FIB, Interfícies dels Computadors
Primer parcial 13-11-2014 (1h45', Full 1/2)

NOM:

COGNOMS:

DNI:

Responeu en aquest mateix full. Cal justificar totes les respostes.

Respostes sense un mínim text explicatiu no es tindran en consideració.

1) En relació al següent codi en assembler del PIC18F (2 punts)

```
PORTD equ  0x0F83
TRISD equ  0x0F95
Delay1 equ 0
Delay2 equ 1
ORG 0
1: CLRF PORTD
2: CLRF TRISD
3: SETF Delay1 ; FFh-> Delay1
4: CLRF Delay2
Delay:
5: INCFSZ Delay1,F
6:GOTO Delay
7:DECFSZ Delay2,F
8:GOTO Delay
```

a) En quina adreça de memòria se situaria la línia etiquetada com a 1:?
Tal com indica la directiva ORG la instrucció que ve a continuació se situa a l'adreça zero.

b) Calculeu la mida en Bytes que ocuparia el programa en la ROM. Indiqueu els càlculs per a cada instrucció del programa.
6 instruccions single-word (1,2,3,4,5 i7) i dues instruccions double-word (6 i 8). Total 20 Bytes.

c) Indiqueu una solució ben senzilla per reduir la seva mida en ROM. Quin estalvi en bytes obteniu amb la solució proposada?
Canviar les instruccions de salt absolut GOTO per un salt relatiu BRA. Reduïm la mida del codi en 4 Bytes (2 Bytes per cada GOTO->BRA).

d) Quantes vegades s'executaria la línia 5 fins a sortir dels dos bucles anidats?
El bucle més intern s'executa la primera vegada 1 cop i les demés vegades 256 cops. El bucle més extern s'executa 256 cops. En total la instrucció 5 s'executa 255*256+1 = 65.281 cops.

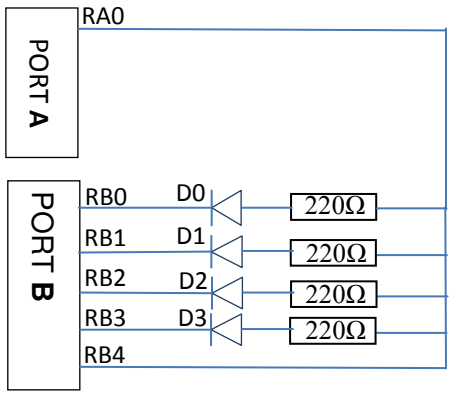
2) En relació als modes d'adreçament indirecte:(1 punt)

a) Els registres d'accés indirecte FSR (File Select Register) poden apuntar a qualsevol espai de la memòria ROM?
Els registres FSR apunten a la RAM i no a la ROM.

b) Quina mida (en bits útils) tenen els registres d'accés indirecte FSR?
12 Bits per adreçar 4 KBytes de RAM. Si la implementada RAM és de 2KBytes, llavors només són 11 els bits útils.

c) Quina mida tenen els registres indirectes INDF?
És un registre de 8 bits que s'utilitza com a àlies del registre FSR.

3) En relació al següent esquema (2 punts)



A) Mantenint els pins de sortida dins dels nivells de voltatge adequats als seus nivells lògics, calcular quants LEDs es podran encendre com a màxim si:

$V_{OH\ MIN}$= 4,7 Volts,$V_{OH\ MAX}$= 5,0 Volts, $V_{IH\ MIN}$= 4,5 Volts
$V_{OL\ MAX}$= 0,3 Volts,$V_{OL\ MIN}$= 0 Volts, $V_{IL\ MAX}$= 0,7 Volts
$I_{OH\ MAX}$= $I_{OL\ MAX}$= 20mA, $I_{Leakage}$= 2,5 µAi la tensió en borns dels díodes ésde **2 V**quan estan encesos.

La combinació que permet encendre més LEDs és RA0 i RB4 configurats de sortida i en estat alt (1 lògic). Aquesta combinació permeten sumar el dos corrents que poden arribar a proporcionar cadascun dels PINs (2 x 20 mA). Mantenint els nivells per sobre dels valors especificats VOH MIN, el corrent que demanaria cada branca dels díodes seria de:

I_D = (4,7 – 2) / 220 Ampers = 12,27mA

Amb tres LEDs encara s'està per sota del corrent màxim (40 mA). Amb quatre LEDs se supera el corrent màxim que els dos pins poden proporcionar conjuntament.

FIB, Interfícies dels Computadors
Primer parcial 13-11-2014 (1h45', Full 2/2)

NOM:

COGNOMS:

DNI:

Responeu en aquest mateix full. Cal justificar totes les respostes.

Respostes sense un mínim text explicatiu no es tindran en consideració.

4) Indica quants bits s'utilitzen per codificar la instrucció **MOVFF f_s, f_d**, i quants s'utilitzen per la instrucció **MOVF f, d, a.** Justifica el motiu d'aquesta diferència.

MOVFF és una instrucció double-word doncs per indicar l'adreça absoluta d'origen i la de destí calen 12 x 2 bits, més 4 bits pel codi d'operació i 4 bits del codi NOP inserit en la segona part de la instrucció; en total 32 bits. La instrucció MOVF es codifica amb 16 bits, on 8 d'aquest 16 bits s'utilitzen per adreçar la RAM. En conseqüència, amb el MOVF només es pot adreçar 256 bytes que cal complementar amb el registre BSR (o l'ACCES BANK segons el cas) per obtenir l'adreça final.

5) Indiqueu quin és el valor final de les posicions de memòria indicades a la taula després de l'execució delasegüentsecció de codi. (2 punts)

@ RAM	Valor inicial	Valor Final
0x000	0x00	0x00
0x001	0x01	0x01
0x002	0x02	0x02
0x003	0x03	0x03
0x004	0x04	0x04
0x010	0x00	0x10
0x110	0x01	0x03
0x210	0x02	0x1A
0x310	0x03	0x41
0x410	0x04	0x04
WREG	0x00	0x2D
BSR	0x00	0x03

d= 0 result in WREG
1 result in file registre
a= 0 Access RAM
1 Banked RAM

```
(...)
        MOVLB 0x01
        MOVLW 0x10
        MOVWF 0x10, 0
        MOVLW 0x03
        MOVWF 0x10, 1
        MOVLB 0x02
        MOVLW 0x1A
        MOVWF 0X10, 1
        MOVLB 0x03
        MOVLW 0x14
        MOVWF 0X10, 1
        MOVLB 0x01
        MOVF 0x10, 0, 0
        ADDWF 0x10, 0, 1
        MOVLB 0x02
        ADDWF 0x10, 0, 1
        MOVLB 0x03
        ADDWF 0x10, 1, 1
(...)
```

6) Assumint que s'utilitza el hardware EASYPIC6, implementaren C una rutina *voidclearGLCD(byte pagei, byte pagef, byte columni, byte columnf)* que "neteja" el GLCD en la zona rectangular delimitada entre *pagei-pagef*i *columni-columnf*. Per implementar aquesta funció clearGLCD podeu assumir que ja teniu implementades les següents funcions: (2 punts)

byte readByte(byte page, byte column);
voidwriteByte(byte page, byte column, byte data);
voidsendGLCDCommand(byte val, byte CS);

```
//  Clear all pixels in pagesfrompagei to pagefandin columnsfromcolumni to comlumnf
// pagei, pagef [0:7] columni, comlumnf[0:127]
voidclearGLCD(byte pagei, byte pagef, byte columni, byte columnf)
{
bytei,j;

        for (i = pagei; i <=pagef; i++)
                for (j=columni;j<=columnf;j++)
                        writeByte(i,j,0);  // Clear Page i atcolumn j
}
```