Tutorial do Debug

O debug não é um recurso profissional para desenvolvimento de aplicações em Assembly. Entretanto, sua simplicidade de operação nos permite rapidamente testar alguns comandos, até reunirmos conhecimento para desenvolver programas assemby para ser montados pelo TASM/TLINK.
Para a criação de um programa em assembler existem 2 opções: usar o TASM –Turbo Assembler da Borland, ou o DEBUGGER. Por enquanto iremos usar o debug, uma vez que podemos encontrá-lo em qualquer PC com o MS-DOS.
Debug pode apenas criar arquivos com a extensão .COM, e devido as características deste tipo de programa, eles não podem exceder os 64 Kb, e também devem iniciar no endereço de memória 0100H dentro do segmento específico. É importante observar isso, pois deste modo os programas .COM não são relocáveis.
Os principais comandos do programa debug são:
A Montar instruções simbólicas em código de máquina
D Mostrar o conteúdo de uma área da memória
E Entrar dados na memória, iniciando num endereço específico
G Rodar um programa executável na memória
N Dar nome a um programa
P Proceder, ou executar um conjunto de instruções relacionadas
Q Sair do programa debug
R Mostrar o conteúdo de um ou mais registradores
T Executar passo a passo as instruções
U Desmontar o código de máquina em instruções simbólicas
W Gravar um programa em disco
É possível visualizar os valores dos registradores internos da CPU usando o programa Debug. Debug é um programa que faz parte do pacote do DOS, e pode ser encontrado normalmente no diretório C:\DOS. Para iniciá-lo, basta digitar Debug na linha de comando:

```
C:/>Debug [Enter]
-
```

Você notará então a presença de um hífen no canto inferior esquerdo da tela. Não se espante, este é o prompt do programa. Para visualizar o conteúdo dos registradores, digite :

```
-r[Enter]
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0D62 ES=0D62 SS=0D62 CS=0D62 IP=0100 NV UP EI PL NZ NA PO NC
0D62:0100 2E CS:
0D62:0101 803ED3DF00 CMP BYTE PTR [DFD3],00
CS:DFD3=03
```

É mostrado o conteúdo de todos os registradores internos da CPU; um modo alternativo para visualizar um único registrador é usar o camando "r" seguido do parâmetro que faz referência ao nome do registrador:

```
-rbx
BX 0000
:
```

Esta instrução mostrará o conteúdo do registrador BX e mudará o indicador do Debug de "-" para ":"

Quando o prompt assim se tornar, significa que é possível, embora não obrigatória, a mudança do valor contido no registrador, bastando digitar o novo valor e pressionar [Enter]. Se você simplesmente pressionar [Enter] o valor antigo se mantém.
Experimente digitar

```
- rf
```

Para visualizar o registrador de *flags.*
2.3.4.Estrutura Assembly.
Nas linhas do código em Linguagem Assembly há duas partes: a primeira é o nome da instrução a ser executada; a segunda, os parâmetros do comando. Por exemplo:

```
add ah,bh
```

Aqui "add" é o comando a ser executado, neste caso uma adição, e "ah" bem como "bh" são os parâmetros.
Por exemplo:

```
mov al, 25
```

No exemplo acima, estamos usando a instrução mov, que significa mover o valor 25 para o registrador al.
O nome das instruções nesta linguagem é constituído de 2, 3 ou 4 letras. Estas instruções são chamadas mnemônicos ou códigos de operação, representando a função que o processador executará.
Às vezes instruções aparecem assim:

```
add al,[170]
```

Os colchetes no segundo parâmetro indica-nos que vamos trabalhar com o conteúdo da célula de memória de número 170, ou seja, com o valor contido no endereço 170 da memória e não com o valor 170, isto é conhecido como "endereçamento direto".

Criando um programa simples em assembly.

Vamos, então, criar um programa para ilustrar o que vimos até agora.
O primeiro passo é iniciar o Debug, o que já vimos como fazer anteriormente. Para montar um programa no Debug, é usado o comando "a" (assemble); quando usamos este comando, podemos especificar um endereço inicial para o nosso programa como o parâmetro, mas é opcional. No caso de omissão, o endereço inicial é o especificado pelos registradores
CS:IP, geralmente 0100h, o local em que programas com extensão .COM devem iniciar. E será este o local que usaremos, uma vez que o Debug só pode criar este tipo de programa.
Embora neste momento não seja necessário darmos um parâmetro ao comando "a", isso é recomendável para evitar problemas, logo:

```
a 100[enter]
mov ax,0002[enter]
mov bx,0004[enter]
add ax,bx[enter]
nop[enter][enter]
```

O que o programa faz? Move o valor 0002 para o registrador ax, move o valor 0004 para o registrador bx, adiciona o conteúdo dos registradores ax e bx, guardando o resultado em ax e finalmente a instrução nop (nenhuma operação) finaliza o programa.
No programa debug, a tela se parecerá com:

```
C:\>debug
-a 100
0D62:0100 mov ax,0002
0D62:0103 mov bx,0004
```

```
0D62:0106 add ax,bx
0D62:0108 nop
0D62:0109
```
Entramos com o comando "t" para executar passo a passo as instruções:
```
-t
AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0D62 ES=0D62 SS=0D62 CS=0D62 IP=0103 NV UP EI PL NZ NA PO NC
0D62:0103 BB0400 MOV BX,0004
```
Vemos o valor 0002 no registrador AX. Teclamos "t" para executar a Segunda instrução:
```
-t
AX=0002 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0D62 ES=0D62 SS=0D62 CS=0D62 IP=0106 NV UP EI PL NZ NA PO NC
0D62:0106 01D8 ADD AX,BX
```
Teclando "t" novamente para ver o resultado da instrução add:
```
-t
AX=0006 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0D62 ES=0D62 SS=0D62 CS=0D62 IP=0108 NV UP EI PL NZ NA PE NC
0D62:0108 90 NOP
```
A possibilidade dos registradores conterem valores diferentes existe, mas AX e BX devem conter os mesmos valores acima descritos.
Para sair do Debug usamos o comando "q" (quit).

Armazenando e carregando os programas.

Não seria prático ter que digitar o programa cada vez que iniciássemos o Debug. Ao invés disso, podemos armazená-lo no disco. Só que o mais interessante nisso é que um simples comando de salvar cria um arquivo com a extensão .COM, ou seja, executável – sem precisarmos efetuar os processos de montagem e ligação, como veremos posteriormente com o TASM.
Eis os passos para salvar um programa que já esteja na memória:
* Obter o tamnho do programa subtraindo o endereço final do
endereço inicial, naturalmente que no sistema hexadecimal.
* Dar um nome ao programa.
* Colocar o tamanho do programa no registrador CX.
* Mandar o debug gravar o programa em disco.

Usando como exemplo o seguinte programa, vamos clarear a idéia de como realizar os passos acima descritos:
```
0C1B:0100 mov ax,0002
0C1B:0103 mov bx,0004
0C1B:0106 add ax,bx
0C1B:0108 int 20
0C1B:010A
```
Para obter o tamanho de um programa, o comando "h" é usado, já que ele nos mostra a adição e subtração de dois números em hexadecimal. Para obter o tamanho do programa em questão, damos como parâmetro o valor do endereço final do nosso programa (10A), e o endereço inicial (100). O primeiro resultado mostra-nos a soma dos endereços, o segundo, a subtração.

```
-h 10a 100
020a 000a
```

O comando "n" permite-nos nomear o programa.

```
-n test.com
```

O comando "rcx" permite-nos mudar o conteúdo do registrador CX para o valor obtido como tamanho do arquivo com o comando "h", neste caso 000a.

```
-rcx
CX 0000
:000a
```

Finalmente, o comando "w" grava nosso programa no disco, indicando quantos bytes gravou.

```
-w
Writing 000A bytes
```

Para já salvar um arquivo quando carregá-lo, 2 passos são necessários:
Dar o nome do arquivo a ser carregado.
Carregá-lo usando o comando "l" (load).
Para obter o resultado correto destes passos, é necessário que o programa acima já esteja criado.
Dentro do Debug, escrevemos o seguinte:

```
-n test.com
-l
-u 100 109
0C3D:0100 B80200 MOV AX,0002
0C3D:0103 BB0400 MOV BX,0004
0C3D:0106 01D8 ADD AX,BX
0C3D:0108 CD20 INT 20
```

O último comando "u" é usado para verificar que o programa foi carregado na memória. O que ele faz é desmontar o código e mostrá-lo em assembly. Os parâmetros indicam ao Debug os endereços inicial e final a serem desmontados.
O Debug sempre carrega os programas na memória no endereço 100h, conforme já comentamos.