Linguagem de Montagem

Instruções Aritméticas e Lógicas

Parte 02

Aula 10

Edmar André Bellorini

Ano letivo 2016

Introdução

- Aula 05

```
ADD r/m, r/m/i
ADC r/m, r/m/i
SUB r/m, r/m/i
SBB r/m, r/m/i
OR  r/m, r/m/i
AND r/m, r/m/i
XOR r/m, r/m/i
NOT r/m
```

Introdução

- Nesta aula
  - Operações Lógicas

```
NEG r/m
SHL r/m, i
SHR r/m, i
SAR r/m, i
SAL r/m, i
ROL r/m, i
ROR r/m, i
```

  - Operações Aritméticas de Inteiros

```
MUL  r/m
IMUL r/m
DIV  r/m
IDIV r/m
```

Operações Lógicas de Deslocamento

- Negação

```
NEG r/m
```

- Deslocamento Lógico

```
SHL r/m, i
SHR r/m, i
```

- Deslocamento Aritmético

```
SAL r/m, i
SAR r/m, i
```

- Rotação

```
ROL r/m, i
ROR r/m, i
```

Negação

- NEG → *Negate*
  - Aplica complemento de 2 no operando

```
NEG r/m 8/16/32/64
```

  - Exemplo

```
MOV AX, 0xEE01 ; -4607
NEG AX         ; +4607
```

```
0xEE01 = 1110 1110 0000 0001 ; Compl. de 1
         0001 0001 1111 1110 ; + 1
         0001 0001 1111 1111 = 0x11FF
```

Exemplo: a10e01.asm (Negação)

```
2       ...
3   section .text
4       global _start
5
6   _start:
7       mov ax, 0xEE01
8   negacao:
9       neg ax
10
11  fim:
12      mov eax, 1
13      mov ebx, 0
14      int 0x80
```

Deslocamento Lógico à Esquerda

- SHL → *Logical Left Shift*
  - Aplica deslocamento lógico em *dest*, *src* bits à esquerda

```
    _start:
    mov ax, 0xEE01
desloc1:
    shl ax, 1 ; 0xDC01
desloc2:
    shl ax, 4 ; 0xC020
```

  - CF = bit expurgado
  - bit de entrada à direita é '0'
    usado em aritmética **não** sinalizada

Deslocamento Lógico à Direita

- SHR → *Logical Right Shift*
  - Aplica deslocamento lógico em *dest*, *src* bits à direita

```
    _start:
    mov bx, 0xEE01
desloc1:
    shr bx, 1 ; 0x7700
desloc2:
    shr bx, 4 ; 0x0770
```

  - CF = bit expurgado
  - bit de entrada à esquerda é '0'
    usado em aritmética **não** sinalizada

Exemplo: a10e02.asm (Desl. Lógico)

```
2       ...
3   section .text
4       global _start
5
6   _start:
7       mov ax, 0xEE01
8       mov bx, 0xEE01
9   desloc1:
10      shl ax, 1
11      shr bx, 1
12  desloc2:
13      shl ax, 4
14      shr bx, 4
15  fim:
16      ...
```

Deslocamento Aritmético à Esquerda

- SAL → *Arithmetic Left Shift*
  - Aplica deslocamento aritmético em *dest*, *src* bits à esquerda

```
    _start:
    mov bx, 0xEE01
desloc1:
    sal bx, 1 ; 0xDC01
desloc2:
    sal bx, 4 ; 0xC020
```

  - CF = bit expurgado
  - bit de entrada à direita é '0'
    bit menos significativo não altera sinal do número
    SAL é sinônimo de SHL
    usado em aritmética **sinalizada**

Deslocamento Aritmético à Direita

- SAR → *Arithmetic Right Shift*
  - Aplica deslocamento aritmético em *dest*, *src* bits à direita

```
    _start:
    mov bx, 0xEE01
desloc1:
    sar bx, 1 ; 0xF700
desloc2:
    sar bx, 4 ; 0xFF00
```

  - CF = bit expurgado
  - bit de entrada à esquerda é mantido
    bit mais significativo **altera** sinal do número
    usado em aritmética **sinalizada**

Exemplo: a10e03.asm (Desl. Aritmético)

```
2       ...
3   section .text
4       global _start
5
6   _start:
7       mov ax, 0xEE01
8       mov bx, 0xEE01
9   desloc1:
10      sal ax, 1
11      sar bx, 1
12  desloc2:
13      sal ax, 4
14      sar bx, 4
15  fim:
16      ...
```

Rotação à Esquerda

- ROL → *Rotate Left*
  - Aplica deslocamento rotacional em *dest*, *src* bits à esquerda

```
    _start:
    mov ax, 0xEE01
desloc1:
    rol ax, 1 ; 0xDC03
desloc2:
    rol ax, 4 ; 0xC03D
```

  - CF = bit expurgado
  - bit de entrada à direita é bit expurgado

Rotação à Direita

- ROR → *Rotate Right*
  - Aplica deslocamento rotacional em *dest*, *src* bits à direita

```
    _start:
    mov bx, 0xEE01
desloc1:
    ror bx, 1 ; 0xF700
desloc2:
    ror bx, 4 ; 0x0F70
```

  - bit de entrada à esquerda é bit expurgado

Exemplo: a10e04.asm (Desl. Rotacional)

```
2     ...
3  section .text
4     global _start
5
6  _start:
7     mov ax, 0xEE01
8     mov bx, 0xEE01
9  desloc1:
10    rol ax, 1 ; 0xDC03
11    ror bx, 1 ; 0xF700
12 desloc2:
13    rol ax, 4 ; 0xC03D
14    ror bx, 4 ; 0x0F70
15 fim:
16    ...
```

Operações Aritméticas de Inteiros

- Multiplicação

```
MUL  r/m
IMUL r/m
```

- Divisão

```
DIV  r/m
IDIV r/m
```

Multiplicação de Inteiros

- MUL → Multiplicação de inteiros **não** sinalizados
- IMUL → Multiplicação de inteiros **sinalizados**

```
 MUL r/m
IMUL r/m
```

- Operandos
  - Implícito: EAX
  - Explícito: $r32/m32$
  - Resultado: EDX:EAX
- Operação:
  EDX:EAX = EAX * $r32/m32$

Divisão de Inteiros

- DIV → Divisão de inteiros **não** sinalizados
- IDIV → Divisão de inteiros **sinalizados**

```
 DIV r/m
IDIV r/m
```

- Operandos
  - Implícito: EDX:EAX
  - Explícito: $r32/m32$
  - Resultado: EAX (Quociente) e EDX (Resto)
- Operação:
  EAX = EDX:EAX $\div$ $r32/m32$ e Resto é armazenado em EDX

Exemplo: a10e05.asm (Mul e Div de Inteiros un/signed)

```
16      ...
17  multiplicacao0:
18      ; EAX <- multiplicando1 = 50
19      ; EDX:EAX <- EAX * multiplicador
20      mov eax,  [multiplicando1]
21      mul dword [multiplicador1] ; pode ser memoria
22      ...
```

```
29      ...
30  divisao0:
31      ; EAX <- dividendo1 = 100
32      ; EDX:EAX <- EDX:EAX / divisor
33      xor edx,  edx ; bytes altos
34      mov eax,  [dividendo1] ; bytes baixos
35      div dword [divisor1]  ; pode ser memoria
36      ...
```

Multiplicação e Divisão de Inteiros - Variações

- Operações em 32 bits
  - Multiplicação
    EDX:EAX = EAX * r32/m32
  - Divisão
    EAX = EDX:EAX ÷ $r32/m32$ e Resto em EDX
- Operações em 16 bits
  - Multiplicação
    DX:AX = AX * r16/m16
  - Divisão
    AX = DX:AX ÷ $r16/m16$ e Resto em DX
- Operações em 64 bits
  - Multiplicação
    RDX:RAX = RAX * r64/m64
  - Divisão
    RAX = RDX:RAX ÷ $r64/m64$ e Resto em RDX

Exercícios de Fixação

- EF1001: Elabore um código que crie a sequência dos 20 primeiros números de uma progressão geométrica com razão 3
  - Tamanho do Inteiro deve ser de 4 bytes
  - A sequência deve ser armazenada em um vetor não-inicializado de 20 posições
  - O primeiro termo é 3
    - Dica: O último termo é 3.486.784.401
    - Pergunta: Usa-se MUL ou IMUL? Por que?
- EF1002: Semelhante o EF1001, porém com razão -2
  - Tamanho do Inteiro deve ser de 4 bytes
  - A sequência deve ser armazenada em um vetor não-inicializado de 20 posições
  - O primeiro termo é -2
    - Dica: O último termo é 1.048.576
    - Pergunta: Usa-se MUL ou IMUL? Por que?

Exercícios de Fixação

- EF1003: Faça um levantamento dos sinais resultantes de uma divisão sinalizada.
  - Crie uma tabela contendo os sinais do Quociente e Resto para divisões com números positivos e negativos
    - Ao todo são 4 combinações possíveis
    - Verifique via GDB

Desafio LM Parte 02 - Faz parte dos Exercícios de Fixação

- D1001: Crie um código que:
  - Permita a entrada pelo teclado de um texto no formato:
    Entrada: $[-]^{*1}[0-9]^{+5}[+|-|*|/][-]^{*1}[0-9]^{+5}$
    - Exemplos:
      Entrada: 9+5
      Entrada: -956-152
      Entrada: 14995*-25615
      Entrada: -540/-150
  - Tratamento de entrada
    - É necessário particionar a string em 3 variáveis não-inicializadas:
      strOperando1, Operador, strOperando2
    - É necessário converter as strings strOperando1 e strOperando2 em números:
      operando1 = conversãoParaNúmero(strOperando1) e
      operando2 = conversãoParaNúmero(strOperando2)

Desafio LM Parte 02 - Continuação

- D1001
  - Solucionar o cálculo solicitado
    - resultado = operando1 Operador operando2
  - Converter resultado numérico para string
    - strResultado = numParaStr(resultado)
  - Apresentar resultado
    - Exemplos:
      Entrada: 9+5
      Resultado : 14
      Entrada: -956-152
      Resultado : -1108
      Entrada: 14995*-25615
      Resultado : -384096925
      Entrada: -540/-150
      Resultado : $Q = 3, R = -90$

Desafio LM Parte 02 - Final

- D1001
  - A execução do programa é única
    - Não requer *endless parrot code*
  - Somente entrega de relatório
    - Contabiliza nota apenas para prática 10
      Relatório deve conter EF1001, EF1002, EF1003 e D1001
  - OU, Somente defesa
    - Contabiliza nota das práticas 08, 09 e 10
- Importante:
  - Desta vez não será permitido o uso de funções prontas
    - Deve ser usado os conceitos desta aula prática

Relatório

- Somente relatório
  - O modelo de relatório para a disciplina de LM está disponível em anexo da Aula 01
    - Arquivo **modeloRelatorioLM.odt**
    - A data do relatório é a data de entrega (ver moodle)
  - Somente serão aceitos os relatórios em formato .pdf com nome do arquivo seguindo o padrão:
    TY.PXX.nome.sobrenome.pdf
    - TY é o número da turma prática (1, 2, 3 ou 4)
    - PXX é o número da prática, neste caso: P10
  - Ex.: aluno Warren Robinett da turma prática 9 (de 1979):
    - T9.P10.Warren.Robinett.pdf