Introduction to Assembly (8085)

Dr. M. Shiple

Microprocessor Architecture, 2019

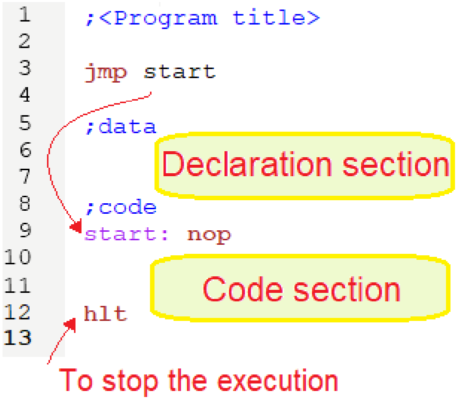
بسم الله الرحمن الرحيم

وقل رب زدني علما

Assembly Sections

Assembly Sections

labels : takes the address of instruction. For example, start= address of "nop" instruction.



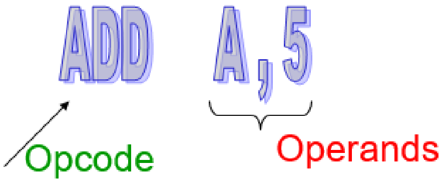
```
;<Program title>

jmp start

;data



;code
start: nop


hlt

```

Addressing Modes in 8085

Register mode

An operand may be in:

- Accumulator A.
- Memory
- Device control registers



Register Addressing Mode :

The data is copied from one register to another.

The data to be operated is available inside registers (the operands are registers).

1. MOV A,B ;*A=B*
2. ADD B ;*A=A+B*
3. INR A ;*A=A+1*

Immediate mode

Immediate addressing mode:

The 8/16-bit data is specified in the instruction itself as one of its operand.

The source operand is always data.

1. MVI B,45 *;B=45*
2. LXI H,3050h *;(load the H-L pair with 3050H immediately)*
3. JMP 0x2030 *;(jump to the operand address immediately)*

Direct Mode

Direct addressing mode

The data is directly copied from the given address to the register.

The data to be operated is available inside a memory location and that memory location is directly specified as an operand.

1. LDA 2050 *;A=[2050h]*
2. LHLD 2030 *;HL=[2030h]*
3. IN 80 *;A=[port B]*

Indirect Mode

Indirect addressing mode

The data is transferred from one register to another by using the address pointed by the register.

The data to be operated is available inside a memory location and that memory location is indirectly specified b a register pair.

1. MOV A, M   *;A=[[HL]]*

Instruction set

Definitions

Instruction:

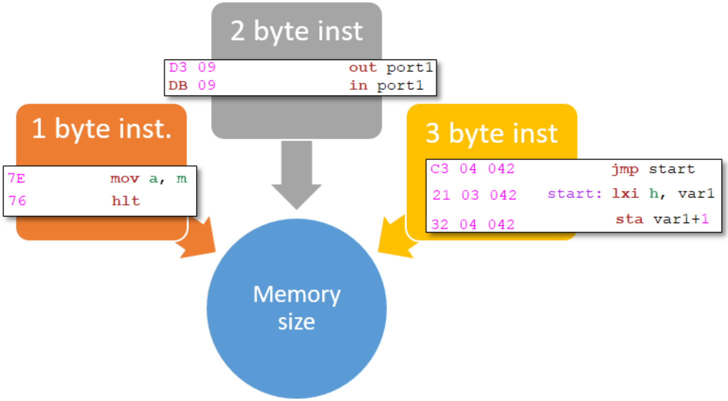
a binary pattern designed inside a microprocessor to perform a specific function.

- 8085 has 246 instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called Op-Code (machine language) or Instruction Byte/mnemonics (Assembly).

Instruction Word Size

Instruction:

a binary pattern designed inside a microprocessor to perform a specific function.

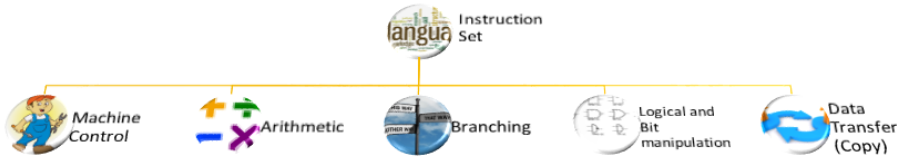


```
D3 09          out port1
DB 09          in port1
```

```
7E             mov a, m
76             hlt
```

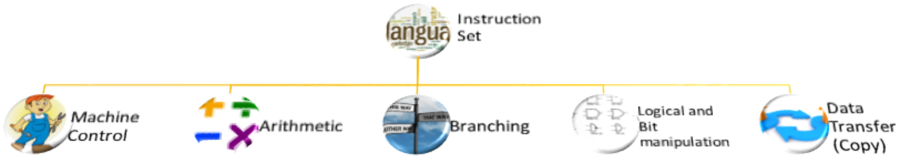
```
C3 04 042                jmp start
21 03 042      start: lxi h, var1
32 04 042                sta var1+1
```

Instruction set



*operation code: (often abbreviated to opcode)

Instruction set: Data transfer (Mov)



Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
MOV	r1,r2	r1=r2	4	1	-----	Move register to register
MOV	M,r	[HL]=r	7	1	-----	Move register to Memory
MOV	r,M	r=[HL]	7	1	-----	Move Memory to register
MVI	r,n	r=n	7	2	-----	Move Immediate
MVI	M,n	[HL]=n	10	2	-----	Move Immediate to Memory

Instruction set: Data transfer (Mov)

Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
LDA	a	A=[a]	13	3	-----	Load Accumulator direct
LDAX	B	A=[BC]	7	1	-----	Load Accumulator indirect
LDAX	D	A=[DE]	7	1	-----	Load Accumulator indirect
LHLD	a	HL=[a]	16	3	-----	Load HL Direct
LXI	B,nn	BC=nn	10	3	-----	Load Immediate BC
LXI	D,nn	DE=nn	10	3	-----	Load Immediate DE
LXI	H,nn	HL=nn	10	3	-----	Load Immediate HL
LXI	SP,nn	SP=nn	10	3	-----	Load Immediate Stack Ptr

STA	a	[a]=A	13	3	-----	Store Accumulator
STAX	B	[BC]=A	7	1	-----	Store Accumulator indirect
STAX	D	[DE]=A	7	1	-----	Store Accumulator indirect

SHLD	a	[a]=HL	16	3	-----	Store HL Direct
SPHL		SP=HL	6	1	-----	Move HL to SP
XCHG		HL<->DE	4	1	-----	Exchange HL with DE
XTHL		[SP]<->HL	16	1	-----	Exchange stack Top with HL

IN	p	A=[p]	10	2	-----	Input
OUT	p	[p]=A	10	2	-----	Output

Instruction set: Data transfer (Mov)

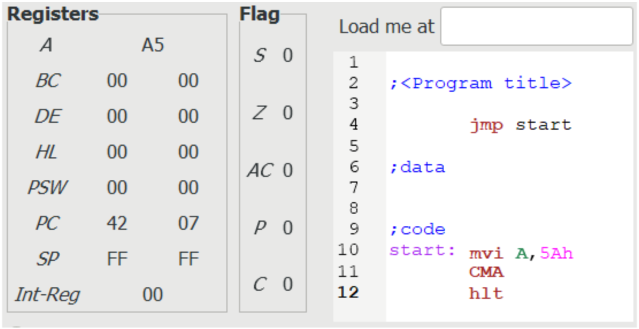
Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
LDA	a	A=[a]	13	3	-----	Load Accumulator direct
LDAX	B	A=[BC]	7	1	-----	Load Accumulator indirect
LDAX	D	A=[DE]	7	1	-----	Load Accumulator indirect
LHLD	a	HL=[a]	16	3	-----	Load HL Direct
LXI	B,nn	BC=nn	10	3	-----	Load Immediate BC
LXI	D,nn	DE=nn	10	3	-----	Load Immediate DE
LXI	H,nn	HL=nn	10	3	-----	Load Immediate HL
LXI	SP,nn	SP=nn	10	3	-----	Load Immediate Stack Ptr
STA	a	[a]=A	13	3	-----	Store Accumulator
STAX	B	[BC]=A	7	1	-----	Store Accumulator indirect
STAX	D	[DE]=A	7	1	-----	Store Accumulator indirect
SHLD	a	[a]=HL	16	3	-----	Store HL Direct
SPHL		SP=HL	6	1	-----	Move HL to SP
XCHG		HL<->DE	4	1	-----	Exchange HL with DE
XTHL		[SP]<->HL	16	1	-----	Exchange stack Top with HL
IN	p	A=[p]	10	2	-----	Input
OUT	p	[p]=A	10	2	-----	Output

Instruction set: Bit manupilation

Instruction Set
Machine Control
Arithmetic
Branching
Logical and Bit manipulation
Data Transfer (Copy)

Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
ANA	r	A=A & r	4	1	SZAP0	AND Accumulator
ANA	M	A=A & [HL]	4	1	SZAP0	AND Accumulator and Memory
ANI	n	A=A & n	7	2	SZ0P0	AND Immediate
CMA		A=~A	4	1	SZAPC	Complement Accumulator
ORA	r	A=A | r	4	1	SZ0P0	Inclusive OR Accumulator
ORA	M	A=A | [HL]	7	1	SZ0P0	Inclusive OR Accumulator
ORI	n	A=A | n	7	2	SZ0P0	Inclusive OR Immediate
XRA	r	A=A⊕r	4	1	SZ0P0	Exclusive OR Accumulator
XRA	M	A=A⊕ [HL]	7	1	SZ0P0	Exclusive OR Accumulator
XRI	n	A=A⊕n	7	2	SZ0P0	Exclusive OR Immediate

Instruction set: Bit manupilation (Examples:CMA)



A	A5	
BC	00	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	07
SP	FF	FF
Int-Reg	00	

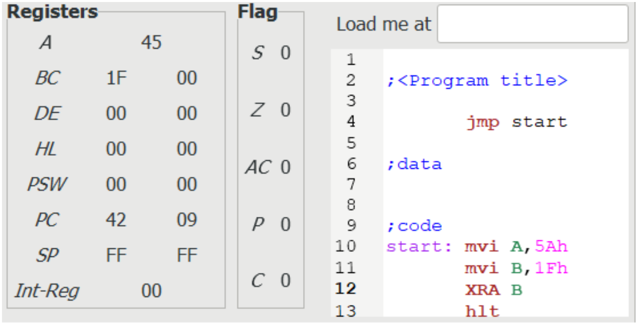
S	0
Z	0
AC	0
P	0
C	0

```
;<Program title>

       jmp start

;data


;code
start: mvi A,5Ah
       CMA
       hlt
```

$A = 5Ah = \quad 01011010$

$\overline{A} = A5h = \quad 10100101$

Instruction set: Bit manupilation (Examples:XOR)



Registers		
A	45	
BC	1F	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	09
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

Load me at

```

;<Program title>

        jmp start

;data


;code
start: mvi A,5Ah
       mvi B,1Fh
       XRA B
       hlt
```

```
A= 5Ah=     01011010
B= 1Fh=     00011111
            ------------
A= A XOR B=45h=  01000101
```

Instruction set: Bit manupilation (Examples:Masking)

```
start: mvi  A,5Bh
       XRA  A
       hlt
```

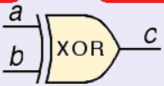
Reset

```
     A= 5bh=    01011011
     A= 5bh=    01011011
             --------------
A xor A=0 = 00000000
```

```
start: mvi  A,5Bh
       XRI  81h
       hlt
```

Alter

```
      A= 5bh=    01011011
         81=     10000001
                --------------
A xor 81=dah= 11011010
```



```
start: mvi  A,5Ah
       ORI  01h
       hlt
```

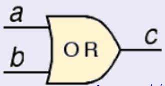
Set

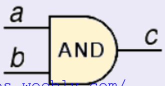
```
      A= 5ah=    01011010
            1=   00000001
               --------------
A OR 1=5bh=  01011011
```

```
start: mvi  A,5Bh
       ani  0feh
       hlt
```

Reset

```
A= 5bh=    01011011
   0Feh=   11111110
          --------------
   5ah = 01011010
```



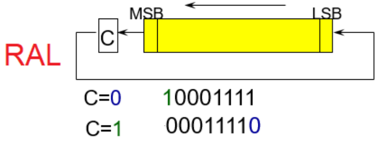


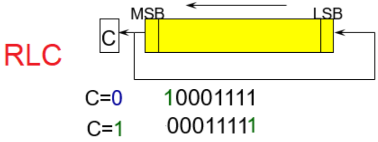
Instruction set: Bit manupilation (Rotation)

Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
RAL		A={CY,A}<-	4	1	SZAPC	Rotate Accumulator Left
RAR		A=->{CY,A}	4	1	SZAPC	Rotate Accumulator Right

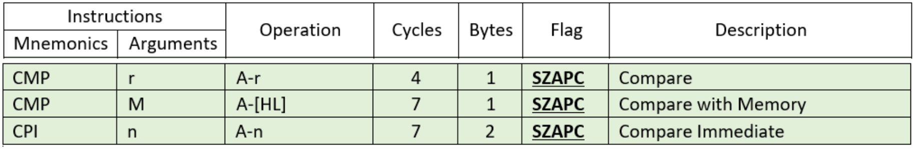
RLC		A=A<-	4	1	SZAPC	Rotate Left Circular
RRC		A=->A	4	1	SZAPC	Rotate Right Circular

Rotate Left





Instruction set: Bit manupilation (Non distructive)



Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
CMP	r	A-r	4	1	**SZAPC**	Compare
CMP	M	A-[HL]	7	1	**SZAPC**	Compare with Memory
CPI	n	A-n	7	2	**SZAPC**	Compare Immediate

CMP is a non-destructive compare

- Sets ZF and CF as appropriate.
- Does not change either of the operands.

Instruction set: Excersize

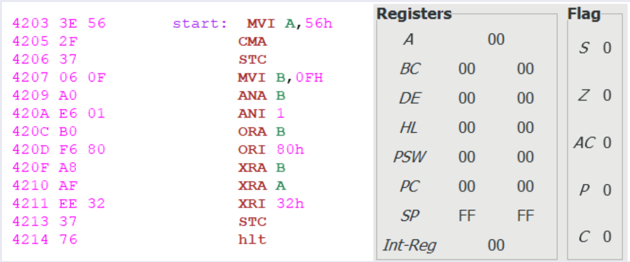
Write programs with effects?

HL= (BC+HL) OR DE (use register pair when necessary), when BC=105h, HL=340h, DE=180h

Change bit patteren as?

Reset bits 0,2 of A and set bits 4,6,7 when A=0A7H

State the



```
4203 3E 56        start:  MVI A,56h
4205 2F                  CMA
4206 37                  STC
4207 06 0F               MVI B,0FH
4209 A0                  ANA B
420A E6 01               ANI 1
420C B0                  ORA B
420D F6 80               ORI 80h
420F A8                  XRA B
4210 AF                  XRA A
4211 EE 32               XRI 32h
4213 37                  STC
4214 76                  hlt
```

Registers

A	00	
BC	00	00
DE	00	00
HL	00	00
PSW	00	00
PC	00	00
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	0
C	0

Memory Size?

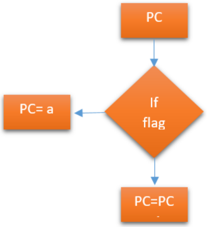
Processing Time?

Instruction set: Branching

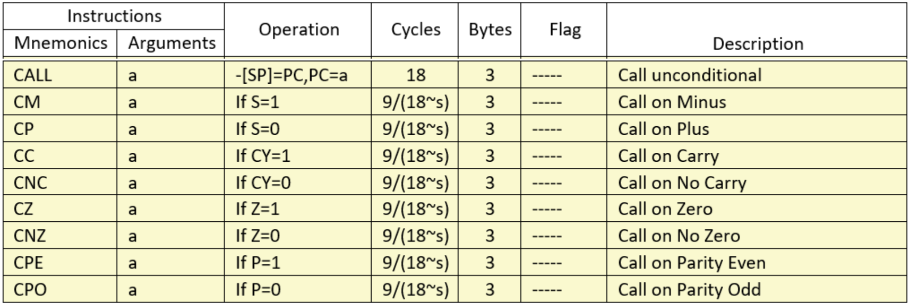
Instruction Set
Machine Control
Arithmetic
Branching
Logical and Bit manipulation
Data Transfer (Copy)

Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
JMP	a	PC=a	7	3	-----	Jump unconditional
JM	a	If S=1	7/(10~s)	3	-----	Jump on Minus
JP	a	If S=0	7/(10~s)	3	-----	Jump on Plus
JC	a	If CY=1	7/(10~s)	3	-----	Jump on Carry
JNC	a	If CY=0	7/(10~s)	3	-----	Jump on No Carry
JZ	a	If Z=1	7/(10~s)	3	-----	Jump on Zero
JNZ	a	If Z=0	7/(10~s)	3	-----	Jump on No Zero
JPE	a	If P=1	7/(10~s)	3	-----	Jump on Parity Even
JPO	a	If P=0	7/(10~s)	3	-----	Jump on Parity Odd

PCHL		PC=[HL]	6	1	-----	Jump HL indirect



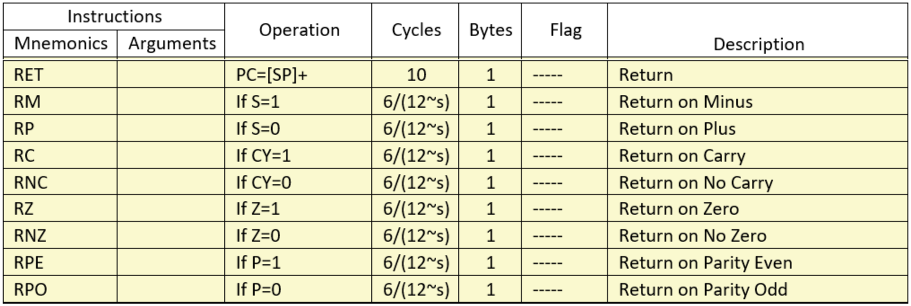
Instruction set: Function call



Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
CALL	a	-[SP]=PC,PC=a	18	3	-----	Call unconditional
CM	a	If S=1	9/(18~s)	3	-----	Call on Minus
CP	a	If S=0	9/(18~s)	3	-----	Call on Plus
CC	a	If CY=1	9/(18~s)	3	-----	Call on Carry
CNC	a	If CY=0	9/(18~s)	3	-----	Call on No Carry
CZ	a	If Z=1	9/(18~s)	3	-----	Call on Zero
CNZ	a	If Z=0	9/(18~s)	3	-----	Call on No Zero
CPE	a	If P=1	9/(18~s)	3	-----	Call on Parity Even
CPO	a	If P=0	9/(18~s)	3	-----	Call on Parity Odd

Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.

Instruction set: Function Return



Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
RET		PC=[SP]+	10	1	-----	Return
RM		If S=1	6/(12~s)	1	-----	Return on Minus
RP		If S=0	6/(12~s)	1	-----	Return on Plus
RC		If CY=1	6/(12~s)	1	-----	Return on Carry
RNC		If CY=0	6/(12~s)	1	-----	Return on No Carry
RZ		If Z=1	6/(12~s)	1	-----	Return on Zero
RNZ		If Z=0	6/(12~s)	1	-----	Return on No Zero
RPE		If P=1	6/(12~s)	1	-----	Return on Parity Even
RPO		If P=0	6/(12~s)	1	-----	Return on Parity Odd

The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

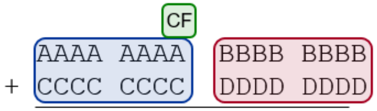
Instruction set: Function Return

Instruction Set
Machine Control
Arithmetic
Branching
Logical and Bit manipulation
Data Transfer (Copy)

Instructions		Operation	Cycles	Bytes	Flag	Description
Mnemonics	Arguments					
ACI	n	A=A+n+CY	7	2	SZAPC	Add with Carry Immediate
ADC	r	A=A+r+CY(21X)	4	1	SZAPC	Add with Carry
ADC	M	A=A+[HL]+CY	7	1	SZAPC	Add with Carry to Memory
ADD	r	A=A+r (20X)	4	1	SZAPC	Add
ADD	M	A=A+[HL]	7	1	SZAPC	Add to Memory
ADI	n	A=A+n	7	2	SZAPC	Add Immediate

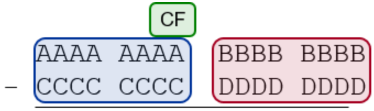
DAD	B	HL=HL+BC	10	1	SZAPC	Double Add BC to HL
DAD	D	HL=HL+DE	10	1	SZAPC	Double Add DE to HL
DAD	H	HL=HL+HL	10	1	SZAPC	Double Add HL to HL
DAD	SP	HL=HL+SP	10	1	SZAPC	Double Add SP to HL

Instruction set: Multi-byte Addition and Subtraction



1. ADD B + D (carry out in CF)
2. ADD A + C **+ CF**

```
    1
    0D  82
 +  00  90
 ---------
    0E  12
```



1. SUB B - D (borrow in CF)
2. SUB A – C **- CF**

```
    1
    0D  82
 -  00  90
 ---------
    0C  F2
```