```
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 6

ENTRADA DE DADOS

*Este capítulo mostra como capturar dados fornecidos via teclado, utilizar rotina do tipo procedimento, usar pilha, a fazer consistência de entrada de dados, além de ler valores numéricos hexadecimais de um e dois dígitos por meio do programa **Enhanced DEBUG**.*

6.1 - Entrada de dados por teclado

Anteriormente os programas desenvolvidos efetuaram operações de processamento e apresentação de sequencias de caracteres (mensagem) e dígitos (valores numéricos em hexadecimais) por intermédio do uso da interrupção **21** que na ocasião utilizou as funções hexadecimais de controle de saída: **02** para a saída de um caractere e **09** para a saída de mais de um caractere, ou seja, saída de um *string*. A entrada de dados também utiliza a interrupção **21**, mas com os códigos de função de controle de entrada **01** para a parte alta do registrador geral **AX**. O programa que fará a leitura de um caractere (que deve ser um dígito hexadecimal válido) é semelhante ao programa de apresentação de dados desenvolvido no capítulo anterior.

Para a realização da entrada de dados numéricos ou alfanuméricos, bem como para a saída desses dados o computador por meio de seu microprocessador faz uso da tabela ASCII (ver apêndice A) para efetivar esta ação. Assim sendo, quando se efetua a entrada de certa sequência de dados por meio do teclado o dado inserido é decomposto nos caracteres que o formam e são então convertidos internamente para seus códigos ASCII em hexadecimal. Por exemplo, a entrada da sequência de caracteres “Brasil3000” será decomposta nos caracteres “B”, “r”, “a”, “s”, “i”, “l”, “3”, “0”, “0” e “0”, ou seja, será decomposta em seus códigos hexadecimais ASCII “42h”, “72h”, “61h”, “73h”, “69h”, “6Ch”, “33h”, “30h”, “30h” e “30h”. Após a inserção de cada um dos caracteres via teclado estes são transferidos para a memória e são nela armazenados no formato hexadecimal. Em seguida os dados são lidos da memória em seu formato hexadecimal e são transferidos para o monitor de vídeo, onde são apresentados visualmente para o usuário convertidos no formato dos caracteres efetivamente fornecidos junto ao teclado. A Figura 5.1 demonstra o esquema de ação de entrada de dados.

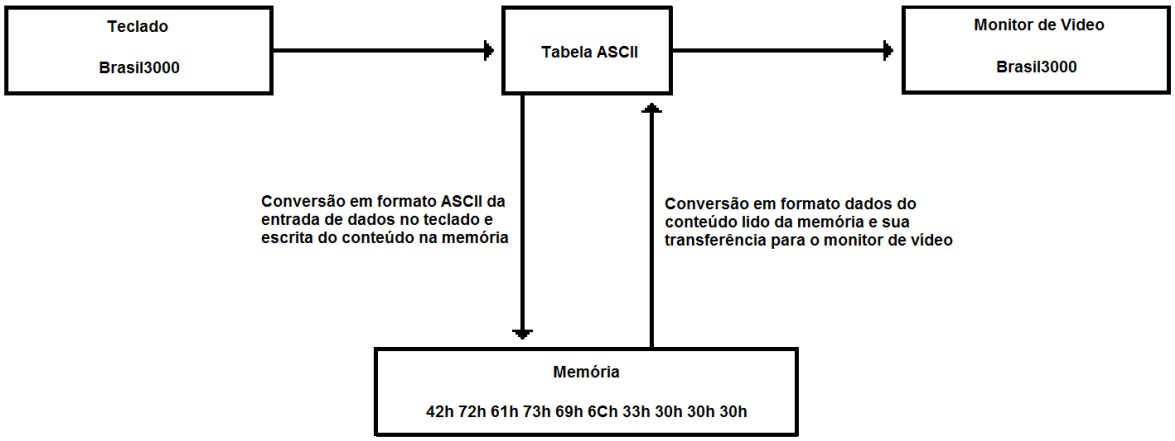

Figura 5.1 - Esquema de ação da entrada de dados.

Antes de iniciar qualquer novo teste apresentado neste capítulo sugere-se sair do programa **Enhanced DEBUG** e inicia-lo novamente para que todos os registradores sejam devidamente inicializados em seus valores padrão.

Em seguida execute a instrução **A 0100** e informe o código seguinte de programa. Atente para o código expresso em *Assembly* no lado esquerdo. O código expresso no lado direito está sendo indicado em *opcodes*:

```
07E9:0100 MOV AH,01    | B4 01
07E9:0102 INT 21       | CD 21
07E9:0104 SUB AL,30    | 2C 30
07E9:0106 CMP AL,09    | 3C 09
07E9:0108 JLE 010C     | 7E 02
07E9:010A SUB AL,07    | 2C 07
07E9:010C INT 20       | CD 20
```

A primeira linha do programa (**07E9:0100**) carrega para o registrador mais significativo (**AH**) do registrador geral **AX** o valor hexadecimal **01** que é o código da função de entrada de dados de um teclado.

A segunda linha (**07E9:0102**) executa a interrupção **21**. Neste momento o programa permite que um valor hexadecimal seja informado e automaticamente armazenado na parte menos significativa (**AL**) do registrador geral **AX**. Ao fazer a entrada de um valor hexadecimal entre **0** e **F,** ele será capturado de forma direta, ou seja, não é necessário acionar a tecla **<Enter>**.

A terceira linha (**07E9:0104**) faz a subtração do valor hexadecimal **30**. Por exemplo, ao entrar o valor hexadecimal **05** pelo teclado, ele é armazenado no registrador **AL** com o valor hexadecimal **35** que é o código hexadecimal ASCII referente ao caractere "**5**", ou seja, ao acionar a tecla **<5>** no teclado, o seu código ASCII é que, na verdade, é armazenado. É preciso tirar o valor hexadecimal **30** para obter o valor correto da tecla acionada.

A quarta linha (**07E9:0106**), por meio da instrução **CMP**, faz a subtração do valor hexadecimal **09** em relação ao conteúdo existente no registrador **AL** e atualiza os *flags* de controle, fornecendo o parâmetro base para a execução da quinta linha de código. O valor **09** no registrador **AL** refere-se ao limite máximo dos dígitos hexadecimais aceitos, que vão de **0** a **9**. Os demais valores são representados pelas letras de **A** até **F**.

A quinta linha (**07E9:0108**), por meio da instrução **JLE,** desvia o código de execução do programa para a linha **010C** caso o valor de **CMP** (quarta linha) seja menor ou igual a **09**. Valores hexadecimais de **0** até **9** serão desviados para a linha de código **010C**. Os valores de **A** até **F** não serão desviados por essa linha, transferindo a execução do programa para a próxima linha (**010A**).

A sexta linha (**07E9:010A**) subtrai do registrador **AL** o valor hexadecimal **07**, além do valor **30** já subtraído pela linha de código **0104**. Isso é necessário, uma vez que na tabela ASCII os valores numéricos de **0** até **9** estão a uma distância de sete posições em relação aos caracteres de **A** até **F**, como pode ser constatado no trecho indicado junto à Tabela 6.1.

Tabela 6.1 - Valores numéricos versus valores caracteres

Caractere	Código ASCII (Hexadecimal)	Caractere	Código ASCII (Hexadecimal)
0	30	8	38
1	31	9	39
2	32	A	41
3	33	B	42
4	34	C	43
5	35	D	44
6	36	E	45
7	37	F	46

Por último a sétima linha (**07E9:010C**) executa a interrupção **20** a qual devolve o controle ao sistema operacional.

Verifique se o ponteiro de deslocamento **IP** está apontando para o endereço de deslocamento **0100**. Antes de iniciar a execução do programa, acione o comando **R** e observe se a informação de tela é semelhante à indicada a seguir:

```
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 B401              MOV     AH,01
```

Para executar passo a passo pode-se Fazer uso do comando **T**, mas agora será utilizado outro comando para a mesma ação. Neste caso, o comando **P** (*Proceeed*). Assim sendo execute a instrução:

```
P <Enter>
```

Será apresentada a seguinte informação, como mostra os trechos em negrito:

```
AX=0100 BX=0000 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 CD21              INT     21
```

O registrador **AX** passa a ter o valor **0100**, sendo o valor **01** o código de função de entrada de dados. Na sequência acione novamente o comando **P** com a instrução:

```
P <Enter>
```

Observe que aparentemente nada acontece. Neste momento forneça um valor hexadecimal válido. Para um teste forneça o valor **5**, com a instrução:

```
5 <Enter>
```

Observe na sequência a apresentação dos valores nos registradores:

```
AX=0135 BX=0000 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 PL NZ NA PO NC
07E9:0104 2C30              SUB     AL,30
```

O registrador **AX** está com o valor **0135**, sendo o valor **01** o código da função de entrada de dados e **35** o valor ASCII referente ao caractere **5**. Está sendo indicada a execução da próxima instrução (**SUB AL,30**) que tirará do registrador **AL** que é **35** o valor **30**. Na sequência execute a instrução:

```
P <Enter>
```

Observe os pontos em negrito:

```
AX=0105 BX=0000 CX=0000 DX=0000 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 NC
07E9:0106 3C09              CMP     AL,09
```

O registrador geral **AX** está atualizado e a próxima instrução efetua a comparação (subtração) do valor **09** com o valor armazenado no registrador **AL**. Acione mais uma vez o comando **P**:

```
P <Enter>
```

Observe os pontos em negrito:

```
AX=0105 BX=0000 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 NG NZ AC PE CY
07E9:0108 7E02              JLE     010C
```

Uma forma de saber a ação do comando **CMP** é observar as informações da sequência de *flags* do registrador de estado. Ao fazer a subtração (**CMP AL,09**), o valor **09** é subtraído do valor **05** que gera um resultado no registrador de estado de um valor negativo, e é esse valor que será usado como parâmetro de desvio para o próximo comando, como indica a próxima linha de código: **07E9:0108 7E02 JLE 010C**. Na sequência acione o comando **P** novamente:

```
P <Enter>
```

Observe em negrito a ocorrência do desvio para a linha **010C** saltando a execução da linha **010A**:

```
AX=0105 BX=0000 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 NG NZ AC PE CY
07E9:010C CD20              INT     20
```

A partir deste ponto o programa já está quase encerrado. Ao acionar mais uma vez o comando **P**, ele apresenta a mensagem "**Program terminated (0000)**"por ter atingido a linha de código **INT 20**.

Procure fazer alguns testes com o comando **P** para alguns valores hexadecimais de um dígito apenas. Lembre--se de ir ajustando o endereço de deslocamento **0100** para o registrador de deslocamento **IP** com a instrução **R IP 0100**, antes de iniciar um novo teste.

A partir deste ponto já é possível, com certo grau de facilidade, desenvolver uma rotina de programa que leia um valor hexadecimal de dois dígitos. Essa rotina é similar à rotina de saída do capítulo anterior, mas exige um controle operacional um pouco maior.

Para auxiliar essa nova tarefa, será utilizado o comando **SHL** (*Shift Logical Left*), que possui sua ação contrária ao modo como o comando **SHR** opera. O comando **SHL** efetua a rotação de *bits* no sentido da esquerda, operando em estruturas de dados do tipo *byte* e *word* e quando executada altera o estado 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).

Basicamente se fará com a rotina de programa de entrada de dois dígitos o sentido inverso realizado pela rotina de saída de dois dígitos. A partir do endereço de deslocamento **0100** com o comando **A** informe o código seguinte definido do lado esquerdo, pois ao lado direito estão indicados os *opcodes* dos comandos apresentados do lado esquerdo:

```
07E9:0100 MOV AH,01     | B4 01
07E9:0102 INT 21        | CD 21
07E9:0104 MOV DL,AL     | 8A D0
07E9:0106 SUB DL,30     | 80 EA 30
07E9:0109 CMP DL,09     | 80 FA 09
07E9:010C JLE 0111      | 7E 03
07E9:010E SUB DL,07     | 80 EA 07
07E9:0111 MOV CL,04     | B1 04
07E9:0113 SHL DL,CL     | D2 E2
07E9:0115 INT 21        | CD 21
07E9:0117 SUB AL,30     | 2C 30
07E9:0119 CMP AL,09     | 3C 09
07E9:011B JLE 011F      | 7E 02
07E9:011D SUB AL,07     | 2C 07
07E9:011F ADD DL,AL     | 02 D0
07E9:0121 INT 20        | CD 20
```

Antes de escrever a rotina de programa anterior, informe o valor **0000** para o registrador **AX** (**R AX 0000**). Estabeleça o endereço de deslocamento **0100** para o registrador de deslocamento **IP**, em seguida execute o comando **R** (**R IP 0100**) e verifique se as posições de memória em uso são equivalentes aos dados seguintes a partir da 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 B401              MOV     AH,01
```

Na sequência acione o comando **P** para que a primeira linha do programa (**07E9:0100**) seja executada, como é apresentado em seguida. Note também o apontamento para a execução da segunda linha de código do programa **07E9:0102**:

```
AX=0100 BX=0000 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 CD21              INT     21
```

O código de função de entrada de dados **01** é estabelecido para a parte mais significativa (**AH**) do registrador geral **AX**. Acione novamente o comando **P** e entre o primeiro dígito de um valor de duas posições.

Para este teste será fornecido o valor hexadecimal **C9**. Entre o caractere **C** (em formato maiúsculo) e automaticamente ocorre um salto da execução do programa da linha **0102** para a linha **0104**, mostrando a seguinte posição de memória:

```
CAX=0143 BX=0000 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 PL NZ NA PO NC
07E9:0104 88C2              MOV     DL,AL
```

O valor **43** (valor hexadecimal correspondente à letra **C** na tabela ASCII) é armazenado na parte menos significativa (**AL**) do registrador geral **AX**. Acione o comando **P**.

```
AX=0143 BX=0000 CX=0000 DX=0043 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0106 NV UP EI PL NZ NA PO NC
07E9:0106 80EA30            SUB     DL,30
```

O valor **43** foi copiado (**MOV DL,AL**) para dentro da parte menos significativa (**DL**) do registrador geral **DX**. Em seguida acione novamente o comando **P** e observe a alteração do valor dentro do registrador geral **DX**.

```
AX=0143 BX=0000 CX=0000 DX=0013 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0109 NV UP EI PL NZ NA PO NC
07E9:0109 80FA09            CMP     DL,09
```

Agora que a parte menos significativa do registrador geral **DX** está com o valor **13**, ele será comparado (**CMP DL,09**), que resultará na subtração do valor **09** do valor **13** do registrador geral **DX**, o qual será avaliado pelo desvio condicional da linha de código **07E9:0109**. Acione o comando **P** e observe a alteração das informações nos registradores de estado:

```
AX=0143 BX=0000 CX=0000 DX=0013 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010C NV UP EI PL NZ AC PE NC
07E9:010C 7E03              JLE     0111
```

A linha de código **07E9:010C** em que se encontra a instrução **JLE 0111** desvia o programa para a linha de código **07E9:0111** caso o resultado da instrução **CMP DL,09** executada na linha de código **07E9:0109** seja menor ou igual a zero. Para verificar essa ação, utilize o comando **P** e observe os detalhes marcados em negrito a seguir:

```
AX=0143 BX=0000 CX=0000 DX=0013 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010E NV UP EI PL NZ AC PE NC
07E9:010E 80EA07            SUB     DL,07
```

A condição de execução da linha de programa **07E9:010C** é considerada falsa. A execução do programa para a linha de código **07E9:010E** fará a subtração do valor **07** do valor existente na parte menos significativa do registrador geral **DX**. Para observar esta ação, acione o comando **P**.

```
AX=0143 BX=0000 CX=0000 DX=000C SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0111 NV UP EI PL NZ AC PE NC
07E9:0111 B104              MOV     CL,04
```

Observe o armazenamento do valor hexadecimal **C** no registrador geral **DL**, sendo este a representação do valor hexadecimal do caractere **C** informado na ação da linha de programa **07E9:0102**.

A partir deste ponto o programa faz o tratamento da entrada do segundo dígito do valor hexadecimal **C9**. Acione o comando **P** e observe o armazenamento do valor **04** na parte menos significativa (**CL**) do registrador geral **CX**.

```
AX=0143 BX=0000 CX=0004 DX=000C SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0113 NV UP EI PL NZ AC PE NC
07E9:0113 D2E2              SHL     DL,CL
```

Lembre-se de que se utiliza o registrador geral **CX** normalmente quando há necessidade de fazer a contagem de algum passo de ação. Neste caso objetiva-se mover em quatro *bits* (uma posição - um *nibble*) o conteúdo do registrador **DX**, ou seja, mover o valor **C** da parte menos significativa uma posição para a esquerda com o comando **SHL** que se encontra na linha de código **07E9:0113**. Acione o comando **P**:

```
AX=0143 BX=0000 CX=0004 DX=00C0 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0115 NV UP EI NG NZ NA PE NC
07E9:0115 CD21              INT     21
```

Observe a posição em que se encontra o valor **C** no registrador geral **DX**. A primeira posição mais à direita do registrador ficou disponível para o armazenamento do próximo valor, que será entrado pela instrução **INT 21** da linha de código **07E9:0115**. Para visualizar essa ação, acione o comando **P** e entre o valor **9** e observe as partes sinalizadas em negrito que se sucedem a ação:

```
9AX=0139 BX=0000 CX=0004 DX=00C0 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0117 NV UP EI NG NZ NA PE NC
07E9:0115 2C30              SUB     AL,30
```

Observe que a parte menos significativa do registrador geral **AX** (que antes da ação possuía o valor **0143** e passa a ter o valor **0139**, mas o que interessa é o valor da parte menos significativa, ou seja, o valor hexadecimal **39**). O valor hexadecimal **39** armazenado na parte menos significativa do registrador geral **AX** corresponde ao código ASCII para a representação do valor numérico **9**.

Em seguida acione o comando **P** para visualizar a ação do comando **SUB AL,30** que fará a subtração do valor hexadecimal **30** do valor atual do registrador **AL**. Observe o ponto marcado em negrito:

```
AX=0109 BX=0000 CX=0004 DX=00C0 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0119 NV UP EI PL NZ NA PE NC
07E9:0119 3C09              CMP     AL,09
```

O registrador geral **AX** tem na sua parte menos significativa o valor hexadecimal **09**. Em seguida, a linha de código **07E9:0119** calcula a subtração interna para definição da comparação, que resulta em um valor **00** e faz com que a instrução da linha de código **07E9:011B** execute o desvio condicional indicado. Acione o comando **P** e observe atentamente os valores do registrador de estado:

```
AX=0109 BX=0000 CX=0004 DX=00C0 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=011B NV UP EI PL ZR NA PE NC
07E9:011B 7E02              JLE     011F
```

Nessa etapa (linha de código **07E9:011B**) o comando **JLE 011F** desvia o programa para a linha de código **07E9:011F**, a qual executa a instrução **ADD DL,AL**. Acione o comando **P** e observe atentamente este detalhe:

```
AX=0109 BX=0000 CX=0004 DX=00C0 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=011F NV UP EI PL ZR NA PE NC
07E9:011F 00C2              ADD     DL,AL
```

Ao ser executado o comando **P** nessa etapa, ocorre a movimentação do valor **09** existente no registrador **AL** para dentro do registrador **DL**. Perceba que após essa ação o registrador geral **DX** ficará com o valor **00C9** referente à entrada **C9**:

```
AX=0109 BX=0000 CX=0004 DX=00C9 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0121 NV UP EI NG NZ NA PE NC
07E9:0121 CD20              INT     20
```

O próximo acionamento do comando **P** finaliza a execução do programa em uso.

6.2 - Utilização de procedimento

O *procedimento* utilizado em linguagem de programação de computadores *Assembly* tem um significado similar às sub-rotinas existentes em linguagens de programação de computadores de alto nível, ou seja, um procedimento é um trecho de código de programa definido em uma determinada área de memória que pode ser chamado e utilizado várias vezes.

Para utilizar o procedimento em linguagem de programação *Assembly,* é necessário trabalhar com dois comandos adicionais: **CALL** (*Call procedure*) que ocupa de 2 até 5 *bytes* de memória e **RET** (*Return*) que ocupa de 1 a 3 *bytes* de memória. Os comandos **CALL** e **RET** não afetam nenhum registrador de estado.

Imagine um programa que apresente em vídeo a sequência de números **0123456789**. Saia do programa **Enhanced DEBUG** e faça um novo carregamento em memória, depois informe as linhas de código a seguir posicionadas do lado esquerdo, lembrando que ao lado direito estão os códigos em *opcodes*:

```
E 0200 "0123456789" 24      | 30 31 32 33 34 35 36 37 38 39 24

A 0100                      |
07E9:0100 MOV AH,09         | B4 09
07E9:0102 MOV DX,0200       | BA 00 02
07E9:0105 INT 21            | CD 21
07E9:0107 INT 20            | CD 20
```

Ao final peça a execução do comando **G** para comprovar o resultado da saída pretendida. No entanto, o objetivo é apresentar caractere por caractere até concluir a apresentação dos dados **0123456789**.

Primeiramente será desenvolvido um programa que não faz uso da técnica de procedimento, em seguida será apresentado uma versão do mesmo programa com uso da técnica de procedimento.

A partir da instrução **A 0100** informe o código seguinte definido do lado esquerdo, lembrando que do lado direito estão indicados os códigos *opcodes* dos comandos apresentados do lado esquerdo:

```
07E9:0100 MOV AH,02     | B4 02
07E9:0102 MOV DL,30     | B2 30
07E9:0104 INT 21        | CD 21
07E9:0106 MOV DL,31     | B2 31
07E9:0108 INT 21        | CD 21
07E9:010A MOV DL,32     | B2 32
07E9:010C INT 21        | CD 21
07E9:010E MOV DL,33     | B2 33
07E9:0110 INT 21        | CD 21
```

```
07E9:0112 MOV DL,34     | B2 34
07E9:0114 INT 21        | CD 21
07E9:0116 MOV DL,35     | B2 35
07E9:0118 INT 21        | CD 21
07E9:011A MOV DL,36     | B2 36
07E9:011C INT 21        | CD 21
07E9:011E MOV DL,37     | B2 37
07E9:0120 INT 21        | CD 21
07E9:0122 MOV DL,38     | B2 38
07E9:0124 INT 21        | CD 21
07E9:0126 MOV DL,39     | B2 39
07E9:0128 INT 21        | CD 21
07E9:012A INT 20        | CD 20
```

Para cada valor apresentado é feita uma chamada à interrupção **21h** (**INT 21**) para que o caractere correspondente ao código hexadecimal seja apresentado na tela do monitor de vídeo. Ao executar o programa com o comando **G,** observe a saída da informação **0123456789**.

Imagine então apresentar os 256 caracteres da tabela ASCII desta forma. Para facilitar esse tipo de trabalho é que entra em ação o *procedimento*. A partir do endereço de deslocamento **0100** será fornecido o código do programa principal. Utilizando o comando **A**, informe o código a seguir indicado do lado esquerdo:

```
07E9:0100 MOV AH,02     | B4 02
07E9:0102 MOV CL,0A     | B1 0A
07E9:0104 MOV DL,30     | B2 30
07E9:0106 CALL 0200     | E8 F7 00
07E9:0109 LOOP 0106     | E2 FB
07E9:010B INT 20        | CD 20
```

A partir do endereço de deslocamento **0200** é fornecido o código de programa correspondente ao trecho da rotina de procedimento. Utilizando o comando **A**, informe o código a seguir indicado do lado esquerdo:

```
07E9:0200 INT 21        | CD 21
07E9:0202 INC DL        | FE C2
07E9:0204 RET           | C3
```

Na sequência execute o comando **G** e será apresentada a mensagem:

```
0123456789
Program terminated (0000)
```

A rotina principal (**0100** até **010B**) do programa define na primeira linha de código (**07E9:0100**) a apresentação de um único caractere. Depois na segunda linha (**07E9:0102**) é definido o valor **0A** (valor decimal **10**) que será usado para contar o laço executado pela linha de código (**07E9:0109**). Toda vez que o comando **LOOP** é executada, ocorre um decremento de **1** no valor armazenado no registrador **CL**.

A terceira linha do programa principal (**07E9:0104**) estabelece o valor do primeiro caractere a ser apresentado (no caso o caractere **0** – zero representado pelo código ASCII do caractere zero como **30h**). A quarta linha de código do programa principal (**07E9:0106**), por meio do comando **CALL**, faz a chamada do trecho secundário do programa, ou seja, de uma sub-rotina.

Antes da execução do comando **CALL** ocorre a atualização do registrador **IP** com o endereço no qual se encontra a primeira instrução após a chamada do comando **CALL** que representa o valor de endereço de retorno da sub-rotina a ser utilizado pelo comando **RET**. Dependendo do tipo de procedimento em uso ter-se-á a atualização do registrador **CS** com o endereço do segmento de memória.

A chamada de uma sub-rotina pode ser feita de forma *direta* ou *indireta*, tanto na modalidade *intra-segmento* (**NEAR**) como na modalidade *inter-segmento* (**FAR**). Uma chamada intra-segmento ocorre no mesmo segmento de memória e uma chamada intra-segmento ocorre em segmentos diferentes de memória.

O uso de procedimentos faz com que o recurso de pilha da memória seja usado, onde *pilha* é uma região da memória que efetua o armazenamento temporário de dados.

Na chamada **direta intra-segmento** o registrador **SP** é subtraído de **02h**, é colocado o valor do registrador **IP** na pilha e efetua-se a soma do deslocamento entre o destino e a instrução seguinte a definição do comando **CALL** ao registrador **IP**.

Na chamada **direta inter-segmento** o registrador **SP** é subtraído de **02h**, é colocado o valor do registrador **CS** na pilha, o registrador **IP** é subtraído de **02h**, é colocado o valor do registrador **IP** na pilha e carrega-se os registradores **CS** e **IP** com o endereço de destino.

Na chamada **indireta intra-segmento** o registrador **SP** é subtraído de **02h**, é colocado o valor do registrador **IP** na pilha e carrega-se o registrador **IP** com o conteúdo do operando.

Na chamada **indireta inter-segmento** o registrador **SP** é subtraído de **02h**, é colocado o valor do registrador **CS** na pilha, o registrador **IP** é subtraído de **02h**, coloca-se o valor do registrador **CS** na pilha e carrega-se os registradores **CS** e **IP** com o conteúdo dos operados.

Os tipos de chamadas de um procedimento em *Assembly* dependem do tipo de operando em uso, que podem ser:

- rótulo **NEAR** usa ação **direta intra-segmento**;
- rótulo **FAR** usa ação **direta inter-segmento**;
- **registrador** usa ação **indireta intra-segmento**;
- variável **word** usa ação **indireta intra-segmento**;
- variável **dword** usa ação **indireta inter-segmento**.

Na primeira linha do trecho secundário do programa (**07E9:0200**) é feita a interrupção **21** que apresenta o caractere correspondente ao código ASCII armazenado no registrador geral **DX**. Na segunda linha (**07E9:0202**) do programa secundário encontra-se a instrução **INC** (*increment*), que faz o incremento de **1** ao valor atual armazenado no registrador geral **DX**.

A título de ilustração a instrução **INC** faz o incremento de mais um, a operação inversa pode ser realziada com o uso da instrução **DEC**.

Observe a terceira linha (**07E9:0204**) do programa secundário que executa a instrução **RET**, a qual retorna a execução do programa para a próxima linha do programa principal, após a instrução **CALL**.

O comando **RET** carrega o registrador **IP** com o valor (conteúdo) que estiver no topo da pilha, incrementa com **02h** o registrador **SP**. Se o retorno vier de uma chamada do tipo inter-segmento carrega o registrador **CS** com o valor do topo da pilha e adiciona **02h** ao registrador **SP**.

É muito importante se tomar o cuidado nas operações com procedimentos de não se alterar o valor do topo da pilha, o qual representa o valor do endereço de retorno para o programa principal. Se este valor for alterado o programa perderá a referência de retorno.

6.3 - Utilização da pilha

Quando os comandos **CALL** e **RET** chamam, respectivamente, um procedimento e seu posterior retorno, elas utilizam como parâmetro de controle operacional as informações armazenadas nos registradores de segmento **CS** (*Code Segment*), **DS** (*Data Segment*), **ES** (*Extra Data Segment*) e **SS** (*Stack Segment*), em conjunto com as informações do registrador de apontamentos **SP** (*Stack Pointer*). A pilha é usada para manter o endereço de retorno de uma instrução quando realizada a chamada de um procedimento ou para manter armazenado qualquer dado manipulado pelos comandos **PUSH** e **POP**, que serão vistos mais adiante.

As principais ações efetuadas pela pilha são: passar parâmetros para sub-rotinas como procedimentos e funções; efetuar a chamada de sub-rotinas e retornar as ações dessas sub-rotinas; preservar os valores dos registradores quando utilizados por sub-rotinas; preservar dados na memória; transferir dados sem que sejam utilizados os registradores; alterar a ordem de dados armazenados.

Para visualizar o controle da operação de manipulação da pilha, o programa atual é executado com auxílio dos comandos **T** e **P**. Antes de iniciar a execução do programa, acione o comando **R** e observe os pontos marcados em negrito, os quais indicam os registradores que são usados para as ações de controle da pilha:

```
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 B402              MOV     AH,02
```

Os registradores de segmento **DS**, **ES**, **SS** e **CS** apontam para o segmento de memória **07E9** e o registrador de apontamento **SP** tem o valor hexadecimal **FFFE** (ou outro endereço qualquer que representa o topo da pilha dos registradores de segmento, sendo este valor apresentado pelo próprio programa **Enhanced DEBUG**).

Os valores dos registradores de segmento **DS**, **ES**, **SS** e **CS** não serão alterados por estar sendo executada a chamada de procedimento no modo intra-segmento. Caso a execução da chamada de um procedimento ocorra fora do segmento apontado por **DS**, **ES**, **SS** e **CS** este será em modo inter-segmento.

Nessa etapa inicial acione o comando **P** por três vezes e observe as informações dos registradores de segmento e de apontamento marcadas em negrito mantendo-se inalteradas:

```
-P
AX=0200 BX=0000 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
07E9:0100 B10A              MOV     CL,0A

-P
AX=0200 BX=0000 CX=000A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0104 NV UP EI PL NZ NA PO NC
07E9:0104 B230              MOV     DL,30

-P
AX=0200 BX=0000 CX=000A DX=0030 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0106 NV UP EI PL NZ NA PO NC
07E9:0106 E8F700           CALL     0200
```

Após a terceira etapa de execução do comando **P** a linha de código **07E9:0106** aponta para a chamada do procedimento por meio do comando **CALL 0200**. Neste momento execute o comando **T**.

```
AX=0200 BX=0000 CX=000A DX=0030 SP=FFFC BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0200 NV UP EI PL NZ NA PE NC
07E9:0200 CD21              INT     21
```

O registrador de apontamento **IP** mostra o valor **0200** que é o valor inicial da rotina do procedimento. Veja também a informação do registrador de apontamento **SP** que anteriormente mostrava o valor **FFFE** e agora apresenta o valor **FFFC** (lembrando que esses valores poderão ser diferentes no computador em uso). Acione o comando **P** mais três vezes e observe os pontos em negrito:

```
-P
AX=0230 BX=0000 CX=000A DX=0030 SP=FFFC BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0202 NV UP EI PL NZ NA PE NC
07E9:0202 FEC2              INC     DL

-P
AX=0230 BX=0000 CX=000A DX=0031 SP=FFFC BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0204 NV UP EI PL NZ NA PO NC
07E9:0204 C3                RET
```

```
-P
AX=0230 BX=0000 CX=000A DX=0031 SP=FFFE BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0109 NV UP EI PL NZ NA PO NC
07E9:0109 E2FB             LOOPW    0106
```

Na última execução do comando **P** o valor do registrador de apontamento **SP** volta a ser **FFFE** e o valor do registrador de apontamento **IP** passa de **0204** para **0109**.

Repita a ação descrita mais algumas vezes observando vagarosamente os detalhes indicados. Os comandos **CALL** e **RET** operam de acordo com as informações dos registradores de segmento e apontamento. **CALL** faz a chamada de um procedimento e armazena o endereço desta chamada na pilha, a instrução **RET** recebe o valor de enderço da pilha e retorna o fluxo de ação do programa de acordo com o deslocamento (*offset*). O mesmo efeito ocorre com o uso do comando **INT** que faz a chamada de sub-rotinas internas para efetuar seu controle operacional.

O uso da pilha é um processo que facilita muitas operações de manipulação de valores (*byte* ou *word*), pois é possível armazenar valores na pilha ou retirá-los via programação pelas instruções específicas para essa finalidade, tais como **PUSH** (envia um dado para a pilha) e **POP** (retira um dado da pilha). Os comandos **PUSH** e **POP** são outra forma de efetuar movimentação de dados na memória.

O comando **PUSH** quando executado não altera nenhum registrador de estado, possuindo como possibilidade de trabalho as operações sobre:

- registrador (para este tipo de ação usa 1 *byte* de memória);
- memória (para este tipo de ação usa de 2 a 4 *bytes* de memória);
- segmento_registro, memória (para este tipo de ação usa 1 *byte* de memória);
- constante (para este tipo de ação usa de 2 a 3 *bytes* de memória).

O comando **POP** quando executado não altera nenhum registrador de estado, possuindo como possibilidade de trabalho as operações sobre:

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

Manipular uma pilha de valores é útil em situações nas quais é preciso guardar um valor de um determinado registrador antes de um procedimento ser executado e recuperar o valor antes da finalização do procedimento.

Para demonstrar essa ação, saia do programa **Enhanced DEBUG** e carregue-o em seguida novamente na memória. Depois entre o valor hexadecimal **000A** para o registrador geral **AX** e o valor **000B** para o registrador geral **BX**, de forma semelhante à seguinte informação:

```
R AX 000A <Enter>
R BX 000B <Enter>
R IP 0100 <Enter>
```

Após execute o comando **R**:

```
AX=000A BX=000B 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
```

A partir do endereço de deslocamento **0100** entre com o comando **A** o código do programa principal indicado a seguir e posicionado do lado esquerdo:

```
07E9:0100 PUSH AX       | 50
07E9:0101 PUSH BX       | 53
07E9:0102 CALL 0200     | E8 FB 00
07E9:0105 INC AX        | 40
07E9:0106 INC BX        | 43
07E9:0107 CALL 0200     | E8 F6 00
07E9:010A POP BX        | 5B
07E9:010B POP AX        | 58
07E9:010C INT 20        | CD 20
```

Na sequência, a partir do endereço de deslocamento **0200** com o comando **A** entre o código da rotina de procedimento indicado a seguir e posicionado do lado esquerdo:

```
07E9:0200 MOV AX,0001   | B8 01 00
07E9:0203 MOV BX,0002   | BB 02 00
07E9:0206 INC AX        | 40
07E9:0207 INC BX        | 43
07E9:0208 RET           | C3
```

No código do programa principal a primeira (**07E9:0100**) e a segunda (**07E9:0101**) linhas guardam na pilha os valores atuais dos registradores gerais **AX** e **BX**.

Na terceira linha (**07E9:0102**) do programa principal é feita a chamada pela instrução **CALL** da rotina de procedimento definida no endereço de deslocamento **0200**.

Dentro do trecho da rotina de procedimento o programa move os valores **0001** e **0002,** respectivamente, para os registradores gerais (primeira linha **07E9:0200**) **AX** e (segunda linha **07E9:0203**) **BX**. Nessa etapa de execução os registradores gerais **AX** e **BX** perdem os valores originais e assumem os novos valores devido ao uso da instrução **MOV**.

Depois na terceira (**07E9:0206**) e quarta (**07E9:0207**) linhas da rotina de procedimento o programa incrementa em **1** os valores existentes nos registradores gerais **AX** e **BX**. A quinta linha (**07E9:0208**) da rotina de procedimento executa a instrução **RET** que devolve o fluxo de execução do programa para a primeira linha do programa principal após a execução do comando **CALL**. Após a primeira chamada do procedimento o retorno ocorre na quarta linha do programa principal (**07E9:0105**).

A quarta (**07E9:0105**) e quinta (**07E9:0106**) linhas do programa principal incrementam o valor **1** aos valores atuais dos registradores gerais **AX** e **BX**. Depois na sexta linha (**07E9:0107**) do programa principal faz nova chamada à rotina de procedimento, que repete sua ação incrementando o valor **1** ao valor atual dos registradores gerais **AX** e **BX** e executando o comando **RET**. Após a segunda chamada da rotina de procedimento o retorno ocorre na sétima linha (**07E9:010A**) do programa principal.

De volta ao programa principal o comando **POP** encontrada na sétima (**07E9:010A**) e oitava (**07E9:010B**) linhas do programa recupera da pilha os valores iniciais dos registradores gerais **AX** e **BX**, ou seja, os registradores gerais **AX** e **BX** voltam, respectivamente, a ter os valores originais **000A** e **000B**. O uso do comando **POP** sempre ocorre no sentido inverso ao uso do comando **PUSH**. Por exemplo, se executada as instruções **PUSH AX**, **PUSH BX**, **PUSH CX** para guadar na pilha os valores dos registradores **AX**, **BX** e **CX** e desejando-se retornar os valores da pilha para seus respectivos registradores deverá ser usada a sequencia de instruções **POP CX**, **POP BX** e **POP AX**.

A seguir, a partir do registrador de deslocamento **IP** posicionado no endereço de deslocamento **0100**, execute o comando **R** uma vez para verificar se os registradores gerais **AX** e **BX** estão com os valores **000A** e **000B**, atentando para o valor **FFFE** do registrador **SP**.

```
AX=000A BX=000B 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 PO NC
07E9:0100 50               PUSH    AX
```

Observe cuidadosamente cada informação apresentada, principalmente nos pontos marcados em negrito. Na sequência execute o comando **T** para que a primeira linha do programa principal seja executada.

```
AX=000A BX=000B CX=0000 DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0101 NV UP EI PL ZR NA PO NC
07E9:0101 53              PUSH    BX
```

O valor do registrador de apontamento **SP** mudou de **FFFE** para **FFFC**. A mudança ocorreu pelo fato de o valor **000A** do registrador geral **AX** ter sido armazenado no topo da pilha. Houve uma atualização do valor para determinar o novo topo da pilha. Na sequência execute o comando **T** para que a segunda linha do programa principal seja executada.

```
AX=000A BX=000B CX=0000 DX=0000 SP=FFFA 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 E8FB00          CALL     0200
```

O valor do registrador de apontamento **SP** mudou de **FFFC** para **FFFA**. A mudança ocorreu pelo fato de o valor **000B** do registrador geral **BX** ter sido armazenado no topo da pilha. Houve uma atualização do valor para determinar o novo topo da pilha. Toda vez que for usado o comando **PUSH** o registrador de apontamento **SP** será atualizado para determinar o próximo topo da pilha.

A próxima instrução a ser executada pelo programa principal é a chamada da execução da rotina de procedimento, que se encontra no endereço de deslocamento **0200**. Na sequência execute o comando **T** para que a terceira linha do programa principal seja executada e ocorra o desvio do fluxo de execução do programa para dentro do procedimento.

```
AX=000A BX=000B CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0200 NV UP EI PL NZ NA PO NC
07E9:0200 B80100            MOV     AX,0001
```

Ao ser executada a rotina de procedimento, o programa atualiza o valor do registrador de apontamento **SP** de **FFFA** para **FFF8**, pois o comando **CALL** também utiliza a informação da pilha para sua execução. O valor **FFFA** será restaurado quando da execução do comando **RET**.

Na posição atual do registrador de apontamento **IP** que marca o endereço **0200** encontra-se a primeira instrução da rotina de procedimento. Execute o comando **T** por duas vezes para que a primeira e segunda linhas da rotina de procedimento sejam executadas.

```
-T
AX=0001 BX=000B CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0203 NV UP EI PL NZ NA PO NC
07E9:0203 BB0200            MOV     BX,0002

-T
AX=0001 BX=0002 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0206 NV UP EI PL NZ NA PO NC
07E9:0206 40                INC     AX
```

Os registradores gerais **AX** e **BX** perdem o valor anterior e assumem um "novo" valor devido ao uso do comando **MOV**. Para preservar os valores originais dos registradores **AX** e **BX** foi utilizada a instrução **PUSH**. Perceba que o valor do registrador **SP** se manteve constante nas duas ações anteriores.

Na sequência execute o comando **T** por mais duas vezes e observe a atualização dos valores dos registradores gerais **AX** e **BX** e o comportamento do registrador de apontamento **SP**.

```
-T
AX=0002 BX=0002 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0207 NV UP EI PL NZ NA PO NC
07E9:0207 43                INC     BX

-T
AX=0002 BX=0003 CX=0000 DX=0000 SP=FFE8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0208 NV UP EI PL NZ NA PE NC
07E9:0208 C3                RET
```

Nesse ponto, ao ser solicitada a execução da última linha de código do programa de procedimento com o comando **T**, ocorre o retorno à quarta linha do programa principal.

```
AX=0002 BX=0003 CX=0000 DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0105 NV UP EI PL NZ NA PE NC
07E9:0105 40                INC     AX
```

Observe nesse ponto a alteração do valor do registrador **SP** que volta a marcar o topo da pilha como sendo **FFFA**, valor este da última posição da pilha antes da chamada da rotina de procedimento. Execute em seguida o comando **T** duas vezes.

```
-T
AX=0003 BX=0003 CX=0000 DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0106 NV UP EI PL NZ NA PE NC
07E9:0106 43                INC     BX

-T
AX=0003 BX=0004 CX=0000 DX=0000 SP=FFFA 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 E8F600            CALL    0200
```

O endereço do registrador de apontamento **SP** se manteve constante durante os passos anteriores executados no programa principal. Nesse último momento o programa principal aponta para a chamada da rotina de procedimento no endereço de deslocamento **0200**. Nessa etapa acione o comando **T** uma vez.

```
AX=0003 BX=0004 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0200 NV UP EI PL NZ NA PO NC
07E9:0200 B80100            MOV     AX,0001
```

Ao ser executado o acesso à rotina de procedimento, o ponteiro de apontamento **SP** volta a ficar com o valor **FFF8**. Na sequência execute o comando **T** quatro vezes.

```
-T
AX=0001 BX=0004 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0203 NV UP EI PL NZ NA PO NC
07E9:0203 BB0200            MOV     BX,0002

-T
AX=0001 BX=0002 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0206 NV UP EI PL NZ NA PO NC
07E9:0206 40                INC     AX

-T
AX=0002 BX=0002 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0207 NV UP EI PL NZ NA PO NC
07E9:0207 43            INC     BX

-T
AX=0002 BX=0003 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=0208 NV UP EI PL NZ NA PE NC
07E9:0208 C3                RET
```

Perceba a mudança dos valores dos registradores gerais **AX** e **BX**. Em relação à anterior os valores mudaram novamente devido ao uso dos comandos **MOV** e **INC**. Nessa etapa, acione o comando **T** e observe o local de retorno do programa, principalmente na informação do registrador **SP** e na indicação da próxima linha de execução do programa.

```
AX=0002 BX=0003 CX=0000 DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010A NV UP EI PL NZ NA PE NC
07E9:010A 5B                POP     BX
```

Acione o comando **T** uma vez. Perceba o retorno do valor **000B** para o registrador geral **BX**. Este foi o último valor inserido na pilha, ou seja, o último valor empilhado é o primeiro a ser desempilhado, pois de outra forma, a operação não pode ser realizada.

```
AX=0002 BX=000B CX=0000 DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=07E9 ES=07E9 SS=07E9 CS=07E9 IP=010B NV UP EI PL NZ NA PE NC
07E9:010B 58                POP     AX
```

Ao ser retirado um valor da pilha, o registrador de apontamento **SP** é atualizado para **FFFC**. Execute mais uma vez o comando **T**.

```
AX=000A BX=000B 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 NZ NA PE NC
07E9:010C CD20              INT     20
```

Note o retorno do valor **000A** para o registrador geral **AX** e o ajuste do valor do registrador de apontamento **SP** para **FFFE**, valor inicial antes da execução da primeira linha de código do programa principal. Depois, para finalizar o programa, acione apenas uma vez o comando **P**.

A pilha permite manipular a preservação de valores, dando a possibilidade de trabalhar em baixo nível do mesmo modo que em linguagens de programação de alto nível, quando se utilizam passagens de parâmetro por valor ou referência em sub-rotinas.

6.4 - Consistência da entrada de dados

A explanação do tópico anterior foi necessária, uma vez que não se deve confiar na ação de um usuário de programa. Devido a esta questão, o desenvolvedor de um programa precisa se preocupar com alguns detalhes, como o que aconteceria na entrada de valores hexadecimais se ele entrasse um caractere inválido, diferente da sequência **0123456789ABCDEF**.

O objetivo do próximo programa é apenas aceitar a entrada de caracteres válidos, ou seja, valores numéricos de 0 até 9 e caracteres alfabéticos de A até F em maiúsculos. Qualquer outro caractere não será aceito pelo programa.

O programa seguinte é formado por três partes de código, sendo uma parte o programa principal e as outras duas partes as rotinas de procedimento (uma secundária e outra terciária).

A rotina principal do programa é responsável por todo o controle aceitando até dois dígitos válidos. Para a definição dessa rotina utilize o comando **A** e a partir do endereço de deslocamento **0100**, informe o código a seguir indicado do lado esquerdo:

```
07E9:0100 SUB DX,DX     | 29 D2
07E9:0102 CALL 0200     | E8 FB 00
07E9:0105 MOV DL,AL     | 88 C2
07E9:0107 MOV CL,04     | B1 04
07E9:0109 SHL DL,CL     | D2 E2
07E9:010B CALL 0200     | E8 F2 00
07E9:010E INT 20        | CD 20
```

A primeira instrução do programa efetua a limpeza do registrador geral **DX** com a instrução **SUB DX,DX** eliminando eventualmente qualquer valor que exista na posição preparando o registrador para as ações do programa.

A segunda instrução **CALL 0200** efetua a chamada do procedimento que se encontra no endereço de deslocamento **0200** (procedimento secundário) responsável pela leitura do primeiro caractere advundo do teclado. O valor numérico aceito precisa ser reposicionado na memória para que a segunda entrada ocorra. Esta ação é realizada pelas próximas três instruções.

A terceira instrução **MOV DL,AL**, após a entrada do caractere numérico validado, pega o conteúdo informado e armazenado no registrador **AL** e o transfere para o registrador **DL**.

A quarta instrução **MOV CL,04** coloca no registrador **CX** o valor **04** para rotacionar os bits do valor numérico informado.

A quinta instrução **MOV CL,DL** faz o rotacionamento para à esquerda dos caracteres junto ao registrador **DL** deixando-o pronto para ser apresentado em conjunto com o segundo caractere lido pela sexta instrução.

A sexta instrução **CALL 0200**, assim como a segunda instrução faz a chamada do procedimento que efetua a leitura do caractere.

A sétima instrução **INT 20** faz o encerramento do programa devolvendo o controle de operação ao sistema operacional.

A rotina de procedimento secundária (será chamada pelo programa principal), responsável pela recepção de apenas um caractere hexadecimal válido por vez. Esta rotina de código deve ser codificada a partir do endereço de deslocamento **0200**, por meio do comando **A**, como apresentado em seguida:

```
07E9:0200 PUSH DX       | 52
07E9:0201 MOV AH,08     | B4 08
07E9:0203 INT 21        | CD 21
07E9:0205 CMP AL,30     | 3C 30
07E9:0207 JL 0203       | 7C FA
07E9:0209 CMP AL,46     | 3C 46
07E9:020B JG 0203       | 7F F6
07E9:020D CMP AL,39     | 3C 39
07E9:020F JG 0219       | 7F 08
07E9:0211 CALL 0300     | E8 EC 00
07E9:0214 SUB AL,30     | 2C 30
07E9:0216 POP DX        | 5A
07E9:0217 JMP 0223      | EB 0A
07E9:0219 CMP AL,41     | 3C 41
07E9:021B JL 0203       | 7C E6
07E9:021D CALL 0300     | E8 E0 00
07E9:0220 SUB AL,37     | 2C 37
07E9:0222 POP DX        | 5A
07E9:0223 RET           | C3
```

A primeira instrução **PUSH DX** pega o valor existente em **DX** e o coloca na pilha, deixando este registrador livre para outro uso no programa. Isto é importante para salvaguardar o valor existente em **DL**. É pertinente perceber que o registrador **DL** também é usado pelo trecho de código do programa no sentido de seu conteúdo ser apresentado quando da execução da instrução **CALL 0300**.

A segunda instrução **MOV AH,08** habilita no registrador **AH** o valor de função **08** responsável por permitir que o teclado opere sem que apresente o caractere lido. Esta função faz o efeito de teclado sem eco. Este recurso está sendo implementado para que não seja apresentado no monitor de vídeo nenhum caractere que seja inválido. O programa somente apresenta caracteres de 0 até 9 e de A até F.

A terceira instrução **INT 21** dá acesso ao uso do teclado configurado para efetuar a leitura de caracteres sem que os mesmos sejam apresentados no monitor de vídeo. A apresentação dos valores hexadecimais ocorrerá somente pelo controle do procedimento no endereço de deslocamento **0300** para os caracteres validados.

A quarta instrução **CMP AL,30** faz a subtração do valor **30** do valor do caractere numérico lido e armazenado no registrador **AL** no sentido de veriticar se o caractere ASCII informado é válido. Desta forma, é possível ter o valor que será tratado condicionalmente por uma instrução de desvio condicional, neste caso **JL** (*Jump on Less*) definido junto a quinta instrução.

A quinta instrução **JL 0203** pega o valor avaliado pela instrução **CMP AL,30** e verifica se o valor numérico do caractere informdo é um valor abaixo de **0** (zero). Se a condição for verdadeira o fluxo do programa será desviado para a linha **0203** (terceira linha) para que a entrada continue. Se o caractere numérico informado for válido o programa acata automaticamente o valor validado.

A sexta instrução, assim como a quarta instrução **CMP AL,46** subtrais o valor **46** do caractere informado para saber se o caractere tem valor acima do código ASCII da letra **F**. esta instrução gera um valor lógico que será tratado pela instrução de desvio condicional **JG** (*Jump on Greater*).

A sétima instrução **JG 0203** desvia o fluxo do programa para o endereço de deslocamento **0203** caso o caractere informado seja maior que a letra **F**. Se o caractere numérico informado for válido o programa acata automaticamente o valor validado.

A oitava instrução **CMP AL,39** efetua a subtração para ter o valor que permita verificar caracteres acima do valor **9**. Se o valor estiver acima de **9** este precisará ser verificado se é ou não um caractere alfabético.

A nona instrução **JG 0219** desvia o fluxo do programa para o endereço de deslocamento **0219** caso o caractere informado seja maior que o número **9**. Não sendo maior que **9** o primeiro caractere já válido pode na sequência ser apresentado no monitor de vídeo por meio da chamada **CALL 0300**.

A décima instrução **CALL 0300** efetua a apresentação do primeiro caractere validado.

A décima primeira instrução **SUB AL,30** ajusta o registrador **AL** subtraindo o valor **30** do valor primeiro valor informado na entrada.

A décima segunda instrução **POP DX** recupera o primeiro valor colocado anteriormente na pilha.

A décima terceira instrução **JMP 0223** efetua um salto incondicional para o endereço de deslocamento **0223** onde é realizado o encerramento do programa. O comando **JMP** (*Jump*) é usado quando se deseja desviar o fluxo de um programa para certo ponto do código sem o uso de avaliação condicional.

A décima quarta instrução **CMP AL,41** efetua a subtração em relação ao segundo valor hexadecimal para ter o valor que permita verificar caracteres abaixo da letra **A**. Se o valor estiver abaixo da letra **A** este precisará ser verificado se é ou não um caractere alfabético.

A décima quinta instrução **JL 0203** desvia para a reentrada de um caractere que tenha sido informado de foma inválida.

A décima sexta instrução **CALL 0300** efetua a apresentação do segundo caractere validado.

A décima sétima instrução **SUB AL,37** ajusta o registrador **AL** subtraindo o valor **37** do valor segundo valor informado na entrada.

A décima oitava instrução **POP DX** recupera o segundo valor colocado anteriormente na pilha.

A décima nona instrução **RET** faz o retorno da chamada do procedimento ao programa principal.

A rotina de procedimento terciário (chamada pelo procedimento secundário) é responsável pela apresentação do caractere válido, deve ser codificada a partir do endereço de deslocamento **0300**, pelo comando **A**, informe o código a seguir indicado do lado esquerdo:

```
07E9:0300 MOV AH,02     | B4 02
07E9:0302 MOV DL,AL     | 88 C2
07E9:0304 INT 21        | CD 21
07E9:0306 RET           | C3
```

A primeira instrução **MOV AH,02** é usada para armazenar o código de função **02** responsável por apresentar o caractere armazenado no registrador **DL**.

A segunda instrução **MOV DL,AL** transporta para **DL** o conteúdo do registrador **AL** que será apresentado.

A terceira instrução **INT 21** efetua a apresentação do caractere em **DL** a partir do código de função **02**.

A quarta instrução **RET** faz o retorno da chamada do procedimento ao procedimento secundário.

Em seguida execute o comando **G**. O programa espera que dois caracteres válidos sejam fornecidos. Se o caractere for inválido o programa não faz sua recepção, aceitando apenas caracteres para a entrada de valores numéricos hexadecimais.

Anotações