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)

El fet de tenir dos espais de memòria independents (Arquitectura Havard) permet accedir al mateix temps a la memòria de programa (i fer el fetch) i a la memòria de dades (llegir i escriure els operands involucrats en l'execució d'una instrucció).

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	0x24
0x051	0xFF
0x150	0x25
0x151	0x00
0x250	0x01
0x251	0x00
WREG	0x01
BSR	0x02

```
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
....
```

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

3 cicles (goto) + 1 cicle * 9 inst (abans del bucle) + (5 cicle)*170 iteracions + 7 cicles (última iteració i 3 inst. fora del bucle) = **869 cicles**

1 cicle = 4*(1/20M)= **0,2 µs** $t_{execució}$= 869*0,2 µs = **173,8 µs**

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

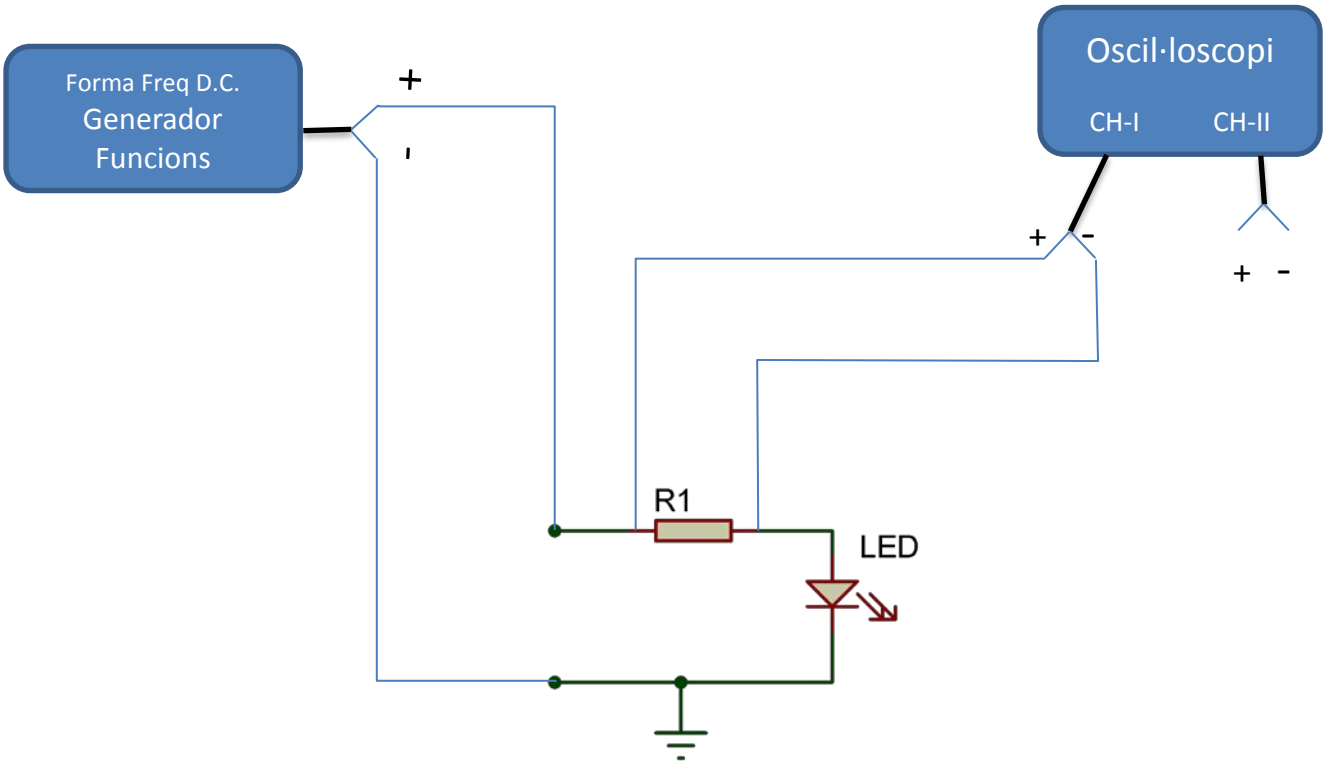
Tenim 1 instrucció double-word (GOTO) i 16 instruccions single word.

1*(4 bytes) + 16*(2 bytes)= 36 bytes

3. Es vol construir un circuit per encendre intermitentment un LED a un ritme de 15 vegades per minut (I_{LED}= 9mA; 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)

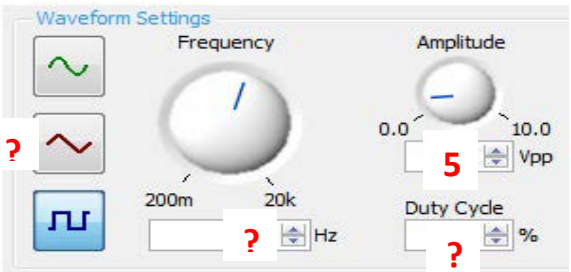
R, C, L, LED, BJT, JFET, Generador de funcions i Oscil·loscopi

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 mA= 333,3 Ω

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



Generador de funcions

Forma del senyal: Senyal Quadrat (senyal amb només dos estats/tensions possibles; encès i apagat)

Freqüència: 0,25 Hz (15 cicles/minut passat a unitats del sistema mètric internacional –Hz-)

Duty Cycle: 50% (temps alt del senyal igual al 50% del període)

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?

Cada element de la taula ocupa dos bytes (encara que el valor siguin 1 byte, guardem en realitat una instrucció de 16 bits).

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

<DESPLA> ha de valdre el doble del desplaçament de la taula donat que cada dada ocupa 2 bytes.
Per tant per accedir a DADA5, DESPLA=10.

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

Donat que estem sumant el desplaçament en registres de 8 bits (al W o a la part baixa del registre PC (PCL)) i que cada dada ocupa 2 bytes com a màxim podem acumular 128 elements a 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.

Si. En el moment de fer el CALL (push) i en el moment de fer el RETLW (pop)

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)

No. Els vectors de reset i d'interrupcions es troben a la memòria de programa i no pas a la memòria de dades. Per tant no hi ha cap conflicte ni restricció.

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

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb	LSb	Status Affected	Notes
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 1111 0000 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	
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 ffff ffff ffff 1111 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	
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 1111 kkkk kkkk kkkk		None	
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