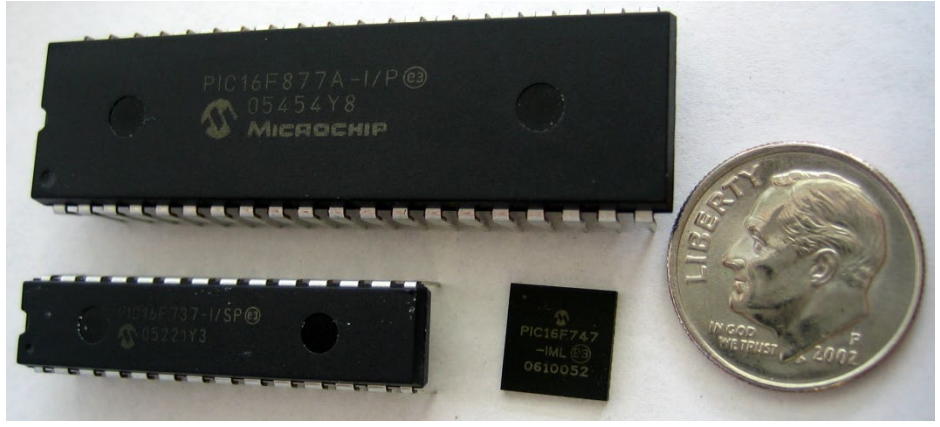
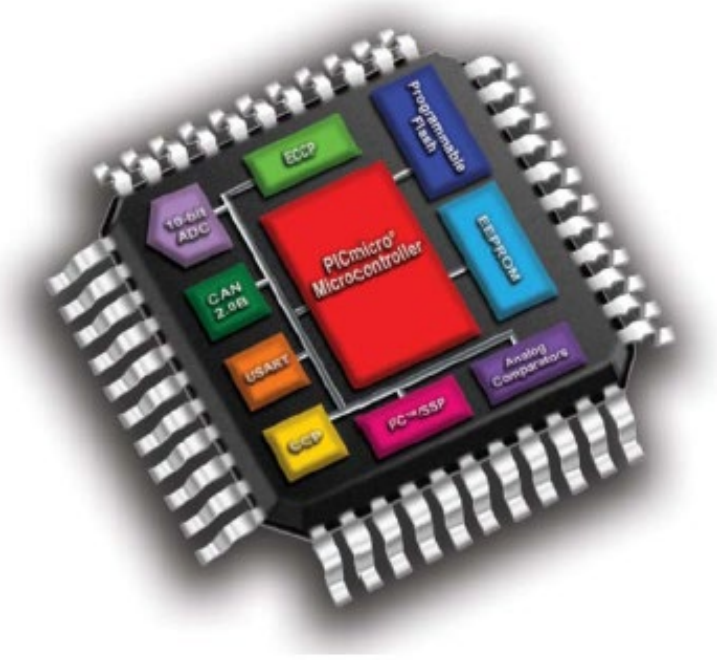
INTRODUÇÃO AOS MICROCONTROLADORES PIC





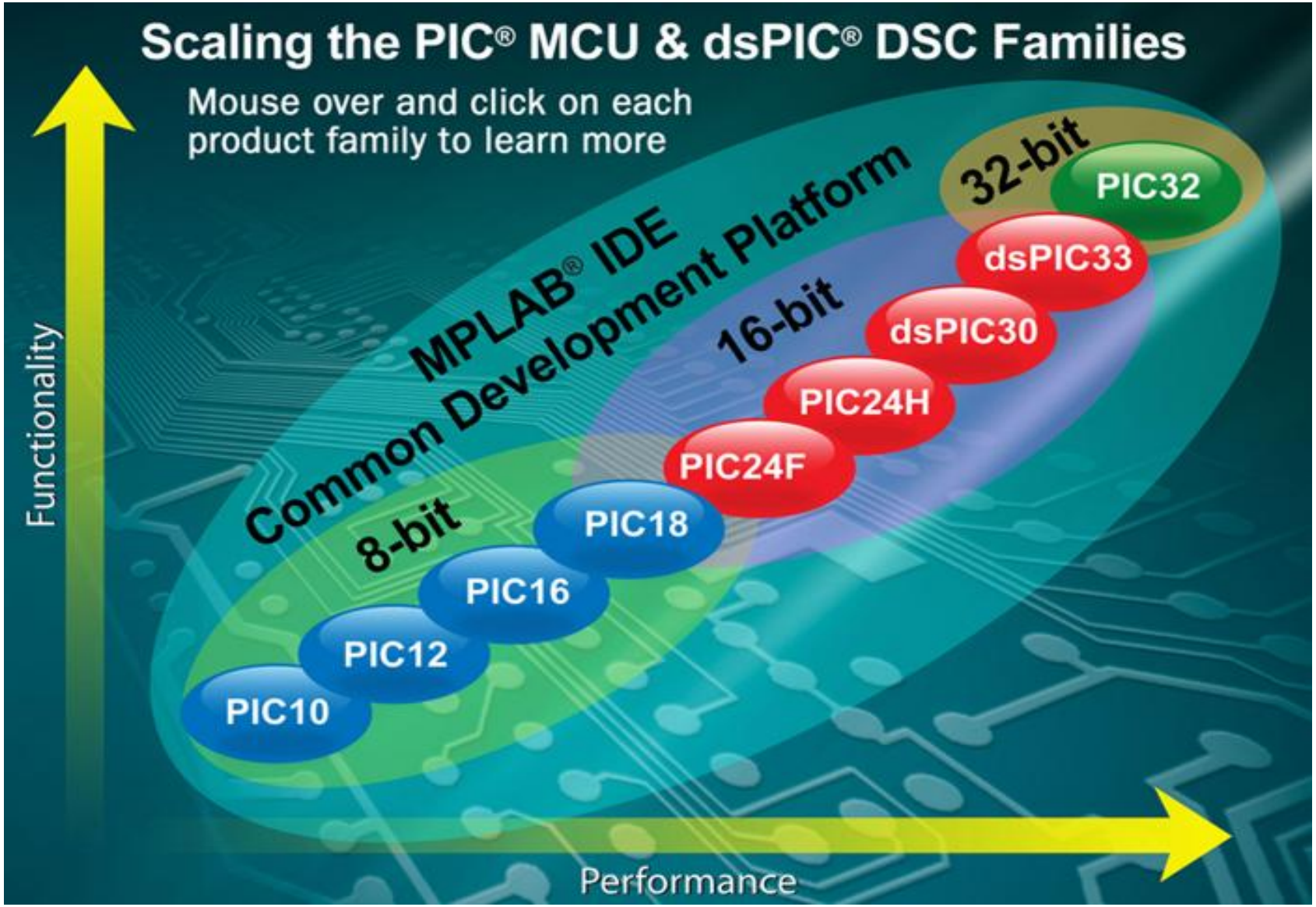
Introdução

- As circunstâncias que se nos deparam hoje no campo dos microcontroladores têm os seus primórdios no desenvolvimento da tecnologia dos circuitos integrados.
- Um crescente aumento do nível de integração, permitiu o aparecimento de circuitos integrados contendo simultaneamente processador e periféricos.
- Foi assim que o primeiro chip contendo um microcomputador e que mais tarde haveria de ser designado por microcontrolador, apareceu.

Microcontroladores versus Microprocessadores

Um microcontrolador difere de um microprocessador em vários aspectos:

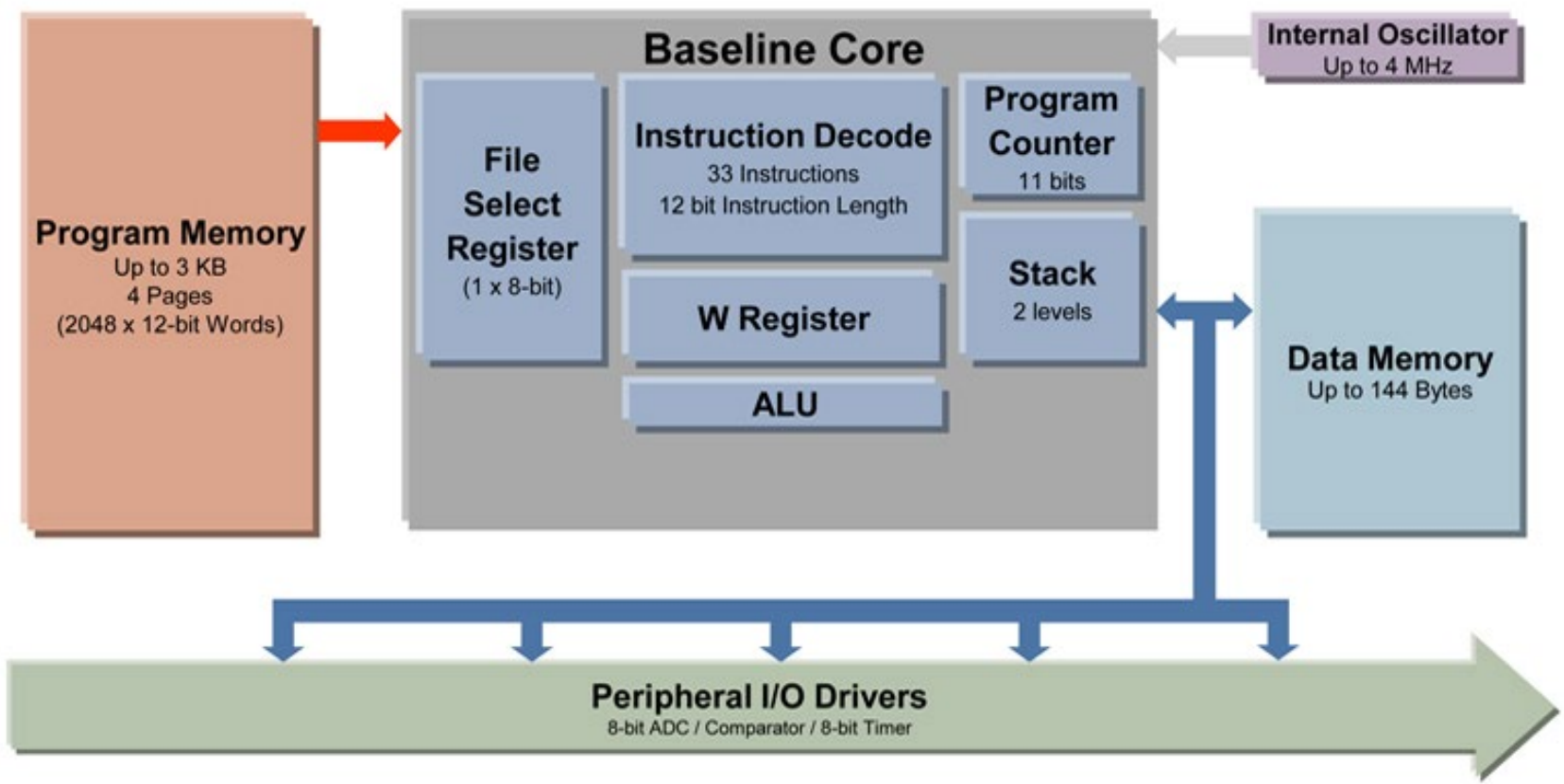
- Primeiro e o mais importante, é a sua funcionalidade. Para que um microprocessador possa ser usado, outros componentes devem-lhe ser adicionados, tais como memória e componentes para receber e enviar dados.
- **O microcontrolador foi projetado para ter tudo num só.**
- Nenhum dos outros componentes externos são necessários nas aplicações, uma vez que todos os periféricos necessários já estão contidos nele.
- Assim, nós poupamos tempo e espaço na construção dos dispositivos.



http://www.microchip.com/stellent/images/mchpsiteimages/en537986.jpg

8-bit Core Architecture

- Harvard (memória de dados separada da memória de programa)
- RISC
- Único acumulador
  - W Register
- Small addressable data space (256 bytes)
  - Banking
- RAM
  - PC, special purpose registers



http://www.microchip.com/_images/BaselineArch_large.jpg

Vantagens & Limitaçoes da Arquitetura 8-bit

Vantagens

- Pequeno conjunto de Instruções (ISA)
- Oscilador interno, com velocidades selecionaveis
- Facilidade de desenvolvimento
- Baixo-custo
- Device variants
- Wide range of interfaces (I2C, SPI, etc.)

Limitações

- Único registrador acumulador (W)
- Bank switching
- Não é possivel acessar os endereços da pilha (sem multi-tasking)

16-bit and 32-bit Architecture

- Maior quantidade de working registers
- No bank switching
- Assignable interrupt vector table
- Maior quantidade de memória flash
- Cache (32-bit architecture)

	Baseline Architecture	Mid-Range Architecture	Enhanced Mid-Range Architecture	PIC18 Architecture
Pin Count	6-40	8-64	8-64	18-100
Interrupts	No	Single interrupt capability	Single interrupt capability with hardware context save	Multiple interrupt capability with hardware context save
Performance	5 MIPS	5 MIPS	8 MIPS	Up to 16 MIPS
Instructions	33, 12-bit	35, 14-bit	49, 14-bit	83, 16-bit
Program Memory	Up to 3 KB	Up to 14 KB	Up to 28 KB	Up to 128 KB
Data Memory	Up to 138 Bytes	Up to 368 Bytes	Up to 1.5 KB	Up to 4 KB
Features	•Comparator •8-bit ADC •Data Memory •Internal Oscillator	In addition to Baseline: •SPI/I²C™ •UART •PWMs •LCD •10-bit ADC •Op Amp	In addition to Mid-Range: •Multiple Communication Peripherals •Linear Programming Space •PWMs with Independent Time Base	In addition to Enhanced Mid-Range: •8x8 Hardware Multiplier •CAN •CTMU •USB •Ethernet •12-bit ADC
Families	PIC10, PIC12, PIC16	PIC12, PIC16	PIC12FXXX, PIC16F1XX	PIC18

- 8-bit architecture

PIC16 ISA:
35 Instructions, 14-bit

Data Transfer Instructions

INSTRUCTION	DESCRIPTION	OPERATION
MOVLW k	Move constant to W	k -> w
MOVWF f	Move W to f	W -> f
MOVF f,d	Move f to d	f -> d
CLRW	Clear W	0 -> W
CLRF f	Clear f	0 -> f
SWAPF f,d	Swap nibbles in f	f(7:4),(3:0) -> f(3:0),(7:4)

W: Working register(Accumulator)

Registers: Memory locations

INSTRUCTION	DESCRIPTION	OPERATION
Data Transfer Instructions		
MOVLW k	Move constant to W	k -> w
MOVWF f	Move W to f	W -> f
MOVF f,d	Move f to d	f -> d
CLRW	Clear W	0 -> W
CLRF f	Clear f	0 -> f
SWAPF f,d	Swap nibbles in f	f(7:4),(3:0) -> f(3:0),(7:4)
Arithmetic-logic Instructions		
ADDLW k	Add W and constant	W+k -> W
ADDWF f,d	Add W and f	W+f -> d
SUBLW k	Subtract W from constant	k-W -> W
SUBWF f,d	Subtract W from f	f-W -> d
ANDLW k	Logical AND with W with constant	W AND k -> W
ANDWF f,d	Logical AND with W with f	W AND f -> d
ANDWF f,d	Logical AND with W with f	W AND f -> d
IORLW k	Logical OR with W with constant	W OR k -> W
IORWF f,d	Logical OR with W with f	W OR f -> d
XORWF f,d	Logical exclusive OR with W with constant	W XOR k -> W
XORLW k	Logical exclusive OR with W with f	W XOR f -> d
INCF f,d	Increment f by 1	f+1 -> f
DECF f,d	Decrement f by 1	f-1 -> f
RLF f,d	Rotate left f through CARRY bit	
RRF f,d	Rotate right f through CARRY bit	
COMF f,d	Complement f	f -> d
Bit-oriented Instructions		
BCF f,b	Clear bit b in f	0 -> f(b)
BSF f,b	Clear bit b in f	1 -> f(b)
Program Control Instructions		
BTFSC f,b	Test bit b of f. Skip the following instruction if clear.	Skip if f(b) = 0
BTFSS f,b	Test bit b of f. Skip the following instruction if set.	Skip if f(b) = 1
DECFSZ f,d	Decrement f. Skip the following instruction if clear.	f-1 -> d skip if Z = 1
INCFSZ f,d	Increment f. Skip the following instruction if set.	f+1 -> d skip if Z = 0
GOTO k	Go to address	k -> PC
CALL k	Call subroutine	PC -> TOS, k -> PC
RETURN	Return from subroutine	TOS -> PC
RETLW k	Return with constant in W	k -> W, TOS -> PC
RETFIE	Return from interrupt	TOS -> PC, 1 -> GIE
Other instructions		
NOP	No operation	TOS -> PC, 1 -> GIE
CLRWDT	Clear watchdog timer	0 -> WDT, 1 -> TO, 1 -> PD
SLEEP	Go into sleep mode	0 -> WDT, 1 -> TO, 0 -> PD

Arithmetic-logic Instructions (partial)

ADDLW k	Add W and constant	W+k -> W
ADDWF f,d	Add W and f	W+f -> d
INCF f,d	Increment f by 1	f+1 -> f
DECF f,d	Decrement f by 1	f-1 -> f
RLF f,d	Rotate left f through CARRY bit	
RRF f,d	Rotate right f through CARRY bit	
COMF f,d	Complement f	f -> d

SUBLW, SUBWF
ANDLW, ANDWF
IORLW, IORWF
XORLW, XORWF

INSTRUCTION	DESCRIPTION	OPERATION
Data Transfer Instructions		
MOVLW k	Move constant to W	k -> w
MOVWF f	Move W to f	W -> f
MOVF f,d	Move f to d	f -> d
CLRW	Clear W	0 -> W
CLRF f	Clear f	0 -> f
SWAPF f,d	Swap nibbles in f	f(7:4),(3:0) -> f(3:0),(7:4)
Arithmetic-logic Instructions		
ADDLW k	Add W and constant	W+k -> W
ADDWF f,d	Add W and f	W+f -> d
SUBLW k	Subtract W from constant	k-W -> W
SUBWF f,d	Subtract W from f	f-W -> d
ANDLW k	Logical AND with W with constant	W AND k -> W
ANDWF f,d	Logical AND with W with f	W AND f -> d
ANDWF f,d	Logical AND with W with f	W AND f -> d
IORLW k	Logical OR with W with constant	W OR k -> W
IORWF f,d	Logical OR with W with f	W OR f -> d
XORWF f,d	Logical exclusive OR with W with constant	W XOR k -> W
XORLW k	Logical exclusive OR with W with f	W XOR f -> d
INCF f,d	Increment f by 1	f+1 -> f
DECF f,d	Decrement f by 1	f-1 -> f
RLF f,d	Rotate left f through CARRY bit	
RRF f,d	Rotate right f through CARRY bit	
COMF f,d	Complement f	f -> d
Bit-oriented Instructions		
BCF f,b	Clear bit b in f	0 -> f(b)
BSF f,b	Clear bit b in f	1 -> f(b)
Program Control Instructions		
BTFSC f,b	Test bit b of f. Skip the following instruction if clear.	Skip if f(b) = 0
BTFSS f,b	Test bit b of f. Skip the following instruction if set.	Skip if f(b) = 1
DECFSZ f,d	Decrement f. Skip the following instruction if clear.	f-1 -> d skip if Z = 1
INCFSZ f,d	Increment f. Skip the following instruction if set.	f+1 -> d skip if Z = 0
GOTO k	Go to address	k -> PC
CALL k	Call subroutine	PC -> TOS, k -> PC
RETURN	Return from subroutine	TOS -> PC
RETLW k	Return with constant in W	k -> W, TOS -> PC
RETFIE	Return from interrupt	TOS -> PC, 1 -> GIE
Other instructions		
NOP	No operation	TOS -> PC, 1 -> GIE
CLRWDT	Clear watchdog timer	0 -> WDT, 1 -> TO, 1 -> PD
SLEEP	Go into sleep mode	0 -> WDT, 1 -> TO, 0 -> PD

Program Control Instructions

BTFSC f,b	Test bit b of f. Skip the following instruction if clear.	Skip if f(b) = 0
BTFSS f,b	Test bit b of f. Skip the following instruction if set.	Skip if f(b) = 1
DECFSZ f,d	Decrement f. Skip the following instruction if clear.	f-1 -> d skip if Z = 1
INCFSZ f,d	Increment f. Skip the following instruction if set.	f+1 -> d skip if Z = 0
GOTO k	Go to address	k -> PC
CALL k	Call subroutine	PC -> TOS, k -> PC
RETURN	Return from subroutine	TOS -> PC
RETLW k	Return with constant in W	k -> W, TOS -> PC
RETFIE	Return from interrupt	TOS -> PC, 1 -> GIE

INSTRUCTION	DESCRIPTION	OPERATION
Data Transfer Instructions		
MOVLW k	Move constant to W	k -> w
MOVWF f	Move W to f	W -> f
MOVF f,d	Move f to d	f -> d
CLRW	Clear W	0 -> W
CLRF f	Clear f	0 -> f
SWAPF f,d	Swap nibbles in f	f(7:4),(3:0) -> f(3:0),(7:4)
Arithmetic-logic Instructions		
ADDLW k	Add W and constant	W+k -> W
ADDWF f,d	Add W and f	W+f -> d
SUBLW k	Subtract W from constant	k-W -> W
SUBWF f,d	Subtract W from f	f-W -> d
ANDLW k	Logical AND with W with constant	W AND k -> W
ANDWF f,d	Logical AND with W with f	W AND f -> d
ANDWF f,d	Logical AND with W with f	W AND f -> d
IORLW k	Logical OR with W with constant	W OR k -> W
IORWF f,d	Logical OR with W with f	W OR f -> d
XORWF f,d	Logical exclusive OR with W with constant	W XOR k -> W
XORLW k	Logical exclusive OR with W with f	W XOR f -> d
INCF f,d	Increment f by 1	f+1 -> f
DECF f,d	Decrement f by 1	f-1 -> f
RLF f,d	Rotate left f through CARRY bit	
RRF f,d	Rotate right f through CARRY bit	
COMF f,d	Complement f	f -> d
Bit-oriented Instructions		
BCF f,b	Clear bit b in f	0 -> f(b)
BSF f,b	Clear bit b in f	1 -> f(b)
Program Control Instructions		
BTFSC f,b	Test bit b of f. Skip the following instruction if clear.	Skip if f(b) = 0
BTFSS f,b	Test bit b of f. Skip the following instruction if set.	Skip if f(b) = 1
DECFSZ f,d	Decrement f. Skip the following instruction if clear.	f-1 -> d skip if Z = 1
INCFSZ f,d	Increment f. Skip the following instruction if set.	f+1 -> d skip if Z = 0
GOTO k	Go to address	k -> PC
CALL k	Call subroutine	PC -> TOS, k -> PC
RETURN	Return from subroutine	TOS -> PC
RETLW k	Return with constant in W	k -> W, TOS -> PC
RETFIE	Return from interrupt	TOS -> PC, 1 -> GIE
Other instructions		
NOP	No operation	TOS -> PC, 1 -> GIE
CLRWDT	Clear watchdog timer	0 -> WDT, 1 -> TO, 1 -> PD
SLEEP	Go into sleep mode	0 -> WDT, 1 -> TO, 0 -> PD

Bit-oriented Instructions

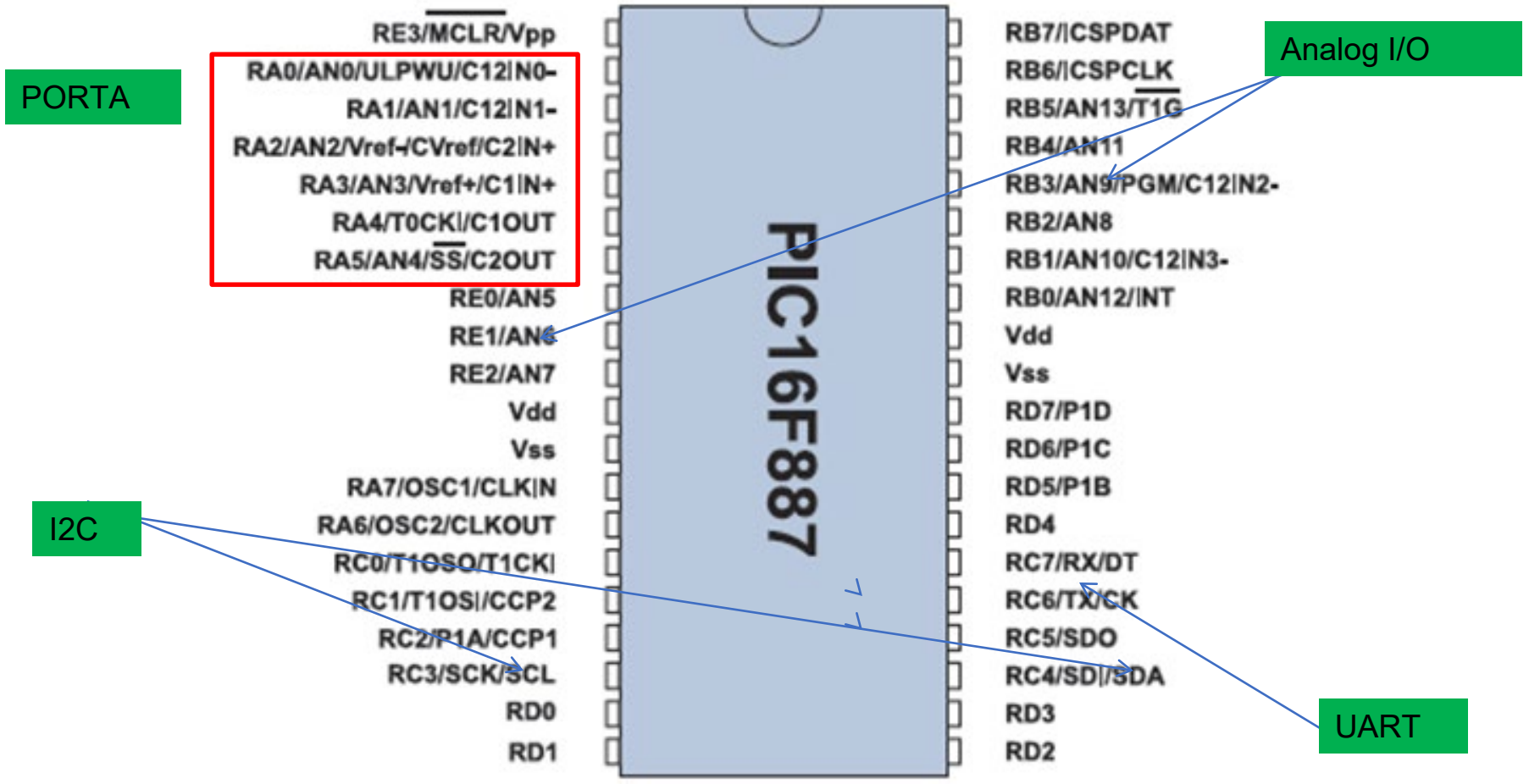
BCF f,b	Clear bit b in f	0 -> f(b)
BSF f,b	Set bit b in f	1 -> f(b)

Other instructions

NOP	No operation	TOS -> PC, 1 -> GIE
CLRWDT	Clear watchdog timer	0 -> WDT, 1 -> TO, 1 -> PD
SLEEP	Go into sleep mode	0 -> WDT, 1 -> TO, 0 -> PD

INSTRUCTION	DESCRIPTION	OPERATION
Data Transfer Instructions		
MOVLW k	Move constant to W	k -> w
MOVWF f	Move W to f	W -> f
MOVF f,d	Move f to d	f -> d
CLRW	Clear W	0 -> W
CLRF f	Clear f	0 -> f
SWAPF f,d	Swap nibbles in f	f(7:4),(3:0) -> f(3:0),(7:4)
Arithmetic-logic Instructions		
ADDLW k	Add W and constant	W+k -> W
ADDWF f,d	Add W and f	W+f -> d
SUBLW k	Subtract W from constant	k-W -> W
SUBWF f,d	Subtract W from f	f-W -> d
ANDLW k	Logical AND with W with constant	W AND k -> W
ANDWF f,d	Logical AND with W with f	W AND f -> d
ANDWF f,d	Logical AND with W with f	W AND f -> d
IORLW k	Logical OR with W with constant	W OR k -> W
IORWF f,d	Logical OR with W with f	W OR f -> d
XORWF f,d	Logical exclusive OR with W with constant	W XOR k -> W
XORLW k	Logical exclusive OR with W with f	W XOR f -> d
INCF f,d	Increment f by 1	f+1 -> f
DECF f,d	Decrement f by 1	f-1 -> f
RLF f,d	Rotate left f through CARRY bit	
RRF f,d	Rotate right f through CARRY bit	
COMF f,d	Complement f	f -> d
Bit-oriented Instructions		
BCF f,b	Clear bit b in f	0 -> f(b)
BSF f,b	Clear bit b in f	1 -> f(b)
Program Control Instructions		
BTFSC f,b	Test bit b of f. Skip the following instruction if clear.	Skip if f(b) = 0
BTFSS f,b	Test bit b of f. Skip the following instruction if set.	Skip if f(b) = 1
DECFSZ f,d	Decrement f. Skip the following instruction if clear.	f-1 -> d skip if Z = 1
INCFSZ f,d	Increment f. Skip the following instruction if set.	f+1 -> d skip if Z = 0
GOTO k	Go to address	k -> PC
CALL k	Call subroutine	PC -> TOS, k -> PC
RETURN	Return from subroutine	TOS -> PC
RETLW k	Return with constant in W	k -> W, TOS -> PC
RETFIE	Return from interrupt	TOS -> PC, 1 -> GIE
Other instructions		
NOP	No operation	TOS -> PC, 1 -> GIE
CLRWDT	Clear watchdog timer	0 -> WDT, 1 -> TO, 1 -> PD
SLEEP	Go into sleep mode	0 -> WDT, 1 -> TO, 0 -> PD

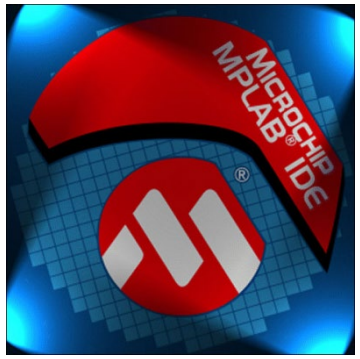
PIC 16F887 layout



http://www.mikroe.com/eng/chapters/view/74/pic-basic-book-chapter-1-world-of-microcontrollers/

Programming a PIC

- Microchip provides the free MPLAB:
  - Assembler and linker
  - Application development
  - Hardware emulation
  - Debugging
  - C or assembly compatible

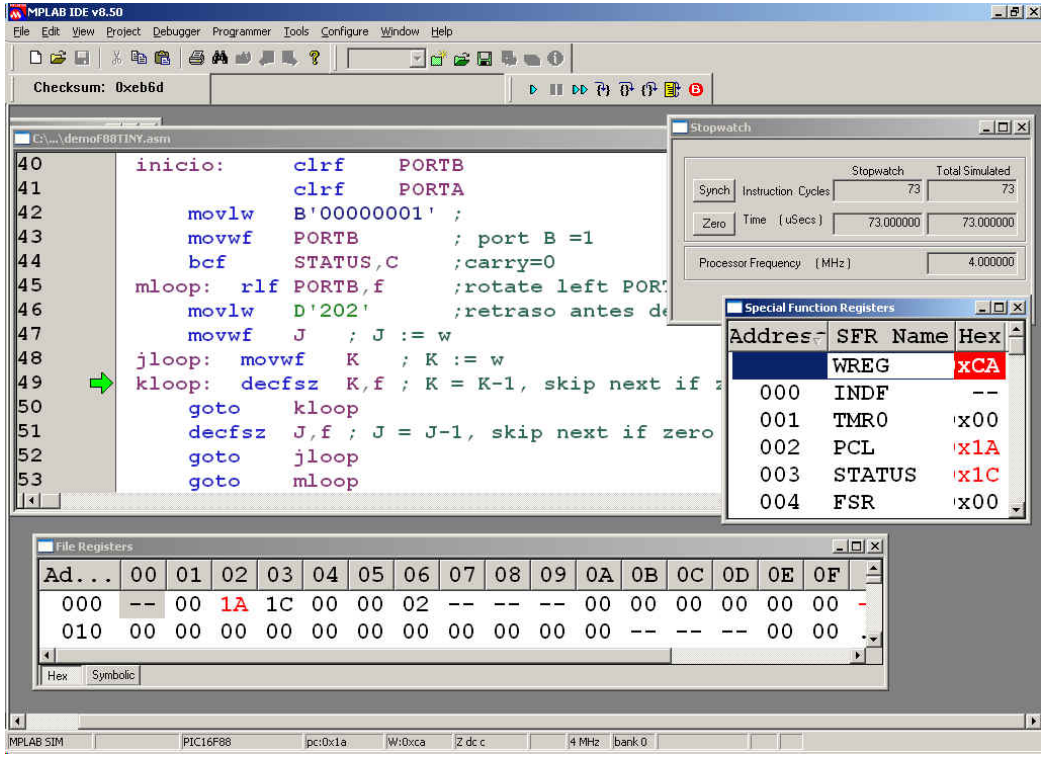


http://www.microchip.com/pagehandler/en-us/family/mplabx/

- Compiler
  - Can be C-based or Basic
  - A free one is the CCS C Compiler for PIC12/24/26/18 (not compatible with all PICS) or the HI-TECH PICC-Lit

Ferramentas de Programação

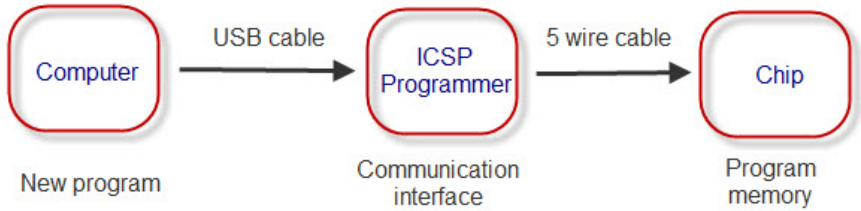
- MPLAB IDE
- Ferramenta básica para programação assembly ou C
- Versão atual MPLAB X (Windows, Linux, MacOS)



http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

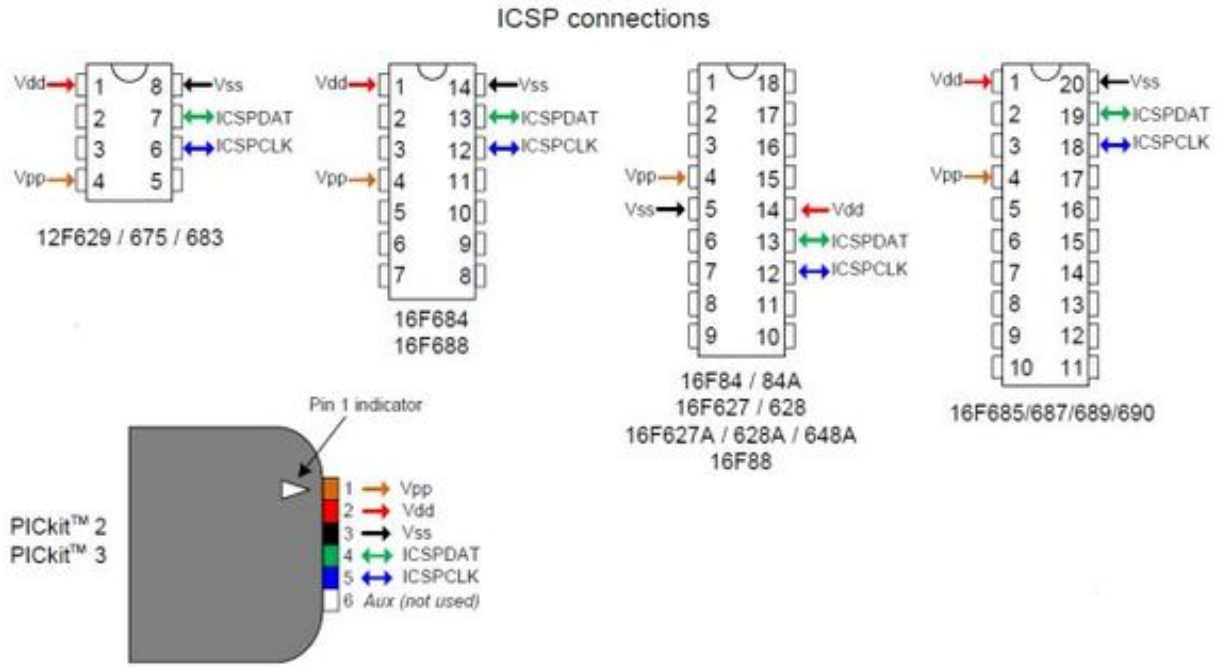
Dispositivo Programador

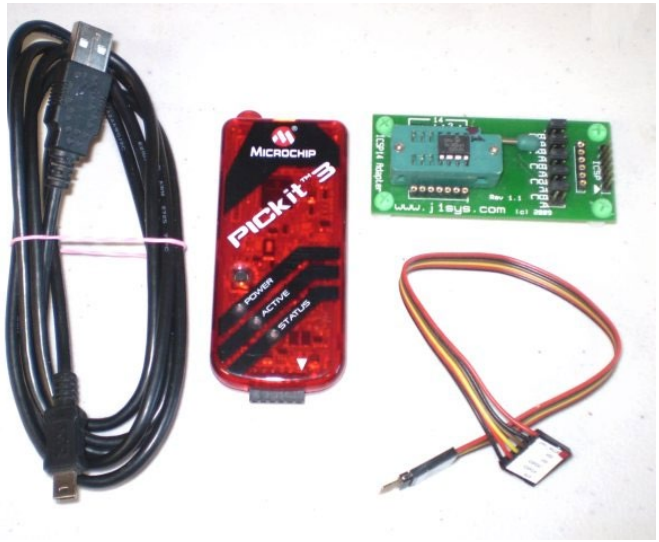
- Need device to store machine code into PIC's memory (EEPROM or Flash)
- Can be external device, but ICSP is easier:
  - Don't have to remove chip from its circuit
  - Provides interface between computer (USB) and PIC
  - Specific to circuit (due to interconnect scheme and surrounding circuit)
  - Communication protocol requ signals



http://en.wikipedia.org/wiki/PIC_microcontroller

Device Programmer





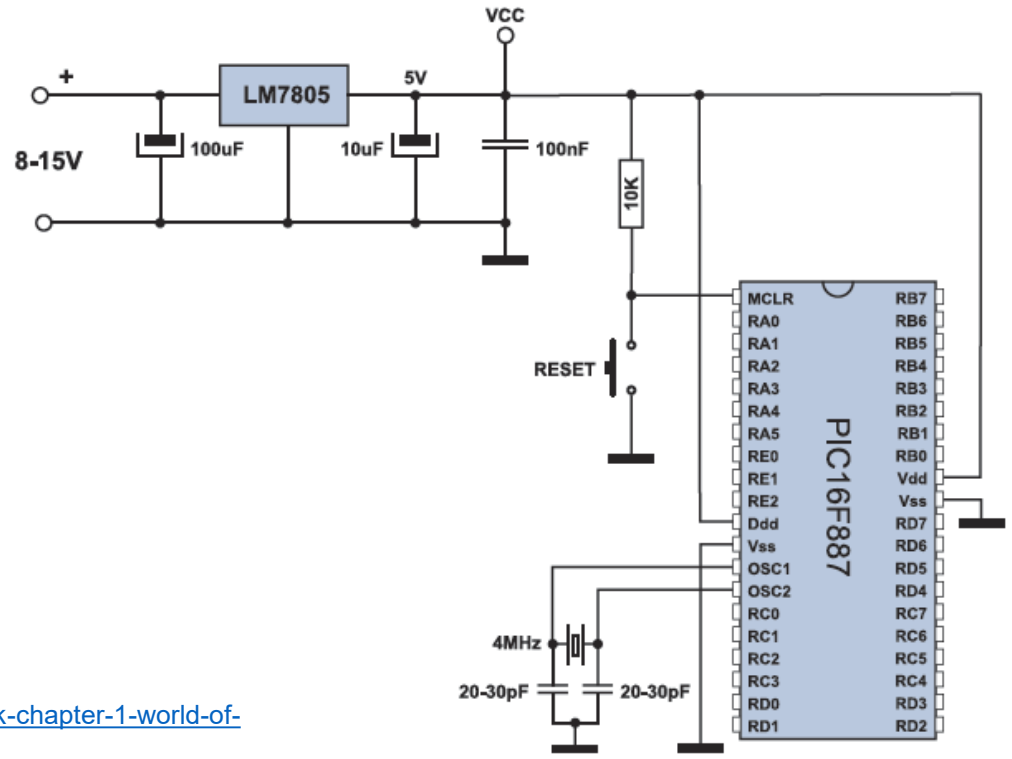
5 Sinais:

- Vpp (programming voltage)
- Vdd (power)
- Vss (ground)
- IC SPCLK (clock)
- IC SPDAT (data)

http://en.wikipedia.org/wiki/PIC_microcontroller

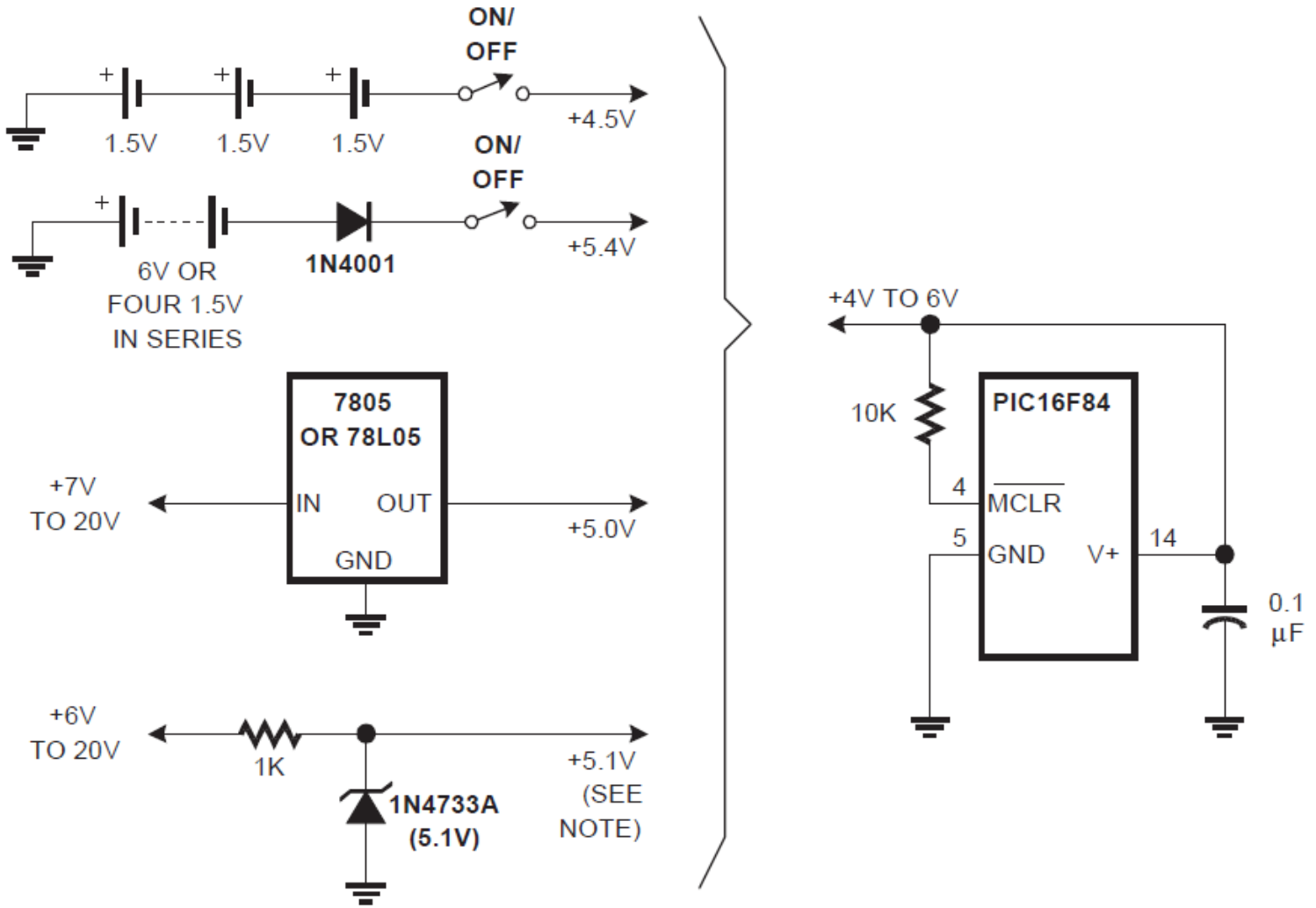
Conexõe para uso (PIC16F877A)

- O PIC pode ser montado na proto-board, com as seguintes conexões:
  - Power
  - Ground
  - Reset signal
  - Crystal (oscillator)

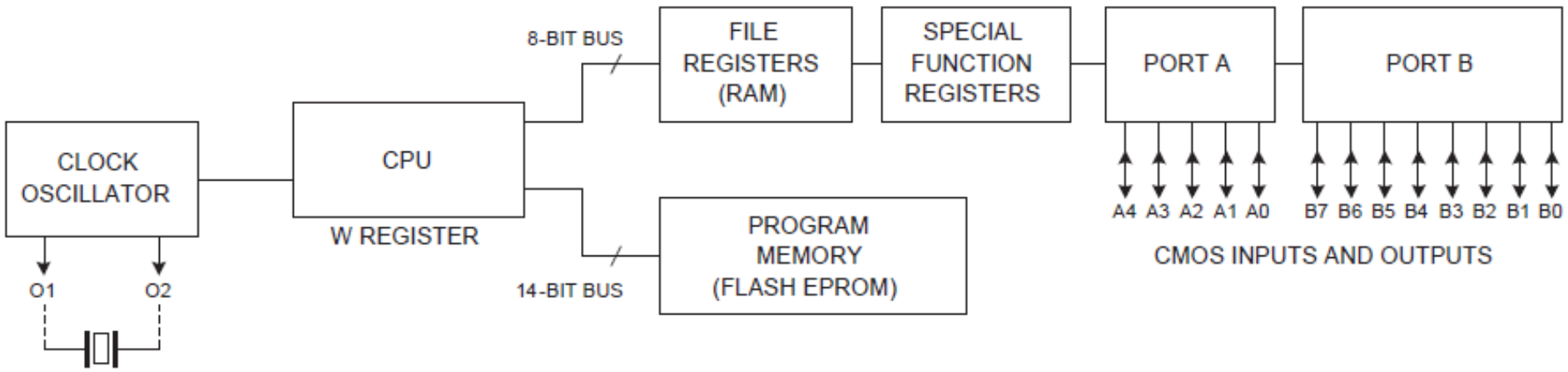


http://www.mikroe.com/eng/chapters/view/74/pic-basic-book-chapter-1-world-of-microcontrollers/

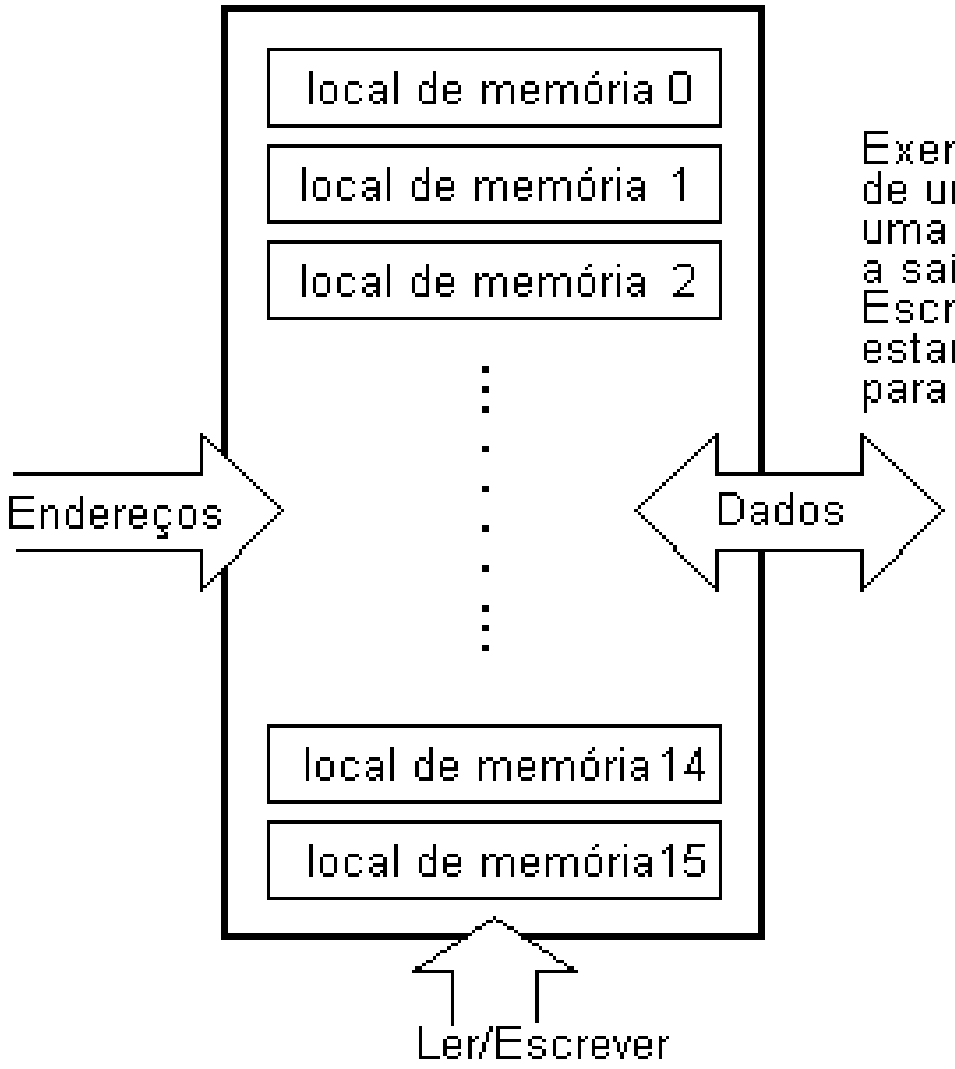
Como ligar


ON/
OFF
1.5V
1.5V
1.5V
+4.5V
ON/
OFF
6V OR
FOUR 1.5V
IN SERIES
1N4001
+5.4V
7805
OR 78L05
+7V
TO 20V
IN
OUT
GND
+5.0V
+6V
TO 20V
1K
1N4733A
(5.1V)
+5.1V
(SEE
NOTE)
+4V TO 6V
10K
PIC16F84
4
MCLR
5
GND
V+
14
0.1
µF

16F84A – 16F628

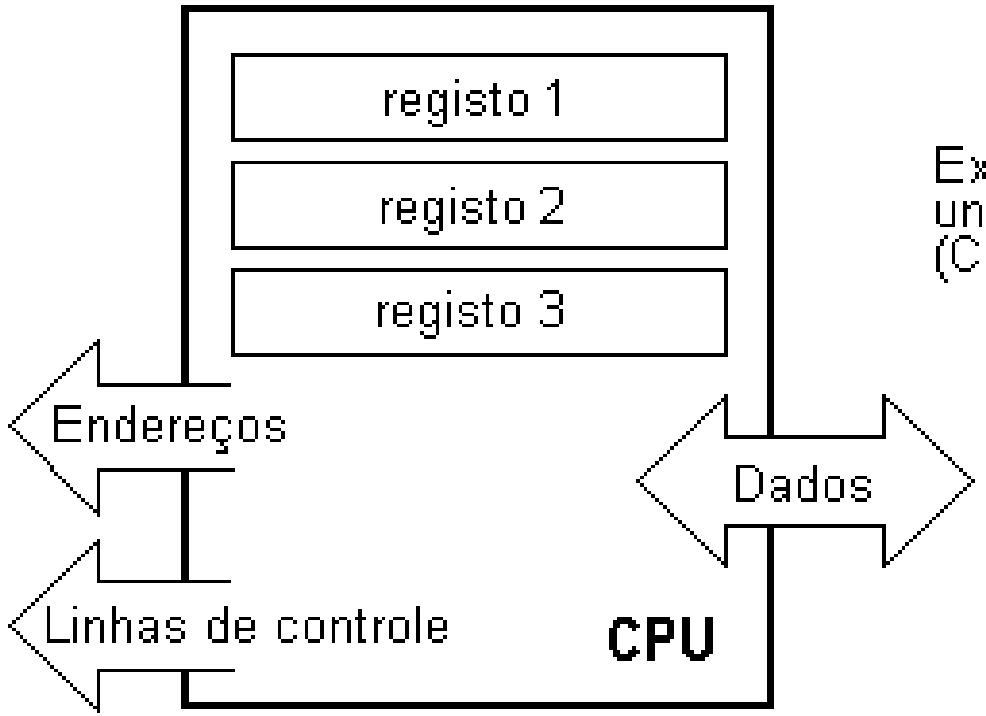


Unidade de Memória



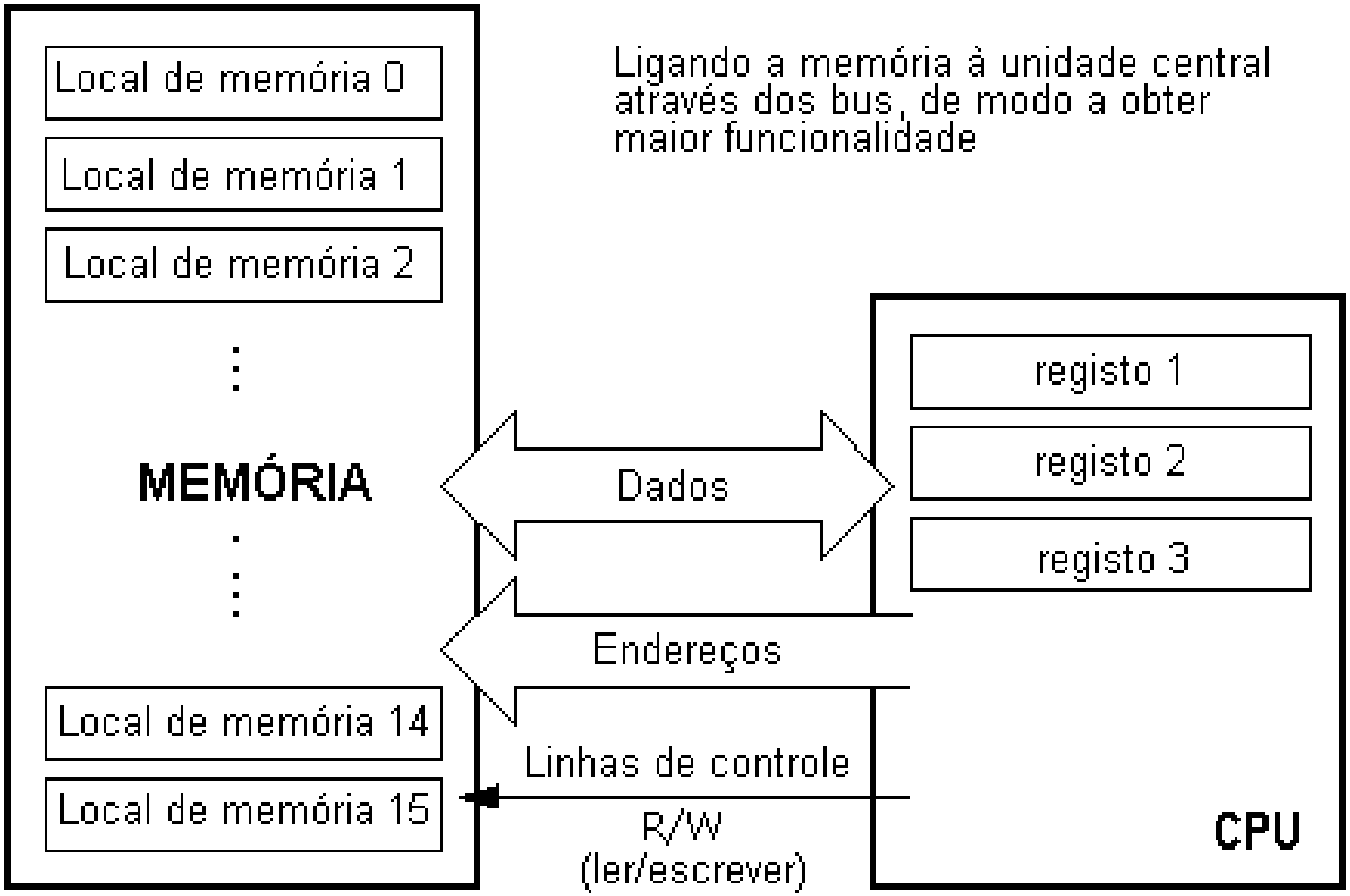
Exemplo de um modelo simplificado de uma unidade de memória. Para uma entrada específica nós obtemos a saída correspondente. A linha Ler/Escrever (R/W) determina quando estamos a ler ou escrever da ou para a memória.

Unidade Central de Processamento

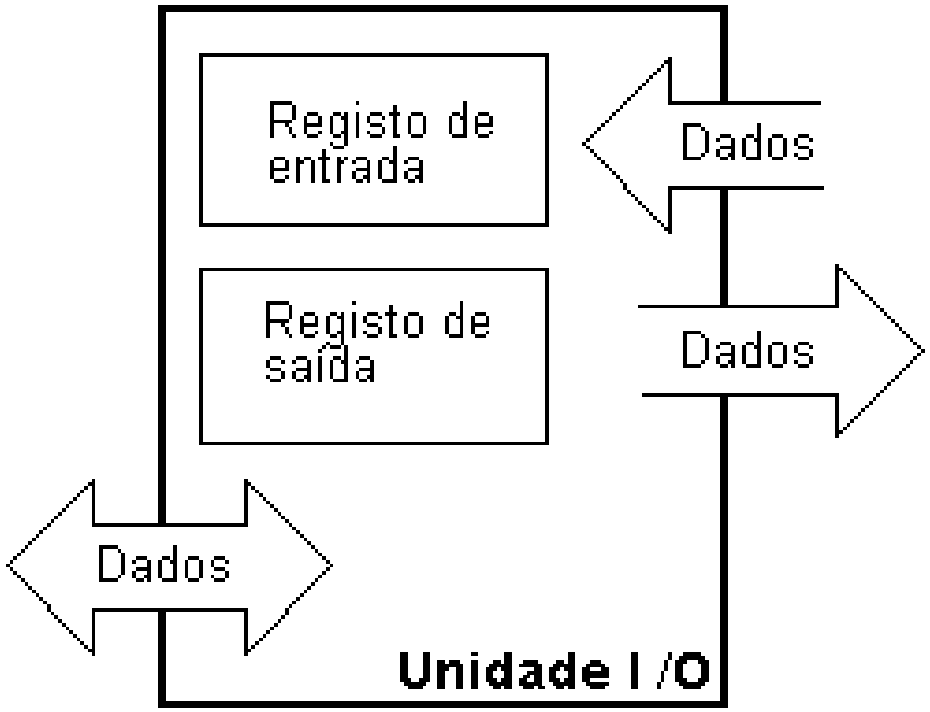


Exemplo simplificado de uma unidade central de processamento (CPU) com três registos.

Barramentos

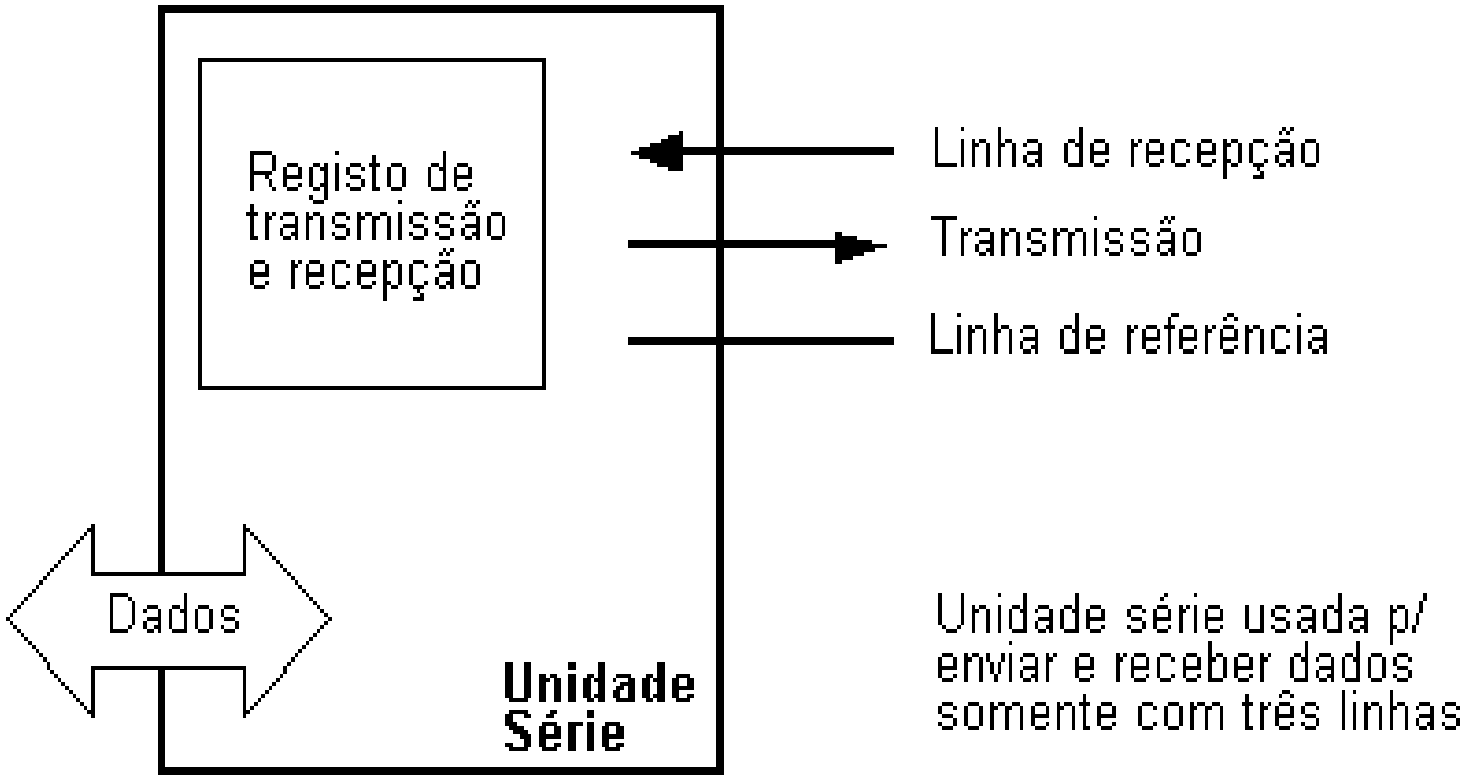


Unidade de Entrada e Saídas (I/O)

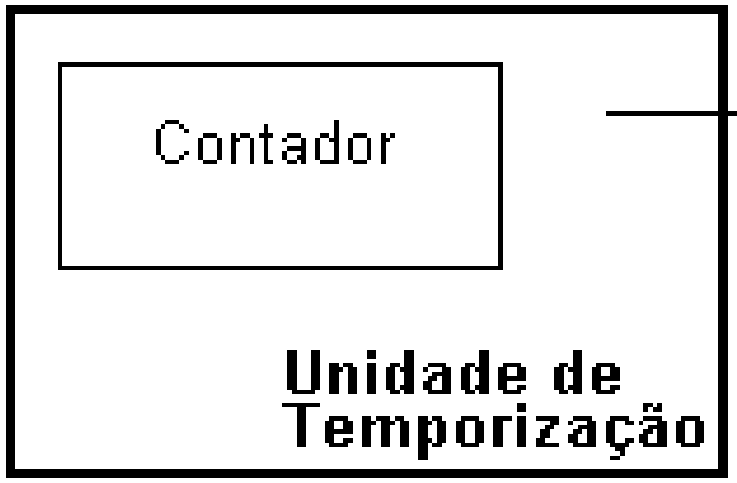


Exemplo simplificado de uma unidade de entrada/ saída (I/O) que fornece comunicação com o mundo exterior

Comunicação Serial



Timer

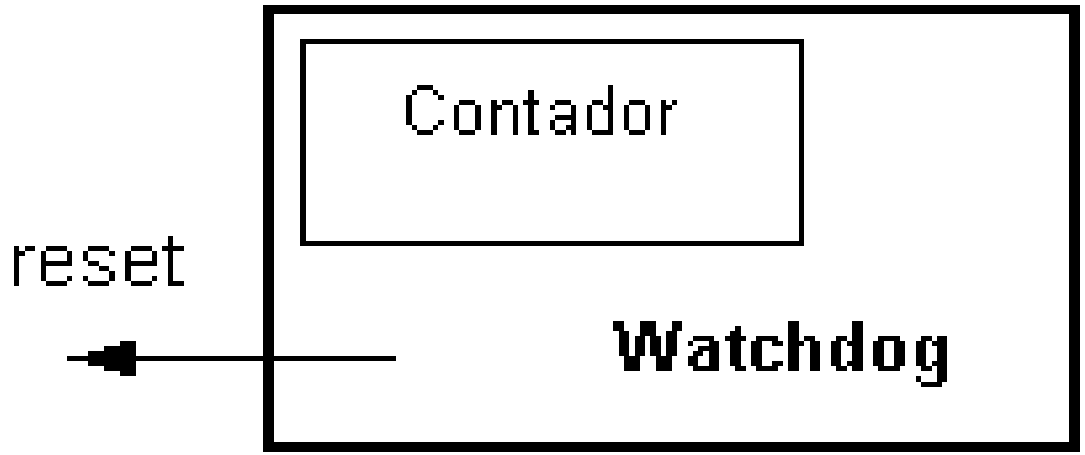


Sinal

A unidade de temporização gera sinais a intervalos de tempo regulares

Watchdog

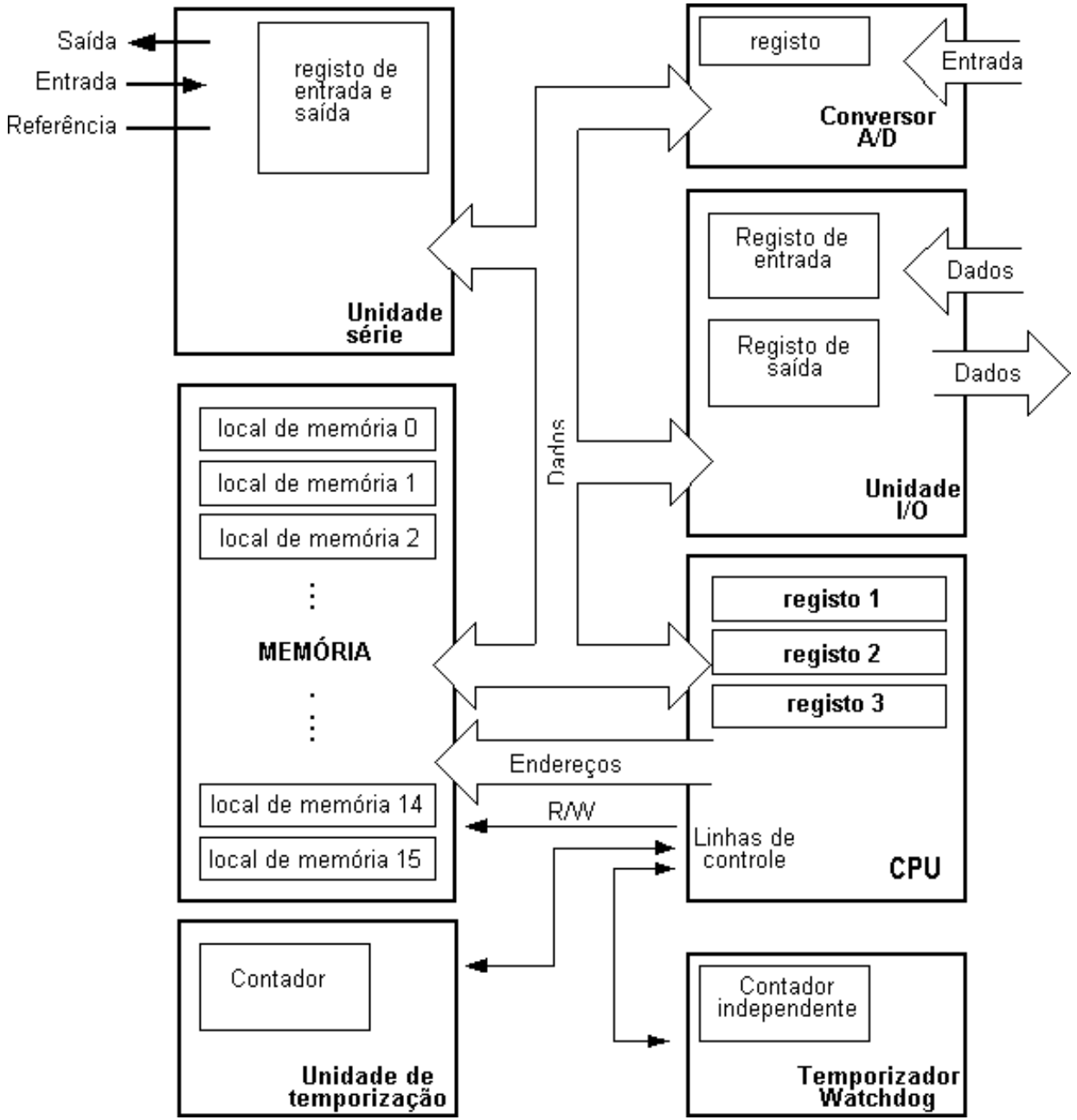
- Responsavel por “destravar/reset” nosso microcontrolador em caso de erro.



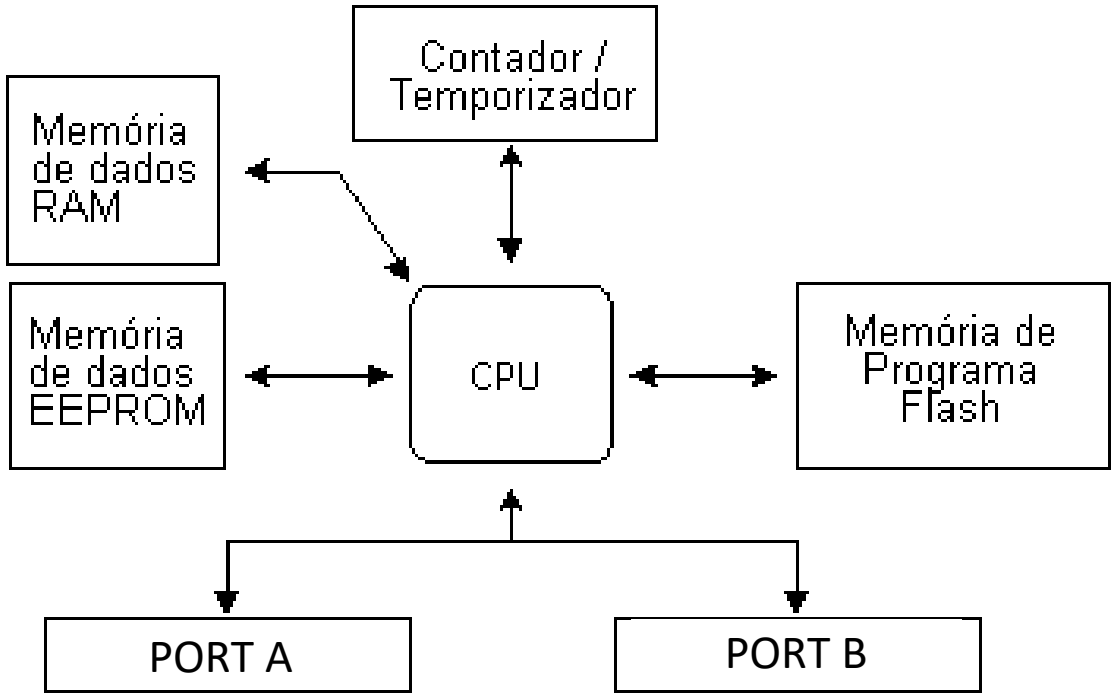
Conversor Analógico – Digital

- Responsável por transformar um nível de tensão em um número para que o microcontrolador possa processar a informação.

Configuração física do interior de um microcontrolador



Microcontrolador PIC16F84



Esquema do microcontrolador PIC16F84

- O **PIC 16F84** pertence a uma classe de microcontroladores de 8 bits, com uma arquitetura RISC.
- A estrutura genérica é a do mapa ao lado, que nos mostra os seus blocos básicos.

Microcontrolador PIC16F84

- **Memória de programa** (FLASH) - para armazenar o programa que se escreveu. Como a memória fabricada com tecnologia FLASH pode ser programa e limpa mais que uma vez. ela torna-se adequada para o desenvolvimento de dispositivos.

Microcontrolador PIC16F84

- **EEPROM** - memória dos dados que necessitam de ser salvaguardados quando a alimentação é desligada.
  - Normalmente é usada para guardar dados importantes que não se podem perder quando a alimentação, de repente, "vai abaixo".
  - Um exemplo deste tipo de dados é a temperatura fixada para os reguladores de temperatura. Se, durante uma queda de luz se perdessem dados, nós precisaríamos ajustar de novo quando a alimentação fosse restabelecida. Assim, o nosso dispositivo, perderia eficácia.

Microcontrolador PIC16F84

- **RAM** - memória de dados usada por um programa, durante a sua execução.
- Na RAM, são guardados todos os resultados intermédios ou dados temporários durante a execução do programa e que não são cruciais para o dispositivo, depois de ocorrer uma falha na alimentação.

- **PORT A e PORT B** são ligações físicas entre o microcontrolador e o mundo exterior. O port A tem cinco pinos e o port B oito pinos.

Microcontrolador PIC16F84

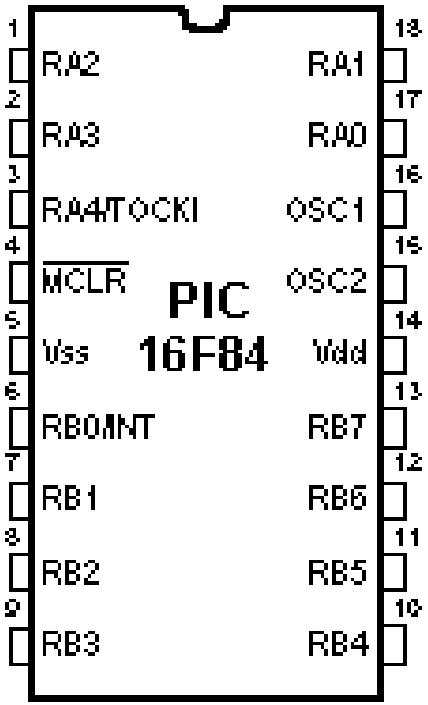
- **CONTADOR/TEMPORIZADOR** é um registo de 8 bits no interior do microcontrolador que trabalha independentemente do programa. No fim de cada conjunto de quatro ciclos de relógio do oscilador, ele incrementa o valor armazenado, até atingir o valor máximo (255), nesta altura recomeça a contagem a partir de zero.
- Como nós sabemos o tempo exato entre dois incrementos sucessivos do conteúdo do temporizador, podemos utilizar esta informação para medir intervalos de tempo, o que o torna muito útil em vários dispositivos.

Microcontrolador PIC16F84

- **UNIDADE DE PROCESSAMENTO CENTRAL** faz a conexão com todos os outros blocos do microcontrolador. Ele coordena o trabalho dos outros blocos e executa o programa do utilizador (firmware).

Microcontrolador PIC16F84

- **Descrição dos Terminais ou "Pinos"**
  - **18 pinos (encapsumaneto DIP)**



RA0 - RA3 : Pins on port A. No additional function

RA4 : TOCK1 which functions as a timer

RB0 : Interrupt input is an additional function.

RB1 - RB5 : Pins on port B. No additional function.

RB6 : 'Clock' line in program mode.

RB7 : 'Data' line in program mode

MCLR : Reset input and Vpp programming voltage

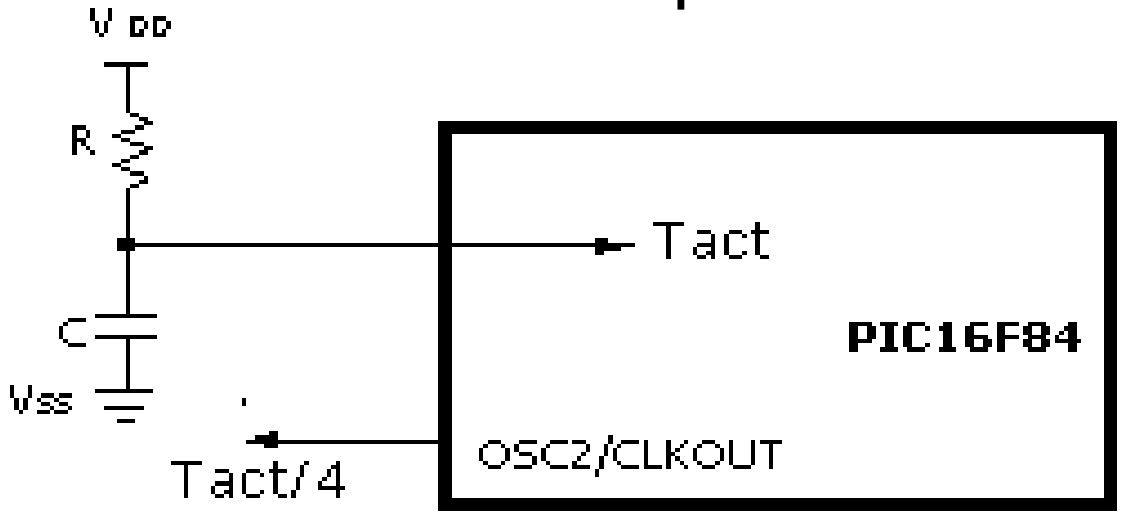
Vss : Ground of power supply.

Vdd : Positive power supply pole.

OSC1 - OSC2 : Pins for connecting with oscillator.

Gerador de Clock – Oscilador

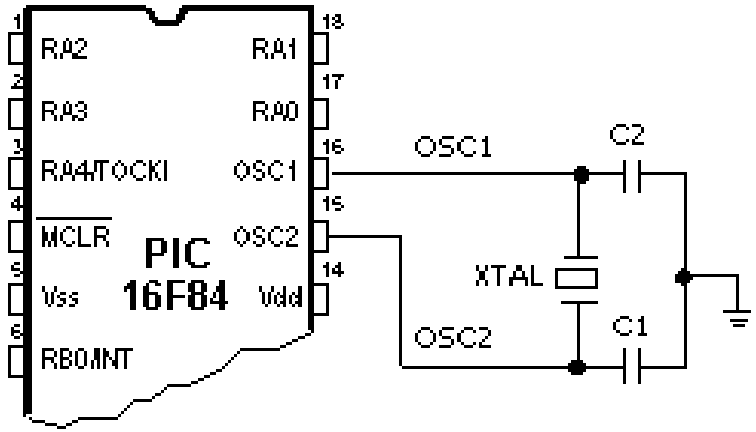
- **OSCILADOR RC**
- O diagrama mostra como um oscilador RC deve ser ligado a um PIC16F84. Com um valor para a resistência R abaixo de 2,2 K, o oscilador pode tornar-se instável ou pode mesmo parar de oscilar.



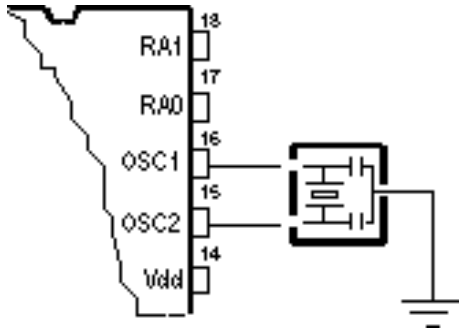
Para um valor muito grande R (1M por exemplo), o oscilador torna-se muito sensível à umidade e ao ruído. É recomendado que o valor da resistência R esteja compreendido entre 3K e 100K. Apesar de o oscilador poder trabalhar sem capacitor externo (C = 0 pF), é conveniente, ainda assim, usar um capacitor acima de 20 pF para evitar o ruído e aumentar a estabilidade.

Gerador de Clock – Oscilador

- **OSCILADOR XT**
- O oscilador de cristal está contido num envolucro de metal com dois pinos onde foi escrita a frequência a que o cristal oscila. Dois condensadores cerâmicos devem ligar cada um dos pinos do cristal à massa. Casos há em que cristal e condensadores estão contidos no mesmo encapsulamento, é também o caso do ressonador cerâmico ao lado representado.

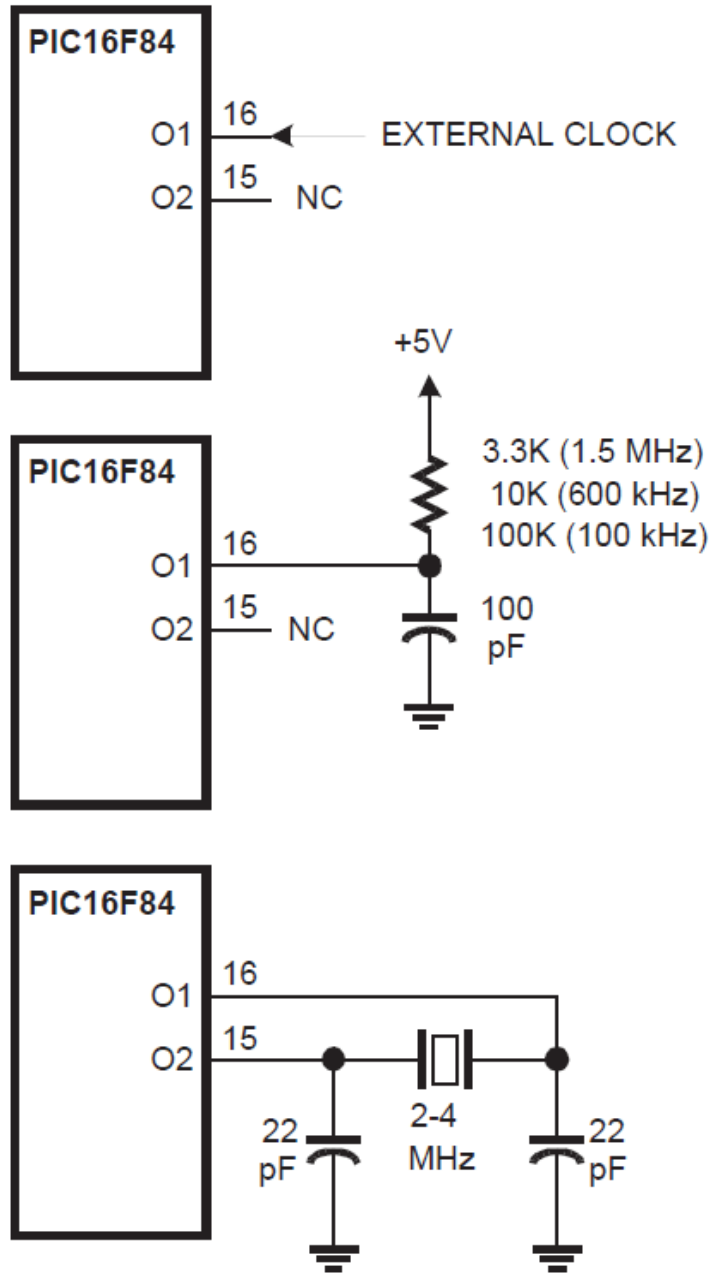


Clock de um microcontrolador a partir de um cristal de quartzo



Clock de um microcontrolador com um ressonador

Este elemento tem três pinos com o pino central ligado à massa e os outros dois pinos ligados aos pinos OSC1 e OSC2 do microcontrolador. Quando projectamos um dispositivo, a regra é colocar o oscilador tão perto quanto possível do microcontrolador, de modo a evitar qualquer interferência nas linhas que ligam o oscilador ao microcontrolador.


PIC16F84
O1
16
EXTERNAL CLOCK
O2
15
NC
PIC16F84
+5V
3.3K (1.5 MHz)
10K (600 kHz)
100K (100 kHz)
O1
16
O2
15
NC
100
pF
PIC16F84
O1
16
O2
15
22
pF
2-4
MHz
22
pF

Reset

- O reset é usado para pôr o microcontrolador num estado conhecido.
- De modo a prevenir a ocorrência de um zero lógico acidental no pino MCLR (a linha por cima de MCLR significa o sinal de reset é ativado por nível lógico baixo), o pino MCLR tem que ser ligado através de uma resistência ao lado positivo da alimentação. Esta resistência deve ter um valor entre 5 e 10K. Uma resistência como esta, cuja função é conservar uma determinada linha a nível lógico alto, é chamada “resistência de pull up”.

Reset

- O microcontrolador PIC16F84, admite várias formas de reset:

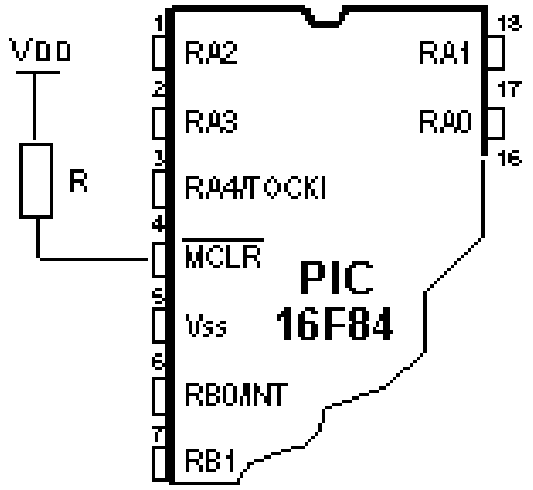
a)Reset quando se liga a alimentação, POR (Power-On Reset)

b) Reset durante o funcionamento normal, quando se põe a nível lógico baixo o pino MCLR do microcontrolador.

c)Reset durante o regime de SLEEP (dormir).

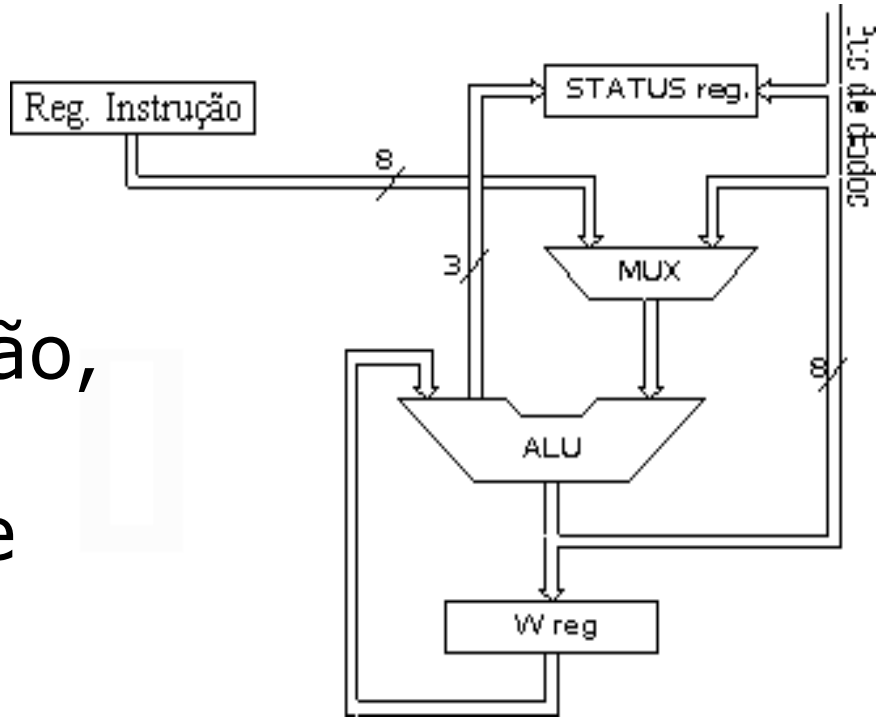
d) Reset quando o temporizador do watchdog (WDT) transborda (passa para 0 depois de atingir o valor máximo).

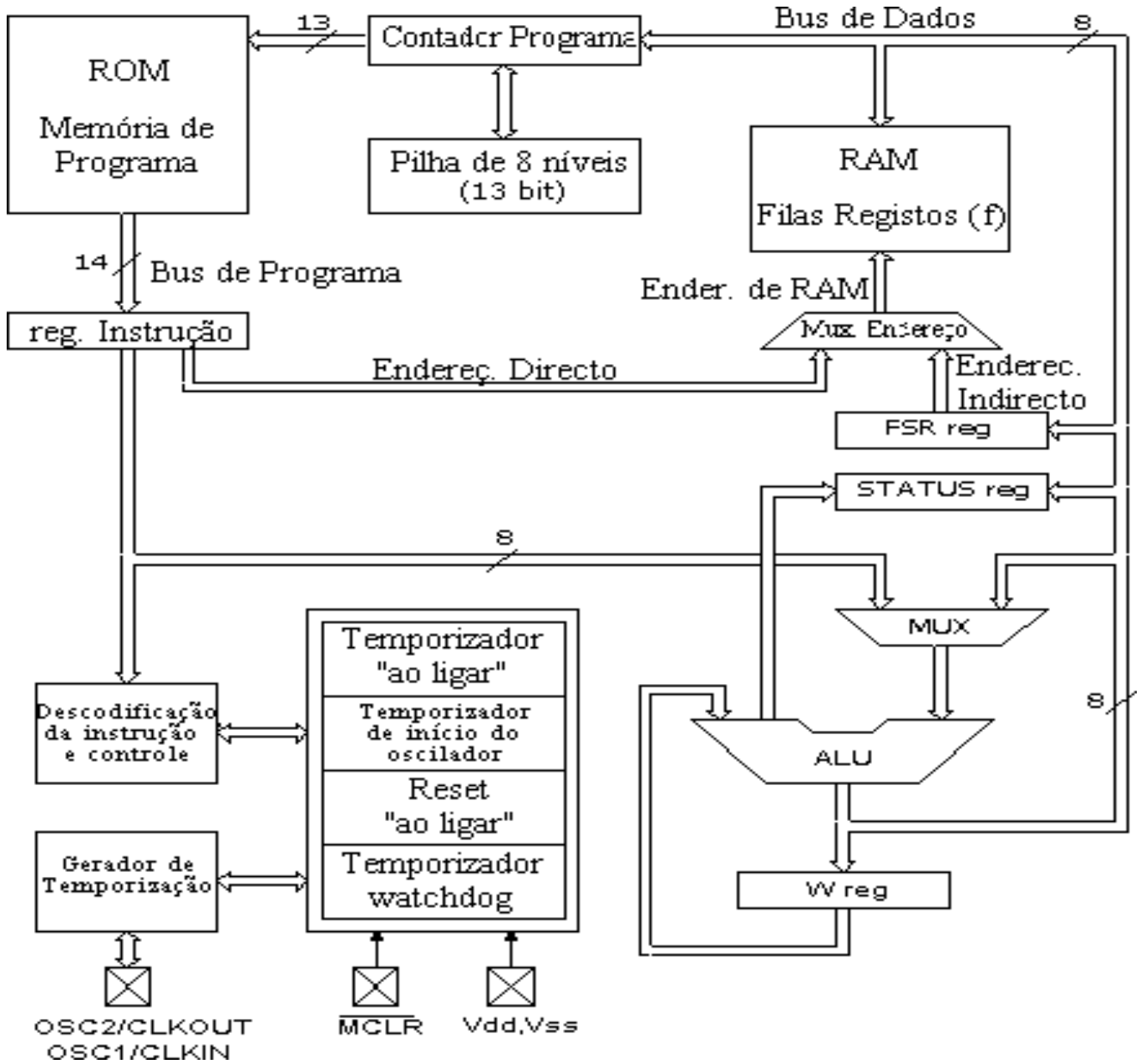
e) Reset quando o temporizador do watchdog (WDT) transborda estando no regime de SLEEP.



Unidade Lógica Aritmética (ALU)

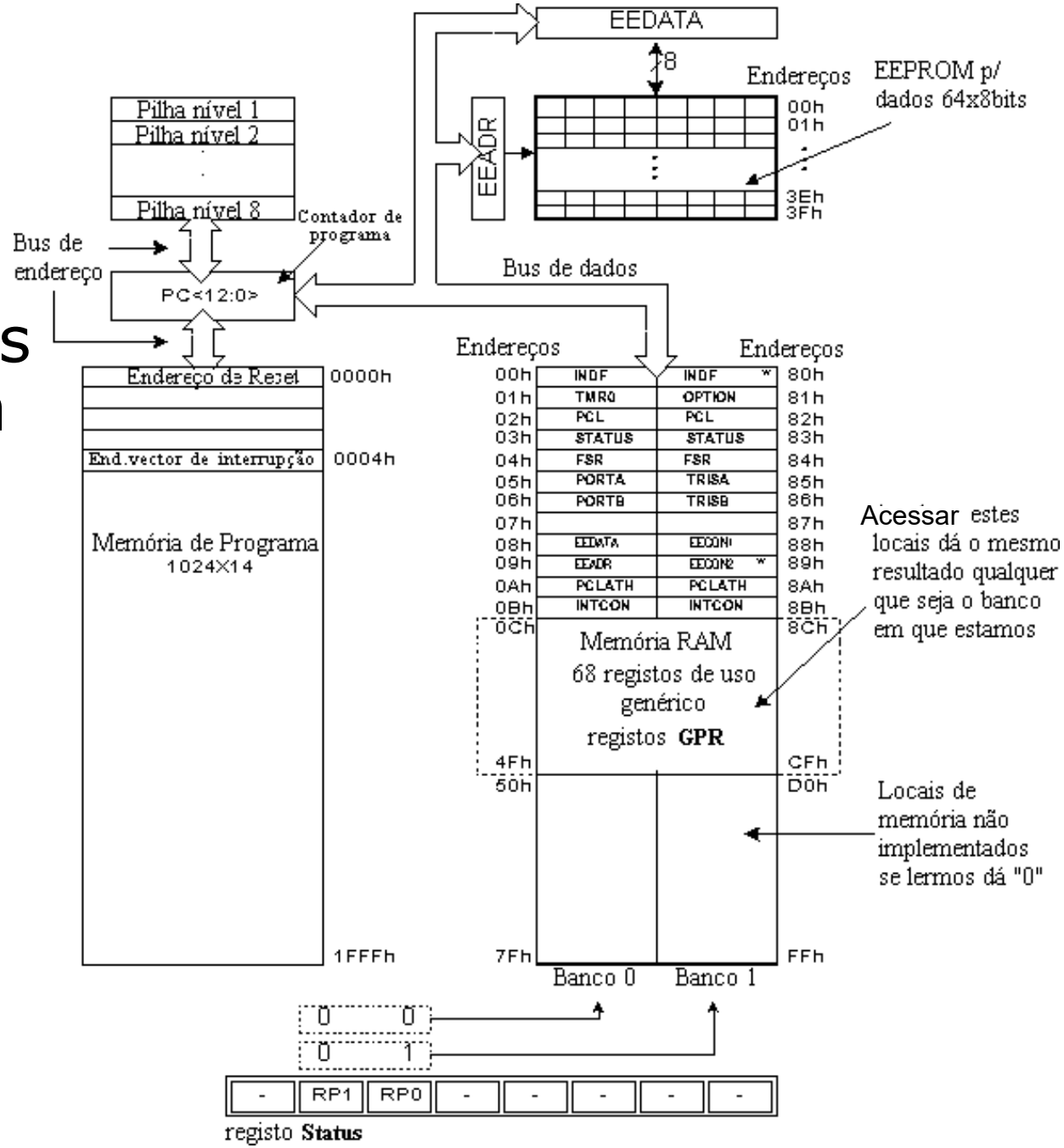
- A unidade lógica aritmética (ALU – Arithmetic Logic Unit), é responsável pela execução de operações de adição, subtração, deslocamento (para a esquerda ou para a direita dentro de um registo) e operações lógicas.
- O PIC16F84 contém uma unidade lógica aritmética de 8 bits e registos de uso genérico também de 8 bits.




ROM
Memória de Programa
13
Contador Programa
Bus de Dados
8
Pilha de 8 níveis (13 bit)
RAM
Filas Registos (f)
14
Bus de Programa
Ender. de RAM
reg. Instrução
Mux Endereço
Endereç. Directo
Enderec. Indirecto
FSR reg
STATUS reg
8
MUX
Temporizador "ao ligar"
Descodificação da instrução e controle
Temporizador de início do oscilador
Reset "ao ligar"
Gerador de Temporização
Temporizador watchdog
ALU
8
W reg
OSC2/CLKOUT
OSC1/CLKIN
MCLR
Vdd,Vss

Organização da Memória

- O PIC16F84 tem dois blocos de memória separados, um para dados e o outro para o programa.
- A memória EEPROM e os registos de uso genérico (GPR) na memória RAM constituem o bloco para dados e a memória FLASH constitui o bloco de programa

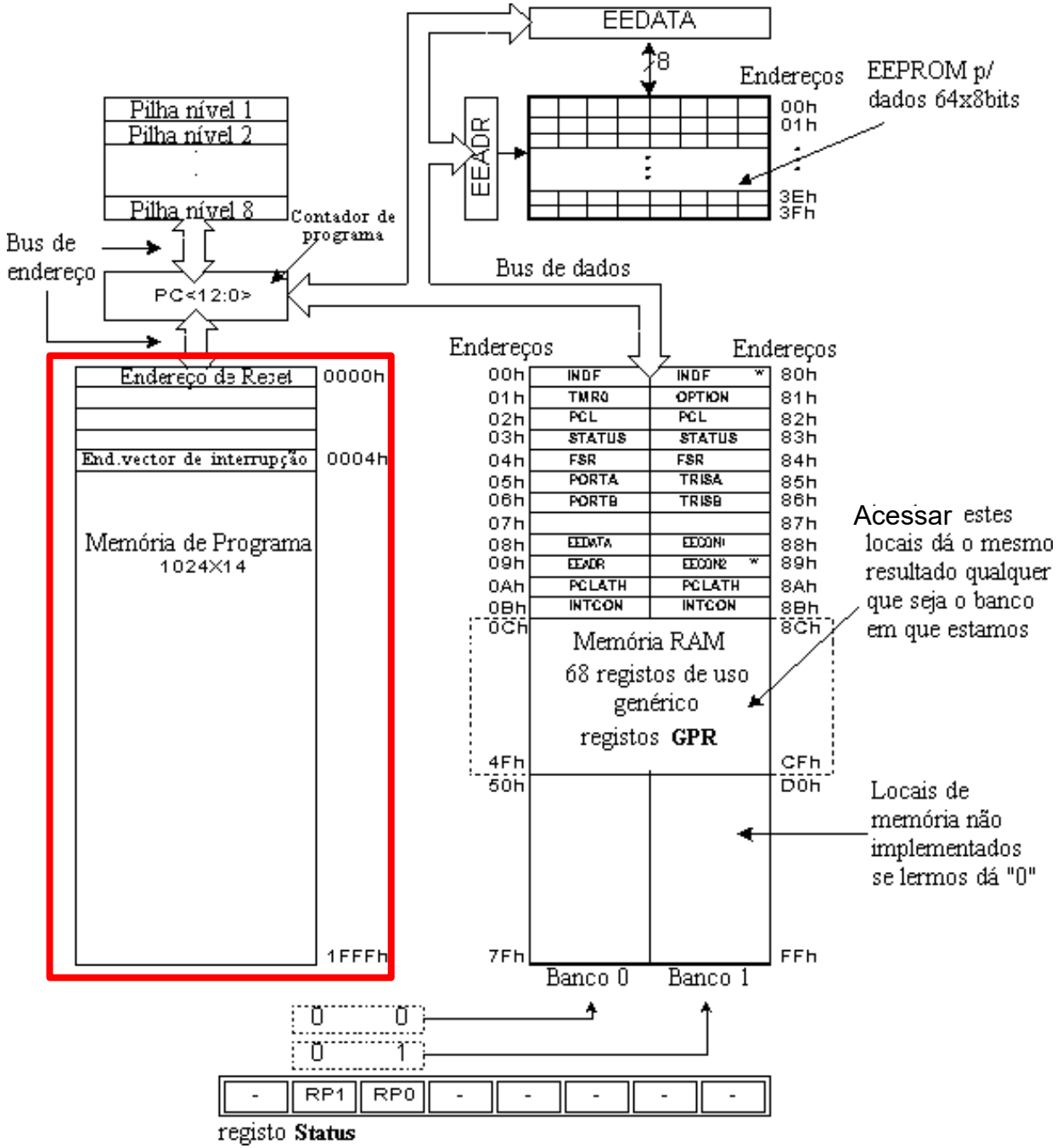


Organização da Memória

- **Memória de programa**

A memória de programa é implementada usando tecnologia FLASH, o que torna possível programar o microcontrolador muitas vezes antes de este ser instalado num dispositivo, e, mesmo depois da sua instalação, podemos alterar o programa e parâmetros contidos.

O tamanho da memória de programa é de 1024 endereços de palavras de 14 bits, destes, os endereços zero e quatro estão reservados respectivamente para o reset e para o vector de interrupção.



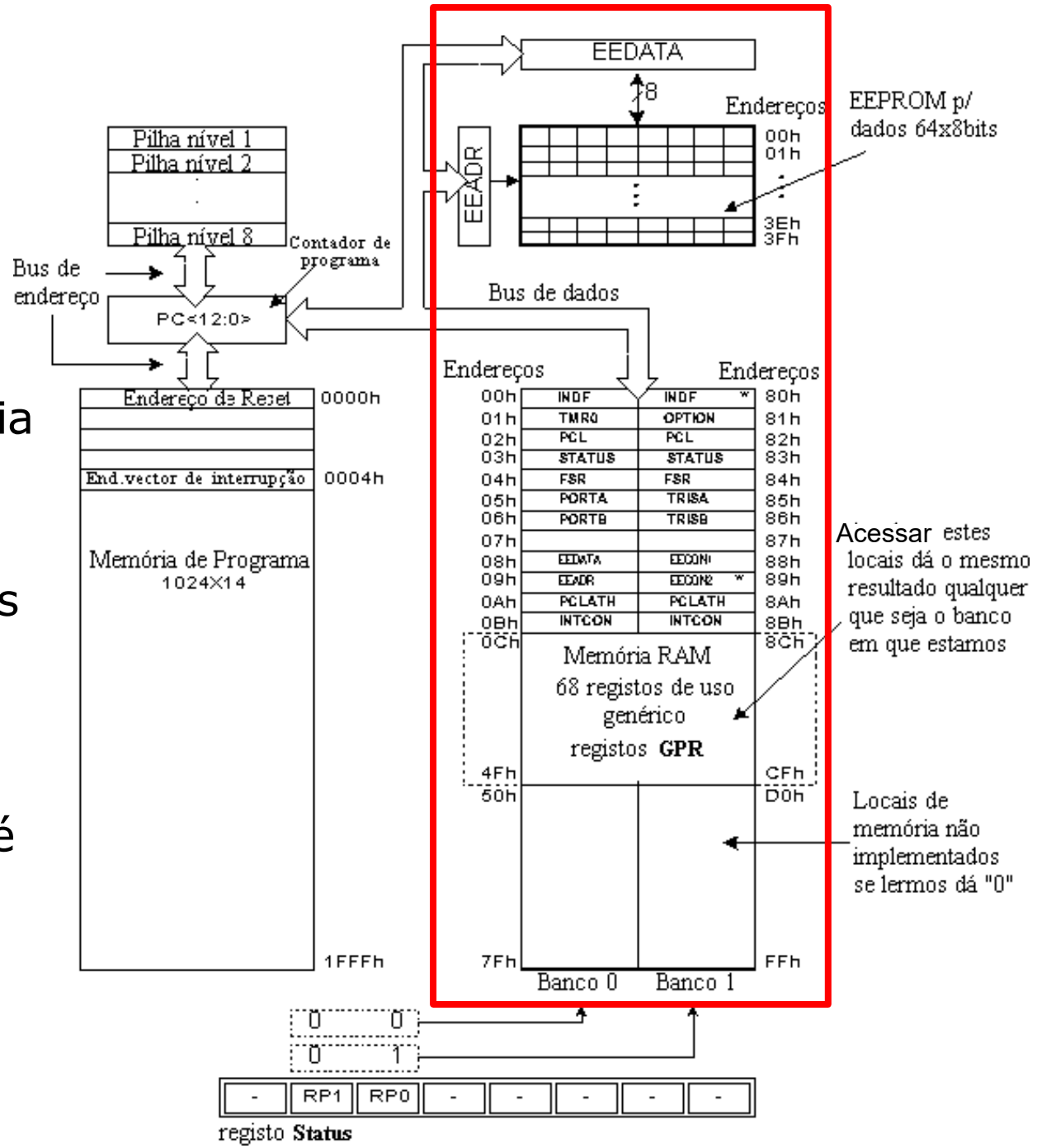
Organização da Memória

- **Memória de dados**

A memória de dados compreende memória EEPROM e memória RAM.

A memória EEPROM consiste em 64 posições para palavras de oito bits e cujos conteúdos não se perdem durante uma falha na alimentação.

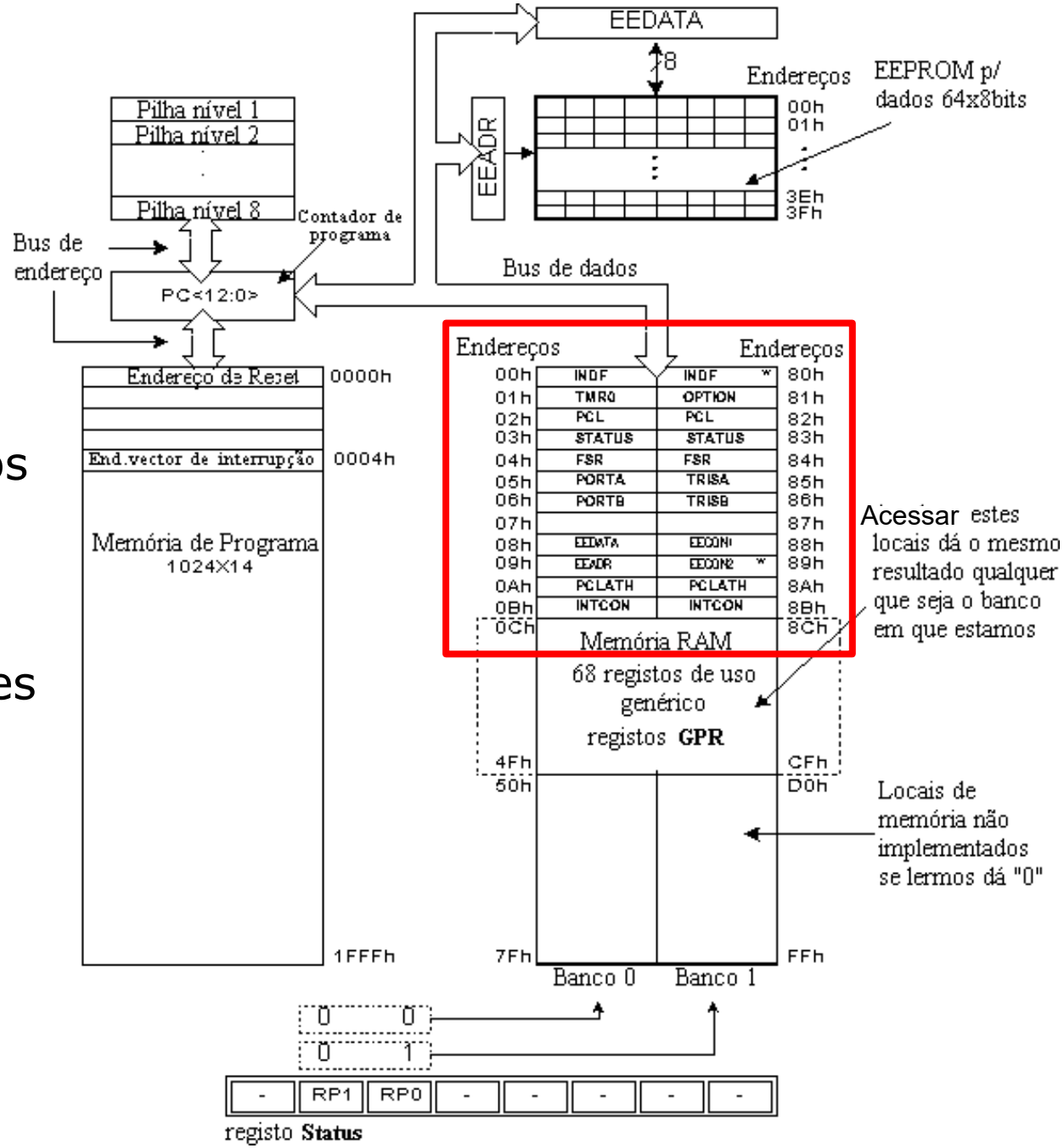
A memória EEPROM não faz parte diretamente do espaço de memória mas é acessada indiretamente através dos registos EEADR e EEDATA



Organização da Memória

Registos SFR

Os registos que ocupam as 12 primeiras localizações nos bancos 0 e 1 são registos especiais e têm a ver com a manipulação de certos blocos do microcontrolador. Estes registos são os **SFR** (Special Function Registers ou Registos de Funções Especiais).



Organização da Memória

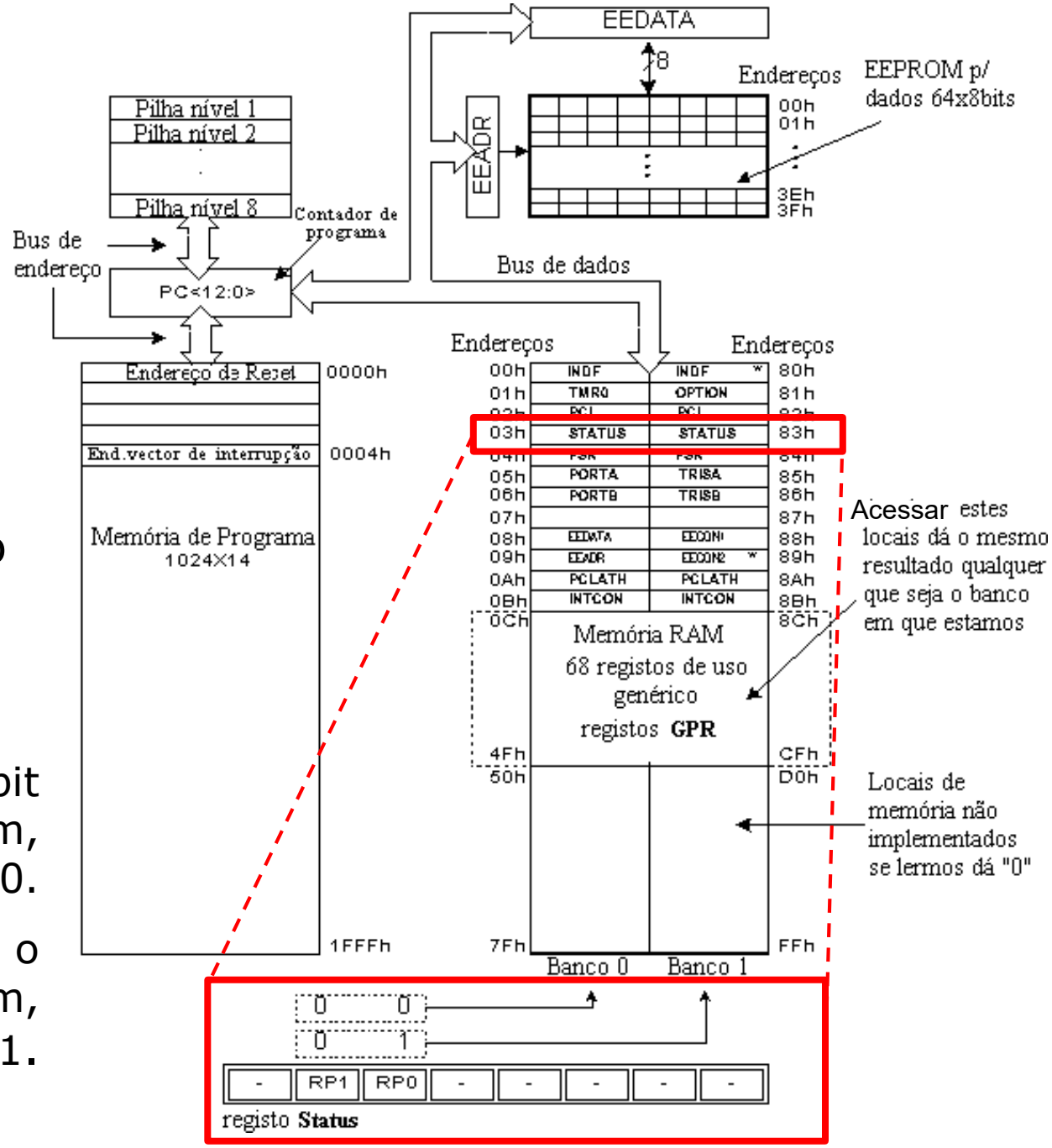
- **Bancos de Memória**

Além da divisão em 'comprimento' entre registos SFR e GPR, o mapa de memória está também dividido em 'largura' em duas áreas chamadas 'bancos'. A seleção de um dos bancos é feita por intermédio dos bits RP0 e RP1 do registo STATUS.

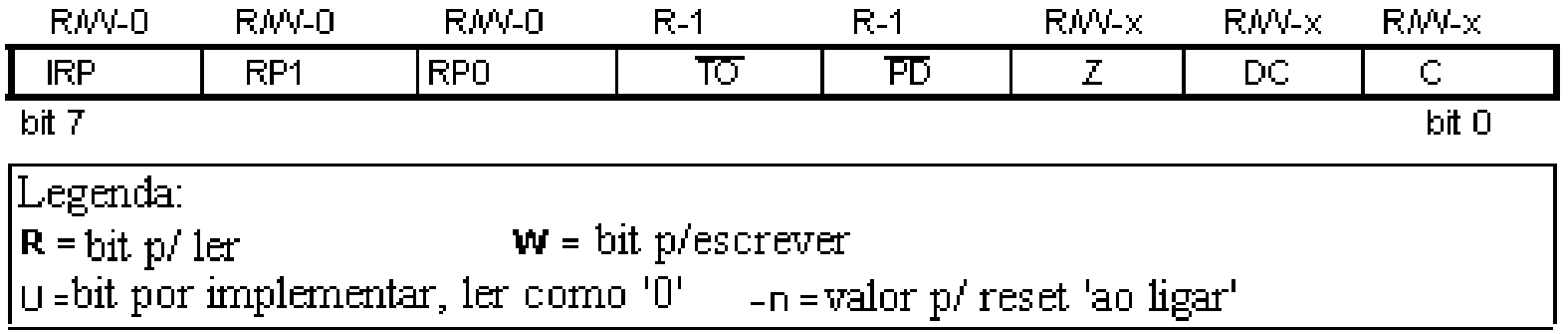
Exemplo :

bcf STATUS, RP0 ; A instrução BCF "limpa" o bit RP0 (RP0 = 0) do registo STATUS e, assim, coloca-nos no banco 0.

bsf STATUS, RP0 ; A instrução BSF põe a um, o bit RP0 (RP0 = 1) do registo STATUS e, assim, coloca-nos no banco 1.



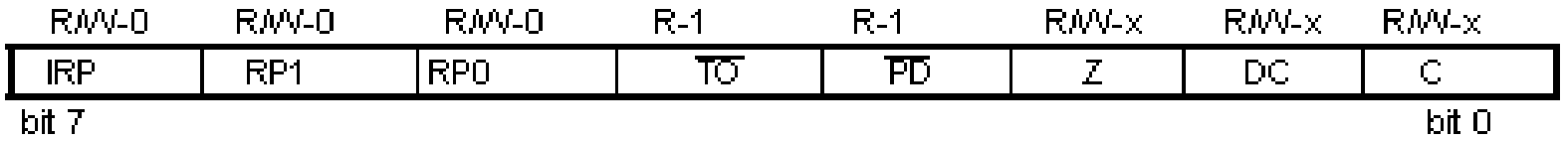
Registo STATUS



O registo de estado (**STATUS**), contém o estado da ALU (C, DC, Z), estado de RESET (TO, PD) e os bits para seleção do banco de memória (IRP, RP1, RP0). Considerando que a seleção do banco de memória é controlada através deste registo, ele tem que estar presente em todos os bancos.

Se o registo STATUS for o registo de destino para instruções que afetem os bits Z, DC ou C a escrita nestes três bits não é permitida

Registo STATUS



- **bit 0 C (Carry)** Transporte

Este bit é afetado pelas operações de adição, subtração e deslocamento. Assume o valor '1' (set), quando um valor mais pequeno é subtraído de um valor maior e assume o valor '0' (reset) quando um valor maior é subtraído de um menor.

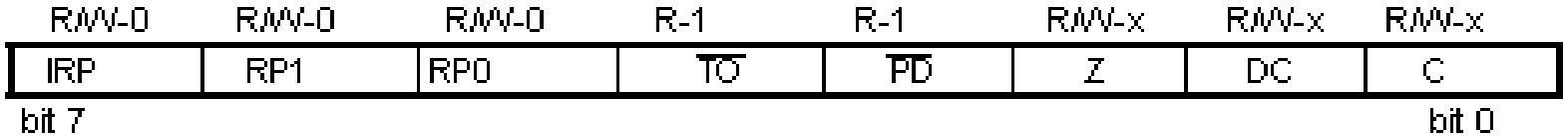
- 1= Ocorreu um transporte no bit mais significativo 0= Não ocorreu transporte no bit mais significativo
- O bit C é afetado pelas instruções ADDWF, ADDLW, SUBLW e SUBWF.

- **bit 1 DC** (Digit Carry) Transporte de dígito

Este bit é afetado pelas operações de adição, subtração. Ao contrário do anterior, DC assinala um transporte do bit 3 para o bit 4 do resultado. Este bit assume o valor '1', quando um valor mais pequeno é subtraído de um valor maior e assume o valor '0' quando um valor maior é subtraído de um menor.

- 1= Ocorreu um transporte no quarto bit mais significativo 0= Não ocorreu transporte nesse bit O bit DC é afetado pelas instruções ADDWF, ADDLW, SUBLW e SUBWF.

Registo STATUS



Legenda:
R = bit p/ ler **W** = bit p/escrever
U =bit por implementar, ler como '0' -n = valor p/ reset 'ao ligar'

- **bit 2 Z** (bit Zero) Indicação de resultado igual a zero.

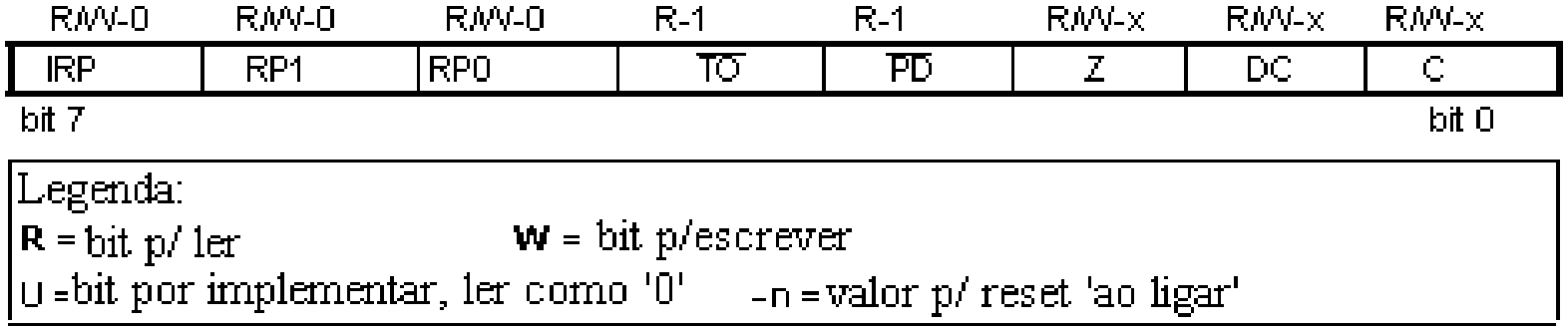
Este bit assume o valor '1' quando o resultado da operação lógica ou aritmética executada é igual a 0.

- 1= resultado igual a zero
- 0= resultado diferente de zero

- **bit 3 PD** (Bit de baixa de tensão – Power Down)
- Este bit é posto a '1' quando o microcontrolador é alimentado e começa a trabalhar, depois de um reset normal e depois da execução da instrução CLRWDT. A instrução SLEEP põe este bit a '0' ou seja, quando o microcontrolador entra no regime de baixo consumo / pouco trabalho. Este bit pode também ser posto a '1', no caso de ocorrer um impulso no pino RB0/INT, uma variação nos quatro bits mais significativos do porta B, ou quando é completada uma operação de escrita na DATA EEPROM ou ainda pelo watchdog.

- 1 = depois de ter sido ligada a alimentação
- 0 = depois da execução de uma instrução SLEEP

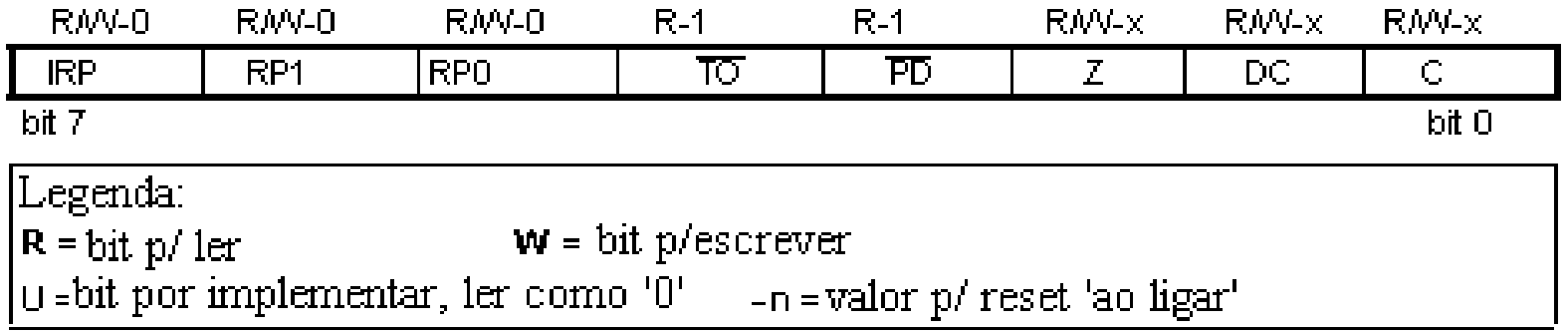
Registo STATUS



- **bit 4 TO** Time-out ; transborda do Watchdog
- Este bit é posto a ‘1’, depois de a alimentação ser ligada e depois da execução das
- instruções CLRWDT e SLEEP. O bit é posto a ‘0’ quando o watchdog consegue chegar ao fim da sua contagem (overflow = transbordar), o que indica que qualquer coisa não esteve bem.
- 1 = não ocorreu transborda 0 = ocorreu transborda

- **bits 5 e 6 RP1:RP0** (bits de seleção de banco de registos)
- Estes dois bits são a parte mais significativa do endereço utilizado para endereçamento direto. Como as instruções que endereçam diretamente a memória, dispõem somente de sete bits para este efeito, é preciso mais um bit para poder endereçar todos os 256 registos do PIC16F84. No caso do PIC16F84, RP1, não é usado, mas pode ser necessário no caso de outros microcontroladores PIC, de maior capacidade.
- 01 = banco de registos 1
- 00 = banco de registos 0

Registo STATUS



- **bit 7 IRP** (Bit de seleção de banco de registos)

Este bit é utilizado no endereçamento indireto da RAM interna, como oitavo bit 1 = bancos 2 e 3

- 0 = bancos 0 e 1 (endereços de 00h a FFh).

Registo OPTION

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Legenda:
R = bit p/ ler **W** = bit p/ escrever
U =não implementado, ler como '0' -n = valor p/ reset 'ao ligar'

- **bits 0 a 2 PS0, PS1, PS2** (bits de seleção do divisor Prescaler)

Estes três bits definem o fator de divisão do prescaler (divisor de frequência de clock, para uso dos contadores).

Bits	TMR0	WDT
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Registo OPTION

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Legenda:
R = bit p/ ler **W** = bit p/ escrever
U =não implementado, ler como '0' -n = valor p/ reset 'ao ligar'

- **bit 3 PSA** (Bit de Atribuição do Prescaler)

Bit que atribui o prescaler ao TMR0 ou ao watchdog. 1 = prescaler atribuído ao watchdog
0 = prescaler atribuído ao temporizador TMR0

- **bit 4 T0SE** (bit de seleção de borda ativo em TMR0)

Se for aplicado impulsos em TMR0, a partir do pino RA4/TOCK1, este bit determina se os impulsos ativos são os impulsos ascendentes (ativados na borda de subida) ou os impulsos descendentes (ativado na borda de descida) .
1 = borda descendente 0 = borda ascendente

- **bit 5 TOCS** (bit de seleção de fonte de clock em TMR0)

Este pino seleciona a fonte de impulsos que vai ligar ao temporizador. Esta fonte pode ser o clock do microcontrolador (frequência de clock a dividir por 4) ou impulsos externos no pino RA4/TOCKI.
1 = impulsos externos 0 = ¼ do clock interno

Registo OPTION

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Legenda:
R = bit p/ ler **W** = bit p/ escrever
U =não implementado, ler como '0' -n = valor p/ reset 'ao ligar'

- **bit 6 INDEDG** (bit de seleção de borda de interrupção)

Se esta interrupção estiver habilitada, é possível definir o borda que vai ativar a interrupção no pino RB0/INT.

1 = borda ascendente
0 = borda descendente

- **bit 7 RBPU** (Habilita pull-up nos bits do porta B)

Este bit introduz ou retira as resistências internas de pull-up do porta B.

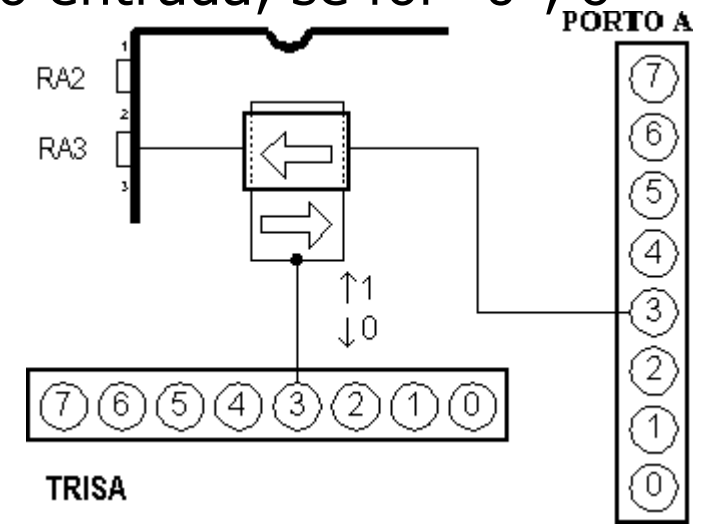
1 = resistências de "pull-up" desligadas
0 = resistências de "pull-up" ligadas

Portas

- As portas, são um grupo de pinos num microcontrolador que podem ser acessados simultaneamente, e, no qual nós podemos colocar uma combinação de zeros e uns ou ler dele o estado existente.
- Fisicamente, porta é um registo dentro de um microcontrolador que está ligado por fios aos pinos do microcontrolador.
- As portas representam a conexão física da Unidade Central de Processamento (CPU) com o mundo exterior.
- O microcontrolador usa-os para observar ou comandar outros componentes ou dispositivos.
- Para aumentar a sua funcionalidade, os mesmos pinos podem ter duas aplicações distintas, como, por exemplo, RA4/TOCKI, que é simultaneamente o bit 4 do porta A e uma entrada externa para o contador/temporizador TMR0. A escolha de uma destas duas funções é feita através dos registos de configuração. Um exemplo disto é o TOCS, quinto bit do registo OPTION. Ao selecionar uma das funções, a outra é automaticamente inibida.

Portas

- Todos os pinos dos portas podem ser definidos como de entrada ou de saída, de acordo com as necessidades do dispositivo que se está a projetar.
- Para definir um pino como entrada ou como saída, é preciso, em primeiro lugar, escrever no registo TRIS, a combinação apropriada de zeros e uns.
- Se no local apropriado de um registo TRIS for escrito o valor lógico "1", então o correspondente pino do porta é definido como entrada, se for "0", o pino é definido como saída.
- Todos os portas, têm um registo TRIS associado. Assim, para o porta A, existe o registo TRISA no endereço 85h e, para o porta B existe o registo TRISB, no endereço 86h.



Exercício

- Verificar tarefa no Moodle