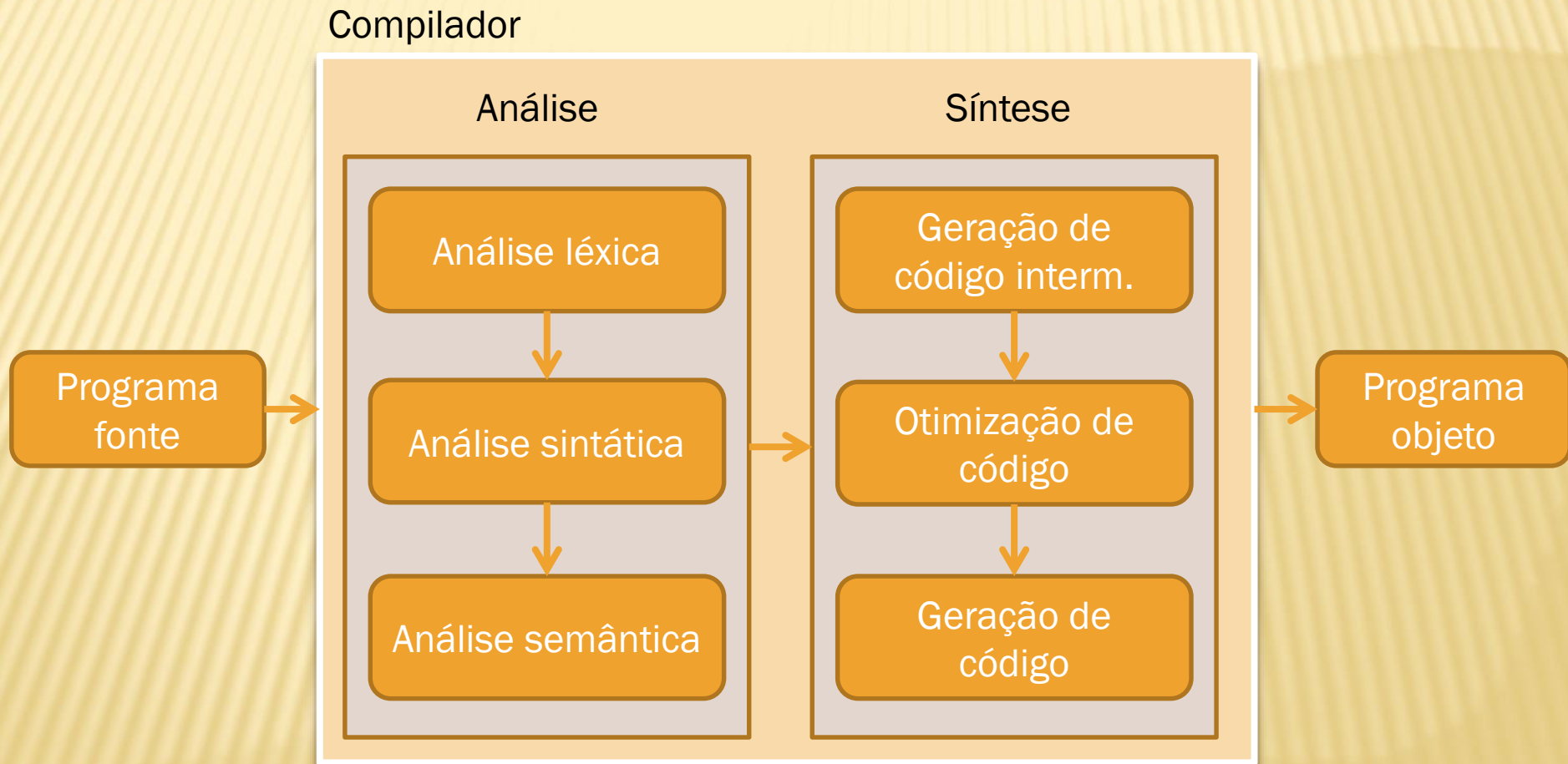


GERAÇÃO DE CÓDIGO INTERMEDIÁRIO

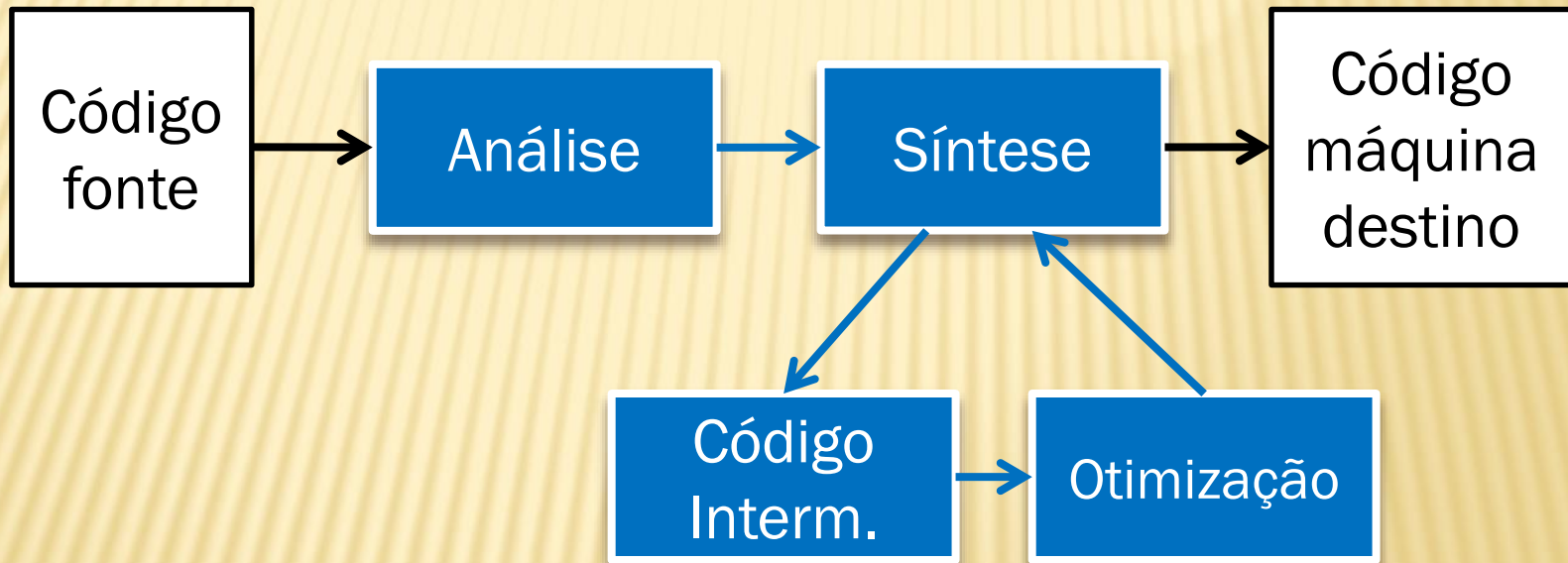
SUMÁRIO

- ✓ Introdução à geração de código intermediário
- ✓ Geração de código intermediário para a LMS:
Início de programa; Declaração de variáveis;
Fim de programa; Uso de constantes
numéricas; Atribuições; Entrada de dados;
Operações aritméticas; Saída de dados; If

O PROCESSO DE COMPILAÇÃO – VISÃO GERAL



POR QUE GERAR CÓD. INTERMEDIÁRIO?



REPRESENTAÇÃO INTERMEDIÁRIA

✓ Alto nível

- ✓ Próxima da linguagem fonte
- ✓ Mais fácil de gerar

✓ Baixo nível

- ✓ Próxima da linguagem da máquina alvo
- ✓ Facilita a alocação de memória e a utilização do conjunto de instruções

REPRESENTAÇÃO INTERMEDIÁRIA

✓ Alto nível

```
void main() {  
    Print("hello world");  
}
```

```
main:  
    BeginFunc 4;  
    _t0 = "hello world";  
    PushParam _t0;  
    LCall _PrintString;  
    PopParams 4;  
    EndFunc;
```

REPRESENTAÇÃO INTERMEDIÁRIA

✓ Baixo nível

```
program abc;  
var  
  i: Integer;  
begin  
  writeln("Hello World");  
end.
```

	Código	Operando1	Operando2
0	AMEM	1	-
1	IMPL	0	-
2	IMPN	-	-
3	PARA	-	-

REPRESENTAÇÃO INTERMEDIÁRIA

- ✓ **Dependente da linguagem**
 - ✓ Semelhante a um pseudo-código
 - ✓ Exemplo: Byte Code Java
- ✓ **Independente de linguagem**
 - ✓ Semelhante a uma linguagem assembly
 - ✓ Código de três endereços

Geração de código intermediário → Representação intermediária

CÓDIGO DE TRÊS ENDEREÇOS

- ✓ Instruções com, no máximo, **3 operandos**
- ✓ **Variações** podem ser utilizadas **com mais ou com menos operandos**
- ✓ Os “endereços” podem ser:
 - ✓ Identificadores
 - ✓ Constantes
 - ✓ Nomes temporários gerados pelo compilador

Geração de código intermediário → Representação intermediária

CÓDIGO DE TRÊS ENDEREÇOS

✓ Exemplo

a = b+c*d;

	Operador	Arg 1	Arg 2	Result.
1	*	c	d	_t1
2	+	b	_t1	a

Geração de código intermediário → Representação intermediária

CÓDIGO DE TRÊS ENDEREÇOS

- ✓ Nomes temporários são usados para computar operações extensas, “quebrando-as” em operações intermediárias
- ✓ Endereços são implementados como ponteiros para o respectivo nome

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

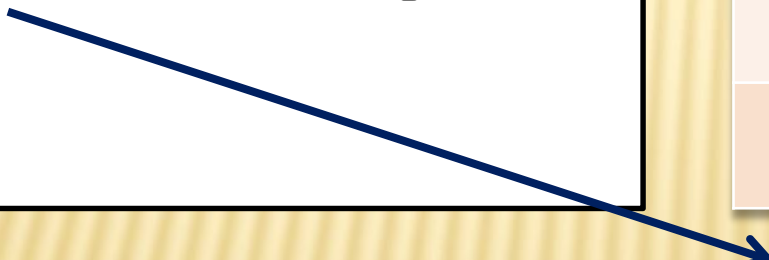
- ✓ Instruções no formato “código de três endereços”, porém, com apenas **2 operandos**

	Instrução	Arg 1	Arg 2
1	AMEM	-	3
2	CALL	2	35

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

- ✓ Início de programa; Declaração de variáveis;
Fim de programa

```
Program Teste1;  
Var  
    a,b : Integer;  
    ...
```



	Instrução	Arg 1	Arg 2
1	AMEM	-	5
2
3	PARA	-	-

Alocação de memória: $3 + N$

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

- ✓ Uso de constantes numéricas
- ✓ Atribuições

```
...  
Var  
    i : Integer;  
Begin  
    i := 2;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	2
3	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Entrada de dados simples

```
...  
Var  
    x : Integer;  
Begin  
    Readln(x) ;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	LEIT	-	-
3	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Entrada de dados composta

```
...  
Var  
    x,y : Integer;  
Begin  
    Readln(x,y) ;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	5
2	LEIT	-	-
3	ARMZ	0	3
4	LEIT	-	-
5	ARMZ	0	4

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Operações aritméticas simples

```
...  
Var  
    x : Integer;  
Begin  
    x := 10 + 20;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	10
3	CRCT	-	20
4	SOMA	-	-
5	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Operações aritméticas simples

```
...  
Var  
    x : Integer;  
Begin  
    x := 10 - 20;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	10
3	CRCT	-	20
4	SUBT	-	-
5	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Operações aritméticas simples

```
...  
Var  
    x : Integer;  
Begin  
    x := 10 * 20;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	10
3	CRCT	-	20
4	MULT	-	-
5	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Operações aritméticas simples

```
...  
Var  
    x : Integer;  
Begin  
    x := 10 / 20;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	10
3	CRCT	-	20
4	DIVI	-	-
5	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Operações aritméticas compostas

```
...  
Var  
    x : Integer;  
Begin  
    x := 10+20*30;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	10
3	CRCT	-	20
4	CRCT	-	30
5	MULT	-	-
6	SOMA	-	-
7	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Operações aritméticas compostas

```
...  
Var  
    x : Integer;  
Begin  
    x := 10*20+30;  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	10
3	CRCT	-	20
4	MULT	-	-
5	CRCT	-	30
6	SOMA	-	-
7	ARMZ	0	3

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Saída de dados simples

```
...  
Var  
    x : Integer;  
Begin  
    x := 123;  
    Writeln(x);  
...
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	123
3	ARMZ	0	3
4	CRVL	0	3
5	IMPR	-	-

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ Saída de dados composta

```
Var
    x,y : Integer;
Begin
    x := 123;
    y := 456;
    Writeln(x,y);
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	5
2	CRCT	-	123
3	ARMZ	0	3
4	CRCT	-	456
5	ARMZ	0	4
6	CRVL	0	3
7	IMPR	-	-
8	CRVL	0	4
9	IMPR	-	-

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ IF

Fim-IF
Início do Else

```
Var
    x : Integer;
Begin
    x := 123;
    If x > 100 Then
        Writeln(1)
    Else
        Writeln(0);
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	123
3	ARMZ	0	3
4	CRVL	0	3
5	CRCT	-	100
6	CMMA	-	-
7	DSVF	-	11
8	CRCT	-	1
9	IMPR	-	-
10	DSVS	-	13
11	CRCT	-	0
12	IMPR	-	-

GERAÇÃO DE CÓDIGO INTERM. PARA A LMS

✓ IF

Fim-IF
Após Else

```
Var
    x : Integer;
Begin
    x := 123;
    If x > 100 Then
        Writeln(1)
    Else
        Writeln(0);
```

	Instrução	Arg 1	Arg 2
1	AMEM	-	4
2	CRCT	-	123
3	ARMZ	0	3
4	CRVL	0	3
5	CRCT	-	100
6	CMMA	-	-
7	DSVF	-	11
8	CRCT	-	1
9	IMPR	-	-
10	DSVS	-	13
11	CRCT	-	0
12	IMPR	-	-