8085 MICROPROCESSOR LAB MANUAL

http://www.8085projects.info/

MICROPROCESSOR LAB MANUAL

CONTENTS

31.   Program to control the speed and direction of stepper motor

1. Write an ALP to move data block starting at location 'X' to location 'Y' without overlap.

PROGRAM	ALGORITHM
START: LXI H, F000H	; Initialize HL rp* with source addr*.
LXI D, F100H	; Initialize DE rp with destination addr.
MVI C, 04	; move the block length into reg.C
LOOP: MOV A, M	; move the data from memory location as pointed by HL rp to reg. A
	; Store the data from reg. A into the dest*.
STAX D	whose addr. is pointed by DE rp.
	; Increment the src*. addr.
INX H	; Increment dest addr.*
INX D	; Decrement the counter.
DCR C	; If counter is zero terminate the program
JNZ LOOP	Else repeat the program for next data.
	; Terminate the program.
HLT	

NOTE: * denotes

Addr ⟶ Address

rp ⟶ register pair

dest ⟶ Destination

src ⟶ Source

RESULT:

STARING SRC. ADDR.= F000

STARTING DEST. ADDR.= F100

BLOCK LENGTH= 04

BEFORE EXECUTION				AFTER EXECUTION			
Src.addr.	Data	Dest.addr.	Data	Src.addr.	Data	Dest.addr.	Data
F000	01	F100	XX	F000	01	F100	01
F001	02	F101	XX	F001	02	F101	02
F002	03	F102	XX	F002	03	F102	03
F003	04	F103	XX	F003	04	F103	04

SIGNATURE OF STAFF-IN-CHARGE

2. **Write an ALP to move a block starting at location 'X' to location 'Y' with overlap.**

PROGRAM	ALGORITHM
START: LXI H, F109	; Initialize HL rp* with last location of source addr.
LXI D, F10E	; Initialize DE rp with last location of Destination addr*.
MVI C, 0A	; Move the block length into reg.C
LOOP: MOV A, M	; move the data from memory location as Pointed by HL rp to reg. A
STAX D	; Store the data from reg. A into the dest*. whose addr. is pointed by DE rp.
DCX H	; Decrement the src*. addr.
DCX D	; Decrement dest addr.*
DCR C	; Decrement the counter.
JNZ LOOP	; If counter is zero terminate the program else repeat the program for next data.
HLT	; Terminate the program.

RESULT:

STARING SRC. ADDR.= F100

STARTING DEST. ADDR.= F105

BLOCK LENGTH= 0A

BEFORE EXECUTION AFTER EXECTION

Src.addr.	Data	Dest.addr.	Data
F100	00	F105	00
F101	01	F106	01
F102	02	F107	02
F103	03	F108	03
F104	04	F109	04
F105	05	F10A	05
F106	06	F10B	06
F107	07	F10C	07
F108	08	F10D	08
F109	09	F10E	09

SIGNATURE OF STAFF-IN-CHARGE

3. Write an ALP to arrange set of 8 bit numbers starting at location in ASCENDING/DESCENDING order. Display the stored vector in address-data field.

PROGRAM	ALGORITHM
START: MVI B, (N-1)	; Load register B with (N-1), No. of passes
MVI C, (N-1)	; Load register C with (N-1) comparisons
NXTPASS: LXI H, F100	; Move starting address of the Data into HL rp.
LOOP: MOV A, M	; Move data to register A
INX H	; Increment the pointer.
CMP M	; Compare with the next element
JC NOSWAP	; If carry jump to NOSWAP, else interchange the data
	; Interchange two data
SWAP: MOV D, M	; Consecutive elements
MOV M,A	; Decrement the memory location
DCX H	
MOV M, D	; Increment register pair.
INX H	; Decrement register C (No. of comparisons)
NOSWAP: DCR C	; If not zero jump to loop, else
JNZ LOOP	; decrement register B (No. of passes)
DCR B	; The data in register B is moved to register C
MOV C, B	; If not zero, jump to next pass
JNZ NXTPASS	; Initialize HL pair with address of the list
DISPLAY: LXI H, F100	(ascending/descending)
	; Initialize counter.
MVI C, N	; Load the element in register A.
NEXT: MOV A, M	; Store the content of register A in FFF1.
STA FFF1	; Push addr, of the data into the Stack
PUSH H	; Push the content into the Stack.
PUSH B	; Display the data on data sheet.
CALL UPDDT	; Wait for some time.
CALL DELAY	; Pop the counter
POP B	; Pop the addr. of the list.
POP H	; Increment pointer
INX H	; Decrement counter
DCR C	; If Counter=0 terminate the program, else take
JNZ NEXT	next data for comparison.
	; Terminate the program.

SIGNATURE OF STAFF-IN-CHARGE

HLT	; Load reg. pair BC with the count for 0.5s delay.
DELAY: LXI B, F424	; Decrement count
WAIT: DCX B	; Check if count is zero
MOV A, C	; Clear the Accumulator contents
ORA B	; If count is not zero jump to WAIT, else return to
JNZ WAIT	main program
RET	

RESULT SHEET:

N = 07

AFTER EXECUTION:

Src.addr.	Data	Data Field
F100	30	04
F101	12	12
F102	A3	23
F103	04	30
F104	46	46
F105	71	71
F106	23	A3

NOTE: "For Descending order Change JC to JNC "

4. Write an ALP to add N one byte binary numbers stored from location X+1' where N is stored at location X. Store the result in location Y & Y+1. Display the result in address field.

PROGRAM	ALGORITHM
START: LXI H, F100	STEP 1: Initialize the starting address of the Data block
MOV C, M	
SUB A	STEP 2: Initialize the count.
MOV B,A	STEP 3: Initialize the initial sum to zero.
LOOP: INX H	STEP 4: Add the data bytes one by one.
ADD M	STEP 5: Increment the memory pointer one by
JNC LOOP1	One for one each addition.
INR B	STEP 6: Decrement the count by one for each
LOOP1: DCR C	Condition. Check for zero condition.
JNZ LOOP	STEP 7: If the count is not zero, repeat step 4 to
MOV H,B	6.
MOV L,A	STEP 8: If the count is zero halt the processor.
SHLD F200	
CALL UPDAD	
HLT	

NOTE: Store the program starting from F000H.

Store the count at location F100H.

Store the data starting from F101H.

Execute the program. The result will be displayed in the display field.

The result is also in location F200H & F201H

• Address for UPDAD is 06BFH

RESULT:

LENGTH OF BLOCK = 04

STORE AT LOCATION F100

BEFORE EXECUTION:	
Data Addr.	Data
F101	01
F102	02
F103	03
F104	04

AFTER EXECUTION:	
Result Addr.	Data
F200	0A
F201	00

ADDRESS FIELD:

0	0	0	A

SIGNATURE OF STAFF-IN-CHARGE

http://www.8085projects.info/

5. Write an ALP to add two multi-byte binary number starting at locations 'X' and 'Y' and store the result at location 'Z'.

PROGRAM	ALGORITHM
START: LXI H, F100	STEP 1: Initialize the starting address of the
LXI B, F200	two multi-byte numbers at result
LXI D, F300	Location (F100, F200 & F300).
STC	STEP 2: Reset the carry flag
CMC	STEP: 3: Add the multi-byte numbers byte by
MVI A, 04	byte by considering the carry
LOOP: STA F400	Condition.
LDAX B	STEP 4: Check the byte count logically
ADC M	without affecting the status.
STAX D	STEP 5: If the byte count is not zero, repeat
INX H	the steps 3 & 4
INX B	STEP 6: If the byte count is zero, count the
INX D	final carry, mean time store the
LDA F400	result.
DCR A	STEP 7: Halt the processor.
JNZ LOOP	
MVI A,00	
RAL	
STAX D	
HLT	

NOTE:

- Store the program starting from F000H.
- Store the first 32 bit number d ata starting from F100H.
- Store the second 32 bit number data starting from F200H.
- Store the result starting from F300.
- Execute the program. The result will be displayed in the display field.
- The result is also in location F200H & F201H

SIGNATURE OF STAFF-IN-CHARGE

RESULT:

BEFORE EXECUTION:					
Addr. Of 1st multibyte number	Data	Addr. Of 2nd multibyte number	Data	Addr. Of the result	Data
F100	01	F200	AA	F300	XX
F101	02	F201	BB	F301	XX
F102	03	F202	CC	F302	XX
F103	04	F203	DD	F303	XX

AFTER EXECUTION:					
Addr. Of 1st multibyte number	Data	Addr. Of 2nd multibyte number	Data	Addr. Of the result	Data
F100	01	F200	AA	F300	AB
F101	02	F201	BB	F301	BD
F102	03	F202	CC	F302	CF
F103	04	F203	DD	F303	E1

http://www.8085projects.info/

6. Write an ALP to add N 2-digit BCD numbers store the result at location X and store the same in address/data field.

PROGRAM	ALGORITHM
START: LXI H,F100H	STEP1. Initialize the starting address of the data block where the two digits BCD numbers are stored.
MOV C,M	
SUB A	
MOV B,A	STEP2. Initialize the counter.
RPT: INX H	STEP3. Initialize the sum 00H.
ADD M	STEP4. Add the data bytes one by one mean time convert the sum into the decimal value
DAA	
JNC NOF	
PUSH PSW	STEP5. Decrement the counter one by one and check for the zero condition.
MOV A,B	
ADI 01H	STEP6. If the counter is not zero repeat step 4 to 6
DAA	
MOV B,A	STEP7. If the counter is zero display the result
POP PSW	
NOF: DCR C	STEP 8.Halt the processor.
JNZ RPT	
MOV L,A	
MOV H,B	
SHLD F200H	
CALL UPDAD	
HLT	

NOTE:

- Store the program starting from F000h.
- Store the counter at F100H
- Store the 2-digit BCD numbers starting at the location F101H
- Execute the program.
- Result will be displayed in the display field and the same will stored at location F200H and F201H

ADDRESS FOR UPDAD is 06BFh.

RESULT:

BEFORE EXECUTION

Address	Data	Address for result	Data
F100	04(cnt)	F200	X X
F101	11	F201	X X
F102	22		
F103	33		
F104	44		

AFTER EXECUTION

Address	Data	Address for result	Data
F100	04(cnt)	F200	10
F101	11	F201	01
F102	22		
F103	33		
F104	44		

ADDRESS FIELD :

0	1	1	0

7. Write an ALP to subtract a 16-bit binary number stored at location 'X' & 'X+1' from another 16-bit number at location 'Y' & 'Y+1''. Display the result in address field.

PROGRAM	ALGORITHM

START : LHLD F100
XCHG
LHLD F102
MOV A,L
SUB E
MOV L,A
MOV A,H
SBB D
MOV H,A
SHLD F104
CALL UPDAD
HLT
	STEP 1: Load the two 16-bit BCD
	numbers from respective
	locations
	STEP 2: Using suitable instructions in
	Decimal mode finds the numbers.
	STEP 3: Store the result in the address
	field.
	STEP 4: Halt the processor.

NOTE:

- Address for UPDAD is 06BFH
- Store the program starting from F000H
- Store the 16-bit data's at LOC F100H, F101H and F102H, F103H
- Respectively.
- Executive the program.
- Result will be displayed in address field.

RESULT:

BEFORE EXECUTION			
Addr. For 1st data	Data	Addr. For 2nd data	Data
F100H	3C	F102H	C3
F101H	3C	F103H	C3

AFTER EXECUTION

Address field:

8	7	8	7

CY=0. Therefore answer is positive.

8. Write an ALP to check the fourth bit of a byte stored at location 'X' is a '0' or '1'. If '0' store '00H' else store 'FFH' at location 'Y'. Display the same in ADDRESS/DATA field.

PROGRAM	ALGORITHM
START: LDA F100H	STEP 1: Load the data byte from LOC 'X'
ANI 10H	to ACC.
JZ STR0	STEP 2: Check the fourth bit using suitable
MVI A,FFH	Instruction.
STR0: STA F101H	STEP 3: Check the respective flag
CALL UPDDT:	Condition.
HLT	STEP 4: Store result/ decision at LOC 'Y'.
	STEP 5: Display the same in DATA field.
	STEP 6: Halt the processor.

NOTE:

- Store the data type in LOC 'X' (F100H).
- Execute the program. Result will be displayed in DATA field and also stored
- in LOC 'Y' (F101H)

   # for UPDAD: is 06BFH.

   # for UPDAD: is 06D6H

RESULT:

BEFORE EXECUTION	
Addr. of LOC X	DATA
F100H	07
F100H	90

AFTER EXECUTION		
Addr. of LOC Y	DATA	Data in Data field
F101H	00	00
F101H	FF	FF

9. **Write an ALP to generate resultant byte whose 7th bit is given by:**

$$A_7 = A_2 \oplus A_5 \oplus A_6$$

PROGRAM	ALGORITHM
START: LDA F100H	STEP 1: Load the data byte from LOC 'X'
MOV B,A	to ACC.
RRC	STEP 2: Check the fourth bit using suitable
RRC	Instruction.
ANI 01H	STEP 3: Check the respective flag
MOV C,A	Condition.
MOV A,B	STEP 4: Store result/ decision at LOC 'Y'.
RLC	STEP 5: Display the same in DATA field.
RLC	STEP 6: Halt the processor.
ANI 01H	
MOV D,A	
MOV A,B	
RLC	
RLC	
RLC	
ANI 01H	
XRA D	
XRA C	
RRC	
STA F300H	
CALL UPDDT:	
HLT	

NOTE:

- Store the program from F000h., Store the data at F100h., Execute the program.
- Result will be stored in the memory location F101h, and the same will be
- displayed in the data field.
- If A7 bit is '1', DATA field will always display '80H'
- If A7 bit is '0', DATA field will always display '00H'

Address for UPDAD: is 06BFH.

RESULT:

BEFORE EXECUTION	
Addr. of LOC 'X'	DATA
F100H	17H

AFTER EXECUTION		
Addr. of LOC 'Y'	DATA	Data in Data field
F300H	80H	80H

10.Write a ALP to find the product of two unsigned binary numbers stored at location 'X' and 'X+1' using successive addition and display the result in address field.

PROGRAM	ALGORITHM

START: LDA F100H
 MOV E,A
 MVI D,00H
 LDA F101H
 MOV C,A
 LXIH, 0000H
 MOV A,E
 CPI 00H
 JZ DISPLY
 MOV A,C
 CPI 00H
 JZ DISPLY
 LOC: DAD D
 DCR C
 JNZ LOC
 DISPLY: SHLD FE73H
 CALL UPDAD
 HLT
	STEP 1: Load the data from locations 'X'
	 and 'X+1'.
	 STEP 2: Find the product using successive
	 Addition method.
	 STEP 3: Display the result in ADDRESS
	 field
	 STEP 4: Halt the processor.

NOTE:

- Store the program starting from F000H
- Store the data at LOC F100H and F101H
- Executive the program.
- Result will be displayed in address field.

    # Address for UPPAD is 06BFH.

RESULT:

BEFORE EXECUTION			
Addr. For 1st multiplier	Data	Addr. For 2nd multiplier	Data
F100H	06	F101H	05

AFTER EXECUTION:

Address Field:

0	0	1	E

11. Write an ALP to find the product of two unsigned binary numbers stored at location 'X' and 'X+1' by shift left and add method and display the product in address field.

PROGRAM	ALGORITHM

START: LXI H, F100H
          MOV E,M
          MVI D,00H
          INX H
          MOV A,M
          LXI H,0000H
          MVI B,08H
MULT: DAD H
          RAL
          JNC SKIP
          DAD D
SKIP: DCR B
          JNZ MULT
          SHLD FE73
          CALL UPDAD
          HLT
	STEP 1: Load the data's from locations 'X'
	          and 'X+1'.
	STEP 2: Find the product using Shift Left
	          method.
	STEP 3: Display the result in ADDRESS
	          field
	STEP 4: Halt the processor.

NOTE:

- Store the program starting from F000H
- Store the data at LOC F100H and F101H
- Executive the program.
- Result will be displayed in address field.

   # Address for UPDAD is 06BFH.

RESULT:

BEFORE EXECUTION			
Addr. For 1st multiplier	Data	Addr. For 2nd multiplier	Data
F100H	06	F101H	05

AFTER EXECUTION:

Address Field:

0	0	1	E

12. Write an ALP to divide a 16 bit number at location X and X+1 by an 8 bit number at loc Y. Display the Quotient in address field and remainder in data filed.

PROGRAM	ALGORITHM

START: LHLD Fl00H
 	
XCHG
 	
LDA Fl02H
 	
MOV L, A
 	
MVI H,00H
 	
LXI B, 0000H
 	
RPTS: MOV A,E
 	
SUB L
 	
MOV E,A
 	
MOV A,D
 	
SBB H
 	
MOV D,A
 	
INX B
 	
JNC RPTS
 	
DCX B
 	
DAD D
 	
MOV A, L
 	
MOV L,C
 	
MOV H,B
 	
PUSH H
 	
CALL UPDDT
 	
POP H
 	
CALL UPDAD
 	
HLT
	STEP 1; Load the 16 bit number from the
	            specified memory location that is
	            F100H and F101H
	STEP2: Load 8 bit number from
	            specified memory location
	            (F102H)
	STEP3: Using successive subtraction
	            principle find the quotient and
	            remainder
	STEP4: Display the result in Data field
	            and address field.
	STEP 5: Terminate the program.

NOTE:

- o Store the program starting from F000H
- o Store the 16 bit number at F100H and F101H(Numerator)
- o Store the 8 bit number at F102H(denominator)r)
- o Execute the program, result will be displayed in the display field
     & addresses for UPDDT: is 06D6H

RESULT:

SIGNATURE OF STAFF-IN-CHARGE

BEFORE EXECUTION:	
Data Addr.	Data(Numerator)
F100	0A
F101	00

BEFORE EXECUTION	
Data Addr	Data(Denominator)
F102	05

AFTER EXECUTION:

Address field:

0	0	0	2

Data field:

0	0

http://www.8085projects.info/

MICROPROCESSOR LAB MANUAL

13. Write an ALP to implement a counter to count from '00 – 99' (UPCOUNTER) in BCD. Use a subroutine to generate a delay of one second between the counts.

PROGRAM	ALGORITHM
START: MVI A,00H	STEP 1: Initiate the minimum number in
RPTD: PUSH PSW	accumulator
CALL UPDDT	STEP 2: Display in the DATA field
CALL DELAY	STEP 3: Add 01 to the present value
POP PSW	displayed
ADI 01H	STEP 4: Use decimal conversion
DAA	Instruction.
JMP RPTD	STEP 5: Repeat the steps 2-4.
HLT	STEP 6: Provide proper display between
	Each display.
DELAY: LXI B, F424H	STEP 7: Terminating Point.
WAIT: DCX B	
MOV A,C	
ORA B	
JNZ WAIT	
RET	

NOTE:
- Store the program starting from F000H.
- Execute the program; the result will be displayed in the DATA field.
  # Address for UPDAD: is 06BFH.

RESULT:
It counts from 00 to 99 with the given delay in DATA field.

0	0
0	1

.   .

9	8
9	9

SIGNATURE OF STAFF-IN-CHARGE

MICROPROCESSOR LAB MANUAL

14. Write a program for decimal down counter 99-00
(DOWN COUNTER) and display the count in DATA/ADDRESS field

PROGRAM	ALGORITHM
START: MVI A,99H	Step 1: Initiate the maximum count Acc
DSPLY: PUSH PSW	Step 2: Display the present counting data
CALL UPDDT	field
CALL DELAY	Step3: Provide proper delay using
POP PSW	subroutine techniques
ADI 99H	Step4: Decrement the counting decimal
DAA	mode(Use 10's complement
JMP DSPLY	method)
HLT	Step5: Repeat steps 2, 3, and 4
DELAY: LXI D, FFFFH	
REPEAT: DCX D	Step6: Terminate the program
MOV A, E	
ORA D	
JNZ REPEAT	
RET	

NOTE:
o   Store the program starting from LOC F000H
o   Execute the program
o   Observe the result on the data field
o   # Address for UPDDT: 06D6H

RESULT:

It counts from 99 to 00 with the given delay on data field.

DATA FIELD

9	9
9	8

0	1
0	0

SIGNATURE OF STAFF-IN-CHARGE

MICROPROCESSOR LAB MANUAL

15. Write an ALP to implement a counter to count from '00 – FF' (UPCOUNTER) in HEX. Use a subroutine to generate a delay of one second between the counts.

PROGRAM	ALGORITHM
START: MVI A,00	STEP 1: Initiate the minimum number in
RPTD: PUSH PSW	accumulator
CALL UPDDT	STEP 2: Display in the DATA field
CALL DELAY	STEP 3: Add 01 to the present value
POP PSW	displayed
ADI 01 H	STEP 4: Repeat the steps 2-4.
JMP RPTD	STEP 5: Provide proper display between
HLT	Each display.
	STEP 6: Terminating Point.
DELAY: LXI B, F424H	
WAIT: DCX B	
MOV A,C	
ORA B	
JNZ WAIT	
RET	

NOTE:

- Store the program starting from F000H.
- Execute the program; the result will be displayed in the DATA field.

  # Address for UPDAD: is 06BFH.

RESULT:

It counts from 00 to FF with the given delay in DATA field.

0	0
0	1

F	E
F	F

SIGNATURE OF STAFF-IN-CHARGE

16. Write an ALP to implement a counter to count from 'FF – 00' (DOWN COUNTER) in HEX. Use a subroutine to generate a delay of one second between the counts.

PROGRAM	ALGORITHM
START: MVI A,FFH	STEP 1: Initiate the minimum number in
RPTD: PUSH PSW	accumulator
CALL UPDDT	STEP 2: Display in the DATA field
CALL DELAY	STEP 3: Subtract 01 to the present value
POP PSW	Displayed.
SBI 01H	STEP 4: Repeat the steps 2-4.
JMP RPTD	STEP 5: Provide proper display between
HLT	Each display.
	STEP 6: Terminating Point.
DELAY: LXI B, F424H	
WAIT: DCX B	
MOV A,C	
ORA B	
JNZ WAIT	
RET	

NOTE:

- Store the program starting from F000h.
- Execute the program; the result will be displayed in the DATA field.

    # Address for UPDAD: is 06BFH.

RESULT:

It counts from FF to 00 with the given delay in DATA field.

F	F
F	E

.   .

0	1
0	0

SIGNATURE OF STAFF-IN-CHARGE

MICROPROCESSOR LAB MANUAL

17. Write an ALP to check whether the 8-bit numbers stored at location 'X' belongs to '2 out of 5 code' or not. Display '00' if invalid and 'FF' if valid, on the data field.

PROGRAM	ALGORITHM
START: LDA F100H	STEP 1: Load the data to be checked for Code status.
ANI E0H	
JNZ DISPLAY	STEP 2: Check the 3 MSB's are zero or not.
LDA F100H	
JPO DISPLAY	STEP 3: If zero proceed further else Display error and halt the processor.
MVI C,05H	
MVI B,00H	
RPT1: RRC	STEP 4: Count the number of ones in the 5-bits
JNC NEXT1	
INR B	STEP 5: Check the count with '02'
NEXT1: DCR C	
JNZ RPT1	STEP 6: If count=02, display 'FFH' on the Data filed and halt the processor.
MOV A,B	
CPI 02H	
MVI A,FFH	STEP 7: If count ≠ 02 display '00' on the Data field and halt the processor.
JZ NEXT2	
DISPLAY: MVI A,00H	
NEXT2: STA FE75H	STEP 8: Halt the processor.
CALL UPDDT	
HLT	

NOTE:

- Store the program starting from F000H, Store the data at LOC F100H., Execute the program.
- Result will be displayed in data field, Displays FF on Data filed for valid condition.
- Displays 00 on Data filed for invalid condition.

   # Address for UPDDT is 06D6H

RESULT:

Eg1:   **Eg2:**

BEFORE EXECUTION	
Address	Data
F100H	12H

BEFORE EXECUTION	
Address	Data
F100H	19H

AFTER EXECUTION: **AFTER   EXECUTION**

Data Field: | F | F | | 0 | 0 |

SIGNATURE OF STAFF-IN-CHARGE

18. WAP to find the smallest of 'N' 1-byte numbers. Value of 'N' is stored in location 'X' & numbers from 'X+1'. Display the number in data field & its address in address field.

PROGRAM	ALGORITHM
START: LXI H,F500H	STEP 1: Initialize the starting address of
MOV C,M	The array of N elements.
INX H	
MOV A,M	STEP 2: Load the count N.
DCR C	
LOOP1: INX H	STEP 3: Find the smallest number by
CMP M	comparing the elements given by
JC AHEAD	Verifying the carry flag.
MOV A,M	
AHEAD: DCR C	STEP 4: Store the address of smallest
JNZ LOOP1	Number at F301.
STA F300H	
LXI H,F500H	
MOV C,M	
INX H	
LOOP3: CMP M	
JZ LOOP2	
INX H	
DCR C	
JNZ LOOP3	
LOOP2: SHLD F301H	
CALL UPDAD	
LDA F300H	
CALL UPDDT	
HLT	

RESULT:

Length of Block = 04

BEFORE EXECUTION	
Address	Data
F500H(cnt)	04
F501H	A1
F502H	11
F503	01
F504	B2

AFTER EXECUTION:

Address				Data
F	5	0	3	01

19. WAP to realize real time clock. Display seconds in data field, minutes and hours in address field.

PROGRAM	ALGORITHM

START: LXI H,0000H
MIN : MVI A,00H
SEC: PUSH PSW
PUSH H
PUSH H
CALL UPDDT
POP H
CALL UPDAD
LXI D,FFFFH
CALL DELAY
CALL DELAY
POP H
POP PSW
ADI 01H
DAA
CPI 60
JNZ SEC
MOV A,L
ADI 01H
DAA
MOV L,A
CPI 60
JNZ MIN
MVI L,00
MOV A,H
ADI 01H
DAA
MOV H,A
CPI 24H
JNZ MIN
JMP START
HL T
	STEP 1: Initialize the data for seconds in
	acc...
	STEP 2: Initialize the data for minutes in L
	reg.
	STEP 3: Initialize the data for hours in H
	reg
	STEP 4: Display the data in the display
	field
	STEP 5: Call proper delay of one second.
	STEP 6: Increment the second by 01 and
	compare it with the value 60
	suitably if it is equal increment the
	minute by one and compare it with
	the value 24 suitably, if not for all
	the above increment the second
	Value and repeat the steps 4-5. STEP7: Termination.

RESULT: AFTER EXECUTION

Address field	
Hours	Min
00	00
..	..
..	..
..	..

Data field
Sec
01
..
..
..

SIGNATURE OF STAFF-IN-CHARGE

20. Write an ALP to convert a BINARY numbers stored at LOC X to its BCD equivalent and display it in the data/addr field.

PROGRAM	ALGORITHM

START: LDA F100H
MOV B,A
MVI D, 64H
CALL BCD
MOV H,C
MVI D,0AH
CALL BCD
MOV A,C
RLC
RLC
RLC
RLC
ORA B
MOV L,A
CALL UPDAD
HLT
BCD: MVI C,00H
MOV A,B
RPTS: SUB D
JC NC
INR C
JMP RPTS
NC: ADD D
MOV B,A
RET
	STEP 1: Load the number to be
	        converted
	STEP 2: On the basis of successive
	        subtraction find the
	        Co-efficient in BCD form.
	STEP 3: Display the result in Address
	        Field.
	
	STEP4: Halt the processor

NOTE:

- Store the program starting from LOC F000H
- Store the program starting from LOC F100H
- Execute the program
- Observe the result on the address field

RESULT:

EXAMPLE 1:

BEFORE EXECUTION:

F100H ⎯⎯⎯~~FFH~~⎯⎯→

AFTER EXECUTION:

BINARY NUMBER   BCD NUMBER

FFH ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯→

0	2	5	5

EXAMPLE 2:

BEFORE EXECUTION:

F100H ⎯⎯⎯⎯⎯⎯→ ACH

AFTER EXECUTION:

BINARY NUMBER   BCD NUMBER

ACH ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯→

0	1	7	2

SIGNATURE OF STAFF-IN-CHARGE

21. Write an ALP to convert a BINARY NUMBER stored at location 'X' to its ASCII EQUIVALENT and display in DATA field.

PROGRAM	ALGORITHM

START: LDA F100H
         MOV B,A
         CALL ASCII
         MOV L,A
         MOV A,B
         RRC
         RRC
         RRC
         RRC
         CALL ASCII
         MOV H,A
         MOV A,B
         PUSH H
         CALL UPDDT
         POP H
         CALL UPDAD
         HLT
ASCII: ANI 0FH
         CPI 0AH
         JC BUS
         ADI 07H
BUS: ADI 30H
RETURN: RET
	STEP 1: Load the binary number from
	         LOC 'X'
	STEP 2: Separate the nibbles
	STEP 3: Convert each nibble to its ASCII
	         Equivalent.
	STEP 4: Add the two converted values.
	STEP 5: Display the result in the DATA
	         Field.
	STEP 6: Halt the processor.

NOTE:

- Store the program starting from F000h.
- Store the binary number at F100h.
- Execute the program; the result will be displayed in the data field.

Address for UPDDT: is 06D6H.

Address for UPDAD: is 06BFH.

RESULT:

BEFORE EXECUTION		
Addr. of LOC 'X'		DATA
1.	F100H	01H
2.	F100H	ABH

AFTER EXECUTION	
Data in Data field	
30	31
41	42

SIGNATURE OF STAFF-IN-CHARGE

http://www.8085projects.info/

22. Write an ALP to convert a ASCII NUMBER stored at location 'X' to its BINARY EQUIVALENT and display in DATA field.

PROGRAM	ALGORITHM
START: LDA F100H	STEP 1: Load the ASCII number from
CPI 40H	LOC 'X'.
JC RPT	STEP 2: Check for digit / alphabets.
JZ HALT	STEP 3: Using suitable logic and
SUI 40H	instructions convert the ASCII
ADI 09H	Number into binary.
JMP DISP	STEP 4: Display it in the DATA field.
RPT: SUI 30H	STEP 6: Halt the processor.
DISP: CALL UPDDT	
HALT: HLT	

NOTE:

- Store the program starting from F000h.
- Store the ASCII number at F100h.
- Execute the program; the result will be displayed in the data field.

  # Address for UPDDT: is 06D6H.

RESULT:

BEFORE EXECUTION		AFTER EXECUTION
Addr. of LOC 'X'	DATA	Data in Data field
1. F100H	30H	00
2. F100H	34	04

23. Write a program to convert a BCD number stored at LOC X to its BINARY equivalent & display it in data field

PROGRAM	ALGORITHM
START: LDA F100H	STEP 1:Load the BCD numbers from
MOV B,A	LOC x to ACC
ANI F0H	
RRC	STEP 2: Separate the higher & lower
RRC	nibbles
RRC	
RRC	STEP 3:Convert the nibbles in to binary
MOV D,A	values by multiplying their
MVI C,0AH	nibbles by its factor
SUB A	
RPTA: ADD D	STEP 4:ADD the two binary numbers
DCR C	
JNZ RPTA:	STEP 5:Display the result in the DATA
MOV D,A	field
MOV A,B	
ANI 0FH	STEP 6:Halt the processor
ADD D	
CALL UPDDT	
HALT: HLT	

NOTE:

- Store the program starting from F000H
- Store the BCD number at F100H
- Execute the program, result will be displayed in the display field

RESULT:

BEFORE EXECUTION:

Eg1: F100H ————————▶ 45

Eg2: F100H ———————— 04

AFTER EXECUTION:

DATA FIELD

Eg1:

2	D

Eg2:

4	0

SIGNATURE OF STAFF-IN-CHARGE

PROGRAMS ON INTERFACING

24. Write a program to generate square wave of given duty cycle using DAC

Display the waveform on a CRO & verify the same.

PROGRAM	ALGORITHM
START: MVI A,80	STEP 1:Write the control word in to the
OUT CWR	PPI of the kit
RPT: XRA A	
OUT P_a	
OUT P_b	STEP 2: Pass the data's for square wave
CALL OFFCOUNT	towards PPI words
MVI A,FF	
OUT P_a	
OUT P_b	STEP 3: Pass the alternative data's for
CALL ONCOUNT	LOW & HIGH alternatively
JMP RPT	with proper delay according to
HLT	the duty cycle given
ONCOUNT: LXI H,08	
LOOP: DCX H	
MOV A,L	S TEP 4: Keep the processor in a
ORA H	continuous loop till termination
JNZ LOOP	
RET	STEP 5: Terminating point
OFFCOUNT: LXI H,03	
LOOP1: DCX H	
MOV A,L	
ORA H	
JNZ LOOP	
RET	

NOTE:

- Store the program starting from F000H
- Connect the interfacing unit to the kit
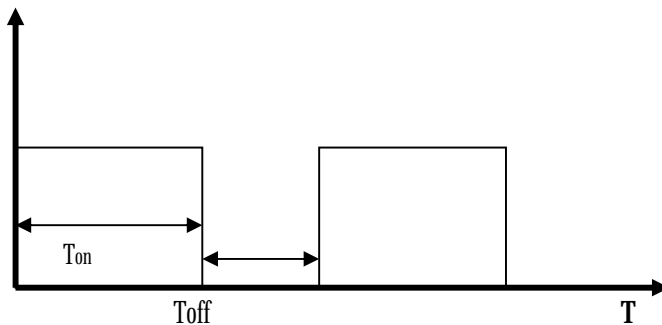- Execute the program
- Observe the waveform on the CRO

SIGNATURE OF STAFF-IN-CHARGE

PORT ADDRESS:

FOR P3		FOR P4	
PORT	ADDRESS	PORT	ADDRESS
PORT A	D8	PORT A	F0
PORT B	D9	PORT B	F1
PORT C	DA	PORT C	F2
CWR	DB	CWR	F3

OUT PUT WAVEFORM:

CALUCLATIONS:

Duty cycle=70%,f=1kHz,T=1/f=1m sec

D=Ton/T

0.7=Ton/1m sec

Ton=0.7msec, T=Ton+Toff

There fore Toff=0.3m Sec

(i)   Ton(delay)=0.7msec

   Total number of T state=$0.7 \times 10^{-3}/0.33 \times 10^{-6}$=2121

Total number of T state=10+(count-1)24+21

2121=10+(count-1)24+21

2121=10+(count -1)24+21=88.09

on count=0058H

(ii)   Toff(delay)=0.3msec

   Total number of T state=$0.3 \times 10^{-3}/0.33 \times 10^{-6}$=909

Total number of T state=10+(count-1)24+21

2121=10+(count-1)24+21

2121=10+(count -1)24+21=37.5

on count=0025H

Note: Caluclate for80, 60%,50% duty cycles

25. Write a program to generate a triangular wave using a DAC. Display the Waveform and measure the slope.

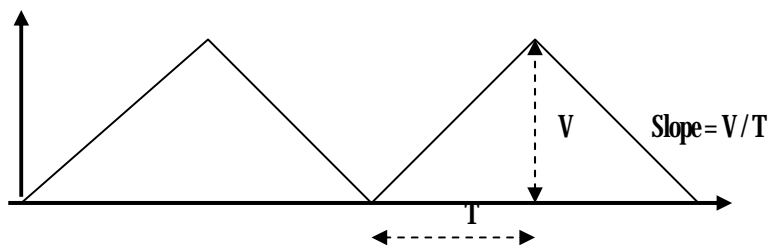| PROGRAM	ALGORITHM

START: MVI   A, 80
      OUT   CWR
REP: XRA   A
VP: OUT P_a
      OUT P_b
      INR   A
      CPI   FF
      JNZ   UP
DN: DCR   A
      OUT P_a
      OUT P_b
      JNZ   DN
      JMP   REP
	STEP 1:Write the control word in to the control register of PPI
	STEP 2:Send the data's towards PPI to generate triangular wave
	STEP3:send the data's for positive slope & negative slope alternatively
	STEP 4:Keep the processor in the continuous loop, till termination
	STEP 5: Terminating point

NOTE:

- Store the program starting from F000H
- Connect the interfacing unit to the kit
- Execute the program
- Observe the waveform on the CRO

PORT ADDRESS:

FOR P3		FOR P4	
PORT	ADDRESS	PORT	ADDRESS
PORT A	D8	PORT A	F0
PORT B	D9	PORT B	F1
PORT C	DA	PORT C	F2
CWR	DB	CWR	F3

OUTPUT WAVEFORM:

SIGNATURE OF STAFF-IN-CHARGE

26. Write a Program to generate a staircase waveform using DAC.

PROGRAM	ALGORITHM
START: MVI A,80	Step 1: Write the control word in to the control register of PPI.
OUT CWR	Step 2: Send the data's towards PPI to generate staircase wave.
GF: MVI A,00	Step 3:send the data's for positive slope & negative slope
NF: OUT P_b	alternatively
PUSH PSW	Step 4:Keep the processor in the continuous loop, till
CALL DELAY	termination
POP PSW	Step 5: Terminating point
ADI 33	
JNC NF	
JMP GF	
DELAY: MVI B,FF	
BACK: DCR B	
JNZ BACK	
RET	

NOTE:

- o Store the program starting from F000H
- o Connect the interfacing unit to the kit
- o Execute the program
- o Observe the waveform on the CRO

PORT ADDRESS:

FOR P3		FOR P4	
PORT	ADDRESS	PORT	ADDRESS
PORT A	D8	PORT A	F0
PORT B	D9	PORT B	F1
PORT C	DA	PORT C	F2
CWR	DB	CWR	F3

SIGNATURE OF STAFF-IN-CHARGE
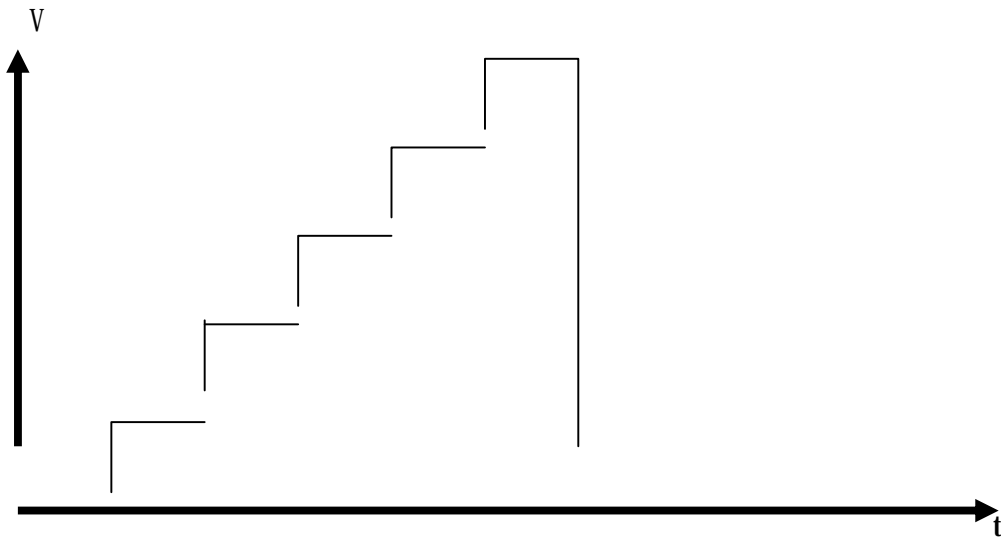
http://www.8085projects.info/

CALCULATION OF STEP SIZE:

NUMBER OF STEPS = N;
STEP SIZE = (MAX. AMPL.) / N

Note: MAX. AMPL. For DAC = 5v (FFH)
Ex.: For 5 steps
  FF / N = 255 / 05 = 51 = 33H
[ADI 33H]

27. Write an Assembly Language Program to Sense a Key board.

PROGRAM	ALGORITHM
START: MVI A, 90H	STEP 1: Write the control word into the control register of PPI.
OUT CWR	STEP 2: Initialize the key board row using suitable data.
BGN: MVI A, 07H	STEP 3 Identify any key closure, if any key closure find the code
OUT PORT C	of the key closed using suitable logic.
DEBOUNCE: IN PORT A	STEP 4: Display the key code in the display field.
ORA A	STEP 5: Repeat the steps 2-4 till termination.
JZ DEBOUNCE	STEP 6: Terminating point.
CALL DELAY	
IN PORT A	
ORA A	
JZ DEBOUNCE	
MVI A, 01H	
MVI C, 00H	
SCAN: MOV B, A	
OUT PORT C	
IN PORT A	
ORA A	
JNZ NXTKEY	
MOV A, C	
ADI 08H	
MOV C, A	
MOV A, B	
RLC	
CPI 08H	
JZ BGN	
JMP SCAN	
NXTKEY: RRC	

```
                    JC FOUND
                    INR C
                    JMP NXTKEY
FOUND:          MOV A, C
                    STA F200H
                    CALL UPDDT
                    JMP BGN
DELAY:          LXI H,00FFH
LOOP:           DCX H
                    MOV A, L
                    ORA H
                    JNZ LOOP
                    RET
```

NOTE:

- Store the program starting from F000H.
- Connect the interfacing unit to the kit.
- Execute the program.
- Press any key in the key board.
- Result will be displayed in the display field.

SIGNATURE OF STAFF-IN-CHARGE

28. Write an ALP to implement a moving display of a given string of digits on a display interface with a suitable delay.

PROGRAM	ALGORITHM
START: MVI A,CW	STEP 1: Initialize all ports
OUT CWR:	STEP 2: Make all rows high
MVI C,04H	STEP 3: Sense the Key board
RPTCD: MVI A,FFH	STEP 4: Is any Key Pressed , if Yes call
CALL DISP	delay
LXI D,FFFFH	STEP 5: If No, Check the Key Pressed
CALL DELY	STEP6: Initialize counter
DCR C	Step 7: Set Row High.
JNZ RPTCD	Step 8:Is any Key Pressed Check first
LXI D,FFFFH	column, If No increment the
CALL DELY	counter by 8 and enable next Row.
LXI H, F100H	Step 9: If Yes Display the counter.
MVI C, 04H	
RPDIS: MOV A,M	
CALL DISP	
INX H	
PUSH H	
PUSH B	
LXI D,FFFFH	
CALL DELY	
POP B	
POP H	
DCR C	
JNZ RPDIS	
LXI D,FFFFH	
CALL DELY	
JMP START	
DISP: MVI E,08H	
MOV B,A	
RPTR: MOV A,B	
OUT PB	
RRC	
MOV B,A	

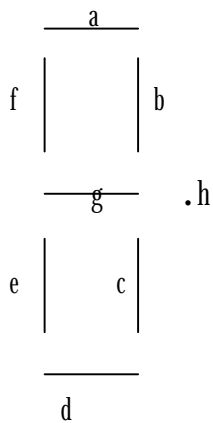|	
MVI A,00H OUT PC CMA OUT PC DCR E JNZ RPTR RETURN: RET	

NOTE:

- Store the program from F000H.
- Store the string of data from F100h.
- Connect the interfacing unit to the PPI of the kit.
- Execute the program.
- Observe the result in the display interface unit.

LED DISPLAY:

```
      ___a___
    f |     | b
      |     |
      ___g___   . h
    e |     | c
      |     |
      _____
         d
```

String for SSIT:

A	b	c	d	e	f	g	h	
0	1	0	0	1	0	0	1	49H(S)
0	1	0	0	1	0	0	1	49H(S)
1	0	0	1	1	1	1	1	9FH(i)
1	1	1	0	0	0	0	1	E1H(t)

SIGNATURE OF STAFF-IN-CHARGE

http://www.8085projects.info/

29. Write a program to display the ASCII equivalent of the key pressed using 8279.

PROGRAM	ALGORITHM
START: MVI A, 0EH	Step 1: Initialise the 8279 IC & initialize
SIM	the interrupt system by suitable
EI	data
CALL RDKBD	Step 2: Convert the received data from
PUSH PSW	the key pressed in to its ASCII
MOV B, A	equivalent
CALL ASCII	Step 3: Display the same in the display
MOV L, A	field
RRC	Step 4: Repeat the steps 1-4 for each key
RRC	pressed till termination
RRC	Step 5: Terminating point
RRC	
CALL ASCII:	
MOV H, A	
POP PSW	
PUSH H	
CALL UPDDT	
POP H	
CALL UPDAD	
JMP START	
HALT: HLT	
ASCII: ANI 0FH	
CPI 0AH	
JC BAT	
ADI 07H	
BAT: ADI 30H	
RET	

NOTE:

o   Store the program from F000H

o   Execute the program

o   Press any key in the key board other than the RESET key

o   The result will be displayed in the display field

The address for RDKBD: is 0634H

30. Write an ALP to simulate 'THROW OF A DICE' using interrupts.

PROGRAM	ALGORITHM
START: MVI A,0BH	STEP 1: Initialize the interrupt system by proper data.
SIM	STEP 2: Write the interrupt service routine at proper location
EI	(memory location)
CAR: MVI A,01H	STEP 3: Interrupt service routine is a program for dice simulation
RPT: INR A	(counting from 0-6).
CPI 06H	STEP 4: Loop the program control in a continuous mode.
JNZ RPT	STEP 5: Terminating point.
JMP CAR	
INRTS: DI	
PUSH PSW	
CALL UPDDT	
POP PSW	
EI	
RETRN: RET	
FFB1: C3 0F F0	

NOTE:
- Store the program starting from F000H.
- Store the interrupt service routine starting from INRTS: address in F00FH.
- Store the instruction JMP INRTS: at memory location FFB1H
- Execute the program.

RESULT:

Press the 'Vect intr' button in the keyboard, for each pressing a display will be there in the display field (data field). I t displays from 00 to 06.

SIGNATURE OF STAFF-IN-CHARGE

http://www.8085projects.info/