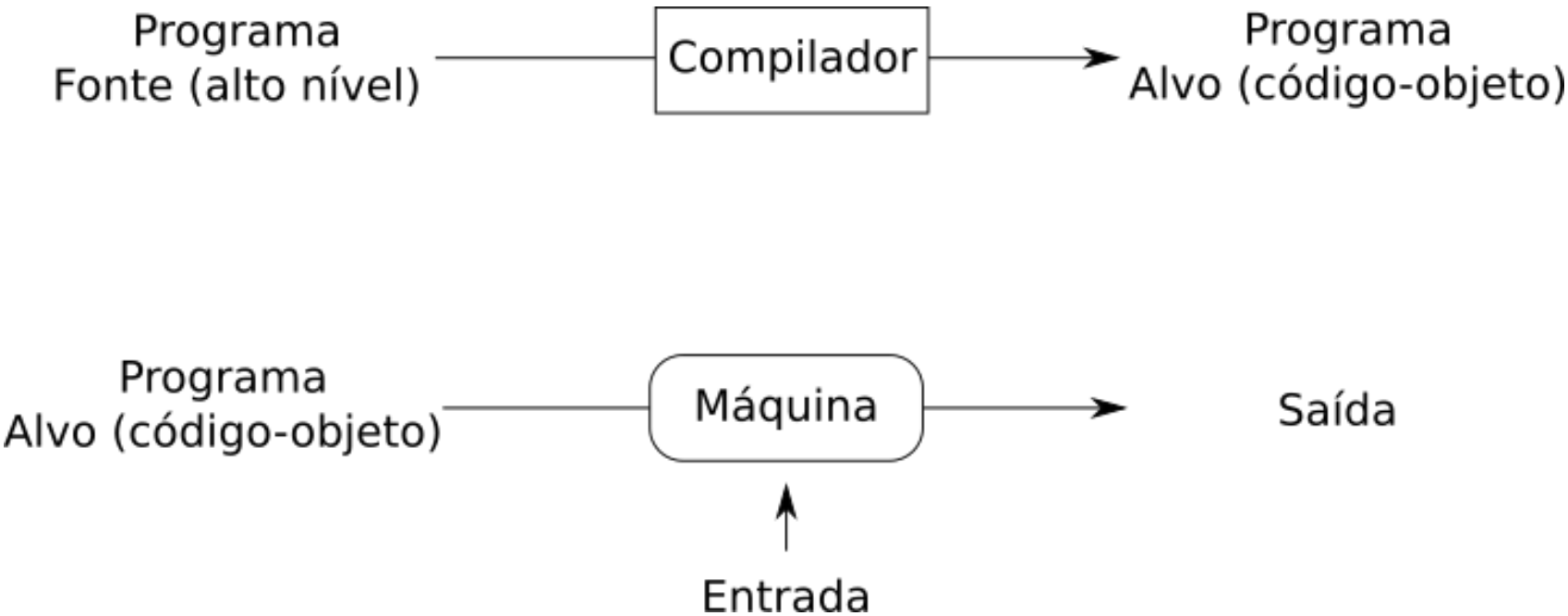
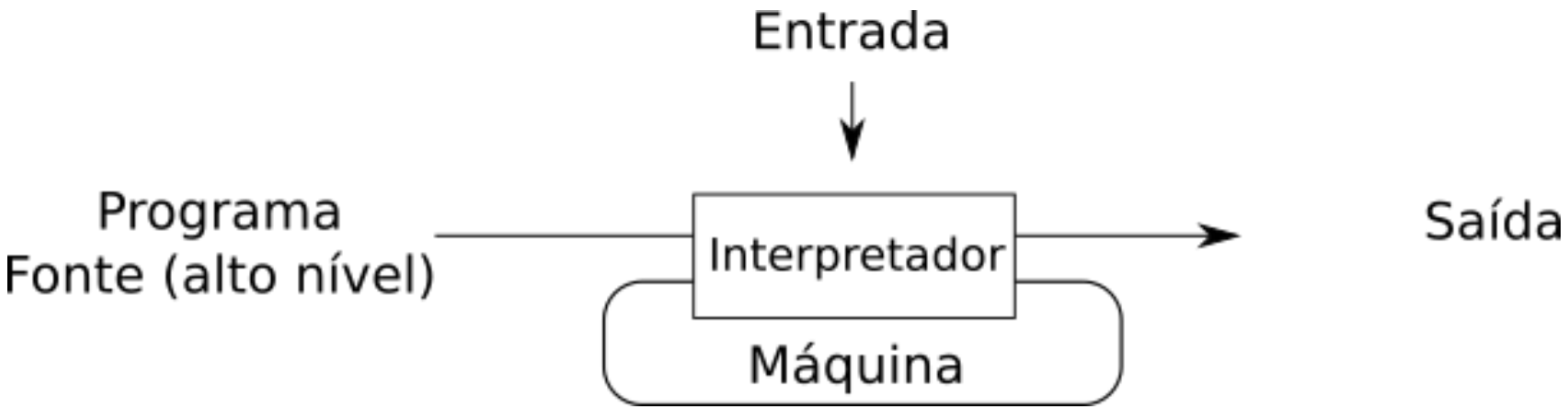
Visão Geral do Processo de Compilação

João Marcelo Uchôa de Alencar
joao.marcelo@ufc.br
UFC-Quixadá

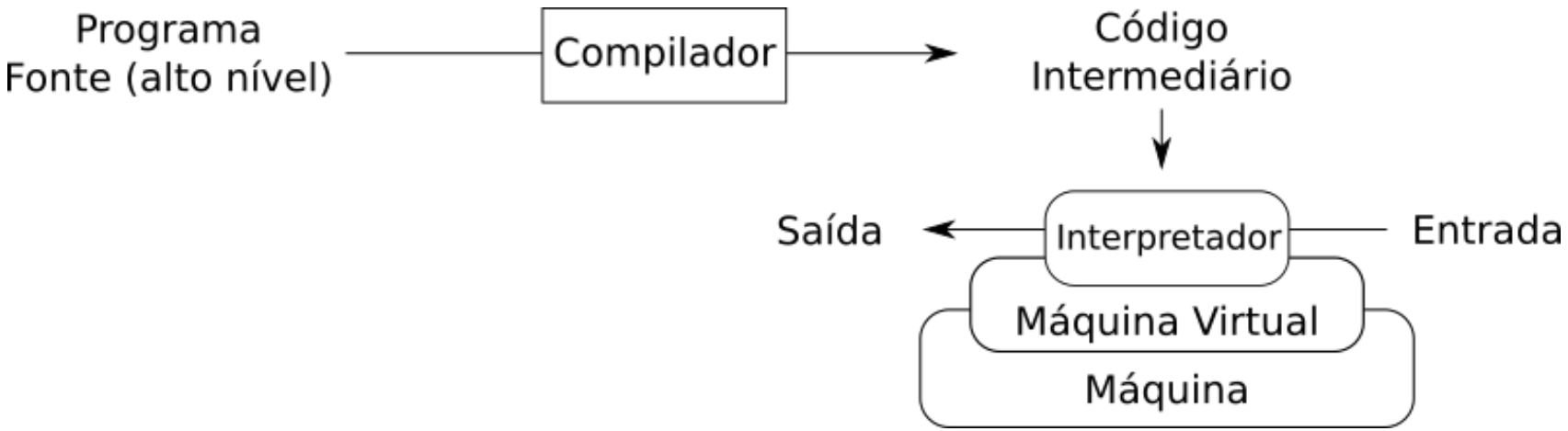
Visão Geral


Programa
Fonte (alto nível)
Compilador
Programa
Alvo (código-objeto)
Programa
Alvo (código-objeto)
Máquina
Saída
Entrada

Visão Geral


Entrada
Programa
Fonte (alto nível)
Interpretador
Saída
Máquina

Visão Geral


Programa
Fonte (alto nível)
Compilador
Código
Intermediário
Saída
Interpretador
Entrada
Máquina Virtual
Máquina

Reflexões

- Qual é a diferença entre um compilador e interpretador?
- Quais as vantagens/desvantagens de cada um?
- Um compilador que ao receber uma linguagem de alto nível produza código em C no lugar de código objeto teria alguma utilidade?

Histórico

Linguagem de Máquina

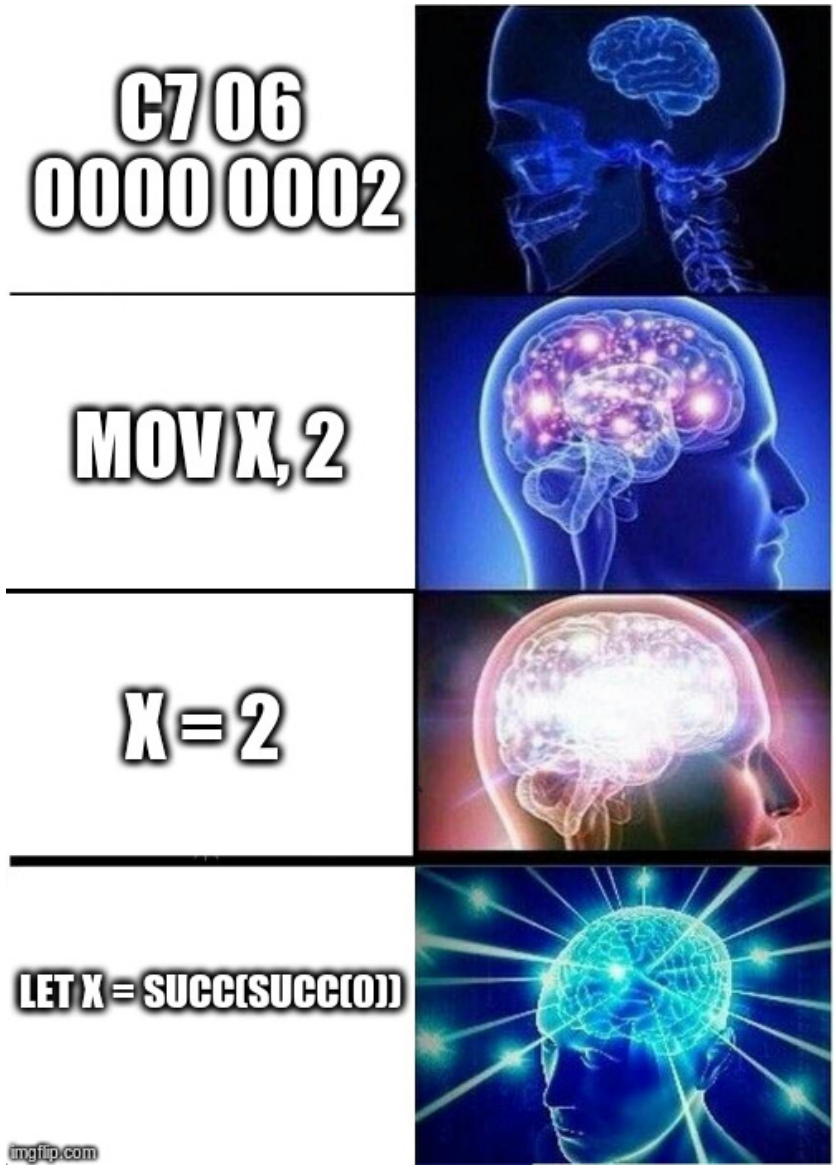
C7 06 0000 0002

Linguagem de Montagem

MOV X, 2

Linguagem de Alto Nível

x = 2;



Histórico

- John Backus: Fortran;
- Noam Chomsky: Gramáticas;
- Geradores de analisadores sintáticos e geradores de sistemas de varredura;
- Verificação de tipos e linguagens funcionais.

Exemplo - Linguagem de Alto Nível

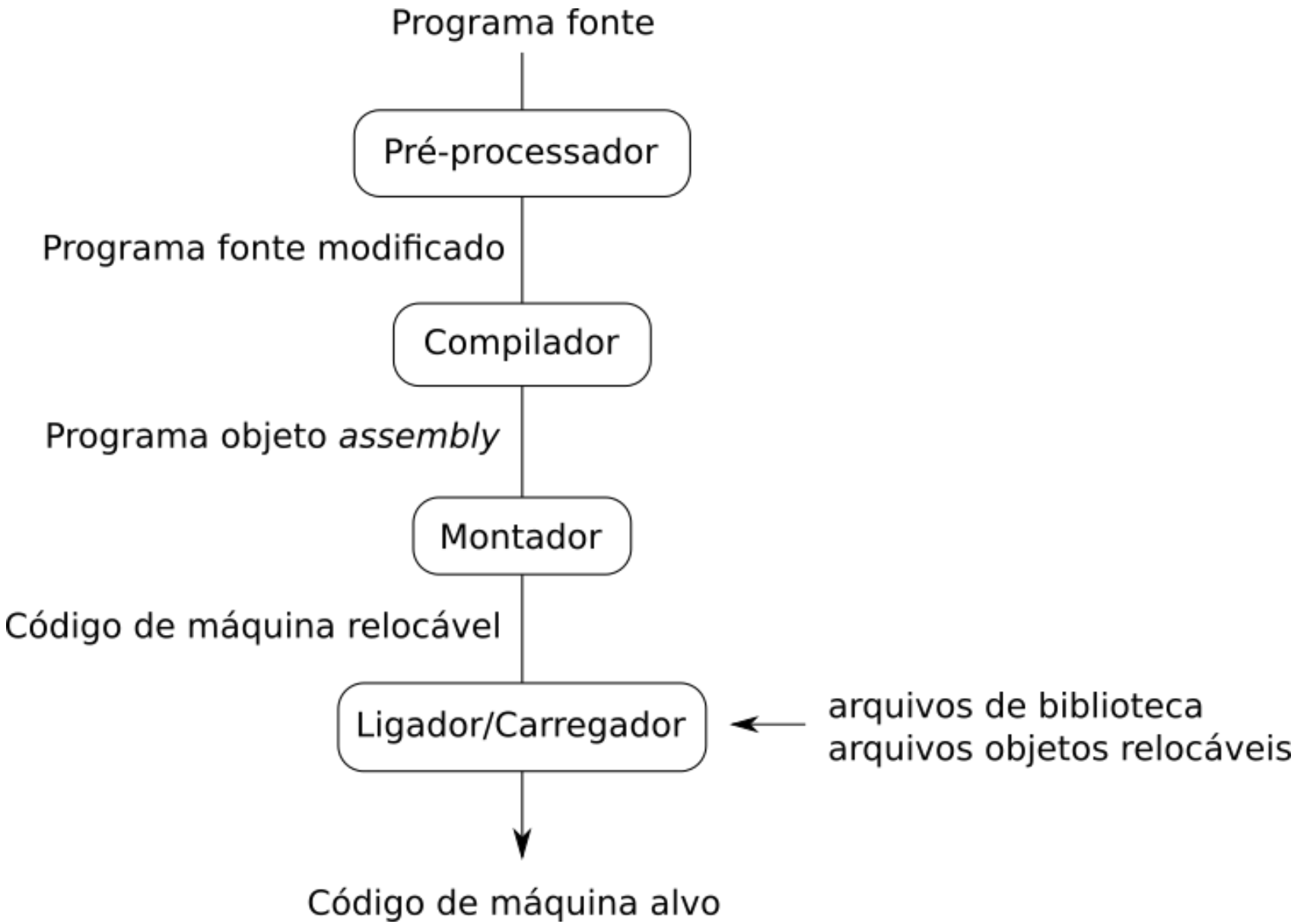
```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    int x = 2;
    return 0;
}
```

Exemplo - Linguagem de Montagem

```
        .file   "historico.c"
        .text
        .globl   main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -20(%rbp)
        movq    %rsi, -32(%rbp)
        movl    $2, -4(%rbp)
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident   "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.6) 5.4.0 20160609"
        .section   .note.GNU-stack,"",@progbits
```

Programas Relacionados



Exemplo de Compilação - Programa em C

- Vamos compilar um simples programa em C para entender as etapas;
- considere uma pasta *projeto* com o seguinte conteúdo:
  - Subdiretório *include* com o arquivo *biblioteca.h*;
  - subdiretórios *lib* e *bin* vazios;
  - subdiretório *src* com os arquivos *main.c* e *biblioteca.c*.
- nós vamos criar uma biblioteca e usá-la em um programa principal;
- a divisão do programa em vários arquivos é regra em programas não triviais.

Exemplo de Compilação - Programa em C

```c
/*** Conteúdo de biblioteca.h ***/
void funcao_da_biblioteca();

/*** Conteúdo de biblioteca.c ***/
#include <stdio.h>
void funcao_da_biblioteca() {
   printf("Olá Mundo da Biblioteca.\n");
   return;
}

/*** Conteúdo de main.c ***/
#include <stdio.h>
#include "biblioteca.h"
int main(int argc, char *argv[]) {
   funcao_da_biblioteca();
   return 0;
}
```

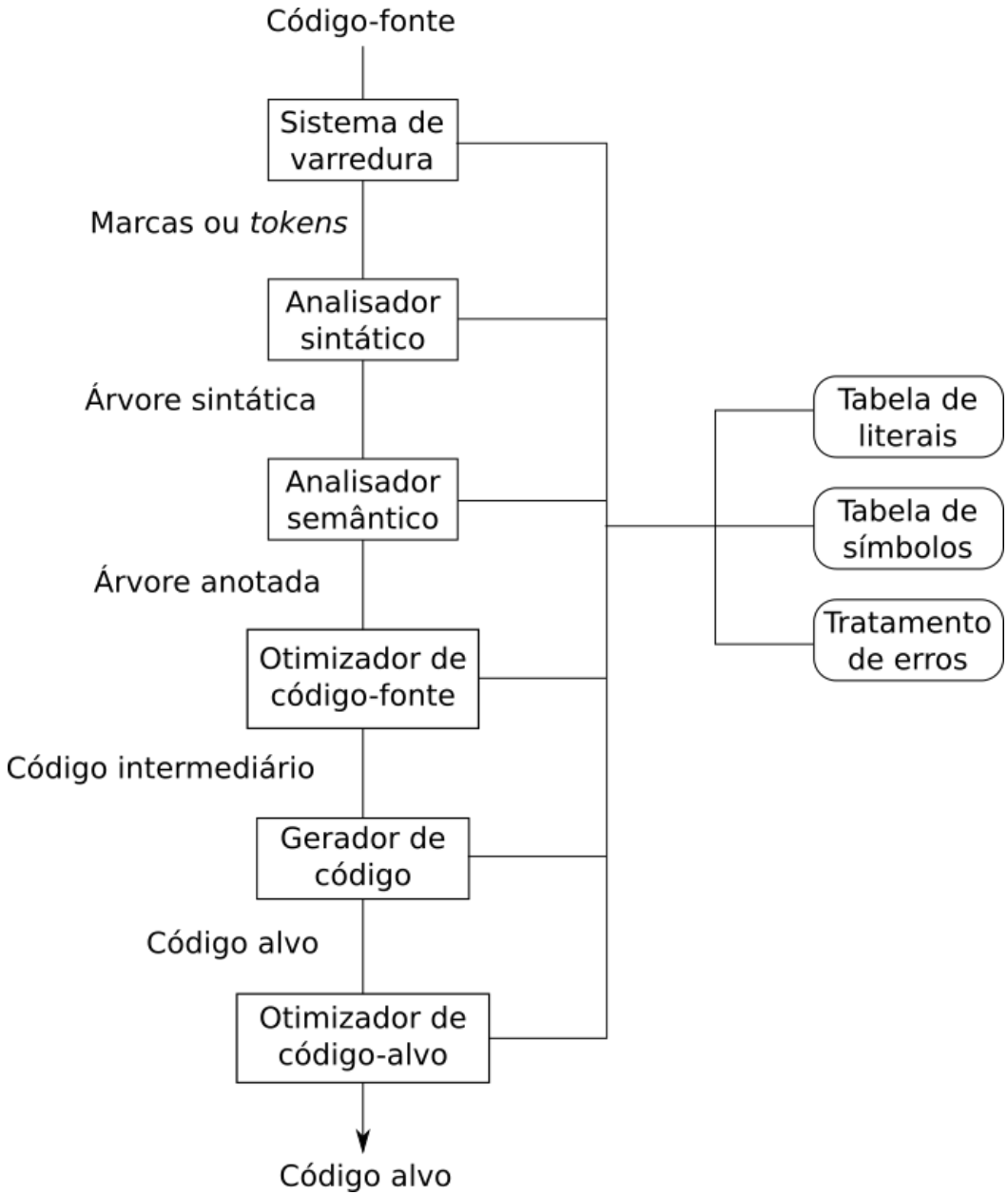
Exemplo de Compilação - Programa em C

- O exemplo é trivial, mas já envolve muitos passos;
- um programa complexo, com centenas de arquivos fonte, compilado da forma que vamos mostrar seria impraticável;
- a maioria utiliza ferramentas (*autoconf*) para automatizar a compilação:
  1. Informações sobre o ambiente são inseridas em um arquivo de configuração;
  2. um *script* chamado *configure* realiza testes e coleta mais informações sobre o sistema;
  3. o comando *make* invoca o compilador várias vezes para construir o projeto.
- porém, não é o foco da disciplina apresentar as ferramentas *autoconf*.

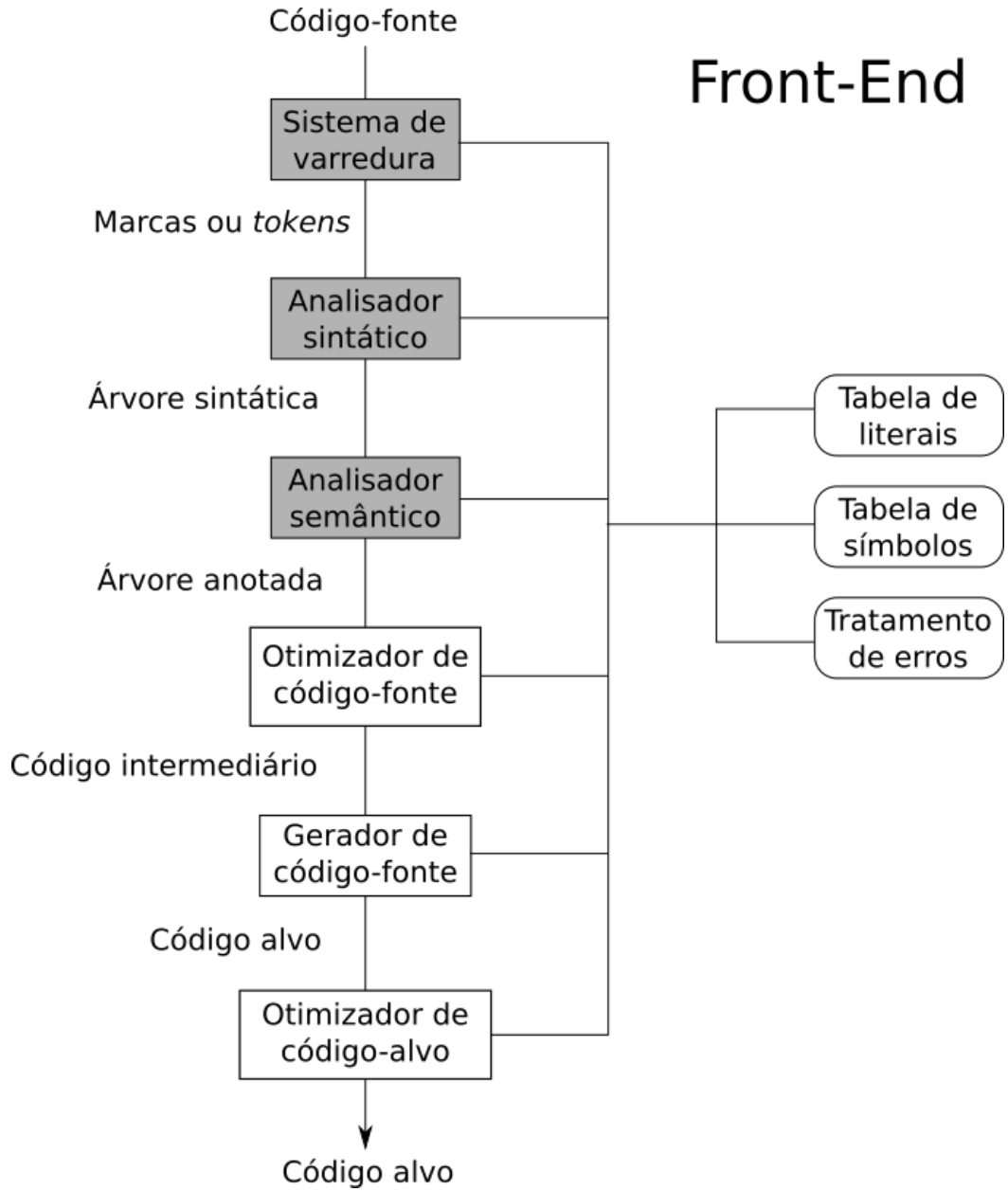
Exemplo de Compilação - Programa em C

```
$ cd projeto
$ gcc -Iinclude/ -c src/main.c -o bin/main.o
$ gcc -Iinclude/ -c src/biblioteca.c -fPIC\\
-o bin/biblioteca.o
$ gcc -shared bin/biblioteca.o -o lib/libbiblioteca.so
$ gcc bin/main.o -Llib/ -lbiblioteca -o bin/a.out
$ bin/a.out
bin/a.out: error while loading shared libraries
$ LD_LIBRARY_PATH=lib/ bin/a.out
Olá Mundo da Biblioteca.
```

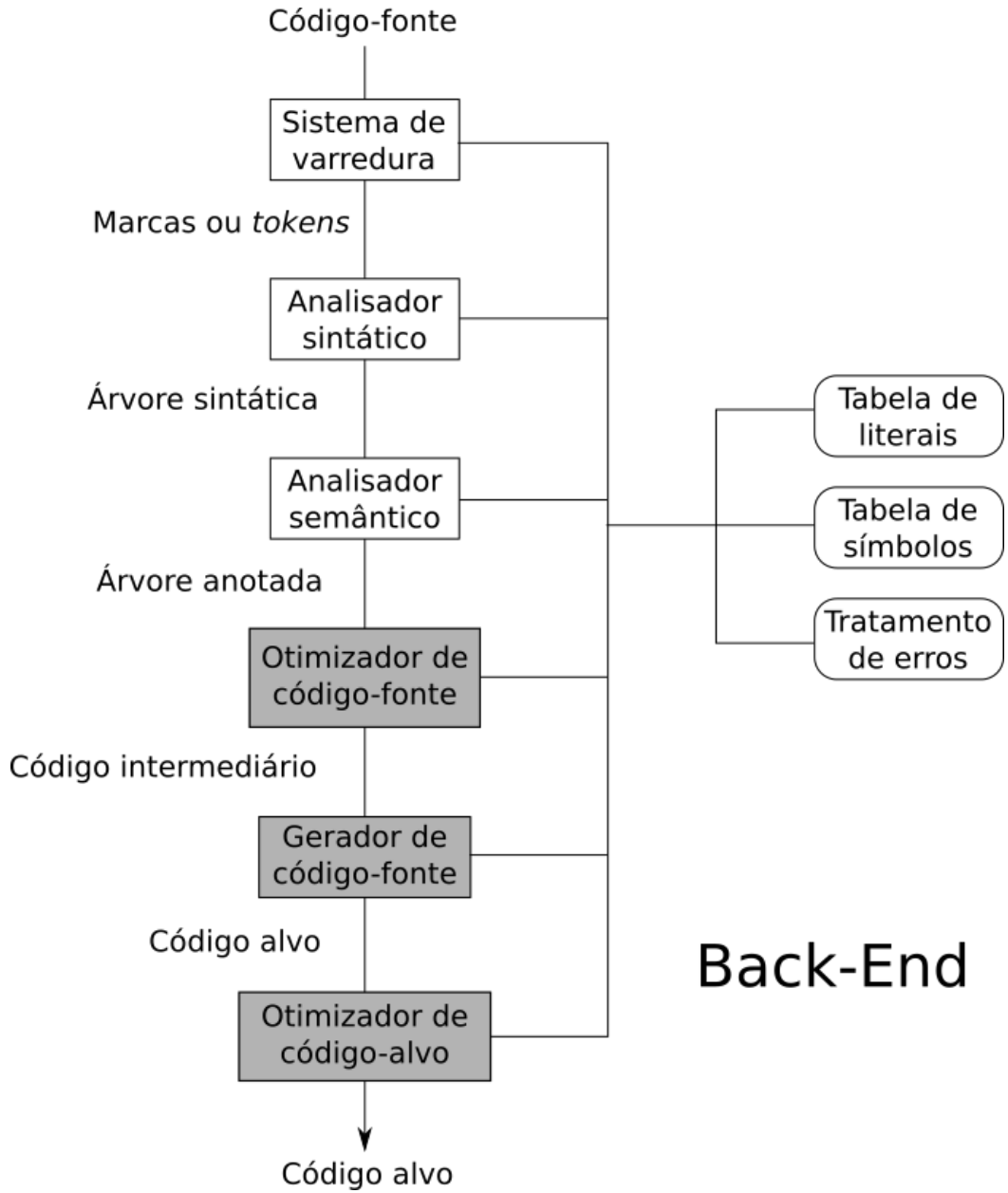
As Fases de um Compilador



As Fases de um Compilador



As Fases de um Compilador



Varredura ou Análise Léxica

Marcas ou *tokens*

Sequências de caracteres organizadas como unidades significativas.

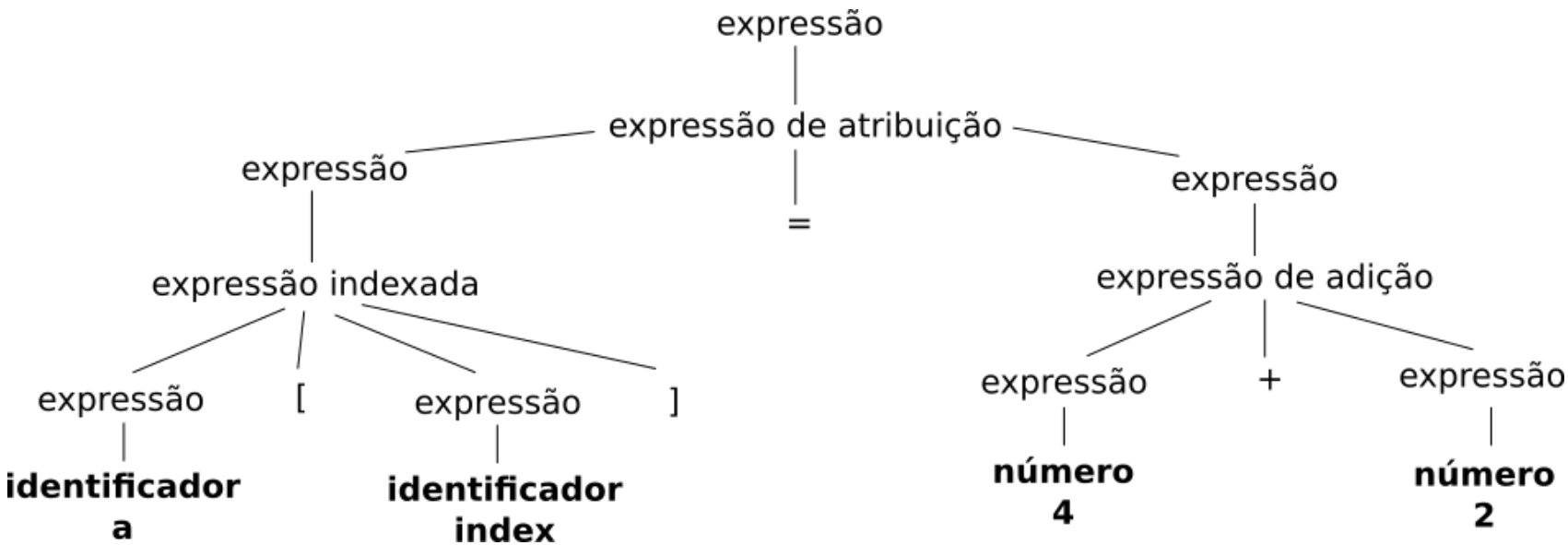
```
a [index] = 4 + 2
```

Símbolo	*Token*
a	identificador
[	colchete à esquerda
index	identificador
]	colchete à direita
=	atribuição
4	número
+	sinal de adição
2	número

Analisador Sintático

Árvore sintática

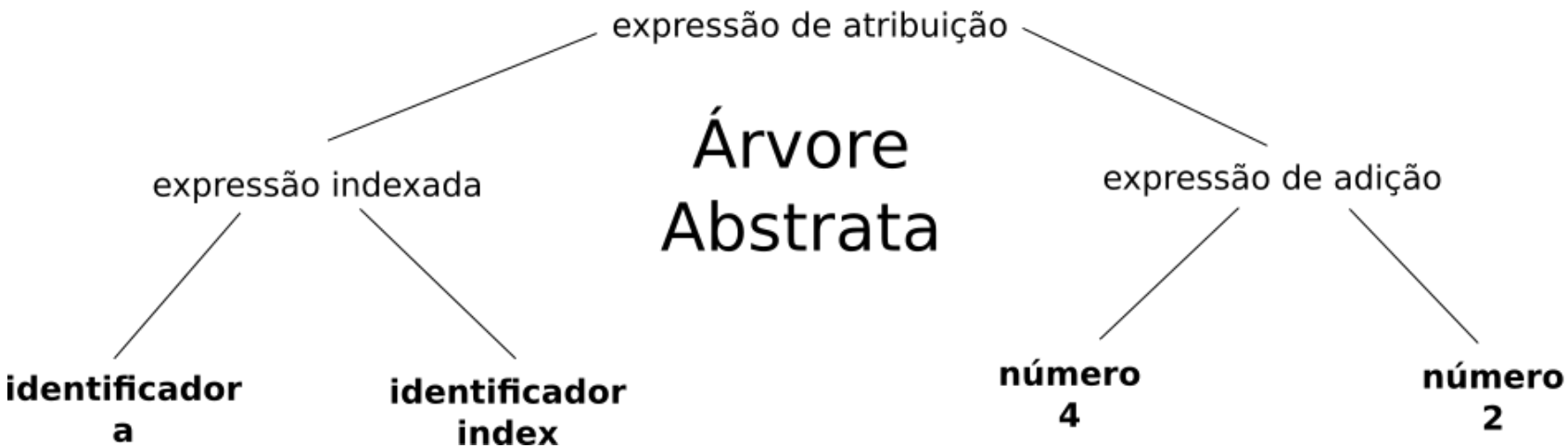
Determina os elementos estruturais do programa e seus relacionamentos.



Analisador Sintático

Árvore sintática

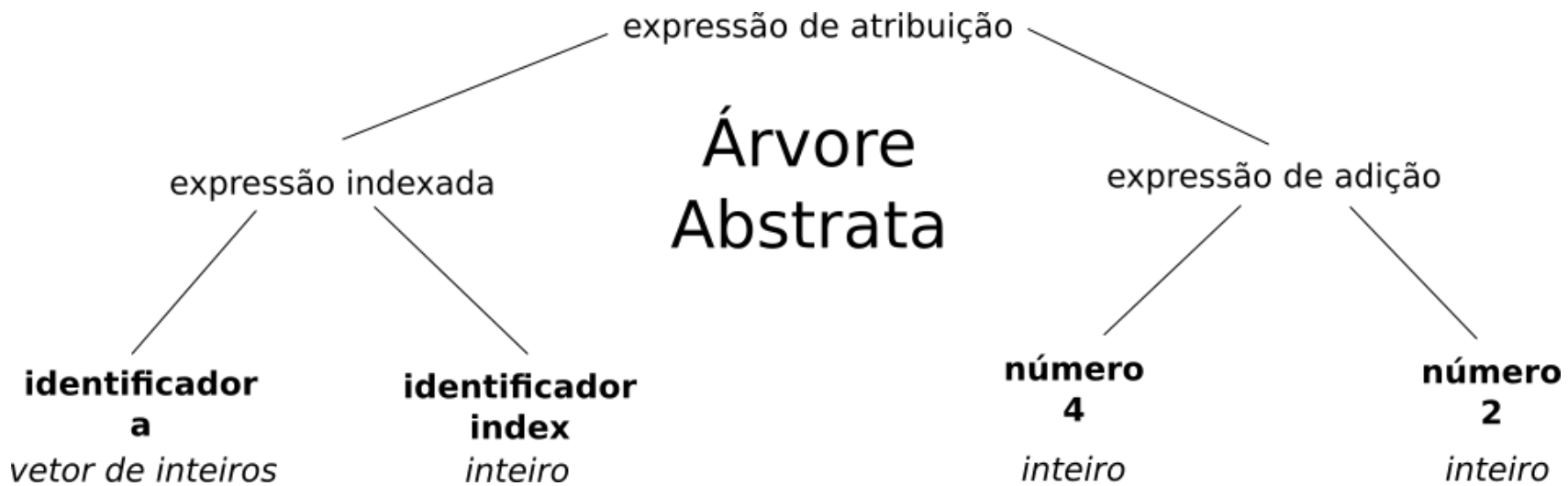
Determina os elementos estruturais do programa e seus relacionamentos.



Analisador Semântico

Verificação de Tipos

Consistência do programa em relação aos tipos empregados.



Otimizador de Código-Fonte

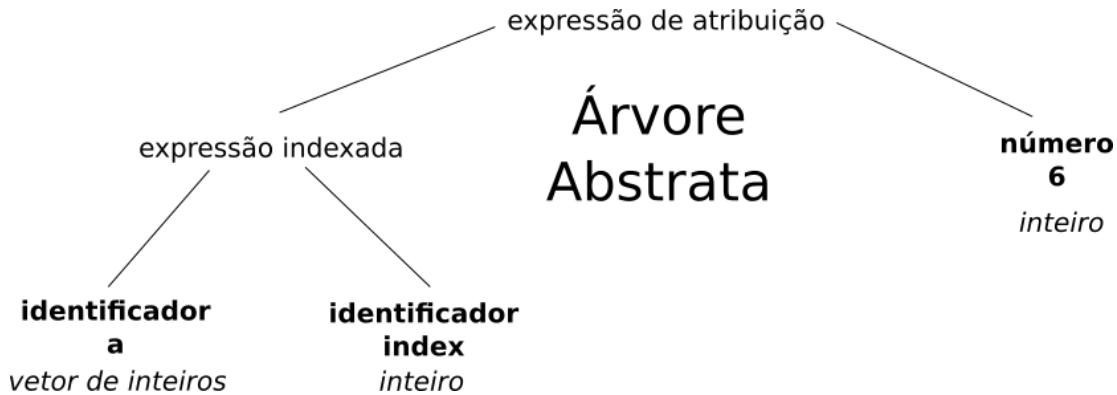
Otimizações à Nível da Linguagem de Alto Nível

Aplicar técnicas padrão de melhoria no código.

```
// Original
t = 4 + 2;
a[index] = t;

// Primeiro passo
t = 6;
a[index] = t;

// Segundo Passo
a[index] = 6;
```



Árvore Abstrata

Gerador de Código

Código Objeto

A partir do código intermediário, gera o código objeto. (Obs: Mostramos código de montagem por fins didáticos).

```
MOV R0, index   ;; valor de index para R0
MUL R0, 2       ;; dobra o valor em R0
MOV R1, &a      ;; endereço de a para R1
ADD R1, R0      ;; adiciona R0 a R1
MOV *R1, 6      ;; constate 6 para o endereço em R1
```

Otimizador de Código-Alvo

Melhorias no Código Objeto

Tira proveiro de características da arquitetura.

```
MOV R0, index ;; valor de index para R0
SHL R0        ;; dobra o valor em R0
MOV &a[R0], 6 ;; constante 6 para endereço a + 10
```

Estruturas de Dados

Marcas ou *Tokens*

Descrição do símbolo e cadeia de caracteres associada.

Arvóre Sintática

Cada nó é um registro com atributos sintáticos e semânticos.

Tabela de Símbolos

Identificadores de funções, variáveis, constantes e tipos de dados.

Estruturas de Dados

Tabela de Literais

Constantes e cadeias de caractares.

Código Intermediário

Estrutura de vida curta que permite manipulações para otimizações.

Arquivos Temporários

Quando a memória era pequena, arquivos eram utilizados para armazenas as estruturas do compilador.

Questões de Projeto

- Análise e síntese;
- frente e fundo;
- passadas;
- definições de linguagens;
- opções e interfaces de um compilador;
- tratamento de erros.

Reflexões

- Em qual linguagem foi feito o primeiro compilador C?
- Em qual linguagem são escritos os compiladores C da atualidade?
- O compilador Java, é escrito em qual linguagem?
- A máquina virtual Java, é escrita em qual linguagem?
- Se vou instalar um sistema em um sensor ou sistema embarcado, devo compilar o código no sensor/sistema?

A Linguagem TINY

- Um curso de compiladores poderia mostrar como construir um compilador de C para a arquitetura x86-64;
- seria cansativo para o professor;
- os alunos teriam que ter alto conhecimento da especificação de C e de instruções x86-64;
- o índice de reprovação seria alto!

Solução

Utilizar uma linguagem de programação *brinquedo* para exercitar os conceitos: TINY.

A Linguagem TINY

```
{ Exemplo de programa em TINY - fatorial }
read x; { inteiro de entrada }
if x > 0 then { não calcula se x <= 0 }
   fact := 1;
   repeat
      fact := fact * x;
      x := x - 1;
   until x = 0;
   write fact { apresenta o fatorial de x }
end
```

A Máquina TM

- Mesmo para uma linguagem simples como TINY, gerar código x86-64 é árduo;
- a solução novamente é adotar uma máquina *brinquedo* hipotética;
- a máquina TM possui apenas instruções básicas, sendo de compreensão mais acessível;
- novamente, trabalhamos com linguagem de montagem para facilitar a compreensão, mas a maioria dos compiladores produz código objeto.

A Máquina TM

O código a[index]=6; equivale a:

```
LDC   1,0(0)    coloca 0 no registrador 1
* a instrução abaixo assume que o índice (index)
* está no endereço 10 da memória
LD    0,10(1)   coloca valor de 10 + R1 no registrador R0
LDC   1,2(0)    coloca 2 no registrador R1
MUL   0,1,0     coloca R1 * R0 em R0
LDC   1,0(0)    coloca 0 no registrador R1
* a instrução abaixo assume que a variável a está no
* endereço 20 da memória
LDA   1,20(1)   coloca 20 + R1 no registrador R1
ADD   0,1,0     coloca R1 + R0 no registrador R0
LDC   1,6(0)    coloca 6 no registrador R1
ST    1,0(0)    armazena o valor de R1 no endereço 0+R0
```

Dúvidas?

Dúvidas ou comentários?