Aula 8 - Aritmética Inteira
Arquitetura de Computadores I

Prof. MSC. Wagner Guimarães Al-Alam

Universidade Federal do Ceará
Campus de Quixadá

2020

Sumário

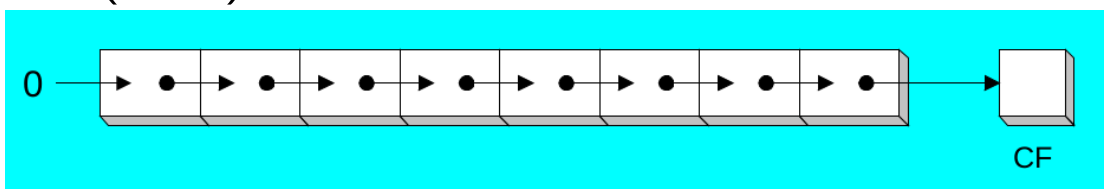
- Instruções Shift e Rotate
- Aplicação das Instruções Shift e Rotate
- Instruções de Multiplicação e Divisão
- Soma e Substração Estendida
- ASCII e Aritmética Unpacked Decimal
- Aritmética Packed Decimal

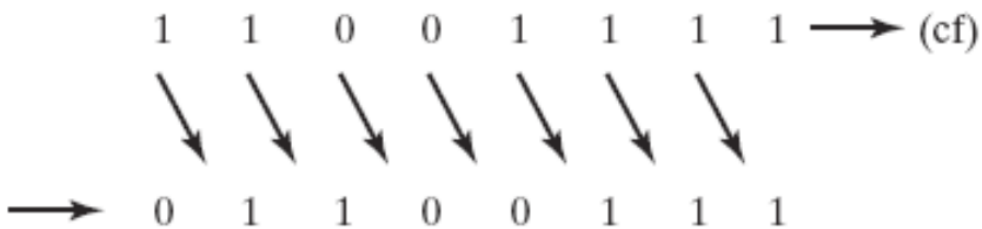
Shift and Rotate Instructions

- Deslocamento Logico vs Aritmético
- Instrução SHL
- Instrução SHR
- Instrução SAL e SAR
- Instrução ROL
- Instrução ROR
- Instrução RCL e RCR
- Instrução SHLD/SHRD

Deslocamento Lógico

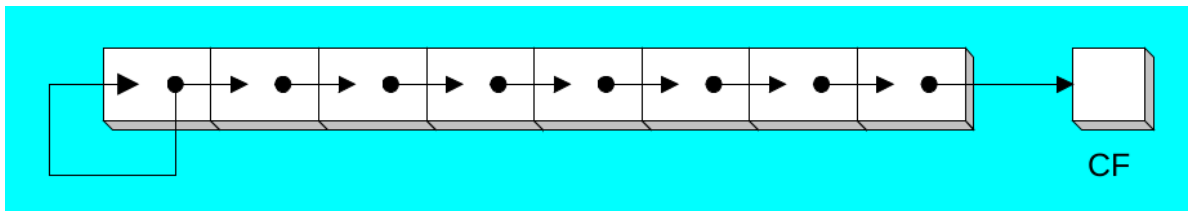
- Um deslocamento lógico completa a posição do novo bit criado com 0(zero).

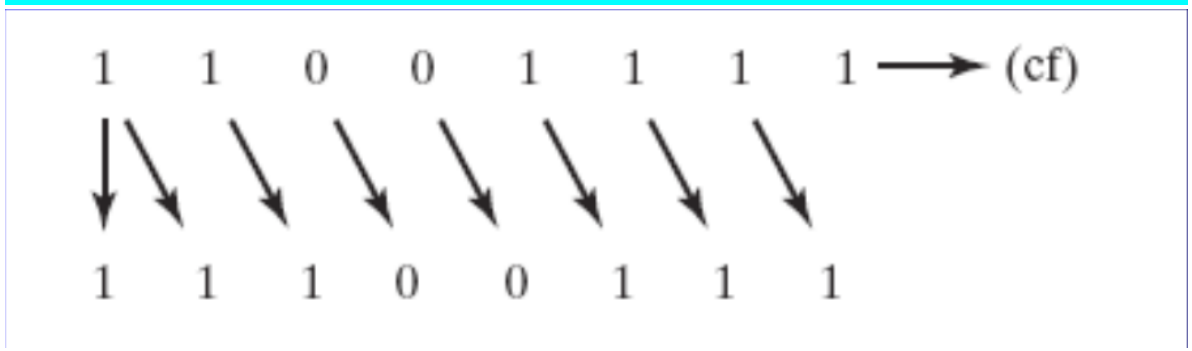




Deslocamento Aritmético

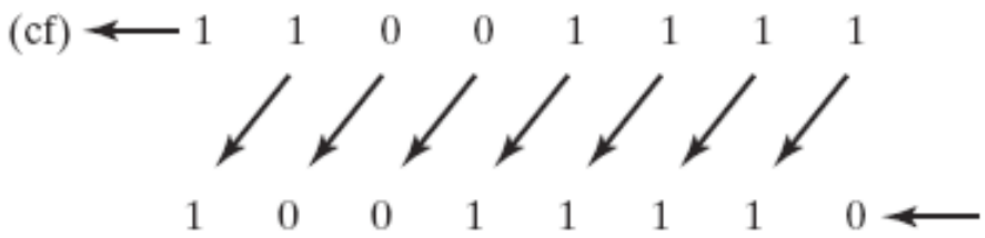
- Um deslocamento lógico completa a posição do novo bit criado com uma cópia do sinal do número.





SHL Instruction

- A instrução SHL (shift left) faz um deslocamento lógico à esquerda no operando destino, preenchendo o bit menos significativo com 0 (zero).



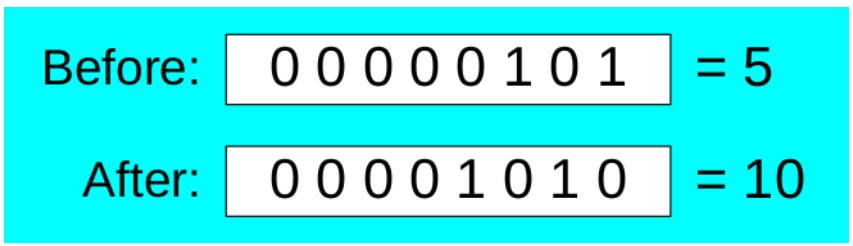
- Tipos de Operandos de SHL:
  - SHL reg,imm8
  - SHL mem,imm8
  - SHL reg,CL
  - SHL mem,CL

Multiplicação Rápida

- Deslocando à esquerda 1 bit multiplica-se o número por 2
- DEslocando n bits à esquerda multiplica-se o operando por 2^n
  - Por Exemplo, $5 * 2^2 = 20$

```
1 mov dl,5
2 shl dl,2              ; DL = 20
```

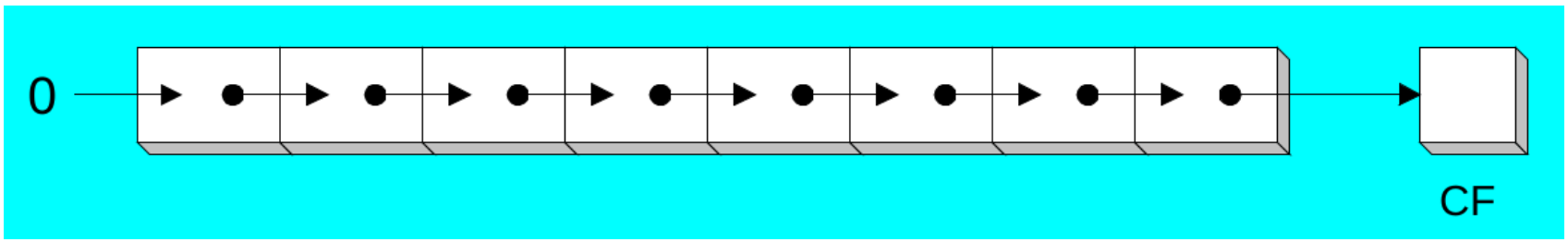
```
mov dl,5
shl dl,1
```



Instrução SHR

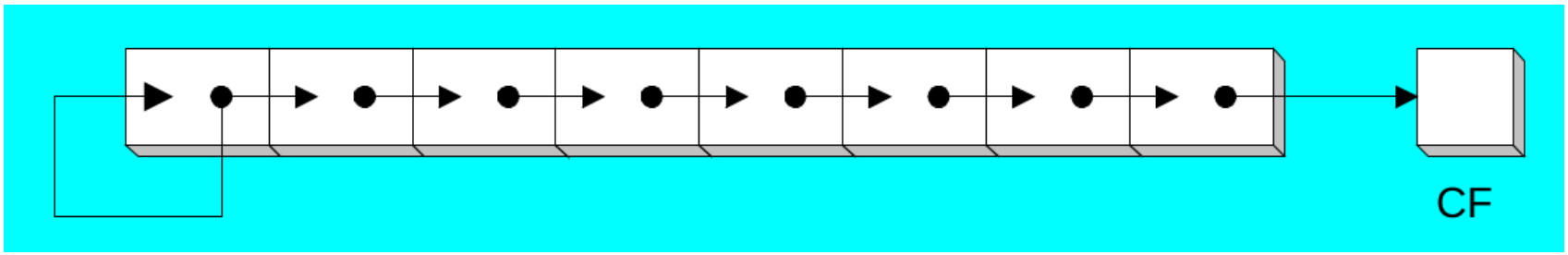
- A instrução SHR (shift right) realiza um deslocamento lógico à direita no operando de destino. O bit mais sibgificativo é preenchido com 0 (zero)
- Deslocando à direita n bits dividimos o operando por 2^n

```
1 mov dl,80
2 shr dl,1          ; DL = 40
3 shr dl,2          ; DL = 10
```



Instruções SAL e SAR

- SAL (shift arithmetic left) é idêntica à SHL.
- SAR (shift arithmetic right) realiza o deslocamento aritmético à direita no operando de destino.

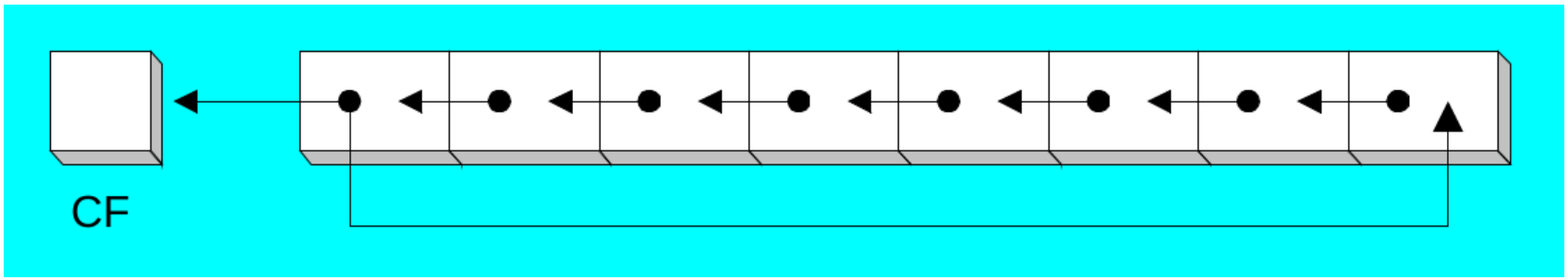


Um deslocamento aritmético preserva o sinal do número.

```
1 mov dl,-80
2 sar dl,1          ; DL = -40
3 sar dl,2          ; DL = -10
```

Instrução ROL

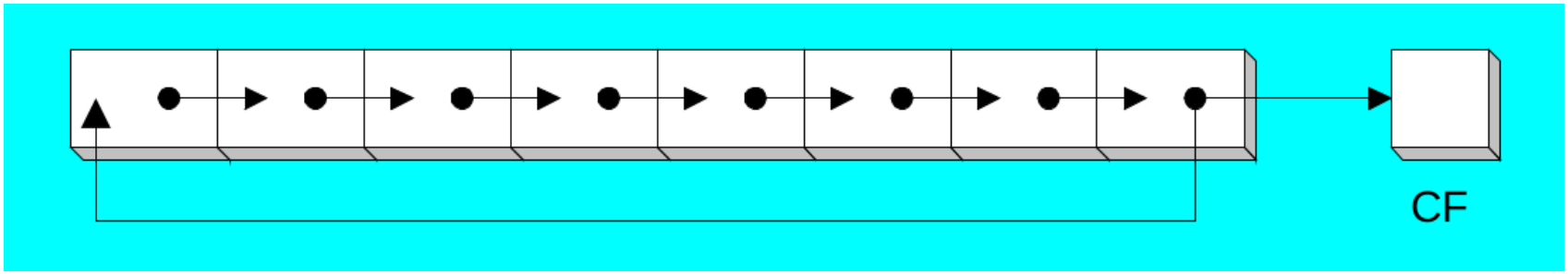
- ROL (rotate) desloca cada bit para a esquerda
- O bit mais sibgificante é copiado no Carry Flag e no bit menos significativo
- Nenhum bit é perdido




```
1 mov al,11110000b
2 rol al,1                  ; AL = 11100001b
3 mov dl,3Fh
4 rol dl,4                  ; DL = F3h
```

Instrução ROR

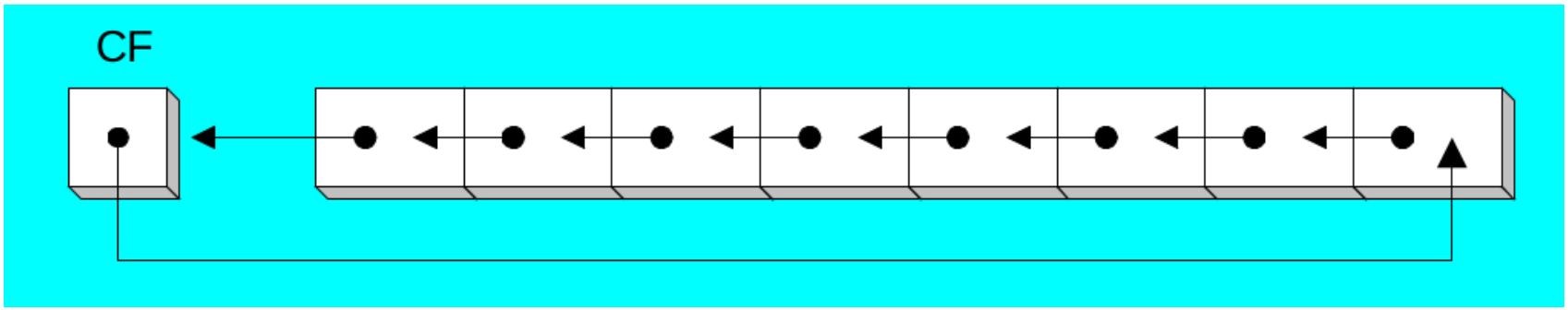
- ROR (rotate right) desloca cada bit à direita
- O bit menos significativo é copiado no Carry flag e no bit mais significativo
- Nenhum bit é perdido




```
1 mov al,11110000b
2 ror al,1              ; AL = 01111000b
3 mov dl,3Fh
4 ror dl,4              ; DL = F3h
```

Instrução RCL

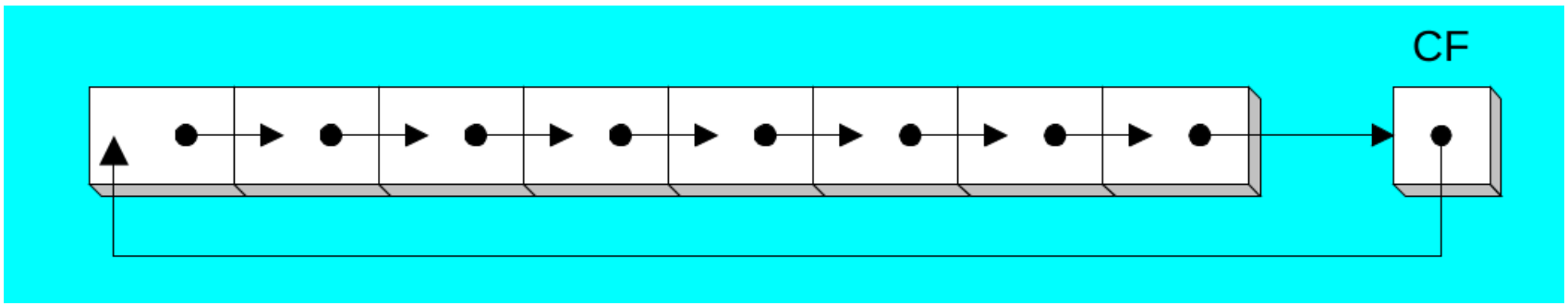
- RCL (rotate carry left) desloca cada bit à esquerda
- Copia o Carry flag para o bit menos significativo
- Copia o bit mais significativo para o Carry flag




```
1 clc                   ;CF = 0
2 mov bl,88h            ;CF,BL = 0 10001000b
3 rcl bl,1              ;CF,BL = 1 00010000b
4 rcl bl,1              ;CF,BL = 0 00100001b
```

Instrução RCR

- RCR (rotate carry right) desloca cada bit à direita
- Copia o Carry flag para o bit mais significativo
- Copia o bit menos significativo para o Carry flag




```
1 stc                   ; CF = 1
2 mov ah,10h            ; CF,AH = 1 00010000b
3 rcr ah,1              ; CF,AH = 0 10001000b
```

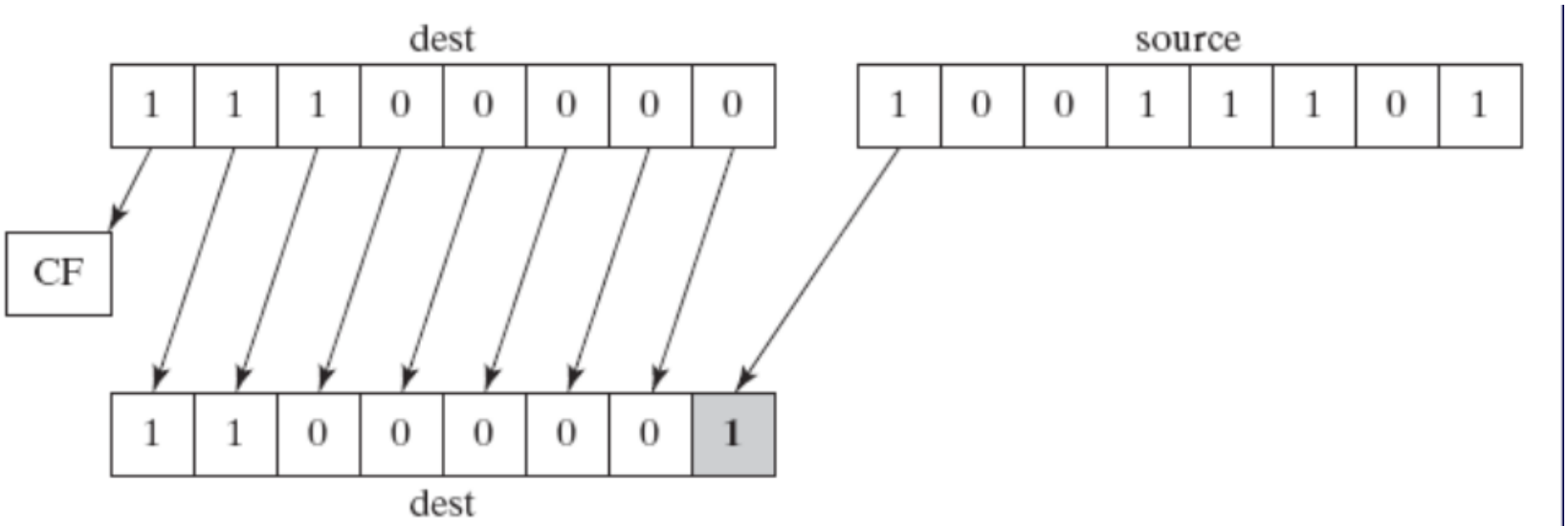
Instrução SHLD

- Desloca o operando de destino n vezes à esquerda
- A posição aberta de bit é preenchida pelo bit mais significativo de operando origem
- O operando de origem não é afetado
- Sintaxe: SHLD destination, source, count
- Operand types:
  - SHLD reg16/32, reg16/32, imm8/CL
  - SHLD mem16/32, reg16/32, imm8/CL

Exemplo de SHLD

- Deslocamento em 1:

```
1 mov al,11100000b
2 mov bl,10011101b
3 shld al,bl,1
```



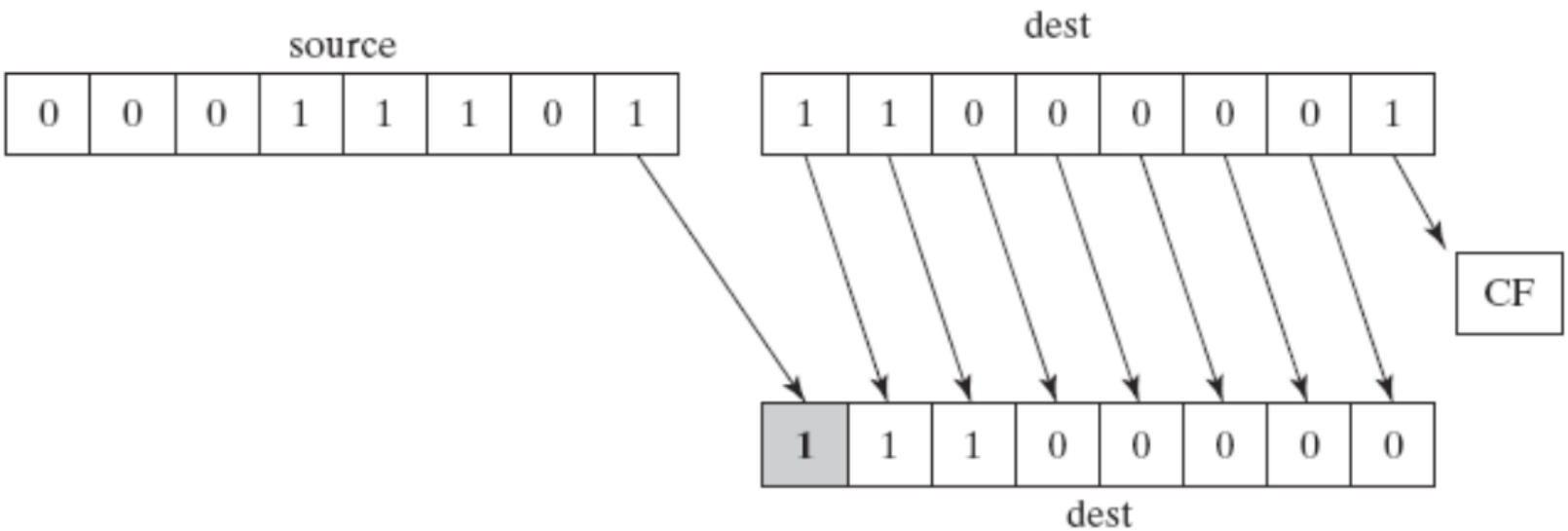
Instrução SHRD

- Desloca o operando de destino n vezes à direita
- As posições abertas pelo deslocamento são preenchidas pelos bits menos significativos do operando de origem
- O operando de origem não é modificado
- Sintaxe: SHRD destination, source, count
- Tipos de Operandos:
  - SHRD reg16/32, reg16/32, imm8/CL
  - SHRD mem16/32, reg16/32, imm8/CL

Exemplo de SHRD

- Deslocamento em 1:

```
1 mov al,11000001b
2 mov bl,00011101b
3 shrd al,bl,1
```



Aplicação de Deslocamentos e Rotações

- Deslocando Múltiplas palavras duplas
- Multiplicação Binária
- Mostrando bits binários
- Isolando um Bit em String

Deslocando Múltiplas Palavras Duplas

- Algumas vezes os programas necessitam deslocar todos bits em um array, como deve quando está sendo movida uma imagem em mapeamento de bits de uma posição da tela para outra.
- Os Bits que seguem em um Array de 3 Palavras Duplas 1 Bit à Direita:

```
1  %include "io.inc"
2
3
4  section .data
5  ArraySize EQU 3
6  array DW 99999999h
7  times ArraySize-1 DW 99999999h     ; 1001 1001...
8
9  section .text
10 global CMAIN
11 CMAIN:
12 mov ebp, esp; for correct debugging
13
14 mov esi,0
15 shr WORD [array+ 4],1        ; high dword
16 rcr WORD [array+ 2],1        ; middle dword, include Carry
17 rcr WORD [array],1           ; low dword, include Carry
18 xor eax, eax
```

Multiplicação Binária

- multiplicar 123×36

	01111011	123
×	00100100	36
	01111011	123 SHL 2
+ 01111011		123 SHL 5
0001000101001100		4428

Multiplicação Binária

- Já sabemos que SHL realiza a multiplicação sem sinal eficientemente quando o multiplicador é uma potência de 2.
- Você poode decompor um número binário em potências de 2.
  - Por exemplo, para multiplicar EAX * 36, fatore 36 em $32 + 4$ e use a propriedade distributiva da multiplicação para realizar a operação:

```
1 EAX * 36
2 = EAX * (32 + 4)
3 = (EAX * 32)+(EAX * 4)
```

```
1 mov eax,123
2 mov ebx,eax
3 shl eax,5          ; mult by 2^5
4 shl ebx,2          ; mult by 2^2
5 add eax,ebx
```

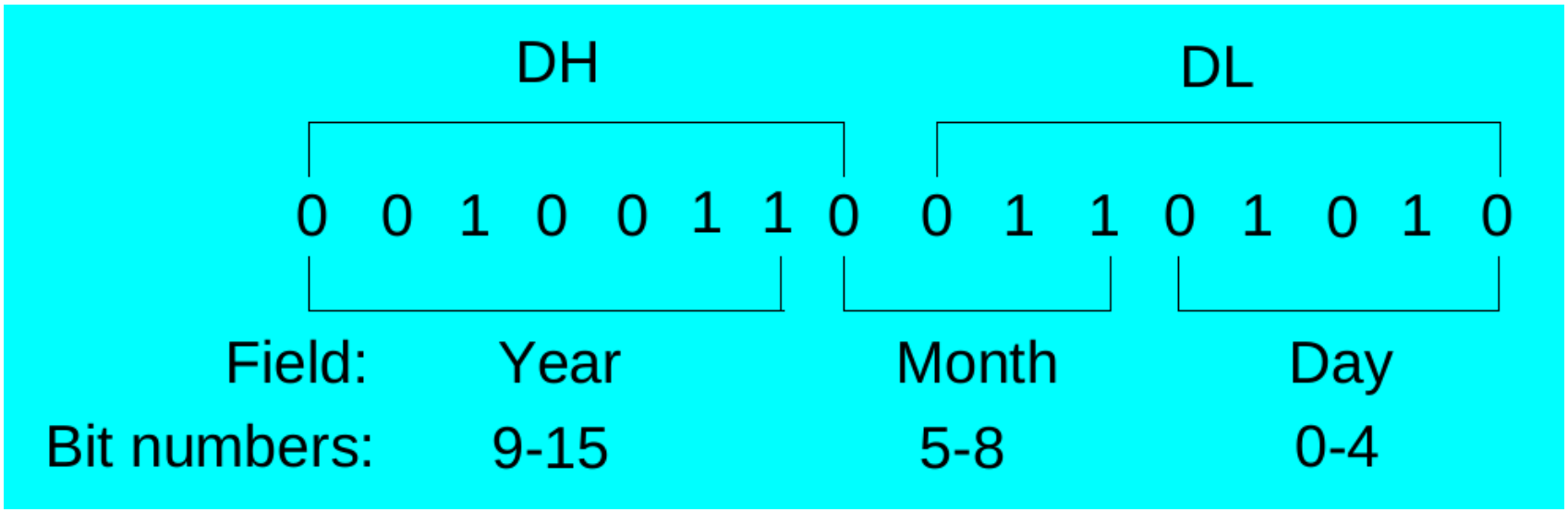
Mostrar Bits Binários

- Algoritmo: Deslocar MSB no Carry flag; Se CF = 1, acrescente um caractere "1"a uma string; caso contrário, acrescente um caractere "0". Repita em um laço 32 vezes.

```
1  .data
2  buffer BYTE 32 DUP(0),0
3  .code
4  mov ecx,32
5  mov esi,OFFSET buffer
6  L1: shl eax,1
7  mov BYTE PTR [esi],'0'
8  jnc L2
9  mov BYTE PTR [esi],'1'
10 L2: inc esi
11 loop L1
```

Isolando bits em uma string

- O campo data dos arquivos do MS-DOS é empacotado em ano, mês, e dia em 16 bits:



Isolar o campo Mês:

```
1 mov ax,dx              ;make a copy of DX
2 shr ax,5               ;shift right 5 bits
3 and al,00001111b       ;clear bits 4-7
4 mov month,al           ;save in month variable
```

Instruções de Multiplicação e Divisão

- Instrução MUL
- Instrução IMUL
- Instrução DIV
- Divisão Inteira com Sinal
- Instruções CBW, CWD, CDQ
- Instrução IDIV
- Implementação de Expressões Aritméticas

Instrução MUL

- No modo 32-bit, a instrução MUL (multiplicação sem sinal) multiplica um operando de 8-, 16-, ou 32-bit por AL, AX, ou EAX.
- The instruction formats are:
  - MUL r/m8
  - MUL r/m16
  - MUL r/m32

TABLE 7-2 MUL Operands.

Multiplicand	Multiplier	Product
AL	reg/mem8	AX
AX	reg/mem16	DX:AX
EAX	reg/mem32	EDX:EAX

Exemplos de MUL

- 100h * 2000h, usando operando de 16-bit:

```
1 .data
2 val1 WORD 2000h
3 val2 WORD 100h
4 .code
5 mov ax,val1
6 mul val2              ; DX:AX = 00200000h, CF=1
```

- 12345h * 1000h, usando operando de 32-bit:

```
1 mov eax,12345h
2 mov ebx,1000h
3 mul ebx               ; EDX:EAX = 0000000012345000h, CF=0
```

Instrução IMUL

- IMUL (multiplicação inteira com sinal) multiplica um operando com sinal de 8-, 16-, ou 32-bit por AL, AX, or EAX
- Preserva o sinal do produto pela extensão de sinal para a metade superior do registrador de destino
- Exemplo: multiplica 48 * 4, usando operando de 8-bit:

```
1 mov al,48
2 mov bl,4
3 imul bl          ; AX = 00C0h, OF=1
```

OF=1 porque AH não é uma extensão de sinal de AL.

Exemplos de IMUL

- Multiplicar $4.823.424 * -423$:

```
1 mov eax,4823424
2 mov ebx,-423
3 imul ebx
4 ; EDX:EAX = FFFFFFFF86635D80h, OF=0
```

OF=0 porque EDX é uma extensão de sinal de EAX.

Instrução DIV

- A instrução DIV (divisão sem sinal) realiza a divisão em operandos inteiros sem sinal de 8-bit, 16-bit, e 32-bit
- Um único operando é informado (operando em registrador ou memória), que é assumido como divisor
- Formatos de Instrução:
  - DIV reg/mem8
  - DIV reg/mem16
  - DIV reg/mem32

Default Operands:

Dividend	Divisor	Quotient	Remainder
AX	*r/m8*	AL	AH
DX:AX	*r/m16*	AX	DX
EDX:EAX	*r/m32*	EAX	EDX

Exemplos de DIV

- Divide 8003h por 100h, usando operandos de 16-bit:

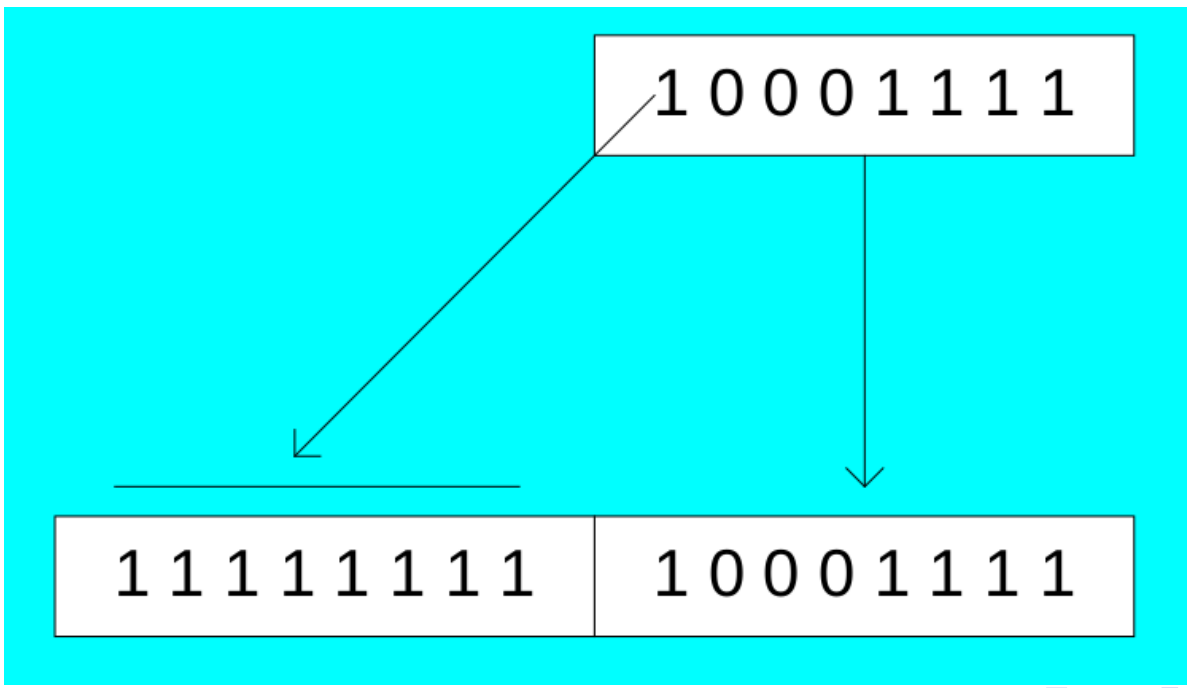
```
1    mov dx,0          ;clear dividend, high
2    mov ax,8003h      ;dividend, low
3    mov cx,100h       ;divisor
4    div cx            ;AX = 0080h, DX = 3
```

Mesma divisão usando operando de 32-bit:

```
1    mov edx,0         ;clear dividend, high
2    mov eax,8003h     ;dividend, low
3    mov ecx,100h      ;divisor
4    div ecx           ;EAX = 00000080h, DX = 3
```

Divisão Inteira com Sinal (IDIV)

- Inteiros com sinal devem ter seu sinal estendido antes da divisão ser realizada
  - preencha o os byte/word/doubleword com a cópia do bits de sinal dos byte/word/doubleword's menos significativos
- Por exemplo, o byte mais significativo contém uma cópia do sinal do byte menos significativo:



Instruções CBW, CWD, CDQ

- As instruções CBW, CWD, e CDQ proveem importantes operações de estenção de sinal:
  - CBW (convert byte to word) estende AL em AH
  - CWD (convert word to doubleword) estende AX em DX
  - CDQ (convert doubleword to quadword) estende EAX em EDX
- Exemplo:

```
1 .data
2 dwordVal SDWORD -101      ; FFFFFF9Bh
3 .code
4 mov eax,dwordVal
5 cdq                       ; EDX:EAX = FFFFFFFFFFFFFF9Bh
```

Instrução IDIV

- IDIV (signed divide) realiza a divisão inteira com sinal
- Mesma sintaxe e operandos da instrução DIV
- Exemplo: divisão em 8-bit de -48 por 5

```
1 mov al,-48
2 cbw                    ; extend AL into AH
3 mov bl,5
4 idiv bl                ; AL = -9, AH = -3
```

Exemplos IDIV

- Exemplo: divisão em 16-bit de -48 por 5

```
1 mov ax,-48
2 cwd           ; extend AX into DX
3 mov bx,5
4 idiv bx       ; AX = -9, DX = -3
```

- Exemplo: divisão em 32-bit de -48 por 5

```
1 mov eax,-48
2 cdq                   ; extend EAX into EDX
3 mov ebx,5
4 idiv ebx              ; EAX = -9, EDX = -3
```

Expressões Aritméticas sem Sinal

- Algumas boas razões para aprender como implementar expressões inteiras:
  - Aprender como compilá-las
  - Testar seu entendimento de MUL, IMUL, DIV, IDIV
  - Verificar overflow (Carry and Overflow flags)
- Example: var4 = (var1 + var2) * var3

```
1 ; Assumir operandos sem sinal
2 mov eax, var1
3 add eax, var2              ; EAX = var1 + var2
4 mul var3                   ; EAX = EAX * var3
5 jc  TooBig                 ; check for carry
6 mov var4, eax              ; save product
```

Expressões Aritméticas Com Sinal(1 of 2)

- Example: eax = (-var1 * var2) + var3

```
1 mov eax,var1
2 neg eax
3 imul var2
4 jo TooBig                ; check for overflow
5 add eax,var3
6 jo TooBig                ; check for overflow
```

- Exemplo: var4 = (var1 * 5) / (var2 - 3)

```
1 mov eax,var1             ; left side
2 mov ebx,5
3 imul ebx                 ; EDX:EAX = product
4 mov ebx,var2             ; right side
5 sub ebx,3
6 idiv ebx                 ; EAX = quotient
7 mov var4,eax
```

Expressões Aritméticas Com Sinal(2 of 2)

- Exemplo: $var4 = (var1 * -5)/(-var2\%var3);$

```
1 mov eax,var2          ; begin right side
2 neg eax
3 cdq                   ;sign-extend dividend
4 idiv var3             ;EDX = remainder
5 mov ebx,edx           ;EBX = right side
6 mov eax,-5            ;begin left side
7 imul var1             ;EDX:EAX = left side
8 idiv ebx              ;final division
9 mov var4,eax          ;quotient
```

Em alguns casos é mais fácil calcular primeiramente o termo mais a direita primeiro.

Adição e Subtração Estendida

- Instrução ADC
- Adição com Precisão Estendida
- Instrução SBB
- Subtração com Precisão Estendida

Adição com Precisão Estendida

- Somando dois operando que são maiores que uma palavra do computador(32 bits).
  - Virtualmente não há limite para o tamanho dos operandos
- A operação aritmética deve ser realizada em passos
  - O valor do Carry para cada passo deve ser passado para o próximo passo.

Instrução ADC

- A instrução ADC (add with carry) soma ambos operandos, origem e destino e o conteúdo de Carry flag armazenando o resultado no operndo de destino.
- Operandos são valores binários
  - Mesma sintaxe do ADD, SUB, etc.
- Example
  - Soma de dois inteiros de $32 - bit (FFFFFFFFh + FFFFFFFFh)$, produzindo uma soma de $64 - bit$ em $EDX : EAX$

```
1 mov edx,0
2 mov eax,0FFFFFFFFh
3 add eax,0FFFFFFFFh
4 adc edx,0                    ;EDX:EAX = 00000001FFFFFFFEh
```

Exemplo de Soma Estendida

- Tarefa: Somar 1 a *EDX* : *EAX*
  - Valor inicial de EDX:EAX: 00000000FFFFFFFFh
  - Somar os 32 bits menos significativos primeiro, definindo o Carry flag.
  - Somar os 32 bits mais significativos, e incluir o Carry flag.

```
1 mov edx,0                    ; set upper half
2 mov eax,0FFFFFFFFh           ; set lower half
3 add eax,1                    ; set lower half
4 adc edx,0                    ; set upper half
```

$EDX : EAX = 0000000100000000$

Instrução SBB

- A instrução SBB (subtract with borrow) subtrai ambos operandos, origem e o valor do Carry flag do operando destino.
- Sintaxe do Operando:
  - Mesmo que a instrução ADC

Exemplo de Subtgração Estendida

- Tarefa: Subtrair 1 de *EDX* : *EAX*
  - Valor inicial de EDX:EAX: 0000000100000000h
  - Subtrair os 32 bits menos significativos primeiro, definindo o Carry flag.
  - Subtrair os 32 bits mais significativos, e incluir o Carry flag.

```
1 mov edx,1             ;set upper half
2 mov eax,0             ;set lower half
3 sub eax,1             ;subtract lower half
4 sbb edx,0             ;subtract upper half
```

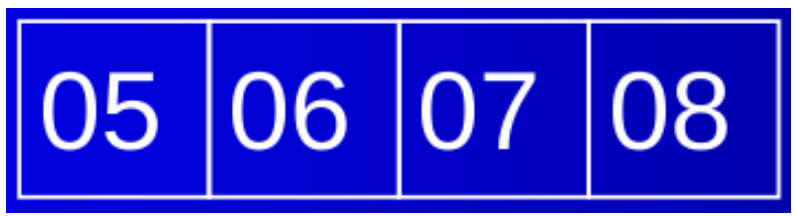
EDX:EAX = 00000000 FFFFFFFF

ASCII e Artimética Decimal Empacotada

- Binary Coded Decimal
- ASCII Decimal
- Instrução AAA
- Instrução AAS
- Instrução AAM
- Instrução AAD
- Packed Decimal Integers
- Instrução DAA
- Instrução DAS

Binary-Coded Decimal

- Inteiros Binary-coded decimal (BCD) usam 4 bits binários para representar cada dígito decimal
- Um número usando representação não empacotada BCD armazena um digito decimal nos 4 menores bits de cada byte
  - Por Exemplo, 5.678 é armazenado com a seguinte sequência de bytes hexadecimais:



ASCII Decimal

- Um número usando representação ASCII Decimal armazena um único dígito ASCII em cada byte
  - Por exemplo, 5.678 é armazenado como a seguinte sequência de bytes hexadecimais:

Instrução AAA

- A instrução AAA (ASCII adjust after addition) ajusta o resultado binário de uma instrução ADD ou ADC. Isso faz o resultado em AL consistente com a representação decimal em ASCII.
  - O valor do Carry, se não termina em AH
- Exemplo: Somar '8' e '2'

```
1 mov ah,0
2 mov al,'8'          ;AX = 0038h
3 add al,'2'          ;AX = 006Ah
4 aaa                 ;AX = 0100h (adjust result)
5 or ax,3030h         ;AX = 3130h = '10'
```

Instrução AAS

- A instrução AAS (ASCII adjust after subtraction) ajusta o resultado binário de uma instrução SUB ou SBB.
- Isso faz o resultado de AL consistente com a representação decimal em ASCII.
  - Isto coloca o valor do Carry, se há algum em AH
- Exemplo: Subtrair '9' de '8'

```
1 mov ah,0
2 mov al,'8'          ;AX = 0038h
3 sub al,'9'          ;AX = 00FFh
4 aas                 ;AX = FF09h, CF=1
5 or al,30h           ;AL = '9'
```

Instrução AAM

- A instrução AAM (ASCII adjust after multiplication) ajusta o resultado binário de uma instrução MUL. A multiplicação deve ser realizada em números BCD não empacotados.

```
1 mov bl,05h          ; first operand
2 mov al,06h          ; second operand
3 mul bl              ; AX = 001Eh
4 aam                 ; AX = 0300h
```

Instrução AAD

- A instrução AAD (ASCII adjust before division) ajusta o número BCD não empacotado dividido em AX antes da operação de divisão

```
1  %include "io.inc"
2
3  section .bss
4  quotient RESB 1
5  remainder RESB 1
6
7  section .text
8  global CMAIN
9  CMAIN:
10 mov ebp, esp; for correct debugging
11 mov ax,0307h          ; dividend
12 aad             ; AX = 0025h
13 mov bl,5              ; divisor
14 div bl             ; AX = 0207h
15 mov [quotient],al
16 mov [remainder],ah
17 xor eax, eax
18 ret
```

Aritmética Decimal Empacotada

- Decimais Inteiros Empacotados armazenam 2 dígitos decimais por byte
  - Por exemplo, 12.345.678 pode ser armazenado como a seguinte sequência de bytes hexadecimais:

12	34	56	78

Decimal empacotado é também conhecido como BCD empacotado. Bom para valores financeiros - precisão estendida é possível, sem erros de arredondamento.

Instrução DAA

- A instrução DAA (decimal adjust after addition) converte o resultado binário de uma operação ADD ou ADC para o formato decimal empacotado.
  - O valor a ser ajustado deve estar em AL
  - Se o dígito menos significativo é ajustado, a Auxiliary Carry flag é ativada.
  - Se o dígito mais significativo é ajustado, a Carry flag é ativada.

DAA Logic

```
1  If (AL(lo) > 9) or (AuxCarry = 1)
2  AL = AL + 6
3  AuxCarry = 1
4  Else
5      AuxCarry = 0
6  Endif
7  If (AL(hi) > 9) or Carry = 1
8  AL = AL + 60h
9  Carry = 1
10 Else
11 Carry = 0
12 Endif
```

Se $AL = AL + 6$ ativa Carry flag, seu valor é usando quando estiver avaliando AL(hi).

DAA Exemplos

- Exemplo: calcular $BCD35 + 48$

```
1 mov al,35h
2 add al,48h        ; AL = 7Dh
3 daa               ; AL = 83h, CF = 0
```

- Exemplo: calcular $BCD35 + 65$

```
1 mov al,35h
2 add al,65h        ; AL = 9Ah
3 daa               ; AL = 00h, CF = 1
```

- Exemplo: calcular $BCD69 + 29$

```
1 mov al,69h
2 add al,29h        ; AL = 92h
3 daa               ; AL = 98h, CF = 0
```

Instrução DAS

- A instrução DAS (decimal adjust after subtraction) converte o resultado binário de uma operação SUB ou SBB para o formato decimal empacotado.
- O valor deve estar em AL
- Exemplo: subtrair BCD 48 de 85

```
1 mov al,48h
2 sub al,35h        ; AL = 13h
3 das               ; AL = 13h CF = 0
```

Lógica DAS

```
1  If (AL(lo) > 9) OR (AuxCarry = 1)
2  AL = AL - 6;
3  AuxCarry = 1;
4  Else
5  AuxCarry = 0;
6  Endif
7  If (AL > 9FH) or (Carry = 1)
8  AL = AL - 60h;
9  Carry = 1;
10 Else
11 Carry = 0;
12 Endif
```

Se $AL = AL - 6$ ativa a Carry flag, seu valor será usado quando avaliar AL no segundo IF.

Exemplo DAS(1 of 2)

- Exemplo: subtrair $BCD48 - 35$

```
1 mov al,48h
2 sub al,35h        ; AL = 13h
3 das               ; AL = 13h CF = 0
```

- Exemplo: subtrair $BCD62 - 35$

```
1 mov al,62h
2 sub al,35h        ; AL = 2Dh, CF = 0
3 das               ; AL = 27h, CF = 0
```

- Exemplo: subtrair $BCD32 - 29$

```
1 mov al,32h
2 add al,29h        ; AL = 09h, CF = 0
3 daa               ; AL = 03h, CF = 0
```

Exemplo DAS (2 of 2)

- Exemplo: subtrair $BCD32 - 39$

```
1 mov al,32h
2 sub al,39h        ; AL = F9h, CF = 1
3 das               ; AL = 93h, CF = 1
```

Passos:
$AL = F9h$
$CF = 1$, então subtrair 6 de $F9h$
$AL = F3h$
$F3h > 9Fh$, então subtrair $60h$ de $F3h$
$AL = 93h, CF = 1$