produce block diagram, flowchart and write the Z80 assembly language. [20]

Question 7

Design a Z80 microprocessor program to search for a given character in a block memory. If the character is found it should be written to an output port, else the number 00H should be written to the output port. Assume the character is 'A' with ASCII code 41H. Assume the block memory starts at address 8000H and has 400H characters in memory. Show the flowchart and the assembly code. [20]

Z80 Instruction Set

8 Bit Transfer Instructions

Z80 Mnemonic	Machine Code	Operation				
LD A,A	7F	A <- A	LD H,A	67		H <- A
LD A,B	78	A <- B	LD H,B	60		H <- B
LD A,C	79	A <- C	LD H,C	61		H <- C
LD A,D	7A	A <- D	LD H,D	62		H <- D
LD A,E	7B	A <- E	LD H,E	63		H <- E
LD A,H	7C	A <- H	LD H,H	64		H <- H
LD A,L	7D	A <- L	LD H,L	65		H <- L
LD A,(HL)	7E	A <- (HL)	LD H,(HL)	66		H <- (HL)
LD A,(BC)	0A	A <- (BC)	LD H,(IX+index)	DD66index		H <- (IX+index)
LD A,(DE)	1A	A <- (DE)	LD H,(IY+index)	FD66index		H <- (IY+index)
LD A,(word)	3Aword	A <- (word)	LD L,A	6F		L <- A
LD A,(IX+index)	DD7Eindex	A <- (IX+index)	LD L,B	68		L <- B
LD A,(IY+index)	FD7Eindex	A <- (IY+index)	LD L,C	69		L <- C
LD A,I	ED57	A <- Interrupt Page	LD L,D	6A		L <- D
LD B,A	47	B <- A	LD L,E	6B		L <- E
LD B,B	40	B <- B	LD L,H	6C		L <- H
LD B,C	41	B <- C	LD L,L	6D		L <- L
LD B,D	42	B <- D	LD L,(HL)	6E		L <- (HL)
LD B,E	43	B <- E	LD L,(IX+index)	DD6Eindex		L <- (IX+index)
LD B,H	44	B <- H	LD L,(IY+index)	FD6Eindex		L <- (IY+index)
LD B,L	45	B <- L	LD (HL),A	77		(HL) <- A
LD B,(HL)	46	B <- (HL)	LD (HL),B	70		(HL) <- B
LD B,(IX+index)	DD46index	B <- (IX+index)	LD (HL),C	71		(HL) <- C
LD B,(IY+index)	FD46index	B <- (IY+index)	LD (HL),D	72		(HL) <- D
LD C,A	4F	C <- A	LD (HL),E	73		(HL) <- E
LD C,B	48	C <- B	LD (HL),H	74		(HL) <- H
LD C,C	49	C <- C	LD (HL),L	75		(HL) <- L
LD C,D	4A	C <- D	LD (IX+index),A	DD77index		(IX+index) <- A
LD C,E	4B	C <- E	LD (IX+index),B	DD70index		(IX+index) <- B
LD C,H	4C	C <- H	LD (IX+index),C	DD71index		(IX+index) <- C
LD C,L	4D	C <- L	LD (IX+index),D	DD72index		(IX+index) <- D
LD C,(HL)	4E	C <- (HL)	LD (IX+index),E	DD73index		(IX+index) <- E
LD C,(IX+index)	DD4Eindex	C <- (IX+index)	LD (IX+index),H	DD74index		(IX+index) <- H
LD C,(IY+index)	FD4Eindex	C <- (IY+index)	LD (IX+index),L	DD75index		(IX+index) <- L
LD D,A	57	D <- A	LD (IX+index),byte	DD76indexbyte		(IX+index) <- byte
LD D,B	50	D <- B	LD (IY+index),A	FD77index		(IY+index) <- A
LD D,C	51	D <- C	LD (IY+index),B	FD70index		(IY+index) <- B
LD D,D	52	D <- D	LD (IY+index),C	FD71index		(IY+index) <- C
LD D,E	53	D <- E	LD (IY+index),D	FD72index		(IY+index) <- D
LD D,H	54	D <- H	LD (IY+index),E	FD73index		(IY+index) <- E
LD D,L	55	D <- L	LD (IY+index),H	FD74index		(IY+index) <- H
LD D,(HL)	56	D <- (HL)	LD (IY+index),L	FD75index		(IY+index) <- L
LD D,(IX+index)	DD56index	D <- (IX+index)	LD (IY+index),byte	FD76indexbyte		(IY+index) <- byte
LD D,(IY+index)	FD56index	D <- (IY+index)	LD A,byte	3Ebyte		A <- byte
LD E,A	5F	E <- A	LD B,byte	06byte		B <- byte
LD E,B	58	E <- B	LD C,byte	0Ebyte		C <- byte
LD E,C	59	E <- C	LD D,byte	16byte		D <- byte
LD E,D	5A	E <- D	LD E,byte	1Ebyte		E <- byte
LD E,E	5B	E <- E	LD H,byte	26byte		H <- byte
LD E,H	5C	E <- H	LD L,byte	2Ebyte		L <- byte
LD E,L	5D	E <- L	LD (HL),byte	36byte		(HL) <- byte
LD E,(HL)	5E	E <- (HL)	LD (IX+index),byte	DD36index byte		(IX+index) <- byte
LD E,(IX+index)	DD5Eindex	E <- (IX+index)	LD (IY+index),byte	FD36index byte		(IY+index) <- byte
LD E,(IY+index)	FD5Eindex	E <- (IY+index)	LD (BC),A	02		(BC) <- A
			LD (DE),A	12		(DE) <- A
			LD (word),A	32word		(word) <- A

16 Bit Transfer Instructions

Z80 Mnemonic	Machine Code	Operation
LD BC,word	01word	BC <- word
LD DE,word	11word	DE <- word
LD HL,word	21word	HL <- word
LD SP,word	31word	SP <- word
LD IX,word	DD21word	IX <- word
LD IY,word	FD21word	IY <- word
LD HL,(word)	2Aword	HL <- (word)
LD BC,(word)	ED4Bword	BC <- (word)
LD DE,(word)	ED5Bword	DE <- (word)
LD HL,(word)	ED6Bword	HL <- (word)
LD SP,(word)	ED7Bword	SP <- (word)
LD IX,(word)	DD2Aword	IX <- (word)
LD IY,(word)	FD2Aword	IY <- (word)
LD (word),HL	22word	(word) <- HL
LD (word),BC	ED43word	(word) <- BC
LD (word),DE	ED53word	(word) <- DE
LD (word),HL	ED6Bword	(word) <- HL
LD (word),IX	DD22word	(word) <- IX
LD (word),IY	DD22word	(word) <- IY
LD (word),SP	ED73word	(word) <- SP
LD SP,HL	F9	SP <- HL
LD SP,IX	DDF9	SP <- IX
LD SP,IY	FDF9	SP <- IY

Register Exchange Instructions

Z80 Mnemonic	Machine Code	Operation
EX DE,HL	EB	HL <-> DE
EX (SP),HL	E3	H <-> (SP+1); L <-> (SP)
EX (SP),IX	DDE3	IXh <-> (SP+1); IXl <-> (SP)
EX (SP),IY	FDE3	IYh <-> (SP+1); IYl <-> (SP)
EX AF,AF'	08	AF <-> AF'
EXX	D9	BC/DE/HL <-> BC'/DE'/HL'

Add Byte Instructions

Z80 Mnemonic	Machine Code	Operation
ADD A,A	87	A <- A + A
ADD A,B	80	A <- A + B
ADD A,C	81	A <- A + C
ADD A,D	82	A <- A + D
ADD A,E	83	A <- A + E
ADD A,H	84	A <- A + H
ADD A,L	85	A <- A + L
ADD A,(HL)	86	A <- A + (HL)
ADD A,(IX+index)	DD86index	A <- A + (IX+index)
ADD A,(IY+index)	FD86index	A <- A + (IY+index)
ADD A,byte	C6byte	A <- A + byte

Add Byte with Carry-In Instructions

Z80 Mnemonic	Machine Code	Operation
ADC A,A	8F	A <- A + A + Carry
ADC A,B	88	A <- A + B + Carry
ADC A,C	89	A <- A + C + Carry
ADC A,D	8A	A <- A + D + Carry
ADC A,E	8B	A <- A + E + Carry
ADC A,H	8C	A <- A + H + Carry

ADC A,L	8D	A <- A + L + Carry
ADC A,(HL)	8E	A <- A + (HL) + Carry
ADC A,(IX+index)	DD8Eindex	A <- A + (IX+index) + Carry
ADC A,(IY+index)	FD8Eindex	A <- A + (IY+index) + Carry
ADC A,byte	CEbyte	A <- A + byte + Carry

Subtract Byte Instructions

Z80 Mnemonic	Machine Code	Operation
SUB A	97	A <- A - A
SUB B	90	A <- A - B
SUB C	91	A <- A - C
SUB D	92	A <- A - D
SUB E	93	A <- A - E
SUB H	94	A <- A - H
SUB L	95	A <- A - L
SUB (HL)	96	A <- A - (HL)
SUB (IX+index)	DD96index	A <- A - (IX+index)
SUB (IY+index)	FD96index	A <- A - (IY+index)
SUB byte	D6byte	A <- A - byte

Subtract Byte With Borrow-In Instructions

Z80 Mnemonic	Machine Code	Operation
SBC A	9F	A <- A - A - Carry
SBC B	98	A <- A - B - Carry
SBC C	99	A <- A - C - Carry
SBC D	9A	A <- A - D - Carry
SBC E	9B	A <- A - E - Carry
SBC H	9C	A <- A - H - Carry
SBC L	9D	A <- A - L - Carry
SBC (HL)	9E	A <- A - (HL) - Carry
SBC (IX+index)	DD9Eindex	A <- A - (IX+index) - Carry
SBC (IY+index)	FD9Eindex	A <- A - (IY+index) - Carry
SBC byte	DEbyte	A <- A - byte - Carry

Double Byte Add Instructions

Z80 Mnemonic	Machine Code	Operation
ADD HL,BC	09	HL <- HL + BC
ADD HL,DE	19	HL <- HL + DE
ADD HL,HL	29	HL <- HL + HL
ADD HL,SP	39	HL <- HL + SP
ADD IX,BC	DD09	IX <- IX + BC
ADD IX,DE	DD19	IX <- IX + DE
ADD IX,IX	DD29	IX <- IX + IX
ADD IX,SP	DD39	IX <- IX + SP
ADD IY,BC	FD09	IY <- IY + BC
ADD IY,DE	FD19	IY <- IY + DE
ADD IY,IY	FD29	IY <- IY + IY
ADD IY,SP	FD39	IY <- IY + SP

Double Byte Add With Carry-In Instructions

Z80 Mnemonic	Machine Code	Operation
ADC HL,BC	ED4A	HL <- HL + BC + Carry
ADC HL,DE	ED5A	HL <- HL + DE + Carry
ADC HL,HL	ED6A	HL <- HL + HL + Carry
ADC HL,SP	ED7A	HL <- HL + SP + Carry

Double Byte Subtract With Borrow-In Instructions

Z80 Mnemonic	Machine Code	Operation
SBC HL,BC	ED42	HL <- HL - BC - Carry
SBC HL,DE	ED52	HL <- HL - DE - Carry
SBC HL,HL	ED62	HL <- HL - HL - Carry
SBC HL,SP	ED72	HL <- HL - SP - Carry

Control Instructions

Z80 Mnemonic	Machine Code	Operation
DI	F3	IFF <- 0
EI	FB	IFF <- 1
IM 0	ED46	---
IM 1	ED56	---
IM 2	ED5E	---
LD I,A	ED47	Interrupt Page <- A
NOP	00	No Operation
HLT	76	NOP;PC <- PC-1

Increment Byte Instructions

Z80 Mnemonic	Machine Code	Operation
INC A	3C	A <- A + 1
INC B	04	B <- B + 1
INC C	0C	C <- C + 1
INC D	14	D <- D + 1
INC E	1C	E <- E + 1
INC H	24	H <- H + 1
INC L	2C	L <- L + 1
INC (HL)	34	(HL) <- (HL) + 1
INC (IX+index)	DD34index	(IX+index) <- (IX+index) + 1
INC (IY+index)	FD34index	(IY+index) <- (IY+index) + 1

Decrement Byte Instructions

Z80 Mnemonic	Machine Code	Operation
DEC A	3D	A <- A - 1
DEC B	05	B <- B - 1
DEC C	0D	C <- C - 1
DEC D	15	D <- D - 1
DEC E	1D	E <- E - 1
DEC H	25	H <- H - 1
DEC L	2D	L <- L - 1
DEC (HL)	35	(HL) <- (HL) - 1
DEC (IX+index)	DD35index	(IX+index) <- (IX+index) - 1
DEC (IY+index)	FD35index	(IY+index) <- (IY+index) - 1

Increment Register Pair Instructions

Z80 Mnemonic	Machine Code	Operation
INC BC	03	BC <- BC + 1
INC DE	13	DE <- DE + 1
INC HL	23	HL <- HL + 1
INC SP	33	SP <- SP + 1
INC IX	DD23	IX <- IX + 1
INC IY	FD23	IY <- IY + 1

Decrement Register Pair Instructions

Z80 Mnemonic	Machine Code	Operation
DEC BC	0B	BC <- BC - 1
DEC DE	1B	DE <- DE - 1
DEC HL	2B	HL <- HL - 1
DEC SP	3B	SP <- SP - 1
DEC IX	DD2B	IX <- IX - 1

DEC IY FD2B IY <- IY - 1

Special Accumulator and Flag Instructions

Z80 Mnemonic	Machine Code	Operation
DAA	27	---
CPL	2F	A <- NOT A
SCF	37	CF (Carry Flag) <- 1
CCF	3F	CF (Carry Flag) <- NOT CF
NEG	ED44	A <- 0-A

Rotate Instructions

Z80 Mnemonic	Machine Code	Operation
RLCA	07	---
RRCA	0F	---
RLA	17	---
RRA	1F	---
RLD	ED6F	---
RRD	ED67	---
RLC A	CB07	---
RLC B	CB00	---
RLC C	CB01	---
RLC D	CB02	---
RLC E	CB03	---
RLC H	CB04	---
RLC L	CB05	---
RLC (HL)	CB06	---
RLC (IX+index)	DDCBindex06	---
RLC (IY+index)	FDCBindex06	---
RL A	CB17	---
RL B	CB10	---
RL C	CB11	---
RL D	CB12	---
RL E	CB13	---
RL H	CB14	---
RL L	CB15	---
RL (HL)	CB16	---
RL (IX+index)	DDCBindex16	---
RL (IY+index)	FDCBindex16	---
RRC A	CB0F	---
RRC B	CB08	---
RRC C	CB09	---
RRC D	CB0A	---
RRC E	CB0B	---
RRC H	CB0C	---
RRC L	CB0D	---
RRC (HL)	CB0E	---
RRC (IX+index)	DDCBindex0E	---
RRC (IY+index)	FDCBindex0E	---
RL A	CB1F	---
RL B	CB18	---
RL C	CB19	---
RL D	CB1A	---
RL E	CB1B	---
RL H	CB1C	---
RL L	CB1D	---
RL (HL)	CB1E	---
RL (IX+index)	DDCBindex1E	---
RL (IY+index)	FDCBindex1E	---

Logical Byte Instructions

Z80 Mnemonic	Machine Code	Operation
AND A	A7	A ← A AND A
AND B	A0	A ← A AND B
AND C	A1	A ← A AND C
AND D	A2	A ← A AND D
AND E	A3	A ← A AND E
AND H	A4	A ← A AND H
AND L	A5	A ← A AND L
AND (HL)	A6	A ← A AND (HL)
AND (IX+index)	DDA6index	A ← A AND (IX+index)
AND (IY+index)	FDA6index	A ← A AND (IY+index)
AND byte	E6byte	A ← A AND byte
XOR A	AF	A ← A XOR A
XOR B	A8	A ← A XOR B
XOR C	A9	A ← A XOR C
XOR D	AA	A ← A XOR D
XOR E	AB	A ← A XOR E
XOR H	AC	A ← A XOR H
XOR L	AD	A ← A XOR L
XOR (HL)	AE	A ← A XOR (HL)
XOR (IX+index)	DDAEindex	A ← A XOR (IX+index)
XOR (IY+index)	FDAEindex	A ← A XOR (IY+index)
XOR byte	EEbyte	A ← A XOR byte
OR A	B7	A ← A OR A
OR B	B0	A ← A OR B
OR C	B1	A ← A OR C
OR D	B2	A ← A OR D
OR E	B3	A ← A OR E
OR H	B4	A ← A OR H
OR L	B5	A ← A OR L
OR (HL)	B6	A ← A OR (HL)
OR (IX+index)	DDB6index	A ← A OR (IX+index)
OR (IY+index)	FDB6index	A ← A OR (IY+index)
OR byte	F6byte	A ← A OR byte
CP A	BF	A - A
CP B	B8	A - B
CP C	B9	A - C
CP D	BA	A - D
CP E	BB	A - E
CP H	BC	A - H
CP L	BD	A - L
CP (HL)	BE	A - (HL)
CP (IX+index)	DDBEindex	A - (IX+index)
CP (IY+index)	FDBEindex	A - (IY+index)
CP byte	FEbyte	A - byte
CPI	EDA1	A - (HL); HL ← HL+1; BC ← BC-1
CPIR	EDB1	A - (HL); HL ← HL+1; BC ← BC-1
CPD	EDA9	A - (HL); HL ← HL-1; BC ← BC-1
CPDR	EDB9	A - (HL); HL ← HL-1; BC ← BC-1

Branch Control/Program Counter Load Instructions

Z80 Mnemonic	Machine Code	Operation
JP address	C3address	PC ← address
JP NZ,address	C2address	If NZ, PC ← address
JP Z,address	CAaddress	If Z, PC ← address
JP NC,address	D2address	If NC, PC ← address

JP C,address	DAaddress	If C, PC ← address
JP PO,address	E2address	If PO, PC ← address
JP PE,address	EAaddress	If PE, PC ← address
JP P,address	F2address	If P, PC ← address
JM M,address	FAaddress	If M, PC ← address
JP (HL)	E9	PC ← HL
JP (IX)	DDE9	PC ← IX
JP (IY)	FDE9	PC ← IY
JR index	18index	PC ← PC + index
JR NZ,index	20index	If NZ, PC ← PC + index
JR Z,index	28index	If Z, PC ← PC + index
JR NC,index	30index	If NC, PC ← PC + index
JR C,index	38index	If C, PC ← PC + index
DJNZ index	10index	B ← B - 1; while B > 0, PC ← PC + index
CALL address	CDaddress	(SP-1) ← PCh; (SP-2) ← PCl; SP ← SP - 2; PC ← address
CALL NZ,address	C4address	If NZ, CALL address
CALL Z,address	CCaddress	If Z, CALL address
CALL NC,address	D4address	If NC, CALL address
CALL C,address	DCaddress	If C, CALL address
CALL PO,address	E4address	If PO, CALL address
CALL PE,address	ECaddress	If PE, CALL address
CALL P,address	F4address	If P, CALL address
CALL M,address	FCaddress	If M, CALL address
RET	C9	PCl ← (SP); PCh ← (SP+1); SP ← (SP+2)
RET NZ	C0	If NZ, RET
RET Z	C8	If Z, RET
RET NC	D0	If NC, RET
RET C	D8	If C, RET
RET PO	E0	If PO, RET
RET PE	E8	If PE, RET
RET P	F0	If P, RET
RET M	F8	If M, RET
RETI	ED4D	Return from Interrupt
RETN	ED45	IFF1 ← IFF2; RETI
RST 0	C7	CALL 0
RST 8	CF	CALL 8
RST 10H	D7	CALL 10H
RST 18H	DF	CALL 18H
RST 20H	E7	CALL 20H
RST 28H	EF	CALL 28H
RST 30H	F7	CALL 30H
RST 38H	FF	CALL 38H

Stack Operation Instructions

Z80 Mnemonic	Machine Code	Operation
PUSH BC	C5	(SP-2) ← C; (SP-1) ← B; SP ← SP - 2
PUSH DE	D5	(SP-2) ← E; (SP-1) ← D; SP ← SP - 2
PUSH HL	E5	(SP-2) ← L; (SP-1) ← H; SP ← SP - 2
PUSH AF	F5	(SP-2) ← Flags; (SP-1) ← A; SP ← SP - 2
PUSH IX	DDE5	(SP-2) ← IXl; (SP-1) ← IXh; SP ← SP - 2
PUSH IY	FDE5	(SP-2) ← IYl; (SP-1) ← IYh; SP ← SP - 2
POP BC	C1	B ← (SP+1); C ← (SP); SP ← SP + 2
POP DE	D1	D ← (SP+1); E ← (SP); SP ← SP + 2
POP HL	E1	H ← (SP+1); L ← (SP); SP ← SP + 2
POP AF	F1	A ← (SP+1); Flags ← (SP); SP ← SP + 2
POP IX	DDE1	IXh ← (SP+1); IXl ← (SP); SP ← (SP+2)
POP IY	FDE1	IYh ← (SP+1); IYl ← (SP); SP ← (SP+2)

Input/Output Instructions

Z80 Mnemonic	Machine Code	Operation
IN A,(byte)	DBbyte	A <- [byte]
IN A,(C)	ED78	A <- [C]
IN B,(C)	ED40	B <- [C]
IN C,(C)	ED48	C <- [C]
IN D,(C)	ED50	D <- [C]
IN E,(C)	ED58	E <- [C]
IN H,(C)	ED60	H <- [C]
IN L,(C)	ED68	L <- [C]
INI	EDA2	(HL) <- [C]; B <- B-1; HL <- HL+1
INIR	EDB2	(HL) <- [C]; B <- B-1; HL <- HL+1; Repeat while B>0
IND	EDAA	(HL) <- [C]; B <- B-1; HL <- HL-1
INDR	EDBA	(HL) <- [C]; B <- B-1; HL <- HL-1; Repeat while B>0
OUT (byte),A	D320	[byte] <- A
OUT (C),A	ED79	[C] <- A
OUT (C),B	ED41	[C] <- B
OUT (C),C	ED49	[C] <- C
OUT (C),D	ED51	[C] <- D
OUT (C),E	ED59	[C] <- E
OUT (C),H	ED61	[C] <- H
OUT (C),L	ED69	[C] <- L
OUTI	EDA3	[C] <- (HL); B <- B-1; HL <- HL+1
OTIR	EDB3	[C] <- (HL); B <- B-1; HL <- HL+1; Repeat while B>0
OUTD	EDAB	[C] <- (HL); B <- B-1; HL <- HL-1
OTDR	EDBB	[C] <- (HL); B <- B-1; HL <- HL-1; Repeat while B>0

Data Transfer Instructions

Z80 Mnemonic	Machine Code	Operation
LDI	EDA0	(DE) <- (HL); HL <- HL+1; DE <- DE+1; BC <- BC-1
LDIR	EDB0	(DE) <- (HL); HL <- HL+1; DE <- DE+1; BC <- BC-1; repeat while BC<>-1
LDD	EDA8	(DE) <- (HL); HL <- HL-1; DE <- DE-1; BC <- BC-1
LDDR	EDB8	(DE) <- (HL); HL <- HL-1; DE <- DE-1; BC <- BC-1; repeat while BC<>-1

Bit Manipulation Instructions

Z80 Mnemonic	Machine Code	Operation
BIT 0,A	CB47	Z flag <- NOT Bit 0
BIT 0,B	CB40	Z flag <- NOT Bit 0
BIT 0,C	CB41	Z flag <- NOT Bit 0
BIT 0,D	CB42	Z flag <- NOT Bit 0
BIT 0,E	CB43	Z flag <- NOT Bit 0
BIT 0,H	CB44	Z flag <- NOT Bit 0
BIT 0,L	CB45	Z flag <- NOT Bit 0
BIT 0,(HL)	CB46	Z flag <- NOT Bit 0
BIT 0,(IX+index)	DDCBindex46	Z flag <- NOT Bit 0
BIT 0,(IY+index)	FDCBindex46	Z flag <- NOT Bit 0
BIT 1,A	CB4F	Z flag <- NOT Bit 1
BIT 1,B	CB48	Z flag <- NOT Bit 1
BIT 1,C	CB49	Z flag <- NOT Bit 1
BIT 1,D	CB4A	Z flag <- NOT Bit 1
BIT 1,E	CB4B	Z flag <- NOT Bit 1
BIT 1,H	CB4C	Z flag <- NOT Bit 1

BIT 1,L	CB4D	Z flag <- NOT Bit 1
BIT 1,(HL)	CB4E	Z flag <- NOT Bit 1
BIT 1,(IX+index)	DDCBindex4E	Z flag <- NOT Bit 1
BIT 1,(IY+index)	FDCBindex4E	Z flag <- NOT Bit 1
BIT 2,A	CB57	Z flag <- NOT Bit 2
BIT 2,B	CB50	Z flag <- NOT Bit 2
BIT 2,C	CB51	Z flag <- NOT Bit 2
BIT 2,D	CB52	Z flag <- NOT Bit 2
BIT 2,E	CB53	Z flag <- NOT Bit 2
BIT 2,H	CB54	Z flag <- NOT Bit 2
BIT 2,L	CB55	Z flag <- NOT Bit 2
BIT 2,(HL)	CB56	Z flag <- NOT Bit 2
BIT 2,(IX+index)	DDCBindex56	Z flag <- NOT Bit 2
BIT 2,(IY+index)	FDCBindex56	Z flag <- NOT Bit 2
BIT 3,A	CB5F	Z flag <- NOT Bit 3
BIT 3,B	CB58	Z flag <- NOT Bit 3
BIT 3,C	CB59	Z flag <- NOT Bit 3
BIT 3,D	CB5A	Z flag <- NOT Bit 3
BIT 3,E	CB5B	Z flag <- NOT Bit 3
BIT 3,H	CB5C	Z flag <- NOT Bit 3
BIT 3,L	CB5D	Z flag <- NOT Bit 3
BIT 3,(HL)	CB5E	Z flag <- NOT Bit 3
BIT 3,(IX+index)	DDCBindex5E	Z flag <- NOT Bit 3
BIT 3,(IY+index)	FDCBindex5E	Z flag <- NOT Bit 3
BIT 4,A	CB67	Z flag <- NOT Bit 4
BIT 4,B	CB60	Z flag <- NOT Bit 4
BIT 4,C	CB61	Z flag <- NOT Bit 4
BIT 4,D	CB62	Z flag <- NOT Bit 4
BIT 4,E	CB63	Z flag <- NOT Bit 4
BIT 4,H	CB64	Z flag <- NOT Bit 4
BIT 4,L	CB65	Z flag <- NOT Bit 4
BIT 4,(HL)	CB66	Z flag <- NOT Bit 4
BIT 4,(IX+index)	DDCBindex66	Z flag <- NOT Bit 4
BIT 4,(IY+index)	FDCBindex66	Z flag <- NOT Bit 4
BIT 5,A	CB6F	Z flag <- NOT Bit 5
BIT 5,B	CB68	Z flag <- NOT Bit 5
BIT 5,C	CB69	Z flag <- NOT Bit 5
BIT 5,D	CB6A	Z flag <- NOT Bit 5
BIT 5,E	CB6B	Z flag <- NOT Bit 5
BIT 5,H	CB6C	Z flag <- NOT Bit 5
BIT 5,L	CB6D	Z flag <- NOT Bit 5
BIT 5,(HL)	CB6E	Z flag <- NOT Bit 5
BIT 5,(IX+index)	DDCBindex6E	Z flag <- NOT Bit 5
BIT 5,(IY+index)	FDCBindex6E	Z flag <- NOT Bit 5
BIT 6,A	CB77	Z flag <- NOT Bit 6
BIT 6,B	CB70	Z flag <- NOT Bit 6
BIT 6,C	CB71	Z flag <- NOT Bit 6
BIT 6,D	CB72	Z flag <- NOT Bit 6
BIT 6,E	CB73	Z flag <- NOT Bit 6
BIT 6,H	CB74	Z flag <- NOT Bit 6
BIT 6,L	CB75	Z flag <- NOT Bit 6
BIT 6,(HL)	CB76	Z flag <- NOT Bit 6
BIT 6,(IX+index)	DDCBindex76	Z flag <- NOT Bit 6
BIT 6,(IY+index)	FDCBindex76	Z flag <- NOT Bit 6
BIT 7,A	CB7F	Z flag <- NOT Bit 7
BIT 7,B	CB78	Z flag <- NOT Bit 7
BIT 7,C	CB79	Z flag <- NOT Bit 7
BIT 7,D	CB7A	Z flag <- NOT Bit 7

BIT	7,E	CB7B	Z flag <- NOT Bit 7
BIT	7,H	CB7C	Z flag <- NOT Bit 7
BIT	7,L	CB7D	Z flag <- NOT Bit 7
BIT	7,(HL)	CB7E	Z flag <- NOT Bit 7
BIT	7,(IX+index)	DDCBindex7E	Z flag <- NOT Bit 7
BIT	7,(IY+index)	FDCBindex7E	Z flag <- NOT Bit 7
RES	0,A	CB87	Bit 0 <- 0
RES	0,B	CB80	Bit 0 <- 0
RES	0,C	CB81	Bit 0 <- 0
RES	0,D	CB82	Bit 0 <- 0
RES	0,E	CB83	Bit 0 <- 0
RES	0,H	CB84	Bit 0 <- 0
RES	0,L	CB85	Bit 0 <- 0
RES	0,(HL)	CB86	Bit 0 <- 0
RES	0,(IX+index)	DDCBindex86	Bit 0 <- 0
RES	0,(IY+index)	FDCBindex86	Bit 0 <- 0
RES	1,A	CB8F	Bit 1 <- 0
RES	1,B	CB88	Bit 1 <- 0
RES	1,C	CB89	Bit 1 <- 0
RES	1,D	CB8A	Bit 1 <- 0
RES	1,E	CB8B	Bit 1 <- 0
RES	1,H	CB8C	Bit 1 <- 0
RES	1,L	CB8D	Bit 1 <- 0
RES	1,(HL)	CB8E	Bit 1 <- 0
RES	1,(IX+index)	DDCBindex8E	Bit 1 <- 0
RES	1,(IY+index)	FDCBindex8E	Bit 1 <- 0
RES	2,A	CB97	Bit 2 <- 0
RES	2,B	CB90	Bit 2 <- 0
RES	2,C	CB91	Bit 2 <- 0
RES	2,D	CB92	Bit 2 <- 0
RES	2,E	CB93	Bit 2 <- 0
RES	2,H	CB94	Bit 2 <- 0
RES	2,L	CB95	Bit 2 <- 0
RES	2,(HL)	CB96	Bit 2 <- 0
RES	2,(IX+index)	DDCBindex96	Bit 2 <- 0
RES	2,(IY+index)	FDCBindex96	Bit 2 <- 0
RES	3,A	CB9F	Bit 3 <- 0
RES	3,B	CB98	Bit 3 <- 0
RES	3,C	CB99	Bit 3 <- 0
RES	3,D	CB9A	Bit 3 <- 0
RES	3,E	CB9B	Bit 3 <- 0
RES	3,H	CB9C	Bit 3 <- 0
RES	3,L	CB9D	Bit 3 <- 0
RES	3,(HL)	CB9E	Bit 3 <- 0
RES	3,(IX+index)	DDCBindex9E	Bit 3 <- 0
RES	3,(IY+index)	FDCBindex9E	Bit 3 <- 0
RES	4,A	CBA7	Bit 4 <- 0
RES	4,B	CBA0	Bit 4 <- 0
RES	4,C	CBA1	Bit 4 <- 0
RES	4,D	CBA2	Bit 4 <- 0
RES	4,E	CBA3	Bit 4 <- 0
RES	4,H	CBA4	Bit 4 <- 0
RES	4,L	CBA5	Bit 4 <- 0
RES	4,(HL)	CBA6	Bit 4 <- 0
RES	4,(IX+index)	DDCBindexA6	Bit 4 <- 0
RES	4,(IY+index)	FDCBindexA6	Bit 4 <- 0
RES	5,A	CBAF	Bit 5 <- 0
RES	5,B	CBA8	Bit 5 <- 0

RES	5,C	CBA9	Bit 5 <- 0
RES	5,D	CBA A	Bit 5 <- 0
RES	5,E	CBAB	Bit 5 <- 0
RES	5,H	CBAC	Bit 5 <- 0
RES	5,L	CBAD	Bit 5 <- 0
RES	5,(HL)	CBAE	Bit 5 <- 0
RES	5,(IX+index)	DDCBindexAE	Bit 5 <- 0
RES	5,(IY+index)	FDCBindexAE	Bit 5 <- 0
RES	6,A	CBB7	Bit 6 <- 0
RES	6,B	CBB0	Bit 6 <- 0
RES	6,C	CBB1	Bit 6 <- 0
RES	6,D	CBB2	Bit 6 <- 0
RES	6,E	CBB3	Bit 6 <- 0
RES	6,H	CBB4	Bit 6 <- 0
RES	6,L	CBB5	Bit 6 <- 0
RES	6,(HL)	CBB6	Bit 6 <- 0
RES	6,(IX+index)	DDCBindexB6	Bit 6 <- 0
RES	6,(IY+index)	FDCBindexB6	Bit 6 <- 0
RES	7,A	CBBF	Bit 7 <- 0
RES	7,B	CBB8	Bit 7 <- 0
RES	7,C	CBB9	Bit 7 <- 0
RES	7,D	CBBA	Bit 7 <- 0
RES	7,E	CBBB	Bit 7 <- 0
RES	7,H	CBBC	Bit 7 <- 0
RES	7,L	CBBD	Bit 7 <- 0
RES	7,(HL)	CBBE	Bit 7 <- 0
RES	7,(IX+index)	DDCBindexBE	Bit 7 <- 0
RES	7,(IY+index)	FDCBindexBE	Bit 7 <- 0
SET	0,A	CBC7	Bit 0 <- 1
SET	0,B	CBC0	Bit 0 <- 1
SET	0,C	CBC1	Bit 0 <- 1
SET	0,D	CBC2	Bit 0 <- 1
SET	0,E	CBC3	Bit 0 <- 1
SET	0,H	CBC4	Bit 0 <- 1
SET	0,L	CBC5	Bit 0 <- 1
SET	0,(HL)	CBC6	Bit 0 <- 1
SET	0,(IX+index)	DDCBindexC6	Bit 0 <- 1
SET	0,(IY+index)	FDCBindexC6	Bit 0 <- 1
SET	1,A	CBCF	Bit 1 <- 1
SET	1,B	CBC8	Bit 1 <- 1
SET	1,C	CBC9	Bit 1 <- 1
SET	1,D	CBCA	Bit 1 <- 1
SET	1,E	CBCB	Bit 1 <- 1
SET	1,H	CBCC	Bit 1 <- 1
SET	1,L	CB CD	Bit 1 <- 1
SET	1,(HL)	CBCE	Bit 1 <- 1
SET	1,(IX+index)	DDCBindexCE	Bit 1 <- 1
SET	1,(IY+index)	FDCBindexCE	Bit 1 <- 1
SET	2,A	CBD7	Bit 2 <- 1
SET	2,B	CBD0	Bit 2 <- 1
SET	2,C	CBD1	Bit 2 <- 1
SET	2,D	CBD2	Bit 2 <- 1
SET	2,E	CBD3	Bit 2 <- 1
SET	2,H	CBD4	Bit 2 <- 1
SET	2,L	CBD5	Bit 2 <- 1
SET	2,(HL)	CBD6	Bit 2 <- 1
SET	2,(IX+index)	DDCBindexD6	Bit 2 <- 1
SET	2,(IY+index)	FDCBindexD6	Bit 2 <- 1

SET	3,A	CBDF	Bit 3 <- 1	SLA	H	CB24	---
SET	3,B	CBD8	Bit 3 <- 1	SLA	L	CB25	---
SET	3,C	CBD9	Bit 3 <- 1	SLA	(HL)	CB26	---
SET	3,D	CBDA	Bit 3 <- 1	SLA	(IX+index)	DDCBindex26	---
SET	3,E	CBDB	Bit 3 <- 1	SLA	(IY+index)	FDCBindex26	---
SET	3,H	CBDC	Bit 3 <- 1	SRA	A	CB2F	---
SET	3,L	CBDD	Bit 3 <- 1	SRA	B	CB28	---
SET	3,(HL)	CBDE	Bit 3 <- 1	SRA	C	CB29	---
SET	3,(IX+index)	DDCBindexDE	Bit 3 <- 1	SRA	D	CB2A	---
SET	3,(IY+index)	FDCBindexDE	Bit 3 <- 1	SRA	E	CB2B	---
SET	4,A	CBE7	Bit 4 <- 1	SRA	H	CB2C	---
SET	4,B	CBE0	Bit 4 <- 1	SRA	L	CB2D	---
SET	4,C	CBE1	Bit 4 <- 1	SRA	(HL)	CB2E	---
SET	4,D	CBE2	Bit 4 <- 1	SRA	(IX+index)	DDCBindex2E	---
SET	4,E	CBE3	Bit 4 <- 1	SRA	(IY+index)	FDCBindex2E	---
SET	4,H	CBE4	Bit 4 <- 1	SRL	A	CB3F	---
SET	4,L	CBE5	Bit 4 <- 1	SRL	B	CB38	---
SET	4,(HL)	CBE6	Bit 4 <- 1	SRL	C	CB39	---
SET	4,(IX+index)	DDCBindexE6	Bit 4 <- 1	SRL	D	CB3A	---
SET	4,(IY+index)	FDCBindexE6	Bit 4 <- 1	SRL	E	CB3B	---
SET	5,A	CBEF	Bit 5 <- 1	SRL	H	CB3C	---
SET	5,B	CBE8	Bit 5 <- 1	SRL	L	CB3D	---
SET	5,C	CBE9	Bit 5 <- 1	SRL	(HL)	CB3E	---
SET	5,D	CBEA	Bit 5 <- 1	SRL	(IX+index)	DDCBindex3E	---
SET	5,E	CBEB	Bit 5 <- 1	SRL	(IY+index)	FDCBindex3E	---
SET	5,H	CBEC	Bit 5 <- 1				
SET	5,L	CBED	Bit 5 <- 1				
SET	5,(HL)	CBEE	Bit 5 <- 1				
SET	5,(IX+index)	DDCBindexEE	Bit 5 <- 1				
SET	5,(IY+index)	FDCBindexEE	Bit 5 <- 1				
SET	6,A	CBF7	Bit 6 <- 1				
SET	6,B	CBF0	Bit 6 <- 1				
SET	6,C	CBF1	Bit 6 <- 1				
SET	6,D	CBF2	Bit 6 <- 1				
SET	6,E	CBF3	Bit 6 <- 1				
SET	6,H	CBF4	Bit 6 <- 1				
SET	6,L	CBF5	Bit 6 <- 1				
SET	6,(HL)	CBF6	Bit 6 <- 1				
SET	6,(IX+index)	DDCBindexF6	Bit 6 <- 1				
SET	6,(IY+index)	FDCBindexF6	Bit 6 <- 1				
SET	7,A	CBFF	Bit 7 <- 1				
SET	7,B	CBF8	Bit 7 <- 1				
SET	7,C	CBF9	Bit 7 <- 1				
SET	7,D	CBFA	Bit 7 <- 1				
SET	7,E	CBFB	Bit 7 <- 1				
SET	7,H	CBFC	Bit 7 <- 1				
SET	7,L	CBFD	Bit 7 <- 1				
SET	7,(HL)	CBFE	Bit 7 <- 1				
SET	7,(IX+index)	DDCBindexFE	Bit 7 <- 1				
SET	7,(IY+index)	FDCBindexFE	Bit 7 <- 1				

Bit Shift Instructions

Z80 Mnemonic	Machine Code	Operation
SLA A	CB27	---
SLA B	CB20	---
SLA C	CB21	---
SLA D	CB22	---
SLA E	CB23	---

Notes:

1. **byte** is an 8-bit data value
2. **index** is an 8-bit value
3. **word** is a 16-bit data value
4. **address** is a 16-bit address value