CSE-3108

Spring, 2016

Interfacing with Dot Matrix Display through 8255

Tasks: Displaying and moving different shapes on the Dot Matrix display using 8086 microprocessor

➢**Dot Matrix Display**

➢There is a 8X8 Dot Matrix LED display(KMD D1288C) embedded on the MDA-8086 trainer toolkit

➢The Dot Matrix inside the MDA – 8086 trainer kit can be used to display any pattern of LEDs in the dot matrix display

➢This requires *PIO-8255* ports which are already connected to the Dot Matrix internally. You don't have to manually setup a connection between the *Dot Matrix* and *8255A*

➢The 8255A used to interface the Dot Matrix with 8086 is different from the 8255A that was used to interface FND with 8086. Hence, MDA-8086 has multiple 8255A embedded in

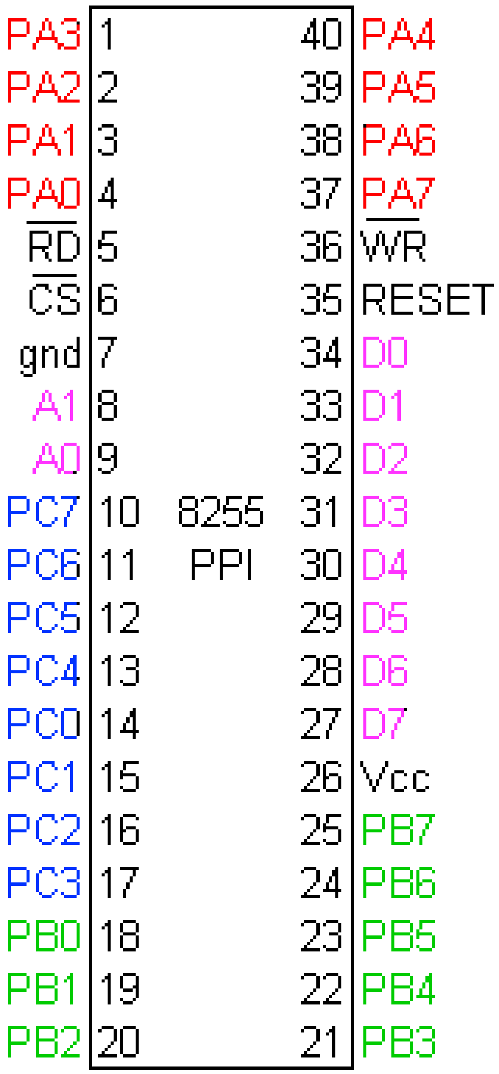
➢By executing proper instructions, we can access these ports and provide binary or hex value to switch the required segment on and off.

➢In order to turn an LED ON, a *logical 0* should be provided to the row and a *logical 1* should be provided to the column because of the following arrangement

➢Each of the Dot Matrix LED can be lit in green, or red, or both(orange)

Introduction to 8255(From Last Slide)

- The Intel 8255 Programmable Peripheral Interface (PPI) chip is a peripheral chip originally developed for the Intel 8085 microprocessor
- 8255 has 40 pins in total, for interfacing with other devices, power supply, chip select etc.
- 24 of these pins are intended towards I/O operations
- These pins are divided intro three 8-bit ports[Port A, Port B, Port C]
- Port A, Port B can be used as 8-bit I/O ports
- Port C can either be used as a 8-bit I/O port, or two 4-bit I/O ports.
- Port C can also be used for producing handshaking signals during handshake data transfer



PA3	1	40	PA4
PA2	2	39	PA5
PA1	3	38	PA6
PA0	4	37	PA7
$\overline{RD}$	5	36	$\overline{WR}$
$\overline{CS}$	6	35	RESET
gnd	7	34	D0
A1	8	33	D1
A0	9	32	D2
PC7	10	31	D3
PC6	11	30	D4
PC5	12	29	D5
PC4	13	28	D6
PC0	14	27	D7
PC1	15	26	Vcc
PC2	16	25	PB7
PC3	17	24	PB6
PB0	18	23	PB5
PB1	19	22	PB4
PB2	20	21	PB3

8255A ports

Introduction to 8255(From Last Slide)

- The three ports are further grouped as follows-
  1. Group A : Port A and upper part of Port C
  2. Group B: Port B and lower part of Port C
- 8255 contains four registers. One for each of Port A, B, C; and another one called ‘Control Word Register’(CWR) for storing the current control state of 8255. CWR stores bits which denote information like whether Port A, Port B, Port C upper or Port C lower are in input mode or output mode.
- Eight data lines (D0-D7) are available (with an 8-bit data buffer) to read/write data into the ports or control register
- The address lines A1 and A0 allow to successively access any one of the ports or the control register
- Port A, B, C are connected with external devices, and data lines(D0-D7) and address lines(A0, A1) are connected with the microprocessor

A 8X8 Dot Matrix Display

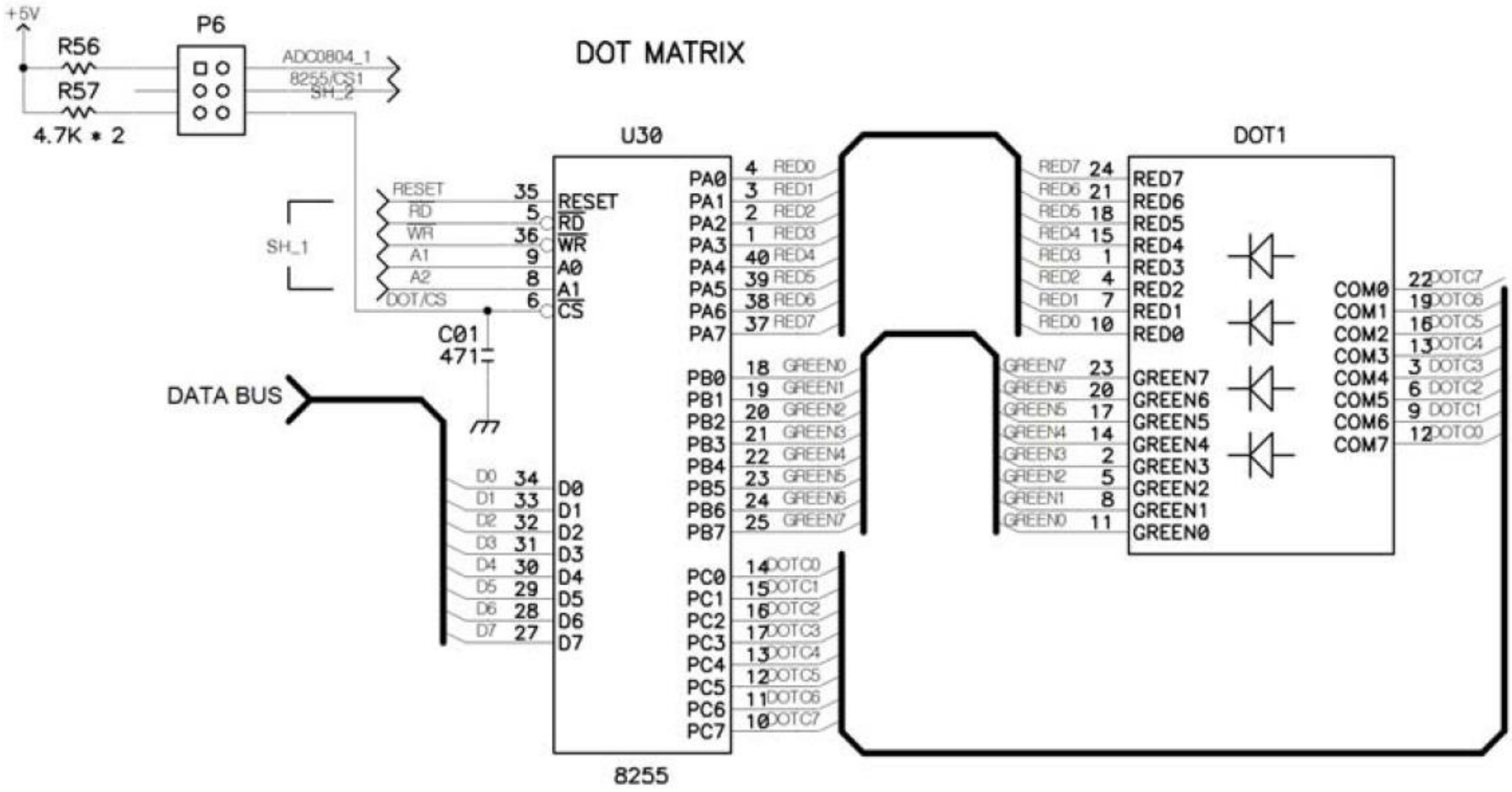
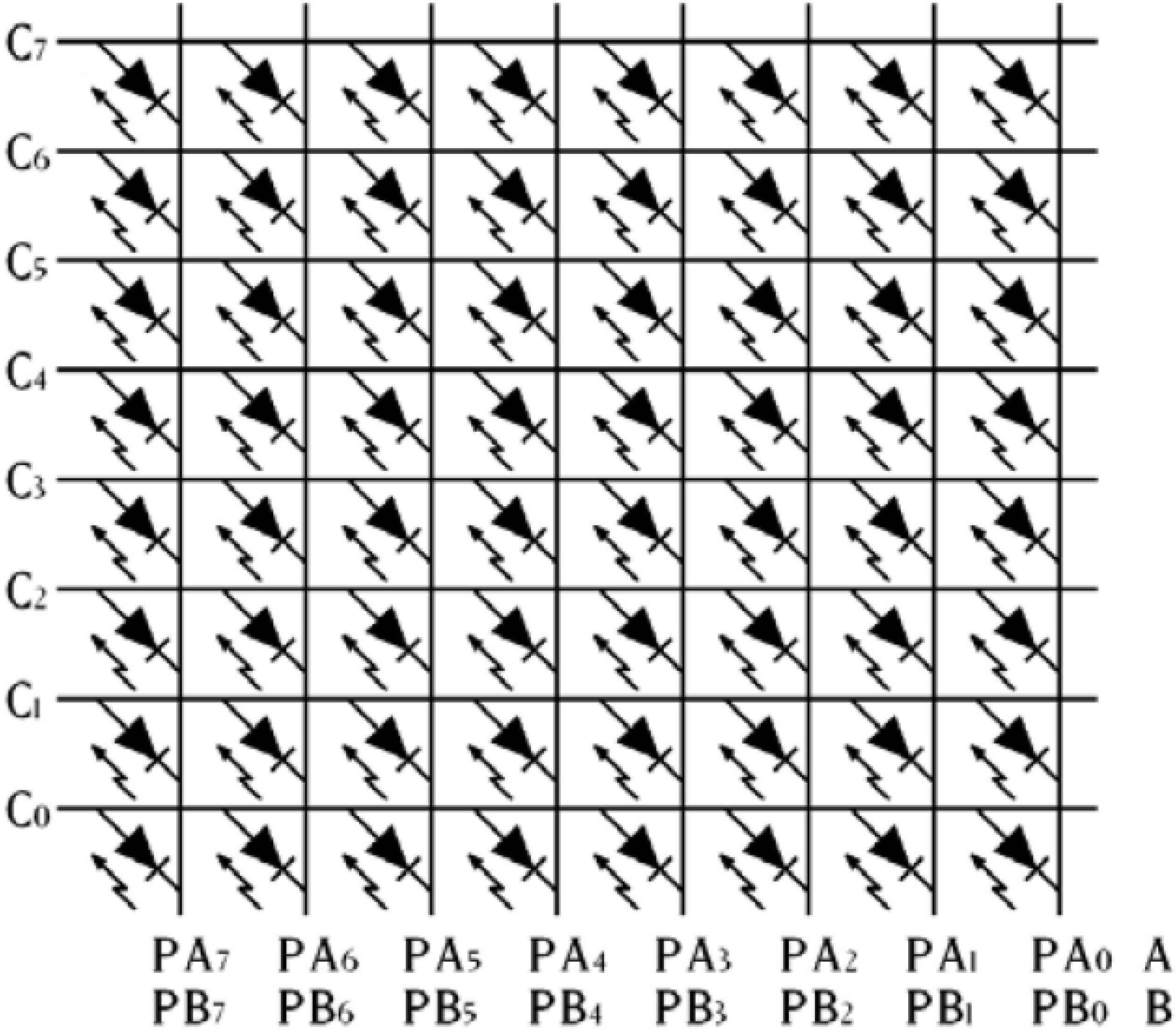

DOT MATRIX
+5V
R56
R57
4.7K * 2
P6
ADC0804_1
8255/CS1
SH_2
SH_1
U30
RESET
RD
WR
A1
A2
DOT/CS
C01
471
DATA BUS
D0 D1 D2 D3 D4 D5 D6 D7
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7
RED0 RED1 RED2 RED3 RED4 RED5 RED6 RED7
GREEN0 GREEN1 GREEN2 GREEN3 GREEN4 GREEN5 GREEN6 GREEN7
DOTC0 DOTC1 DOTC2 DOTC3 DOTC4 DOTC5 DOTC6 DOTC7
8255
DOT1
COM0 COM1 COM2 COM3 COM4 COM5 COM6 COM7

Fig: Dot Matrix LED Interface



Dot Matrix Display internal wiring

How to display a shape on the Dot Matrix

- The Dot Matrix display requires (3 * 8bits) input from 8225A ports. Check the interfacing diagram for clarifications

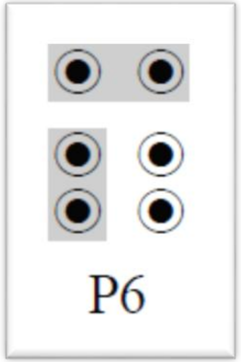
- **Port A** denotes whether or not to display green lights along the rows
- **Port B** denotes whether or not to display red lights along the rows
- **Port C** denotes whether or not to display any light along the columns

- Remember to throw P6 like this before watching an output on the Dot Matrix Display

How to display a shape on the Dot Matrix

- Consider the bitmasks to be sent to different ports of 8255A-
  - Port A, Port B : For the bitmask sent to these ports, LSB denotes the lowest row, and MSB denotes the highest row of the Dot Matrix Display
  - Port C : For the bitmask sent to this port, LSB denotes the leftmost column, and MSB denotes the rightmost column of the Dot Matrix Display

- Remember to throw P6 like this before displaying an output on the Dot Matrix Display



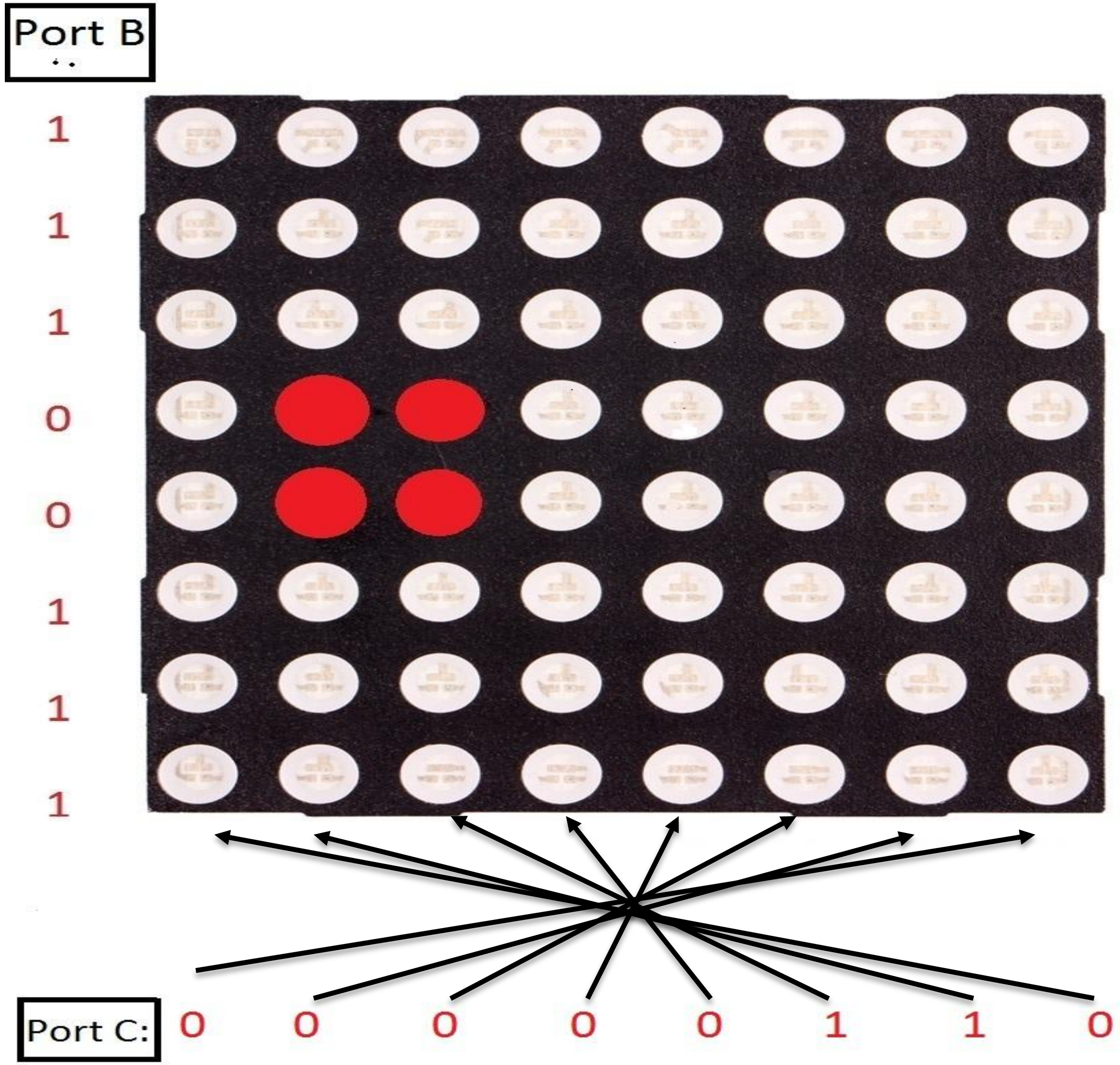
How to display a shape on the Dot Matrix

(This was erroneous on the slide previously provided)

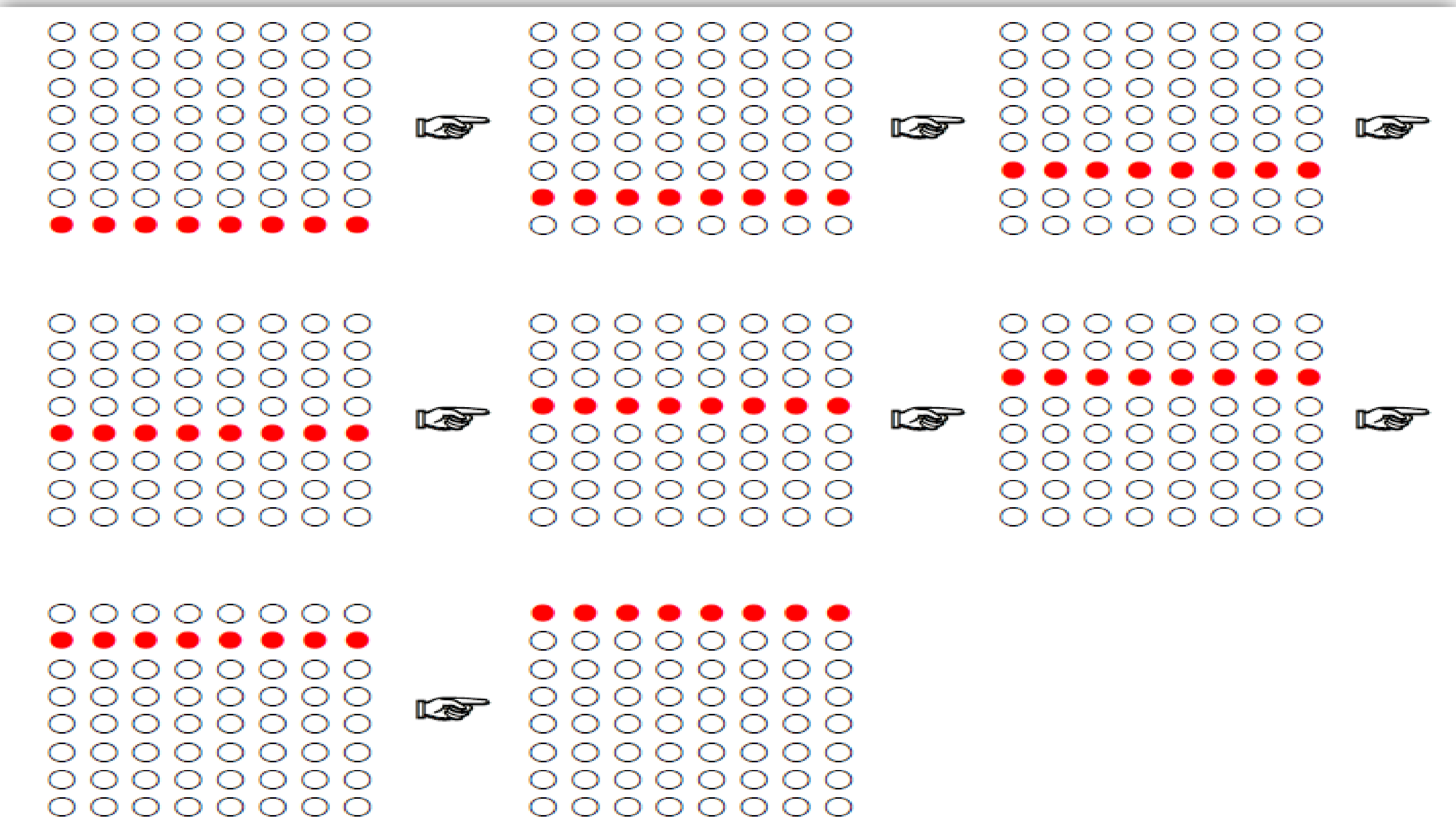
Let’s say we output

```
PortA = 11111111
PortB = 11100111
PortC = 00000110
```

What will we see?



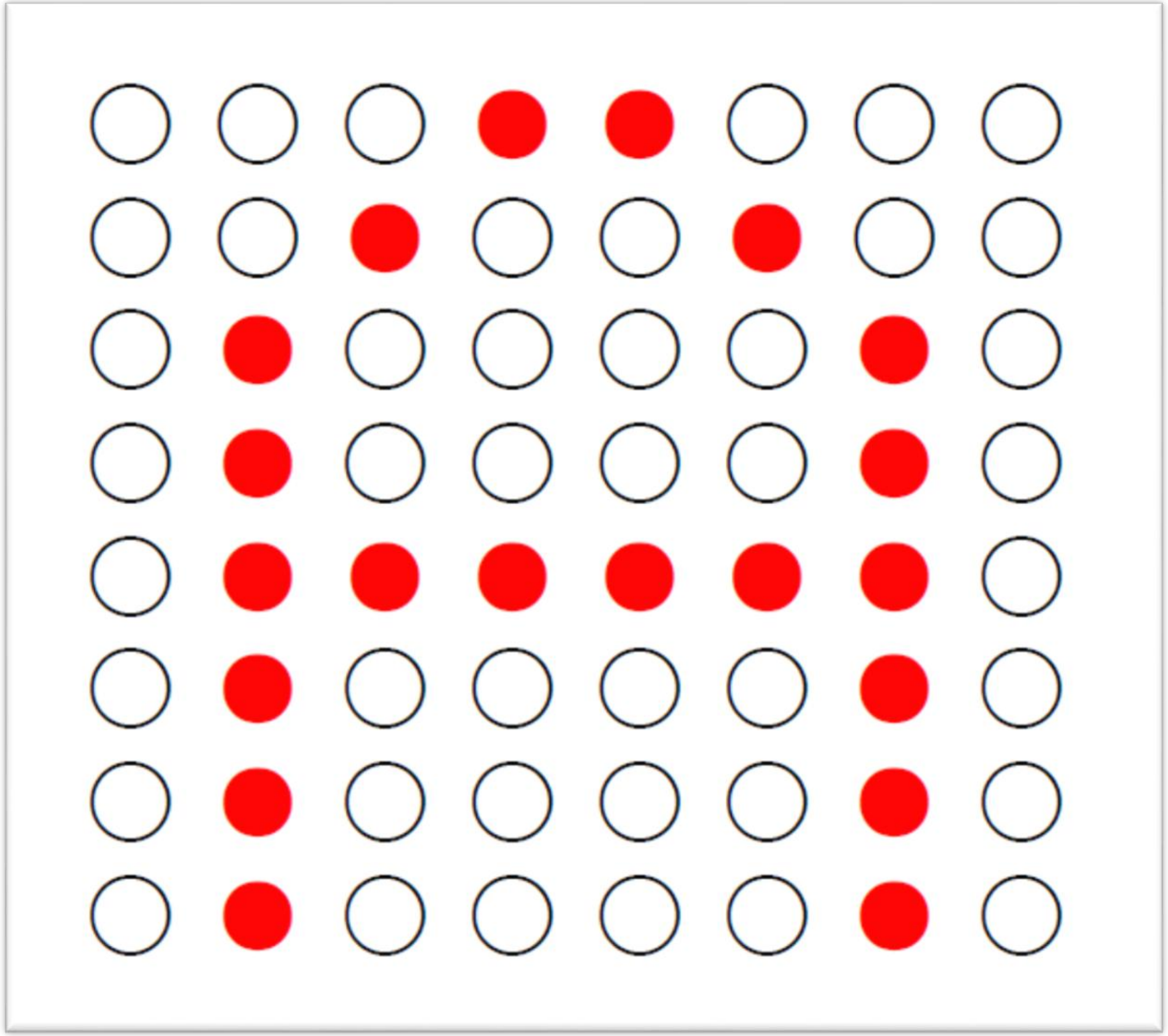
Task-1: Display a red horizontal bar that rotates from bottom to top



```
CODE SEGMENT
ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE

    PPIC_C EQU 1EH
    PPIC EQU 1CH
    PPIB EQU 1AH
    PPIA EQU 18H

    ORG 1000H

    MOV AL, 10000000B
    OUT PPIC_C, AL              ; Take PortA, PortB, PortC to
                                ; output modes

    MOV AL, 11111111B
    OUT PPIC, AL                ; Since all columns should be
                                ; lit at the same time

    MOV AL, 11111111B
    OUT PPIA, AL                ; We'll never light up the
                                ; green lights
                                ; So, output 11111111 to turn
                                ; off all green outputs

L1: MOV AL, 11111110B           ; Since only one row to be
                                ; lit at a time

    MOV CX, 08H
L2: OUT PPIB, AL
    CALL TIMER
    STC
    ROL AL, 1
    LOOP L2
    JMP L1
    INT 3
```

```
; TIMER procedure
TIMER: PUSH CX
       MOV CX, 8FFFH
TIMERLOOP: NOP
    NOP
    NOP
    NOP
    LOOP TIMERLOOP
    POP CX
    RET
    ;
CODE ENDS
END
```

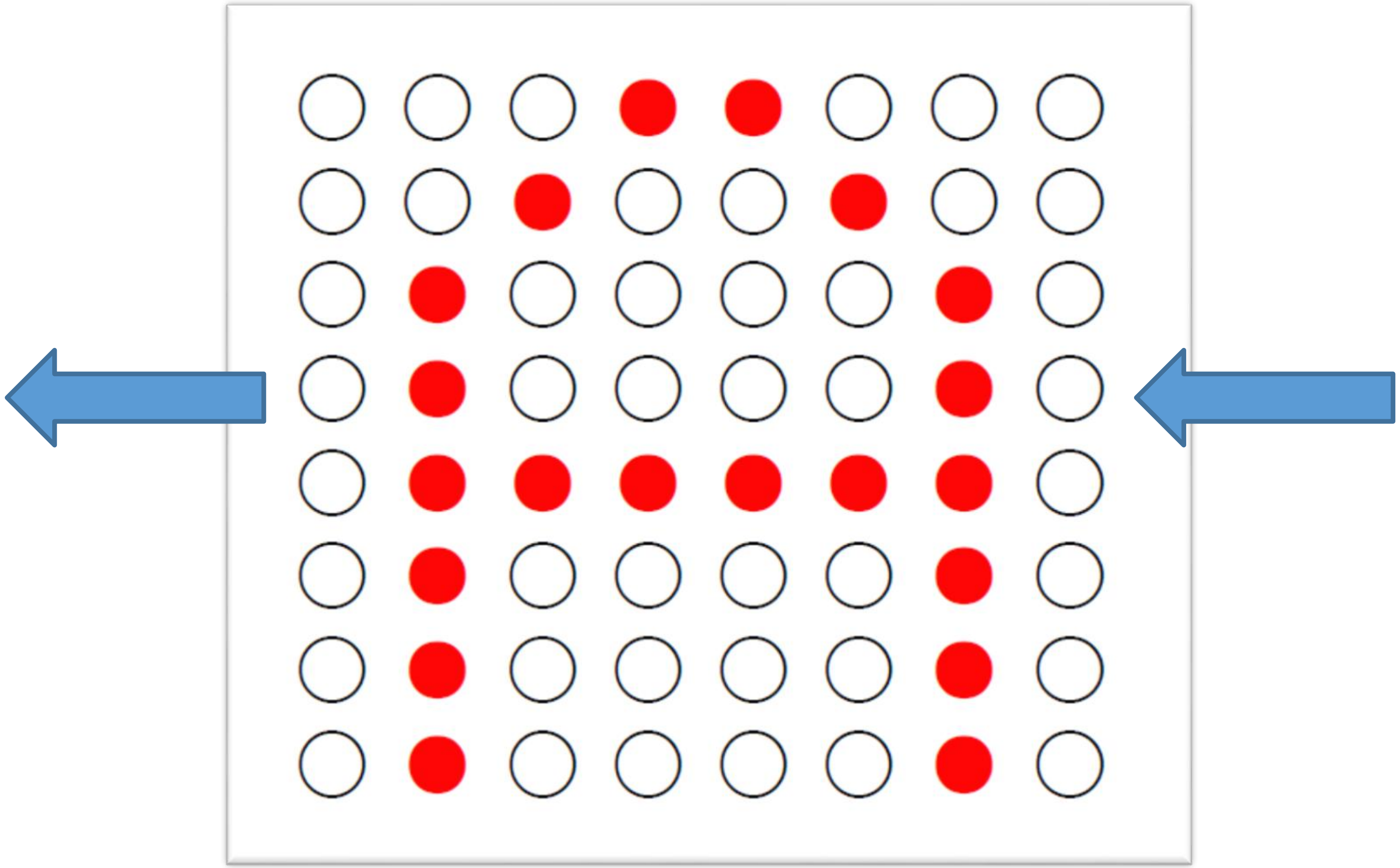
Task-2: Display the letter A



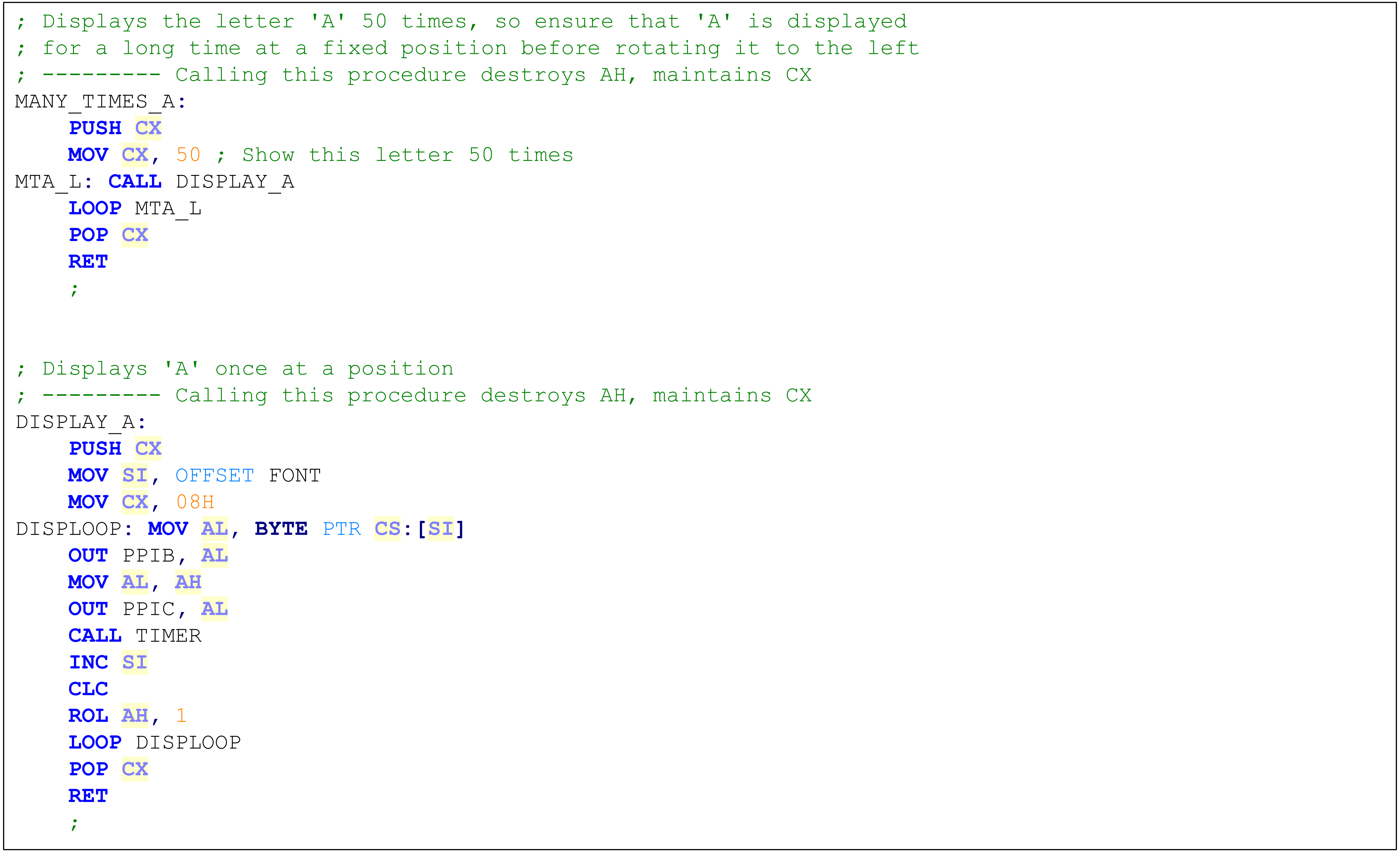
```
CODE SEGMENT
    ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE

    PPIC_C EQU 1EH
    PPIC EQU 1CH
    PPIB EQU 1AH
    PPIA EQU 18H

    ORG 1000H
    MOV AL, 10000000B
    OUT PPIC_C, AL    ; Port A, B, C to output mode

    MOV AL, 11111111B
    OUT PPIA, AL      ; We'll never light up the green lights

L1: MOV AH, 00000001B
    CALL DISPLAY_A
    JMP L1


; Calling this procedure destroys AH
DISPLAY_A:
    MOV SI, OFFSET FONT
    MOV CX, 08H
DISPLOOP: MOV AL, BYTE PTR CS:[SI]
    OUT PPIB, AL
    MOV AL, AH
    OUT PPIC, AL
    CALL TIMER
    INC SI
    CLC
    ROL AH, 1
    LOOP DISPLOOP
    RET
```

```
; TIMER procedure
TIMER:  PUSH CX
        MOV CX, 300
TIMER1: NOP
        NOP
        NOP
        NOP
        LOOP TIMER1
        POP CX
        RET
        ;


FONT:  DB 11111111B; Mask for the column on the far left
       DB 11000000B
       DB 10110111B
       DB 01110111B
       DB 01110111B
       DB 10110111B
       DB 11000000B
       DB 11111111B; Mask for the column on the far right
       ;

CODE ENDS
END
```

Task-2: Display the letter A, rotating from right to left



```
CODE SEGMENT
    ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE

    PPIC_Control EQU 1EH
    PPIC EQU 1CH
    PPIB EQU 1AH
    PPIA EQU 18H

    ORG 1000H
    MOV AL, 10000000B
    OUT PPIC_Control, AL

    MOV AL, 11111111B
    OUT PPIA, AL

    MOV BL, 1H
L1: MOV AH, BL
    CALL MANY_TIMES_A
    CLC
    ROR BL, 1
    JMP L1
```



```
; Displays the letter 'A' 50 times, so ensure that 'A' is displayed
; for a long time at a fixed position before rotating it to the left
; --------- Calling this procedure destroys AH, maintains CX
MANY_TIMES_A:
    PUSH CX
    MOV CX, 50 ; Show this letter 50 times
MTA_L: CALL DISPLAY_A
    LOOP MTA_L
    POP CX
    RET
    ;


; Displays 'A' once at a position
; --------- Calling this procedure destroys AH, maintains CX
DISPLAY_A:
    PUSH CX
    MOV SI, OFFSET FONT
    MOV CX, 08H
DISPLOOP: MOV AL, BYTE PTR CS:[SI]
    OUT PPIB, AL
    MOV AL, AH
    OUT PPIC, AL
    CALL TIMER
    INC SI
    CLC
    ROL AH, 1
    LOOP DISPLOOP
    POP CX
    RET
    ;
```

```
TIMER:  PUSH CX
        MOV CX, 300

TIMER1: NOP
        NOP
        NOP
        NOP
        LOOP TIMER1
        POP CX
        RET
        ;


FONT:   DB 11111111B
        DB 11000000B
        DB 10110111B
        DB 01110111B
        DB 01110111B
        DB 10110111B
        DB 11000000B
        DB 11111111B
        ;

CODE ENDS
END
```