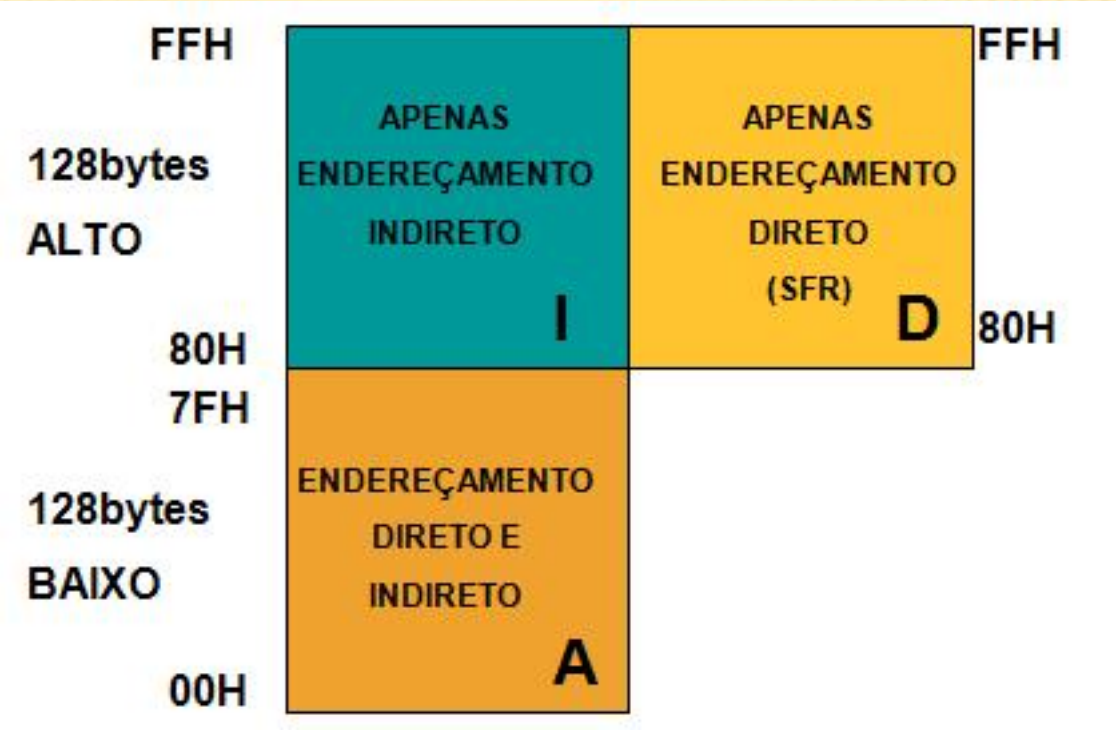
# 8051

# AULA 2

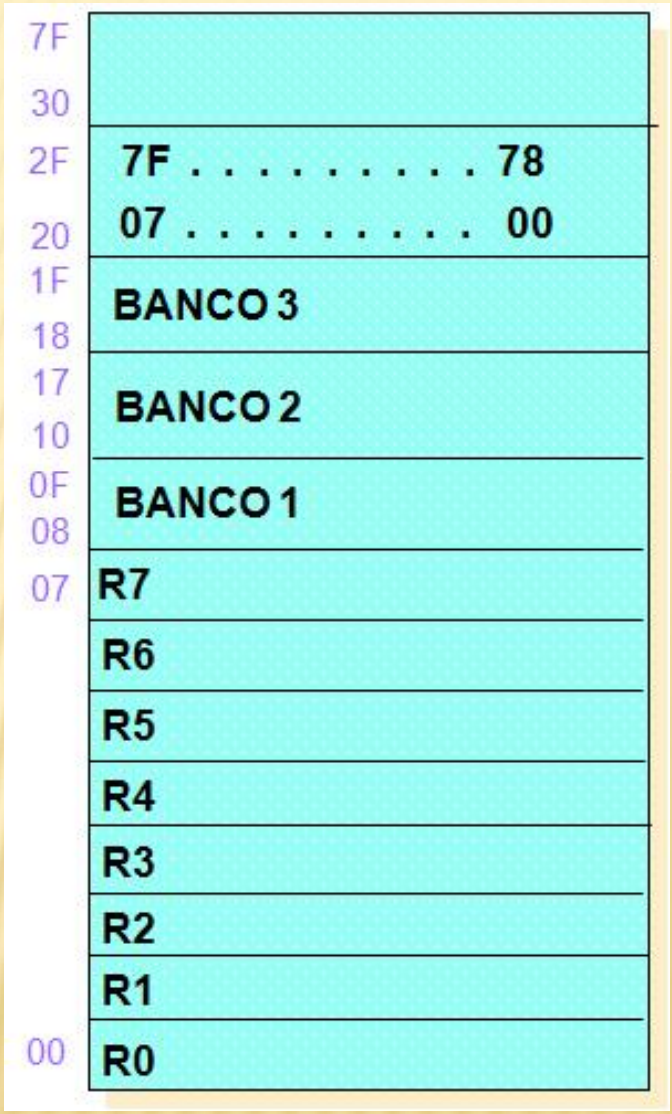
Prof. Adilson Gonzaga

# Memória de Dados Interna (RAM Interna)




- O endereçamento é feito com 8 bits
- Chips com 128 bytes de RAM não possuem a área I (Apenas Endereçamento Indireto)

# Memória de Dados Interna (Area A)



# Banco de Registradores


- Cada banco é formado pelos registradores R0 a R7.
- A seleção entre os Bancos de registradores é feita pelos bits 3 e 4 do byte PSW (Program Status Word)

PSW : Program Status Word

| CY | AC | F0 | RS1 | RS0 | OV | ---- | P |
|---|---|---|---|---|---|---|---|

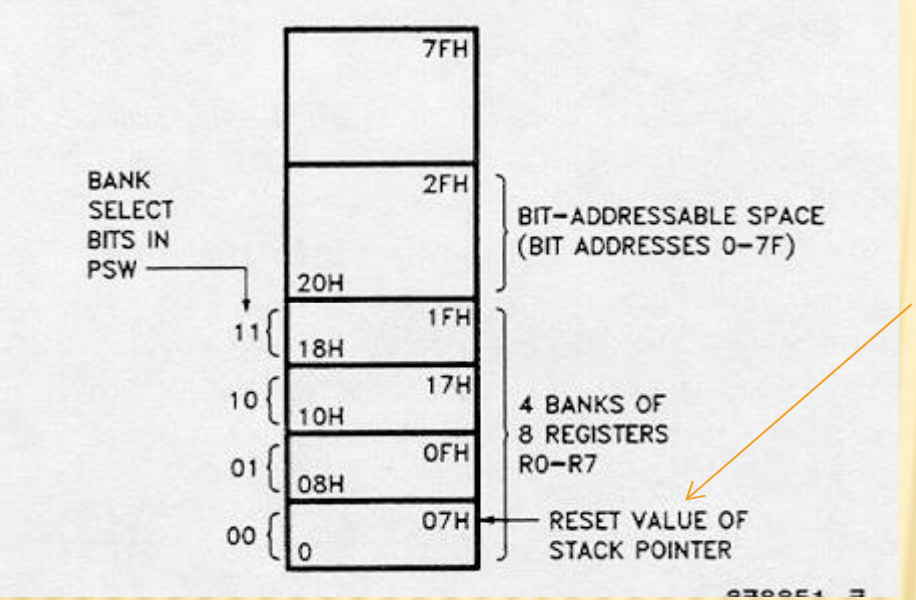
| RS1 | RSO | Register Bank | Address |
|---|---|---|---|
| 0 | 0 | 0 | 00h - 07h |
| 0 | 1 | 1 | 08h - 0Fh |
| 1 | 0 | 2 | 10h - 17h |
| 1 | 1 | 3 | 18h - 1Fh |

Ex: Para selecionar o Banco 3

```
MOV     PSW,#00011000B
```

# Pilha

A Pilha é a região da RAM Interna onde serão armazenados os endereços de Retorno das Sub-rotinas




• Ao se resetar a CPU, RS1 e RS0 são 0 , portanto o banco de registradores 'default' é o **Banco 0.**

**• O reset inicializa o Stack Pointer (SP) na posição 07h da RAM Interna.**

**• O SP é incrementando a cada vez que é usado.**

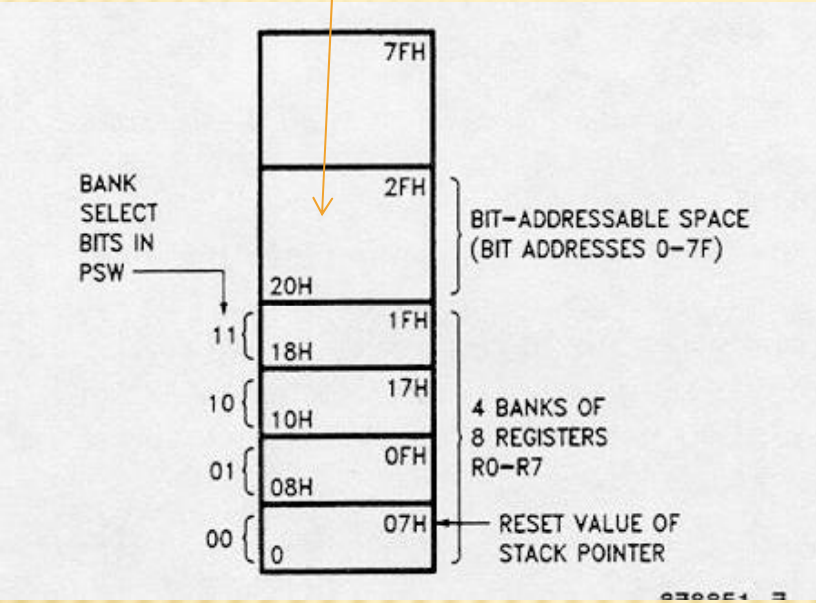
Para que se possa usar mais que um banco de registradores, o SP deve ser inicializado no programa em uma outra posição da RAM (por exemplo 30h).

Ex: Estabelecendo o endereço da Pilha em 30h

```
MOV     SP,#30H
```

# Memória endereçável a Bit

Região da RAM interna ( de 20h a 2Fh) na qual cada bit pode ser acessado por uma instrução de Bit.

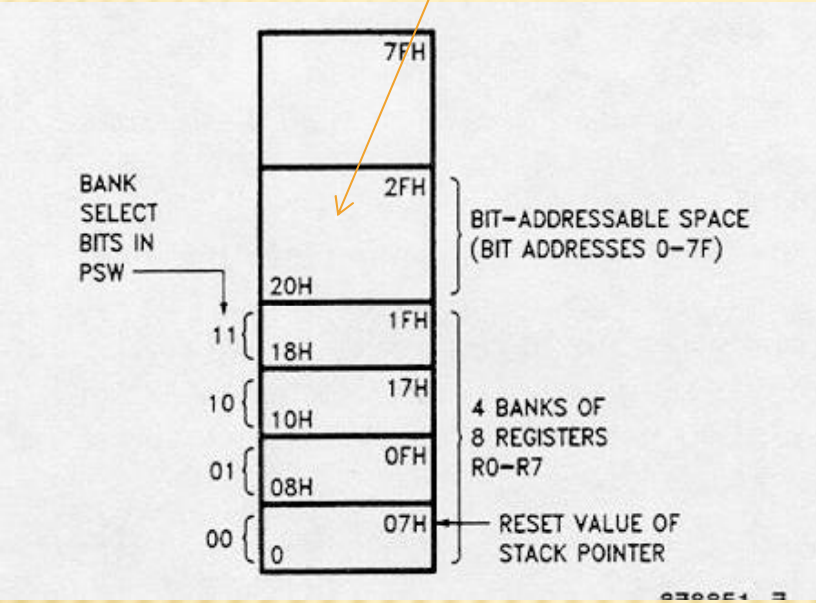



**As instruções de Bit são :**

***CLR bit*** zera o bit diretamente
***SETB bit*** seta o bit diretamente
***CPL bit*** complementa o bit diretamente
***ANL C,bit*** AND entre o bit e o carry
***ANL C,/bit*** AND entre o complemento do bit e o carry
***ORL C,bit*** OR entre o bit e o carry
***ORL C,/bit*** OR entre o complemento do bit e o carry
***MOV C,bit*** move o bit para o carry
***MOV bit,C*** move o carry para o bit
***JB bit,rel*** jmp para rel se bit = 1
***JNB bit,rel*** jmp para rel se bit = 0
***JBC bit,rel*** jmp para rel se bit = 1 e zere o bit

# Memória endereçável a Bit

Cada uma dessas posições de memória pode também ser acessada direta ou indiretamente por byte .




**Exemplo:**

**a) Endereçamento Direto**

***MOV 20h,#0AAh***

**b) Endereçamento Indireto**

***MOV R0,#20h***

***MOV @R0,#0AAh***

# Memória endereçável a Bit

**Exemplo :** Setar o bit 2 da posição 21h

***SETB 0Ah***

ou

***SETB 21h.2***

Cada uma dessas posições de memória pode também ser acessada direta ou indiretamente por byte .

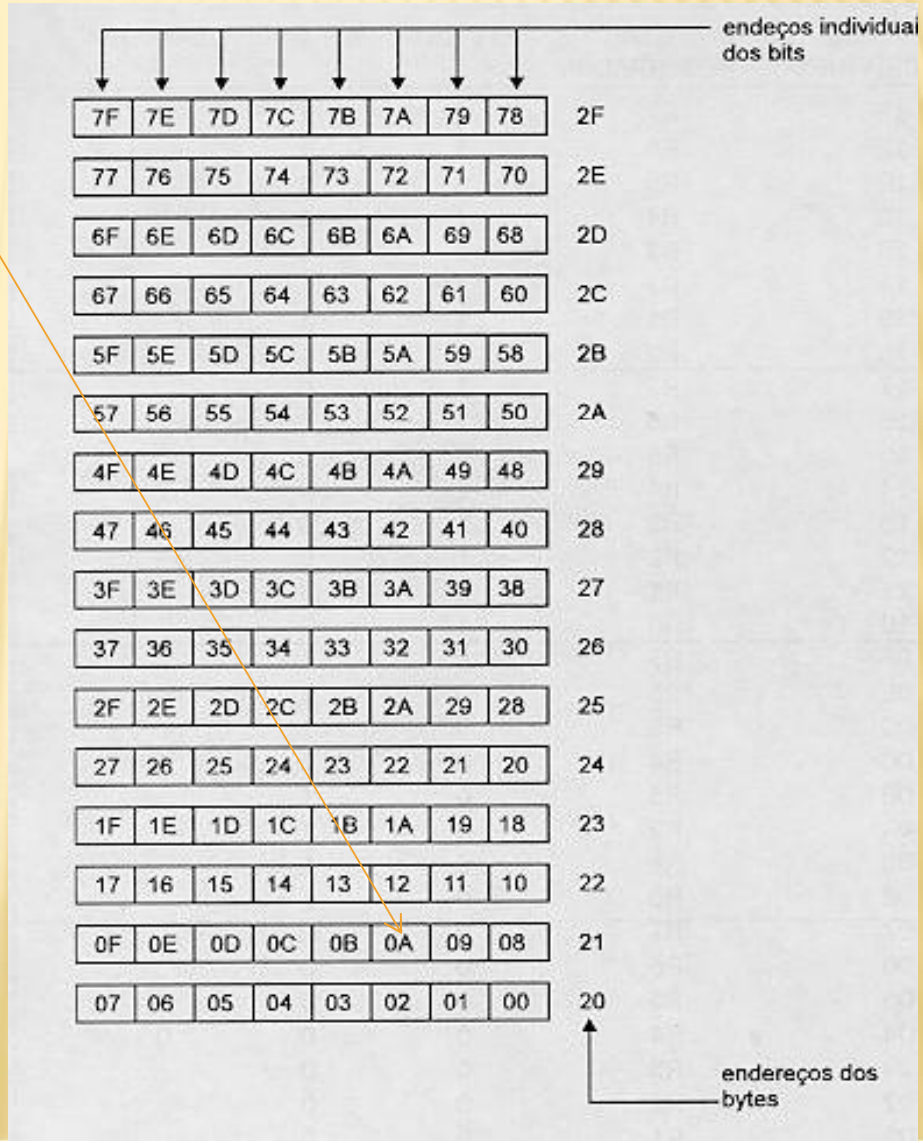
**Exemplo:**

***MOV 20h,#0AAh***

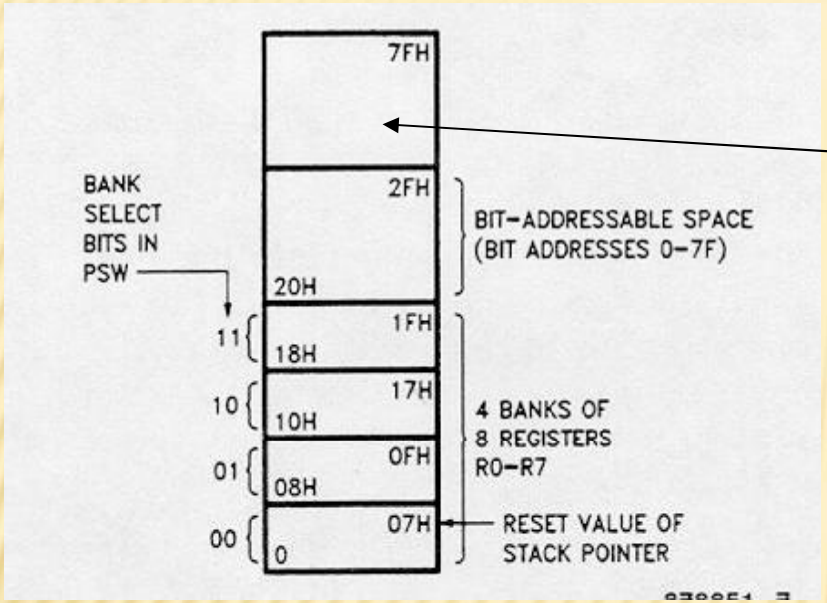
***ou***

***MOV R0,#20h***
***MOV @R0,#0AAh***

Endereços individuais dos Bits:



# Area de Rascunho (Scratch Pad)




- **As posições de 30h a 7Fh da RAM interna são endereçaveis apenas a Byte.**
- **São usadas para leitura e/ou escrita, por endereçamento direto ou indireto.**

Exemplo:
Escrever o dado A4 no endereço 30h

```
MOV     30h,#0A4h
```
Direto

```
MOV     R0,#30h
MOV     @R0,#0A4h
```
Indireto

Exemplo:
Ler o dado armazenado
no endereço 70h

```
MOV     A,70h
```
Direto

```
MOV     R0,#70h
MOV     A,@R0
```
Indireto

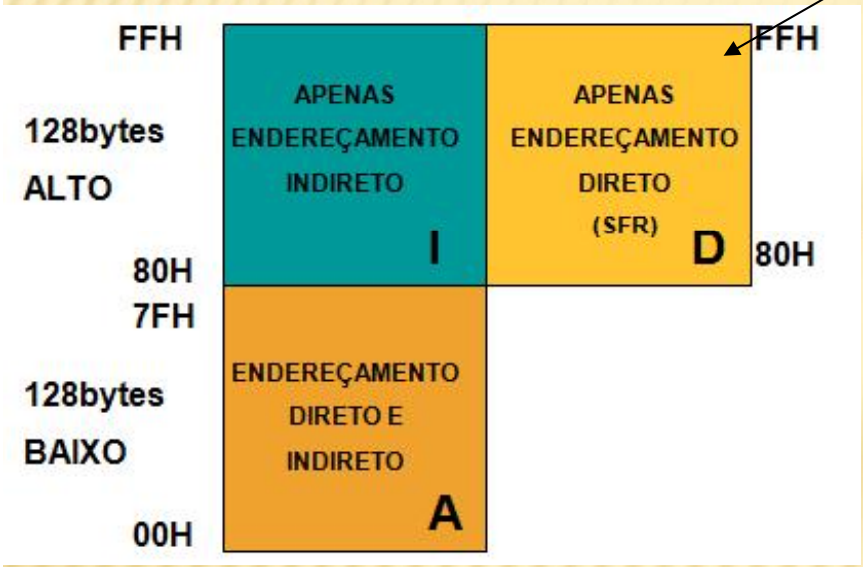
# EXEMPLO:

Somar o conteúdo do Registrador R0 do Banco 0 com o conteúdo do Registrador R0 do Banco 3

```
        ORG     0
        CLR     A           ; ZERANDO ACUMULADOR
        MOV     R0,#31H     ;CARREGANDO O REGISTRADOR R0 DO BANCO 0
        MOV     PSW,#00011000B      ;CHAVEANDO PARA O BANCO 3
        MOV     R0,#33H     ;CARREGANDO O REGISTRADOR R0 DO BANCO 3
        ADD     A,R0        ; SOMANDO ACUMULADOR COM R0 DO BANCO 3
        CLR     RS0         ; VOLTANDO PARA O BANCO 0
        CLR     RS1
        ADD     A,R0        ; SOMANDO O ACUMULADOR A R0 DO BANCO 0
        SJMP    $
        END
```

31h + 33h = 64h

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)




**80H a FFH, endereçável diretamente.**

**Este espaço contém :**

- **Registradores da CPU para funções especiais.**
- **Registradores de controle de I/O.**

**16 posições dos SFR's são endereçáveis por bit (endereços terminando em 0 ou 8)**

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)

**Registradores da CPU:**

**ACC - Accumulador.**

**B - Registrador B.**

**PSW - Program Status Word.**

**SP - Stack Pointer.**

**DPTR - Data Pointer (DPH, DPL).**

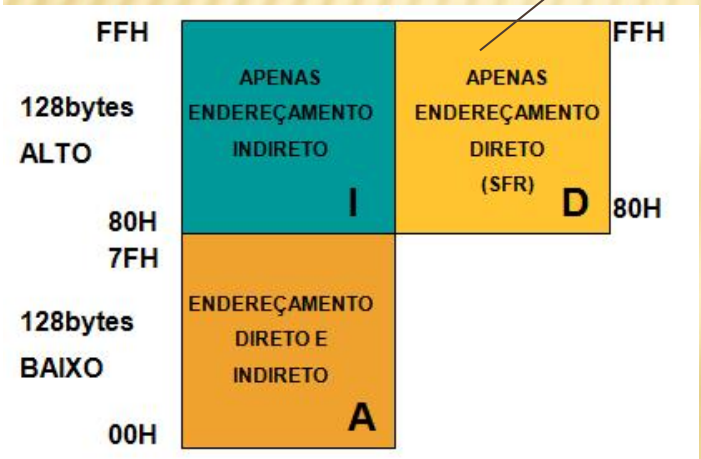



**Figure 5.** SFR Memory Map

| | 8 Bytes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW* | | | | | | | | D7 |
| C8 | T2CON*+ | T2MOD+ | RCAP2L+ | RCAP2H+ | TL2+ | TH2+ | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | IP* | | | | | | | | BF |
| B0 | P3 | | | | | | | | B7 |
| A8 | IE* | | | | | | | | AF |
| A0 | P2 | | | | | | | | A7 |
| 98 | SCON* | SBUF | | | | | | | 9F |
| 90 | P1 | | | | | | | | 97 |
| 88 | TCON* | TMOD* | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON* | 87 |

↑ Bit Addressable

**Portas de I/O:**

**P0 - Port 0.**

**P1 - Port 1.**

**P2 - Port 2.**

**P3 - Port 3.**

**Controle de interrupção:**

**IE - Interrupt Enable.**

**IP - Interrupt Priority.**

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)

**Contadores/Temporizadores:**

**TMOD - modo do Timer**

**TCON - controle do Timer**

**TH0 - MSB do Timer 0**

**TL0 - LSB do Timer 0**

**TH1 - MSB do Timer 1**

**TL1 - LSB do Timer1**

**Figure 5.** SFR Memory Map

| | 8 Bytes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW* | | | | | | | | D7 |
| C8 | T2CON*+ | T2MOD+ | RCAP2L+ | RCAP2H+ | TL2+ | TH2+ | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | IP* | | | | | | | | BF |
| B0 | P3 | | | | | | | | B7 |
| A8 | IE* | | | | | | | | AF |
| A0 | P2 | | | | | | | | A7 |
| 98 | SCON* | SBUF | | | | | | | 9F |
| 90 | P1 | | | | | | | | 97 |
| 88 | TCON* | TMOD* | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON* | 87 |

↑ Bit Addressable

**Comunicação Serial :**

**SCON – Controle da Serial.**

**SBUF – Registrador de Entrada e Saída da porta Serial.**

**Registrador de uso misto:**
**PCON – Controle de Potência, Bits de Proteção, etc...**

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)

Qualquer dos SFRs podem ser endereçados a **byte** diretamente através do endereço de cada um ou do nome.

**Figure 5.** SFR Memory Map

| | 8 Bytes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW* | | | | | | | | D7 |
| C8 | T2CON*+ | T2MOD+ | RCAP2L+ | RCAP2H+ | TL2+ | TH2+ | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | IP* | | | | | | | | BF |
| B0 | P3 | | | | | | | | B7 |
| A8 | IE* | | | | | | | | AF |
| A0 | P2 | | | | | | | | A7 |
| 98 | SCON* | SBUF | | | | | | | 9F |
| 90 | P1 | | | | | | | | 97 |
| 88 | TCON* | TMOD* | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON* | 87 |

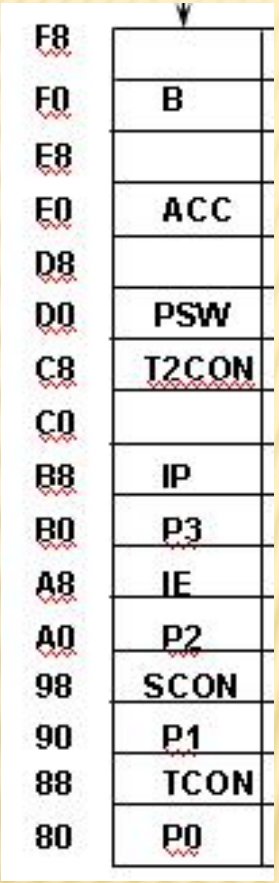
↑ Bit Addressable

**Exemplo:**

***MOV P0,#3Fh*** **ou** ***MOV 80h,#3fh***

***MOV DPL,DPH*** **ou** ***MOV 82h,83h***

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)




## SFRs endereçáveis a Bit

Os SFR's cujos endereços terminam em 0 ou 8h podem também ser endereçados a bit .

**Modos de acesso ao Bit**

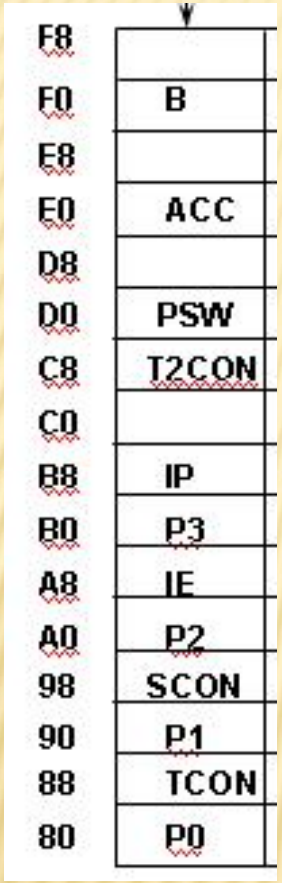
**a) por endereço do Bit dentro do Byte:**

***SETB 80h.1*** ; seta o bit 1 do endereço 80h (Porta P0)

***CLR 80h.2*** ; zera o bit 2 do endereço 80h (Porta P0)

| P0 | P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |
|---|---|---|---|---|---|---|---|---|
| 80h | 80.7 | 80.6 | 80.5 | 80.4 | 80.3 | 80.2 | 80.1 | 80.0 |

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)




**SFRs endereçáveis a Bit**

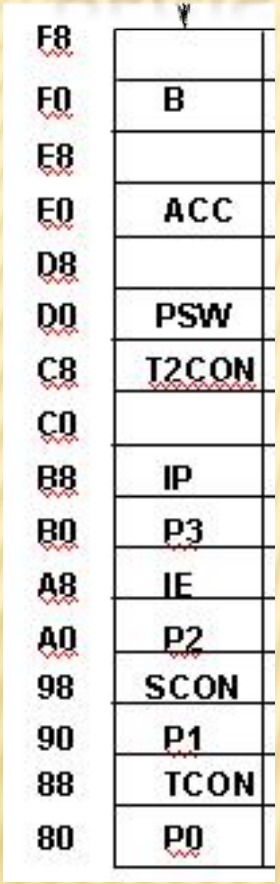
**b) por nome do bit:**

***SETB P0.1*** ; seta o bit 1 da Porta P0

***CLR P0.2*** ; zera o bit 2 da Porta P0

| P0 | P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |
|---|---|---|---|---|---|---|---|---|
| 80h | 80.7 | 80.6 | 80.5 | 80.4 | 80.3 | 80.2 | 80.1 | 80.0 |

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)




## SFRs endereçáveis a Bit

**c) pelo endereço absoluto do bit :**

***SETB 81h*** ; seta o bit 1 do endereço 80h (Porta P0)

***CLR 82h*** ; zera o bit 2 do endereço 80h (Porta P0)

P0

80h

| P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |
|---|---|---|---|---|---|---|---|
| 80.7 | 80.6 | 80.5 | 80.4 | 80.3 | 80.2 | 80.1 | 80.0 |
| 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 |

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)

## SFRs endereçáveis a Bit

Endereços absolutos dos Bits:

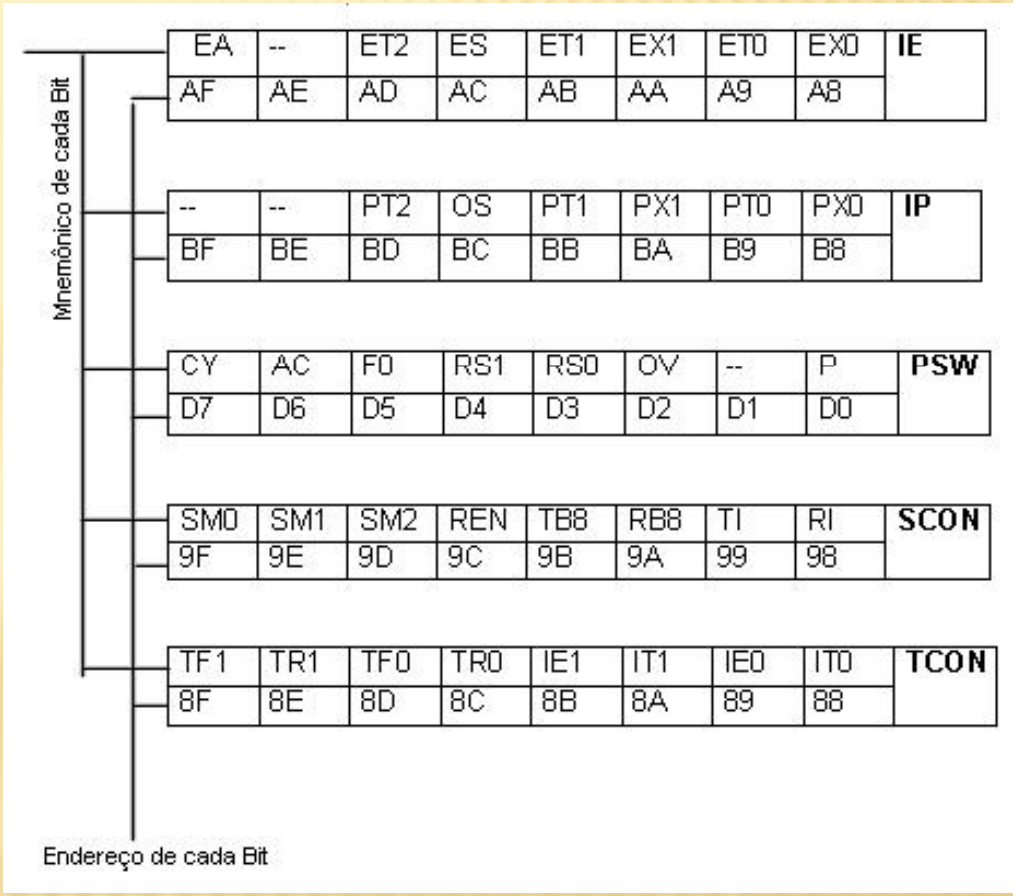
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| FF | FE | FD | FC | FB | FA | F9 | F8 | |
| F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 | **B** |
| | | | | | | | E8 | |
| | | | | | | | E0 | **Acc** |
| | | | | | | | D8 | |
| | | | | | | | D0 | **PSW** |
| | | | | | | | C8 | **T2CON** |
| | | | | | | | C0 | |
| | | | | | | | B8 | **IP** |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | **P3** |
| | | | | | | | A8 | **IE** |
| | | | | | | | A0 | **P2** |
| | | | | | | | 98 | **SCON** |
| 97 | 96 | 95 | 94 | 93 | 92 | 91 | 90 | **P1** |
| 8F | 8E | 8D | 8C | 8B | 8A | 89 | 88 | **TCON** |
| 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 | **P0** |

Endereço de cada Bit

# REGISTRADORES DE FUNÇÕES ESPECIAIS (SFR)

## SFRs endereçáveis a Bit

Os SFRs endereçáveis a bit que determinam funções, podem ser endereçados através do Mnemônico de cada bit:




**Exemplo:**

***SETB EA*** ; faz o bit 7 de IE=1

***SETB 0AFh*** ; idem

**Atenção! :**

***CLR AC*** *; zera o bit 6 do PSW (Carry auxiliar)*

***CLR 0ACh*** *; zera o bit de endereço 0ACh, ou seja, o bit 4 do registrador IE*

# FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

**<Rótulo> <Operação> <Operando> <Comentário>**

1) **Campo do Rótulo**:

- o primeiro caractere deve ser alfabético e pode ter no máximo 13 caracteres
- espaço, "tab" e " : " são considerados como caracteres finais do Rótulo
- corresponde ao endereço da instrução
- é opcional
- para identação do programa usar "tab" antes do próximo campo
- alinhar o primeiro caractere do Rótulo à esquerda

Exemplo:

Campo do Rótulo (Label)

```
;*******************************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*******************************************************************************

            ORG     0
            CLR     A           ; Inicializa o contador em zero
LOOP:       JB      P1.0,$      ; Testa se a chave está aberta
            INC     A           ; Incrementa o contador
            MOV     P2,A        ; Mostra o valor do contador na Porta P2
            CALL    ATRASO      ; Realiza o debounce da chave
            JNB     P1.0,$      ; Verifica se a chave continua fechada
            SJMP    LOOP        ; Retorna para nova contagem de pulso

;*******************************************************************************
```

# FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

**<Rótulo> <Operação> <Operando> <Comentário>**

**2) Campo da Operação :**

- contém o mnemônico da instrução ou diretivas do programa,
- não diferencia entre maiúsculas e minúsculas.

Exemplo:

```
;*******************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*******************************************************************

          ORG     0
          CLR     A           ; Inicializa o contador em zero
LOOP:     JB      P1.0,$      ; Testa se a chave está aberta
          INC     A           ; Incrementa o contador
          MOV     P2,A        ; Mostra o valor do contador na Porta P2
          CALL    ATRASO      ; Realiza o debounce da chave
          JNB     P1.0,$      ; Verifica se a chave continua fechada
          SJMP    LOOP        ; Retorna para nova contagem de pulso

;*******************************************************************
```

Campo da Operação (Mnemônicos)

# FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

**<Rótulo> <Operação> <Operando> <Comentário>**

**3) Campo do Operando:**

- especifica o dado a ser operado pela instrução.

Exemplo:

```
;*****************************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************************

            ORG     0
            CLR     A           ; Inicializa o contador em zero
LOOP:       JB      P1.0,$      ; Testa se a chave está aberta
            INC     A           ; Incrementa o contador
            MOV     P2,A        ; Mostra o valor do contador na Porta P2
            CALL    ATRASO      ; Realiza o debounce da chave
            JNB     P1.0,$      ; Verifica se a chave continua fechada
            SJMP    LOOP        ; Retorna para nova contagem de pulso

;*****************************************************************************
```

Campo do Operando

# FORMATO DO PROGRAMA FONTE

As declarações do programa fonte são constituídas pelos seguintes campos:

**<Rótulo> <Operação> <Operando> <Comentário>**

**4) Campo do Comentário:**

- Usado pelo programador para comentar a função da instrução no contexto do programa.
- É opcional.
- Sempre começa com ";" .
- Se o comentário mudar de linha, deve vir precedido de " ; "

Exemplo:

Campo do Comentário

```
;*****************************************************************************
; Programa de debounce de chave mecânica
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************************

            ORG     0
            CLR     A           ; Inicializa o contador em zero
LOOP:       JB      P1.0,$      ; Testa se a chave está aberta
            INC     A           ; Incrementa o contador
            MOV     P2,A        ; Mostra o valor do contador na Porta P2
            CALL    ATRASO      ; Realiza o debounce da chave
            JNB     P1.0,$      ; Verifica se a chave continua fechada
            SJMP    LOOP        ; Retorna para nova contagem de pulso

;*****************************************************************************
```

## TIPOS DE INFORMAÇÕES NO CAMPO DO OPERANDO

1. **Rótulo -** é um conjunto de caracteres com valor numérico associado a ele, e geralmente representando um endereço. Pode ter no máximo 13 caracteres, sendo o primeiro obrigatoriamente uma letra . Os demais caracteres podem ser letras, dígitos e ponto.

Exemplo:

```
;*************************************************************************
; Programa: Contador de pulsos com debounce de chave mecânica.
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*************************************************************************

            ORG     0
            MOV     A,#00000000b ; Inicializa o contador em zero
LOOP:       JB      P1.0,$       ; Testa se a chave está aberta
            INC     A            ; Incrementa o contador
            MOV     P2,A         ; Mostra o valor do contador na Porta P2
            CALL    ATRASO       ; Realiza o debounce da chave
            JNB     P1.0,$       ; Verifica se a chave continua fechada
            MOV     P3,#'H'      ; Envia o Caracter H para a Porta P3
            SJMP    LOOP         ; Retorna para nova contagem de pulso

;*************************************************************************

ATRASO:     MOV R0,#0FFH    ; Valor do atraso para debounce da chave de
            DJNZ R0,$       ; aproximadamente 500 us
            RET
;*************************************************************************
```

## TIPOS DE INFORMAÇÕES NO CAMPO DO OPERANDO

2. **Constante numérica –**

- **Decimal -** é o default; o final D é opcional.

- **Hexadecimal -** a constante deve ser finalizada com H; quando inicia com uma letra deve ser precedida por 0(zero) .

- **Binária -** deve ser finalizada com B.

```
;*************************************************************************
; Programa: Contador de pulsos com debounce de chave mecânica.
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*************************************************************************

            ORG     0
            MOV     A,#00000000b  ; Inicializa o contador em zero
LOOP:       JB      P1.0,$        ; Testa se a chave está aberta
            INC     A             ; Incrementa o contador
            MOV     P2,A          ; Mostra o valor do contador na Porta P2
            CALL    ATRASO        ; Realiza o debounce da chave
            JNB     P1.0,$        ; Verifica se a chave continua fechada
            MOV     P3,#'H'       ; Envia o Caracter H para a Porta P3
            SJMP    LOOP          ; Retorna para nova contagem de pulso

;*************************************************************************

ATRASO:     MOV R0,#0FFH  ; Valor do atraso para debounce da chave de
            DJNZ R0,$     ; aproximadamente 500 us
            RET
;*************************************************************************
```

## TIPOS DE INFORMAÇÕES NO CAMPO DO OPERANDO

2. **Constante numérica –**

- **Octal -** deve ser finalizada com Q

- **Caracteres ASCII** - A constante ASCII deve vir entre cotas únicas.

- **Contador de posição -** o valor corrente do PC pode ser usado em expressões colocando-se um $ na posição desejada da expressão..

```
;*****************************************************************************
; Programa: Contador de pulsos com debounce de chave mecânica.
; Uma sub-rotina de atraso deve ser inserida para garantir o tempo de
; estabilização mecânica da chave. Para a maioria das chaves um atraso de
; 500 us é suficiente.
;*****************************************************************************

            ORG     0
            MOV     A,#00000000b ; Inicializa o contador em zero
LOOP:       JB      P1.0,$       ; Testa se a chave está aberta
            INC     A            ; Incrementa o contador
            MOV     P2,A         ; Mostra o valor do contador na Porta P2
            CALL    ATRASO       ; Realiza o debounce da chave
            JNB     P1.0,$       ; Verifica se a chave continua fechada
            MOV     P3,#'H'      ; Envia o Caracter H para a Porta P3
            SJMP    LOOP         ; Retorna para nova contagem de pulso

;*****************************************************************************

ATRASO:     MOV R0,#0FFH    ; Valor do atraso para debounce da chave de
            DJNZ R0,$       ; aproximadamente 500 us
            RET
;*****************************************************************************
```

# PSEUDO-INSTRUÇÕES OU DIRETIVAS DO ASSEMBLER

**As diretivas não geram código de máquina!!.**

São utilizadas para complementar as informações que permitam a montagem efetiva do programa.

- Indicar o Endereço Inicial do Programa.
- Reservar área de Dados
- Definir equivalência entre valores
- Etc…

## Principais Diretivas:

**1) Diretiva ORG** – define a Origem do programa

**ORG** ***endereço***

A diretiva ORG deve ser usada para instruir ao Assembler em qual endereço deve começar a colocar o código do programa compilado.
Por default, na ausência da diretiva ORG, o código do programa começa no endereço 0000h, que é o endereço de reset dos microcontroladores da família MCS-51.

O valor do endereço deve ser uma expressão válida. Ou seja, o endereço pode ser um valor numérico válido ou conter uma expressão com contador de posição.

```
ORG 0 ; inicia o código do programa no endereço zero (endereço de reset do microcontrolador)
```

```
ORG 10h ; inicia o código do programa no endereço 10 hexadecimal.
```

Principais Diretivas:

**2) Diretiva DB** – Define Byte

**DB databyte1** ***[ , databyte2, [databyte3... ] ]***
**DB "string1"** ***[, "string2" [, "string3"... ] ]***

A diretiva DB permite ao programador inserir bytes de dados diretamente no programa na posição de memória corrente.

Os valores numéricos de 8 Bits são inseridos respeitando-se o seu formato (decimal, hexadecimal, binário, octal). Se mais de um valor for inserido eles devem vir separados por vírgula.

Exemplo:

```
ORG 0010h
DB 05h, 0CFh
```

Armazena na posição 0010h da Memória de Programa, o Byte 05h e na posição seguinte (0011h) o Byte CFh

## Principais Diretivas:

**2) Diretiva DB** – Define Byte

**DB databyte1** ***[ , databyte2, [databyte3... ] ]***
**DB "string1"** ***[, "string2" [, "string3"... ] ]***

Caracteres ASCII isolados ou "Strings" de caracteres ASCII devem estar contidos entre aspas.

**Obs:** Esta diretiva deve ser colocada sempre depois do fim lógico do programa para que os dados inseridos não sejam confundidos com instruções executáveis.

```
ORG     0010H
DB      05H,0CFH,'ISTO E UM TESTE',00H
```

; esta diretiva insere diretamente a partir da
; posição de memória 0010h os seguintes
; códigos hexadecimais ( 05, CF, 49, 53, 54,
; 4F, 20, 45, 20, 55 ,4D, 20, 54, 45, 53, 54, 45,
; 00)

Principais Diretivas:

### 3) Diretiva DW - Define Word

**DW dataword1** ***[ , dataword2, [dataword3... ] ]***
**DW "string1"** ***[. string2 [, string3... ] ]***

A diretiva DW permite ao programador inserir palavras de dados (2 bytes) diretamente no programa na posição de memória corrente.

Os valores numéricos de 16 Bits (2 Bytes) são inseridos respeitando-se o seu formato (decimal, hexadecimal, binário, octal). Se mais de um valor for inserido eles devem vir separados por vírgula. Se apenas um Byte for inserido o MSB será adotado como 00.

Caracteres ASCII isolados ou "Strings" de caracteres ASCII devem estar contidos entre aspas. Se apenas um caractere ASCII for inserido, o LSB será 00.

**Exemplo:**

```
ORG 0100h
DW 567Fh, "TESTE", 05H, "A"
```

```
; esta diretiva insere diretamente a partir da
; posição de memória 0100h os seguintes
; códigos hexadecimais (56, 7F, 54, 45, 53, 54, 45, 00,
; 05, 41, 00)
```

Principais Diretivas:

4) Diretiva EQU ---- (Equate) Igual

literal EQU valor

• Atribui um valor a um literal.

• O literal só pode receber um único valor.

• O valor pode ser um valor numérico ou uma expressão.

• Uma vez declarado o valor do literal este não poderá ser redefinido.

Exemplo:

```
                ORG 0
Controle        EQU     10h             ; atribui 10h ao literal Controle
                MOV     A, #Controle    ; Acumulador = 10h
```

## Exemplo de escrita de um Programa Fonte:

```
;*****************************************************************************
; Título do Programa: Programa Principal                                     *
; Este é um programa exemplo para mostrar como escrever um código de programa *
; fonte e comentá-lo, facilitando  futuras correções.                        *
;*****************************************************************************

; Atribuição das variáveis do programa

Var1    EQU 30h          ; Esta variável estabelece o início da contagem
Var2    EQU 02h          ; Variável que determina o número de incrementos
;*****************************************************************************

        ORG 0            ; Início do programa principal no endereço 0000h

Init:   MOV A, #Var1     ; Usar o Valor de Var1 para iniciar a contagem
        ADD A, #Var2     ; Incrementar a contagem de 02 unidades
Loop:   ACALL Rot1       ; Chamar a Sub-rotina de análise dos dados
        SJMP Loop        ; Voltar para o Loop e permanecer calculando

; Fim Lógico do Programa Principal. (O Fim Lógico não permite que o programa
; principal ultrapasse este ponto, evitando executar lixo que esteja residente na
; memória)
;*****************************************************************************
```

```
;*****************************************************************
; Área das Sub-rotinas: as sub-rotinas devem ficar após o Fim Lógico do programa *
; Principal, pois, serão chamadas por instruções específicas quando for necessário  *
; executá-las.                                                                      *
;*****************************************************************
```

```
;*****************************************************************
; Sub-rotina Rot1 :                                                 *
; Esta sub-rotina analisa os dados e executa o cálculo dos máximos valores  *
;*****************************************************************

Rot1:   ADD A, #Var1
        MOV R0,#Var2
        DJNZ R0, Pulo              ; Loop de controle do sistema
        SJMP Rot1
Pulo:   RET

; Fim da Sub-rotina Rot1.
;*****************************************************************
```

```
;*****************************************************************
; Área de Dados para criação da Tabela                           *
;*****************************************************************

Tab1:   DB 12h, 45h, 0DFh, "abcd"
Tab2:   DW "Erro. Entrar com outro Valor", 34h, 5656h

; Fim da área de dados
;*****************************************************************

;*****************************************************************
; Fim físico do Programa. Define para o programador e para o compilador a região *
; de código de todo o programa.                                  *
; Atenção: O Fim Físico é apenas simbólico. Não é um código que faz o programa  *
; parar! É preciso ter um fim Lógico!                            *
;*****************************************************************
END
```

# CONTROLE DE FLUXO DE PROGRAMA

Comparação de Bytes

**A instrução CJNE (compare e salte se não for igual) faz o flag de carry = 1 depois da execução, se o dado em comparação for maior que o conteúdo do registrador em questão (A, Rn ou @Ri).**

**CJNE A,direct,rel**

**CJNE A,#data,rel**

**CJNE Rn,#data,rel**

**CJNE @Ri,#data,rel**

# CONTROLE DE FLUXO DE PROGRAMA

```
        CJNE    A,Valor,Test    ;Desvie se A < Valor.
Test:   JC      LT
```

```
        CJNE    A,Valor,Test    ;Desvie se A >= Valor.
Test:   JNC     GTE
```

```
        CJNE    A,Valor,Test    ;Desvie se A > Valor.
        SJMP    Else
Test:   JNC     GT
Else:   -----
```

```
        CJNE    A,Valor,Test    ;Desvie se A <= Valor.
        SJMP    LTE
Test:   JC      LTE
```

# CONTROLE DE FLUXO DE PROGRAMA

Teste de Bits

Testa o bit e salta para o endereço rel se bit=1

```
JB      bit,rel
```

```
        ORG     0
        JB      P1.0,SAI
        SJMP    CONT
SAI:    ------
```

# CONTROLE DE FLUXO DE PROGRAMA

Teste de Bits

Testa o bit e salta para o endereço rel se bit=0

```
JNB     bit,rel
```

```
        ORG     0
        JNB     P1.0,SAI
        SJMP    CONT
SAI:    ------
```

# CONTROLE DE FLUXO DE PROGRAMA

Teste de Bits

Testa o bit , salta para o endereço rel se bit=1  e complementa o bit

```
JBC     bit,rel
```

```
        ORG     0
        JBC     P1.0,SAI
        SJMP    CONT
SAI:    ------
```

# DADOS ARMAZENADOS NA MEMÓRIA DE PROGRAMA

Dados são armazenados na Memória de Programa somente usando as Psudo-Instruções DB ou DW, durante a fase de gravação do programa.

Existem duas instruções apenas que permitem ler estes dados armazenados na Memória de Programa

```
MOVC    A, @A+DPTR
```

```
MOVC    A, @A+PC
```

# EXEMPLO:

Somar dois dados armazenados na Memória de Programa.

```
        ORG     0
        MOV     DPTR,#TAB       ; APONTA O PONTEIRO PARA O ENDEREÇO TAB
        CLR     A               ; ZERA O ACUMULADOR
        MOVC    A,@A+DPTR       ; LÊ O DADO DO ENDEREÇO TAB
        MOV     R0,A            ; SALVA EM R0
        INC     DPTR            ; APONTA O PONTEIRO PRA A PRÓXIMA POSIÇÃO (TAB+1)
        CLR     A               ; ZERA O ACUMULADOR
        MOVC    A,@A+DPTR       ; LÊ O DADO DE TAB+1
        ADD     A,R0            ; SOMA OS DOIS VALORES LIDOS
        SJMP    $
TAB:    DB      23H,32H
        END
```