Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Engenharia Elétrica
ENG04476-Microprocessadores II

Roteiro de Laboratório 1
Ambiente de Desenvolvimento

Prof. Walter Fetter Lages

25 de fevereiro de 2005

1 Objetivo

O objetivo deste laboratório é familiarizar os alunos com a operação dos diversos utilitários utilizados para programação de computadores, em especial na linguagem Assembly. Ao final, espera-se que os alunos sejam capazes de montar e linkar tanto programas compostos por um único módulo quanto programas compostos por vários módulos.

2 Configuração do Ambiente

O programa MASM deve ser instalado no diretório `C:\ENG04476\MASM`. A variável de ambiente `PATH` deve incluir os diretórios `C:\ENG04476\MASM\BIN` e `C:\ENG04476\MASM\BINB`. A variável de ambiente `INCLUDE` deve incluir o diretório `C:\ENG04476\MASM\INCLUDE`. A variável de ambiente `HELPFILES` deve incluir `C:\ENG04476\MASM\HELP\*.HLP`. A variável de ambiente `ML` deve incluir as opções *default* do montador, ou seja `/c /Cp /W3 /VM`.

Se tudo estiver configurado corretamente, os comandos `ml /?` e `link /?` devem apresentar o resumo dos parâmetros que podem ser passados na linha de comando do montador e do linker, respectivamente. Por outro lado, os comandos `ml /help` e `link /help` devem apresentar um *help* detalhado da utilização do montador e do linker.

Crie um diretório para colocar os seus arquivos. Ao final do laboratório, limpe-o.

Não altere as configurações dos computadores do laboratório. Isto inclui o papel de parede e a criação de ícones e atalhos no *desktop*.

Usuários que colocarem arquivos em local inapropriado ou alterarem as configurações do computador serão penalizados na avaliação do relatório do laboratório.

3 Desenvolvimento dos Experimentos

3.1 Programa em um Único Módulo

O programa abaixo imprime a mensagem `Hello, World!` na tela. Isto é feito através da subrotina imprime. Note que este programa está estruturado para ser montado e linkado como um programa do tipo executável (`.exe`).

```
CSEG            segment         public
                assume          cs:CSEG

inicio:         push            ds
                mov             ax,DSEG
                mov             ds,ax
                assume          ds:DSEG

                mov             dx,offset msg
                call            imprime
                pop             ds
                mov             ax,4c00h
                int             21h

imprime         proc            near
                mov             ah,09
                int             21h
                ret
imprime         endp

CSEG            ends

DSEG            segment         public
msg             db              0ah,0dh
                db              'Hello, World!'
                db              0ah,0dh,'$'
```

```
DSEG            ends

SSEG            segment         stack
                db              256 dup('pilha')
SSEG            ends
                end             inicio
```

1. Edite este programa e coloque em um arquivo denominado, por exemplo, `hello1.asm`. Sugestão: utilize o editor edit do DOS.

2. Monte o programa utilizando o Microsoft Macro-assembler, digitando

   ```
   ML /Fl hello1.lst hello1.asm
   ```

   Note que o parâmetro `/Fl hello.asm` é opcional. Ele faz com que o montador gere um arquivo de listagem do módulo sendo montado. Este arquivo é um arquivo ASCII. Utilize o editor para verificar o conteúdo deste arquivo. Se não houveram erros de digitação, deve ter sido gerado o arquivo `hello1.obj`. Verifique se isto realmente ocorreu. Caso contrário, retorne ao item 2. O parâmetro `/c` faz com que o montador não invoque automaticamente o *linker*. O parâmetro `W3` habilita todos os *warnings*.

3. Linke o arquivo `hello1.obj`, utilizando o comando

   ```
   link /M hello1.obj;
   ```

   O Parâmetro `/M` é opcional. Ele força o linker a gerar uma arquivo de mapa do programa sendo linkado (`hello1.map`). Este arquivo também é um arquivo ASCII, portanto utilize o editor para verificar o conteúdo deste arquivo. Caso não tenham havido erros de linkagem, deve ter sido gerado também um arquivo denominado `hello1.exe`.

4. Execute o programa `hello1.exe` para verificar o seu funcionamento.

3.2 Programas com Vários Módulos

Considere a subrotina `imprime`, utilizada no programa anterior. Esta subrotina é relativamente genérica, ou seja, é bastante provável que outros programas necessitem de uma subrotina semelhante. Imagine agora que esta subrotina seja bastante grande. Neste caso incluir esta rotina em todos os programas que necessitam dela torna-se bastante indesejável, pois além de ser cansativo, aumenta a probabilidade

de introdução erros e também o tempo necessário para montagem do programa. O que se faz nestes casos é colocar a subrotina em um módulo separado, que é montado uma única vez e linkado com todos os programas que necessitam desta subrotina. Em outras palavras, cria-se um arquivo `.asm` com o programa principal e um arquivo `.asm` com a(s) subrotina(s). Ambos são montados separadamente e linkados em único arquivo `.exe`. O mesmo arquivo `.obj` que contém a subrotina pode ser utilizado em qualquer outro programa que necessite desta subrotina, sem a necessidade passar pelo processo de montagem novamente.

Dividindo-se o programa anterior tem-se o seguinte módulo correspondente ao programa principal:

```
                extrn           imprime:near

CSEG            segment         public
                assume          cs:CSEG

inicio:         push            ds
                mov             ax,DSEG
                mov             ds,ax
                assume          ds:DSEG

                mov             dx,offset msg
                call            imprime
                pop             ds
                mov             ax,4c00h
                int             21h

CSEG            ends

DSEG            segment         public
msg             db              0ah,0dh
                db              'Hello, World!'
                db              0ah,0dh,'$'
DSEG            ends

SSEG            segment         stack
                db              256 dup('pilha')
SSEG            ends
                end             inicio
```

O módulo contendo a subrotina `imprime` seria:

```
                public          imprime

CSEG            segment         public

                assume          cs:CSEG

imprime         proc            near
                mov             ah,9
                int             21h
                ret
imprime         endp

CSEG            ends
                end
```

1. Digite o programa principal em um arquivo denominado, por exemplo `hello2.asm`. (Ou copie e modifique o arquivo `hello1.asm`).

2. Digite o módulo com a subrotina em um arquivo denominado, por exemplo `imprime.asm`.

3. Monte cada um dos módulos do programa separadamente, gerando os arquivos `hello2.obj` e `imprime.obj` (e opcionalmente os arquivos `hello2.lst` e `imprime.lst`).

4. Linke ambos os módulos utilizando o comando

   ```
   link /m hello2.obj+imprime.obj;
   ```

   Note que os mesmos comentários feitos no item 3 da seção 2 com relação ao parâmetro `/m` também aplicam-se aqui.

5. Execute o novo programa `hello2.exe` para verificar o seu funcionamento.

6. Compare os arquivos de listagem e de mapa com os gerados na seção 3.1.

3.3 Programas com bibliotecas

Suponha que além da subrotina imprime, já se tenha desenvolvido uma rotina para fazer soar um beep no auto-falante do computador, como mostrado abaixo. Salve

este novo arquivo com o nome `beep.asm`. Note que o nome do arquivo com o código fonte do módulo não tem relação nenhuma com o nome das subrotinas contidas no mesmo. Como os módulos desenvolvidos aqui contém apenas uma rotinas, torna-se conveniente dar para o módulo o mesmo nome da rotina.

```
                public          beep

CSEG            segment         public

                assume          cs:CSEG

beep            proc            near
                mov             ah,0eh
                xor             bh,bh
                mov             al,07h
                int             10h
                ret
beep            endp

CSEG            ends
                end
```

1. Monte o módulo `beep.asm`, gerando o arquivo `beep.obj`.
2. Crie uma biblioteca contendo este módulo, através do comando

   ```
   lib tela.lib +imprime.obj +beep.obj, tela.lst;
   ```

   Note que a vírgula e o arquivo de listagem são opcionais. Este arquivo também está em ASCII. Utilize o editor para verificar o seu conteúdo.
3. Crie e monte o programa `hello3.asm` a partir do programa `hello2.asm`, incluindo no cabeçalho a diretiva `extern beep:near` e incluindo um `call beep` após o `call imprime`.
4. Linke o arquivo `hello3.obj` e o arquivo `tela.lib` para produzir o programa `hello3.exe`. Para isto utilize o comando:

   ```
   link /m hello3.obj, hello3.exe, hello3.map, tela.lib;
   ```
5. Execute o programa `hello3.exe` e verifique o seu funcionamento.
6. Compare os arquivos de listagem e de mapa gerados com os obtidos nas seções 3.1 e 3.2.