O modelo do computador

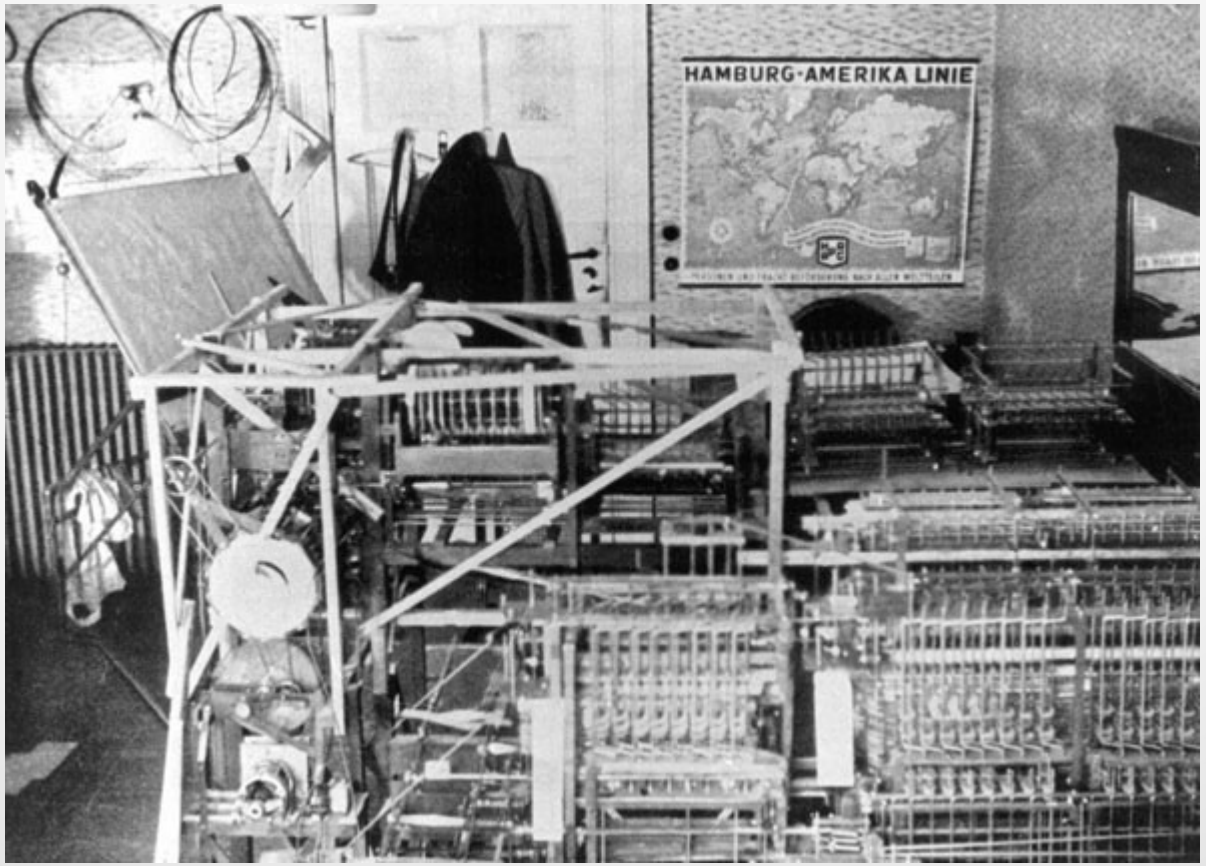
Objetivos:

- Mostrar como é o funcionamento dos computadores modernos

- Mostrar as limitações a que estamos sujeitos quando programamos

Histórico

- Os primeiros computadores são da década de 1930. Konrad Zuse construiu o primeiro computador (Z1) eletro-mecânico binário programável em 1936.



Histórico

- Eles eram feitos para rodar programas específicos
- Engenheiros reconstruiam os computadores para que ele fizesse cálculos diferentes
- Esta tarefa consumia 3 semanas
- Entrada e saída era feita por cartões perfurados

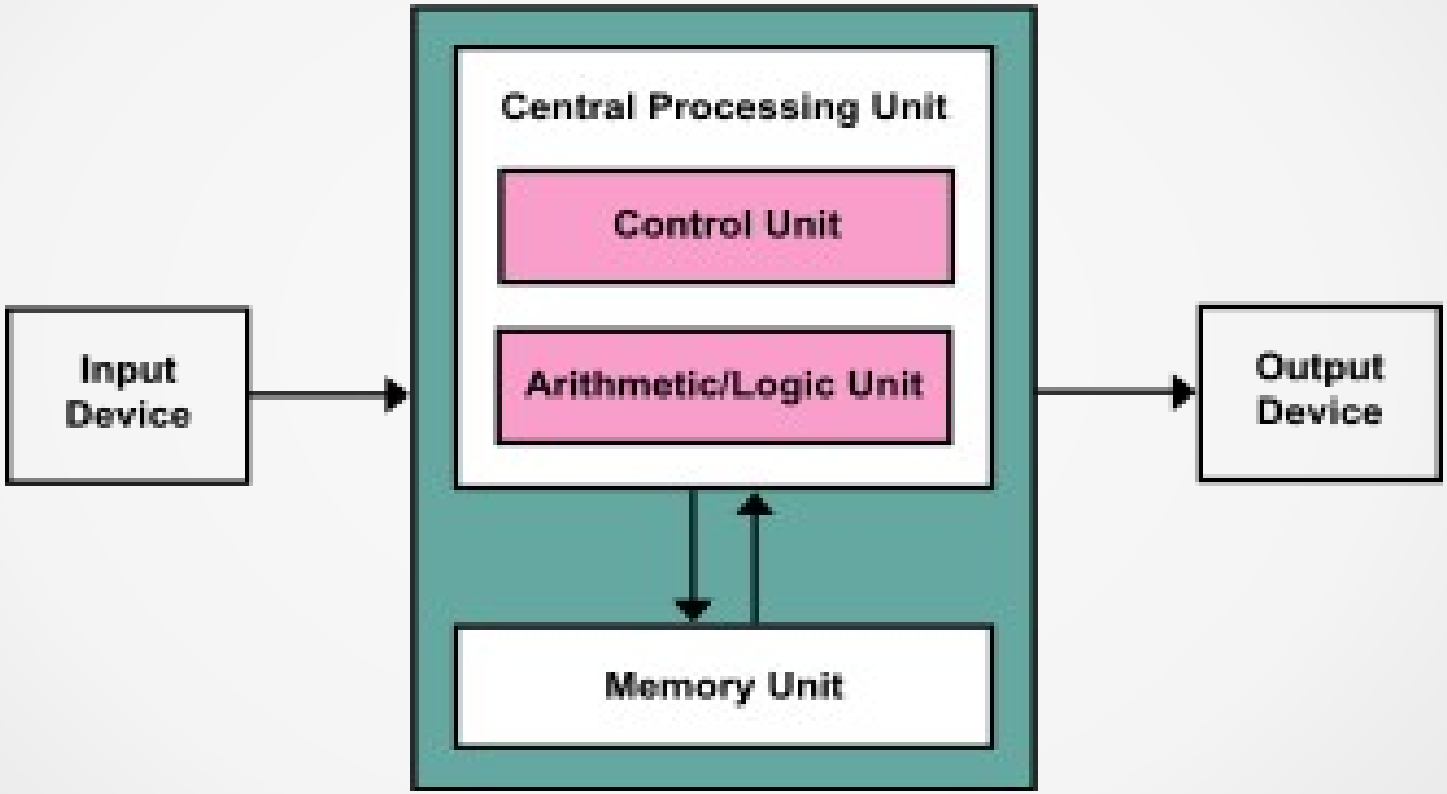
Histórico

- Evidentemente, não era muito prático de se reprogramar, além de caro
- Exemplo ainda hoje: calculadoras de bolso
- Era preciso um novo modelo que facilitasse a programação
- John Von Neumann implementou o modelo proposto por Alan Turing, em que tanto o programa quanto os dados ficam em memória

O modelo von Neumann

- Uma unidade de central de processamento (CPU)
  - Unidade lógica e aritmética
  - Registradores
  - Uma unidade de controle
    - Registrador de instruções
    - Contador do programa
- Memória interna
  - Dados e programa
- Unidade de entrada e saída

Esquema do modelo von Neumann



O ciclo de execução de instruções

- É um elemento adicional que controla a execução do programa
- Assim, tanto o programa quanto os dados ficam armazenados na RAM
- A unidade de controle, segundo o ciclo de execução de instruções, define a operação que é executada a cada passo

O hardware para o modelo Von Neumann

- Cada fabricante define qual o conjunto de instruções básicas para seu computador
- Este conjunto é implementado no circuito integrado e **é tudo o que o computador sabe fazer**!
- Processadores modernos implementam apenas algumas dezenas de instruções
- Programar significa realizar operações complexas usando apenas este pequeno conjunto de instruções

Exemplo: O computador da BCC

- BCC é a sigla da *Big Computer Company,* e em termos teóricos tem o mesmo poder das suas concorrentes multinacionais
- O computador da BCC implementa apenas 9 (nove!) instruções
- Ele tem uma memória RAM, um teclado e um monitor

Endereço	Conteúdo	Endereço	Conteúdo	Endereço	Conteúdo
0	1	20	47	40	57
1	54	21	49	41	56
2	2	22	6	42	54
3	1	23	52	43	8
4	50	24	51	44	57
5	4	25	3	45	9
6	7	26	53	46	33
7	46	27	46	47	2
8	4	28	52	48	76
9	47	29	5	49	67
10	46	30	55	50	76
11	46	31	53	51	124
12	7	32	54	52	14
13	48	33	8	53	47
14	4	34	55	54	235
15	49	35	2	55	35
16	50	36	56	56	23
17	48	37	46	57	78
18	3	38	52	58	243
19	51	39	5	59	27

Endereços vs conteúdos

- Denotamos por [p] o conteúdo do endereço p
  - [0] = ?
  - [[0]] = ?
  - [0] + 1 = ?
  - [0] + [1] = ?
  - [0 + 1] = ?
  - [[0] + [1]] = ?
  - [[0] + 1] = ?

O conjunto de instruções da BCC

1	load	Escreva em [p+1] o valor de [p+2]. Some 3 em p
2	add	Escreva em [p + 1] a soma [[p + 2]] e [[p + 3]]. Some 4 em p
3	sub	Escreva em [p + 1] a diferença [[p + 2]] e [[p + 3]]. Some 4 em p
4	mult	Escreva em [p + 1] o produto [[p + 2]] por [[p + 3]]. Some 4 em p
5	div	Escreva em [p + 1] a divisão [[p + 2]] por [[p + 3]]. Some 4 em p
6	sqrt	Escreva em [p + 1] a raiz quadrada de [[p + 2]]. Some 3 em p
7	read	Leia um número do teclado e guarde em [p + 1]. Some 2 em p
8	write	Escreva [[p + 1]] na tela. Some 2 em p
9	stop	Pare

O ciclo de execução de instruções

1) Começe com p valendo zero (p = 0)
2) Interprete [p] de acordo com a tabela de instruções e só pare quando a instrução for uma ordem de parar (instrução 9)

Acompanhamento do programa

- Vamos nos colocar no lugar do computador e tentar entender como ele trabalha

- Vamos acompanhar o funcionamento dele a partir da fotografia da memória acima e do conjunto de instruções da BCC

1	load	Escreva em [p+1] o valor de [p+2]. Some 3 em p
2	add	Escreva em [p + 1] a soma [[p + 2]] e [[p + 3]]. Some 4 em p
3	sub	Escreva em [p + 1] a diferença [[p + 2]] e [[p + 3]]. Some 4 em p
4	mult	Escreva em [p + 1] o produto [[p + 2]] por [[p + 3]]. Some 4 em p
5	div	Escreva em [p + 1] a divisão [[p + 2]] por [[p + 3]]. Some 4 em p
6	sqrt	Escreva em [p + 1] a raiz quadrada de [[p + 2]]. Some 3 em p
7	read	Leia um número do teclado e guarde em [p + 1]. Some 2 em p
8	write	Escreva [[p + 1]] na tela. Some 2 em p
9	stop	Pare

Aspectos observados

- O computador não sabe o que está fazendo
- Ele obedece as instruções contidas na memória
- O ser humano percebe o que está havendo
- Existe uma diferença grande entre a compreensão do ser humano e a mecânica do computador

- O ser humano pode visualizar o processo de outra maneira!

Separando as instruções

Endereço	Instrução	Operando	Operando	Operando
0	1	54	2	
3	1	50	4	
6	7	46		
8	4	47	46	46
12	7	48		
14	4	49	50	48
18	3	51	47	49
22	6	52	51	
25	3	53	46	52
29	5	55	53	54
33	8	55		
35	2	56	46	52
39	5	57	56	54
43	8	57		
45	9			

Efeito desta nova visualização

- O computador não mudou o modo de operar
- Esta visualização apresentada apenas facilita o ser humano a perceber melhor o que o computador está fazendo
- Isto é um aspecto puramente humano, não computacional

Abstração dos endereços

Endereço	Instrução	Operando	Operando	Operando
0	1	54	2	
3	1	50	4	
6	7	46		
8	4	47	46	46
12	7	48		
14	4	49	50	48
18	3	51	47	49
22	6	52	51	
25	3	53	46	52
29	5	55	53	54
33	8	55		
35	2	56	46	52
39	5	57	56	54
43	8	57		
45	9			

Abstração dos endereços

- Os endereços não são relevantes para o ser humano
- Eles podem ser removidos perfeitamente que ainda podemos compreender o processo.
- Para o computador, o endereço também não precisa ser fixo. O importante é que o contador de programa inicie na primeira instrução.

Abstração dos endereços

Instrução	Operando	Operando	Operando
1	54	2	
1	50	4	
7	46		
4	47	46	46
7	48		
4	49	50	48
3	51	47	49
6	52	51	
3	53	46	52
5	55	53	54
8	55		
2	56	46	52
5	57	56	54
8	57		
9			

Abstração dos códigos das instruções

- Os seres humanos preferem usar nomes no lugar de números
- Por exemplo, eu sou mais conhecido pelo meu nome, embora para alguns sistemas eu seja o meu RG ou o meu CPF
- Logo, podemos trocar os códigos das instruções por nomes

Abstração dos códigos das instruções

Instrução	Operando	Operando	Operando
load	54	2	
load	50	4	
read	46		
mult	47	46	46
read	48		
mult	49	50	48
sub	51	47	49
sqrt	52	51	
sub	53	46	52
div	55	53	54
write	55		
add	56	46	52
div	57	56	54
write	57		
stop			

Abstração do repertório de instruções

- Vamos agora abstrair as instruções baseado nas tabelas seguintes:
  - X = [p+1], Y = [p+2], Z = [p+3]

[p]	Instrução	Notação
1	**load** x y	X ← [y]
2	**add** x y z	X ← [y] + [z]
3	**sub** x y z	X ← [y] - [z]
4	**mult** x y z	X ← [y] * [z]
5	**div** x y z	X ← [y] / [z]
6	**sqrt** x y	X ← √[y]
7	**read** x	X ← v
8	**write** x	□ ← [x]
9	**stop**	●

Abstração do repertório de instruções

54 ← 2
50 ← 4
46 ← v
47 ← [46] * [46]
48 ← v
49 ← [50] * [48]
51 ← [47] - [49]
52 ← √[51]
53 ← [46] - [52]
55 ← [53] / [54]
□ ← [55]
56 ← [46] + [52]
57 ← [56] / [54]
□ ← [57]
●

Abstração dos endereços de memória

- Os endereços de memória ainda estão sob a forma numérica
- Mais uma vez, preferimos nomes para isto
- Um exemplo é o celular de hoje em dia, ligamos para um nome, não mais para um número (raramente o contrário)
- Então vamos dar (bons) nomes para as variáveis

Abstração dos endereços de memória

Endereço	Nome
54	dois
50	quatro
46	B
47	quadradoB
48	C
49	quadruploC
51	discriminante
52	raizdiscriminante
53	dobromenorraiz
55	menorraiz
56	dobromaiorraiz
57	maiorraiz

Abstração dos endereços de memória

Dois	← 2
quatro	← 4
B	← v
QuadradoB	← B * B
C	← v
quadruploC	← quatro * C
discriminante	← quadradoB - quadruploC
raizdiscriminante	← √discrinante
dobromenorraiz	← B - raizdiscriminante
menorraiz	← dobromenorraiz / dois
□	← menorraiz
dobromaiorraiz	← B + raizdiscriminante
maiorraiz	← dobromaiorraiz / dois
□	← maiorraiz
●	

Último graus de abstração direta

- Esta última forma de se visualizar o programa é a última forma de abstração direta
- A partir desta, é possível voltar à primeira forma (da fotografia da memória) apenas por traduções reversas
- A partir deste ponto, para 'vermos' o programa na forma de alguma linguagem, precisamos dos **compiladores**

Abstração das instruções (linguagem)

- Com o uso de programas *tradutores* ou *compiladores*, é possível melhorar o grau de facilidade visual dos programas

- Os compiladores são programas que recebem um texto escrito em um formato adequado e geram como saída um programa no formato da máquina

Linguagens

- As linguagens de programação são baseadas em gramáticas rígidas e portanto o formato dos texto é bastante restrito

- As linguagens seguem diferentes paradigmas, que basicamente definem o grau de restrição delas
  - Linguagens estruturadas, funcionais, lógicas, orientadas a objetos, etc.

Linguagens

- As linguagens chamadas de 'alto nível' são aquelas que permitem ao ser humano um alto grau de compreensão do texto que está escrito, embora ainda longe da nossa chamada língua natural

- Um dos motivos da rigidez das linguagens de programação é a eliminação de ambiguidade dos textos (linguagens livres de contexto)

- A seguir veremos nosso programa escrito em uma linguagem de alto nível (procedura e imperativa) chamada de **PASCAL**

Versão do programa em PASCAL

```
Program Bhaskara (input,output);
Var b,c,raizdiscriminante : real;

Begin
   Read (b);
   Read (c);
   Raizdiscriminante:= sqrt (b*b - 4*c);
   Write ((b - raizdiscriminante)/2);
   Write ((b + raizdiscriminante)/2);
End.
```

O que é uma linguagem de programação?

- É meramente uma notação convencionada visando facilitar a vida do ser humano que programa o computador

- Esta notação trata de como um texto se traduz em um programa executável em um determinado sistema operacional

- Um programa que traduz um texto em linguagem de máquina é chamado de **compilador**

O que é programar?

- Programar um computador em alto nível é, basicamente, conhecer e dominar uma notação através da qual textos (ou **programas fonte**) são traduzidos para **programas executáveis**

- Programar significa concatenar as instruções disponíveis dentro de um repertória a fim de transformar dados de entrada em dados de saída para **resolver um problema**

Quais as limitações do compilador

- O compilador depende de duas coisas para poder transformar textos em programas executáveis:
  - Do sistema operacional (Linux, MacOs, Android, etc.)
  - Do hardware (Intel I386, Intel 64, ARM, etc)

O que se precisa ter para programar?

- Ter a disposição um editor de textos (ASCII) para codificar o algoritmo em forma de programa fonte
- Ter a disposição um compilador para a linguagem escolhida para transformar automaticamente um programa fonte em um programa executável
- Evidentemente, ter habilidade de escrever textos, que são programas que resolvem problemas

Evolução e alternativas

- Inteligência artificial voltada para linguagens de ainda mais alto nível, por exemplo linguagens sensíveis ao contexto

- Computação quântica: escapa do modelo von Neumann, podem dar origem a outros tipos de linguagens