

```
;-----  
; Programa: primeiro.asm  
; Move dado da acumulador para o registro B e vice-versa.  
; Autor: Vargas  
; Data:16:52 16/02/01  
;-----
```

```
org 0000h ;endereço inicial da ROM  
  
mov a,#09h  
mov b,a  
mov a,#00h ;limpa o acumulador  
mov a,b ;restaura o Acc  
  
end
```

```
;-----  
; Programa: segundo.asm  
;  
; Este programa mostra os efeitos das diretivas de compilação.  
;  
; Autor: Prof. Vargas  
; Data:17:57 27/04/01  
;-----
```

```
org 0000h  
  
movx @dptr,a ; move A para onde dptr aponta  
; efeito do @  
mov a,#23h ; move 23 para o acumulador (#)  
mov a,23h ; carrega no A, o conteúdo da  
; memória interna de endereço 23h  
  
end
```

```
;-----  
; Programa: Mover  
; Exemplo do uso de propriedade  
; do compilador (#)  
; 12:17 23/02/01  
;-----
```

```
org 0000H  
ender equ 1000H  
  
mov R1,10H ; Carrega o dado do endereço 10H em R1  
mov R1,#10H ; Carrega 10H em R1  
  
end
```

```
;-----  
; Programas: Memext.asm  
; Como acessar a memória externa.  
; 12:20 23/02/01  
; Autor: Prof. Vargas  
;-----
```

```
org 0000h  
newadd equ 1027h  
  
mov a,#23h ;carrega a referência  
mov dptr,#478h ;carrega o endereço da RAM externa  
movx @dptr,a ;envia dado á RAM externa
```

```

;-----
; Endereçando com o EQUate
;-----
    mov     dptr,#newadd    ;Aponta o novo endereço
    movx   @dptr,a         ;envia dado á RAM externa

    end

;-----
; Programa: Loop.asm
; Carrega um valor 3Eh a partir de 3000h usando um contador
; 12:39 23/02/01
; Autor: Prof. Vargas
;-----
    org    0000
    mov    a,#3EH
loop:
    dec    A
    jnz   loop

    end

;-----
; programa: compara.asm
; compara o dado do endereço 1777h e 2398h
; Guarda ffh em 1000h se forem diferentes e 11h se forem iguais
; 12:35 23/02/01
; Autor: Prof. Vargas
;-----
    org    0000h
num1    equ    1777h
num2    equ    2398h
result  equ    1000h

    mov    dptr,#num1      ;aponta o primeiro valor
    movx   a,@dptr        ;move o conteúdo para o acumulador
    mov    r0,a           ;preserva o dado em R0
    mov    a,#00h         ;zera (ou limpa) o acumulador
    mov    dptr,#num2     ;Aponta o segundo valor
    movx   a,@dptr        ;Carrega o 2º valor no acumulador
    subb   a,r0           ;executa acumulador-R0
    mov    dptr,#result   ;Aponta para o local onde guardará o resultado
    jnz   diferent       ;Se Acc é diferente de zero N1 é diferente de N2
    mov    a,#ffh        ;sinaliza ffh
    movx   @dptr,a
    jmp    final

diferent:
    mov    a,#11h        ;se Acumulador=0 então N1=N2
    movx   @dptr,a       ;sinaliza 11h

final:
    end

```

```

;-----
; Programa: Compara2.asm
; Exercício que usa decremento, salto e os Flags
; Data: 11:31 09/03/01
; Autor: Prof. Vargas
;-----
        org     0000H
refere equ 2010H

        mov     a,#0AH           ; Prepara contador de 10
        mov     dptr,#refere     ; Aponta para o endereço inicial
loop1:
        movx    @dptr,A         ; envia o dado
        inc     dptr            ; aponta para o próximo endereço
        dec     a               ; decrementa o contador de dados
        jnz     loop1          ; Se não terminou faça novamente...

        end

;-----
; Programa:Multip.asm
; Este programa carrega M1 em 2100H,M2 em 2101H e executa M=M1*M2 de duas
; formas:
; a) M = M1 x M2
; b) M= M1+M1+...+M1 (M2 vezes)
; O 1º resultado ficará no endereço 2102H e o 2º em 2103H
; Data : 11:39 22/03/01
; Autor: Prof. Vargas
;-----
        org     0000H
m1      equ     2100H
ma      equ     2102H
mb      equ     2103H

        mov     dptr,#m1
        movx    A,@dptr
        mov     B,A             ; recolhe M1 e guarda em B
        mov     R2,B           ; preserva M1
        inc     dptr
        movx    A,@dptr       ; recolhe M2
        mov     R0,A           ; preserva M2
        mul     AB
        mov     R1,A           ; preserva resultado da multiplicação
        mov     dptr,#ma
        movx    @dptr,A

;-----
; inicia segunda multiplicação
;-----
; Para isso, R0 contém M2 que na verdade é o contador de somas,
; e deve ser decrementado de 1
;-----
        dec     R0             ; corrige o contador
        mov     A,R2           ; recupera M1
        mov     B,R2           ; recupera M1
denovo:
        add     A,B
        dec     R0
        cjne    R0,#00,denovo
        mov     dptr,#mb
        movx    @dptr,A       ; guarda resultado em 2103H

        end

```

```

;-----
; Programa: TIO.ASM
; Teste de I/O deve examinar o bit 0 e grava 99h no endereço 1000h quando este
; evento ocorrer.
;
; Data: 16:36 30/03/01
; Autor: Prof. Vargas
;-----

```

```

    org     0000h
aguard:
    mov     A,P0
    jnb     A.0,aguard     ;não é 1
    mov     A,#99h        ;foi ligado
    mov     dptr,#1000h
    movx    @dptr,A

    end

```

```

;-----
; Programa: Contabit.asm
; Grava um byte qualquer em 1E3FH, e, em seguida, conta o número de bits 1
; desse elemento.
; r1= numero de bits 1
; Data: 11:31 22/03/01
; Autor: Prof. Vargas
;-----

```

```

    org 0000H
refer equ 1E3FH
    mov     r1,00H        ;contador de bit
    mov     dptr,#refer
    movx    A,@dptr      ;recolhe o dado ao acumulador
    jnb     A.0,loop1     ;não é zero...pule
    inc     r1            ;atualiza contador de bit=1
loop1:
    jnb     A.1,loop2     ; não é zero...pule
    inc     r1            ;atualiza contador de bit=1
loop2:
    jnb     A.2,loop3     ;não é zero...pule
    inc     r1            ;atualiza contador de bit=1
loop3:
    jnb     A.3,loop4     ;não é zero...pule
    inc     r1
loop4:
    jnb     A.4,loop5
    inc     r1
loop5:
    jnb     A.5,loop6
    inc     r1
loop6:
    jnb     A.6,loop7
    inc     r1
loop7:
    jnb     A.7,fim
    inc     r1
fim:
    end

```

```

;-----
; Programa: Troca.asm
; Troca 10 bytes da posição 2010H para 1000H
; Data: 11:31 09/03/01
; Autor: Prof. Vargas
;-----
        org 0000H
adres1 equ    2010H
adres2 equ    1000H
dentro equ    20H           ; endereço interno intermediário

;-----
; Carrega dados de 0AH a 01H no primeiro endereço
;-----

        mov     a,#0AH       ; Prepara contador de 10
        mov     dptr,#adres1 ; Aponta para o endereço inicial
loop1:
        movx    @dptr,A      ; envia o dado
        inc     dptr         ; aponta para o próximo endereço
        dec     a           ; decrementa o contador de dados
        jnz    loop1        ; Se não terminou faça novamente...

;-----
; Recupera dado a dado
;-----
        mov     A,#0AH       ; a quantidade de dados
        mov     R1,#dentro   ; endereço intermediário interno
        mov     dptr,#adres1
again:
        mov     R0,A         ; preservo o contador
        movx    A,@dptr     ; recupero o dado
        mov     @R1,A
        inc     dptr
        inc     R1           ; salvo o dado
        mov     A,R0        ; recupero contador
        dec     A           ; chegou ao fim ?
        jnz    again        ; não, faça de novo

;-----
; transfere para o segundo endereço
;-----
        mov     A,#0AH       ; a quantidade de dados
        mov     R1,#dentro   ; endereço intermediário interno
        mov     dptr,#adres2 ; aponto para o destino
denovo:
        mov     R0,A         ; preservo o contador
        mov     A,@R1        ; recupero o dado
        movx    @dptr,a     ; envio o dado
        inc     dptr
        inc     R1           ; atualizo o destino
        mov     A,R0        ; recupero o contador de dados
        dec     A           ; chegou ao fim ?
        jnz    denovo       ; não, faça de novo

        end

```

```

;-----
; Programa: Troca2.asm
; Grava 10 bytes em ordem decrescente na posição 2010H e em seguida reconstrói
; em ordem crescente
; Data: 16:00 23/03/01
; Autor: Prof. Vargas
;-----
        org 0000H
adres1 equ    2010H
dentro equ    20H          ; endereço interno intermediário

;-----
; Carrega dados de 0AH a 01H no primeiro endereço
;-----
        mov     a,#0AH      ; Prepara contador de 10
        mov     dptr,#adres1 ; Aponta para o endereço inicial
loop1:
        movx    @dptr,A     ; envia o dado
        inc     dptr        ; aponta para o próximo endereço
        dec     a           ; decrementa o contador de dados
        jnz    loop1       ; Se não terminou faça novamente...

;-----
; Recupera dado a dado
;-----
        mov     A,#0AH      ; a quantidade de dados
        mov     R1,#dentro  ; endereço intermediário interno
        mov     dptr,#adres1

again:
        mov     R0,A        ; preservo o contador
        movx    A,@dptr    ; recupero o dado
        mov     @R1,A
        inc     dptr
        inc     R1          ; salvo o dado
        mov     A,R0       ; recupero contador
        dec     A           ; chegou ao fim ?
        jnz    again      ; não, faça de novo

;-----
; transfere para o segundo endereço
;-----
        mov     R1,#dentro  ; endereço intermediário interno
        mov     A,#0AH      ; a quantidade de dados
        add     A,R1
        dec     A           ; corrige A
        mov     R1,A        ; Aponta para o final da lista
        mov     A,#0BH
        mov     dptr,#adres1 ; aponto para o destino
denovo:
        mov     R0,A        ; preservo o contador
        mov     A,@R1      ; recupero o dado
        movx    @dptr,a    ; envio o dado
        inc     dptr
        dec     R1          ; atualizo o destino
        mov     A,R0       ; recupero o contador de dados
        dec     A           ; chegou ao fim ?
        jnz    denovo      ; não, faça de novo

        end

```

```

;-----
; Programa: Pilha.asm
; Entrada: Primeiro e segundo números passa dos pela pilha operacional
;-----
; Subrotina soma
; Saída : R1=0 se a soma>255 (1 byte)
; : R1=1 se a soma<255
; : R0=resultado da soma
; Ambos os valores são recebidos diretamente da pilha operacional
; ** Afeta todos os flags **
;
; Data:15:55 20/04/01
; Autor: Prof. Vargas
;-----

```

```

org 0000h
mov SP,#30h ;ajusta a pilha
mov a,#c8h ;Carrega N1
push a
mov a,#1eh ;Carrega N2
push a
lcall soma
pop a
pop a ;recupera result
prende:
cjne a,#00,prende
ljmp fim
;=====

```

```

soma:
pop R0 ;preserva end1
pop R1 ;Preserva end2
pop a ;recupera N2
pop R2 ;recupera N1
add a,R2 ;soma
jc maior ;S>255
mov R3,#01h ;sinaliza menor
ljmp saida
maior:
mov R3,#00h ;sinaliza maior
saida:
push R3 ;maior ou menor
push a ;guarda a soma
push R1
push R0
ret
;=====
fim:
end

```

```

;-----
; Programa: Subr1.asm
;
; Entrada: R2=primeiro número
; : R3=segundo número
;-----
; Subrotina soma
; Saída : R1=0 se a soma>255 (1 byte)
; : R1=1 se a soma<255
; : R0=resultado da soma
; Ambos os valores são recebidos diretamente dos registros R2 e R3
; ** Afeta todos os flags **
;-----
org 0000h

```

```

        mov     R2,#c8h           ;carrega N1
        mov     R3,#1eh          ;carrega N2
        lcall   soma
prende:
        mov     R2,#0h           ;limpa N1
        mov     R3,#0h           ;limpa N2
        ljmp    prende
;=====
soma:
        mov     a,R2             ;prepara N1
        add     a,R3             ;soma
        jc      maior           ;S>255
        mov     R1,#01h         ;sinaliza menor
        ljmp    saida
maior:
        mov     R1,#0h           ;sinaliza maior

saida:
        mov     R0,a             ;soma em R0
        ret
;=====
        end

```

```

;-----
; Programa: Subr2.asm
; Esta subrotina recebe dois dados em R6 e R7 do programa principal e informa
; qual deles é o maior retornando no Acumulador
; o resultado:
; Acc=00 ..... R6 > R7
; Acc=FF ..... R6 <= R7
; O programa principal, por sua vez, recolhe R6 de 2000h e R7 de 2001h
; O resultado será guardado em P0:
; P0=01010101 (R6 > R7)
; P0=00001111 (R6 <= R7)
; permanecendo lá por um determinado tempo, após o que será apagado
; (P0=00000000)
;
; Data:16:34 06/04/01
; Autor: Prof. Vargas
;-----

```

```

        org     0000h

        mov     dptr,#2000h
        movx    A,@dptr          ;pega o 1º numero
        mov     R6,A
        inc     dptr
        movx    A,@dptr          ;pega o 2º numero
        mov     R7,A
        call    decide
        cjne    A,#06h,proxim    ;R6 é o maior
        mov     P0,#55h
        ajmp    frente
proxim:
        mov     P0,#0Fh          ;R7 é o maior
frente:
        call    delay
        mov     P0,#00h
        ajmp    fim

```

```

;=====
; Subrotina Decide
;=====

```



```

decide:
    clr    c                ;limpa o carry
    mov    A,R6
    subb   A,R7             ; A=R6-R7
    jc     negat
    mov    A,#06h          ; R6 é o maior
    ajmp   final
negat:
    mov    A,#07h          ; R7 é o maior
final:
    ret                    ;fim de Decide

```

```

;=====
; Subrotina delay
;=====

```

```

delay:
    mov    R0,#D0h
    mov    R1,#02h
loop:
    djnz   R0,loop
    djnz   R1,loop
    ret                    ;fim do Delay
fim:
    end

```

```

;-----
; Programa: Interrup.asm
; Simula um pedido de interrupção de Int 0. Uma vez reconhecido o pedido de
; interrupção a rotina de tratamento desta será alocada em 0050h. O tratamento
; básico consiste em acionar o port de I/O P3 (ligado teoricamente á um LED de
; alarme). Após o alarme, um temporizador desliga automaticamente o alarme
; aproximadamente 5 seg após o evento, independentemente do pedido de
; interrupção.
; Autor: Prof. Vargas
; Data:15:18 25/05/01
;-----

```

```

control    equ    81h    ;hab. int 0
prior      equ    01h    ;prioridade alta de int0
tipoin     equ    01h    ;ativo em transição

```

```

;=====
    org    0000h
    ljmp   inter
;=====

```

```

    org    0003h
    ljmp   intru        ;tratamento da int 0
;=====

```

```

    org    0030h
inter:
    mov    IE,#control
    mov    IP,#prior
    mov    TCON,#tipoin
    mov    SP,#0060h
    mov    P3,#00        ;limpa ports de I/O
loop:
    ljmp   loop          ;aguarda a int 0
;=====

```

```

    org    0050h
;-----
; Tratamento da interrupção INT 0
;-----

```

```

intru:
    push    dptr          ;preserva registros
    push    a
    push    R0
    setb    P3.4         ; liga Led 4 de P3

;-----[ Timer de 5 segundos ]-----
    mov     R0,#14       ; 14 decimal
    mov     Th1,#07h     ; programa High=07h
    mov     Tl1,#53h     ; programa Low=53h
rep:
    setb    TR1          ; liga timer 1
    jnb     TF1,$        ; se TF1=0 fica aqui
    clr     TF1          ; reseta overflow flag
    djnz   R0,rep        ; repete 14 ciclos
    clr     TR1          ; limpa timer 1
;-----[ Fim do timer ]-----
    clr     P3.4         ; apaga Led 4 de P3
    pop     R0           ; Restaura registros
    pop     a
    pop     dptr
    reti

    end

;-----
; Programa: Interrup.asm
; Simula um pedido de interrupção de Int 0. Uma vez reconhecido o pedido de
; interrupção a rotina de tratamento desta será alocada em 0050h. O tratamento
; básico consiste em acionar o port de I/O P3 (ligado teoricamente á um LED de
; alarme). Após o alarme, um temporizador desliga automaticamente o alarme
; aproximadamente 5 seg após o evento, independentemente do pedido de
; interrupção.
; Autor: Prof. Vargas
; Data:15:18 25/05/01
;-----
control equ    83h          ;hab. int 0 e Timer 0
prior    equ    01h        ;prioridade alta de int0
tipoin   equ    01h        ;ativo em transição

;=====
    org     0000h
    ljmp    inter
;=====
    org     0003h
    ljmp    intru          ;tratamento da int 0
;=====
    org     000Bh
    ljmp    tempo         ;tratamento de Tim 0
;=====
    org     0030h
inter:
    mov     IE,#control
    mov     IP,#prior
    mov     TCON,#tipoin
    mov     SP,#0060h
    mov     P3,#00        ;limpa ports de I/O
loop:
    ljmp    loop          ;aguarda a int 0

;=====
    org     0050h

```

```
;-----  
;Tratamento da interrupção INT 0  
;-----
```

```
intru:  
    push    dptr          ;preserva registros  
    push    a  
    setb    P3.4          ; liga Led 4 de P3  
    mov     Th0,#07h      ; programa High=07h  
    mov     Tl0,#53h      ; programa Low=53h  
    setb    TR0           ; liga timer 0  
    pop     a  
    pop     dptr  
    reti
```

```
;=====
```

```
    org     006Ah  
tempo:  
    push    dptr          ;preserva registros  
    push    a  
    clr     TR0           ; limpa timer 0  
    clr     P3.4          ; Apaga Led 4 de P3  
    pop     a  
    pop     dptr  
    reti
```

```
;=====
```

```
end
```

```
;-----  
; Programa Timer1.asm  
; este programa foi compilado á partir do exemplo 1 do livro  
; "8051" do engº Vidal P. Silva Jr. da Ed. Érica.  
;  
; Considerando um clock de 10,695 MHz (valor comercial) então teremos um pulso  
; de 0,0935 uS. Como o clock é dividido por 12 pelo 8051 para efeito de  
; relógio, então teremos um pulso de relógio á cada 1,122 uS. Assim, para termos  
; ~1 S, teremos de contar 891265,6 pulsos. O timer do 8051 conta até 65535,  
; portanto insuficiente. Assim utilizamos RAM auxiliar. Fazendo as contas:  
; 14 x 63660 ~ 891240 e portanto adotamos 14 vezes. Considerando que contaremos  
; 63660, então programamos o ; limite do Timer 1 para contar a partir de  
; 65535-63660 = 1875 que em Hexadecimal é 0753h que dividimos em 07h e 53h.  
;  
; Autor: Prof. Vargas  
; Data:17:18 27/04/01  
;-----
```

```
    org     00h  
    mov     R0,#14        ; 14 decimal  
    mov     Th1,#07h      ; programa High=07h  
    mov     Tl1,#53h      ; programa Low=53h  
rep:  
    setb    TR1           ; liga timer 1  
    jnb     TF1,$         ; enquanto TF1=0 fica aqui  
    clr     TF1           ; reseta flag de overflow  
    djnz    R0,rep        ; repete 14 ciclos  
    clr     TR1           ; limpa timer 1  
end
```