UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Fundamentos de Programação

CP41F

Fundamentos de computação. Linguagens de alto nível. Compilador/Interpretador. História e Introdução a C.

Aula 3

Prof. Daniel Cavalcanti Jeronymo

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação – 1º Período
2016.1

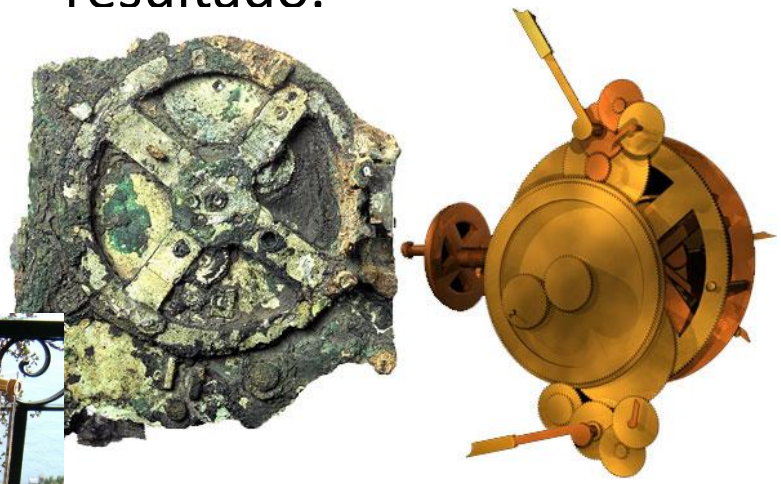
Plano de Aula

- Fundamentos de computação.
  - Computador
  - Analógico
  - Digital
  - Arquitetura
  - Memória
  - Codificação
  - Processadores
- Linguagens de alto nível.
- Compilador/Interpretador.
- História e Introdução a C.

Fundamentos de Computação

- **Computador:**

Dispositivo que recebe informações e a manipula utilizando sequências de instruções, os programas, para obter um resultado.



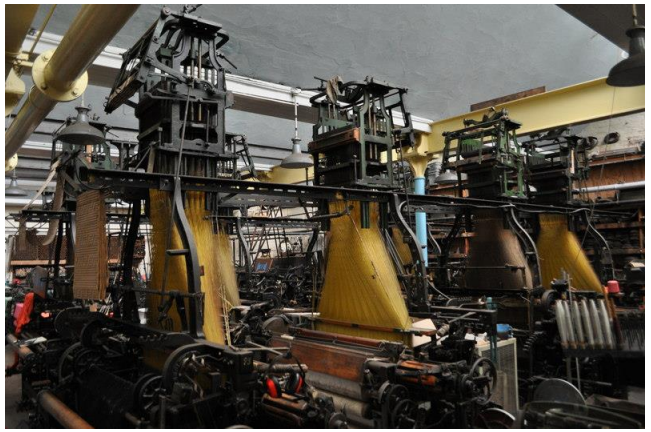
Dispositivo de Antikhytera: computador de posições astronômicas e eclipses (~200 a.C.)

Autômatos de Jaquet-Droz – A Escritora: memória de quarenta caracteres (~1700)

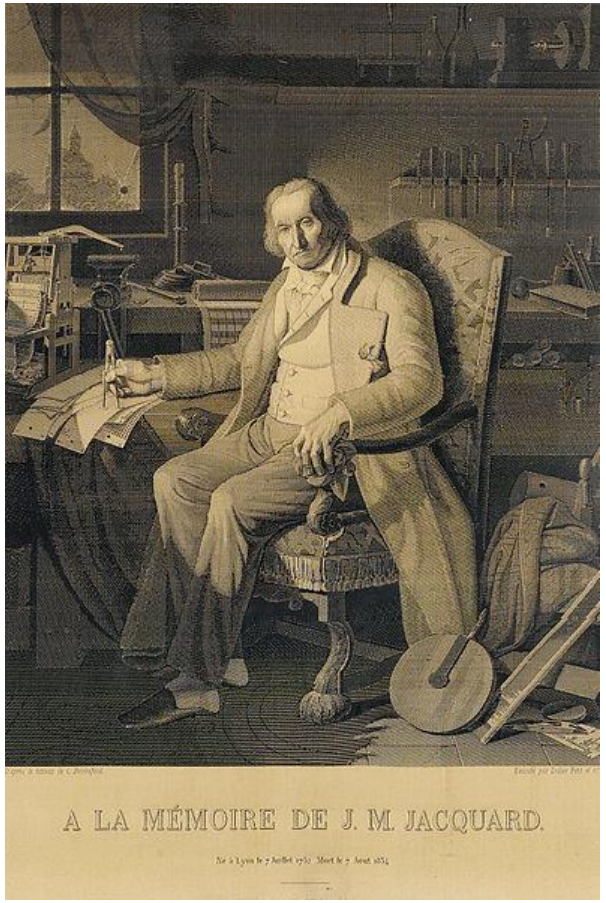
Computador de marés: previsão da altura das marés (~1800, Lord Kelvin)

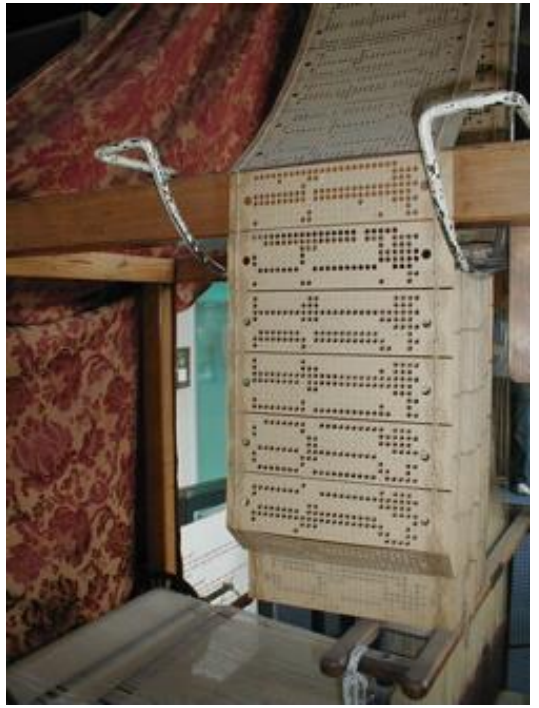
Fundamentos de Computação

- Tear de Jacquard: utilizava cartões perfurados para programação dos padrões (1839).



O retrato de Jacquard foi tecido em seda usando uma de suas máquinas, necessitando de 24.000 cartões perfurados.

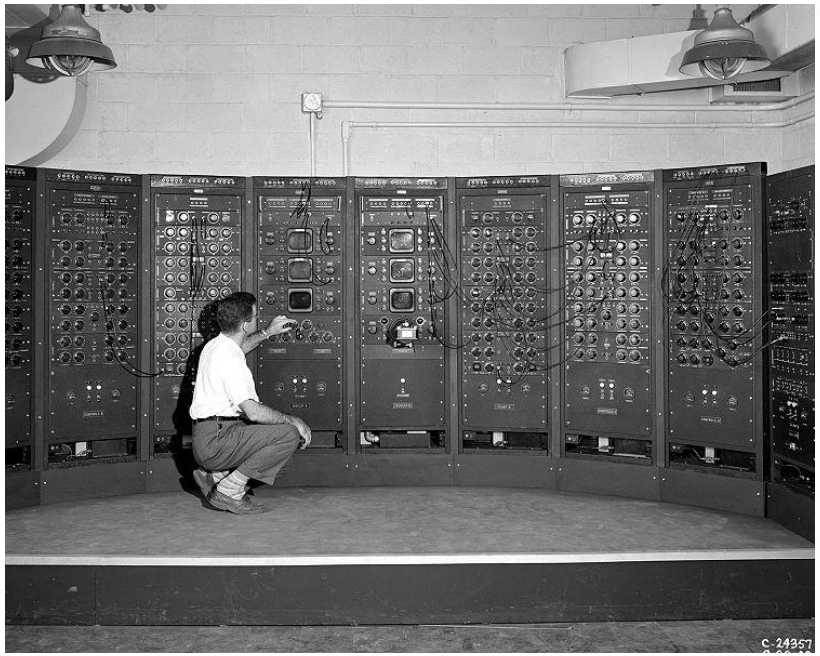




Fundamentos de Computação

- **Computador analógico (eletrônico):**

  No lugar de eixos mecânicos estes computadores empregavam conexões elétricas (~1900).

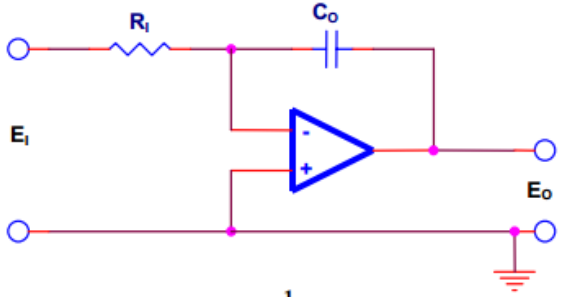


Fundamentos de Computação

- **Computador analógico (eletrônico):**

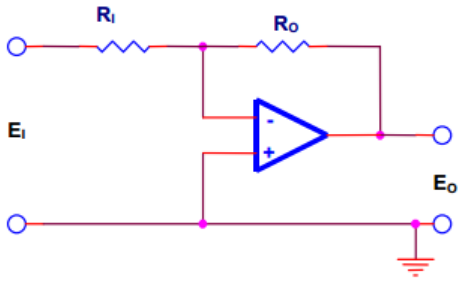
Utilizando amplificadores operacionais.

Integrator



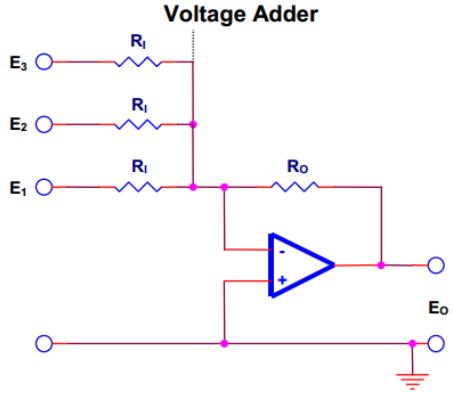
$$E_O = \frac{-1}{R_I C_O} \int E_I dt$$

INVERTING AMPLIFIER



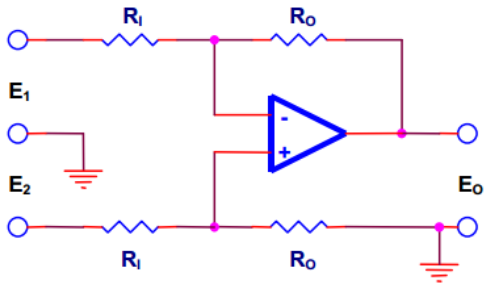
$$\frac{E_O}{E_I} = \frac{-R_O}{R_I}$$

Voltage Adder



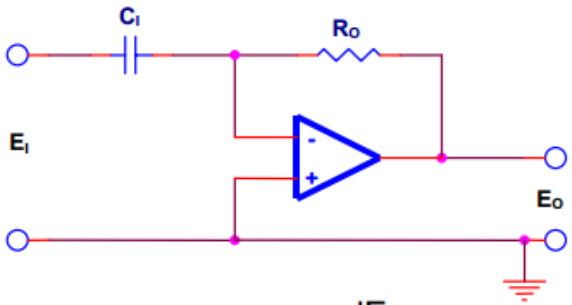
$$E_O = \frac{-R_O}{R_I}(E_1 + E_2 + E_3 + \ldots)$$

Differential Input Amplifier



$$E_O = \frac{R_O}{R_I}(E_2 - E_1)$$

Differentiator



$$E_O = -R_O C_I \frac{dE_I}{dt}$$

TEXAS INSTRUMENTS

Application Report
SBOA092A –October 2001

HANDBOOK OF OPERATIONAL AMPLIFIER APPLICATIONS

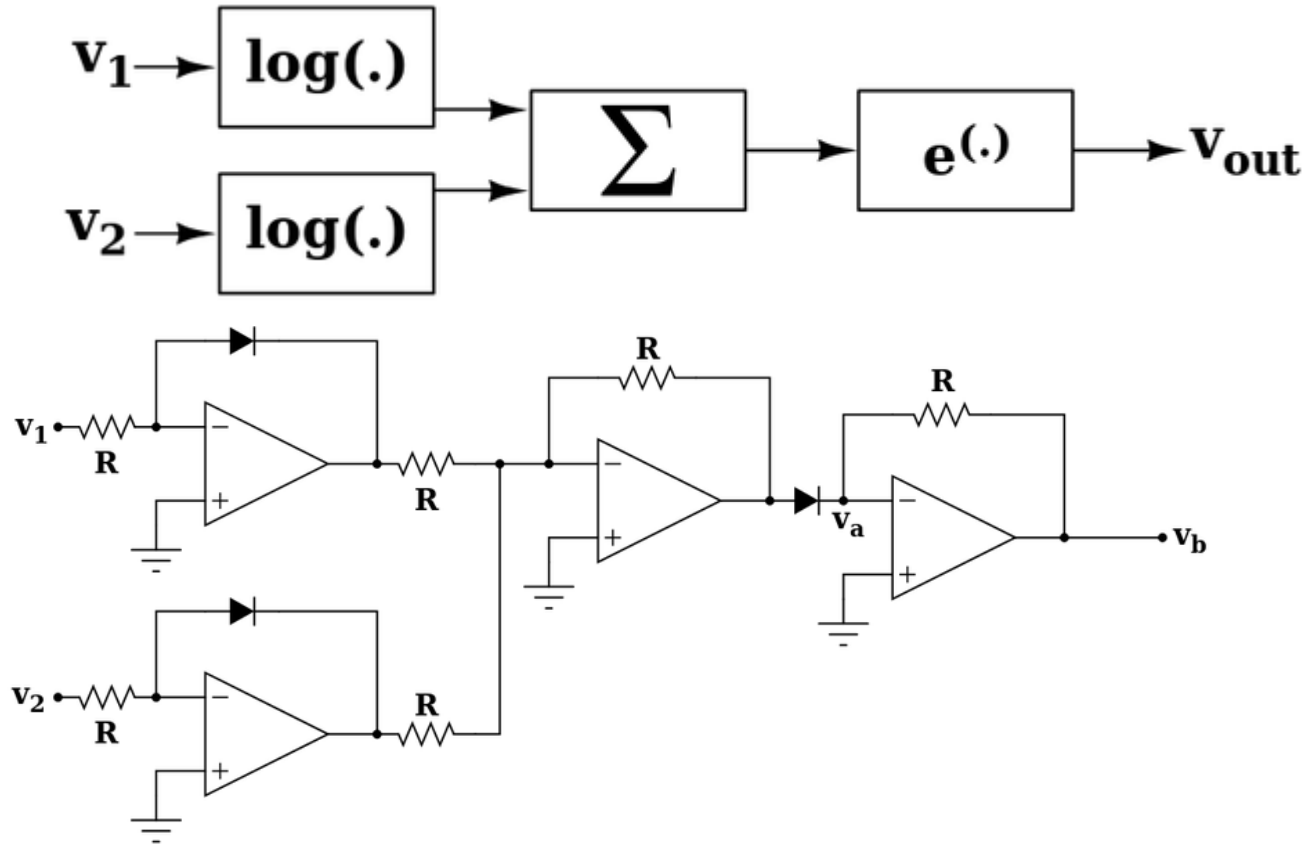
Bruce Carter and Thomas R. Brown

Fundamentos de Computação

- **Computador analógico (eletrônico):**

Multiplicação usando a cabeça...

$$\log(a \cdot b) = \log a + \log b$$



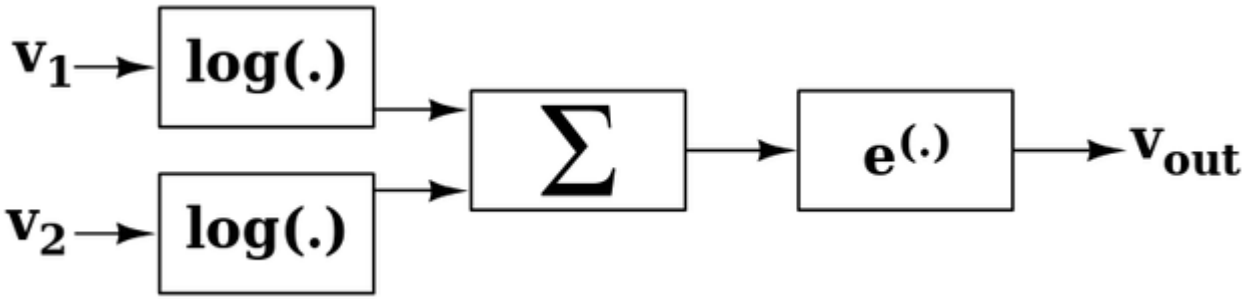
$$-\frac{v_1 \cdot v_2}{RI_s} - (v_1 + v_2)$$

Fundamentos de Computação

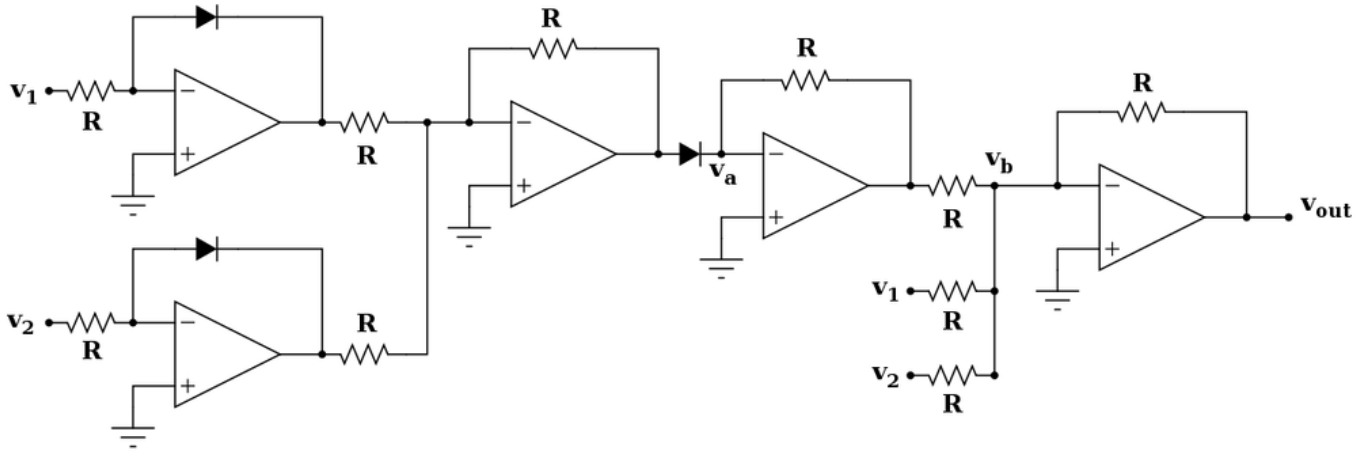
- **Computador analógico (eletrônico):**

Multiplicação usando a cabeça... e adicionando um termo.

$$\log(a \cdot b) = \log a + \log b$$



Como fazer divisão?
Considere o logaritmo!

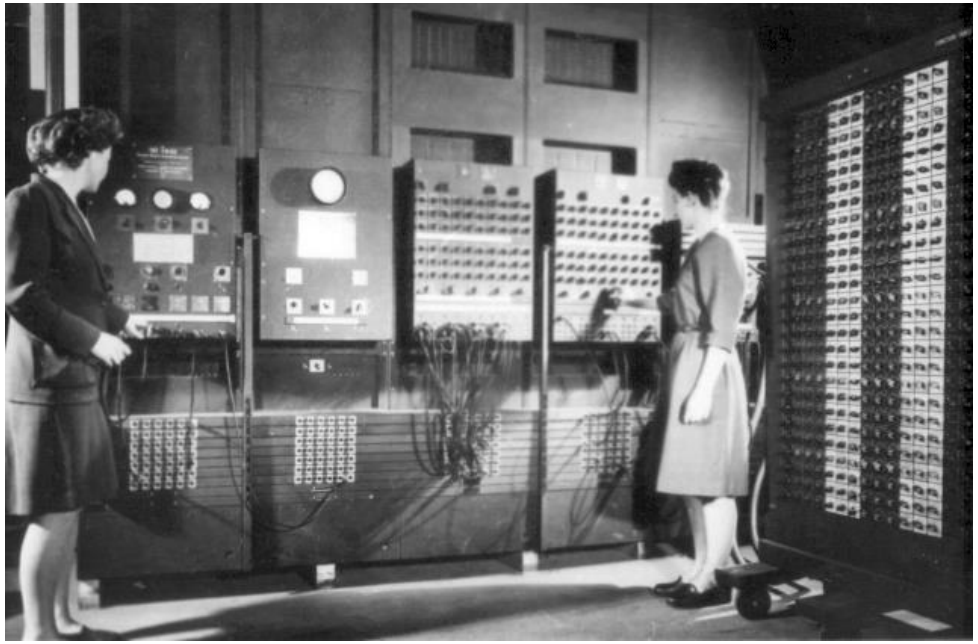


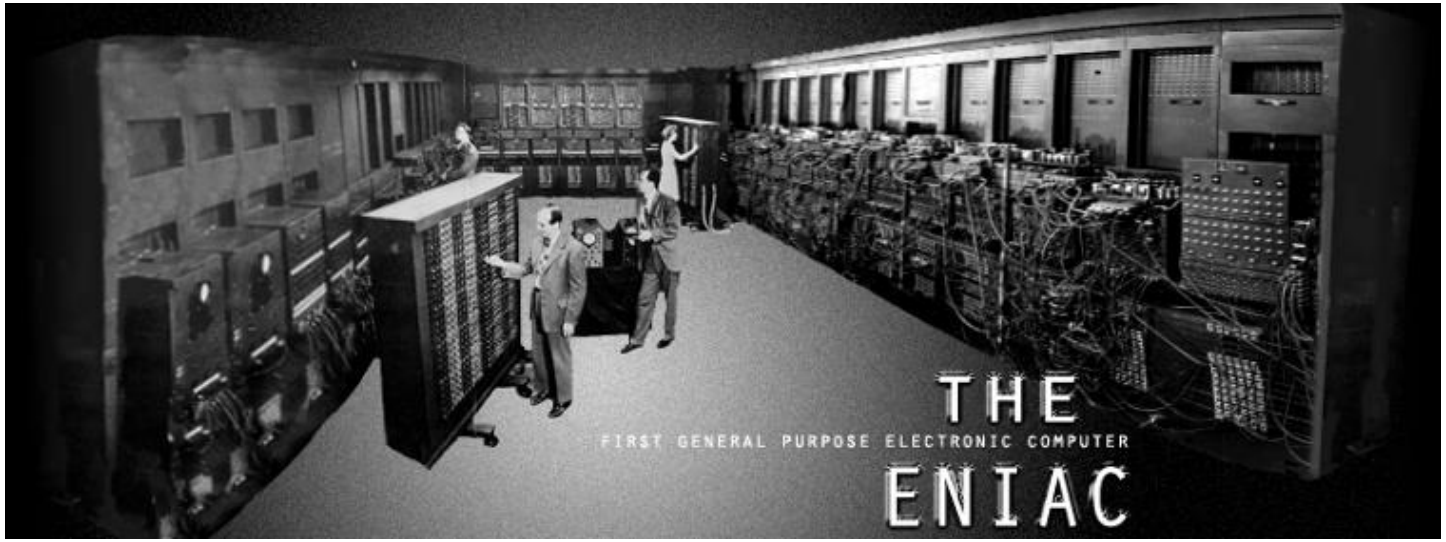
$$\frac{v_1 \cdot v_2}{RI_s}$$

Fundamentos de Computação

- **Computador digital:**

Circuitos substituídos por válvulas...





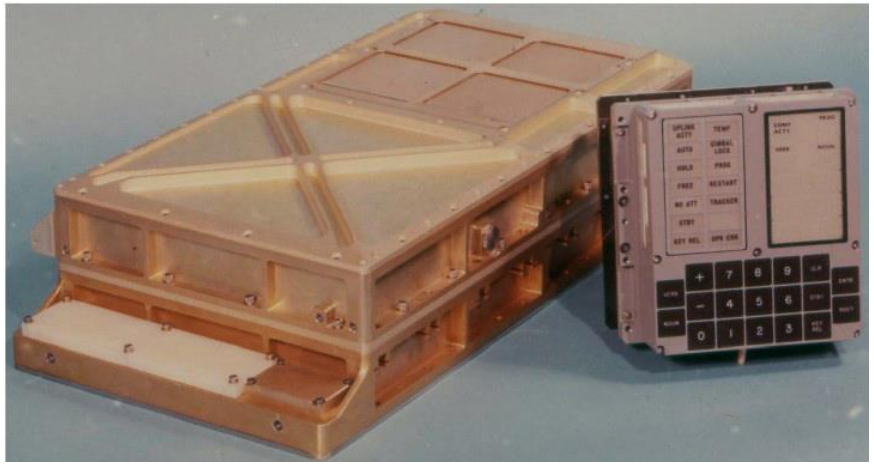
ENIAC - **E**lectronic **N**umerical **I**ntegrator **A**nd **C**omputer: Computador para cálculo de trajetórias balísticas. (~1945)

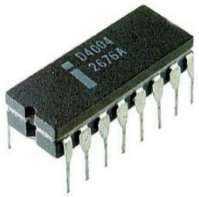
Fundamentos de Computação

- **Computador digital:**

Circuitos substituídos por válvulas... e posteriormente transistores.

IBM 608 – Calculadora Transistorada (1955)



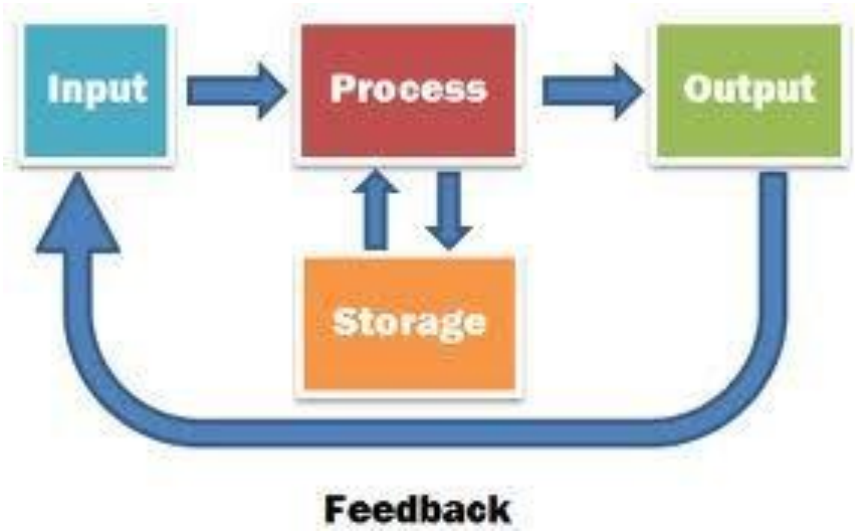


Intel 4004 (1971): primeiro processador, após o chute inicial dos programas Apollo e Minuteman na indústria de circuitos integrados.

Fundamentos de Computação

- **Computador digital (moderno):**

  Ciclo IPOS (*Input, Processing, Output, Storage*)

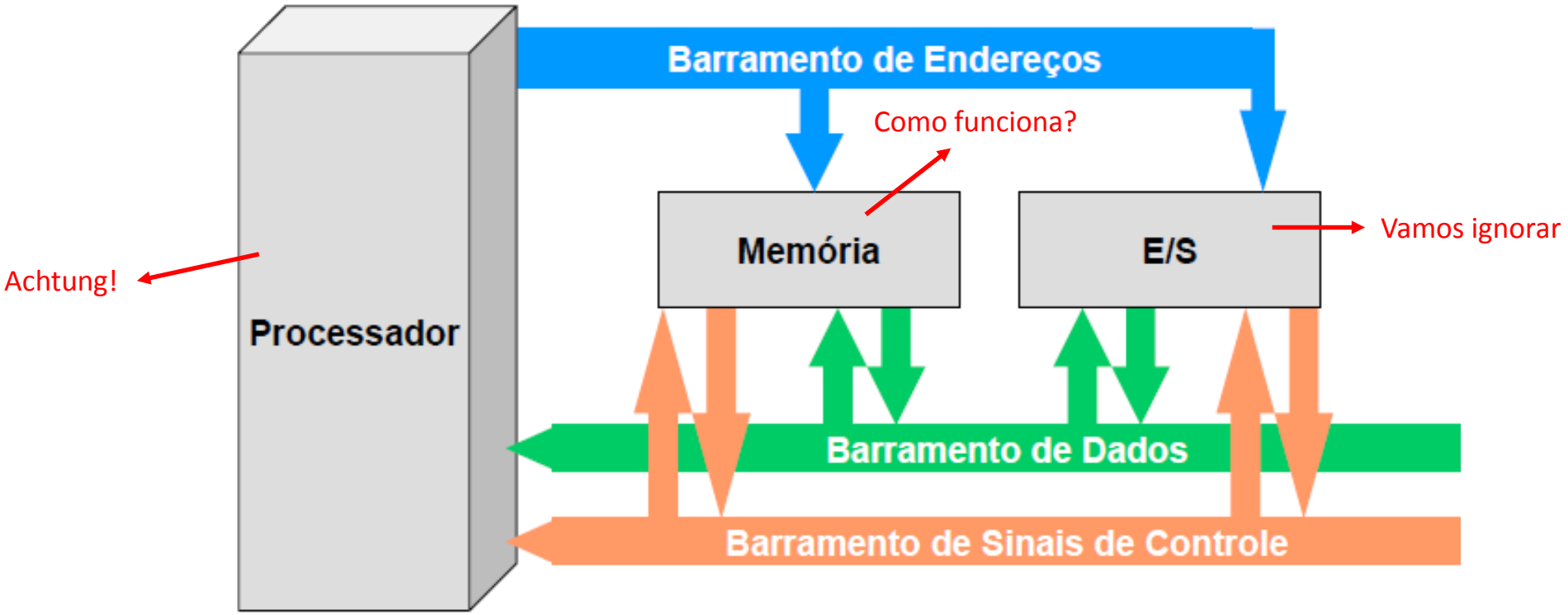


Sistema de Computação: integração de componentes numa única entidade. *peopleware + hardware + software*

componentes: humana + física + lógica

Fundamentos de Computação

- **Visão geral da arquitetura de computadores:**

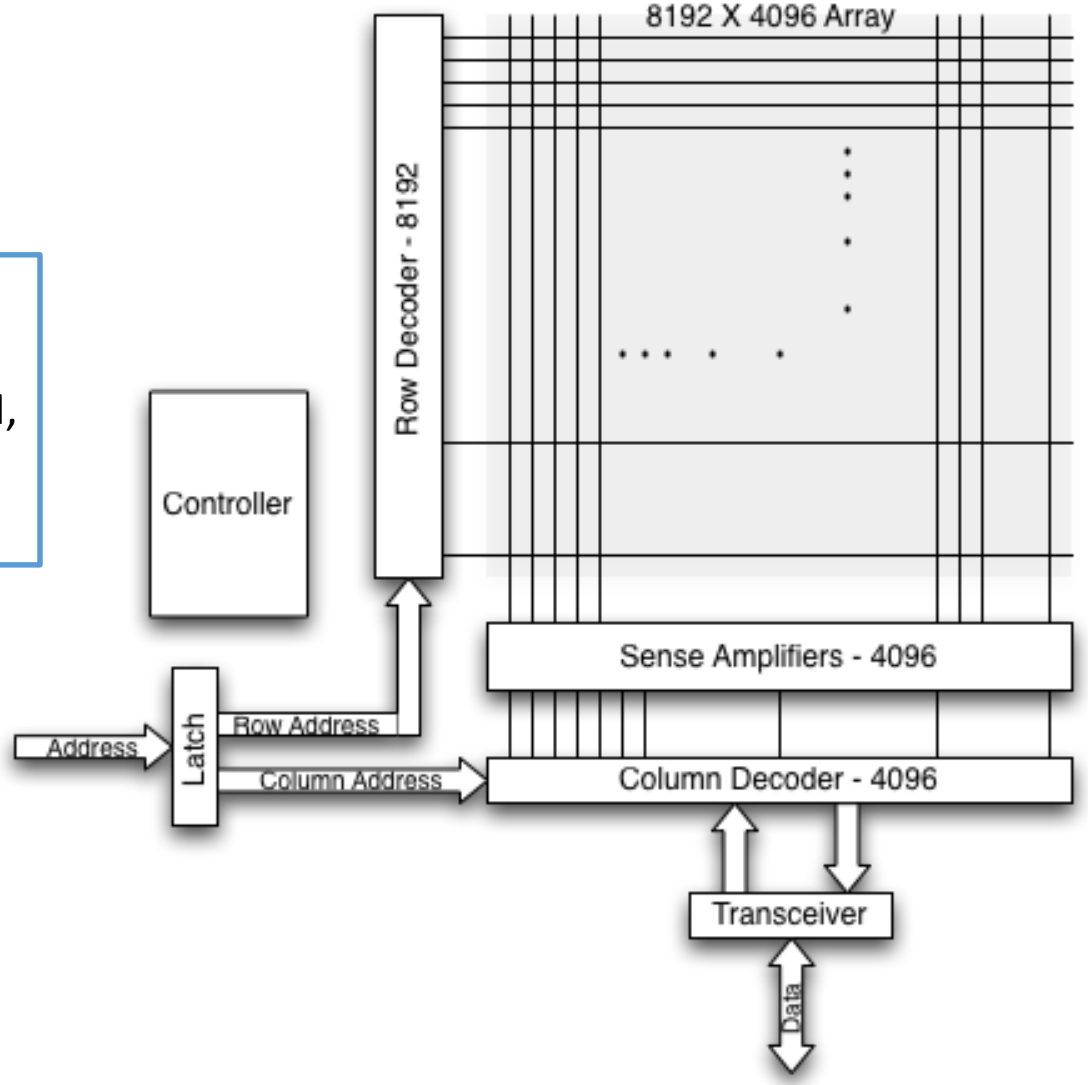


Fundamentos de Computação

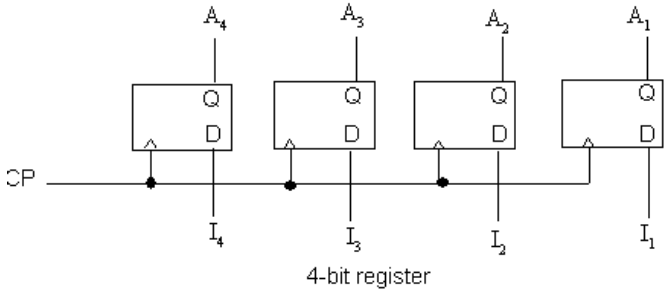
- **Visão geral da memória:**

Matriz de memória

Cada intersecção liga uma célula de memória, há diferentes compromissos entre diferentes tipos de células, SRAM, DRAM, ROM, etc (densidade, velocidade e volatilidade)



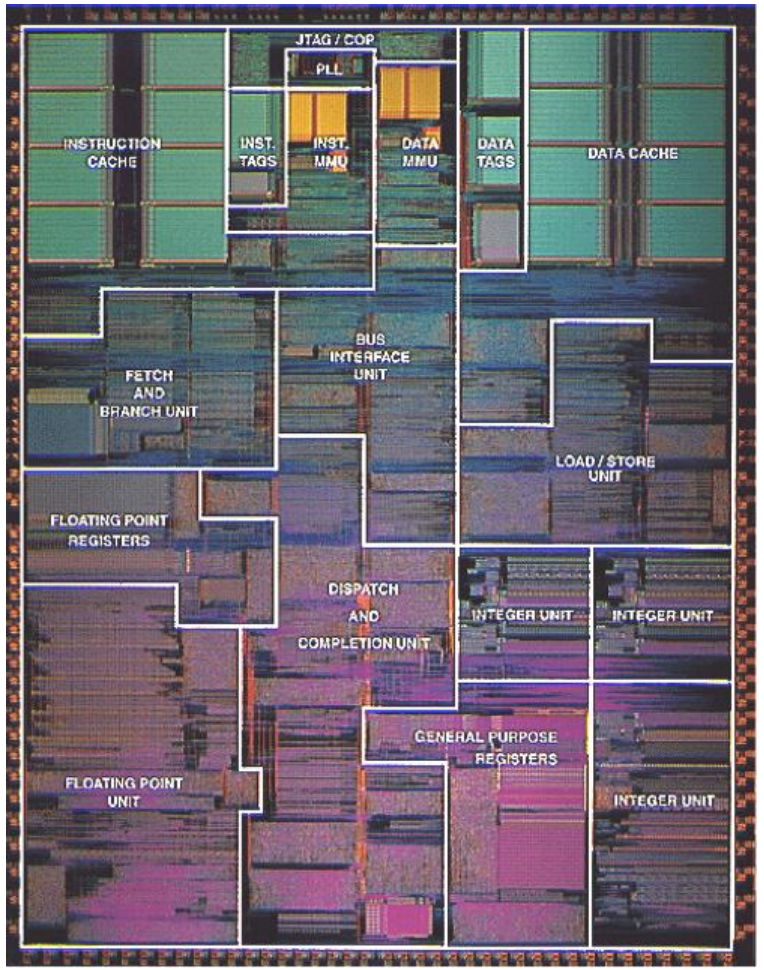
Registrador – o primo isolado



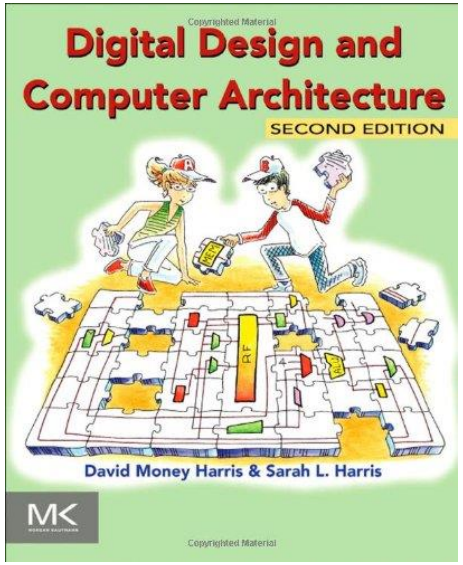
Fundamentos de Computação

- **Visão geral da memória:**
  - ***"Location, location, location"*** **– posição da memória referente ao processador importa.**
  - **Registradores (D flip flops) e Caches (SRAM) são localizados próximos.**
  - **A RAM (DRAM) se encontra distante.**

Motorola PowerPC 604



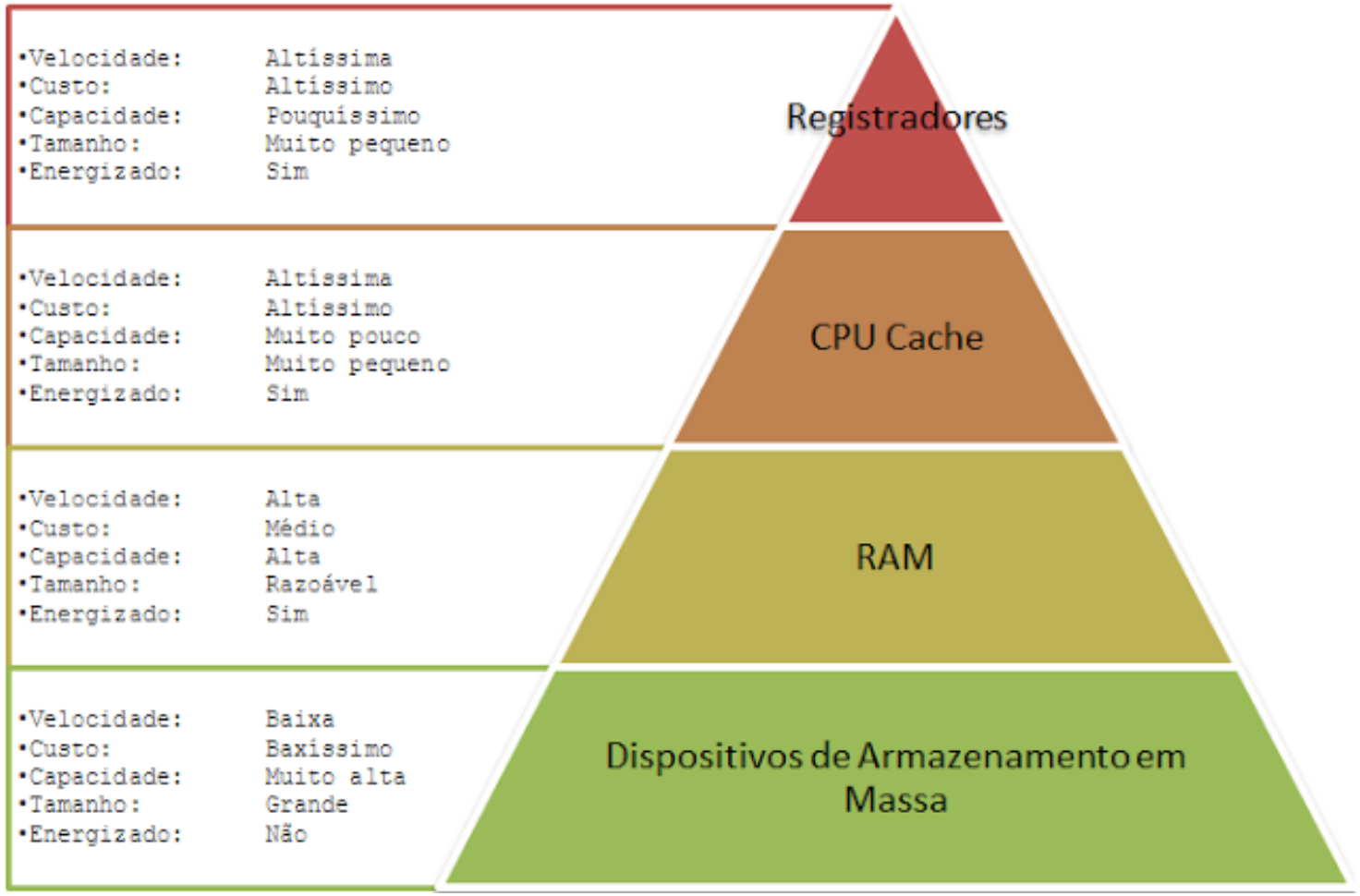
Memory Type	Transistors per Bit Cell	Latency
flip-flop	~20	fast
SRAM	6	medium
DRAM	1	slow



p. 267

Fundamentos de Computação

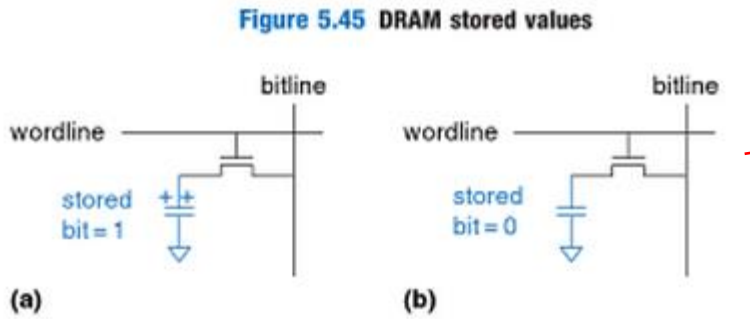
- **Visão geral da memória:** hierarquia de memória.



Fundamentos de Computação

- **Visão geral da memória:**

O bit é a unidade elementar de armazenamento de dados. Valor 0 ou 1, não existe bit vazio.



Célula de memória DRAM

Um conjunto de oito bits é chamado de byte, possui 256 (2^8) valores possíveis.

Palavra, ou *word*, é a unidade natural de bits processados em uma arquitetura .

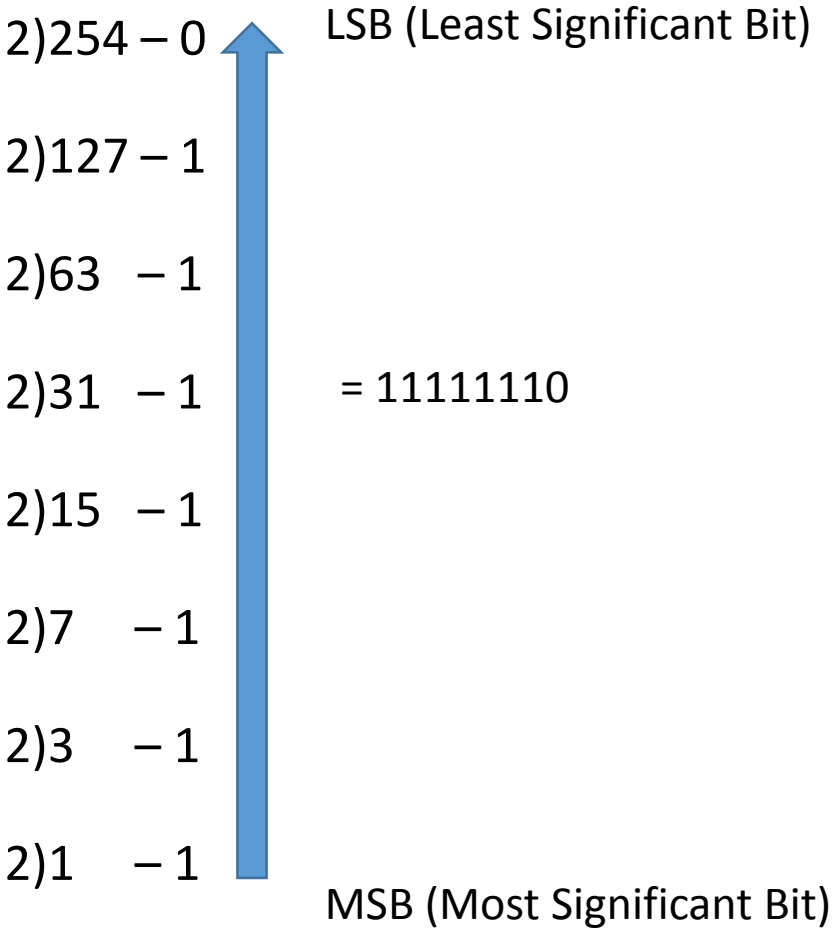
Fundamentos de Computação

- **Representações numéricas:** Um valor pode ser representado em diferentes “alfabetos”.
  - Hexadecimal possui 16 caracteres: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
  - Decimal possui 10 caracteres: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
  - Octal possui 8 caracteres: 0, 1, 2, 3, 4, 5, 6, 7.
  - Binário possui 2 caracteres: 0, 1.
  - Hexadecimal e octal possuem uma relação semi-direta com binário e são amplamente utilizados para **melhor visualização de valores**.
  - Notação: $número_{base}$

Fundamentos de Computação

- **Representações numéricas:** Decimal para binário.
  - Método do teste de resto na divisão por 2

Procure pela operação **shift right**, consegue perceber a similaridade com o que acabamos de fazer?



Fundamentos de Computação

- **Representações numéricas:** Binário para decimal.
  - Método da notação posicional
  - $2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$
    1 1 1 1 1 1 1 0

Agora procure pela operação **shift left**, consegue perceber a similaridade com o que acabamos de fazer?

$= 1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+0*2^0$ **= 254**

Fundamentos de Computação

- **Representações numéricas:** Octal e hexadecimal para decimal e vice-versa.
  - Valores tabelados, são equivalências alfabéticas

Agora procure pela operação **shift left**, consegue perceber a similaridade com o que acabamos de fazer?

Decimal	Hexadecimal	Octal
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	-
9	9	-
10	A	-
11	B	-
12	C	-
13	D	-
14	E	-
15	F	-

Fundamentos de Computação

- **Representações numéricas:** Octal e hexadecimal para binário e vice-versa.
  - O valor máximo de um dígito octal é 7, equivale a 111 em binário, portanto há uma equivalência de 3 bits para cada dígito octal.
  - Similarmente para um dígito hexadecimal, cujo máximo é F (15 dec.), equivale a 1111 em binário, equivalência de 4 bits por dígito.

Agora procure pela operação **shift left**, consegue perceber a similaridade com o que acabamos de fazer?

Binário	Decimal	Hexadecimal	Octal
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	8	-
1001	9	9	-
1010	10	A	-
1011	11	B	-
1100	12	C	-
1101	13	D	-
1110	14	E	-
1111	15	F	-

Fundamentos de Computação

- **Esquemas de codificação:**
  - No computador, letras são números. O significado de cada número é resultado de um padrão ou convenção.
  - Esquemas comuns: ASCII, EBCDIC, Unicode
  - ASCII: Representa caracteres de 8 bits.
  - Unicode: Acomoda alfabetos com mais de 256 caracteres, utiliza 16 bits para representação. 65.536 valores possíveis.

Fundamentos de Computação

- **Esquemas de codificação:** ASCII.
  - American Standard Code for Information Interchange

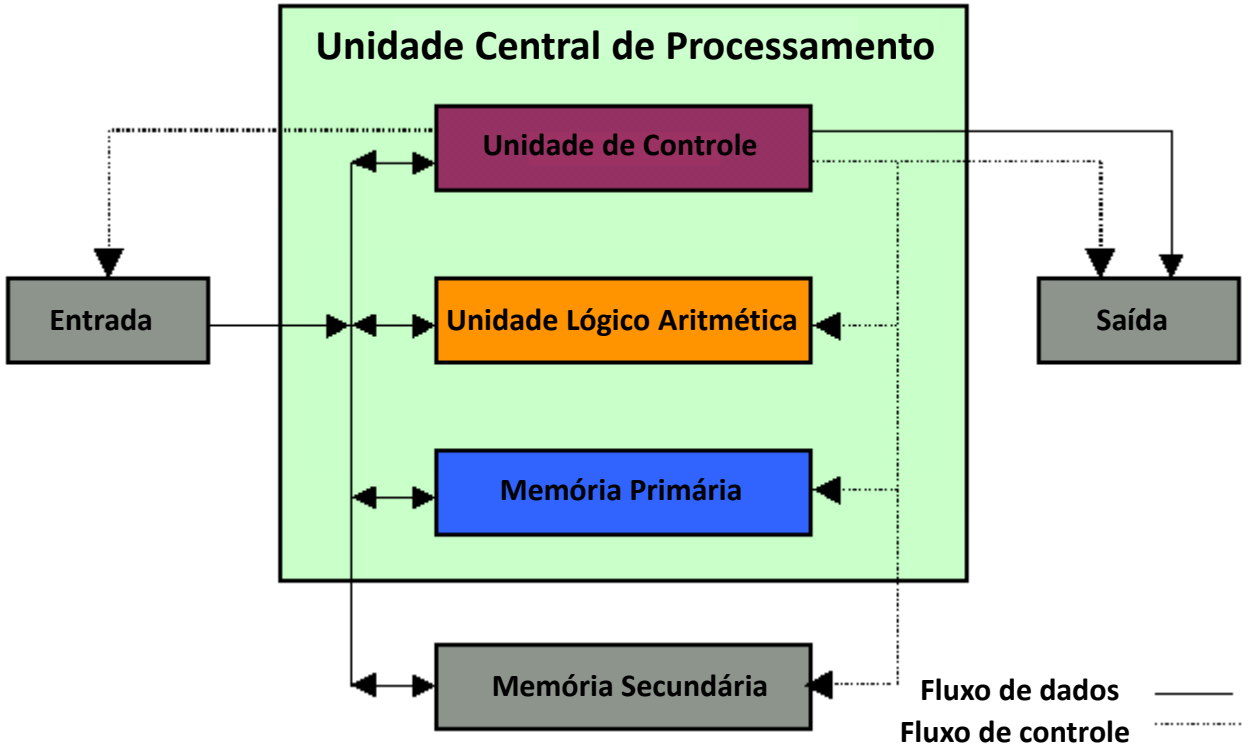
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Quando estiver programando, faça um programa que escreva todos os caracteres de 0 a 255 no console.
Há outros caracteres que não estão na tabela ASCII? Por que?

Fundamentos de Computação

- **Visão geral da arquitetura de processadores:**

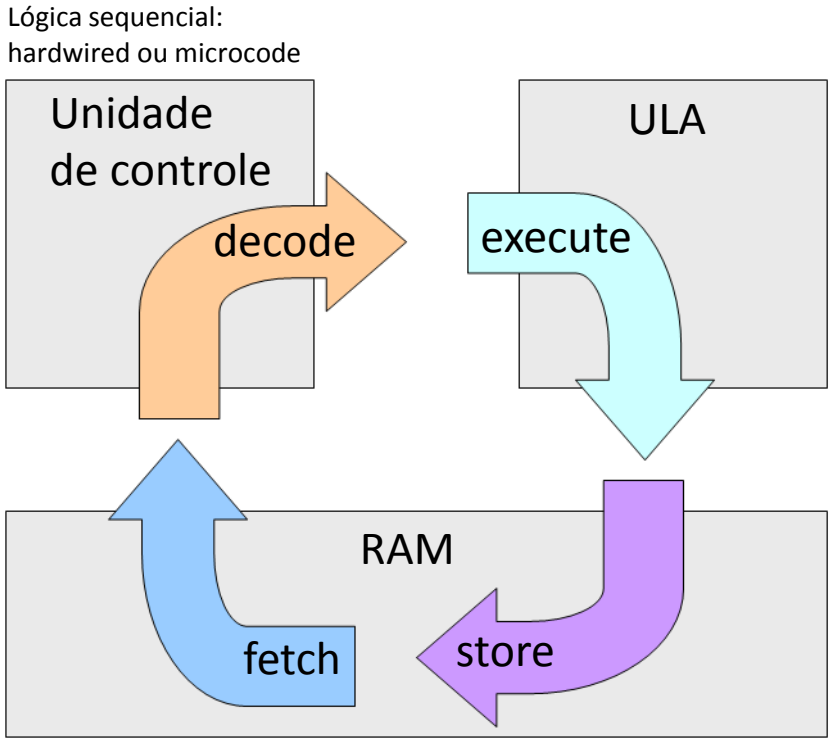


Fundamentos de Computação

- **Unidade de Controle:**

Responsável por manter o ciclo

Fetch – Decode – Execute – Store



Fetch: busca a instrução endereçada pelo PC

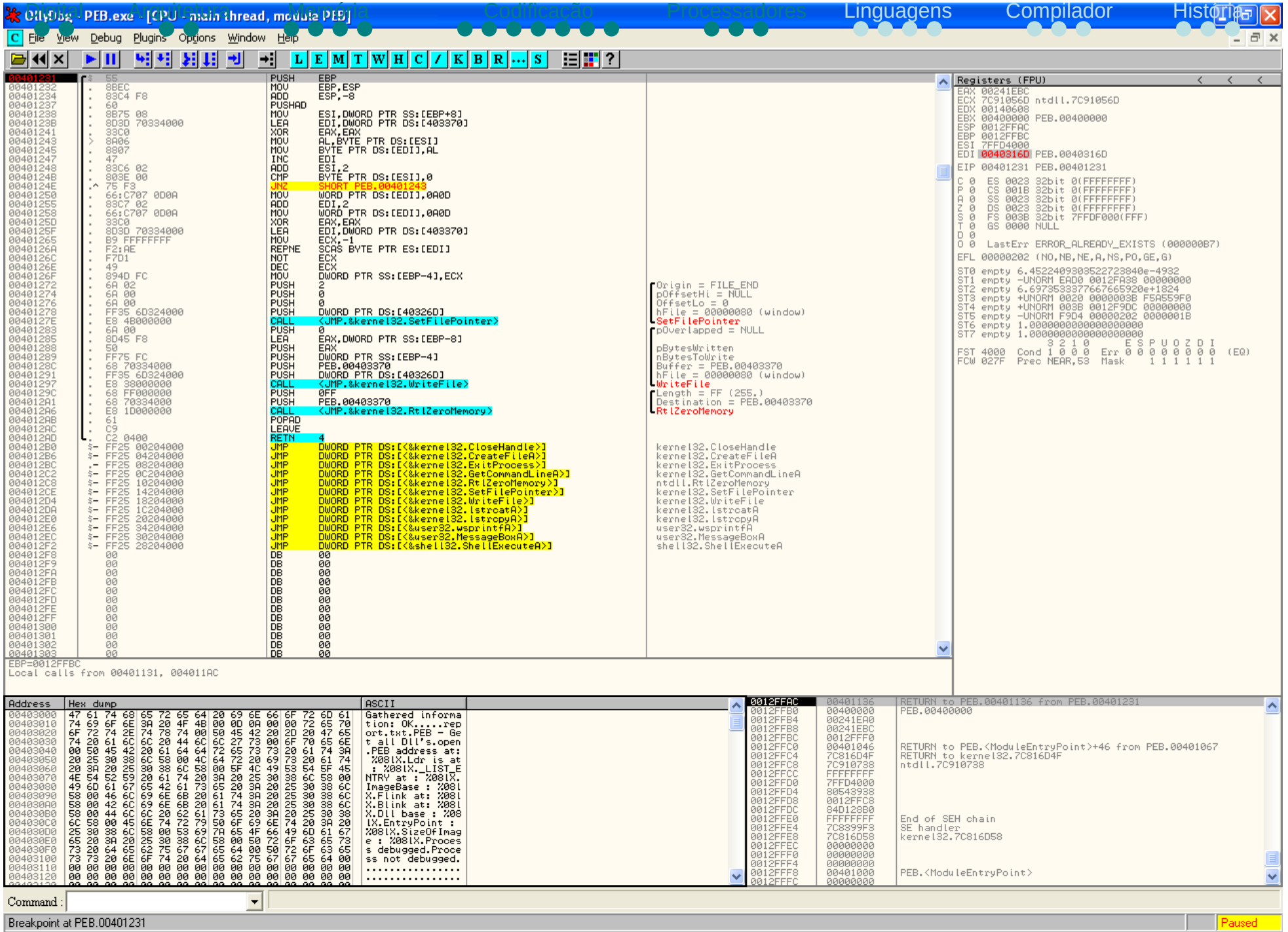
Decode: determina qual o estado da instrução

Execute: efetua a operação da instrução

Store: armazena dados na memória

Observar:

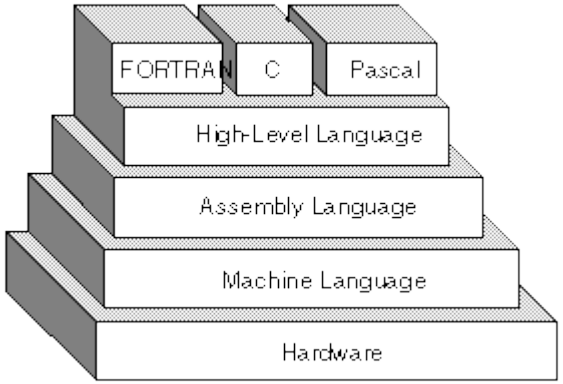
- Instruction Pointer
- Ciclo fetch-decode-execute-store
- Pilha (stack)
- Heap
- Registradores
- Registradores EBP e ESP
- Instruções



Linguagens de Alto Nível

- **Tipos de Linguagens de Programação:**
  - Máquina: códigos numéricos relativos a instruções e operandos (*bit encoding*).

    75 F3



  - Assembly (montagem): representação simbólica do código de máquina, com suporte a labels, macros e estruturas básicas (*symbolic encoding*).

    JNZ PEB.00401243

Linguagens de Alto Nível

- **Problemas do código de montagem:**
  - Difícil compreensão humana.
  - Por consequência, difícil manutenção e dificuldades de depuração.
  - Específico ao hardware.
  - Ausência de código estruturado leva ao código espaguete!

Linguagens de Alto Nível

- **Problemas do código de montagem:**
  - O código é fixo, uma vez definido não muda mais.
  - Controvérsia da localização da complexidade. No processador (CISC), no compilador (RISC), na máquina virtual?

Linguagens de Alto Nível

- **Tipos de Linguagens de Programação:**
  - Alto nível: abstraem detalhes da implementação de hardware e introduzem conceitos de código estruturado.

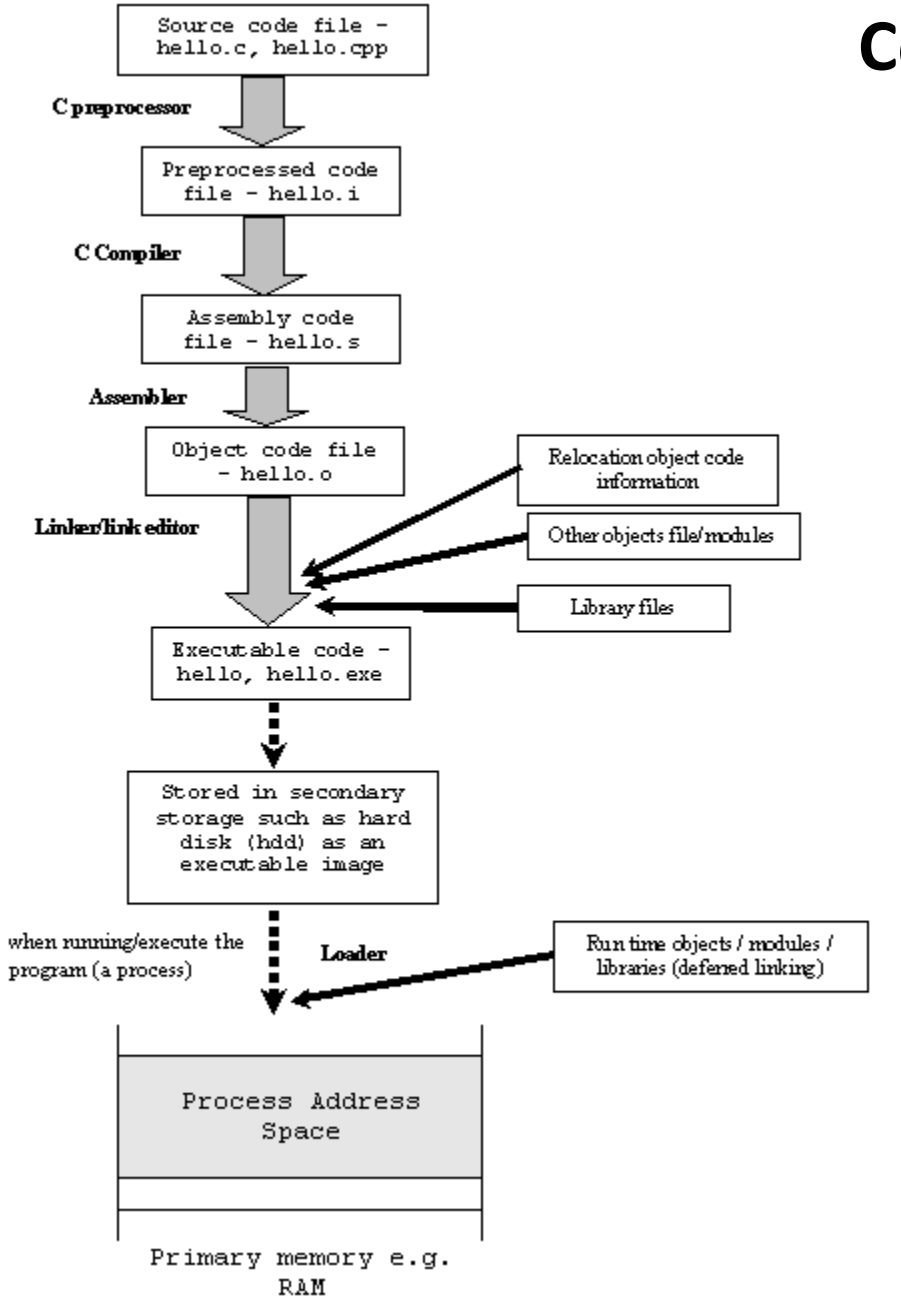
O “nível” é relativo. C já foi uma linguagem de alto nível. 
Atualmente é considerada apenas um nível acima de abstração de assembly, alguns consideram C uma linguagem de médio nível. (tópico controverso)

Compilador/Interpretador

- **Dois paradigmas distintos:**
  - Compiladores processam o código fonte e geram um executável. C é uma linguagem compilada.
  - Interpretadores processam o código fonte durante o tempo de execução.

Compilador/Interpretador

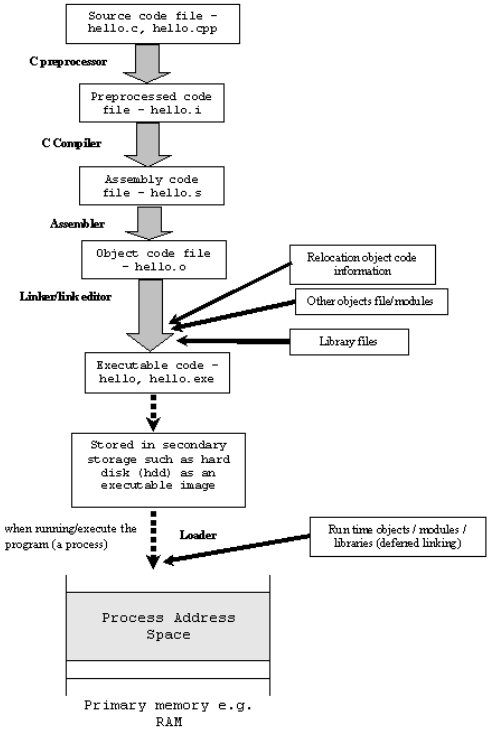
- **Compilador:**



Compilador/Interpretador

Etapas da compilação:

- **Pré-processador: altera os arquivos fontes de acordo com diretivas de pré-processamento.**
- **Compilador: traduz o código para assembly, considerando as análises léxica, sintática e semântica. Efetua otimizações.**
- **Assembler: traduz o código assembly para código de máquina, mantendo alguns parâmetros (labels para endereços, por exemplo).**
- **Linker: efetua a ligação de todos os objetos e bibliotecas em um código de máquina final.**



História e Introdução a C

- **Origens:**
  - Primeira versão do Unix: escrito em assembly por Ken Thompson e Dennis Ritchie

  - Depois traduzido para B, uma implementação simplificada do BCPL por Thompson
  - B era limitado: orientada a *words* e não poderia acessar bytes individuais ou valores de ponto flutuante, problemático em novos hardwares

História e Introdução a C

- **Evolução:**
  - Mudanças no acesso a memória e funcionamento de ponteiros
  - Alterações sintáticas
  - C!

Algol	International Group
BCPL	Martin Richards
B	Ken Thomson
Traditional C	Dennis Ritchie
K&R C	kernighan & Ritchie
ANSI C	ANSI Commitee
ANSI/ISO C	ISO Commitee
C99	Standerd Commitee

The Development of the C Language*

Dennis M. Ritchie
Bell Labs/Lucent Technologies
Murray Hill, NJ 07974 USA

Leitura!

História e Introdução a C

- **Genealogia:**