Montadores, macropocessadores, carregadores e ligadores

Programação Aplicada a Ciência da Computação.

Prof. Dr. Eduardo S. Pereira.

http://eduardopereira.upcursosetreinamentosonline.com/

5 de março de 2018

Introdução

Compilador

- Montadores, macropocessadores, carregadores e ligadores

Introdução

Introdução

- O resultado do processo de compilação é um arquivo contendo um programa em assembly equivalente ao original escrito em linguagem de alto nível
- O montador é o programa do sistema responsável por traduzir código assembly em linguagem de máquina, traduzindo cada instrução do programa para a sequência de bits que codifica a instrução da máquina;
- A ligação, que resolve as referências que tenham sido feitas a dados e rotinas em outros programas;
- Carregamento, que transfere o programa montado para a memória principal e da início à sua execução.

Introdução

Introdução

- O resultado do processo de compilação é um arquivo contendo um programa em assembly equivalente ao original escrito em linguagem de alto nível
- O montador é o programa do sistema responsável por traduzir código assembly em linguagem de máquina, traduzindo cada instrução do programa para a sequência de bits que codifica a instrução da máquina;
- A ligação, que resolve as referências que tenham sido feitas a dados e rotinas em outros programas;
- Carregamento, que transfere o programa montado para a memória principal e da início à sua execução.

Introdução

Introdução

- O resultado do processo de compilação é um arquivo contendo um programa em assembly equivalente ao original escrito em linguagem de alto nível
- O montador é o programa do sistema responsável por traduzir código assembly em linguagem de máquina, traduzindo cada instrução do programa para a sequência de bits que codifica a instrução da máquina;
- A ligação, que resolve as referências que tenham sido feitas a dados e rotinas em outros programas;
- Carregamento, que transfere o programa montado para a memória principal e da início à sua execução.

Introdução

Introdução

- O resultado do processo de compilação é um arquivo contendo um programa em assembly equivalente ao original escrito em linguagem de alto nível
- O montador é o programa do sistema responsável por traduzir código assembly em linguagem de máquina, traduzindo cada instrução do programa para a sequência de bits que codifica a instrução da máquina;
- A ligação, que resolve as referências que tenham sido feitas a dados e rotinas em outros programas;
- Carregamento, que transfere o programa montado para a memória principal e da início à sua execução.

Montadores

Montadores

- Recebe como entrada um arquivo de texto contendo o código fonte do programa assembly e gera como saída um arquivo binário, o módulo objeto.
- O módulo objeto contém o código de máquina e outras informações relevantes para a execução do código gerado.
- um programa fonte para o montador pode conter diretivas ou pseudo-instruções definidas para o montador, como macro-instruções;
- Macro-instruções: uma sequência de instruções que será inserida no código ao ser referenciada pelo nome.

Montadores

Montadores

- Recebe como entrada um arquivo de texto contendo o código fonte do programa assembly e gera como saída um arquivo binário, o módulo objeto.
- O módulo objeto contém o código de máquina e outras informações relevantes para a execução do código gerado.
- um programa fonte para o montador pode conter diretivas ou pseudo-instruções definidas para o montador, como macro-instruções;
- Macro-instruções: uma sequência de instruções que será inserida no código ao ser referenciada pelo nome.

Montadores

Montadores

- Recebe como entrada um arquivo de texto contendo o código fonte do programa assembly e gera como saída um arquivo binário, o módulo objeto.
- O módulo objeto contém o código de máquina e outras informações relevantes para a execução do código gerado.
- um programa fonte para o montador pode conter diretivas ou pseudo-instruções definidas para o montador, como macro-instruções;
- Macro-instruções: uma sequência de instruções que será inserida no código ao ser referenciada pelo nome.

Montadores

Montadores

- Recebe como entrada um arquivo de texto contendo o código fonte do programa assembly e gera como saída um arquivo binário, o módulo objeto.
- O módulo objeto contém o código de máquina e outras informações relevantes para a execução do código gerado.
- um programa fonte para o montador pode conter diretivas ou pseudo-instruções definidas para o montador, como macro-instruções;
- Macro-instruções: uma sequência de instruções que será inserida no código ao ser referenciada pelo nome.

Montadores

Montadores

- Um montador que suporte a definição e utilização de macro-instruções é usualmente denominado um macro-montador (macro-assembler).
- Um montador multiplataforma (cross-assembler) é um montador que permite gerar código para um processador-alvo diferente daquele no qual o montador está sendo executado.

Montadores

Montadores

- Um montador que suporte a definição e utilização de macro-instruções é usualmente denominado um macro-montador (macro-assembler).
- Um montador multiplataforma (cross-assembler) é um montador que permite gerar código para um processador-alvo diferente daquele no qual o montador está sendo executado.

Estrutura de programa assembly

Estrutura de programa assembly

- Um programa assembly é tipicamente composto por pelo menos dois segmentos:

1: segmento de dados que define o espaço associado ao armazenamento das variáveis e constantes usadas pelo programa

2: segmento de instruções, onde o código do programa é armazenado

- Além dessas duas seções, um programa-fonte assembly pode conter uma seção de definições, usadas na descrição dos programas e que não produzem nenhum efeito no código gerado.

Estrutura de programa assembly

Estrutura de programa assembly

- Um programa assembly é tipicamente composto por pelo menos dois segmentos:

1: segmento de dados que define o espaço associado ao armazenamento das variáveis e constantes usadas pelo programa

2: segmento de instruções, onde o código do programa é armazenado

- Além dessas duas seções, um programa-fonte assembly pode conter uma seção de definições, usadas na descrição dos programas e que não produzem nenhum efeito no código gerado.

Estrutura de programa assembly

Estrutura de programa assembly

- Um programa assembly é tipicamente composto por pelo menos dois segmentos:

1: segmento de dados que define o espaço associado ao armazenamento das variáveis e constantes usadas pelo programa

2: segmento de instruções, onde o código do programa é armazenado

- Além dessas duas seções, um programa-fonte assembly pode conter uma seção de definições, usadas na descrição dos programas e que não produzem nenhum efeito no código gerado.

Estrutura de programa assembly

Estrutura de programa assembly

- Um programa assembly é tipicamente composto por pelo menos dois segmentos:

1: segmento de dados que define o espaço associado ao armazenamento das variáveis e constantes usadas pelo programa

2: segmento de instruções, onde o código do programa é armazenado

- Além dessas duas seções, um programa-fonte assembly pode conter uma seção de definições, usadas na descrição dos programas e que não produzem nenhum efeito no código gerado.

Estrutura de programa assembly

Estrutura de programa assembly

- Por conveniência da leitura do código fonte, a seção de definições é tradicionalmente alocada ao início do código.
- Para o montador, o posicionamento dos diferentes trechos de programa no código fonte deve ser irrelevante.

Estrutura de programa assembly

Estrutura de programa assembly

- Por conveniência da leitura do código fonte, a seção de definições é tradicionalmente alocada ao início do código.
- Para o montador, o posicionamento dos diferentes trechos de programa no código fonte deve ser irrelevante.

Estrutura de programa assembly

- Considere o código:

```
1       START   ADD.L   D0,D1
2               JMP     NEXT
3       LOOP    ADD.L   #1,D1
4       NEXT    CLR.L   D5
5               JMP     LOOP
```

- Na linha 2 desse trecho de programa há uma referência a um símbolo, NEXT, cujo valor ainda não havia sido determinado – essa definição só acontecerá na linha 4.

Estrutura de programa assembly

- Considere o código:

```
1       START   ADD.L   D0,D1
2               JMP     NEXT
3       LOOP    ADD.L   #1,D1
4       NEXT    CLR.L   D5
5               JMP     LOOP
```

- Na linha 2 desse trecho de programa há uma referência a um símbolo, NEXT, cujo valor ainda não havia sido determinado – essa definição só acontecerá na linha 4.

Estrutura de programa assembly

- Considere o código:

```
1       START   ADD.L   D0,D1
2               JMP     NEXT
3       LOOP    ADD.L   #1,D1
4       NEXT    CLR.L   D5
5               JMP     LOOP
```

- Há duas possibilidades de lidar com essas referências futuras.

Estrutura de programa assembly

Estrutura de programa assembly

- deixar uma lacuna reservada no código gerado associada ao operando da instrução da linha 2. Posteriormente, quando houvesse uma definição desse valor – provavelmente quando o fim do arquivo com o código fonte fosse alcançado – essa lacuna seria preenchida.
- Neste caso, seria possível gerar o código de máquina realizando um único passo (uma única leitura) sobre o arquivo.
- Entretanto, haveria um maior custo na complexidade de implementação do montador, que deveria manter referências a todas as lacunas que devem ser preenchidas ao final da montagem.

Estrutura de programa assembly

Estrutura de programa assembly

- deixar uma lacuna reservada no código gerado associada ao operando da instrução da linha 2. Posteriormente, quando houvesse uma definição desse valor – provavelmente quando o fim do arquivo com o código fonte fosse alcançado – essa lacuna seria preenchida.
- Neste caso, seria possível gerar o código de máquina realizando um único passo (uma única leitura) sobre o arquivo.
- Entretanto, haveria um maior custo na complexidade de implementação do montador, que deveria manter referências a todas as lacunas que devem ser preenchidas ao final da montagem.

Estrutura de programa assembly

Estrutura de programa assembly

- deixar uma lacuna reservada no código gerado associada ao operando da instrução da linha 2. Posteriormente, quando houvesse uma definição desse valor – provavelmente quando o fim do arquivo com o código fonte fosse alcançado – essa lacuna seria preenchida.
- Neste caso, seria possível gerar o código de máquina realizando um único passo (uma única leitura) sobre o arquivo.
- Entretanto, haveria um maior custo na complexidade de implementação do montador, que deveria manter referências a todas as lacunas que devem ser preenchidas ao final da montagem.

Estrutura de programa assembly

Estrutura de programa assembly

- realizar o processo montagem em dois passos.
- O primeiro passo simplesmente lê o arquivo com o objetivo de criar a Tabela de Símbolos, ou seja, obter os valores associados a todas as constantes simbólicas definidas no programa.
- No segundo passo, uma nova leitura sobre o arquivo é realizada para gerar o código de máquina; nesse passo, a informação da tabela de símbolos criada no primeiro passo é utilizada.

Estrutura de programa assembly

Estrutura de programa assembly

- realizar o processo montagem em dois passos.
- O primeiro passo simplesmente lê o arquivo com o objetivo de criar a Tabela de Símbolos, ou seja, obter os valores associados a todas as constantes simbólicas definidas no programa.
- No segundo passo, uma nova leitura sobre o arquivo é realizada para gerar o código de máquina; nesse passo, a informação da tabela de símbolos criada no primeiro passo é utilizada.

Estrutura de programa assembly

Estrutura de programa assembly

- realizar o processo montagem em dois passos.
- O primeiro passo simplesmente lê o arquivo com o objetivo de criar a Tabela de Símbolos, ou seja, obter os valores associados a todas as constantes simbólicas definidas no programa.
- No segundo passo, uma nova leitura sobre o arquivo é realizada para gerar o código de máquina; nesse passo, a informação da tabela de símbolos criada no primeiro passo é utilizada.

Pseudo-Instruções

Pseudo-Instruções

- Além das instruções do processador, um programa assembly preparado para um montador também pode conter pseudo-instruções que estabelecem a conexão entre referências simbólicas e valores a serem efetivamente referenciados.
- Cada montador pode oferecer um conjunto de pseudo-instruções diferenciado.

Pseudo-Instruções

Pseudo-Instruções

- Além das instruções do processador, um programa assembly preparado para um montador também pode conter pseudo-instruções que estabelecem a conexão entre referências simbólicas e valores a serem efetivamente referenciados.
- Cada montador pode oferecer um conjunto de pseudo-instruções diferenciado.

Pseudo-Instruções

Pseudo-Instruções

- substituição simbólica, EQU, associa um valor definido pelo programador a um símbolo.
- SIZE EQU 100
- associa o valor decimal 100 ao símbolo SIZE, que pode ser posteriormente referenciado em outras instruções, como em: MOVE #SIZE,D0

Pseudo-Instruções

Pseudo-Instruções

- substituição simbólica, EQU, associa um valor definido pelo programador a um símbolo.
- SIZE EQU 100
- associa o valor decimal 100 ao símbolo SIZE, que pode ser posteriormente referenciado em outras instruções, como em: MOVE #SIZE,D0

Pseudo-Instruções

Pseudo-Instruções

- substituição simbólica, EQU, associa um valor definido pelo programador a um símbolo.
- SIZE EQU 100
- associa o valor decimal 100 ao símbolo SIZE, que pode ser posteriormente referenciado em outras instruções, como em: MOVE #SIZE,D0

Macro-instruções

Macro-instruções

- Uma macro-instrução é um sinônimo para um grupo de instruções que pode ser usado como uma instrução ao longo do código-fonte.
- O uso de macros facilita a especificação de trechos repetitivos de código, que podem ser invocados pelo programador como um única linha no programa.
- Por esse motivo, diversos montadores apresentam extensões com funcionalidades para a definição e utilização de macros.

Macro-instruções

Macro-instruções

- Uma macro-instrução é um sinônimo para um grupo de instruções que pode ser usado como uma instrução ao longo do código-fonte.
- O uso de macros facilita a especificação de trechos repetitivos de código, que podem ser invocados pelo programador como um única linha no programa.
- Por esse motivo, diversos montadores apresentam extensões com funcionalidades para a definição e utilização de macros.

Macro-instruções

Macro-instruções

- Uma macro-instrução é um sinônimo para um grupo de instruções que pode ser usado como uma instrução ao longo do código-fonte.
- O uso de macros facilita a especificação de trechos repetitivos de código, que podem ser invocados pelo programador como um única linha no programa.
- Por esse motivo, diversos montadores apresentam extensões com funcionalidades para a definição e utilização de macros.

Macro-instruções

Macro-instruções

- Na sua forma mais simples, uma macro é simplesmente uma abreviatura para um grupo de instruções:
- nome MACRO [argumentos]
  corpo
  ENDM

Macro-instruções

Macro-instruções

- Na sua forma mais simples, uma macro é simplesmente uma abreviatura para um grupo de instruções:
- nome MACRO [argumentos]
  corpo
  ENDM

Montagem

Montagem

- O processo de montagem de um código assembly pode apresentar pequenas diferenças em função das opções adotadas no projeto do montador, mas em linhas gerais as funcionalidades a seguir são suportadas.
- Uma etapa inicial que pode ser suportada é o pré-processamento do código, onde informação não relevante pode ser eliminada.
- Por exemplo, o pré-processador do montador as elimina comentários e converte constantes em formato caráter para as correspondentes constantes em valores numéricos.
- Na sequência, o montador realiza o pré-processamento de macros, obtendo um código pronto para a criação do módulo objeto.

Montagem

Montagem

- O processo de montagem de um código assembly pode apresentar pequenas diferenças em função das opções adotadas no projeto do montador, mas em linhas gerais as funcionalidades a seguir são suportadas.
- Uma etapa inicial que pode ser suportada é o pré-processamento do código, onde informação não relevante pode ser eliminada.
- Por exemplo, o pré-processador do montador as elimina comentários e converte constantes em formato caráter para as correspondentes constantes em valores numéricos.
- Na sequência, o montador realiza o pré-processamento de macros, obtendo um código pronto para a criação do módulo objeto.

Montagem

Montagem

- O processo de montagem de um código assembly pode apresentar pequenas diferenças em função das opções adotadas no projeto do montador, mas em linhas gerais as funcionalidades a seguir são suportadas.
- Uma etapa inicial que pode ser suportada é o pré-processamento do código, onde informação não relevante pode ser eliminada.
- Por exemplo, o pré-processador do montador as elimina comentários e converte constantes em formato caráter para as correspondentes constantes em valores numéricos.
- Na sequência, o montador realiza o pré-processamento de macros, obtendo um código pronto para a criação do módulo objeto.

Montagem

Montagem

- O processo de montagem de um código assembly pode apresentar pequenas diferenças em função das opções adotadas no projeto do montador, mas em linhas gerais as funcionalidades a seguir são suportadas.
- Uma etapa inicial que pode ser suportada é o pré-processamento do código, onde informação não relevante pode ser eliminada.
- Por exemplo, o pré-processador do montador as elimina comentários e converte constantes em formato caráter para as correspondentes constantes em valores numéricos.
- Na sequência, o montador realiza o pré-processamento de macros, obtendo um código pronto para a criação do módulo objeto.

Montagem

manipulação de arquivos

- O montador estará recebendo como entrada um arquivo em formato texto, do qual ele deverá ler cada linha para fazer o processamento que for necessário.
- Assim, um dos primeiros grupos de funcionalidades que se faz necessário é a manipulação de arquivos.
- Uma vez obtida a linha do arquivo, a tarefa de extrair de cada linha o campo de interesse estará representada através dos seguintes procedimentos, todos recebendo como argumento uma referência para a linha a ser processada.

Montagem

manipulação de arquivos

- O montador estará recebendo como entrada um arquivo em formato texto, do qual ele deverá ler cada linha para fazer o processamento que for necessário.
- Assim, um dos primeiros grupos de funcionalidades que se faz necessário é a manipulação de arquivos.
- Uma vez obtida a linha do arquivo, a tarefa de extrair de cada linha o campo de interesse estará representada através dos seguintes procedimentos, todos recebendo como argumento uma referência para a linha a ser processada.

Montagem

manipulação de arquivos

- O montador estará recebendo como entrada um arquivo em formato texto, do qual ele deverá ler cada linha para fazer o processamento que for necessário.
- Assim, um dos primeiros grupos de funcionalidades que se faz necessário é a manipulação de arquivos.
- Uma vez obtida a linha do arquivo, a tarefa de extrair de cada linha o campo de interesse estará representada através dos seguintes procedimentos, todos recebendo como argumento uma referência para a linha a ser processada.

Montagem

Processamento de macro-instruções

- um processador de macro deve realizar quatro tarefas básicas
  1- Reconhecer as definições de macros;
  2- Salvar as definições de macros de forma possibilitar a posterior expansão;
  3- Reconhecer as invocações a macros;
  4- Expandir as invocações, possivelmente substituindo argumentos e verificando condições.

Montagem

Processamento de macro-instruções

- um processador de macro deve realizar quatro tarefas básicas
  1- Reconhecer as definições de macros;
  2- Salvar as definições de macros de forma possibilitar a posterior expansão;
  3- Reconhecer as invocações a macros;
  4- Expandir as invocações, possivelmente substituindo argumentos e verificando condições.

Montagem

Processamento de macro-instruções

- um processador de macro deve realizar quatro tarefas básicas
  1- Reconhecer as definições de macros;
  2- Salvar as definições de macros de forma possibilitar a posterior expansão;
  3- Reconhecer as invocações a macros;
  4- Expandir as invocações, possivelmente substituindo argumentos e verificando condições.

Montagem

Processamento de macro-instruções

- um processador de macro deve realizar quatro tarefas básicas
  1- Reconhecer as definições de macros;
  2- Salvar as definições de macros de forma possibilitar a posterior expansão;
  3- Reconhecer as invocações a macros;
  4- Expandir as invocações, possivelmente substituindo argumentos e verificando condições.

Montagem

Processamento de macro-instruções

- um processador de macro deve realizar quatro tarefas básicas
  1- Reconhecer as definições de macros;
  2- Salvar as definições de macros de forma possibilitar a posterior expansão;
  3- Reconhecer as invocações a macros;
  4- Expandir as invocações, possivelmente substituindo argumentos e verificando condições.

Montagem

Processamento de macro-instruções

- O processador de macros pode ser visto como um programa independente do montador, que é invocado antes do processo de montagem propriamente dito. Sua implementação mais simples pode ser realizada em dois passos.

Montagem

Processamento de macro-instruções

- No primeiro passo, cada linha do arquivo com o programa fonte (já sem comentários) é lida. Caso contenha na coluna do campo de operação a pseudo-instrução MACRO, então o que se segue é uma definição de macro, que deve ser armazenada.
- Uma estrutura de dados, a Tabela de Definição de Macro, é usada para guardar essas definições. A chave nessa tabela é o nome da macro, definido no campo de rótulo dessa mesma linha.

Montagem

Processamento de macro-instruções

- No primeiro passo, cada linha do arquivo com o programa fonte (já sem comentários) é lida. Caso contenha na coluna do campo de operação a pseudo-instrução MACRO, então o que se segue é uma definição de macro, que deve ser armazenada.
- Uma estrutura de dados, a Tabela de Definição de Macro, é usada para guardar essas definições. A chave nessa tabela é o nome da macro, definido no campo de rótulo dessa mesma linha.

Montagem

Processamento de macro-instruções

- Associado a cada nome de macro-instrução, a tabela de definição de macro contém dois valores.
- Um valor é o corpo da definição da macro
- o outro a sua lista de parâmetros formais.

Montagem

Processamento de macro-instruções

- Associado a cada nome de macro-instrução, a tabela de definição de macro contém dois valores.
- Um valor é o corpo da definição da macro
- o outro a sua lista de parâmetros formais.

Montagem

Processamento de macro-instruções

- Associado a cada nome de macro-instrução, a tabela de definição de macro contém dois valores.
- Um valor é o corpo da definição da macro
- o outro a sua lista de parâmetros formais.

Montagem

Processamento de macro-instruções

- Para obter a lista de parâmetros formais, o processador de macro verifica se essa lista está presente no campo de operandos da linha.
- Se estiver, cada membro da lista será associado a uma entrada em outra estrutura de dados auxiliar, a Tabela da Lista de Argumentos, que será referenciada na tabela de definição de macro

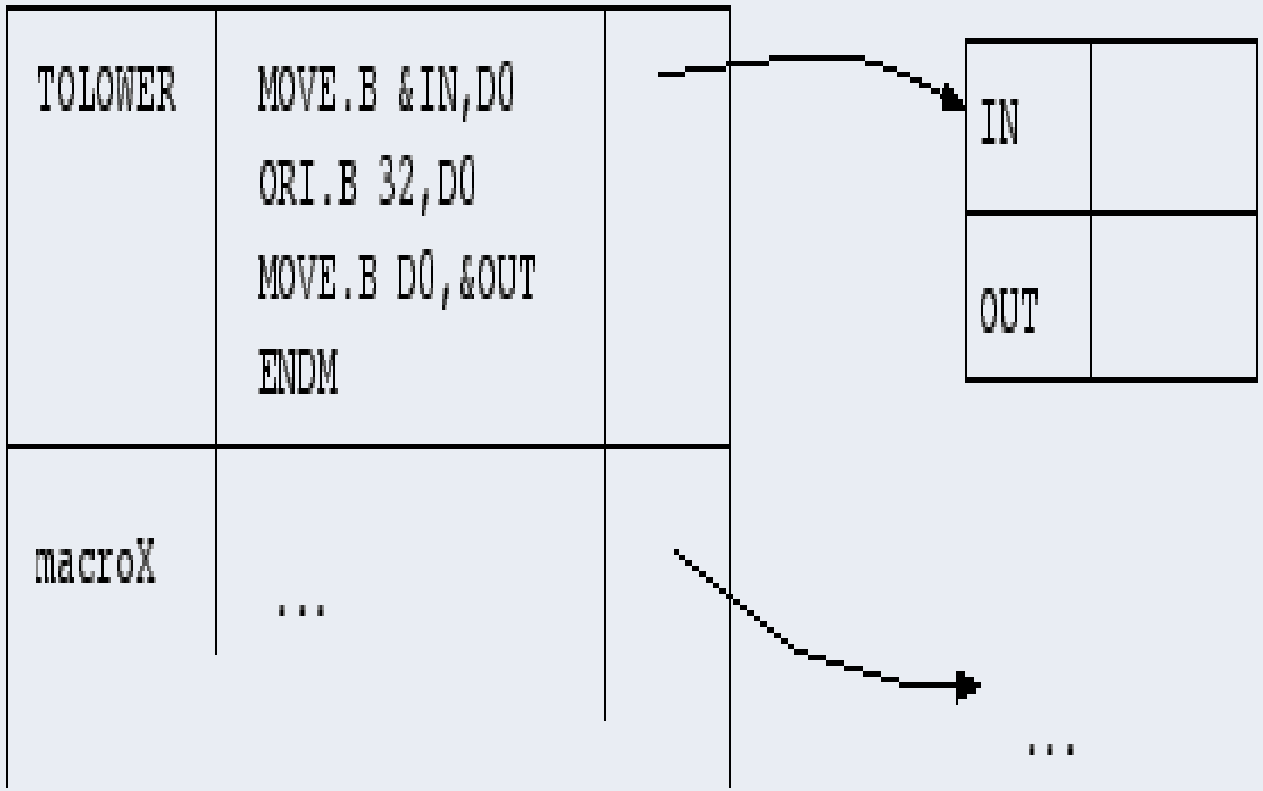
Montagem

Processamento de macro-instruções

- Para obter a lista de parâmetros formais, o processador de macro verifica se essa lista está presente no campo de operandos da linha.
- Se estiver, cada membro da lista será associado a uma entrada em outra estrutura de dados auxiliar, a Tabela da Lista de Argumentos, que será referenciada na tabela de definição de macro

Montagem

Processamento de macro-instruções



Montagem

Processamento de macro-instruções

- Para armazenar o corpo da macro, a linha a seguir é lida e copiada literalmente para a tabela.
- Verifica-se então se o campo de operação da linha copiada era ENDM; se sim, então a definição dessa macro é concluída.
- Caso contrário, o procedimento repete para a linha seguinte.

Montagem

Processamento de macro-instruções

- Para armazenar o corpo da macro, a linha a seguir é lida e copiada literalmente para a tabela.
- Verifica-se então se o campo de operação da linha copiada era ENDM; se sim, então a definição dessa macro é concluída.
- Caso contrário, o procedimento repete para a linha seguinte.

Montagem

Processamento de macro-instruções

- Para armazenar o corpo da macro, a linha a seguir é lida e copiada literalmente para a tabela.
- Verifica-se então se o campo de operação da linha copiada era ENDM; se sim, então a definição dessa macro é concluída.
- Caso contrário, o procedimento repete para a linha seguinte.

Montagem

Processamento de macro-instruções

- O primeiro passo do processador de macro é encerrado quando a pseudo-instrução END é encontrada, sinalizando o fim do código fonte.

Processamento de macro-instruções

- No segundo passo, cada linha de entrada é novamente lida e o campo de operação é obtido
- Se a operação especificada for MACRO, esta linha e todas as que a seguem, até aquela que contenha a operação ENDM, são ignoradas.
- Caso contrário, verifica-se se a operação está presente na tabela de definição de macro. Se não estiver, a linha é copiada para o arquivo de saída na sua forma original.
- Caso contrário, a linha contém uma invocação de macro-instrução que deve ser expandida.

Montagem

Processamento de macro-instruções

- No segundo passo, cada linha de entrada é novamente lida e o campo de operação é obtido
- Se a operação especificada for MACRO, esta linha e todas as que a seguem, até aquela que contenha a operação ENDM, são ignoradas.
- Caso contrário, verifica-se se a operação está presente na tabela de definição de macro. Se não estiver, a linha é copiada para o arquivo de saída na sua forma original.
- Caso contrário, a linha contém uma invocação de macro-instrução que deve ser expandida.

Montagem

Processamento de macro-instruções

- No segundo passo, cada linha de entrada é novamente lida e o campo de operação é obtido
- Se a operação especificada for MACRO, esta linha e todas as que a seguem, até aquela que contenha a operação ENDM, são ignoradas.
- Caso contrário, verifica-se se a operação está presente na tabela de definição de macro. Se não estiver, a linha é copiada para o arquivo de saída na sua forma original.
- Caso contrário, a linha contém uma invocação de macro-instrução que deve ser expandida.

Montagem

Processamento de macro-instruções

- No segundo passo, cada linha de entrada é novamente lida e o campo de operação é obtido
- Se a operação especificada for MACRO, esta linha e todas as que a seguem, até aquela que contenha a operação ENDM, são ignoradas.
- Caso contrário, verifica-se se a operação está presente na tabela de definição de macro. Se não estiver, a linha é copiada para o arquivo de saída na sua forma original.
- Caso contrário, a linha contém uma invocação de macro-instrução que deve ser expandida.

Montagem

Processamento de macro-instruções

- Na expansão da macro, verifica-se se a tabela da lista de argumentos contém algum elemento.
- Caso haja argumentos, as strings com os nomes dos argumentos são associadas, como valores na tabela, aos parâmetros, que são as chaves nessa tabela.

Montagem

Processamento de macro-instruções

- Na expansão da macro, verifica-se se a tabela da lista de argumentos contém algum elemento.
- Caso haja argumentos, as strings com os nomes dos argumentos são associadas, como valores na tabela, aos parâmetros, que são as chaves nessa tabela.

Montagem

Processamento de macro-instruções

- A expansão da macro continua pela leitura de linhas de código da definição, a partir da tabela de definição de macros.
- Caso a linha contenha no campo de operação a pseudo-instrução ENDM, o processo de expansão está concluído e continua a leitura do arquivo de entrada.

Montagem

Processamento de macro-instruções

- A expansão da macro continua pela leitura de linhas de código da definição, a partir da tabela de definição de macros.
- Caso a linha contenha no campo de operação a pseudo-instrução ENDM, o processo de expansão está concluído e continua a leitura do arquivo de entrada.

Montagem

Processamento de macro-instruções

- Caso contrário, caso o campo de operando contenha algum nome iniciado pelo símbolo &, então esse nome é buscado na tabela da lista de argumentos para ser substituído pela string correspondente nesta expansão.
- Após essa substituição (se houver), a linha resultante é passada para o arquivo de saída.

Montagem

Processamento de macro-instruções

- Caso contrário, caso o campo de operando contenha algum nome iniciado pelo símbolo &, então esse nome é buscado na tabela da lista de argumentos para ser substituído pela string correspondente nesta expansão.
- Após essa substituição (se houver), a linha resultante é passada para o arquivo de saída.

Montagem

Processamento de macro-instruções

- O processamento de macros conclui quando a pseudo-instrução END é copiada para o arquivo de saída, o qual então conterá apenas linhas com instruções do processador ou pseudo-instruções, já sem comentários ou macro-instruções.
- O processamento de macros pode ocorrer em um único passo caso se restrinja que todas as invocações a uma macro só podem ocorrer no código-fonte após sua definição, quando então os dois passos podem ser combinados.

Montagem

Processamento de macro-instruções

- O processamento de macros conclui quando a pseudo-instrução END é copiada para o arquivo de saída, o qual então conterá apenas linhas com instruções do processador ou pseudo-instruções, já sem comentários ou macro-instruções.
- O processamento de macros pode ocorrer em um único passo caso se restrinja que todas as invocações a uma macro só podem ocorrer no código-fonte após sua definição, quando então os dois passos podem ser combinados.

Criação de Tabela de Simbolos

- A tabela de símbolos tem como chaves as strings com os nomes simbólicos definidos no programa assembly.
- Símbolos podem ser definidos como resultado de duas situações:
  1. Como um rótulo em uma pseudo-instrução EQU; neste caso, o valor do símbolo está definido no campo do operando
  2. Como um rótulo em uma outra instrução; neste caso, o valor do símbolo está relacionado a posição de memória dentro do segmento onde ocorre a definição do símbolo.

Criação de Tabela de Simbolos

Criação de Tabela de Simbolos

- A tabela de símbolos tem como chaves as strings com os nomes simbólicos definidos no programa assembly.
- Símbolos podem ser definidos como resultado de duas situações:
  1. Como um rótulo em uma pseudo-instrução EQU; neste caso, o valor do símbolo está definido no campo do operando
  2. Como um rótulo em uma outra instrução; neste caso, o valor do símbolo está relacionado à posição de memória dentro do segmento onde ocorre a definição do símbolo.

Criação de Tabela de Simbolos

Criação de Tabela de Simbolos

- A tabela de símbolos tem como chaves as strings com os nomes simbólicos definidos no programa assembly.
- Símbolos podem ser definidos como resultado de duas situações:
  1. Como um rótulo em uma pseudo-instrução EQU; neste caso, o valor do símbolo está definido no campo do operando
  2. Como um rótulo em uma outra instrução; neste caso, o valor do símbolo está relacionado à posição de memória dentro do segmento onde ocorre a definição do símbolo.

Criação de Tabela de Simbolos

Criação de Tabela de Simbolos

- A tabela de símbolos tem como chaves as strings com os nomes simbólicos definidos no programa assembly.
- Símbolos podem ser definidos como resultado de duas situações:
  1. Como um rótulo em uma pseudo-instrução EQU; neste caso, o valor do símbolo está definido no campo do operando
  2. Como um rótulo em uma outra instrução; neste caso, o valor do símbolo está relacionado à posição de memória dentro do segmento onde ocorre a definição do símbolo.

Criação de Tabela de Simbolos

Criação de Tabela de Simbolos

- O Montador usa tabelas auxiliares:
  1. A Tabela de Instruções da Máquina (ou MOT, de machine operations table) contém toda a informação necessária para permitir a tradução de um mnemônico para o código de máquina correspondente.
  2. A Tabela de Pseudo-Instruções (ou POT, de pseudo-operations table) tem seu conteúdo definido pelos projetistas do montador. é uma tabela com conteúdo exclusivamente para consulta, não sendo modificado durante a execução do programa.

Criação de Tabela de Simbolos

- O Montador usa tabelas auxiliares:
  1. A Tabela de Instruções da Máquina (ou MOT, de machine operations table) contém toda a informação necessária para permitir a tradução de um mnemônico para o código de máquina correspondente.
  2. A Tabela de Pseudo-Instruções (ou POT, de pseudo-operations table) tem seu conteúdo definido pelos projetistas do montador. é uma tabela com conteúdo exclusivamente para consulta, não sendo modificado durante a execução do programa.

Criação de Tabela de Simbolos

- O Montador usa tabelas auxiliares:
  1. A Tabela de Instruções da Máquina (ou MOT, de machine operations table) contém toda a informação necessária para permitir a tradução de um mnemônico para o código de máquina correspondente.
  2. A Tabela de Pseudo-Instruções (ou POT, de pseudo-operations table) tem seu conteúdo definido pelos projetistas do montador. é uma tabela com conteúdo exclusivamente para consulta, não sendo modificado durante a execução do programa.

Criação de Tabela de Simbolos

Criação de Tabela de Simbolos

- A partir da informação derivada das tabelas do montador é possível saber quanto espaço de memória cada linha de instrução do programa fonte irá ocupar no módulo de carregamento gerado.
- Será a partir dessa informação que o montador poderá definir qual a posição a ser alocada para cada instrução do programa.

Criação de Tabela de Simbolos

Criação de Tabela de Simbolos

- A partir da informação derivada das tabelas do montador é possível saber quanto espaço de memória cada linha de instrução do programa fonte irá ocupar no módulo de carregamento gerado.
- Será a partir dessa informação que o montador poderá definir qual a posição a ser alocada para cada instrução do programa.

Ligadores

Ligadores

- O Montador traduz diversos arquivos fonte em arquivos objeto;
- A função do ligador é combinar todos os arquivos objetos em um único arquivo executável;
- Ele utiliza a informação de relocação e a tabela de símbolos para resolver referências entre arquivos;
- Ele também liga estaticamente (cópia) o código de funções de bibliotecas pré-compiladas (.so)

Ligadores

Ligadores

- O Montador traduz diversos arquivos fonte em arquivos objeto;
- A função do ligador é combinar todos os arquivos objetos em um único arquivo executável;
- Ele utiliza a informação de relocação e a tabela de símbolos para resolver referências entre arquivos;
- Ele também liga estaticamente (cópia) o código de funções de bibliotecas pré-compiladas (.so)

Ligadores

Ligadores

- O Montador traduz diversos arquivos fonte em arquivos objeto;
- A função do ligador é combinar todos os arquivos objetos em um único arquivo executável;
- Ele utiliza a informação de relocação e a tabela de símbolos para resolver referências entre arquivos;
- Ele também liga estaticamente (cópia) o código de funções de bibliotecas pré-compiladas (.so)

Ligadores

Ligadores

- O Montador traduz diversos arquivos fonte em arquivos objeto;
- A função do ligador é combinar todos os arquivos objetos em um único arquivo executável;
- Ele utiliza a informação de relocação e a tabela de símbolos para resolver referências entre arquivos;
- Ele também liga estaticamente (cópia) o código de funções de bibliotecas pré-compiladas (.so)

Ligadores

Ligação Estática e Dinâmica

- Há duas formas de se utilizar funções de bibliotecas em um programa
  1. Funções de Bibliotecas Estáticas;
  2. Funções de Bibliotecas Dinâmicas

Ligadores

Ligação Estática e Dinâmica

- Há duas formas de se utilizar funções de bibliotecas em um programa
  1. Funções de Bibliotecas Estáticas;
  2. Funções de Bibliotecas Dinâmicas

Ligadores

Ligação Estática e Dinâmica

- Há duas formas de se utilizar funções de bibliotecas em um programa
  1. Funções de Bibliotecas Estáticas;
  2. Funções de Bibliotecas Dinâmicas

Ligadores

Ligação Estática e Dinâmica

- Na Ligação Estática o código objeto da função a ser ligada é copiado para o arquivo executável;

Ligadores

Ligação Estática e Dinâmica

- Na ligação dinâmica, o SO carrega a priori ou sob demanda o código das bibliotecas (.lib, .dll)
  1. Símbolos externos são adicionados identificando as funções dinâmicas;
  2. O endereço real destas funções é definido durante o processo de carregamento do programa

Ligadores

Ligação Estática e Dinâmica

- Na ligação dinâmica, o SO carrega a priori ou sob demanda o código das bibliotecas (.lib, .dll)
  1. Símbolos externos são adicionados identificando as funções dinâmicas;
  2. O endereço real destas funções é definido durante o processo de carregamento do programa

Ligadores

Ligação Estática e Dinâmica

- Na ligação dinâmica, o SO carrega a priori ou sob demanda o código das bibliotecas (.lib, .dll)
  1. Símbolos externos são adicionados identificando as funções dinâmicas;
  2. O endereço real destas funções é definido durante o processo de carregamento do programa

Carregadores

Carregadores

- Parte do Sistema Operacional;
- Responsável por carregar um programa executável em memória

Carregadores

Carregadores

- Parte do Sistema Operacional;
- Responsável por carregar um programa executável em memória

Carregadores

Carregadores

- Na realidade, O carregador faz diversas chamadas ao Sistema Operacional:
  1. Cria um Descritor de Processo
  2. Selecionar um espaço de endereçamento
  3. Carregar na memória os dados estáticos;
  4. Carregar na memória o texto do programa;Resolver os endereços dinâmicos de chamadas de funções
  5. Agendar o novo programa como pronto para execução;

Carregadores

Carregadores

- Na realidade, O carregador faz diversas chamadas ao Sistema Operacional:
  1. Cria um Descritor de Processo
  2. Selecionar um espaço de endereçamento
  3. Carregar na memória os dados estáticos;
  4. Carregar na memória o texto do programa;Resolver os endereços dinâmicos de chamadas de funções
  5. Agendar o novo programa como pronto para execução;

Carregadores

Carregadores

- Na realidade, O carregador faz diversas chamadas ao Sistema Operacional:
  1. Cria um Descritor de Processo
  2. Selecionar um espaço de endereçamento
  3. Carregar na memória os dados estáticos;
  4. Carregar na memória o texto do programa;Resolver os endereços dinâmicos de chamadas de funções
  5. Agendar o novo programa como pronto para execução;

Carregadores

Carregadores

- Na realidade, O carregador faz diversas chamadas ao Sistema Operacional:
  1. Cria um Descritor de Processo
  2. Selecionar um espaço de endereçamento
  3. Carregar na memória os dados estáticos;
  4. Carregar na memória o texto do programa;Resolver os endereços dinâmicos de chamadas de funções
  5. Agendar o novo programa como pronto para execução;

Carregadores

Carregadores

- Na realidade, O carregador faz diversas chamadas ao Sistema Operacional:
  1. Cria um Descritor de Processo
  2. Selecionar um espaço de endereçamento
  3. Carregar na memória os dados estáticos;
  4. Carregar na memória o texto do programa;Resolver os endereços dinâmicos de chamadas de funções
  5. Agendar o novo programa como pronto para execução;

Carregadores

Carregadores

- Na realidade, O carregador faz diversas chamadas ao Sistema Operacional:
  1. Cria um Descritor de Processo
  2. Selecionar um espaço de endereçamento
  3. Carregar na memória os dados estáticos;
  4. Carregar na memória o texto do programa;Resolver os endereços dinâmicos de chamadas de funções
  5. Agendar o novo programa como pronto para execução;

Bibliografia

Livros Texto

- AHO, A; ULLMANN, J; REVI, S. Compiladores: Princípios, técnicas e ferramentas 3 ed. Rio de Janeiro: LTC-Livros
- LOUDEN, Kenneth C; SILVA, Flávio Soares Corrêa. Compiladores : princípios e práticas. 1a ed. São João da Boa Vista: Pioneira - Thomson Learning, 2004.
- PRICE, Ana M. A.; TOSCANI, Simão S.. Implementação de linguagens de programação: compiladores. 3a ed. Porto AlegreBookman, 2008.
- SETZER, V.W.. Construção de um Compilador. 1a ed. Rio de Janeiro: Campus - Elsevier, 1983.

FIM

Grato

MUITO OBRIGADO.