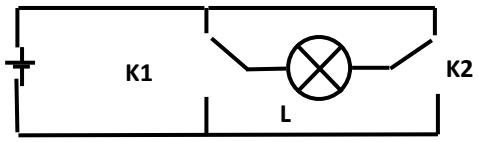
Errata: Dans le circuit XOR du cours précédent, j'ai été un peu vite en mettant une LED. Mais Problème la LED ne s'allume que dans un seul sens. Il faut utiliser le circuit ci-dessous



Les Microprocesseurs

I) Qu'est-ce Programme?

Supposons que vous soyez le chef de chœur d'une chorale, et que vous donniez des ordres aux choristes. Ces ordres pourraient être:

Chanter la chanson 1

Chanter la chanson 2

Chanter la chanson 3

Chanter la chanson 4

Faire la salutation

L'ensemble de ces ordres formes le programme de la chorale. Les choristes on appris à exécuter ces ordres.

Un microprocesseur (µP) sait lui aussi exécuter certains ordres? Ceux-ci on été choisis par le fabricant du microprocesseur. Comme les choristes n'interprètent qu'une chanson à la fois, un µP n'execute qu'un ordre à la fois, mais il le fait beaucoup plus vite (de l'ordre de la microseconde = 0.000001s) et encore certains vont même beaucoup plus vite.

II) Les différents ordres

1) les commandes

Lorsque le chef de chœur donne l'ordre de chanter la chanson 1 et les choristes effectuent cet ordre immédiatement, cet ordre s'appelle une commande. Lorsque sous windows vous faites demarrer/executer vous lancez une commande.

2) les instructions

Les ordres qui sont dans un programme s'appellent des instructions. Elles ne sont exécuter que quand les programme est exécuté.

III) Comment sont codés les instructions?

En informatique tout est en binaire. Une instruction est donc codée avec des 0 et des 1. Voici par exemple une instruction mettent à 0 un endroit particulier de la mémoire:

% 00000100000000. Pour s'en souvenir plus facilement, on peut l'écrire en hexadécimal. Pour cela on place les bits 4 par 4 de la façon suivante: % 00 0001 0000 0000

Puis on utilise le tableau de conversion que l'on a vu dans les leçons précédentes. Cela donne:

$ 0100.

IV) Le langage assembleur.

Bien que le nombre hexadécimal $100 soit un peu plus facile à retenir, il ne nous renseigne pas sur ce que fait l'instruction. De plus un programme du genre:

$100	**% 00 0001 0000 0000**
$001	**% 00 0000 0000 0001**
$ 3 FFF	**% 11 1111 1111 1111**

Qu'il soit écrit en binaire ou en hexadécimal est bien difficile à comprendre, à lire et à modifier. (Bien que les pionniers de l'informatique ont fait ainsi!) Pour remédier à ceci on a utilisé un langage appelé langage assembleur. A chaque instruction on a fait correspondre un groupe de quelques lettres appelées mnémoniques, expliquant ce que fait l'instruction.

Exemple: A la place de $100 on écrit CLRW (de l'anglais Clear W efface la mémoire w)

Un programme écrit en langage assembleur peut avoir l'allure suivante:

CLRW	**(efface la mémoire w)**
NOP	**(ne fait rien du tout)**
MOVLW 0	**(met 0 dans w)**

Ce programme en langage assembleur n'est pas compréhensible directement par le microprocesseur, il faut transformer ces instructions en binaire. Pour cela on utilise un programme spécial appelé assembleur qui transforme le langage assembleur en binaire. (On dit que l'on assemble le programme.)

V) L'assembleur des microcontrôleurs PIC.

Les PIC du type les plus courant (16F84) utilisent un nombre d'instructions très faible. On dit qu'ils sont à structure RISC (reduced instruction set computer). Ils ont 35 instructions qui contiennent toutes 14 bits au total. (Comme le CLRW 00 0001 0000 0000)

Le tableau suivant vous donne les 35 mnémonics et leur correspondance en binaire?

PIC16F87X

TABLE 13-2: PIC16F87X INSTRUCTION SET

Les 35 mnémoniques

Les 35 nombres binaires correspondants

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

VI) mémoire de microprocesseur

1) une définition pour l'informatique

L'informatique est la sciences étudie les échanges et le traitement dles informations.

(Par exemple: Les informations sont prises quelques part, éventuellement modifiées, puis enregistrées autre part)

{les informations étant prises et stockées dans des mémoires}

2) Entrée

Lorsqu'un microprocesseur lit une information, l'information va du périphérique dans le processeur. On parle d'une entrée.

Exemple de périphériques d'entrée: scanner, souris, webcam

3) Sortie

Lorsqu'un microprocesseur place une information à, un périphérique, ona unpériphérique de sortie

Exemple: imprimante, écran (s'il n'est pas tactile), haut parleur etc…

Remarque ilexiste des périphériques d'entrée sortie exemple disque dur etc…

Certaines broches des microcontrôleurs peuvent être des sorties (le microprocesseur la place à 0V ou à 5V), d'autres des broches d'entrée, le microprocesseur regrade si l'on a appliqué une tension 0 ou 5V.

4) différence entre un microprocesseur et un microcontrôleur

Dans un ordinateur, on peut trouver:

-De la mémoire Ram que l'on peut lire et écrire et qui se réinitialise à 0 lorsque l'on éteint l'ordinateur. Ce sont en particulier les barrettes de mémoire (DDR3) que vous pouvez installer dans le PC.

-De la mémoire Rom (ou équivalent) qui contiennent par exemple les programmes du BIOS (exemple le mot de passe du bios). En général on ne fait que les lire.

- un (ou plusieurs) microprocesseurs. (Exemple un intel pour les PC). Ces microprocesseurs

Prennent les informations dans la mémoire, éventuellement peuvent les traiter, et replacent les résultats en mémoire. Cependant, le mémoires Rom ou ram citées ont un temps d'accès assez long au vu de la vitesse d'un microprocesseur. Les microprocesseurs ont des mémoires

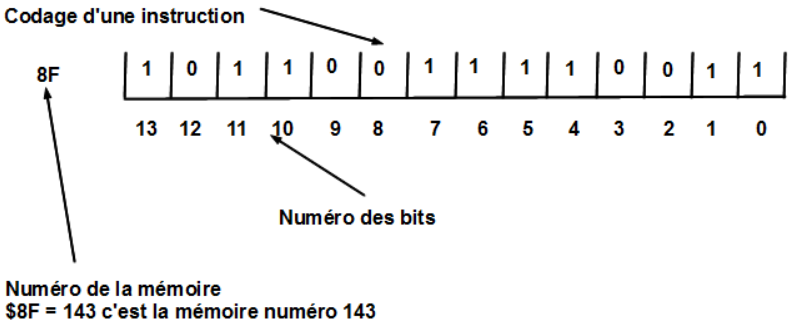
intégrées appelées des REGISTRES, qui permettent de stocker des informations. Si le temps d'accès à ces registres est très court (car insérés par le fabriquant), ils sont par contre moins nombreux (en général moins d'une centaines).

Pour les pics de milieu de gamme genre 16F84, les registres ont une contenance de 8bits (sauf exception), ils ne peuvent pas contenir un nombres supérieur à 255, c'est-à-dire qu'ils s' écrivent avec deux chiffres exadécimaux. (exemple FF).

Les microcontrôleurs eux contiennent toutes ces mémoires (en moins grand nombre que dans les PC, plus quelques périphériques comme des « chronomètres », des comparateurs, des « voltmètres » etc…

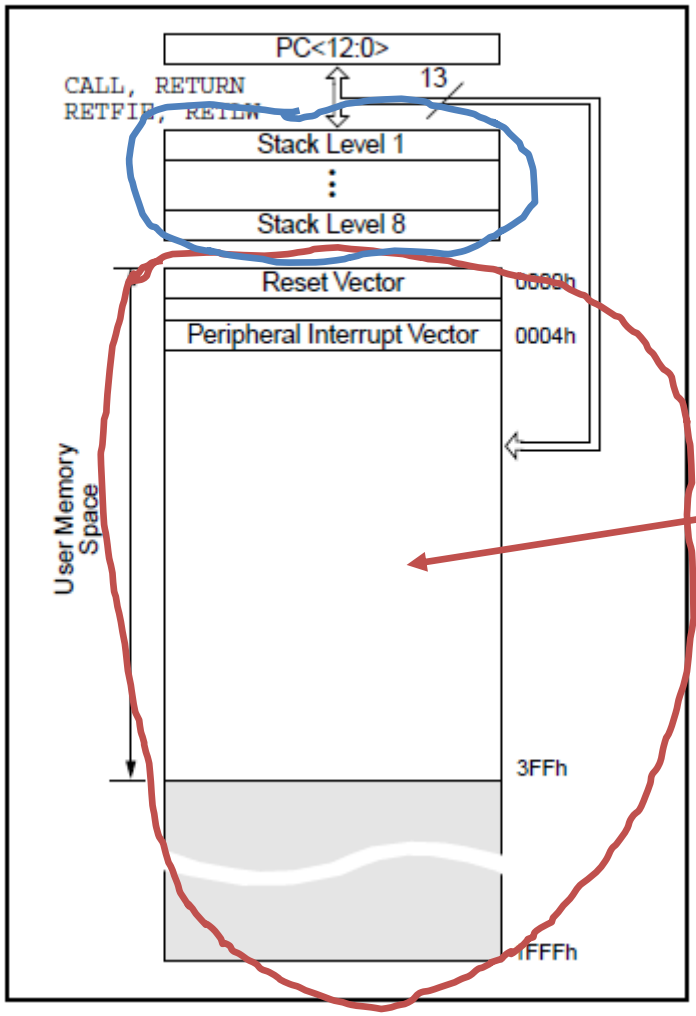
5) les mémoires contenues dans un microcontrôleur PIC

La mémoire de programme: Ces petites bêtes se programment, ils faut donc qu'elles puissent mémoriser des programmes. Ces programmes sont sous la forme d'un nombre binaire de 14 bits numérotés de 0 à 13



Cette mémoire de programme n'est en général écrite lors de la programmation du pic.

FIGURE 4-2: PROGRAM MEMORY MAP AND STACK - PIC16F84/CR84



La mémoire de programme.

La première mémoire a pour numéro $0000(on dit adresse), et la dernière $3FF = 1023.

Soit en tout 1024 mémoires = 1kilo octet.

On ne pourra pas avec le 16F84 faire des programmes de plus de 1K oct. (Mais programmées en assembleur, c'est déjà beaucoup) .

A l'adresse 0000 se trouve le vecteur de reset. Le contenu de cette mémoire indique au 16F84 ou commence le programme lors du démarrage ou d'une remise à zéro.

A L'adresse $0004 se trouve le vecteur d'interruption. Le contenu de cette mémoire indique au microprocesseur où se trouve la partie du programme traitant cette interruption. Supposons que l'on ait demander au chronomètre (Timer) de nous prévenir quand il se sera écouler 1ms. Si une interruption a été mise , le programme s'arrêtera et ira continuer à l'adresse contenue dans le vecteur d'interruption.

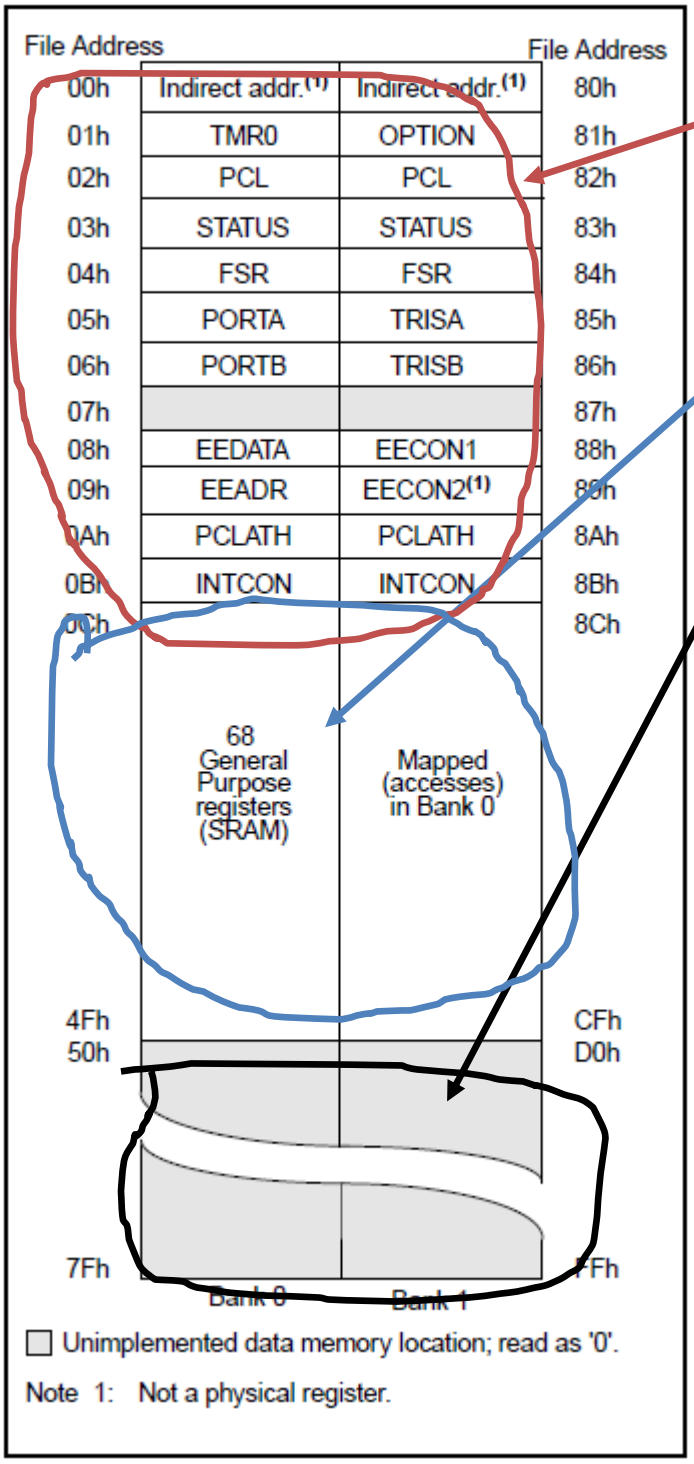
Remarquer la façon d'écrire les nombres hexadécimaux: 0000h au lieu de $0000. souvent l'on trouve 0x0000

La pile, permet de sauvegarder des adresse pour les sauts et sous-programmes. (on verra plus tard)

Les registres:

Ce sont les mémoires situées dans le microprocesseur lui-même. Comme dans les microcontrôleur tout est embarqué (mémoires, périphériques etc…) toutes les ram pourraient par construction être des registres. De fait, la ram sert à stocker des données et une partie de celles-ci sont des registres pouvant agir sur le fonctionnement du microcontrôleur.

FIGURE 4-2: REGISTER FILE MAP - PIC16F84/CR84



Voici la mémoire des données.

Elle contient des registres

Et

Mémoires que l'on utilisera suivant notre gré (des variables)

Et des zones inutilisables

Remarque, ces mémoires sont réparties en deux banques (on verra cela plus tard)

La mémoire EEPROM: La microcontrôleur peuvent aussi contenir de la mémoire EEPPROM. Elle peut servir à enregistrer des données qui seront rarement modifiées, votre indicatif par exemple.

Regardons le microcontrôleur 12F675 et voyons ce qu'il contient:

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)				
PIC12F629	1024	64	128	6	–	1	1/1
PIC12F675	1024	64	128	6	4	1	1/1

* 8-bit, 8-pin devices protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

1024 mémoires de programme (1Koct) en mots de 14 bits

64 octets utilisables pour stocker vos données qui s'effacent quand l'alimentation est supprimée (ou un RESET)

128 octets d'EEPROM pour un stockage après arrêt de l'alimentation

6 broches qui seront au chois en entrée (on lit s'il y a 5V ou 0V) ou en sortie (on met 5V ou 0V)

1 convertisseur analogique->digital (votltmètre) permettant de lire 4 tensions différentes.

1 comparateur (pour savoir si une tension est plus grande qu'une autre)

1 timer (chronomètre) pouvant compter jusqu'à 256 (8bits)

1 timer 16 bits pouvant compter jusqu'à 65536