Nom i Cognoms: _____Possible solució_____

1. Tenim el següent codi fet per un estudiant d'una altra facultat, que pretén posar a 1 el bit 0 del PORTA. Utilitza un PIC18F45K22 amb un oscil·lador extern de 12MHz. **(1 PUNT)**

```
MOVLB  0Fh
BCF    ANSELA, 0, 1
BCF    TRISA, 0, 1
BSF    LATA, 0, 1
```

Durant quant de temps no té controlada la sortida al pin A0? Justifica-ho!

Inicialment el TRIS està tot a 1, per tant configurat com a entrada (veure la descripció del registre 10-8). És lògic; no pot arrencar i treure "coses" a l'atzar. Anem a veure les instruccions:

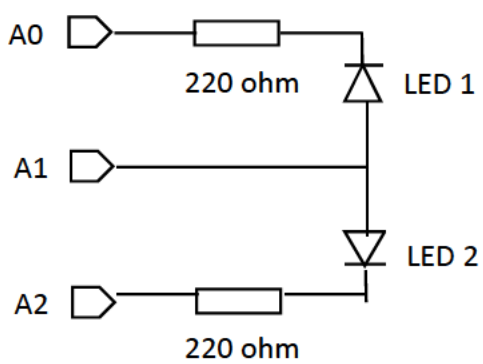
```
MOVLB   0Fh          F  E          (carrega el BSR a 'f')
BCF     ANSELA, 0, 1    F  E       (posa el pin a digital (si fos input)
BCF     TRISA, 0, 1        F  E    (posa el pin en mode sortida)
BSF     LATA, 0, 1            F  E (posa la sortida a 1)
```

La instrucció 3 posa el pin com a sortida, sense saber què hi ha carregat al LAT, per tant durant un cicle d'instrucció no controlem la sortida. 1 cicle = 4 * 1/12 MHz = 333ns
Seria millor fer primer BSF LATA,0,1 i després BCF TRISA,0,1

2. Indica si són certes(C) o falses(F) les afirmacions següents.**(2 PUNTS (+0.5 encert/-0.5 fallada))**

- ☐ La tensió de sortida a nivell alt és independent de la tensió a la que alimentem el xip (dins del rang de funcionament). **FALS**
- ☐ La tensió de sortida a nivell baix és independent de la tensió a la que alimentem el xip (dins del rang de funcionament). **CERT**
- ☐ Una tensió de 1.5V sempre serà llegida com un '1' lògic per les entrades digitals del microcontrolador. **FALS**
- ☐ El marge de soroll a nivell baix és independent de la tensió d'alimentació del xip (VDD). **FALS**

3. Escriu el codi necessari perquè s'encenguin els dos LEDs de la figura. Suposeu que el micro acaba d'arrancar. Programeu en C. **(1 PUNT)**



Pot ser tan fàcil com:
ANSELA = 0x00;
TRISA = 0x00;
LATA (o PORTA) = 0x02;

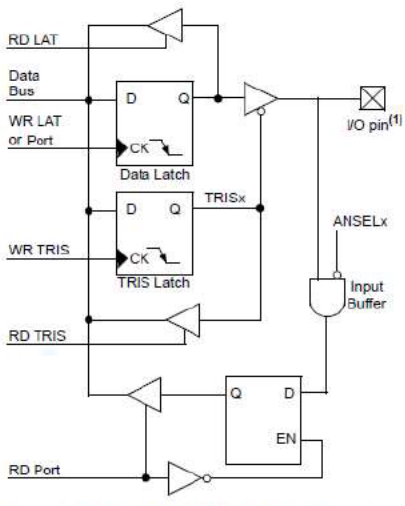
També valdría amb màscares o tocant els bits un a un.

4. Podem garantir que els dos LEDs s'encenen simultàniament? Per què? **(1 PUNT)**
Ho garantirem si, un cop passat a codi màquina es copia al LAT un valor de 8 bits de cop (ex. MOVLW 02h; MOVWF LATA,A) si toquem els pins bit a bit, no.

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		$\overline{MCLR}$, 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
		with I²C levels	0.7 VDD	—	—	V	
		with SMBus levels	2.1	—	—	V	2.7V ≤ VDD ≤ 5.5V
D149		$\overline{MCLR}$	0.8 VDD	—	—	V	
D150A		OSC1 (HS mode)	0.7 VDD	—	—	V	
D150B		OSC1 (RC mode)(1)	0.9 VDD	—	—	V	

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.

Note 2: -n = Value at Power-On for the bit.

Nom i Cognoms: _____Possible solució_____

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

```
CLRF    00, 1       ; borra 200h
CLRF    01, 1       ; borra  201h
MOVLW   04          ; W =4
MOVLB   00          ; BSR = 0
DECF    00, 1, 1    ; 000h = 0Dh
DECF    01, 0, 1    ; W = (001h) – 1 = 00
MOVLB   01          ; BSR = 1
ADDWF   01, 1, 1    ; (101h) = (101h) + 0
MOVFF   201h, 200h  ; mou 00 a 200h
MOVFF   201h, 101h  ; mou 00 a 101h
```

	Valor inicial	Valor final
WREG	12h	00h
BSR	02h	01h
000h	0Eh	0Dh
001h	01h	01h
100h	01h	01h
101h	89h	00h
200h	0Ah	00h
201h	23h	00h

6. Quants bytes ocuparà el programa a la memòria de programa? **(0,5 PUNTS)**
Són 8 instruccions single word i 2 double word; 8x2 + 2x4 = 24 bytes.

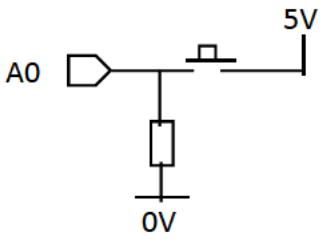
7. Quant temps triga a executar-se el codi de Espera si Fosc= 8 MHz? Justifica la resposta. **(1,5 PUNTS)**

```
Espera:   MOVLW   5         W=5
Etiq:     MOVWF   00h, 0    00h de l'access=W (5 el primer cop)
          DECFSZ  00h, 0, 0 W = (00h) -1 -> guarda al W 4,3,2,1...
          BRA     Etiq      Quatre cops salta a Etiq
          RETURN            El cinquè cop fa skip de BRA i RETURN
```

1 cop MOVLW(1)(2 si teniu en compte el fetch)+**4 cops**(MOVWF(1)+DECFSZ(1)+BRA(2)) + 1 cop MOVWF(1)+1 cop DECFSZ(2 perquè fa skip de BRA)+ 1 cop RETURN(2) = 22/23 cicles = 11/11.5 us.

8. Si al final del següent tros de programa (suposeu que A0 ja està configurat com a entrada digital) que llegeix l'estat d'un botó connectat a A0: **(2 PUNTS)**

```
          ...
          CLRF    00h
Wait:     BTFSS   PORTA, 0, 0
          BRA     Wait
Count:    INCF    00h, 1, 0
          BTFSC   PORTA, 0, 0
          BRA     Count
          ...
```



La posició de memòria 00h de l'*access bank* val 17, podem saber durant quant temps ha estat premut el botó? Justifica la resposta! (Fosc = 8MHz)

El primer bucle (Wait) espera a que premin el botó. El segon fa iteracions mentre no el deixin anar, tot incrementant una variable. **NO** podem saber el temps per:
PROBLEMA 1 (el és greu des del punt de vista informàtic). Com que la variable està en un registre de 8 bits, no sabem si ha fet 17, 256+17, 512+17,... iteracions.
PROBLEMA 2: Imprecisió. Tant un bucle com l'altre miren l'estat del botó en un cicle de 3 o 4 instruccions. Això ens genera una imprecisió 1.5 o 2 us a cada un.
PROBLEMA EXTRA: rebots. Si tenim rebots en el moment de prémer tampoc anirà bé.

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)