Introdução aos Computadores e Programação

Eng. Química
&
Bioquímica

Bibliografia

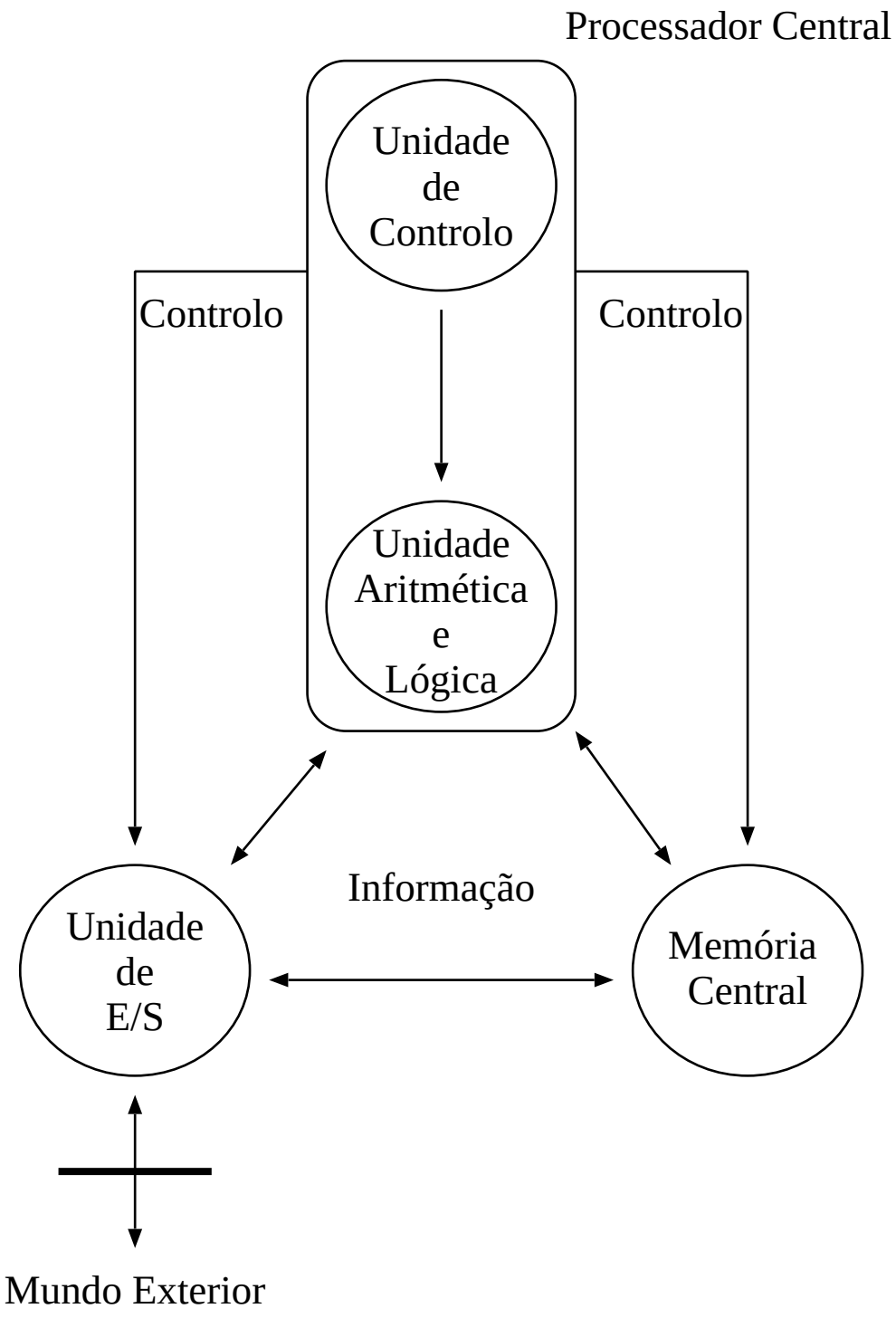
- Stephen J, Chapman, *Introduction to Fortran 90/95*, McGraw-Hill, New York, 1998.
- A. Varandas et. al., *introdução à programação Fortran e cálculo científico*, Minerva, Coimbra, 1994.

Noções Gerais:

- Informática (Teoria da Informação) — Ciência do tratamento e transmissão da informação.
- Computador Digital — Sistema digital que permite armazenar grandes quantidades de informação, e realizar sobre essa informação, a velocidades muito elevadas, manipulações e operações aritméticas e lógicas elementares.

Organização básica de um Computador Digital

Modelo de Von Newmann



- Processador Central (CPU)
  - Unidade Aritmética e Lógica (UAL) — local onde se executam as operações aritméticas e lógicas elementares estipuladas pelos programas;
  - Unidade de Controlo — Extrai da memória as instruções dos programas e os dados, analisa-as e executa a consequente manipulação.
- Memória Central (RAM), local de armazenamento de:
  - programas — sequências de instruções que definem as tarefas a executar e por que ordem;
  - dados — informação sobre a qual se vão executar as operações (manipulações) definidas pelos programas;
  - resultados — informação resultante das operações (manipulações) efectuadas sobre os dados.
- Unidades de Entrada/Saída — unidades especializadas na troca de informação com o exterior. Estas unidades comunicam com os dispositivos periféricos, estabelecendo a ligação entre o mundo exterior e o processador central (ou a memória) e vice-versa.

Periféricos

- Comunicação com o exterior
  - Teclado (entrada)
  - Ecrã (saída)
  - Dispositivos de posicionamento, *"rato"* (entrada)
  - Impressora (saída)
  - Traçadora de curvas (saída)
  - Desmodeladores (entrada/saída)
    - analógico para digital (entrada)
    - digital para analógico (saída)
- Memórias auxiliares
  - Discos Magnéticos (discos fixos, disquetes)
  - Fitas Magnéticas (cartuchos, bobines, cassetes)
  - Discos Ópticos (CDs, DVDs)

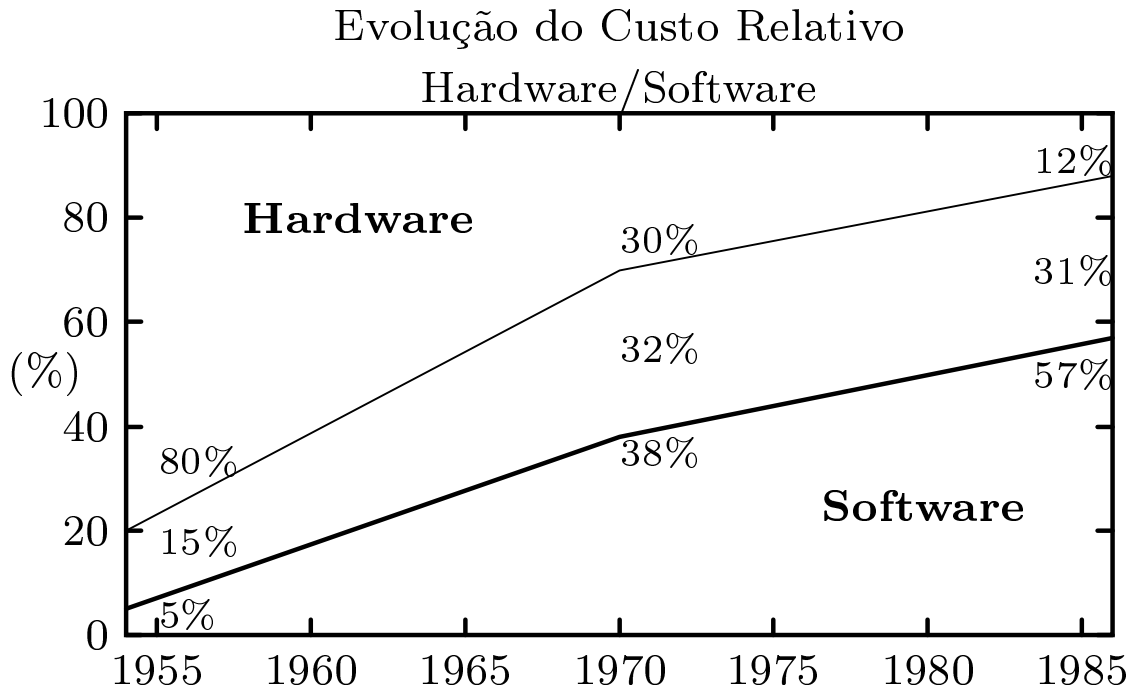
Hardware/Software

Hardware (Equipamento Físico) Dispositivos mecânicos, magnéticos, eléctricos, e electrónicos.

- Hardware
  - Aspecto Tecnológico
  - Aspecto Lógico
  - Aspecto Arquitectónico

Software (programas) Componente Lógica. Domínio da programação.

- Software
  - Do sistema
    - sistema operativo
  - De suporte
    - compiladores
    - editores
    - bibliotecas de rotinas auxiliares
  - De aplicação
    - programas dos utilizadores



Exemplos (hardware):

ENIAC (1946)	área	1400m^2
	Peso	30 Toneladas
	Consumo	150kW
	Velocidade	5000 adições/s
PC 386 (1985)	área	0.5m^2
	Peso	30Kg
	Consumo	590W
	Velocidade	2.43MIPS

Informação

$$\begin{cases} \text{Analógica — variação contínua} \\ \text{Digital — variação discreta} \end{cases}$$

Computadores — Informação Digital Binária

Dois estados diferentes — 0 | 1

O elemento de informação contido na alternativa "O ou 1" designa-se por **Dígito Binário**, em inglês, **B**inary **I**nformation **D**igit (bit).

1 bit	$\rightarrow$	0	2 estados diferentes
		1	
2 bit	$\rightarrow$	0 0	4 estados diferentes
		0 1	
		1 0	
		1 1	
3 bits	$\rightarrow$	...	8 estados diferentes
	$\vdots$		
n bits	$\rightarrow$	...	2^n estados diferentes

Byte (Binary Term): 8 bits.

Palavra: número de bits que constituem as unidades manipuláveis por um determinado computador (número de bits agrupados num elemento endereçável de uma só vez).

Memória: sequência numerada de células, cada uma permitindo o armazenamento de uma palavra. Cada célula é identificável pelo seu endereço.

A capacidade de memória mede-se pelo número de palavras que consegue armazenar:

$$
\begin{array}{lcll}
1024 & = & 1\text{K} & \\
2^{16} & = & 2^6 \times 2^{10} & = \quad 64\text{K} \\
2^{20} & = & 2^{10} \times 2^{10} & = \quad 1\text{M (1 Mega)} \\
2^{30} & = & 2^{20} \times 2^{10} & = \quad 1\text{G (1 Giga)}
\end{array}
$$

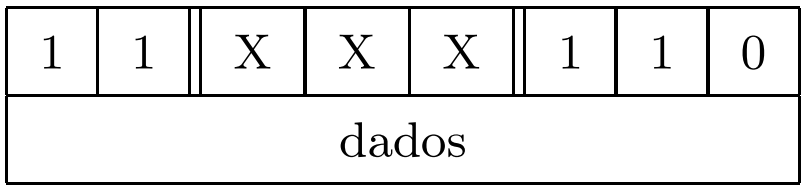
Representação das várias entidade manipuláveis, exemplo para um processador de 8bits (Z80).

representação interna			possíveis interpretações
1 0 0 0 0 1 1 0	$\xrightarrow{1}$	ADD M	instrução para executar uma adição.
	$\xrightarrow{2}$	134	número inteiro (representação, valor absoluto)
	$\xrightarrow{3}$	à	caracter (extensão à tabela ASCII)

1. Conjunto de instruções que constituem a "linguagem máquina" de um dado processador.
2. Representação de número através da conversão de bases ($10000110_2 = 134_{10}$).
3. Representação de caracteres através de uma tabela de conversão ($10000110_2 = 134_{10} \overset{\text{ASCII}}{=}$ à).

"Linguagem Máquina" — Cada micro-processador tem um conjunto de instruções que constituem a sua "linguagem", para facilitar a sua leitura existe uma (primeira) interpretação na linguagem simbólica, que é também própria a cada micro-processador.

Por exemplo: instruções aritmética e lógicas imediatas para os micro-processadores P8080 e P8085.



x x x	Instrução		
	simbólica	mnemónica	hex.
0 0 0	(A) ← (A) + dados	ADI ,dados	C 6
0 0 1	(A) ← (A) + (cy) + dados	ACI ,dados	C E
0 1 0	(A) ← (A) - dados	SUI ,dados	D 6
0 1 1	(A) ← (A) - (cy) - dados	SBI ,dados	D E
1 0 0	(A) ← (A) $\wedge$ dados	ANI ,dados	E 6
1 0 1	(A) ← (A) $\dot{\vee}$ dados	XRI ,dados	E E
1 1 0	(A) ← (A) $\vee$ dados	ORI ,dados	F 6
1 1 1	(A) - dados	CPI ,dados	F E

Caracteres — Tabela ASCII.

	0	1	2	3	4	5	6	7	8	9
00x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
01x	LF	VT	FF	CR	SO	SI	DBL	DC1	DC2	DC3
02x	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
03x	RS	US	SP	!	"	#	$	%	&	´
04x	(	)	*	+	,	-	.	/	0	1
05x	2	3	4	5	6	7	8	9	:	;
06x	<	=	>	?	@	A	B	C	D	E
07x	F	G	H	I	J	K	L	M	N	O
08x	P	Q	R	S	T	U	V	W	X	Y
09x	Z	[	\	]	^	_	`	a	b	c
10x	d	e	f	g	h	i	j	k	l	m
11x	n	o	p	q	r	s	t	u	v	w
12x	x	y	z	{	|	}	~	DEL		

Informação Numérica $\left\{ \begin{array}{l} \text{Inteiros} \\ \text{Reais} \end{array} \right.$

Inteiros — A conversão é exacta.

Conversão base 2 para base 10

$$\begin{aligned} 10000110_2 &= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + \\ &\quad +0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 134_{10}(1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0) \end{aligned}$$

Conversão de base 10 para base 2. Dividir por 2, reter o resto e repetir tudo até não ser possível realizar a divisão, o número obtêm-se "lendo" os restos por ordem inversa.

dividendo	resto	
134	0	
67	1	
33	1	
16	0	
8	0	
4	0	
2	0	
1	1	$\uparrow$

Representação Interna.

representação **sinal e valor absoluto**

1 bit	7 bits						
X	X	X	X	X	X	X	X
Sinal	Valor						

Por exemplo:

1	0	0	0	0	1	1	0	= -6

Gama de Variação.

1	1	1	1	1	1	1	1	=	-127
				⋮					
1	0	0	0	0	0	0	0	=	-0
0	0	0	0	0	0	0	0	=	0
				⋮					
0	1	1	1	1	1	1	1	=	127

- A representação é exacta.
- As operações não induzem em erros (excepto no que diz respeito à ultrapassagem da capacidade).
- A gama de variação é da ordem de -32768 a 32767 (16 bits).

Reais — A conversão **não** é (em geral) exacta.

$$
\begin{aligned}
23.48_{10} & \\
23_{10} &= 10111_2 \\
0.48_{10} &= 0.01111010\ldots_2
\end{aligned}
$$

A conversão da base 10 para a base 2 da parte decimal:
multiplica-se por dois e retêm-se a parte inteira,
repete-se todo o processo até se obter zero no resultado.
Obtêm-se o número fazendo a leitura directa das partes inteiras que se foram obtendo.

Por exemplo: $0.1_{10} = 0.0(0011)_2$

↓	0	0.1
		×2
	0	0.2
		×2
	0	0.4
		×2
	0	0.8
		×2
	1	1.6
		0.6
		×2
	1	1.2
		0.2
		×2
	0	0.4
		⋮

Representação Vírgula Flutuante – representação usada para minimizar os erros de representação.

$$x = m \times B^e, \quad -E < e < E \quad -M < m < M$$

m – Mantissa; B – Base; e – Expoente.

Na forma canónica tem-se

$$\frac{M}{B} \leq m < M$$

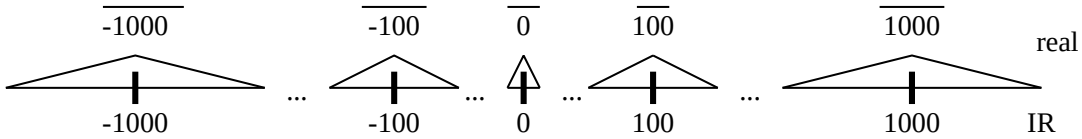
Para $B = 10, M = 1$ tem-se $0.1 \leq m < 1$.

Por exemplo: $23.48 = 0.2348 \times 10^2$

Representação Interna ($4 \times 8 = 32$bits).

expoente	mantissa
8 bits	24 bits

- A representação **não** é, em geral, exacta.
- As operações podem induzir mais erros (além da ultrapassagem da capacidade).
- Na forma normal a densidade de representantes decresce exponencialmente com o crescimento de $|x|$.



- O número de "bits" reservado para a mantissa dá o grau de precisão que uma dada representação interna possui.
- O número de bits reservado para o expoente dá a extensão da gama representada.

Conclusão:

$$\texttt{real} \subsetneq \mathbb{R}$$

– Gama de variação finita.

– Representação discreta.

Programação — Transformar Problemas em Programas.

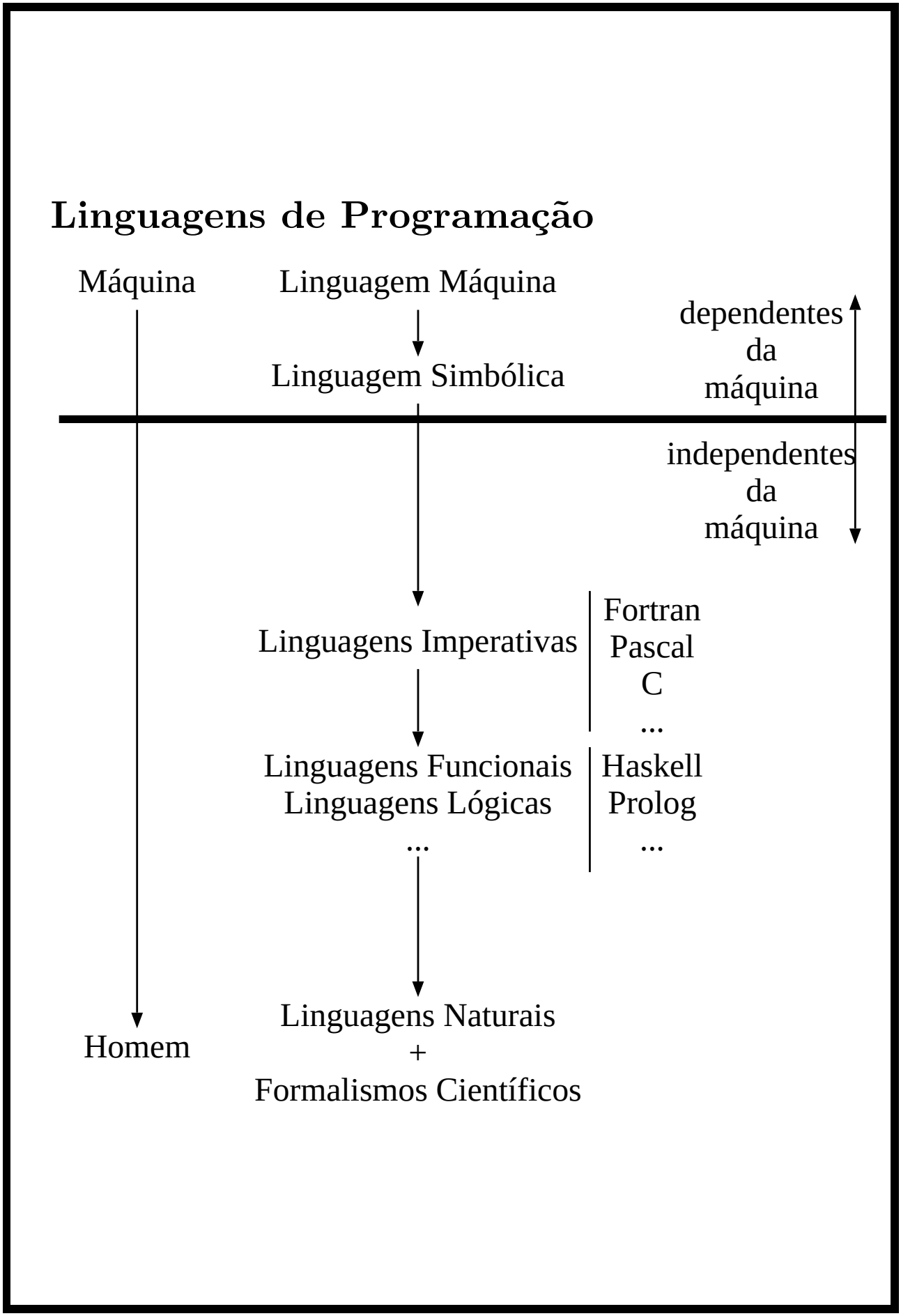
Programas

=

Estruturas de Dados + Algoritmos

Estrutura de Dados: Um Conjunto de Valores e um Conjunto de Operações sobre esses valores.

Algoritmo: Procedimento mecânico (automático) que produz sempre o mesmo resultado ao fim de um número finito de passos.


Linguagens de Programação
Máquina
Linguagem Máquina
Linguagem Simbólica
dependentes da máquina
independentes da máquina
Linguagens Imperativas
Fortran
Pascal
C
...
Linguagens Funcionais
Linguagens Lógicas
...
Haskell
Prolog
...
Homem
Linguagens Naturais
+
Formalismos Científicos

Linguagem Máquina e Linguagem Simbólica ("Assembler")

Excerto de uma rotina na linguagem simbólica do processador Z80.

Por colunas temos:

Nome da rotina; Instrução; Argumento; Comentário.

```
INSE: MVI  A,02H   ;SALTA DUAS LINHAS
      CALL MDLH
      LXI  D,INARG ;ESCREVE MENSAGEM "INTRODUZA ARGUMENTO ..."
      CALL PSTR
      CALL LIMPB   ;VAI "LIMPAR" O BUFFER PARA NAO HAVER INTRODUCAO DE LIXO
      LXI  H,BUFI  ;BUFFER PARA RESPOSTA COM CARG OCTETOS
      LDA  CARG
      MOV  M,A
      XCHG
      CALL RSTR    ;CHAMA SUBROTINA DE LEITURA
      LDA  ORDE
      CPI  'N'
      CZ   ALIN    ;VERIFICA E ALINHA A ENTRADA NO CASO NUMERICO
      MVI  A,02H   ;MUDA DE LINHA
      CALL MDLH
      LXI  D,INVAL ;ESCREVE MENSAGEM "INTRODUZA VALOR..."
      CALL PSTR
      LXI  H,BUFI  ;HL:=BUFI+CARG+2 ISTO E A ZONA DE MEMORIA O
      LDA  CARG    ;VALOR E IMEDIATAMENTE A SEGUIR A DO ARGUEMENTO
      ADI  02H

(...)
```

Vantagens e desvantagens da utilização de uma linguagem simbólica.

+ Há uma correspondência biunívoca entre instruções em linguagem simbólica e instruções em linguagem máquina.
+ Muito eficientes (se bem utilizadas).
+ Total controlo da máquina.
– Programas dependentes do processador central utilizado.
– Linguagens de difícil utilização, tanto para programar, como para modificar ou compreender um programa já existente.
– Programação muito sujeita a erros.

Linguagens de Alto Nível

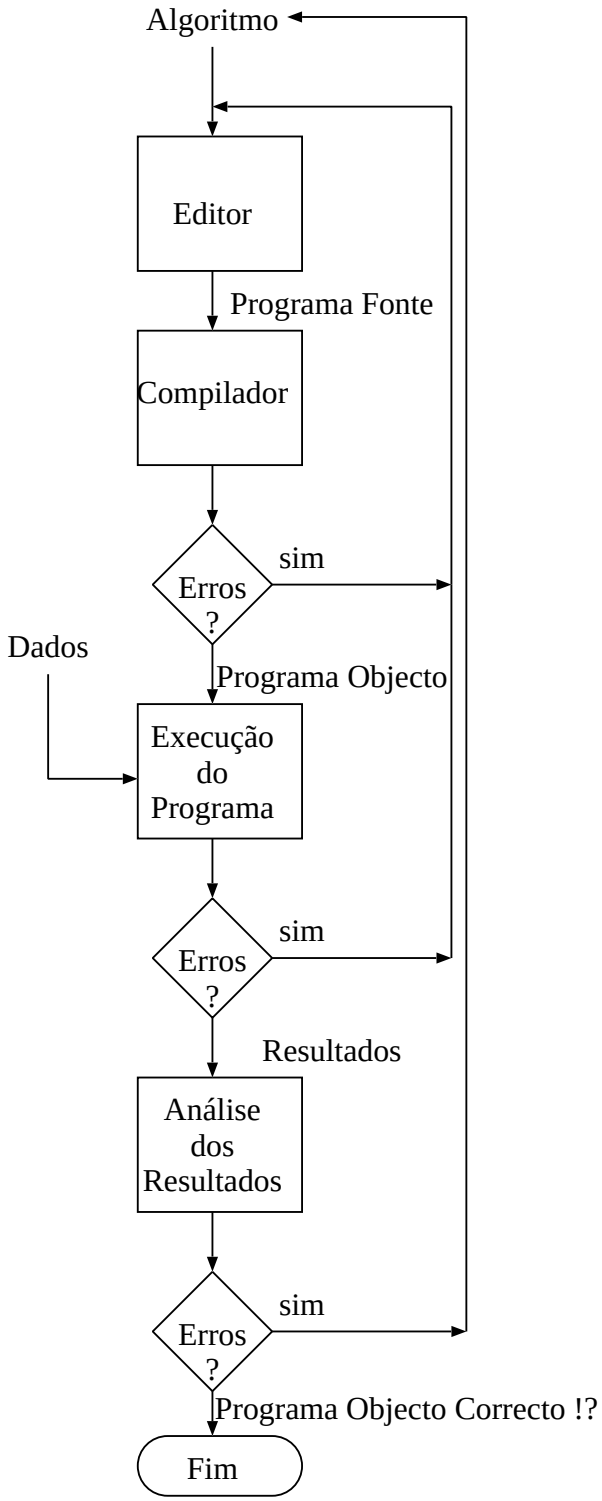
Exemplo (em Fortran): $y = |x|$

```
IF (x>=0) THEN
   y=x
ELSE
   y=-x
END IF
```

Vantagens e desvantagens da utilização de uma linguagem de alto nível.

+ Mais próximas dos conceitos expressos em linguagem natural e/ou científica.
+ Programas independentes do processador central utilizado.
+ Linguagens de fácil (razoavelmente fácil) utilização, tanto para programar, como para modificar ou compreender um programa já existente.
– Não há uma correspondência biunívoca entre instruções em linguagem de alto nível e instruções em linguagem máquina.
– Menos eficientes.
– Menor (ou nenhum) controlo da máquina.

Ciclo de desenvolvimento de um programa



Um exemplo de resolução de um problema através da escrita de um programa em Fortran

Problema 1 *Dado o instante de partida de um avião (horas, minutos,segundos), e dado o seu tempo de voo (segundos), calcule o instante de chegada.*

1º Passo - Especificação fazer uma descrição do problema mais rigorosa em termos dos dados (entradas), e dos resultados (saídas).

dados instante de partida (hh,mm,ss); tempo de voo (ss)

resultados instante de chegada (hh,mm,ss)

¿ e se houver mudança de dia?

¿ e as diferenças de fusos horários?

Correcções na especificação inicial

dados instante de partida (hh,mm,ss); tempo de voo (ss)

resultados instante de chegada (dd,hh,mm,ss)

dd - dá conta da mudança (caso haja) de dia.

não se vai ter em conta as diferenças dos fusos horários.

2º passo - Estrutura de dados discutir o tipo de informação (tipos de dados) que se vai manipular.

dd,hh,mm,ss,tv - quatro valores naturais.

Conclusão - vai-se usar o tipo de dados "INTEGER" (Inteiros).

3º Passo - Algoritmo Descrição dos passos a dar em ordem a resolver o problema.

1. Ler dados (hh,mm,ss,tv)
2. Calcular o resultado
   dd,hh',mm',ss' ← hh,mm,ss+tv
3. Escrever o resultado

Particularizando o segundo passo

2. Calcular o resultado
   (a) cálculo dos segundos
   ss' ← resto da divisão inteira de
   (ss+tv)/60
   (b) cálculo dos minutos
   mm' ← resto da divisão inteira de
   (mm+quocs)/60
   com quocs o resultado da divisão inteira
   (ss+tv)/60
   (c) cálculo das horas
   hh' ← resto da divisão inteira de
   (hh+quocm)/24
   com quocs o resultado da divisão inteira
   (mm+quocs)/60
   (d) cálculo dos dias
   dd ← resultado da divisão inteira
   (hh+quocm)/24

Particularizando o terceiro passo

se dd=0 então escrever(hh',mm',ss')
senão escrever(dd,hh',mm',ss')

Temos então

ler(hh,mm,ss)
ler(tv)
ssl ← mod((ss+tv),60)
quocs ← (ss+tv) / 60
mml ← mod((mm+quocs),60)
quocm ← (mm+quocs) / 60
hhl ← mod((hh+quocm),24)
dd ← (hh+quocm) / 24
se dd = 0 então escrever (hhl,mml,ssl)
senão escrever(dd,hhl,mml,ssl)
fimse

O Programa

```
PROGRAM aviao

IMPLICIT NONE

INTEGER :: hh,mm,ss,tv ! Horas, Minutos, Segundos, Tempo de Viagem
INTEGER :: dd,hhl,mml,ssl ! Dias, Horas', Minutos', Segundos'
INTEGER :: quocs,quocm ! Quociente segundo, Quoc. min.

! Leitura dos Dados
WRITE (*,*) 'Diga qual a hora de descolagem (hh,mm,ss): '
READ (*,*) hh,mm,ss
WRITE (*,*) 'Diga qual o tempo de viagem (ss): '
READ (*,*) tv

! Cálculo
ssl = MOD((ss+tv),60)
quocs = (ss+tv) / 60
mml = MOD((mm+quocs),60)
quocm = (mm+quocs) / 60
hhl = MOD((hh+quocm),24)
dd = (hh+quocm) / 24

! Escrita
IF (dd == 0) THEN
   WRITE (*,*) 'Tempo de Chegada (hh,mm,ss) :'
   WRITE (*,*) hhl,mml,ssl
ELSE
   WRITE (*,*) 'Tempo de Chegada (dd,hh,mm,ss) :'
   WRITE (*,*) dd,hhl,mml,ssl
END IF
END PROGRAM
```

Exemplos de Utilização:

```
> ./exemplo1
Diga qual a hora de descolagem (hh,mm,ss):
1 0 0
Diga qual o tempo de viagem (ss):
3745
Tempo de Chegada (hh,mm,ss) :
          2          2          25

> ./exemplo1
Diga qual a hora de descolagem (hh,mm,ss):
23 12 56
Diga qual o tempo de viagem (ss):
10892
Tempo de Chegada (dd,hh,mm,ss) :
          1          2          14          28
```

Programação

Objectivos a atingir $\begin{cases} \text{Correcção} \\ \text{Clareza} \\ \text{Eficiência} \end{cases}$

Pretende-se obter um programa **correcto** que seja **fácil de compreender** e/ou **modificar**, mesmo que seja por uma programador diferente daquele que o escreveu.

Além disso entre duas soluções, ambas correctas e escritas de uma forma clara escolhe-se, naturalmente, a **mais eficiente**.

Correcção: O Programa deve estar de acordo com a Especificação.

- Simplicidade.
- Aproximação sistemática — Metodologia de Programação Estruturada e descendente.
- Testes e/ou Verificação formal.

Clareza: O Programa deve reflectir a estrutura do algoritmo, e além disso, deve ser fácil de ler.

- Separação em blocos funcionais.
- Uso adequado das estruturas da linguagem.
- Escolha cuidadosa dos identificadores.
- Documentação interna.
- Aspecto gráfico:
  - linhas em branco;
  - indentação.

Eficiência: Comparação entre várias soluções. Avaliada em termos de:

- tempo gasto pelo computador;
- espaço de memória usado.