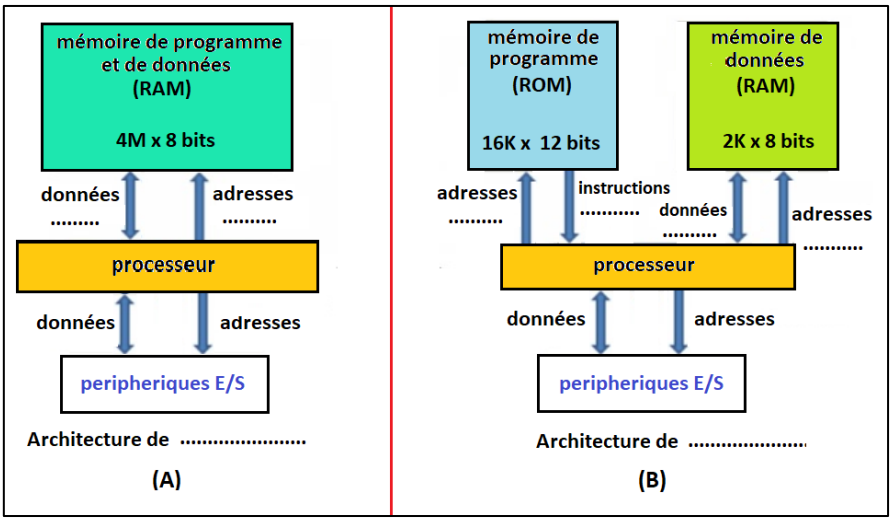
Série de TD N° 7-8

Exercice 01 :

1. Déterminer de quel type d'architecture est conçu chaque schéma bloc.
2. Déterminer la taille des bus (adresses, données, instructions) sur chaque schéma bloc.



3. Où est-ce qu'on peut trouver les microcontrôleurs ?
4. Est-ce que tous les microcontrôleurs doivent être obligatoirement d'architecture Harvard.
5. Citez 6 composants qu'on peut trouver dans un microcontrôleur.

Exercice 02 :

Questions de cours

1. Quel type d'architecture le PIC16F84 a-t-il (RISC/CISC). Donnez le nombre total de ses instructions.
2. Quelle est la taille en bits, des instructions et des données mémoire du PIC16F84 ?
3. Quels sont les circuits qui sont indispensables au fonctionnement du microcontrôleur.
4. Si l'horloge externe du microcontrôleur PIC16F84 est dotée d'un quartz de 8Mhz. Déterminer le temps de cycle d'une instruction.
5. De quel type est la pile du PIC16F84 (FIFO, LIFO…etc.), et combien de niveau a-t-elle ?
6. Quelle est la taille de la mémoire programme du PIC16F84.
7. Quel est le nombre d'interruptions périphériques qui peuvent être gérées par le PIC16F84. Citez-les ?
8. Quel bit est positionné à '1' est utilisé pour autoriser toutes les interruptions du PIC16F84.
9. Si l'interruption TMR0 est autorisée (T0IE =1), Déterminer à quel moment l'interruption survient-elle ?
10. Quelle instruction annonce la fin de sous-programme d'interruption et oblige le processeur à retourner au programme principal.

Exercice 03 :

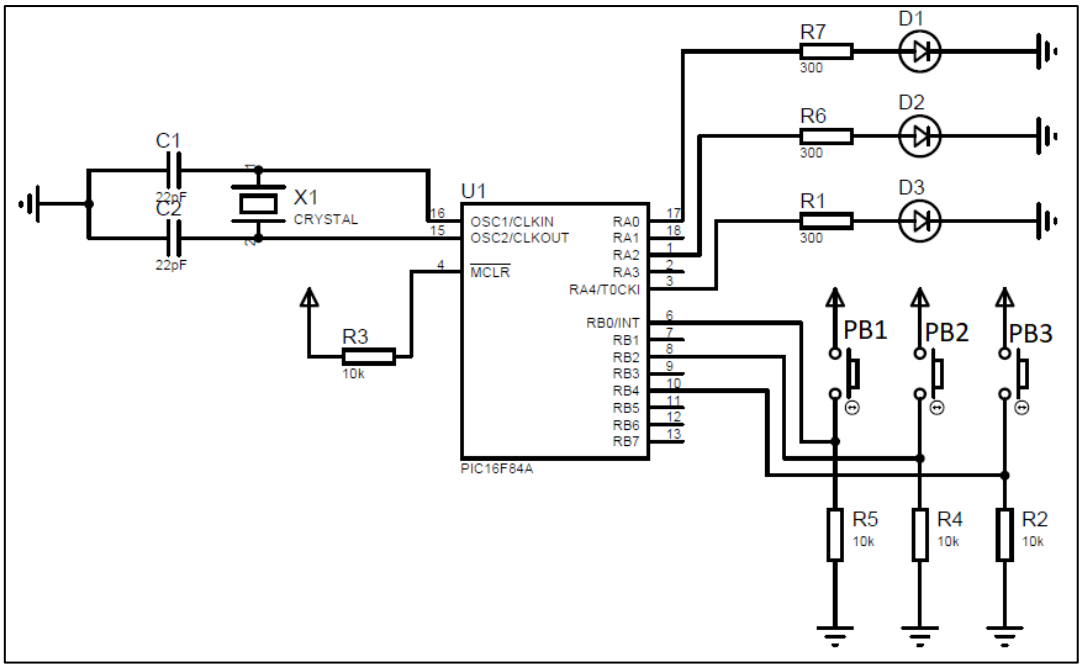
Répondez par (vrai) ou (faux) aux affirmations suivantes, et corrigez les affirmations fausses

1. Dans une architecture Harvard, le processeur peut facilement lire/écrire des données et accéder aux instructions à tout moment.
2. Lors de la mise sous tension du PIC16F84, le processeur pointe vers l'adresse 0000H de la mémoire du programme.
3. Un niveau de tension haut sur la broche $\overline{\text{MCLR}}$ provoque un reset du PIC16F84.
4. Si un périphérique provoque une interruption, le processeur se positionne à l'adresse 0004H.
5. Les registres TRISA et TRISB déterminent respectivement le sens du PORTA et du PORTB.

6. Le mode SLEEP est utilisé pour minimiser la consommation d'énergie dans les applications alimentées par batterie.
7. Une interruption sur la broche RB0/INT peut faire sortir le PIC du mode SLEEP (réveiller le PIC).
8. Dès la mise sous tension du PIC16F84 le registre TMR0 s'incrémente automatiquement.
9. La pile du PIC16F84 permet la sauvegarde de l'adresse du PC de retour lors des appels des sous-programmes (CALL) ou lorsqu'une interruption survient.
10. Le programmeur doit effacer le drapeau qui a signalé l'interruption après l'avoir traitée.
11. L'instruction RETFIE permet de restaurer le contenu du registre PC à partir de la pile pour permettre au processeur de reprendre l'exécution du programme qu'il avait interrompu.
12. Le PIC16F84 peut sauvegarder des données mémoire en pile.
13. Dans une architecture de Van Neumann, la vitesse d'exécution est plus rapide car le processeur récupère simultanément les données et les instructions.

Exercice 04 :

Soit le schéma ci-dessous, composé de 3 boutons poussoirs PB1, PB2 et PB3 liés respectivement à RB0, RB2 et RB4, et de trois LEDs D1, D2 et D3 liées respectivement à RA0, RA2 et RA4. L'ensemble est géré par un microcontrôleur PIC16F84.



Nous allons décrire le fonctionnement du montage ci-dessus pour le bouton PB1 et la LED1, mais en fait le raisonnement est le même pour les composants restants.

- Si le bouton PB1 est enfoncé l'entrée RB0 est lue comme un '1' logique, la sortie RA0 doit être mise à un niveau de tension haut ≈ 5Volts ('1' logique) et la LED s'allume.
- Si le bouton PB1 est relâché l'entrée RB0 est lue comme un '0' logique, la sortie RA0 doit être mise à un niveau de tension bas (0 logique) et la LED s'éteint.

1. Ecrire un programme qui permet de lire les boutons poussoirs PB1, PB2 et PB3 et afficher leurs états respectivement sur D1, D2 et D3. Ci-dessous, une proposition de la structure du programme assembleur, qui peut être adoptée par le programmeur.

```
                org     0x000           ; Adresse de départ après reset
                goto    init            ; Adresse 0: initialiser

        init
; INITIALISATION DES PORTS
                .
                .
                .
                .
                goto    start           ; sauter au programme principal
; PROGRAMME PRINCIPAL
        Start
                .
                .
                .
                goto    start           ; boucler
        END                             ; directive de fin de programme
```