

Introdução à Arquitetura e Organização de Computadores

Professor: Diego de Assis Monteiro Fernandes

Aula 7 - Programas: Do código fonte ao executável

Essa aula mostra quais são as etapas em que um código fonte passa até chegar ao código executável.

1 Execução de um Programa

A seguir estão descritos os passos para transformar um programa em C em um arquivo executável no disco. Apesar de serem descritos quatro passos, alguns sistemas combinam alguns desses passos para reduzir o tempo de conversão. A Figura 1 ilustra a hierarquia de tradução de um código fonte, ressaltando o que utilizado para transformar o código entre as etapas.

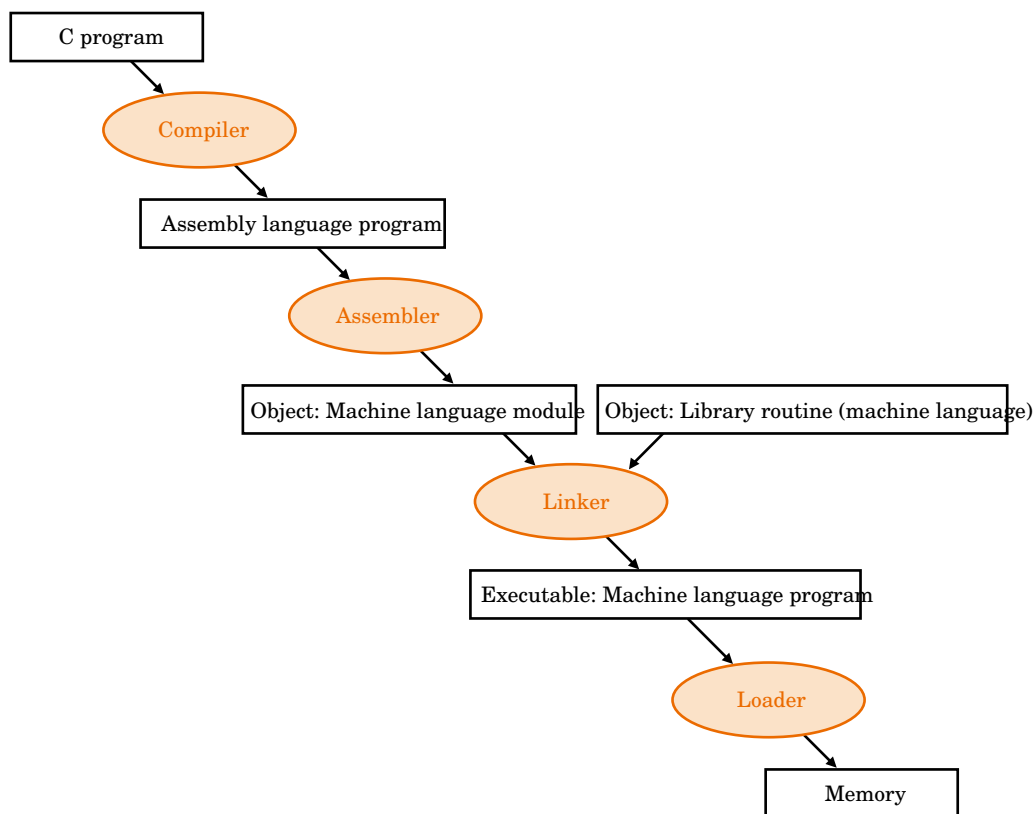


Figura 1: A hierarquia de tradução de um programa.

As entidades que transformam o código são o compilador (compiler), o montador (assembler), o ligador (linker) e o carregador (loader). Cada uma das entidades estão descritas a seguir.

1.1 Compilador

Basicamente, a função do compilador é a de transformar o programa em C em um programa em linguagem de montagem, por exemplo, na linguagem de montagem MIPS.

1.2 Montador (Assembler)

A linguagem de montagem pode ser considerada como a interface para a linguagem de alto nível. Tal linguagem é compreendida pelo montador. É comum que nestas linguagens estejam incluídas variações de algumas instruções contida na linguagem de máquina. É função do montador tratar essas variações comuns das instruções de linguagem de máquina, compreendendo portanto instruções que não estão implementadas diretamente no *hardware*. Tais instruções, chamadas **pseudo-instruções** facilitam a programação provendo operações mais intuitivas. Por exemplo, uma pseudo-instrução compreendida pelo montador da linguagem de montagem MIPS é a **move**, que transfere o conteúdo de um registrador para outro. A transferência feita por **move \$t0, \$t1** pode ser convertida na instrução **add \$t0,\$zero,\$t1** que é efetivamente compreendida pelo *hardware*. Montadores também aceitam números em algumas bases numéricas, geralmente binária, decimal e hexadecimal, ao contrário do hardware que trabalha somente na base binária.

Entretanto, a tarefa principal do montador é criar o código de máquina. Para isso, o montador converte um programa em linguagem de montagem em um **arquivo objeto**, que é a combinação de instruções em linguagem de máquina, dados e informações que permitem o armazenamento adequado das instruções na memória. Um arquivo objeto para sistemas Unix contém tipicamente seis partes distintas:

- **cabeçalho**: descreve o tamanho e a posição das outras partes do arquivo
- **segmento de texto**: contém o código de máquina
- **segmento de dados**: contém todo o tipo de dado manipulado no programa, sejam eles estáticos ou dinâmicos.
- **informação de relocação**: identifica as instruções e dados que dependem do endereço absoluto quando o programa é carregado na memória.
- **tabelas de símbolos**: contém os rótulos restantes que não estão definidos.
- **informação de depuração**: contém a descrição de como os módulos foram compilados de modo que o depurador seja capaz de associar as instruções de máquina ao programa fonte em C.

1.3 Linker

Para evitar que todo o programa seja compilado quando o código fonte de um único procedimento é alterado, cada procedimento é compilado e montado independentemente. Desse modo, a alteração do código fonte de um único procedimento só requer sua compilação, uma vez que os procedimentos restantes permaneceram inalterados. O Linker é o programa que realiza a união desses procedimentos, pois toma programas em linguagem de máquina independentes e os une. Para fazer isso, a informação de relocação e a tabela de símbolos de cada módulo é utilizada.

A Figura 2 mostra a convenção utilizada pelo MIPS para alocar endereços de memória a programas e dados. Considerando que os arquivos foram montados separadamente, o montador não sabe a posição relativa das instruções e dos dados de cada um dos módulos. Quando o linker coloca o módulo na memória, todas as referências são absolutas, ou seja, endereços de memória que não sejam relativos a um registrador precisam ser relocados para refletir sua verdadeira posição.

O linker produz um **arquivo executável** que finalmente pode executar em um computador. Tipicamente, este arquivo tem o mesmo formato do arquivo objeto, exceto pelo fato de não conter referências não resolvidas, nem informações de relocação e nem tabela de símbolos.

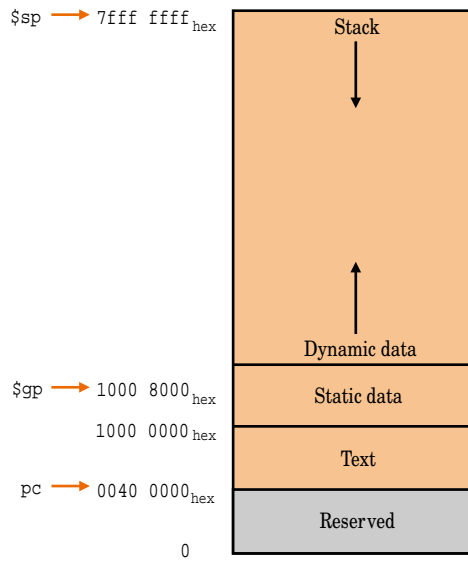


Figura 2: Alocação do espaço de memória no MIPS, para código e dados.

1.4 Loader

Depois que o arquivo executável é gerado, o sistema operacional o lê na memória e inicia sua execução. Para isso os seguintes passos são seguidos no Unix:

1. Lê o cabeçalho do arquivo executável para determinar o tamanho dos segmentos de texto e dados.
2. Cria um espaço de endereçamento grande o suficiente para dados e texto
3. Copia as instruções e dados do arquivo para a memória.
4. Copia os parâmetros (se houver) para o programa principal na pilha.
5. Inicializa os registradores e atribui o primeiro endereço livre para o ponteiro da pilha (`$sp` em MIPS).
6. Desvia para a rotina de inicialização que copia os parâmetros para os registradores de argumento e invoca a rotina principal do programa.