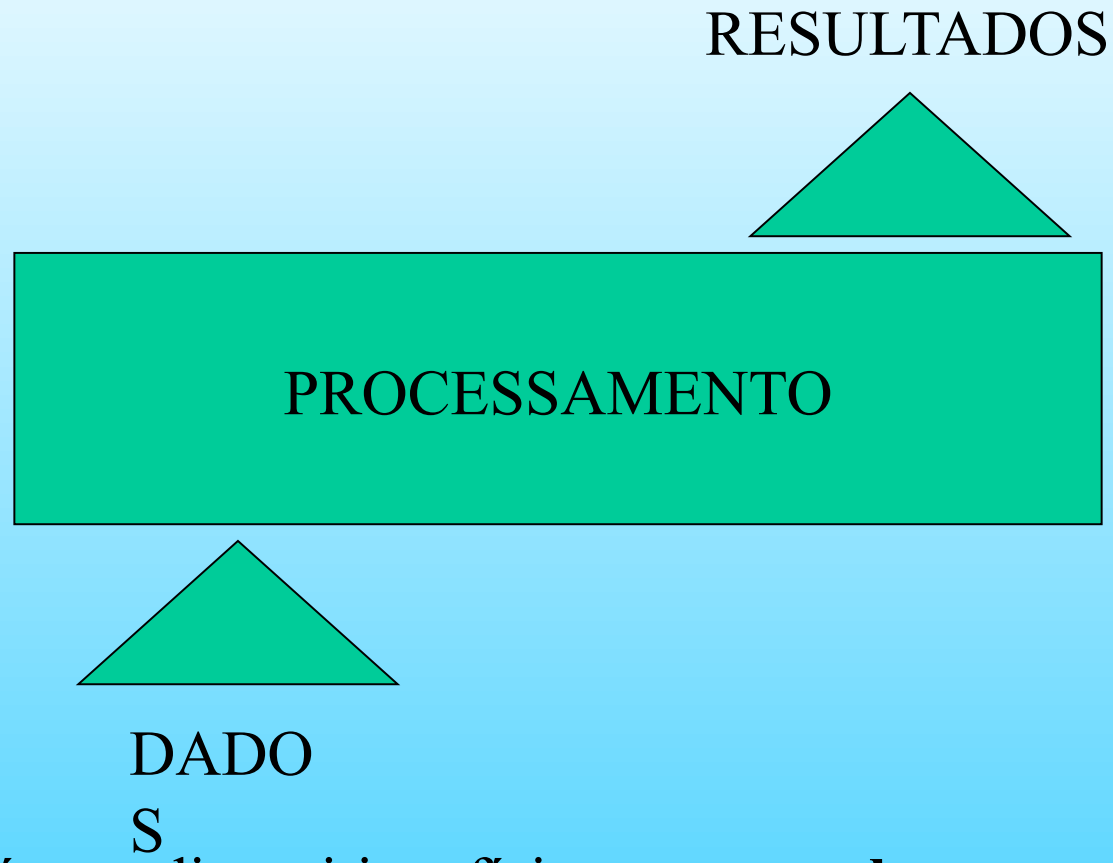
5955001

COMPILADORES

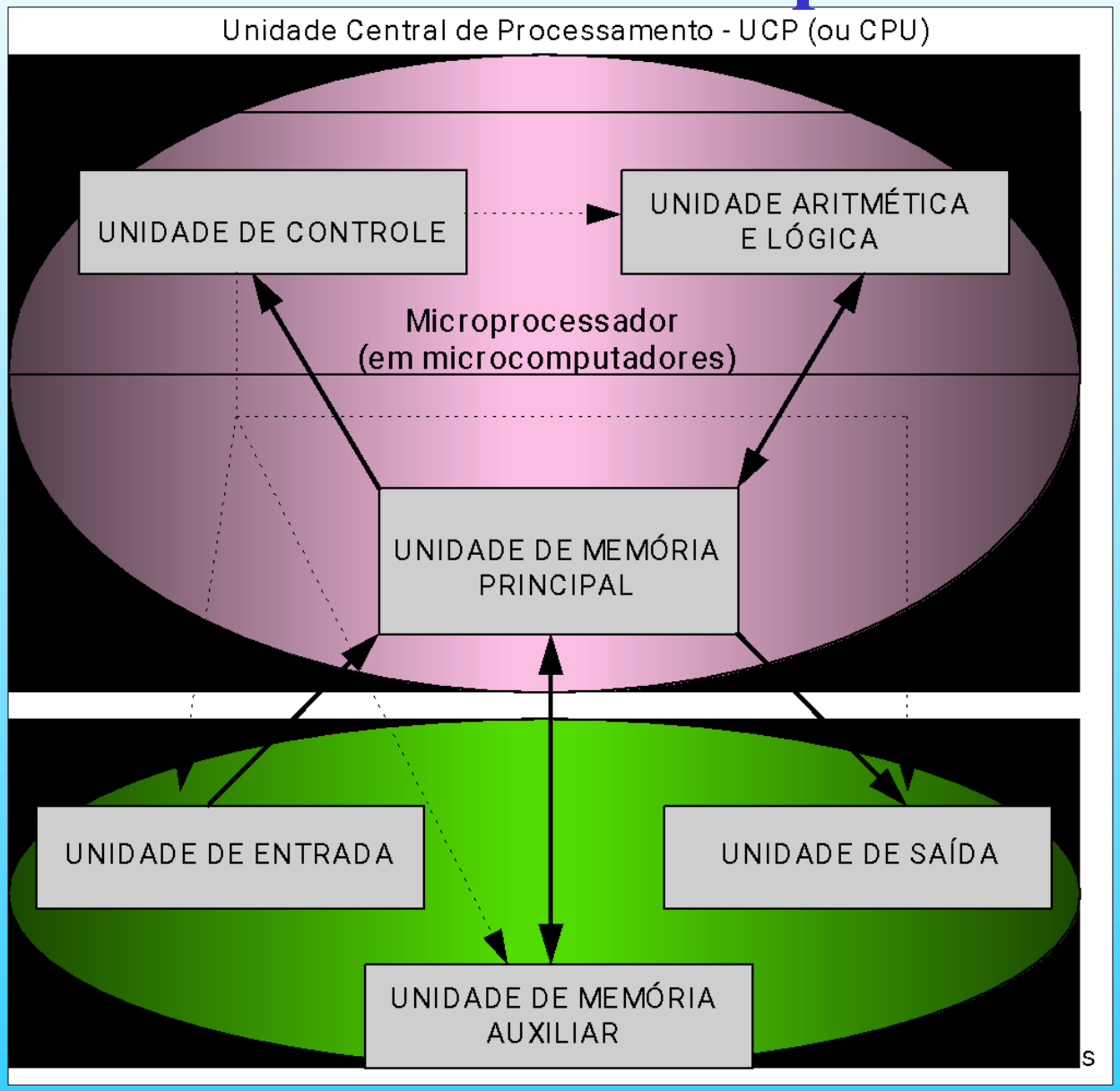
Prof. Zhao Liang

O Sistema de Computador

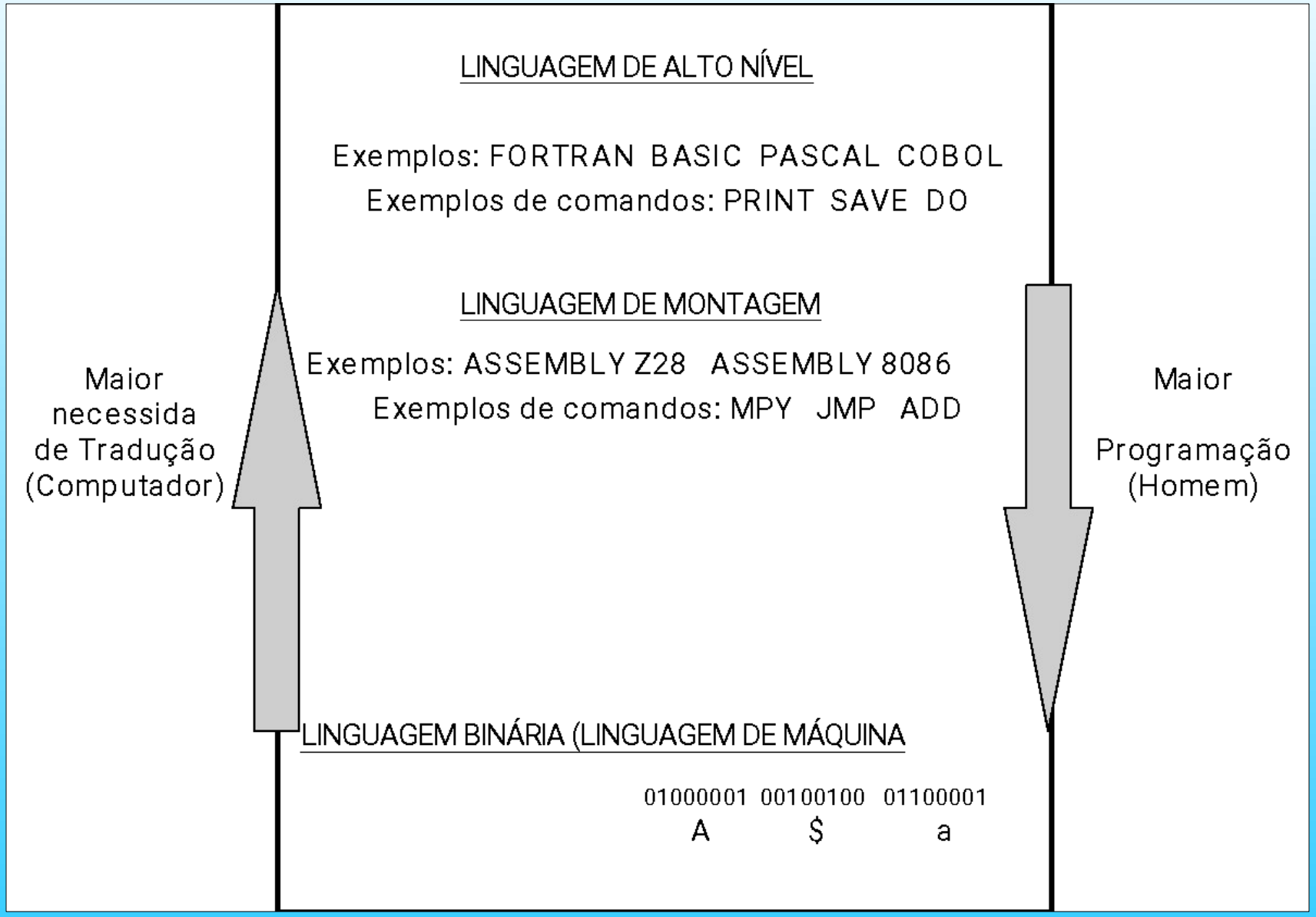


Um **computador** é um dispositivo físico que **recebe** dados como entrada, **transforma** esses dados pela execução de um **programa** armazenado e **envia** informações para diversos dispositivos.

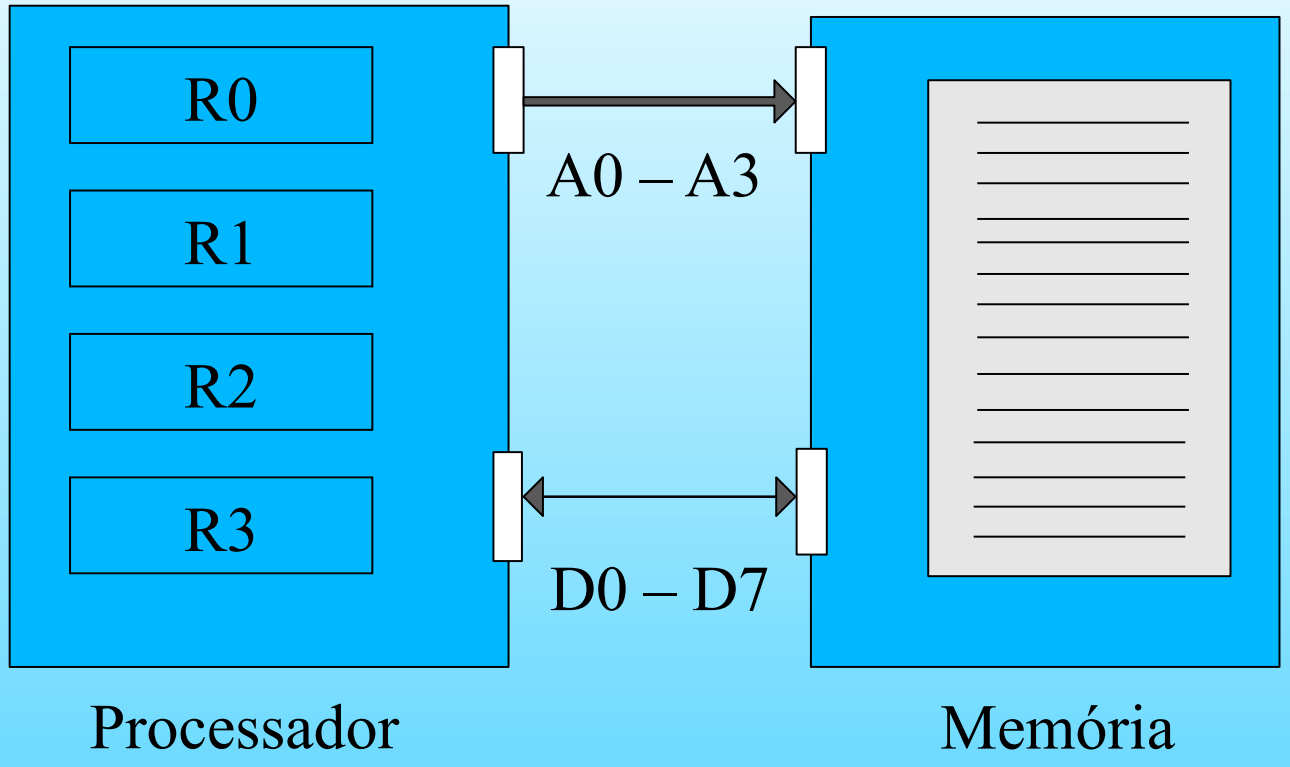
Arquitetura Básica de Computadores



Evolução das linguagens de programação

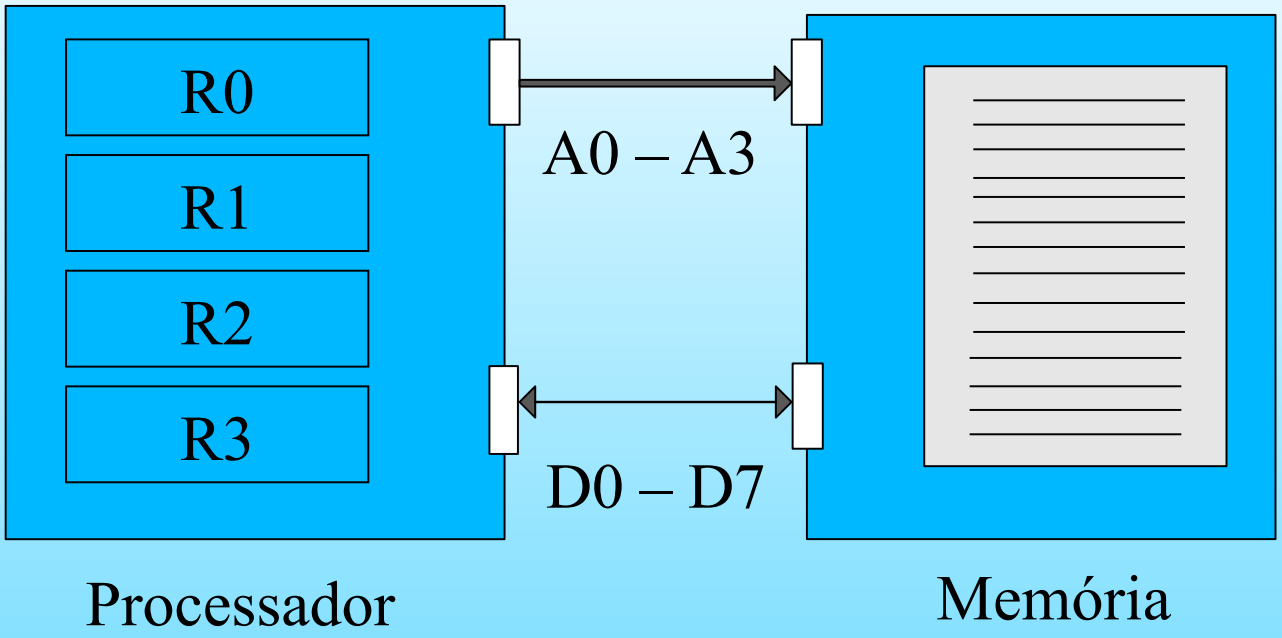


Arquitetura de um Processador Hipotético



Um processador de 8 bits (dimensão do barramento de dados)
A capacidade de endereçamento é 16 posições, pois tem quatro linhas de endereço.

Arquitetura de um Processador Hipotético



LOAD: Transferir conteúdo de uma posição de memória para um registrador.
STORE: Transferir o conteúdo de um registrador para uma posição de memória.
ADD: Adicionar o conteúdo de dois registradores e armazenar o resultado em outro registrador.
BZERO: Mudar o conteúdo do contador de programas para a posição de memoria especificada se o conteúdo do registrador indicado foi igual a zero

Arquitetura de um Processador Hipotético

LOAD 10, R1		00101001
LOAD 11, R2		00101110
ADD R1, R2, R0		10011000
STORE R0, 12		01001100

LOAD: 00; STORE: 01; ADD: 10; BZERO: 11
R0: 00; R1: 01; R2: 10; R3: 11

Linguagem de Alto Nível

As instruções são expressas na forma de um texto que independe do processador. Esse texto, o código fonte, é composto por uma combinação de palavras e expressões que se aproximam mais daquelas usadas na linguagem humana do que daquelas usadas para comandar o processador.

if, while, print, etc., ao inves de LOAD, MOVE.

Compiladores

- **Definição**

  Programa que

-Lê um programa escrito numa linguagem: a "linguagem-fonte"

-Traduz em um programa equivalente escrito em outra linguagem: a "linguagem-objeto"

-Reporta a presença de erros no "programa-fonte"

Compiladores

Possíveis linguagens-fontes:

-Mais comuns: tradicionais de linguagens de programação: FORTRAN, PASCAL, C, etc.

-Linguagens especializadas: banco de dados, simulação, experimentos: SQL, MATLAB, etc.

Possíveis linguagens-objetos:

-Mais comuns: linguagens de “assembly” ou binária de várias máquinas;

- Outra linguagem de programação.

Compiladores

Nesta matéria, o enfoque principal é a tradução de uma linguagem de programação tradicional para a linguagem assembly de alguma máquina

O Contexto de um Compilador

Além do compilador, outros programas são exigidos para criar um programa executável em alguma máquina

Compiladores

Preprocessador: realiza várias tarefas antes da compilação

-Inclui outros arquivos no programa.

Exemplo: #include <arq.h> em C

-Processa marcos.

Exemplo em C, com as marcos:

#define EHPAR(x) (((x) % 2) ? 0 : 1)

#define ERRO (meus) printf("erro: %s\n", meus)

Compiladores

Preprocessador: realiza várias tarefas antes da compilação

O preprocessador substitui o primeiro argumento do #define pelo segundo.

No programa, pode-se escrever comandos como:

if(EHPAR(a+b)) ..... if((((a+b) % 2) ? 0 : 1))

if(valor > MAX) ERRO("valor muito grande");

if(valor > MAX) printf("erro: %s\n", "valor muito grande");

Compiladores

Montador: transforma o código assembly produzido pelo compilador em código de máquina realocável.

Carregador / Ligador (loader/link-editor):

-O código gerado pelo montador pode não carregado a partir do endereço zero da memória

-Os endereços dos instruções precisam ser corrigidos

-Muitas rotinas da biblioteca e do usuário podem ser ligadas ao programa

-Tudo isso é feito pelo “Carregador-Ligador”

-Feito isso, o programa já pode ser executado

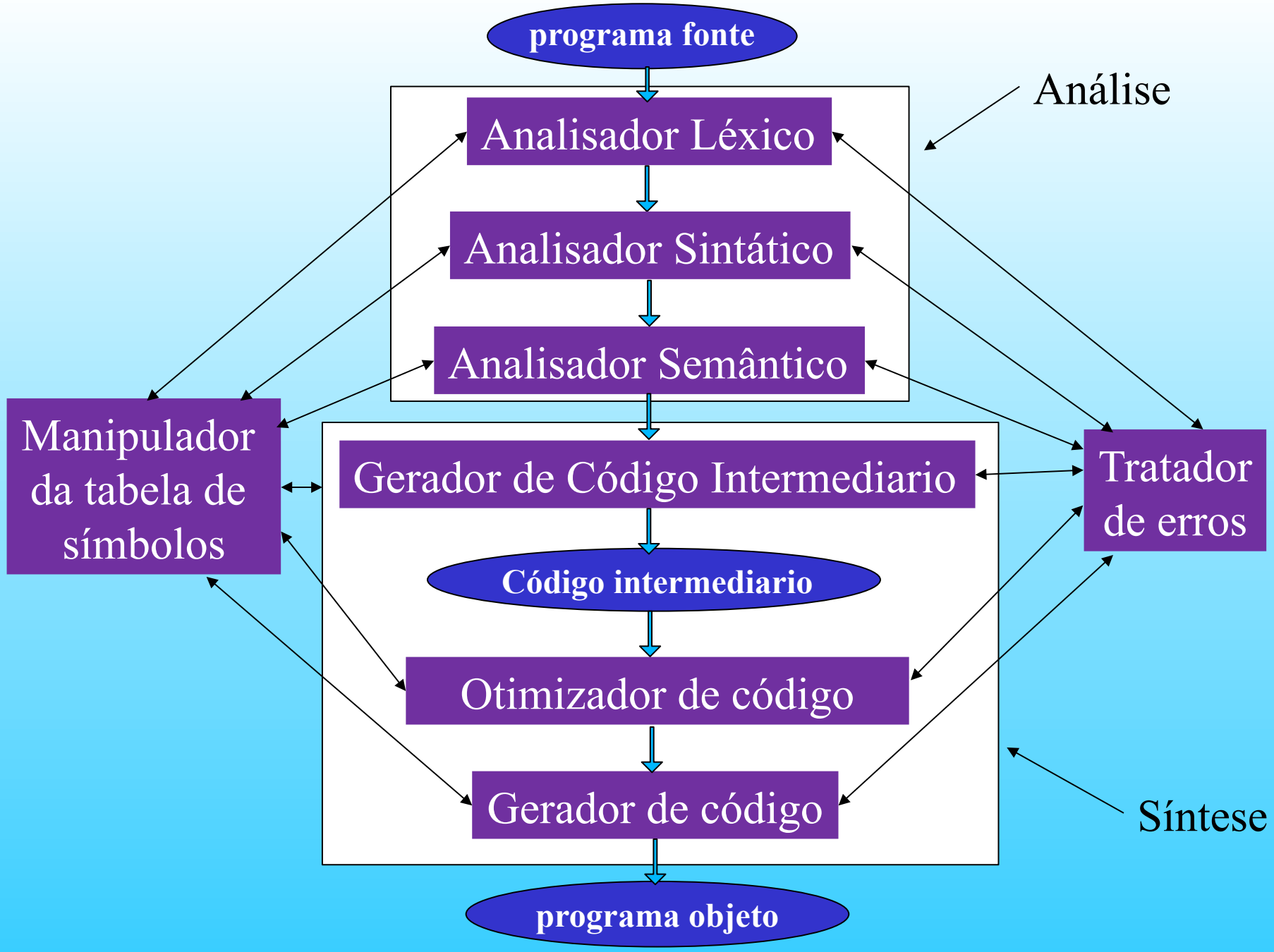
Estrutura de um Compilador

Componentes de um Compilador

O trabalho de compilação é divido em 2 fases:

- **Fase de análise**
- **Fase de síntese**

Existem atividades que fazem partes das duas fases


programa fonte
Análise
Analisador Léxico
Analisador Sintático
Analisador Semântico
Manipulador da tabela de símbolos
Gerador de Código Intermediario
Tratador de erros
Código intermediario
Otimizador de código
Gerador de código
Síntese
programa objeto

A Fase de Análise

São realizadas três tipos de análise:

- **Análise linear de léxica**
- **Análise hierárquica ou sintática**
- **Análise semântica**

Análise Léxica

- **As caracteres do texto são agrupados em átomos (tokens);**
- **A validade dos átomos é verificada;**
- **Os átomos recebem uma classificação.**

Exemplo:

Frase da língua portuguesa:

Ajbx o homen alto apanhou a laranja madura na laranjeira tdhf

Análise Léxica

átomo	classe	átomo	classe
Ajbx	inválido	laranja	substantivo
o	artigo	madura	adjetivo
homen	substantivo	na	contração
alto	adjetivo	laranjeira	substantivo
apanhou	flexão verbal	tdhf	inválido
a	artigo		

Análise Léxica

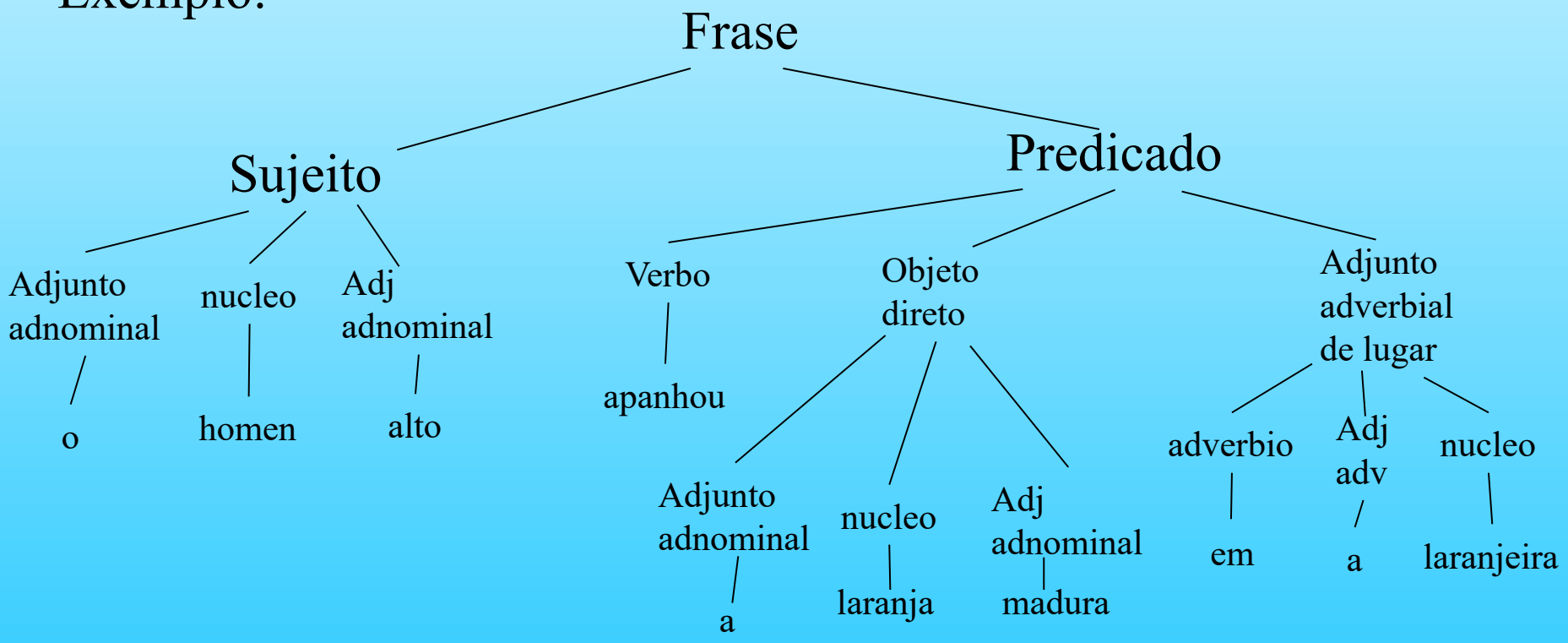
Exemplo: Um commando **while**:
while num < 50 **do** num := num *2

átomo	**classe**	**átomo**	**classe**
while	palavra reservada	num	identificador
num	identificador	:=	operador
<	operador	num	identificador
50	constante	*	operador
do	palavra reservada	2	constante

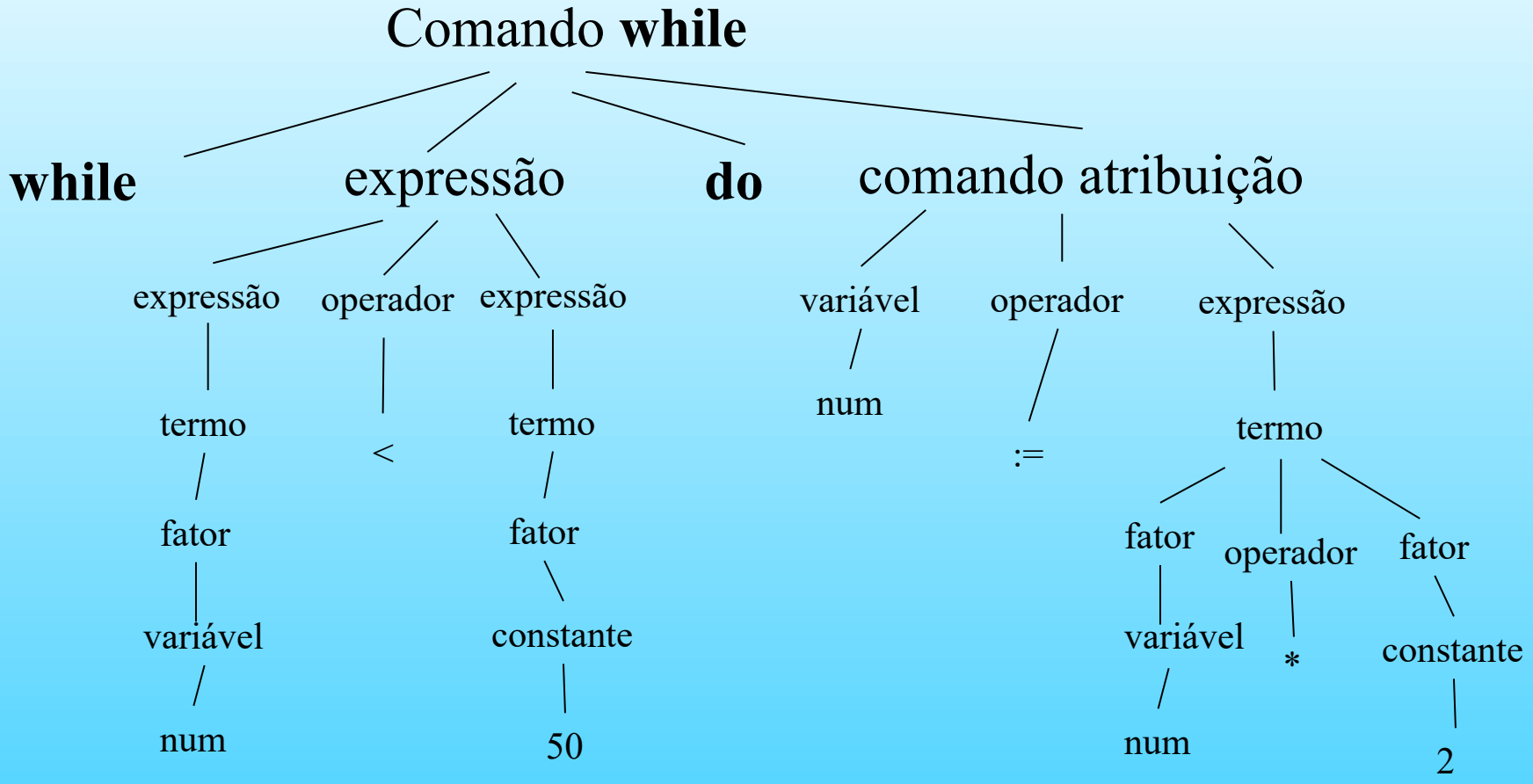
Análise Sintática

- Os átomos são agrupados em frases, em estrutura de árvore
  (árvore sintática)
- A validade da posição dos átomos é verificada

Exemplo:



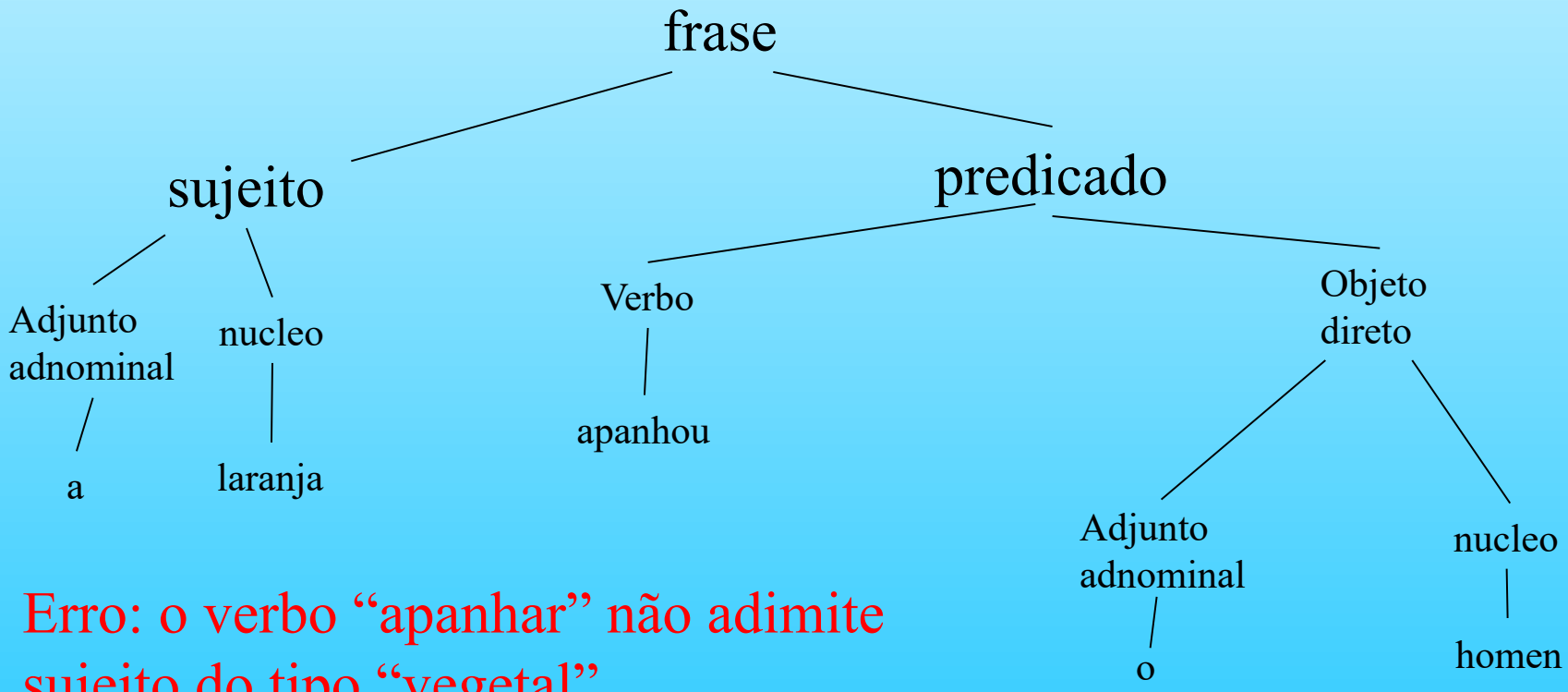
Análise Sintática



Análise Semântica

- Verificar se a árvore sintática tem sentido
- Coleta informação de tipos para a fase de síntese

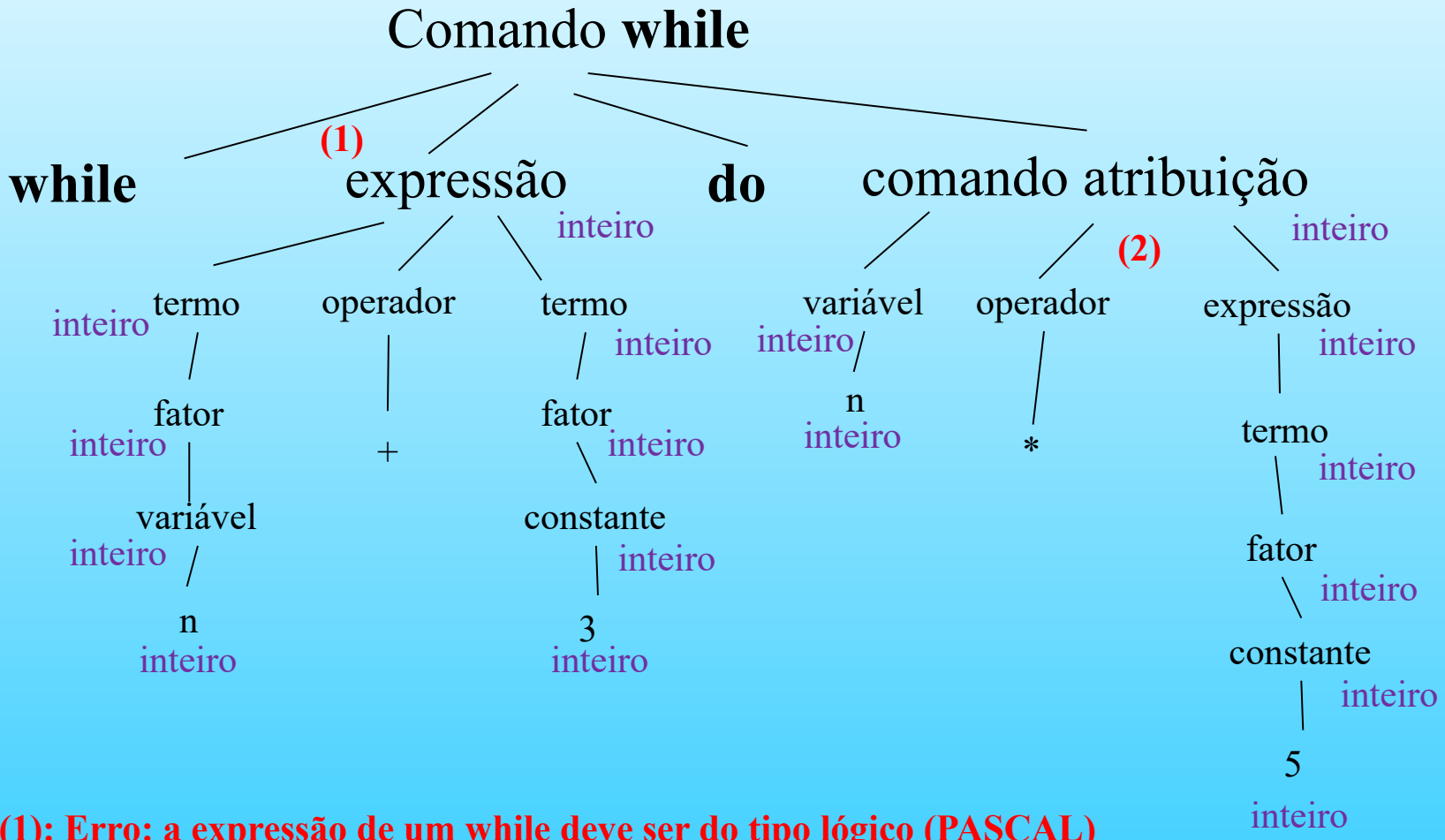
Exemplo: A laranja apanhou o homen



Erro: o verbo “apanhar” não adimite sujeito do tipo “vegetal”.

Análise Semântica

Exemplo: **while** n+3 **do** n*5



(1): Erro: a expressão de um while deve ser do tipo lógico (PASCAL)

(2): Erro: o operador de um commando de atribuição só pode ser o ":=".

A Fase de Síntese

Depois da análise, essa fase constroi o programa objeto

São realizadas três tarefas:

-Geração do código intermediário

-Otimização deste código

-Geração do código objeto

átomo	classe	átomo	classe
while	palavra reservada	num	identificador
num	identificador	:=	operador
<	operador	num	identificador
50	constante	*	operador
do	palavra reservada	2	constante

Geração do Código Intermediário

A transformação Programa fonte para Programa objeto é brutal.
Deve ser aliviada mediante uma fase intermediária.
O código intermediário deve ser fácil de ser produzido e ser traduzido em código objetivo (assembly da máquina)

Exemplo: while n < 50 do n := n *2

1	<, n, 50, L1
2	JF, L1, 6, -
3	*, n, 2, T1
4	:=, T1, -, n
5	JUMP, 1, -, -
6	……

Código “Quaduplas” ou código de “Três Endereços”
-2 endereços para operandos
-1 endereço para o resultado

Otimização de Código Intermediário

- Elimina operações desnecessárias e repetidas
- Simplifica o código intermediário
- Torna o programa executável mais rápido
- Economiza memória

x := a + b + c

y := a + b + c + d

Código intermediário

+, a, b, T1
+, T1, c, T2
:=, T2, -, x
+, a, b, T3
+, T3, c, T4
+, T4, d, T5
:=, T5, -, y

Código otimizado

+, a, b, T1
+, T1, c, x
+, x, d, y

Geração do Código Objeto

- Transformação do código intermediário otimizado no código objeto (normalmente o Assembly da máquina alvo)
- O código objeto também sofre otimização

while n < 50 do n = n *2

Código intermediário

1	<, n, 50, L1
2	JF, L1, 5, -
3	*, n, 2, n
4	JUMP, 1, -, -
5	……

Código Assembly

E1:	Load n
	Sub 50
	Jz E2
	Jp E2
	Load n
	Mult 2
	Store n
	Jump E1
E2:	……

Os Componentes de Comunicação entre as Fases

- **Manipulador de tabela de símbolos**
- **Tratador de erros**

Tabela de símbolos: guarda informações sobre todos os identificadores usados em um programa

Informações sobre os identificadores:

-Tipo: variável, constant, definição, nome de função, nome de procedimento, rótulo

-Escopo: em que trecho ele é válido

-Endereço de memória e espaço a ser alocado

-No caso de subprogram: número e tipo dos parâmetros, método de passagem de parâmetros, tipo a ser retornado

Os Componentes de Comunicação entre as Fases

A tabela de símbolos é manipulada pelos vários componentes:

-Análise léxica detecta e armazena o identificador

-Análise sintática armazena seu tipo

-Análise semantica consulta a tabela

-Geração de código introduz e usa sobre espaço alocado

Os Componentes de Comunicação entre as Fases

Detecção e Tratamento de Erros:

-Cada componente do compilador pode encontrar erros

-Encontrado um erro, um componente deve tratá-lo de forma a permitir que outros erros sejam detectados

-Existem erros léxicos, sintáticos e semânticos