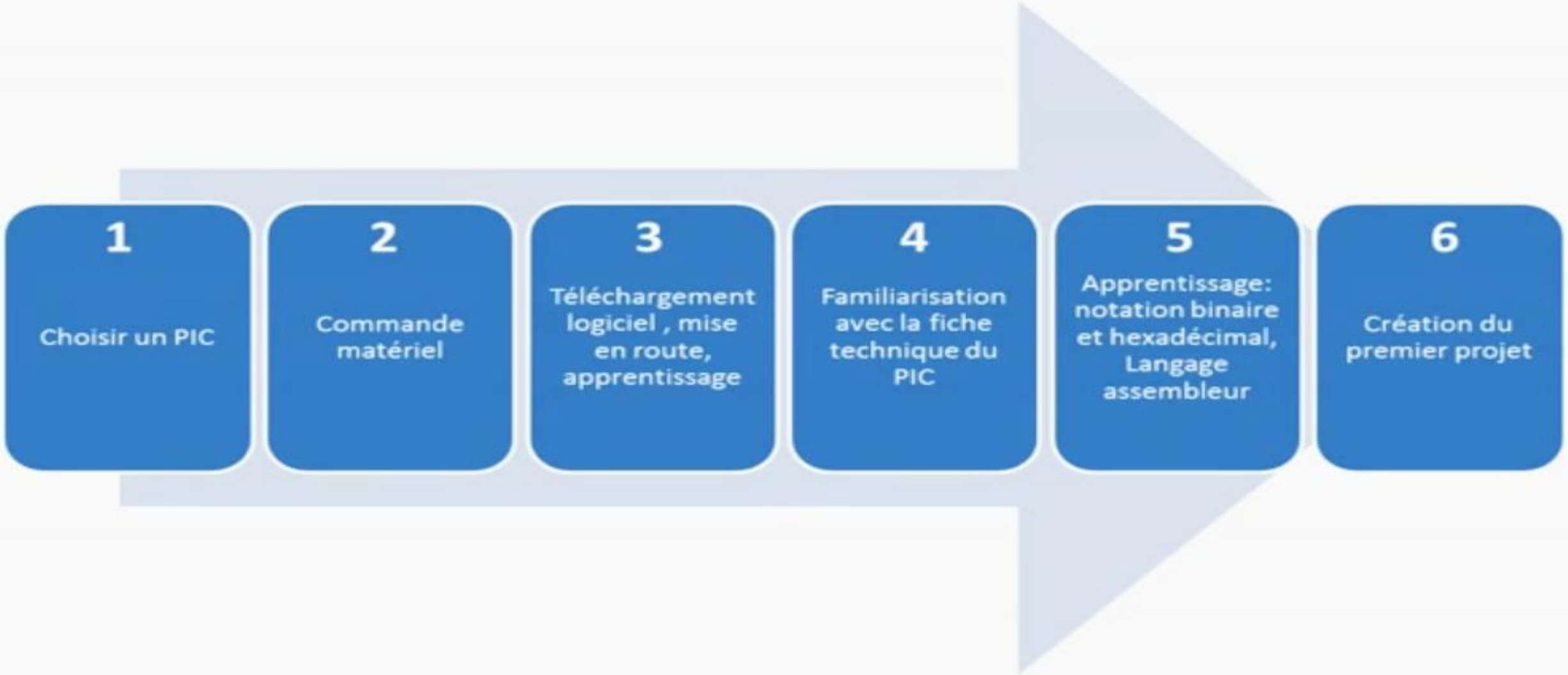
UNIVERSITÉ CHOUAIB DOUKKALI
Ecole Supérieure de Technologie
Sidi Bennour

Cours: Physique pour l'Informatique

« PIC, programmation assembleur»

Année, 2018-2019

PIC, programmation assembleur

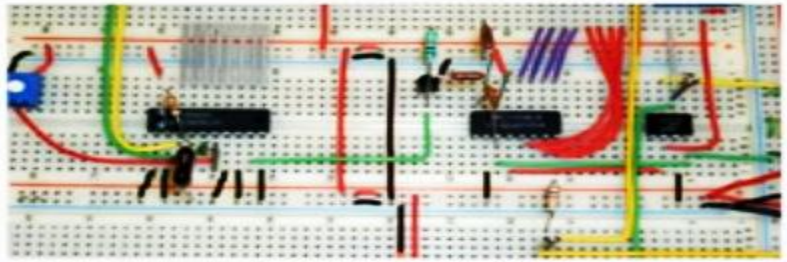


Choix du PIC

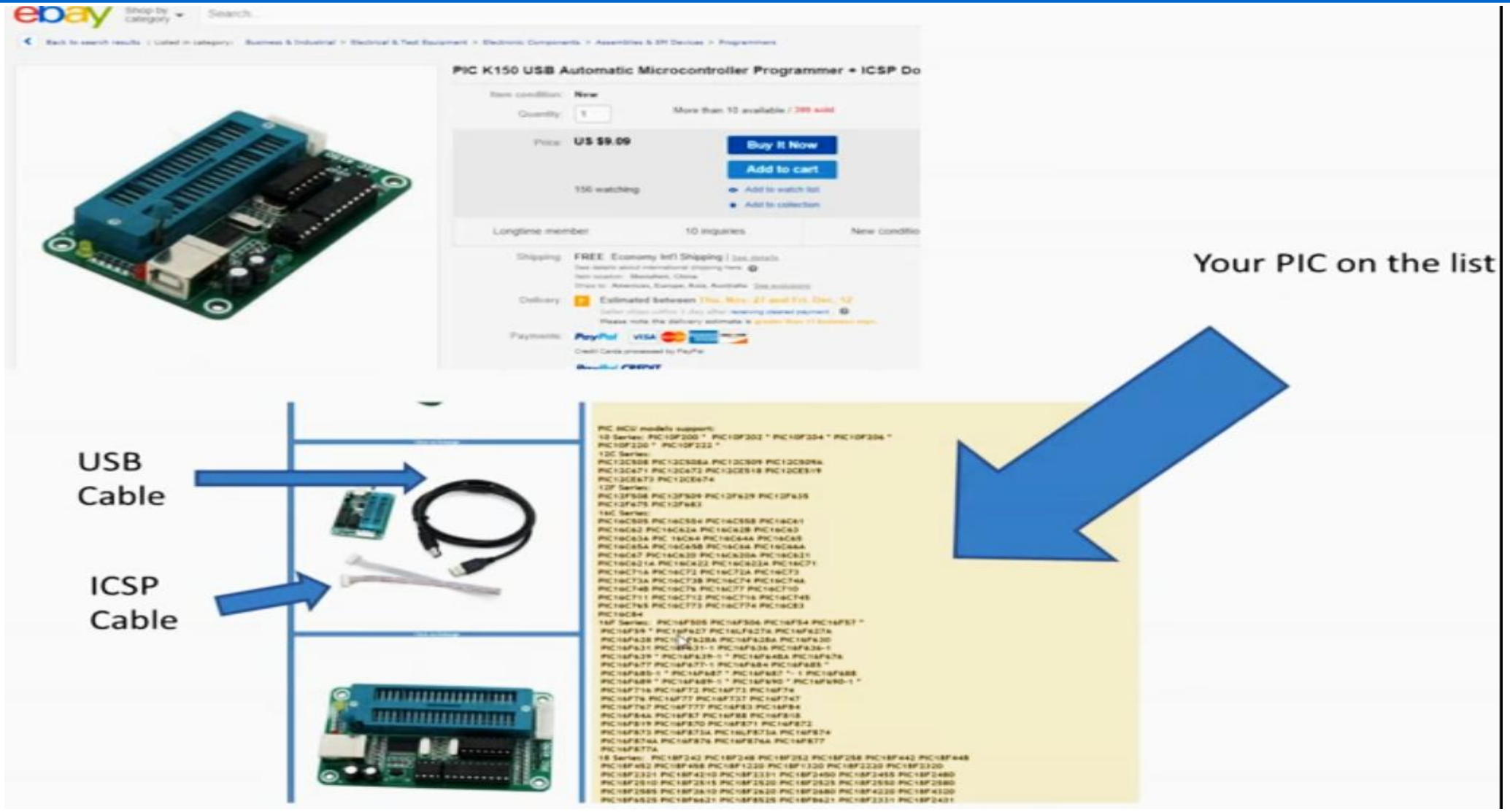
- Choisir un PIC simple et commun
  - PIC 16F84 de Microchip est un bon choix
- Recherchez le # du PIC sur Google
  - Quantité et qualité d'information
  - Assurez-vous de trouver la fiche technique
- Recherchez un projet simple
  - DEL clignotant (Google avec pic #)
  - Identifiez le matériel requis
  - Recherchez le code ASM (assembleur)

Commande du matériel

- List of matériel
  - PIC
  - Xtal, resistor, capacitor, LED
  - PIC programmer
  - Platine d'essaie (Breadboard)



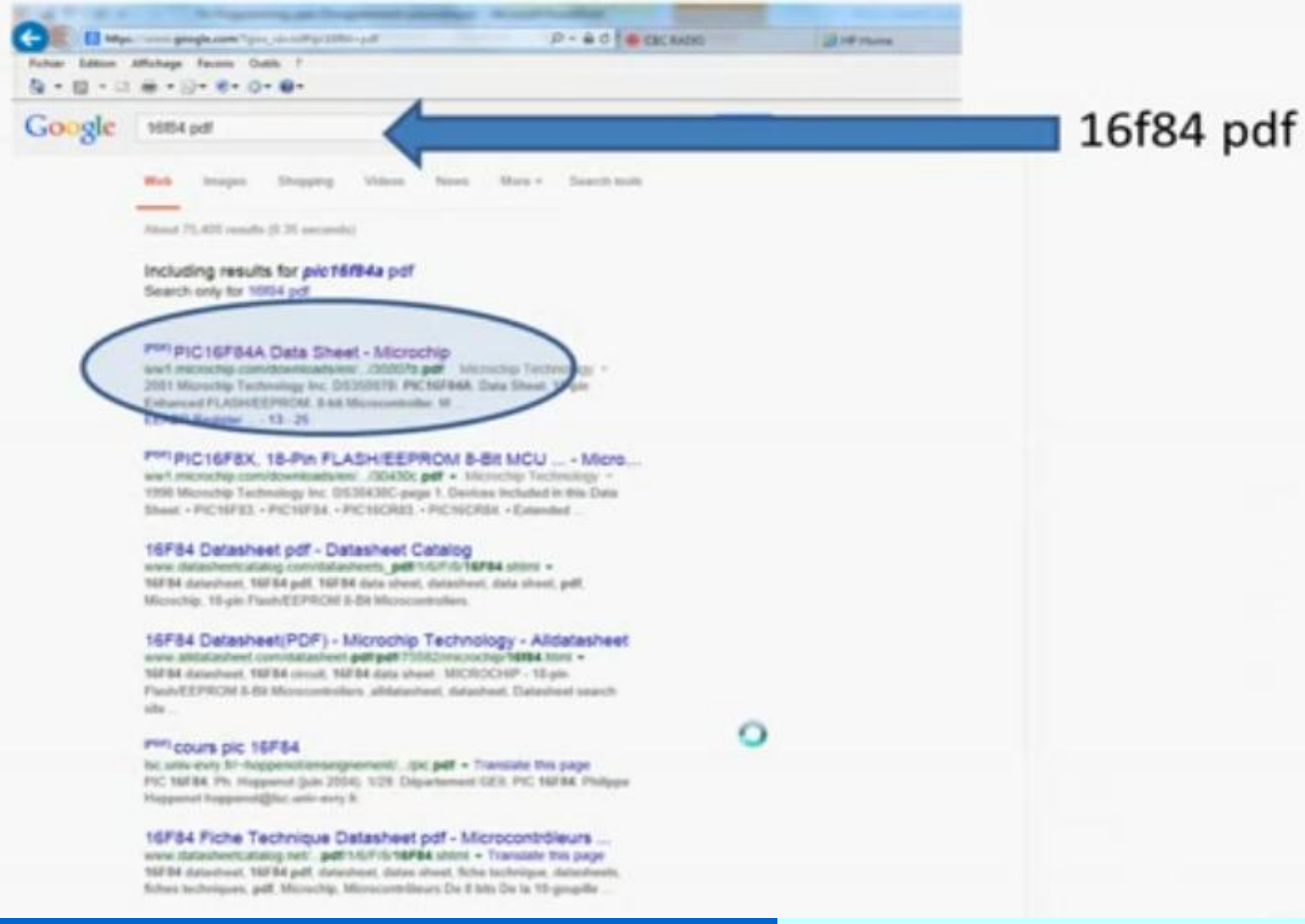
PIC, programmation assembleur



Integrated Development Environment (IDE)

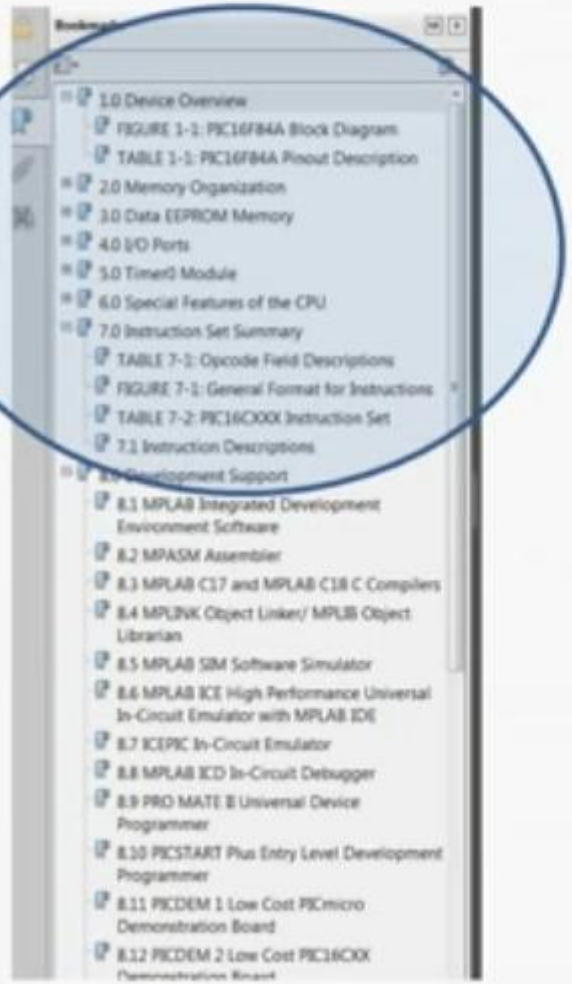
- IDE est un logiciel qui fonctionne sur un PC (Windows®, Mac OS®, Linux®) pour développer des applications pour les microcontrôleurs Microchip et les contrôleurs de signaux numériques.
- C'est ce qu'on appelle un environnement de développement intégré (IDE), car il fournit un «environnement» intégré unique pour développer du code pour les microcontrôleurs intégrés.

Get familiar with Datasheet



16f84 pdf

Révision de la fiche technique



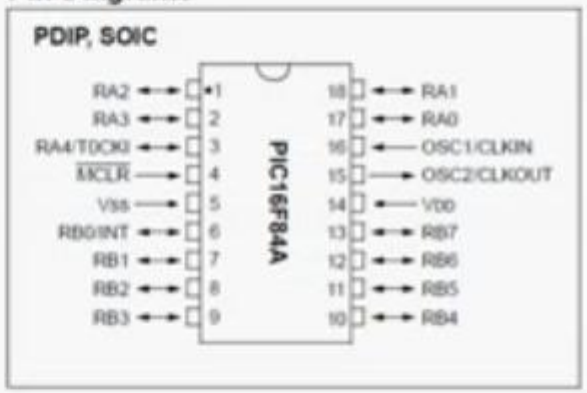
High Performance RISC CPU Features:

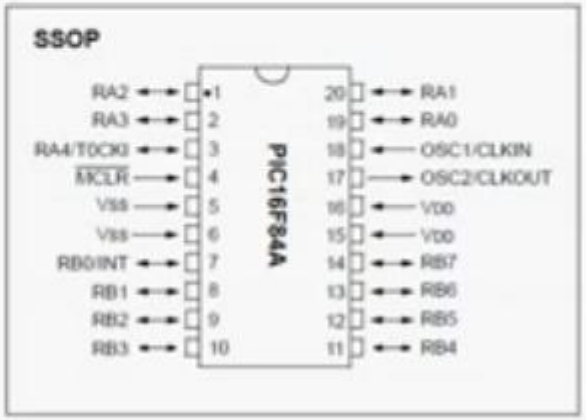
- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit

Pin Diagrams

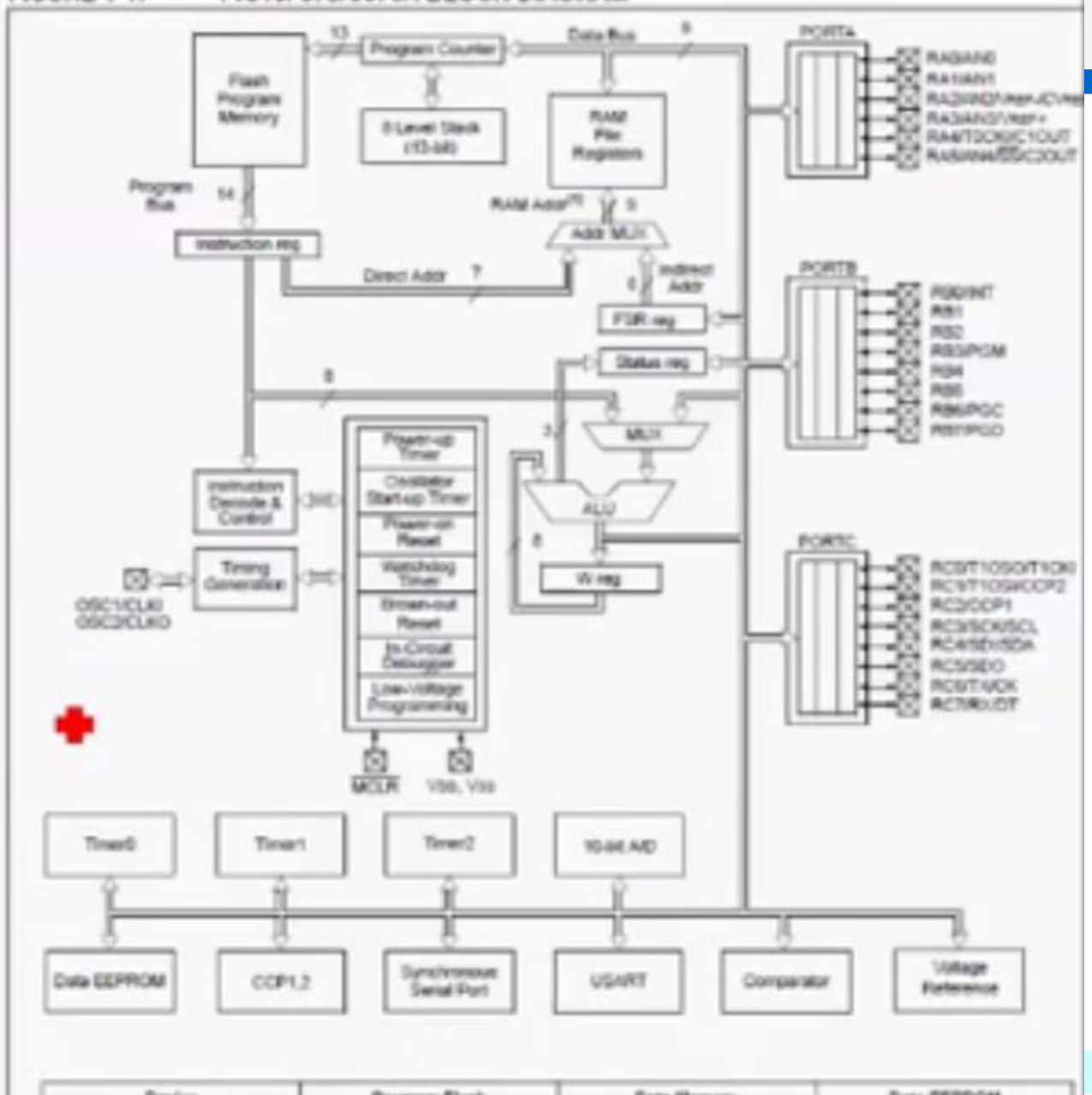




Bookmarks

PIC16F87XA

FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM



- **Bit-oriented** operations
- **Literal and control** operations

Each PIC16 instruction is a 14-bit word divided into an **opcode** which specifies the instruction type and one or more **operands** which further specify the operation of the instruction. The formats for each of the categories is presented in Figure 15-1, while the various opcode fields are summarized in Table 15-1.

Table 15-2 lists the instructions recognized by the MPASM™ Assembler. A complete description of each instruction is also available in the PIC® Mid-Range MCU Family Reference Manual (DS33023).

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven-bit constant or literal value.

One instruction cycle consists of four oscillator periods; for an oscillator frequency of 4 MHz, this gives a normal instruction execution time of 1 µs. All instructions are executed within a single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of an instruction. When this occurs, the execution takes two instruction cycles with the second cycle executed as a NOP.

> **Note:** To maintain upward compatibility with future PIC16F87XA products, do not use the OPTION and TRIS instructions.

All instruction examples use the format 'nnnh' to

TABLE 15-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

FIGURE 15-1: GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations

13 … 8	7	6 … 0
OPCODE	d	f (FILE #)

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Bit-oriented file register operations

13 … 10	9 … 7	6 … 0
OPCODE	b (BIT #)	f (FILE #)

b = 3-bit bit address
f = 7-bit file register address

Literal and control operations

General

eur

TABLE 15-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

TABLE 7-2: PIC16CXXX INSTRUCTION SET

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	

Assembler/Linker/Librarian User's Guide

4.2.1 Control Directives

Control directives control how code is assembled.

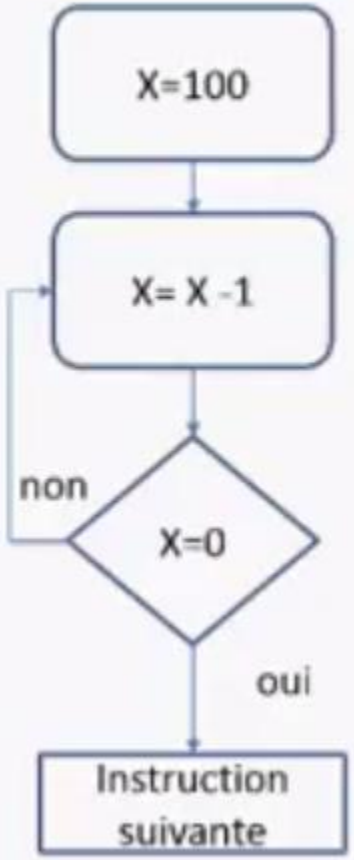
- `#define` - Define a Text Substitution Label .....
- `#include` - Include Additional Source File .....
- `#undefine` - Delete a Substitution Label.....
- `bankisel` - Generate Indirect Bank Selecting Code (PIC12/16 MCUs)..
- `banksel` - Generate Bank Selecting Code .....
- `constant` - Declare Symbol Constant.....
- `end` - End Program Block.....
- `equ` - Define an Assembler Constant.....
- `org` - Set Program Origin.....
- `pagesel` - Generate Page Selecting Code (PIC10/12/16 MCUs).....
- `pageselw` - Generate Page Selecting Code Using WREG Commands (PIC10/12/16 MCUs).....
- `processor` - Set Processor Type .....
- `radix` - Specify Default Radix .....
- `set` - Define an Assembler Variable .....
- `variable` - Declare Symbol Variable .....

4.2.2 Conditional Assembly Directives

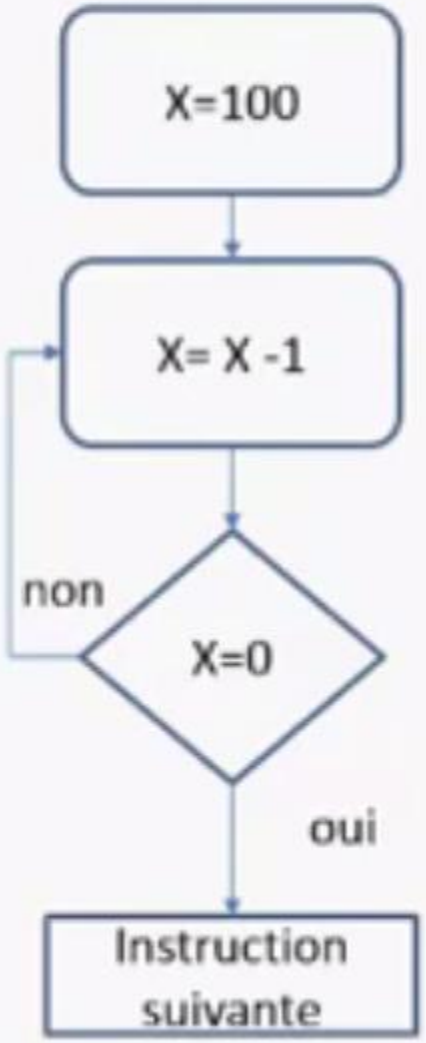
Conditional assembly directives permit sections of conditionally assemble
These are not run-time instructions like their C language counterparts. The
which code is assembled, not how the code executes.

- `else` - Begin Alternative Assembly Block to `if` Conditional .....
- `endif` - End Conditional Assembly Block .....
- `endw` - End a `while` Loop .....
- `if` - Begin Conditionally Assembled Code Block.....
- `ifdef` - Execute If Symbol Has Been Defined.....
- `ifndef` - Execute If Symbol Has Not Been Defined .....
- `while` - Perform Loop While Condition is True

Algorithme d'une boucle de délais



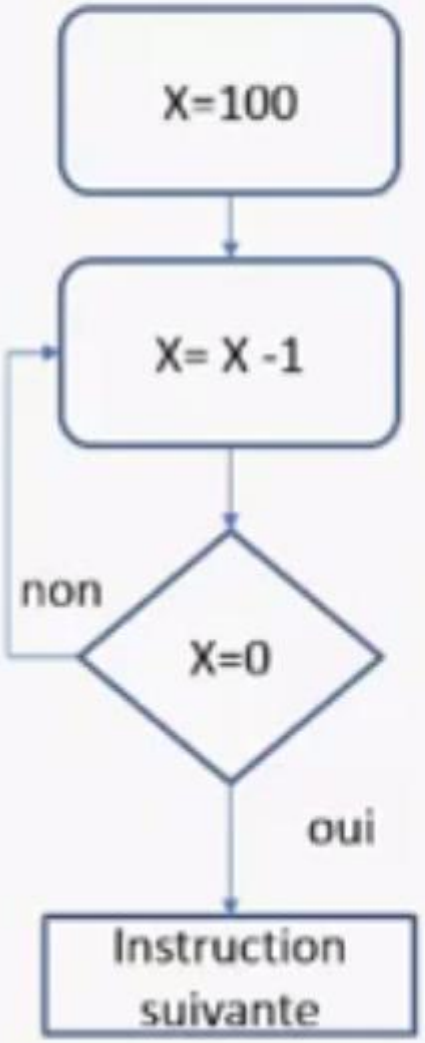
Algorithme d'une boucle de délais et code assembleur




```
COUNT1 equ 08h      ;reserve emplacement mémoire pour compteur
movlw   64h         ; charge la valeur 64 hex (=100 decimal) dans W
movwf   COUNT1      ;pousse le contenu de W dans le compteur

casino  decfsz      COUNT1,1    ;soustrait 1 de count1 .si le résultat =0
                                ;le programme saute l'instruction suivante, si
                                ;non l'instruction suivante est excécuté

        goto        casino      ;rendez vous a l'instruction ayant "casino" comme
                                ;étiquette

;instruction suivante se place ici
```

Decfsz = réduire le contenu de f de 1 et sauter l'instruction suivante si le résultat est zéro.

Algorithme d'une boucle de délais et code assembleur




```
COUNT1 equ 08h      ;reserve emplacement mémoire pour compteur
movlw   64h         ; charge la valeur 64 hex (=100 decimal) dans W
movwf   COUNT1      ;pousse le contenu de W dans le compteur

casino  decfsz      COUNT1,1   ;soustrait 1 de count1 .si le résultat =0
                               ;le programme saute l'instruction suivante, si
                               ;non l'instruction suivante est excécuté

        goto        casino     ;rendez vous a l'instruction ayant "casino" comme
                               ;étiquette
```

Decfsz = réduire le contenu de f de 1 et sauter l'instruction suivante si le résultat est zéro.

Variable «d » qui peu prendre la valeur 0 ou 1 , si 0 le résultat de la Soustraction est poussé dans le registre w , si 1 le résultat reste dans la case mémoire pointé par f

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. If the result is '1', the next instruction is executed. If the result is '0', then a NOP is executed instead, making it a 2 TCY instruction.

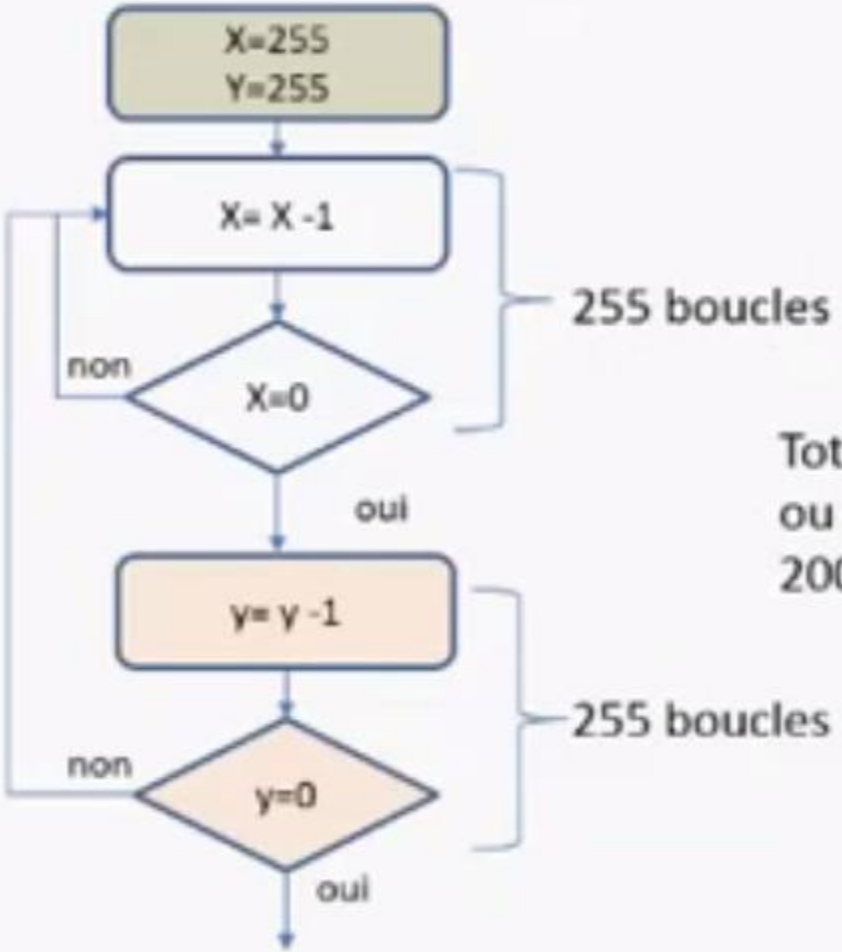
Calcul du délais en secondes

- Avec un cristal de 20 M Hz, la période de l'oscillateur est de un .000 000 005 seconde, P=1/f
- Un cycle d'instruction prend 4 périodes d'oscillateur (information disponible dans le datasheet)
- Donc un cycle prend .000 000 2 secondes à exécuter
- Le nombre de cycle par instruction est disponible sur le jeu d'instruction
- Dans le petit programme de délais suivant , chaque boucle dure 3 cycles
- Le programme répète la boucle 100 fois, donc le délais dure 300 cycles
- Le délais total = 300 X .000 000 2 sec = .000 06 sec
- La valeur maximale du compteur est de 255 décimale ou FF hex.
- Le délais maximale d'une simple boucle est donc de 255 X 3 X .000 000 2 = .000 153 sec (153 micro secondes)

Alors comment faire pour obtenir un délais plus long ?

```
casino    decfsz    COUNT1,1    ;1 cycle
          goto      casino      ;2 cycles
```

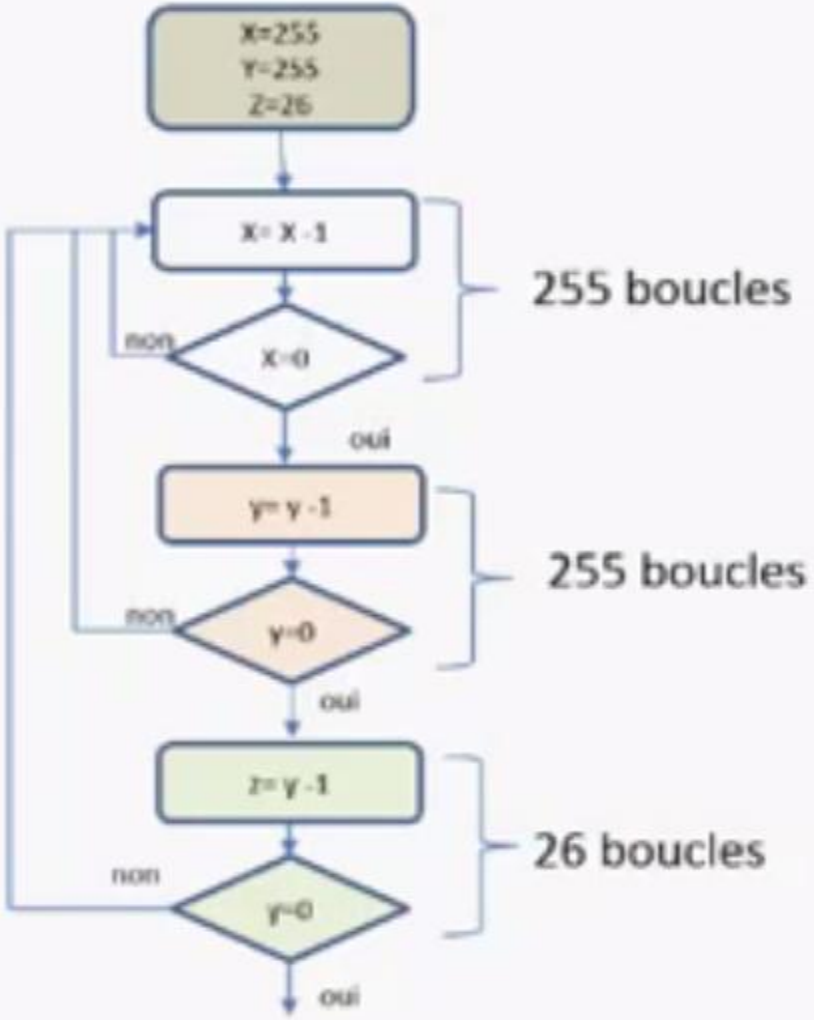
Algorithme d'une boucle de délais à 2 paliers



Total = 255 X 255 = 65 025 boucles
ou +- 200K cycles
200k X .000 000 2 = .04 sec

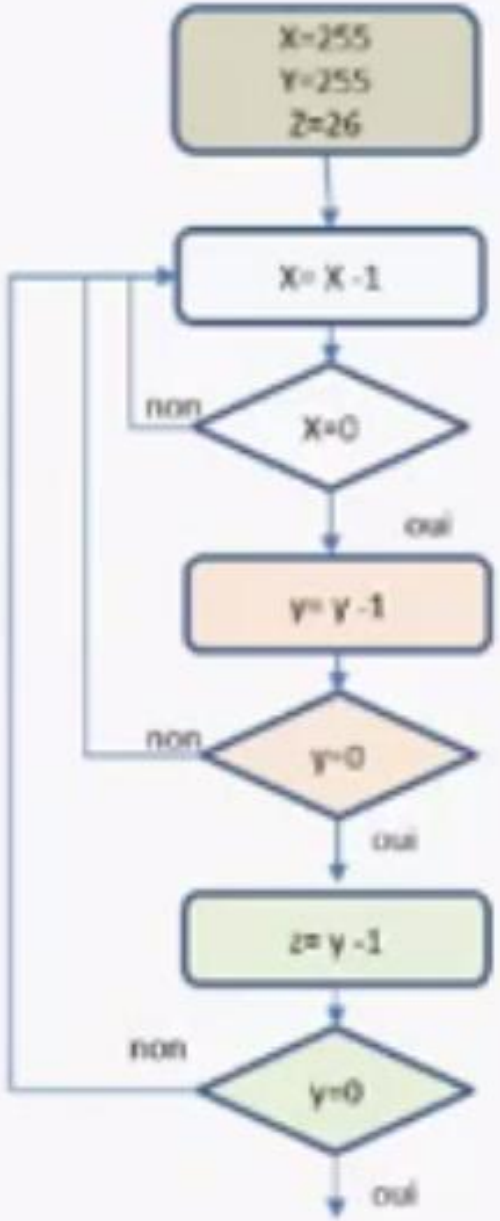
Si on ajoute un troisième boucle : 255 X 255 X 255 X 3 X .000 000 02= +- 10 secondes.

Algorithme d'une boucle de délais à 3 paliers pour 1 sec



Total = 255 X 255 X 26 = +- 1.7M boucles
ou +- 5M cycles
5M X .000 000 2 = 1sec

Code asm pour boucle de 1 sec




```
COUNTX    equ 08 h    ;reserve emplacement mémoire pour compteur
COUNTY    equ 09 h    ;reserve emplacement mémoire pour compteur
COUNTZ    equ 0A h    ;reserve emplacement mémoire pour compteur

movlw     ff h        ; charge la valeur FF hex (=255 decimal) dans W
movwf     COUNTX      ;pousse le contenu de W dans le compteur x
movlw     ff h        ; charge la valeur FF hex (=255 decimal) dans W
movwf     COUNTY      ;pousse le contenu de W dans le compteur y
movlw     1A h        ; charge la valeur 1a hex (=26 decimal) dans W
movwf     COUNTZ      ;pousse le contenu de W dans le compteur z

casino    decfsz      COUNTX,1    ;soustrait 1 de count1 .si le résultat =0
          goto        casino      ;rendez vous a l'instruction ayant "casino" comme
          decfsz      COUNTY,1    ;soustrait 1 de count1 .si le résultat =0
          goto        casino      ;rendez vous a l'instruction ayant "casino" comme
          decfsz      COUNTZ,1    ;soustrait 1 de count1 .si le résultat =0
          goto        casino      ;rendez vous a l'instruction ayant "casino" comme

;instruction suivante se place ici
```

TABLE 7-2: PIC16CXXX INSTRUCTION SET

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	

Les interruptions

Clignotant DEL avec délais

Clignotant sans délais

```
Insertion .....

STATUS equ 03h
TRISA equ 85h
PORTA equ 05h

bsf STATUS,5
moviw 00h
movwf TRISA
bcf STATUS,5

Start   moviw 02h
        movwf PORTA
délais  .....
        moviw 00h
        movwf PORTA
délais  .....
        goto Start
```

Routine délais 1 sec

```
COUNTX   equ 08 h
COUNTY   equ 09 h
COUNTZ   equ 0A h

moviw    ff h
movwf    COUNTX
moviw    ff h
movwf    COUNTY
moviw    1A h
movwf    COUNTZ

casino
decfsz   COUNTX,1
goto     casino
decfsz   COUNTY,1
goto     casino
decfsz   COUNTZ,1
goto     casino
```

Clignotant DEL avec délais
Avec sous routine

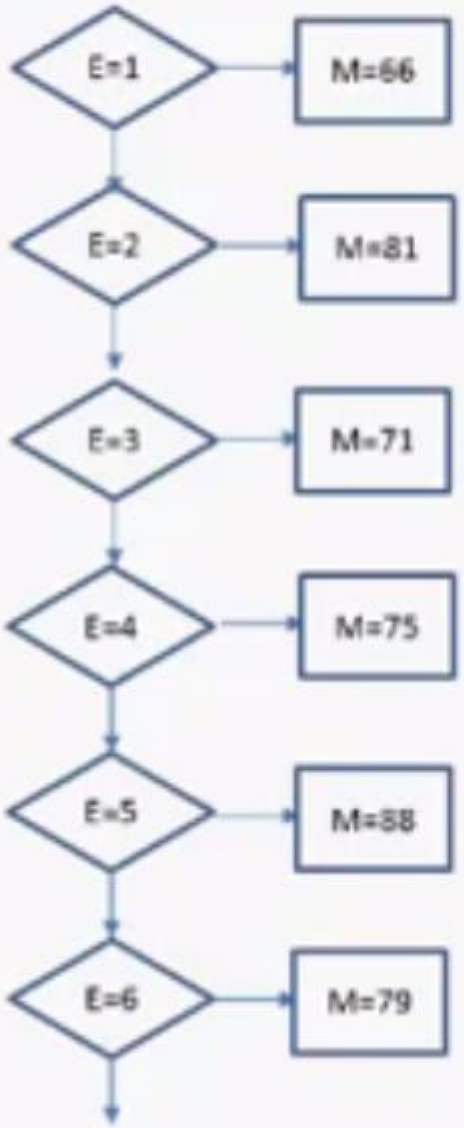
```
            COUNTX    equ 08 h
            COUNTY    equ 09 h
            COUNTZ    equ 0A h

            STATUS equ 03h
            TRISA equ 85h
            PORTA equ 05h

            bsf STATUS,5
            moviw 00h
            movwf TRISA
            bcf STATUS,5

Start       moviw 02h
            movwf PORTA
            call delais
            moviw 00h
            movwf PORTA
            call delais
            goto Start
------------------------------------
delais      moviw   ff h
            movwf   COUNTX
            moviw   ff h
            movwf   COUNTY
            moviw   1A h
            movwf   COUNTZ

casino      decfsz    COUNTX,1
            goto      casino
            decfsz    COUNTY,1
            goto      casino
            decfsz    COUNTZ,1
            goto      casino
            return
```

A la fin du programme

TABLE 7-2: PIC16CXXX INSTRUCTION SET

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	

Choix multiple



échantillon	masse
1	66
2	81
3	71
4	75
5	88
6	79

Sous routine avec table

échantillon	masse
1	66
2	81
3	71
4	75
5	88
6	79

```
        movfw   echan   ;pousser le # échantillon dans W
        Call    table1  ;sauter a la sous routine Table1
        movwf   masse   ;pousser le contenu de W dans masse

Table1  addwf   PCL     ;additionné W a l'adresse pointé par le compteur de programme
        nop             ;aucune opération
        retlw   D'66'   ;retour avec la valeur 66d dans W
        retlw   D'81'   ;retour avec la valeur 81d dans W
        retlw   D'71'   ;retour avec la valeur 71d dans W
        retlw   D'75'   ;retour avec la valeur 75d dans W
        retlw   D'88'   ;retour avec la valeur 88d dans W
        retlw   D'79'   ;retour avec la valeur 79d dans W
```

Sous routine avec table

échantillon	masse
1	66
2	81
3	71
4	75
5	88
6	79

1 → W

```
        movfw   echan   ;pousser le # échantillon dans W
        Call    table1  ;sauter a la sous routine Table1
        movwf   masse   ;pousser le contenu de W dans masse
```

66

PCL + 1

```
Table1  addwf   PCL     ;additionné W a l'adresse pointé par le compteur de programme
        nop             ;aucune opération
        retlw   D'66'   ;retour avec la valeur 66d dans W
        retlw   D'81'   ;retour avec la valeur 81d dans W
        retlw   D'71'   ;retour avec la valeur 71d dans W
        retlw   D'75'   ;retour avec la valeur 75d dans W
        retlw   D'88'   ;retour avec la valeur 88d dans W
        retlw   D'79'   ;retour avec la valeur 79d dans W
```

+1

Sous routine avec table

échantillon	masse
1	66
2	81
3	71
4	75
5	88
6	79

W = 4

```
        movfw   echan   ;pousser le # échantillon dans W
        Call    table1  ;sauter a la sous routine Table1
        movwf   masse   ;pousser le contenu de W dans masse
                75

Table1  addwf   PCL+4   ;additionné W a l'adresse pointé par le compteur de programme
        nop             ;aucune opération
        retlw   D'66'   ;retour avec la valeur 66d dans W
        retlw   D'81'   ;retour avec la valeur 81d dans W
        retlw   D'71'   ;retour avec la valeur 71d dans W
        retlw   D'75'   ;retour avec la valeur 75d dans W
        retlw   D'88'   ;retour avec la valeur 88d dans W
        retlw   D'79'   ;retour avec la valeur 79d dans W
```

TABLE 7-2: PIC16CXXX INSTRUCTION SET

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	

Interruption

- Une interruption est une procédure ou un signal qui arrête le micro d'exécuter son programme pour laisser une sous-routine s'exécuter.
  - Sources d'interruptions
    - Interne: débordement d'un compteur
    - Externe: changement d'état d'une borne du micro