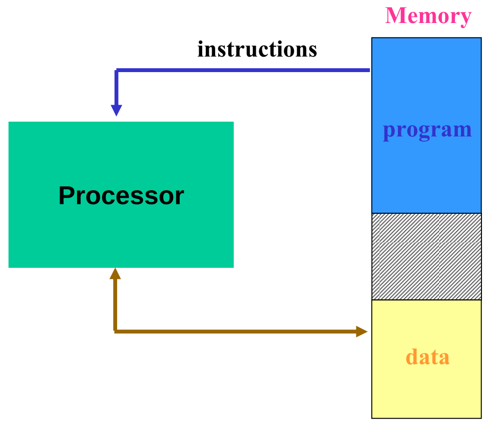
Chapter 3

Z80 Instructions & Assembly Language

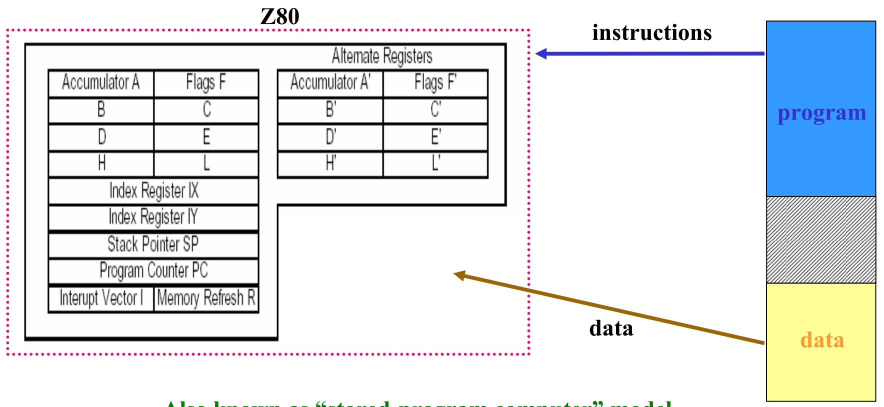
Von Neumann Architecture

- The **von Neumann architecture is a** computer design model that uses a processing unit and a separate storage to hold both instructions and data
- To run a machine, program and data must be loaded into memory



Z80 Programming Model

- **Instructions are loaded into memory in program segment**
- **Data are loaded into memory at data segment**
- **Data are processed by processor according to instructions by loading them into internal temporary storage called "registers"**



Also known as "stored-program computer" model

Register Set (revisited)

- **A** : Accumulator Register
- **F** : Flag register
- Two sets of six general-purpose registers
  - may be used as 8-bit A, F, B, C, D, E, H, L (A', F', B', C', D', E', H', L')
  - or in pairs as 16-bit AF, BC, DE, HL (AF' BC' DE' HL')
- The Alternative registers (A', F', B', C', D', E', H', L') not visible to the programmer but can access via special instructions:
  - EXX (BC)<->(BC') , (DE)<->(DE') , (HL)<->(HL')
  - EX AF, AF ' (AF)<->(AF')

Register Set (revisited)

- 4 (16-bit) registers hold memory address (pointers)
  - index registers (**IX**) and (**IY**) are 16-bit memory pointers
  - 16 bit stack pointer (**SP**)
  - Program counter (**PC**)
- Program counter (**PC**)
  - **PC** points to the next opcode to be fetched from ROM
  - when the µP places an address on the address bus to fetch the byte from memory, it then increments the program counter by one to the next location
- Special purpose registers
  - **I** : Interrupt vector register.
  - **R** : memory Refresh register

Z80 CPU Instructions

- **158 different instructions**
- Including all 78 of the 8080A CPU.
- Instruction groups
  - **Load and Exchange**
    - **Move data to / from memory to registers**
    - **Exchange data in main registers and alternate registers**
  - **Block Transfer and Search**
    - **Move / search data in memory as a block**
  - **Arithmetic and Logical**
    - **Performing arithmetic / logical operations**
  - **Rotate and Shift**
    - **Performing arithmetic / logical rotation / shifting**
  - **Bit Manipulation (Set, Reset, Test)**
  - **Jump, Call, and Return**
  - **Input/Output**
  - **Basic CPU Control**

Z80 Instructions

Load and Exchange

LD – Load instruction

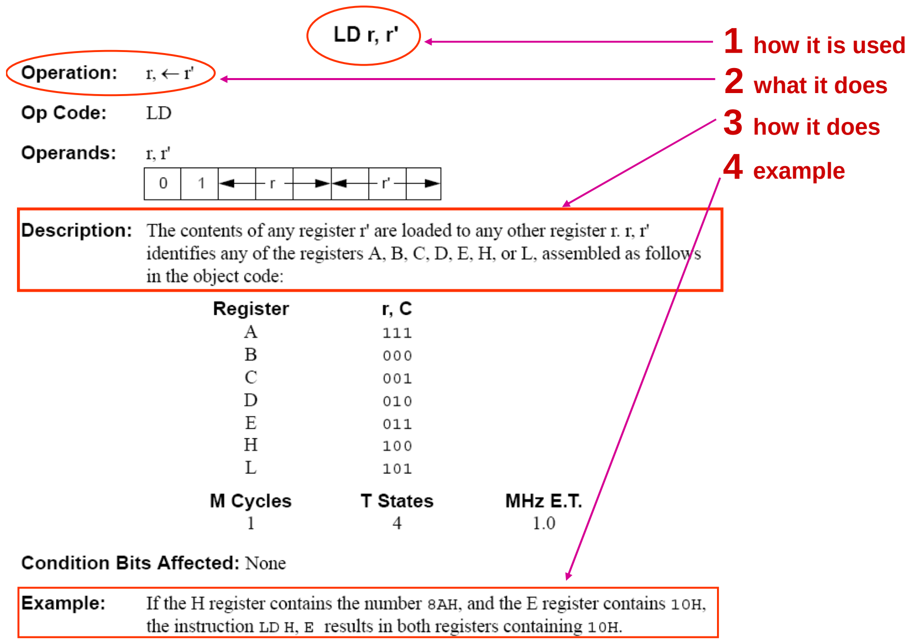
- One of the most useful instruction in the Z80.
- It is used to transfer data between registers, and to and from memory.
- The most simple type of LD instruction is to fill a register with a number:

Example **LD A, 6**
This loads the **A** register with the number 6

- You can transfer this data to other registers

Example **LD H, A**
This transfer content of the **A** register to the **H** register

Instruction Description in Z80 User Manual Document


1 how it is used
2 what it does
3 how it does
4 example

LD r, r'

Operation: r, ← r'

Op Code: LD

Operands: r, r'

0	1	← r →	← r' →

Description: The contents of any register r' are loaded to any other register r. r, r' identifies any of the registers A, B, C, D, E, H, or L, assembled as follows in the object code:

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

M Cycles	T States	MHz E.T.
1	4	1.0

Condition Bits Affected: None

Example: If the H register contains the number 8AH, and the E register contains 10H, the instruction LD H, E results in both registers containing 10H.

Z80 Instructions

LD r,n

Operation: r ← n

Op Code: LD

Operands: r, n

0	0	← r →	1	1	0

← n →

Description: The 8-bit integer n is loaded to any register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

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

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: At execution of `LD E, A5H` the contents of register E are `A5H`.

Z80 Instructions

16-bit Load instructions

LD dd, nn

Operation: dd ← nn

Op Code: LD

Operands: dd, nn

0	0	d	d	0	0	0	1

← n →
← n →

Description: The 2-byte integer nn is loaded to the dd register pair, where dd defines the BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the Op Code is the low order byte.

M Cycles	T States	4 MHz E.T.
2	10 (4, 3, 3)	2.50

Condition Bits Affected: None

Example: At execution of `LD HL, 5000H` the contents of the HL register pair is `5000H`.

Z80 Instructions

8-bit Load instructions with memory reference

LD r, (HL)

Operation: r ← (HL)

Op Code: LD

Operands: r, (HL)

0	1	← r →	1	1	0

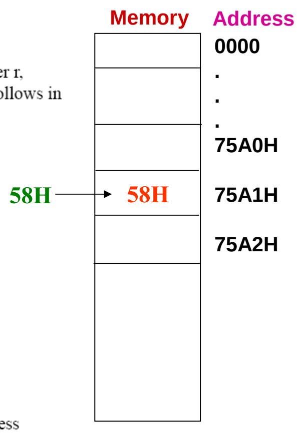
Description: The 8-bit contents of memory location (HL) are loaded to register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

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

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If register pair HL contains the number `75A1H`, and memory address 75A1H contains byte `58H`, the execution of `LD C, (HL)` results in `58H` in register C.



Z80 Instructions

8-bit Load instructions with memory reference

LD (HL), r

Operation: (HL) ← r

Op Code: LD

Operands: (HL), r

0	1	1	1	0	← r →

Description: The contents of register r are loaded to the memory location specified by the contents of the HL register pair. The symbol r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

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

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If the contents of register pair HL specifies memory location `2146H`, and the B register contains byte `29H`, at execution of `LD (HL), B` memory address `2146H` also contains `29H`.

Z80 Instructions

8-bit Load instructions with memory reference

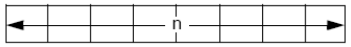
LD (HL), n

Operation: (HL) ← n

Op Code: LD

Operands: (HL), n

0	0	1	1	0	1	1	0	36



Description: Integer n is loaded to the memory address specified by the contents of the HL register pair.

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

Condition Bits Affected: None

Example: If the HL register pair contains `4444H`, the instruction `LD (HL), 28H` results in the memory location `4444H` containing byte `28H`.

Z80 Instructions

8-bit Load instructions with memory reference

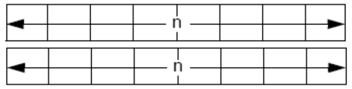
LD A, (nn)

Operation: A ← (nn)

Op Code: LD

Operands: A, (nn)

0	0	1	1	1	0	1	0	3A



Description: The contents of the memory location specified by the operands nn are loaded to the Accumulator. The first n operand after the Op Code is the low order byte of a 2-byte memory address.

M Cycles	T States	4 MHz E.T.
4	13 (4, 3, 3, 3)	3.25

Condition Bits Affected: None

Example: If the contents of nn is number `8832H`, and the content of memory address `8832H` is byte `04H`, at instruction `LD A, (nn)` byte `04H` is in the Accumulator.

Z80 Instructions

8-bit Load instructions with memory reference

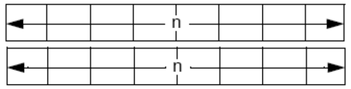
LD (nn), A

Operation: (nn) ← A

Op Code: LD

Operands: (nn), A

0	0	1	1	0	0	1	0	32



Description: The contents of the Accumulator are loaded to the memory address specified by the operand nn. The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
4	13 (4, 3, 3, 3)	3.25

Condition Bits Affected: None

Example: If the contents of the Accumulator are byte `D7H`, at execution of `LD (3141 H),AD7H` results in memory location `3141H`.

Z80 Instructions

16-bit Load instructions with memory reference

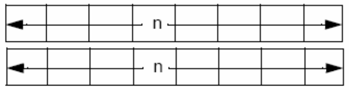
LD HL, (nn)

Operation: H ← (nn+1), L ← (nn)

Op Code: LD

Operands: HL, (nn)

0	0	1	0	1	0	1	0	2A



Description: The contents of memory address (nn) are loaded to the low order portion of register pair HL (register L), and the contents of the next highest memory address (nn+1) are loaded to the high order portion of HL (register H). The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
5	16 (4, 3, 3, 3, 3)	4.00

Condition Bits Affected: None

Example: If address `4545H` contains `37H`, and address `4546H` contains `A1H`, at instruction `LD HL, (4545H)` the HL register pair contains `A137H`.

Z80 Instructions

16-bit Load instructions with memory reference

LD (nn), HL

Operation: (nn+1) ← H, (nn) ← L

Op Code: LD

Operands: (nn), HL

0	0	1	0	0	0	1	0	22
◄			n				►	
◄			n				►	

Description: The contents of the low order portion of register pair HL (register L) are loaded to memory address (nn), and the contents of the high order portion of HL (register H) are loaded to the next highest memory address (nn+1). The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
5	16 (4, 3, 3, 3, 3)	4.00

Condition Bits Affected: None

Example: If the content of register pair HL is `483AH`, at instruction `LD (B2291-1), HL` address `B229H` contains `3AH`, and address `B22AH` contains `48H`.

Z80 Instructions

16-bit Load instructions with memory reference

LD dd, (nn)

Operation: ddh ← (nn+1) ddl ← (nn)

Op Code: LD

Operands: dd, (nn)

1	1	1	0	1	1	0	1	ED
0	1	d	d	1	0	1	1	
◄			n				►	
◄			n				►	

Description: The contents of address (nn) are loaded to the low order portion of register pair dd, and the contents of the next highest memory address (nn+1) are loaded to the high order portion of dd. Register pair dd defines BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the Op Code is the low order byte of (nn).

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If Address `2130H` contains `65H`, and address `2131M` contains `78H`, at instruction `LD BC, (2130H)` the BC register pair contains `7865H`.

Z80 Instructions

16-bit Load instructions with memory reference

LD (nn), dd

Operation: (nn+1) ← ddh, (nn) ← ddl

Op Code: LD

Operands: (nn), dd

1	1	1	0	1	1	0	1	ED
0	1	d	d	0	0	1	1	
◄			n				►	
◄			n				►	

Description: The low order byte of register pair dd is loaded to memory address (nn); the upper byte is loaded to memory address (nn+1). Register pair dd defines either BC, DE, HL, or SP, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the Op Code is the low order byte of a two byte memory address.

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If register pair BC contains the number `4644H`, the instruction `LD (1000H), BC` results in `44H` in memory location `1000H`, and `46H` in memory location `1001H`.

Z80 Assembly Language Programming

Program Template (for C16 assembler)

```
cpu   "z80.tbl"   ←  Directive to indicate instruction table
hof   "INT8"      ←  Directive to indicate number scheme
org   2000h       ←  Directive for relocatable assembler
      .                    ; Comment follows semicolon
      .                    ; Line by line
      .                    ; Your assembly codes start after
      .                    ; org directive
      .
      .
      .
      .
      end
```

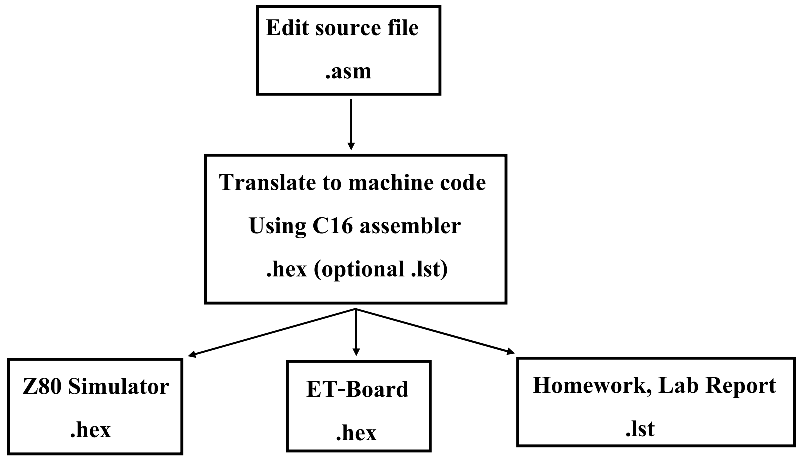
Accumulator and Flag Registers

Program example

```
        cpu   "z80.tbl"
        hof   "INT8"
        org   2000h
              xor   a           ;clear reg A
Loop:         out   (40h), a    ;output to port 40h
              inc   a           ;increment reg A
              jp    loop        ;jump to loop
              halt              ;terminate
              end
```

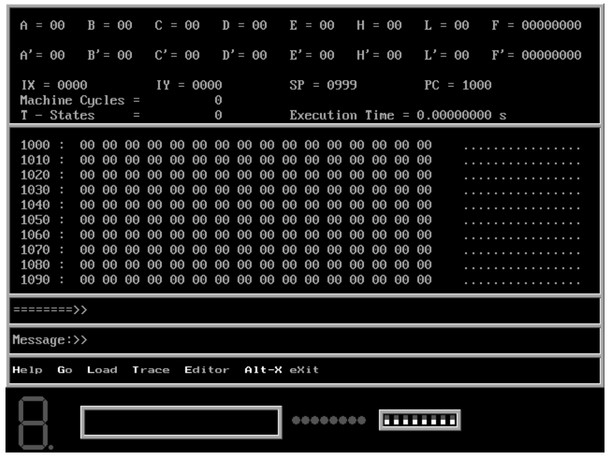
Special-Purpose Registers

Developing Z80 Program in assembly language



Special-Purpose Registers

Demo Session



```
A = 00   B = 00   C = 00   D = 00   E = 00   H = 00   L = 00   F = 00000000

A'= 00   B'= 00   C'= 00   D'= 00   E'= 00   H'= 00   L'= 00   F'= 00000000

IX = 0000          IY = 0000          SP = 0999          PC = 1000
Machine Cycles =            0
T - States     =            0          Execution Time = 0.00000000 s

1000 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1010 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1020 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1030 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1040 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1050 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1060 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1070 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1080 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
1090 :  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................

========>>

Message:>>

Help  Go  Load  Trace  Editor  Alt-X eXit
```

Z80 Instructions

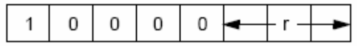
8-bit Arithmetic Group

ADD A, r

Operation: $A \leftarrow A + r$

Op Code: ADD

Operands: A, r



Description: The contents of register r are added to the contents of the Accumulator, and the result is stored in the Accumulator. The symbol r identifies the registers A, B, C, D, E, H, or L,

Condition Bits Affected:

S is set if result is negative; reset otherwise

Z is set if result is zero; reset otherwise

H is set if carry from bit 3; reset otherwise

P/V is set if overflow; reset otherwise

N is reset

C is set if carry from bit 7; reset otherwise

Z80 Instructions

Flag (Revisited)

- **Register F (F’) has special purpose**
- **Each bit of F (F’) has unique meaning**

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

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

Z80 Instructions

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

Carry Flag

The Carry Flag (C) is set or cleared depending on the operation performed.
- If **ADD** instruction generates a Carry, the <u>Carry Flag sets</u>.
- If **SUB** instructions generates a Borrow, the <u>Carry Flag sets</u>.
- Otherwise, the Carry Flag is reset by an ADD/SUB instructions.

- The Carry Flag is also used in Logical rotation instructions.
- Logical operations AND, OR, XOR always reset the Carry Flag.

Z80 Instructions

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

The Add/Subtract Flag (N)

Used to indicate the last executed instruction whether it was ADD or SUB
- For ADD instructions, N is cleared to 0.
- For SUB instructions, N is set to 1.

Z80 Instructions

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

Parity/Overflow Flag (P/V)

This flag is set to a specific state depending on the operation performed.
- For arithmetic operations,

  P/V is set when the result in the Accumulator is greater than the maximum possible number (+127) or is less than the minimum possible number (–128). ➔ **Overflow Condition**
- For logical operations,

  P/V is set to indicate the resulting parity is Even. The number of 1 bits in a byte are counted.
  If the total is Odd, P/V is cleared (P/V = 0).

Z80 Instructions

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

Half Carry Flag

The Half-Carry Flag (H) is set (1) or cleared (0) depending on the Carry and Borrow status between Bits 3 and 4 of an 8-bit arithmetic operation.

H Flag	Add	Subtract
1	A Carry occurs from Bit 3 to Bit 4	A Borrow from Bit 4 occurs
0	No Carry occurs from Bit 3 to Bit 4	No Borrow from Bit 4 occurs

Z80 Instructions

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

Zero Flag

The Zero Flag (Z) is set (1) or cleared (0) if the result generated by the execution of certain instructions is 0.
For 8-bit arithmetic and logical operations,

- the Z flag is set (Z = 1) if the resulting byte in the Accumulator is 0.
- If the byte is not 0, the Z flag is reset (Z = 0).

Z80 Instructions

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

Sign Flag

The Sign Flag (S) stores the state of the most-significant bit of the Accumulator (bit 7).

Z80 Instructions

8-bit Arithmetic Group

ADD A, n

Operation: A ← A + n

Op Code: ADD

Operands: A, n

1	1	0	0	0	1	1	0	C6
←			n				→	

Description: The integer n is added to the contents of the Accumulator, and the results are stored in the Accumulator.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 7; reset otherwise

Example: If the contents of the Accumulator are `23H`, at execution of `ADD A, 33H` the contents of the Accumulator are `56H`.

Z80 Instructions

8-bit Arithmetic Group

SUB s

Operation: A ← A - s

Op Code: SUB

Operands: s

This s operand is any of r, n, (HL), (IX+d), or (IY+d) as defined for the analogous ADD instruction.

Description: The s operand is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if overflow; reset otherwise
N is set
C is set if borrow; reset otherwise

Example: If the Accumulator contents are `29H`, and register D contains `11H`, at execution of `SUB D` the Accumulator contains `18H`.

Z80 Instructions

8-bit Arithmetic Group

AND s

Operation: A ← A ∧ s

Op Code: AND

Operands: s

The s operand is any of r, n, (HL), (IX+d), or (IY+d), as defined for the analogous ADD instructions.

Description: A logical AND operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set
P/V is reset if overflow; reset otherwise
N is reset
C is reset

Example: If the B register contains `7BH` (`0111 1011`), and the Accumulator contains `C3H` (`1100 0011`), at execution of `AND B` the Accumulator contains `43H` (`0100 0011`).

Z80 Instructions

8-bit Arithmetic Group

OR s

Operation: A ← A ∨ s

Op Code: OR

Operands: s

XOR s

Operation: A ← A ⊕ s

Op Code: XOR

Operands: s

Z80 Instructions

8-bit Arithmetic Group

CP s

Operation: A - s

Op Code: CP

Operands: s

The s operand is any of r, n, (HL), (IX+d), or (IY+d), as defined for the analogous ADD instructions.

Description: The contents of the s operand are compared with the contents of the Accumulator. If there is a true compare, the Z flag is set.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if overflow; reset otherwise
N is set
C is set if borrow; reset otherwise

Example: If the Accumulator contains `63H`, the HL register pair contains `6000H`, and memory location `6000H` contains `60H`, the instruction `CP (HL)` results in the PN flag in the F register resetting.

Z80 Instructions

8-bit Arithmetic Group

INC r

Operation: r ← r + 1

Op Code: INC

Operands: r

0	0	←	r →	1	0	0

Description: Register r is incremented and register r identifies any of the registers A, B, C, D, E, H, or L

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if r was 7FH before operation; reset otherwise
N is reset
C is not affected

Example: If the contents of register D are `28H`, at execution of `INC D` the contents of register D are `29H`.

Z80 Instructions

8-bit Arithmetic Group

DEC m

Operation: m ← m- 1

Op Code: DEC

Operands: m

The m operand is any of r, (HL), (IX+d), or (IY+d), as defined for the analogous INC instructions.

Description: The byte specified by the m operand is decremented.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4, reset otherwise
P/V is set if m was 80H before operation; reset otherwise
N is set
C is not affected

Example: If the D register contains byte 2AH, at execution of `DEC D` register D contains 29H.

Z80 Instructions

8-bit Arithmetic Group

DAA

Operation: Decimal Adjust Accumulator

Description: This instruction conditionally adjusts the Accumulator for BCD addition and subtraction operations.

Example: If an addition operation is performed between 15 (BCD) and 27 (BCD), simple decimal arithmetic gives this result:

```
 15
+27
 42
```

But when the binary representations are added in the Accumulator according to standard binary arithmetic.

```
  0001      0101
+ 0010      0111
  0011      1100      = 3C
```

the sum is ambiguous. The `DAA` instruction adjusts this result so that the correct BCD representation is obtained:

```
  0011      1100
+ 0000      0110
  0100      0010      = 42
```

Z80 Instructions

General-Purpose Group

CPL

Operation: $A \leftarrow \overline{A}$

Op Code: CPL

Description: The contents of the Accumulator (register A) are inverted (one’s complement).

Condition Bits Affected:

S is not affected
Z is not affected
H is set
P/V is not affected
N is set
C is not affected

Example: If the contents of the Accumulator are `1011 0100`, at execution of `CPL` the Accumulator contents are `0100 1011`.

Z80 Instructions
Processor Control Group

NOP

Operation: —

Op Code: NOP

Description: The CPU performs no operation during this machine cycle.

Condition Bits Affected: None

Very useful instruction!!!

Z80 Instructions
Processor Control Group

HALT

Operation: —

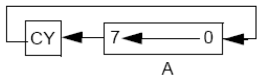
Description: The HALT instruction suspends CPU operation until a subsequent interrupt or reset is received. While in the HALT state, the processor executes NOPs to maintain memory refresh logic.

Condition Bits Affected: None

Z80 Instructions
Rotate and Shift Group

RLCA

Operation:



Description: The contents of the Accumulator (register A) are rotated left 1-bit position. The sign bit (bit 7) is copied to the Carry flag and also to bit 0. Bit 0 is the least-significant bit.

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 7 of Accumulator

Example: If the contents of the Accumulator are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

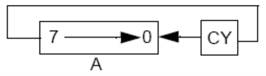
at execution of `RLCA` the contents of the Accumulator and Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

Z80 Instructions
Rotate and shift Group

RLA

Operation:



Description: The contents of the Accumulator (register A) are rotated left 1-bit position through the Carry flag. The previous content of the Carry flag is copied to bit 0. Bit 0 is the least-significant bit.

Condition Bits Affected: Condition Bits Affected

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 7 of Accumulator

Example: If the contents of the Accumulator and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	1	0

at execution of `RLA` the contents of the Accumulator and the Carry flag are

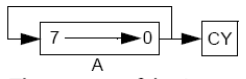
C	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0	1

Z80 Instructions

Rotate and shift Group

RRCA

Operation:



Description: The contents of the Accumulator (register A) are rotated right 1-bit position. Bit 0 is copied to the Carry flag and also to bit 7. Bit 0 is the least-significant bit.

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 0 of Accumulator

Example: If the contents of the Accumulator are

7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1

at execution of `RRCA` the contents of the Accumulator and the Carry flag are

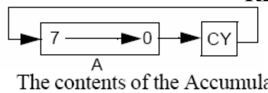
7	6	5	4	3	2	1	0	C
1	0	0	1	1	0	0	0	1

Z80 Instructions

Rotate and shift Group

RRA

Operation:



Description: The contents of the Accumulator (register A) are rotated right 1-bit position through the Carry flag. The previous content of the Carry flag is copied to bit 7. Bit 0 is the least-significant bit.

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 0 of Accumulator

Example: If the contents of the Accumulator and the Carry Flag are

7	6	5	4	3	2	1	0	C
1	1	1	0	0	0	0	1	0

at execution of `RRA` the contents of the Accumulator and the Carry flag are

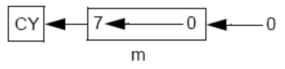
7	6	5	4	3	2	1	0	C
0	1	1	1	0	0	0	0	1

Z80 Instructions

Rotate and shift Group

SLA

Operation:



Description: An arithmetic shift left 1-bit position is performed on the contents of operand m. The content of bit 7 is copied to the Carry flag. Bit 0 is the least-significant bit.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity is even; reset otherwise
N is reset
C is data from bit 7

Example: If the contents of register L are

7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1

at execution of `SLA L` the contents of register L and the Carry flag are

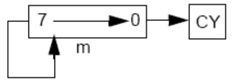
C	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1	0

Z80 Instructions

Rotate and shift Group

SRA

Operation:



Description: An arithmetic shift right 1-bit position is performed on the contents of operand m. The content of bit 0 is copied to the Carry flag and the previous content of bit 7 is unchanged. Bit 0 is the least-significant bit.

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity is even; reset otherwise
N is reset
C is data from bit 0 of source register

Example: If the contents of the Index Register IX are `1000H`, and the contents of memory location `1003H` are

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0

at execution of `SRA (IX+3H)` the contents of memory location 1003H and the Carry flag are

7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	0	0

Z80 Instructions

Unconditional Jump Group

JP nn

Operation: PC ← nn

Description: Operand nn is loaded to register pair PC (Program Counter). The next instruction is fetched from the location designated by the new contents of the PC.

Condition Bits Affected: None

Z80 Instructions

Unconditional Jump Group

JP cc, nn

Operation: IF cc true, PC ← nn

Description:

- If condition cc is true, the instruction loads operand nn to register pair PC (Program Counter)
- If condition cc is false, the PC is incremented as usual, and the program continues with the next sequential instruction.
- Condition cc is programmed as one of eight status that corresponds to condition bits in the Flag Register (register F).

Example

```
restart:  LD A, (HL)
          SUB B
          JP z, restart
          LD D, A
```

```
restart:  LD A, (HL)
          SUB B
          JP nz, restart
          LD D, A
```

```
restart:  LD A, (HL)
          SUB B
          JP c, restart
          LD D, A
```

Z80 Instructions

Unconditional Jump Group

JR e

Operation: PC ← PC + e

Description:

- This instruction provides for unconditional branching to other segments of a program.
- The value of the displacement e is added to the PC and the next instruction is fetched from the location designated by the new contents of the PC.

Example: To jump forward five locations from address 480, the following assembly language statement is used `JR $+5`

The resulting object code and final PC value is shown below:

Location	Instruction
480	18
481	03
482	-
483	-
484	-
485	← PC after jump

Z80 Instructions

Conditional Jump Group

JR C, e

Operation: If C = 0, continue
If C = 1, PC ← PC+ e

JR NC, e

Operation: If C = 1, continue
If C = 0, PC ← PC + e

JR Z, e

Operation: If Z = 0, continue
If Z = 1, PC ← PC +e

JR NZ, e

Operation: If Z = 1, continue
If Z = 0, PC ← pc + e

Z80 Instructions

Conditional Jump Group

DJNZ e

Description:

- The B register is decremented, and if a non zero value remains, the value of the displacement e is added to the PC.
- The next instruction is fetched from the location designated by the new contents of the PC.
- The jump has a range of -126 to +129 bytes.
- If the result of decrementing leaves B with a zero value, the next instruction executed is taken from the location following this instruction.

Example

```
                LD B, 03
loop:           .
                .
                .
                DJNZ loop
                LD A, (HL)
```

Conclusion

Z80 instructions

- **Data transfer**
- **Arithmetic**
- **CPU Control**
- **Shift-Rotate**
- **Conditional/Unconditional Jump**

Q & A

"*Do you program in Assembly ?* "she asked.

"*NOP*" he said.