Instruções de uso do montador DAEDALUS

(baseadas em texto extraído da monografia apresentada como trabalho de diplomação no curso de Bacharelado em Ciência da Computação por Luís Ricardo Schwengber, sob orientação do Professor Raul Fernando Weber)

Porque usar um montador?

Quando a complexidade do conjunto de instruções de um computador ou o tamanho dos programas aumenta, a correção de erros nos programas escritos em linguagem simbólica fica mais difícil.

No caso do modo de endereçamento direto usado por alguns dos processadores usados em nosso curso, não é possível usar endereços simbólicos. Ou seja, não se pode fazer referência a posições de memória através de nomes. É necessário utilizar os próprios endereços numéricos. Nesses casos, quando for necessário inserir instruções no meio do código para corrigir algum erro de programação, todas as instruções que referenciam os endereços posteriores a esse ponto devem, em geral, serem modificadas.

Por outro lado, com um *montador*, pode-se associar nomes às instruções ou variáveis que são referenciadas pelas instruções. Quando a posição dessas instruções ou variáveis é alterada, o próprio montador encarrega-se de recalcular os endereços e acertar as referências a elas. Além disso, um montador representa uma ferramenta indispensável para se desenvolver programas com maior robustez e confiabilidade, evitando erros que poderiam ocorrer se esse trabalho fosse realizado manualmente.

Além disso, deve-se incorporar a esse montador recursos que tornem mais natural a tarefa de codificação da linguagem simbólica, o que é feito através dos seguintes mecanismos:

- *Rótulos*: possibilidade da declaração de nomes, que são usados para designar tanto valores de variáveis/constantes como instruções. Um rótulo corresponde a uma posição de memória e substitui um endereço, seja este de um operando ou um endereço de desvio.
- *Diretivas de montagem*: também denominadas de pseudo-instruções ou comandos do montador. A maioria das diretivas são usadas para controlar a montagem do programa objeto e não geram código. Esses comandos só atuam durante a montagem, uma vez que durante a execução só podem ser executadas instruções do processador. Outras diretivas são usadas para reservar espaço para variáveis ou definir constantes, atribuindo nomes às mesmas.

Pode haver diferentes implementações de montadores para uma mesma arquitetura. Entretanto, em geral, os mesmos mnemônicos são aceitos por todas as implementações, de modo que a principal diferença entre as implementações está na quantidade e nos tipos de diretivas.

Formato de uma linha de código fonte:

Um programa fonte para o montador **Daedalus** é composto por uma série de linhas de código fonte, cada linha contendo instruções ou diretivas do montador.

Linhas contendo instruções

Uma linha de instrução que contém um mnemônico de uma instrução irá gerar exatamente uma instrução de máquina. Seu formato geral apresenta vários campos. Os colocados entre colchetes no formato abaixo são opcionais:

[RÓTULO:] MNEMÔNICO [OPERANDOS] [;COMENTÁRIO]

- RÓTULO: é um símbolo definido pelo usuário ao qual é atribuído o valor corrente do contador de programa, e que é introduzido na tabela de símbolos do montador. Um rótulo não pode ser redefinido e deve começar na primeira posição da linha (coluna 1) e ser seguido por um caractere dois-pontos (**:**), que não faz parte do rótulo. Para criar rótulos veja as regras de formação de símbolos mais adiante.
- MNEMÔNICO: identifica uma instrução de uma das máquinas descritas (Ahmes, Ramses ou Cesar) e corresponde a uma única instrução executável. Deve ser colocado após o "**:**" que segue o rótulo, separado deste por pelo menos um espaço. Se não houver rótulo na linha, o mnemônico deve a direita da coluna 1.
- OPERANDOS: pode conter zero ou mais operandos separados por delimitadores. A quantidade de operandos depende da instrução especificada pelo mnemônico. Os operandos devem ser separados do mnemônico por pelo menos um espaço.
- COMENTÁRIOS: deve começar com um ponto-e-vírgula (**;**) e termina com o caractere de nova linha. Ele deve ser separado do campo a esquerda (operandos ou mnenônico) por pelo menos um espaço.

Linhas contendo diretivas

Uma linha de diretiva também contém vários campos opcionais:

[NOME:] DIRETIVA [OPERANDOS] [;COMENTÁRIO]

Diferente de um campo de mnemônico, uma diretiva (pseudo-instrução) desenvolve alguma função durante o processo de montagem, **não produzindo qualquer código executável**, mas pode reservar e inicializar espaço de dados do programa. Para criar os nomes usados nas diretivas, veja as regras de formação de símbolos, adiante.

Dentre as várias diretivas implementadas encontram-se as seguintes:

Diretiva:	**DB**
Função:	reserva um byte da memória, permitindo opcionalmente inicializá-lo com um valor decimal ou hexadecimal.
Formato:	[nome] DB valorInicial

Diretiva:	**DW**
Função:	reserva dois bytes da memória (uma palavra do processador César = 16 bits), permitindo opcionalmente inicializá-los com um valor decimal ou hexadecimal.
Formato:	[nome] DW valorInicial

Diretiva:	**DAB**
Função:	reserva um *array* de bytes da memória. Tem duas formas: com inicialização individual ou com a definição do tamanho do *array*. No caso da inicialização individual pode-se indicar uma série de valores decimais ou hexadecimais. Para a definição do tamanho deve-se especificar um número entre colchetes, o que reservará um *array* onde todos os seus elementos serão inicializados com "0" (zero).
Formato:	[nome-variável] DAB valor1,valor2,...,valorN ou [nome-variável] DAB [númeroDeBytes]

Diretiva:	**DAW**
Função:	reserva um array de palavras (16 bits) da memória. Tem duas formas: com inicialização individual ou com a definição do tamanho do array. No caso da inicialização individual pode-se indicar uma série de valores decimais ou hexadecimais. Para a definição do tamanho deve-se especificar um número entre colchetes, o que reservará um array onde todos os seus elementos serão inicializados com "0" (zero). Notar que, independentemente da máquina utilizada o montador gerará as palavras usando-se a notação *big endian*.
Formato:	[nome-variável] DAW valor1,valor2,...,valorN ou [nome-variável] DAW [númeroDeWord's]

Diretiva:	**ORG**
Função:	Altera o valor do apontador de posição. Após encontrar uma diretiva ORG, o montador passa a colocar o código gerado a partir do endereço especificado pelo ORG. Essa diretiva é útil, por exemplo, para separar a área de códigos da área de variáveis. O montador toma o cuidado de emitir um aviso (Warning) sobre o fato de se estar ultrapassando o tamanho máximo da memória existente na máquina específica.
Formato:	ORG posiçãoMemória

Símbolos e Constantes

Um símbolo (usado como rótulo de uma instrução ou nome nas diretivas que geram áreas de dados, bem como para referenciar as mesmas nas instruções) pode conter até 1024 caracteres, mas é recomendável utilizar poucos para não comprometer a performance de gerência de strings. As regras para a definição de um símbolo são:

- O primeiro caractere não pode ser numérico;
- Os outros caracteres podem ser a-z, A-Z, 0-9 e sublinha (_);

Constantes são usadas na codificação de instruções e na definição de valores em diretivas. Uma constante pode ser representada por um valor decimal, escrito em sua forma natural ou então por um valor hexadecimal (cujo primeiro dígito deve ser numérico),

representado com o valor precedido pela letra H. Nas diretivas DB, DW, DAB e DAW podem adicionalmente ser utilizados caracteres ASCII individuais (entre apóstrofes, como em 'A').

Exemplos:

```
MOV #15,R0
LDR A #15
MOV #H0ABC,2002
DB 5
DB H1A
DW 0
```

```
DW H2A1
DAB 10, H15, 5, H0D
DAW 65000, H0FFF
DB 'A'
DB 'A','B','C'
DW 'A'
```

Implementação

A interface básica oferece a possibilidade de edição de arquivos fonte que podem ser carregados de arquivos gravados anteriormente pelo montador ou escritos desde seu início. Para obter-se o mapa de memória gerado para uma máquina em específico basta escolher uma das opções disponíveis e clicar no botão para executar a montagem.

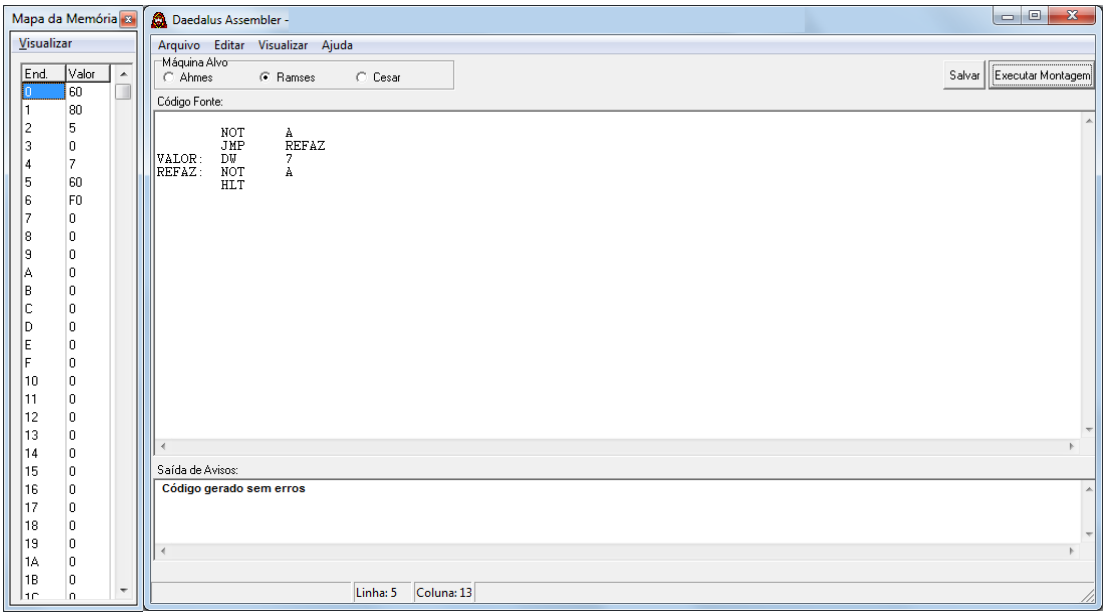

Figura 3.1 - Edição e montagem de um arquivo fonte

Na parte inferior da janela de edição existe um campo de saída, onde o programa exibe avisos eventualmente gerados durante o processo de montagem. Ele é muito útil na verificação de erros no texto, em tempo de programação.

É possível visualizar o mapa de memória gerado através do menu “Visualizar” do montador, sendo que se pode escolher valores decimais ou hexa decimais para sua apresentação. A visualização do mapa de memória da máquina hipotética CESAR, por ser de 64kbytes, pode ser um tanto demorada, mas é feita apenas após um processo de montagem.

Após a montagem é gerado um arquivo “.mem”, que contém uma “imagem de memória”, que pode ser carregado nos simuladores correspondentes, sem a necessidade de configurações adicionais.

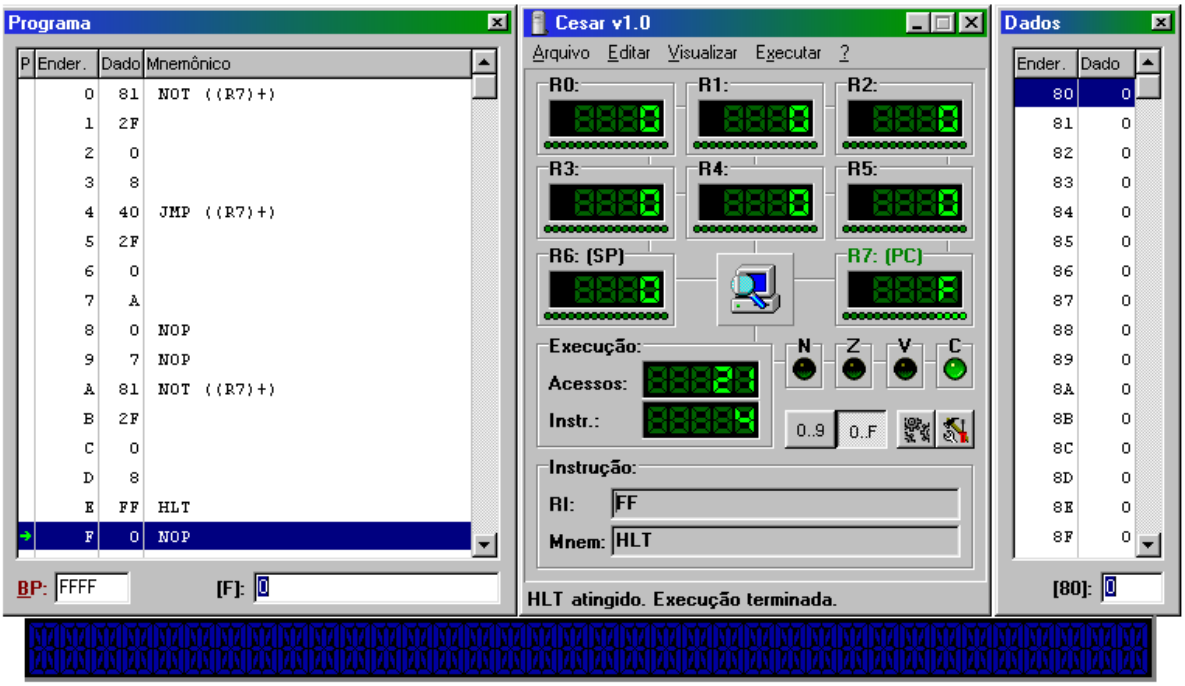

Figura 3.2 - Carga do mapa de memória gerado num dos simuladores disponíveis

Listagem da tabela de símbolos

O montador, além de gerar a imagem de memória para a carga nos simuladores correspondentes, também gera um arquivo “.map”, onde é listada a tabela de símbolos e seus respectivos valores, tanto em decimal quanto em hexa decimal.

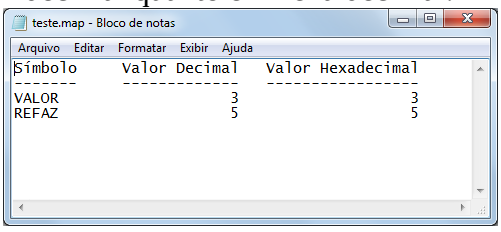

Figura 3.3 - Listagem da Tabela de Símbolos