ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Instruções de multiplicação

- **Instruções de multiplicação**

  MUL fonte

  IMUL fonte

  - **MUL (*multiply*) → usada com números em representação não-sinalizada;**
  - **IMUL (*integer multiply*) → usada com números sinalizados.**

- **Multiplicação com números em formato *byte*:**
  - **registradores de 8 bits e variáveis de tipo DB;**
  - **segundo operando é assumido em AL;**
  - **resultado (destino) pode atingir 16 bits e se encontra em AX.**

- **Multiplicação com números em formato *word*:**
  - **registradores de 16 bits e variáveis de tipo DW**
  - **segundo operando é assumido em AX**
  - **resultado pode atingir 32 bits (tamanho *doubleword*) e se encontra em:**
    - **DX → 16 bits mais significativos (*high word*)**
    - **AX → 16 bits menos significativos (*low word*)**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Instruções de multiplicação

Observação:

- **Para números positivos (MSB = 0), MUL e IMUL dão o mesmo resultado**
- **Flags afetados:**
  - **SF, ZF, AF, PF -> indefinidos;**
  - **após MUL, CF/OF (ambos) = 0 , se a metade superior do resultado é 0;**
    = 1 , caso contrário;
  - **após IMUL, CF/OF (ambos) = 0 , se a metade superior do resultado for extensão do sinal da metade inferior;**
    = 1 , caso contrário
- **Exemplos de casos de multiplicação:**

Suponha que AX contenha 0FFF h:

antes: AX = 0FFF h = 0000 1111 1111 1111 h = 4095 ou + 4095

	Resultado				
Instrução	Decimal	Hexadecimal	DX	AX	CF/OF
MUL AX					
IMUL AX					

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Instruções de multiplicação

Observação:

- **Para números positivos (MSB = 0), MUL e IMUL dão o mesmo resultado**
- **Flags afetados:**
  - **SF, ZF, AF, PF -> indefinidos;**
  - **após MUL, CF/OF (ambos) = 0 , se a metade superior do resultado é 0;**
    = 1 , caso contrário;
  - **após IMUL, CF/OF (ambos) = 0 , se a metade superior do resultado for extensão do sinal da metade inferior;**
    = 1 , caso contrário
- **Exemplos de casos de multiplicação:**

Suponha que AX contenha 0FFF h:

antes: AX = 0FFF h = 0000 1111 1111 1111 h = 4095 ou + 4095

	Resultado				
Instrução	Decimal	Hexadecimal	DX	AX	CF/OF
MUL AX	16769025	00FFE001h	00FFh	E001h	1
IMUL AX	16769025	00FFE001h	00FFh	E001h	1

Instruções de multiplicação

- **Exemplos:**
  Dados os registradores e respectivos conteúdos, de o resultado e o valor de CF e OF após MUL e após IMUL (resultado e CF e OF):

1) Suponha que AX contenha 0001h BX contenha FFFFh:

2) Suponha que AL contenha 80h BL contenha FFh:

Instruções de multiplicação

- **Exemplos de casos de multiplicação:**

1) Suponha que AX contenha 0001h BX contenha FFFFh:

antes: AX = 0001 h = 0000 0000 0000 0001b = 1 ou +1

BX = 0FFFF h = 1111 1111 1111 1111b = 65535 ou -1

	Resultado				
Instrução	Decimal	Hexadecimal	DX	AX	CF/OF
MUL BX	65535	0000FFFFh	0000h	FFFFh	0
IMUL BX	-1	FFFFFFFFh	FFFFh	FFFFh	0

2) Suponha que AL contenha 80h BL contenha FFh:

antes: AL = 80 h = 1000 0000 b = 128 ou -128

BL = FF h = 1111 1111 b = 255 ou -1

	Resultado				
Instrução	Decimal	Hexadecimal	AH	AL	CF/OF
MUL BL	326407	7F80h	7Fh	80h	1
IMUL BL	128	0080h	00h	80h	1

Instruções de divisão

- **Instruções de divisão**

DIV fonte
IDIV fonte

  - **DIV (*divide*) → usada com números em representação não-sinalizada**
  - **IDIV *(integer* divide) → usada com números sinalizados**
    - **fonte deve ser considerado como divisor (não pode ser uma constante)**

- **Divisão com números em formato *byte*:**
  - **o divisor é assumido ser de 8 bits (1 byte)**
  - **o dividendo é assumido estar em AX (16 bits)**
  - **após a execução: o quociente de 8 bits estará em AL o resto de 8 bits estará em AH**

- **Divisão com números em formato *word*:**
  - **o divisor é assumido ser de 16 bits (1 word)**
  - **o dividendo é assumido ser de 32 bits:**
    - **DX → 16 bits mais significativos do dividendo (*high word*)**
    - **AX → 16 bits menos significativos do dividendo (*low word*)**
    - **após a execução: o quociente de 16 bits estará em AX o resto de 16 bits estará em DX**

- **Para números positivos (MSB = 0), DIV e IDIV fornecem o mesmo resultado.**

Instruções de divisão

- **Flags afetados: todos ficam indefinidos**
- **Em divisão de números em representação sinalizada, o resto possui o mesmo sinal do dividendo.**
- **Exemplos de casos de divisão:**

Suponha que DX e AX contenham 0000h e 0005h, e BX contenha FFFEh:

antes: DX : AX = 0000 0005 h = 5 ou + 5

BX = FFFE h = 65534 ou - 2

	QUOCIENTE			
Instrução	Quociente	Resto	AX	DX
DIV BX				
IDIV BX				

Instruções de divisão

- **Flags afetados: todos ficam indefinidos**
- **Em divisão de números em representação sinalizada, o resto possui o mesmo sinal do dividendo.**
- **Exemplos de casos de divisão:**

Suponha que DX e AX contenham 0000h e 0005h, e BX contenha FFFEh:

antes: DX : AX = 0000 0005 h = 5 ou + 5

BX = FFFE h = 65534 ou - 2

	QUOCIENTE			
Instrução	Quociente	Resto	AX	DX
DIV BX	0	5	0000h	0005h
IDIV BX	-2	1	0FFFEh	0001H

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Instruções de divisão

- **Exemplos de casos de divisão:**

2) Suponha que AX contenha 0005h e BL contenha FFh:
antes: AX = 0005 h = 5 ou + 5
BL = FF h = 256 ou - 1

	QUOCIENTE			
Instrução	Quociente	Resto	AL	AH
DIV BL				
IDIV BL				

3) Suponha que AX contenha 00FB h e BL contenha FF h:
antes: AX = 00FB h = 251 ou + 251
BL = FF h = 256 ou - 1

	QUOCIENTE			
Instrução	Quociente	Resto	AL	AH
DIV BX				
IDIV BX				

Instruções de divisão

- **Exemplos de casos de divisão:**

2) Suponha que AX contenha 0005h e BL contenha FFh:
antes: AX = 0005 h = 5 ou + 5
BL = FF h = 256 ou - 1

	QUOCIENTE			
Instrução	Quociente	Resto	AL	AH
DIV BL	0	5	00h	05h
IDIV BL	-5	0	0FBh	00h

3) Suponha que AX contenha 00FB h e BL contenha FF h:
antes: AX = 00FB h = 251 ou + 251
BL = FF h = 255 ou - 1

	QUOCIENTE			
Instrução	Quociente	Resto	AL	AH
DIV BX	0	251	00h	0FBh
IDIV BX	-251	0	-	-

*** → como -251 não cabe em AL (8 bits) -> ocorre *DIVIDE OVERFLOW* que é situação de erro que faz com que o programa termine.**

Extensão do sinal do dividendo

- **Em operações em formato word: Caso o dividendo de uma divisão (composto de DX : AX) ocupe apenas AX, DX deve ser preparado, pois é sempre considerado.**
  - **Em DIV → DX deve ser zerado**
  - **EM IDIV → DX deve ter a extensão de sinal de AX**

CWD

- **Instrução sem operandos (zero operandos) que converte *word* para *doubleword* e estende o sinal de AX para DX. Deve ser usada com IDIV.**

- **Em operações em formato byte: Caso o dividendo de uma divisão (composto por AX) ocupe apenas AL, AH deve ser preparado, pois é sempre considerado.**
  - **Em DIV → AH deve ser zerado**
  - **EM IDIV → AH deve ter a extensão de sinal de AL**

CBW

- **Instrução sem operandos (zero operandos) que converte *byte* para *word* e estende o sinal de AL para AH. Deve ser usada com IDIV**

Exemplos de divisões com ajuste de extensão:

1) Crie um trecho de programa que divida -1250 por 7.

```
...
MOV  AX, -1250     ;AX recebe o dividendo
CWD                ;estende o sinal de AX para DX
MOV  BX,7          ;BX recebe o divisor
IDIV  BX           ;executa a divisão
                   ;após a execução, AX recebe o
                   ;quociente e DX recebe o resto
...
```

2) Crie um trecho de programa que divida a variável sinalizada XBYTE por -7.

```
...
MOV  AL,XBYTE      ;AL recebe o dividendo
CBW                ;estende o sinal (eventual) de
                   ;AL para AH
MOV BL, -7         ;BL recebe o divisor
IDIV BL            ;executa a divisão
                   ;após a execução, AL recebe o
                   ;quociente e AH recebe o resto
...
```

Obs.: Não há efeito de CBW e CWD sobre os FLAGS.

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplos de aplicação

- **Entrada de números decimais:**
  - **cadeia de caracteres números de 0 a 9, fornecidos pelo teclado;**
  - **CR é o marcador de fim de cadeia;**
  - **AX é assumido como registrador de armazenamento;**
  - **valores decimais permitidos na faixa de - 32768 a + 32767;**
  - **sinal negativo deve ser apresentado.**

- **Algoritmo básico em linguagem de alto nível:**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplos de aplicação

- **Entrada de números decimais:**
  - **cadeia de caracteres números de 0 a 9, fornecidos pelo teclado;**
  - **CR é o marcador de fim de cadeia;**
  - **AX é assumido como registrador de armazenamento;**
  - **valores decimais permitidos na faixa de -32768 a +32767;**
  - **sinal negativo deve ser apresentado.**

- **Algoritmo básico em linguagem de alto nível:**

```
total = 0
negativo = FALSO
ler um caractere
CASE  caractere  IS
      ' - '        :  negativo = VERDADEIRO   e   ler um caractere
      ' + '        :  ler um caractere
END_CASE
REPEAT
      converter caractere em valor binário
      total  = 10 x total + valor binário
      ler um caractere
UNTIL   caractere é um carriage return (CR)
IF  negativo = VERDADEIRO
      THEN  total = - (total)
END_IF
```

Obs.: O *loop* do tipo CASE pode ser entendido como um IF múltiplo, que testa simultaneamente os vários "casos": se algum deles for verdadeiro, executa as instruções relacionadas; se todos os "casos" forem falsos, não executa nada e vai para o fim do CASE.

123

CX = 0

31h → 1

CX *10 + AL = CX → 1

32h → 2

CX *10 + AL = CX → 12

33h → 3

CX *10 + AL = CX → 123

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplos de aplicação

- **Saída de números decimais:**
  - **AX é assumido como registrador de armazenamento;**
  - **valores decimais na faixa de - 32768 a + 32767;**
  - **exibe sinal negativo, se o conteúdo de AX for negativo;**
  - ***string* de caracteres números de 0 a 9, exibidos no monitor de vídeo.**

- **Algoritmo básico em linguagem de alto nível:**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplos de aplicação

- **Saída de números decimais:**
  - **AX é assumido como registrador de armazenamento;**
  - **valores decimais na faixa de - 32768 a + 32767;**
  - **exibe sinal negativo, se o conteúdo de AX for negativo;**
  - ***string* de caracteres números de 0 a 9, exibidos no monitor de vídeo.**

- **Algoritmo básico em linguagem de alto nível:**

```
IF AX < 0
      THEN    exibe um sinal de menos substitui-se AX pelo seu complemento de 2
END_IF
contador = 0
REPEAT
      dividir quociente por 10
      colocar o resto na pilha
      contador = contador + 1
UNTIL   quociente = 0
FOR  contador vezes  DO
      retirar um resto (número) da pilha
      converter para caracter ASCII
      exibir o caracter no monitor
END_FOR
```

- **Ideia básica da técnica de decomposição decimal do número em AX:**

		Pilha
24618	**dividido por 10 = 2461**	**com resto 8 → 0008h**
2461	**dividido por 10 = 246**	**com resto 1 → 0001h**
246	**dividido por 10 = 24**	**com resto 6 → 0006h**
24	**dividido por 10 = 2**	**com resto 4 → 0004h**
2	**dividido por 10 = 0**	**com resto 2 → 0002h ← Topo**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

- **Entrada de números decimais:**
  - **cadeia de caracteres números de 0 a 9, fornecidos pelo teclado;**
  - **CR é o marcador de fim de cadeia;**
  - **AX é assumido como registrador de armazenamento;**
  - **valores decimais permitidos na faixa de -32768 a +32767;**
  - **sinal negativo deve ser apresentado.**

- **Algoritmo básico em linguagem de alto nível:**

```
total = 0
negativo = FALSO
ler um caractere
CASE  caractere  IS
      ' - '        :   negativo =
      VERDADEIRO   e   ler um
      caractere
      ' + '        :   ler um caractere
END_CASE
REPEAT
      converter caractere em valor
      binário
      total  = 10 x total + valor binário
      ler um caractere
UNTIL   caractere é um carriage
      return (CR)
IF  negativo = VERDADEIRO
      THEN  total = - (total)
END_IF
```

- **Saída de números decimais:**
  - **AX é assumido como registrador de armazenamento;**
  - **valores decimais na faixa de - 32768 a + 32767;**
  - **exibe sinal negativo, se o conteúdo de AX for negativo;**
  - ***string* de caracteres números de 0 a 9, exibidos no monitor de vídeo.**

- **Algoritmo básico em linguagem de alto nível:**

```
IF AX < 0
      THEN    exibe um sinal de menos
      substitui-se AX pelo seu
      complemento de 2
END_IF
contador = 0
REPEAT
      dividir quociente por 10
      colocar o resto na pilha
      contador = contador + 1
UNTIL   quociente = 0
FOR  contador vezes  DO
      retirar um resto (número) da
      pilha
      converter para caractere ASCII
      exibir o caractere no monitor
END_FOR
```

ENTRADA DECIMAL

```
ENTDEC PROC
; lê um numero entre -32768 A 32767
; entrada nenhuma
; saída numero em AX

   PUSH BX
   PUSH CX
   PUSH DX
@INICIO:
; imprime prompt ?
   MOV AH,2
   MOV DL,'?'
   INT 21H     ; imprime ?
;  total = 0
   XOR BX,BX

; negativo = falso
   XOR CX,CX

; le caractere
   MOV AH,1
   INT 21H

; case caractere lido eh?
   CMP AL,'-'
   JE @NEGT
   CMP AL,'+'
   JE @POST
   JMP @REP2
```

```
@NEGT:
   MOV CX,1
@POST:
   INT 21H
;end case
@REP2:
; if caractere esta entre 0 e 9
   CMP AL, '0'
   JNGE @NODIG
   CMP AL, '9'
   JNLE @NODIG
; converte caractere em digito
   AND AX,000FH
   PUSH AX
; total = total X 10 + digito
   MOV AX,10
   MUL BX   ; AX = total X 10
   POP BX
   ADD BX,AX ; total - total X 10 +
digito
; le caractere
   MOV AH,1
   INT 21H
   CMP AL,13  ;CR ?
     JNE @REP2    ; não, continua
```

```
; ate CR
   MOV AX,BX   ; guarda numero em AX
; se negativo
   OR CX,CX    ; negativo ?
   JE @SAI     ; sim, sai
; entao
   NEG AX
; end if
@SAI:
   POP DX    ; restaura registradores
   POP CX
   POP BX

   RET   ; retorna
@NODIG:
; se caractere ilegal
   MOV AH,2
   MOV DL, 0DH
   INT 21H
   MOV DL, 0AH
   INT 21H
   JMP @INICIO
ENTDEC ENDP
```

```
SAIDEC PROC
; imprime numero decimal sinalizado em AX
;entrada AX
;sa?da nenhuma


   PUSH AX
   PUSH BX
   PUSH CX
   PUSH DX
; if AX < 0
OR AX,AX      ; AX < 0 ?
   JGE @END_IF1
;then
   PUSH AX     ;salva o numero
   MOV DL, '-'
   MOV AH,2
   INT 21H         ; imprime -
   POP AX          ; restaura numero
   NEG AX

; digitos decimais
@END_IF1:
   XOR CX,CX      ; contador de d?gitos
   MOV BX,10      ; divisor
```

```
@REP1:
   XOR DX,DX      ; prepara parte alta do dividendo
   DIV BX         ; AX = quociente   DX = resto
   PUSH DX        ; salva resto na pilha
   INC CX         ; contador = contador +1

;until
   OR AX,AX       ; quociente = 0?
   JNE @REP1      ; nao, continua

; converte digito em caractere
   MOV AH,2

; for contador vezes
@IMP_LOOP:
   POP DX         ; digito em DL
   OR DL,30H
   INT 21H
   LOOP @IMP_LOOP
; fim do for

   POP DX
   POP CX
   POP BX
   POP AX
   RET
SAIDEC ENDP
```