Computer Interfacing (CI) FIB. Primer parcial **04/11/2021**

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

P1. (2,5 punts)

Tenim la següent rutina que calcula el mínim d'un vector de 256 elements guardat al banc 2 de memòria del PIC deixant el resultat al WREG.

```
            vector_ad   EQU  200h
r_min:      SETF        WREG,A
            LFSR        FSR0, vector_ad
next:       CPFSGT      INDF0,A      ; els modes indirectes van amb Access Bank o es pot
            MOVF        INDF0,W,A    ; deixar en blanc perquè és l'opció per defecte.
            INCFSZ      FSR0L,F,A
            BRA         next
            RETURN
```

1.1 Calcula el temps d'execució de la rutina desde que se li fa la crida (CALL r_min, fins que torna al principal). Fosc=12Mhz.

A la taula veiem l'anàlisi de l'execució des del CALL fins que es torna de la funció rmin.

Instr / Cicle		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...	1281	1282	1283	1284	1285	1286
(pre CALL)	E																..						
CALL rmin	F	**E**															..						
(post CALL)		F	-														..						
SETF			F	E													..						
LFSR				F	F	E											..						
CPFSGT						F	E										..						
MOVF o SKIP							F	?									..						
INCFSZ								F	E	FSR0L 0 a 1							..						
BRA									F	E							..						
(return)										F	-						..						
CPFSGT											F	E					..						
MOVF o SKIP												F	?				..						
INCFSZ													F	E	FSR0L 1 a 2								
BRA														F	E		..						
(return)															F	-	..						
.. *iteracions*..																	..						
CPFSGT																		E					
MOVF o SKIP																		F	E				
INCFSZ																			F	E	FSR0L FF a 0		
BRA																				F	-		
RETURN																					F	E	
(post RETURN)																						F	-
(post CALL)																							F

Hem comptat 1286 cicles d'instrucció des del CALL fins que es retorna (s'accepten interpretacions + - 1 o 2).
t = 1286 cicles x (4 tics rellotge / 1 cicle) x (1 tic rellotge / $12x10^6$ Hz) = 428.6 us (microsegons)

Alternativament podem comptar:

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

sef(1)+ lfsr(2)+ 255 (cpfsgt(1 o 2)+movf(1 o 0)+incfsz(1)+bra(2)) + 1(cpfsgt(1 o 2)+movf(1 o 0)+incfsz(2)+return(2)= 3+255(5)+1(6)=1284, amb els 2 d'entrar a la funció, 1286.

1.2 El temps d'execució és independent del contingut del vector de 256 elements? Si no ho és, indica el temps d'execució en el cas millor i pitjor.

Si, és independent.
Suposem que el nombre de la posició apuntada per INDF0 és menor o igual que WREG.

Instrucció / cicle	x	x+1	x+2	x+3
CPFSGT INDF0,A (compara W amb on apunta INDF0, no fa skip	F	E		
MOVF INDF0,W,A (actualitza W amb el nou mínim)		F	E	
INCFSZ FSR0L,F,A (incrementa el punter 0 al següent)			F	E
...				F

Suposem ara que el nombre de la posició apuntada per INDF0 és major que WREG.

Instrucció / cicle	x	x+1	x+2	x+3
CPFSGT INDF0,A (compara W amb on apunta INDF0, farà SKIP	F	E		
MOVF INDF0,W,A (se li havia fet FETCH, no fa EXE per l'SKIP)		F	-	
INCFSZ FSR0L,F,A (incrementa el punter 0 al següent)			F	E
...				F

1.3 Quants Bytes ocuparà la rutina a la memòria de programa?

```
            vector_ad   EQU  200h          no ocupa codi, defineix una etiqueta
r_min:      SETF        WREG,A            1 instrucció single word (2B)
            LFSR        FSR0, vector_ad   1 instrucció double word (4B)
next:       CPFSGT      INDF0,A           1 instrucció single word (2B)
            MOVF        INDF0,W,A         1 instrucció single word (2B)
            INCFSZ      FSR0L,F,A         1 instrucció single word (2B)
            BRA         next              1 instrucció single word (2B)
            RETURN                        1 instrucció single word (2B)
```

En total la rutina ocupa 16 Bytes a memòria de programa.

P2. (1 punt)

Indica maneres de, amb un màxim de dues instruccions i dos cicles, posar a zero el contingut d'un registre REG (omple la taula, fins 8). Exemple:

Suposarem que REG està accessible al ACCESS BANK (,A)

MOVLW 0 MOVWF REG	CLRF REG,A	COMF REG,W,A ANDWF REG,F,A	SETF REG,A INCF REG,F,A

CLRF WREG
MOVWF REG,A
(qualsevol altre que posi WREG a 0)
	MOVLW 0
	ANDWF REG,F,A
	(qualsevol altre que posi WREG a 0)
		MOVF REG,W,A
		SUBWF REG,F,A
		(també XORWF, compte amb els borrow!)
			SETF REG,A
			COMF REF,F,A
			(COMF nega bit a bit)

Entre d'altres solucions.

Computer Interfacing (CI) FIB. Primer parcial **04/11/2021**

Cognoms, Nom _____ **DNI** _____

Tota resposta sense justificar es considerarà nul·la !

P3. (1 punt)
Indica quin serà el valor del registre STATUS després d'executar les següents instruccions. Si un bit del registre no es veu afectat indica-ho amb una 'x'

Instruccions	STATUS				
	N	OV	Z	DC	C

MOVLW 0
MOVWF 0x00, A
MULWF 0x00, F, A
Cap instrucció modifica l'STATUS
	X	X	X	X	X

CLRF 0x00, B [0xB00]=0 INCF 0x00, B [0xB00]=1 INCF modifica tots els bits	0	0	0	0	0
MOVLB 3 CLRF 0x00, B [0x300]=0 DECF 0x00, B [0x300]=FFh DECF modifica tots els bits	1	0	0	0	0
MOVLW 0xE0 WREG=E0h MOVWF 0x00, A [0x000]=E0h RLNCF 0x00, F, A [0x000]=C1h RLNCF modifica N i Z	1	X	0	X	X

REGISTER 5-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(2)
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 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{\text{Borrow}}$ bit(1)
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{\text{Borrow}}$ bit(2)
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

P4. (1 punts)

En un fòrum d'internet, un usuari asegura que no hi ha problemes en connectar els pins I/O d'un PIC18F45K22 (VDD=5V) amb els pins I/O d'una Raspberry PI (VDD=3V3). Estàs d'acord amb aquesta afirmació?

Raspebrry PI GPIO input/output pin electrical characteristics	
Output low voltage V_{OL}	<0,4V
Output high voltage V_{OH}	>2,4V
Input low voltage V_{IL}	<0,8V
Input high voltage V_{IH}	>2V

Hem de comprovar els 4 casos posibles:

1 - Enviar un 0. Raspberry com a output i PIC com a input:
La senyal de la raspberry estarà entre 0V i 0,4V. El PIC llegirà un 0 entre 0V i 0,8V. És correcte i tenim un marge de soroll de 0,4V

2 - Enviar un 0. PIC com a output i raspberry com a input:
La senyal del PIC estarà entre 0V i 0,6V. La raspberry llegirà un 0 entre 0V i 0,8V. És correcte i tenim un marge de soroll de 0,2V

3 - Enviar un 1. Raspberry com a output i PIC com a input:
La senyal de la raspberry estarà entre 2,4V i 3,3V. El PIC llegirà un 1 a partir dels 2V. És correcte i tenim un marge de soroll de 0,4V

4 - Enviar un 1. PIC com a output i raspberry com a input:
La senyal del PIC estarà entre 4,3V i 5V. La raspberry llegirà un 1 a partir de 2V. Però el pin de la raspberry pi és tolerant a un input més gran que el seu VDD? L'enunciat no ens dóna informació sobre aquest punt. En cas afirmatiu, no hi hauria problema i l'usuari tindria raó. Si no fos així, podem malmetre els GPIO de la raspberry.

Computer Interfacing (CI) FIB. Primer parcial **04/11/2021**

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

P5. (2 punts)

Tenim la següent rutina que gestiona les interrupcions d'un PIC18F45K22 amb una Fosc=4MHZ

```
org 0x0008
        BTFSS INTCON3, INT1IE, A
        BRA exit
        BTFSS INTCON3, INT1IF, A
        BRA exit
        INCF 0x00, A
        BCF INTCON3, INT1IF, A
exit:   RETFIE FAST
```

2.1 Quina és la freqüència màxima a la que podem rebre interrupcions sense perdre cap al pin RB1 (INT1)? Suposeu que tot està ben inicialitzat i que el micro no té configurada cap altra font d'interrupció. (Fosc=4MHz)

Perdrem interrupcions si aquestes arriben més ràpid del que les podem tractar:

3 o 4 cicles de latència + 8 cicles que triga l'execució de l'RSI (sabem que no hi ha cap altra font d'interrupció i si s'executa és degut a la INT1). En total 11 o 12 cicles a 4MHz.

Cas millor: 11 cicles *4/4M= 0,000011s -> 90909Hz

Cas pitjor: 12 cicles *4/4M= 0,000012s -> 83333Hz

2.2 Quines modificacions trobaríem al codi de l'RSI anterior si aquesta fos de baixa prioritat?

Si la interrupció dos de baixa prioritat la trobarem l'RSI a l'espai reservat per a la rutina de servei a la interrupció de baixa prioritat: 0x0018

Al ser de baixa prioritat no es faran servir els shadows registres llavors el codi ha de guardar i restaurar l'estat, per exemple ho pot fer amb

```
MOVFF WREG, WREG_TMP
MOVFF BSR, BSR_TMP
MOVFF STATUS, STATUS_TMP
…
MOVFF WREG_TMP, WREG
MOVFF BSR_TMP, BSR
MOVFF STATUS_TMP, STATUS
```

I al no fer servir els shadows register, retornarem amb un RETFIE normal, no FAST

2.3 Quins bits configuraries, i amb quin valor, per a activar la INT1 sense prioritats?

És una interrupció de core. Necessitem configurar:
ANSELB[1] = 0 (Digital)
TRISB[1] = 1 (input)
INT1EDG = x (configurem la detecció de flanc ascendent o descendent)
INT1IF = 0
INT1IE = 1
GIE = 1

2.4 Quins bits configuraries, i amb quin valor, per a activar la INT1 com a baixa prioritat?

Necessitem configurar:
ANSELB[1] = 0 (Digital)
TRISB[1] = 1 (input)
INT1EDG = x (configurem la detecció de flanc ascendent o descendent)
INT1IP = 0 (low priority)
IPEN = 1 (activem les prioritats)
INT1IF = 0
INT1IE = 1
GIEL = 1 (habilitem les interrupts de baixa prioritat)
GIE = 1

Computer Interfacing (CI) FIB. Primer parcial **04/11/2021**

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

P6. (1 punt)

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.

És exactament el contrari. Sortim de les RSI d'alta prioritat amb un RETFIE FAST (així es recupera el context des dels Shadow Registers). I són les de baixa prioritat les que han d'afegir codi extra i sortir amb RETFIE.

C El PIC té una pila de 31 posicions a on es guarden adreces de memòria de programa, per a que es pugui retornar de crides a subrutines del programa o d'atenció a interrupcions.

Cada cop que fem un CALL o RCALL, o bé es crida a una RSI, es guarda en aquesta pila l'adreça de retorn (valor actual del PC). Veure capítol 5.1.2 Return Address Stack al datasheet.

F En un pin configurat com a Output, el sentit de la corrent elèctrica mai pot ser de l'exterior cap a l'interior del pin

El fet de configurar un pin com a Output indica que controlem el valor lògic (i per tant el voltatge) que hi ha en el pin. Una cosa diferent és el flux de corrent elèctrica que s'hi pugui generar en funció del circuit que connectem al pin. Fixeu-vos, com a exemple, en un circuit R-LED que activem per "0". La corrent entrarà de l'exterior cap a l'interior del pin quan encenguem el LED.

C Per a executar una instrucció "MOVFF dada1, dada2" no cal haver especificat abans amb el BSR el Banc de memòria on estan les dades.

El BSR és necessari per especificar el codi de Bank (4 bits), quan l'adreça que es codifica a les instruccions és de 8 bits. Però el MOVFF té codificades a la pròpia instrucció les adreces completes de 12 bits tant de dada1, com de dada2. Aquest és el motiu pel qual necessita més espai que l'habitual, essent una de les instruccions double-word que tenim al PIC.

F En l'algoritme d'escaneig d'un Keypad "row-by-row", els díodes de protecció que posem a cada fila són per evitar curtcircuits en el cas que es premin més d'1 tecla dins la mateixa fila.

Quan escanegem per files, no hi ha cap problema en prémer més d'1 tecla dins la mateixa fila. El problema podria venir si la pulsació de varis botons dins la mateixa columna unís el valor 0 de la fila escanejada, amb el valor 1 d'alguna de les inactives. Els díodes prohibeixen aquesta corrent de curtcircuit.

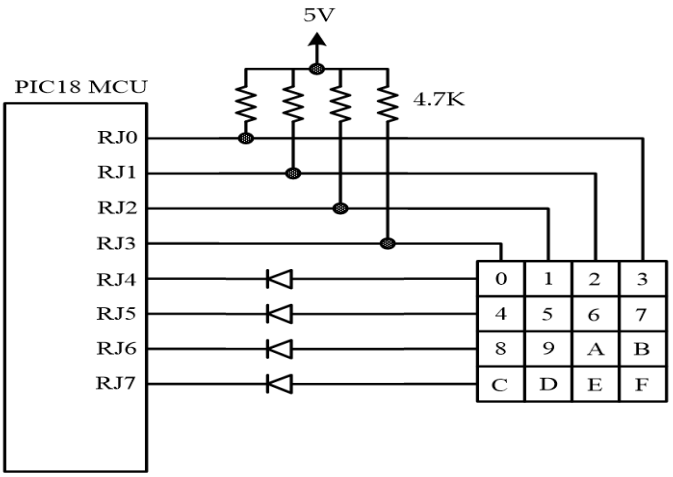


Figure 7.26b Sixteen-key keypad connected to PIC18 (used in all SSE demo boards)

Situació amb botons “0” i “4” premuts alhora, que podria haver causat un curtcircuit si no estiguessin els díodes

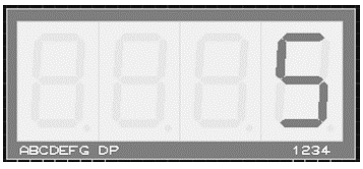
P7. (1,5 punts)

Un alumne ha programat el següent codi per a mostrar si un botó està premut o no, usant un display de 7segments amb 4 caràcters (com el de la placa EasyPIC del laboratori). Sempre que el botó estigui apretat, ha d’aparèixer “PULS”, si no ho està ha de sortir “FREE”.

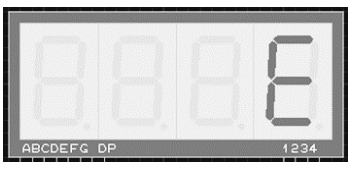
```
void main() {
   configPorts(); // ports digitals, i ajusta pins I/O adequadament

   while(1) {
      if (PORTBbits.RB0 == 1) {
         pintaDisp(3,CODI_P);
         pintaDisp(2,CODI_U);
         pintaDisp(1,CODI_L);
         pintaDisp(0,CODI_S);
         while (PORTBbits.RB0 == 1);
      }
      else {
         pintaDisp(3,CODI_F);
         pintaDisp(2,CODI_R);
         pintaDisp(1,CODI_E);
         pintaDisp(0,CODI_E);
         while (PORTBbits.RB0 == 0);
      }
   }
}
```

Però el resultat no és l’esperat, i en comptes d’aparèixer les paraules senceres, només s’encén el display de més a la dreta, com es veu a continuació:

El que es veu amb botó premut:



El que es veu amb botó lliure:



Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

Expliqueu per què passa això, i si creieu que el problema ve del software, proposeu una modificació del programa per tal que es vegin les paraules "PULS" i "FREE" senceres.

Com ens diuen que el display de l'exercici és com el de la placa EasyPIC, sabem que amb un Port del PIC especifiquem els segments encesos o apagats dels dígits (7segments). I que tenim 4 pins del PIC addicionals per a escollir quin dígit dels quatre volem encendre. El codi crida a una rutina anomenada pintaDisp, que selecciona un dígit concret d'entre els 4 possibles i col·loca un valor a un port per a encendre/apagar els segments.

Però recordem que aquest sistema de tenir un únic Port compartit pels 4 dígits, requereix que fem multiplexació en el temps, i contínuament anem activant tots els dígits, en un cicle sense fi. Si en algun moment s'atura aquest cicle, ens quedarem mostrant només el darrer dígit demanat.

Si ens fixem al codi, després de pintar les 4 lletres de "PULS", fem una **espera activa** fins que deixem de polsar. Tanmateix, després de pintar "FREE", fem una **espera activa** fins que tornem a polsar.

Aquestes esperes actives innecessàries, bloquegen l'execució del bucle infinit del main, i per tant s'atura l'algoritme d'escriptura al display. La solució és tan simple com eliminar aquestes esperes actives.

```
while(1) {
   if (PORTBbits.RB0 == 1) {
      pintaDisp(3,CODI_P);
      pintaDisp(2,CODI_U);
      pintaDisp(1,CODI_L);
      pintaDisp(0,CODI_S);
      // ELIMINEM AIXO: while (PORTBbits.RB0 == 1);
   }
   else {
      pintaDisp(3,CODI_F);
      pintaDisp(2,CODI_R);
      pintaDisp(1,CODI_E);
      pintaDisp(0,CODI_E);
      // ELIMINEM AIXO: while (PORTBbits.RB0 == 0);
   }
}
```

Evidentment hi ha altres alternatives de codi que solucionen el problema. L'important és que no parem mai de refrescar l'escriptura dels 4 dígits. P.ex: podríem mantenir els dos "*while problemàtics*", movent les 4 instruccions pintaDisp a dins de cada while, com veiem aquí:

```
while(1) {
   if (PORTBbits.RB0 == 1) {
      while (PORTBbits.RB0 == 1) {
         pintaDisp(3,CODI_P);
         pintaDisp(2,CODI_U);
         pintaDisp(1,CODI_L);
         pintaDisp(0,CODI_S);
      }
   }
   else {
      while (PORTBbits.RB0 == 0) {
         pintaDisp(3,CODI_F);
         pintaDisp(2,CODI_R);
         pintaDisp(1,CODI_E);
         pintaDisp(0,CODI_E);
      }
   }
```