```
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
0D82:0109 64            DB      64
0D82:010A 61            DB      61
0D82:010B 64            DB      64
0D82:010C 65            DB      65
0D82:010D 206573        AND     [DI+73],AH
0D82:0110 7065          JO      0177
0D82:0112 63            DB      63
0D82:0113 69            DB      69
0D82:0114 66            DB      66
0D82:0115 69            DB      69
0D82:0116 63            DB      63
0D82:0117 61            DB      61
```

Parte II

Programação com Enhanced DEBUG

```
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 3

FERRAMENTAS DE DESENVOLVIMENTO

*Este capítulo descreve a obtenção, instalação e configuração do ambiente de trabalho para a produção de programas escritos em linguagem Assembly. São apresentadas as ferramentas operacionais **DEBUGs** e **emu8086** que serão usadas ao longo do estudo desta obra.*

3.1 - Como começou, como está, como fica

Este trabalho surgiu com o objetivo de ser um material introdutório para a apresentação inicial da linguagem de programação *Assembly* voltada aos microprocessadores padrão 8086/8088 da empresa Intel e demais microprocessadores compatíveis como os da empresa AMD. O modelo 8086/8088 do microprocessador Intel foi usado no desenvolvimento do microcomputador pessoal da empresa IBM lançado no ano de 1981 e conhecido como IBM-PC.

Em 1980, o programador Tim Paterson trabalhando para a empresa **SCP** (*Seattle Computer Products*) desenvolveu o sistema operacional **QDOS** (*Quick and Dirty Operating System*) de 16 *bits*, posteriormente chamado de **86-DOS** para uma placa de circuitos integrados, chamada **8086 S-100**, baseada no microprocessador 8086 da empresa Intel, a qual o tinha projetado um ano antes.

Na fase de desenvolvimento do sistema operacional *QDOS* Tim Paterson escreveu um pequeno programa de depuração que veio a facilitar o desenvolvimento de rotinas internas desse sistema e de um dos chips de **ROM** (*Read Only Memory*) usado na placa *8086 S-100*. Um tempo depois Tim Paterson adaptou o código de seu programa depurador para ser executado dentro de seu sistema operacional com extensão **.COM** chamando-o de **DEBUG** (lê-se *debâgui*), adicionando a este programa a capacidade de desmontar o código de máquina do microprocessador *8086*.

No decorrer dessas ações a empresa Microsoft comprou os direitos do sistema operacional *86-DOS* e o adaptou para uso na linha de computadores pessoais da família IBM. Nesse período a Microsoft tinha como seu principal cliente a empresa *IBM* que estava desenvolvendo secretamente o primeiro microcomputador de 16 *bits* para concorrer com os microcomputadores de 8 *bits* que estavam sendo comercializados no mercado.

A Microsoft contratou temporariamente o programador Tim Paterson como principal autor de “seu” primeiro produto: o sistema operacional **IBM-DOS 1.00**. Assim que o “novo” sistema operacional foi concluído Tim Paterson incluiu o utilitário *DEBUG.COM* neste “novo” sistema operacional.

Posteriormente a empresa Microsoft lança o sistema operacional *MS-DOS* para os computadores clone *IBM-PC*, mantendo dentro de seu conjunto de utilitários o programa *DEBUG*, existente até os dias de hoje em todas as versões de 32 *bits* do sistema operacional Windows.

O utilitário *DEBUG* passou ao longo dos anos de sua existência por diversas mudanças e correções de erros. Nas mãos de um bom programador é uma ferramenta muito eficaz para operar arquivos e escrever pequenos programas em baixo nível. No entanto, com o surgimento do sistema operacional Windows acabou por perder um pouco de seu *glamour*. O programa *DEBUG* basicamente deixou de ser apreciado e utilizado quando seus comandos de I/O se tornaram menos confiáveis, seja devido a um 'bug' ou ao próprio sistema operacional Windows. O fato é que os comandos de I/O no Windows não são confiáveis para acesso direto a discos rígidos via *DEBUG*.

Embora tenha sido criado no início da era do microprocessador de 16 *bits* as versões mais recentes do utilitário *DEBUG* são muito úteis para técnicos de PCs no que tange ao acesso direto a determinadas posições de memória. O *DEBUG* nos dias de hoje pode ser útil para fins educacionais como será demonstrado neste trabalho no sentido de auxiliar o entendimento de alguns princípios internos da arquitetura do microprocessador *8086* e da apresentação de códigos em linguagem de máquina e linguagem *Assembly*.

Um dos maiores problemas do programa *DEBUG* desde seu lançamento no MS-DOS é a disponibilidade de uma documentação efetiva para seu uso. Isso se explica pelo fato de ter sido um programa criado para uso particular de Tim Paterson na elaboração do sistema operacional QDOS. Não era intenção inicial de que tal ferramenta fizesse parte de um sistema operacional comercial como ocorreu com o MS-DOS. A ferramenta *DEBUG* é um programa *nerd* voltado para *nerds* e *hackers* e sua documentação passa a não necessária a este público. No entanto, isso não justifica plenamente sua ausência e neste sentido um material mais acentuado a respeito da ferramenta *DEBUG* pode ser consultado no sítio **http://debug.manzano.pro.br/**.

Além do programa *DEBUG,* Tim Paterson escreveu um programa assembler (compilador de linguagem de montagem) chamado **ASM** que foi usado para escrever o sistema operacional *86-DOS*. Este programa foi base para que a Microsoft tivesse para seu sistema operacional *MS-DOS* um compilador de linguagem de montagem chamado **MASM** (Microsoft ASseMbler) que foi evoluindo ao longo dos anos, desde o lançamento de sua primeira versão em 1981, sendo disponibilizado atualmente em conjunto com o ambiente de programa *Visual Studio*.

Em paralelo a existência da ferramenta *MASM* outras ferramentas de compilação para linguagem de montagem surgiram em sua época, destacando-se o **TASM** (Turbo ASseMbler) de 1989 da empresa Borland (não mais distribuída), **FASM** (Flat ASseMbler) de 2000 escrito por Tomasz Grysztar e **NASM** (Netwide ASseMbler) de 2016 escrito por Simon Tatham e Julian Hall.

Os programas *MASM*, *TASM*, *FASM* e *NASM* são ferramentas para a produção de software, principalmente, comerciais podendo ser usadas pela área de educação. No entanto, aprendizagem de programação de baixo nível para novatos pode ser considerada difícil e ferramentas com toque profissional podem ser intimidadoras para alguns. Assim sendo, para uso didático há como alternativa o programa **emu8086** para o sistema operacional Windows que pode ser executado no sistema operacional Linux via programa **Wine**.

3.2 - Os Programas de depuração DEBUG

A partir do desenvolvimento do utilitário *DEBUG.COM* escrito para o sistema operacional *QDOS/86-DOS* e sua aquisição pela Microsoft para a IBM ocorreu a transição do sistema operacional *QDOS/86-DOS* para o *IBM-DOS* e posteriormente para o *MS-DOS*. Desta forma, passam a existir duas versões simultâneas do utilitário *DEBUG.COM* que foram amplamente usadas até o surgimento do sistema operacional Windows que, em modo 32 *bits*, possui internamente uma versão do utilitário *DEBUG* de 16 *bits* que pode ser normalmente executado em uma janela de comando do sistema.

No entanto, não é possível executar nas edições de 64 *bits* do sistema operacional Windows programas escritos para o modo 16 *bits* estando o programa *DEBUG* nesta condição, a menos que se utilize alguma versão clone do utilitário de depuração *DEBUG* (evitando-se a prática de pirataria ao fazer uso de uma versão própria da Microsoft) executado dentro de algum ambiente de virtualização que emule o ambiente do sistema operacional *MS-DOS*, como são os casos dos emuladores **vDos** (**https://vdos.info/**) ou **vDosPlus** (**http://vdosplus.org/**).

Alternativamente ao sistema operacional MS-DOS de 16 *bits* da Microsoft há um sistema operacional de 32 *bits* chamado **FreeDOS** que trás entre seus utilitários um clone do programa *DEBUG* chamado *DOS Debug* que pode ser executado em sistemas operacionais Windows de 64 bits por meio de ambientes de emulação **vDos** ou **vDosPlus**. O programa de depuração **DOS Debug** foi escrito pelo Professor Paul Vojta, sendo atualmente mantido pelo programador Andreas Grech (**https://www.ibiblio.org/pub/micro/pc-stuff/freedos/files/repositories/1.3/base/debug.zip**). A partir da versão 1.26 o utilitário de depuração do Professor Paul Vojta vem também sendo mantido, como uma variante, pelo programador Vernon Brooks (**https://pcdosretro.github.io/enhdebug.htm**) com o nome **Enhanced DEBUG** (*DEBUG* Aprimorado), na versão **1.32b**. O programa *Enhanced DEBUG* inclui em seu pacote o utilitário *DEBUG.COM* que é um substituto para o *DEBUG* original da Microsoft e o utilitário *DEBUGX.COM*, que depura programas em modo **DPMI** (*DOS Protected Mode Interface*), possuindo mais recursos, além dos recursos existentes no *DEBUG* (original) e no *DOS Debug*.

Além dos programas *DOS Debug/Enhanced DEBUG* há alternativamente um programa de depuração chamado **GRDB** (Get Real DeBugger) desenvolvido pela **LADSoft** (**http://ladsoft.tripod.com/grdb_debugger.html**) compatível com as ferramentas *DEBUG* comentadas que pode também ser utilizado dentro de ambientes de emulação.

Para o estudo desta obra será utilizado o pacote *Enhanced DEBUG* versão 1.32b a partir do utilitário *DEBUGX.COM* que será executado dentro do programa de emulação *vDos*.

3.3 - Programa emulador assembler emu8086

O programa **emu8086** foi lançado em 1997 sendo escrito pelos programadores Barry Allyn e Yuri Margolin. Este programa possui uma estrutura operacional e visual muito didática por permitir a visualização simultânea de diversos detalhes operacionais da execução de instruções de baixo nível de um microprocessador 8086.

O *emu8086* é um programa emulador do microprocessador 8086 possuindo um modo montador (assemblador) integrado, além de conter um conjunto de tutoriais e outros documentos voltados a iniciantes. O emulador executa programas como se fosse um microprocessador real permitindo visualizar a execução desses programas em modo passo a passo, mostrando as ocorrências geradas em todos os registradores, memória, pilha e variáveis, os quais podem ser acompanhados e editados com um duplo clique. As instruções de um programa podem ser executadas no *emu8086* passo a passo tanto para frente como para trás (LVSEVEN, 2018).

O objetivo do programa *emu8086* é ser um instrumento que facilite a aprendizagem da linguagem *Assembly 8086* com alto grau de facilidade de uso não encontrado nos programas montadores. O pacote de software inclui vários dispositivos virtuais externos, como: robô, motor de passo, display de LED e interseção de semáforos. Além de emular um microprocessador 8086 esta ferramenta executa dentro de sua instância (máquina virtual) de operação programas escritos em modo 16 *bits* mesmo em um sistema operacional Windows de 64 *bits*. Além de emular didaticamente um ambiente de 16 *bits* para o microprocessador 8086, o programa *emu8086* pode ser usado como um programa montador por ter integrado a ele o programa *FASM*.

É propício comentar que apesar do programa *emu8086* ser uma excelente ferramenta de estudo ela possui um pequeno problema de distribuição. Algumas vezes fica seu sítio oficial fora do ar o que prejudica a compra da licença para seu uso uma vez que o *emu8086* não é uma ferramenta *freeware*. Quanto a obtenção do programa *emu8086* este pode ser conseguido a partir de diversos serviços de *downloads* existente na *Internet* e estará seu uso restrito ao tempo de teste que é oferecido para um usuário não registrado. Por exemplo, durante o segundo semestre de 2018 o sítio do programa **http://emu8086.com/** esteve fora do ar, sendo esta a segunda vez desta ocorrência desde 1997. Além do sítio oficial, na documentação da versão do programa usada para a elaboração deste trabalho encontra-se como opção de contato o e-mail **air.plane@dr.com**.

3.4 - Preparação do ambiente de trabalho

Para o estudo deste livro é necessário o uso de três programas: **emu8086, vDos** (ou **vDosPlus**) e **Enhanced DEBUG**. O programa *vDos* ou o programa *vDosPlus* são necessários para a execução do programa *Enhanced DEBUG* junto ao sistema operacional Windows de 64 *bits*. Escolha um dos programas

Se a escolha for o programa **vDos** acesse o sítio **https://vdos.info/** e selecione em seu menu a opção **Download** e na página apresentada acesse o *link* **Installation program 2022.05.01** em **Download vDos: Installation program 2022.05.01 | Source files or older versions** para que o programa de instalação seja copiado para seu computador. A partir da cópia do programa **vDosSetup.exe** selecione-o com um duplo clique do ponteiro do *mouse* e proceda a instalação junto a pasta **vDos** no disco **C:** de seu computador.

Se a escolha for o programa **vDosPlus** acesse o sítio **https://vdos.info/** e selecione o *link* **Standalone (offline) installer [Recommended]** para que o programa de instalação seja copiado para seu computador. A partir da cópia do programa **vDosPlus-201511-setup.exe** selecione-o com um duplo clique do ponteiro do *mouse* e proceda a instalação junto a pasta **vDosPlus** no disco **C:** de seu computador.

Após copiar e instalar o programa *vDos* ou o programa *vDosPlus* é necessário obter e instalar o programa de depuração *Enhanced DEBUG*. Para tanto, acesse o sítio **https://pcdosretro.github.io/enhdebug.htm**, e selecione no final da página o *link* **Download DEBUG 1.32b** que copiará o arquivo **DEBUGX.ZIP**. A partir da cópia do arquivo **DEBUGX.ZIP** descompacte-o e copie a pasta gerada para a raiz do disco **C:**.

A partir da instalação dos programas *vDos/vDosPlus* e *Enhanced DEBUG* é necessário fazer a configuração para uso do programa *Enhanced DEBUG* dentro dos ambientes *vDos/vDosPlus*. Para tanto, utilizando-se do programa **Explorer** do Windows abra uma das pastas *vDos* ou *vDosPlus* do disco *C:* e selecione o arquivo **autoexec.txt** acrescentando no final do arquivo a linha de instrução **use e: c:\debugx**.

A partir da instalação e configuração dos programas anteriores é necessário obter e instalar o programa de aprendizagem *emu8086*. Assim sendo, acesse o sítio **http://www.emu8086.com/** (ou outro serviço de sua confiança que possa fornece-lo) e selecione o *link* **emu8086v408r10.zip** (17 de novembro de 2015 quando a versão **4.08r10** estava disponível) e baixe para sua pasta de trabalho, descompacte o arquivo e execute o programa **setup** e siga as orientações apresentadas acionando os botões **Next** e **Finish** quando apresentados.

Observação

Caso seja apresentada mensagem de erro com arquivo **dll** vá ao sítio **http://support.microsoft.com/gp/vbruntime**, selecione a partir da seção **Visual Basic 6.0** selecione a opção **FILE: VBRUN60.EXE Installs Visual Basic Run-Time Files**. Na página apresentada selecione o *link* **Baixe o pacote VBRun60.exe agora** e baixe-o para sua pasta de trabalho. Após obtenção do arquivo **vbrum60.exe** proceda sua instalação.

Concluído o processo de instalação, o programa pode ser acionado por intermédio do ícone na área de *desktop* do Windows **emu8086**. Assim sendo, dê um duplo clique com o ponteiro do *mouse* sobre o ícone **emu8086**, selecione o botão **New** na caixa de diálogo **welcome**. Na caixa de diálogo **choose code template** mantenha a seleção da opção **COM template** e acione o botão **OK**.

Após os passos anteriores será apresentada a tela do ambiente de trabalho do programa **emu8086**. Inicialmente, como primeira execução será definida a configuração do tipo de caractere em que os programas serão apresentados dentro do ambiente do **emu8086**. Acione na barra de ferramentas do programa o botão **options** (nono botão) e aparece a caixa de diálogo **options** para ter acesso a caixa de diálogo **options**.

Acione o botão **set font** da área **fonts and colors** e defina para a opção **control** como **source code editor** a fonte de trabalho **Courier New** no formato **Negrito**, deixe como tamanho o valor **12** e acione o botão **OK**. No quadro **backgroud color** da área **fonts and colors** dê um clique e selecione no quadro de cores apresentados a opção para cor branca. Na área **fonts and colors** selecione para **control** a opção **memory list** e defina para **background color** a cor branca e em seguida para a opção **disassembler list** defina a mesma opção de **memory list**. No quadro **keywords** junto à opção **item** selecione com o botão *drop-box* a opção **line numbers** e para a opção **backgroud color** selecione a cor cinza claro e para a opção **foreground color** selecione a cor cinza escuro. Ao final acione o botão **close** para fechar a janela de caixa de diálogo **options**.