MAC0344 - ARQUITETURA DE COMPUTADORES

tópicos de cada aula

SLIDES 01

- arquitetura e organização
- estrutura e função
- componentes de um computador
- arquitetura de von Neumann
- como foi o meu aprendizado: arquitetura x organização

SLIDES 02

- evolução do computador
- tecnologias expressas em gerações
- história da computação
- ciclo de relógio e frequência
- ExaFLOPS e ZettaFLOPS

TOP 500

- computação paralela
- avanço da tecnologia VLSI
- primeiro computador com EXAFLOPS
- máquina da USP no top 500
- performance: crescimento exponencial
- avanço da microeletrônica - pastilhas de silício
- lei de Moore
- tamanho de um transistor
- como foi o meu aprendizado: verdadeiro x falso.

SLIDES 03

- tecnologia VLSI
- transistor MOS (chave, resistor, capacitor)
- portas lógicas NOT, NAND e NOR produzidas por transistores MOS
- analogia: circuitos elétricos × circuito de água
- Lei de Moore: número de transistores numa pastilha de silício dobra a cada 18 meses.

VLSI

- processo de fabricação do VLSI
- sala limpa: poucas partículas
- VLSI é usada para processadores e memória
- array sistólico
  - rede de processadores que calcula e passa dados ritmicamente pelo sistema
  - GOOGLE TPU: search, street view, translate
  - acelerar as computações de redes neurais em aprendizado de máquina.
- fabricantes de VLSI

SLIDES 03

- evolução do desempenho do processador
- pipelining
- pré-busca de instruções
- predicção de desvios
- paralelismo
- dependência de dados
- VLIW
  - processador super escalar
- execução especulativa
- processador multicore
- lei de Amdahl: ganho obtido com o uso de múltiplos processadores

SLIDES 04

- hierarquia de memória
- importância da memória cache em agilizar o acesso de dados e instruções
- fenômeno de localidade
- função de mapeamento
- analogia com biblioteca
- mapeamento por conjunto : mais usado em processadores modernos

SLIDES 05

- memória interna
- código de detecção / correção de erros
- DRAM : volátil
- SRAM : estática
- ROM : não-volátil
- memória flash
- código de Hamming

MAC0344 - ARQUITETURA DE COMPUTADORES

SLIDES 01 - introdução

ARQUITETURA: se refere aos atributos do sistema visíveis a um programador de linguagem de máquina, com um impacto direto na execução de um programa.

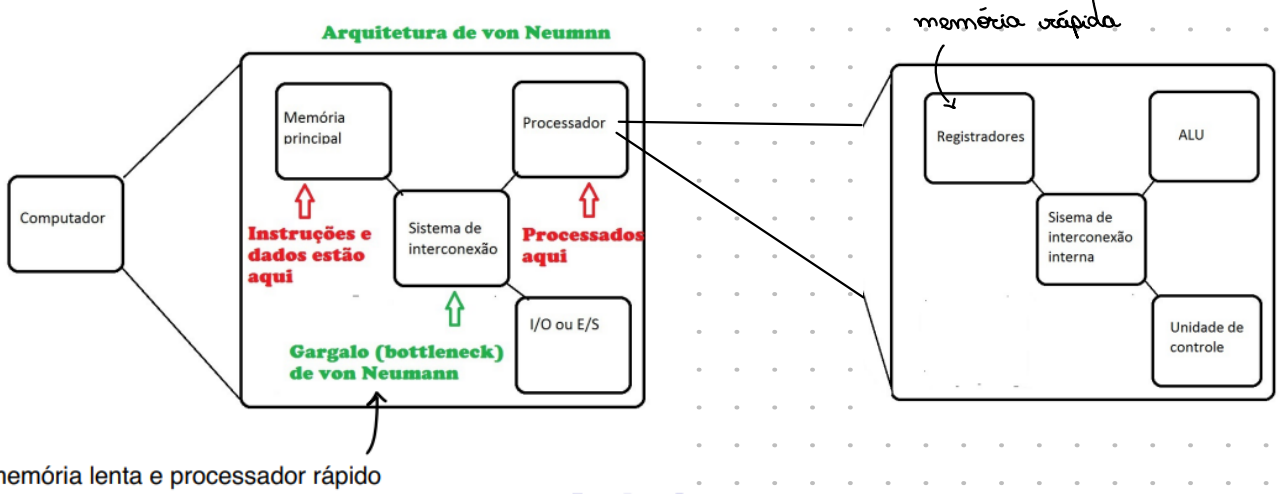
ORGANIZAÇÃO: as unidades operacionais e sua interconexão que realizam as especificações arquiteturais, invisíveis ao programador.

ESTRUTURA: a maneira em que os componentes são inter-relacionados. Como estão conectados?

FUNÇÃO: a operação de cada componente individual como parte da estrutura. Para que serve?

ESTRUTURA DE UM COMPUTADOR

Um computador tem como componentes:



- Processador ou CPU: tem a função de controlar a operação do computador e realizar o processamento de dados.
- Memória principal: a função é armazenar dados e instruções.
- I/O (ou E/S - entrada e saída): movimenta dados entre o computador e o ambiente externo.
- Sistema de interconexão: para comunicação entre CPU, memória e I/O, através de um barramento de sistema (*bus*).
- Unidade de controle: controla a operação da CPU e portanto do computador.
- ALU (unidade aritmética e lógica): realiza as operações da função de processamento de dados.
- Registradores: fornece armazenamento interno para a CPU.
- Interconexão interna: mecanismo que faz a comunicação entre a unidade de controle, ALU e registradores.

ARQUITETURA DE VON NEUMANN

É uma arquitetura de computador que se caracteriza pela possibilidade de uma máquina digital armazenar seus programas no mesmo espaço de memória que os dados, podendo assim manipular tais programas.

ARQUITETURA x ORGANIZAÇÃO

A • Representação de um número de ponto flutuante de dupla precisão.
A • Níveis de prioridade na execução de um processo.
O • Implementação do circuito somador com a técnica *carry-lookahead*.
A • Projeto do conjunto de instruções de máquina.
O • Como implementar o conjunto de instruções.
O • Usar um co-processador para aritmética de ponto flutuante.
O • Usar um co-processador especializado para processamento de imagem.
A • Técnicas de endereçamento.
O • Usar memória cache para acelerar o acesso.
O • Adotar técnicas de correção automática de erros de acesso à memória.

SLIDES 02 - evolução

- Tecnologia expressa em gerações
  - Primeira geração: válvulas
  - Segunda geração: transistores
  - Terceira geração: circuito integrado VLSI
  - Novas gerações
- Evolução caracterizada por:
  - Aumento da velocidade do processador
  - Diminuição do tamanho dos componentes
  - Aumento da capacidade de I/O e velocidade

HISTÓRIA DA COMPUTAÇÃO

- ábacos
- baguça e o sistema binário
- régua de cálculo
- geração 0: computador mecânico
- geração 1: invenção da válvula
  - as válvulas tem seu funcionamento baseado no fluxo de elétrons no vácuo.
- geração 2: invenção do transistor
  - componente de circuito eletrônico
  - aumentar e chavear os sinais elétricos.
  - primeiro computador na USP
- geração 3: circuito integrado
  - conjunto de transistores, resistores e capacitores
- geração 4: VLSI - hoje
  - Very large Scale of Integration
  - componentes eletrônicos minúsculos implementados em silício

top 500

- Vocês verão que hoje todos os supercomputadores usam computação paralela, alguns com milhões de processadores (*cores* ou núcleos).
- Isso se deve ao avanço da tecnologia VLSI (microeletrônica) com o aumento da capacidade de uma pastilha de Silício, que pode conter cada vez mais componentes eletrônicos minúsculos.
- Aumentar a frequência de relógio é uma forma de aumentar o desempenho, mas tem o limitante de dissipação de calor que impede o seu rápido aumento. Daí a solução por computação paralela.

VERDADEIRO OU FALSO

F • Pela lista TOP500, vivemos hoje na era de PetaFLOPS. já estamos em ExaFlops

V • Todos os computadores da lista TOP500 hoje possuem mais do que um processador.

F • A Lei de Moore, por ser lei, vale sempre, no presente e no futuro.

F • O Brasil ainda não conseguiu colocar nenhum computador na lista TOP500. o da USP

F • Pela Lei de Moore, a ~~frequência do relógio~~ dobra em cada 18 meses.

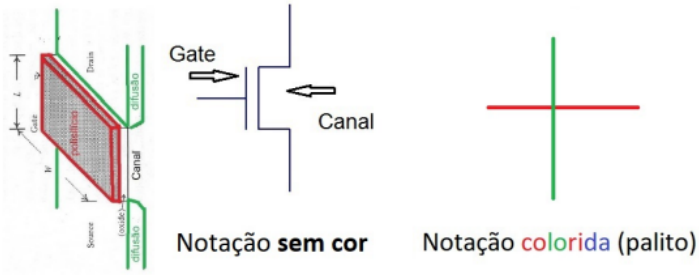
número de transistores

slides 03 - VLSI

- integra uma grande quantidade de dispositivos eletrônicos (transistores) numa pastilha (chip) de silício.
- very large scale of integration
- bilhões de transistores
- analogia: circuitos elétricos × circuito de água
- MOS = Metal Oxide Semiconductor
  - chave liga e desliga feito de semicondutor (silício - Si)
- explicação simplificada: NMOS
- tecnologia mais usada: CMOS

CHAVE LIGA-DESLIGA (MOS)

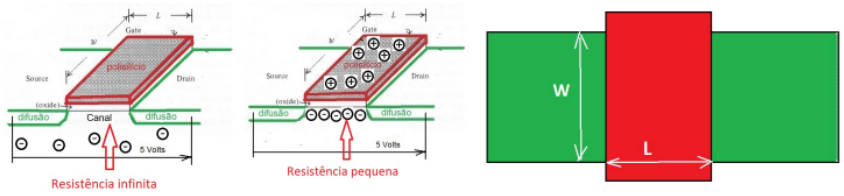
- transistor MOS: trilha de polisilício cruza uma trilha de difusão
  - chave liga e desliga
- a trilha de difusão está interrompida e não passa corrente
- injetando corrente elétrica entre o Gate e Source V_{GS}, as cargas positivas do polisilício atraem elétrons para onde a trilha de difusão está interrompida e passa corrente elétrica.



CAPACITOR (MOS)

- armazena carga elétrica
- Voltagem alta no Gate carrega cargas elétricas no capacitor. Voltagme zero no Gate descarrega as cargas do capacitor. Um transistor pode então implementar um bit de memória.

RESISTOR OU RESISTÊNCIA



Um transistor que não conduz corrente apresenta uma resistência 'infinita' pois a trilha difusão está interrompida no Canal.

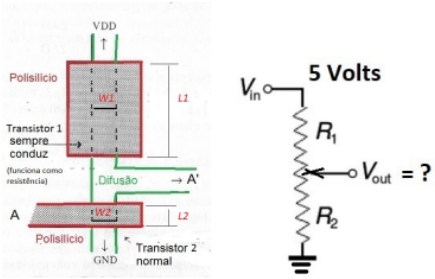
Mas um transistor conduzindo ou passando corrente possui uma pequena resistência R cujo valor é diretamente proporcional ao comprimento L e inversamente proporcional à largura W.

$$R = \alpha \frac{L}{W}, \text{ onde } \alpha \text{ é uma constante.}$$

- O comprimento L e a largura W são medidas na região de interseção entre Polissilício e Difusão (ver figura).
- L é a medida na direção do fluxo da corrente
- W é a medida ortogonal ao comprimento.

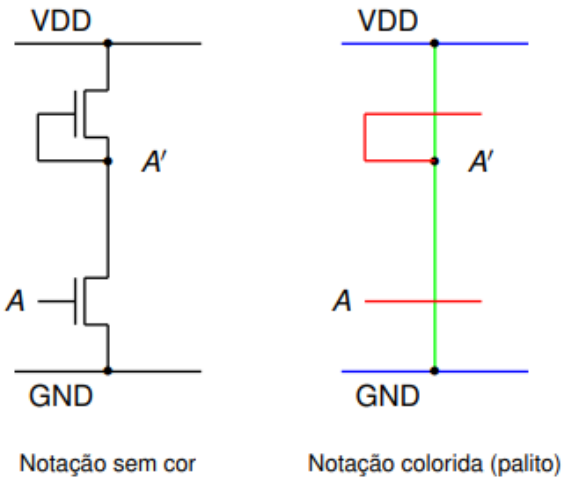
PORTA NOT

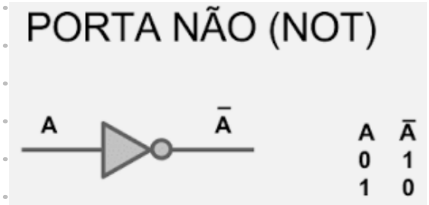
- dois transistores



Para uma porta NOT funcionar, basta fazer a resistência de condução do transistor de cima R_1 ser 4 vezes a resistência de condução do transistor de baixo R_2:

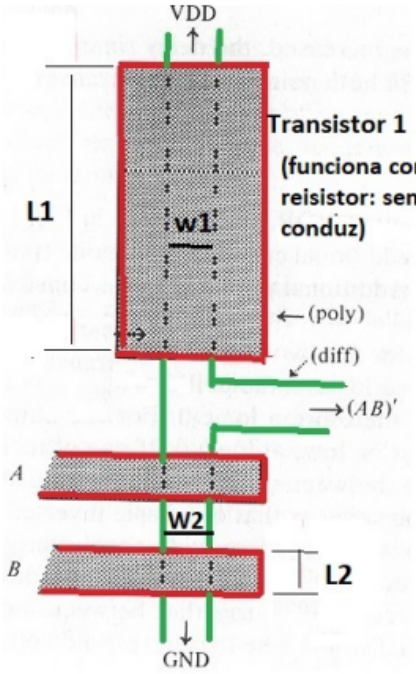
$$R_1 = 4R_2$$
$$\frac{L_1}{W_1} = 4\frac{L_2}{W_2}$$

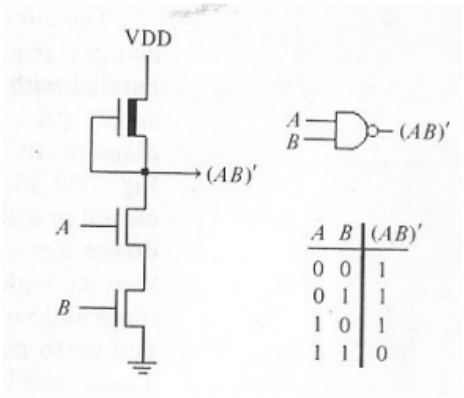




PORTA NAND

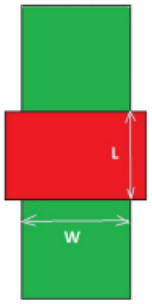
- três transistores

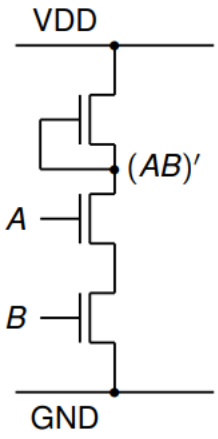




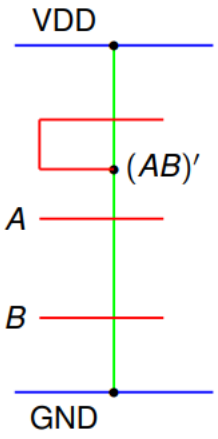
Precisamos fazer L1/W1=8 L2/W2 pois as resistências de condução dos transistores A e B se somam.

O efeito final é que a resistência do transistor 1 fica 4 vezes a resistência equivalente de A e B.

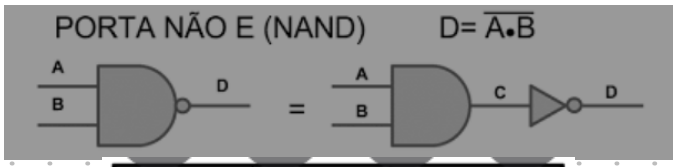




Notação sem cor



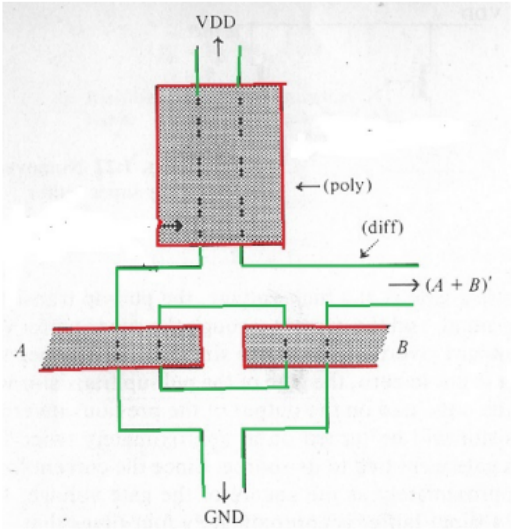
Notação colorida (palito)

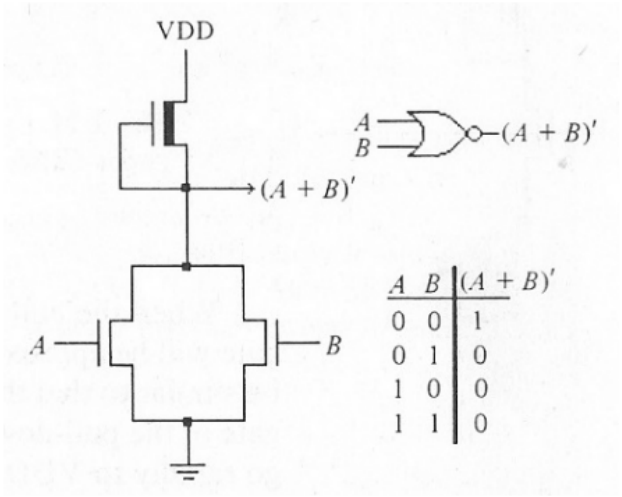


A	B	S = (A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

PORTA NOR

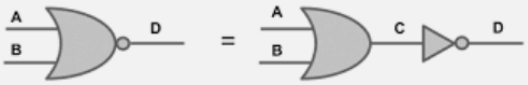
- três transistores





A	B	(A + B)'
0	0	1
0	1	0
1	0	0
1	1	0

PORTA NÃO OU (NOR) $D = \overline{A+B}$



A	B	S = (A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

slides 03 - performance

- Aumentar a frequência do relógio tem o problema de dissipação de calor. Então outras técnicas devem ser investigadas para melhorar a velocidade do processador.
- Ao longo dos anos, várias técnicas foram desenvolvidas para essa finalidade.
- Várias dessas técnicas procuram atenuar o gargalo de von Neumann: pré-busca de instruções, VLIW (very Large Instruction Word), etc.
- Várias técnicas procuram explorar o paralelismo: pipelining, processador superescalar, processadores multicore, execução foram de ordem ou de forma concorrente (se não houver dependência), etc.
- Lei de Amdahl sobre a limitação da computação paralela.

DEPENDÊNCIA DE DADOS

- Dependência verdadeira ou de fluxo
- Anti-dependência
- Dependência de saída

Anti-dependência e dependência de saída podem ser removidas renomeando variáveis.

- Dependência verdadeira ou dependência de fluxo ou *Read-After-Write* (RAW): quando uma instrução depende do resultado de outra.

  Modelo: $A = \ldots$
  $\ldots = A \ldots$

  1: $A = A + 2$
  2: $B = 2 \times A$
  3: $C = B - A$

  Instrução 2 depende verdadeiramente da instrução 1 (escrevemos $1 \rightarrow^{v} 2$).

  Instrução 3 depende verdadeiramente da instrução 1 (escrevemos $1 \rightarrow^{v} 3$).

  Instrução 3 depende verdadeiramente da instrução 2 (escrevemos $2 \rightarrow^{v} 3$).

- Anti-dependência ou *Write-After-Read* (WAR): quando uma instrução usa uma variável que depois vai ser alterada: a ordem de executar essas duas instruções não pode ser alterada, nem executadas em paralelo.

  Modelo: $\begin{array}{l} \ldots = A \ldots \\ A = \ldots \end{array}$

  1: $B = A + 5$
  2: $A = 7$

  A instrução 2 anti-depende da instrução 1 (escrevemos $1 \rightarrow^{anti} 2$):

- Suponha que gostaríamos muito de poder executar as instruções 1 e 2 ao mesmo tempo. Isso é possível se removermos a anti-dependência. Veremos isso agora.

- Anti-dependência pode ser removida ao renomear variáveis. Isso permite executar instruções que tinham anti-dependências em paralelo.

Modelo da anti-dependência	Remoção da anti-dependência
	$0 : A1 = A$
$1 : \ldots = \ldots A \ldots$	$1 : \ldots = \ldots A1 \ldots$
$2 : A = \ldots$	$2 : A = \ldots$

- A instrução $0 : A1 = A$ deve ser executada antes das outras duas instruções.
- Depois disso, as instruções 1 e 2 podem ser executadas em qualquer ordem. A anti-dependência foi removida.

  Sejam as duas instruções com anti-dependência.

  1: $B = A \times X$
  2: $A = Y \times Z$

  Renomeamos a variável A:

  0: $A1 = A$
  1: $B = A1 \times X$
  2: $A = Y \times Z$

  Após a renomeação da variável na instrução 0 e a execução da instrução 0, podemos executar instruções 1 e 2 em paralelo. Mas note que introduzimos uma dependência verdadeira entre as instruções 0 e 1.

- Dependência de saída ou *Write After Write* (WAW): quando a ordem das instruções afeta o valor final de saída de uma variável.

Modelo:
$A = \ldots$
$\ldots$
$A = \ldots$

1: $A = X * X$
2: $B = A + 5$
3: $A = Y * Y$

Instução 3 tem dependência de saída em relação à instrução 1 (escrevemos $1 \rightarrow^{saida} 3$).

- Suponha que gostaríamos muito de poder executar as instruções 1 e 3 ao mesmo tempo. Isso é possível se removermos a dependência de saída. Veremos isso agora.

- Dependência de saída também pode ser removida ao renomear variáveis.

Modelo da dependência de saída	Remoção da dependência de saída
$1 : A = \ldots$	$1 : A1 = \ldots$
$2 : \ldots = \ldots$ se aparecer $A \ldots$	$2 : \ldots = \ldots$ trocar por $A1 \ldots$
$3 : A = \ldots$	$3 : A = \ldots$

- A instrução $1 : A1 = \ldots$ deve ser executada antes da instrução 2.
- As instruções 1 e 3 podem ser executadas em qualquer ordem. A dependência de saída foi removida.

Considerem instruções 1 e 3 com dependência de saída:

1: $A = X * X$
2: $B = A + 5$
3: $A = Y * Y$

Renomeanos a variável A:

1: $A1 = X * X$
2: $B = A1 + 5$
3: $A = Y * Y$