<u>Note:</u> Team is not responsible for any issues regarding execution and exam.

MPMC Lab Record

8086 Programs

- PASSWORD.
- ASCII.
- PARITY.

8051

- ARITHMETIC & LOGIC OPERATIONS.
- SIGNIFICANCE OF ZERO AND CARRY FLAG
- 10kHz TIME DELAY SQUARE WAVE GENERATOR

HARDWARE INTERFACING:

- STEPPER MOTOR.
- DAC (RAMP WAVE GENERATOR).
- LOGIC CONTROLLER.

PASSWORD

Program:

```
assume cs:code , ds:data
data segment
         message db 'enter password$'
         password db 'jntuh'
         strlen equ($-password)
         correct db 'password verified$'
         notcorrect db 'invalid password$'
data ends
code segment
start:
         mov ax,data
         mov ds,ax
         mov cx,strlen
         mov bx,offset password   ;password in bx
         mov dx,offset message
         mov ah,09h
         int 21h
again:
         mov ah,08h
         int 21h
         cmp al,[bx]
         jnc error
         inc bx
         loop again
         mov dx, offset Correct
         mov ah, 09h
         int 21h
         jmp over
error:
         mov dx,offset notcorrect
         mov ah,09h
         int 21h
over:
         mov ah,4ch
         int 21h
code ends
end start
```

ASCII

Program:

```
assume cs:code,ds:data
data segment
     n1 db 10,13,'enter the first number:$'
     n2 db 10,13,'enter the second number:$'
     n3 db 10,13,'the result number:$'
     num1 db 00h
     num2 db 00h
data ends
code segment
start:      mov ax,data
     mov ds,ax
     mov es,ax                ; extra segment this is comment
     lea dx,n1
          mov ah,9
          int 21h
          xor bx,bx
          mov cx,00h
          mov ah,1
          int 21h
          mov num1,al
     lea dx,n2
          mov ah,9
          int 21h
          xor ax,ax
          mov ah,1
          int 21h
          mov num2,al
          mov bl,num1
          mov dl,num2
          add bx,dx
          mov cx,bx
      lea dx,n3
          mov ah,9
          int 21h
          mov ah,02h
          mov dl,cl
          int 21h
Exit:
      int 3
     code ends
     end start
```

PARITY

Program:

```
assume cs:code , ds:data
data segment
n1 dd 35435544h   ;data double
count equ 04h
data ends
code segment
start:
      mov ax,data
      mov ds,ax
      mov dh,count
      xor ax,ax
      xor cx,cx
      lea si,n1       ;   Load effective address

nxt: add al,[si]
jp evenp
inc cl

evenp: inc si
      xor al,al
      dec dh
      jnz nxt
      xor dl,dl
      rcr cl,1
      jnc clear
      inc dl


Clear:
      mov ah,4ch
      int 21h
      int 21h
code ends
end start
```

8051

Time Delay Generation Using Timers of 8051

Program:

```
MOV P2,#0000B
OV TMOD,#0001BAIN: SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP MAIN
DELAY: MOV TH0,#0FFH
        MOV TL0,#0CEH
        SETB TR0
HERE:JNB TF0,HERE
        CLR TR0
        CLR TF0
        SETB P2.0
RET
END
```

LINK: http://www.circuitstoday.com/delay-using-8051-timer

FOR EXECUTION OVER KIT

```
MOV 9000,#0000
MOV 9003,#0001
```

```
        SETB 9000
        ACALL 900E
        CLR 9000
        ACALL 900E
        SJMP 9006
        MOV 9010,#0FF
        MOV 9013,#0CE
        SETB 9016
        JNB 9019,9018
        CLR 9016
        CLR 9019
        SETB 9000
RET
LCALL 0003
```

Use Values of end , lcall up_dad addresses here

SIGNIFICANCE OF ZERO AND CARRY FLAG

Program:

```
677D                UP_DAD          EQU   677DH
9000                                ORG   9000H
9000
9000  78 35                        MOV   R0,#35H
9002  79 35                        MOV   R1,#50H
9004  E6                          MOV   A,@R0
9005  C3                          CLR   C
9006  99                          SUBB  A,R1
9007  60 0A                       JZ    DIS_00
9009  40 04                       JC    DIS_CC
900B  7E FF                       MOV   R6,#FFH
900D  80 06                       SJMP  END
900F  7E CCDIS_CC:          MOV   R6,#CCH
9011  80 02                       SJMP  END
9013  7E 00         DIS_00:      MOV   R6,#00H

9015  12 67 7D      END:         LCALL UP_DAD
9018  80 FB                       SJMP  END
```

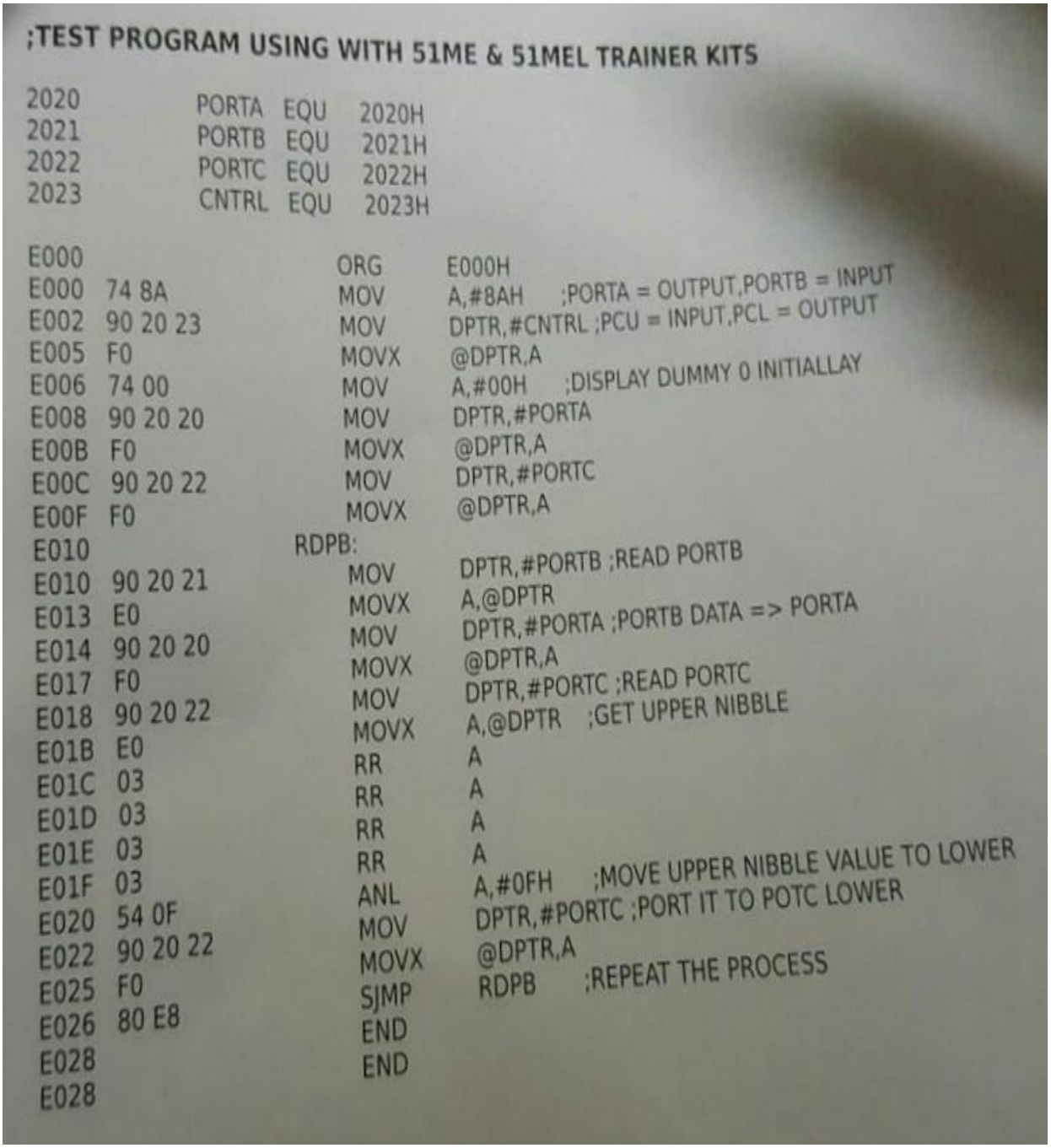
8051 INTERFACING

Ramp wave generator

```
;PROGRAM 2 - RAMP WAVEFORM GENERATION

2023            PORTCP      EQU   2023H  ;8255 control port address
2020            PORTA       EQU   2020H  ;port A address
2021            PORTB       EQU   2021H  ;port B address
2022            PORTC       EQU   2022H  ;port C address

E000                        ORG   E000H

E000  90 20 23              MOV   DPTR,#PORTCP
E003  74 80                 MOV   A,#80H          ;initialise 8255
E005  F0                    MOVX  @DPTR,A         ;port c high as o/p
E006                START:
E006  7C FF                 MOV   R4,#FFH
E008  74 00                 MOV   A,#00H          ;initialise temp.reg to 00H
E00A                A2:
E00A  90 20 20              MOV   DPTR,#PORTA     ;o/p to both D to A
E00D  F0                    MOVX  @DPTR,A
E00E  90 20 21              MOV   DPTR,#PORTB
E011  F0                    MOVX  @DPTR,A
E012  04                    INC   A               ;increment the digital code
E013  00                    NOP
E014  00                    NOP
E015  DC F3                 DJNZ  R4,A2
E017  02 E0 06              LJMP  START
E01A                        END
E01A                        END
```

LED

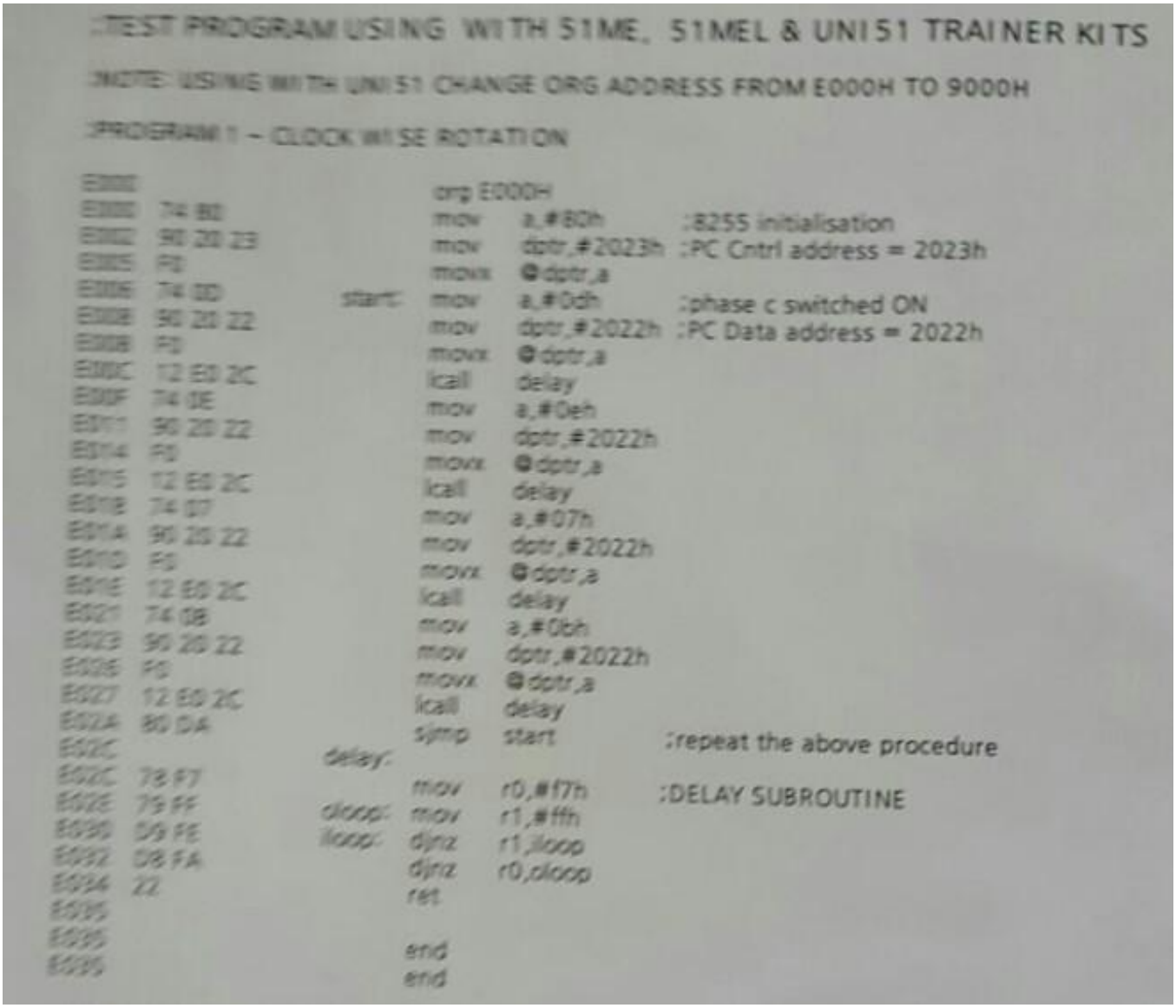
Program:



```
;TEST PROGRAM USING WITH 51ME & 51MEL TRAINER KITS

2020          PORTA  EQU   2020H
2021          PORTB  EQU   2021H
2022          PORTC  EQU   2022H
2023          CNTRL  EQU   2023H

E000                       ORG     E000H
E000  74 8A                MOV     A,#8AH     ;PORTA = OUTPUT,PORTB = INPUT
E002  90 20 23             MOV     DPTR,#CNTRL ;PCU = INPUT,PCL = OUTPUT
E005  F0                   MOVX    @DPTR,A
E006  74 00                MOV     A,#00H     ;DISPLAY DUMMY 0 INITIALLAY
E008  90 20 20             MOV     DPTR,#PORTA
E00B  F0                   MOVX    @DPTR,A
E00C  90 20 22             MOV     DPTR,#PORTC
E00F  F0                   MOVX    @DPTR,A
E010                  RDPB:
E010  90 20 21             MOV     DPTR,#PORTB ;READ PORTB
E013  E0                   MOVX    A,@DPTR
E014  90 20 20             MOV     DPTR,#PORTA ;PORTB DATA => PORTA
E017  F0                   MOVX    @DPTR,A
E018  90 20 22             MOV     DPTR,#PORTC ;READ PORTC
E01B  E0                   MOVX    A,@DPTR    ;GET UPPER NIBBLE
E01C  03                   RR      A
E01D  03                   RR      A
E01E  03                   RR      A
E01F  03                   RR      A
E020  54 0F                ANL     A,#0FH     ;MOVE UPPER NIBBLE VALUE TO LOWER
E022  90 20 22             MOV     DPTR,#PORTC ;PORT IT TO POTC LOWER
E025  F0                   MOVX    @DPTR,A
E026  80 E8                SJMP    RDPB       ;REPEAT THE PROCESS
E028                       END
E028                       END
```

STEPPER MOTOR

Program:



```
;TEST PROGRAM USING WITH 51ME, 51MEL & UNI51 TRAINER KITS

;NOTE: USING WITH UNI51 CHANGE ORG ADDRESS FROM E000H TO 9000H

;PROGRAM 1 - CLOCK WISE ROTATION

E000                    org E000H
E000 74 80              mov   a,#80h        ;8255 initialisation
E002 90 20 23           mov   dptr,#2023h   ;PC Cntrl address = 2023h
E005 F0                 movx  @dptr,a
E006 74 0D      start:  mov   a,#0dh        ;phase c switched ON
E008 90 20 22           mov   dptr,#2022h   ;PC Data address = 2022h
E00B F0                 movx  @dptr,a
E00C 12 E0 2C           lcall delay
E00F 74 0E              mov   a,#0eh
E011 90 20 22           mov   dptr,#2022h
E014 F0                 movx  @dptr,a
E015 12 E0 2C           lcall delay
E018 74 07              mov   a,#07h
E01A 90 20 22           mov   dptr,#2022h
E01D F0                 movx  @dptr,a
E01E 12 E0 2C           lcall delay
E021 74 0B              mov   a,#0bh
E023 90 20 22           mov   dptr,#2022h
E026 F0                 movx  @dptr,a
E027 12 E0 2C           lcall delay
E02A 80 DA              sjmp  start         ;repeat the above procedure
E02C
E02C 78 F7      delay:
E02E 79 FF              mov   r0,#f7h       ;DELAY SUBROUTINE
E030 D9 FE      dloop:  mov   r1,#ffh
E032 D8 FA      iloop:  djnz  r1,iloop
E034 22                 djnz  r0,dloop
E035                    ret
E035
E035                    end
E035                    end
```