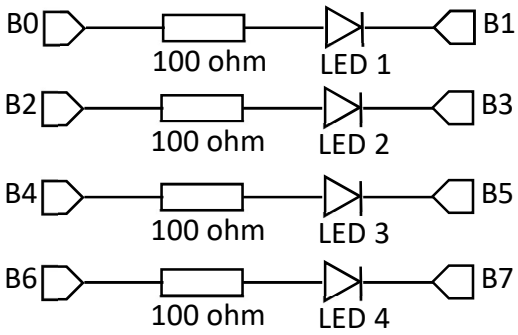
Nom i Cognoms: _____POSSIBLE SOLUCIÓ_____

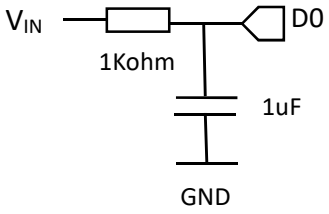
1. Tenim el següent muntatge als Pins del PORTB, i executem el tros de codi de sota. Indica quin o quins LED s'encendran (Els registres ANSELB, TRISB i PORTB es troben al Bank F de memòria). Justifica la resposta. **(1,5 PUNTS)**




```
MOVLB   0Fh        // al bank F hi ha els tres registres involucrats.
CLRF    ANSELB, B
CLRF    TRISB, B
MOVLW   7Ah        Els bits 7, 2 i 0 es posen a 0. Els bits 6,5,4,3,2 a 1. Sols apareix
MOVWF   PORTB, B   diferència de potencial en el sentit del LED entre 6 i 7. Per tant
                   només s'encén el LED 4.
```

2. Tenim connectat al PIN D0 un circuit resistència-condensador, que es carregarà segons l'equació de càrrega del condensador vista a classe (**2,5 PUNTS**):

$$V_c = V_{in} (1 - e^{-t/RC})$$



Si V_{IN} = 5V, V_{DD} = 5V, el condensador està inicialment descarregat i el Pin D0 està configurat com a entrada digital:

- Calcula durant quant temps l'entrada està amb seguretat a "0" lògic.

  Per sota V_{IL} serà segur un 0. Per tant: $0{,}8V = 5V (1 - e^{-t/\,0{,}001})$
  Aillem l'exponencial i fent logaritme a les dues bandes ens dóna:
  t= 174 us.

- Calcula a partir de quin instant de temps l'entrada a D0 serà amb seguretat un "1" lògic.

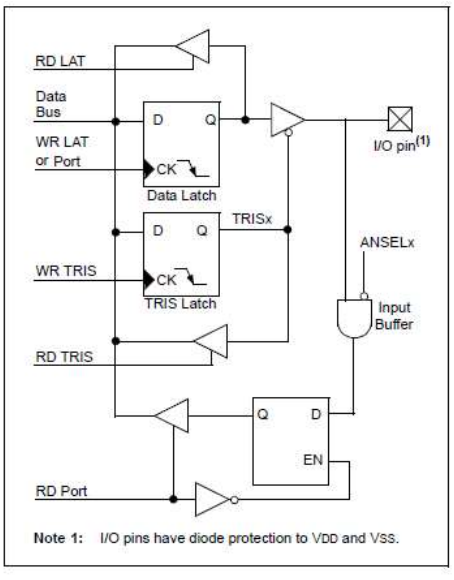
  A partir de que arribem a V_{IH} serà segur un 1. Per tant: $2V = 5V (1 - e^{-t/\,0{,}001})$
  Aillem l'exponencial i fent logaritme a les dues bandes ens dóna:
  t= 511 us.

- Què ha passat entremig amb el valor lògic de l'entrada digital?
  Entre t = 174 us i t= 511us no es pot garantir si l'entrada caurà a l'estat 0 o a l'estat 1 (però serà una de les dues, és un sistema digital).

27.8 DC Characteristics: Input/Output Characteristics, PIC18(L)F2X/4XK22

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
		I/O PORT:					
D140		with TTL buffer	—	—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D140A			—	—	0.15 VDD	V	1.8V ≤ VDD ≤ 4.5V
D141		with Schmitt Trigger buffer	—	—	0.2 VDD	V	2.0V ≤ VDD ≤ 5.5V
		with I2C levels	—	—	0.3 VDD	V	
		with SMBus levels	—	—	0.8	V	2.7V ≤ VDD ≤ 5.5V
D142		MCLR, OSC1 (RC mode)(1)	—	—	0.2 VDD	V	
D142A		OSC1 (HS mode)	—	—	0.3 VDD	V	
	VIH	**Input High Voltage**					
		I/O ports:		—	—		
D147		with TTL buffer	2.0	—	—	V	4.5V ≤ VDD ≤ 5.5V
D147A			0.25 VDD + 0.8	—	—	V	1.8V ≤ VDD ≤ 4.5V
D148		with Schmitt Trigger buffer	0.8 VDD	—	—	V	2.0V ≤ VDD ≤ 5.5V

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D159	VOL	**Output Low Voltage** I/O ports	—	—	0.6	V	IOL = 8 mA, VDD = 5V IOL = 6 mA, VDD = 3.3V IOL = 1.8 mA, VDD = 1.8V
D161	VOH	**Output High Voltage(3)** I/O ports	VDD - 0.7	—	—	V	IOH = 3.5 mA, VDD = 5V IOH = 3 mA, VDD = 3.3V IOH = 1 mA, VDD = 1.8V



Note 1: I/O pins have diode protection to VDD and VSS.

REGISTER 10-8: TRISx: PORTx TRI-STATE REGISTER(1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISx7	TRISx6	TRISx5	TRISx4	TRISx3	TRISx2	TRISx1	TRISx0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **TRISx<7:0>:** PORTx Tri-State Control bit
1 = PORTx pin configured as an input (tri-stated)
0 = PORTx pin configured as an output

Note 1: Register description for TRISA, TRISB, TRISC and TRISD.

Nom i Cognoms: _____POSSIBLE SOLUCIÓ_____

3. Omple la taula amb els valors resultants dels registres després d'executar aquest codi: **(1 PUNT)**

```
Dada1    equ  05h
Dada2    equ  019h
CLRF     00, A
MOVLW    Dada1
MOVWF    01, A
BTFSC    01, 0, A   //aquest mira 001h
INCF     00, F, A
BTFSC    00, 1, A   //aquests no!!
INCF     00, F, A
BTFSC    00, 2, A
INCF     00, F, A
BTFSC    00, 3, A
INCF     00, F, A
```

	Valor inicial	Valor final
WREG	AAh	05h
BSR	02h	02h
000h	0Eh	01h
001h	01h	05h
002h	45h	45h
018h	89h	89h
019h	4Ah	4Ah
020h	32h	32h

4. Quants bytes ocuparà el programa a la memòria de programa? **(0,5 PUNTS)**

Tenim 11 instruccions single-word, per tant el codi ocupa 22 bytes a memòria.

5. Calcula el temps que triga a executar-se la següent funció, des de que entra fins que retorna al codi que la crida (recordeu: RETURN 2 cicles, single WORD; BRA 2 CICLES, single WORD) amb un Fosc=20 MHz **(2 PUNTS)**

```
CopiaBank:    LFSR     000h, FSR0
              LFSR     100h, FSR0
    Etiq:     MOVF     POSTINC0, W
              MOVWF    POSTINC1
              BTFSC    FSR0H, 0
              RETURN
              BRA      Etiq
```

Si ens adonem que el segon LFSR posa al punter 0 el valor 100h (256), veiem que mai s'executa el bucle ja que el BTFSC no fa skip, per tant el codi triga:

El registre FSR0 queda escrit amb el valor 100h (256) per tant el BTFSC fallarà a la primera i hi haurà Return!!
2(LFSR) + 2(LFSR) + 1(MOVF) + 1(MOVWF) + 1(BTFSC) + 2(RETURN) = 9 cicles.

Cada cicles d'instrucció són 4 cicles de clock i un cicle de clock és 1/20MHZ = 50ns.
Per tant el temps total és: 9x4x50ns= 1,8 us

(Si interpretem que es copiarà el bank 0 al bank 1 i que per tant el bucle fa 255 iteracions fins la última on no fa skip tindrem:
2+2+255(1+1+2+2) + 1+1+1+2 = 1539 cicles = 307,8 us.)

TABLE 25-2: PIC18(L)F2X/4XK22 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word to FSR(f) 1st word	2	1110 1111	1110 0000	00ff kkkk	kkkk kkkk	None	
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)

Nom i Cognoms: _____POSSIBLE SOLUCIÓ_____

6. Completa el codi per garantir que si hi ha un canvi al PORTB es saltarà a una interrupció de baixa prioritat (RSI_LOW). **(1,5 PUNTS)**

```
Org 000h
        GOTO Codi
Org 008h
        GOTO RSI_HIGH
Org 018h
        GOTO RSI_LOW

Codi:
        MOVLB   0Fh         // Triem el Bank F de registres (és on
        CLRF    ANSELB,B    // hi ha ANSELB i TRISB)
        SETF    TRISB,B     // Posem PORTB com a entrada digital

        BCF   INTCON2, 0, B // prioritat baixa
        BCF   INTCON , 0, B // flag a 0 inicialment
        BSF   INTCON , 0, B // prioritat baixa
        BSF   INTCON , 3, B // habilitem IE del port B
        BSF   INTCON , 6, B // habilitem interrupcions de baixa prio
        BSF   INTCON , 7, B // habilitem (sempre al final) les de alta

Final:  BRA     Final       // El programa acaba aquí
        End
```

7. Hi ha algun error en el codi de la interrupció RSI_LOW que tenim aquí sota?, explica'l: **(1 punt)**

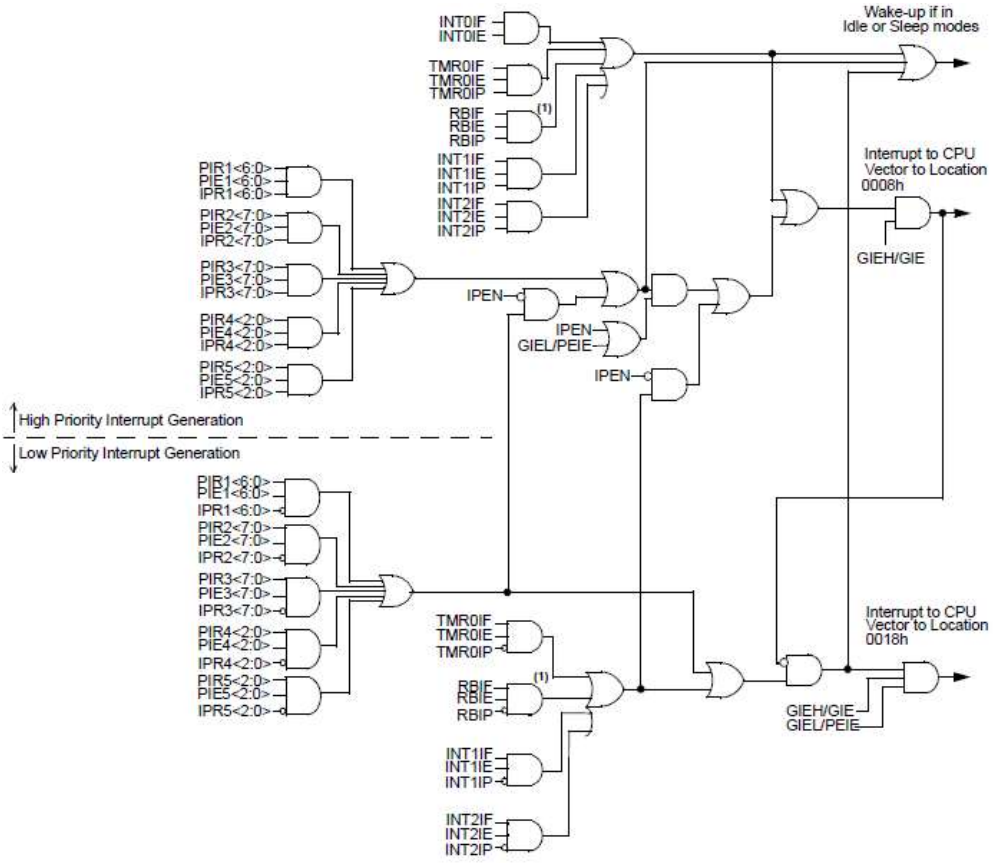
```
RSI_LOW:

        BTFSC   INTCON, 0, A    // Mirem si el flag és del PORTB
        RETFIE FAST
        INCF    Counter, F, A   // Comptem la interrupció
        BCF     INTCON, 0, A    //Borrem el flag perquè no reentri
        RETFIE FAST             // Sortim de la interrupció
```

MAI podem posar un RETFIE FAST en una interrupció de baixa prioritat!!

TABLE 9-1: REGISTERS ASSOCIATED WITH INTERRUPTS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—
IPR1	—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	CTMUIP	TMR5GIP	TMR3GIP	TMR1GIP
IPR4	—	—	—	—	—	CCP5IP	CCP4IP	CCP3IP
IPR5	—	—	—	—	—	TMR6IP	TMR5IP	TMR4IP
PIE1	—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	CTMUIE	TMR5GIE	TMR3GIE	TMR1GIE
PIE4	—	—	—	—	—	CCP5IE	CCP4IE	CCP3IE
PIE5	—	—	—	—	—	TMR6IE	TMR5IE	TMR4IE
PIR1	—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	CTMUIF	TMR5GIF	TMR3GIF	TMR1GIF
PIR4	—	—	—	—	—	CCP5IF	CCP4IF	CCP3IF
PIR5	—	—	—	—	—	TMR6IF	TMR5IF	TMR4IF
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RCON	IPEN	SBOREN	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for Interrupts.

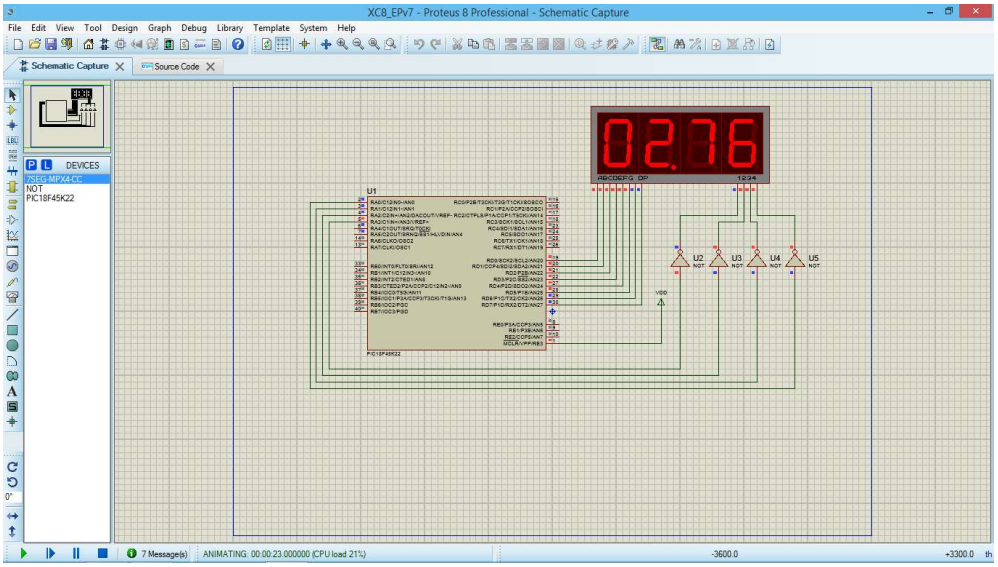


Nom i Cognoms: _____POSSIBLE SOLUCIÓ_____

PREGUNTA CONTROL DE LABORATORI

(els repetidors amb LAB convalidat no l'heu de fer)

A la pràctica de laboratori del display 7 segments (veure figura), heu fet una funció per pintar un nombre amb decimals a la pantalla. Indica si són certes (**C**) o falses (**F**) les afirmacions següents: **(+1,25 encert, -0,75 fallada)**



F El punt decimal s'ha de posar sempre i en cas que el nombre sigui enter s'ha de posar després de la última xifra.

C Es pot fer (canviant el codi) que uns dígits brillin més que altres.

F Aquesta pràctica va ser la última feta en Assembler del PIC18F45K22.

C Si volem fer sortir un dígit pels quatre displays alhora, podem treure els seus bits pel PORTD i activar 4 uns al PORTA.

F Si estem pintant el nombre '2019' i parem el programa per debug o aquest acaba amb un "while (1);", quedarà pintat el nombre a la pantalla.

F Si un cop tenim el codi acabat canviem l'oscil.lador del xip de 8MHz a 16 MHz utilitzarem la meitat de dígits per representar el mateix nombre.

F A la placa de laboratori (circuit real) també hi ha portes NOT al control de l'activació dels 7-segments.

C Es podria dedicar el primer 7-segments (per software) a representar el signe del nombre i així pintar negatius també.