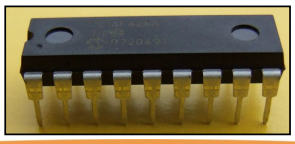
MICROPROCESSADORES

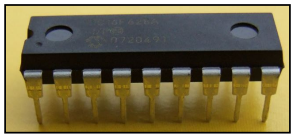
Prof. Ivair Teixeira

Labhardware.fav@unianhanguera.edu.br

ivairt@yahoo.com.br

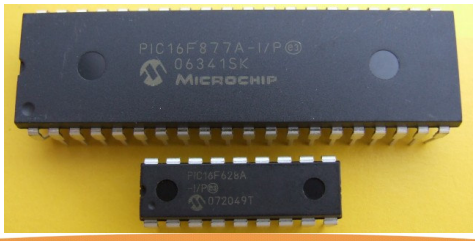


Conceito



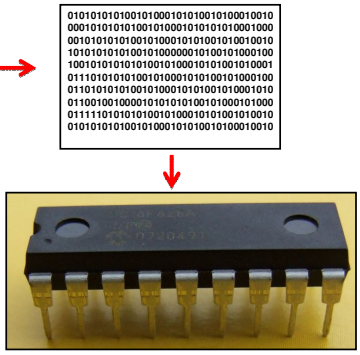
Microcontroladores - Conceito

Componente eletrônico dotado de uma memória programável, na qual pode ser gravada uma seqüência de instruções ou comandos, estruturadas na forma de um programa.



Microcontroladores - Conceito

```
PRINCIPAL
    MOVLW  B'00000001'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'00000010'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'00000100'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'00001000'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'00010000'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'00100000'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'01000000'
    MOVWF  PORTB
    CALL   TEMPO
    MOVLW  B'10000000'
    MOVWF  PORTB
    CALL   TEMPO
    GOTO   PRINCIPAL
```



Microcontroladores - Conceito

Microprocessador

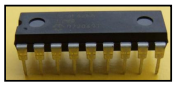
HD = 250.000.000.000 bytes

Clock = 2.000.000.000 Hz

RAM = 1.000.000.000 bytes

E/S = Serial, Paralela, USB...

Gabinete, placa mãe, cabos teclado, mouse...



Microcontrolador

Flash = **2.000** bytes

Clock = **1.000.000** Hz

RAM = **224** bytes

E/S = 16 pinos

...tudo em um chip

Microcontroladores - Conceito

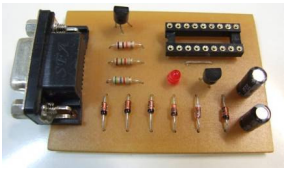
- **Automação Industrial**
- **Casa inteligente**
- **Alarme automotivo / residencial.**
- **Robô / braço mecânico.**
- **Relógio / Cronômetro / Contador.**
- **Sistemas Cliente-Servidor WEB.**
- **Aplicações de I.A.**
- **Comunicação com PC, celular, GPS.**
- **Brinquedos.**

Microcontroladores - Conceito

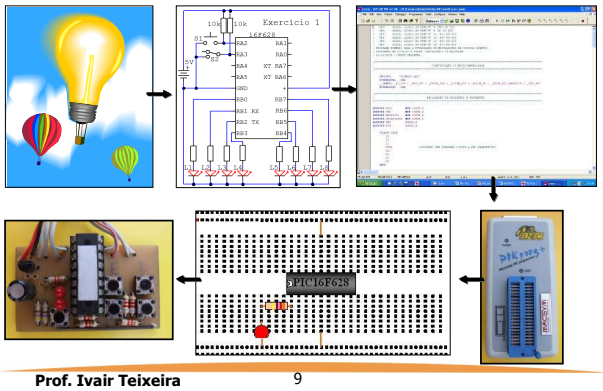
- Controle de motores.
  - Corrente contínua,
  - Servo-motores,
  - Motores de passo.
- Gerenciador comunicação.
  - RS232,
  - RS485,
  - i2c,
  - Infra-Vermelho,
  - Óptica.

Microcontroladores - Conceito

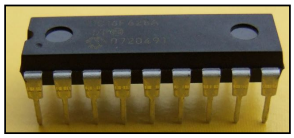
Gravadores



Microcontroladores - Conceito



Linguagem de máquina



Conjunto de instruções

Quem executa um programa é o hardware, e ele só entende uma sequência de instruções de máquina em código binário.

01010100101001001001000010010101
01001001000100101110010100010000
01001111000100000100111010100010
01010100010001010010010001010100

Conjunto de instruções

Instrução de máquina = é composta de códigos binários representando instruções, endereços e dados (set).

Formato das instruções = OPCODE + Op1 + Op2 ...

- **Opcode (Código da Operação):** é operação a ser realizada pelo processador. Cada opcode deverá ter um código único que o identifique.
- **Op (Operando):** é o dado, ou o endereço de memória onde está o dado, ou o endereço onde vai ser salvo o resultado.

Opcode	Op1	Op2
0001	100010100	101010001

Conjunto de instruções

OPCODE -Tipos de operações:

Matemáticas:
- Aritméticas, lógicas, de complemento, de deslocamento, ...

Movimentação de dados
- Memória ←→ Acumulador.

Controle
- Desvio da sequência de execução, parar, ...

Entrada e Saída
- Leitura e escrita em dispositivos de Entrada / Saída.

Instruções de máquina

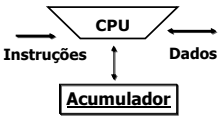
Instrução	Significado		Código
Load	Carregar no acumulador	AC ← op	0001
Store	Salvar na memória	op← AC	0010
Add	Somar	AC←AC+op	0011
Sub	Subtrair	AC←AC-op	0100
Mul	Multiplicar	AC←AC*op	0101
Div	Dividir	AC←AC/op	0110
Goto	Desviar	PC←op	0111
Read	Ler	Op←Entrada	1000
Print	Imprimir	Saída←op	1001

Conjunto de instruções

Memória de Programa

HD ou Flash

end	opcode
00000001	000100000111
00000010	001100000010
00000011	001000001010
00000100	
00000101	
00000110	
00000111	
00001000	
00001001	
00001010	
00001011	
00001100	
00001101	



Carregar = 0001
Salvar = 0010
Somar = 0011

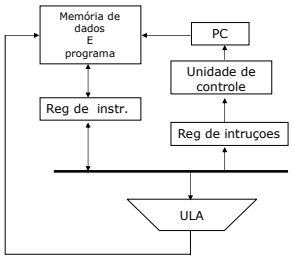
Memória de dados (RAM)

Registradores

end	conteúdo
00000001	10100010
00000010	00000001
00000011	00000010
00000100	00000011
00000101	01010111
00000110	11100110
00000111	00001111
00001000	
00001001	
00001010	
00001011	
00001100	
00001101	

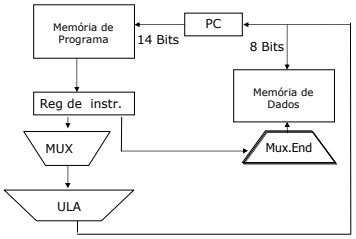
Arquitetura Von Newmann

Só há uma memória para dados e programas. Utiliza a tecnologia **CISC** (*Complex Instruction Set Computer*), Computador com um Conjunto Complexo de Instruções



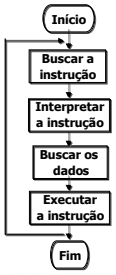
Arquitetura Harvard

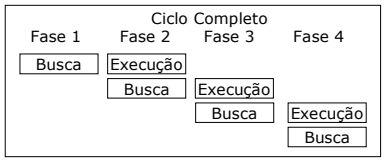
A memória de dados está separada da memória de programa. Utiliza a tecnologia **RISC** *(Reduced Instruction Set Computer).* Computador com Conjunto Reduzido de Instruções.



Arquitetura Harvard

A frequência de *clock* é dividida em quatro fases, isso permite a técnica conhecida por PIPELINE, na qual a CPU realize a busca de uma ou mais instruções enquanto outra é executada





Arquitetura

HARVARD vs VON NEUMANN

Harvard

Von Neumann

Instruções de máquina

•Arquitetura: Harvard – RISC – 35 instruções
•Palavras de 12, 14 e 16 bits.

Exemplo: somar 12 ao conteúdo do endereço 0x27

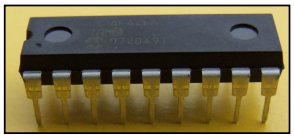
Microcontrolador comum: 3 palavras 8 bits

1 – Opcode somar.
2 – Operando 12.
3 – Operando 0x27.

Microcontrolador PIC: 1 palavra 14 bits.

1 – Somar + 12 + local

Básico



Conceito Geral

Registradores (F de File Registers) são espaços de memória que armazenam um byte (uma informação de 8 bits 0 ou 1), como uma variável.

Existem operações (comandos) para **ler ou escrever** em um registrador:

- •Um bit por vez (A operação acessa apenas 1 dos 8 bits do registrador).
- •Um byte inteiro (A operação acessa todos os 8 bits do registrador)

Acumulador (W de Work) são registradores especiais usados para intercâmbio geral de dados.

No PIC16F628, até o endereço 0X19 a memória é reservada para a CPU (registradores de uso especial), após esse endereço podemos alocar nossas variáveis (registradores de propósito geral).

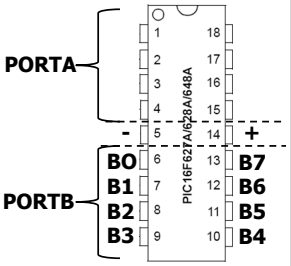
```
CBLOCK  0X20
    VAR1
    VAR2
    . . .
ENDC
```


Conceito Geral

Um exemplo de **registrador** do PIC é o PORTB, utilizado pelo sistema para controlar quais terminais estão ligados ou desligados. O PORTB é composto por 8 bits, cada um controla um terminal do PIC, conforme a figura.

Registrador: PORTB

Bits:7 6 5 4 3 2 1 0



Conceito Geral

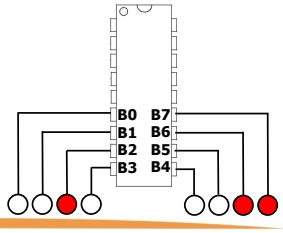
Se escrever o valor 1, em qualquer bit do registrador, a saída correspondente ficará LIGADA, ou seja com 5 Volts.

Se escrever o valor 0, a saída ficará DESLIGADA, ou seja com 0 Volts

Observe que a posição dos bits na variável é invertida em relação aos pinos no esquema elétrico.

PORTB

Valor:	1	1	0	0	0	1	0	0
Bits:	7	6	5	4	3	2	1	0

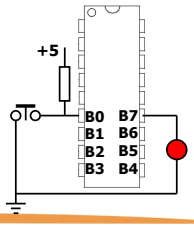


Conceito Geral

O básico da programação consiste em:

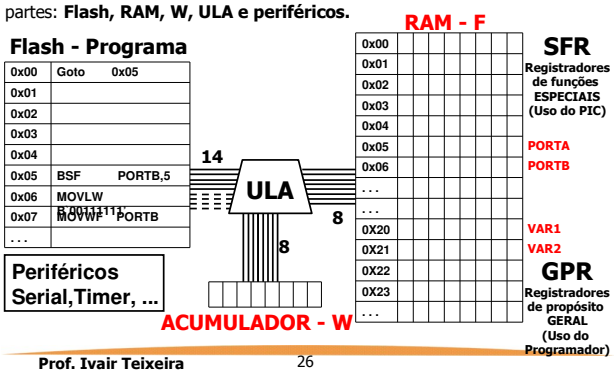
- Testar Entradas (verificar se um botão foi pressionado...)
- Acionar saídas (ligar um LED...)
- Aguardar um tempo determinado (visualização dos eventos..)
- Realizar cálculos (soma, subtração, decremento, operações lógicas...)

•Por exemplo: se o botão conectado ao pino **PORTB,0** for pressionado, pisque o LED conectado ao pino **PORTB,7** com um intervalo de 1 segundo.



Conceito Geral

Neste momento o PIC pode ser dividido, de uma forma bem simples, em partes: **Flash, RAM, W, ULA e periféricos.**



Conceito Geral

Na **Flash,** está armazenado o programa que será executado.

A memória **RAM,** divide-se em:

- Registradores especiais, área reservada ao PIC para os registradores de controle, como PORTA, PORTB, TRISA, TRISB, STATUS, INTCON, ...
- Registradores Gerais, área que inicia no endereço 0x20 (pic16f628A) e é destina a criação das variáveis do usuário.

O acumulador, conhecido com **W,** trabalha como um registrador de intercâmbio de dados. No PIC não é possível mover o conteúdo de um registrador(ou variável) diretamente para outro, deve-se usar o acumulador como intermediário.

A **ULA** é a unidade lógica Aritmética, ou seja o processador da instruções.

Os **periféricos** são Timers, comparadores analógicos, módulo de comunicação serial, PWM, etc.

E, como o PIC é um computador logicamente contém PC, pilha, etc.

Conceito Geral

As instruções em assembly são compostas por letras que representam a abreviação de uma frase. Sua composição utiliza termos como:

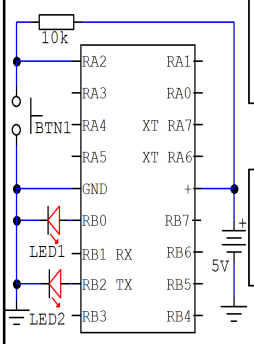
L = **L**iteral (valor, número).
W = **W**ork (Acumulador de trabalho)
F = **F**ile Register (Arquivo de registro, variável na memória)
B = **B**it (um dos 8 bits de um registrador qualquer)
S = **S**et (Valor 1, positivo)
C = **C**lear (limpar, Valor 0 ou negativo)
Z = **Z**ero (valor 0, quando usado em testes de decremento)
S = **S**kip (saltar a próxima instrução, após um teste comparativo)

Observe que S pode ser "set"ou "skip" dependendo da operação.

Exemplo:

MOVLW .25 = **MOV**a a **L**iteral 25 para o Acumulador (**W**ork)
BSF PORTB,5 = **S**et (1) o **B**it 5 do **F** (registrador) PORTB

Interpretar esquema elétrico



BTN1 é um botão ligado ao pino **PORTA, 2**
Nessa configuração RA2 é uma entrada que está em nível 1 (5 volts) pelo resistor de 10k.

Ao pressionar o botão, o pino RA2 é aterrado e recebe um nível 0.

PORTA = (BTN1)

LED1 é um LED ligado ao pino **PORTB, 0**
LED2 é um LED ligado ao pino **PORTB, 2**
Um dos terminais do LED está conectado ao negativo, portanto para o LED ligar deve-se impor nível 1 (5 Volts) no respectivo pino do PIC

PORTB = (LED2, LED1)

Acionar Saídas

Para controlar as saídas individuais:

```
BSF     PORTB,0     ;Bit Set File → Liga (5 Volts)
BCF     PORTB,0     ;Bit Clear File → Desliga ( 0 Volts)
```

Para controlar as saídas em conjunto
W= acumulador.

```
MOVLW   B'00000101'   ;Move a literal 00000101 para W
MOVWF   PORTB         ;Move W para o PORTB
```

PORTB =	0	0	0	0	0	5V	0	5V

Testar Entradas

Para testar os botões:

```
TESTA
   BTFSS   PORTA,2      ;Testa RA2 e salta a próxima se for 1
   GOTO    LIGA         ;RA2 = 0 vai para a rotina LIGA
   GOTO    TESTA        ;RA2 = 0 vai para a rotina TESTA

                        ou
TESTA
   BTFSC   PORTA,2      ;Testa RA2 e salta a próxima  se for 0
   GOTO    TESTA        ;RA2 = 0 vai para a rotina TESTA
   GOTO    LIGA         ;RA2 = 0 vai para a rotina LIGA
```

Testa(0 ou 1?) o **B**it(2) de **F**(PORTA) e **S**alta(próxima) se for 1 (**S**et)
Testa(0 ou 1?) o **B**it(2) de **F**(PORTA) e **S**alta(próxima) se for 0 (**C**lear)

Repetir Eventos

```
PISCA
   BSF     LED1     ;Liga (5 Volts) o PORTB,0
   CALL    TEMPO    ;Chama a rotina tempo (deve ser implementada)
   BCF     LED1     ;Desliga (5 VOLTS) o PORTB,0
   CALL    TEMPO    ;. . .
   DECFSZ  AUX,F    ;Decrementa AUX e salta próxima linha se aux = 0
   GOTO    PISCA    ;Vai para PISCA (cai aqui se AUX diferente de 0)
   GOTO    INICIO   ;Vai para INICIO (cai aqui se AUX ficou igual a 0)
```

```
INICIO
   . . .
   MOVLW   .10      ;Move a Literal 10 para a acumulador (W)
   MOVWF   AUX      ;Move W para a variável AUX
   GOTO    PISCA    ;Vai para a rotina pisca
```


Notações

Tipos de literais

```
MOVLW   .15          ;Literal em decimal.
MOVLW   B'00001111'  ;Literal em binário.
MOVLW   0XF          ;Literal em hexadecimal.
MOVLW   'a'          ;Literal em ASCII.
```


Saltar Linhas com $+k e $-k

```
PISCA
   BSF     LED1     ;Liga (5 Volts) o PORTB,0
   CALL    TEMPO    ;Chama a rotina tempo (deve ser implementada)
   BCF     LED1     ;Desliga (0 VOLTS) o PORTB,0
   CALL    TEMPO
   DECFSZ  AUX1,F   ;Decrementa AUX e salta próxima linha se AUX==0
   GOTO    PISCA    ;Vai para PISCA (cai aqui se AUX != 0)
   BSF     LED2     ;Liga (5 Volts) o PORTB,0 (cai aqui se AUX == 0)
   CALL    TEMPO
   BCF     LED2     ;Desliga (5 VOLTS) o PORTB,0
   CALL    TEMPO
   DECFSZ  AUX2,F   ;Decrementa AUX2 e salta próx. linha se AUX2==0
   GOTO    $-5      ;Volta 5 linhas e repete somente essas linhas
   GOTO    INICIO   ;Vai para INICIO
```


Declarar de Variáveis

```
CBLOCK 0X20          ;0X20 É O INÍCIO DA MEMÓRIA RAM
   T1                ;ALOCAÇÃO DE T1 NO ENDEREÇO 0X20
   T2                ;ALOCAÇÃO DE T2 NO ENDEREÇO 0X21
   T3                ;ALOCAÇÃO DE T3 NO ENDEREÇO 0X22
ENDC
```

OU

```
T1    EQU  0X20      ;ALOCAÇÃO DE T1 NO ENDEREÇO 0X20
T2    EQU  0X21      ;ALOCAÇÃO DE T2 NO ENDEREÇO 0X21
T3    EQU  0X22      ;ALOCAÇÃO DE T3 NO ENDEREÇO 0X22
```


Definir Macros e Constantes

Constantes e macros, ajudam a criar "apelidos"que facilitam o entendimento do código.

```
;Constantes: o "apelido" substitui o pino do PIC
#DEFINE   SAIDA    PORTB       ;SAIDA = PORTB
#DEFINE   BTN_1    PORTA, 0    ;BTN1 = PORTA, 0
#DEFINE   BTN_2    PORTA, 1

;Macros: o "apelido" substitui a operação
#DEFINE   BANCO1   BSF   STATUS,RP0
#DEFINE   LIGA     BSF   PORTB, 0
#DEFINE   DESL     BCF   PORTB, 0
```

Comparar Valores

O bit **Z** do registrador **STATUS** sempre assume o valor **1** se o resultado da operação anterior resultou em **0**.

STATUS – Armazena *flags* matemáticos e de estados da CPU.

					1		
IRP	RP0	RP1	T0	PD	**Z**	DC	C

Para comparar se os valores são iguais deve-se: Copiar o valor em W,Realizar uma operação XORLW entre W e a variável e **Testar se a operação resultou em 0** (STATUS, Z = 1).

```
MOVLW   .10       ;COPIA 10 EM W
XORWF   AUX,W     ;XOR ENTRE AUX E W, RESULTADO EM W
BTFSS   STATUS,Z  ;SE STATUS,Z = 1, PULA PRÓXIMA LINHA
GOTO    ROT_NAO   ;NÃO SÃO IGUAIS
GOTO    ROT_SIM   ;SÃO IGUAIS
```

AUX= 00001010
W = 00001010
XOR= 00000000

Subtração em assembly

Subtração entre o valor da variável AUX e W.

```
SUBWF   f,d     ;d = f-W
```

Se **AUX = 10** **W= 8** → Resultado = **2**

```
SUBWF   AUX, W  ;salva o resultado em w
SUBWF   AUX, F  ;salva o resultado em auxiliar
```

Subtração entre o valor da literal 8 e W.

```
SUBLW   k       ;k-W)
```

Se **k = 8** **W= 2** → Resultado = **6**

```
SUBLW   .8      ;w=6
```


Inverter Bits E Nibbles

•Para inverter os **BITs** de uma variável:

```
CONF    f,d
```

•Exemplo:

```
MOVLW   B'11001111'
MOVWF   AUX             ;AUX = 11001111
CONF    AUX, F          ;AUX = 00110000
```

•Para inverter os **NIBBLEs (4 bits)** de uma variável:

```
SWAPF   f,d
```

•Exemplo:

```
MOVLW   B'10011111'
MOVWF   AUX             ;AUX = 10011111
SWAPF   AUX, F          ;AUX = 11111001
```


Rotina Tempo

```
TEMPO
   MOVLW    .12     ;MOVE 12 PARA O ACUMULADOR W
   MOVWF    T1      ;MOVE W PARA T1     (T1=12)
   MOVLW    .166
   MOVWF    T2      ;T2 = 166
   MOVLW    .166
   MOVWF    T2      ;T2 = 166
   DECFSZ   T1      ;DECREMENTA T1 E SALTA A PROX. SE T1==0
   GOTO     $-1     ;VOLTA 1 LINHA
   DECFSZ   T2      ;DECREMENTA T2 E SALTA A PROX. SE T2==0
   GOTO     $-5     ;VOLTA 5 LINHAS
   DECFSZ   T3      ;DECREMENTA T3 E SALTA A PROX. SE T3==0
   GOTO     $-9     ;VOLTA 9 LINHAS
   RETURN
```


Instruções de máquina

Instruções = Palavras de 14 bits.

As instruções do PIC podem ser de 4 tipos.

1 - MNEMÔNICO k

- •O operando k é armazenado no acumulador w
  - •MOVLW .200
  - •*Opcode* + operando
- •O operando k é parte do endereço em uma instrução de desvio (Instruções de desvio utilizam 2 ciclos de máquina. Por que?)
  - •GOTO PRINCIPAL
  - •CALL TESTA_ENTRADA
  - •*Opcode* + operando

Instruções de máquina

2 - MNEMÔNICO reg

- •O operador reg representa um registrador.
  - •MOVWF PORTA
  - •*Opcode* + operando

3 - MNEMÔNICO reg, des

- •Existem 2 operandos reg e des.
  - •reg = Um registrador (geral ou especial).
  - •des = bit indicador do destino do resultado.
    - •W ou 0 = o resultado é armazenado em W;
    - •F ou 1 = o resultado é armazenado no registrador).
  - •ADDWF SOMA, F
  - •*Opcode* + operando + operando

Instruções de máquina

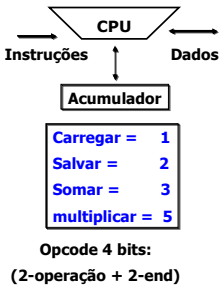
4 - MNEMÔNICO reg, bit

- Existem 2 operandos reg e bit.
  - reg = Um registrador (geral ou especial).
  - bit = um dos bits que compõe o registrador.
  - BSF PORTA, 2
  - *Opcode* + operando + operando

Exercícios 1

Memória de Programa (HD)

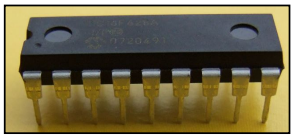
End (hex)	Opcode (hex)
01	0107
02	0304
03	0503
04	020D
05	
06	
07	
08	
09	
0A	
0B	
0C	
0D	



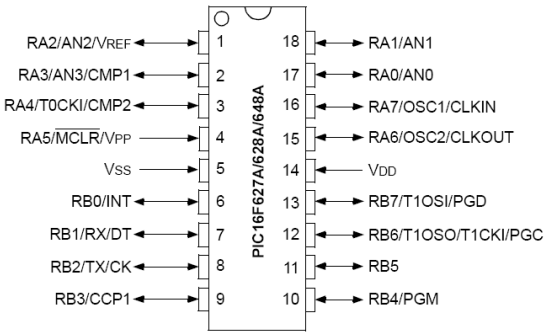
Memória de dados (RAM)

End.(hex)	Conteúdo(dec)
01	12
02	50
03	02
04	05
05	09
06	25
07	15
08	
09	
0A	
0B	
0C	
0D	

PIC 16F628A

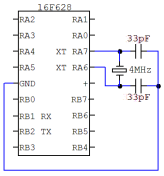


Microcontrolador PIC



Microcontrolador PIC

- Capacidade de **Pipeline**
- 1 instrução por ciclo de máquina (exceto instr. de desvio).



$$\frac{\text{Frequência}}{4} = \frac{4.000.000}{4} = 1\text{Mhz}$$

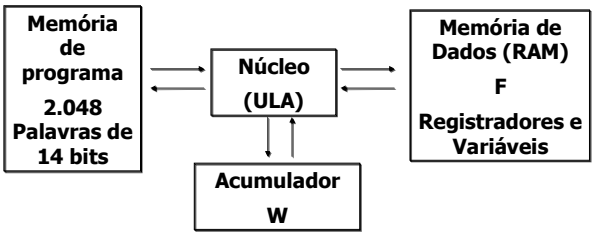
LP – Cristal até 200KHz.
XT – Cristal até 4Mhz.
HS – Cristal até 20Mhz.
Resistor/Capacitor externo sem saída de *clock*
Resistor/Capacitor externo com saída de *clock*
Oscilador RC interno sem saída de *clock*
Oscilador RC interno com saída de *clock*
Oscilador externo
Ver tabela – Palavra de configuração

PIC 16F628 - Características

- Memória de programa – Flash → 2048 palavras de 14 bits.
- Memória de dados – RAM → 224 palavras de 8 bits.
- Memória persistente – EEPROM → 128 palavras de 8 bits.
- Processamento – 1.000.000 instruções/s → 1us cada.
- Portas – 16 pinos Entradas e/ou Saídas.
- Alimentação – 3 a 5.5 Volts (tipicamente 5 V).
- Apenas 35 instruções.

PIC 16F628 - Características

Organização do PIC.



PIC 16F628 - Características

Memória de programa.

- **ROM** (*Read Only Memory*) – Já saem programados de fábrica.
- **OTP** (*One Time Programmable*) – ou PROM (*programmable read-only memory*) Podem ser gravados apenas uma vez.
- **EPROM** (*Erasable Programmable Read-Only Memory*) – *Dispositivos* que contém uma janela pela qual pode se apagado por luz ultravioleta.
- **FLASH** – Possuem memória do tipo flash que podem ser escrita/apagada entre e 100.000 1.000.000 de vezes.

PIC 16F628 - Características

Mapa da memória de programa

Vetor de *reset*	000	0
Vetor de interrução	004	4
Memória de programa	005	5
	7FF	2048

Endereços com 11 bits.

PIC 16F628 - Características

Acumulador

Espaço na memória de 8 bits com as seguintes características:

- Pode ser usado como destino de operações matemáticas e lógicas.
- Não está mapeado na memória RAM.
- É utilizado como "ponte" entre registradores.
- É referenciado, na programação como **W** (*work*).

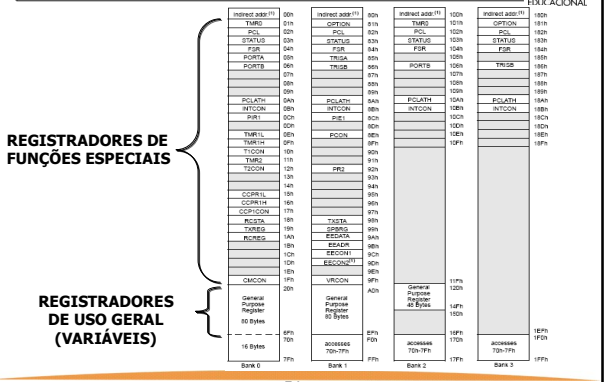
PIC 16F628 - Características

Memória RAM

É utilizada para abrigar registradores internos e é dividida em duas partes distintas:

- **SFR** (*Special Function Registers*) – São utilizados para controlar periféricos e dispositivos internos.
- **GPR** (*General Purpose Registers*) – São utilizados para o armazenamento temporário de dados e informações do usuário.
- São referenciados na programação com **F** (*File registers*).

PIC 16F628 - Características



PIC 16F628 - Características

Memória RAM.

Dentre os registradores podem ser destacados.

- **STATUS –** Armazena *flags* matemáticos e de estados da CPU.
- **PORTx** – Leitura ou escrita de informações nos pinos do PIC.
- **TRISx** – Configuração dos pinos do PIC (entrada ou saída)
- **INTCON –** Utilizado para controle das interrupções.

PIC 16F628 - Registradores

REGISTRADORES.

STATUS – Armazena *flags* matemáticos e de estados da CPU.

IRP	RP1	RP0	TO\	PD\	Z	DC	C

Bit0

C – Nível 1 se houve transporte ou empréstimo de bit, no byte, após uma operação.

DC – 1 se houve Transporte ou empréstimo de bit, no nibble, após uma operação.

Z – 1 se a operação resultou em 0.

PD\ – Quando em 0 indica que o ocorreu uma instrução *Sleep.*

TO\ – Quando em nível 0 indica que o Watchdog gerou uma interrupção.

RP1 e RP0 – Seleção do banco de memória.

- **0 0 = Banco 0.**
- **0 1 = Banco 1.**

IRP – Seleção do banco para endereçamento indireto.

PIC 16F628 - Registradores

REGISTRADORES.

INTCON– Controle de interrupções.

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Bit0

RBIF – *Flag* sinalizador de alteração de nível nos pinos RB4 a RB7.

INTF – *Flag* sinalizador de interrupção externa.

T0IF – *Flag* sinalizador de transbordamento do TMR0 (Timer 0).

RBIE – Habilitação de interrupção por mudança de nível nos pinos RB4 a RB7.

INTE – Habilitação de interrupção externa (pino INT).

T0IE – Habilitação da interrupção por transbordamento do TMR0 (Timer 0).

PEIE – Habilitação das interrupções periféricas.

GIE – Habilitação geral das interrupções.

PIC 16F628 - Registradores

REGISTRADORES.

PORTA – Leitura ou escrita de informações nos pinos do PIC

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit0

RA0 / AN0 – E/S digital ou entrada 0 do comparador analógico.

RA1 / AN1– E/S digital ou entrada 1 do comparador analógico.

RA2 / AN2 Vref– E/S digital ou entrada 2 do comparador analógico ou tensão de referência do comparador analógico).

RA3 / AN3 /CMP1– E/S digital ou entrada 2 do comparador analógico ou saída do comparador 1).

RA4 / T0CK1 /CMP1– E/S digital ou entrada de *clock* para o TMR0 ou saída do comparador 1).

RA5 / MCLR\ /THV– E digital ou reset ou tensão de programação(13 Volts).

RA6 / OSC2 /CLKOUT– E/S digital ou entrada osc2 ou saída de *clock* (f/4).

RA7 / OSC1 /CLKIN– E/S digital ou entrada osc1 ou entrada de *clock* ext.

PIC 16F628 - Registradores

REGISTRADORES.

TRISA – Leitura ou escrita de informações nos pinos do PIC.

0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Bit0

0 = O pino funciona como saída.

1 = O pino funciona como entrada.

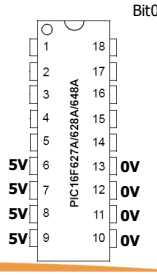
Configurando o PORTB como SAÍDA DIGITAL

TRISB = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Cada pino pode ser acessado pelo seu respectivo bit.

Por exemplo:

PORTB= | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |



PIC 16F628 - Características

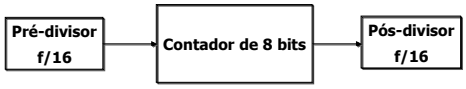
3 Timer (contadores).

0	0	0	0	0	0	0	0

até

1	1	1	1	1	1	1	1

- Contar eventos externos por um período.
- Gerar **interrupção** quando estourar sua capacidade.



PIC 16F628 - Características

Timer 0 – Contador de 8 bits – até 256

- Contagem de eventos externos (*clock* externo).
- Contagem de tempo (*clock* interno).
- Pre-divisor 1:2 até 1:256.

Timer 2 – Contador de 8 bits – até 256.

- Contagem de tempo (*clock* interno).
- Pré-divisor 1:1 até 1:16.
- Pós-divisor 1:1,1:4 e 1:16.
- Registrador de período (até quanto vai contar)

PIC 16F628 - Características

Timer 1 – Contador de 16 bits – até 65.535

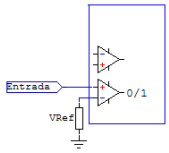
- Contagem de eventos externos (*clock* externo).
- Contagem de tempo (*clock* interno).
- Pre-divisor por 1, 2, 4, 8.

Estoura com 65.535 uSeg= 11111111 11111111

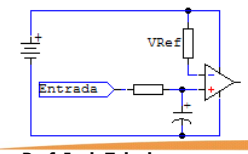
- Configurar como contagem de tempo.
- Pré-dividor 1:8 (estoura com 524.280uSeg)

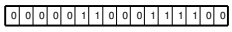
PIC 16F628 - Características

2 Módulos comparadores analógicos



Conversor analógico digital Delta-sigma

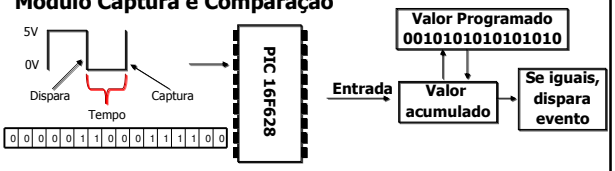




- Descarrega o capacitor
- Inicia o contador
- Aguarda nível 1 no comparador
- Captura a leitura no contador

PIC 16F628 - Características

Módulo Captura e Comparação



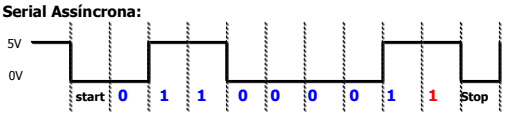
Módulo PWM (modulação por largura de pulso)

PIC 16F628 - Características

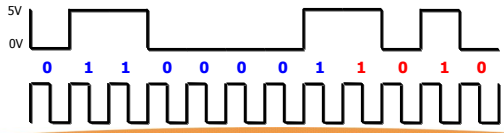
USART (Transmissão/Recepção Assíncrona ou Síncrona Universal)

Transmissão do caractere **'a'** e do **código de verificação de erro**:

Serial Assíncrona:



Serial Síncrona:



PIC 16F628 - Características

Power-Up

- Retardo de 72 ms na inicialização do oscilador, para estabilização da fonte de alimentação.

Brown-Out

- Provoca o reset da CPU se a tensão de alimentação cair abaixo de 4 Volts.

Watchdog

- Temporizador independente que provoca o reset da CPU, caso não seja executada a instrução "CLRWDT", em um determinado intervalo (18 ms até 2.304 ms).
- Esta ação reinicia o sistema em caso de travamento.

PIC 16F628 - Características

ICSP (programação serial no circuito)

- Técnica de programação no qual o chip é gravado no circuito de aplicação, utilizando três pinos do PIC(Clock,Dados e tensão de programação).

Sleep

- Modo de baixo consumo de energia (2uA).
- Retorno por três modos:
  - Reset externo (pino MCLR).
  - Termino da temporização do *Watchdog*.
  - Ocorrência de uma interrupção.

PIC 16F628 - Características

PC (Contador de programa)

- Cada linha de código tem um número, o PC é um contador que controla a linha que está sendo executada no momento.

Pilha

- Nas chamadas de rotinas (desvio), o endereço atual (PC) é armazenado na pilha, para retorno após o término da rotina.
- A pilha do PIC contem 8 níveis, e é função do programador controlar para que não sejam executadas mais que 8 desvios consecutivos.

PIC 16F628 - Características

Interrupções - Conceito.

A interrupção é um dos mais poderosos recursos do PIC.

Em alguns casos funciona como um processamento paralelo, transparente ao programador.

Quando ocorre uma interrupção a execução do código é desviada para o endereço de memória 0x04.

Neste endereço é verificado qual foi a fonte da interrupção e deve existir uma chamada para a rotina de tratamento da interrupção.

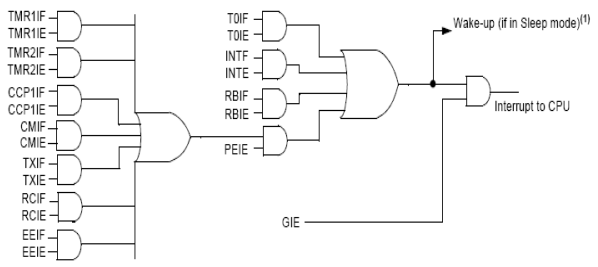
Após o tratamento, a execução retorna ao endereço original.

PIC 16F628 - Características

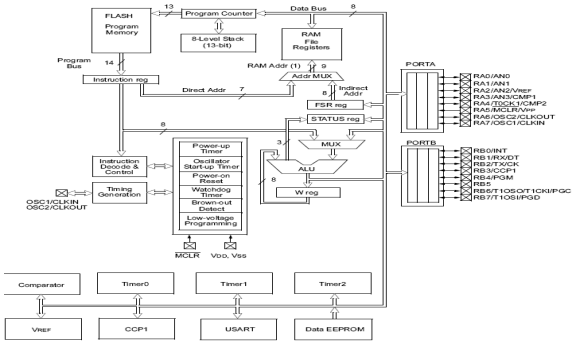
Fontes de interrupções.

- Estouro de contagem do Timer 0, Timer 1 e Timer 2.
- Mudança de nível de sinal no PORTB.
- Interrupção externa (pino INT).
- Término de escrita na EEPROM;
- Comparação/captura no módulo CPP.
- Recepção de dados na USART.
- Fim de transmissão na USART.
- Comparador analógico.

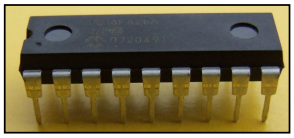
PIC 16F628 - Características



PIC 16F628 - Características



MPLAB IDE



MBLab IDE - Novo Projeto

•O MPLab utiliza o conceito de projeto, ou seja, um conjunto de informações ou arquivos relacionados a um trabalho em particular**.**

•Crie a pasta "TesteMPLAB" em um diretório no qual tenha permissão de gravação (G: ou Z:).

MBLab IDE - Novo Projeto

Método 1

Crie uma pasta no seu drive virtual

(RAxxxxxx z:)

MBLab IDE - Novo Projeto

Criar um novo projeto:

•Inicie o MPLAB e finalize qualquer projeto anterior (*File → Close Workspace*).

•*Project → Project Wizard*...

•Avançar → PIC16F628A → Avançar.

•**SE HOUVER** um "X" vermelho antes de MPASM, vá no botão "*browse*",busque o endereço do arquivo "mpasmwin.exe (C:\Arquivos de programas \Microchip\MPASM Suite\mpasmwin.exe).Repita para mplink.exe e mplibexe.→ Avançar

•**SE NÃO HOUVER .** → Avançar

MBLab IDE - Novo Projeto

No botão "*browse*" encontre a pasta "TesteMPLab" e digite um nome para o projeto (PiscaLed).

Avançar → Avançar → Concluir.

Criar um Arquivo fonte:

File → New.

File → Save As... e digite o nome PiscaLed.asm (importante " .asm").

Project → Add Files to Project... e encontre o arquivo que foi salvo.

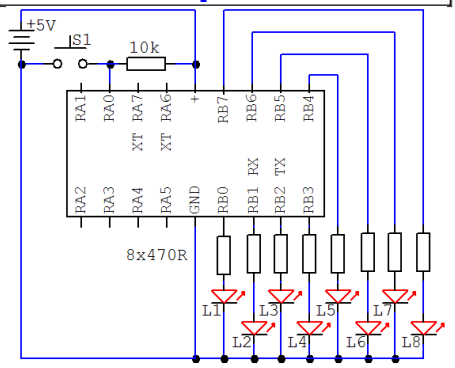
MBLab IDE - Padronização

Para manter um padrão todo o código será escrito em letras maiúsculas e utilizada a tecla "TAB" para endentação em colunas.

Qualquer texto após o " ; " é interpretado pelo compilador como comentário.

```
; SEQUENCIAL DE LEDS.
; PROGRAMA EXEMPLO PARA EXPLORAR A IDE MPLAB.
; IVAIR TEIXEIRA.
```

Pisca Led – Esquema elétrico



Pisca Led – Código

```
; *********************************************************
;                CONFIGURAÇÃO DO MICROCONTROLADOR.
; *********************************************************
  LIST P=16F628A                    ;INFORMA O PIC UTILIZADO.
  INCLUDE  <P16F628A.INC>           ;INCLUSÃO DA BIBLIOTECA.
  __CONFIG _CP_OFF  &  _WDT_OFF  &  _PWRTE_OFF  &  _BODEN_OFF
  & _MCLRE_ON  &  _INTRC_OSC_NOCLKOUT  &  _LVP_OFF
  ERRORLEVEL -302                   ;RETIRA MSG OPERAÇÃO NO BANCO 1.
```

Antes do config são 2 traços.
Todo o config em uma única linha

Pisca Led – Código

```
; *********************************************************
;                DECLARAÇÃO DE VARIÁVEIS E CONSTANTES.
; *********************************************************
  CBLOCK 0X20               ;0X20 É O INÍCIO DA MEMÓRIA RAM
     T1                     ; T1 NO ENDEREÇO 0X20
     T2                     ; T2 NO ENDEREÇO 0X21
  ENDC

  #DEFINE S1 PORTA,0        ;DEFINE QUE S1 = PORTA,0
```

S1 será substituído por PORTA,0 pelo compilador

Pisca Led – Código

```
; *********************************************************
;                VETORES DE RESET E INTERRUPÇÃO.
; *********************************************************
  ORG 0X00              ;A EXECUÇÃO INICIA NESSE ENDEREÇO.
  GOTO      INICIO      ;VAI PARA A SUB-ROTINA INÍCIO.
  ORG  0X04             ;VEM NESSE ENDEREÇO COM A INTERRUPÇÃO.
  RETFIE                ;RETORNA DA INTERRUPÇÃO.


; *********************************************************
;                ROTINAS GERAIS.
; *********************************************************
```


Pisca Led – Código

```
SEQUENCIAL
  MOVLW     B'00000001'     ;MOVE 00000001 PARA W
  MOVWF     PORTB           ;MOVE W PARA O PORTB
  CALL      TEMPO           ;CHAMA A ROTINA TEMPO
  MOVLW     B'00000010'     ;MOVE 00000010 PARA W
  MOVWF     PORTB           ; MOVE W PARA O PORTB
  CALL      TEMPO
  . . .                     ;completar o código

  MOVLW     B'10000000'
  MOVWF     PORTB
  CALL      TEMPO
  GOTO      SEQUENCIAL      ;LOOP INFINITO
```


Pisca Led – Código

```
; *********************************************************
;                ROTINAS DE TEMPORIZAÇÃO.
; *********************************************************
TEMPO
  MOVLW     .255
  MOVWF     T2              ;INICIALIZA T2 COM 255
  MOVLW     .255
  MOVWF     T1              ;INICIALIZA T1 COM 255
  DECFSZ    T1, F           ;DECREMENTA T1 E SALTA SE =0
  GOTO      $-1             ; VOLTA UMA LINHA
  DECFSZ    T2, F
  GOTO      $-5
  RETURN                    ;RETORNA DA CHAMADA CALL
```

Pisca Led – Código

```
; ********************************************************
;                  CONFIGURAÇÃO INICIAL.
; ********************************************************
INICIO
   BSF     STATUS,RP0              ;VAI PARA O BANCO 1
   MOVLW   B'00000001'
   MOVWF   TRISA                   ;PORTA,0 ENTRADA
   CLRF    TRISB                   ;PORTB TODO SAÍDA
   BCF     STATUS,RP0              ;VOLTA PARA O BANCO 0
   MOVLW   B'00000111'             ;MOVE 00000111 PARA W
   MOVWF   CMCON             ;DESABILITA OS COMP ANALÓGICOS
   CLRF    PORTA                   ;INICIALIZA PORTA COM 0
   CLRF    PORTB
   GOTO    SEQUENCIAL
   END
```


MBLab IDE - Novo Projeto

Método 2

Crie uma pasta no seu drive virtual

(RAxxxxxx z:)

Salve dentro da pasta o arquivo codbase.asm

MBLab IDE - Novo Projeto

Criar um novo projeto:

- Inicie o MPLAB e finalize qualquer projeto anterior
- *Project* → *Project Wizard*...
- (*Welcome*) → Avançar
- (*Device*) →PIC16F628A → Avançar.
- (*Select Language* – não mude nada) →Avançar
- (*Create new project file*) → BROWSE, encontre a pasta do projeto e coloque o nome do projeto. → SALVAR → AVANÇAR.

MBLab IDE - Novo Projeto

- (*Add existing files*...) Selecione o arquivo codbase.asm e adicione ao projeto (ADD>>)
- Avançar → Concluir.
- Abra o código fonte.

Simulação – Saídas

***Watch*:**

- ***Wiew → Watch.***
- **Nessa janela é possível inserir e visualizar o valor de todos os registradores (*Add SFR*) ou variáveis (*Add Symbol*) do PIC.**
- ***Click* com o botão direito sobre o registrador e em "*properties...*" escolha o formato da exibição.**

Para compilar: F10

Para simular: Debugger →Select tools → MPLab SIM

F6=reset / F8=passo-a-passo

Simulação – Entradas

***Stimulus*:**

- ***Debugger → Stimulus → New Workbook → Asynch***
- **Em "Pin / SFR" escolha o "PORT" desejado, e em "*Action*" escolha "*Toggle*" (inverter).**
- **Para inverter o nível digital (0 ou 1) pressione o botão "*Fire*".**

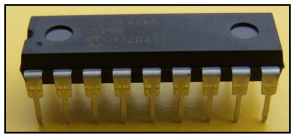
- **Obs: F10 → F6 → F8 → F8... até passar pela inicialização do PORTA, após isso "*click*" em "*Fire*", selecione novamente a janela de código e pressione novamente F8 para atualizar a janela *Watch.***

Simulação – Tempo

StopWatch **:**

- ***Debugger → StopWatch*****.**
- ***Debugger → Settings → Processor Frequency*** **= 4.**
- **Nessa janela é possível visualizar o tempo de execução de uma determinada rotina.**
- **Colocar um *breakpoint* (2 clicks) na primeira instrução e outro na última instrução da rotina.**
- **Pressionar F9 para o código ser executado até o primeiro *breakpoint*, zerar o *StopWatch* e pressionar novamente F9.**

Conjunto de Instruções



PIC 16F628 - Instruções

Estrutura do código

- **Inclusão de bibliotecas.**

  INCLUDE "P16F628.inc"

  - Faz com que o compilador reconheça um conjunto de termos visando facilitar a programação.
  - O que é mais fácil de lembrar: PORTA ou H'0005' ?

- **Configuração de gravação (palavra de configuração).**

  __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_OFF & _BODEN_OFF & _MCLRE_ON & _INTRC_OSC_NOCLKOUT & _LVP_OFF

  - Define, no momento da gravação do PIC, o modo de funcionamento de vários periféricos.
  - Ver tabela – palavra de configuração.

PIC 16F628 - Instruções

Estrutura do código

- **Definições das constantes.**

  ```
  #DEFINE   BTN1    PORTA, 0
  ```

  - Nossas definições para facilitar a programação.

- **Declaração das variáveis.**

  ```
  CBLOCK 0X20
     T1
     T2
  ENDC
  ```

  - Reserva o endereço de memória 0x20 para T1 e 0x21 para T2.
  - Em C é o mesmo que: char T1,T2;

PIC 16F628 - Instruções

Estrutura do código

- **Vetor de reset.**

  ```
  ORG 0x00
  ```

  - Define o local onde o código inicia quando o PIC e alimentado ou quando acontece um *reset*
  - Após esta linha deve haver uma chamada para a primeira rotina do programa.

- **Vetor de interrupções.**

  ```
  ORG 0x04
  ```

  - Define o local para onde o código é desviado quando acontece uma interrupção.
  - Após esta linha deve haver a rotina, ou chamada, para tratamento da interrupção e no final o comando RETFIE.

PIC 16F628 - Instruções

Estrutura do código

```
PRINCIPAL
   CALL     TEMPO      ;CHAMA A ROTINA TEMPO
   BTFSC    PORTA, 1   ;SE O BOTÃO FOR 0 PULA PRÓXIMA
   GOTO     DESLIGA    ;VAI PARA LIGA
   GOTO     LIGA       ;VAI PARA DESLIGA
LIGA
   MOVLW    B'11111111'  ;MOVE 11111111 PARA W
   MOVWF    PORTB      ;MOVE W PARA O PORTB
   GOTO     PRINCIPAL  ;VOLTA PARA A ROTINA PRINCIPAL
DESLIGA
   CLRF     PORTB       ;LIMPA (ZERO) O PORTB
   GOTO     PRINCIPAL
TEMPO
   ...
   RETURN               ;RETORNA APÓS A CHAMADA
```

PIC 16F628 - Instruções

Conjunto de Instruções do PIC 16.

Sintaxe das instruções.

W = (*Work*), Acumulador ou registrador de trabalho

•**CLRW**

f = (*File Register*) Qualquer registrador, geral ou especial.

•**MOVWF PORTB** **(MOVWF f)**

d = (*Destination*) Destino do resultado da operação (W=0 ou F=1)

•**INCF CONT, F** **(INCF f,d)**

k = Literal, número, constante, nome de uma sub-rotina.

•**CALL TESTA_PINO** **(CALL k)**

b = 1 dos 8 bits do registrador.

•**BCF PORTA,5** **(BCF f,b)**

PIC 16F628 - Instruções

Conjunto de Instruções do PIC 16.

O conjunto de instruções do PIC utiliza MNEMÔNICOS construídos com partes de palavras, visando facilitar a programação.

Por exemplo:

MOVLW .140 - ***MOV****e* ***L****iteral of* ***W*** (*Work*). – Move o valor para W.

DECF TOTAL, W - ***DEC****rement* ***F****ile* – decremente o registrador (variável) TOTAL e armazene o resultado em W.

SUBWF TOTAL, F -***SUB****tract* ***W*** *of the* ***F****ile* – Subtrai W do registrador e armazene no próprio registador.

BTFSS PORTA, 5 -***B****it* ***Te****st of* ***F****ile and* ***S****kip if* ***S****et* – Teste o bit B do registrador F e pule se for 1.

W=Work, F=File, B=bit, Z=Zero, C=Clear, S=Set ou Skip.

PIC 16F628 - Instruções

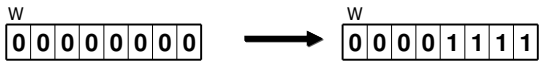
Manipulação de registradores.

MOVLW k

- •Copia o valor da constante k para o acumulador W (sobrescreve).
- •A constante (número) K é um valor entre 0 e 255.
- •Exemplo: Suponha W=0

MOVLW B'00001111'

- •Copia o valor binário 00001111 (.15 ou 0xF) para o acumulador W.
- •W=15



PIC 16F628 - Instruções

Manipulação de registradores.

MOVWF f

- •Copia o conteúdo do acumulador W para o registrador f.
- •O operando f pode ser qualquer registrador geral ou especial
- •Exemplo: Suponha W=15 e PORTB=0

MOVWF PORTB

- •Copia o conteúdo acumulado em W para o registrador PORTB.
- •PORTB = W;

PIC 16F628 - Instruções

Manipulação de registradores.

MOVF f,d

- •Copia o conteúdo do registrador f para o destino especificado por d.
- •d pode ser W ou f (o próprio registrador – comparação se é 0)
- •Exemplo: Suponha PORTB=9 W=0

MOVF PORTB, W

- •Copia o valor presente no PORTB para o acumulador W.
- •W=PORTB;

PIC 16F628 - Instruções

Manipulação de registradores.

CLRW

- •Apaga o conteúdo do acumulador W.
- •Exemplo: Suponha W=67

CLRW

- •Armazena zeros no acumulador.
- •W=0;

PIC 16F628 - Instruções

Manipulação de registradores.

CLRF f

- Apaga o conteúdo do registrador especificado pelo operando f.
- Exemplo: Suponha PORTB=3

  CLRF PORTB

  - Armazena zeros no registrador PORTB.
  - PORTB = 0;

PORTB 0 0 0 0 0 0 1 1 → PORTB 0 0 0 0 0 0 0 0

PIC 16F628 - Instruções

Manipulação de registradores.

INCF f, d

- Soma 1 ao conteúdo do registrador f e armazena o resultado no destino, especificado pelo operando d.
- Exemplo: Suponha NUM = 8 W=0

  INCF NUM, F **OU** **INCF NUM, W**

  - O valor armazenado na variável num é incrementado salvo na própria variável ou em W de acordo com o destino d.

NUM 0 0 0 0 1 0 0 0 → NUM 0 0 0 0 1 0 0 1

OU

NUM 0 0 0 0 1 0 0 0 → W 0 0 0 0 1 0 0 1

PIC 16F628 - Instruções

Manipulação de registradores.

DECF f, d

- Diminui 1 ao conteúdo do registrador f e armazena o resultado no destino, especificado pelo operando d.
- Exemplo: Suponha NUM = 8 W=0

  DECF NUM, F **OU** **DECF NUM, W**

  - O valor armazenado na variável num é incrementado salvo na própria variável ou em W de acordo com o destino d.

NUM 0 0 0 0 1 0 0 0 → NUM 0 0 0 0 0 1 1 1

OU

NUM 0 0 0 0 1 0 0 0 → W 0 0 0 0 0 1 1 1

PIC 16F628 - Instruções

Manipulação de registradores.

BSF f, b

- Insere nível 1, liga (*set*) o bit representado pelo operando b, no registrador representado pelo operando f.
- Exemplo:

  BSF PORTB, 5

  - O bit 5 do PORTB recebe o valor 1.

PORTB

X	X	1	X	X	X	X	X

BSF INTCON, GIE

PIC 16F628 - Instruções

Manipulação de registradores.

BCF f, b

- Insere nível 0, desliga (*clear*) o bit representado pelo operando b, no registrador representado pelo operando f.
- Exemplo:

  BCF PORTB, 5

  - O bit 5 do PORTB recebe o valor 0.

- PORTB

X	X	0	X	X	X	X	X

PIC 16F628 - Instruções

Manipulação de registradores.

SWAPF f, d

- Troca os *nibbles* (meio byte) do registrador representado pelo operando f e armazena o resultado no destino, especificado pelo operando d.
- Exemplo:

  SWAPF PORTB, F **OU** **SWAPF PORTB,W**

  - Inverte os *nibles* do registrador PORTB
  - Inverte os *nibles* do registrador PORTB e salva o resultado em W

PORTB 1 1 0 0 1 1 1 1 → PORTB 1 1 1 1 1 1 0 0

PORTB 0 0 0 0 1 1 1 1 → W 1 1 1 1 0 0 0 0

PIC 16F628 - Instruções

Operações de desvio.

GOTO k

- Desvio incondicional do programa para uma sub-rotina especificada por k, sem retorno.
- Exemplo:

```
PRINCIPAL
    GOTO   TESTE2
    GOTO   INICIO
TESTE1
    GOTO   PRINCIPAL
TESTE2
    GOTO   TESTE1
```


PIC 16F628 - Instruções

Operações de desvio.

CALL k

- Chamada de uma sub-rotina especificada por k. O valor PC + 1 é armazenado na pilha para o retorno após o termino da sub-rotina.
- Exemplo:

```
PRINCIPAL
    CALL   TESTE2
    GOTO   TESTE1
TESTE1
    GOTO   PRINCIPAL
TESTE2
    RETURN
```


PIC 16F628 - Instruções

Operações de desvio.

RETURN

- Retorno de sub-rotina, desempilha o ultimo endereço da pilha e retorna ao ponto de chamada.
- Exemplo:

```
PRINCIPAL
    CALL   TESTE2
    GOTO   TESTE1
TESTE1
    GOTO   PRINCIPAL
TESTE2
    RETURN
```


PIC 16F628 - Instruções

Operações de desvio.

RETLW k

- Retorno de sub-rotina, com o valor especificado pela literal k carregada no acumulador W.
- Exemplo: Suponha W=3

```
PRINCIPAL                CONVERTE
    CALL    CONVERTE         ADDWF   PCL,F
    MOVWF   AUX              RETLW   .0
    . . .                    RETLW   .5
                             RETLW   .10
AUX = 15                     RETLW   .15
                             RETLW   .20
                             RETLW   .25
                             . . .
```


PIC 16F628 - Instruções

Operações de desvio.

RETFIE

- Retorno de uma interrupção. Liga o bit INTCON, GIE e retorna ao ponto onde ocorreu a interrupção
- Exemplo:

```
TRATA_INT
    . . .
    . . .
    RETFIE
```


PIC 16F628 - Instruções

Operações aritméticas.

ADDLW k

- Adição do valor representado pelo operando k ao conteúdo do acumulador W
- Exemplo: Suponha W=4

```
ADDLW .32
```

- O valor decimal 32 é adicionado ao conteúdo de W.

Literal	0	0	1	0	0	0	0	0
+ W	0	0	0	0	0	1	0	0
=								
W	0	0	1	0	0	1	0	0

PIC 16F628 - Instruções

Operações aritméticas.

ADDWF f, d

- Adição do conteúdo do acumulador W ao registrador representado pelo operando f, o resultado é armazenado no destino indicado pelo operando d.
- Exemplo: Suponha W=10 e NUM=8

ADDWF NUM, F **OU** **ADDWF NUM, W**

- O conteúdo de W é somado ao conteúdo da variável NUM, e o resultado é armazenado na própria variável OU em W.

NUM	0	0	0	0	1	0	1	0	NUM	0	0	0	0	1	0	1	0
+ W	0	0	0	0	1	0	0	0	+ W	0	0	0	0	1	0	0	0
=									=								
NUM	0	0	0	1	0	0	1	0	W	0	0	0	1	0	0	1	0

PIC 16F628 - Instruções

Operações aritméticas.

SUBLW k

- Subtrai do conteúdo valor representado pelo operando k o conteúdo do acumulador W (k – W)
- Exemplo: Suponha W = 4

SUBLW .16

- Subtrai o conteúdo de W do valor decimal 16.

Literal	0	0	0	1	0	0	0	0
- W	0	0	0	0	0	1	0	0
=								
W	0	0	0	0	1	1	0	0

PIC 16F628 - Instruções

Operações aritméticas.

SUBWF f, d

- Subtração do conteúdo do registrador f o conteúdo do acumulador W, o resultado (f - W) é armazenado no destino d.
- Exemplo: Suponha NUM=2 e W=10

SUBWF NUM, F **OU** **SUBWF NUM, W**

- O conteúdo de NUM é subtraído do conteúdo W, e o resultado é armazenado na própria variável **OU** em W.

NUM	0	0	0	0	0	0	1	0	NUM	0	0	0	0	0	0	1	0
- W	0	0	0	0	1	0	1	0	- W	0	0	0	0	1	0	1	0
=									=								
NUM	1	1	1	1	1	0	0	0	W	1	1	1	1	1	0	0	0

PIC 16F628 - Instruções

Operações lógicas.

ANDLW k

- AND lógico bit a bit da constante representada por k com o conteúdo do registrador W.
- Exemplo: Suponha W=170

ANDLW B'00001111'

10101010
00001111
00001010

- E lógico do conteúdo de W, com o binário 00001111

0 e 0 = 0
0 e 1 = 0
1 e 0 = 0
1 e 1 = 1

PIC 16F628 - Instruções

Operações lógicas.

ANDWF f, d

- AND lógico do conteúdo do registrador W com o conteúdo do registrador especificado por f. O resultado é armazenado em d.
- Exemplo: Suponha W=170 num = 15

ANDWF NUM, F

- E lógico do conteúdo de W, com valor de NUM, com o resultado armazenado em NUM

NUM 10101010
W 00001111
NUM 00001010

PIC 16F628 - Instruções

Operações lógicas.

IORLW k

- OR lógico da constante representada por k com o conteúdo do registrador W.
- Exemplo: Suponha W=170

IORLW B'00001111'

10101010
00001111
10101111

- OU lógico do conteúdo de W, com o binário 00001111

0 ou 0 = 0
0 ou 1 = 1
1 ou 0 = 1
1 ou 1 = 1

PIC 16F628 - Instruções

Operações lógicas.

IORWF f, d

- OR lógico do conteúdo do registrador W com o conteúdo do registrador especificado por f. O resultado é armazenado em d.
- Exemplo: Suponha W=170 num = 15

  IORWF NUM, F

  - OU lógico do conteúdo de W, com valor de NUM

NUM 10101010
W 00001111
NUM 10101111

PIC 16F628 - Instruções

Operações lógicas.

XORLW k

- XOR lógico bit a bit da constante representada por k com o conteúdo do registrador W.
- Exemplo: Suponha W=170

  XORLW B'00001111' **10101010**
  00001111
  10100101

  - OU exclusivo lógico do conteúdo de W, com o binário 00001111.

0 xor 0 = 0
0 xor 1 = 1
1 xor 0 = 1
1 xor 1 = 0

PIC 16F628 - Instruções

Operações lógicas.

XORWF f, d

- XOR lógico do conteúdo do registrador W com o conteúdo do registrador especificado por f. O resultado é armazenado em d.
- Exemplo: Suponha W=170 num = 15

  XORWF NUM, F

  - OU exclusivo lógico do conteúdo de W, com valor de NUM

NUM 10101010
W 00001111
NUM 10100101

PIC 16F628 - Instruções

Operações lógicas.

COMF f, d

- Complementa (inverte), o nível lógico dos bits do registrador especificado por f.
- Exemplo:

  COMF NUM, F **10101010**
  01010101

  - Inversão bit a bit do conteúdo de NUM

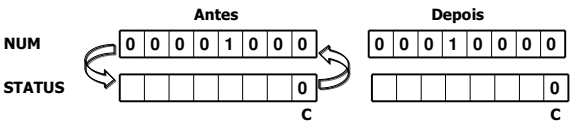
PIC 16F628 - Instruções

Operações lógicas.

RLF f, d

- O conteúdo do registrador especificado por f é deslocado um bit a esquerda. O valor do *flag* STATUS, C é armazenado no bit 0 do registrador e o bit excedente é armazenado no *flag*.
- Exemplo:

  RLF NUM, F



Aula 10/03 até aqui

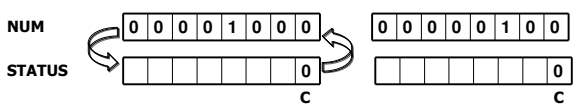
PIC 16F628 - Instruções

Operações lógicas.

RRF f, d

- O conteúdo do registrador especificado por f é deslocado um bit a direita. O valor do *flag* STATUS, C é armazenado no bit 7 do registrador e o bit excedente é armazenado no *flag*.
- Exemplo:

  RRF NUM, F



Aula 10/03 até aqui

PIC 16F628 - Instruções

Operações de desvio.

BTFSS f,b

- Desvio condicional. Testa o bit especificado por b, do registrador f e pula a próxima instrução se o bit estiver em nível 1

```
PRINCIPAL                TESTE1
    BTFSS NUM, 3             GOTO    PRINCIPAL
    GOTO  TESTE1
    GOTO  TESTE2         TESTE2
                             GOTO    PRINCIPAL
```

- Testa o bit 3 do registrador NUM e pula a próxima linha se for **1**
- Se o bit for igual a **1** pula a próxima linha e vai para a rotina TESTE2, senão, não pula a próxima linha e vai para a rotina TESTE1

PIC 16F628 - Instruções

Operações de desvio.

BTFSC f,b

- Desvio condicional. Testa o bit especificado por b, do registrador f e pula a próxima instrução se o bit estiver em nível 0

```
PRINCIPAL                TESTE1
    BTFSC NUM, 3             GOTO    PRINCIPAL
    GOTO  TESTE1
    GOTO  TESTE2         TESTE2
                             GOTO    PRINCIPAL
```

- Testa o bit 3 do registrador NUM e pula a próxima linha se for **0**
- Se o bit for igual a **0** pula a próxima linha e vai para a rotina TESTE2, senão, não pula a próxima linha e vai para a rotina TESTE1

PIC 16F628 - Instruções

Operações de desvio.

DECFSZ f, d

- Desvio condicional. Decrementa o conteúdo do registrador f, e pula a próxima instrução se o resultado for igual a 0. O resultado e armazenado no destino indicado por d.

```
PRINCIPAL                TESTE1
    DECFSZ  NUM, F           GOTO    PRINCIPAL
    GOTO    TESTE1
    GOTO    TESTE2       TESTE2
                             GOTO    PRINCIPAL
```

Subtrai 1 do valor da variável NUM, se o resultado for igual a 0 pula a próxima instrução e vai para a rotina TESTE2. Senão vai para a rotina TESTE1.

PIC 16F628 - Instruções

Operações de desvio.

INCFSZ f,d

- Desvio condicional. Incrementa o conteúdo do registrador f, e pula a próxima instrução se o resultado for igual a 0. O resultado e armazenado no destino indicado por d.

```
PRINCIPAL                TESTE1
    INCFSZ  NUM, F           GOTO    PRINCIPAL
    GOTO    TESTE1
    GOTO    TESTE2       TESTE2
                             GOTO    PRINCIPAL
```

Adiciona 1 ao valor da variável NUM, se o resultado for igual a 0 pula a próxima instrução e vai para a rotina TESTE2. Senão vai para a rotina TESTE1.

PIC 16F628 - Instruções

Controle.

NOP

- Não executa nenhuma operação. Gasta um ciclo de máquina.

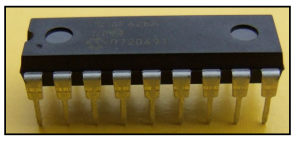
CLRWDT

- Limpa o contador *watchdog*. Quando o *watchdog* estiver habilitado o programa deve executar esta instrução periodicamente para que não ocorra o reset da CPU.

SLEEP

- Coloca a CPU no modo de economia de energia.

Avançado



Usar Tabelas

Cada comando do programa está em uma linha e tem um número sequencial. O registrador PCL armazena a linha em execução, e é incrementado no fim de cada linha. Se somar um valor ao PCL realiza-se um "salto"para outra linha.

```
100 MULTIPLICA
101   ADDWF PCL,F    ;W(?) +PCL (esta é a linha 101)  + 1
102   RETLW .0       ;Cai aqui se W=0 → 0 + 101 + 1 = 102
103   RETLW .3       ;Cai aqui se W=1 → 1 + 101 + 1 = 103
104   RETLW .6       ;Cai aqui se W=2 → 2 + 101 + 1 = 104
105   RETLW .9       ;Cai aqui se W=3 → 3 + 101 + 1 = 105
106   RETLW .12      ;Cai aqui se W=4 → 4 + 101 + 1 = 106
107   RETLW .15      ;Cai aqui se W=5 → 5 + 101 + 1 = 107
108   RETLW .18      ;Cai aqui se W=6 → 6 + 101 + 1 = 108
109   RETLW .21      ;Cai aqui se W=7 → 7 + 101 + 1 = 109
110   . . .
```


Usar Tabelas

Se o PCL = 101 e somar 5 a ele, seu novo valor será 106, assim, quando for incrementado ocorrerá um salto para a linha 107. A instrução **RETLW** retorna da rotina com a literal em W. Esse exemplo faz a multiplicação de W por 3.
Sempre no fim da a instrução ocorre o incremento (PCL+1)

```
100 MULTIPLICA
101   ADDWF  PCL,F    ;W(?) +PCL (esta é a linha 101)  + 1
102   RETLW  .0       ;Cai aqui se W=0 → 0 + 101 + 1 = 102
103   RETLW  .3       ;Cai aqui se W=1 → 1 + 101 + 1 = 103
104   RETLW  .6       ;Cai aqui se W=2 → 2 + 101 + 1 = 104
105   RETLW  .9       ;Cai aqui se W=3 → 3 + 101 + 1 = 105
106   RETLW  .12      ;Cai aqui se W=4 → 4 + 101 + 1 = 106
107   RETLW  .15      ;Cai aqui se W=5 → 5 + 101 + 1 = 107
108   RETLW  .18      ;Cai aqui se W=6 → 6 + 101 + 1 = 108
109   RETLW  .21      ;Cai aqui se W=7 → 7 + 101 + 1 = 109
110   . . .
```


Endereçamento Indireto

FSR = Armazena o endereço de memória
INDF = Acessa o endereço de FSR
Exemplo:

```
•MOVLW  0X25
•MOVWF  FSR
•MOVLW  .100
•MOVWF  INDF
•INCF   FSR
•MOVLW  .200
•MOVWF  INDF
• . . .
```

Endereço	Conteúdo
0X20	
0X21	
0X22	
0X23	
0X24	
0X25	100
0X26	200
0X27	
0X28	

Endereçamento Indireto

A cada chamada a rotina SALVA armazena sequencialmente o valor de W em uma posição da memória.

```
PRINCIPAL
   . . .
SALVA
   INCF    FSR,F
   MOVWF   INDF
   RETURN
INÍCIO
   . . .
   MOVLW   0X30
   MOVWF   FSR
   GOTO    PRINCIPAL
```

FSR →
FSR ⋯→

Endereço	Conteúdo
0X30	
0X31	
0X32	
0X33	
0X34	
0X35	
0X36	
0X37	
0X38	

Usar Flags de controle

•Declarar a variável e inicializá-la (**BCF** AUX, 0)
•Interrupção do TMR1 a cada 2 minutos.

```
TRATA_INT
   BSF     AUX, 0
   RETFIE
TESTA_FLAG
   BTFSS   AUX, 0
   GOTO    TESTA_FLAG
   GOTO    SEQUENCIAL
```

```
SEQUENCIAL
        MOVLW   B'00000001'
        MOVWF   PORTB
        CALL    TEMPO
        MOVLW   B'00000010'
        MOVWF   PORTB
        CALL    TEMPO
        . . .   . . .
        BCF     AUX,0
        GOTO    TESTA_FLAG
```


Interrupções

```
ORG  0X00
GOTO        INICIO
ORG  0X04
GOTO        TRATA_INT
. . .
PRINCIPAL
   MOVLW    B'00000000'
   . . .
   GOTO     PRINCIPAL
TRATA_INT
   . . .
   RETFIE
INICIO
   BSF      STATUS,RP0
   . . .
   GOTO     PRINCIPAL
```

Quando ocorre uma interrupção, o fluxo é direcionado para o endereço 0X04 no qual deve haver um desvio para a rotina de tratamento de interrupção.

Próximo Slide.

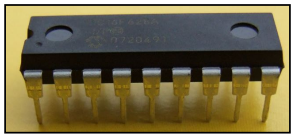
Interrupções

Após o tratamento da interrupção a instrução **RETFIE** realiza o retorno para a linha posterior a ocorrência da interrupção

```
TRATA_INT
   MOVWF   W_TEMP      ;SALVA O CONTEÚDO DE W
   . . .               ;EXECUTA AS AÇÕES DESEJADAS...
   . . .               ;...POR EXEMPLO TESTAR UM BOTÃO...
   . . .               ;...MOSTRAR UM VALOR NO DISPLAY
   MOVF    W_TEMP,W    ;RESTAURA O CONTEÚDO DE W
   RETFIE              ;RETORNA DA INTERRUÇÃO
```

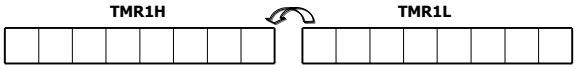
Obs: Criar a variável W_TEMP se for necessário salvar W

TIMER 1



Interrupções – Timer 1

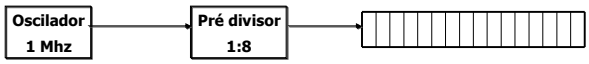
O Timer 1 (TMR1) é um contador de 16 bits capaz de contar até 65535 (11111111 11111111)



Gera interrupção por "estouro" da capacidade.
Temporizador quando conectado ao oscilador
Contador de eventos quando conectado ao pino 12
Tem um pré divisor configurável (1:1,1:2,1:4 ou 1:8)
Pode ser ligado ou desligado em tempo de execução

Interrupções – Timer 1

TIMER 1 = CONTADOR DE 16 BITS = 65.535



1 pulso a cada 0,000001 seg. ou 1 microseg.

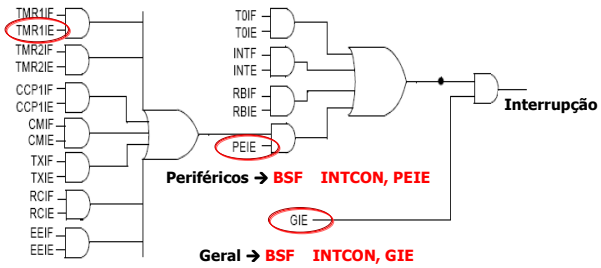
A cada 8 pulsos sai 1

Estoura a cada 65.535 x 8 = 524.280 pouco mais de 0,5 seg.

Interrupções – Timer 1

"Chaves" que devem ser habilitadas

Timer 1 → BSF PIE1, TMR1IE



Periféricos → BSF INTCON, PEIE

Geral → BSF INTCON, GIE

Interrupções – Timer 1

INTCON– Registrador de controle de interrupções.

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

PEIE – Habilitação das interrupções periféricas.

GIE – Habilitação geral das interrupções.

BSF INTCON, PEIE
BSF INTCON, GIE

PIE1– Registrador de habilitação de interrupções periféricas

----	----	----	-----	-----	-----	-----	TMR1IE

TMR1IE = 1 Interrupção do Timer 1 habilitada

BSF PIE1, TMR1IE

Interrupções – Timer 1

T1CON– Registrador de controle do Timer 1.

---	---	T1CKPS1	T1CKPS0	---	---	---	TMR1ON

Bit0

TMR1ON – Liga / desliga o Timer 1

T1CKPS1 →

T1CKPS0 →

T1CKPS1	T1CKPS0	Divisor
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

BSF T1CON, T1CKPS0
BSF T1CON, T1CKPS1
BSF T1CON, TMR1ON

8 X 65.535 = 524.280

Interrupções – Timer 1

Para um "ajuste fino" os registradores do Timer 1 podem ser inicializados com valores que alteram a quantidade de ciclos necessária para ocorrer o estouro, por exemplo:

Para gerar um tempo de 500.000 ms. (0,5 seg.).

Cada ciclo de máquina 0,000001 seg.

Estoura com: 524.280 – 500.000 = (-)24.280 ciclos de diferença

24280 / 8 = 3.035 (a diferença também vai ser dividida pelo prescaler) os registradores devem ser inicializados com:

3.035 = 00001011 11011011

TMR1H **TMR1L**

0	0	0	0	1	0	1	1	1	1	0	1	1	0	1	1

Interrupções – Timer 1

Para que o Timer 1 inicie a contagem deve ser previamente ligado (BSF T1CON, TMR1ON)

T1CON– Registrador de controle do Timer 1.

----	----	----	-----	-----	-----	-----	TMR1ON

TMR1ON – Liga o Timer 1 Bit0

Quando ocorre uma interrupção o bit PIR1, TMR1IF assume nível 1, e **deve ser apagado** (BSF PIR1, TMR1IF) após o tratamento da interrupção.

PIR1– Registrador de interrupções periféricas

----	----	----	-----	-----	-----	-----	TMR1IF

TMR1IF = 1 Ocorreu o transbordamento do TIMER1 Bit0

Interrupções – Timer 1

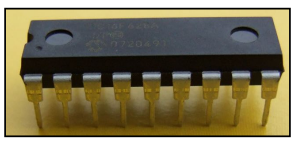
```
ORG  0X04
   GOTO      TRATA_INT      ;VAI PARA A ROTINA DE TRATAMENTO
...                         ;DE INTERRUPÇÃO
TRATA_INT
   BCF       PIR1, TMR1IF   ;LIMPA O FLAG DE SINALIZAÇÃO
   MOVWF     W_TEMP         ;SALVA O CONTEÚDO DE W
   ...                      ;EXECUTA AS AÇÕES DESEJADAS
   ...                      ;POR EXEMPLO TESTAR UM BOTÃO
   ...                      ;MOSTRAR UM VALOR NO DISPLAY
   MOVF      W_TEMP,W       ;RESTAURA O CONTEÚDO DE W
   RETFIE                   ;RETORNA DA INTERRUÇÃO
```

Obs: Criar a variável W_TEMP se for necessário salvar W

Interrupções – Timer 1

```
INICIO
   BSF       STATUS, RP0    ;MUDA PARA O BANCO 1
   BSF       PIE1, TMR1IE   ;HABILITA A INTERRUPÇÃO DO TIMER 1
   CLRF      PORTA          ;PORTA TODO SAÍDA
   CLRF      PORTB          ;PORTB TODO SAÍDA
   BCF       STATUS, RP0    ;MUDA PARA O BANCO 0
   MOVLW     B'00000111'    ;MOVE 00000111 PARA W
   MOVWF     CMCON          ;DESABILITA OS COMP. ANALÓGICOS
   BSF       T1CON, T1CKPS1 ;PRÉ DIVISOR COM...
   BSF       T1CON, T1CKPS0 ;...FATOR DE 1:8
   BSF       T1CON, TMR1ON  ;TIMER 1 LIGADO
   BSF       INTCON, PEIE   ;HABILITA INT. DOS PERIFÉRICOS
   BSF       INTCON, GIE    ;HABILITA A CHAVE GERAL DE INT.
   ... (CLRF PORTA, CLRF PORTB, GOTO PRINCIPAL)
```


Comunicação Serial



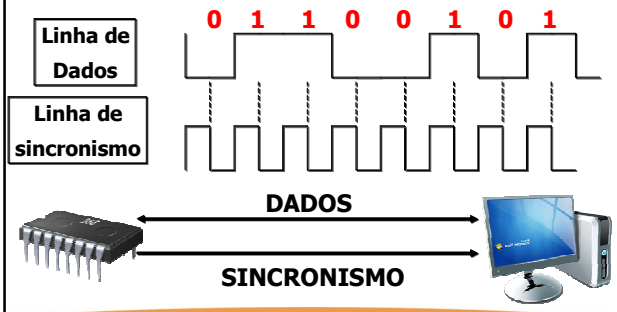
Conversão ASCII → Decimal

ASCII	Binario	Decimal	ASCII - 48
0	00110000	48	00000000
1	00110001	49	00000001
2	00110010	50	00000010
3	00110011	51	00000011
4	00110100	52	00000100
5	00110101	53	00000101
6	00110110	54	0000110
7	00110111	55	00000111
8	00111000	56	00001000
9	00111001	57	00001001

Comunicação serial

SÍNCRONA

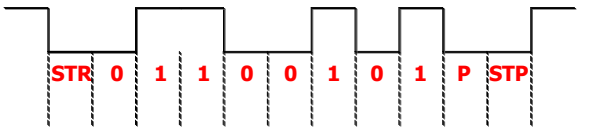
Caractere 'e' = 01100101



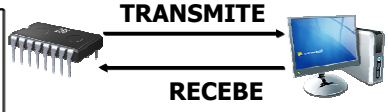
Comunicação serial

ASSÍNCRONA

Caractere 'e' = 01100101



O StartBit dispara um temporizador no receptor Baud Rate



Comunicação serial

Baud Rate = Velocidades de comunicação serial

300 – 600 – 1.200 – 2.400 – 4.800 – **9.600** – 19.200
38.400 – 76.800 – 96.000 – 115.200 – 230.400 –
300.000 – 500.000 **(bps)**

BRGH = 0 - Modo baixa velocidade:
Baud Rate = Fosc/[64(**x**+1)]

BRGH = 1 - Modo alta velocidade:
Baud Rate = Fosc/[16(**x**+1)]

X = valor no registrador SPBRG

Comunicação serial

Configurar BRGH e SPBRG para 9600 bps, no modo que apresentar o menor erro.

Clock 4 Mhz - **BRGH=0** (Modo de baixa velocidade)			
Velocidade	Vel. Real	Erro	SPBRG
600	613	1.13	102
1.200	1.201	0,16	51
9.600	8.928	-6,99	6
19.200	19.800	8,51	2

Erro = ((8928-9600)/9600)*100 = -6.99

Comunicação serial

Configurar BRGH e SPBRG para 9600 bps, no modo que apresentar o menor erro.

Clock 4 Mhz - **BRGH=1** (Modo de alta velocidade)			
Velocidade	Vel. Real	Erro	SPBRG
600	---	---	---
1.200	1.201	0,16	207
9.600	9.615	0,16	25
230.400	250.000	8,51	0
300.000	---	---	---

Erro = ((9615-9600)/9600)*100 = 0,16

Comunicação serial

Banco 1

TXSTA– Registrador de controle da transmissão serial.

---	---	TXEN	SYNC	---	BRGH	TRMT	---

TXEN – Habilita a transmissão serial

SYNC – Assíncrono = 0

Síncrono = 1

BRGH – Baixa velocidade = 0

Alta velocidade = 1

TRMT – Registrador de transmissão cheio = 0

Registrador de transmissão vazio = 1

Comunicação serial

Banco 1

SPBRG– Registrador do gerador de Baud Rate

0	0	0	1	1	0	0	1

25 = 00011001

PIE1– Registrador de habilitação de interrupção com. serial

----	----	RCIE	TXIE	-----	-----	-----	-----

RCIE = 1 Int. de recepção de caractere pela serial habilitada*

TXIE = 1 Int. de transmissão de caractere pela serial habilitada*

TRISB– Configuração de entrada e/ou saída do PORTB

----	----	----	-----	-----	-----	RX	-----

RX = 1 - PORTB, 1 (**Entrada** da comunicação serial)

Comunicação serial

Banco 0

RCSTA– Registrador de controle da recepção serial.

SPEN	---	---	CREN	---	---	---	---

SPEN = 1 Comunicação serial habilitada

CREN = 1 Recepção contínua habilitada

INTCON– Registrador de controle de interrupções

GIE	PEIE	-----	-----	-----	-----	-----	-----

GIE = 1 Habilitação geral das interrupções.

PEIE = 1 Habilitação das interrupções periféricas.

Comunicação serial

Banco 0

PIR1– Registrador de interrupções periféricas

----	----	RCIF	TXIF	-----	-----	-----	---

RCIF = 1 Caractere recebido pela comunicação serial

TXIF = 1 Fim da transmissão do caractere pela serial

RCIF = Flag que sinaliza quando terminou de receber um caractere pela porta serial (entrada de dados).

TXIF = Flag que sinaliza quando acabou a transmissão de um caractere pela serial (saída de dados)

Comunicação serial

Recepção Serial - Comandos

- Copiar para W o valor do registrador **RCREG** que contém o byte recebido pela serial.

```
•MOVF    RCREG, W    ;Salva em w
         OU
•MOVF    RCREG, W    ;Salva em w
•MOVWF   DADO_RX     ;Salva em variável
```


Comunicação serial

Recepção Serial – Exemplo sem interrupção

```
PRINCIPAL
    BTFSC    PIR1, RCIF    ;se NÃO chegou caractere, pula
    CALL     RECEBE        ;ou, chama a rotina recebe
    . . .                  ;códigos gerais
    . . .

RECEBE
    MOVF     RCREG, W      ;copia o byte recebido em W.
    MOWF     AUX           ;copia W para uma variável.
    RETURN                 ;retorna na linha após a chamada
```

Obs: Somente detecta/recebe o caractere ao passar pelo ponto de teste na rotina principal.

Comunicação serial

Recepção Serial – Exemplo com interrupção

```
ORG 0X00
    GOTO        TRATA_INT     ;vem aqui se ocorrer interrupção

PRINCIPAL
    . . .                     ;códigos gerais
    . . .

TRATA_INT
    MOVF        RCREG, W      ;copia o byte recebido em W.
    MOWF        AUX           ;copia W para uma variável.
    RETFIE                    ;retorna na linha após a interrupção
```

Obs: Detecta/recebe o caractere no momento que chegar.

Comunicação serial

Transmissão Serial sem interrupção

- Mover o valor para o registrador **TXREG** e aguardar o fim da transmissão testando o bit **PIR1,TXIF.**

```
•MOVWF      TXREG       ; inicia a transmissão
•BTFSS      PIR1,TXIF   ;salta se já terminou
•GOTO       $-1
```

O flag **PIR1,TXIF** ficará **automaticamente** com **valor 1** quando a transmissão for concluída.

Comunicação serial

Exemplo de repetidor de comunicação serial:

```
PRINCIPAL
    BTFSS       PIR1, RCIF    ;testa se chegou caractere
    GOTO        PRINCIPAL     ;volta
    CALL        RECEBE        ;chama a rotina recebe
    GOTO        TRANSMITE     ;vai para a rotina transmite

RECEBE
    MOVF        RCREG, W      ;copia o byte recebido (em ASCII).
    RETURN                    ;retorna na linha após a chamada

TRANSMITE
    MOVWF       TXREG         ;inicia a transmissão
    BTFSS       PIR1,TXIF     ;se já terminou a transmissão, salta
    GOTO        $-1           ;senão, linha atual menos 1
    GOTO        PRINCIPAL
```


Comunicação serial

Configuração para Recepção (RX) e Transmissão (TX):

Banco 1

- **Habilitar a transmissão serial (TX).**
- **Gerador de baud rate no modo de alta velocidade (TX/RX).**
- **Deixar o registrador de transmissão vazio (TX).**
- **Para 9600 Set point do gerador = 25 (TX/RX).**
- **Habilitar a interrupção da recepção serial (RX). (se for utilizar)**
- **Habilitar a interrupção de transmissão serial (TX). (se for utilizar)**
- **PORTB,1 = Entrada (RX).**

Banco 0

- **Habilitar a recepção serial (RX).**
- **Habilitar a recepção contínua (RX).**
- **Habilitar a interrupção dos periféricos (RX). (se for utilizar)**
- **Habilitar a interrupção geral (RX). (se for utilizar)**

Comunicação serial

```
INICIO
    BSF         STATUS,RP0    ;MUDA PARA O BANCO 1
    CLRF        PORTA         ;PORTA TODO SAÍDA
    MOVLW       B'00000010'
    MOVWF       TRISB         ;PORTB,1 ENTRADA, RESTANTE SAÍDA
    BSF         PIE1, RCIE    ;*HABILITA A INTERRUPÇÃO DE REC NA SERIAL
    BSF         TXSTA, TXEN   ;HABILITA A TRANSMISSÃO SERIAL
    BSF         TXSTA, TRMT   ;LIMPA O REGISTRADOR DE TRANSMISSÃO
    BSF         TXSTA, BRGH   ;MODO ALTA VELOCIDADE
    MOVLW       .25           ;BAUD RATE PARA 9.600 BPS
    MOVWF       SPBRG         ;
    BCF         STATUS,RP0    ;VOLTA PARA O BANCO 0
    MOVLW       B'00000111'   ;DESABILITA OS...
    MOVWF       CMCON         ;... COMPARADORES ANALÓGICOS
    BSF         RCSTA,SPEN    ;HABILITA A COMUNICAÇÃO SERIAL
    BSF         RCSTA,CREN    ;HABILITA A RECEPÇÃO CONTÍNUA
    BSF         INTCON, PEIE  ;*HABILITA A INTERRUPÇÃO DOS PERIFÉRICOS
    BSF         INTCON, GIE   ;*HABILITAÇÃO GERAL DE INTERRUPÇÕES
    ... (CLRF   PORTA,  CLRF  PORTB,  GOTO  PRINCIPAL)
```