

Cognoms i Nom: _____

Doc. Identitat: _____

Totes les respostes han d'estar degudament justificades

1. Quin(s) motiu(s) i/o característica(ques) permet que un microcontrolador PIC18F pugui simultàniament, executar una instrucció i realitzar la lectura (*fetch*) de la següent instrucció. Justifiqueu la resposta. (1 punt)

2. Donat el següent codi

(2 punts)

a. Indicar el valor dels registres de la taula després de la seva execució.
Assumint que tota la memòria de dades està a 0 a l'inici de l'execució.

@ RAM	Valor final
0x050	
0x051	
0x150	
0x151	
0x250	
0x251	
WREG	
BSR	

b. Calcular el temps d'execució del codi (des del *power-on* o activació del MCLR fins l'última instrucció del llistat), si $F_{osc} = 20$ MHz.

c. Indica el nombre de bytes de memòria de programa que ocupa el codi del llistat.

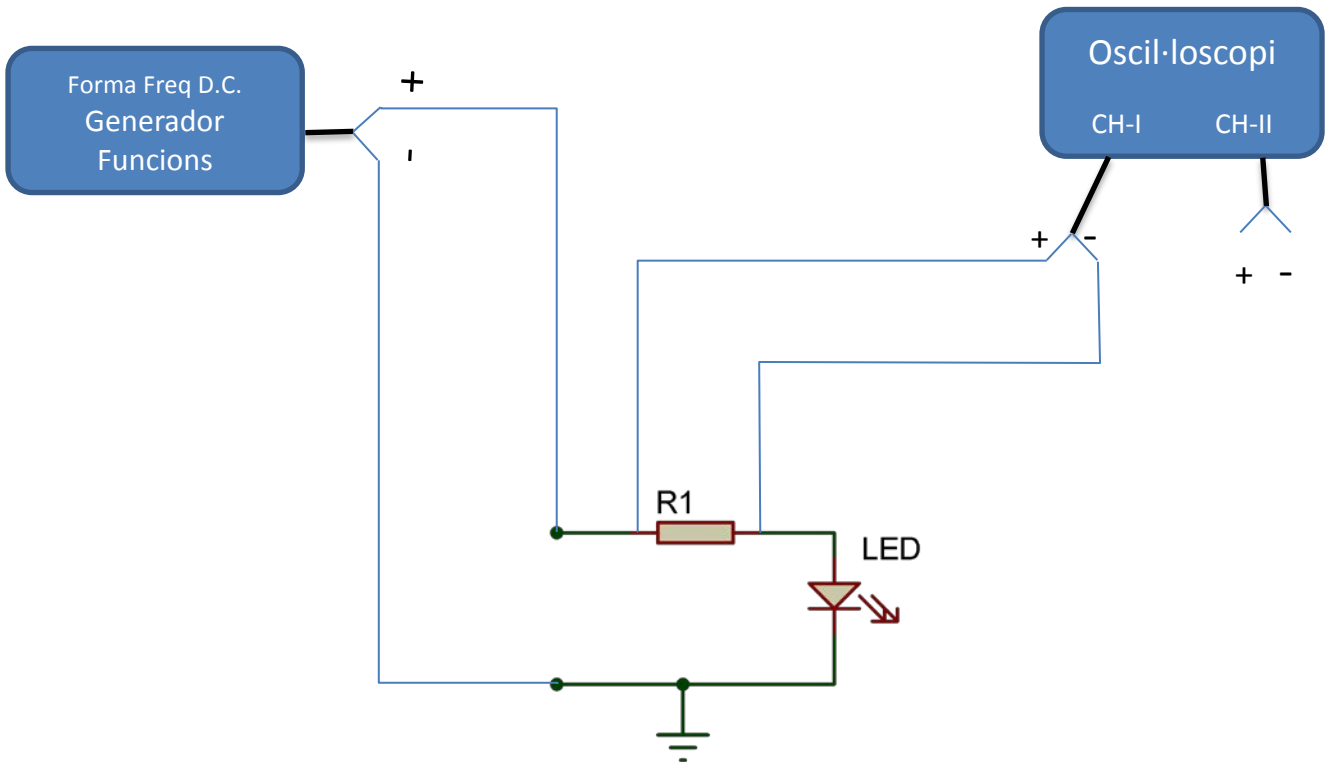
```
A equ 0x50
B equ 0x51
C equ 0x15
ORG 0
GOTO START
....
ORG 50
START
CLRF A, 0
MOVLW 0x10
MOVLB 0x01
SETF B, 0
ADDLW C
MOVWF A, 1
MOVLB 0x02
MOVWF A, 0
MOVLW 0xAB
Delay1
NOP
NOP
DECFSZ WREG, 0, 0
BRA Delay1
INCF A, 0, 1
DECF A, 1, 0
ADDWF A, 1, 1
....
```

3. Es vol construir un circuit per encendre intermitentment un LED a un ritme de 15 vegades per minut ($I_{LED} = 9\text{mA}$; temps LED encès igual al temps LED apagat). El senyal que activa el LED prové d'un generador de funcions configurat per generar un senyal amb 5 volts d'amplitud. A més, es vol mesurar la caiguda de potencial en borns de la resistència. Es disposa dels següents components i instruments: (2 punts)

R, C, L, LED, BJT, JFET, Generador de funcions i Oscil·loscopi

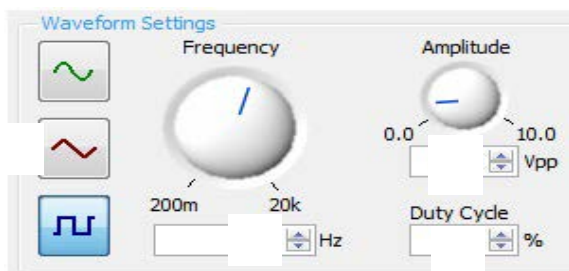


a) Realitzeu el diagrama electrònic o esquemàtic, a partir dels components disponibles. El diagrama electrònic ha d'incloure les seves connexions al generadors de funcions i a l'oscil·loscopi amb la resta de components que formin part del circuit. Calculeu els valors dels components que utilitzeu (R's, C's o L's).



$$R1 = (5-2)/9 \text{ mA} = 333,3 \, \Omega$$

b) Indiqueu com configurariu el generador de funcions indicant la forma, la freqüència i el *duty cycle* del senyal.



Generador de funcions

Cognoms i Nom: _____ Doc. Identitat: _____

Totes les respostes han d'estar degudament justificades

4. Es habitual que en les aplicacions es defineixin taules de valors que es troben dins del mateix programa. Això vol dir que necessitem un mecanisme per accedir a dades que es troben en **memòria de programa**. Una de les formes per llegir taules en memòria de programa es mitjançant la instrucció **RETLW <dada>**. Aquesta instrucció retorna d'una crida CALL guardant en el working registre el literal <dada> especificat.

Un exemple de rutina per accedir a una dada que es troba en una taula en memòria de programa seria:

```
..
MOVF <DESPLA>, W      ; ACUMULEM EL DESPLAÇAMENT A LA TAULA A W
CALL TAULA             ; FEM CRIDA A TAULA
..
..
ORG <INICIA TAULA>
TAULA ADDWF PCL         ; SUMEM AL PC EL DESPLAÇAMENT
      RETLW <DADA0>     ; RETORNEM GUARDANT LA DADA0 EN W
      RETLW <DADA1>     ; RETORNEM GUARDANT LA DADA1 EN W
      ..
      ..
      RETLW <DADAn>     ; RETORNEM GUARDANT LA DADAn EN W
```

Fent servir aquesta implementació:

(4 punts)

a) Quan ocupa en memòria cada element de la taula?

b) Si volem accedir a la DADA5 de la taula quin ha de ser el valor de <DESPLA>.

c) Amb aquesta implementació quina és la dimensió màxima de la taula?

d) Durant l'execució del codi que permet accedir a una dada de la taula, es modificarà en algun moment el contingut de la pila? Justifica la resposta.

5. Donat que el vector de reset i el vectors d'interrupcions es troben a l'adreça 0x00, 0x08 i 0x18 respectivament, tenim alguna restricció o conflicte per reservar variables en la memòria de dades en aquestes posicions de memòria? Justifica la resposta. (1 punt)

Extracte / Resum del conjunt d'instruccions del PIC18F4550 (Secció 26 del manual)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb		LSb			
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
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
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
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
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	1110	00ff	kkkk	None	
				1111	0000	kkkk	kkkk		
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	
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f _s , f _d	Move f _s (source) to f _d (destination) 2nd word	2	1100	ffff	ffff	ffff	None	
				1111	ffff	ffff	ffff		
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	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call Subroutine 1st word 2nd word	2	1110	110s	kkkk	kkkk	None	
				1111	kkkk	kkkk	kkkk		
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f