LAB MANUAL

Microprocessor and Microcontroller Lab

Bachelor of Technology

in

Electronics & Communication Engineering

Department of Electronics & Communication Engineering

School of Studies of Engineering & Technology

Guru Ghasidas Vishwavidyalaya

Bilaspur-495009 (C. G.)

Website: www.ggu.ac.in

SCHOOL OF STUDIES OF ENGINEERING & TECHNOLOGY
GURU GHASIDAS VISHWAVIDYALAYA, BILASPUR (C.G.)
(A CENTRAL UNIVERSITY)

CBCS-NEW SYLLABUS

B. TECH. THIRD YEAR (Electronics and Communication Engineering)

Vision and Mission of the Institute

Vision		To be a leading technological institute that imparts transformative education to create globally competent technologists, entrepreneurs, researchers and leaders for a sustainable society
Mission	1	To create an ambience of teaching learning through transformative education for future leaders with professional skills, ethics, and conduct.
	2	To identify and develop sustainable research solutions for the local and global needs.
	3	To build a bridge between the academia, industry and society to promote entrepreneurial skills and spirit

Vision and Mission of the Department

Vision		The Department endeavours for academic excellence in Electronics & Communication Engineering by imparting in depth knowledge to the students, facilitating research activities and cater to the ever-changing industrial demands, global and societal needs with leadership qualities.
Mission	1	To be the epitome of academic rigour, flexible to accommodate every student and faculty for basic, current and future technologies in Electronics and Communication Engineering with professional ethics.
	2	To develop an advanced research centre for local & global needs.
	3	To mitigate the gap between academia, industry & societal needs through entrepreneurial and leadership promotion.

Program Educational Objectives (PEOs)

The graduate of the Electronics and Communication Engineering Program will

PEO1: Have fundamental and progressive knowledge along with research initiatives in the field of Electronics & Communication Engineering.

PEO2: Be capable to contrive solutions for electronic & communication systems for real world applications which are technically achievable and economically feasible leading to academia, industry, government and social benefits.

PEO3: Have performed effectively in a multi-disciplinary environment and have self-learning & self-perceptive skills for higher studies, professional career or entrepreneurial endeavors to be confronted with a number of difficulties.

PEO4: Attain team spirit, communication skills, ethical and professional attitude for lifelong learning.

Programme Outcomes: Graduates will be able to:

PO1: Fundamentals: Apply knowledge of mathematics, science and engineering.

PO2: Problem analysis: Identify, formulate and solve real time engineering problems using first principles.

PO3: Design: Design engineering systems complying with public health, safety, cultural, societal and environmental considerations

PO4: Investigation: Investigate complex problems by analysis and interpreting the data to synthesize valid solution.

PO5: Tools: Predict and model by using creative techniques, skills and IT tools necessary for modern engineering practice.

PO6: Society: Apply the knowledge to assess societal, health, safety, legal and cultural issues for practicing engineering profession.

PO7: Environment: Understand the importance of the environment for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics, and responsibilities and norms of the engineering practice.

PO9: Teamwork: Function effectively as an individual and as a member or leader in diverse teams and multidisciplinary settings.

PO10: Communication: Communicate effectively by presentations and writing reports.

PO11: Management: Manage projects in multidisciplinary environments as member or a team leader.

PO12: Life-long learning: Engage in independent lifelong learning in the broadest context of technological change.

Programme Specific Outcomes:

PSO1: Identify, formulate and apply concepts acquired through Electronics & Communication Engineering courses to the real-world applications.

PSO2: Design and implement products using the cutting-edge software and hardware tools to attain skills for analyzing and developing subsystem/processes.

PSO3: Ability to adapt and comprehend the technology advancement in research and contemporary industry demands with demonstration of leadership qualities and betterment of organization, environment and society.

List of Program

Name of Lab Microprocessor and Microcontroller

1. Program to move a data block without overlap.
2. Program to execute ascending/descending order.
3. Program to add N one byte numbers.
4. Write ALP to load the Hexadecimal numbers 9BH and A7H I register D and E respectively, add the numbers. If sum is greater than FFH, display 01 at memory location 2050H otherwise display the sum
5. Program to add BCD numbers
6. program to subtract two 8 bit numbers
7. Program to implement multiplication by successive addition method
8. Program to implement HEX up counter
9. Program to implement HEX down counter
10. Program to implement square wave generation using DAC
11. Program to implement triangular wave generation using DAC
12. Program to display using seven segment display scrolling.
13. Program to display ASCII equivalent of the key pressed
14. Program to control the speed and direction of stepper motor.
15. Write a Program to add a data byte located at offset 0500H in 2000H segment to another data byte available at 0600H in the same segment and store the result at 0700H in the segment.
16. Add the contents of the memory location 2000H:0500H to contents of 3000H:0600H and store the result in 5000H:0700H.
17. Program to multiply 25 by 10 using the technique of repeated addition
18. Write a program to load the accumulator with the values 55H and complement the accumulator 700 times.
19. Write a program to add the first ten natural numbers.
20. To add two numbers such as 25H and 34H, and the result is saved in other register.

Program 1

Program to move a data block without overlap

```
START: LXI H, F000H
LXI D, F100H MVI C, 04
LOOP :        MOV A, M
STAX D
INX    H
INX    D
DCR C
JNZ    LOOP


HLT
```

RESULT:

STARING SRC. ADDR.= F000 STARTING DEST. ADDR.= F100 BLOCK LENGTH= 04

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

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

Program 2

Program to execute ascending/descending order.

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 STA	; Store the content of register A in FFF1.
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.

RESULT SHEET:

N = 07

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

For Descending order Change JC to JNC *"*

Program 3

Program to add N one byte numbers.

PROGRAM	ALGORITHM
START: LXI H, F100	STEP 1: Initialize the starting address of the Data block
MOV C, M	STEP 2: Initialize the count.
SUB A	STEP 3: Initialize the initial sum to zero.
MOV B,A	STEP 4: Add the data bytes one by one.
LOOP: INX H	STEP 5: Increment the memory pointer one by One for one each addition.
ADD M	STEP 6: Decrement the count by one for each Condition. Check for zero condition.
JNC LOOP1	STEP 7: If the count is not zero, repeat step 4 to 6.
INR B	STEP 8: If the count is zero halt the processor.
LOOP1: DCR C	
JNZ LOOP	
MOV H,B	
MOV L,A	
SHLD F2OO	
CALL UPDAD	
HLT	

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

Result Addr.	Data
F200	0A
F201	00

Program 4

Write ALP to load the Hexadecimal numbers 9BH and A7H I register D and E respectively, add the numbers. If sum is greater than FFH, display 01 at memory location 2050H otherwise display the sum.

```
MVI D, 9B H
MVI E, A7H
MOV A, D
ADD E
JNC DSPLY
MVI A, 01 H
DSPLY   STA 2050H
HLT
```

Program 5

Program to add BCD numbers

```
MVI A, 05H
MVI B, 05H
ADD B
STA 2050H
```

Result 10

Program 6

Program to subtract two 8 bit numbers

```
MVI A, 13 H
MVI B, 12 H
SUB B
STA 2070H

Result 01 H
```

Program 7

Program to implement multiplication by successive addition method

LXI H,8000H	Load first operand address			
MOV B, M	Store first operand to B			
F004	23		INX H	Increase HL pair
F005	AF		XRA A	Clear accumulator
F006	4F		MOV C, A	Store 00H at register C
F007	86	LOOP	ADD M	Add memory element with Acc
F008	D2, 0C, F0		JNC SKIP	When Carry flag is 0, skip next task
F00B	0C		INR C	Increase C when carry is 1
F00C	05	SKIP	DCR B	Decrease B register
F00D	C2, 07, F0		JNZ LOOP	Jump to loop when Z flag is not 1
F010	21, 50, 80		LXI H,8050H	Load Destination address
F013	71		MOV M, C	Store C register content into memory
F014	23		INX H	Increase HL Pair
F015	77		MOV M, A	Store Acc content to memory
F016	76		HLT	Terminate the program

Result 8050 93

8051 D0

Program 8

Program to implement HEX up counter

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

RESULT:

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

0	0
0	1

F	E
F	F

Program 9

Program to implement HEX down counter

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

RESULT:

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

F	F
F	E

. .

0	1
0	0

Program 10

Program to implement square wave generation using DAC

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

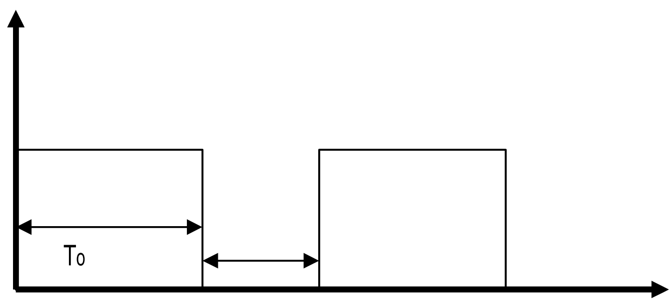
NOTE:

- 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

OUT PUT WAVEFORM:



Program 11

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

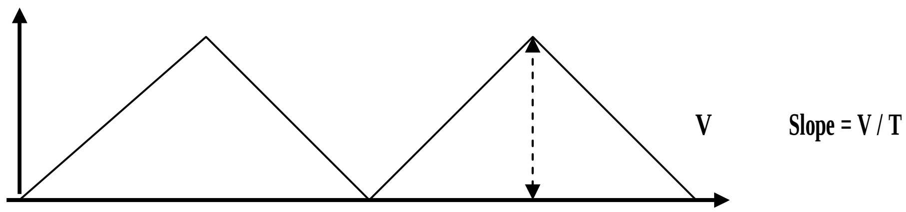
NOTE:

- 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:



Program 12

Program to display using seven segment display scrolling

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

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.

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)

Program 13

Program to display ASCII equivalent of the key pressed

START: MVI A, 0EH
SIM
EI
CALL RDKBD
PUSH PSW
MOV B, A
CALL ASCII
MOV L, A
RRC
RRC
RRC
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
	Step 1: Initialise the 8279 IC & initialize the interrupt system by suitable data
	Step 2: Convert the received data from the key pressed in to its ASCII equivalent
	Step 3: Display the same in the display field
	Step 4: Repeat the steps 1-4 for each key pressed till termination
	Step 5: Terminating point

NOTE:

- Store the program from F000H
- Execute the program
- Press any key in the key board other than the RESET key
- The result will be displayed in the display field # The address for RDKBD: is 0634H

Program 14
Control the speed and direction of stepper motor

Stepper motor Interfacing/Control using 8085 and 8051

Stepper Motor
A stepper motor is a device that translates electrical pulses into mechanical movement in steps of fixed step angle.

- The stepper motor rotates in steps in response to the applied signals.
- It is mainly used for position control.
- It is used in disk drives, dot matrix printers, plotters and robotics and process control circuits.

Structure
Stepper motors have a permanent magnet called rotor (also called the shaft) surrounded by a stator. The most common stepper motors have four stator windings that are paired with a center-tap. This type of stepper motor is commonly referred to as a four-phase or unipolar stepper motor. The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator.

Interfacing

Even a small stepper motor require a current of 400 mA for its operation. But the ports of the microcontroller cannot source this much amount of current. If such a motor is directly connected to the microprocessor/microcontroller ports, the motor may draw large current from the ports and damage it. So a suitable driver circuit is used with the microprocessor/microcontroller to operate the motor.

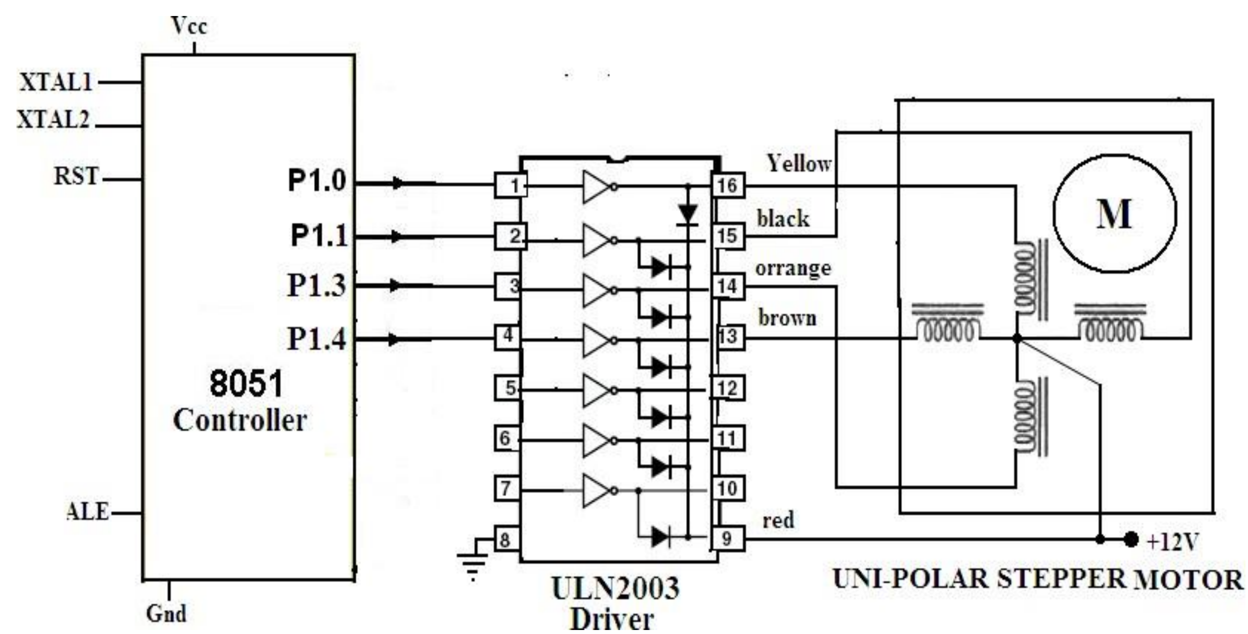
Motor Driver Circuit (ULN2003)
Stepper motor driver circuits are available readily in the form of ICs. ULN2003 is one such driver IC which is a High-Voltage High-Current Darlington transistor array and can give a current of 500mA.This current is sufficient to drive a small stepper motor. Internally, it has protection diodes used to protect the motor from damage due to back emf and large eddy currents. So, this ULN2003 is used as a driver to interface the stepper motor to the microcontroller.

Operation

The important parameter of a stepper motor is the **step angle**. It is the minimum angle through which the motor rotates in response to each **excitation pulse**. In a four phase motor if there are 200 steps in one complete rotation then then the step angle is 360/200 = 1.8^{O} . So to rotate the stepper motor we have to apply the excitation pulse. For this the controller should send a hexa decimal code through one of its ports. **The hex code mainly depends on the construction of the stepper motor**. So, all the stepper motors do not have the same Hex code for their rotation. (refer the operation manual supplied by the manufacturer.)

For example, let us consider the hex code for a stepper motor to rotate in clockwise direction is 77H , BBH , DDH and EEH. This hex code will be applied to the input terminals of the driver through the assembly language program. To rotate the stepper motor in anti-clockwise direction the same code is applied in the reverse order.

Stepper Motor interface- Schematic Diagram (for 8051)



The assembly language program for 8051 is given below.

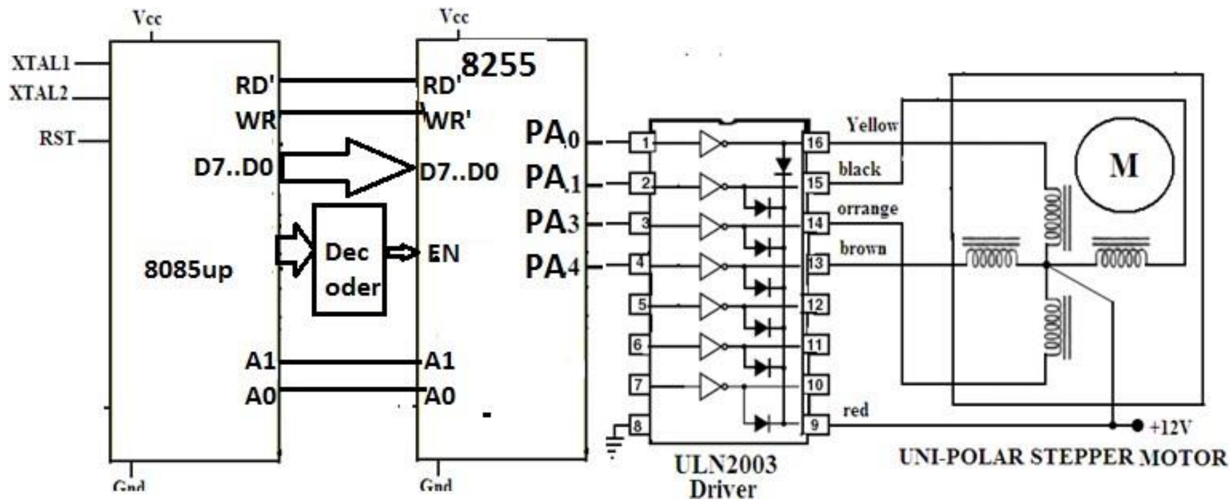
ASSEMBLY LANGUAGE PROGRAM (8051)

```
Main    : MOV A, # 0FF H        ;   Initialization of Port 1
          MOV P1, A              ;
          MOV A, #77 H           ; Code for the Phase 1
          MOV P1, A              ;
          ACALL DELAY            ;  Delay subroutine
          MOV A, # BB H          ; Code for the Phase II
          MOV P1, A              ;
          ACALL DELAY            ;  Delay subroutine.
          MOV A, # DD H          ; Code for the Phase III
          MOV P1, A              ;
          ACALL DELAY            ; Delay subroutine
          MOV A, # EE H          ;  Code for the Phase 1
          MOV P1, A              ;
          ACALL DELAY            ; Delay subroutine
          SJMP MAIN; Keep the motor rotating continuously.
```

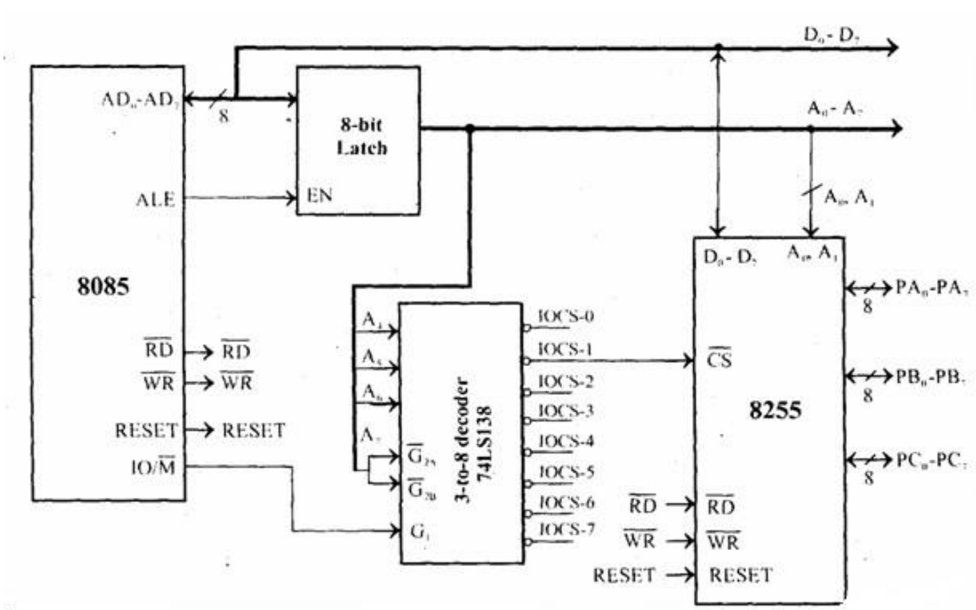
DELAY Subroutine

```
          MOV R4, #0FF H         ;  Load R4 with FF
          MOV R5, # 0FF          ; Load R5 with FF
LOOP1:  DJNZ R4, LOOP1           ; Decrement R4 until zero,wait
LOOP2:  DJNZ R5, LOOP2           ; Decrement R5 until zero,wait
          RET                    ; Return to main program .
```

Stepper Motor interface - Schematic Diagram for (8085)



Detailed Connection diagram between 8085 and 8255



ASSEMBLY LANGUAGE PROGRAM (8085)

```
Main    : MVI A, 80            ;  80H → Control word to configure PA,PB,PC in O/P
          OUT CWR_Address      ;  Write control word in CWR of 8255
          MVI A, 77            ; Code for the Phase 1
          OUT PortA_Address    ; sent to motor via port A of 8255        ;
          CALL DELAY           ;  Delay subroutine
          MVI A,  BB           ; Code for the Phase II
          OUT PortA_Address    ; sent to motor via port A of 8255
          CALL DELAY           ;  Delay subroutine.
          MVI A,  DD           ; Code for the Phase III
          OUT PortA_Address    ; sent to motor via port A of 8255;
          CALL DELAY           ; Delay subroutine
          MVI A,  EE H         ;  Code for the Phase 1
          OUT PortA_Address    ; sent to motor via port A of 8255                  ;
          CALL DELAY           ; Delay subroutine
          JMP MAIN             ; Keep the motor rotating continuously.
```

DELAY Subroutine

```
          MVI C, FF            ; Load C with FF -- Change it for the speed variation
LOOP1:  MVI D,FF               ; Load D with FF
LOOP2:  DCR D
          JNZ  LOOP2
          DCR C
          JNZ LOOP1
          RET                  ; Return to main program .
```

Program 15

Write a Program to add a data byte located at offset 0500H in 2000H segment to another data byte available at 0600H in the same segment and store the result at 0700H in the segment.

```
MOV AX, 2000H
MOV DS, AX
MOV AX, [500H]
ADD AX, [600H]
MOV [700H], AX
HLT
```

Program 16

Add the contents of the memory location 2000H:0500H to contents of 3000H:0600H and store the result in 5000H:0700H

```
MOV CX, 200H
MOV DS, CX
MOV AX, [500H]
MOV CX, 3000H
MOV DS, CX
MOV BX, [0600H]
ADD AX, BX
MOV CX, 5000H
MOV DS, CX
MOV [0700H], AX
HLT
```

Program 17

Program to multiply 25 by 10 using the technique of repeated addition

```
MOV A, #  0
MOV R2, #10
Again ADD A, #25
DJNZ R2, Again
MOV R5, A
```

Program 18

Write a program to load the accumulator with the values 55H and complement the accumulator 700 times.

```
MOV A, #55H
MOV R3, # 10
Next   MOV R2, #70
Again CPL A
DJNZ R2, Again
DJNZ R3, Next
```

Program 19

Write a program to add the first ten natural numbers.

```
MOV A, #0
MOV R2, #10
MOV R0, #0
Again  INC R0
ADD A, R0
DJNZ  R2, Again
MOV 46H, A
```

Program 20

To add two numbers such as 25H and 34H, and the result is saved in other register.

```
MOV A,  # 0
MOV R2, # 25H
MOV R3, # 34 H
ADD A, R2
ADD A, R3
MOV R4, A
```