Arquitetura e Organização de Computadores I

Como os computadores evoluíram? Por que? Quais os marcos desta evolução?

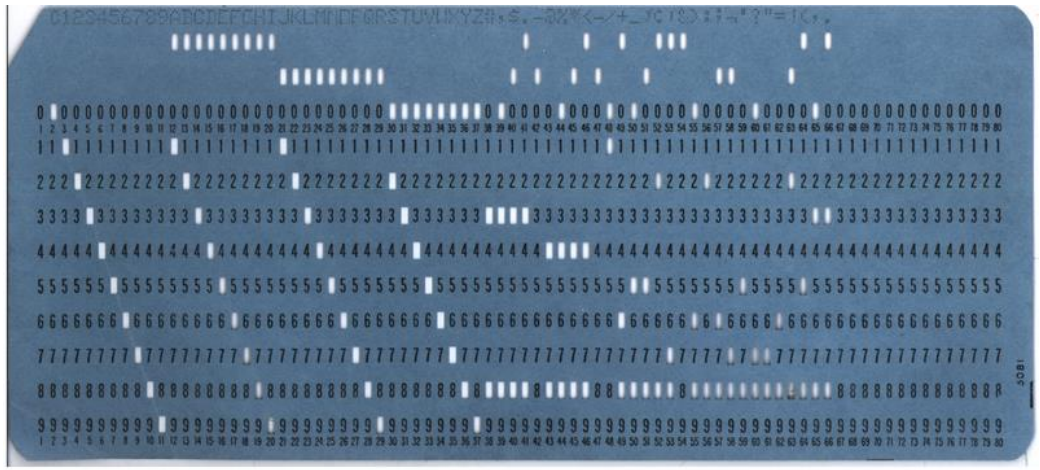
Evolução dos Computadores

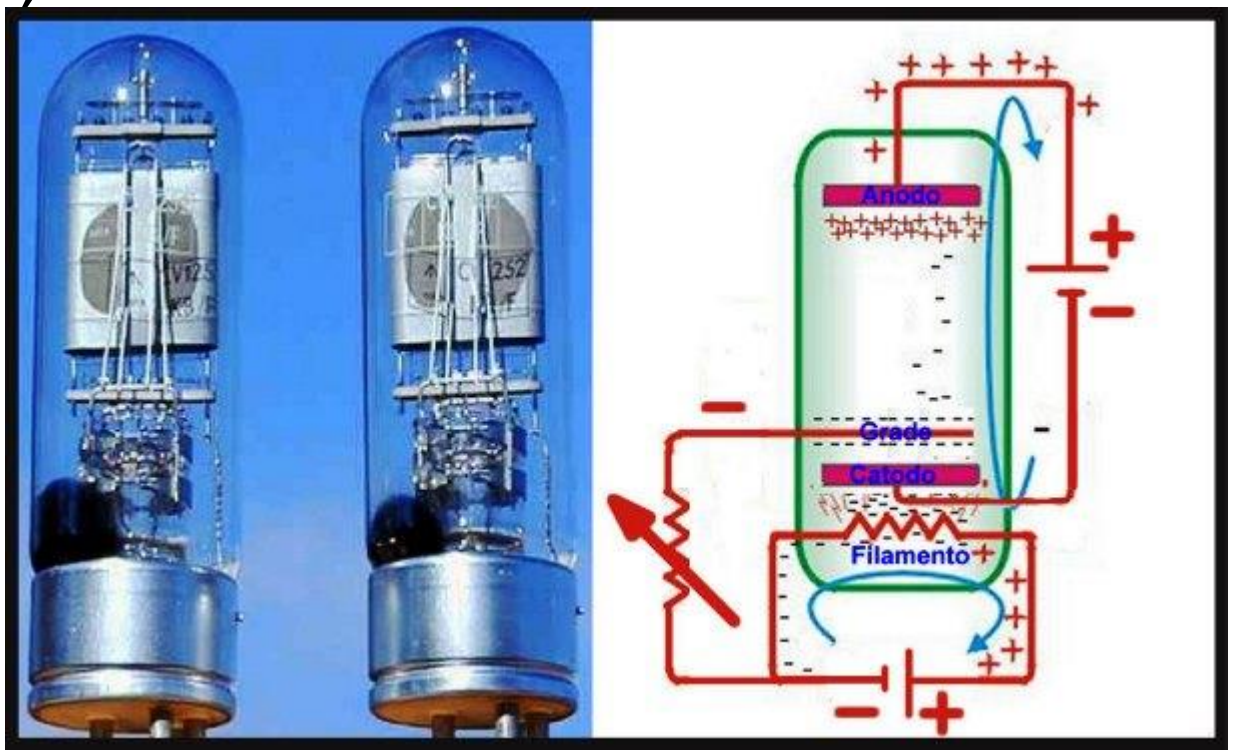
Gerações de Computadores

- Três gerações bem definidas:
  - Válvulas;
  - Transistores;
  - Circuitos Integrados;
  - Demais gerações variam;
- Cada geração caracterizada por um avanço tecnológico;
- Mudanças fundamentais em termos de:
  - Tamanho, custo, poder, eficiência, confiança;

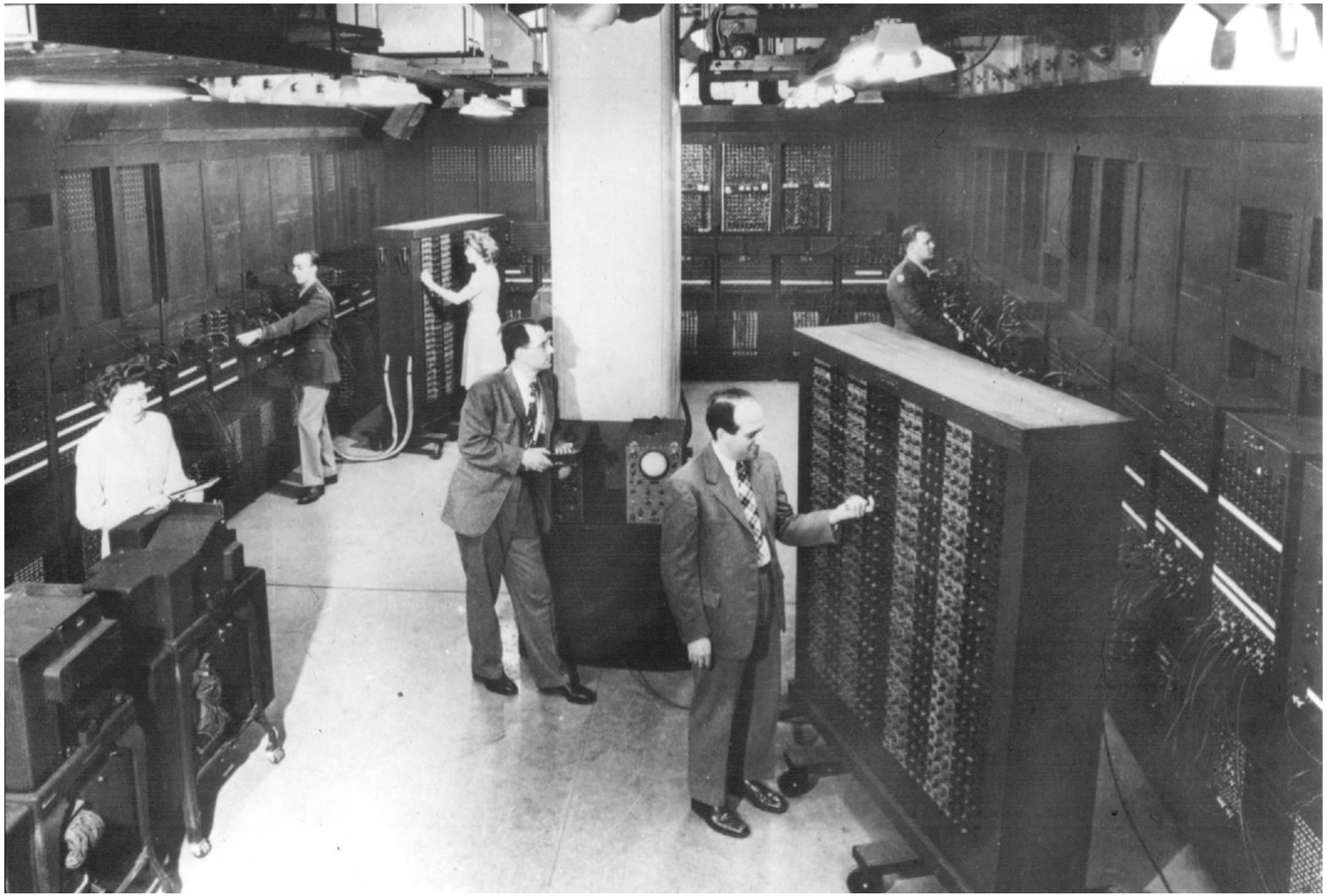
Primeira Geração – 40's e 50's
Tubos de Vácuo

- Caros, volumosos, não confiáveis, beberrões de energia;
- Usavam: cartões perfurados (dados) / fitas, memórias de tambor magnético, linguagem de máquina;





O ENIAC



ENIAC - background

- **E**lectronic **N**umerical **I**ntegrator **A**nd **C**omputer:
  - Eckert and Mauchly;
  - University of Pennsylvania:
- Objetivo:
  - Trajetória balística para armas na Guerra;
- Construção:
  - 1943 -1946;
  - Tarde de mais para a guerra, foi utilizado para cálculos científicos, inclusive bombas atômicas;
- Aposentado em1955;

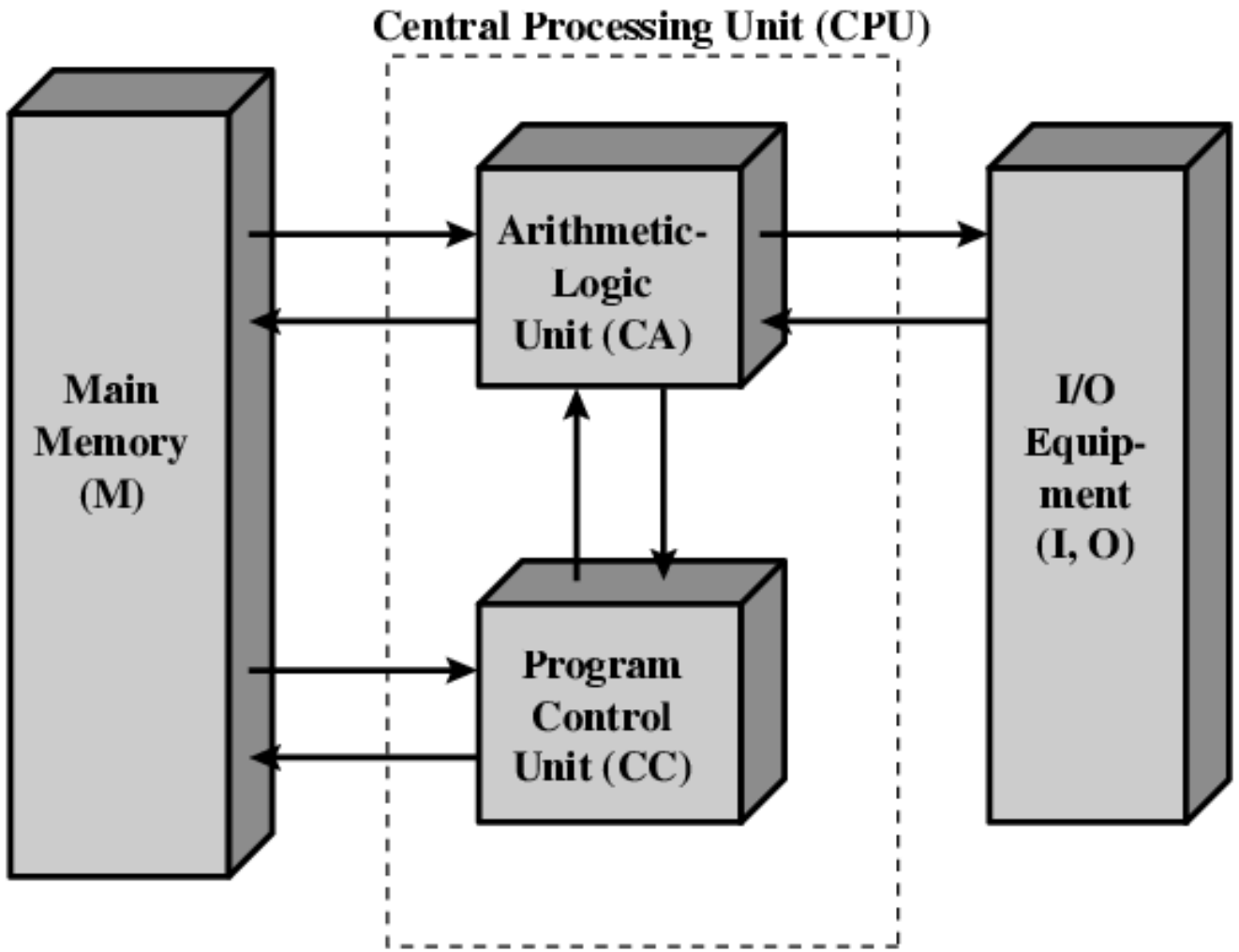
ENIAC - detalhes

- Computador Decimal (not binary);
- Programado manualmente;
- 18,000 tubos de vácuo;
- 30 toneladas;
- 5.000 somas por segundo;

Von Neumann/Turing

- Após as dificuldades do ENIAC:
  - Conceito de programa armazenado;
  - O programa e dados são armazenados em memória;
- Operações matemáticas em Binário;
- Unidade de controle interpreta e executa as instruções da memória;
- I/O controlado pela unidade de controle;
- Novo computador:
  - Princeton Institute for Advanced Studies;
    - IAS;
  - Completo em 1952;

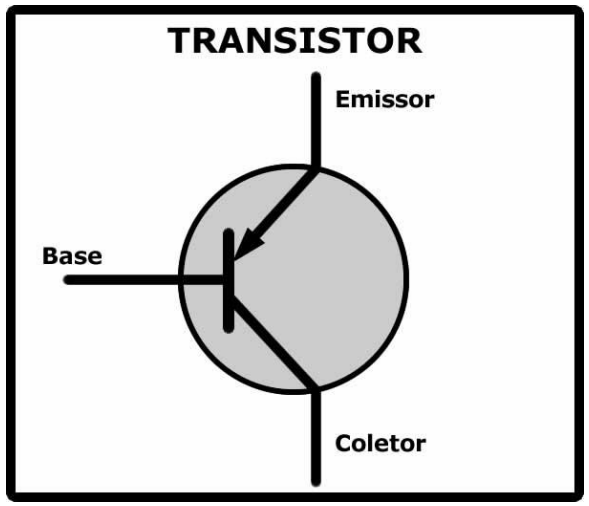
Máquina de von Neumann

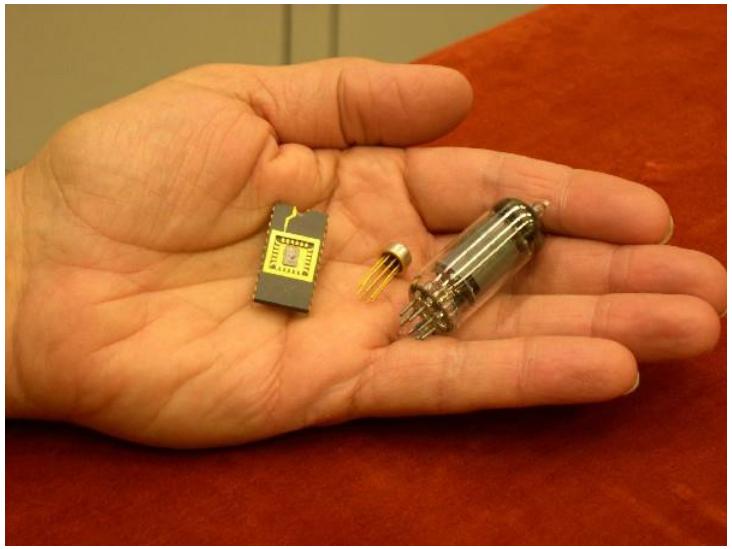


Segunda Geração – 50's e 60's
Transistores

- Menor, mais rápido, mais barato, mais eficiente em termos de energia e mais confiável em comparação aos tubos de vácuo;

- Linguagens de montagem (ASSEMBLY) e as primeiras versões do FORTRAN e COBOL;

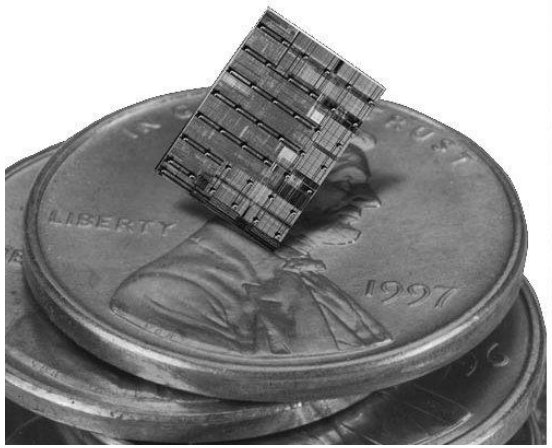


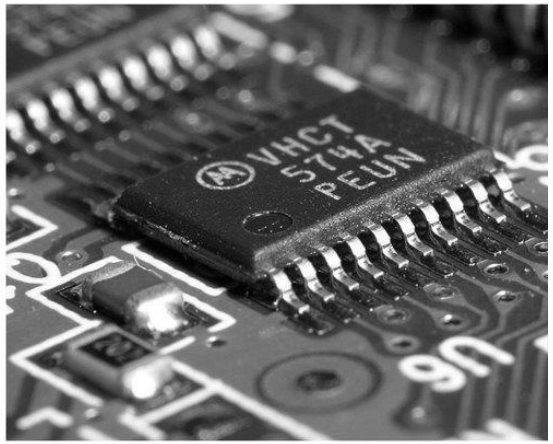


IBMs

Terceira Geração – 60's e 70's
Circuitos Integrados (microchips)

- Um grande número de transistores sobre um único chip:
  - LSI (Large Scale Integration - 100 transistores)
- Rapidez e eficiência aumentou drasticamente;
- Teclados e monitores;
- Os sistemas operacionais;





IBM Series 360

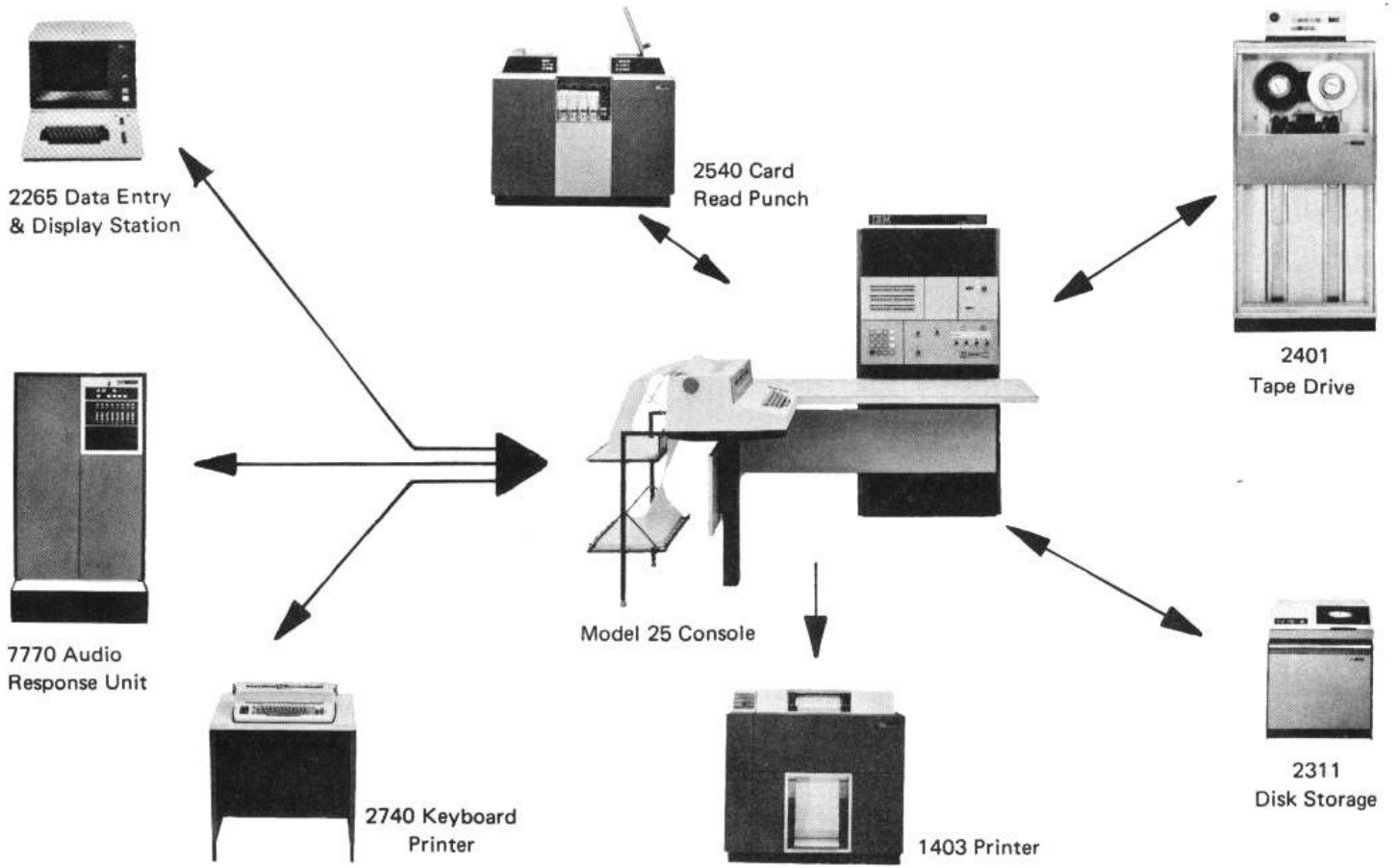


Figure 16. Machine-to-machine communication

IBM Series 360

Quarta Geração – 70’ to 90’s Microprocessadores

- Permitiram que milhares de transistores fossem colocados em um circuito integrado;
  - VLSI (Very Large Scale Integration - 1.000 transistors)
- Computação pessoal e embarcada possíveis;
- Gráficos e mouse;
- Dispositivos portáteis;

Quinta Geração – presente e futuro
IA e Conectividade

- ULSI (Ultra-Large Scale Integration - milhões de transistores);
- Miniaturização dos computadores;
- Discos de estado sólido;
- IA;
- Computação Ubíqua;
- Computação Quântica;
- Biocomputação;
- Computação ótica;

Arquiteturas Computacionais

Máquina de von Neumann

- John von Neumann, “First Draft of a Report on the EDVAC”,
- Moore School of Electrical Engineering, Univ. of Pennsylvania
- June, 30, 1945

(Decembro 28, 1903 – Fevereiro 8, 1957)

Três contribuições fundamentais:

1. Conceito dos programas armazenados
2. Organização básica de um computador
3. Arquitetura básica (tipos de instruções)

Principais Características

- Sistema com três subsistemas básicos:
  - CPU (unidade central de processamento);
  - memória principal de leitura e escrita ;
  - sistema de entrada e saída;

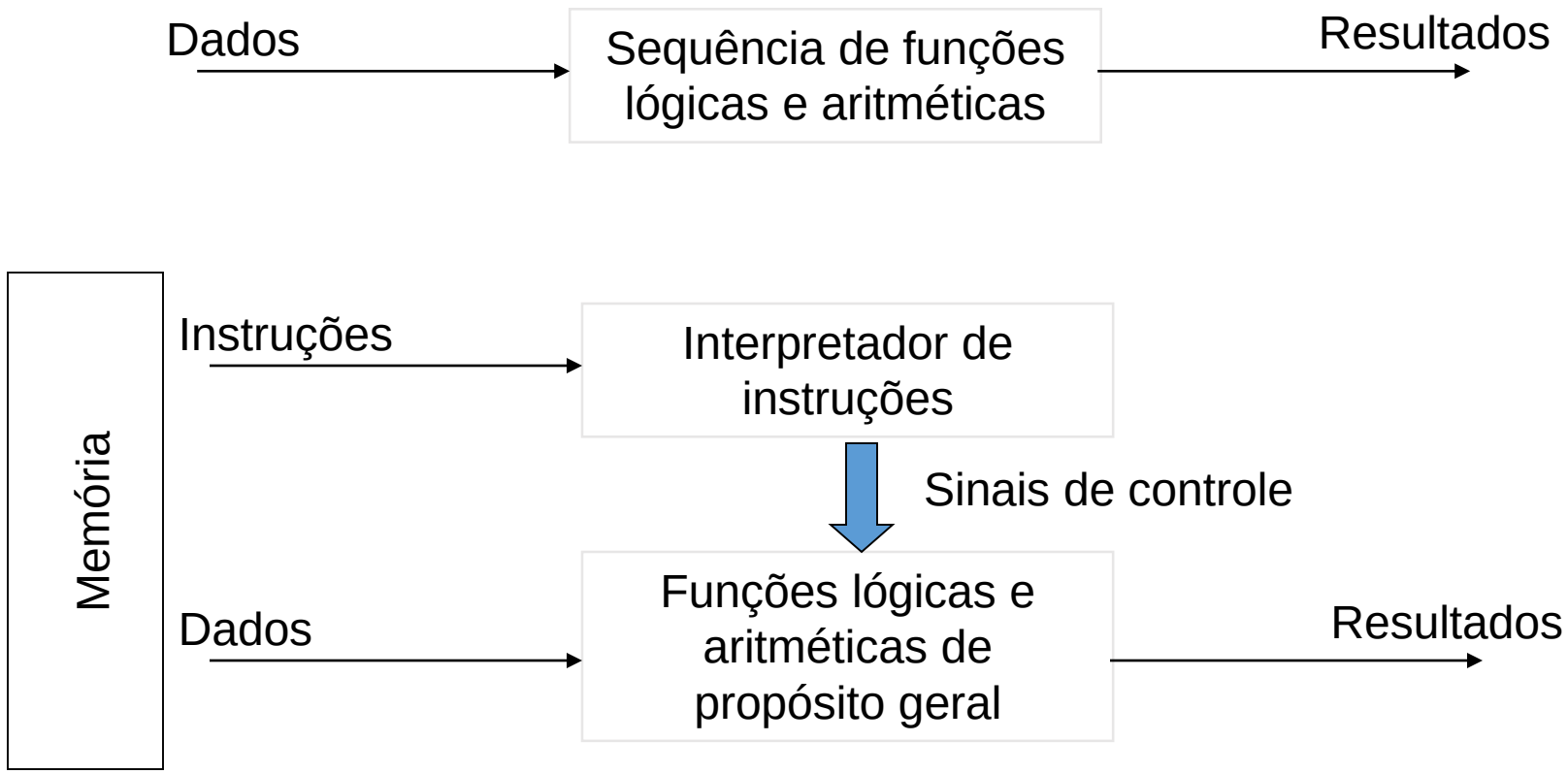
- Utilização do conceito de programa armazenado
  - Execução sequencial de instruções

- Existência de um caminho único entre memória e unidade de controle

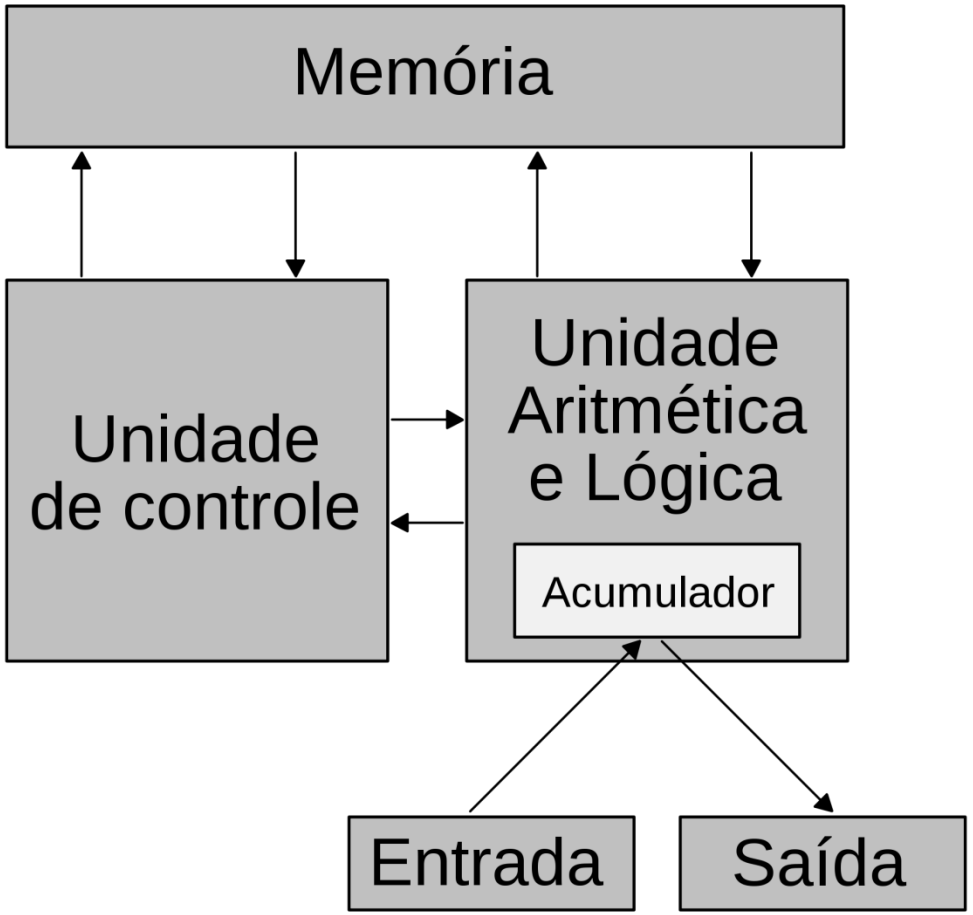
A arquitetura de von Neumann

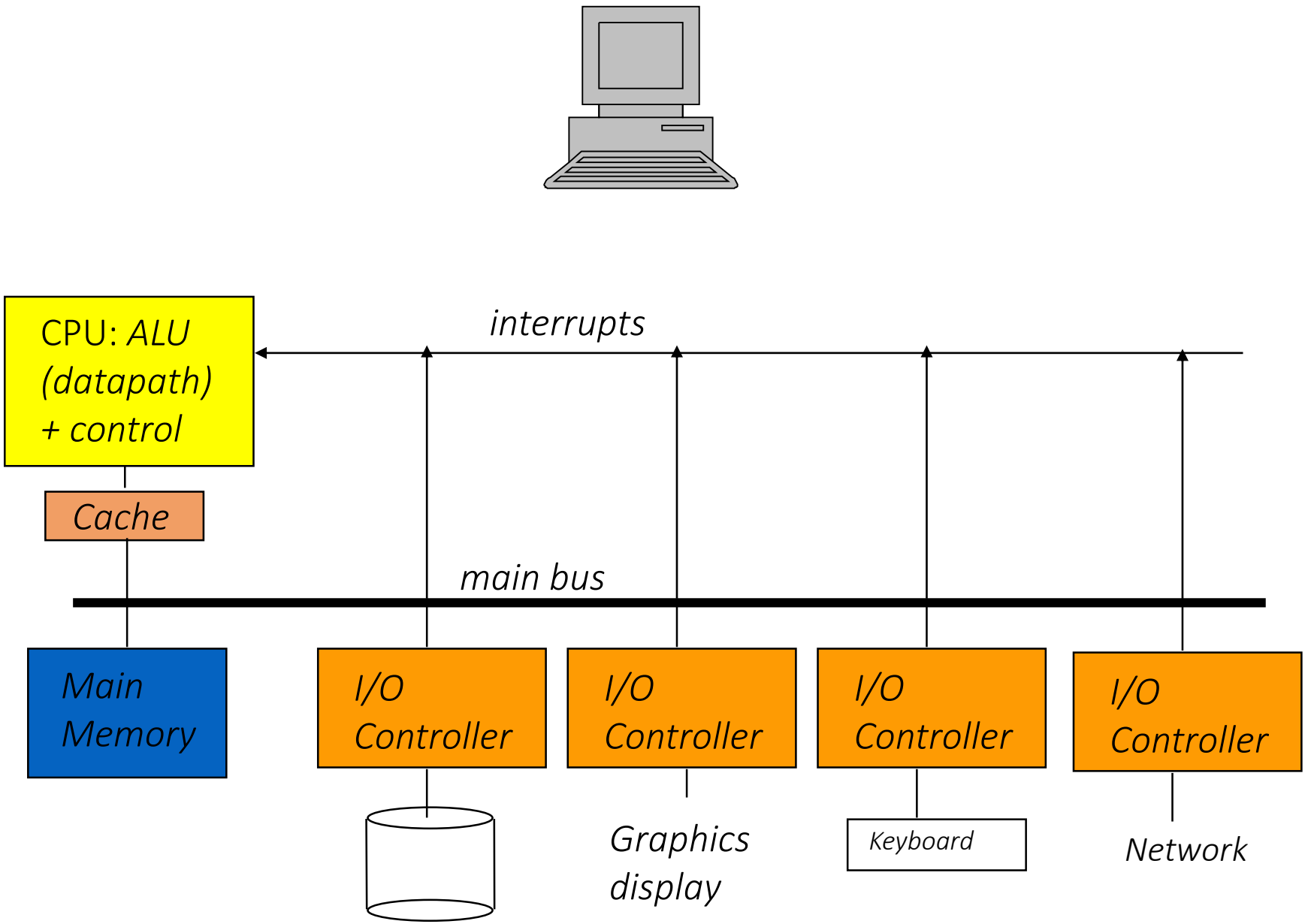
- Conceito dos programas armazenados:
  - O programa consiste em instruções binárias, que são executadas sequencialmente, e que estão armazenadas em posições consecutivas de memória;
  - A unidade de controle decodifica cada instrução e gera os sinais de controle necessários para que os componentes restantes executem essa instrução;
  - O computador pode ser reprogramado, alterando apenas o conteúdo da memória

A arquitetura de von Neumann



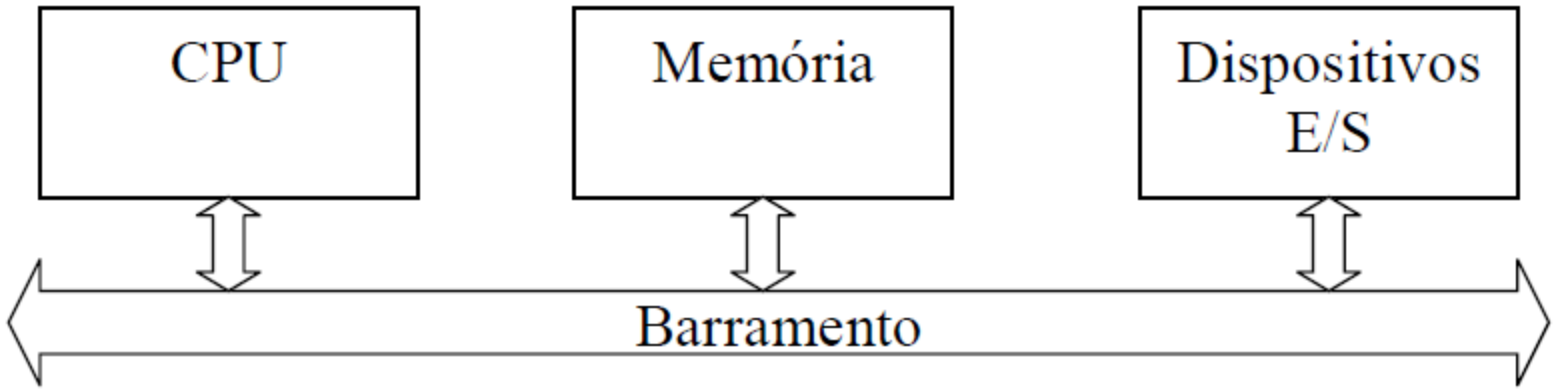
A arquitetura de von Neumann




CPU: ALU (datapath) + control
interrupts
Cache
main bus
Main Memory
I/O Controller
I/O Controller
I/O Controller
I/O Controller
Graphics display
Keyboard
Network

O que é um Computador?

Arquitetura Básica de um Computador



O que é um computador?

- **Componentes:**
  - Entrada (mouse, teclado, USB);
  - Saída (monitor, impressora);
  - Memória (disk drives, DRAM, SRAM, CD, DVD, flash memory);
  - Rede;
- **Foco inicial: o processador (controle e via de dados)**
  - Implementado usando milhões de transistores;
  - Impossível entender “olhando cada transistor”;
  - Precisamos de...

Abstrações do Computador

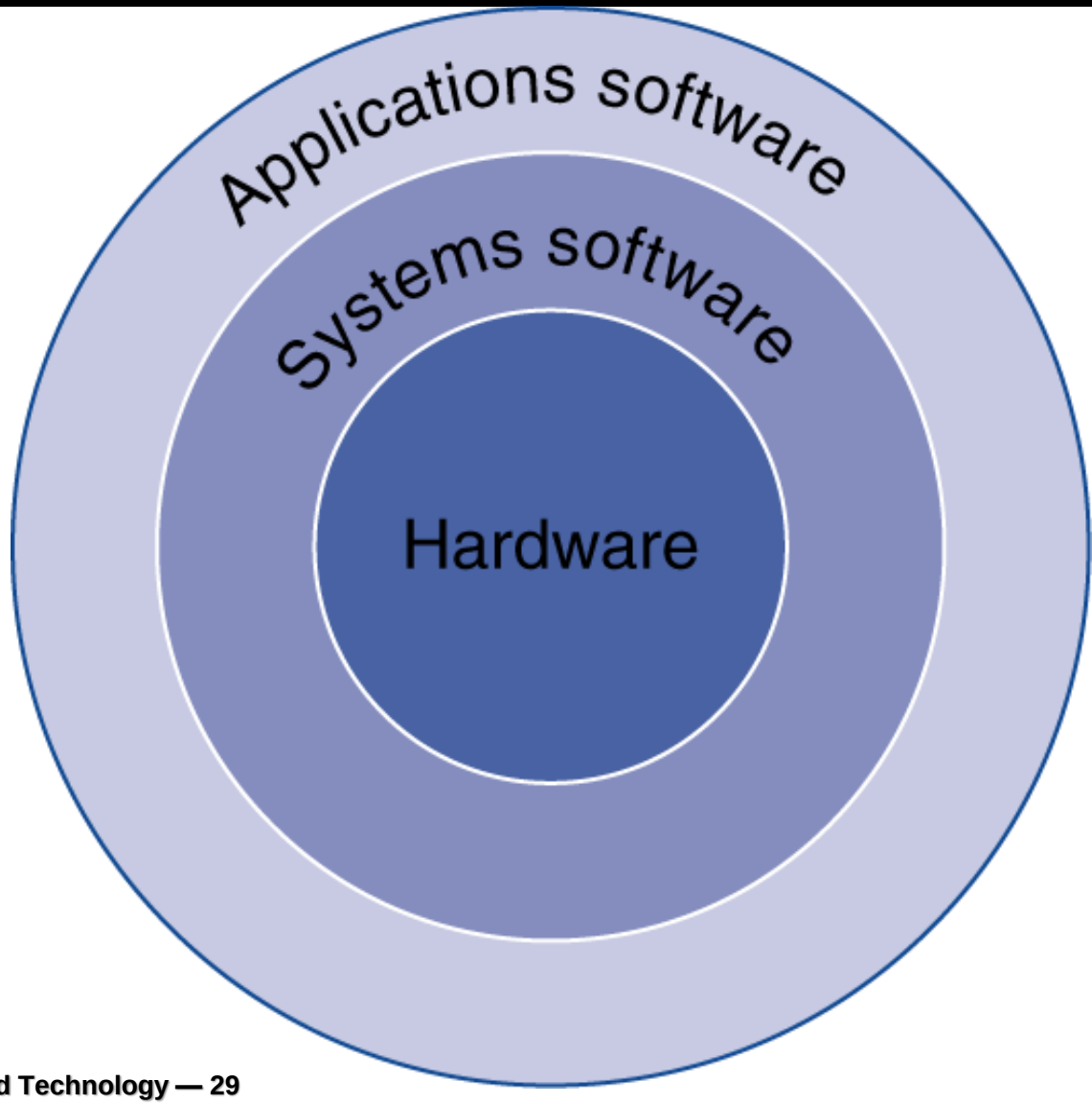
Abstração

- Investigando algo a fundo (removendo camadas), são reveladas mais informações;

- Uma abstração omite detalhes desnecessários, ajudando a reduzir a complexidade;

Below Your Program

- Software de aplicação
  - Escrito em linguagem de alto nível
- Software do sistema
  - Compilador: traduz o código HLL para código de máquina
  - Sistema Operacional: código de serviço
    - Manipulação de entrada / saída
    - Gerenciando memória e armazenamento
    - Agendamento de tarefas e compartilhamento de recursos
- Hardware
  - Processador, memória, controladores de E / S



Como se fazer entender pelo computador?

- “Linguagem” do computador : 100011000001 (*bits*)
  - Números binários: base da teoria computacional

1. Primórdios: uso da linguagem nativa em binário!!!
2. Linguagem de Montagem *(Assembly)*
   - Montador: traduz uma versão simbólica das instruções para sua representação binária na arquitetura

add A, B → montador → 100011000001

3. Linguagem de Programação de alto-nível
   - Compilador: traduz instruções de alto-nível para instruções binárias diretamente ou via um montador

A + B → compilador → add A, B → montador →100011000001

ou

A + B → compilador → 100011000001

Mais camadas

- Existe ainda um programa que gerencia os recursos da máquina durante a execução dos programas: o **SISTEMA OPERACIONAL** (SO):
  - Operações de Entrada/Saída (E/S), “carga” do programa na memória, exceções, etc.;
  - O SO funciona como um gerente dos recursos, escondendo o acesso direto ao hardware dos usuários;
  - Mais ainda: multiprocessamento, gerência de arquivos, processamento distribuído, ...;
- Assim, existem diversas camadas e serviços disponíveis para auxiliar nossa comunicação com a máquina:
  - Um modelo em camadas e serviços é uma forma interessante de abstração para a comunicação;

Exemplo de uma abstração

- O que é isto?

  00000000001000100100000000100000000000001100100010010000010000000010000100100101000010010

- Um pouco mais abstrato

  00000000 00100010 01000000 00100000
  00000000 01100100 01001000 00100000
  00000001 00001001 00101000 00100010

Subindo o nível

- Que tal decimal?

  0 34 64 32
  0 100 72 32
  1 9 40 34

- Assembly ...

  add $8, $1, $2
  add $9, $3, $4
  sub $5, $8, $9

Finalmente Compreensível

- Melhorando ainda mais...

  $8 = $1 + $2
  $9 = $3 + $4
  $5 = $8 - $9

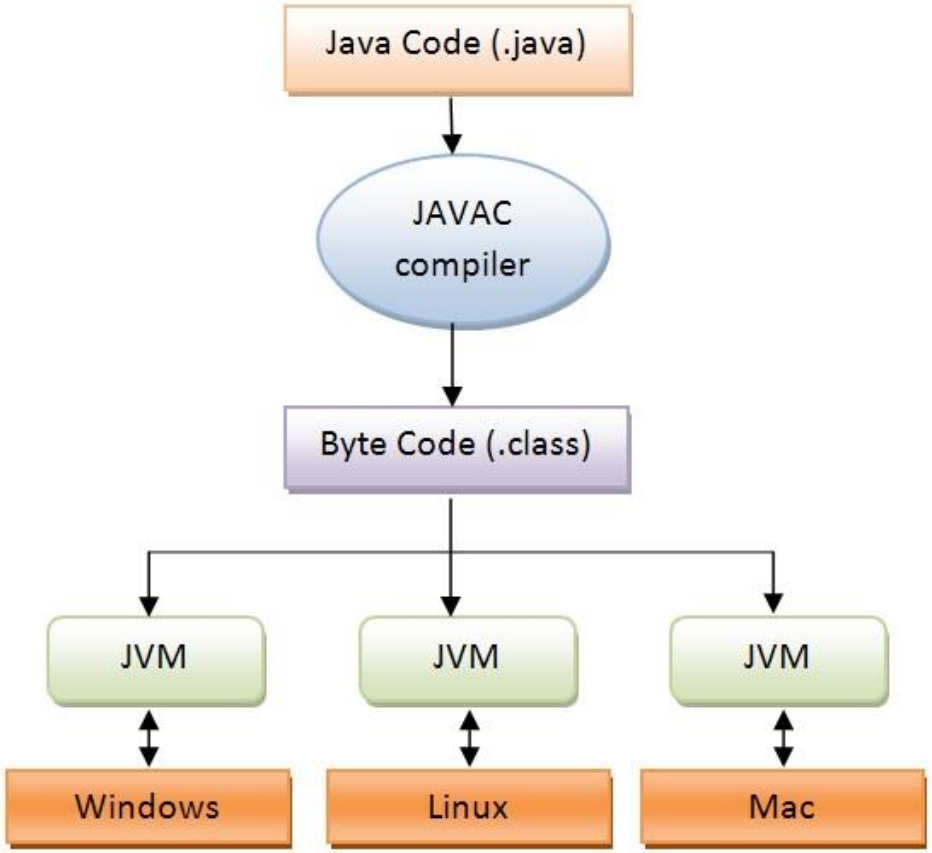
- Claro agora?

  u = a + b
  v = c + d
  x = u - v

- Sim, é claro:

  x = (a+b) - (c+d)

Java e Máquina Virtual



Código em Linguagem de Máquina MIPS

```
00100111101111011111111111100000
10101111101111110000000000010100
10101111101001000000000000100000
10101111101001010000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101110000000000000011000
00000001110011100000000000011001
00100101110010000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
00000000000000000111100000010010
00000011000011111110010000010001
00010100001000001111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
00000011111000000000000000001000
00000000000000000001000000100001
```

Mesmo código em Assembly MIPS

```
        .text
        .align    2
        .globl    main
main:
        subu      $sp, $sp, 32
        sw        $ra, 20($sp)
        sd        $a0, 32($sp)
        sw        $0,  24($sp)
        sw        $0,  28($sp)
loop:
        lw        $t6, 28($sp)
        mul       $t7, $t6, $t6
        lw        $t8, 24($sp)
        addu      $t9, $t8, $t7
        sw        $t9, 24($sp)
        addu      $t0, $t6, 1
        sw        $t0, 28($sp)
        ble       $t0, 100, loop
        la        $a0, str
        lw        $a1, 24($sp)
        jal       printf
        move      $v0, $0
        lw        $ra, 20($sp)
        addu      $sp, $sp, 32
        jr        $ra


        .data
        .align    0
str:
        .asciiz   "The sum from 0 .. 100 is %d\n"
```

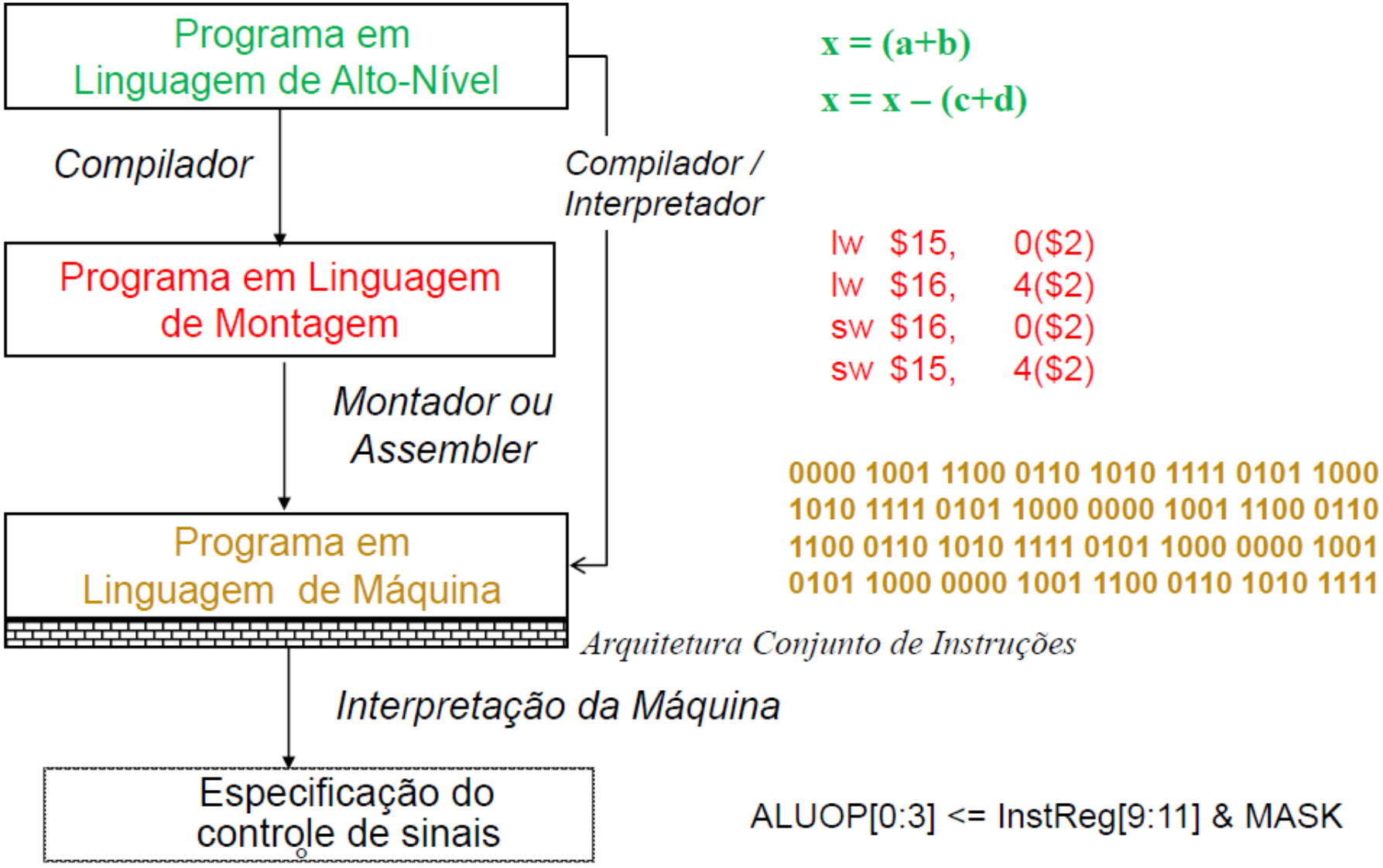
Mesma rotina anterior em C

```c
#include <stdio.h>

int
main (int argc, char *argv[])
{
    int i;
    int sum = 0;

    for (i = 0; i <= 100; i = i + 1) sum = sum + i * i;
    printf ("The sum from 0 .. 100 is %d\n", sum);
}
```

Níveis de Representação

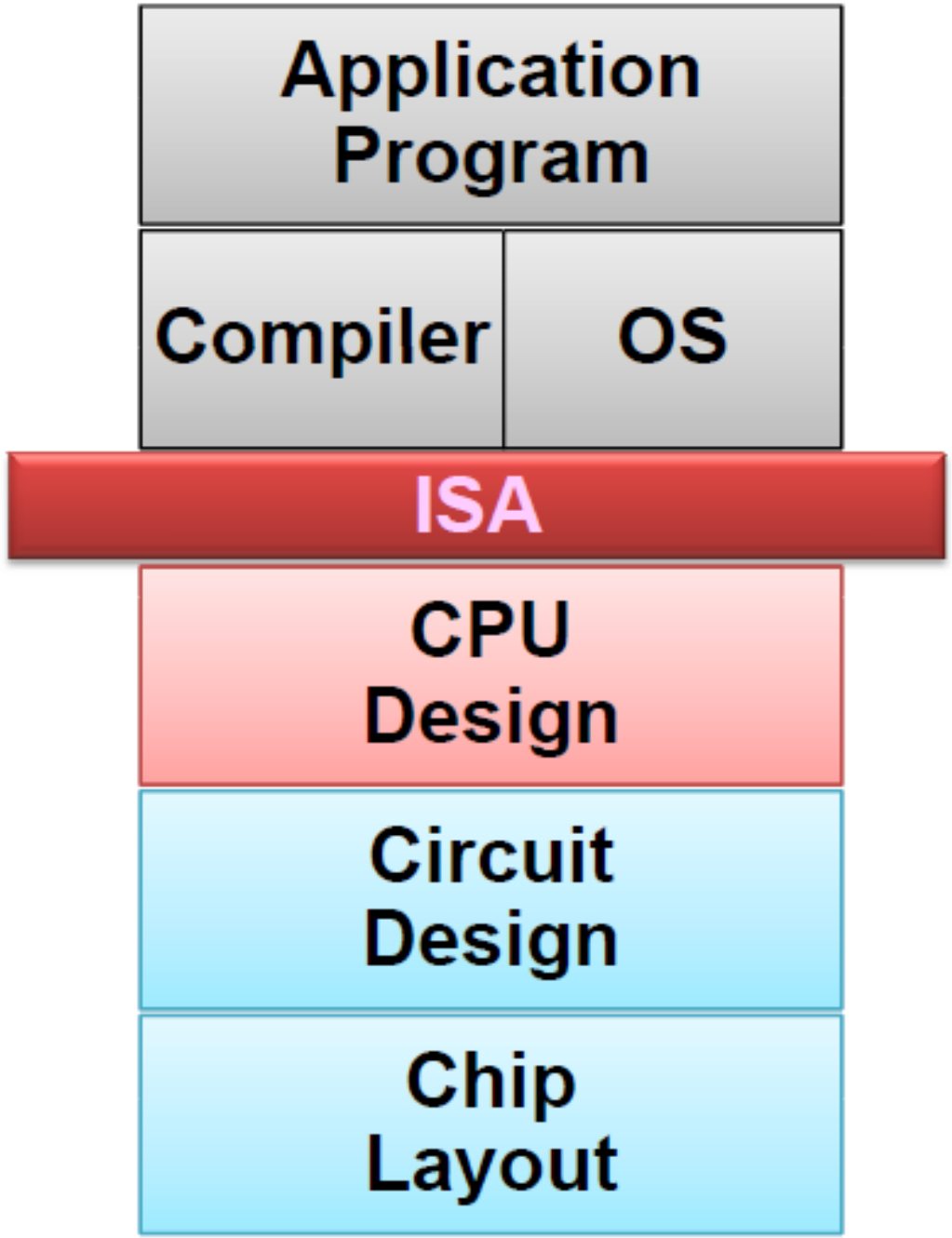


Níveis de Arquitetura

- Arquitetura do conjunto de instruções (ISA):
  - Conjunto de instruções e registradores visíveis ao programador;
  - Interface entre nível de hardware e software de baixo nível;
  - Padroniza instruções, padrões de bits da linguagem de máquina, etc.;
  - Vantagem: *Implementações diferentes da mesma arquitetura;*
  - Desvantagem: *As vezes dificulta o uso de "novas invenções";*

Níveis de Arquitetura

- Micro-arquitetura:
  - blocos tais como sistema de memória, barramentos, CPU;
  - mais de uma implementação do mesmo conjunto de instruções (AMD e Intel, 80{,1,2,3,4,5,6}86);
  - 80x86/Pentium/K6, PowerPC, DEC Alpha, **MIPS**, SPARC, HP, ARM;
- Hardware:
  - tecnologia de implementação;
  - circuitos integrados (CMOS), pipelining vs ciclo-longo;


Application Program
Compiler
OS
ISA
CPU Design
Circuit Design
Chip Layout

Perguntas?

See ya!