```
0D82:0100 B402          MOV     AH,02
0D82:0102 B241          MOV     DL,41
0D82:0104 CD21          INT     21
0D82:0106 CD20          INT     20
0D82:0108 69            DB      69
```

Capítulo 5

APRESENTAÇÃO DE DADOS

*Este capítulo aborda as instruções para a exibição de dados no monitor de vídeo. Ele ensina a apresentar um único caractere, movimentação de dados, sequência de caracteres (apresentação de strings), valores binários, registradores de estado e aplicação simples de saltos condicionais. Novas instruções de uso das ferramentas **Enhanced DEBUG** e novos códigos de máquina para controlar a apresentação de dados no monitor de vídeo do compu¬tador em uso também são assuntos estudados no capítulo.*

5.1 - Apresentação de um caractere

Para apresentar apenas um caractere na tela do monitor de vídeo, é necessário utilizar um recurso chamado *interrupções*. Até este momento o uso das ferramentas **Enhanced DEBUG**, bem como a manipulação de dados numéricos, ocorreu somente em memória. Assim sendo, carregue para a memória o programa **Enhanced DEBUG**.

A interrupção responsável por controlar a apresentação de um caractere no console padrão de saída (que pode ser a tela do monitor de vídeo) é definida com o código hexadecimal **21** (essa interrupção permite também controlar a entrada de dados, como será visto no próximo capítulo). É importante ressaltar que a comunicação do microprocessador de um computador digital que é realizada com os seus periféricos ocorre por meio das interrupções e a interrupção **21** é usada para controlar o acesso aos periféricos de entrada e saída de dados.

A apresentação de um caractere é realizada com o uso dos registradores gerais **AX** e **DX**, em que **AX** armazena o código de controle para a exibição de apenas um caractere e **DX** armazena o código do caractere a ser apresentado. O registrador geral **DX** normalmente não deve ser usado para o armazenamento de valores para entrada ou saída de dados (como está aqui sendo exemplificado), pois sua finalidade é apontar para um endereço de memória que possui algum dado a ser operacionalizado pelo programa. Mas no presente momento, isto pode ser realizado por ser a apresentação de apenas um caractere. Para apresentar um *string* o procedimento de trabalho deve ser um pouco diferente como será ainda constatado.

Com o programa **Enhanced DEBUG** carregado informe ao registrador geral **AX** o valor hexadecimal **0200** (AH = 02 e AL = 00) que tem por finalidade permitir a escrita de um caractere no periférico padrão de saída; caso contrário, não é possível apresentar o caractere na tela do monitor de vídeo. Neste caso, informe na linha de *prompt* do programa a instrução:

```
R AX 0200 <Enter>
```

Na sequência informe para o registrador geral **DX** o valor hexadecimal **0040** (valor do caractere **@** na tabela ASCII)[1], como demonstrado a seguir:

```
R DX 0040 <Enter>
```

[1] *Para visualizar a tabela ASCII com os 128 códigos padrão definidos em decimal, binário e hexadecimal, consulte o apêndice A.*

Acione o comando **R** e observe os valores existentes nos registradores gerais **AX** e **DX**, como é indicado a seguir em negrito.

```
AX=0200 BX=0000 CX=0000 DX=0040 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3                RET
```

É pertinente salientar que o valor **0200** do registrador geral **AX** será usado para indicar ao sistema operacional que deve ser apresentado um caractere no monitor de vídeo. O registrador de 16 *bits* **AX** é formado pelos registradores **AH** (*bits* mais significativos) e **AL** (*bits* menos significativos), de 8 *bits* cada um. Neste caso o valor **0200** possui a parte do valor hexadecimal **02** alocada no registrador **AH** e a parte **00** alocada no registrador **AL**. Se fosse colocado outro valor no registrador **AH** diferente de **02**, ele informaria ao sistema operacional para executar uma ação diferente de apresentar algo na tela. O registrador mais significativo **AH** caracteriza-se por ser utilizado para informar o código de controle padrão que efetua a saída de um caractere no monitor de vídeo. Neste caso, o valor hexadecimal **02** que é o código de função para uso da interrupção do monitor de vídeo.

Outro detalhe a ser observado com relação ao valor **0040** do registrador geral **DX** é que ele também possui sua parte mais significativa (registrador **DH**) e menos significativa (registrador **DL**), e o código hexadecimal **40** está sendo alocado na parte menos significativa do registrador geral **DX**.

A seguir é necessário informar nos endereços de deslocamento **0100** e **0101** o par de valores hexadecimais (*opcodes*) **CD** (que representa o comando de interrupção) e **21** (que representa a interrupção a ser acionada) que executam o código em linguagem de máquina que ativa o serviço de acionamento do monitor de vídeo para que um determinado caractere seja apresentado através do código de serviço **0200** armazenado no registrador geral **AX**. Informe o comando:

```
E 0100 CD 21 <Enter>
```

Na sequência verifique se o registrador de deslocamento **IP** está apontando para o endereço de deslocamento **0100**. Caso não esteja (o que é difícil, pois o programa **Enhanced DEBUG** inicia sua execução nesse endereço), execute a instrução **R IP 0100**.

Depois acione o comando **R** para que seja apresentado o estado atual dos registradores, como é mostrado em seguida. Atente para os pontos em negrito e verifique se esses valores estão presentes na tela do programa **Enhanced DEBUG**.:

```
AX=0200 BX=0000 CX=0000 DX=0040 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 CD21              INT     21
```

Observe atentamente as informações dos registradores **AX**, **DX** e **IP** e a terceira linha, na qual se encontra a definição da instrução **INT** (*interrupt*) indicando o valor hexadecimal **21** (**INT 21**). À esquerda veja a definição do par de códigos de máquina (*opcode*) **CD21**.

A instrução **INT** quando executada altera o estado dos registradores de estados (*flags*) **TF** e **IF**, possuindo como possibilidade de trabalho a definição direta de um valor (constante) e para este tipo de ação consome-se 1 byte de memória.

O *opcode* **CD21** (**INT 21**) é uma instrução de interrupção do sistema operacional responsável por efetuar acesso aos periféricos de entrada e saída conectados ao computador. Neste caso particular será realizada uma apresentação de dado no monitor de vídeo devido ao valor **0200** armazenado no registrador **AX**, ou seja, o valor **02h** que está no em **AH**. O valor **02h** caracteriza-se por representar a sub função responsável por indicar para a interrupção **21h** a ação de apresentação de um caractere no periférico de saída.

Será solicitada a execução dos dados em memória, mas desta vez não será usado o comando **T**, mas o comando **G** (*go*), porque o comando **T** executa linha a linha, e isso gera uma execução inconveniente, pois ao ser executada a interrupção (**INT 21**), ela desvia para uma sub-rotina interna, a qual dará muito trabalho para ser executada passo a passo. A interrupção acessada pelo código **INT 21**, busca no registrador menos significativo **DL** a informação a ser

apresentada no monitor de vídeo, que está sendo controlado pelo código **02h** armazenado no registrador mais significativo **AH**.

O comando **G** age de forma direta, necessitando apenas que seja informado até que ponto de um deslocamento de um segmento ele deve ir, ou seja, é necessário informar o endereço de parada para a execução. Neste caso (considerando que o registrador de deslocamento **IP** está no endereço de deslocamento **0100**) será solicitado que o comando **G** execute os endereços de deslocamento **0100**, **0101** e pare no endereço de deslocamento **0102**. Informe na linha de *prompt* do programa o comando:

```
G 0102 <Enter>
```

O programa apresenta o caractere **@** (note o ponto marcado em negrito) e indica a listagem de estado dos registrados, conforme a seguir:

```
@AX=0240 BX=0000 CX=0000 DX=0040 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PO NC
07E9:0102 F726FC26          MUL     WORD PTR [26FC]               DS:26FC=0000
```

A primeira linha tem a indicação do caractere **@**, e os valores existentes no registrador geral **AX** e no registrador de deslocamento **IP**. O registrador geral **AX** assumiu no seu registrador geral menos significativo (**AL**) o código **40** correspondente ao caractere **@**.

A instrução **INT 21** (*opcode* **CD21**) executa a chamada de uma sub-rotina do sistema operacional que acessa o serviço de entrada ou de saída. Para a execução de sua ação é necessário passar e receber parâmetros por meio de registradores (WEBER, 2004, p. 247). No caso apresentado o registrador **AH** foi usado com o código **02** (código que informa ao sistema operacional que será usado o monitor de vídeo para a apresentação de um caractere) para indicar a função que deverá ser executada pela instrução **INT 21** e o registrador **AL** foi usado para receber a resposta da ação, ou seja, o valor **40** referente ao caractere a ser apresentado. Veja a seguir alguns exemplos de ações normalmente executadas com a instrução **INT 21**:

- Registrador AH com valor 01h espera que um caractere seja informado via teclado, mostra o caractere na tela e retorna seu código ASCII no registrador **AL**;
- Registrador **AH** com valor **08h** espera que um caractere seja informado via teclado, não mostra o caractere na tela e retorna seu código ASCII no registrador **AL**;
- Registrador **AH** com valor **09h** mostra a cadeia de caracteres (finalizada com o caractere **$** - valor **24h**) que se encontra apontada pelo par de registradores **DS:DX**.

Todos os programas em código de máquina desenvolvidos até aqui utilizaram apenas uma linha de código para suas ações. No entanto, um programa costuma ter bem mais do que isso.

Mesmo um programa de apresentação de um único caractere acaba precisando de mais de um código, pois é sempre bom após a execução de um programa solicitar o retorno do controle ao sistema operacional, ou seja, solicitar o encerramento do programa. Neste caso, deve-se utilizar a interrupção **20**, e é claro que ela deve ser colocada no final do código de programa.

Lembre-se de que o programa de apresentação do caractere **@** está utilizando os endereços de deslocamento **0100** e **0101** para a definição da interrupção **21**. A interrupção **20** deve ser colocada nos endereços de segmentos **0102** e **0103** com os códigos de máquina (*opcode*) **CD** e **20.** Informe o comando:

```
E 0102 CD 20 <Enter>
```

Na sequência verifique se o registrador de deslocamento **IP** está apontando para o endereço de deslocamento **0102**. Caso não esteja, execute o comando **R IP** e forneça como endereço de apontamento inicial o valor hexadecimal **0102**. Depois acione o comando **R** para que seja apresentado o estado atual dos registradores, como é mostrado em seguida:

```
AX=0240 BX=0000 CX=0000 DX=0040 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PO NC
07E9:0102 CD20              INT     20
```

A terceira linha traz a definição do rótulo **INT** indicando o valor hexadecimal **20** (**INT 20**) e também à esquerda a definição do par de códigos de máquina **CD20**.

Para executar o programa que agora tem duas instruções, ajuste o registrador de deslocamento **IP** para o deslocamento **0100** e execute o comando **G** sem a indicação do endereço de término, pois como foi definida a interrupção **20**, ela informa o término da execução do programa, retornando o controle de operação do computador para o sistema operacional sem necessidade de indicar em que segmento deve a execução parar.

```
R IP 0100 <Enter>
G         <Enter>

@
Program terminated (0000)
```

Assim que o comando **G** é acionado, o programa apresenta o caractere **@**, em seguida sinaliza o término da sua ação com a mensagem "**Program terminated (0000)**". É apresentada a sequência de execução dos comandos e do programa a partir do ajuste do registrador de deslocamento **IP**.

Para ver uma listagem do código de programa criado, que agora possui duas instruções, será utilizado o comando **U** (*unassemble*) que faz a listagem do programa de um ponto inicial até um ponto definido como intervalo de deslocamentos. Informe no *prompt* do programa o comando:

```
U 0100 0102 <Enter>
```

O programa apresenta algo semelhante a:

```
07E9:0100 CD21              INT     21
07E9:0102 CD20              INT     20
```

A listagem anterior apresenta apenas os códigos de instrução situados nos deslocamentos **0100** e **0102**. Os deslocamentos **0101** e **0103** foram usados apenas para auxiliar a entrada dos pares de códigos de máquina no programa que controlam respectivamente a apresentação do caractere e o encerramento do programa. O primeiro bloco de números separados por dois pontos mostram a posição de memória onde uma instrução encontra-se definida, o segundo bloco apresenta o código operacional (*opcode*) e o terceiro bloco apresenta a instrução escrita em linguagem *Assembly*. Note que a instrução **INT 21** está na posição de memória **0100** e a instrução **INT 20** está situada duas posições a frente, ou seja, na posição **0102**. Isto mostra que a instrução **INT 20** está ocupando dois *bytes* de memória e certamente uma próxima instrução após a interrupção **INT 20** seria então posicionada na posição de memória **0104**. A distância de ocupação de memória usada por uma linha de instrução pode ser de um a seis *bytes*, dependendo da instrução em uso. O apêndice B apresenta este tipo de informação.

O uso de código de máquina pelo programa **Enhanced DEBUG** pode ser simplificado com o fornecimento direto de códigos de programas na linguagem *Assembly* para o modo de montagem (*assembler*).

Para testar essa nova possibilidade, saia dos programas **Enhanced DEBUG** com o comando **Q** e retorne ao programa imediatamente. Essa ação fará com que todos os registradores utilizados sejam zerados.

Para o próximo teste forneça para o registrador geral **AX** e para o registrador geral **DX**, respectivamente, os valores hexadecimais **0200** e **0046** (valor do caractere **F** na tabela ASCII).

```
R AX 0200 <Enter>
R DX 0046 <Enter>
```

Acione o comando **R** e observe os valores existentes nos registradores gerais **AX** e **DX**, como é indicado a seguir em negrito.

```
AX=0200 BX=0000 CX=0000 DX=0046 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3                RET
```

A partir do ponto em que os valores de controle do programa estão armazenados nos registradores, é necessário informar as interrupções **21** e **20**, as quais serão feitas com códigos em *Assembly* e não em código de máquina, por meio de *opcodes*. Acione o comando **A** (*assemble*) fornecendo como ponto inicial de armazenamento de código o deslocamento **0100**, como indicado a seguir:

```
A 0100 <Enter>
```

Automaticamente será apresentada a linha (segmento:deslocamento) em que o código deve ser definido. É sempre bom lembrar que o endereço de segmento é um valor escolhido pelos programas **Enhanced DEBUG**. Nesta etapa forneça as seguintes linhas de código:

```
07E9:0100 INT 21 <Enter>
07E9:0102 INT 20 <Enter>
07E9:0104 <Enter>
```

Ao informar a primeira linha de código e acionar a tecla **<Enter>,** é apresentado automaticamente o segundo endereço de deslocamento. Nesse ponto, ao entrar a segunda linha de código e acionar **<Enter>**, passa-se para o próximo segmento, que se nada for entrado e for acionada a tecla **<Enter>**, o modo de trabalho do comando **A** é encerrado. A sequência de comandos apresenta um exemplo visual da ocorrência aqui descrita:

Observação

A entrada de instruções no programa **Enhanced DEBUG** pode ser feita tanto com letras minúsculas como maiúsculas. O programa sempre converte os caracteres no seu formato maiúsculo quando da apresentação das instruções listadas.

Para fazer um teste de execução do programa, utilize o comando **G** sem indicar nenhum endereço de deslocamento inicial. Veja o resultado que será apresentado:

```
F
Program terminated (0000)
```

O uso de comandos na forma mnemônica é bem mais fácil de que comandos em código de máquina. A partir deste ponto os próximos programas serão criados com esse recurso.

5.2 - Movimentação de dados

Os programas feitos até aqui utilizam as informações armazenadas nos registradores de forma manual. No entanto, é possível controlar essas ações por meio de um programa de forma mais automática.

Para conseguir esse intento, é necessário utilizar uma instrução *Assembly* denominada **MOV** (*move*)[2], que faz a movimentação de dados (*byte* ou *word*) nos registradores e entre os registradores. No entanto, é necessário levar em consideração que não é possível movimentar dados de uma posição da memória para outra posição de forma direta; para este tipo de ação, é sempre necessário fazer uso de um registrador. O dado posicionado em um local de origem deve ser posicionado primeiramente em um registrador, para depois, ser movimentado do registrador para um local de destino.

[2] *Lembre-se de que já foram transmitidas noções sobre as instruções ADD* (addition), *SUB* (subtract), *MUL* (multiply), *DIV* (divide) e *INT* (interrupt).

A instrução **MOV** quando executada não altera o estado dos registradores de estados (*flags*), possuindo como possibilidade de trabalho as operações sobre:

- registrador, registrador (para este tipo de ação usa 2 *bytes* de memória);
- memória, registrador (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, memória (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, constante (para este tipo de ação usa de 2 a 3 *bytes* de memória);
- memória, constante (para este tipo de ação usa de 3 a 6 *bytes* de memória);
- segmentoregistro, registrador (para este tipo de ação usa 2 *bytes* de memória);
- segmentoregistro, memória (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- memória, segmentoregistro (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, segmentoregistro (para este tipo de ação usa 2 *bytes* de memória).

Para assimilar um pouco de intimidade com essa nova instrução, serão executadas algumas movimentações de valores entre os registradores. Para começar, informe para o registrador geral **AX** o valor **1122** e para o registrador **DX** o valor **AABB**.

```
R AX 1122 <Enter>
R DX AABB <Enter>
```

Verifique com o comando **R** se os valores estão como os seguintes:

```
AX=1122 BX=0000 CX=0000 DX=AABB SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 CD21              INT     21
```

Como exemplo será realizada a movimentação da parte menos significativa do registrador **DX** para a parte mais significativa do operador **AX**. Na linha de *prompt* dos programas **Enhanced DEBUG** execute as instruções:

```
A 0100 <Enter>
```

Informe na linha a instrução **MOV AH,DL** e acione a tecla **<Enter>** por duas vezes.

Observe na sequência a aparência que deve ter a tela do programa após as execuções anteriores:

```
07E9:0100 MOV AH,DL <Enter>
07E9:0102           <Enter>
```

Caso queira poderá fazer uso do par de *opcodes* **8A E2** para os endereços **0100** e **0101** informados com o uso do comando **E** a partir da instrução **E 0100 8A E2**. Execute em seguida o comando **T** para visualizar o estado atual dos registradores durante a execução do programa, como indicado a seguir:

```
AX=BB22 BX=0000 CX=0000 DX=AABB SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PO NC
07E9:0102 CD20              INT     20
```

O valor hexadecimal **BB** armazenado na parte menos significativa do registrador **DL** pertencente ao registrador geral **DX** foi copiado (foi atribuído ou implicado) para o registrador mais significativo do registrador geral **AX**.

Na sequência, atualize o valor do registrador de deslocamento **IP** para **0100** e execute o comando **A 0100** como indicado a seguir e informando as ações grafadas em negrito seguintes:

```
R IP 0100 <Enter>
A 0100    <Enter>

07E9:0100 MOV AX,DX <Enter>
07E9:0102 <Enter>
```

Caso queira poderá fazer uso do par de *opcodes* **8B C2** para os endereços **0100** e **0101** informados com o uso do comando **E**. Execute o comando **T** para visualizar o estado atual dos registradores durante a execução do programa, como indicado a seguir:

```
AX=AABB BX=0000 CX=0000 DX=AABB SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PO NC
07E9:0102 CD20              INT     20
```

O valor hexadecimal **AABB** armazenado no registrador geral **DX** foi copiado por inteiro para o registrador geral **AX**.

Na execução anterior foi copiado entre os registradores gerais **AX** e **DX** o conteúdo armazenado na estrutura *word*. No entanto, no exemplo anterior ao último foi copiado entre os registradores **AH** e **DL** o conteúdo armazenado em um *byte*.

É importante estar atento à instrução **MOV** que opera sempre entre pares da mesma estrutura. Não é possível, por exemplo, executar uma linha de código como **MOV AX,DL**.

Após essa noção básica execute o comando **A 0100** e informe as linhas de código indicadas em negrito a seguir:

```
07E9:0100 MOV AH,02 <Enter>
07E9:0102 MOV DL,41 <Enter>
07E9:0104 INT 21    <Enter>
07E9:0106 INT 20    <Enter>
07E9:0108           <Enter>
```

O código anterior (**MOV AH,02**) movimenta o valor hexadecimal **02** (serviço de apresentação em vídeo realizado pelo sistema operacional) para o registrador **AH** (parte mais significativa do registrador geral **AX**) e movimenta também (**MOV DL,41**) o valor hexadecimal **41** (código que representa o caractere maiúsculo **A**) para o registrador **DL** (registrador menos significativo do registrador geral **DX**). Esse tipo de ação não é possível realizar manualmente quando utilizar o comando **R** para a definição de valores nos registradores, que somente aceita valores no tamanho de um *word*. As linhas **INT 21** e **INT 20** são usadas para acionar, respectivamente, as interrupções **21** (comunicação com o monitor de vídeo) e **20** (sinalização de encerramento de programa pelo sistema operacional).

Caso queira a partir do uso do comando **E** poderá fazer a entrada do programa com uso dos conjuntos de *opcodes* **B4 02** para os endereços **0100** e **0101**, **B2 41** para os endereços **0102** e **0103**, **CD 21** para os endereços **0104** e **0105** e por fim **CD 20** para os endereços **0106** e **0107**.

Com o comando **R IP** posicione o endereço de deslocamento em **0100** e execute o comando **G** para visualizar a apresentação do caractere **A** na tela, como é mostrado a seguir:

```
A
Program terminated (0000)
```

O último programa composto por um conjunto de quatro linhas de código já pode ser gravado. No entanto, a gravação de um programa deve seguir alguns passos básicos de ação:

1. É necessário saber o número de *bytes* que o programa ocupará em disco. Pegue o endereço de deslocamento da última instrução do programa e da primeira instrução (que deve sempre ser iniciada por **0100**) e calcule a diferença entre esses valores. No programa anterior a primeira linha após a última instrução é a de valor de deslocamento **0108**. Neste caso, é fácil saber o tamanho, porém em programas maiores sugere-se usar o comando **H** para obter o valor correto. Neste caso poder-se-ia usar o comando como **H 0108 0100** que resultaria nos valores **0208** e **0008**, sendo a segunda resposta o valor da diferença.
2. Com o valor do número de *bytes* em mãos é necessário informá-lo aos registradores gerais **BX** e **CX**. O registrador geral **BX** somente será usado no caso de o valor do tamanho em *bytes* não caber no registrador geral **CX**.

Com o comando **R CX** informe o valor **0008**. Este valor determinará para o registrado **CX** a quantidade de bytes que deverá ser contada para a execução da gravação.

3. Após executar os procedimentos anteriores, use o comando **N** (*name*) para atribuir um nome ao programa. Para esse teste forneça o nome **MOSTRA.COM**. Os programas gravados com a ferramenta **Enhanced DEBUG** devem ter a extensão de identificação **.COM**. Utilize a linha de código **N MOSTRA.COM**. um detalhe importante nesta ação o nome do arquivo deve respeitar o padrão **8.3** (8 caracteres máximo para o nome e 3 caracteres máximo para o nome da extensão).

4. Em seguida é necessário gravar o programa com o comando **W** (*write*). Basta apenas acionar **W** e aparece a mensagem de que o arquivo foi gravado. Essa mensagem informa também o número de *bytes* usados na gravação.

Agora que o programa foi criado, saia do programa **Enhanced DEBUG**. No *prompt* do sistema operacional execute o programa **MOSTRA.COM** e observe a apresentação da letra **A**.

Volte ao programa **Enhanced DEBUG** e execute novamente o comando **N MOSTRA.COM** para associá-lo à memória. Depois execute o comando **L** *(load)* para que o programa associado seja carregado na memória.

Execute o comando **U** (*unassemble*) para que o código *decompilado* seja mostrado com a instrução **U 0100**, a qual mostrará uma listagem semelhante à:

```
0727:0100 B402          MOV     AH,02
0727:0102 B241          MOV     DL,41
0727:0104 CD21          INT     21
0727:0106 CD20          INT     20
0727:0108 0000          ADD     [BX+SI],AL
0727:010A 0000          ADD     [BX+SI],AL
0727:010C 0000          ADD     [BX+SI],AL
0727:010E 0000          ADD     [BX+SI],AL
0727:0110 0000          ADD     [BX+SI],AL
0727:0112 0000          ADD     [BX+SI],AL
0727:0114 0000          ADD     [BX+SI],AL
0727:0116 0000          ADD     [BX+SI],AL
0727:0118 0000          ADD     [BX+SI],AL
0727:011A 0000          ADD     [BX+SI],AL
0727:011C 0000          ADD     [BX+SI],AL
0727:011E 0000          ADD     [BX+SI],AL
```

Leve em consideração apenas as quatro primeiras linhas, as quais estão grafadas em negrito e correspondem ao programa montado anteriormente. As demais linhas apresentam dados que estão dispostos na memória do computador, mas não interferem no programa criado, os quais poderão ser diferentes em seu computador. Depois saia dos programas **Enhanced DEBUG**.

Na listagem apresentada são indicadas algumas informações importantes, tais como a posição da instrução no endereço de memória (por exemplo, do endereço **0727:0100** até **0727:0106**, onde cada uma das posições de memória é formada pelo par **SEGMENTO:DESLOCAMENTO**), a instrução em código de máquina representada pelo valor hexadecimal (como, por exemplo, **B402** existente na primeira linha de código) ao lado do endereço de memória é o código definido em *Assembly* em modo *opcode* (**B402** corresponde ao uso da instrução **MOV AH,02**, encontrada na primeira linha de código).

No pequeno programa apresentado nota-se que no endereço de memória **0727** a um pulo nos valores associados ao **DESLOCAMENTO** de dois em dois (**0100**, **0102**, **0104** e **0106**). Esse salto é calculado automaticamente pelo microprocessador e se refere a quantidade de *bytes* que cada instrução ocupa na memória quando usada. Por exemplo, a instrução **MOV AH, 02** para fazer a movimentação do valor **02** (constante) para o registrador **AH** ocupa 2 *bytes* de memória. Toda vez que se faz uma operação do tipo **MOV registrador, constante** usa-se de 2 a 3 *bytes* de memória.

Encerre a execução do programa **Enhanced DEBUG**.

5.3 - Apresentar sequência de caracteres

O desafio agora é escrever uma linha de texto na tela do monitor de vídeo. É necessário saber que se deseja escrever e codificar cada caractere do *string* em seu equivalente hexadecimal. Imagine a necessidade de apresentar a mensagem "**Alo mundo!** " (sem nenhuma acentuação, se existir).

Para efetuar a conversão da mensagem "**Alo mundo!**", é necessário estar com a tabela ASCII em mãos para saber quais são os valores numéricos hexadecimais de cada caractere da frase a ser escrita. No caso da mensagem em uso os valores hexadecimais são os apresentados na Tabela 5.1.

Tabela 5.1 - Valores ASCII da mensagem "Alo mundo!"

A	l	o		M	u	n	d	o	!
41	**6C**	**6F**	**20**	**6D**	**75**	**6E**	**64**	**6F**	**21**

É preciso colocar os códigos do *string* na memória. Carregue um dos programas **Enhanced DEBUG**, execute o comando **E** e informe o endereço de deslocamento a partir de **0200**.

Informe cada *byte* da tabela anterior, separando-os por espaços em branco. Ao final acrescente o código hexadecimal **24** que representa o símbolo **$**, o qual define para os programas **Enhanced DEBUG** o fim do *string*. A seguir são apresentados os dados após a entrada dos valores hexadecimais da mensagem "**Alo mundo!**".

```
E 0200 41 6C 6F 20 6D 75 6E 64 6F 21 24
```

É necessário informar para o registrador geral **DX** o início do endereço de deslocamento em que se encontra a sequência de caracteres. Neste caso, o endereço é **0200**. É necessário também definir o valor hexadecimal **09** (função para apresentação de uma sequencia de caracteres no monitor de vídeo) no registrador mais significativo **AH**, para que um *string* seja impresso. Execute o comando **A 0100**.

```
07E9:0100 MOV AH,09   <Enter>
07E9:0102 MOV DX,0200 <Enter>
07E9:0105 INT 21      <Enter>
07E9:0107 INT 20      <Enter>
07E9:0109             <Enter>
```

No código anterior não está sendo utilizado o código **02** para o registrador mais significativo **AH**, mas sim o código **09** quando do uso da instrução **MOV AH,09** (caso prefira poderá usar o *opcode* **B4 09**). O código **02** aciona o recurso de impressão de apenas um caractere, enquanto o código **09** aciona o serviço de apresentação de uma sequência de caracteres.

A instrução **MOV DX,0200** (*opcode* **BA 00 02**) para ser usada ocupa três posições de memória, pois armazena no registrador **DX** o endereço de memória onde se encontra o *string* a ser apresentado. O registrador geral **DX** é normalmente usado para apontar o endereço onde certo dado se encontra na memória.

Execute o comando **G** para que a mensagem seja apresentada, como é indicado a seguir:

```
Alo mundo!
Program terminated (0000)
```

O comando **U** permite visualizar a sequência de instruções de um programa. No entanto, é possível também visualizar as informações referentes ao armazenamento dos códigos hexadecimais dos caracteres da mensagem. Utilize o comando **D** (*dump*) com o endereço do deslocamento desejado. Neste caso, acione o comando:

```
D 0200 <Enter>
```

Observe os detalhes apresentados na sequência:

```
07E9:0200  41 6C 6F 20 6D 75 6E 64-6F 21 24 00 00 00 00 00   Alo mundo!$.....
07E9:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07E9:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07E9:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07E9:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07E9:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07E9:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07E9:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Observe o trecho em negrito. Os demais dados são restos de conteúdo da memória e certamente serão apresentados de forma diferente no computador do leitor. Perceba a definição do caractere **$** indicando o fim da mensagem. A informação apresentada é composta das informações de segmento e de deslocamento (segmento:deslocamento) e de 16 *bytes* hexadecimais com os códigos referentes aos caracteres que também são apresentados no extremo direito da relação.

Na parte extrema à direita da relação há também alguns pontos (**.**). Isso ocorre quando o caractere é um ponto ou outro caractere especial, ou seja, o comando **D** apresenta apenas 96 códigos de caracteres dos 256 códigos encontrados na tabela ASCII. Os outros 160 códigos são apresentados como pontos.

Agora o programa de escrita de mensagem deve ser gravado. Este programa ocupa um espaço de memória bem maior que o programa anterior, pois a sequência de caracteres está armazenada a partir do endereço de deslocamento **0200**. Para verificar essa informação, execute o comando:

```
U 0200 <Enter>
```

Observe os detalhes apresentados na listagem a seguir:

```
07E9:0200 41            INC     CX
07E9:0201 6C            INSB
07E9:0202 6F            OUTSW
07E9:0203 206D75        AND     [DI+75],CH
07E9:0206 6E            OUTSB
07E9:0207 646F          FS:OUTSW
07E9:0209 2124          AND     [SI],SP
07E9:020B 0000          ADD     [BX+SI],AL
07E9:020D 0000          ADD     [BX+SI],AL
07E9:0211 0000          ADD     [BX+SI],AL
07E9:0213 0000          ADD     [BX+SI],AL
07E9:0215 0000          ADD     [BX+SI],AL
07E9:0217 0000          ADD     [BX+SI],AL
07E9:0219 0000          ADD     [BX+SI],AL
07E9:021B 0000          ADD     [BX+SI],AL
07E9:021D 0000          ADD     [BX+SI],AL
07E9:021F 0000          ADD     [BX+SI],AL
-
```

O trecho marcado em negrito corresponde exatamente à sequência de caracteres definida para a escrita do *string* entre os endereços **0200** e **0209**. Considere a primeira posição de memória após a indicação do endereço **0209**, ou seja, considere o valor do endereço **020B**.

A sequência de caracteres que forma o *string* "**Alô mundo!$** " é visível na definição dos valores de códigos de máquina existentes após a indicação dos endereços de memória, como mostra o trecho em negrito de código a seguir.

```
07E9:0200 41            INC     CX
07E9:0201 6C            INSB
07E9:0202 6F            OUTSW
07E9:0203 206D75        AND     [DI+75],CH
07E9:0206 6E            OUTSB
07E9:0207 646F          FS:OUTSW
07E9:0209 2124          AND     [SI],SP
```

Não se preocupe com os mnêmicos apresentados do lado direito dos valores hexadecimais que forma o texto do *string* Em seguida calcule a diferença entre os valores final e inicial para determinar o tamanho do programa em memória. Note que o programa começa no endereço de memória **0100** e faz uso da sequência de caracteres definida a partir do endereço **0200**, a qual se estende até o endereço **020B**. Desta forma, basta fazer o cálculo com a instrução:

```
H 020B 0100 <Enter>
```

Os resultados apresentados são **030B** para a soma e **010B** para a diferença. Considerando o valor de diferença **010B** coloque-o no registrador **CX** com a instrução:

```
R CX 010B <Enter>
```

Assim sendo, execute o comando **R** e atente para os pontos marcados em negrito como mostrado a seguir:

```
AX=0000  BX=0000  CX=010B  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=07E9  ES=07E9  SS=07E9  CS=07E9  IP=0100   NV UP EI PL NZ NA PO NC
07E9:0100 B409          MOV     AH,09
```

Execute o comando **N MENSA.COM** e depois o comando **W**. Saia de um dos programas **Enhanced DEBUG** e execute o programa no *prompt* do sistema operacional para que a mensagem seja apresentada.

Há outra forma um pouco mais simples de definir sequências de caracteres na memória. Para tanto, execute a instrução:

```
E 0200 "Ola, leitor" 24 <Enter>

A 0100                  <Enter>
07E9:0100 MOV AH,09     <Enter>
07E9:0102 MOV DX,0200   <Enter>
07E9:0105 INT 21        <Enter>
07E9:0107 INT 20        <Enter>
07E9:0109               <Enter>
```

Observe que sequência de caracteres está definida entre aspas. É importante manter ao final o código hexadecimal **24** para a definição do caractere de controle **$**. Com o comando **G** execute o programa e observe a apresentação da mensagem **Olá, leitor**.

5.4 - Apresentar valores binários

A seguir serão criados alguns programas que apresentam valores numéricos binários. Para este contexto são indicados novos comandos de operação.

Mas antes de apresentar os valores numéricos, é necessário considerar outros detalhes relacionados ao estouro da capacidade de cálculo. Por exemplo, carregue novamente um dos programas **Enhanced DEBUG** e para o registrador geral **AX** forneça o valor hexadecimal **FFFF**, depois forneça para o registrador geral **BX** o valor hexadecimal **0001** a partir das instruções:

```
R AX FFFF <Enter>
R BX 0001 <Enter>
```

Execute o comando **R** para ver a listagem da posição de memória e observe se a primeira linha está semelhante à indicação em negrito a seguir:

```
AX=FFFF BX=0001 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3                RET
```

Na segunda linha da encontra-se um conjunto de oito pares de registradores de estado (*flags*), como indicado em negrito:

```
AX=FFFF BX=0001 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3                RET
```

Direcione a sua atenção para o último par de valores que deve estar sinalizado com as letras **NC** - *No Carry* (dependendo da ocorrência em uso, esse par de valores poderá estar identificado com as letras **CY** - *Carry Yes*). Os valores **NC** e **CY** são as respostas geradas para o estado de operação do registrador **CF** (*Carry Flag*). O registrador **CF** será sinalizado com o valor **CY** quando determinado cálculo gerar um estouro da capacidade máxima de representação numérica na memória.

Os registradores gerais **AX** e **BX** possuem, respectivamente, os valores hexadecimais **FFFF** e **0001**. Assim, execute a instrução:

```
E 0100 01 D8 <Enter>
```

Os valores **01** e **D8** são os *opcodes* utilizados para realizar a adição do valor do registrador geral **BX** sobre o valor do registrador geral **AX**. Com a certeza de que o registrador de deslocamento **IP** aponta para o endereço **0100,** execute o comando **T** e observe o que ocorreu.

```
AX=0000 BX=0001 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL ZR AC PE CY
07E9:0102 F726FC26          MUL     WORD PTR [26FC]                DS:26FC=0000
```

Os valores apresentados nos registradores gerais **AX** e **BX** são, respectivamente, **0000** e **0001**. O último valor do registrador de estado que era NC agora é **CY**. Se prestar atenção, verá que outros três registradores de estado também foram afetados (NZ para **ZR**, NA para **AC** e PO para **PE**). Mas no momento, interessa apenas fazer um rápido estudo do último valor do registrador de estado, pois com ele será realizada a apresentação de valores binários.

O que ocorreu? Já é sabido que os registradores gerais operam com valores numéricos de 16 *bits*, e quando um valor ultrapassa esta capacidade, ocorre automaticamente um estouro. Foi exatamente isso que ocorreu ao tentar somar o valor hexadecimal **0001** com o valor hexadecimal **FFFF** (que é o maior possível).

Ocorreu um estouro no valor do registrador **AX** que deveria ser **10000** e apresenta apenas o valor **0000**, ou seja, são considerados apenas os quatro dígitos da direita. O valor **1** (primeiro dígito à esquerda "desprezado") gerou um estouro da capacidade numérica de armazenamento. Ao ocorrer o estouro, o décimo quinto *bit* (está em uso um valor hexadecimal de 16 *bits.* Se fosse um valor de 8 *bits,* seria então considerado o sétimo *bit)*, o registrador de *Carry*, é automaticamente "setado" em **1**, indicando a ocorrência do estouro, pois o registrador de estado *Carry* indica o conceito de vai-um quando ocorre um estouro, e neste caso esse registrador passa do estado **NC** (sem estouro, quando está com valor binário zero) para o estado **CY** (com estouro, passando a possuir o binário um).

Se neste exato momento for executada a soma do valor hexadecimal **0001** no registrador geral **AX**, o registrador de estado *Carry* volta ao seu estado original sinalizado como **NC**. Posicione o registrador **IP** no endereço **0100** e execute o comando **T**, que apresenta a indicação do registrador de estado *Carry*, como exibido em negrito:

```
AX=0001 BX=0001 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PO NC
07E9:0102 F726FC26          MUL     WORD PTR [26FC]                DS:26FC=0000
```

A primeira forma de apresentação numérica será de valores binários. Neste caso, serão utilizados apenas os recursos do registrador de estado *Carry* para a finalidade de representar um valor numérico na forma binária. Considere a apresentação do valor hexadecimal **00AA** (valor **170** em decimal) na forma binária **10101010**. Forneça para o registrador geral **AX** o valor hexadecimal **0000**, para o registrador **BX** o valor hexadecimal **00AA** e para o registrador **IP** o valor hexadecimal **0100**. Acione o comando **R** e veja o resultado conforme a seguir:

```
AX=0000 BX=00AA CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 01D8              ADD     AX,BX
```

Para conseguir o intento de escrever um número em formato binário, é necessário utilizar o comando **RCL** *(Rotate through Carry Left)*, que tem por finalidade mover para a esquerda o ponteiro de *bit* dentro do registrador de estado *Carry Flag* (**CF**), percorrendo todos os *bits* existentes, a partir da primeira posição (posição zero) até a última (que pode ser a posição sete para a varredura de um *byte* ou a posição dezesseis para a varredura de um *word)*. Entre com as instruções:

```
A 0100              <Enter>
07E9:0100 RCL BL, 1 <Enter>
07E9:0102           <Enter>
```

A instrução **RCL BL, 1** tem por finalidade somar o valor **1** à parte baixa do registrador geral **BX**.

O comando **RCL** quando executada altera o estado dos registradores de estados (*flags*) **CF** e **OF**, possuindo como possibilidade de trabalho as operações sobre:

- registrador, 1 (para este tipo de ação usa 2 *bytes* de memória);
- memória, 1 (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, CL (para este tipo de ação usa 2 *bytes* de memória);
- memória, CL (para este tipo de ação usa de 2 a 4 *bytes* de memória).

Observação

A instrução **RCL** também pode operar a movimentação do registrador de estado *Carry Flag* para dados do tipo *word*.

Isso fará com que o valor hexadecimal **00AA** (em modo binário **10101010**) seja somado com o valor **1**. Toda vez que ocorrer internamente um estouro da soma do valor **1** do registrador menos significativo **BL** com o valor armazenado no registrador geral **BX**, este sinalizará o registrador de estado *Carry Flag* com o valor **CY** (**1**) ou **NC** (**0**).

A primeira ação ocorre no *bit* zero (primeiro *bit* da direita para a esquerda). Quando for solicitada uma segunda ação do comando **RCL**, ele passa a operação para o segundo *bit*. É necessário repetir a ação por oito vezes (*byte*) para conseguir obter os oito valores binários relacionados ao valor hexadecimal **00AA**. Tendo uma ideia do processo de execução, acompanhe cada passo indicado a seguir:

1. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **1** numa folha de papel, pois o registrador *Carry Flag* está indicando **CY**.
2. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **0** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **NC**.
3. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **1** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **CY**.
4. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **0** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **NC**.
5. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **1** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **CY**.
6. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **0** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **NC**.
7. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **1** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **CY**.
8. Atualize o registrador **IP** com o valor de endereço **0100**, execute o comando **T** e anote o valor **0** numa folha de papel ao lado do valor anteriormente obtido, pois o registrador de estado *Carry Flag* está indicando **NC**.

Observe na Tabela 4.2 o resumo da anotação dos valores de *Carry* durante os oito passos com o comando **T**. Ao executar as ações indicadas, confira cada etapa do processo com a Tabela 5.2.

Tabela 5.2 - Valores de *carry* após uso do comando T

Valor Inicial do Registrador BX: 00AA (ou apenas AAh)					
Ação	**IP**	**Execução**	**Carry Flag**	**Anotação**	**(BX) BL**
1	R IP 0100	T	CY	1	0054
2	R IP 0100	T	NC	0	00A9
3	R IP 0100	T	CY	1	0052
4	R IP 0100	T	NC	0	00A5
5	R IP 0100	T	CY	1	004A
6	R IP 0100	T	NC	0	0095
7	R IP 0100	T	CY	1	002A
8	R IP 0100	T	NC	0	0055

O valor do registrador geral **BL** está sendo indicado na tabela anterior para facilitar o acompanhamento de cada passo. Em nenhum momento o valor existente nesse registrador será usado. Será usada apenas a informação **CY** e **NC** do registrador de estado *Carry Flag*.

Se por curiosidade forem executados mais uma vez os comandos descritos na tabela anterior, obter-se-á o valor **00AA** no registrador geral **BX**, provando que o comando **RCL** efetuou a varredura *bit* a *bit* sempre no sentido esquerdo.

Até o presente momento foram informadas as instruções para movimentação dos *bits* pela instrução **RCL**, sendo apenas uma parte do processo de apresentação de um valor hexadecimal na forma binária.

É necessário ainda fazer com que o registrador de estado *Carry Flag* quando "setado", ou seja, sinalizado (**CY**), forneça para o programa condição de escrever o valor **1** (um) e quando não estiver "setado" (**NC**), forneça condição de escrever **0** (zero). Além disso, é necessário desenvolver um laço de repetição para automatizar o processo de tradução de valor hexadecimal em valor binário.

O laço de repetição será construído com o comando **LOOP** que determina o número de vezes a ser executado o trecho de código subordinado ao laço. Essa instrução utiliza o registrador geral **CX** para armazenar o valor do contador, ocupa 2 *bytes* de memória e não afeta os registradores de estado. No exemplo apresentado anteriormente o programa deve executar o trecho de conversão definido no laço oito vezes.

Primeiramente saia do programa **Enhanced DEBUG** em uso e faça o seu carregamento para a memória novamente a fim de inicializa-lo de forma zerada. Acrescente ao registrador geral **BX** o valor hexadecimal **00AA**, depois execute o comando **A 0100** e informe as linhas seguintes:

```
07E9:0100 MOV  CX,0008 <Enter>
07E9:0103 RCL  BL,1    <Enter>
07E9:0105 LOOP 0103    <Enter>
07E9:0107 INT  20      <Enter>
07E9:0109              <Enter>
```

A primeira linha **07E9:0100** faz a colocação do valor hexadecimal **0008** no registrador geral **CX**. A segunda linha **07E9:0103** movimenta um *bit* à esquerda no registrador de estado *Carry Flag*. A terceira linha **07E9:0105** executa o laço desviando a execução do programa para a linha de deslocamento **0103**. Esse processo será executado oito vezes. A cada execução do comando **LOOP** (que será internamente considerado como comando **LOOPW** para o programa **Enhanced DEBUG**) será realizado um decréscimo de **0001** até que o registrador geral **CX** chegue ao valor **0000**, quando o fluxo de execução do programa será desviado para a quarta linha do programa (**07E9:0107**).

Para realizar um teste de execução, utilize o comando **T** que possibilita a execução de um programa passo a passo até que seja apresentada a mensagem de término do programa. A seguir é apresentada a sequência completa de execução dessa etapa. Observe detalhadamente cada trecho marcado em negrito na sequência indicada a seguir.

```
-T
AX=0000 BX=00AA CX=0008 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 NV UP EI PL NZ NA PO NC
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=0054 CX=0008 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO CY
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=0054 CX=0007 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO CY
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=00A9 CX=0007 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO NC
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=00A9 CX=0006 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO NC
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=0052 CX=0006 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO CY
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=0052 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO CY
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=00A5 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO NC
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=00A5 CX=0004 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO NC
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=004A CX=0004 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO CY
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=004A CX=0003 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO CY
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=0095 CX=0003 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO NC
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=0095 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO NC
07E9:0103 D0D3              RCL     BL,1
```

```
-T
AX=0000 BX=002A CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 OV UP EI PL NZ NA PO CY
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=002A CX=0001 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0103 OV UP EI PL NZ NA PO CY
07E9:0103 D0D3              RCL     BL,1

-T
AX=0000 BX=0055 CX=0001 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 NV UP EI PL NZ NA PO NC
07E9:0105 E2FC              LOOPW   0103

-T
AX=0000 BX=0055 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0107 NV UP EI PL NZ NA PO NC
07E9:0107 CD20          INT     20

-T

Program terminated (0000)
```

Observe pela sequência anterior que o registrador geral **CX** é decrementado de **0001** em **0001** até chegar a **0000**, em que a execução passo a passo deve ser interrompida.

Falta pouco para conseguir a apresentação do valor no formato binário. A próxima etapa é interpretar no registrador de estado *Carry Flag* cada **CY** como **1** e cada **NC** como **0**. Será utilizado o comando **ADC** (*Add Carry Flag*). O comando **ADC** tem como diferença do comando **ADD** a capacidade de, além de adicionar dois valores, efetuar a adição do *bit* do estado do registrador de estado *Carry Flag*.

Para conseguir o efeito de apresentação de valor binário (que será formado pelos caracteres **0** e **1**, os quais possuem, respectivamente, os valores ASCII **30** (indica o caractere **0**) e **31** (indica caractere **1**) em formato hexadecimal), será adicionado o valor atual do registrador de estado *Carry Flag* com o valor hexadecimal **30**. Neste caso, se o *bit* do *Carry Flag* for **NC** (valor 0), será apresentado o caractere de código hexadecimal **30;** caso o *bit* do *Carry Flag* venha a ser **CY** (valor 1), será apresentado o caractere de código hexadecimal **31**.

Assim sendo, defina para o registrador **BX** o valor **00AA** e para o registrador **IP** o valor **0001**, depois insira o código a seguir a partir do deslocamento **0100**. Faça-o com o comando **A 0100**.

```
07E9:0100 MOV  AH,02   <Enter>
07E9:0102 MOV  CX,0008 <Enter>
07E9:0105 MOV  DL,00   <Enter>
07E9:0107 RCL  BL,1    <Enter>
07E9:0109 ADC  DL,30   <Enter>
07E9:010C INT  21      <Enter>
07E9:010E LOOP 0105    <Enter>
07E9:0110 INT  20      <Enter>
07E9:0112              <Enter>
```

A primeira linha (**07E9:0100**) do código anterior define o valor hexadecimal **02** para o registrador geral **DL**. Lembre-se de que esse valor informa ao sistema operacional que é para imprimir um único caractere armazenado no registrador **DL** (registrador geral menos significativo do registrador geral **DX**).

A segunda linha (**07E9:0102**) define o valor total do contador de laço, representado pelo registrador geral **CX**. É importante lembrar que o registrador geral **CX** é exclusivo para a definição de contadores, por esta razão a instrução **LOOP** o utiliza. Outro ponto a ser observado é que a instrução **LOOP** efetua sempre um laço decrescente, diminuindo o valor do registrador **CX**.

A terceira linha (**07E9:0105**) movimenta o valor **00** para o registrador geral menos significativo **DL**. Isso faz com que o registrador seja limpo (zerado) toda vez que for executada essa linha de código. Esta atitude retira o último valor utilizado. Lembre-se de que os registradores possuem o efeito de ir acumulando a carga de valores que são a eles imputados.

A quarta linha (**07E9:0107**) utiliza a instrução **RCL** que movimenta para a esquerda um *bit* durante todo o *byte* utilizado do registrador geral menos significativo **BL** para representar o valor **00AA** (o valor interno realmente considerado é **AA;** o **00** à esquerda não possui nenhum valor).

A quinta linha (**07E9:0109**) utiliza a instrução **ADC** que efetua a adição do valor hexadecimal **30** (caractere **0** - zero) ao valor existente no registrador menos significativo **DL**, que pode ser **1** ou **0,** dependendo do valor do registrado de estado *Carry Flag*. Toda vez que a instrução **ADC** efetua uma adição, ela também muda o valor do registrador de estado *Carry Flag* de **NC** (não sinalizado) para **CY** (sinalizado) e vice-versa. Desta forma, é possível simular a geração e a apresentação de um valor binário convertido a partir de um valor hexadecimal definido.

A sexta linha (**07E9:010C**) efetua a apresentação na tela do monitor de vídeo do valor existente no registrador geral menos significativo **DL** pela instrução **INT 21**.

A sétima linha (**07E9:010E**) efetua o decremento em **1** do valor existente no registrador geral **CX** e transfere a execução do fluxo de programa para a linha de código identificada pelo valor **0105**. Tudo isso é controlado pela instrução **LOOPW**.

A oitava (**07E9:0110**) e última linha de código do programa efetua a devolução do controle operacional do programa ao sistema operacional por meio da instrução **INT 20**.

Observe a seguir um paralelo entre o código escrito em *opcodes* e linguagem *Assembly* equivalentes do programa anterior que pode ser obtido a partir da instrução **U 0100 0110**.

```
07E9:0100    B4 02       MOV    AH,02
07E9:0102    B9 08 00    MOV    CX,0008
07E9:0105    B2 00       MOV    DL,00
07E9:0107    D0 D3       RCL    BL,1
07E9:0109    80 D2 30    ADC    DL,30
07E9:010C    CD 21       INT    21
07E9:010E    E2 F5       LOOPW  0105
07E9:0110    CD 20       INT    20
```

Execute o programa com o comando **G** para visualizar na forma binária o valor hexadecimal **10101010** definido para o registrador geral **BX**.

5.5 - Registradores de estado

No tópico anterior foi bastante utilizado o registrador de estado **CF** (*Carry Flag*), que manifesta sua ação quando uma operação de cálculo matemático é efetuada. No entanto, o conjunto de registradores de estado possui outros registradores que se alteram quando uma operação de cálculo é realizada, e por esta razão merecem alguma atenção. Por exemplo, quando se utiliza os comandos **ADD** e **SUB** os registradores **AF**, **CF**, **OF**, **PF**, **SF** e **ZF** são alterados; ao se usar os comandos **MUL CF** e **OF**; o uso do comando **DIV** não altera nenhum registrador de *flag*.

O modelo 8086/8088 possui nove registradores de estado, sendo este valor de 32 em modelos mais recentes de microprocessadores da Intel e AMD. Os registradores de estado são elementos que sinalizam um estado de ação para que o microprocessador "saiba" como efetuar certa tarefa.

Entre o conjunto disponível estão os registradores de estado que serão apresentados neste tópico: registrador de estado *Zero Flag* (**ZF**), que pode estar sinalizado com os valores **ZR** - *zero* - ou com **NZ** - *not zero*; registrador de estado *Sign Flag* (**SF**), que pode estar sinalizado com os valores **NG** - *negative* - ou com **PL** - *plus* – positivo, registrador de estado *Overflow Flag* (**OF**), que pode estar sinalizado com os valores **OV** - *overflow* ou com **NV** - *not overflow*, o registrador de estado **PF** (*Parity Flag*) que indica se a paridade após uma ação é impar com valor **PO** - *parity odd* ou par com valor **PE** - *parity even* e o registrador de estado **AF** (*Auxiliary Carry Flag*) que opera com os valores **NA** - *not auxiliary carry* e **AC** - *auxiliary carry*.

Os registradores de estado *Direction Flag* (**DF**) que opera com os valores **DN** - *direction down* e **UP** - *direction up* é usado nas ações de contagem com os registradores *Source Index* (**SI**) e *Destination Index* (**DI**) quando da operação e tratamento de sequências de caracteres, *Interrup Flag* (**IF**) que opera com os valores **EI** - *enabled interrupt* e **DI** - *disabled interrupt* quando ações de interrupção são habilitadas e desabilitadas e *Trap Flag* (**TF**) que não tem seus valores de estado visíveis no programa **Enhanced DEBUG** e efetua uma interrupção na execução do programa, permitindo ao comando **T** por exemplo, executar um programa passo-a-passo e ter acesso a ação de cada uma das instruções separadamente. Os registradores **DF**, **IF** e **TF** não serão demonstrados além do que aqui está sendo exposto por não ser isto tão necessário.

Para realizar testes com os registradores de estado *Zero Flag*, *Sign Flag* e *Overflow Flag*, sugere-se sair do programa **Enhanced DEBUG** em uso e fazer novo carregamento do programa em memória, para que todos os registradores sejam iniciados com seus valores padrão, como é indicado em seguida após a execução do comando **R**:

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3        RET
```

Observe atentamente o lado direito da segunda linha. Em especial, note os registradores sinalizados com os valores **NV** (primeiro na lista), **PL** (quarto na lista) e **NZ** (quinto na lista) que estão sinalizados em negrito.

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3        RET
```

Para um primeiro teste será verificada uma operação de subtração de valores. Forneça o valor **0005**, respectivamente, para os registradores gerais **AX** e **BX**. Assim sendo execute as instruções:

```
R AX 0005 <Enter>
R BX 0005 <Enter>
```

Observe a seguir como devem estar os dados em memória após executar o comando **R**.

```
AX=0005 BX=0005 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 C3        RET
```

Para realizar a subtração, execute a instrução **E 0100 29 D8** e acione a tecla **<Enter>**. Lembre-se de que este é o código de máquina para realização de uma subtração.

Observação

Verifique sempre se o registrador **IP** está com o valor de deslocamento **0100**. Caso não esteja, lembre-se de executar o comando **R IP** e fornecer o valor correto.

Na sequência acione o comando **R** para que seja apresentado o estado atual dos registradores. Confirme se o registrador **IP** aponta para o endereço **0100** como é mostrado em seguida:

```
AX=0005 BX=0005 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 29D8              SUB     AX,BX
```

Execute o comando **T** para que o resultado da subtração seja armazenado no registrador geral **AX** (neste caso será armazenado o valor hexadecimal **0000**) e a informação do registrador de estado *Zero Flag* indicado como NZ seja apresentado sinalizado com o identificador **ZR**, indicando que a operação executada zerou o valor do registrador geral **AX**. Além do registrador de estado *Zero Flag* o registrador de estado *Parity Flag* também é alterado de PO para **PE**.

```
AX=0000 BX=0005 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL ZR NA PE NC
07E9:0102 F726FC26          MUL     WORD PTR [26FC]              DS:26FC=0000
```

O registrador de estado *Overflow Flag* é sinalizado todas as vezes que ocorre um estouro da capacidade numérica a ser armazenada na estrutura de um *word*. Para testar essa ocorrência, entre para o registrador geral **AX** o valor hexadecimal **6000** e para o registrador geral **BX** o valor hexadecimal **7000**. Ajuste o registrador de deslocamento **IP** para o endereço de deslocamento **0100** e entre para a adição o código em linguagem de máquina **01** e **D8**, a partir do endereço **0100** por meio do comando **E**. Assim sendo, execute as instruções:

```
R AX 6000    <Enter>
R BX 7000    <Enter>
R IP 0100    <Enter>
E 0100 01 D8 <Enter>
```

Observe se os dados de seu computador estão semelhantes aos pontos marcados em negrito após execução do comando **R**:

```
AX=6000 BX=7000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL ZR NA PE NC
07E9:0100 01D8              ADD     AX,BX
```

Execute o comando **T** e observe o valor armazenado no registrador **AX** após a operação (valor hexadecimal negativo **D000**).

```
AX=D000 BX=7000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 OV UP EI NG NZ NA PE NC
07E9:0102 F726FC26          MUL     WORD PTR [26FC]              DS:26FC=0000
```

Observe também a indicação dos registradores de estado *Overflow Flag* e *Sign Flag*, os quais mostram, respectivamente, os valores **OV** (indicando que ocorreu um estouro da capacidade de cálculo, ou seja, ocorreu um erro) e **NG** (indicando que o valor apresentado é considerado negativo).

O registrador de estado *Sign Flag* também pode manifestar-se quando ocorrer a subtração de um valor menor em relação a um valor maior. Por exemplo, estabeleça para o registrador **AX** o valor hexadecimal **0000** e para o registrador geral **BX** o valor hexadecimal **0001**. Ajuste o registrador de deslocamento **IP** para o endereço de deslocamento **0100** e entre para a subtração o código em linguagem de máquina **29** e **D8**, a partir do endereço **0100** por meio do comando **E** a partir das instruções:

```
R AX 0000    <Enter>
R BX 0001    <Enter>
R IP 0100    <Enter>
E 0100 29 D8 <Enter>
```

Observe se os dados de seu computador estão semelhantes aos pontos marcados em negrito após execução do comando **R**:

```
AX=0000 BX=0001 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 OV UP EI NG NZ NA PE NC
07E9:0100 29D8              SUB     AX,BX
```

Execute o comando **T** e observe o valor armazenado no registrador geral **AX** após a operação (valor hexadecimal negativo **FFFF**).

```
AX=FFFF BX=0001 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI NG NZ AC PE CY
07E9:0102 F726FC26          MUL     WORD PTR [26FC]               DS:26FC=0000
```

O registrador de estado *Overflow* com o valor **NV**, anteriormente marcado como **OV** sinaliza a ocorrência de um estouro na capacidade numérica do cálculo efetivado.

Outros dois registradores de estado que tiveram seus valores alterados são: *Auxiliary Carry Flag* (indicado que ocorreu na operação o conceito pega-um) que teve seu valor de **NA** alterado para **AC** e *Parity Flag* (indicando se a quantidade de *bits* manipulado na memória é par ou impar) que neste caso mantem o valor **PE** ao indicar que o resultado **FFFF** é um valor com quantidade de *bits* par.

É comum utilizar o registrador de estado (*flags*) em operações de programação, principalmente naquelas que envolvem saltos condicionais. Cada *flag* que compõe o conjunto do registrador de estado possui um valor particular, como mostra a Tabela 5.3.

Tabela 5.3 - Valores dos registradores de *flag*

Bit	Flag	Nome	Finalidade	Ativado	Desativado
11	**OF**	Overflow	Sinaliza uma operação aritmética com estouro da capacidade de cálculo.	**OV**	**NV (*)**
10	**DF**	Direction	Determina a direção de contagem de SI/DI para cima ou para baixo em sequências de caracteres.	**DN**	**UP (*)**
9	**IF**	Interrupt	Identifica se alguma interrupção de hardware está ou não ativa quando do uso do comando **INT**.	**EI (*)**	**DI**
8	**TF**	Trap	Permite que um programa seja executado linha a linha durante a fase de depuração (comando **T**).	Não é visível para o **DEBUG**	
7	**SF**	Sign	Indica a obtenção de um valor positivo ou negativo em uma operação aritmética.	**NG**	**PL (*)**
6	**ZF**	Zero	Indica se o resultado após uma operação aritmética é zero, ou se o resultado de comparação após uma ação lógica é igual.	**ZR**	**NZ (*)**
4	**AF**	Auxiliary Carry	Indica a sinalização de *carry* quando da adição e subtração de valores binários baseados no formato BCD.	**AC**	**NA (*)**
2	**PF**	Parity	Indica se a paridade de um valor é par ou ímpar após uma operação lógica ou aritmética.	**PE**	**PO (*)**
0	**CF**	Carry	Indica a sinalização de *carry* (vai um) para a próxima posição, após a efetivação de uma operação aritmética.	**CY**	**NC (*)**

Na Tabela 4.3 as colunas **Ativado** e **Desativado** indicam os valores que cada um dos registradores de estado pode assumir. Internamente os valores de ativação serão **1** e os valores de desativação serão **0**. As células da Tabela 4.3 com a indicação dos *status* assinalados com asterisco mostram os valores dos registradores definidos automaticamente por padrão pelo sistema.

Assim como é possível fazer a definição e alteração de valores para os registradores gerais **AX**, **BX**, **CX** e **DX** e de apontamento como **IP** os registradores de estado podem ter seus valores manualmente alterados a partir do uso do comando **R F** (*Register Flag*), que pode ser usado como **RF**.

Quando o comando **RF** é usado ocorre a apresentação de um *prompt* com os valores dos registradores de estado que estão ativos. Neste instante basta informar o valor referente ao registrador que se deseja alterar para que a mudança ocorra. Por exemplo, para mudar o valor de **CY** para **NC** do registrador de estado **CF** basta informar no *prompt* apresentado pelo comando **RF** o valor **NC**.

5.6 - Instruções de salto condicional

Anteriormente foi apresentado o comando **LOOPW** que efetua a ação de laço incondicional tendo seu controle definido junto ao registrador geral **CX**. No entanto, existem outros comandos de salto controlados por ações condicionadas aos registradores de estado, sendo esses comandos reesposáveis por ações de saltos condicionais.

Os comandos usados para a realização de saltos condicionais são recursos que possibilitam executar ações de desvio em um programa, a partir da verificação de condições e valores dos registradores de estado (*flags*). Os comandos de saltos condicionais ocupam de 2 a 4 *bytes* de memória, sendo:

- **JZ** (*jump on zero*) - salta para um determinado endereço de deslocamento caso o registrador de estado *Zero Flag* esteja sinalizado como zero (**ZR**);
- **JNZ** (*jump on not zero*) - salta para um determinado endereço de deslocamento caso o registrador de estado *Zero Flag* esteja sinalizado como não zero (**NZ**).

Para demonstrar o salto condicional com o comando **JNZ**, considere apresentar na tela de vídeo a letra **A** enquanto o registrador geral menos significativo **BL** for maior que zero. Quando o registrador geral **BL** estiver com zero, o programa apresenta a letra **B**, e encerra a sua execução. Informe o código seguinte a partir do endereço de deslocamento **0100** com o comando **A**:

```
07E9:0100 MOV BL,03 <Enter>
07E9:0102 MOV AH,02 <Enter>
07E9:0104 MOV DL,41 <Enter>
07E9:0106 INT 21    <Enter>
07E9:0108 SUB BL,01 <Enter>
07E9:010B JNZ 0106  <Enter>
07E9:010D MOV DL,42 <Enter>
07E9:010F INT 21    <Enter>
07E9:0111 INT 20    <Enter>
07E9:0113           <Enter>
```

A primeira linha do programa (**07E9:0100**) determina o valor hexadecimal **03** ao registrador geral menos significativo **BL**. Esse registrador será utilizado para armazenar o valor de contagem do laço de repetição até que ele seja zero, ou seja, até que o registrador de deslocamento *Zero Flag* seja sinalizado com **NZ**.

A segunda linha do programa (**07E9:0102**) determina o código hexadecimal **02** de controle da apresentação de caracteres (apenas um caractere) ao registrador geral mais significativo **AH**.

A terceira linha do programa (**07E9:0104**) determina o armazenamento do código hexadecimal **41** corres-pondente ao valor da letra **A**. O código **41** é armazenado no registrador geral menos significativo **DL**.

A quarta linha do programa (**07E9:0106**), por meio da instrução **INT 21,** apresenta o caractere armazenado no registrador **DL**.

A quinta linha do programa (**07E9:0108**) efetua com a instrução **SUB** (*subtract*) a subtração de **01** (**SUB BL,01**) no valor armazenado no registrador **BL**, fazendo com que o valor passe a ser **02**, depois **01** e por último **00**.

A sexta linha do programa (**07E9:010B**) salta o programa voltando a execução para o endereço de deslocamento **0106** (quarta linha, que efetua uma nova apresentação do caractere armazenado no registrador **DL**). Esse ciclo se repete até o momento em que o registrador menos significativo **BL** se torna **00** e gera o sinal **ZR** para o registrador de estado *Zero*.

A sétima linha do programa (**07E9:010D**) determina o armazenamento do código hexadecimal **42** correspon-dente ao valor da letra **B** no registrador geral menos significativo **DL**. Essa ação somente ocorre quando a sexta linha deixar de operar, passando o fluxo de execução do programa para a próxima linha.

A oitava linha do programa (**07E9:010F**), por meio da instrução **INT 21,** apresenta o caractere armazenado no registrador **DL**, que nesse momento é **B**.

A nona e última linha do programa (**07E9:0111**) finaliza o programa e retorna o controle de execução para o sistema operacional.

Observe a seguir um paralelo entre o código escrito em *opcodes* e linguagem *Assembly* equivalentes do programa anterior que pode ser obtido a partir da instrução **U 0100 0111**.

```
07E9:0100    B3 03       MOV BL,03
07E9:0102    B4 02       MOV AH,02
07E9:0104    B2 41       MOV DL,41
07E9:0106    CD 21       INT 21
07E9:0108    80 EB 01    SUB BL,01
07E9:010B    75 F9       JNZ 0106
07E9:010D    B2 42       MOV DL,42
07E9:010F    CD 21       INT 21
07E9:0111    CD 20       INT 20
```

Em seguida execute o programa com o comando **G** para visualizar na tela do monitor de vídeo a sequência de caracteres **AAAB**.

Além dos comandos de salto **JZ** e **JNZ**, há ainda o comando **CMP** (*compare*), a qual não altera o valor inicial de um determinado registrador geral. Sua finalidade é apenas comparar os valores. Se eles forem iguais, esse comando altera a sinalização do registrador de estado *Zero Flag* para **ZR**.

O comando **CMP** não efetua a comparação de locais de memória ou dois valores informados na mesma instrução. Os conteúdos (operandos) informados para o comando **CMP** devem ser do mesmo tamanho, podendo ser valores do tipo *byte* ou *word*.

O comando **CMP** quando executada altera os valores dos registradores de estados (*flags*) **AF**, **CF**, **OF**, **PF**, **SF** e **ZF**, possuindo como possibilidade de trabalho as operações sobre:

- registrador, registrador (para este tipo de ação usa 2 *bytes* de memória);
- memória, registrador (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, memória (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, constante (para este tipo de ação usa de 3 a 4 *bytes* de memória);
- memória, constante (para este tipo de ação usa de 3 a 6 *bytes* de memória).

Por exemplo, entre o valor hexadecimal **AAAA** para os registradores gerais **AX** e **BX** e ajuste o registrador **IP** para **0100**, como é mostrado a seguir:

```
R AX AAAA <Enter>
R BX AAAA <Enter>
R IP 0100 <Enter>
```

Observe se os dados de seu computador estão semelhantes aos pontos marcados em negrito após execução do comando **R**:

```
AX=AAAA BX=AAAA CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 B303              MOV     BL,03
```

Com o comando **A 0100** entre a linha de código **SUB AX, BX**, verifique se o registrador **IP** aponta para o endereço de deslocamento **0100** e execute o comando **T**. Note o valor alterado do registrador geral **AX** e, em especial, do registrador de estado *Zero Flag*:

```
AX=0000 BX=AAAA CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL ZR NA PE NC
07E9:0102 B402          MOV     AH,02
```

Saia do programa **Enhanced DEBUG** e retorne ao programa novamente. Defina o valor hexadecimal **AAAA** nos registradores **AX** e **BX**, acione o comando **A 0100** entre a linha de código **CMP AX, BX**, verifique se o registrador **IP** aponta para o endereço de deslocamento **0100** e execute o comando **R**.

```
AX=AAAA BX=AAAA CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0101 NV UP EI PL NZ NA PO NC
07E9:0100 39D8              CMP     AX,BX
```

Na sequência execute o comando **T** e observe atentamente todos os pontos indicados a seguir em negrito:

```
AX=AAAA BX=AAAA CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL ZR NA PE NC
07E9:0102 7261          JB      0165
```

Os valores dos registradores **AX** e **BX** estão inalterados, mas a sinalização do registrador de estado *Zero Flag* indica **ZR** quando anteriormente indicava **NZ** e o registrador *Parity Flag* indica **PE** quando anteriormente indicava **PO**. Mais adiante esta característica de comportamento do comando **CMP** será utilizada.

5.7 - Execução básica de desvios

A partir do conjunto de detalhes anteriormente apresentado torna-se possível trabalhar em algo um pouco mais sofisticado, como, por exemplo, escrever na tela do monitor de vídeo um valor de um dígito expresso em notação hexadecimal.

O programa deve apresentar um valor hexadecimal entre **0** (zero) e **F** (quinze em decimal). É preciso levar em consideração o código ASCII em hexadecimal de cada caractere que compõe a formação de um único valor hexadecimal. Considere o trecho apresentado na Tabela 5.4.

Tabela 5.4 - Trecho com caracteres e seus valores hexadecimais

Caractere	Código ASCII (Hexadecimal)
0	30
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39
:	3A
;	3B

Caractere	Código ASCII (Hexadecimal)
<	3C
=	3D
>	3E
?	3F
@	40
A	41
B	42
C	43
D	44
E	45
F	46

A parte numérica de um valor hexadecimal representado na Tabela 4.4 vai do código ASCII **30** até o código ASCII **39**, enquanto a parte alfabética de um valor hexadecimal vai do código ASCII **41** até o código ASCII **46**. Para formar a apresentação de um valor hexadecimal, é necessário usar dois grupos de caracteres com sequências diferentes da tabela ASCII com uma disparidade de sete posições. O programa em questão deve considerar este fato.

Para conseguir a apresentação na tela do monitor de vídeo de um valor hexadecimal, é necessário utilizar o comando de comparação **CMP** (*compare*) em conjunto com os comandos de salto condicional. O comando **CMP** efetua a subtração dos valores existentes entre os registradores gerais **AX** e **BX** sem que o resultado da operação seja armazenado na memória e sem efetuar a alteração do valor do registrador **AX**. O resultado da operação altera apenas os valores dos ponteiros dos registradores de estado, permitindo que se utilize na sequência algum dos comandos de saltos condicionais.

Antes de se preocupar com a rotina de apresentação de um único valor hexadecimal, é pertinente testar um pouco o funcionamento do comando **CMP** em conjunto com um dos vários comandos de saltos condicionais existentes. Neste

caso, será utilizado o comando **JG** (*jump on greater*) que não afeta nenhum dos registradores de estado e ocupa de 2 a 4 bytes de memória.

Para realizar o teste condicional do comando **JG**, defina, respectivamente, para os registradores gerais **AX**, **BX** e **IP** os valores hexadecimais **0003, 0002** e **0100** com as instruções:

```
R AX 0003 <Enter>
R BX 0002 <Enter>
R IP 0100 <Enter>
```

Em seguida, execute a instrução **A 0100** e entre a sequência de código seguinte:

```
07E9:0100 CMP AX,BX <Enter>
07E9:0102 JG 0108   <Enter>
07E9:0104 MOV DL,BB <Enter>
07E9:0106 JMP 010A  <Enter>
07E9:0108 MOV DH,AA <Enter>
07E9:010A SUB DX,DX <Enter>
07E9:010C           <Enter>
```

O programa executa na primeira linha (**07E9:0100**) a comparação dos valores dos registradores gerais **AX** e **BX** com o comando **CMP**, o qual realiza uma subtração entre os valores dos registradores, afetando apenas valores dos registradores de estado: **AF** *(Auxiliar Flag)*, **CF** *(Carry Flag)*, **OF** *(Overflow Flag)*, **PF** *(Paity Flag)*, **SF** *(Signal Flag)* e **ZF** *(Zero Flag).*

Na segunda linha (**07E9:0102**) o programa efetua um desvio condicional, caso o valor do registrador geral **AX** seja maior que o valor do registrador geral **BX,** por meio da instrução condicional **JG**. O fluxo do programa passa para a linha de código **0108**, caso a condição seja verdadeira. Para uma instrução condicional funcionar, ela necessita do uso prévio da instrução **CMP**.

A terceira linha (**07E9:0104**) carrega o valor hexadecimal **BB** para dentro da parte menos significativa (**DL**) do registrador geral **DX**. Essa linha somente será executada caso a linha **0102** não faça o desvio programado.

A quarta linha (**07E9:0106**), por meio da instrução **JMP**, realiza um salto incondicional para a linha de código **010A**. Essa linha somente será executada caso a linha **0102** não efetue o desvio programado. A instrução **JMP** não afeta nenhum registrador de estado, ocupando de 2 até 4 *bytes* de memória. Esta instrução será apresentada em mais detalhes no capítulo 8.

A quinta linha (**07E9:0104**) carrega o valor hexadecimal **AA** para dentro da parte mais significativa (**DH**) do registrador geral **DX**. Essa linha somente será executada caso a condição da linha **0102** efetue o desvio programado.

A sexta e última linha (**07E9:010A**) faz a subtração de todo o conteúdo do registrador geral **DX** pelo próprio conteúdo do registrador geral **DX**, ou seja, desta forma o valor do registrador é zerado.

Observe a seguir um paralelo entre o código escrito em *opcodes* e linguagem *Assembly* equivalentes do programa anterior que pode ser obtido a partir da instrução **U 0100 010A**.

```
07E9:0100    39 D8       CMP AX,BX
07E9:0102    7F 04       JG 0108
07E9:0104    B2 BB       MOV DL,BB
07E9:0106    EB 02       JMP 010A
07E9:0108    B6 AA       MOV DL,AA
07E9:010A    29 D2       SUB DX,DX
```

Ao executar o programa, utilize o comando **T** para fazê-lo passo a passo, mas antes disso verifique se o registrador de deslocamento **IP** aponta para o endereço de deslocamento **0100**. Caso não esteja, execute o comando **R IP** e forneça o valor **0100**.

Na sequência execute o comando **R** e observe a apresentação das informações pertinentes ao estado atual dos valores em memória marcados em negrito.

```
AX=0003 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL ZR NA PE NC
07E9:0100 39D8              CMP     AX,BX
```

Em seguida execute o programa com o comando **T**. A seguir veja o resultado das operações solicitadas:

```
AX=0003 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PO NC
07E9:0102 7F04          JG      0108
```

Nesta etapa pode-se perceber que, ao ser executado o comando **T**, o registrador de deslocamento **IP** encontra--se na posição **0102**, na qual se encontra a instrução **JG 0108**.

Pelo fato de o valor do registrador geral **AX** ser maior que o valor do registrador geral **BX**, ele ocasiona um resultado da operação de subtração positivo, o que caracteriza que o primeiro valor é maior que o segundo valor e desvia o fluxo de execução do programa para a linha **0108**, como pode ser constatado na próxima execução do comando **T**.

```
AX=0003 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0108 NV UP EI PL NZ NA PO NC
07E9:0108 B6AA              MOV     DH,AA
```

Ocorreu o desvio para a linha **0108** na qual se encontra a instrução **MOV DH,AA** que carrega o registrador geral **DX** com o valor **AA00**, como pode ser constatado na próxima execução do comando **T**.

```
AX=0003 BX=0002 CX=0000 DX=AA00 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010A NV UP EI PL NZ NA PO NC
07E9:010A 29D2              SUB     DX,DX
```

Observe os pontos marcados em negrito. O registrador de deslocamento **IP** está apontando para o endereço de deslocamento **010A** e neste momento mostra o registrador geral **DX** com o valor **AA00**. Dê sequência à execução do programa com o comando **T**.

```
AX=0003 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010C NV UP EI PL ZR NA PE NC
07E9:010C 26283F            SUB     ES:[BX],BH                      ES:0002=FF
```

Após a última execução do comando **T** o registrador geral **DX**, devido à execução das linhas **SUB DX,DX,** passa a ter o valor **0000**. Perceba também a posição de deslocamento em que se encontra o registrador de deslocamento **IP**.

Para fazer outro teste, altere o valor do registrador geral **AX** para **0001** e do registrador de deslocamento **IP** para **0100** com as instruções:

```
R AX 0001 <Enter>
R IP 0100 <Enter>
```

Em seguida acione uma vez o comando **R** e mais quatro vezes o comando **T**, prestando muita atenção no que ocorre. A seguir é apresentada a sequência de ocorrência da ação solicitada:

```
-R
AX=0001 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL ZR NA PE NC
07E9:0100 39D8              CMP     AX,BX

-T
AX=0001 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI NG NZ AC PE CY
07E9:0102 7F04              JG      0108
```

```
-T
AX=0001 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0104 NV UP EI NG NZ AC PE CY
07E9:0104 B2BB          MOV     DL,BB

-T
AX=0001 BX=0002 CX=0000 DX=00BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0106 NV UP EI NG NZ AC PE CY
07E9:0106 EB02              JMP     010A

-T
AX=0001 BX=0002 CX=0000 DX=00BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010A NV UP EI NG NZ AC PE CY
07E9:010A 29D2              SUB     DX,DX

-T
AX=0001 BX=0002 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010C NV UP EI PL ZR NA PE NC
07E9:010C 26283F            SUB     ES:[BX],BH                      ES:0002=FF
```

Nesta sequência ocorreu algo de diferente na execução do programa. Pelo fato de o valor do registrador geral **AX** ser menor que o valor do registrador geral **BX** (observe as indicações do registrador de estado **CF** com a indicação **NC** e **CY**), ele gerou um resultado negativo. A instrução da linha **0102** não é considerada, passando o controle do fluxo de execução do programa para a linha seguinte (linha **0104** que atribui o valor **00BB** ao registrador geral **DX**). Depois o programa encontra a linha **0106** que por meio da instrução **JMP 010A** desvia a execução do programa para a linha **010A**, que limpa o valor existente no registrador geral **DX**.

A partir da explicação anteriormente exposta, já é possível escrever um trecho de programa que apresente na tela o valor de um número hexadecimal de uma única posição. Para tanto, ajuste o valor do registrador de deslocamento **IP** para **0100**; zere os valores dos registradores gerais **AX** e **BX** com as instruções:

```
R AX 0000 <Enter>
R BX 0000 <Enter>
R IP 0100 <Enter>
```

A partir da execução da instrução **A 0100** entre as instruções do seguinte programa:

```
07E9:0100 MOV AH,02 <Enter>
07E9:0102 MOV DL,BL <Enter>
07E9:0104 ADD DL,30 <Enter>
07E9:0107 CMP DL,39 <Enter>
07E9:010A JLE 010F  <Enter>
07E9:010C ADD DL,07 <Enter>
07E9:010F INT 21    <Enter>
07E9:0111 INT 20    <Enter>
07E9:0113           <Enter>
```

O programa anterior possui alguns detalhes já familiares. A primeira linha (**07E9:0100**) estabelece para a parte mais significativa (**AH**) do registrador geral **AX** o valor hexadecimal **02**, responsável por apresentar um único caractere na tela do monitor de vídeo.

A segunda linha (**07E9:0102**) movimenta para a parte menos significativa do registrador geral **DX** o conteúdo armazenado na parte menos significativa do registrador geral **BX**.

A terceira linha (**07E9:0104**) adiciona o valor hexadecimal **30** ao valor atual armazenado no registrador geral **BX**. O valor hexadecimal **30** equivale ao código ASCII do caractere **0** (zero). Os valores **30**, **31**, **32**, **33**, **34**, **35**, **36**, **37**, **38** e **39** correspondem ao código ASCII equivalente para a apresentação dos caracteres **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** e **9**.

A quarta linha (**07E9:0107**) faz a comparação (subtração) do valor armazenado na parte menos significativa do registrador geral **DX** com o valor **39**, que é o código ASCII do número **9** (nove).

A quinta linha (**07E9:010A**) utiliza a instrução condicional **JLE** para verificar se o resultado da comparação (linha de código **0107**) do registrador geral **DX** é menor ou igual ao valor **39**. Se a condição for verdadeira, o fluxo de execução do programa desvia a execução para a linha **010F** que apresenta o valor equivalente encontrado no registrador geral **BX**. Caso a condição seja falsa, automaticamente o programa executa a linha de código **010C** (sexta linha). A instrução **JLE** não afeta nenhum registrador de estado, ocupando de 2 até 4 *bytes* de memória.

A sexta linha (**07E9:010C**) será executada quando a quinta linha for falsa. Neste caso o valor armazenado no registrador **BX** corresponde às sete letras que compõem a representação numérica de um valor em formato hexadecimal. A letra **A** possui código ASCII hexadecimal **41**. Ela está distante do valor nove posições. Por esta razão a sexta linha faz a adição do valor hexadecimal **07** com o valor existente na parte menos significativa do registrador geral **DX**.

A sétima (**07E9:010F**) e a oitava (**07E9:0111**) linhas do programa executam, respectivamente, a ativação da tela do monitor de vídeo para apresentação dos caracteres e também devolvem o controle de execução do programa para o sistema operacional.

Observe a seguir um paralelo entre o código escrito em *opcodes* e linguagem *Assembly* equivalentes do programa anterior que pode ser obtido a partir da instrução **U 0100 0111**.

```
07E9:0100    B4 02       MOV AH,02
07E9:0102    88 DA       MOV DL,BL
07E9:0104    80 C2 30    ADD DL,30
07E9:0107    80 FA 39    CMP DL,39
07E9:010A    7E 03       JLE 010F
07E9:010C    80 C2 07    ADD DL,07
07E9:010F    CD 21       INT 21
07E9:0111    CD 20       INT 20
```

Estabeleça para o registrador geral **BX** um valor hexadecimal na faixa de **0000** até **000F**. Para um teste execute o comando **R BX** e informe o valor **0007**, mantenha **IP** em **0100**. Depois execute os comandos **R** e **G** e observe os dados apresentados em tela, como indicado a seguir:

```
-R
AX=0000 BX=0007 CX=0000 DX=0000 SP=FFE8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL ZR NA PE NC
07E9:0100 B402              MOV     AH,02

-G
7
Program terminated (0000)
```

Experimente alterar os valores hexadecimais da parte menos significativa do registrador geral **BX**. Lembre-se de limitar as alterações entre os valores de **00h** até **0Fh**.

5.8 - Apresentar valor hexadecimal

Anteriormente foi conseguido o efeito de apresentar em tela um valor hexadecimal de um único dígito. A questão agora é apresentar um número hexadecimal com mais de um dígito, ou seja, com dois dígitos.

Para apresentar um número hexadecimal com dois dígitos, é necessário utilizar o comando **SHR** (*Shift Logical Right*) para fazer a rotação de *bits* no sentido da direita. Lembre-se de que anteriormente esse trabalho foi feito pelo comando **RCL**, a qual fazia a rotação no sentido da esquerda. É pertinente, nesta etapa, sair do programa **Enhanced DEBUG** em uso e retornar a ele em seguida para que todos os registradores sejam inicializados com seus valores padrão.

O comando **SHR** possibilita a rotação de *bits* no sentido da direita com uma pequena diferença em relação ao comando **RCL**, uma vez que é possível efetuar a movimentação de mais de um *bit* na mesma instrução. É necessário apenas fornecer para o comando **SHR** o número de vezes que a rotação deve ser realizada (considerando o fato de ser um tipo *byte* ou um tipo *word*). O valor da rotação fica definido na parte menos significativa (**CL**) do registrador geral **CX**, estando limitado ao valor máximo de dezesseis (**10** em hexadecimal) que é o tamanho de um tipo *word*.

O comando **SHR** quando executada altera os valores dos registradores de estados (*flags*) **CF**, **OF**, **PF**, **SF** e **ZF**, possuindo como possibilidade de trabalho as operações sobre:

- registrador, 1 (para este tipo de ação usa 2 *bytes* de memória);
- memória, 1 (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, CL (para este tipo de ação usa 2 *bytes* de memória);
- memória, CL (para este tipo de ação usa de 2 a 4 *bytes* de memória).

Para executar a apresentação em tela de valor numérico hexadecimal de dois dígitos, é necessário fazer a rotação de **4** *bits* no sentido da direita, para que o valor hexadecimal armazenado na parte menos significativa (**DL**) do registrador geral **DX** seja manipulado adequadamente. Para tanto, execute as instruções:

```
R DX 006B    <Enter>
R CX 0004    <Enter>
E 0100 D2 EA <Enter>
```

O *opcode* **D2 EA** é código de operação para a instrução **SHR DL,CL** e execute o comando **R** que apresenta valores semelhantes à relação seguinte:

```
AX=0000 BX=0000 CX=0004 DX=006B SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PO NC
07E9:0100 D2EA              SHR     DL,CL
```

Antes de executar qualquer ação, preste bem atenção nos pontos marcados em negrito, principalmente nos valores armazenados nos registradores gerais **CX** (contador de deslocamento de *bits* que será de quatro em quatro, ou seja, de *nibble* em *nibble*) e **DX** (valor que será rotacionado à direita). Observe atentamente o registrador de estado **CF** que do valor **NC** passará para o valor **CY**.

Em seguida execute o comando **T**.

```
AX=0000 BX=0000 CX=0004 DX=0006 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ AC PE CY
07E9:0102 F726FC26          MUL     WORD PTR [26FC]               DS:26FC=0000
```

O valor do registrador geral **DX** passou a ser **0006**, ou seja, ele foi movimentado quatro *bits* (um *nibble*) para a direita. Além disso, o registrador de estado **CF** passou do valor **NC** para o valor **CY**, indicando a movimentação de "*carry*", neste caso para a direita.

Para entender melhor essa ação execute a instrução **A 0100** e entre o código:

```
07E9:0100 MOV AH,02 <Enter>
07E9:0102 MOV DL,BL <Enter>
07E9:0104 MOV CL,04 <Enter>
07E9:0106 SHR DL,CL <Enter>
07E9:0108 ADD DL,30 <Enter>
07E9:010B CMP DL,39 <Enter>
07E9:010E JLE 0113  <Enter>
07E9:0110 ADD DL,07 <Enter>
07E9:0113 INT 21    <Enter>
07E9:0115 INT 20    <Enter>
07E9:0117           <Enter>
```

Em seguida execute as instruções:

```
R BX 0012 <Enter>
R IP 0100 <Enter>
```

Na sequência execute o comando **R**.

```
AX=0000 BX=0012 CX=0004 DX=0006 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0100 NV UP EI PL NZ NA PE CY
07E9:0100 B402              MOV     AH,02
```

Para fazer um teste passo a passo da execução do programa, utilize o comando **T** e não o comando **G**. Na primeira execução do comando **T** o valor hexadecimal **02** é armazenado no segmento mais significativo do registrador geral **AX** e o registrador de deslocamento **IP** aponta para o endereço **0102**, conforme a seguir:

```
AX=0200 BX=0012 CX=0004 DX=0006 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0102 NV UP EI PL NZ NA PE CY
07E9:0102 88DA              MOV     DL,BL
```

Execute novamente o comando **T** e observe a movimentação do valor existente no registrador menos significativo **BL** para o registrador menos significativo **DL**. Observe também o valor do registrador de deslocamento **IP** apontando para o endereço **0104**.

```
AX=0200 BX=0012 CX=0000 DX=0012 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0104 NV UP EI PL NZ NA PE CY
07E9:0104 B104              MOV     CL,04
```

Na sequência execute o comando **T** novamente e observe a movimentação do valor **04** para dentro do registrador menos significativo **CL**. Mais uma vez observe o valor do registrador de deslocamento **IP** apontando para o endereço **0106**.

```
AX=0200 BX=0012 CX=0004 DX=0012 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0106 NV UP EI PL NZ NA PE CY
07E9:0106 D2EA              SHR     DL,CL
```

Novamente acione o comando **T** e observe que no registrador geral **DL**, ocorre um deslocamento à direita de quatro *bits*. Note a alteração do valor do registrador de deslocamento **IP** apontando para o endereço **0108**.

```
AX=0200 BX=0012 CX=0004 DX=0001 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0108 NV UP EI PL NZ NA PE CY
07E9:0108 80C230            ADD     DL,30
```

A partir da linha **0108** o programa é idêntico ao modelo do tópico anterior. Neste caso acione desse ponto o comando **G** para a conclusão e término adequado do programa, pois o valor hexadecimal **0001** do registrador geral **DX** será apresentado como **1**:

```
1
Program terminated (0000)
```

Observe a seguir um paralelo entre o código escrito em *opcodes* e linguagem *Assembly* equivalentes do programa anterior obtido a partir da instrução **U 0100 0115**.

```
07E9:0100    B4 02       MOV AH,02
07E9:0102    88 DA       MOV DL,BL
07E9:0104    B1 04       MOV CL,04
07E9:0106    D2 EA       SHR DL,CL
07E9:0108    80 C2 30    ADD DL,30
07E9:010B    80 FA 39    CMP DL,39
07E9:010E    7E 03       JLE 0113
07E9:0110    80 C2 07    ADD DL,07
07E9:0113    CD 21       INT 21
07E9:0115    CD 20       INT 20
```

Até este ponto foi conseguido o mesmo efeito do tópico anterior, mas não é este o objetivo. É preciso ainda o suporte de mais uma instrução para conseguir a impressão do segundo dígito do valor. Entrará em uso o comando de ação lógica **AND** (*and logical*) que faz a conjunção de dois valores e os considera com resultado verdadeiro se todos os valores forem também verdadeiros.

O comando **AND** quando executado altera o estado dos registradores de estados (*flags*) **CF**, **OF**, **PF**, **SF** e **ZF**, possuindo como possibilidade de trabalho as operações sobre:

- registrador, registrador (para este tipo de ação usa 2 *bytes* de memória);
- memória, registrador (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, memória (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- registrador, constante (para este tipo de ação usa de 3 a 4 *bytes* de memória);
- memória, constante (para este tipo de ação usa de 3 a 6 *bytes* de memória).

Observação

O comando **T** (*trace*) utilizado até aqui foi executado sempre passo a passo. No entanto, é possível definir para ele o número de passos que deseja que ele execute. Por exemplo, experimente no *prompt* do programa **Enhanced DEBUG** a seguinte linha de comando **T = 0100 4** e observe que será apresentado o resultado de quatro passagens do comando. O comando **T** pode ocasionar execuções desconfortáveis quando direciona sua ação para dentro de uma rotina de interrupção (como é o caso do comando **INT 21**, que é na verdade uma sub-rotina adjacente ao código em uso). No entanto, o procedimento pode ser executado com o comando **P** (*proceed*) que não entra nas rotinas adjacentes, executando-as como se fossem comandos comuns. O comando **P** aceita os mesmos parâmetros que o comando **T**. Após o uso dos comandos **T** e **P** é sempre importante retornar o valor do registrador de deslocamento **IP** para **0100**.

Para o modo de programação em baixo nível, um valor é considerado verdadeiro quando está "setado" em **1** (um) e um valor é considerado falso quando está "setado" em **0** (zero). Isso é semelhante ao utilizado pelos registradores de estado. Considere a Tabela 5.5 para dois valores quaisquer avaliados em um estado de conjunção:

Tabela 5.5 - Trecho para ação de conjunção

Valor 1	Valor 2	Resultado
1	1	1
1	0	0
0	1	0
0	0	0

Na tabela o resultado será considerado verdadeiro (valor **1**) quando todos os valores (no caso valores **1** e **2**) forem também verdadeiros, ou seja, definidos com o valor **1** (um).

O comando **AND** faz a comparação em um determinado valor *bit* a *bit* (modo binário) em dados do tipo *byte* ou *word,* tendo seu uso destinado a três situações típicas (DANDAMUDI, 2000, p. 301):

1. Apoiar as composições de expressões lógicas;

2. Limpar um ou mais *bits* de um *byte* ou *word;*

3. Isolar um ou mais *bits* de um *byte* ou *word*.

Para os objetivos propostos o comando **AND** será usada para fazer a limpeza (zerar) dos *bits* do *byte* em uso que não forem de interesse. Por exemplo, imagine que se possua o valor hexadecimal **12**, o qual equivale em binário ao valor **0001 0010,** e que se deseja pegar apenas a parte da direita do valor, ou seja, o número **2** desprezando-se assim o valor **1**. Assim sendo, basta comparar o valor binário **0001 0010** com o valor binário **0000 1111** (que equivale ao valor hexadecimal **0F**), como mostra a Tabela 5.6.

Tabela 5.6 - Ação de limpeza de *bits*

Byte	***Nibble* mais alto**				***Nibble* mais baixo**			
Bits	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Valores comparados com instrução lógica AND								
Valor original	0	0	0	1	0	0	1	0
Código para limpeza	0	0	0	0	1	1	1	1
Resultado	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

Observe na tabela a indicação dos valores hexadecimais **12** (equivalente a **0001 0010** em binário) e **0F** (equivalente a **0000 1111** binário). Ao ser realizada uma comparação *bit* a *bit* com o comando **AND** de cada parte de ambos os valores e respeitando a regra definida na tabela-verdade anterior, fica evidente que no *nibble* mais alto sua posição torna-se zerada e no *nibble* mais baixo o valor fica mantido.

A partir deste fato é possível fazer a outra parte do trabalho. Já existe o conhecimento para apresentar o primeiro valor de um número hexadecimal de um *byte*. No entanto, será demonstrado um programa que apresenta apenas o segundo valor de um número hexadecimal para depois ambas as técnicas serem trabalhadas em conjunto.

Observação

Uma forma rápida de fazer verificações condicionais com valores dos tipos binários e hexadecimais é utilizar a calculadora existente no ambiente Windows. Neste caso, selecione o modo de operação científica, o modo de operação Bin (binário) ou Hex (hexadecimal). Depois entre o primeiro valor e acione o botão [And], entre o segundo valor e acione o botão [=]. Experimente fazer o teste com o valor hexadecimal 12 no sentido de obter a parte menos significativa de seu valor, ou seja, o valor 2. Execute na calculadora 12 [And] F [=] e observe a apresentação do valor 2.

Em seguida execute as instruções:

```
R BX 0012 <Enter>
R IP 0100 <Enter>
```

Na sequência entre o código a seguir a partir da execução da instrução **A 0100**.

```
07E9:0100 MOV AH,02 <Enter>
07E9:0102 MOV DL,BL <Enter>
07E9:0104 AND DL,0F <Enter>
07E9:0107 ADD DL,30 <Enter>
07E9:010A CMP DL,39 <Enter>
07E9:010D JLE 0112  <Enter>
07E9:010F ADD DL,07 <Enter>
07E9:0112 INT 21    <Enter>
07E9:0114 INT 20    <Enter>
07E9:0116           <Enter>
```

Após entrar o código anterior, execute o comando **G** e observe a apresentação do valor hexadecimal **2**. Se desejar fazer uma execução passo a passo, utilize o comando **T,** tomando o cuidado de parar a ação do comando na linha **0112**, para não cair na rotina de programa que realiza o comando **INT 21**.

Observe a seguir um paralelo entre o código escrito em *opcodes* e linguagem *Assembly* equivalentes do programa anterior

```
07E9:0100    B4 02       MOV AH,02
07E9:0102    88 DA       MOV DL,BL
07E9:0104    80 E2 0F    AND DL,0F
07E9:0107    80 C2 30    ADD DL,30
07E9:010A    80 FA 39    CMP DL,39
07E9:010D    7E 03       JLE 0112
07E9:010F    80 C2 07    ADD DL,07
07E9:0112    CD 21       INT 21
07E9:0114    CD 20       INT 20
```

Para apresentar dois valores numéricos basicamente basta unir os dois trechos de programa conhecidos e apresentados anteriormente em um código só. O trecho de programa que utiliza o comando **SHR** deve vir à frente do trecho de programa que usa o comando **AND**.

Trechos anteriores	Código completo
07E9:0100 MOV AH,02 07E9:0102 MOV DL,BL 07E9:0104 MOV CL,04 07E9:0106 SHR DL,CL 07E9:0108 ADD DL,30 07E9:010B CMP DL,39 07E9:010E JLE 0113 07E9:0110 ADD DL,07 07E9:0113 INT 21 07E9:0115 INT 20	07E9:0100 MOV AH,02 07E9:0102 MOV DL,BL 07E9:0104 MOV CL,04 07E9:0106 SHR DL,CL 07E9:0108 ADD DL,30 07E9:010B CMP DL,39 07E9:010E JLE 0113 07E9:0110 ADD DL,07 07E9:0113 INT 21
07E9:0100 MOV AH,02 07E9:0102 MOV DL,BL 07E9:0104 AND DL,0F 07E9:0107 ADD DL,30 07E9:010A CMP DL,39 07E9:010D JLE 0112 07E9:010F ADD DL,07 07E9:0112 INT 21 07E9:0114 INT 20	07E9:0115 MOV DL,BL 07E9:0117 AND DL,0F 07E9:011A ADD DL,30 07E9:011D CMP DL,39 07E9:0120 JLE 0125 07E9:0122 ADD DL,07 07E9:0125 INT 21 07E9:0127 INT 20

Informe o código de programa indicado na coluna da esquerda sem pular linhas em branco a partir do comando **A 0100**: Observe a seguir as instruções a serem definidas:

```
07E9:0100 MOV AH,02 <Enter>
07E9:0102 MOV DL,BL <Enter>
07E9:0104 MOV CL,04 <Enter>
07E9:0106 SHR DL,CL <Enter>
07E9:0108 ADD DL,30 <Enter>
07E9:010B CMP DL,39 <Enter>
07E9:010E JLE 0113  <Enter>
07E9:0110 ADD DL,07 <Enter>
07E9:0113 INT 21    <Enter>
07E9:0115 MOV DL,BL <Enter>
07E9:0117 AND DL,0F <Enter>
07E9:011A ADD DL,30 <Enter>
07E9:011D CMP DL,39 <Enter>
07E9:0120 JLE 0125  <Enter>
07E9:0122 ADD DL,07 <Enter>
07E9:0125 INT 21    <Enter>
07E9:0127 INT 20    <Enter>
07E9:0129           <Enter>
```

Ao rodar o programa com o comando **G**, você obterá na tela o valor definido no registrador menos significativo (**BL**) do registrador geral **BX**. Altere alguns valores para a parte menos significativa do registrador geral **BX**. Se você desejar, execute o programa passo a passo com o comando **P** até a apresentação da mensagem: **Program terminated**.