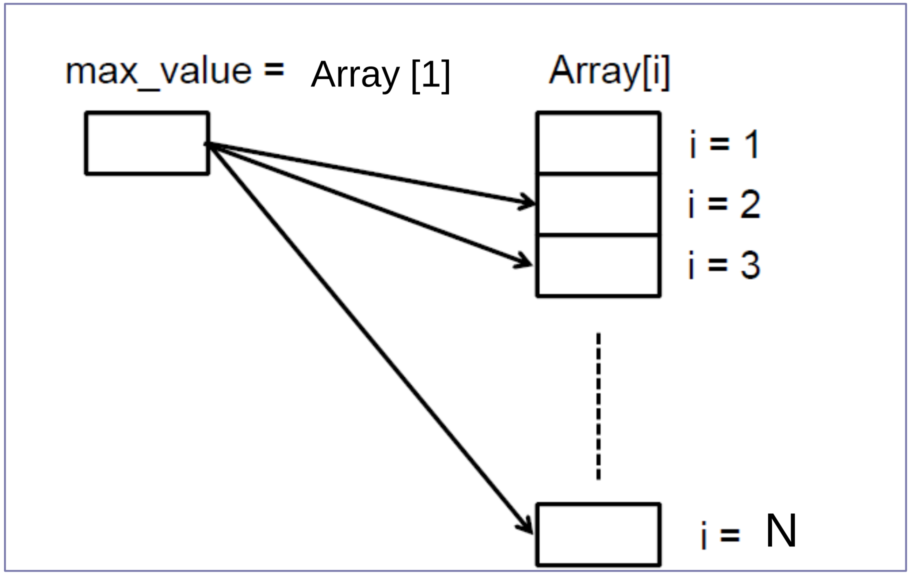
Task I: Find the Max value in a given Array

Task I: Find the max value in an Array

- To find the maximum element in a given array (each element is a byte size). The array starts from the location $1000



1. Max_value = Array [1]
2. Scan the array from Array[2] to Array [N]
3. For loop i=2 to N:

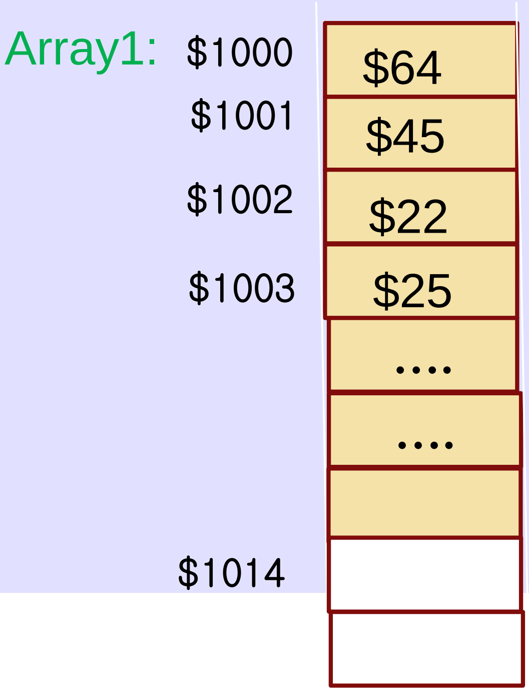
```
 if Max_value < Array[i]  then
          Max_value = Array[i]
```

4 After scanning all the array elements,
max_value = the max. element in the array

Task I: Find the max in an Array (II)

- How to build the Array Data

```
        ORG   RAMStart        ; RAMStart is defined as $1000 in mc9s12dg256.inc
Array1  DC.B  $64, $45,$22, $25, $52, $66, $48, $53, $50,$AF
N       EQU   10
```



Task I: Find the max in an Array (III)

1. Load the first array element into register **A**
2. Load the address of the second element into register X

```
LDX     #Array1+1                        ; X = the address of the second element = $1001
```

3. Load the array length N-1 into register B

```
LDAB     #N -1
```

Loop:

1. Compare register A with the content at the memory location given by register X.
2. If [**A**] >= Mem[X], branch to **SKIP** ; use unsigned branch instruction **BHS**

   if not, replace A with Mem[x]

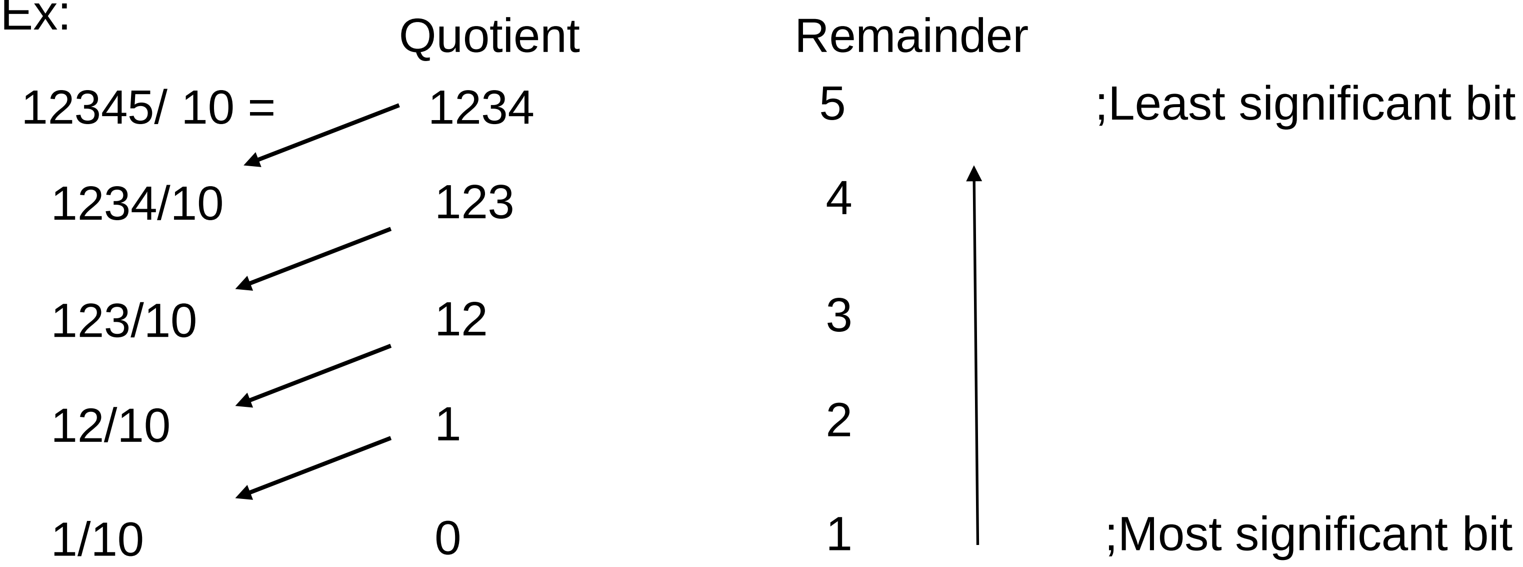
SKIP:

4. Increment register X ;X will hold the address of the next element in the array
5. Decrement the counter B register
6. If B register is not equal to 0, branch to **Loop**

Task II: Number Conversion to the BCD Format

Task2 : Binary to BCD Conversion

- A binary number can be converted to the BCD format **using repeated division by 10.**
- The **first division by 10** generates **the least significant digit** in the remainder
- Ex:



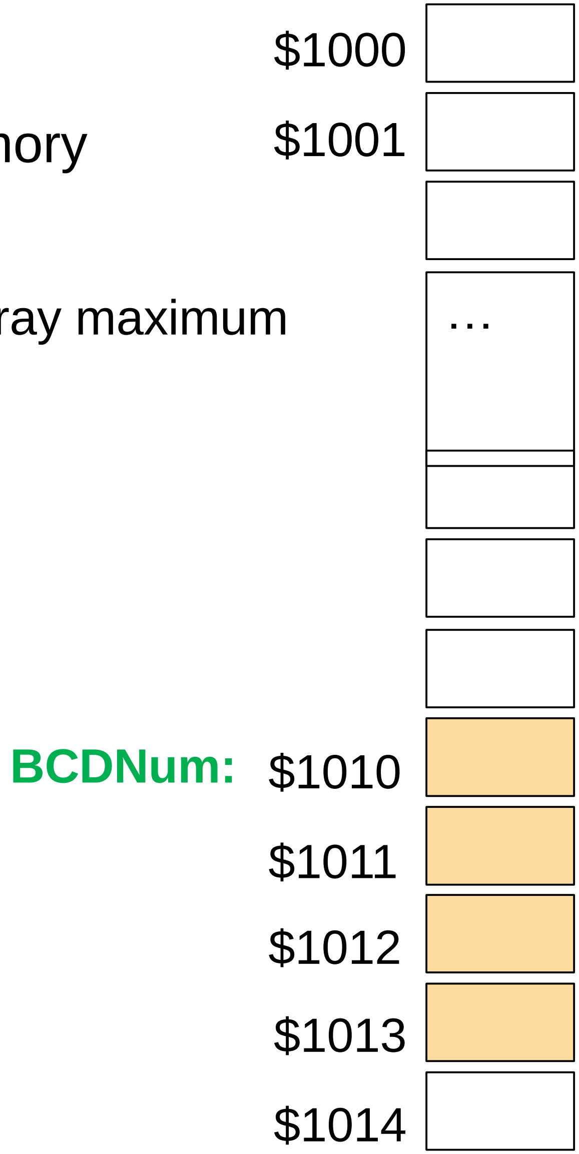
	Quotient	Remainder	
12345/ 10 =	1234	5	;Least significant bit
1234/10	123	4	
123/10	12	3	
12/10	1	2	
1/10	0	1	;Most significant bit

Binary to BCD Conversion

Data allocation for BCD digits

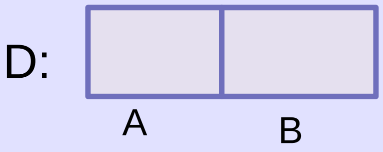
- The converted BCD digits are saved in the memory locations from $1010 (= BCDNum) to $1013.

```
; The memory allocation for the BCD number for the array maximum

            ORG $1010

BCDNum:   DS.B   4
```



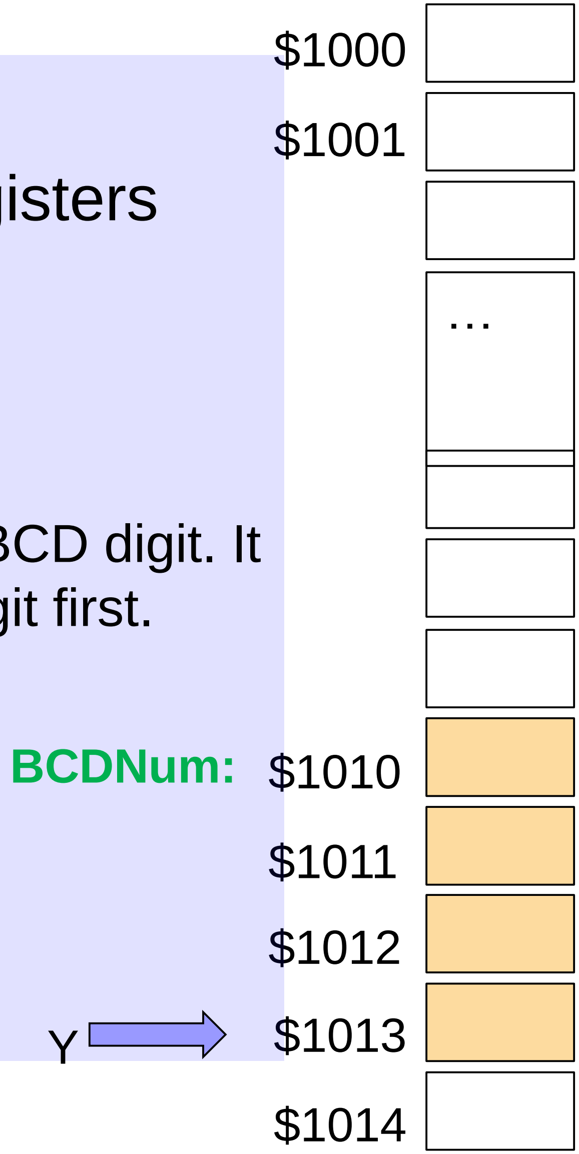
Binary to BCD Conversion

1. Transfer A register into B register,
2. Clear A register. So D= A and B registers



3. Index register Y holds the address of the BCD digit. It saves starting from the least significant digit first.

```
LDY  #BCDNum+3
```



Binary to BCD Conversion

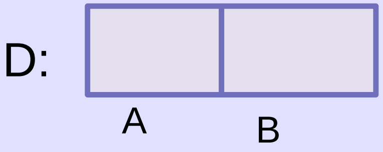
BCDLoop:

i) Assign X register with the value of the Divider

ii) Do the integer division:

```
        IDIV                    ; D/X,   X=Quotient, D=Remainder
```

iii) Store the remainder(in B register) at the address in Y register



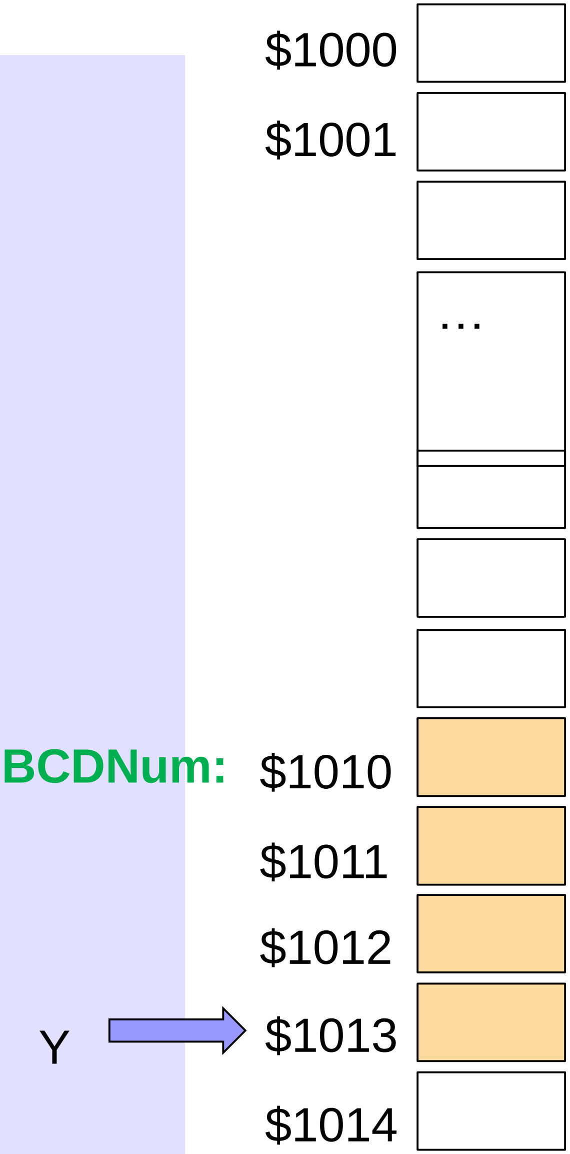
iv) Decrement Y

v) Exchange X register with D register

```
    XGDX
```

vi) If D register is not equal to 0 , then go to **BCDLoop**

vii) Move $00 to **BCDNum.**



PART III 7-SEGMENT LED DISPLAY

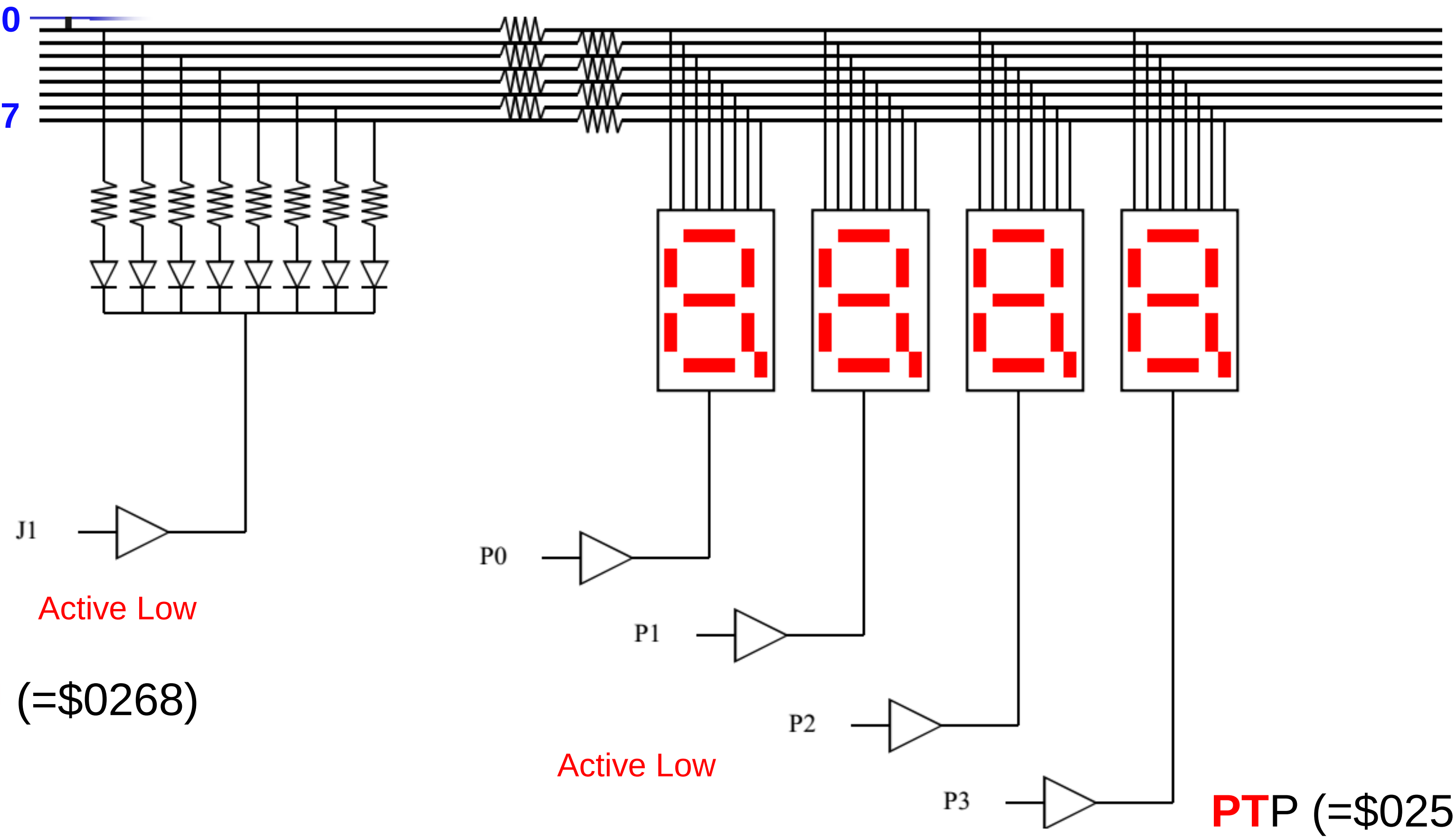
Drangon12plus - LEDs

- **Port B** : All of these LED outputs are wired to **share the Port B pins** for controlling the output on the LEDs.
- **J1 pin (Bit 1)** in port J is **used to enable/disable 8 flashing LEDs**
- **P0** to **P3 pins (Bit 0 to Bit 3) in port P** is used to **select one of 7-segment LEDs**

PORTB
(=$0001)

B0

B7



Active Low

PTJ (=$0268)

Active Low

PTP (=$0258)

7-segment LED Display

A. Defining Ports:

1) First, you need to define the data direction registers for PORT B, PORT P and PORT J.

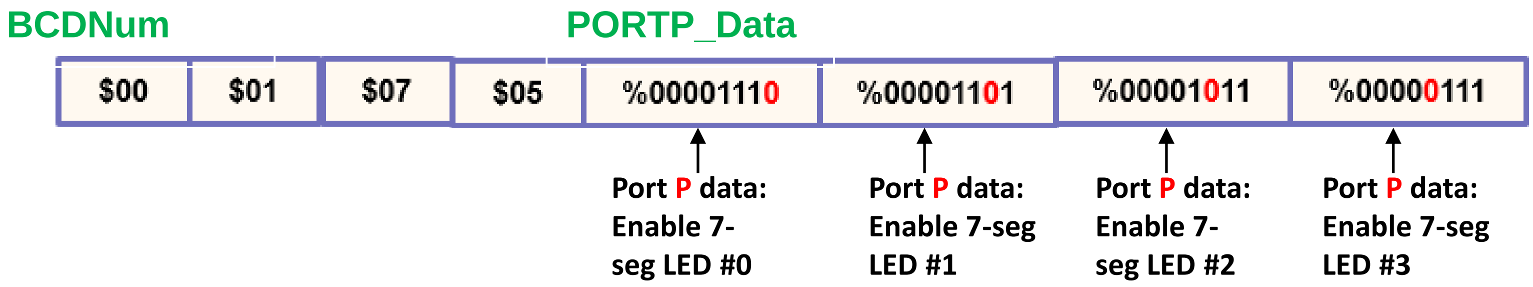
2) Then disable the flashing LEDs by setting a value in PJ1 pin in PORT J.

```
BSET      DDRB,    %11111111            ;configure Port B as a output

 BSET     DDRJ,    %00000010            ;configure PJ1 pin  as a output pin

 BSET     DDRP,    %00001111           ; configure P port as a output port



 ;Disabling PJ1 by setting  the value 1

BSET     PTJ,    %00000010                   ; Disable flashing LEDs
```

7-segment LED Display

- **Data needed to display on the 7-segment LEDs**
- PORT P data to enable the 7-segment LED#0 - LED #3

```
PORTP_Data:  DC.B %00001110, %00001101, %00001011, %00000111
```



BCDNum

PORTP_Data

$00	$01	$07	$05	%00001110	%00001101	%00001011	%00000111

Port P data: Enable 7-seg LED #0

Port P data: Enable 7-seg LED #1

Port P data: Enable 7-seg LED #2

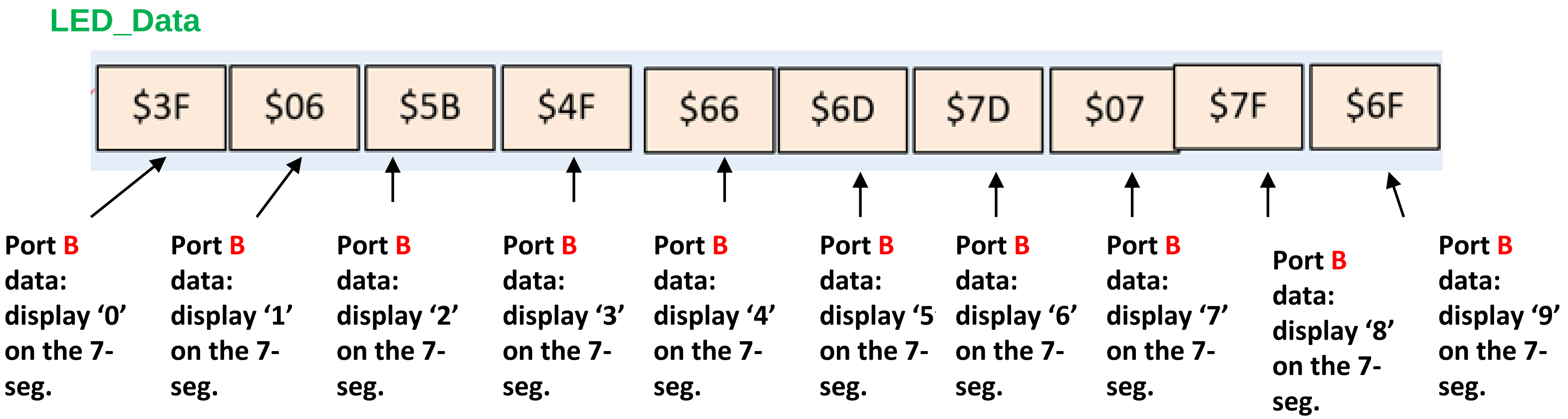
Port P data: Enable 7-seg LED #3

7-segment LED Display

- **Data needed to display on the 7-segment LEDs**

- Port B Data: The LED_Data contains the hex-values for PORT B to represent the digit 0-9.

LED_Data: DC.B $3F, $06, $5B, $4F, $66, $6D, $7D, $07, $7F, $6F

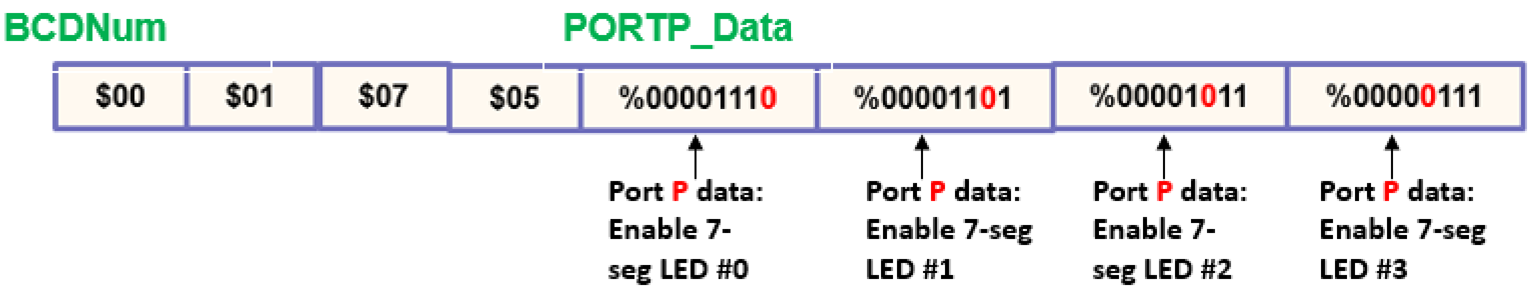


LED_Data

$3F	$06	$5B	$4F	$66	$6D	$7D	$07	$7F	$6F
Port B data: display ‘0’ on the 7-seg.	**Port B data: display ‘1’ on the 7-seg.**	**Port B data: display ‘2’ on the 7-seg.**	**Port B data: display ‘3’ on the 7-seg.**	**Port B data: display ‘4’ on the 7-seg.**	**Port B data: display ‘5 on the 7-seg.**	**Port B data: display ‘6’ on the 7-seg.**	**Port B data: display ‘7’ on the 7-seg.**	**Port B data: display ‘8’ on the 7-seg.**	**Port B data: display ‘9’ on the 7-seg.**

7-segment LED Display

- **Overall Data :to display the digit on the 7-segment LEDs**



BCDNum

PORTP_Data

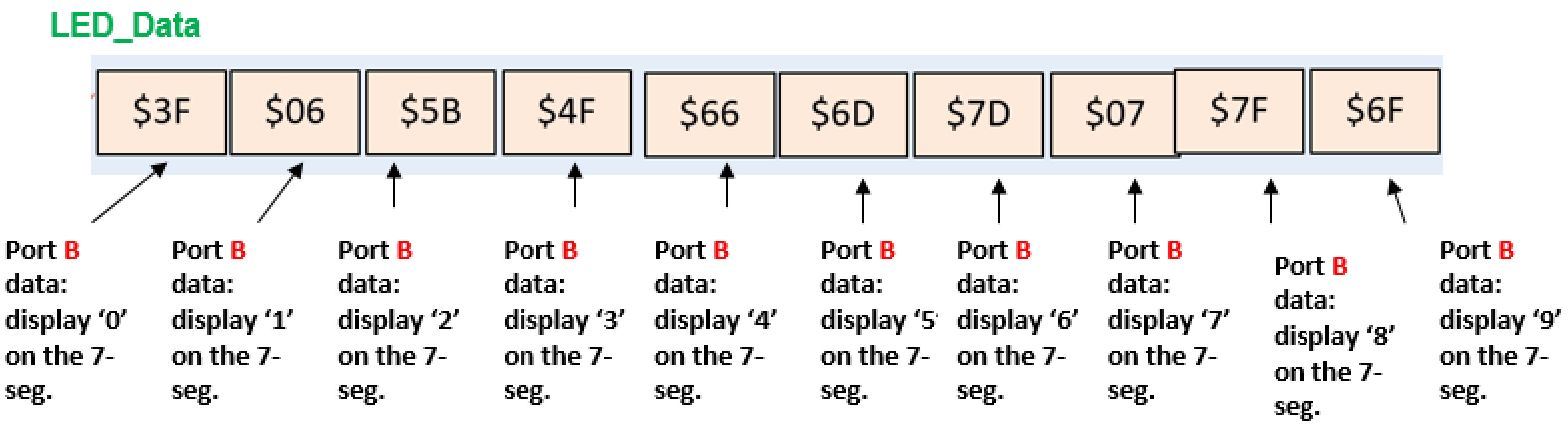
$00	$01	$07	$05	%00001110	%00001101	%00001011	%00000111

Port P data: Enable 7-seg LED #0

Port P data: Enable 7-seg LED #1

Port P data: Enable 7-seg LED #2

Port P data: Enable 7-seg LED #3



LED_Data

$3F	$06	$5B	$4F	$66	$6D	$7D	$07	$7F	$6F

Port B data: display ‘0’ on the 7-seg.

Port B data: display ‘1’ on the 7-seg.

Port B data: display ‘2’ on the 7-seg.

Port B data: display ‘3’ on the 7-seg.

Port B data: display ‘4’ on the 7-seg.

Port B data: display ‘5 on the 7-seg.

Port B data: display ‘6’ on the 7-seg.

Port B data: display ‘7’ on the 7-seg.

Port B data: display ‘8’ on the 7-seg.

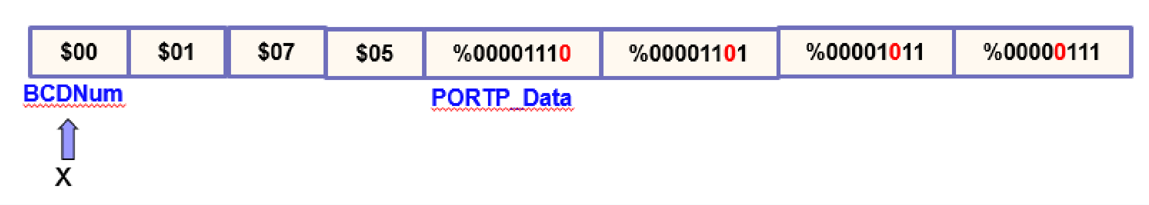
Port B data: display ‘9’ on the 7-seg.

7-segment LED Display

- To display BCD digits on 7-segment LEDs, following steps are required.

i) X register is holding the beginning address of BCD

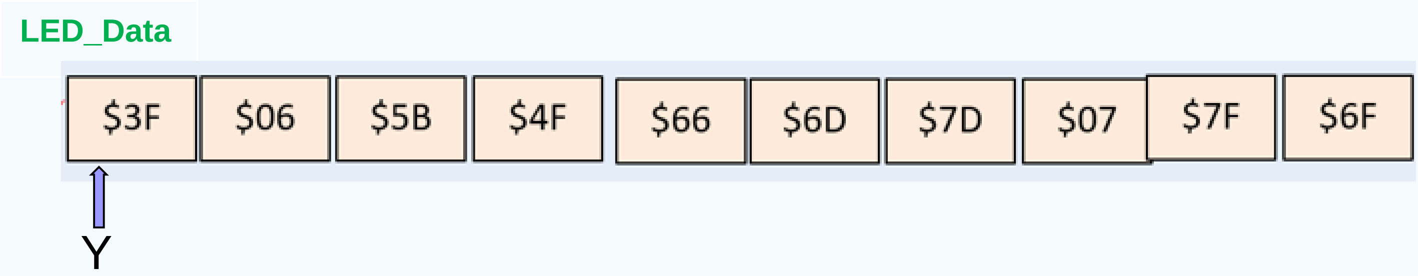
LDX #BCDNum



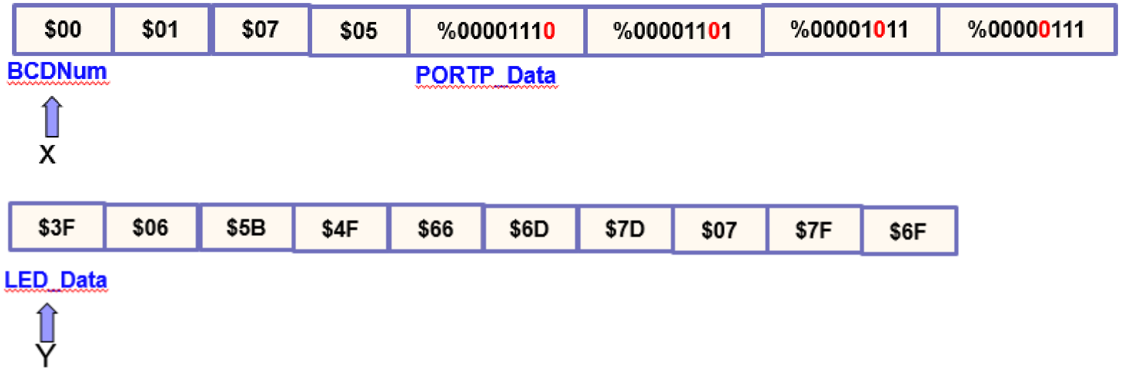
$00	$01	$07	$05	%00001110	%00001101	%00001011	%00000111

ii) Y register is holding the beginning address of **LED_Data**

LDY #LED_Data



$3F	$06	$5B	$4F	$66	$6D	$7D	$07	$7F	$6F



$00	$01	$07	$05	%00001110	%00001101	%00001011	%00000111

BCDNum ← X

PORTP_Data

$3F	$06	$5B	$4F	$66	$6D	$7D	$07	$7F	$6F

LED_Data ← Y

LEDLoop:

i) Load A register from the memory location pointed by X

ex) A= $00

ii) Load B register for the address Y + A ; B register has the data for PORT B

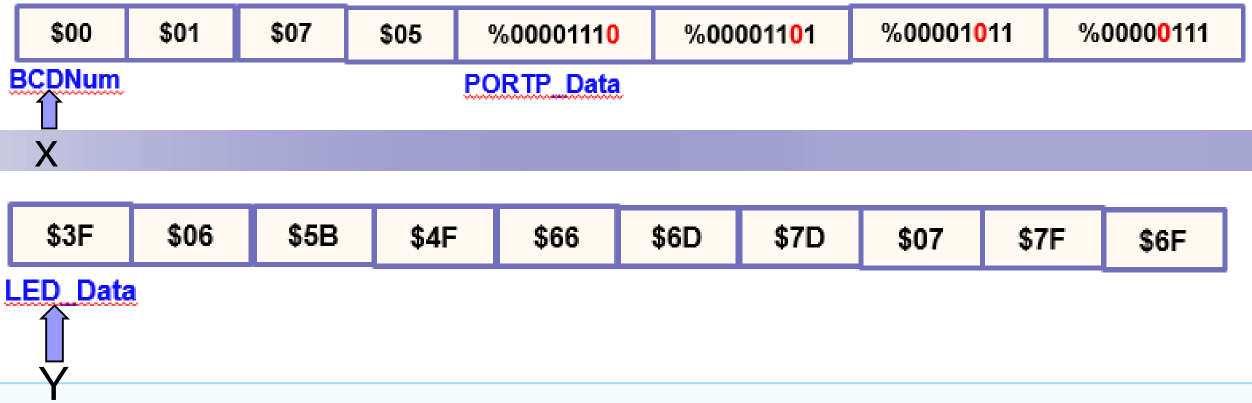
ex) B = $3F

LDAB, A,Y

iii) Save B register to PORT B

IV) The data in memory location X+4 to PORT P.

Move the byte in X+4 to PTP



$00	$01	$07	$05	%00001110	%00001101	%00001011	%00000111

$3F	$06	$5B	$4F	$66	$6D	$7D	$07	$7F	$6F

LEDLoop:

……

cont.

vi) 1 ms delay:

Before assign 1ms to Y register, save Y register into Stack : **PSH**Y
Assign 1ms to Y register
Call the subroutine Delay_yms
Pull Y (restore Y register) : **PUL**Y

vii) Increment X register

viii) Compare X register with #BCDNum+4

viiii) If X is Less than #BCDNum+4, branch **LEDLoop**

else Branch to **Forever**