Computer Interfacing (CI) FIB. Primer parcial **06/04/2022**

Cognoms, Nom _____ **DNI** _____

Tota resposta sense justificar es considerarà nul·la !

P1. (1,5 punts)

1.1 (1 punt) Calcula quant triga el següent codi a executar-se si es fa servir una Fosc=8MHz

```
mistery_routine
      valor1 equ 000
      valor2 equ 0x1FF

      LFSR FSR0, valor1 ; 2 cicles
      LFSR FSR1, valor2 ; 2 cicles
acaba
      MOVFF POSTINC0, POSTDEC1 ; 2 cicles
      BTFSS FSR0H, 0, A ; 1 cicle si no salta, 2 cicles si salta (3 si la següent
      instrucció fos de double word)
      BRA acaba ; 2 cicles
```

Un anàlisi de l'execució es pot trobar en la següent taula

Inst/Cicle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...	1280	1281	1282	1283
LFSR FSR0	F	E	E																
LFSR FSR1			F	E	E														
MOVFF					F	E	E												
BTFSS							F	E	No salta										
BRA								F	E	E									
(INST)									F	-									
MOVFF										F	E	E							
BTFSS												F	E	No salta					
BRA													F	E	E				
[...]														F	-	253 iteracions més fins que FSR0H[bit0]=1			
MOVFF															F	E	E		
BTFSS																	F	E	E SALTA
BRA																		F	-

De forma alternativa podem comptar:
FSR(2)+FSR(2)+255*(MOVFF(2)+BTFSS(1)+BRA(2))+MOVFF(2)+BTFSS(2)=1283 cicles
1283 cicles *(4/8MHz)=641,5us

1.2 (0,5 punts) Indica, de forma breu, què fa aquest codi i quant ocupa a memòria de programa
Aquest codi el que fa és copiar el Bank0 de memòria de dades al Bank1, però de forma inversa:
[000] -> [1FF]
[001] -> [1FE]
[..]
[0FF] -> [100]

En total ocupa a memòria de programa 8 words:
LFSR(x2) 2w, MOVFF 2w, BTFSS 1w, BRA 1w

P2. (2 punts)

Indica si són certes (C) o falses (F) les afirmacions següents (cada encert suma 0.2 / cada error resta 0.2)

F Utilitzem la instrucció RETFIE FAST per a sortir de les RSI de baixa prioritat. Les d'alta prioritat han d'afegir un codi en assembler per a guardar/restaurar el WREG, STATUS i BSR en variables de memòria RAM, i sortir usant RETFIE.

F El flag de petició d'interrupció IF no s'activa si aquesta està deshabilitada mitjançant el bit d'enable IE.

C L'avantatge d'una arquitectura Harvard enfront d'una Von Neumann és que permet la concurrència en l'accés a dades i instruccions.

C Totes les instruccions del PIC18F45k22 ocupen 16 bits, excepte GOTO, LFSR, MOVFF i CALL.

C Accedint a la memòria de dades, si ho fem en mode banked podem accedir a més posicions de memòria que si ho fem en mode Access Bank.

C "CLRF 00, A" i "CLRF 00, B" accedeixen a la mateixa posició de memòria si BSR=0.

F El registre ANSELA no es troba a l'Access Bank. Per accedir-hi ho fem en mode banked i BSR=0.

C El byte menys significatiu d'una instrucció, s'emmagatzema sempre en una adreça parell de la memòria de programa.

C Usant un clock de CPU de 4MHz, el temps d'execució (en microsegons) d'una instrucció que provoca un salt de tipus skip i precedeix a una instrucció 'double word' serà de 3µs.

F Podem triar la prioritat de la interrupció externa INT0 fent servir el bit INT0IP.

Computer Interfacing (CI) FIB. Primer parcial **06/04/2022**

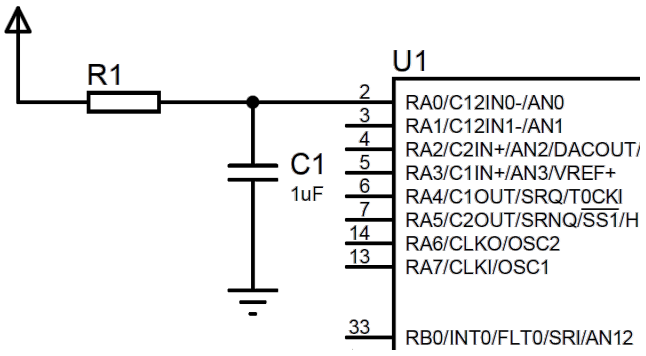
Cognoms, Nom _____ **DNI** _____

Tota resposta sense justificar es considerarà nul·la !

P3. (1,5 punts)

Tenim connectat al PIN RA0 un circuit resistència-condensador que es carregarà segons l'equació de càrrega del condensador vista al laboratori

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



Suposant que Vin són 5V i que el condensador estigui descarregat a temps 0, quina R1 triaries si volguessis assegurar que l'entrada es troba a 0 lògic com a mínim 100 milisegons.

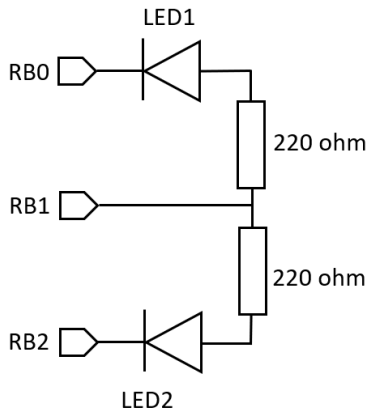
Sabem que el pin es llegirà com a 0 sempre que el valor a l'entrada sigui més petit que $V_{IL\ MAX}$. A les condicions d'V_{IN}=5V, $V_{IL\ MAX}$=0,8V.

$$0{,}8 = 5\,(1 - e^{-0{,}1/R*1u})$$

Aïllant R trobem que la resistència ha de ser més gran que 573,5KΩ

P4. (1,5 punts)

Escriu el codi necessari en ensamblador del PIC18 per aconseguir encendre el LED1 i mantenir el LED2 apagat.



Per tal d'encendre el LED1 haurem de posar més tensió a RB1 que a RB0: per tant a RB1 posarem "1 lògic" i a RB0 posarem "0 lògic". Per tal de posar un valor digital a un pin, l'hem de configurar com a Output!

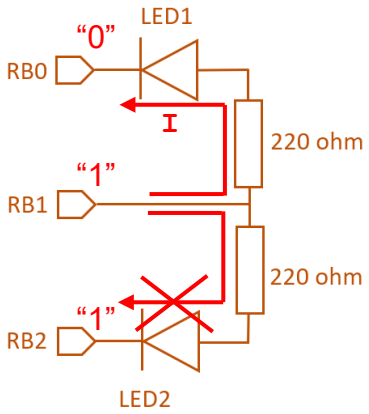
Per mantenir el LED2 apagat, tenim dues possibilitats:

1) podem fixar el valor del pin RB2 a "1 lògic", configurant el pin a Output. D'aquesta manera no hi ha diferència de potencial entre RB1 i RB2.

2) Podem configurar el pin RB2 com a Input. D'aquesta manera el micro no hi estableix cap voltatge, i el pin no interacciona amb el circuit connectat (*pin en alta impedància*)

El següent codi mostra una possible solució fent que el RB2 sigui un Output que tregui "1 lògic". Hem de recordar que per treballar amb els pins com a digitals, cal ajustar els bits corresponents del registre ANSELx (bits a 0 per a que els pins siguin Digitals). Els registres ANSELx no estàn al Access Bank.

La solució mostra diferents formes d'establir el valor als bits d'un registre.



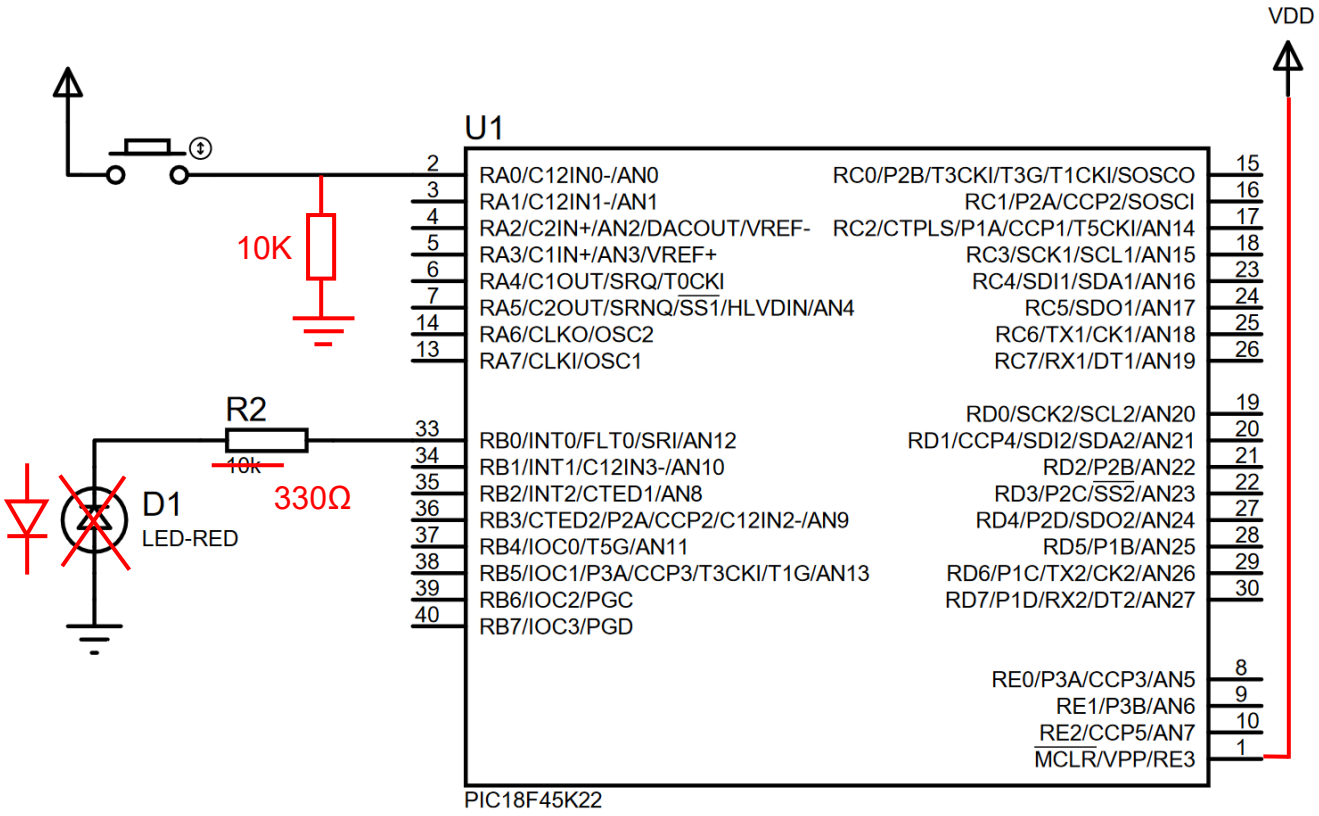

```
MOVLB 0x0F          ; BSR = 0x0F
CLRF ANSELB, B      ; posem a 0 tots els bits de ANSELB (pins digitals)
                    ; B= banked access (posem BSR=0x0F abans)

MOVLW 0x00          ; WREG = 0
MOVWF TRISB, A      ; posem a 0 tots els bits de TRISB (pins a Output)

BCF LATB, 0, A      ; LATB0 = 0
BSF LATB, 1, A      ; LATB1 = 1
BSF LATB, 2, A      ; LATB2 = 1
```

P5. (1,5 punts)

Un estudiant ha entregat un esquemàtic d'un projecte Proteus on vol encendre un led en funció de l'estat d'un botó. Si el botó está premut el led s'ha d'encendre, si el botó no està premut el led ha d'estar apagat. Tot i copiar el codi de github, el projecte no funciona. Indica tots els errors que trobes en l'esquemàtic



1) Cal connectar el símbol de VDD al pin 1 (/MCLR) per tal d'evitar que el micro es resetegi aleatòriament. La funció de reset (MCLR= Master Clear) funciona amb lògica negada, per tant hem de posar un "1 lògic" per a que no es reinicií.

2) Al circuit del polsador cal posar una resistència de pull-down, per tal que es llegeixi un 0 quan el botó no està premut. Altrament, el pin quedaria a l'aire.

3) Per tal que s'encengui el led, l'ànode ha d'estar a una tensió més gran que el càtode. Al diagrama, l'ànode està a Ground (0V), per tant serà impossible col·locar al càtode una tensió més petita. Una possible solució és girar el Led (tal com es veu a la solució proposada al dibuix).

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

Una altra solució seria connectar l'ànode a VDD en comptes de a 0V: això requeriria encendre el led amb lògica negada, és a dir el pin RB0 hauria de treure un "0" per encendre'l, i un "1" per apagar-lo.

4) El valor de 10K (que proposa Proteus per defecte) és massa gran i provoca que la corrent que circula sigui massa petita com per a encendre el led de la simulació. Valors adequats poden ser 470Ω, 330Ω, o propers.

P6. (2 punts)

Volem programar una sèrie d'interrupcions i hem trobat el següent codi a una coneguda web d'ajuda per a estudiants. Al copiar-ho han quedat espais en blanc. Completa els espais emmarcats amb el codi necessari per a que funcioni bé el programa sense modificar el codi que sí que s'ha copiat bé.

```
void interrupt low_priority lowRSI()
{
   if (TMR0IF==1 && TMR0IE==1)
   {
      tractament_T0(); // doneu per fet que aquesta
                       // subrutina esta ben programada
      TMR0IF=0;
   }
}
void interrupt highRSI()
{
   if (ADIF==1 && ADIE==1)
   {
      tractament_AD(); // doneu per fet que aquesta
                       // subrutina esta ben programada
      ADIF=0;
   }

}

void main(void)
{
   iniTimer();  // doneu per fet que aquestes
   iniAD();     // subrutines estan ben programades


   //inicialitzacio interrupcions
```

```
// Sempre cal fer el següent: habilitar cada font d'interrupció involucrada, configurar prioritats...
// ... i activar els enables globals necessaris.
// A la solució següent veureu diferents formes d'accedir als bits dels registres en C.

INTCONbits.TMR0IF=0;        // opcional: podem netejar el Flag per si hagués succeït interrupció prèvia
INTCON2bits.TMR0IP=0;       // cal posar la prioritat a 0 (low priority), doncs el TMR0 es tracta a lowRSI
INTCONbits.TMR0IE=1;        // habilitem font d'interrupció TMR0

ADIF=0;         // opcional: podem netejar el Flag per si hagués succeït interrupció prèvia
ADIP=1;         // cal posar la prioritat a 1 (high priority), doncs el AD es tracta a highRSI
ADIE=1;         // habilitem font d'interrupció AD

RCON = RCON | 0x80;         // IPEN=1. Habilitem nivells de prioritat, doncs tenim les 2 RSI programades
INTCON = INTCON | 0x80;     // GIEH=1. Global Enable de les interrupcions d'alta prioritat
INTCON = INTCON | 0x40;     // GIEL=1. Global Enable de les interrupcions de baixa prioritat
```

```
   while (1);

}
```