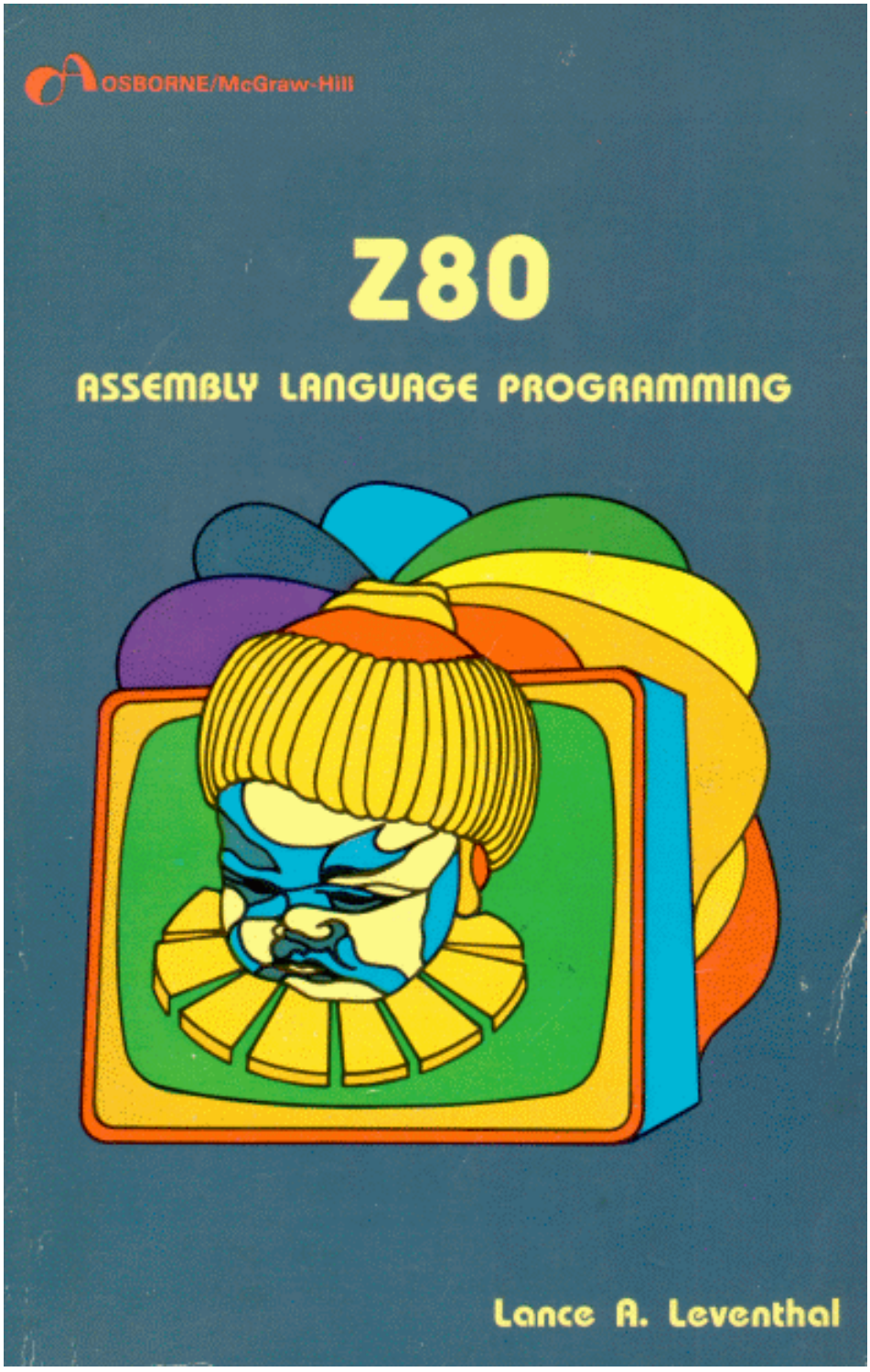


Scanned and converted to PDF by HansO, 2001

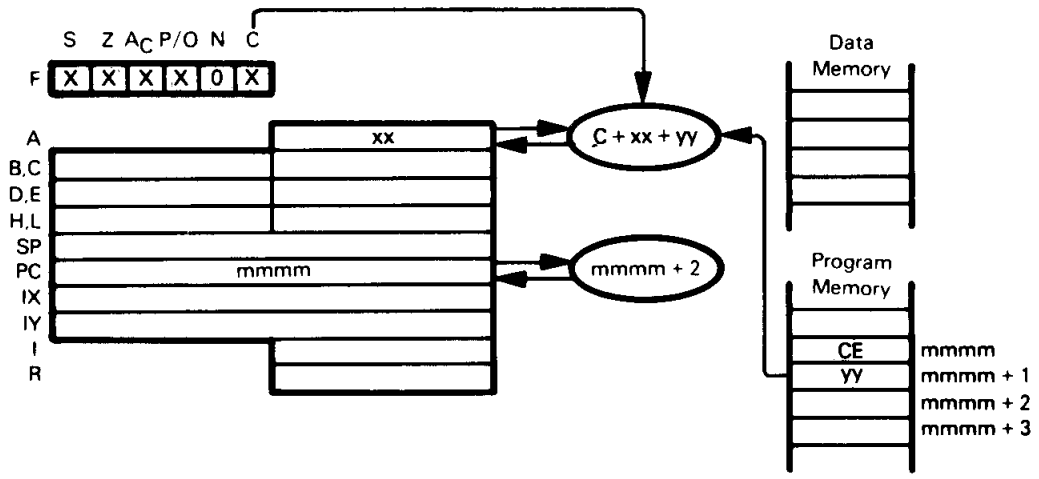
Chapter 3 page 43-163

Contents (Continued)

Chapter		Page
3	The Z80 Assembly Language Instruction Set	3-1
	CPU Registers and Status Flags	3-2
	Z80 Memory Addressing Modes	3-4
	Implied	3-5
	Implied Block Transfer with Auto-Increment/Decrement	3-7
	Implied Stack	3-8
	Indexed	3-10
	Direct	3-11
	Program Relative	3-12
	Base Page	3-13
	Register Direct	3-14
	Immediate	3-15
	Abbreviations	3-18
	Instruction Mnemonics	3-21
	Instruction Object Codes	3-21
	Instruction Execution Times	3-21
	Status	3-21
	Instruction Descriptions	3-43
	8080A/Z80 Compatibility	3-164
	Zilog Z80 Assembler Conventions	3-170
	Assembler Field Structure	3-170
	Labels	3-170
	Reserved Names	3-170
	Pseudo-operations	3-170
	Examples	3-171
	Labels with Pseudo-operations	3-172
	Addresses	3-172
	Conditional Assembly	3-174
	Macros	3-174

In this PDF: Chapter 3 page 43-163

ADC A,data — ADD IMMEDIATE WITH CARRY TO ACCUMULATOR



ADC A, data
CE yy

Add the contents of the next program memory byte and the Carry status to the Accumulator.

Suppose xx=$3A_{16}$, yy=$7C_{16}$, and Carry=0. After the instruction

ADC A,7CH

has executed, the Accumulator will contain $B6_{16}$:

```
3A    =  0 0 1 1   1 0 1 0
7C    =  0 1 1 1   1 1 0 0
Carry =                0
        1 0 1 1   0 1 1 0
```

1 sets S to 1

No carry, set C to 0

0 ∀ 1=1, set P/O to 1

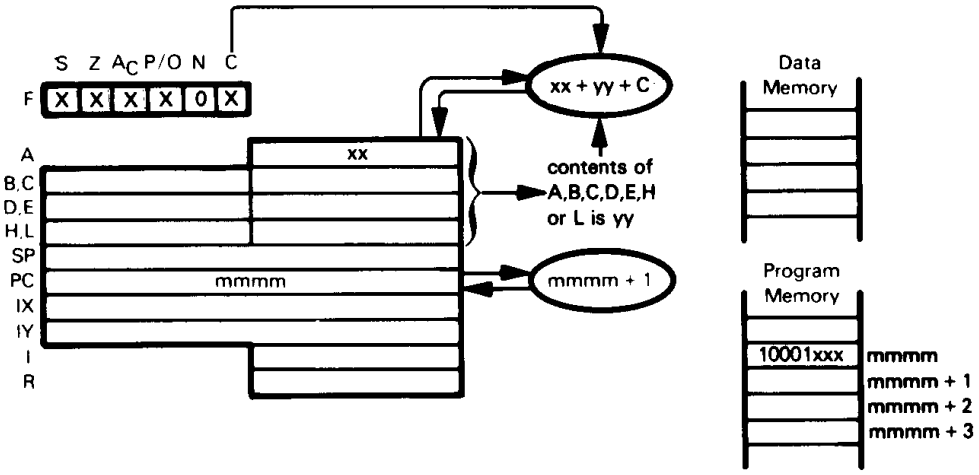
Non-zero result, set Z to 0

Carry, set A_C to 1

Addition instruction, set N to 0

The ADC instruction is frequently used in multibyte addition for the second and subsequent bytes.

ADC A,reg — ADD REGISTER WITH CARRY TO ACCUMULATOR



ADC A, reg

10001 xxx

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

Add the contents of Register A, B, C, D, E, H or L and the Carry status to the Accumulator.

Suppose xx=$E3_{16}$, Register E contains $A0_{16}$, and Carry=1. After the instruction

ADC A,E

has executed, the Accumulator will contain 84_{16}:

```
E3    =  1 1 1 0   0 0 1 1
A0    =  1 0 1 0   0 0 0 0
Carry =                  1
       ---------------------
       1 0 0 0   0 1 0 0
```

1 sets S to 1

Non-zero result, set Z to 0

Carry, set C to 1

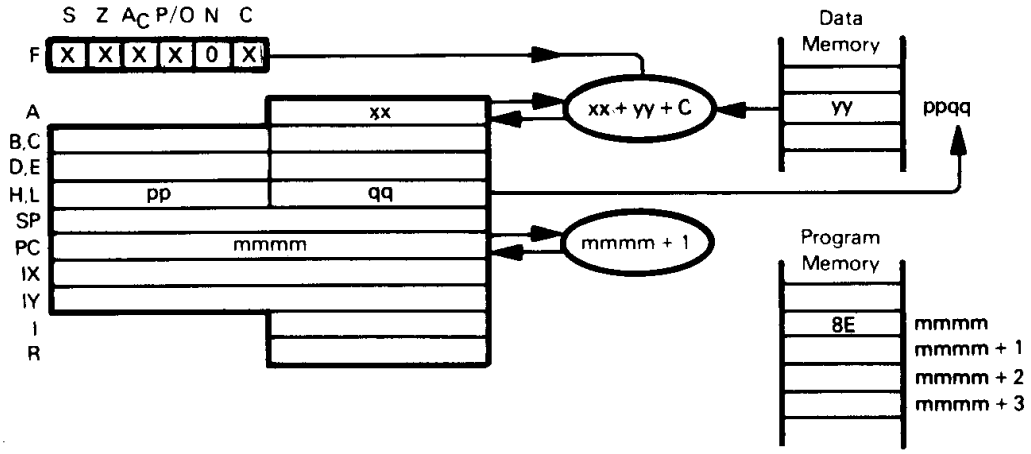
No carry, set A_C to 0

1 ⊻ 1=0, set P/O to 0

Addition instruction, set N to 0

The ADC instruction is most frequently used in multibyte addition for the second and subsequent bytes.

ADC A,(HL) — ADD MEMORY AND CARRY TO ACCUMULATOR
ADC A,(IX+disp)
ADC A,(IY+disp)



The illustration shows execution of ADC A,(HL):

ADC A,(HL)
8E

Add the contents of memory location (specified by the contents of the HL register pair) and the Carry status to the Accumulator.

Suppose xx=$E3_{16}$, yy=$A0_{16}$, and Carry=1. After the instruction

ADC A,(HL)

has executed, the Accumulator will contain 84_{16}:

```
E3    =  1 1 1 0   0 0 1 1
A0    =  1 0 1 0   0 0 0 0
Carry =                  1
        ---------------------
        1 0 0 0   0 1 0 0
```

1 sets S to 1

Non-zero result, set Z to 0

Carry, set C to 1

No carry, set A_C to 0

1 ⊻ 1=0, set P/O to 0

Addition instruction, set N to 0

ADC A,(IX+disp)
DD 8E d

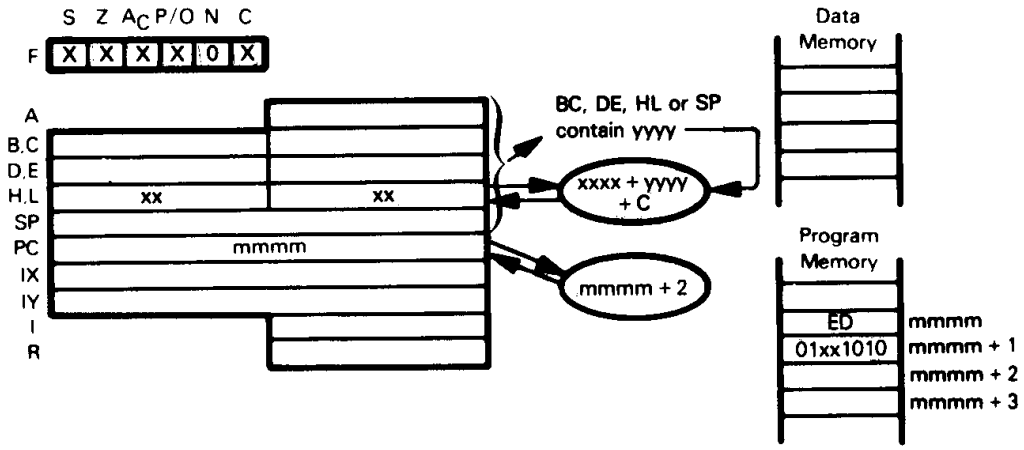
Add the contents of memory location (specified by the sum of the contents of the IX register and the displacement digit d) and the Carry to the Accumulator.

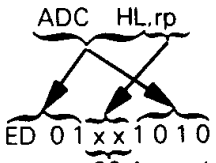
ADC A,(IY+disp)
FD 8E d

This instruction is identical to ADC A,(IX+disp), except that it uses the IY register instead of the IX register.

The ADC instruction is most frequently used in multibyte addition for the second and subsequent bytes.

ADC HL,rp — ADD REGISTER PAIR WITH CARRY TO H AND L





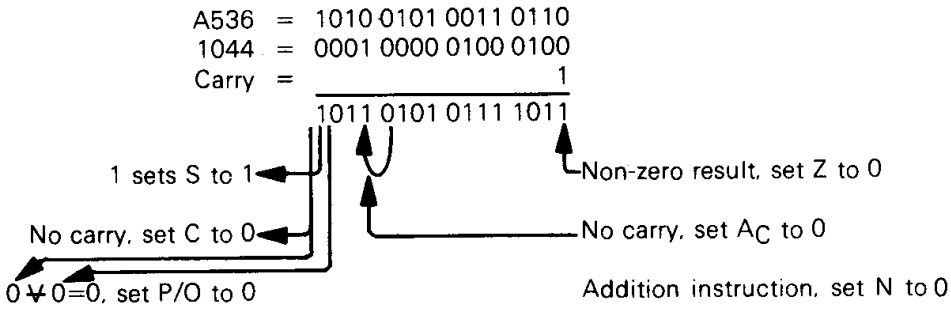
00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Add the 16-bit value from either the BC, DE, HL register pair or the Stack Pointer, and the Carry status, to the HL register pair.

Suppose HL contains $A536_{16}$, BC contains 1044_{16}, and Carry=1. After execution of

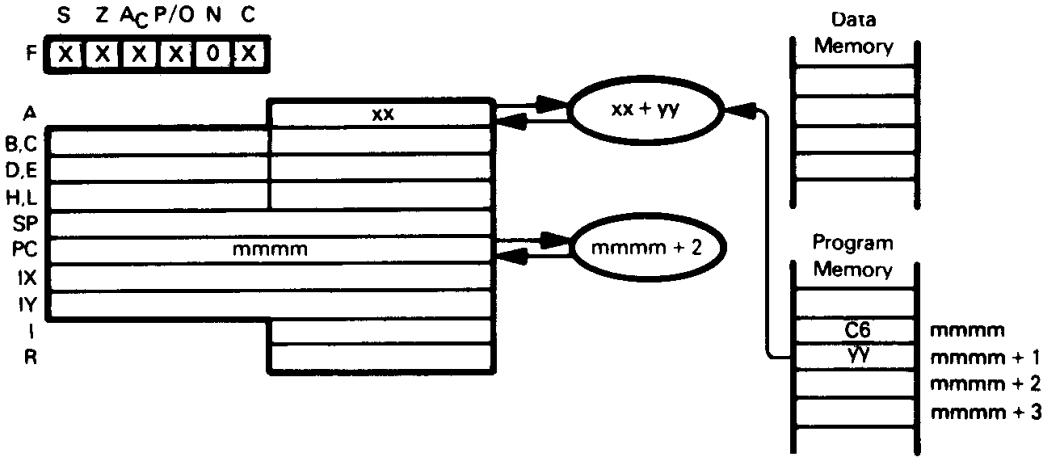
ADC HL,BC

the HL register pair will contain:



The ADC instruction is most frequently used in multibyte addition for the second and subsequent bytes.

ADD A,data — ADD IMMEDIATE TO ACCUMULATOR



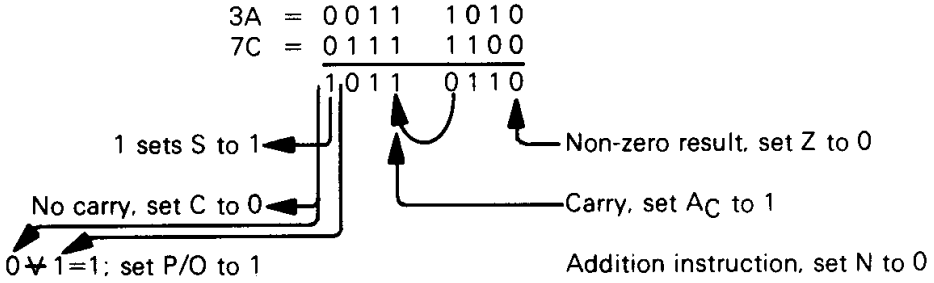
ADD A, data
C6 yy

Add the contents of the next program memory byte to the Accumulator.

Suppose xx=$3A_{16}$, yy=$7C_{16}$, and Carry=0. After the instruction

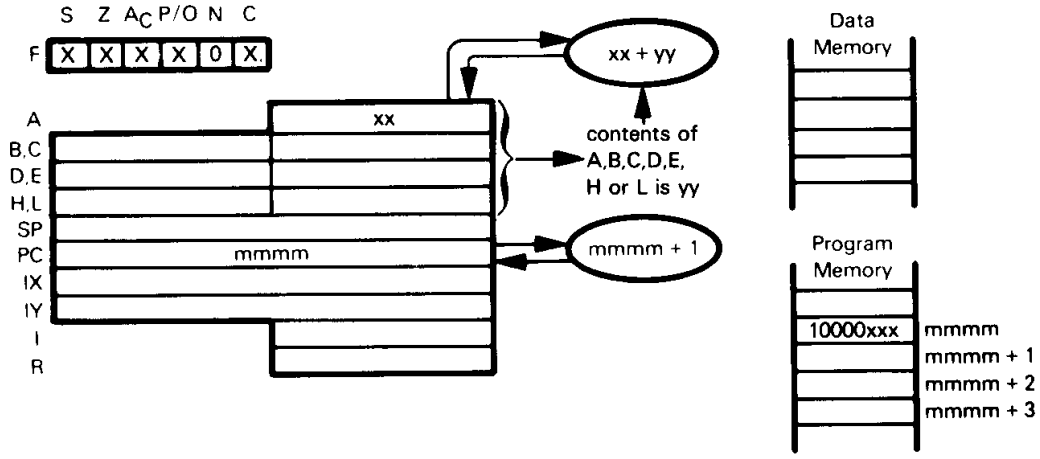
ADD A,7CH

has executed, the Accumulator will contain $B6_{16}$:



This is a routine data manipulation instruction.

ADD A,reg — ADD CONTENTS OF REGISTER TO ACCUMULATOR



ADD reg

10000 xxx

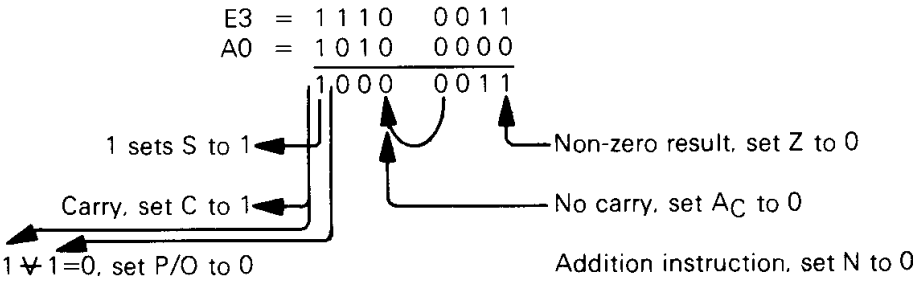
000	for reg=B
001	for reg=C
010	for reg=D
011	for reg=E
100	for reg=H
101	for reg=L
111	for reg=A

Add the contents of Register A, B, C, D, E, H or L to the Accumulator.

Suppose xx=$E3_{16}$, Register E contains $A0_{16}$. After execution of

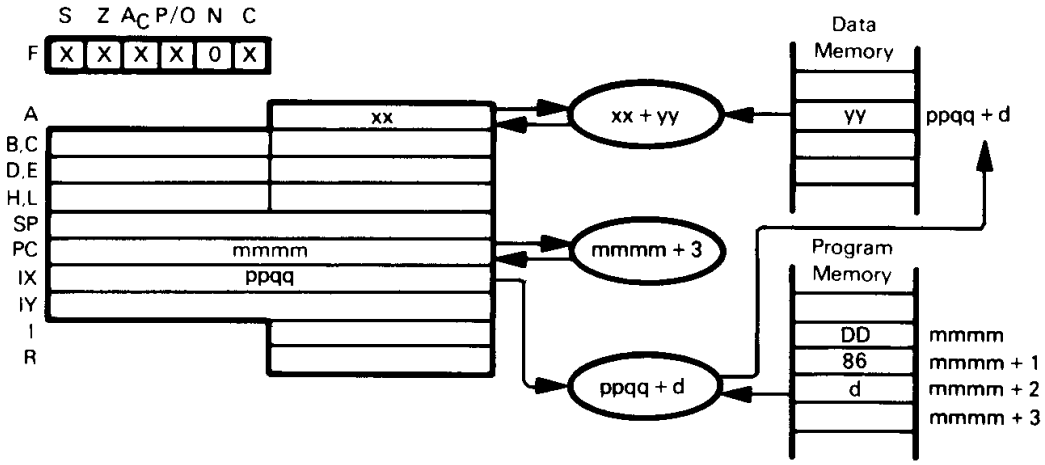
ADD A,E

the Accumulator will contain 83_{16}:



This is a routine data manipulation instruction

ADD A,(HL) — ADD MEMORY TO ACCUMULATOR
ADD A,(IX+disp)
ADD A,(IY+disp)



The illustration shows execution of ADD A,(IX+disp).

ADD A,(IX+disp)

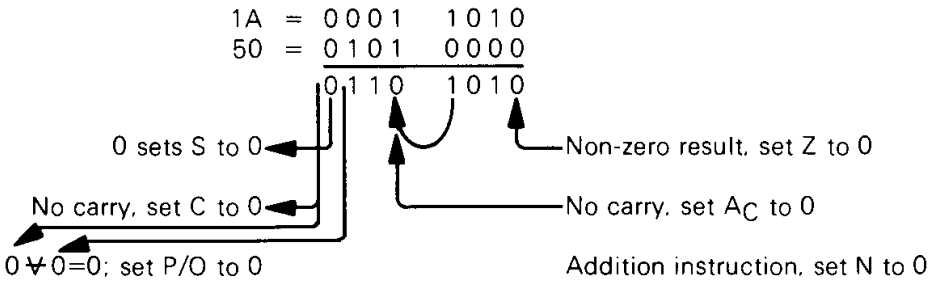
DD 86 d

Add the contents of memory location (specified by the sum of the contents of the IX register and the displacement digit d) to the contents of the Accumulator.

Suppose ppqq=4000_{16}, xx=$1A_{16}$, and memory location $400F_{16}$ contains 50_{16}. After the instruction

ADD A,(IX+0FH)

has executed, the Accumulator will contain $6A_{16}$.



ADD A,(IY+disp)

FD 86 d

This instruction is identical to ADD A,(IX+disp), except that it uses the IY register instead of the IX register.

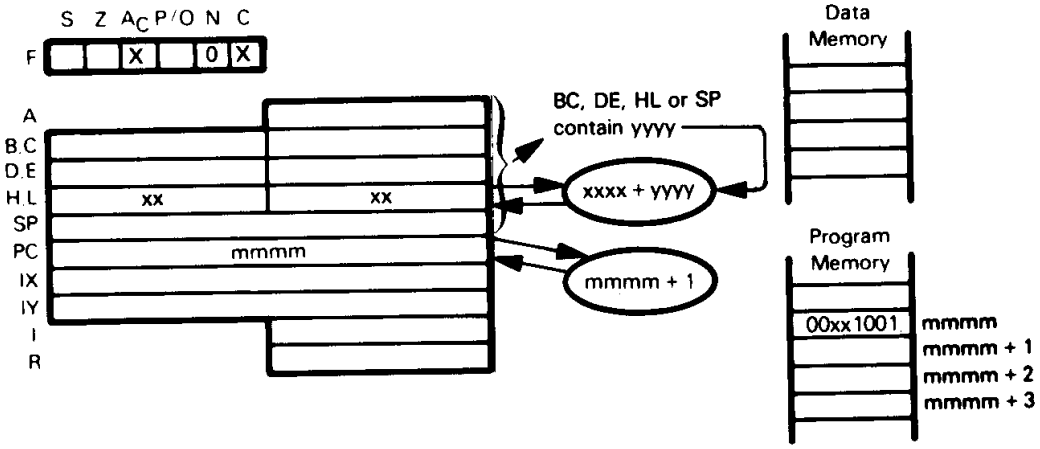
ADD A,(HL)

86

This version of the instruction adds the contents of memory location, specified by the contents of the HL register pair, to the Accumulator.

The ADD instruction is a routine data manipulation instruction.

ADD HL,rp — ADD REGISTER PAIR TO H AND L



00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Add the 16-bit value from either the BC, DE, HL register pair or the Stack Pointer to the HL register pair.

Suppose HL contains $034A_{16}$ and BC contains $214C_{16}$. After the instruction

ADD HL,BC

has executed, the HL register pair will contain 2496_{16}.

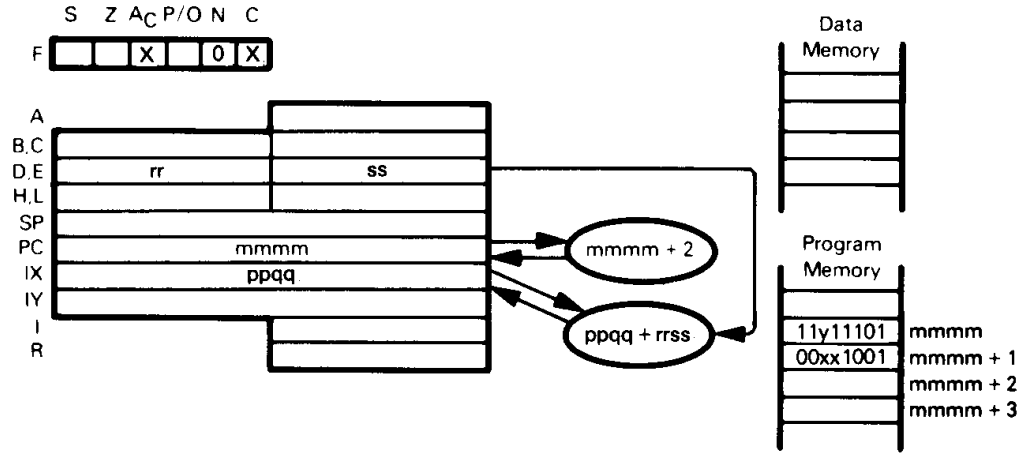
```
034A = 0000 0011 0100 1010
214C = 0010 0001 0100 1100
       ---------------------
       0010 0100 1001 0110
```

No carry, set C to 0 ← | → No carry, set A_C to 0

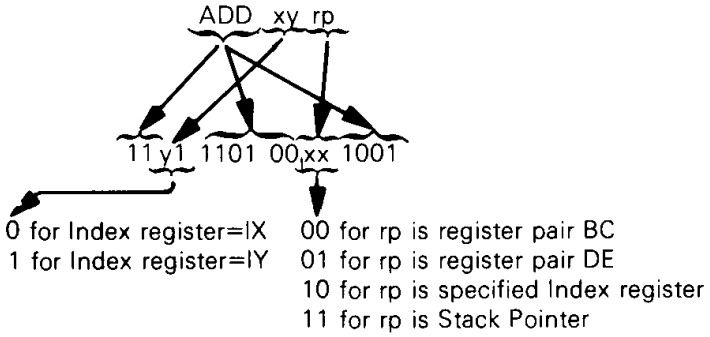
Addition instruction, set N to 0

The ADD HL,HL instruction is equivalent to a 16-bit left shift.

ADD xy,rp — ADD REGISTER PAIR TO INDEX REGISTER



The illustration shows execution of ADD IX,DE.



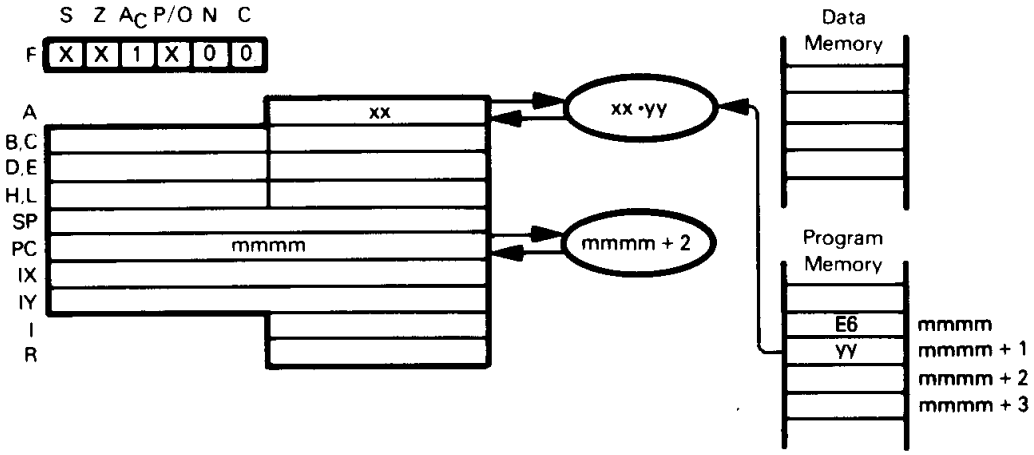
Add the contents of the specified register pair to the contents of the specified Index register.

Suppose IY contains $4FF0_{16}$ and BC contains $000F_{16}$. After the instruction

ADD IY,BC

has executed, Index Register IY will contain $4FFF_{16}$.

AND data — AND IMMEDIATE WITH ACCUMULATOR



AND data

E6 yy

AND the contents of the next program memory byte to the Accumulator.

Suppose xx=$3A_{16}$. After the instruction

AND 7CH

has executed, the Accumulator will contain 38_{16}.

```
3A = 0011  1010
7C = 0111  1100
     0011  1000
```

0 sets S to 0

Three 1 bits, set P/O to 0

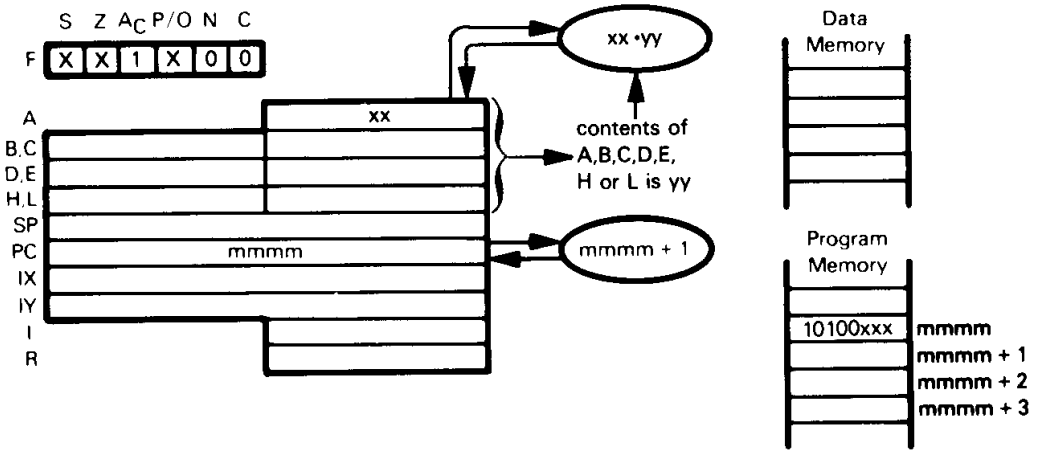
Non-zero result, set Z to 0

This is a routine logical instruction; it is often used to turn bits "off". For example, the instruction

AND 7FH

will unconditionally set the high order Accumulator bit to 0.

AND reg — AND REGISTER WITH ACCUMULATOR



AND reg

10100 xxx

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

AND the Accumulator with the contents of Register A, B, C, D, E, H or L. Save the result in the Accumulator.

Suppose xx=$E3_{16}$, and Register E contains $A0_{16}$. After the instruction

AND E

has executed, the Accumulator will contain $A0_{16}$.

```
E3 = 1 1 1 0   0 0 1 1
A0 = 1 0 1 0   0 0 0 0
     1 0 1 0   0 0 0 0
```

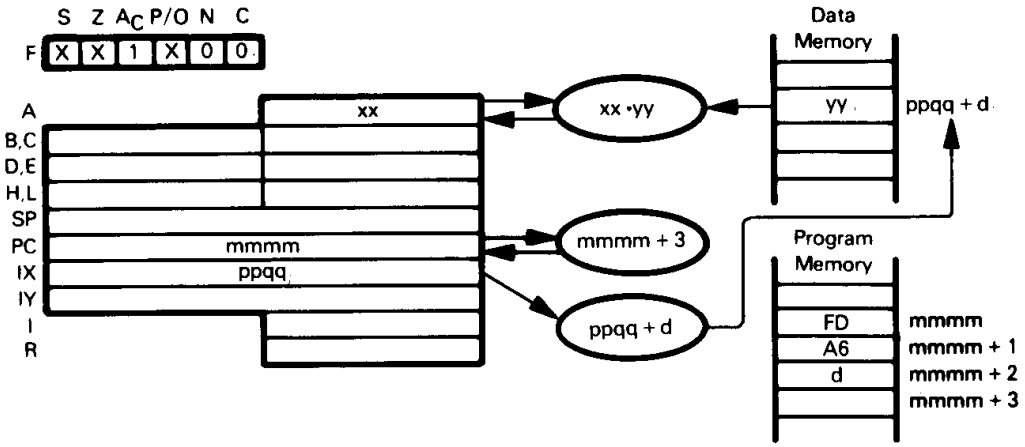
1 sets S to 1

Two 1 bits, set P/O to 1

Non-zero result, set Z to 0

AND is a frequently used logical instruction.

AND (HL) — AND MEMORY WITH ACCUMULATOR
AND (IX+disp)
AND (IY+disp)



The illustration shows execution of AND (IY+disp).

AND (IY+disp)
FD A6 d

AND the contents of memory location (specified by the sum of the contents of the IY register and the displacement digit d) with the Accumulator.

Suppose xx=$E3_{16}$, ppqq=4000_{16}, and memory location $400F_{16}$ contains $A0_{16}$. After the instruction

AND (IY+0FH)

has executed, the Accumulator will contain $A0_{16}$.

```
E3 = 1 1 1 0   0 1 1 1
A0 = 1 0 1 0   0 0 0 0
     1 0 1 0   0 0 0 0
```

1 sets S to 1

Two 1 bits, set P/O to 1

Non-zero result, set Z to 0

AND (IX+disp)
DD A6 d

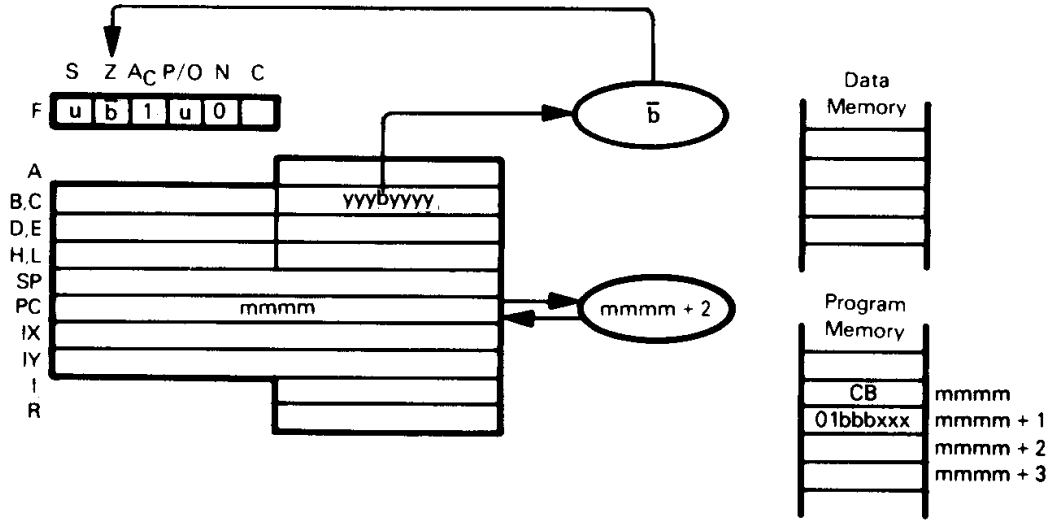
This instruction is identical to AND (IY+disp), except that it uses the IX register instead of the IY register.

AND (HL)
A6

AND the contents of the memory location (specified by the contents of the HL register pair) with the Accumulator.

AND is a frequently used logical instruction.

BIT b,reg — TEST BIT b IN REGISTER reg

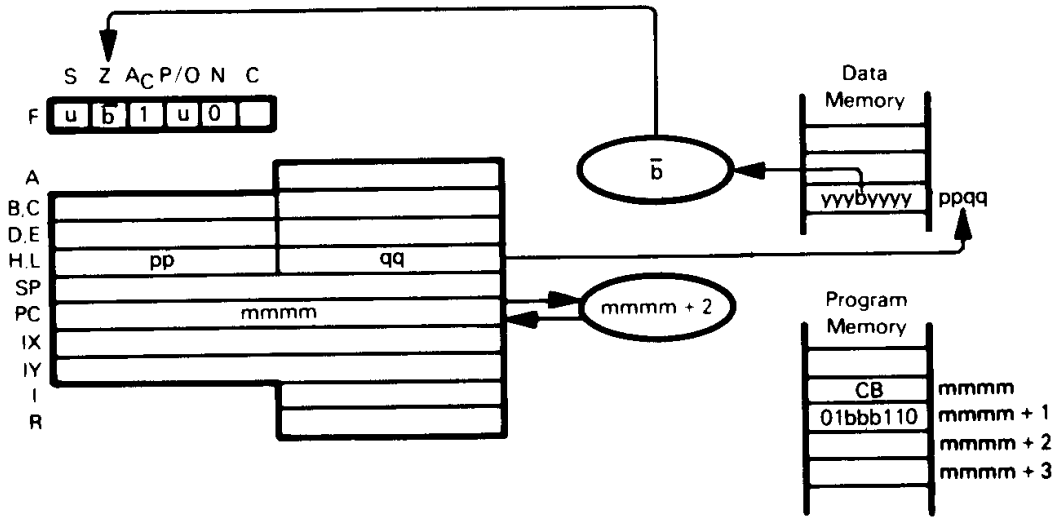


BIT	b,	reg	
CB 01	bbb	xxx	
Bit Tested			Register
0	000	000	B
1	001	001	C
2	010	010	D
3	011	011	E
4	100	100	H
5	101	101	L
6	110	111	A
7	111		

Place complement of indicated register's specified bit in Z flag of F register.

Suppose Register C contains 1110 1111. The instruction BIT 4,C will then set the Z flag to 1, while bit 4 in Register C remains 0. Bit 0 is the least significant bit.

BIT b,(HL) — TEST BIT b OF INDICATED MEMORY POSITION
BIT b,(IX+disp)
BIT b,(IY+disp)



The illustration shows execution of BIT 4,(HL). Bit 0 is the least significant bit.

BIT	b,	(HL)
CB 01	bbb	110

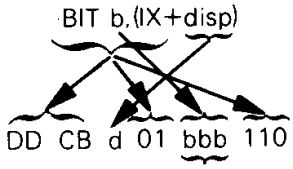
Bit Tested	bbb
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Test indicated bit within memory position specified by the contents of Register HL, and place bit's complement in Z flag of the F register.

Suppose HL contains 4000H and bit 3 in memory location 4000H contains 1. The instruction

BIT 3,(HL)

will then set the Z flag to 0, while bit 3 in memory location 4000H remains 1.



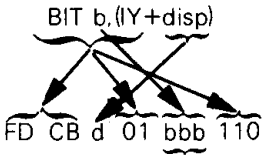
bbb is the same as in BIT b,(HL)

Examine specified bit within memory location indicated by the sum of Index Register IX and disp. Place the complement in the Z flag of the F register.

Suppose Index Register IX contains 4000H and bit 4 of memory location 4004H is 0. The instruction

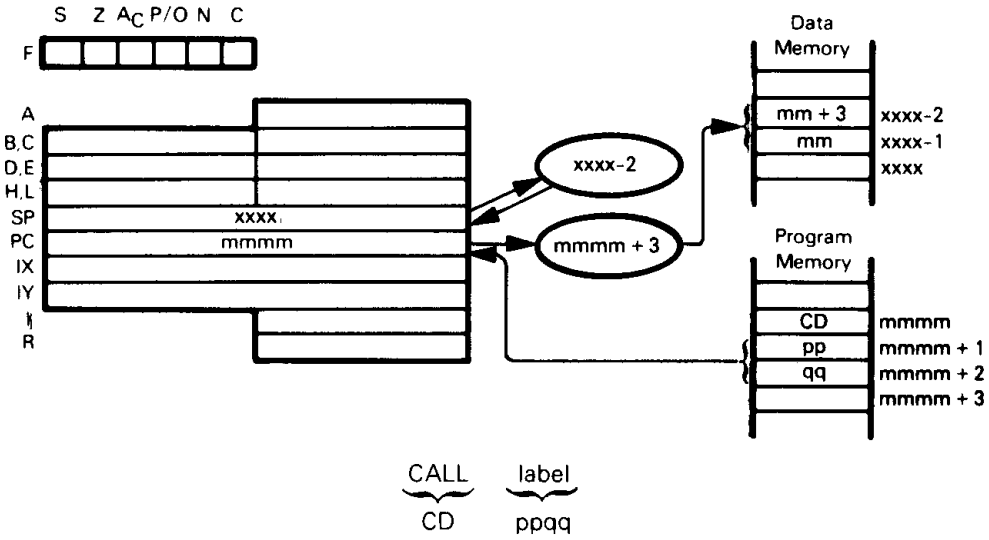
BIT 4,(IX+4H)

will then set the Z flag to 1, while bit 4 of memory location 4004H remains 0.



This instruction is identical to BIT b,(IX+disp), except that it uses the IY register instead of the IX register.

CALL label — CALL THE SUBROUTINE IDENTIFIED IN THE OPERAND



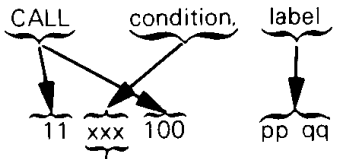
Store the address of the instruction following the CALL on the top of the stack: the top of the stack is a data memory byte addressed by the Stack Pointer. Then subtract 2 from the Stack Pointer in order to address the new top of stack. Move the 16-bit address contained in the second and third CALL instruction object program bytes to the Program Counter. The second byte of the CALL instruction is the low-order half of the address, and the third byte is the high-order byte.

Consider the instruction sequence:

```
        CALL    SUBR
        AND     7CH
        -
        -
        -
SUBR
```

After the instruction has executed, the address of the AND instruction is saved at the top of the stack. The Stack Pointer is decremented by 2. The instruction labeled SUBR will be executed next.

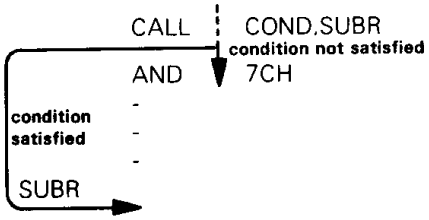
CALL condition,label — CALL THE SUBROUTINE IDENTIFIED IN THE OPERAND IF CONDITION IS SATISFIED



	Condition		Relevant Flag
000	NZ	Non-Zero	Z
001	Z	Zero	Z
010	NC	Non-Carry	C
011	C	Carry	C
100	PO	Parity Odd	P/O
101	PE	Parity Even	P/O
110	P	Sign Positive	S
111	M	Sign Negative	S

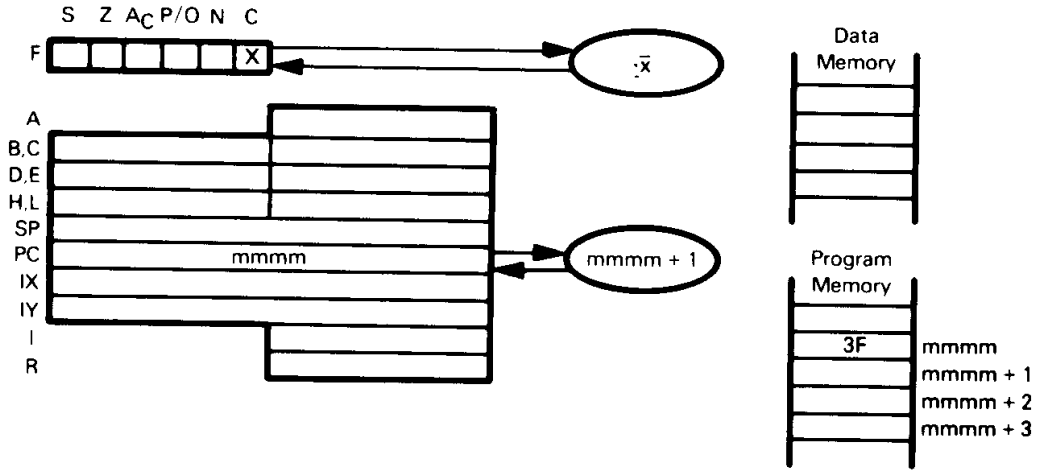
This instruction is identical to the CALL instruction, except that the identified subroutine will be called only if the condition is satisfied; otherwise, the instruction sequentially following the CALL condition instruction will be executed.

Consider the instruction sequence:



If the condition is not satisfied, the AND instruction will be executed after the CALL COND,SUBR instruction has executed. If the condition is satisfied, the address of the AND instruction is saved at the top of the stack, and the Stack Pointer is decremented by 2. The instruction labeled SUBR will be executed next.

CCF — COMPLEMENT CARRY FLAG

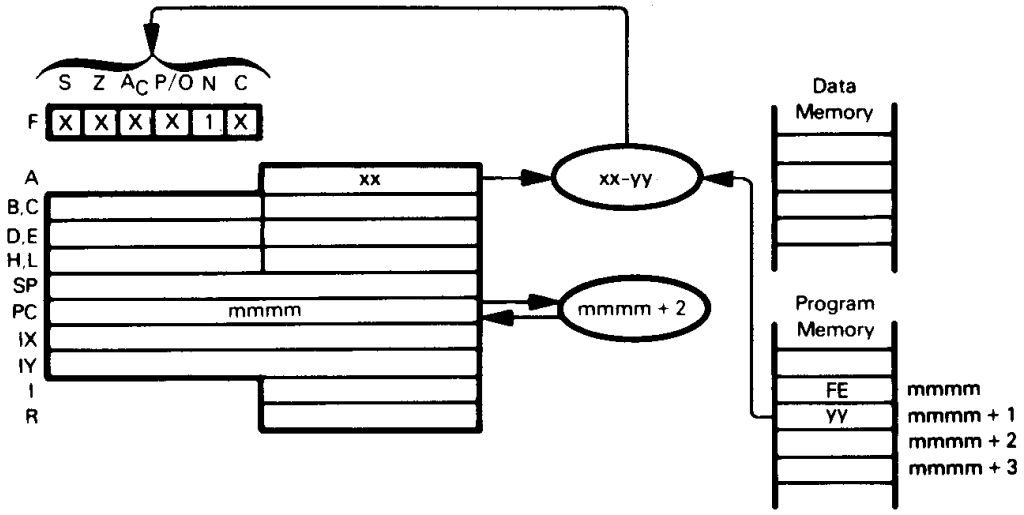


CCF

3F

Complement the Carry flag. No other status or register contents are affected.

CP data — COMPARE IMMEDIATE DATA WITH ACCUMULATOR



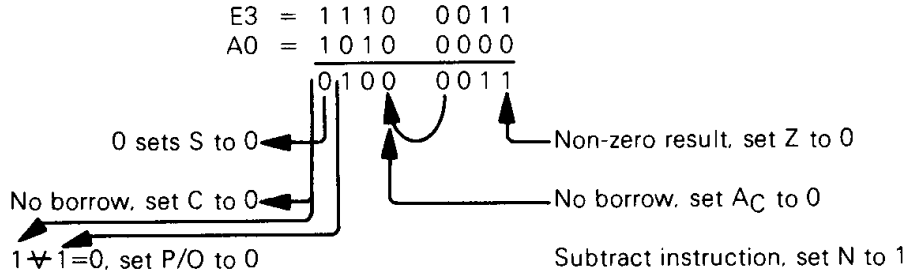
CP data
FE yy

Subtract the contents of the second object code byte from the contents of the Accumulator, treating both numbers as simple binary data. Discard the result; i.e., leave the Accumulator alone, but modify the status flags to reflect the result of the subtraction.

Suppose xx=$E3_{16}$ and the second byte of the CP instruction object code contains $A0_{16}$. After the instruction

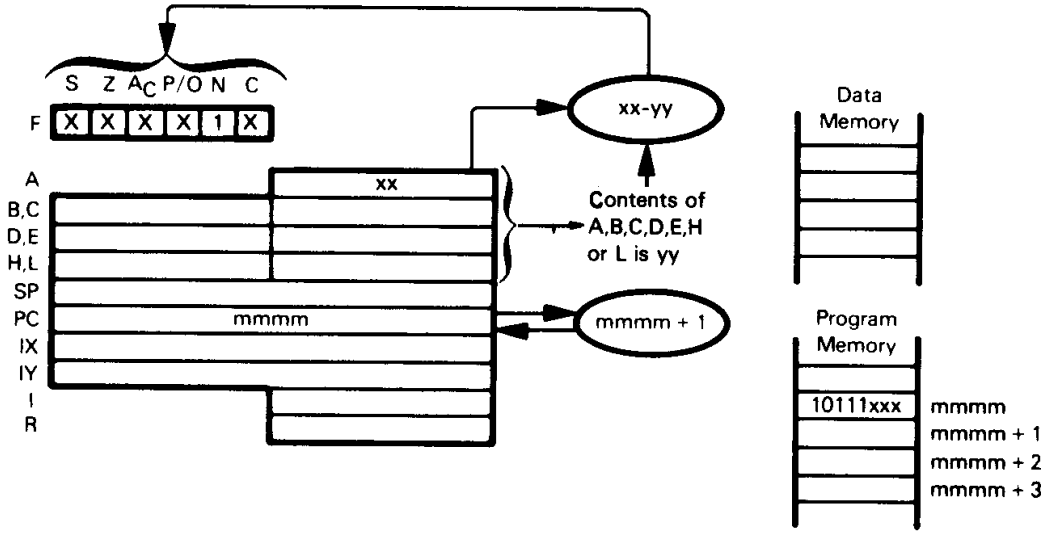
CP 0A0H

has executed, the Accumulator will still contain $E3_{16}$, but statuses will be modified as follows:



Notice that the resulting carry is complemented.

CP reg — COMPARE REGISTER WITH ACCUMULATOR



CP reg

10111 xxx

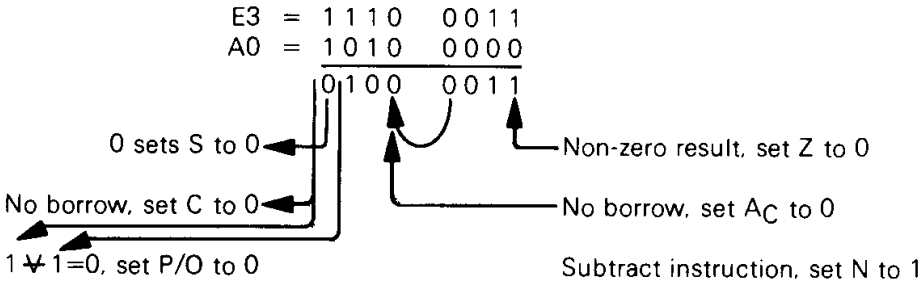
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Subtract the contents of Register A, B, C, D, E, H or L from the contents of the Accumulator, treating both numbers as simple binary data. Discard the result; i.e., leave the Accumulator alone, but modify status flags to reflect the result of the subtraction.

Suppose xx=$E3_{16}$ and Register B contains $A0_{16}$. After the instruction

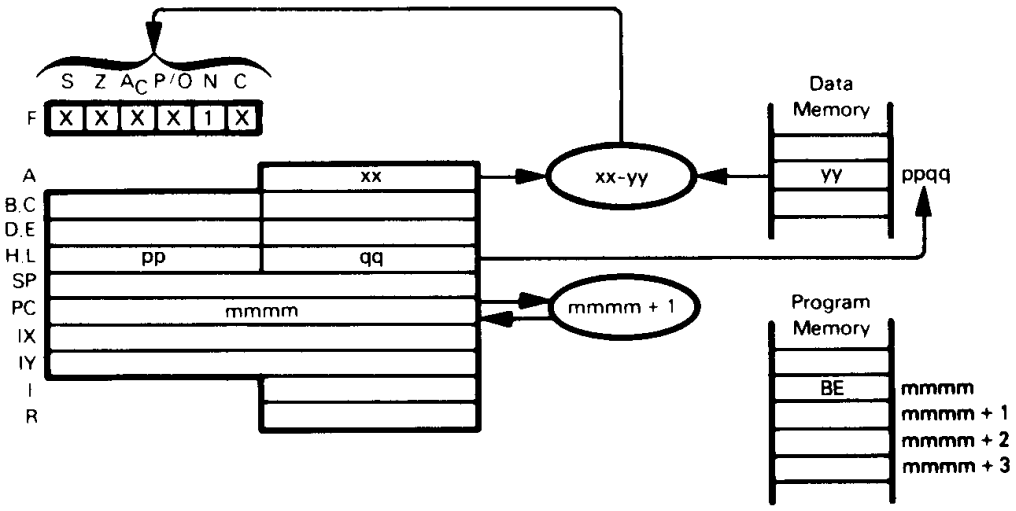
CP B

has executed, the Accumulator will still contain $E3_{16}$, but statuses will be modified as follows:



Notice that the resulting carry is complemented.

CP (HL) — COMPARE MEMORY WITH ACCUMULATOR
CP (IX+disp)
CP (IY+disp)



The illustration shows execution of CP (HL):

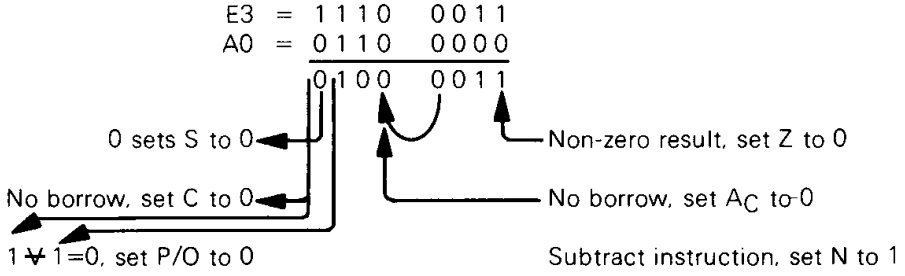
CP (HL)
BE

Subtract the contents of memory location (specified by the contents of the HL register pair) from the contents of the Accumulator, treating both numbers as simple binary data. Discard the result; i.e., leave the Accumulator alone, but modify status flags to reflect the result of the subtraction.

Suppose xx=$E3_{16}$ and yy=$A0_{16}$. After execution of

CP (HL)

the Accumulator will still contain $E3_{16}$, but statuses will be modified as follows:



Notice that the resulting carry is complemented.

CP (IX+disp)
DD BE d

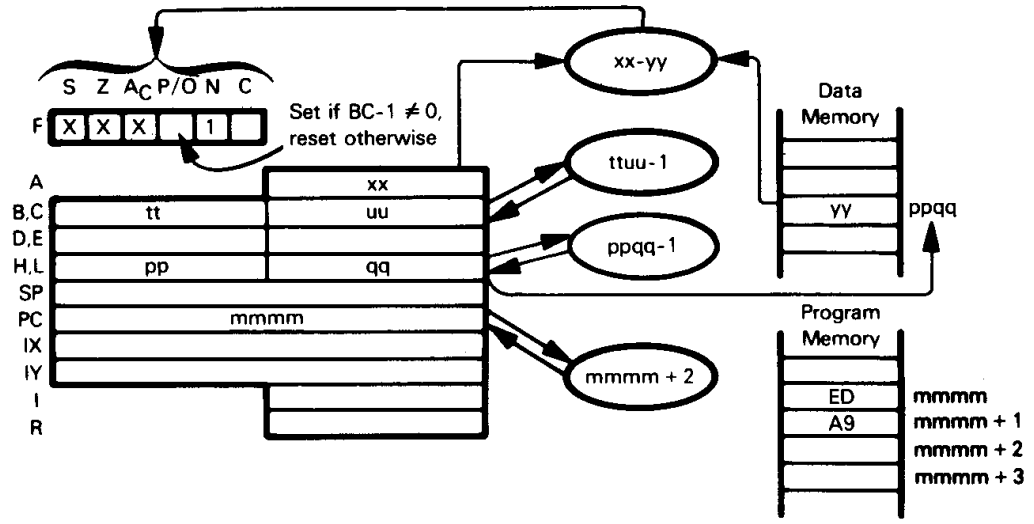
Subtract the contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) from the contents of the Accumulator, treating both numbers as simple binary data. Discard the result; i.e., leave the Accumulator alone, but modify status flags to reflect the result of the subtraction.

CP (IY+disp)

FD BE d

This instruction is identical to CP (IX+disp), except that it uses the IY register instead of the IX register.

CPD — COMPARE ACCUMULATOR WITH MEMORY. DECREMENT ADDRESS AND BYTE COUNTER



CPD

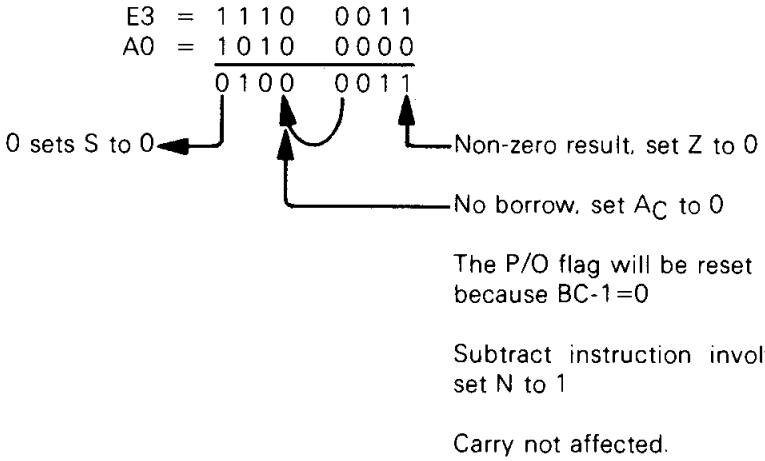
ED A9

Compare the contents of the Accumulator with the contents of memory location (specified by the HL register pair). If A is equal to memory, set Z flag. Decrement the HL and BC register pairs. (BC is used as the Byte Counter.)

Suppose xx=$E3_{16}$, ppqq=4000_{16}, BC contains 0001_{16}, and yy=$A0_{16}$. After the instruction

CPD

has executed, the Accumulator will still contain $E3_{16}$, but statuses will be modified as follows:



The HL register pair will contain $3FFF_{16}$, and BC=0.

CPDR — COMPARE ACCUMULATOR WITH MEMORY. DECREMENT ADDRESS AND BYTE COUNTER. CONTINUE UNTIL MATCH IS FOUND OR BYTE COUNTER IS ZERO

CPDR
ED B9

This instruction is identical to CPD, except that it is repeated until a match is found or the byte counter is zero. After each data transfer, interrupts will be recognized and two refresh cycles will be executed.

Suppose the HL register pair contains 5000_{16}, the BC register pair contains $00FF_{16}$, the Accumulator contains $F9_{16}$, and memory has contents as follows:

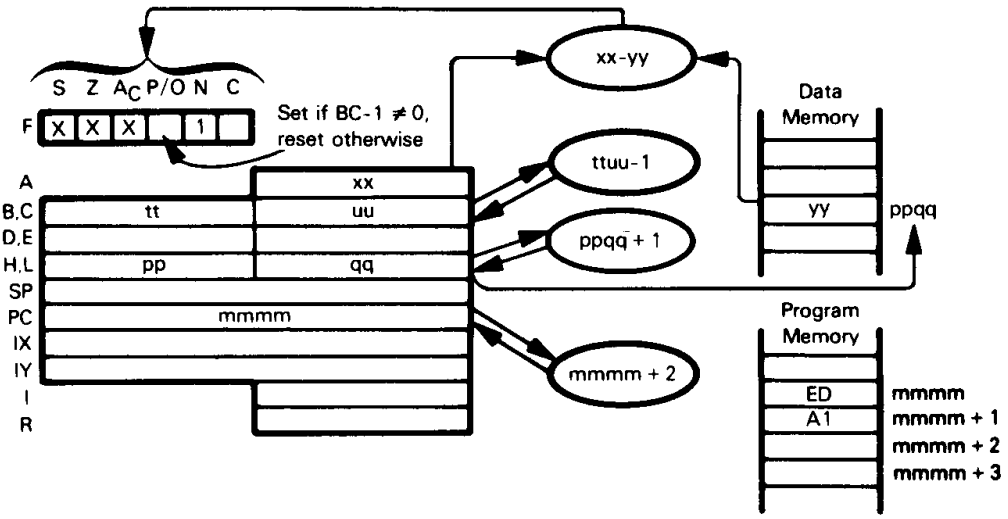
Location	Contents
5000_{16}	AA_{16}
$4FFF_{16}$	BC_{16}
$4FFE_{16}$	19_{16}
$4FFD_{16}$	$7A_{16}$
$4FFC_{16}$	$F9_{16}$
$4FFB_{16}$	DD_{16}

After execution of

CPDR

the P/O flag will be 1, the Z flag will be 1, the HL register pair will contain $4FFB_{16}$, and the BC register pair will contain $00FA_{16}$.

CPI — COMPARE ACCUMULATOR WITH MEMORY. DECREMENT BYTE COUNTER. INCREMENT ADDRESS



CPI
ED A1

Compare the contents of the Accumulator with the contents of memory location (specified by the HL register pair). If A is equal to memory, set the Z flag. Increment the HL register pair and decrement the BC register pair (BC is used as Byte Counter).

Suppose xx=$E3_{16}$, ppqq=4000_{16}, BC contains 0032_{16}, and yy=$E3_{16}$. After the instruction

CPI

has executed, the Accumulator will still contain $E3_{16}$, but statuses will be modified as follows:

```
E3  =  1111  0011
-E3 =  0000  1101
       0000  0000
```

0 sets S to 0

Result is 0, set Z to 1

No borrow, set A_C to 0

The P/O flag will be set because BC-1 ≠ 0.

Subtract instruction involved, set N to 1.

Carry not affected.

The HL register pair will contain 4001_{16}, and BC will contain 0031_{16}.

CPIR — COMPARE ACCUMULATOR WITH MEMORY. DECREMENT BYTE COUNTER. INCREMENT ADDRESS. CONTINUE UNTIL MATCH IS FOUND OR BYTE COUNTER IS ZERO

CPIR

ED B1

This instruction is identical to CPI, except that it is repeated until a match is found or the byte counter is zero. After each data transfer interrupts will be recognized and two refresh cycles will be executed.

Suppose the HL register pair contains 4500_{16}, the BC register pair contains $00FF_{16}$, the Accumulator contains $F9_{16}$, and memory has contents as follows:

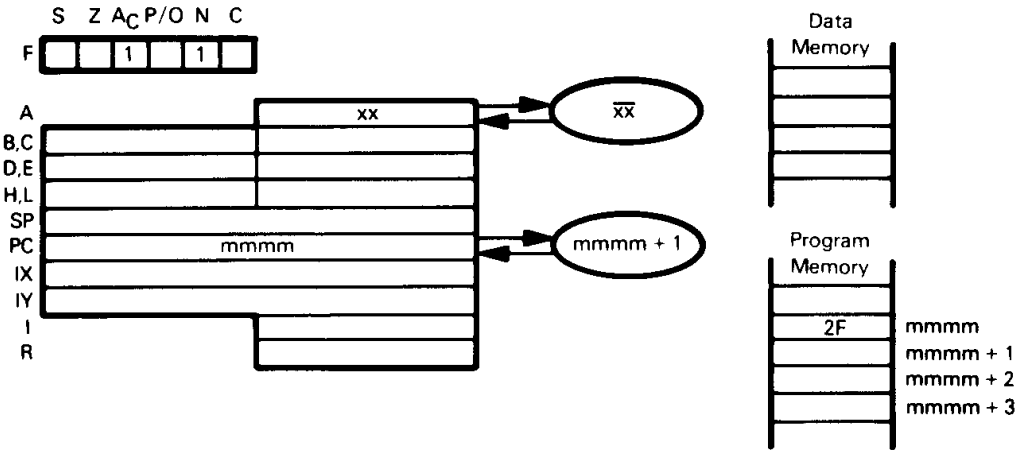
Location	Contents
4500_{16}	AA_{16}
4501_{16}	15_{16}
4502_{16}	$F9_{16}$

After execution of

CPIR

the P/O flag will be 1, and the Z flag will be 1. The HL register pair will contain 4503_{16}, and the BC register pair will contain $00FC_{16}$.

CPL — COMPLEMENT THE ACCUMULATOR



CPL
2F

Complement the contents of the Accumulator. No other register's contents are affected.

Suppose the Accumulator contains $3A_{16}$. After the instruction

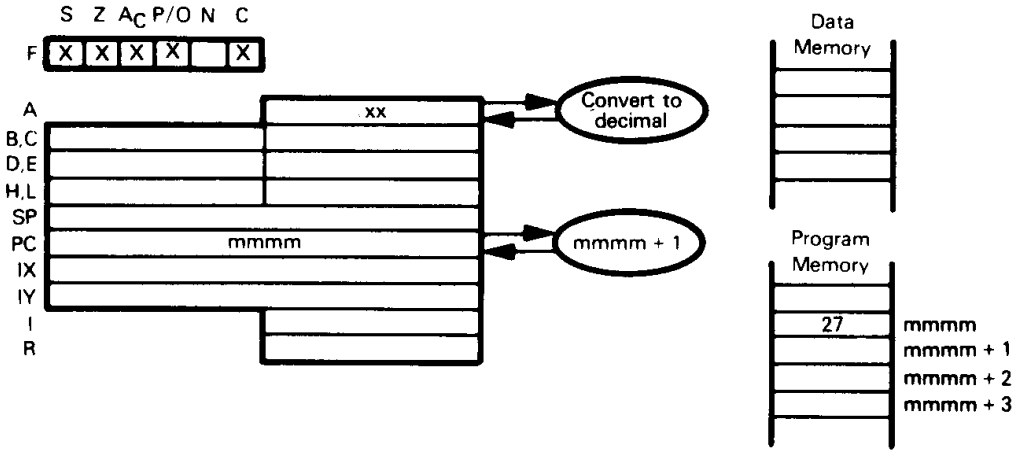
CPL

has executed, the Accumulator will contain $C5_{16}$.

3A = 0 0 1 1 1 0 1 0
Complement = 1 1 0 0 0 1 0 1

This is a routine logical instruction. You need not use it for binary subtraction; there are special subtract instructions (SUB, SBC).

DAA — DECIMAL ADJUST ACCUMULATOR



DAA
27

Convert the contents of the Accumulator to binary-coded decimal form. This instruction should only be used after adding or subtracting two BCD numbers; i.e., look upon ADD DAA or ADC DAA or INC DAA or SUB DAA or SBC DAA or DEC DAA or NEG DAA as compound, decimal arithmetic instructions which operate on BCD sources to generate BCD answers.

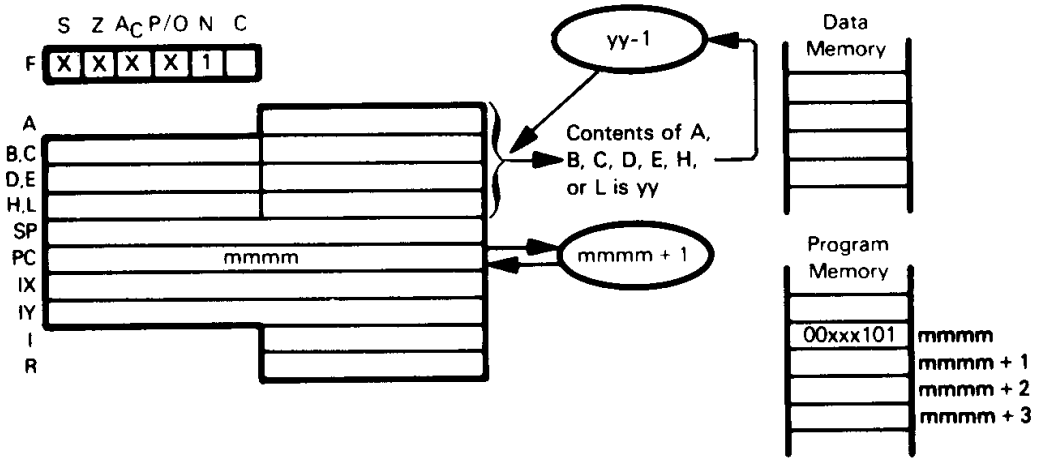
Suppose the Accumulator contains 39_{16} and the B register contains 47_{16}. After the instructions

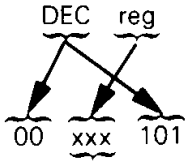
ADD B
DAA

have executed, the Accumulator will contain 86_{16}, not 80_{16}.

Z80 CPU logic uses the values in the Carry and Auxiliary Carry, as well as the Accumulator contents, in the Decimal Adjust operation.

DEC reg — DECREMENT REGISTER CONTENTS





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

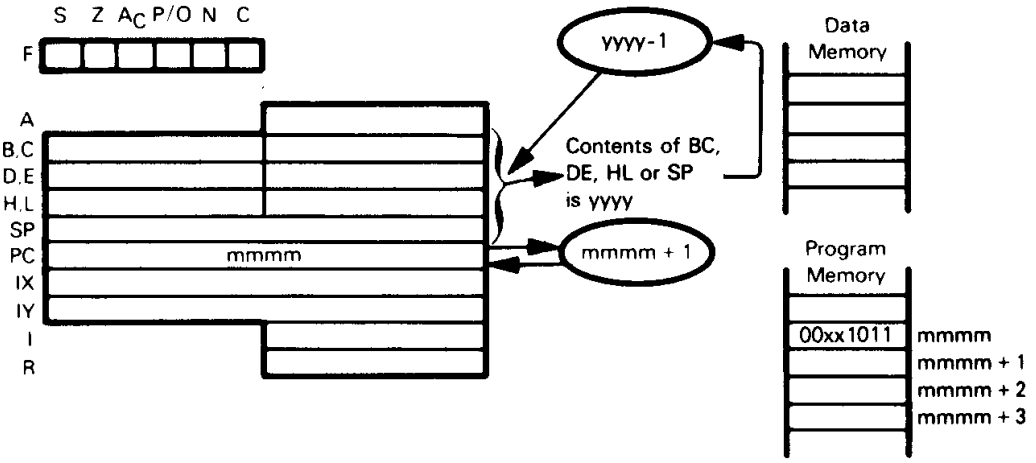
Subtract 1 from the contents of the specified register.

Suppose Register A contains 50_{16}. After execution of

DEC A

Register A will contain $4F_{16}$.

DEC rp — DECREMENT CONTENTS OF SPECIFIED REGISTER PAIR
DEC IX
DEC IY



The illustration shows execution of DEC rp:

DEC rp

00 xx 1011

00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Subtract 1 from the 16-bit value contained in the specified register pair. No status flags are affected.

Suppose the H and L registers contain $2F00_{16}$. After the instruction

DEC HL

has executed, the H and L registers will contain $2EFF_{16}$.

DEC IX

DD 2B

Subtract 1 from the 16-bit value contained in the IX register.

DEC IY

FD 2B

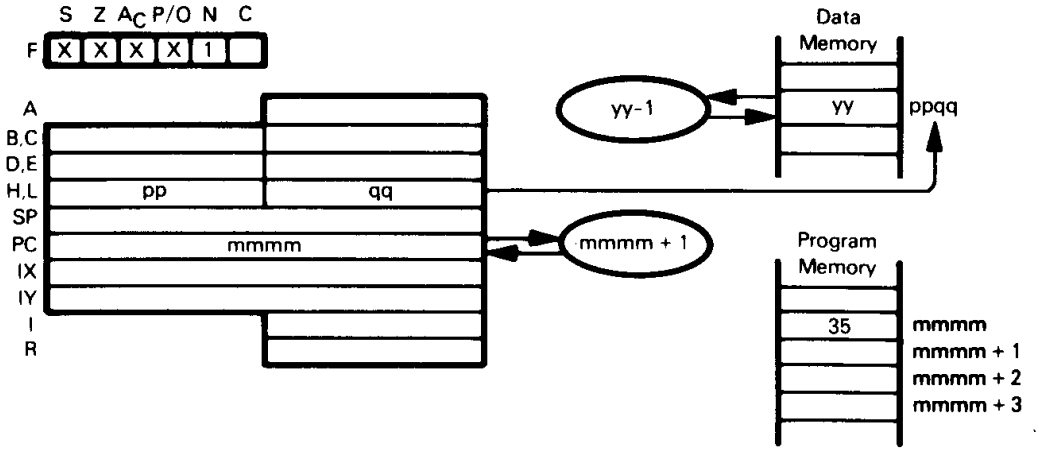
Subtract 1 from the 16-bit value contained in the IY register.

Neither DEC rp, DEC IX nor DEC IY affects any of the status flags. This is a defect in the Z80 instruction set, inherited from the 8080. Whereas the DEC reg instruction is used in iterative instruction loops that use a counter with a value of 256 or less, the DEC rp (DEC IX or DEC IY) instruction must be used if the counter value is more than 256. Since the DEC rp instruction sets no status flags, other instructions must be added to simply

test for a zero result. This is a typical loop form:

```
        LD      DE,DATA   ;LOAD INITIAL 16-BIT COUNTER VALUE
LOOP    -                 ;FIRST INSTRUCTION OF LOOP
        -
        -
        DEC     DE        ;DECREMENT COUNTER
        LD      A,D       ;TO TEST FOR ZERO, MOVE D TO A
        OR      E         ;THEN OR A WITH E
        JP      NZ,LOOP   ;RETURN IF NOT ZERO
```

DEC (HL) — DECREMENT MEMORY CONTENTS
DEC (IX+disp)
DEC (IY+disp)



The illustration shows execution of DEC (HL):

DEC (HL)
35

Subtract 1 from the contents of memory location (specified by the contents of the HL register pair).

Suppose ppqq=4500_{16}, yy=$5F_{16}$. After execution of

DEC (HL)

memory location 4500_{16} will contain $5E_{16}$.

```
5F  =  0 1 0 1   1 1 1 1
-01 =  1 1 1 1   1 1 1 1
       0 1 0 1   1 1 1 0
```

0 sets S to 0

Non-zero result, set Z to 0

1 ⊻ 1=0, set P/O to 0

No borrow, set A_C to 0

Subtract instruction, set N to 1

DEC (IX+disp)

DD 35 d

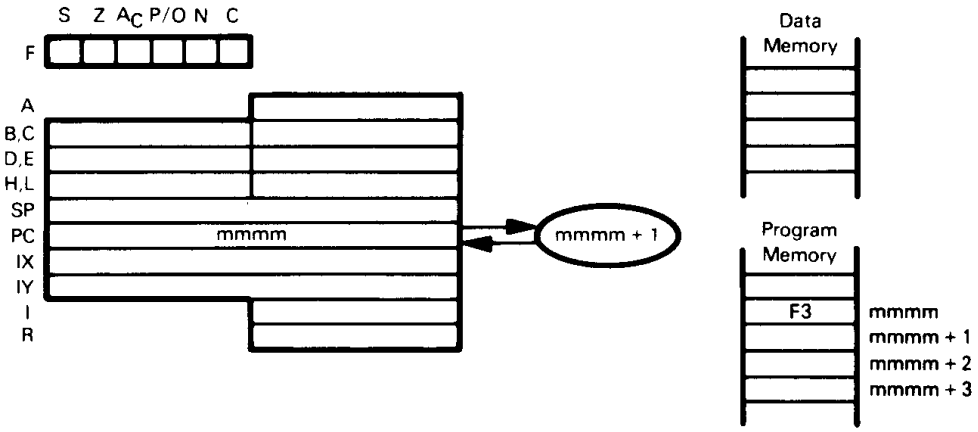
Subtract 1 from the contents of memory location (specified by the sum of the contents of the IX register and the displacement value d).

DEC (IY+disp)

FD 35 d

This instruction is identical to DEC (IX+disp), except that it uses the IY register instead of the IX register.

DI — DISABLE INTERRUPTS



DI

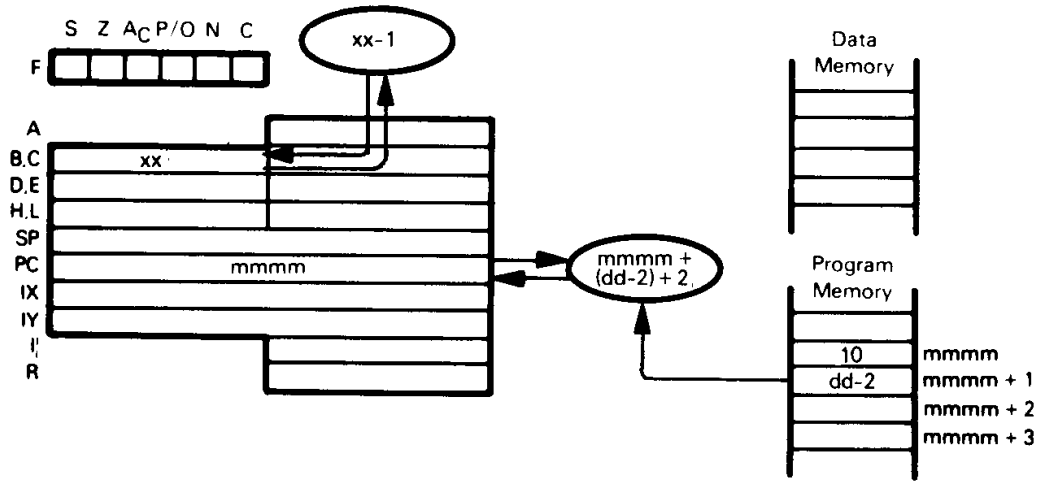
F3

When this instruction is executed, the maskable interrupt request is disabled and the $\overline{INT}$ input to the CPU will be ignored. Remember that when an interrupt is acknowledged, the maskable interrupt is automatically disabled.

The maskable interrupt request remains disabled until it is subsequently enabled by an EI instruction.

No registers or flags are affected by this instruction.

DJNZ disp — JUMP RELATIVE TO PRESENT CONTENTS OF PROGRAM COUNTER IF REG B IS NOT ZERO



DJNZ disp

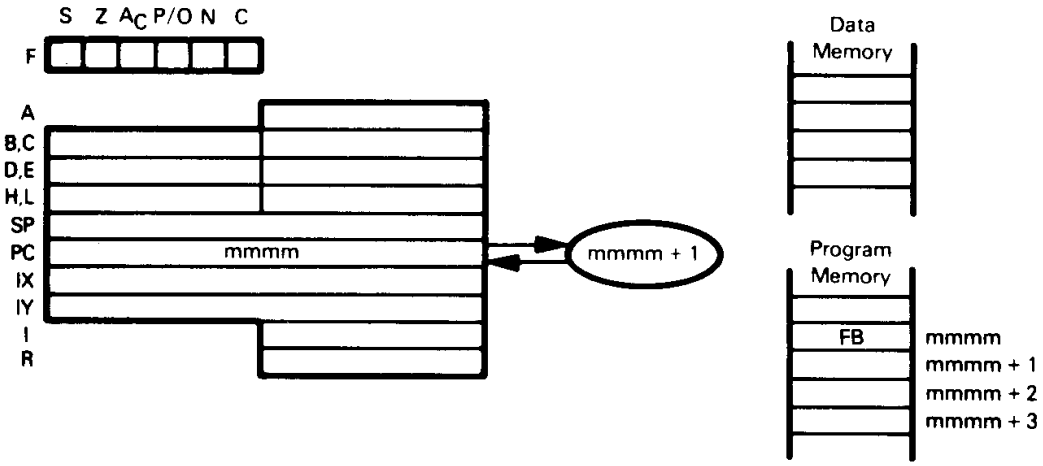
10 dd-2

Decrement Register B. If remaining contents are not zero, add the contents of the DJNZ instruction object code second byte and 2 to the Program Counter. The jump is measured from the address of the instruction operation code, and has a range of -126 to +129 bytes. The Assembler automatically adjusts for the twice-incremented PC.

If the contents of B are zero after decrementing, the next sequential instruction is executed.

The DJNZ instruction is extremely useful for any program loop operation, since the one instruction replaces the typical "decrement-then-branch on condition" instruction sequence.

EI — ENABLE INTERRUPTS



EI

FB

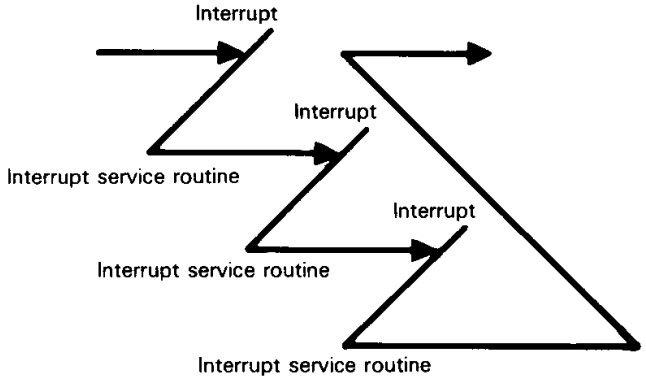
Execution of this instruction causes interrupts to be enabled, but not until one more instruction executes.

Most interrupt service routines end with the two instructions:

```
EI              ;ENABLE INTERRUPTS
RET             ;RETURN TO INTERRUPTED PROGRAM
```

If interrupts are processed serially, then for the entire duration of the interrupt service routine all maskable interrupts are disabled — which means that in a multi-interrupt application there is a significant possibility for one or more interrupts to be pending when any interrupt service routine completes execution.

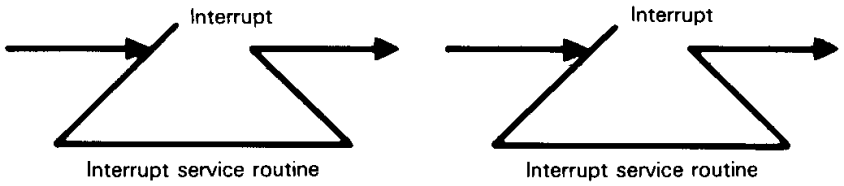
If interrupts were acknowledged as soon as the EI instructions had executed, then the Return instruction would not be executed. Under these circumstances, returns would stack up one on top of the other — and unnecessarily consume stack memory space. This may be illustrated as follows:



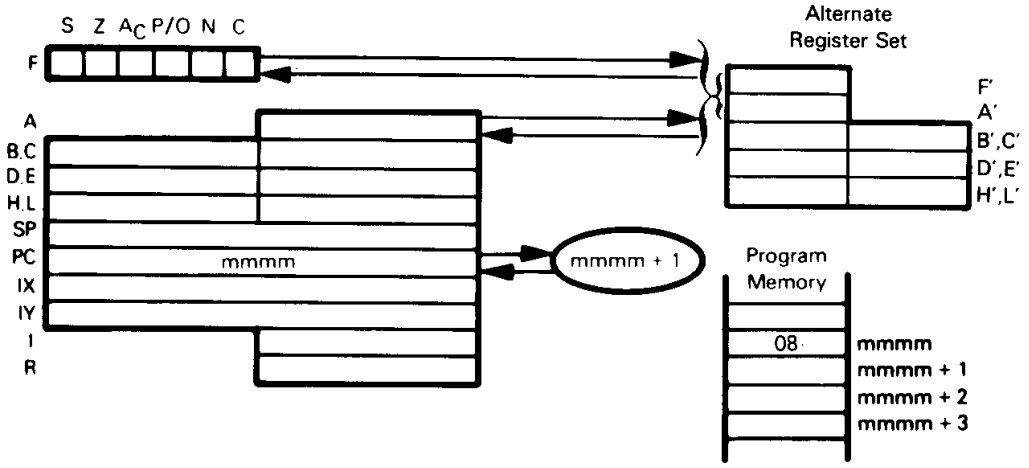
By inhibiting interrupts for one more instruction following execution of EI, the Z80 CPU ensures that the RET instruction gets executed in the sequence:

```
-
-
-
EI              ;ENABLE INTERRUPTS
RET             ;RETURN FROM INTERRUPT
```

It is not uncommon for interrupts to be kept disabled while an interrupt service routine is executing. Interrupts are processed serially:



EX AF,AF' — EXCHANGE PROGRAM STATUS AND ALTERNATE PROGRAM STATUS



EX AF,AF'

08

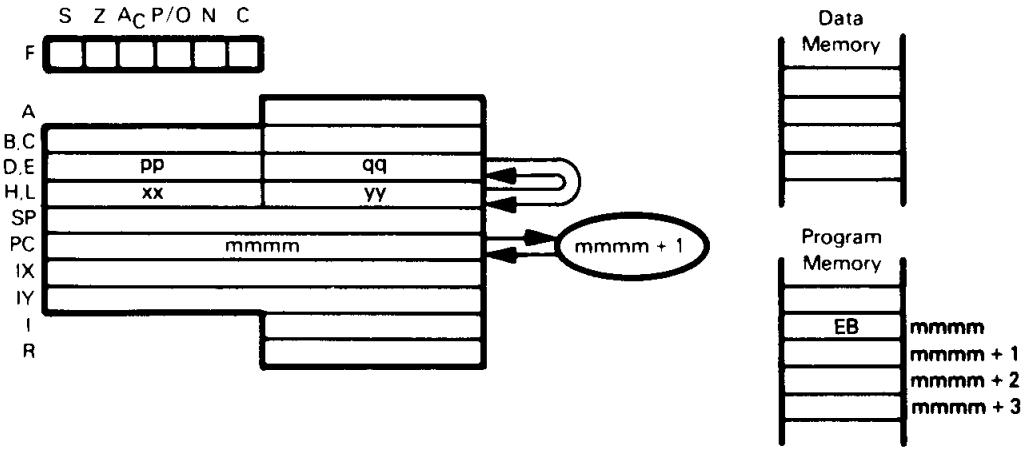
The two-byte contents of register pairs AF and A'F' are exchanged.

Suppose AF contains $4F99_{16}$ and A'F' contains $10AA_{16}$. After execution of

EX AF,AF'

AF will contain $10AA_{16}$ and AF' will contain $4F99_{16}$.

EX DE,HL — EXCHANGE DE AND HL CONTENTS



EX DE,HL
EB

The D and E registers' contents are swapped with the H and L registers' contents.

Suppose pp=03_{16}, qq=$2A_{16}$, xx=41_{16} and yy=FC_{16}. After the instruction

```
EX DE,HL
```

has executed, H will contain 03_{16}, L will contain $2A_{16}$, D will contain 41_{16} and E will contain FC_{16}.

The two instructions:

```
EX DE,HL
LD A,(HL)
```

are equivalent to:

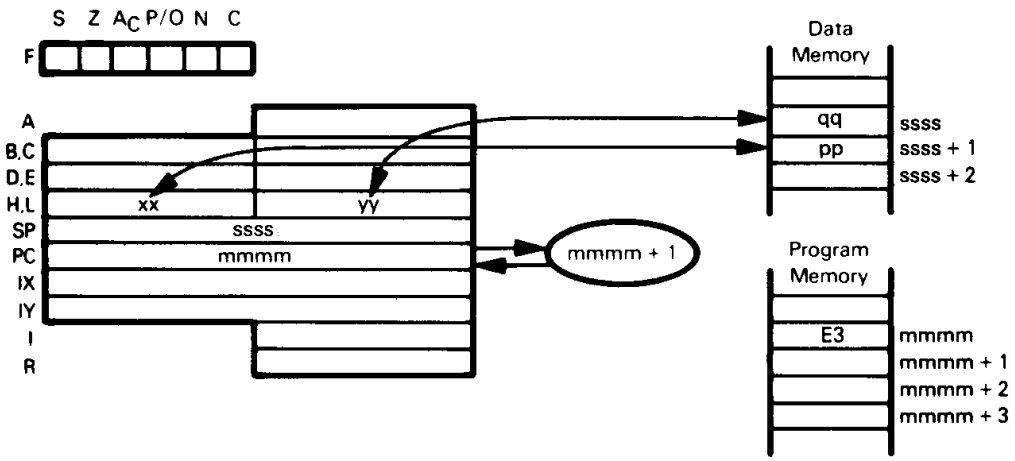
```
LD A,(DE)
```

but if you want to load data addressed by the D and E register into the B register,

```
EX DE,HL
LD B,(HL)
```

has no single instruction equivalent.

EX (SP),HL — EXCHANGE CONTENTS OF REGISTER AND
EX (SP),IX TOP OF STACK
EX (SP),IY



The illustration shows execution of EX (SP),HL.

EX (SP),HL
E3

Exchange the contents of the L register with the top stack byte. Exchange the contents of the H register with the byte below the stack top.

Suppose xx=21_{16}, yy=FA_{16}, pp=$3A_{16}$, qq=$E2_{16}$. After the instruction

EX (SP),HL

has executed, H will contain $3A_{16}$, L will contain $E2_{16}$ and the two top stack bytes will contain FA_{16} and 21_{16} respectively.

The EX (SP),HL instruction is used to access and manipulate data at the top of the stack.

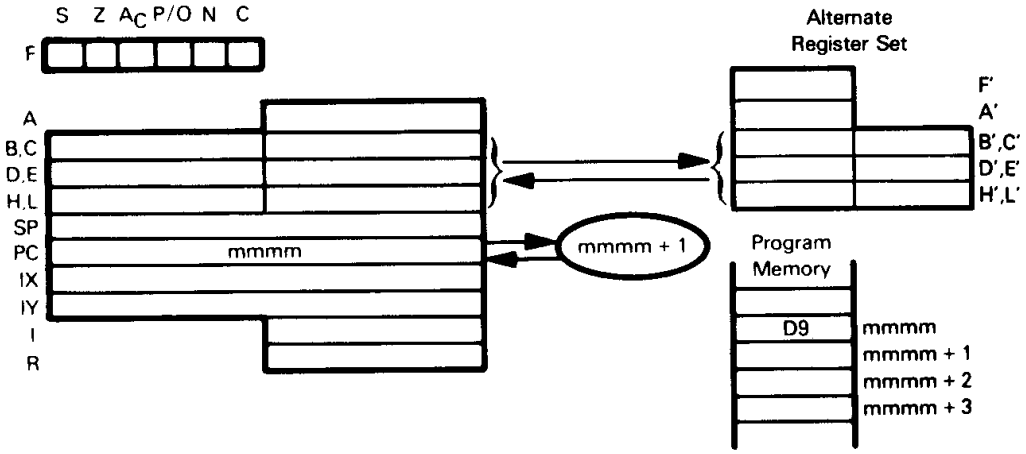
EX (SP),IX
DD E3

Exchange the contents of the IX register's low-order byte with the top stack byte. Exchange the IX register's high-order byte with the byte below the stack top.

EX (SP),IY
FD E3

This instruction is identical to EX (SP),IX, but uses the IY register instead of the IX register.

EXX — EXCHANGE REGISTER PAIRS AND ALTERNATE REGISTER PAIRS



EXX
D9

The contents of register pairs BC, DE and HL are swapped with the contents of register pairs B'C', D'E', and H'L'.

Suppose register pairs BC, DE and HL contain 4901_{16}, $5F00_{16}$ and 7251_{16} respectively, and register pairs B'C', D'E', H'L' contain 0000_{16}, $10FF_{16}$ and 3333_{16} respectively. After the execution of

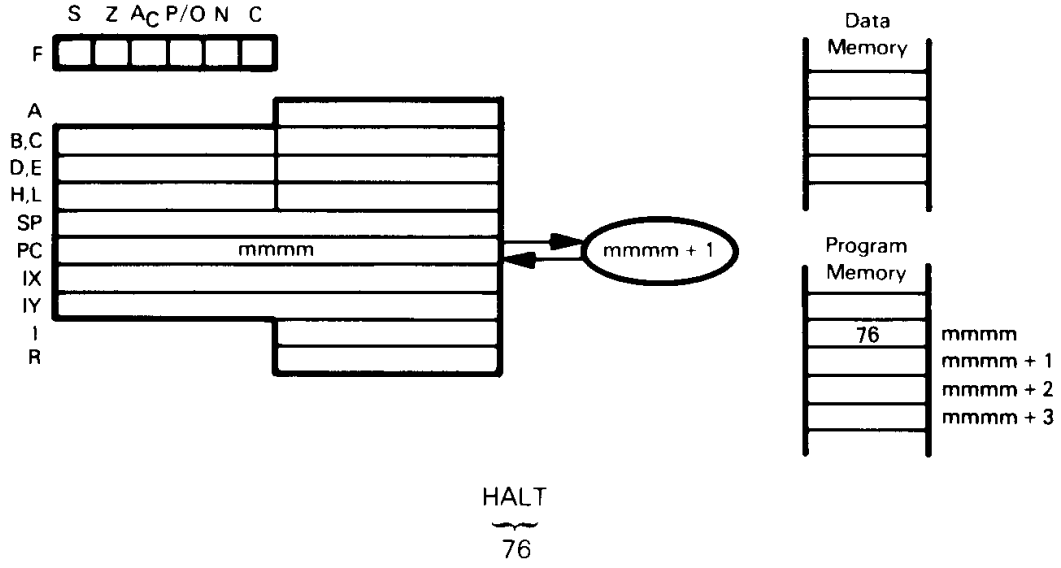
EXX

the registers will have the following contents:

BC: 0000_{16}; DE: $10FF_{16}$; HL: 3333_{16};
B'C': 4901_{16}; D'E': $5F00_{16}$; H'L': 7251_{16}

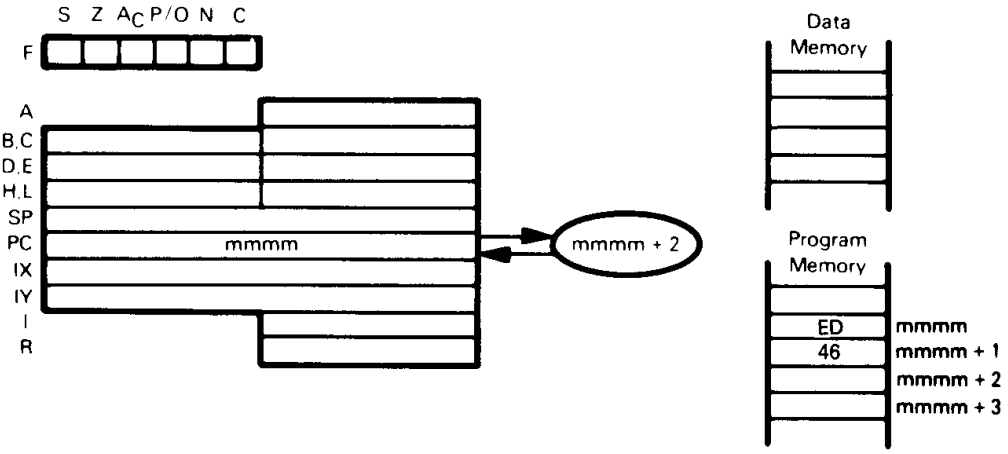
This instruction can be used to exchange register banks to provide very fast interrupt response times.

HALT



When the HALT instruction is executed, program execution ceases. The CPU requires an interrupt or a reset to restart execution. No registers or statuses are affected; however, memory refresh logic continues to operate.

IM 0 — INTERRUPT MODE 0



IM 0
ED 46

This instruction places the CPU in interrupt mode 0. In this mode, the interrupting device will place an instruction on the Data Bus and the CPU will then execute that instruction. No registers or statuses are affected.

IM 1 — INTERRUPT MODE 1

IM 1
ED 56

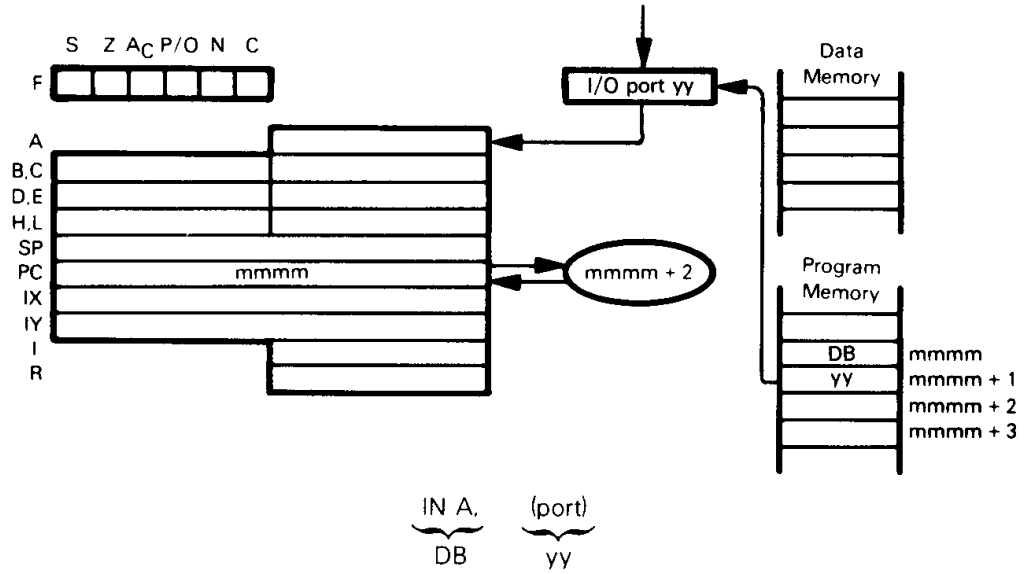
This instruction places the CPU in interrupt mode 1. In this mode, the CPU responds to an interrupt by executing a restart (RST) to location 0038_{16}.

IM 2 — INTERRUPT MODE 2

IM 2
ED 5E

This instruction places the CPU in interrupt mode 2. In this mode, the CPU performs an indirect call to any specified location in memory. A 16-bit address is formed using the contents of the Interrupt Vector (I) register for the upper eight bits, while the lower eight bits are supplied by the interrupting device. Refer to Chapter 12 for a full description of interrupt modes. No registers or statuses are affected by this instruction.

IN A,(port) — INPUT TO ACCUMULATOR



Load a byte of data into the Accumulator from the I/O port (identified by the second IN instruction object code byte).

Suppose 36_{16} is held in the buffer of I/O port $1A_{16}$. After the instruction

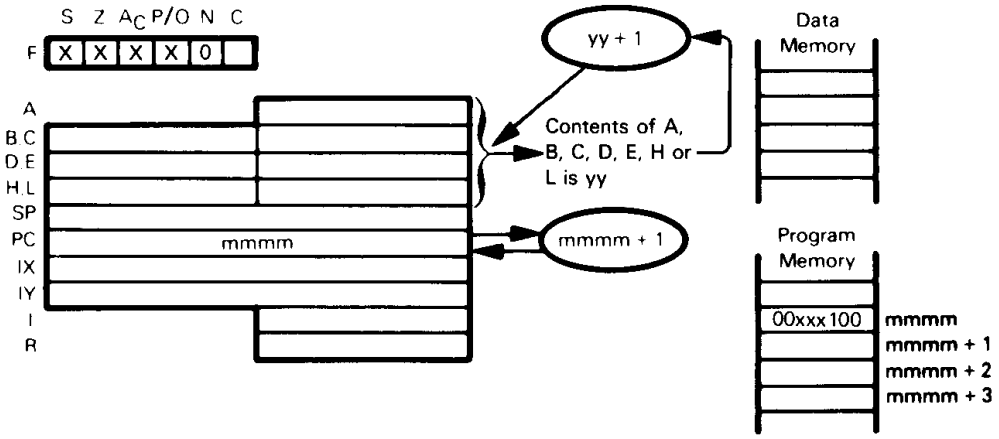
IN A,(1AH)

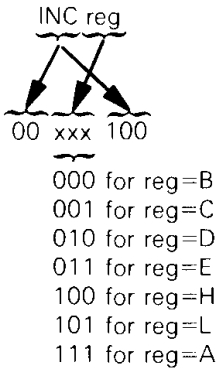
has executed, the Accumulator will contain 36_{16}.

The IN instruction does not affect any statuses.

Use of the IN instruction is very hardware dependent. Valid I/O port addresses are determined by the way in which I/O logic has been implemented. It is also possible to design a microcomputer system that accesses external logic using memory reference instructions with specific memory addresses.

INC reg — INCREMENT REGISTER CONTENTS





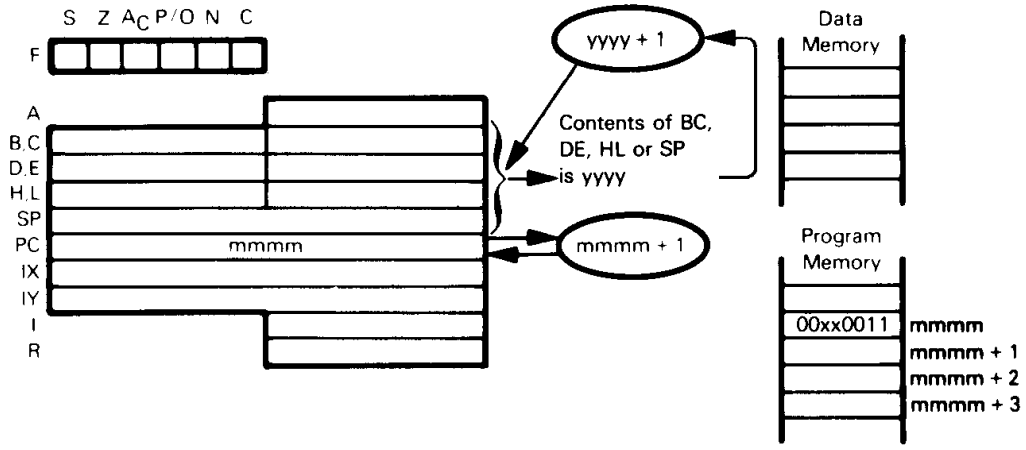
Add 1 to the contents of the specified register.

Suppose Register E contains $A8_{16}$. After execution of

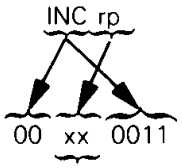
INC E

Register E will contain $A9_{16}$.

INC rp — INCREMENT CONTENTS OF SPECIFIED REGISTER PAIR
INC IX
INC IY



The illustration shows execution of INC rp:



00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Add 1 to the 16-bit value contained in the specified register pair. No status flags are affected.

Suppose the D and E registers contain $2F7A_{16}$. After the instruction

INC DE

has executed, the D and E registers will contain $2F7B_{16}$.

INC IX
DD 23

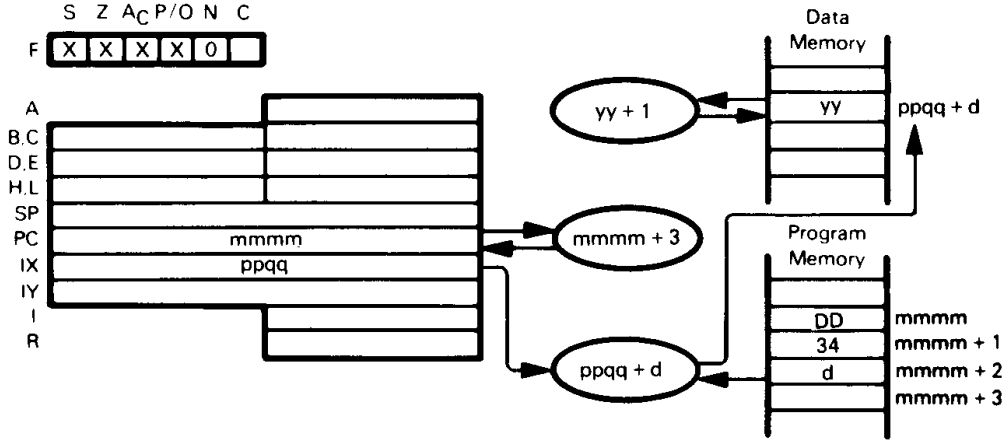
Add 1 to the 16-bit value contained in the IX register.

INC IY
FD 23

Add 1 to the 16-bit value contained in the IY register.

Just like the DEC rp, DEC IX and DEC IY, neither INC rp, INC IX nor INC IY affects any status flags. This is a defect in the Z80 instruction set inherited from the 8080.

INC (HL) — INCREMENT MEMORY CONTENTS
INC (IX+disp)
INC (IY+disp)



The illustration shows execution of INC (IX+d):

INC (IX+disp)

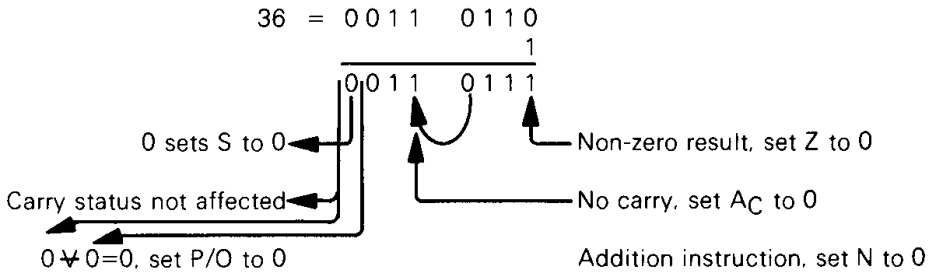
DD 34 d

Add 1 to the contents of memory location (specified by the sum of the contents of Register IX and the displacement value d).

Suppose ppqq=4000_{16} and memory location $400F_{16}$ contains 36_{16}. After execution of the instruction

INC (IX+0FH)

memory location $400F_{16}$ will contain 37_{16}.



INC (IY+disp)

FD 34 d

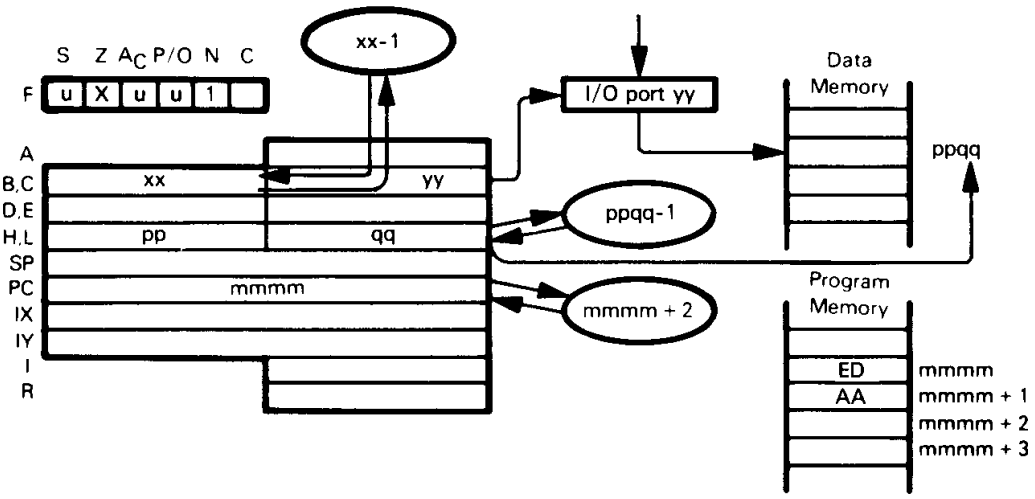
This instruction is identical to INC (IX+disp), except that it uses the IY register instead of the IX register.

INC (HL)

34

Add 1 to the contents of memory location (specified by the contents of the HL register pair).

IND — INPUT TO MEMORY AND DECREMENT POINTER



IND
ED AA

Input from I/O port (addressed by Register C) to memory location (specified by HL). Decrement Registers B and HL.

Suppose xx=05_{16}, yy=15_{16}, ppqq=2400_{16}, and 19_{16} is held in the buffer of I/O port 15_{16}. After the instruction

IND

has executed, memory location 2400_{16} will contain 19_{16}. The B register will contain 04_{16} and the HL register pair $23FF_{16}$.

INDR — INPUT TO MEMORY AND DECREMENT POINTER UNTIL BYTE COUNTER IS ZERO

INDR
ED BA

INDR is identical to IND, but is repeated until Register B=0.

Suppose Register B contains 03_{16}, Register C contains 15_{16}, and HL contains 2400_{16}. The following sequence of bytes is available at I/O port 15_{16}:

17_{16}, 59_{16} and AE_{16}

After the execution of

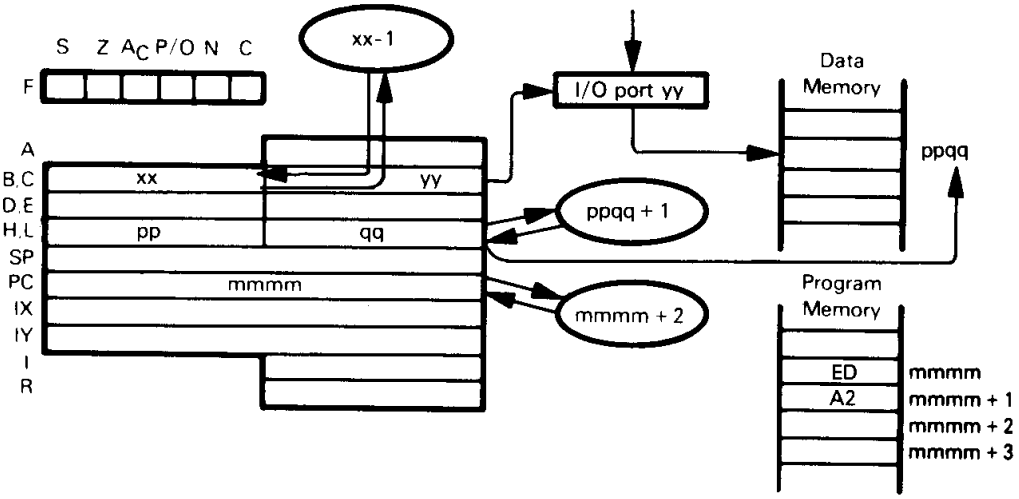
INDR

the HL register pair will contain $23FD_{16}$ and Register B will contain zero, and memory locations will have contents as follows:

Location	Contents
2400	17_{16}
23FF	59_{16}
23FE	AE_{16}

This instruction is extremely useful for loading blocks of data from an input device into memory.

INI — INPUT TO MEMORY AND INCREMENT POINTER



INI
ED A2

Input from I/O port (addressed by Register C) to memory location (specified by HL). Decrement Register B; increment register pair HL.

Suppose xx=05_{16}, yy=15_{16}, ppqq=2400_{16}, and 19_{16} is held in the buffer of I/O port 15_{16}.

After the instruction

INI

has executed, memory location 2400_{16} will contain 19_{16}. The B register will contain 04_{16} and the HL register pair 2401_{16}.

INIR — INPUT TO MEMORY AND INCREMENT POINTER UNTIL BYTE COUNTER IS ZERO

INIR
ED B2

INIR is identical to INI, but is repeated until Register B=0.

Suppose Register B contains 03_{16}, Register C contains 15_{16}, and HL contains 2400_{16}. The following sequence of bytes is available at I/O port 15_{16}:

17_{16}, 59_{16} and AE_{16}

After the execution of

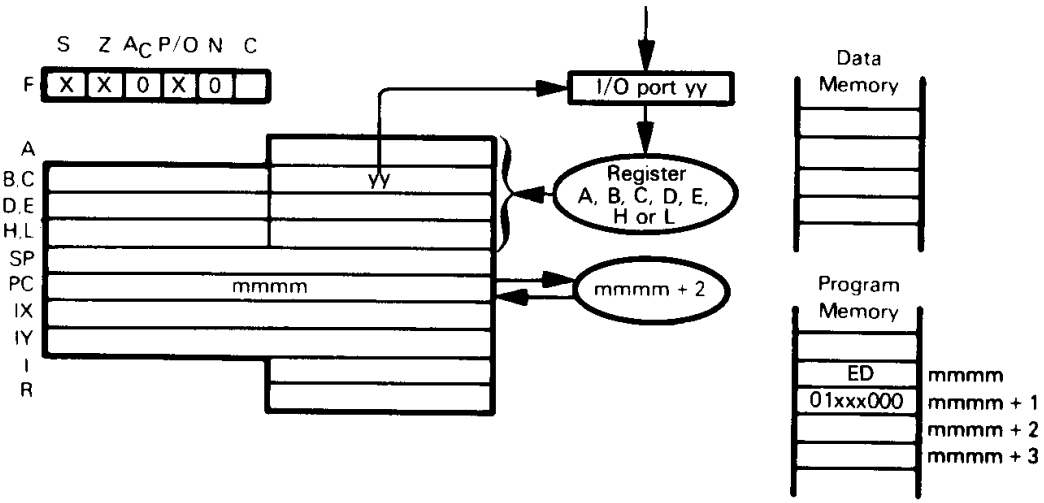
INIR

the HL register pair will contain 2403_{16} and Register B will contain zero, and memory locations will have contents as follows:

Location	Contents
2400	17_{16}
2401	59_{16}
2402	AE_{16}

This instruction is extremely useful for loading blocks of data from a device into memory.

IN reg,(C) — INPUT TO REGISTER



000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A
110 for setting of status flags without changing registers

Load a byte of data into the specified register (reg) from the I/O port (identified by the contents of the C register).

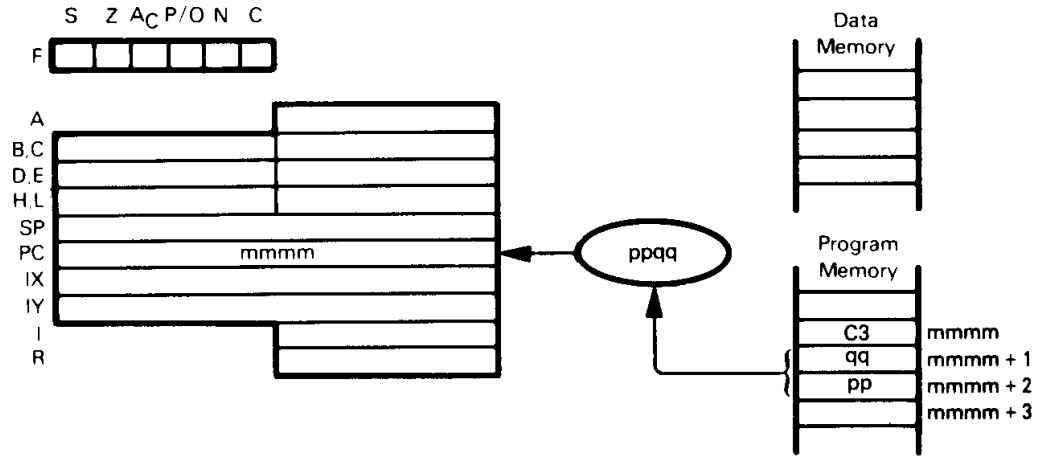
Suppose 42_{16} is held in the buffer of I/O port 36_{16}, and Register C contains 36_{16}. After the instruction

IN D,(C)

has executed, the D register will contain 42_{16}.

During the execution of the instruction, the contents of Register B are placed on the top half of the Address Bus, making it possible to extend the number of addressable I/O ports.

JP label — JUMP TO THE INSTRUCTION IDENTIFIED IN THE OPERAND



JP label

C3 ppqq

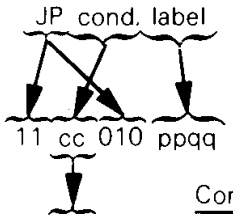
Load the contents of the Jump instruction object code second and third bytes into the Program Counter; this becomes the memory address for the next instruction to be executed. The previous Program Counter contents are lost.

In the following sequence:

```
        JP      NEXT
        AND     7FH
        -
        -
NEXT    CPL
```

The CPL instruction will be executed after the JP instruction. The AND instruction will never be executed, unless a Jump instruction somewhere else in the instruction sequence jumps to this instruction.

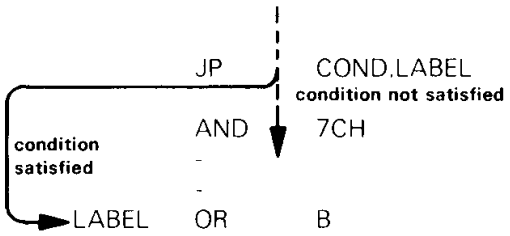
JP condition,label — JUMP TO ADDRESS IDENTIFIED IN THE OPERAND IF CONDITION IS SATISIFED



		Condition	Relevant Flag
000	NZ	Non-Zero	Z
001	Z	Zero	Z
010	NC	No Carry	C
011	C	Carry	C
100	PO	Parity Odd	P/O
101	PE	Parity Even	P/O
110	P	Sign Positive	S
111	M	Sign Negative	S

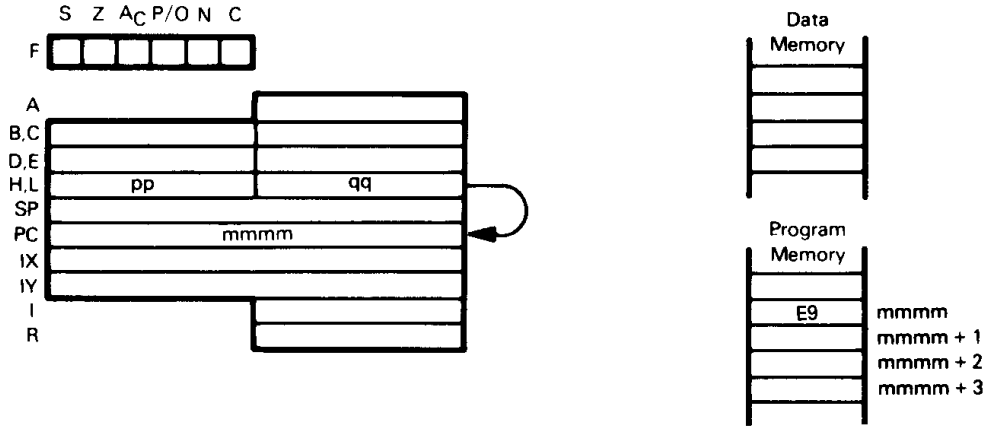
This instruction is identical to the JP instruction, except that the jump will be performed only if the condition is satisfied; otherwise, the instruction sequentially following the JP condition instruction will be executed.

Consider the instruction sequence



After the JP cond,label instruction has executed, if the condition is satisfied then the OR instruction will be executed. If the condition is not satisfied, the AND instruction, being the next sequential instruction, is executed.

JP (HL) — JUMP TO ADDRESS SPECIFIED BY CONTENTS
JP (IX) OF 16-BIT REGISTER
JP (IY)



The illustration shows execution of JP (HL):

JP (HL)
E9

The contents of the HL register pair are moved to the Program Counter; therefore, an implied addressing jump is performed.

The instruction sequence

```
LD      H,ADDR
JP      (HL)
```

has exactly the same net effect as the single instruction

```
JP      ADDR
```

Both specify that the instruction with label ADDR is to be executed next.

The JP (HL) instruction is useful when you want to increment a return address for a subroutine that has multiple returns.

Consider the following call to subroutine SUB:

```
CALL    SUB     ;CALL SUBROUTINE
JP      ERR     ;ERROR RETURN
                ;GOOD RETURN
```

Using RET to return from SUB would return execution of JP ERR; therefore, if SUB executes without detecting error conditions, return as follows:

```
POP     HL      ;POP RETURN ADDRESS TO HL
INC     HL      ;ADD 3 TO RETURN ADDRESS
INC     HL
INC     HL
JP      (HL)    ;RETURN
```

JP (IX)
DD E9

This instruction is identical to the JP (HL) instruction, except that it uses the IX register

instead of the HL register pair.

JP (IY)

FD E9

This instruction is identical to the JP (HL) instruction, except that it uses the IY register instead of the HL register pair.

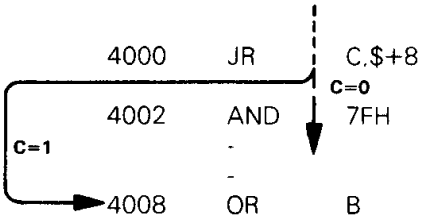
JR C,disp — JUMP RELATIVE TO CONTENTS OF PROGRAM COUNTER IF CARRY IS SET

JR C, disp

38 dd-2

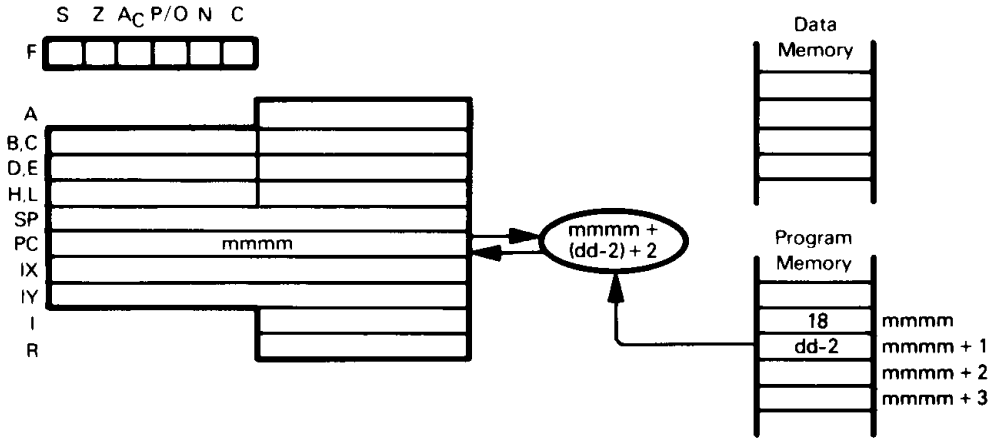
This instruction is identical to the JR disp instruction, except that the jump is only executed if the Carry status equals 1; otherwise, the next instruction is executed.

In the following instruction sequence:



After the JR C,$+8 instruction, the OR instruction is executed if the Carry status equals 1. The AND instruction is executed if the Carry status equals 0.

JR disp — JUMP RELATIVE TO PRESENT CONTENTS OF PROGRAM COUNTER



JR disp

18 dd-2

Add the contents of the JR instruction object code second byte, the contents of the Program Counter, and 2. Load the sum into the Program Counter. The jump is measured from the address of the instruction operation code, and has a range of -126 to +129 bytes. The Assembler automatically adjusts for the twice-incremented PC.

The following assembly language statement is used to jump four steps forward from address 4000_{16}.

JR $+4

Result of this instruction is shown below:

Location	Instruction	
4000	18	
4001	02	
4002	-	
4003	-	
4004	-	← new PC value

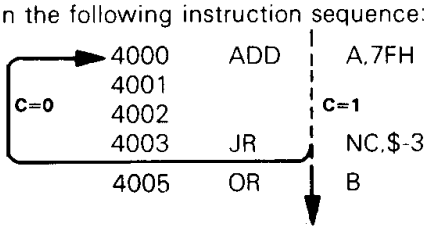
JR NC,disp — JUMP RELATIVE TO CONTENTS OF PROGRAM COUNTER IF CARRY FLAG IS RESET

JR NC,disp

30 dd-2

This instruction is identical to the JR disp instruction, except that the jump is only executed if the Carry status equals 0; otherwise, the next instruction is executed.

In the following instruction sequence:



After the JR NC,$-3 instruction, the OR instruction is executed if the Carry status equals 1. The ADD instruction is executed if the Carry status equals 0.

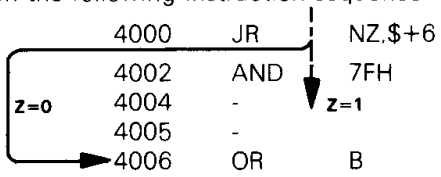
JR NZ,disp — JUMP RELATIVE TO CONTENTS OF PROGRAM COUNTER IF ZERO FLAG IS RESET

JR NZ,disp

20 dd-2

This instruction is identical to the JR disp instruction, except that the jump is only executed if the Zero status equals 0; otherwise, the next instruction is executed.

In the following instruction sequence:



After the JR NZ,$+6 instruction, the OR instruction is executed if the Zero status equals 0. The AND instruction is executed if the Zero status equals 1.

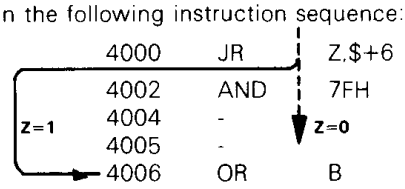
JR Z,disp — JUMP RELATIVE TO CONTENTS OF PROGRAM COUNTER IF ZERO FLAG IS SET

JR Z,disp

28 dd-2

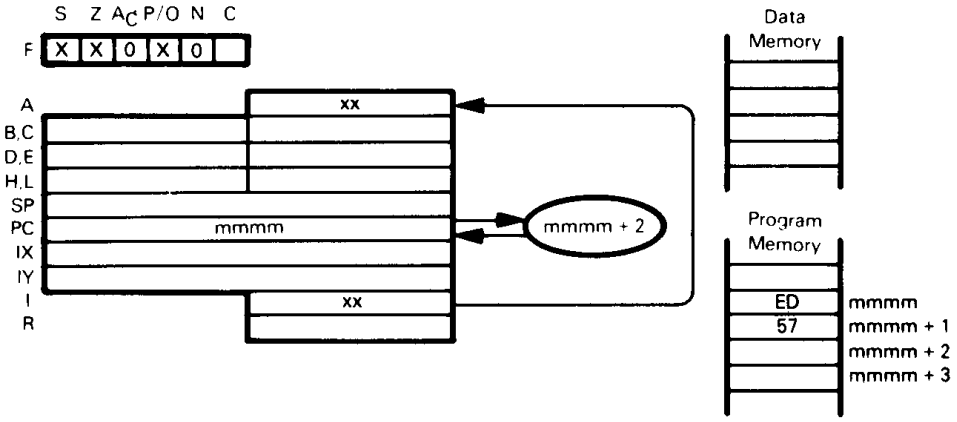
This instruction is identical to the JR disp instruction, except that the jump is only executed if the Zero status equals 1; otherwise, the next instruction is executed.

In the following instruction sequence:



After the JR Z,$+6 instruction, the OR instruction is executed if the Zero status equals 1. The AND instruction is executed if the Zero status equals 0.

LD A,I — MOVE CONTENTS OF INTERRUPT VECTOR OR LD A,R REFRESH REGISTER TO ACCUMULATOR



The illustration shows execution of LD A,I:

LD A,I

ED 57

Move the contents of the Interrupt Vector register to the Accumulator, and reflect interrupt enable status in Parity/Overflow flag.

Suppose the Interrupt Vector register contains $7F_{16}$, and interrupts are disabled. After execution of

LD A,I

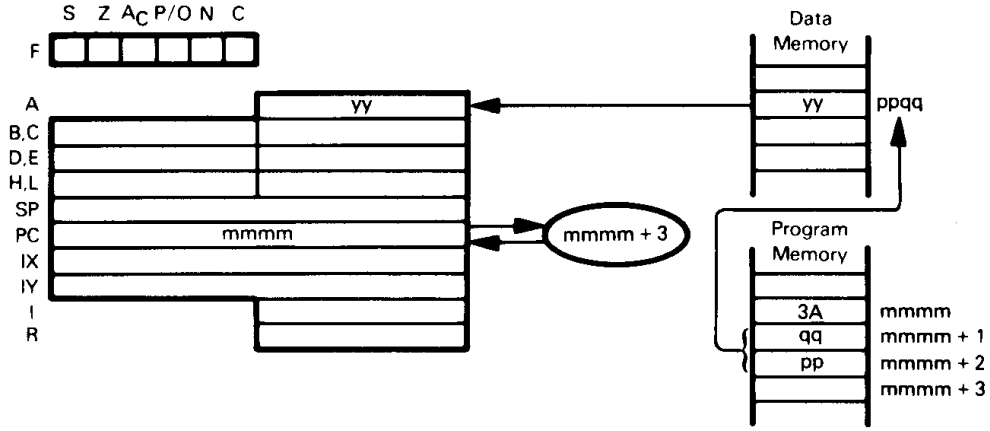
Register A will contain $7F_{16}$, and P/O will be 0.

LD A,R

ED 5F

Move the contents of the Refresh register to the Accumulator. The value of the interrupt flip-flop will appear in the Parity/Overflow flag.

LD A,(addr) — LOAD ACCUMULATOR FROM MEMORY USING DIRECT ADDRESSING



LD A, (addr)

3A ppqq

Load the contents of the memory byte (addressed directly by the second and third bytes of the LD A,(addr) instruction object code) into the Accumulator. Suppose memory byte $084A_{16}$ contains 20_{16}. After the instruction

```
label   EQU     084AH
-
-
-
LD      A,(label)
```

has executed, the Accumulator will contain 20_{16}.

Remember that EQU is an assembler directive rather than an instruction; it tells the Assembler to use the 16-bit value $084A_{16}$ wherever the label appears.

The instruction

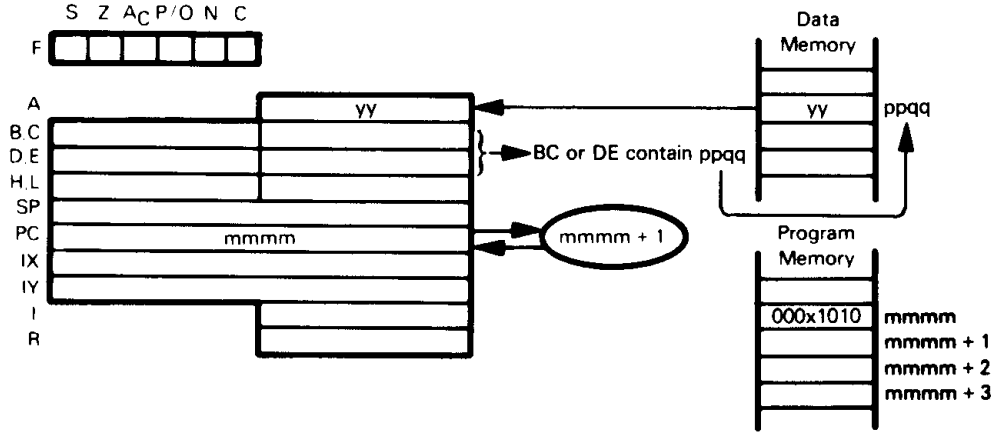
```
LD      A,(label)
```

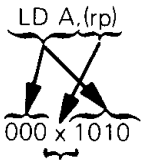
is equivalent to the two instructions

```
LD      HL,label
LD      A,(HL)
```

When you are loading a single value from memory, the LD A,(label) instruction is preferred; it uses one instruction and three object program bytes to do what the LD HL,label, LD A,(HL) combination does in two instructions and four object program bytes. Also, the LD HL,label, LD A,(HL) combination uses the H and L registers, which LD A,(label) does not.

LD A,(rp) — LOAD ACCUMULATOR FROM MEMORY LOCATION ADDRESSED BY REGISTER PAIR





Load the contents of the memory byte (addressed by the BC or DE register pair) into the Accumulator.

Suppose the B register contains 08_{16}, the C register contains $4A_{16}$, and memory byte $084A_{16}$ contains $3A_{16}$. After the instruction

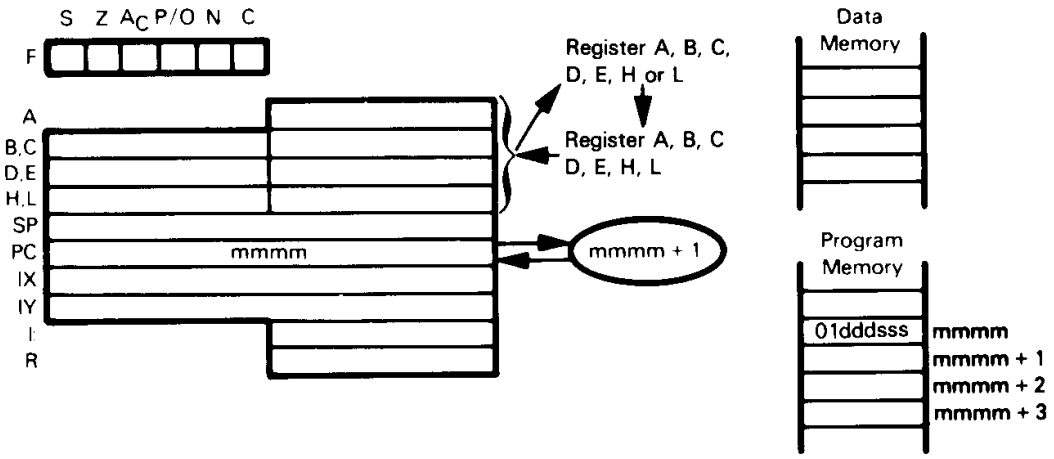
LD A,(BC)

has executed, the Accumulator will contain $3A_{16}$.

Normally, the LD A,(rp) and LD rp,data will be used together, since the LD rp,data instruction loads a 16-bit address into the BC or DE registers as follows:

```
LD      BC,084AH
LD      A,(BC)
```

LD dst,src — MOVE CONTENTS OF SOURCE REGISTER TO DESTINATION REGISTER



LD dst, src

01 ddd sss

000 for dst or src=B
001 for dst or src=C
010 for dst or src=D
011 for dst or src=E
100 for dst or src=H
101 for dst or src=L
111 for dst or src=A

The contents of any designated register are loaded into any other register.

For example:

LD A,B

loads the contents of Register B into Register A.

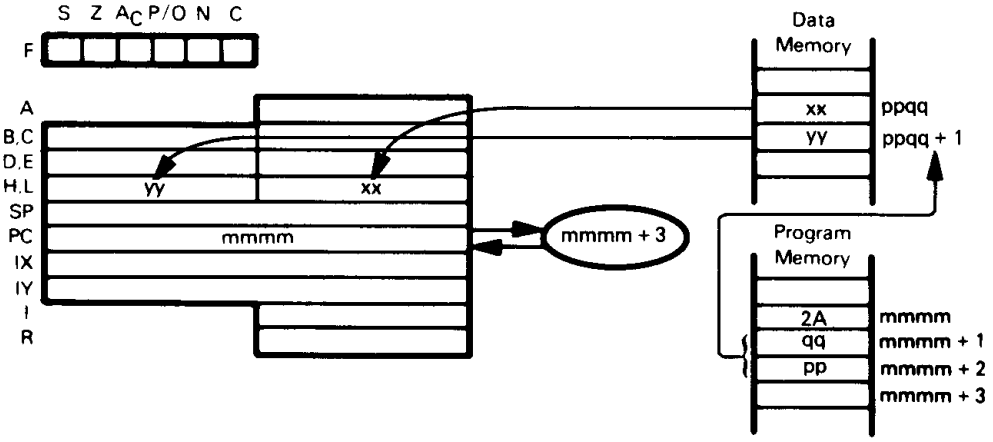
LD L,D

loads the contents of Register D into Register L.

LD C,C

does nothing, since the C register has been specified as both the source and the destination.

LD HL,(addr) — LOAD REGISTER PAIR OR INDEX REGISTER
LD rp,(addr) FROM MEMORY USING DIRECT ADDRESSING
LD IX,(addr)
LD IY,(addr)



The illustration shows execution of LD HL(ppqq):

LD HL,addr
2A ppqq

Load the HL register pair from directly addressed memory location.

Suppose memory location 4004_{16} contains AD_{16} and memory location 4005_{16} contains 12_{16}. After the instruction

LD HL,(4004H)

has executed, the HL register pair will contain $12AD_{16}$.

LD rp, (addr)
ED 01 dd 1011 ppqq

00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Load register pair from directly addressed memory.

Suppose memory location $49FF_{16}$ contains BE_{16} and memory location $4A00_{16}$ contains 33_{16}. After the instruction

LD DE,(49FFH)

has executed, the DE register pair will contain $33BE_{16}$.

LD IX,(addr)
DD 2A ppqq

Load IX register from directly addressed memory.

Suppose memory location $D111_{16}$ contains FF_{16} and memory location $D112_{16}$ contains 56_{16}. After the instruction

LD IX,(D111H)

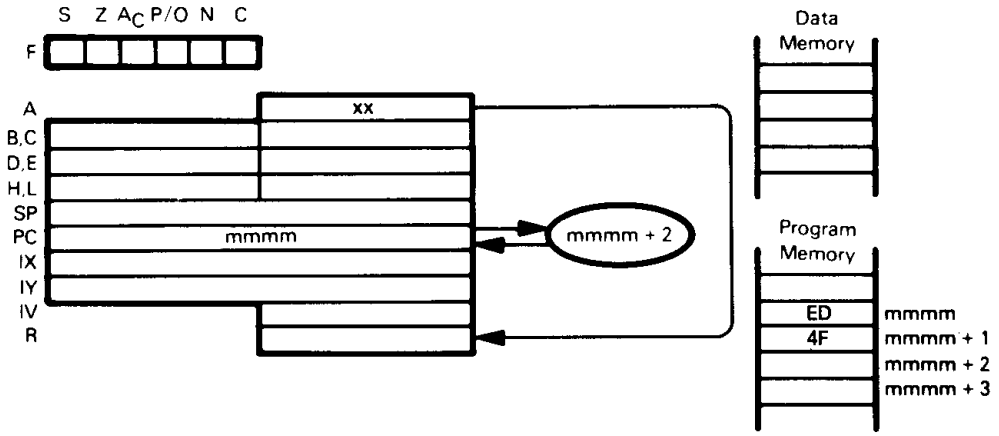
has executed, the IX register will contain $56FF_{16}$.

LD IY,(addr)
FD 2A ppqq

Load IY register from directly addressed memory.

Affects IY register instead of IX. Otherwise identical to LD IX(addr).

LD I,A — LOAD INTERRUPT VECTOR OR REFRESH
LD R,A REGISTER FROM ACCUMULATOR



The illustration shows execution of LD R,A:

LD R,A
ED 4F

Load Refresh register from Accumulator.

Suppose the Accumulator contains $7F_{16}$. After the instruction

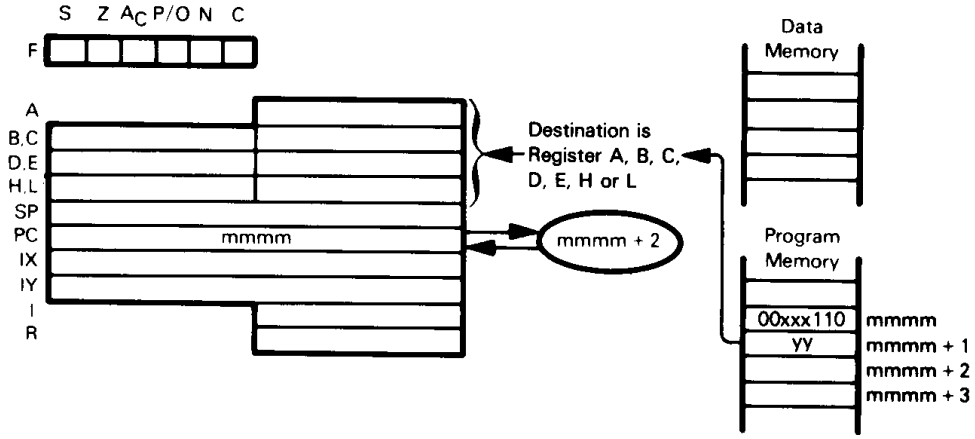
LD R,A

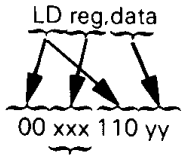
has executed, the Refresh register will contain $7F_{16}$.

LD I,A
ED 47

Load Interrupt Vector register from Accumulator.

LD reg,data — LOAD IMMEDIATE INTO REGISTER





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

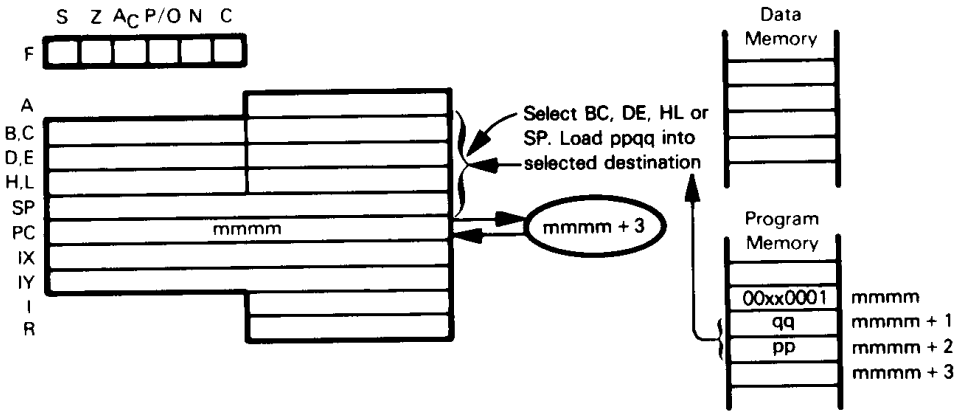
Load the contents of the second object code byte into one of the registers.

When the instruction

LD A,2AH

has executed, $2A_{16}$ is loaded into the Accumulator.

LD rp,data — LOAD 16 BITS OF DATA IMMEDIATE INTO
LD IX,data REGISTER
LD IY,data



The illustration shows execution of LD rp,data:

LD rp, data

00 xx 0001 ppqq

00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Load the contents of the second and third object code bytes into the selected register pair. After the instruction

LD SP,217AH

has executed, the Stack Pointer will contain $217A_{16}$.

LD IX, data

DD 21 ppqq

Load the contents of the second and third object code bytes into the Index register IX.

LD IY, data

FD 21 ppqq

Load the contents of the second and third object code bytes into the Index Register IY.

Notice that the LD rp,data instruction is equivalent to two LD reg,data instructions.

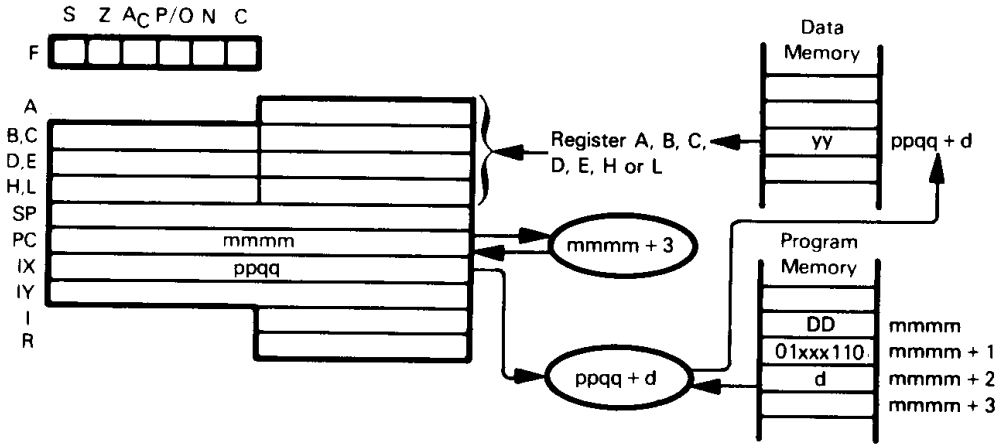
For example:

```
LD      HL,032AH
```

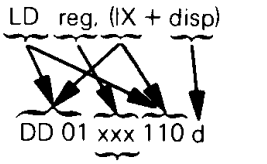
is equivalent to

```
LD      H,03H
LD      L,2AH
```

LD reg,(HL) — LOAD REGISTER FROM MEMORY
LD reg,(IX+disp)
LD reg,(IY+disp)



The illustration shows execution of LD reg,(IX+disp):



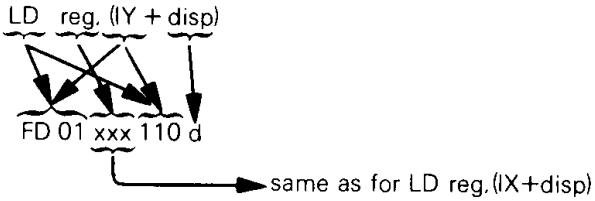
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Load specified register from memory location (specified by the sum of the contents of the IX register and the displacement digit d).

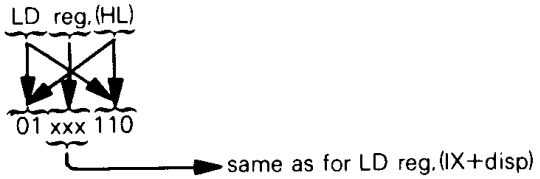
Suppose ppqq=4004_{16} and memory location 4010_{16} contains FF_{16}. After the instruction

LD B(IX+0CH)

has executed, Register B will contain FF_{16}.

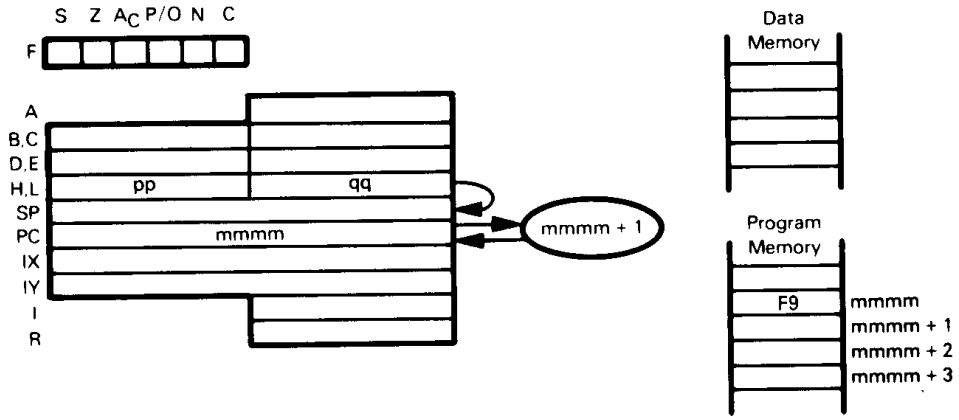


This instruction is identical to LD reg,(IX+disp), except that it uses the IY register instead of the IX register.



Load specified register from memory location (specified by the contents of the HL register pair).

LD SP,HL — MOVE CONTENTS OF HL OR INDEX REGISTER TO STACK POINTER
LD SP,IX
LD SP,IY



The illustration shows execution of LD SP,HL:

LD SP,HL
F9

Load contents of HL into Stack Pointer.

Suppose pp=08_{16} and qq=$3F_{16}$. After the instruction

LD SP,HL

has executed, the Stack Pointer will contain $083F_{16}$.

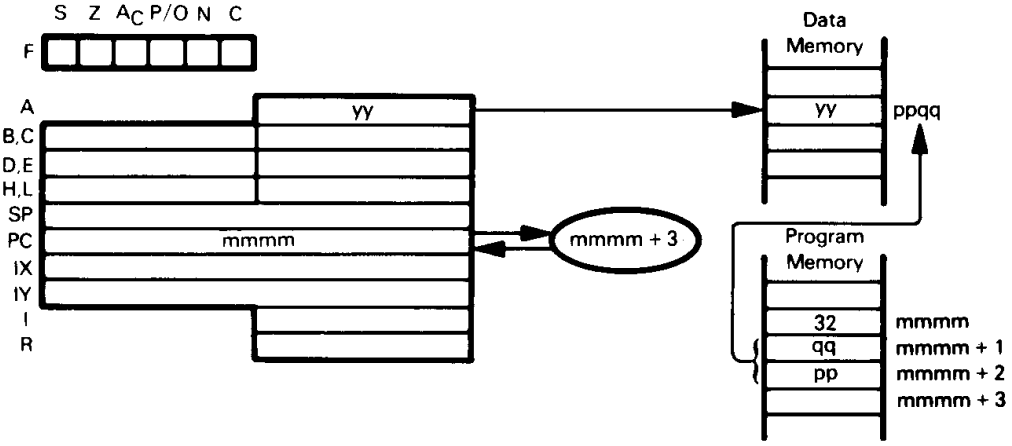
LD SP,IX
DD F9

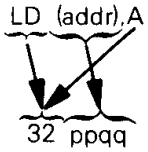
Load contents of Index Register IX into Stack Pointer.

LD SP,IY
FD F9

Load contents of Index Register IY into Stack Pointer.

LD (addr),A — STORE ACCUMULATOR IN MEMORY USING DIRECT ADDRESSING





Store the Accumulator contents in the memory byte addressed directly by the second and third bytes of the LD (addr),A instruction object code.

Suppose the Accumulator contains $3A_{16}$. After the instruction

```
label   EQU     084AH
        -
        -
        LD      (label),A
```

has executed, memory byte $084A_{16}$ will contain $3A_{16}$.

Remember that EQU is an assembler directive rather than an instruction; it tells the Assembler to use the 16-bit value 084AH whenever the word "label" appears.

The instruction

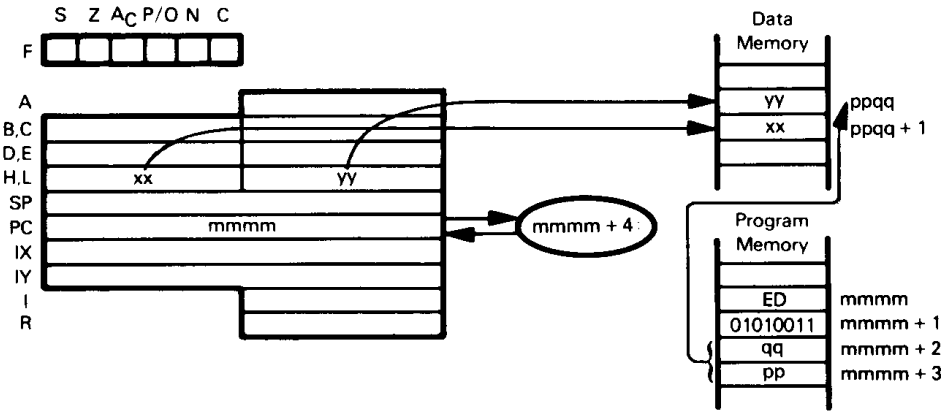
```
LD (addr),A
```

is equivalent to the two instructions

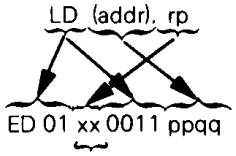
```
LD H,label
LD (HL),A
```

When you are storing a single data value in memory, the LD (label),A instruction is preferred because it uses one instruction and three object program bytes to do what the LD H(label), LD (HL),A combination does in two instructions and four object program bytes. Also, the LD H(label), LD (HL),A combination uses the H and L registers, while the LD (label),A instruction does not.

LD (addr),HL — STORE REGISTER PAIR OR INDEX REGISTER IN MEMORY USING DIRECT ADDRESSING
LD (addr),rp
LD (addr),xy



The illustration shows execution of LD (ppqq),DE:



00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

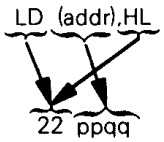
Store the contents of the specified register pair in memory. The third and fourth object code bytes give the address of the memory location where the low-order byte is to be written. The high-order byte is written into the next sequential memory location.

Suppose the BC register pair contains $3C2A_{16}$. After the instruction

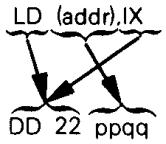
```
label   EQU   084AH
-
-
-
LD    (label),BC
```

has executed, memory byte $084A_{16}$ will contain $2A_{16}$. Memory byte $084B_{16}$ will contain $3C_{16}$.

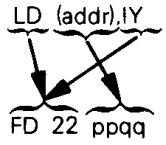
Remember that EQU is an assembler directive rather than an instruction; it tells the Assembler to use the 16-bit value $084A_{16}$ whenever the word "label" appears.



This is a three-byte version of LD (addr),rp which directly specifies HL as the source register pair.

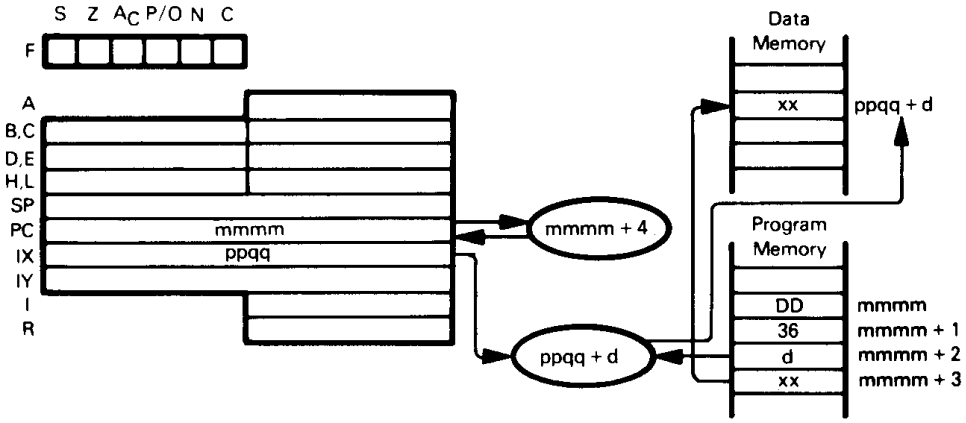


Store the contents of Index register IX in memory. The third and fourth object code bytes give the address of the memory location where the low-order byte is to be written. The high-order byte is written into the next sequential memory location.



This instruction is identical to the LD (addr),IX instruction, except that it uses the IY register instead of the IX register.

LD (HL),data — LOAD IMMEDIATE INTO MEMORY
LD (IX+disp),data
LD (IY+disp),data



The illustration shows execution of LD (IX+d),xx:

LD (IX+disp),data

DD 36 d xx

Load Immediate into the Memory location designated by base relative addressing.

Suppose ppqq=5400_{16}. After the instruction

LD (IX+9),FAH

has executed, memory location 5409_{16} will contain FA_{16}.

LD (IY+disp),data

FD 36 d xx

This instruction is identical to LD (IX+disp),data, but uses the IY register instead of the IX register.

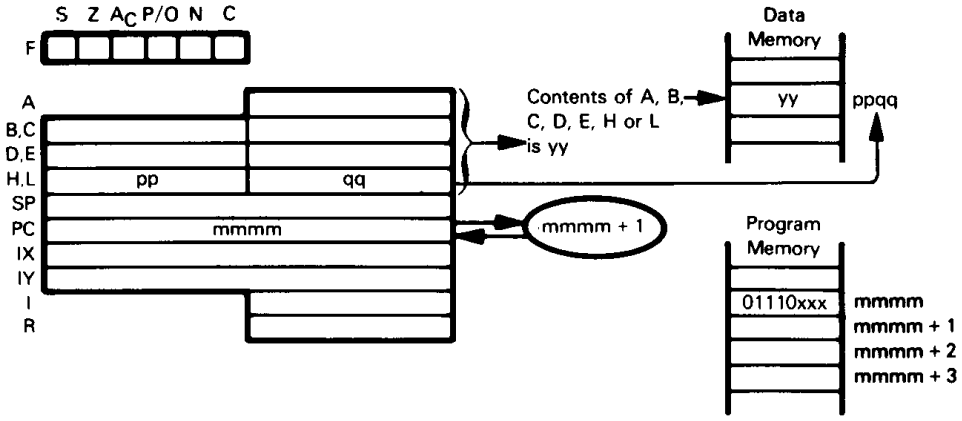
LD (HL),data

36 xx

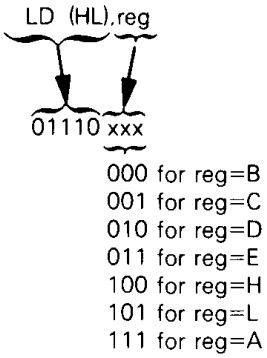
Load Immediate into the Memory location (specified by the contents of the HL register pair).

The Load Immediate into Memory instructions are used much less than the Load Immediate into Register instructions.

LD (HL),reg — LOAD MEMORY FROM REGISTER
LD (IX+disp),reg
LD (IY+disp),reg



The illustration shows execution of LD (HL),reg:

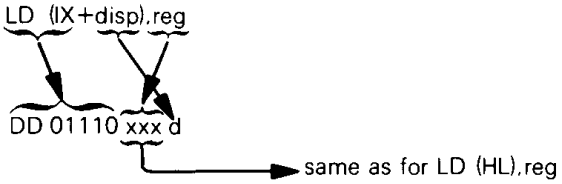


Load memory location (specified by the contents of the HL register pair) from specified register.

Suppose ppqq=4500_{16} and Register C contains $F9_{16}$. After the instruction

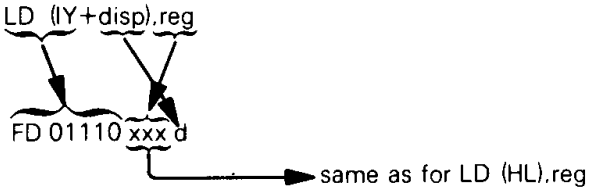
LD (HL),C

has executed, memory location 4500_{16} will contain $F9_{16}$.



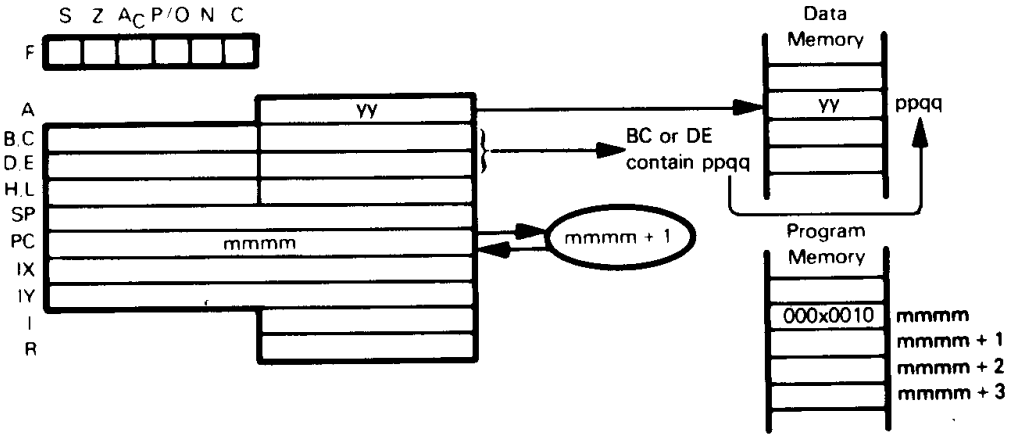
Load memory location (specified by the sum of the contents of the IX register and the

displacement value d) from specified register.



This instruction is identical to LD (IX+disp),reg, except that it uses the IY register instead of the IX register.

LD (rp),A — LOAD ACCUMULATOR INTO THE MEMORY LOCATION ADDRESSED BY REGISTER PAIR



0 if register pair=BC
1 if register pair=DE

Store the Accumulator in the memory byte addressed by the BC or DE register pair.

Suppose the BC register pair contains $084A_{16}$ and the Accumulator contains $3A_{16}$. After the instruction

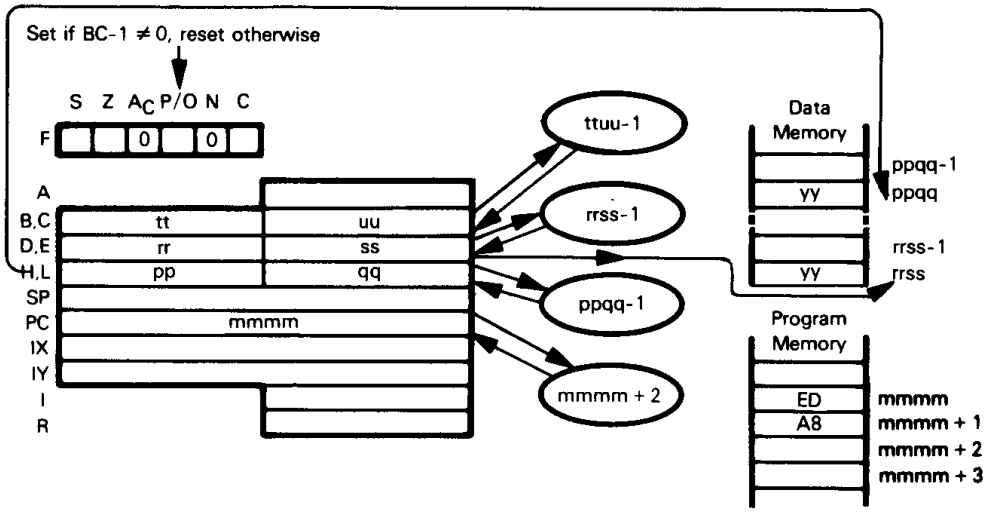
LD (BC),A

has executed, memory byte $084A_{16}$ will contain $3A_{16}$.

The LD (rp),A and LD rp,data will normally be used together, since the LD rp,data instruction loads a 16-bit address into the BC or DE registers as follows:

```
LD BC,084AH
LD (BC),A
```

LDD — TRANSFER DATA BETWEEN MEMORY LOCATIONS, DECREMENT DESTINATION AND SOURCE ADDRESSES



LDD

ED A8

Transfer a byte of data from memory location addressed by the HL register pair to memory location addressed by the DE register pair. Decrement contents of register pairs BC, DE, and HL.

Suppose register pair BC contains $004F_{16}$, DE contains 4545_{16}, HL contains 2012_{16}, and memory location 2012_{16} contains 18_{16}. After the instruction

LDD

has executed, memory location 4545_{16} will contain 18_{16}, register pair BC will contain $004E_{16}$, DE will contain 4544_{16}, and HL will contain 2011_{16}.

LDDR — TRANSFER DATA BETWEEN MEMORY LOCATIONS UNTIL BYTE COUNTER IS ZERO. DECREMENT DESTINATION AND SOURCE ADDRESSES

LDDR
ED B8

This instruction is identical to LDD, except that it is repeated until the BC register pair contains zero. After each data transfer, interrupts will be recognized and two refresh cycles will be executed.

Suppose we have the following contents in memory and register pairs:

Register/Contents		Location/Contents	
HL	2012_{16}	2012_{16}	18_{16}
DE	4545_{16}	2011_{16}	AA_{16}
BC	0003_{16}	2010_{16}	25_{16}

After execution of

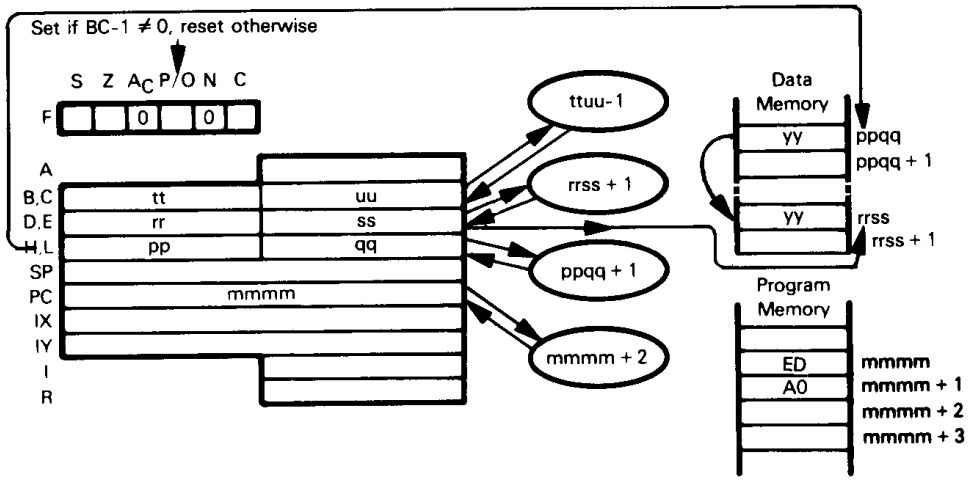
LDDR

register pairs and memory locations will have the following contents:

Register/Contents		Location/Contents		Location/Contents	
HL	2009_{16}	2012_{16}	18_{16}	4545_{16}	18_{16}
DE	4542_{16}	2011_{16}	AA_{16}	4544_{16}	AA_{16}
BC	0000_{16}	2010_{16}	25_{16}	4543_{16}	25_{16}

This instruction is extremely useful for transferring blocks of data from one area of memory to another.

LDI — TRANSFER DATA BETWEEN MEMORY LOCATIONS, INCREMENT DESTINATION AND SOURCE ADDRESSES



LDI
ED A0

Transfer a byte of data from memory location addressed by the HL register pair to memory location addressed by the DE register pair. Increment contents of register pairs HL and DE. Decrement contents of the BC register pair.

Suppose register pair BC contains $004F_{16}$, DE contains 4545_{16}, HL contains 2012_{16}, and memory location 2012_{16} contains 18_{16}. After the instruction

LDI

has executed, memory location 4545_{16} will contain 18_{16}, register pair BC will contain $004E_{16}$, DE will contain 4546_{16}, and HL will contain 2013_{16}.

LDIR — TRANSFER DATA BETWEEN MEMORY LOCATIONS UNTIL BYTE COUNTER IS ZERO. INCREMENT DESTINATION AND SOURCE ADDRESSES

LDIR

ED B0

This instruction is identical to LDI, except that it is repeated until the BC register pair contains zero. After each data transfer, interrupts will be recognized and two refresh cycles will be executed.

Suppose we have the following contents in memory and register pairs:

Register/Contents		Location/Contents	
HL	2012_{16}	2012_{16}	18_{16}
DE	4545_{16}	2013_{16}	CD_{16}
BC	0003_{16}	2014_{16}	$F0_{16}$

After execution of

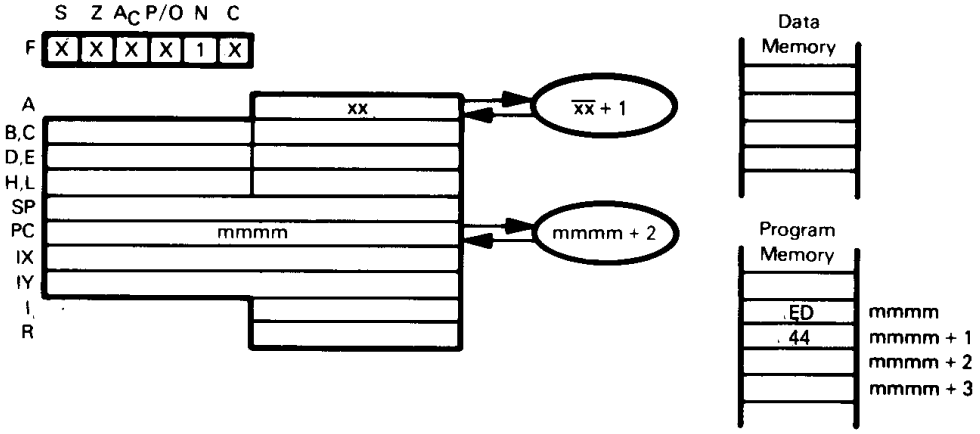
LDIR

register pairs and memory will have the following contents:

Register/Contents		Location/Contents		Location/Contents	
HL	2015_{16}	2012_{16}	18_{16}	4545_{16}	18_{16}
DE	4548_{16}	2013_{16}	CD_{16}	4546_{16}	CD_{16}
BC	0000_{16}	2014_{16}	$F0_{16}$	4547_{16}	$F0_{16}$

This instruction is extremely useful for transferring blocks of data from one area of memory to another.

NEG — NEGATE CONTENTS OF ACCUMULATOR



Negate contents of Accumulator. This is the same as subtracting contents of the Accumulator from zero. The result is the two's complement. 80H will be left unchanged.

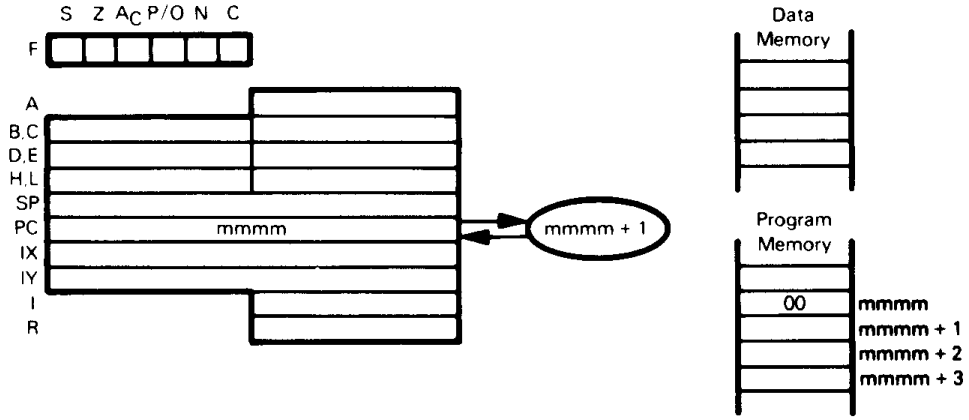
Suppose xx=$5A_{16}$. After the instruction

NEG

has executed, the Accumulator will contain $A6_{16}$.

5A = 0 1 0 1 1 0 1 0
Two's complement = 1 0 1 0 0 1 1 0

NOP — NO OPERATION



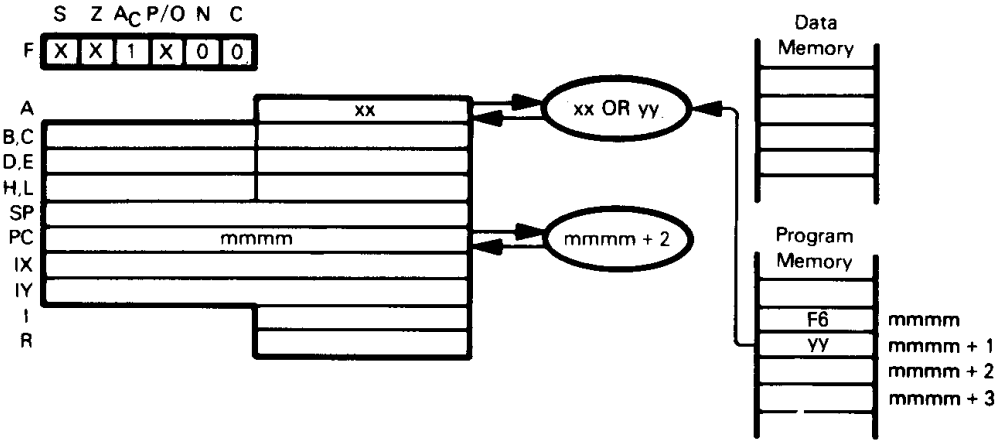
NOP
00

This is a one-byte instruction which performs no operation, except that the Program Counter is incremented and memory refresh continues. This instruction is present for several reasons:

1) A program error that fetches an object code from non-existent memory will fetch 00. It is a good idea to ensure that the most common program error will do nothing.
2) The NOP instruction allows you to give a label to an object program byte:
   HERE NOP
3) To fine-tune delay times. Each NOP instruction adds four clock cycles to a delay.

NOP is not a very useful or frequently used instruction.

OR data — OR IMMEDIATE WITH ACCUMULATOR



OR data

F6 yy

OR the Accumulator with the contents of the second instruction object code byte.

Suppose xx=$3A_{16}$. After the instruction

OR 7CH

has executed, the Accumulator will contain $7E_{16}$.

```
3A = 0011 1010
7C = 0111 1100
     0111 1110
```

0 sets S to 0

Six 1 bits, set P/O to 1

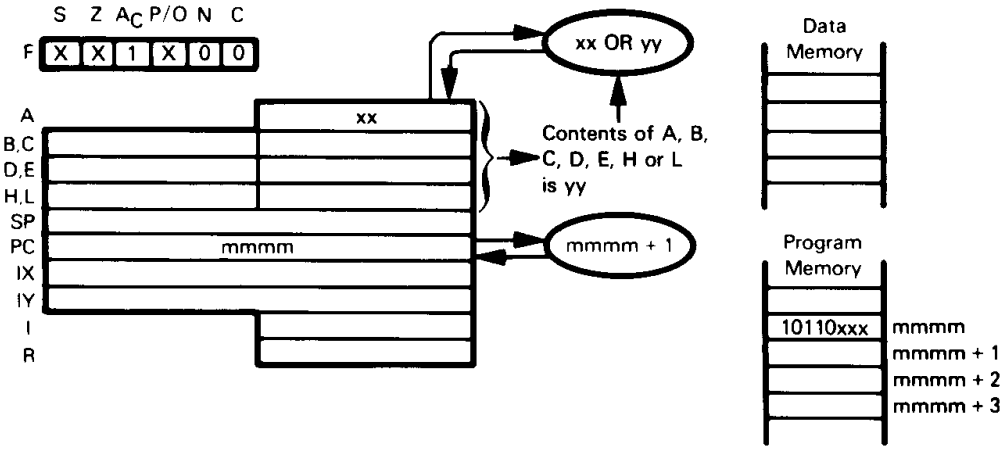
Non-zero result, set Z to 0

This is a routine logical instruction; it is often used to turn bits "on". For example, the instruction

OR 80H

will unconditionally set the high-order Accumulator bit to 1.

OR reg — OR REGISTER WITH ACCUMULATOR



OR reg

10110 xxx

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

Logically OR the contents of the Accumulator with the contents of Register A, B, C, D, E, H or L. Store the result in the Accumulator.

Suppose xx=$E3_{16}$ and Register E contains $A8_{16}$. After the instruction

OR E

has executed, the Accumulator will contain EB_{16}.

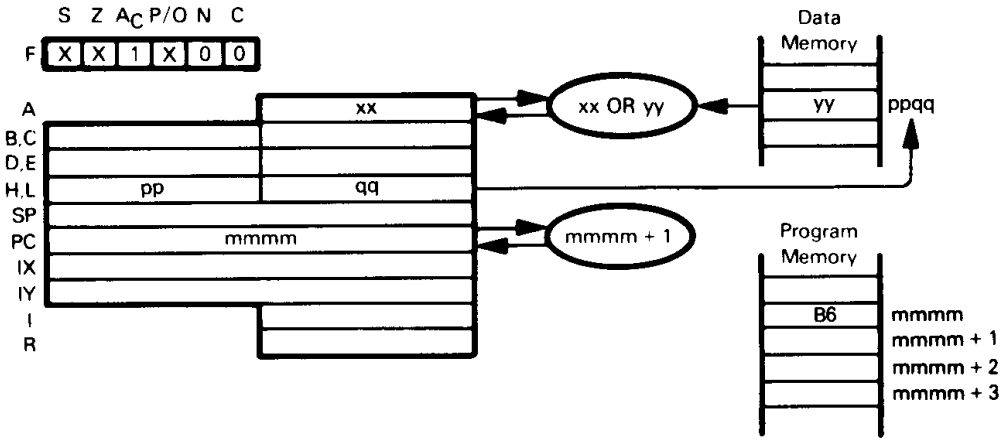
```
E3 = 1110 0011
A8 = 1010 1000
     1110 1011
```

1 sets S to 1

Six 1 bits, set P/O to 1

Non-zero result, set Z to 0

OR (HL) — OR MEMORY WITH ACCUMULATOR
OR (IX+disp)
OR (IY+disp)



The illustration shows execution of OR (HL):

OR (HL)
B6

OR contents of memory location (specified by the contents of the HL register pair) with the Accumulator.

Suppose xx=$E3_{16}$, ppqq=4000_{16}, and memory location 4000_{16} contains $A8_{16}$. After the instruction

OR (HL)

has executed, the Accumulator will contain EB_{16}.

```
E3 = 1110  0011
A8 = 1010  1000
     ----  ----
     1110  1011
```

1 sets S to 1

Six 1 bits, set P/O to 1

Non-zero result, set Z to 0

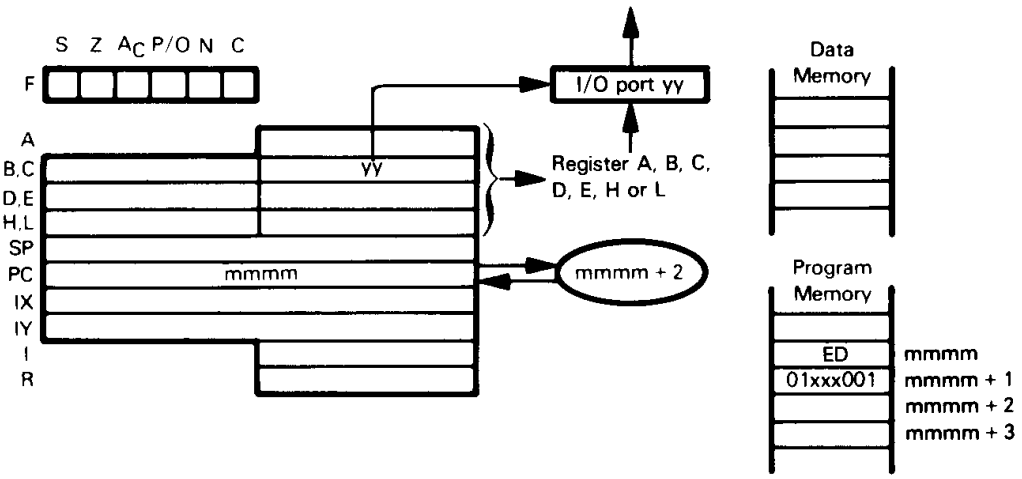
OR (IX+disp)
DD B6 d

OR contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) with the Accumulator.

OR (IY+disp)
FD B6 d

This instruction is identical to OR (IX+disp), except that it uses the IY register instead of the IX register.

OUT (C),reg — OUTPUT FROM REGISTER



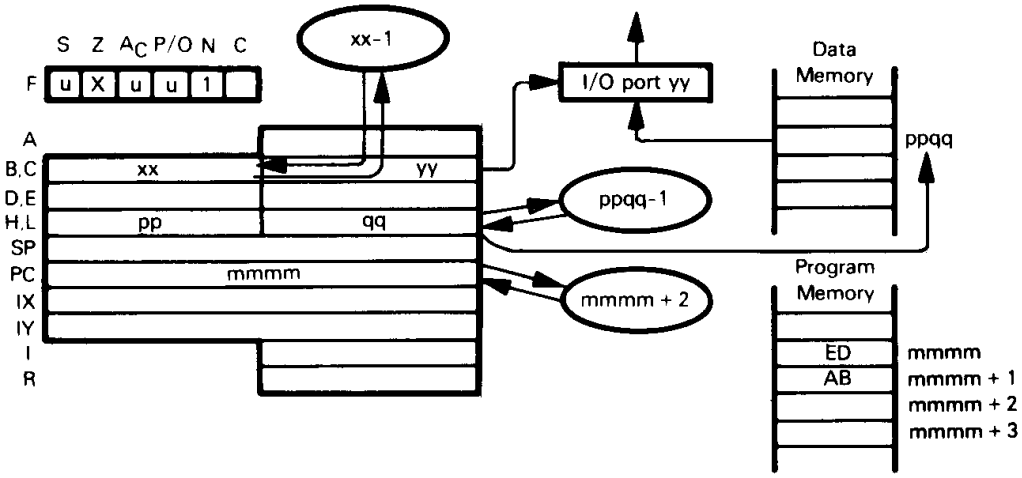
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Suppose yy=$1F_{16}$ and the contents of H are AA_{16}. After the execution of

OUT (C),H

AA_{16} will be in the buffer of I/O port $1F_{16}$.

OUTD — OUTPUT FROM MEMORY. DECREMENT ADDRESS



OUTD
ED AB

Output from memory location specified by HL to I/O port addressed by Register C. Registers B and HL are decremented.

Suppose xx=$0A_{16}$, yy=FF_{16}, ppqq=5000_{16}, and memory location 5000_{16} contains 77_{16}. After the instruction

OUTD

has executed, 77_{16} will be held in the buffer of I/O port FF_{16}. The B register will contain 09_{16}, and the HL register pair $4FFF_{16}$.

OTDR — OUTPUT FROM MEMORY. DECREMENT ADDRESS, CONTINUE UNTIL REGISTER B=0

OTDR
ED BB

OTDR is identical to OUTD, but is repeated until Register B contains 0.

Suppose Register B contains 03_{16}, Register C contains FF_{16}, and HL contains 5000_{16}. Memory locations $4FFE_{16}$ through 5000_{16} contain:

Location	Contents
$4FFE_{16}$	CA_{16}
$4FFF_{16}$	$1B_{16}$
5000_{16}	$F1_{16}$

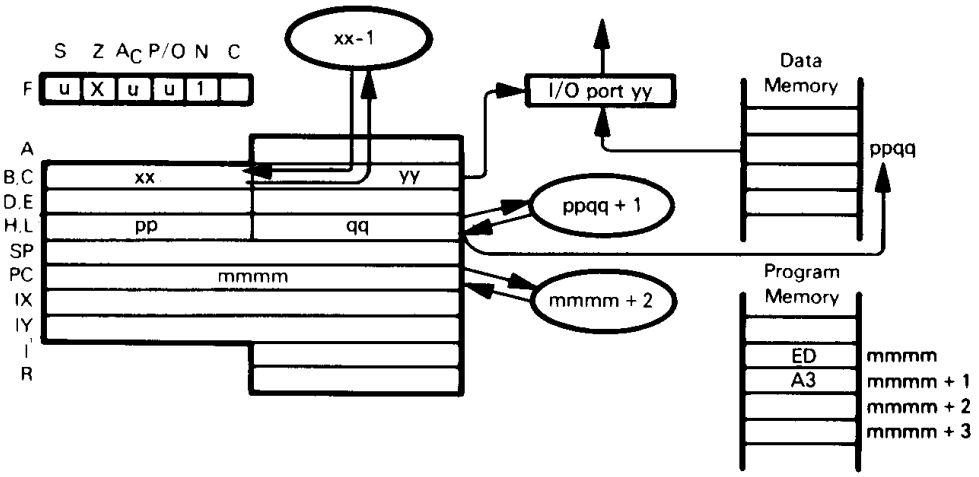
After execution of

OTDR

register pair HL will contain $4FFD_{16}$, Register B will contain zero, and the sequence $F1_{16}$, $1B_{16}$, CA_{16} will have been written to I/O port FF_{16}.

This instruction is very useful for transferring blocks of data from memory to output devices.

OUTI — OUTPUT FROM MEMORY. INCREMENT ADDRESS



OUTI

ED A3

Output from memory location specified by HL to I/O port addressed by Register C. Register B is decremented and the HL register pair is incremented.

Suppose xx=$0A_{16}$, yy=FF_{16}, ppqq=5000_{16}, and memory location 5000_{16} contains 77_{16}. After the instruction

OUTI

has executed, 77_{16} will be held in the buffer of I/O port FF_{16}. The B register will contain 09_{16} and the HL register pair will contain 5001_{16}.

OTIR — OUTPUT FROM MEMORY. INCREMENT ADDRESS, CONTINUE UNTIL REGISTER B=0

OTIR

ED B3

OTIR is identical to OUTI, except that it is repeated until Register B contains 0.

Suppose Register B contains 04_{16}, Register C contains FF_{16}, and HL contains 5000_{16}. Memory locations 5000_{16} through 5003_{16} contain:

Location	Contents
5000_{16}	CA_{16}
5001_{16}	$1B_{16}$
5002_{16}	$B1_{16}$
5003_{16}	AD_{16}

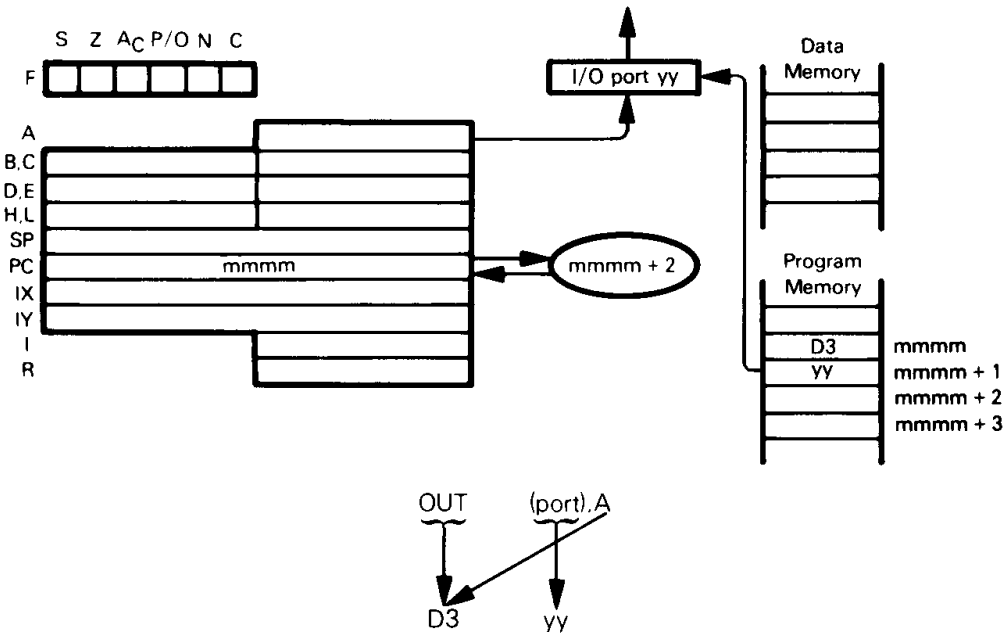
After execution of

OTIR

register pair HL will contain 5004_{16}, Register B will contain zero and the sequence CA_{16}, $1B_{16}$, $B1_{16}$ and AD_{16} will have been written to I/O port FF_{16}.

This instruction is very useful for transferring blocks of data from memory to an output device.

OUT (port),A — OUTPUT FROM ACCUMULATOR



Output the contents of the Accumulator to the I/O port identified by the second OUT instruction object code byte.

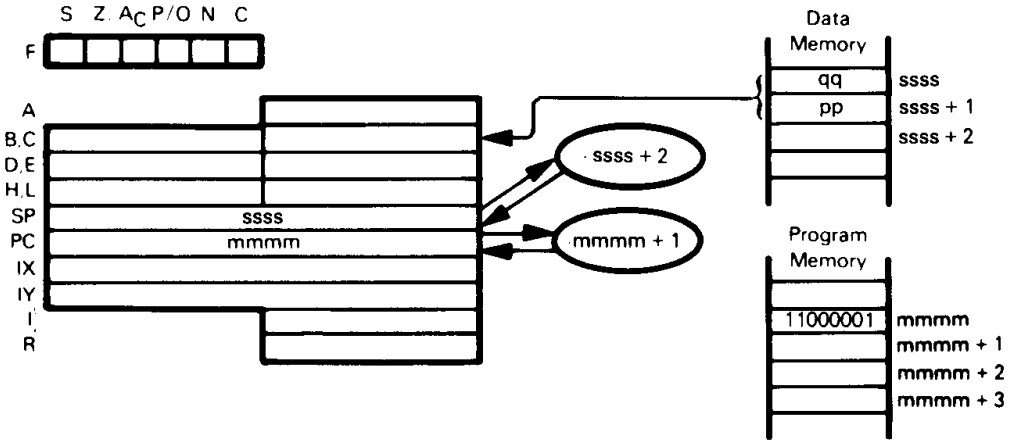
Suppose 36_{16} is held in the Accumulator. After the instruction

OUT (1AH),A

has executed, 36_{16} will be in the buffer of I/O port $1A_{16}$.

The OUT instruction does not affect any statuses. Use of the OUT instruction is very hardware-dependent. Valid I/O port addresses are determined by the way in which I/O logic has been implemented. It is also possible to design a microcomputer system that accesses external logic using memory reference instructions with specific memory addresses. OUT instructions are frequently used in special ways to control microcomputer logic external to the CPU.

POP rp — READ FROM THE TOP OF THE STACK
POP IX
POP IY



The illustration shows execution of POP BC:

POP rp

11 xx 0001

00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is register pair A and F

POP the two top stack bytes into the designated register pair.

Suppose qq=01_{16} and pp=$2A_{16}$. Execution of

POP HL

loads 01_{16} into the L register and $2A_{16}$ into the H register. Execution of the instruction

POP AF

loads 01 into the status flags and $2A_{16}$ into the Accumulator. Thus, the Carry status will be set to 1 and other statuses will be cleared.

POP IX

DD E1

POP the two top stack bytes into the IX register.

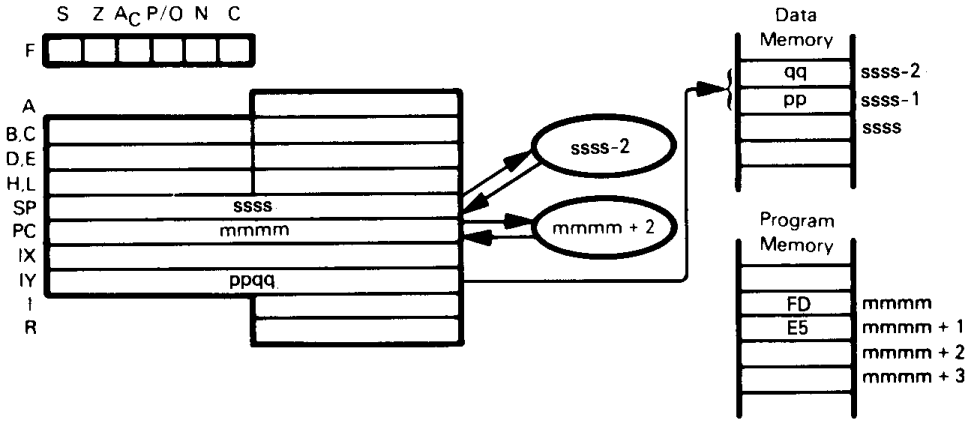
POP IY

FD E1

POP the two top stack bytes into the IY register.

The POP instruction is most frequently used to restore register and status contents which have been saved on the stack; for example, while servicing an interrupt.

PUSH rp — WRITE TO THE TOP OF THE STACK
PUSH IX
PUSH IY



The illustration shows execution of PUSH IY:

PUSH IY
FD E5

PUSH the contents of the IY register onto the top of the stack.

Suppose the IY register contains $45FF_{16}$. Execution of the instruction

PUSH IY

loads 45_{16}, then FF_{16} onto the top of the stack.

PUSH IX
DD E5

PUSH the contents of the IX register onto the top of the stack.

PUSH rp
11 xx 0101

00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is register pair A and F

PUSH contents of designated register pair onto the top of the stack.

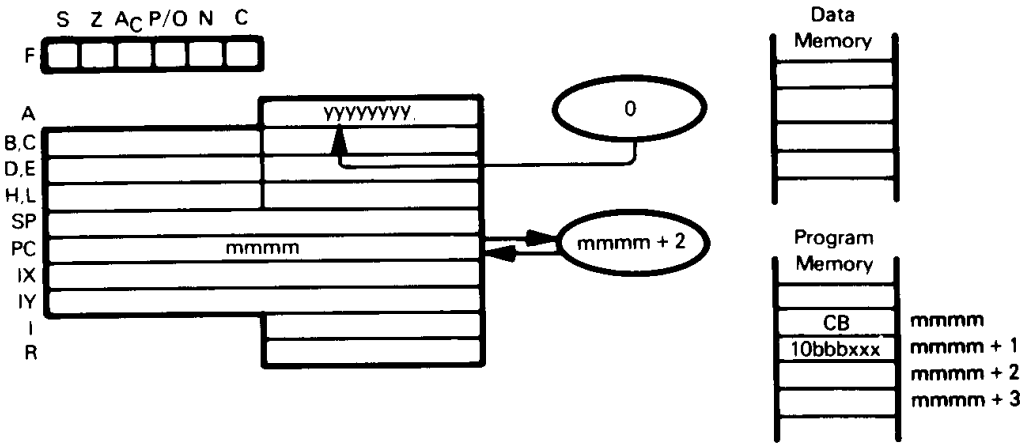
Execution of the instruction

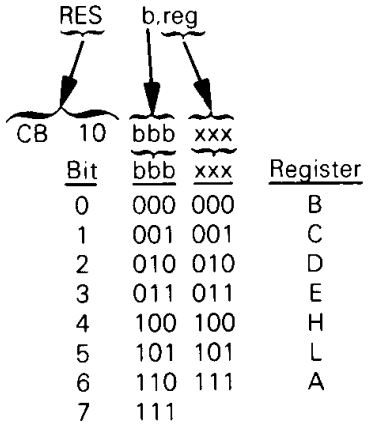
PUSH AF

loads the Accumulator and then the status flags onto the top of the stack.

The PUSH instruction is most frequently used to save register and status contents; for example, before servicing an interrupt.

RES b,reg — RESET INDICATED REGISTER BIT





Bit	bbb	xxx	Register
0	000	000	B
1	001	001	C
2	010	010	D
3	011	011	E
4	100	100	H
5	101	101	L
6	110	111	A
7	111		

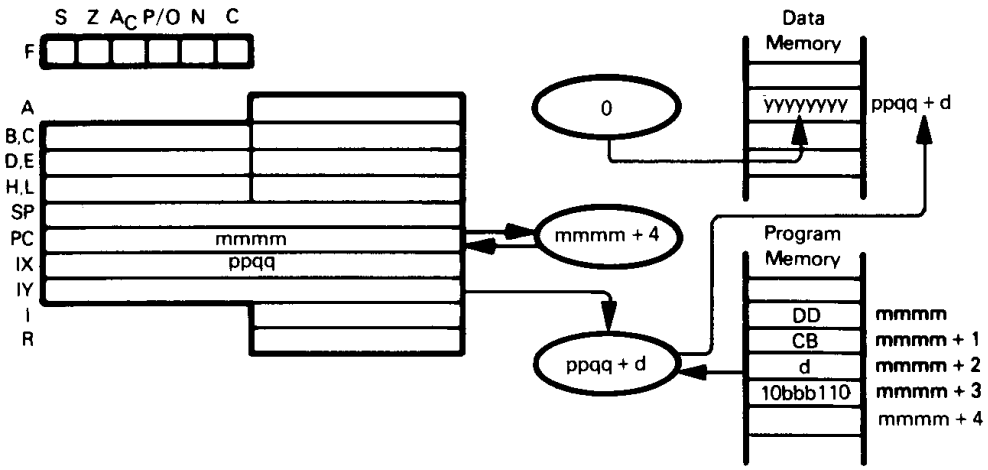
Reset indicated bit within specified register.

After the instruction

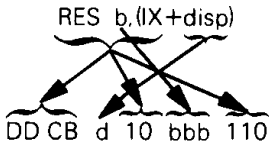
RES 6,H

has executed, bit 6 in Register H will be reset. (Bit 0 is the least significant bit.)

RES b,(HL) — RESET BIT b OF INDICATED MEMORY POSITION
RES b,(IX+disp)
RES b,(IY+disp)



The illustration shows execution of SET b,(IX+disp). Bit 0 is execution of SET b,(IX+disp). Bit 0 is the least significant bit.



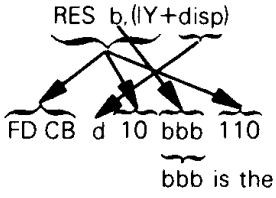
bbb	Bit Reset
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Reset indicated bit within memory location indicated by the sum of Index Register IX and d.

Suppose IX contains 4110_{16}. After the instruction

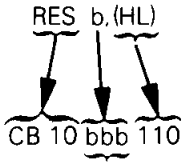
RES 0,(IX+7)

has executed, bit 0 in memory location 4117_{16} will be 0.



This instruction is identical to RES b,(IX+disp), except that it uses the IY register instead

of the IX register.



bbb is the same as in RES b,(IX+disp)

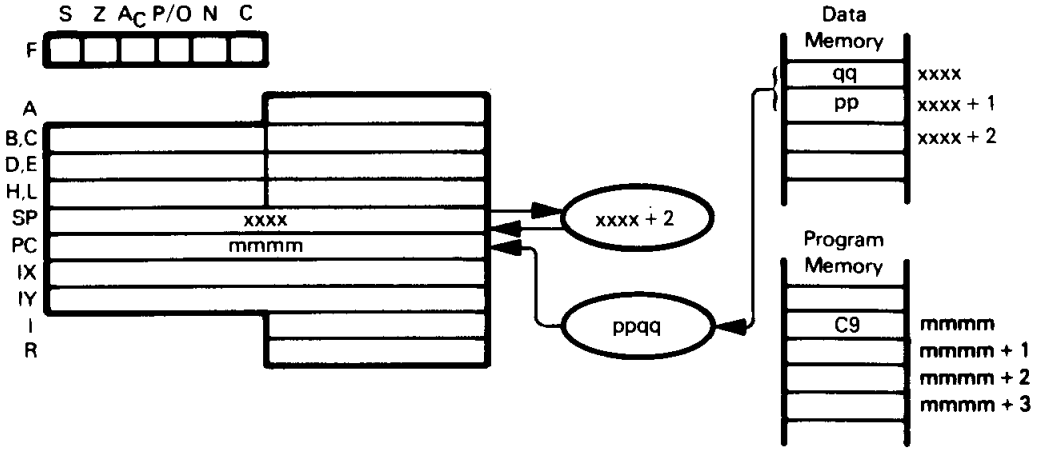
Reset indicated bit within memory location indicated by HL.

Suppose HL contains 4444_{16}. After execution of

RES 7,(HL)

bit 7 in memory location 4444_{16} will be 0.

RET — RETURN FROM SUBROUTINE



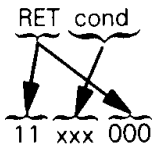
RET

C9

Move the contents of the top two stack bytes to the Program Counter; these two bytes provide the address of the next instruction to be executed. Previous Program Counter contents are lost. Increment the Stack Pointer by 2, to address the new top of stack.

Every subroutine must contain at least one Return (or conditional Return) instruction; this is the last instruction executed within the subroutine, and causes execution to return to the calling program.

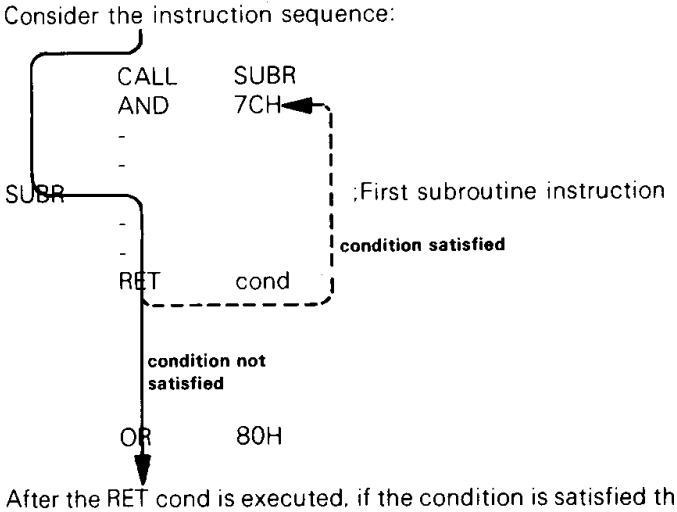
RET cond — RETURN FROM SUBROUTINE IF CONDITION IS SATISFIED



		Condition	Relevant Flag
000	NZ	Non-Zero	Z
001	Z	Zero	Z
010	NC	Non-Carry	C
011	C	Carry	C
100	PO	Parity Odd	P/O
101	PE	Parity Even	P/O
110	P	Sign Positive	S
111	M	Sign Negative	S

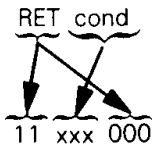
This instruction is identical to the RET instruction, except that the return is not executed unless the condition is satisfied; otherwise, the instruction sequentially following the RET cond instruction will be executed.

Consider the instruction sequence:



After the RET cond is executed, if the condition is satisfied then execution returns to the AND instruction which follows the CALL. If the condition is not satisfied, the OR instruction, being the next sequential instruction, is executed.

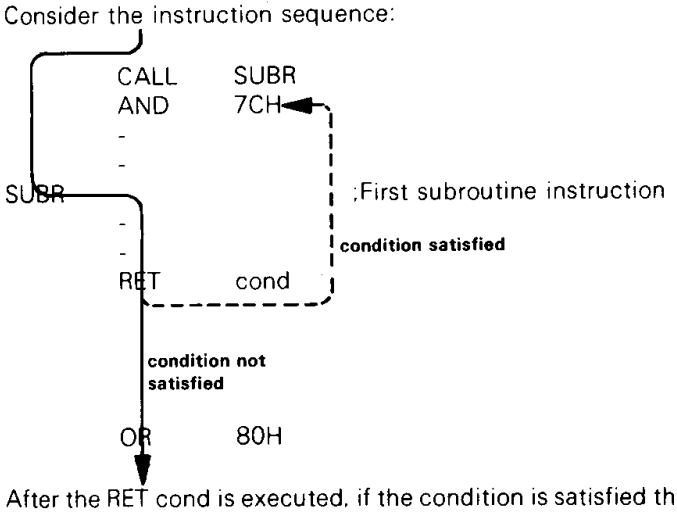
RET cond — RETURN FROM SUBROUTINE IF CONDITION IS SATISFIED



		Condition	Relevant Flag
000	NZ	Non-Zero	Z
001	Z	Zero	Z
010	NC	Non-Carry	C
011	C	Carry	C
100	PO	Parity Odd	P/O
101	PE	Parity Even	P/O
110	P	Sign Positive	S
111	M	Sign Negative	S

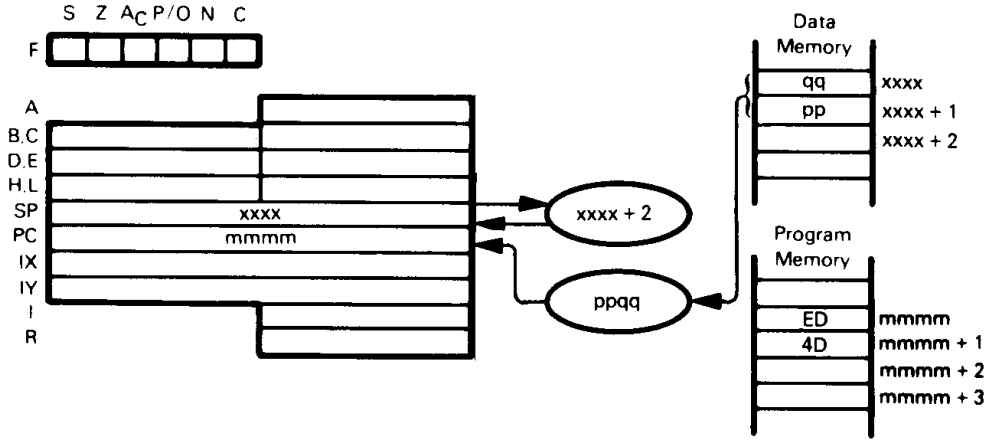
This instruction is identical to the RET instruction, except that the return is not executed unless the condition is satisfied; otherwise, the instruction sequentially following the RET cond instruction will be executed.

Consider the instruction sequence:



After the RET cond is executed, if the condition is satisfied then execution returns to the AND instruction which follows the CALL. If the condition is not satisfied, the OR instruction, being the next sequential instruction, is executed.

RETI — RETURN FROM INTERRUPT



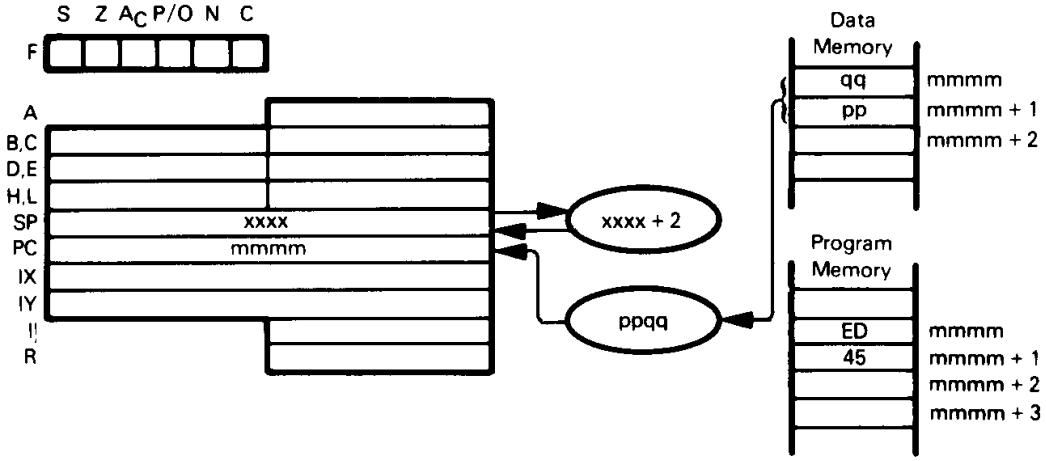
RETI

ED 4D

Move the contents of the top two stack bytes to the Program Counter; these two bytes provide the address of the next instruction to be executed. Previous Program Counter contents are lost. Increment the Stack Pointer by 2, and address the new top of stack.

This instruction is used at the end of an interrupt service routine, and, in addition to returning control to the interrupted program, it is used to signal an I/O device that the interrupt routine has been completed. The I/O device must provide the logic necessary to sense the instruction operation code: refer to An Introduction to Microcomputers: Volume 2 for a description of how the RETI instruction operates with the Z80 family of devices.

RETN — RETURN FROM NON-MASKABLE INTERRUPT

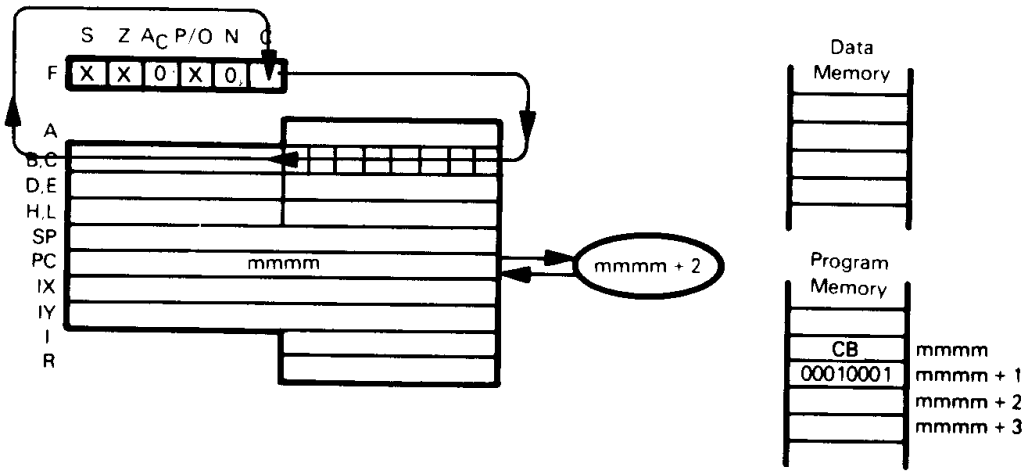


RETN
ED 45

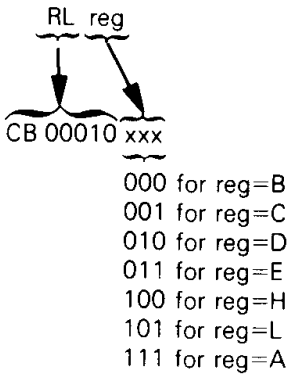
Move the contents of the top two stack bytes to the Program Counter; these two bytes provide the address of the next instruction to be executed. Previous Program Counter contents are lost. Increment the Stack Pointer by 2 to address the new top of stack. Restore the interrupt enable logic to the state it had prior to the occurrence of the non-maskable interrupt.

This instruction is used at the end of a service routine for a non-maskable interrupt, and causes execution to return to the program that was interrupted.

RL reg — ROTATE CONTENTS OF REGISTER LEFT THROUGH CARRY



The illustration shows execution of RL C:

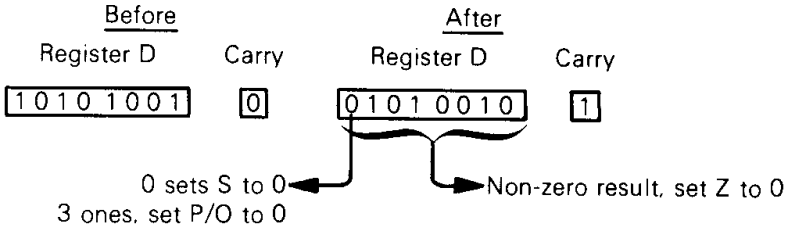


Rotate contents of specified register left one bit through Carry.

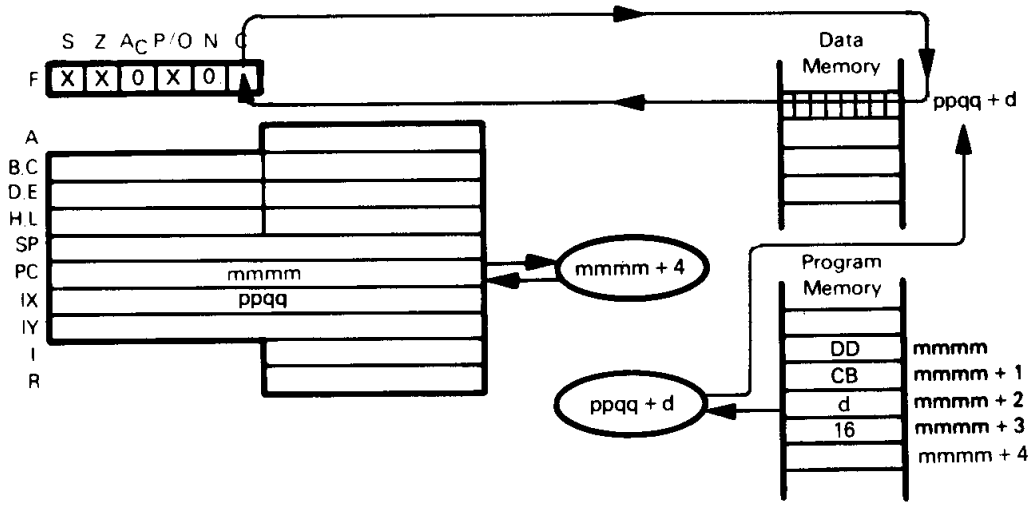
Suppose D contains $A9_{16}$ and Carry=0. After the instruction

RL D

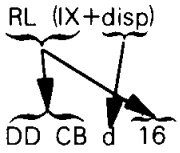
has executed, D will contain 52_{16} and Carry will be 1:



RL (HL) — ROTATE CONTENTS OF MEMORY LOCATION
RL (IX+disp) LEFT THROUGH CARRY
RL (IY+disp)



The illustration shows execution of RL (IX+disp):

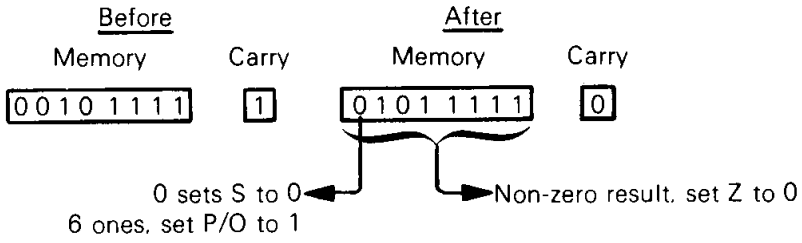


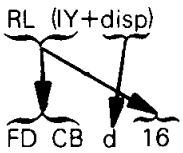
Rotate contents of memory location (specified by the sum of the contents of Index Register IX and displacement integer d) left one bit through Carry.

Suppose the IX register contains 4000_{16}, memory location 4007_{16} contains $2F_{16}$, and Carry is set to 1. After execution of the instruction

RL (IX+7)

memory location 4007_{16} will contain $5F_{16}$, and Carry is 0:





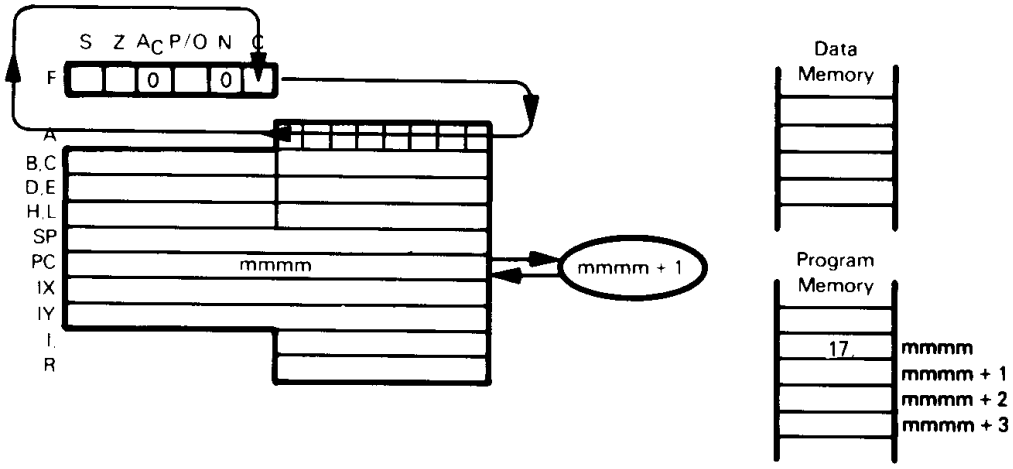
This instruction is identical to RL (IX+disp), but uses the IY register instead of the IX register.

RL (HL)

CB 16

Rotate contents of memory location (specified by the contents of the HL register pair) left one bit through Carry.

RLA — ROTATE ACCUMULATOR LEFT THROUGH CARRY



RLA

17

Rotate Accumulator contents left one bit through Carry status.

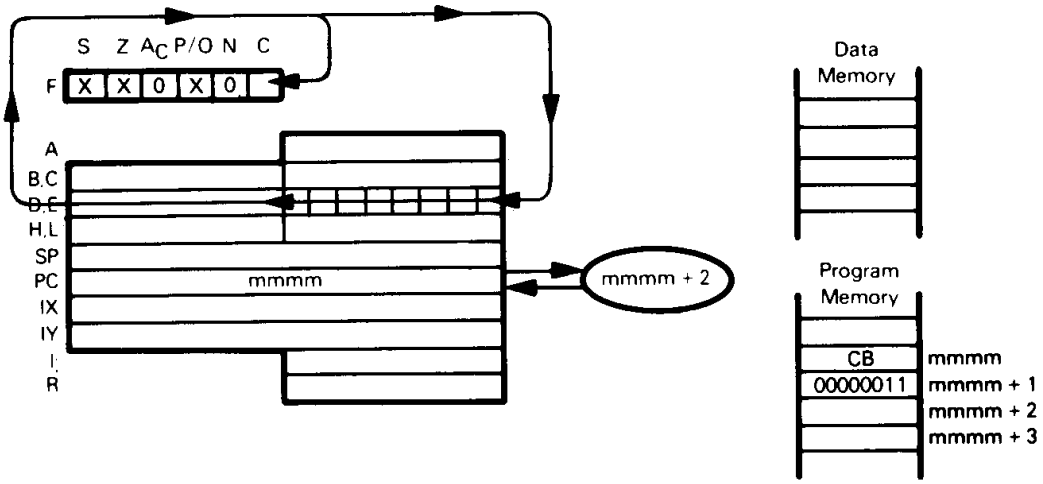
Suppose the Accumulator contains $2A_{16}$ and the Carry status is set to 1. After the instruction

RLA

has executed, the Accumulator will contain $F5_{16}$ and the Carry status will be reset to 0:

Before		After	
Accumulator	Carry	Accumulator	Carry
0 1 1 1 1 0 1 0	1	1 1 1 1 0 1 0 1	0

RLC reg — ROTATE CONTENTS OF REGISTER LEFT CIRCULAR



The illustration shows execution of RLC E:

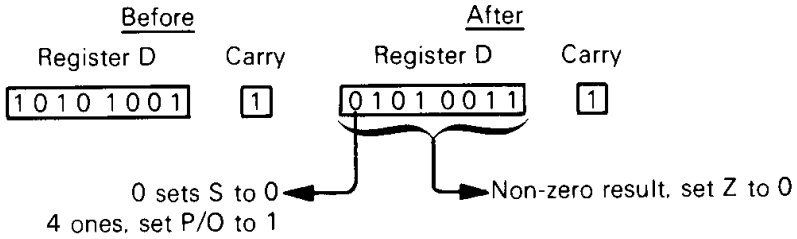
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Rotate contents of specified register left one bit, copying bit 7 into Carry.

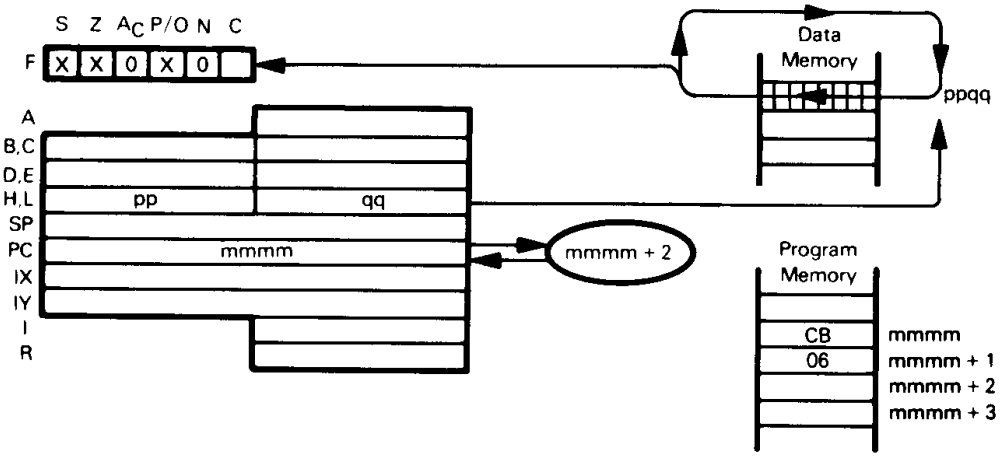
Suppose Register D contains $A9_{16}$ and Carry is 1. After execution of

RLC D

Register D will contain 53_{16} and Carry will be 1:



RLC (HL) — ROTATE CONTENTS OF MEMORY LOCATION
RLC (IX+disp) LEFT CIRCULAR
RLC (IY+disp)



The illustration shows execution of RLC (HL):

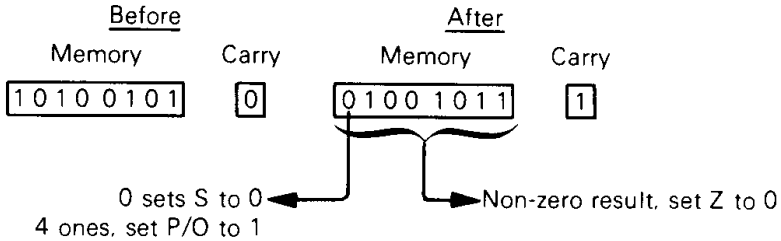
$$\underbrace{\text{RLC (HL)}}_{\text{CB 06}}$$

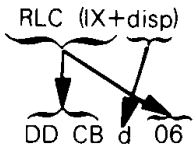
Rotate contents of memory location (specified by the contents of the HL register pair) left one bit, copying bit 7 into Carry.

Suppose register pair HL contains $54FF_{16}$. Memory location $54FF_{16}$ contains $A5_{16}$, and Carry is 0. After execution of

RLC (HL)

memory location $54FF_{16}$ will contain $4B_{16}$, and Carry will be 1:



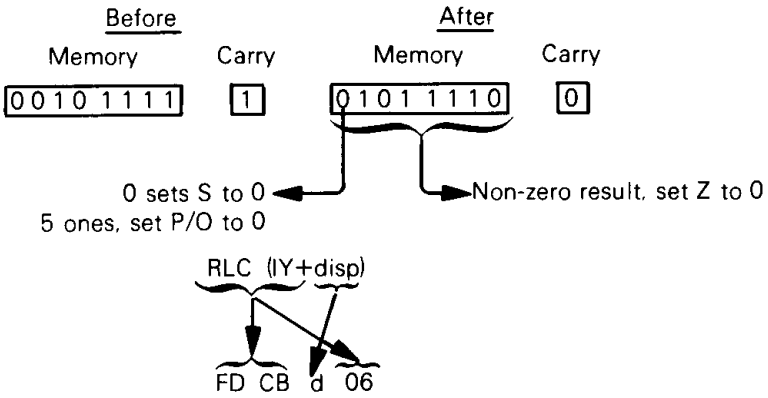


Rotate memory location (specified by the sum of the contents of Index register IX and displacement integer d) left one bit, copying bit 7 into Carry.

Suppose the IX register contains 4000_{16}. Carry is 1, and memory location 4007_{16} contains $2F_{16}$. After the instruction

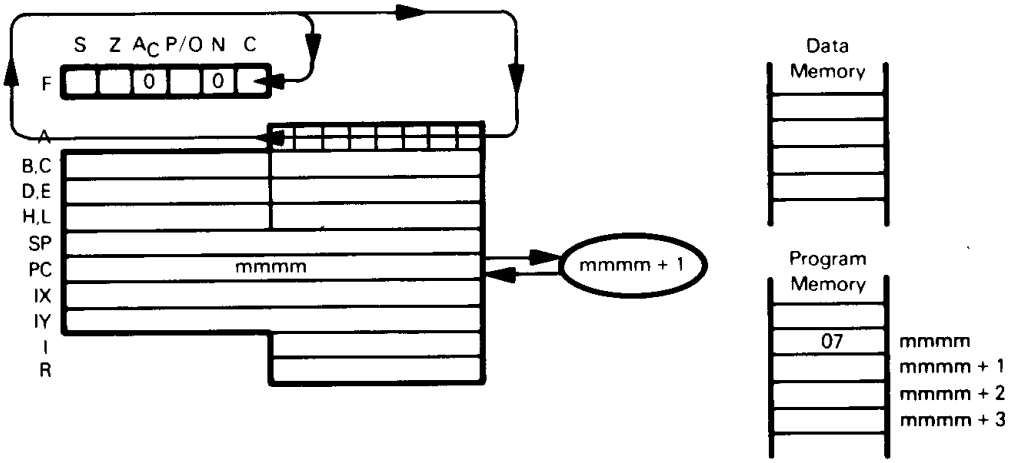
RLC (IX+7)

has executed, memory location 4007_{16} will contain $5E_{16}$, and Carry will be 0:



This instruction is identical to RLC (IX+disp), but uses the IY register instead of the IX register.

RLCA — ROTATE ACCUMULATOR LEFT CIRCULAR



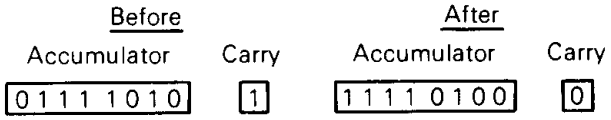
RLCA
07

Rotate Accumulator contents left one bit, copying bit 7 into Carry.

Suppose the Accumulator contains $7A_{16}$ and the Carry status is set to 1. After the instruction

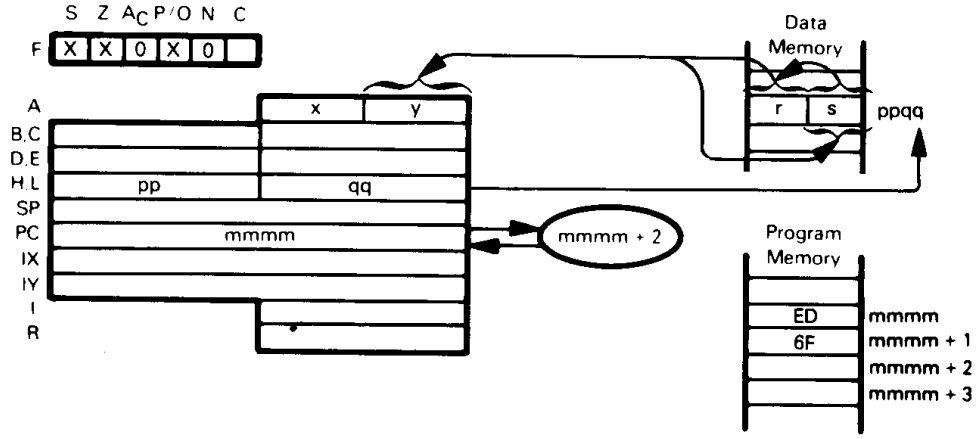
RLCA

has executed, the Accumulator will contain $F4_{16}$ and the Carry status will be reset to 0:



RLCA should be used as a logical instruction.

RLD — ROTATE ONE BCD DIGIT LEFT BETWEEN THE ACCUMULATOR AND MEMORY LOCATION



RLD

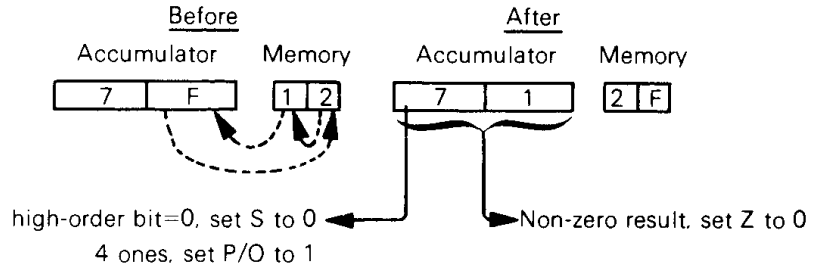
ED 6F

The four low-order bits of a memory location (specified by the contents of register pair HL) are copied into the four high-order bits of the same memory location. The previous contents of the four high-order bits of that memory location are copied into the four low-order bits of the Accumulator. The previous four low-order bits of the Accumulator are copied into the four low-order bits of the specified memory location.

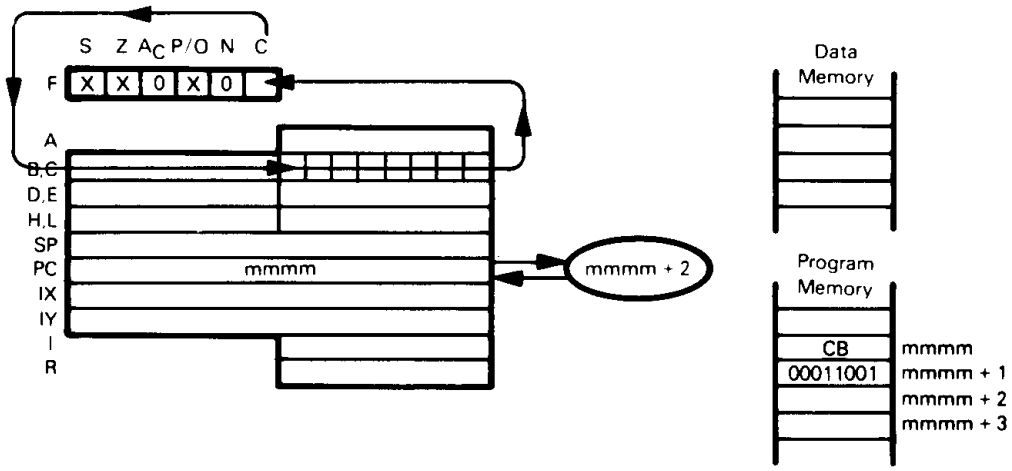
Suppose the Accumulator contains $7F_{16}$, HL register pair contains 4000_{16}, and memory location 4000_{16} contains 12_{16}. After execution of the instruction

RLD

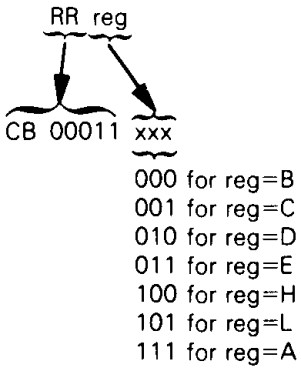
the Accumulator will contain 71_{16} and memory location 4000_{16} will contain $2F_{16}$:



RR reg — ROTATE CONTENTS OF REGISTER RIGHT THROUGH CARRY



The illustration shows execution of RR C:

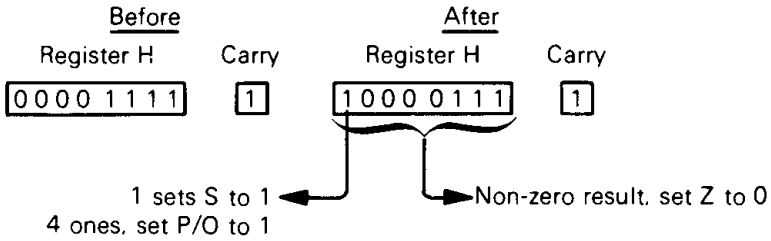


Rotate contents of specified register right one bit through Carry.

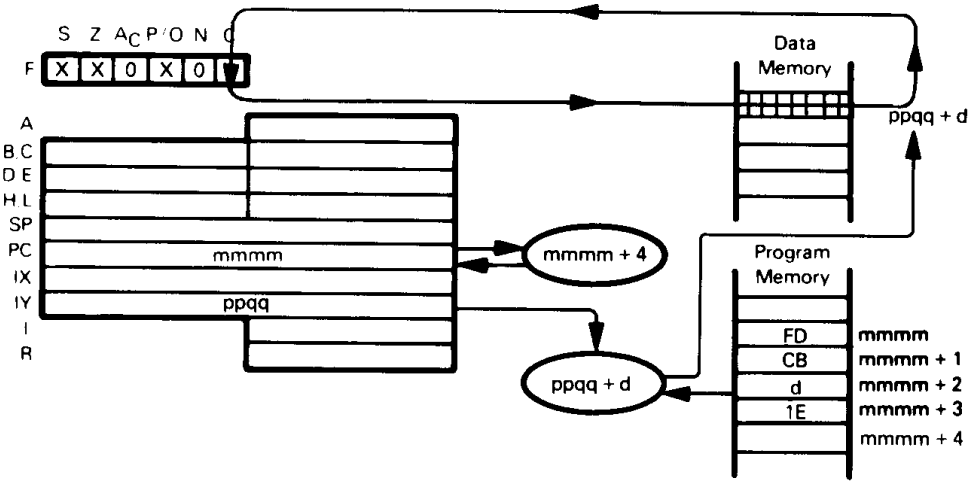
Suppose Register H contains $0F_{16}$ and Carry is set to 1. After the instruction

RR H

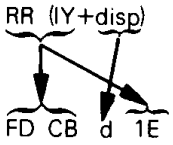
has executed, Register H will contain 87_{16}, and Carry will be 1:



RR (HL) — ROTATE CONTENTS OF MEMORY LOCATION RIGHT THROUGH CARRY
RR (IX+disp)
RR (IY+disp)



The illustration shows execution of RR (IY+disp):

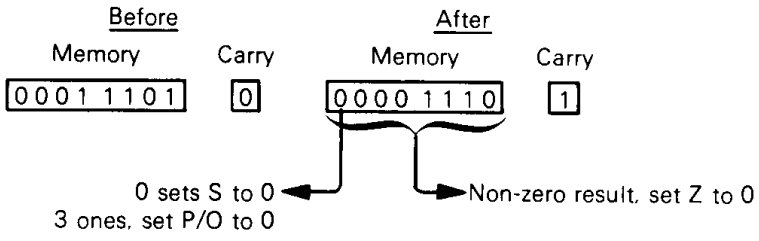


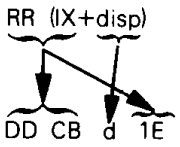
Rotate contents of memory location (specified by the sum of the contents of the IY register and the displacement value d) right one bit through Carry.

Suppose the IY register contains 4500_{16}, memory location $450F_{16}$ contains $1D_{16}$, and Carry is set to 0. After execution of the instruction

RR (IY+0FH)

memory location $450F_{16}$ will contain $0E_{16}$, and Carry will be 1:





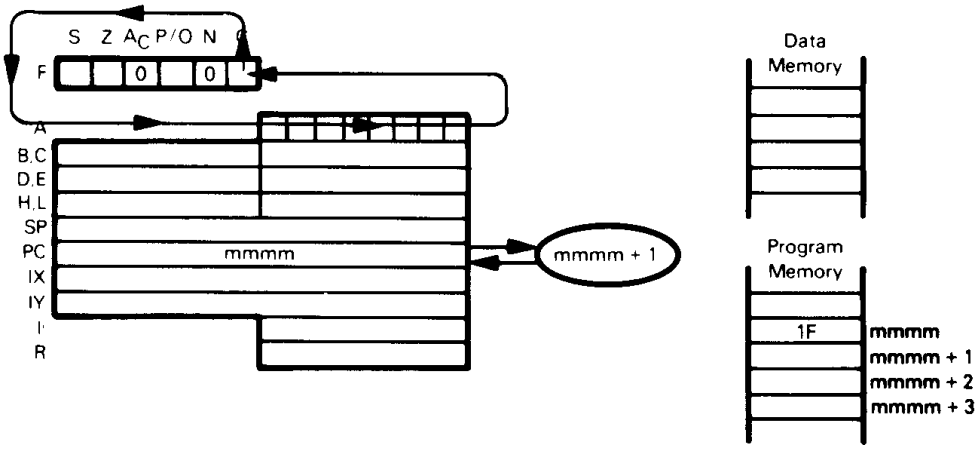
This instruction is identical to RR (IY+disp), but uses the IX register instead of the IY register.

RR (HL)

CB 1E

Rotate contents of memory location (specified by the contents of the HL register pair) right one bit through Carry.

RRA — ROTATE ACCUMULATOR RIGHT THROUGH CARRY



RRA

1F

Rotate Accumulator contents right one bit through Carry status.

Suppose the Accumulator contains $7A_{16}$ and the Carry status is set to 1. After the instruction

RRA

has executed, the Accumulator will contain BD_{16} and the Carry status will be reset to 0:

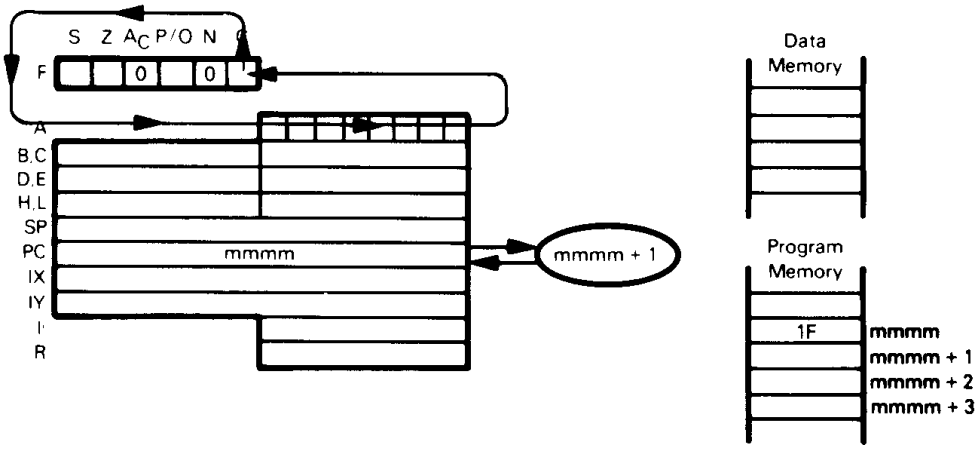
Before		After	
Accumulator	Carry	Accumulator	Carry
0 1 1 1 1 0 1 0	1	1 0 1 1 1 1 0 1	0

RR (HL)

CB 1E

Rotate contents of memory location (specified by the contents of the HL register pair) right one bit through Carry.

RRA — ROTATE ACCUMULATOR RIGHT THROUGH CARRY



RRA

1F

Rotate Accumulator contents right one bit through Carry status.

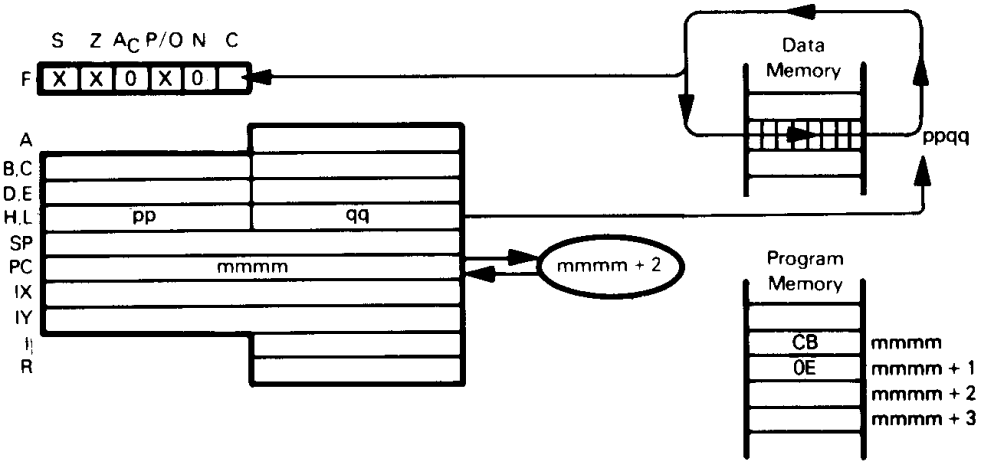
Suppose the Accumulator contains $7A_{16}$ and the Carry status is set to 1. After the instruction

RRA

has executed, the Accumulator will contain BD_{16} and the Carry status will be reset to 0:

Before		After	
Accumulator	Carry	Accumulator	Carry
0 1 1 1 1 0 1 0	1	1 0 1 1 1 1 0 1	0

RRC (HL) — ROTATE CONTENTS OF MEMORY LOCATION
RRC (IX+disp) RIGHT CIRCULAR
RRC (IY+disp)



The illustration shows execution of RRC (HL):

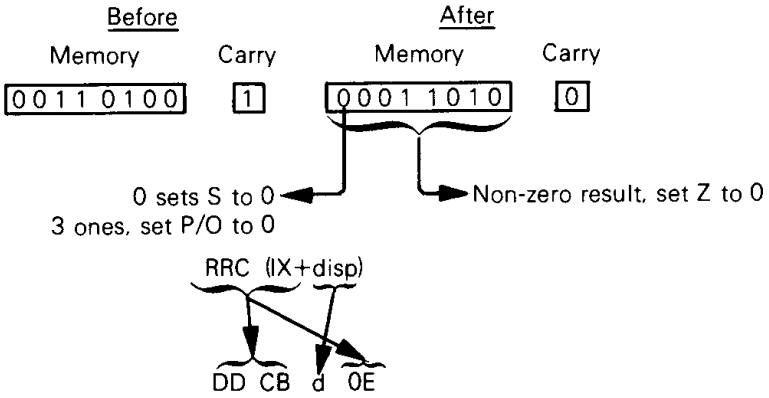
RRC (HL)
CB 0E

Rotate contents of memory location (specified by the contents of the HL register pair) right one bit circularly, copying bit 0 into the Carry status.

Suppose the HL register pair contains 4500_{16}, memory location 4500_{16} contains 34_{16}, and Carry is set to 1. After execution of

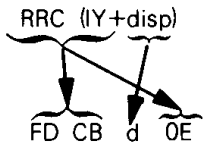
RRC (HL)

memory location 4500_{16} will contain $1A_{16}$, and Carry will be 0:



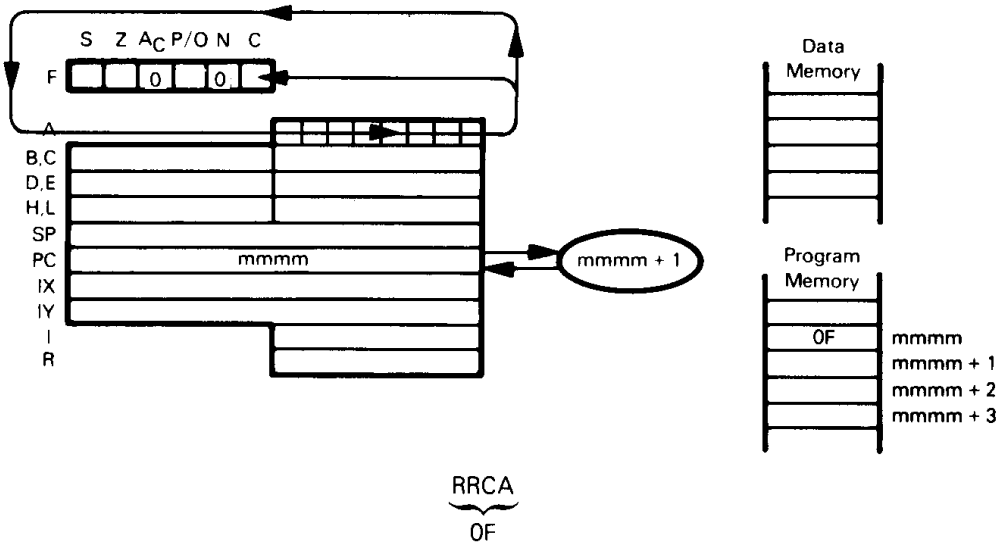
Rotate contents of memory location (specified by the sum of the contents of the IX

register and the displacement value d) right one bit circularly, copying bit 0 into the Carry status.



This instruction is identical to the RRC (IX+disp) instruction, but uses the IY register instead of the IX register.

RRCA — ROTATE ACCUMULATOR RIGHT CIRCULAR



RRCA

0F

Rotate Accumulator contents right one bit circularly, copying bit 0 into the Carry status.

Suppose the Accumulator contains $7A_{16}$ and the Carry status is set to 1. After the instruction

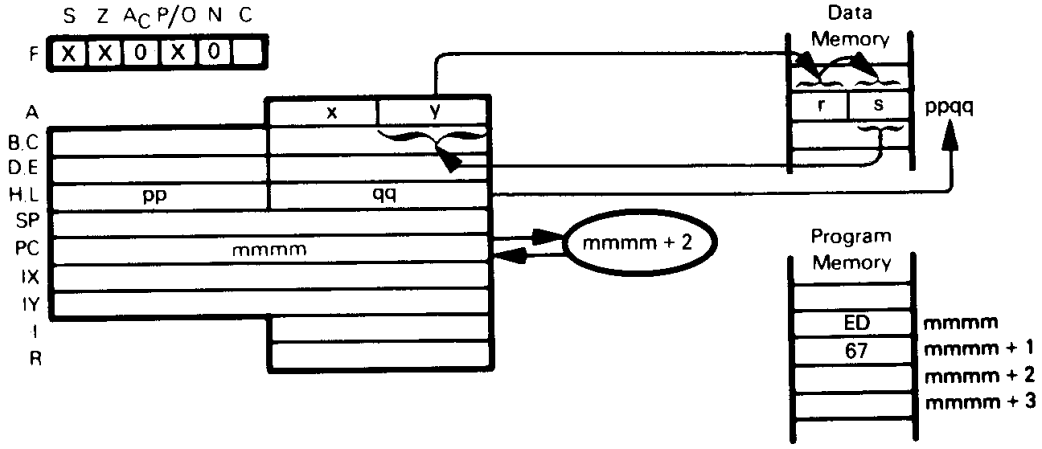
RRCA

has executed, the Accumulator will contain $3D_{16}$ and the Carry status will be reset to 0:

Before		After	
Accumulator	Carry	Accumulator	Carry
0 1 1 1 1 0 1 0	1	0 0 1 1 1 1 0 1	0

RRCA should be used as a logical instruction.

RRD — ROTATE ONE BCD DIGIT RIGHT BETWEEN THE ACCUMULATOR AND MEMORY LOCATION



RRD

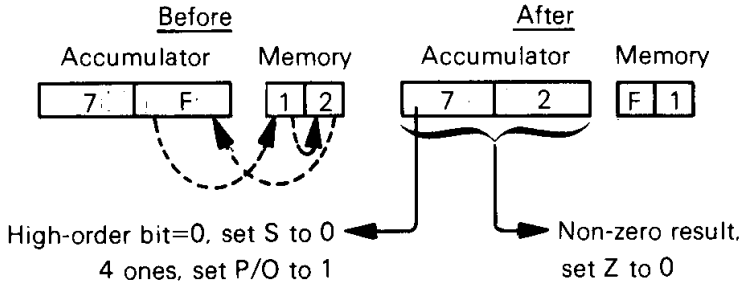
ED 67

The four high-order bits of a memory location (specified by the contents of register pair HL) are copied into the four low-order bits of the same memory location. The previous contents of the four low-order bits are copied into the four low-order bits of the Accumulator. The previous four low-order bits of the Accumulator are copied into the four high-order bits of the specified memory location.

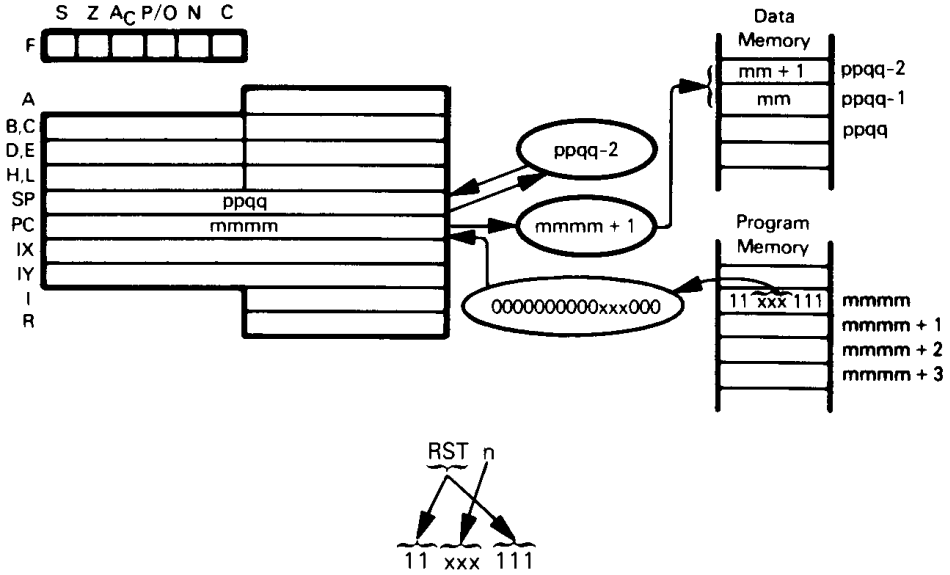
Suppose the Accumulator contains $7F_{16}$, HL register pair contains 4000_{16}, and memory location 4000_{16} contains 12_{16}. After execution of the instruction

RRD

the Accumulator will contain 72_{16} and memory location 4000_{16} will contain $F1_{16}$:



RST n — RESTART



Call the subroutine origined at the low memory address specified by n.

When the instruction

RST 18H

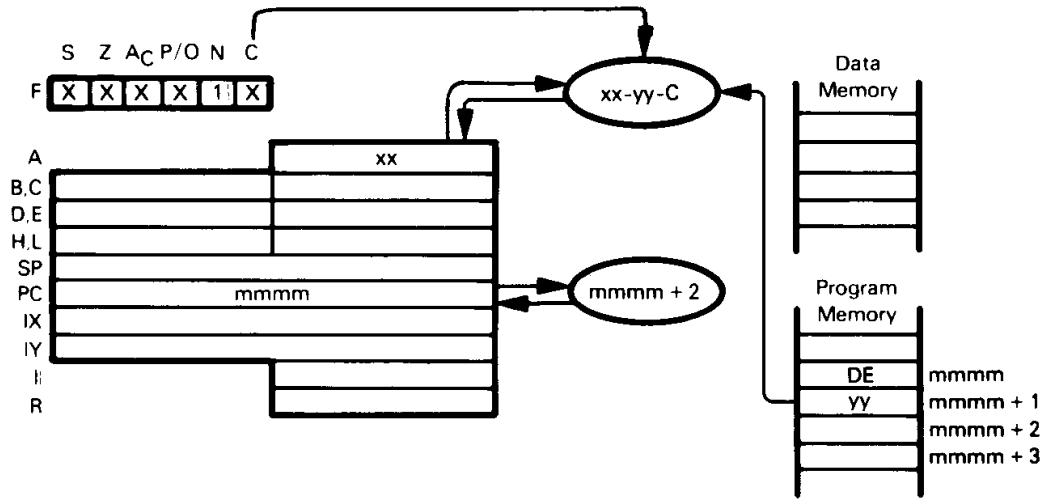
has executed, the subroutine origined at memory location 0018_{16} is called. The previous Program Counter contents are pushed to the top of the stack.

Usually, the RST instruction is used in conjunction with interrupt processing, as described in Chapter 12.

SUBROUTINE CALL USING RST

If your application does not use all RST instruction codes to service interrupts, do not overlook the possibility of calling subroutines using RST instructions. Origin frequently used subroutines at appropriate RST addresses, and these subroutines can be called with a single-byte RST instruction instead of a three-byte CALL instruction.

SBC A,data — SUBTRACT IMMEDIATE DATA FROM ACCUMULATOR WITH BORROW



SBC A, data
DE yy

Subtract the contents of the second object code byte and the Carry status from the Accumulator.

Suppose xx=$3A_{16}$ and Carry=1. After the instruction

SBC A,7CH

has executed, the Accumulator will contain BD_{16}.

```
3A                 =  0 0 1 1   1 0 1 0
Twos comp of 7C    =  1 0 0 0   0 1 0 0
Twos comp of Carry =  1 1 1 1   1 1 1 1
                      1 0 1 1   1 1 0 1
```

1 sets S to 1

Borrow, set C to 1

1 ⊻ 1=0, set P/O to 0

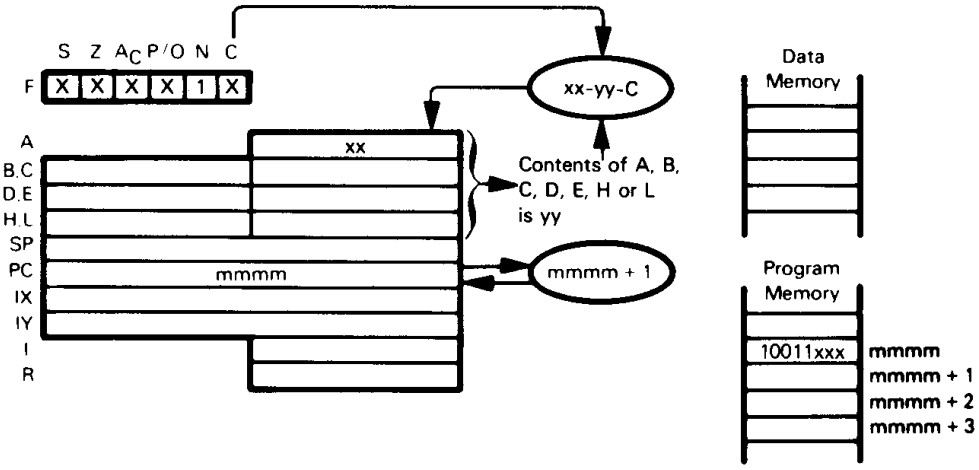
Non-zero result, set Z to 0

Borrow, set A_C to 1

Subtract instruction, set N to 1

The Carry flag is set to 1 for a borrow and reset to 0 if there is no borrow.

SBC A,reg — SUBTRACT REGISTER WITH BORROW FROM ACCUMULATOR



SBC A, reg

10011 xxx

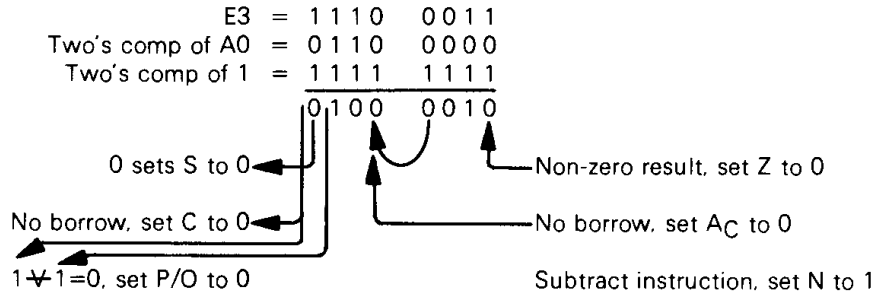
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Subtract the contents of the specified register and the Carry status from the Accumulator.

Suppose xx=$E3_{16}$, Register E contains $A0_{16}$, and Carry=1. After the instruction

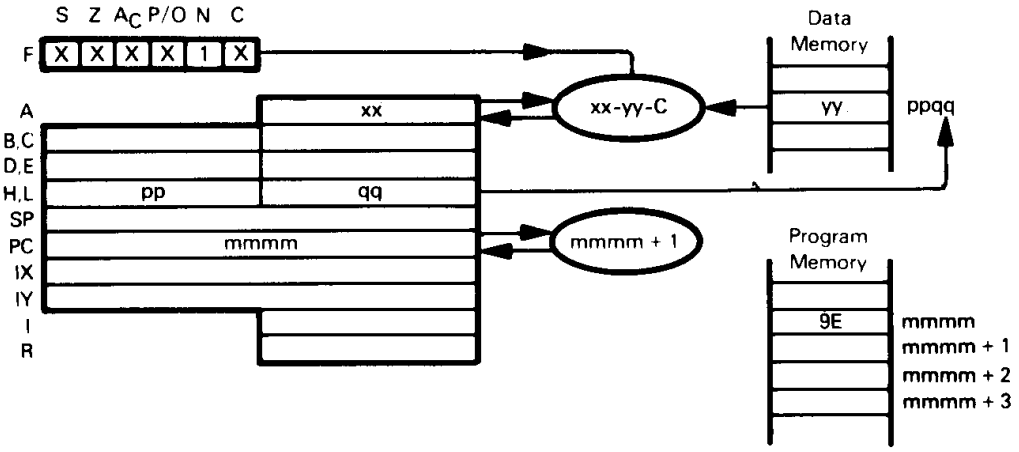
SBC A,E

has executed, the Accumulator will contain 42_{16}.



The Carry flag is set to 1 for a borrow and reset to 0 if there is no borrow.

SBC A,(HL) — SUBTRACT MEMORY AND CARRY FROM ACCUMULATOR
SBC A,(IX+disp)
SBC A,(IY+disp)



The illustration shows execution of SBC A,(HL):

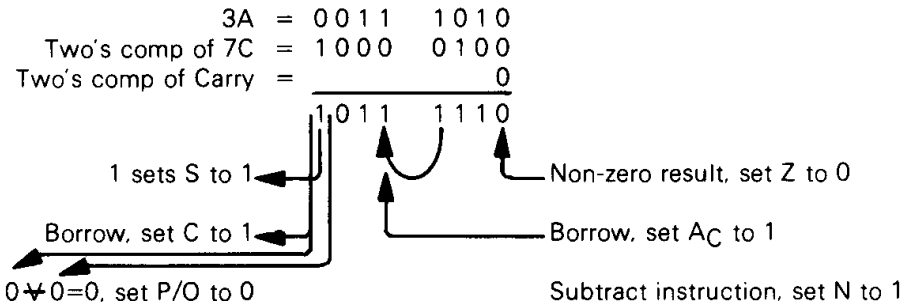
SBC A,(HL)
9E

Subtract the contents of memory location (specified by the contents of the HL register pair) and the Carry from the Accumulator.

Suppose Carry=0, ppqq=4000_{16}, xx=$3A_{16}$, and memory location 4000_{16} contains $7C_{16}$. After execution of the instruction

SBC A,(HL)

the Accumulator will contain BE_{16}.



The Carry flag is set to 1 for a borrow and reset to 0 if there is no borrow.

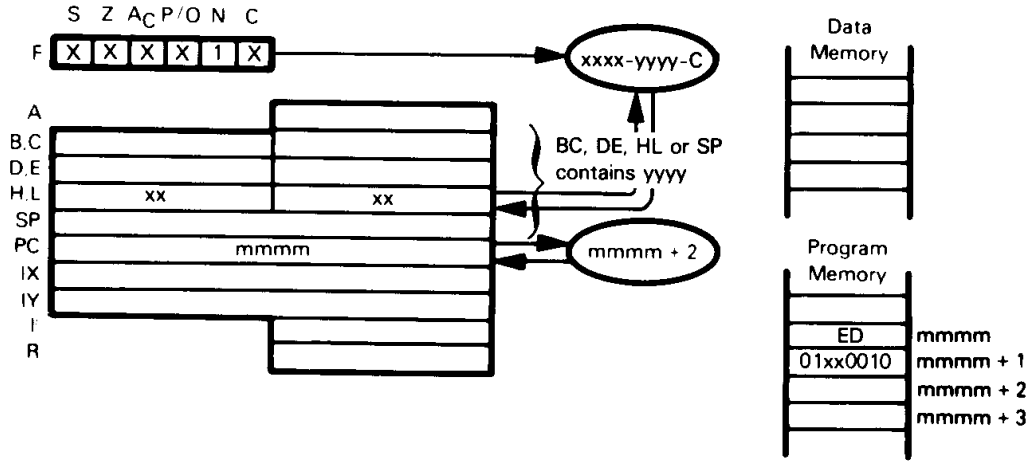
SBC A,(IX+disp)
DD 9E d

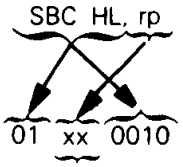
Subtract the contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) and the Carry from the Accumulator.

SBC A,(IY+disp)
FD 9E d

This instruction is identical to the SBC A,(IX+disp) instruction, except that it uses the IY register instead of the IX register.

SBC HL,rp — SUBTRACT REGISTER PAIR WITH CARRY FROM H AND L





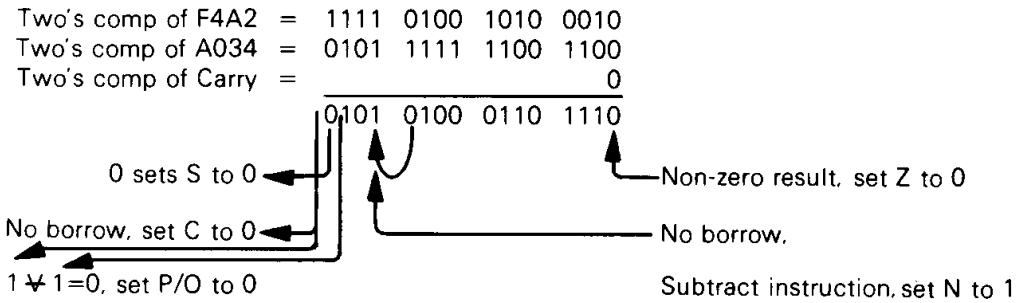
00 for rp is register pair BC
01 for rp is register pair DE
10 for rp is register pair HL
11 for rp is Stack Pointer

Subtract the contents of the designated register pair and the Carry status from the HL register pair.

Suppose HL contains $F4A2_{16}$, BC contains $A034_{16}$, and Carry=0. After the instruction

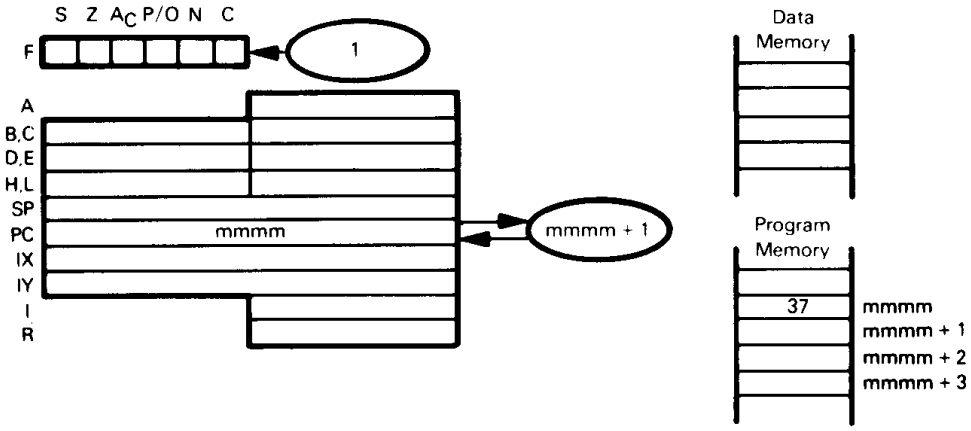
SBC HL,BC

has executed, the HL register pair will contain $546E_{16}$:



The Carry flag is set to 1 for a borrow and reset to 0 if there is no borrow.

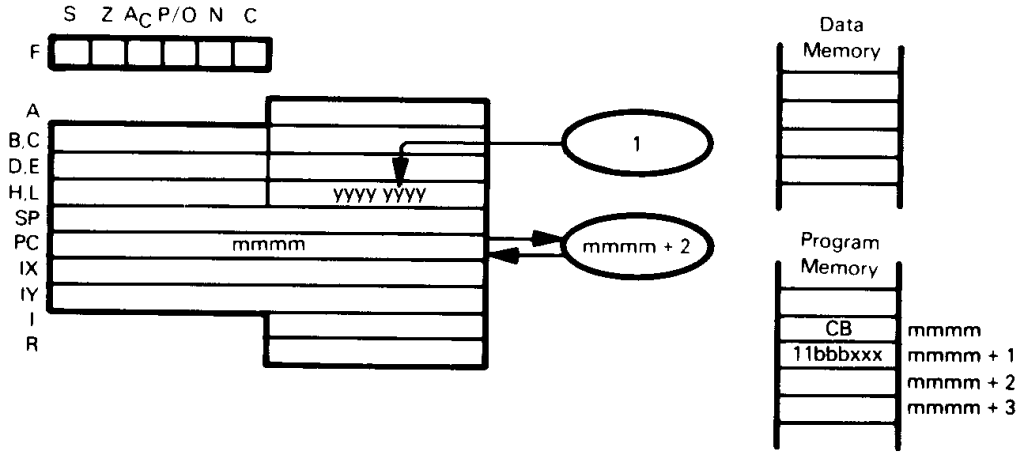
SCF — SET CARRY FLAG

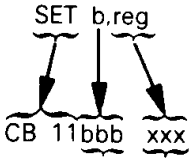


SCF
37

When the SCF instruction is executed, the Carry status is set to 1 regardless of its previous value. No other statuses or register contents are affected.

SET b,reg — SET INDICATED REGISTER BIT





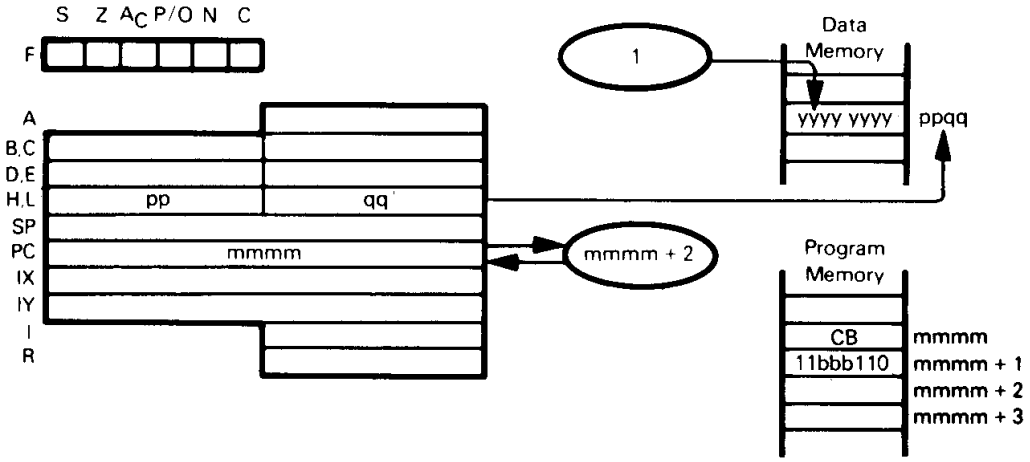
Bit	bbb	xxx	Register
0	000	000	B
1	001	001	C
2	010	010	D
3	011	011	E
4	100	100	H
5	101	101	L
6	110	111	A
7	111		

SET indicated bit within specified register. After the instruction

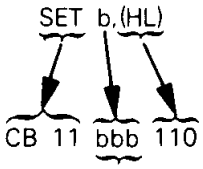
SET 2,L

has executed, bit 2 in Register L will be set. (Bit 0 is the least significant bit.)

SET b,(HL) — SET BIT b OF INDICATED MEMORY POSITION
SET b,(IX+disp)
SET b,(IY+disp)



The illustration shows execution of SET b,(HL). Bit 0 is the least significant bit.



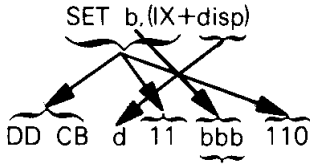
Bit Set	bbb
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Set indicated bit within memory location indicated by HL.

Suppose HL contains 4000_{16}. After the instruction

SET 5,(HL)

has executed, bit 5 in memory position 4000_{16} will be 1.



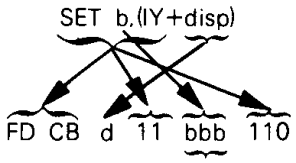
bbb is the same as in SET b,(HL)

Set indicated bit within memory location indicated by the sum of Index Register IX and displacement.

Suppose Index Register IX contains 4000_{16}. After execution of

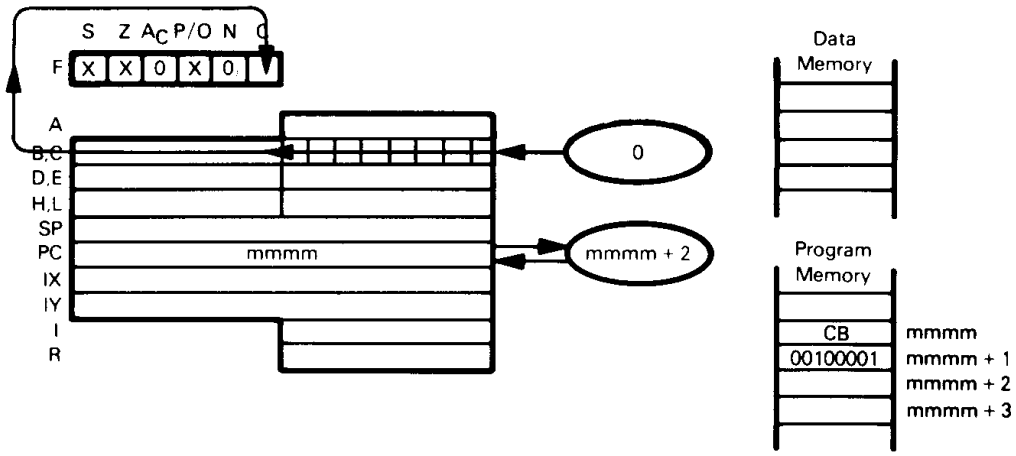
SET 6,(IX+5H)

bit 6 in memory location 4005_{16} will be 1.

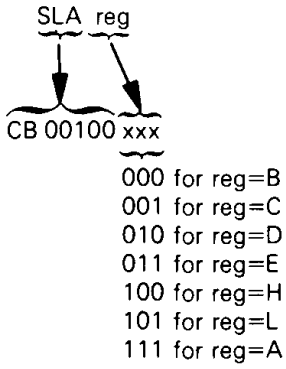


This instruction is identical to SET b,(IX+disp), except that it uses the IY register instead of the IX register.

SLA reg — SHIFT CONTENTS OF REGISTER LEFT ARITHMETIC



The illustration shows execution of SLA C:

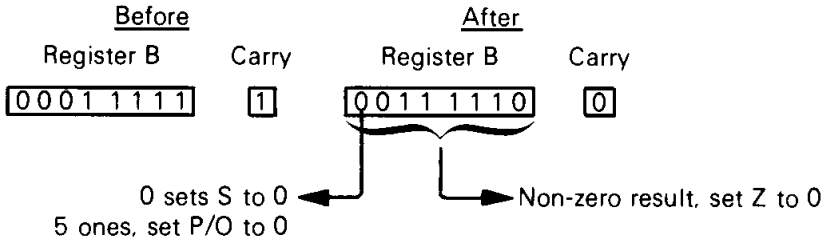


Shift contents of specified register left one bit, resetting the least significant bit to 0.

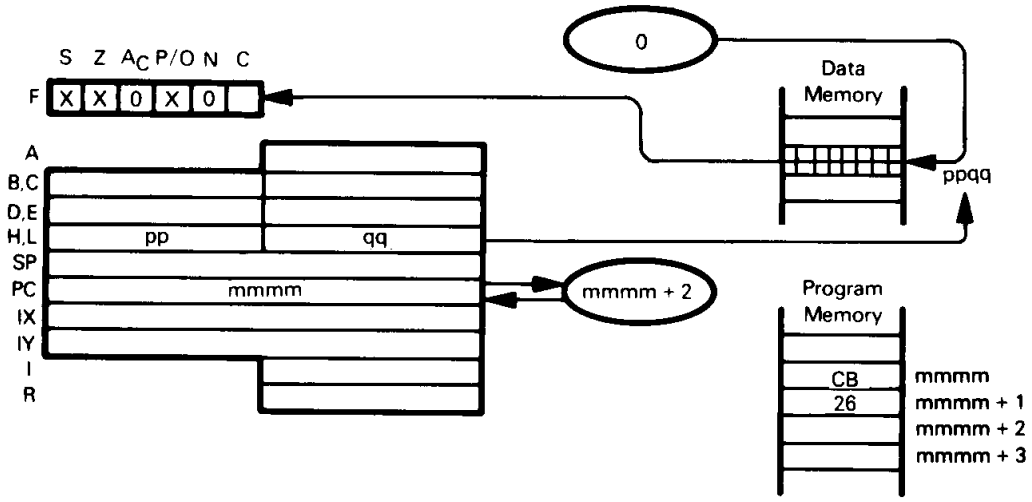
Suppose Register B contains $1F_{16}$, and Carry=1. After execution of

SLA B

Register B will contain $3E_{16}$ and Carry will be zero.



SLA (HL) — SHIFT CONTENTS OF MEMORY LOCATION
SLA (IX+disp) LEFT ARITHMETIC
SLA (IY+disp)



The illustration shows execution of SLA (HL):

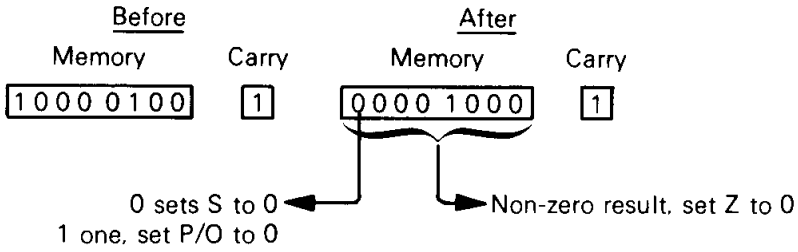
SLA (HL)
CB 26

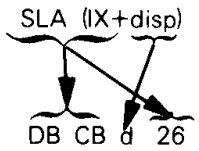
Shift contents of memory location (specified by the contents of the HL register pair) left one bit, resetting the least significant bit to 0.

Suppose the HL register pair contains 4500_{16}, memory location 4500_{16} contains 84_{16}, and Carry=0. After execution of

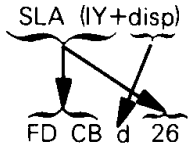
SLA (HL)

memory location 4500_{16} will contain 08_{16}, and Carry will be 1.



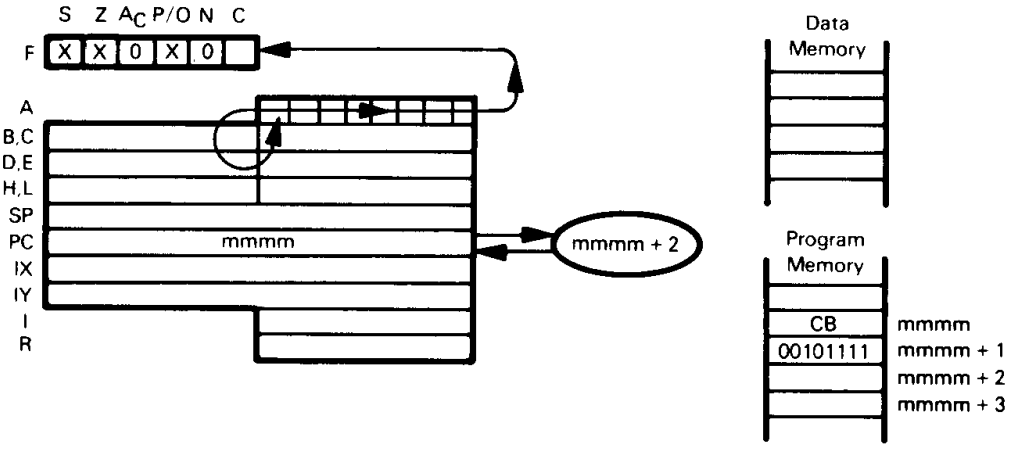


Shift contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) left one bit arithmetically, resetting least significant bit to 0.

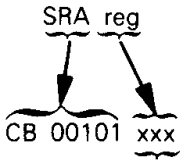


This instruction is identical to SLA (IX+disp), but uses the IY register instead of the IX register.

SRA reg — ARITHMETIC SHIFT RIGHT CONTENTS OF REGISTER



The illustration shows execution of SRA A:



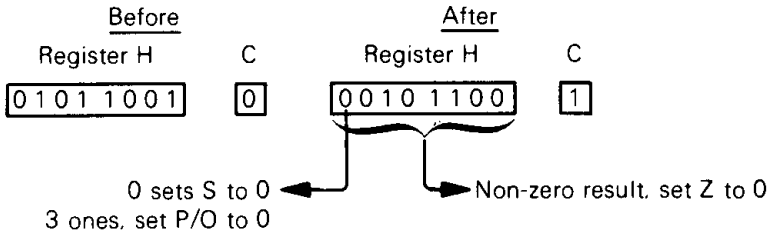
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Shift specified register right one bit. Most significant bit is unchanged.

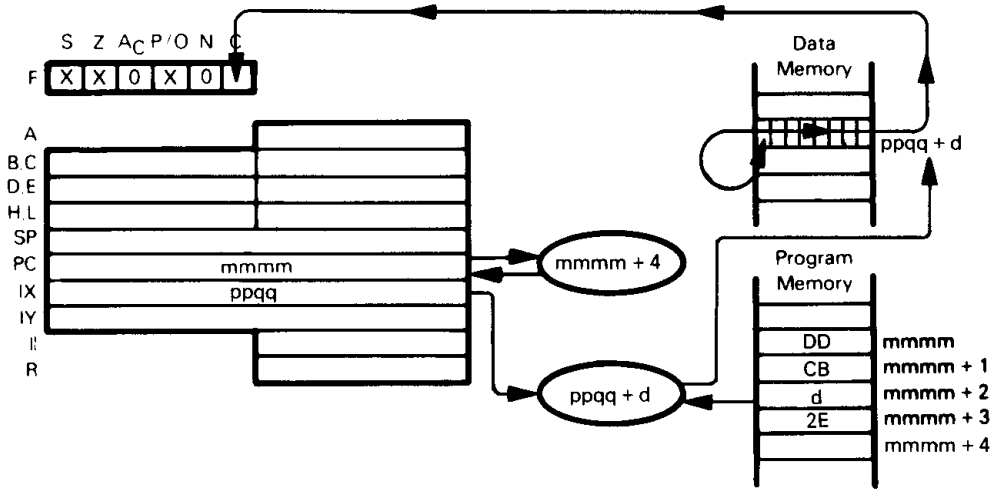
Suppose Register H contains 59_{16}, and Carry=0. After the instruction

SRA H

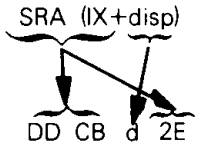
has executed, Register H will contain $2C_{16}$ and Carry will be 1.



SRA (HL) — ARITHMETIC SHIFT RIGHT CONTENTS OF SRA (IX+disp) MEMORY POSITION SRA (IY+disp)



The illustration shows execution of SRA (IX+disp):

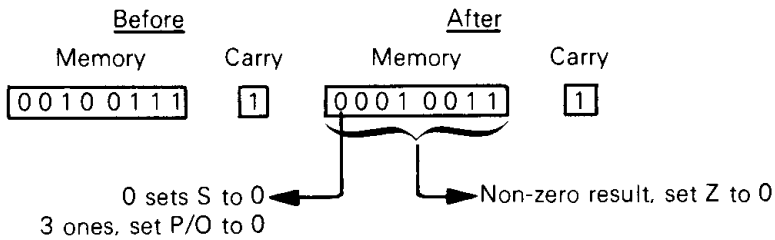


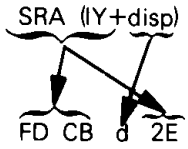
Shift contents of memory location (specified by the sum of the contents of Register IX and the displacement value d) right. Most significant bit is unchanged.

Suppose Register IX contains 3400_{16}, memory location $34AA_{16}$ contains 27_{16}, and Carry=1. After execution of

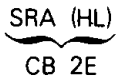
SRA (IX+0AAH)

memory location $34AA_{16}$ will contain 13_{16}, and Carry will be 1.



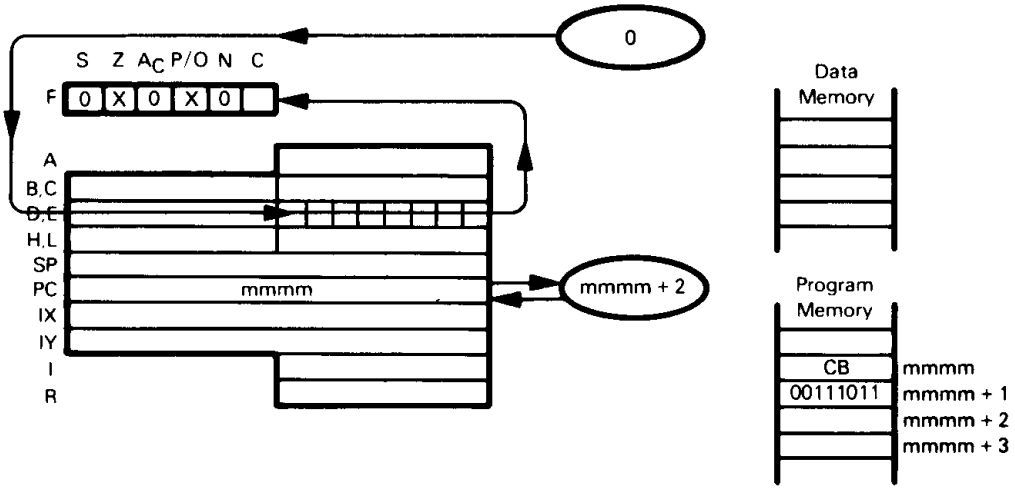


This instruction is identical to SRA (IX+disp), but uses the IY register instead of the IX register.

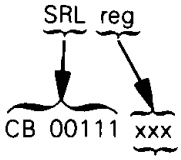


Shift contents of memory location (specified by the contents of the HL register pair) right one bit. Most significant bit is unchanged.

SRL reg — SHIFT CONTENTS OF REGISTER RIGHT LOGICAL



The illustration shows execution of SRL E:



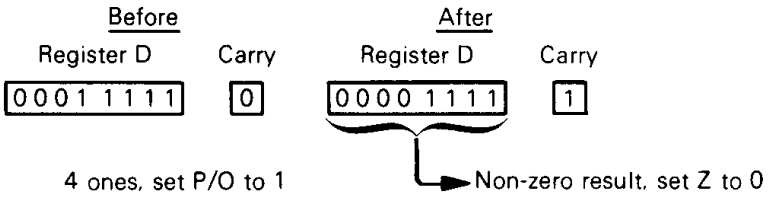
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Shift contents of specified register right one bit. Most significant bit is reset to 0.

Suppose Register D contains $1F_{16}$, and Carry=0. After execution of

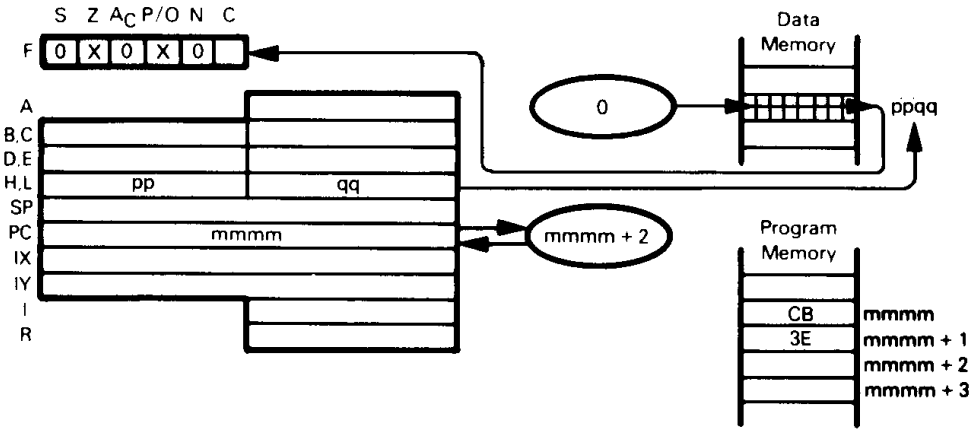
SRL D

Register D will contain $0F_{16}$, and Carry will be 1.



SRL (HL) —
SRL (IX+disp)
SRL (IY+disp)

SHIFT CONTENTS OF MEMORY LOCATION RIGHT LOGICAL



The illustration shows execution of SRL (HL):

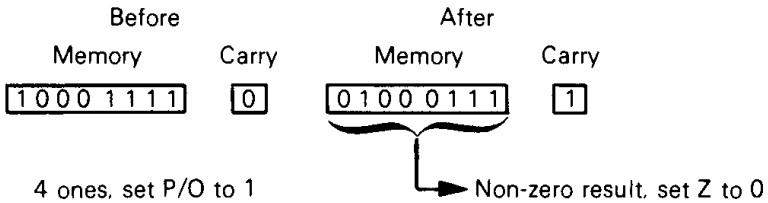
SRL (HL)
CB 3E

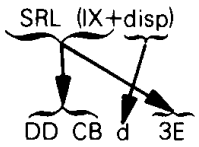
Shift contents of memory location (specified by the contents of the HL register pair) right one bit. Most significant bit is reset to 0.

Suppose the HL register pair contains 2000_{16}, memory location 2000_{16} contains $8F_{16}$, and Carry=0. After execution of

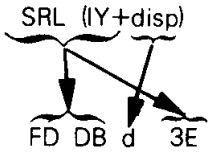
SRL (HL)

memory location 2000_{16} will contain 47_{16}, and Carry will be 1.



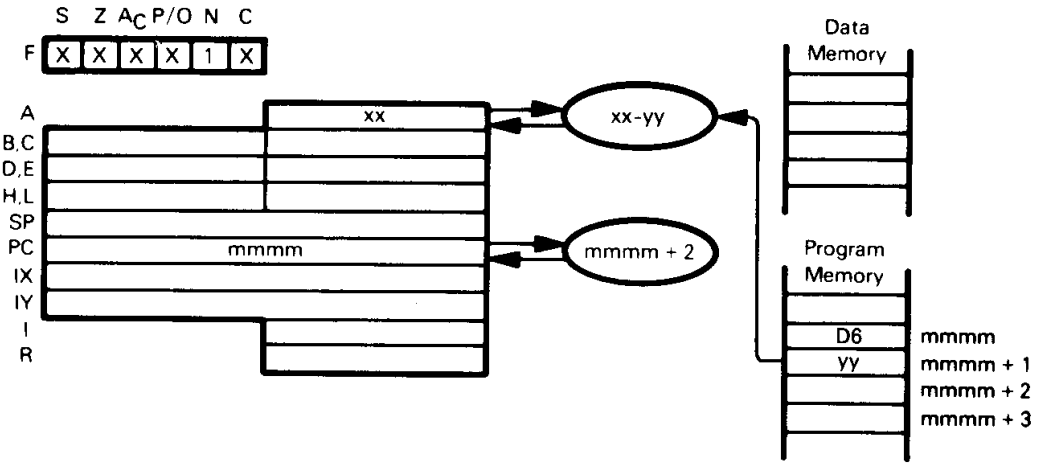


Shift contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) right one bit. Most significant bit is reset to 0.



This instruction is identical to SRL (IX+disp), but uses the IY register instead of the IX register.

SUB data — SUBTRACT IMMEDIATE FROM ACCUMULATOR



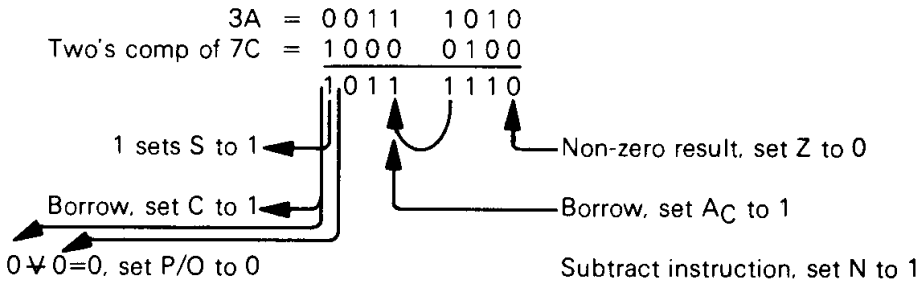
SUB data
D6 yy

Subtract the contents of the second object code byte from the Accumulator.

Suppose xx=$3A_{16}$. After the instruction

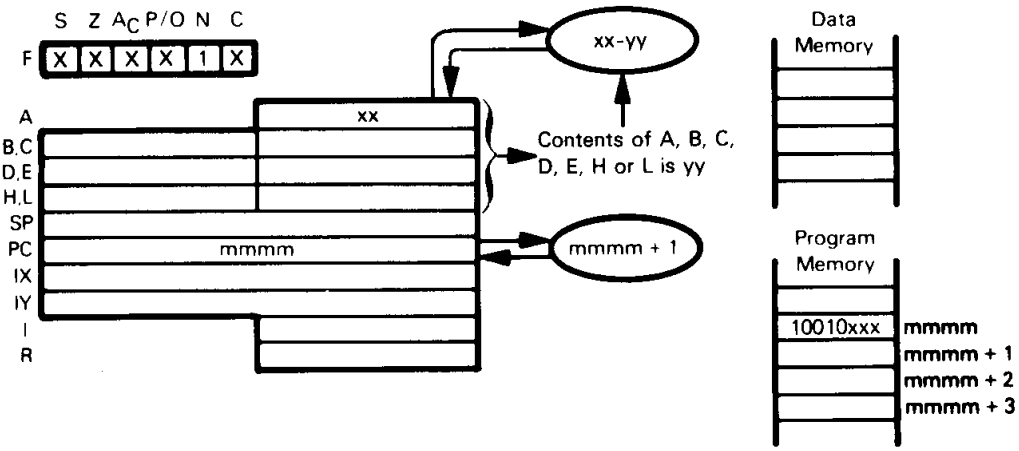
SUB 7CH

has executed, the Accumulator will contain BE_{16}.



Notice that the resulting carry is complemented.

SUB reg — SUBTRACT REGISTER FROM ACCUMULATOR



SUB reg

10010 xxx

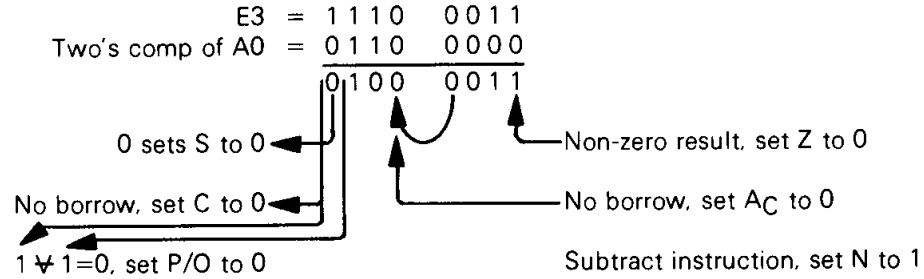
000 for reg=B
001 for reg=C
010 for reg=D
011 for reg=E
100 for reg=H
101 for reg=L
111 for reg=A

Subtract the contents of the specified register from the Accumulator.

Suppose xx=E3 and Register H contains $A0_{16}$. After execution of

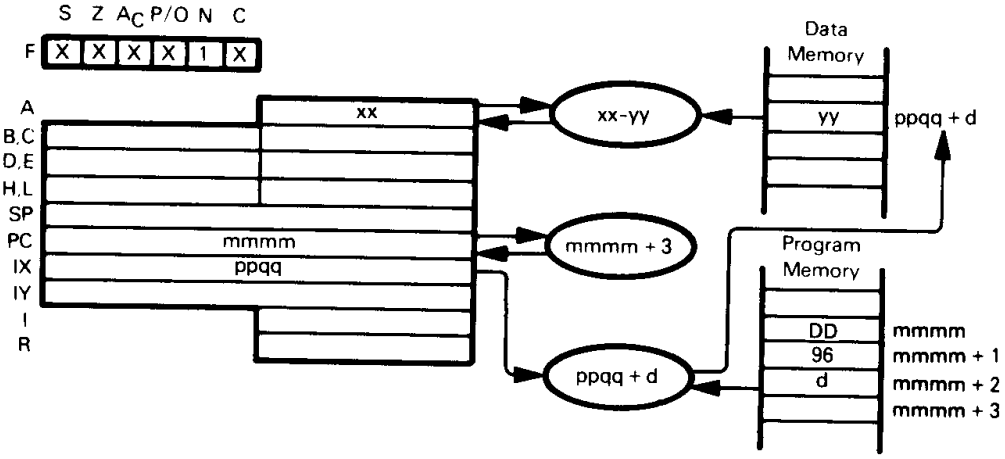
SUB H

the Accumulator will contain 43_{16}.



Notice that the resulting carry is complemented.

SUB (HL) — SUBTRACT MEMORY FROM ACCUMULATOR
SUB (IX+disp)
SUB (IY+disp)



The illustration shows execution of SUB (IX+d):

SUB (IX+disp)

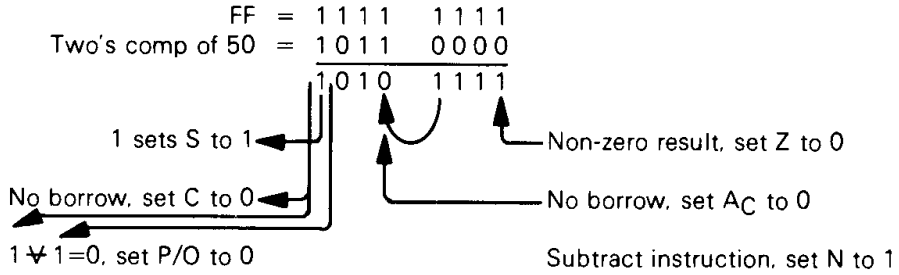
DD 96 d

Subtract contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) from the Accumulator.

Suppose ppqq=4000_{16}, xx=FF_{16}, and memory location $40FF_{16}$ contains 50_{16}. After execution of

SUB (IX+0FFH)

the Accumulator will contain AF_{16}.



Notice that the resulting carry is complemented.

SUB (IY+disp)

FD 96 d

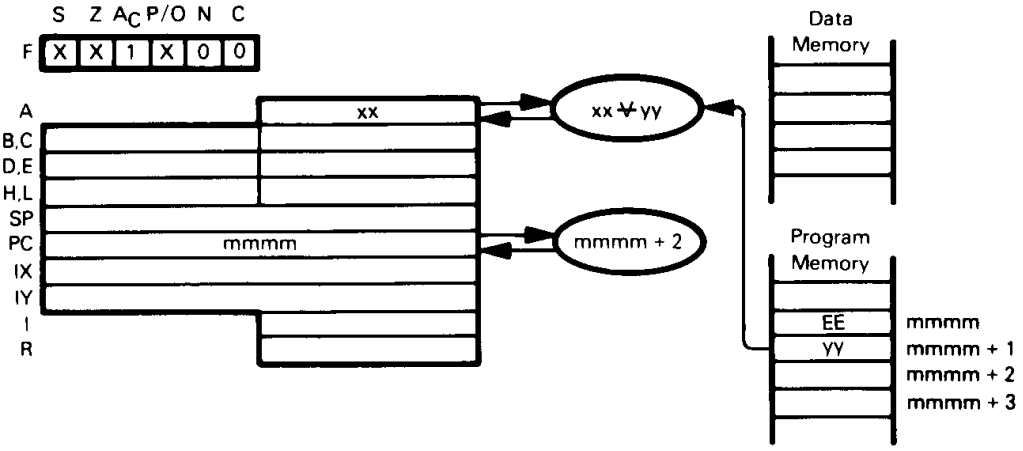
This instruction is identical to SUB (IX+disp), except that it uses the IY register instead of the IX register.

SUB (HL)

96

Subtract contents of memory location (specified by the contents of the HL register pair) from the Accumulator.

XOR data — EXCLUSIVE-OR IMMEDIATE WITH ACCUMULATOR



XOR data

EE yy

Exclusive-OR the contents of the second object code byte with the Accumulator.

Suppose xx=$3A_{16}$. After the instruction

XOR 7CH

has executed, the Accumulator will contain 46_{16}.

```
3A = 0011 1010
7C = 0111 1100
     0100 0110
```

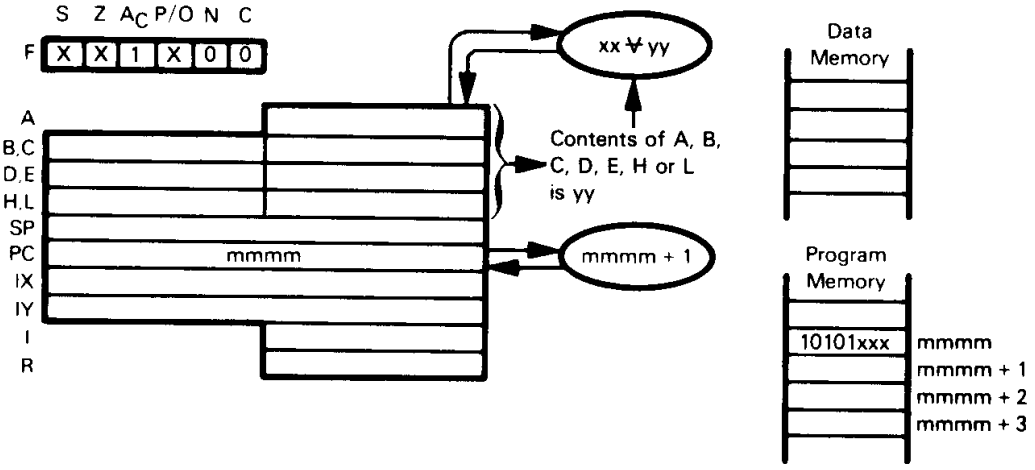
0 sets S to 0

Non-zero result, set Z to 0

Three 1 bits, set P/O to 0

The Exclusive-OR instruction is used to test for changes in bit status.

XOR reg — EXCLUSIVE-OR REGISTER WITH ACCUMULATOR



XOR reg

10101 xxx

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

Exclusive-OR the contents of the specified register with the Accumulator.

Suppose xx=$E3_{16}$ and Register E contains $A0_{16}$. After the instruction

XOR E

has executed, the Accumulator will contain 43_{16}.

```
E3 = 1110 0011
A0 = 1010 0000
     0100 0011
```

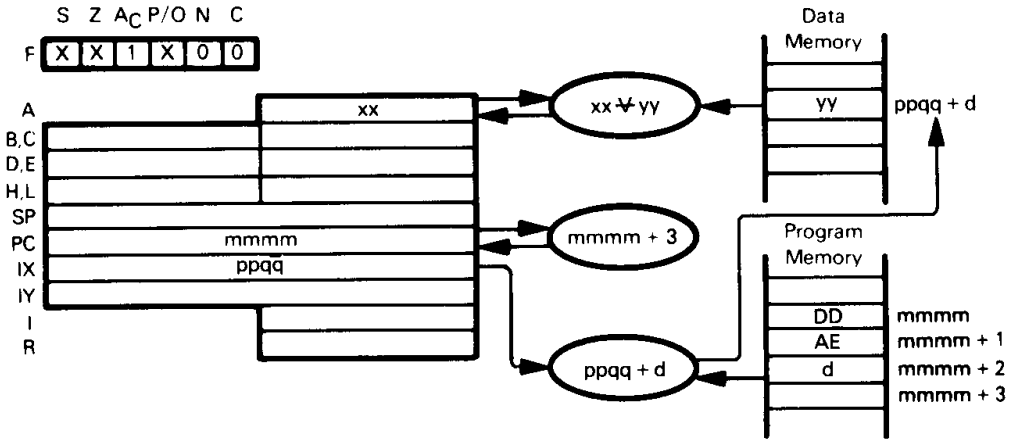
0 sets S to 0

Non-zero result, set Z to 0

Three 1 bits, set P/O to 0

The Exclusive-OR instruction is used to test for changes in bit status.

XOR (HL) — EXCLUSIVE-OR MEMORY WITH ACCUMULATOR
XOR (IX+disp)
XOR (IY+disp)



The illustration shows execution of XOR (IX+disp):

XOR (IX+disp)

DD AE d

Exclusive-OR contents of memory location (specified by the sum of the contents of the IX register and the displacement value d) with the Accumulator.

Suppose xx=$E3_{16}$, ppqq=4500_{16}, and memory location $45FF_{16}$ contains $A0_{16}$. After the instruction

XOR (IX+0FFH)

has executed, the Accumulator will contain 43_{16}.

```
E3 = 1110 0011
A0 = 1010 0000
     0100 0011
```

0 sets S to 0

Non-zero result, set Z to 0

Three 1 bits, set P/O to 0

XOR (IY+disp)

FD AE d

This instruction is identical to XOR (IX+disp), except that it uses the IY register instead of the IX register.

XOR (HL)

AE

Exclusive-OR contents of memory location (specified by the contents of the HL register pair) with the Accumulator.