Faculté des Sciences
Aix*Marseille Université

Année universitaire 2012/2013

Site : ☐ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS

Sujet session de : ☐ 1er semestre - ☐ 2ème semestre - ☐ Session 2 Durée de l'épreuve :1 H 30.

Examen de : ☐L1/ ☐L2 /☐L3 - ☐M1/☐M2 - ☐ LP - ☐DU Nom diplôme : Licence SPI (L2)

Code Apogée du module : SPI42AJ ……..Libellé du module : Informatique industrielle

Document autorisé : ☐ Oui (le polycopié de cours uniquement) Calculatrices autorisées : ☐ OUI

Mercredi 15 Mai 2013 14H00-15H30 *Licence 2 – SPI*

Aix Marseille Université

Informatique industrielle

Le polycopié de cours est autorisé.

Une calculatrice de type collège est autorisée.

Cet examen de 1 heure 30 est composé de 2 parties :

- *des questions de cours* (approximativement 45mn),
- *un problème en langage Assembleur* (approximativement 45 mn).

Note :

Des extraits de la documentation technique nécessaires à cet examen vous sont donnés en annexe.

Questions de cours

Attention

Vous êtes notés sur clarté de vos réponses. Pour avoir le maximum de points, elles devront être justes, courtes, et précises. Utilisez vos brouillons avant de vous jeter sur vos copies d'examen.

[A] Questions générales

1. Quels sont les principaux critères de performance d'un microcontrôleur ? Comment ces critères ont-ils évolué ces dernières années ?
2. Pour une instruction en assembleur, à quoi correspond l'opcode ? A quoi correspond l'opérande ? Donnez un exemple.
3. Quelle est la différence entre l'adressage inhérent et l'adressage immédiat ?

[B] Questions « architecture »

1. Identifier les éléments ci-dessous sur le schéma bloc de la *figure 1* :

-le compteur programme,
-le bus de données,
-le bus d'instructions,
-l'unité arithmétique et logique,
-la mémoire programme,
-les ports entrées-sorties.

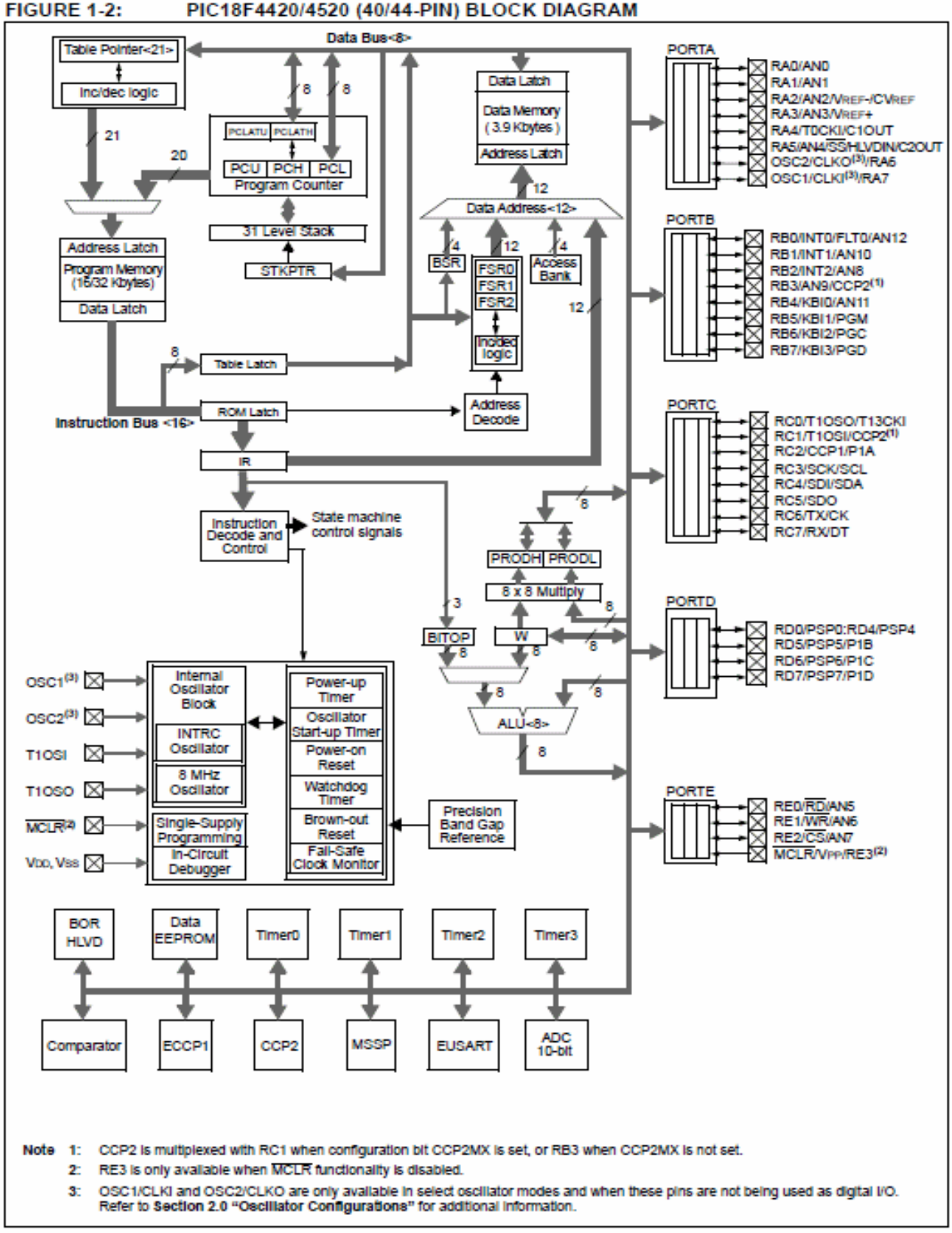
2. Le micro-contrôleur présenté *figure 1* est-il de structure *Von-Neumann* ou *Harvard* (justifiez votre choix)? Est-ce un microprocesseur 8 bits, 16 bits ou 32 bits ?

[C] Question « codage binaire, hexadécimal et décimal »

1. Convertissez a=0x5E en binaire
2. Convertissez b=0x6F en binaire
3. En utilisant les questions 1 et 2, effectuez l'opération a-b en binaire, puis donnez le résultat sous forme hexadécimale.

4. Effectuez l'opération *logique* (binaire) suivante : c xor w où c=0010 et w=1001 (xor : ou exclusif).
5. Effectuez l'opération *logique* (binaire) suivante : d xor w où d=1010 et w=1001 (xor : ou exclusif).
6. D'après les résultats obtenus en 4 et 5, comment peut-on appliquer l'opération 'ou exclusif' à l'inversion d'une LED dans le cadre d'une application impliquant un microcontrôleur ? Votre réponse ne doit pas excéder 5 lignes.

FIGURE 1-2: PIC18F4420/4520 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.
2: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.
3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

Figure 1 : Schéma de principe du PIC 18f4520.

Problème :

On utilise le logiciel MPLAB pour créer un programme assembleur, que l'on implante dans un microprocesseur PIC18F4520. Ce microprocesseur est ensuite placé sur une plaque de test comportant notamment des LEDs connectées au PORTB du microprocesseur.

Le programme est conçu selon ce *cahier des charges :*

On veut inverser l'état de la LED L1 connectée à la broche RB0 du PORTB, de façon à ce qu'elle reste à l'état haut pendant 1 seconde, et à l'état bas pendant 1 seconde.

1) Dans l'algorithme présenté pages suivantes, que vous rendrez avec votre copie d'examen, repérez les étapes principales d'une interruption (avec des accolades pour les étapes constituées de plusieurs lignes).

2) Quelle est la durée d'une itération de la boucle d'attente infinie du programme principal ? Servez-vous des données présentées en annexe pour répondre.

3) Quel registre faut-il considérer dans chaque cas pour :

   (a) configurer le TIMER0 (on/off, prescaler, etc.) ?

   (b) configurer et mettre en œuvre les interruptions (autorisation, etc.) ?

   Complétez le programme en conséquence aux emplacements désignés par _ _.

4) En faisant le test sur une plaquette PICDEM 2 PLUS, on constate que le programme ne fait pas ce qu'on voulait : qu'observe-t-on ? Pourquoi ? Que faudrait-il faire pour que le *cahier des charges* soit respecté ?

5) Dessinez deux algorigrammes succincts qui décrivent le fonctionnement du programme principal, et du programme d'interruption. Indiquez par un point d'exclamation l'endroit où le programme comporte une erreur.

```
; Programme Clignotant avec Interruption
        LIST P=18F4520
        #include <P18F4520.inc>
        #include <CONFIG.inc>

;----- Déclaration de variables
        CBLOCK 0x00
                W_TEMP  : 1
                STATUS_TEMP : 1
                BSR_TEMP : 1
                VARIABLEX : 1
        ENDC
;----- Programme principal
        org     h'0000'
        goto    init
        org     h'0008'
        goto    prog_int

init    clrf    PORTB           ;Remise à zéro du port B
        movlw   h'00'
        movwf   TRISB           ;Le port B est défini en sortie

        movlw   h'83'
        movwf   _ _             ;TIMER0 On, 16bits, Prescaler 16

        rcall   tmr0_init       ;Init TMR0 pour 1s pile
        movlw   h'A0'
        movwf   _ _             ;Autorisation Générale des interruptions et des interruptions du TIMER0.

        movlw   h'01'
        movwf   VARIABLEX

boucle
        incf    VARIABLEX
        decf    VARIABLEX
        goto    boucle          ;Boucle d'attente infinie
```

```
;----- Programme d'interruption
prog_int
        movwf  W_TEMP
        movff  STATUS, STATUS_TEMP
        movff  BSR, BSR_TEMP

        btfsc       _ _      ,2

        rcall    tmr0_overflow

        movff  BSR_TEMP, BSR
        movff  W_TEMP, WREG
        movff  STATUS_TEMP, STATUS
        retfie

;----- Init du registre TMR0
tmr0_init
        movlw  h'0B'
        movwf  TMR0H
        movlw  h'DB'
        movwf  TMR0L
        return

;----- Interruption de débordement du TIMER0
tmr0_overflow
        bcf          _ _      ,2
        rcall    tmr0_init
        movlw    h'08'
        xorwf  PORTB
        return

        END
```

Annexe

Dans tout le problème, les caractéristiques du PIC utilisé sont telles que la période de l'horloge est TOSC= 0.25 10^{-6} sec. Il y a quatre coups d'horloge par cycle.

L'instruction goto nécessite deux cycles pour être exécutée ; le descriptif complet des instructions incf et decf est donné ci-dessous:

INCF **Increment f**

Syntax:	INCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	0010 | 10da | ffff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `INCF CNT, 1, 0`

DECF **Decrement f**

Syntax:	DECF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	0000 | 01da | ffff | ffff
Description:	Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `DECF CNT, 1, 0`

Faculté des Sciences et Techniques

Centre de Marseille – Saint-Jérôme ☐ Centre d'Aix-en-Provence – Montperrin ■

Sujet session de : ……Mai 2014……..

Examen de Licence ■ Master ☐ DU ☐ Libellé diplôme : …Licence SPI

Code Apogée du module : SPI42AA ……..Libellé du module : Informatique industrielle

---

Jeudi 15 Mai 2014 8H30-10H00 *Licence 2 – SPI*

Aix Marseille Université

Informatique industrielle

Le polycopié de cours est autorisé.

Une calculatrice de type collège est autorisée.

Cet examen de 1 heure 30 est composé de 2 parties :

- *des questions de cours et exercices (14 points)*
- *un problème en langage Assembleur (6 points)*

Note :

Des extraits de la documentation technique nécessaires à cet examen vous sont donnés en annexe.

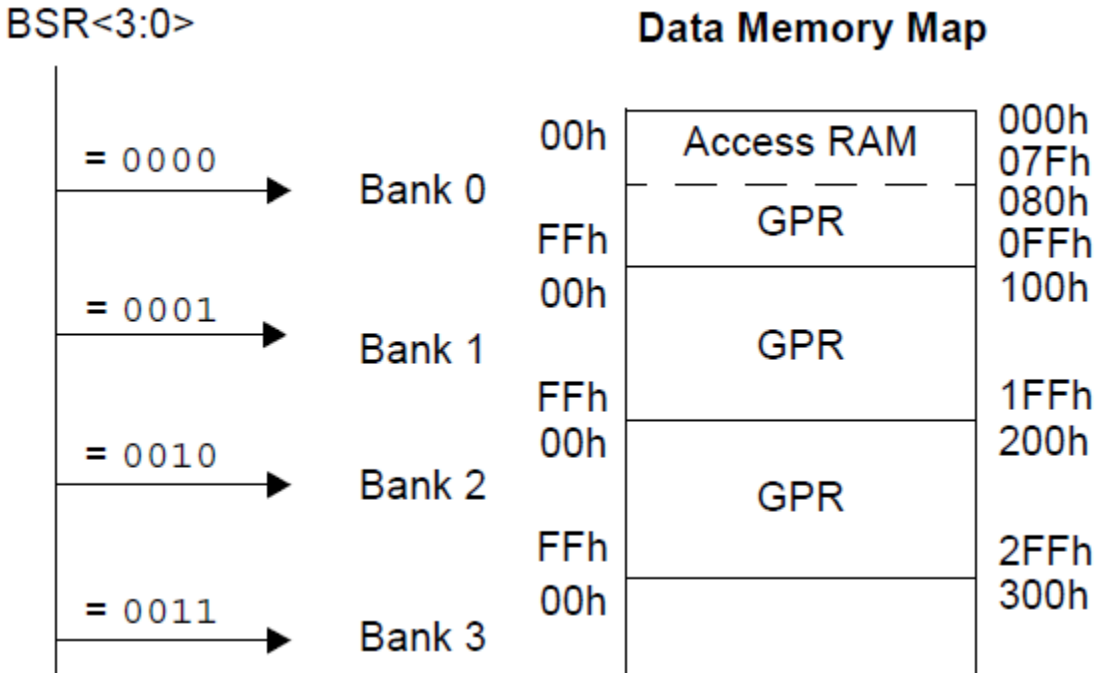
Questions de cours

Attention

Vous êtes notés sur clarté de vos réponses. Pour avoir le maximum de points, elles devront être justes, courtes, et précises. Utilisez vos brouillons avant de vous jeter sur vos copies d'examen.

[A] Questions générales

1. Lorsque l'on étudie le mode de fonctionnement du cerveau, on parle de 'potentiels d'action'. Quel est le parallèle que l'on peut faire entre les potentiels d'action et le terme de 'bit'que l'on utilise en informatique ?
2. Pour une instruction en assembleur, à quoi correspond l'opcode ? A quoi correspond l'opérande ? Donnez un exemple.
3. Cette question concerne les modes d'adressage : quel est le mode d'adressage utilisé quand on utilise l'adresse des données dans la RAM ? Quel est le rôle du registre BSR (voir figure ci-dessous) dans ce contexte ?

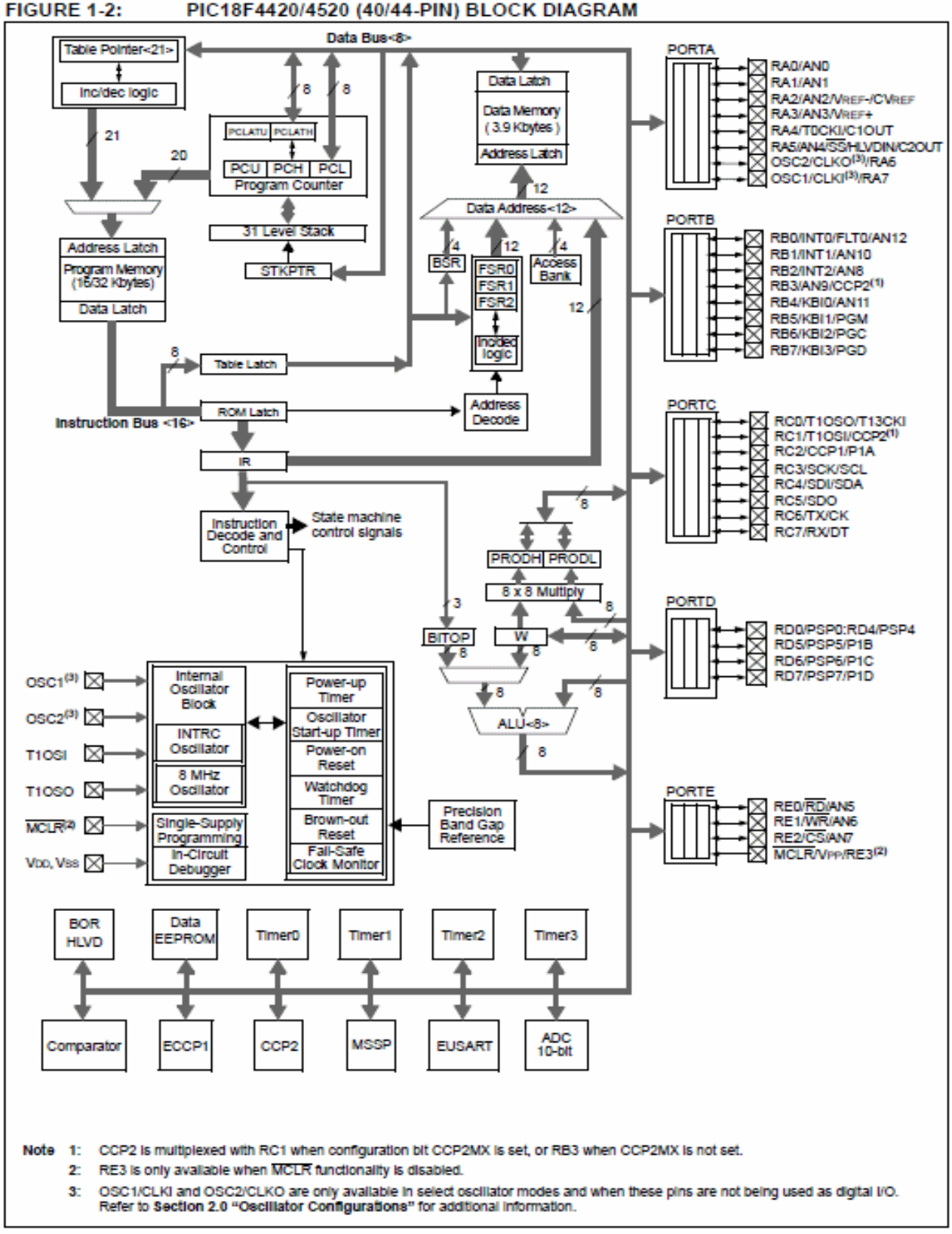


[B] Questions « architecture »

Identifier les éléments ci-dessous sur le schéma bloc de la *figure 1* :

-le compteur programme,
-le bus de données,
-le bus d'instructions,
-l'unité arithmétique et logique,
-la mémoire programme,
-les ports entrées-sorties.

FIGURE 1-2: PIC18F4420/4520 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.
2: RE3 is only available when MCLR functionality is disabled.
3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

Figure 1 : Schéma de principe du PIC 18f4520.

[C] Question « codage binaire, hexadécimal et décimal »

1. Convertissez a=0x5E en binaire
2. Convertissez b=0x6F en binaire
3. En utilisant les questions 1 et 2, effectuez l'opération a-b en binaire, puis donnez le résultat sous forme hexadécimale.

4. Effectuez l'opération *logique* (binaire) suivante : c xor w où c=1010 et w=0001 (xor : ou exclusif).

[D] Question « Temporisation »

On souhaite réaliser une temporisation. Des informations utiles sont fournies en annexe.

On considère le programme suivant, où t10 est une valeur fixe 'initiale' de t1, telle que t10>1 :

```
tempo
        movlw       t10
        movwf       t1
comp1   dcfsnz      t1
        return
comp2   infsnz      t1
        goto         comp1
        return
```

1. Dessinez l'algorigramme correspondant à ce programme
2. Quel est le nombre de cycles et la durée en microsecondes du programme 'tempo' ?
3. Est-ce que la valeur de t10 a une influence sur cette durée ?
4. Réécrivez le programme (en utilisant une partie de ses instructions) de façon à ce que le programme soit une temporisation correcte.

Problème

De même qu'un programme de temporisation, il existe un moyen de compter le temps qui s'écoule avec un microcontrôleur. Il s'agit des interruptions de débordement du TIMER0. L'annexe contient des informations utiles à la résolution du problème.

Dans ce problème on souhaite configurer le TIMER0 pour qu'une interruption se produise toutes les 1 sec.

La période du TIMER0 est donnée par l'expression suivante :

Où est la durée d'un cycle d'instruction, n le nombre de bits sur lesquels le timer compte.

Le tableau 1 donne les valeurs de T pour plusieurs valeurs de prescaler et de n

	8	*16*
1	2,56E-004	6,55E-002
2	5,12E-004	1,31E-001
4	1,02E-003	2,62E-001
8	2,05E-003	5,24E-001
16	4,10E-003	1,05E+000
32	8,19E-003	2,10E+000
64	1,64E-002	4,19E+000
128	3,28E-002	8,39E+000
256	6,55E-002	1,68E+001

Tableau 1 : valeurs de T, en fonction du prescaler et de *n*

1) Quelle est la valeur de ?

2) Quel registre faut-il considérer dans chaque cas pour :
   (a) configurer le TIMER0 (on/off, prescaler, etc.) ?
   (b) configurer et mettre en œuvre les interruptions (autorisation, etc.) ?
   Complétez les extraits de programme page suivante en conséquence aux emplacements désignés par _ _. Vous rendrez cette page avec votre copie d'examen.

3) Dans les extraits de programme présentés page suivante, repérez avec des accolades les étapes principales d'une interruption.

```
;;;;;;;;;;;;;;;;;;; Programme principal

init    clrf    PORTB;Remise à zéro du port B
        movlw   h'00'
        movwf   TRISB           ;Le port B est défini en sortie

        movlw   h'83'
        movwf   _ _             ;TIMER0 On, 16bits, Prescaler 16

        rcall   tmr0_init       ;Init TMR0 pour 1s pile
        movlw   h'A0'
        movwf   _ _             ;Autorisation des interruptions

boucle
        nop
        goto    boucle          ;Boucle d'attente infinie

;;;;;;;;;;;;;;;;;;; Programme d'interruption
prog_int
        movwf   W_TEMP
        movff   STATUS, STATUS_TEMP
        movff   BSR, BSR_TEMP

        btfsc       _ _     ,2

        rcall   tmr0_overflow

        movff   BSR_TEMP, BSR
        movff   W_TEMP, WREG
        movff   STATUS_TEMP, STATUS
        retfie

tmr0_overflow
        bcf         _ _     ,2
        movlw   h'0B'
        movwf   TMR0H
        movlw   h'DB'
        movwf   TMR0L
        movlw    h'01'
        xorwf   PORTB
        return
```

Annexe

Dans tout l'énoncé, les caractéristiques du PIC utilisé sont telles que la période de l'horloge est TOSC= 0.25 10^{-6} sec. Il y a quatre coups d'horloge par cycle.

Voici quelques instructions et le nombre de cycles qui leur est associé :

movlw : 1 cycle

movwf : 1 cycle

goto : 2 cycles

return : 2 cycles

nop : 1 cycle

Le descriptif complet des instructions infsnz et dcfsnz est donné ci-dessous:

INFSNZ	**Increment f, skip if not 0**
Syntax:	INFSNZ f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest, skip if result ≠ 0
Status Affected:	None
Encoding:	0100 | 10da | ffff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

DCFSNZ	**Decrement f, skip if not 0**
Syntax:	DCFSNZ f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – 1 → dest, skip if result ≠ 0
Status Affected:	None
Encoding:	0100 | 11da | ffff | ffff
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Année universitaire 2014/2015

Site : ☐ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ■ Aix-Montperrin ☐ Aubagne-SATIS

Sujet session de : ☐ 1er semestre - ■ 2ème semestre - ☐ Session 2 Durée de l'épreuve :…1H30……..

Examen de : ☐L1/■ L2/☐L3 - ☐M1/☐M2 - ☐ LP - ☐ DU Nom diplôme : ……Licence SPI……...

Code Apogée du module : SPI42ATA Libellé du module : ……Informatique Industrielle

Document autorisé : ■ OUI - ☐ NON Calculatrices autorisées : ■ OUI - ☐ NON

Mercredi 13 Mai 2015 8H30-10H00 *Licence 2 – SPI*

Aix Marseille Université

Informatique industrielle

Le polycopié de cours est autorisé.

Une calculatrice de type collège est autorisée.

Cet examen de 1 heure 30 est composé de 2 parties :

- *des questions de cours et exercices*
- *un problème en langage Assembleur*

Note :

Des extraits de la documentation technique utile à cet examen vous sont donnés en annexe.

! Vous rendrez le sujet d'examen avec votre copie. Il contient des pages à compléter !

Questions de cours et exercices

Attention

Vous êtes notés sur clarté de vos réponses. Pour avoir le maximum de points, elles devront être justes, courtes, et précises. Utilisez vos brouillons avant de vous jeter sur vos copies d'examen.

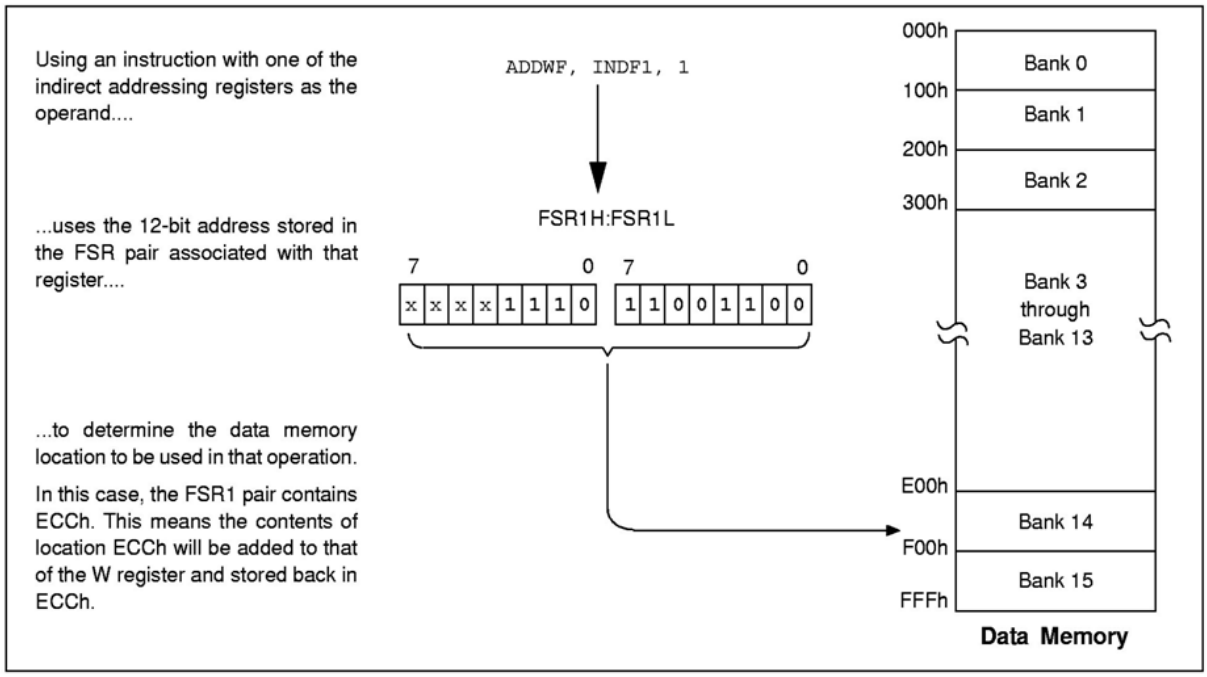
[A] Questions générales

1) Il existe deux grandes catégories de processeurs. Quelles sont-elles ? Quel est l'avantage des processeurs à nombre réduit d'instructions ?

2) Une instruction assembleur est constituée au minimum de deux parties. Quelles sont-elles ? De combien de bits sont-elles constituées ?

3) L'adressage indirect fait intervenir les registres FSR (FSR0, FSR1, ...) qui contiennent l'adresse de la case mémoire à modifier ou utiliser par un code assembleur (voir figure ci-dessous).

FIGURE 5-8: INDIRECT ADDRESSING



Considérez le code composé des trois instructions suivantes:

```
MOVLW 0x55    ; W est chargé avec la valeur 0x55
LFSR  1, 0x40    ; le registre FSR1 est chargé avec la valeur d’adresse 0x40
MOVWF INDF1 ; INDF1 est l’adresse de FSR1.
```

Et répondez succinctement aux questions suivantes:

a) Vers quel emplacement de la RAM le registre FSR1 pointe-t-il ?

b) Quelle est la valeur inscrite dans cet emplacement RAM après exécution de ces trois instructions ?

c) L'instruction MOVWF est-elle utilisé en adressage immédiat ou direct ? Justifiez votre réponse.

d) En utilisant les termes 'FSR1' et 'INDF1', justifiez en deux ou trois lignes maximum le fait que l’on appelle le mode d’adressage utilisé un mode d'adressage ‘indirect’.

[B] Exercice « codage binaire, hexadécimal et décimal »

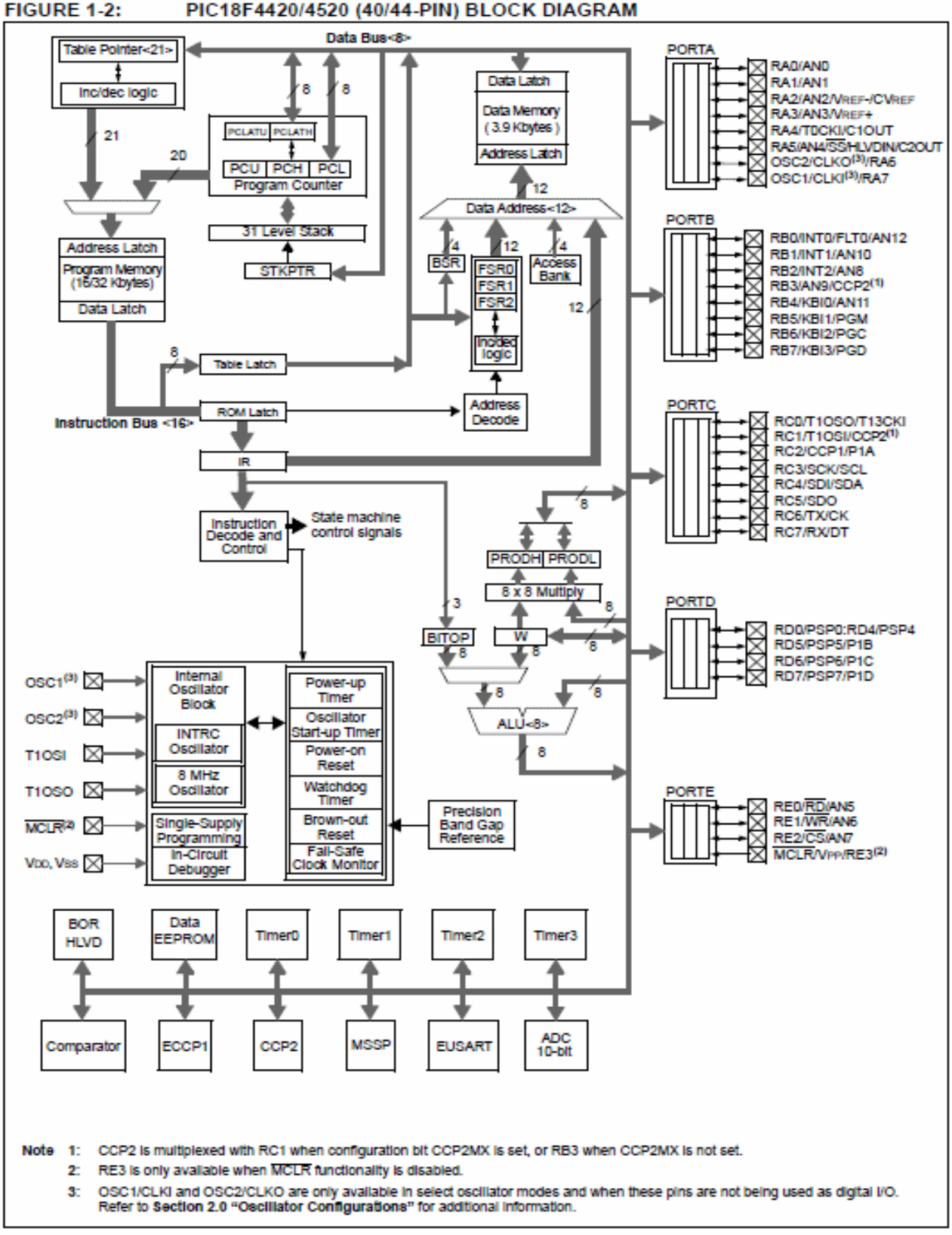
1. Convertissez a=0xAB en binaire.
2. Convertissez b=0xEF en binaire.
3. Calculez le complément à 2 de la version binaire de b.
4. En utilisant les questions 1, 2 et 3, effectuez l'opération a-b en binaire. Le résultat est-il positif ou négatif ? Justifiez votre réponse en vous fondant sur le résultat en binaire de l'opération a-b.
5. Donnez le résultat sous forme hexadécimale.
6. Effectuez l'opération *logique* suivante : c xor w où c=0110 et w=1001 (xor : ou exclusif).

[C] Questions « architecture »

Identifier les éléments ci-dessous sur le schéma bloc du microcontrôleur montré en *figure 1* :

1. le compteur programme,
2. le bus de données,
3. le bus d'instructions,
4. l'unité arithmétique et logique,
5. la mémoire programme,
6. les ports entrées-sorties.

FIGURE 1-2: PIC18F4420/4520 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when MCLR functionality is disabled.

3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

Figure 1 : Schéma de principe du PIC 18f4520.

[D] Exercice « complexité »

Considérez la fonction ‘poweroftwo’ ci-dessous qui calcule successivement les valeurs de 2^n avec n nombre entier entre 0 et 3, et place ces valeurs dans le PORTB.

```
poweroftwo
        movlw       1
        movwf       PORTB
mult    movlw       d'2'
        _ _ _ _     PORTB
        movff       PORTB,WREG
        _ _ _ _     d’ _ _ _ _’
        return
        goto        mult
```

1) Remplacez les trous dans le programme (signalés ainsi : _ _ _ _ _) par un nombre décimal ou une instruction à choisir parmi les instructions ci-dessous :

-**cpfseq** “compare file skip if equal”
-**cpfslt** “compare file skip if less than”
-**cpfsgt** “compare file skip if greater than”

Pour les trois instructions ci-dessus une comparaison est faite entre le registre W et le nombre en opérande. Le nombre de cycles nécessaires pour chacune d’elle est déterminé ainsi : 1 cycle si on ne saute pas l’instruction suivante, 2 cycles si on saute l’instruction suivante et que c’est une instruction sur un mot, 3 cycles si on saute l’instruction suivante et que c’est une instruction sur deux mots.

-**mullw** effectue la multiplication entre w et le nombre en opérande
-**mulwf** effectue la multiplication entre w et le registre en opérande et place le résultat dans le registre en opérande.

Pour les deux instructions ci-dessus une multiplication est faite entre W et un nombre ou un registre (1 cycle chacune).

Nombre de cycles pour les autres instructions : return 2, movlw 1, movwf 1, movff 2.

2) Donnez l'algorigramme du programme ‘poweroftwo’

3) On utilise une carte de test PICDEM2PLUS munie de 4 LEDs et de boutons poussoirs. Les boutons poussoirs sont connectés au PORTA, les LEDs sont connectées au PORTB. Qu’observe-t-on lorsqu’on met la carte sous tension ?

4) On utilise un quartz à 4 Mhz. Donnez le nombre de cycles nécessaires pour faire tourner l’algorithme et sa durée en microsecondes.

Problème : interruption de débordement du TIMER0

Considérons le programme assembleur suivant :

```
        LIST P=18F4520
        #include <P18F4520.inc>
        #include <CONFIG.inc>

;----- Déclaration de variables
        CBLOCK 0x00
                W_TEMP  : 1
                STATUS_TEMP : 1
                BSR_TEMP : 1
        ENDC


;----- Programme
        org     h'08'
        goto    init

        org     h'08'
        goto    routine_int

init    clrf    PORTB
        movlw   h'00'
        movwf   TRISB
        movlw   h'83'
        movwf   T0CON
        rcall   tmr0_init
        movlw   h'A0'
        movwf   INTCON

boucle  nop
        goto    boucle

tmr0_init
        movlw   h'FF'
        movwf   TMR0H
        movlw   h'FD'
        movwf   TMR0L
        return

;----- Routine d'interruption
routine_int
        movwf   W_TEMP
        movff   STATUS, STATUS_TEMP
        movff   BSR, BSR_TEMP
        btfsc   INTCON,2
        rcall   tmr0_overflow
        movff   BSR_TEMP, BSR
        movff   W_TEMP, WREG
        movff   STATUS_TEMP, STATUS
        retfie

;----- Interruption de débordement du TIMER0
tmr0_overflow
        bcf     INTCON,2
        rcall   tmr0_init
        movlw   h'03'
        xorwf   PORTB
        return

        END
```

Questions :

1) Repérez par des accolades les étapes de l'interruption.
2) Il y a une erreur dans le vecteur RESET : laquelle et pourquoi ? Quelle est la conséquence de cette erreur ?
3) Supposons cette erreur corrigée, quel est l'effet de l'interruption sur les LEDs connectées au PORTB ?
4) Marquez par a) et b) les instructions qui permettent:
   a) de paramétrer l'autorisation Générale des IT et TMRO IT;
   b) de choisir TIMER0 On, 16bits, Prescaler 16.
5) La routine d'interruption inclut un temps d'attente entre deux débordements du TIMER0. Ce temps d'attente est-il à votre avis inférieur ou supérieur à 1 seconde ? Justifiez brièvement votre réponse.

ANNEXE : extrait du datasheet du PIC 18F4520

CPFSEQ — **Compare f with W, skip if f = W**

Syntax: CPFSEQ f {,a}
Operands: $0 \le f \le 255$, $a \in [0,1]$
Operation: (f) – (W), skip if (f) = (W) (unsigned comparison)
Status Affected: None
Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words: 1
Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```
Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

MULLW — **Multiply literal with W**

Syntax: MULLW k
Operands: $0 \le k \le 255$
Operation: (W) x k → PRODH:PRODL
Status Affected: None
Encoding:

0000	1101	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example:
```
MULLW   0C4h
```
Before Instruction
W = E2h
PRODH = ?
PRODL = ?
After Instruction
W = E2h
PRODH = ADh
PRODL = 08h