

5. Instruções de controle de fluxo

Instruções de **laço** (*loop*) e de **salto** (*jump*) permitem que:

- o programa "tome" certas decisões, alterando seu curso;
- certas partes de um programa sejam repetidas um número controlado de vezes.

Exemplo preliminar: exibição na tela de todos os caracteres ASCII.

```

TITLE  EXIBICAO DE CARACTERES ASCII
.MODEL  SMALL
.STACK  100H
.CODE
;inicializacao de alguns registradores
;
    MOV AH,2          ;funcao DOS para exibicao de caracter
    MOV CX,256        ;contador com o numero total de caracteres
    MOV DL,00H        ;DL inicializado com o primeiro ASCII
;
;definicao de um processo repetitivo de 256 vezes
;
PRINT_LOOP:
    INT 21H           ;exibir caracter na tela
    INC DL            ;incrementa o caracter ASCII
    DEC CX            ;decrementa o contador
    JNZ PRINT_LOOP ;continua exibindo enquanto CX nao for 0
;
;quando CX = 0, o programa quebra a sequencia do loop
;saida para o DOS
;
    MOV AH,4CH
    INT 21H

```

END

5.1 A instrução de comparação

CMP destino,fonte

CMP (Compare) compara os conteúdos destino e fonte, que podem ser:

- registrador e registrador
- registrador e uma posição de memória
- um número diretamente como operando fonte

Combinações legais de operandos:

Operando fonte	Operando destino	
	Registrador de dados	Posição de memória
Reg. de dados	sim	sim
Posição de memória	sim	não
Constante	sim	sim

CMP calcula a subtração: **(destino) - (fonte)**

Todos os Flags de Estado são afetados e o resultado **não é armazenado**.

Exemplos de instruções válidas:

CMP DX,BX ;compara os conteúdos de DX e BX

CMP AX,WORD1 ;compara o conteúdo do registrador AX com o da
;posição de memória WORD1

CMP AH,'A' ;compara o conteúdo de AH com o caracter ASCII 'A'

5.2 Saltos condicionais

JXXX rótulo_de_destino

- **XXX** é uma **condição** dependente de algum dos **Flags de Estado**
- Se a condição XXX é **verdadeira**:
 - a próxima instrução a ser executada é aquela definida pelo **rótulo_de_destino**;
 - a CPU ajusta o registrador **IP** para apontar para a posição de memória dada por rótulo_de_destino.
- Se a condição XXX é **falsa**:
 - a próxima instrução é aquela que imediatamente segue o salto.
- Faixa de endereçamento do rótulo_de_destino:
 - deve preceder JXXX não mais do que **126 bytes**;
 - deve suceder JXXX não mais do que **127 bytes**.
- Há três classes de saltos condicionais:
 - **saltos sinalizados**: dependem de um resultado na forma de um número sinalizado;
 - **saltos não-sinalizados**: dependem de um resultado na forma de um número não-sinalizado;
 - **saltos de Flag simples**: que dependem do *status* de algum dos Flags.
- A instrução **JXXX** não altera nenhum Flag.

Tipos de saltos condicionais

Símbolo		Descrição	Condições
Saltos sinalizados			
JG JNLE	ou	salto se maior do que salto se não menor do que ou igual a	OU ZF = 0 E SF = OF
JGE JNL	ou	salto se maior do que ou igual a salto se não menor do que	OU SF = OF
JL JNGE	ou	salto se menor do que salto se não maior do que ou igual a	OU SF ≠ OF
JLE JNG	ou	salto se menor do que ou igual a salto se não maior do que	OU ZF = 1 OU SF ≠ OF
Saltos não-sinalizados			
JA JNBE	ou	salto se acima de OU salto se não abaixo de ou igual a	CF = 0 E ZF = 0
JAE JNB	ou	salto se acima de ou igual a salto se não abaixo de	OU CF = 0
JB JNAE	ou	salto se abaixo de OU salto se não acima de ou igual a	CF = 1
JBE JNA	ou	salto se abaixo de ou igual a salto se não acima de	OU CF = 1 OU ZF = 1
Saltos de Flag simples			
JE JZ	ou	salto se igual OU salto se igual a zero	ZF = 1
JNE JNZ	ou	salto se não igual salto se não igual a zero	OU ZF = 0
JC		salto se há VAI-UM (<i>carry</i>)	CF = 1
JNC		salto se não há VAI-UM (<i>not carry</i>)	CF = 0
JO		salto se há <i>overflow</i>	OF = 1
JNO		salto se não há <i>overflow</i>	OF = 0
JS		salto se o sinal é negativo	SF = 1

JNS	salto se o sinal é não-negativo (+)	SF = 0
JP ou JPE	salto se a paridade é PAR (<i>even</i>)	PF = 1
JNP ou JPO	salto se a paridade é IMPAR (<i>odd</i>)	PF = 0

Diferença entre Saltos sinalizados e não-sinalizados

a) Trecho de programa que supõe **quantidades não-sinalizadas**:

;supondo que AX contem 7FFFh e BX contem 8000h

```
...
CMP AX,BX
```

```
JA PT2 ;o salto não ocorre porque 7FFFh < 8000h
```

```
...
```

```
...
```

```
PT2: MOV ... ;continuação do programa
```

b) Trecho de programa que supõe **quantidades sinalizadas**:

;supondo que AX contem 7FFFh e BX contem 8000h

```
...
CMP AX,BX
```

```
JG PT2 ;o salto ocorre porque 7FFFh (+) > 8000h (-)
```

```
...
```

```
...
```

```
PT2: MOV ... ;continuação do programa
```

Exemplo: Supondo que AX e BX contenham números sinalizados, escreva um trecho de programa que coloque o maior deles em CX.

```
...
MOV CX,AX
CMP AX,BX
```

```
;AX já é pressuposto ser o maior deles
```


5.3 Salto incondicional

JMP rótulo_de_destino

- **Rótulo_de_destino** é uma posição no programa, no mesmo segmento de código onde JMP aparece
- Não há restrição de faixa de endereçamento como em JXXX
- JMP pode ajudar a solucionar o problema de faixa de endereçamento das instruções JXXX

Exemplo: trecho utilizando JMP e JXXX

```

TOPO:      ...
           ;mais do que 126 bytes de instruções: limitação para JXXX
           ...
           ;corpo de algum laço
           ;
           DEC CX
           JNZ ABAIXO
           JMP CONTINUA
ABAIXO:    JMP TOPO
CONTINUA:  MOV ...           ;programa continua
           ...

```

Exercícios sugeridos:

- 1) Escreva um programa que apresente uma '?', leia em seguida duas letras maiúsculas e exiba-as na próxima linha, em ordem alfabética.
- 2) Modifique o programa de exibição de caracteres ASCII (pág. 5.1), de forma a exibir 16 caracteres por linha separados por espaços em branco.
- 3) Escreva um programa que pergunte ao usuário para teclar um dígito hexadecimal, exiba na próxima linha o seu valor decimal e pergunte ao usuário se deseja continuar a utilizar o programa: se for digitado S (sim), o programa se repete desde o começo; se for digitado outro caracter, o programa termina. Teste se o dígito hexa está na faixa de valores correta. Se não estiver, exiba uma mensagem para o usuário tentar de novo.
- 4) Crie um trecho de código modificando o programa do exercício (3) acima, tal que se o usuário falhar em entrar com um dígito hexa na faixa correta mais do que três tentativas, o programa exibe uma mensagem adequada e termina.
- 5) Crie um programa que implemente uma multiplicação por meio de somas sucessivas. Faça as considerações que achar necessárias.
- 6) Crie um programa que implemente uma divisão por meio de subtrações sucessivas, exibindo o quociente e o resto com mensagens adequadas. Faça as considerações que achar necessárias.

5.4 Algumas estruturas de linguagens de alto nível

1) Estrutura IF - THEN - ELSE

Em linguagem de alto nível:

```

IF    (condição)
    THEN    (seqüência 1)
    ELSE    (seqüência 2)
END_IF

```

Exemplo: Suponha que AL e BL contêm dois caracteres ASCII; exiba aquele que seja o primeiro em ordem alfabética.

Em linguagem de alto nível:

```

IF    AL (menor ou igual a) BL
    THEN    (exibir AL)
    ELSE    (exibir BL)
END_IF

```

Em linguagem montadora:

```

      ....
;if AL menor ou igual a BL
      MOV AH,2h
      CMP AL,BL
      JA  TROCA

;then
      MOV DL,AL
      INT 21h
      JMP FIM

;else
      TROCA: MOV DL, BL
            INT 21h

;end_if

```

FIM:

LOOP rótulo_de_destino

- Tem como contador implícito o registrador CX, que deve ser inicializado antes do laço.
- Salta para rótulo_de_destino enquanto o conteúdo de CX não for zero.
- Quando CX = 0, a próxima instrução após LOOP será executada.
- CX é decrementado automaticamente quando LOOP é executada.
- Nenhum FLAG é afetado.

Exemplo de instruções válidas:

```
LOOP    PT1
```

```
LOOP    TOPO
```

```
LOOP    RETORNO
```

Obs: são equivalentes as seqüências

```

                MOV CX, (valor_inicial)
TOPO:          ...
                ...
                LOOP TOPO

```

```

                MOV CX, (valor_inicial)
TOPO:          ...
                ...
                DEC CX
                JNZ TOPO

```

2) FOR loop

Em linguagem de alto nível:

```

FOR   (número_de_vezes)   DO
      (seqüência de instruções)

END_FOR

```

Exemplo: Exiba uma seqüência de 80 asteriscos no monitor de vídeo.

Em linguagem de alto nível:

```

FOR   (80 vezes)   DO
      (exibir " * ")
END_FOR

```

Em linguagem montadora:

```

...
;for 80 vezes
                MOV CX,80d
                MOV AH,2h
                MOV DL," * "
;do
    TOPO:      INT 21h
                LOOP TOPO
;end_for
...

```

Exercício: modifique o programa que exibe todos os caracteres ASCII (página 5.1), utilizando apenas a instrução LOOP.

3) WHILE loop

Em linguagem de alto nível:

```

WHILE   (condição_verdadeira)   DO
      (seqüência de instruções)

END_WHILE

```

Exemplo: Ler caracteres ASCII do teclado, contando sua quantidade, até que o caracter **Carriage Return** (CR) apareça.

Em linguagem de alto nível:

```

WHILE   (caracter diferente de CR)   DO
      (ler caracter do teclado e armazená-lo)
      (contador = contador +1)
END_WHILE

```

Em linguagem montadora:

```

      ...
      MOV DX,0h           ;inicialização
      MOV AH,1h
      INT 21h
;while
LOOP:  CMP AL,0Dh        ;é o caracter CR?
      JE  FIM           ;salto quando caracter é igual a CR
      MOV AL, (algum lugar) ;salvando o caracter lido
      INC DX           ;conta número de caracteres
      INT 21h
      JMP LOOP         ;fecha o loop WHILE
;end_while

```

FIM:

4) REPEAT loop

Em linguagem de alto nível:

REPEAT

(seqüência de instruções)

UNTIL (condição_verdadeira)

Exemplo: Ler caracteres ASCII do teclado, contando sua quantidade, até que o caracter **Carriage Return** (CR) apareça.

Em linguagem de alto nível:

REPEAT

(ler caracter do teclado e armazená-lo)

(contador = contador + 1)

UNTIL (caracter igual a CR)

Em linguagem montadora:

```

...
MOV DX,0h           ;inicialização
MOV AH,1h
;repeat
LOOP:  INT 21h
        MOV AL, (algun lugar) ;salvando o caracter lido
        INC DX                ;conta número de caracteres
        CMP AL,0Dh           ;é o caracter CR?
        JNE LOOP             ;salto enquanto caracter não é CR
;until
...

```