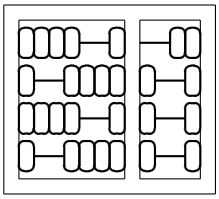

MC404 - Organização de Computadores e Linguagem de Montagem
Instituto de Computação
Universidade Estadual de Campinas

Lab. 01

1 Objetivo

O objetivo deste laboratório é fazer com que o aluno se familiarize com o ciclo de desenvolvimento de programas escritos em linguagem assembly 8086.

2 Preliminares

O ciclo de desenvolvimento de programas em linguagem de montagem geralmente consiste da edição do programa fonte (através de um editor `ASCII`), montagem do programa fonte (através de um montador) e linkedição, gerando o programa executável. A figura 1 mostra esse fluxo de desenvolvimento.

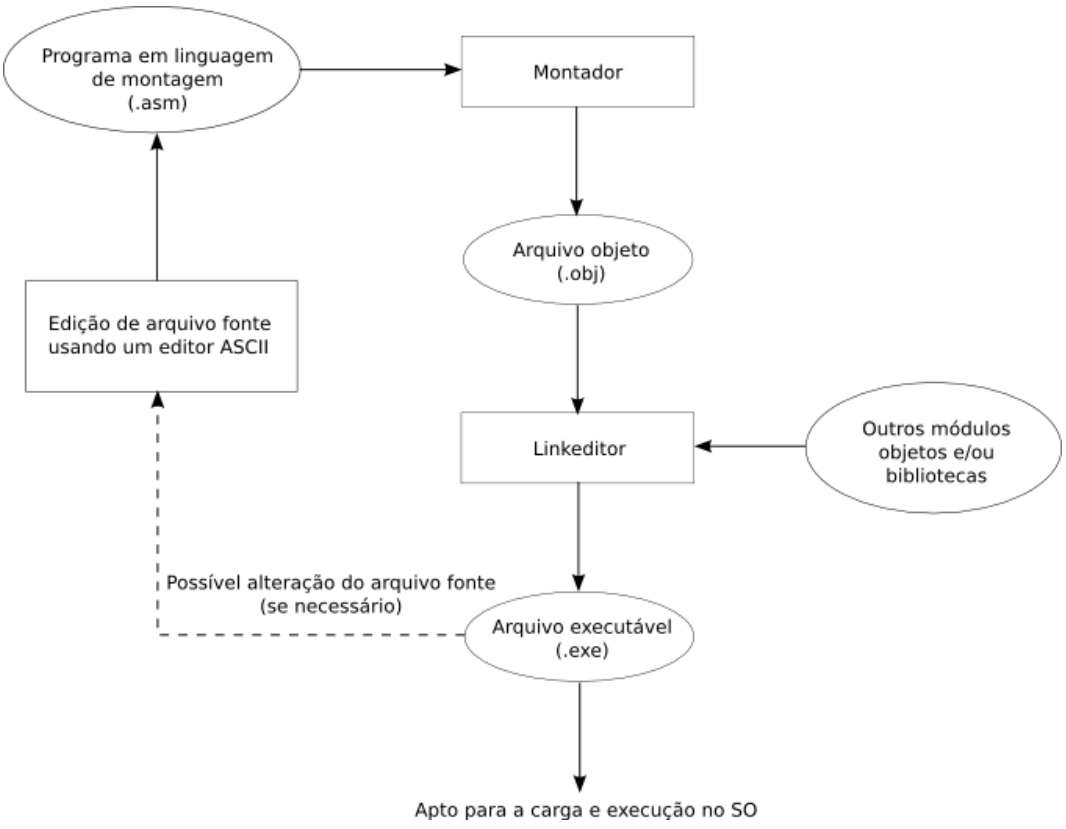

Figura 1: Ciclo de desenvolvimento *Assembly*

Um depurador também pode ser usado durante a fase de testes do programa, possibilitando a verificação, instrução por instrução, de seu comportamento.

3 Ambiente de trabalho

Você pode utilizar tanto o Windows como o Linux para realizar as tarefas do laboratório de MC404. Como vamos trabalhar com o processador 8086 em modo real, teremos que utilizar o sistema operacional DOS. As instruções para entrar no ambiente DOS a partir do Windows ou Linux estão a seguir:

Windows - basta abrir um prompt de comando (`Iniciar -> Executar ->` digite `cmd`)

Linux - você precisa executar um programa que emule o ambiente DOS. Um modo é utilizar o `dosbox`. Verifique se tal ferramenta está instalada em sua máquina abrindo um terminal e digitando `dosbox`. Para montar uma unidade de disco virtual no `dosbox` você deve usar o comando `mount <unidade> <caminho-linux>`. Por exemplo, `mount C /home/aluno` monta na unidade virtual `C` do `dosbox` o diretório `/home/aluno` do linux. Para acessar a unidade execute `<unidade>:` e seu ambiente estará pronto para o laboratório. Qualquer dúvida procure o professor responsável.

Para obter as ferramentas de desenvolvimento acesse a página do curso localizada em `http://www.ic.unicamp.br/~ducatte/mc404`. Copie a pasta `tools` e, se desejar, `docs` para seu ambiente de trabalho. É recomendado a seguinte estrutura de diretório:

```
+ <area-aluno>
  + mc404              <--- diretório do curso
    - tools            <--- ferramentas de desenvolvimento
    + labs             <--- exercícios de laboratório
      - lab01          <--- laboratório de hoje
    - docs             <--- documentos
```

É recomendável ainda ajustar o caminho para as ferramentas através do comando `PATH`:

```
PATH=<area-aluno>\mc404\tools;%PATH%
```

Assim, se sua área estiver em `C:`, o comando fica:

```
PATH=C:\mc404\tools;%PATH%
```

Dentro da pasta `tools` estão as seguintes ferramentas:

- EDIT.COM - editor ASCII
- TASM.EXE - Turbo assembler (montador)
- NASM.EXE - Netwide assembler (montador)
- TLINK.EXE - Turbo linker (ligador)
- TD.EXE - Turbo Debugger (depurador)

Você pode escolher entre os montadores `TASM` e `NASM`. As vantagens deste último é que possui uma documentação eletrônica excelente (veja pasta `docs`), além de contar com versões também para o Linux. Já para obter informações sobre o TASM, você deve consultar o guia do usuário (*Borland Turbo Assembler, User's Guide*), disponível na biblioteca. A diferença entre os montadores está principalmente nas diretivas de montagem disponíveis.

NOTA: Os laboratórios futuros vão utilizar o NASM.

4 Atividade 1

Nesta atividade você vai escrever em assembly um programa que imprime *Hello World*. Não se preocupe em entender o que as instruções do programa fazem, mas sim em utilizar as ferramentas de desenvolvimento.

4.1 Passo 1 – Edição

Abra o editor `EDIT.EXE` (ou algum outro de sua preferência) no Linux ou Windows e escreva o programa apresentado na figura 2 se utilizar o `TASM`, ou o da figura 3, se utilizar o `NASM`.

```
.MODEL small                    ; modelo de memoria do programa

.STACK 100h                     ; tamanho da pilha (em hexa)

.DATA                           ; segmento de dados
  ; define mensagem
  HelloMessage DB 'Hello World', 13, 10, '$'

.CODE                           ; segmento de codigo
start:
  mov ax, @data                 ; faz 'ds' apontar para segmento de dados
  mov ds, ax

  mov ah, 9                     ; ds:dx -> aponta para mensagem
  mov dx, OFFSET HelloMessage
  int 21h                       ; chama servico do DOS para escrita no video

  mov ah, 4Ch                   ; devolve controle para SO
  int 21h
END start                       ; seta ponto de entrada para start
```

Figura 2: *Hello World* em assembly do 8086 (TASM)

```
segment data                    ; segmento de dados
  ; define mensagem
HelloMessage: db 'Hello World', 13, 10, '$'

segment code                    ; segmento de codigo
..start:                        ; ponto de entrada
  mov ax, data                  ; faz 'ds' apontar para segmento de dados
  mov ds, ax

  mov ah, 9                     ; ds:dx -> aponta para mensagem
  mov dx, HelloMessage
  int 0x21                      ; chama servico do DOS para escrita no video

  mov ah, 0x4C                  ; devolve controle para SO
  int 0x21

segment stack stack             ; segmento da pilha
  resb 0x100
```

Figura 3: *Hello World* em assembly do 8086 (NASM)

Após editar o programa salve-o com o nome `hello.asm` na pasta `lab01` e saia do editor.

4.2 Passo 2 – Montagem

Vá até o diretório em que salvou o arquivo `hello.asm` no passo anterior. Para montar o programa com o `TASM`, digite:

```
C:\mc404\labs\lab01>TASM hello
```

Já para montar utilizando o `NASM`, digite:

```
C:\mc404\labs\lab01>NASM -fobj hello.asm
```

Em ambos os casos não deve aparecer nenhuma mensagem de erro. Execute o comando `dir` do DOS e note que o arquivo objeto `hello.obj` foi criado.

4.3 Passo 3 – Linkedição

A linkedição (ou ligação) é feita pelo TLINK (Turbo linker). Digite o comando abaixo:

```
C:\mc404\labs\lab01>TLINK hello
```

Esse passo deve ter criado o arquivo executável `hello.exe` em seu diretório.

4.4 Passo 4 – Execução

Para executar o programa simplesmente digite no prompt:

```
C:\mc404\labs\lab01>hello
```

Uma mensagem com `Hello World` deve ter aparecido!

5 Atividade 2 – Depuração

Nesta atividade será visto a ferramenta de depuração da Borland, Turbo Debugger (TD). Antes, porém, é necessário informar ao ligador que queremos informações extras de depuração no arquivo `.exe`. Assim sendo, volte a executar o `TLINK` usando agora a flag de linha de comando `/v`. Essa flag informa ao `TLINK` que desejamos incorporar ao arquivo `.exe` informacões de depuração.

```
C:\mc404\labs\lab01>TLINK /v hello
```

Agora o arquivo executável `hello.exe` está apto para ser depurado no `TD`. Digite a seguinte linha de comando para executar o Turbo Debugger:

```
C:\mc404\labs\lab01>TD hello
```

As principais janelas do TD estão destacadas na figura 4. Neste primeiro laboratório vamos apenas mostrar os comandos mais simples de depuração.

Uma seta na janela de segmento de código mostra a instrução corrente (que ainda não foi executada). Para caminhar pelo programa, instrução por instrução, use `F7` ou `F8`. A diferença é que

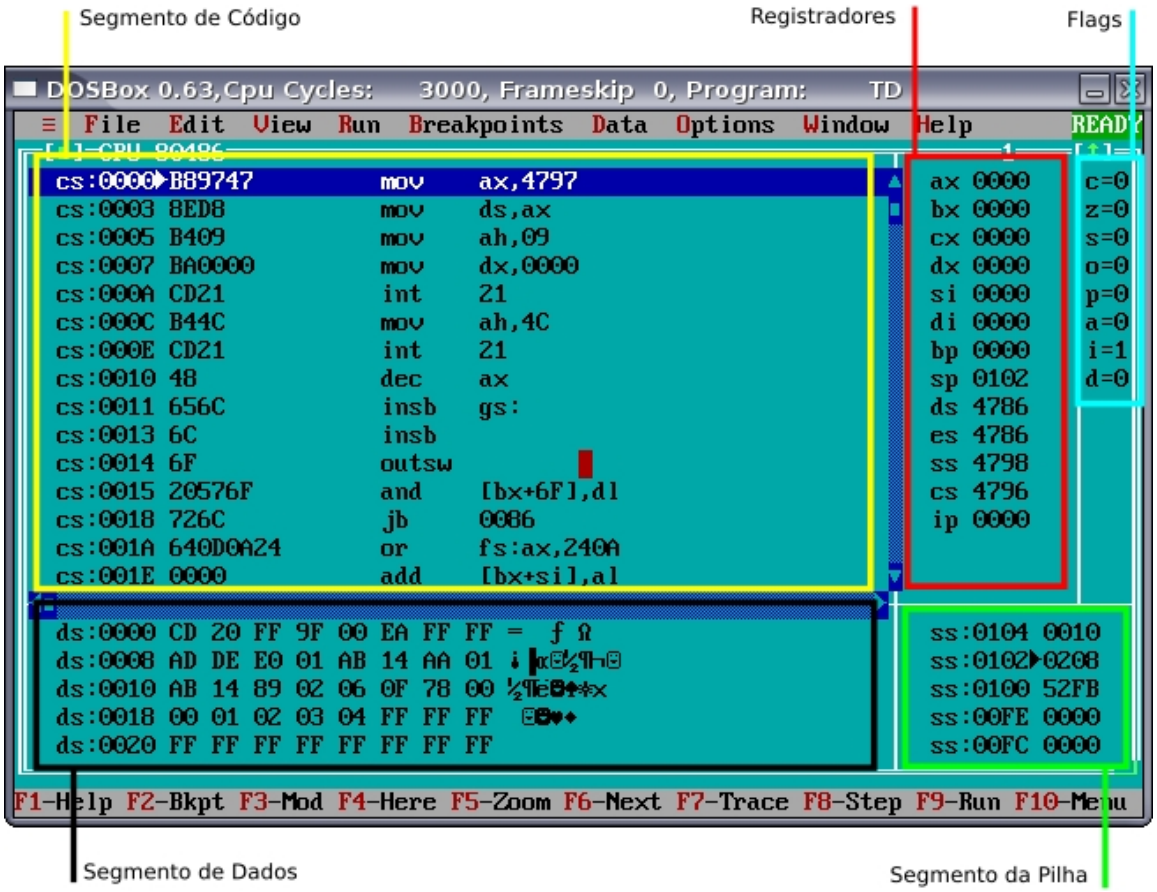

Figura 4: Tela inicial do Turbo Debugger

com `F7` executa-se subrotinas, se houver (*trace into*), passo a passo, enquanto que com `F8` a subrotina é executada inteiramente em um único passo (*step over*). Para o exemplo do laboratório elas têm o mesmo comportamento. Caminhe pelas instruções até o programa terminar com a mensagem *Terminated, exit code xx*.

Para resetar o programa, use `CTRL+F2`. A tecla `F9` executa o programa todo de uma vez só. É possível criar pontos de parada (*breakpoints*) e executar o programa até esses pontos. Para isso, mova com o teclado a linha azul horizontal até alguma instrução e aperte `F2`. Essa linha vai ficar marcada (vermelha). Teclando `F9` o programa será executado até que o primeiro breakpoint seja encontrado (ou o programa termine).

Como exemplo, use `CTRL+F2` e resete o programa. Agora caminhe até a instrução `mov ah,4C`. Em seguida pressione `F2` para marcar um ponto de parada nessa instrução. A figura 5 mostra esta situação.

Agora tecle `F9` e note que todas as instruções desde o começo do programa até o ponto de parada foram executadas. Para desmacar o breakpoint basta usar novamente `F2`. *Breakpoints* são muito úteis quando se deseja investigar uma determinada área do programa (por exemplo, um procedimento) sem perder tempo caminhando por todas as outras instruções.

Para sair do Turbo Debugger utilize a tecla `ALT+X`.

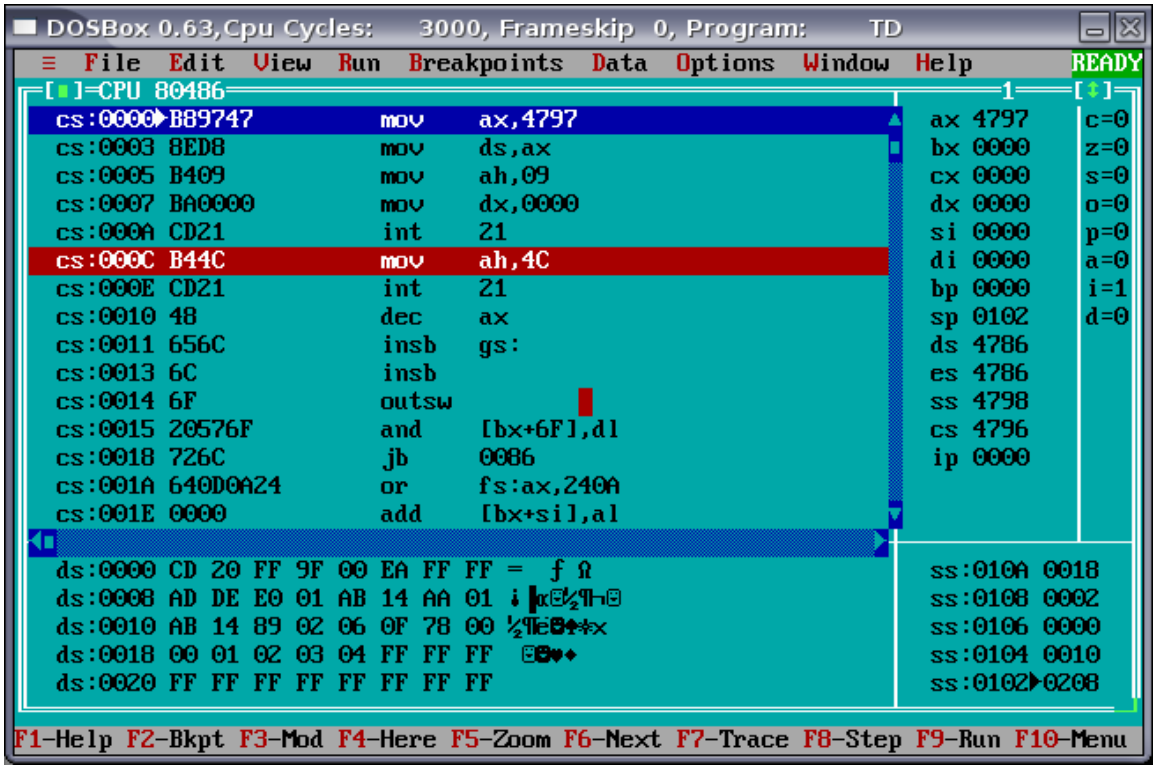

Figura 5: Determinando pontos de parada