MC202 - Estruturas de Dados

Guilherme P. Telles

IC

3 de março de 2023

Avisos

- Estes slides contêm erros.
- Estes slides são incompletos.
- Estes slides usam português anterior à reforma ortográfica de 2009.

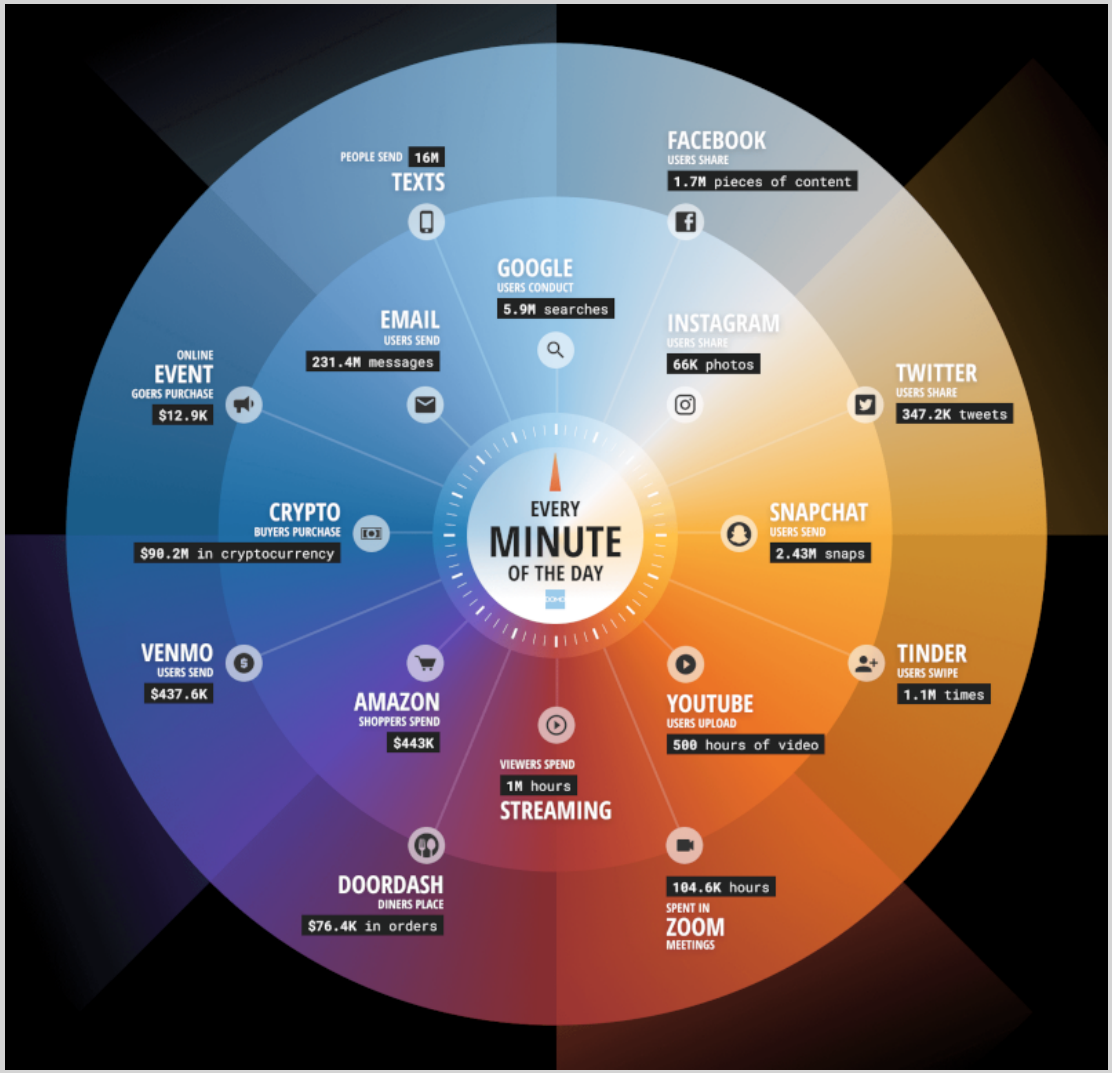
porque estruturas de dados?

Estrutura de dados

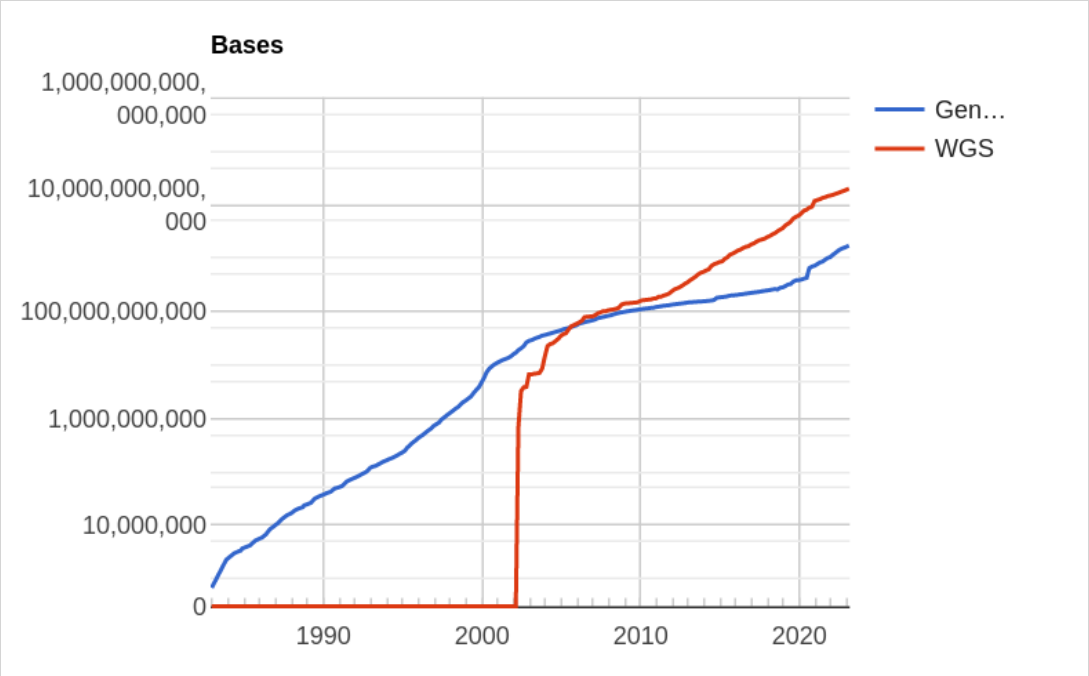
- Uma **estrutura de dados** é uma forma de organizar dados na memória para permitir operações eficientes sobre eles.

Por que estudar/pesquisar estruturas de dados?

- O volume de dados cresce muito rapidamente.
- O número de operações que se deseja realizar sobre os dados é muito grande.



[Data never sleeps 10.0, 2022, domo.com]



[www.ncbi.nlm.nih.gov/genbank/statistics]

Alternativas

- Adicionar mais memória RAM: alto custo, dados crescem muito mais rápido.
- Processar dados em memória secundária (SSD, HDD) ou na nuvem: ordens de grandeza mais lento.
- Processar dados em paralelo: alto custo, complexidade da paralelização e dependências entre dados, dados crescem muito mais rápido.
- Boas escolhas/bons projetos de estruturas de dados: podem permitir ganhos muito grandes de desempenho com a mesma quantidade de memória e CPUs.

Exemplo

- O genoma humano tem aproximadamente 3 bilhões de bases.
- Uma operação de interesse amplo é encontrar as ocorrências de uma seqüência de letras (padrão) no genoma.

  Por exemplo, encontrar onde o padrão

  `CTCAAAAGTCTAGAGCCACCGTCCAGGGAGCAGGTAGCT`

  ocorre no genoma, se ocorrer.

- Temos 4 bases no DNA (A, C, G, T), 2 bits são suficientes para codificá-las.
- Podemos então colocar todas as letras que representam o genoma humano na memória usando 750 megabytes de RAM.

Encontrar um padrão no genoma

ℓ é o tamanho do padrão
g é o tamanho do genoma
occ é o número de ocorrências do padrão no genoma
σ é o tamanho do alfabeto (4 para DNA)

proporcional significa que há fatores constantes multiplicativos

método	espaço aprox.	tempo prop. a
comparação direta (eficiente)	750 MB	$\ell + g + occ$
árvore de sufixos	31 GB	$\ell + occ$
vetor de sufixos	12 GB	$\ell + \log_2 g + occ$
FM-index	5.5 GB	$\ell \log_2 \sigma + occ \log_2 g$

Todos os genomas

- Todos os 59519 genomas no Genbank (23/fev/2021) têm juntos 4.088×10^{12} bases.
- O tamanho do FM-index seria mais ou menos 7.4 Tb.
- Para volumes de dados dessa ordem, as estruturas de dados são semi-externas: mantêm parte dos dados na memória e combinam estratégias de uso da RAM e da memória secundária.

Elementos

- Uma **estrutura de dados** é uma forma de organizar dados na memória para permitir operações eficientes sobre eles.
  - organizar dados na memória: manipulação de memória.
  - dados: a composição dos dados e a forma como mudam.
  - operações: quais são e qual a freqüência delas.
  - eficiência: tempo e memória.

MC202

- Linguagem C: manipulação da memória.
- Estruturas de dados básicas.
- É uma disciplina teórica e prática.

revisão/contexto

Computador

- É uma máquina que essencialmente faz só três coisas:
  1. recebe dados,
  2. faz contas e armazena os resultados, e
  3. devolve dados.
- Mas os computadores não fazem um conjunto fixo de contas: eles podem ser programados.
- E eles fazem contas muito rapidamente.

Programar

- Programar um computador é especificar instruções para ele.
- A programação de um computador é feita em uma linguagem de programação.

Linguagens são muitas

- Muitas linguagens de programação já foram propostas.
- Algumas foram projetadas para programar principalmente certos tipos de aplicações (científicas, comerciais, web etc.)
- Outras foram projetadas para serem flexíveis o bastante para serem consideradas linguagens de uso geral.

- As linguagens de programação podem ser classificadas de acordo com vários critérios.
- Uma distinção importante entre linguagens de programação é o nível de detalhes que o programador tem que saber sobre o computador.

- Em uma linguagem de baixo nível (p.ex. assembly), os detalhes do computador estão explícitos todo o tempo (número de registradores da CPU, instruções de máquina, comunicação entre componentes, sinais, temporização etc).
- O modelo fornecido pela linguagem é muito distante das aplicações práticas.
- Em baixo nível o controle da CPU e da memória é completo e é possível fazer programas extremamente eficientes.
- Mas é mais difícil programar, testar e encontrar erros.

- Em uma linguagem de alto nível (p.ex. Python), os detalhes da máquina estão quase todos escondidos.
- Em alto nível é mais fácil programar, testar e encontrar erros.
- Mas há um uso maior de CPU e memória e conseqüentemente de tempo para a execução do programa.

- C oferece um modelo que está próximo da máquina e permite construir código eficiente, mas sem estar atrelado a detalhes de uma máquina em particular.
- É freqüentemente chamada de “linguagem de nível médio”.

Binários

- Computadores são máquinas digitais binárias: toda informação (dados e programas) é armazenada e processada como sinais elétricos com dois níveis de tensão que representam seqüências de dígitos binários.
  - A palavra **bit** é usada para dar nome a um dígito binário.
  - Um bit pode ter valor 0 ou valor 1.
  - A palavra **byte** é usada para uma seqüência de 8 bits.

- A base 16 é conveniente para representar números binários:

base 10	base 2	base 16	base 10	base 2	base 16
0	0	0	8	1000	8
1	1	1	9	1001	9
2	10	2	10	1010	A
3	11	3	11	1011	B
4	100	4	12	1100	C
5	101	5	13	1101	D
6	110	6	14	1110	E
7	111	7	15	1111	F

- Para representar um número binário em hexadecimal, agrupam-se os bits de 4 em 4, a partir do menos significativo:

  $101 = 0101 = 5$
  $1010 = A$
  $10 = 0010 = 2$
  $11000111011010 = 11\ 0001\ 1101\ 1010 = 0011\ 0001\ 1101\ 1010 = 31DA$

- Quando uma informação é armazenada em computador ela é convertida para binário fazendo as operações necessárias.
  - P.ex. o sinal elétrico produzido ao apertarmos uma tecla é capturado como bytes que representam um símbolo,
  - luz é captada por sensores, discretizada e convertida em uma seqüência de bytes que representam cada ponto de uma imagem fotográfica,
  - o valor da temperatura de um ambiente é discretizado e convertido em bytes que representam um número.

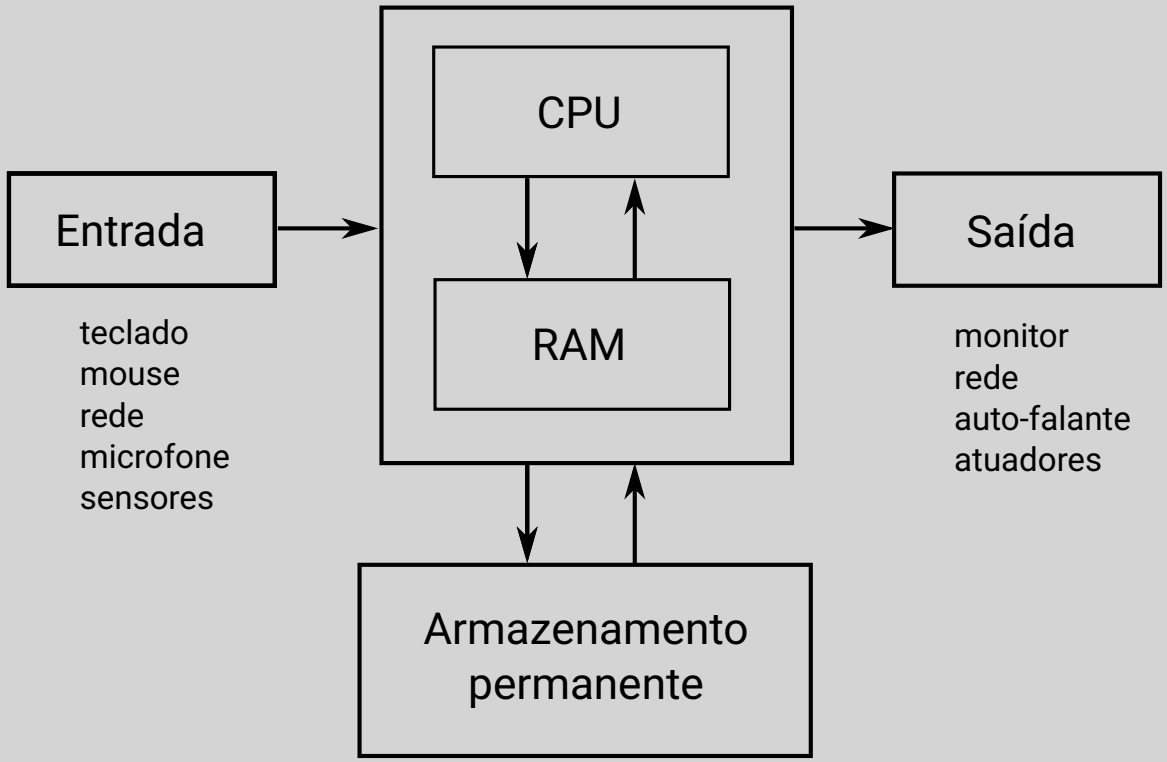
- Quando uma informação em computador é transmitida para pessoas ela é convertida de binário para uma forma adequada.
  - P.ex. um arquivo MP3 é convertido em som,
  - um arquivo texto é mapeado na emissão de luz na forma de letras e símbolos.

Componentes

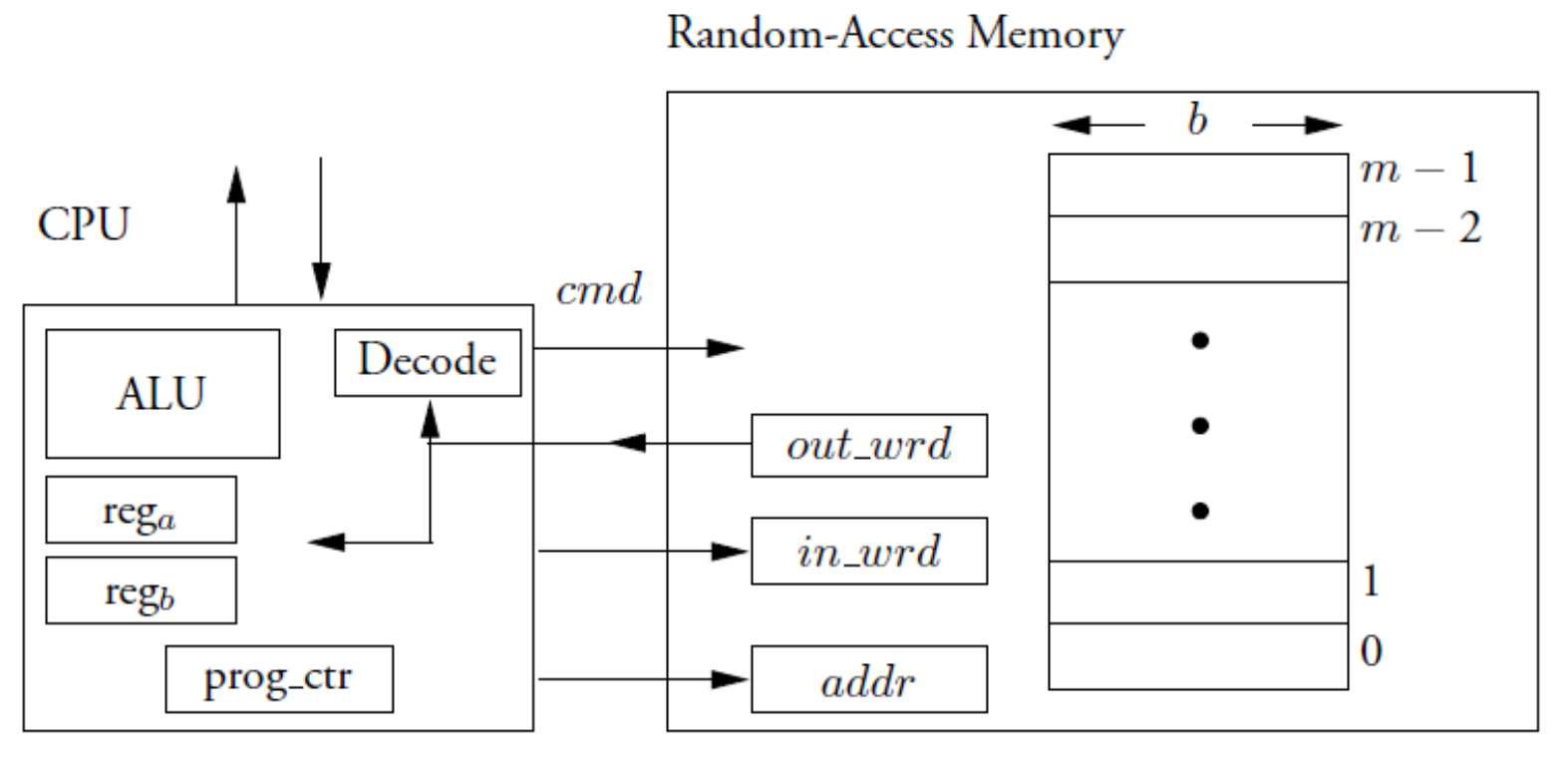
- Um computador é um sistema que tem vários componentes tangíveis e intangíveis.
- Usamos os termos:
  - *Hardware* para os componentes físicos.
  - *Software* para os programas.
  - *Firmware* para programas que são fortemente acoplados ao hardware, *i.e.* o hardware não funcionaria sem eles nem eles funcionariam em outro hardware.

Hardware

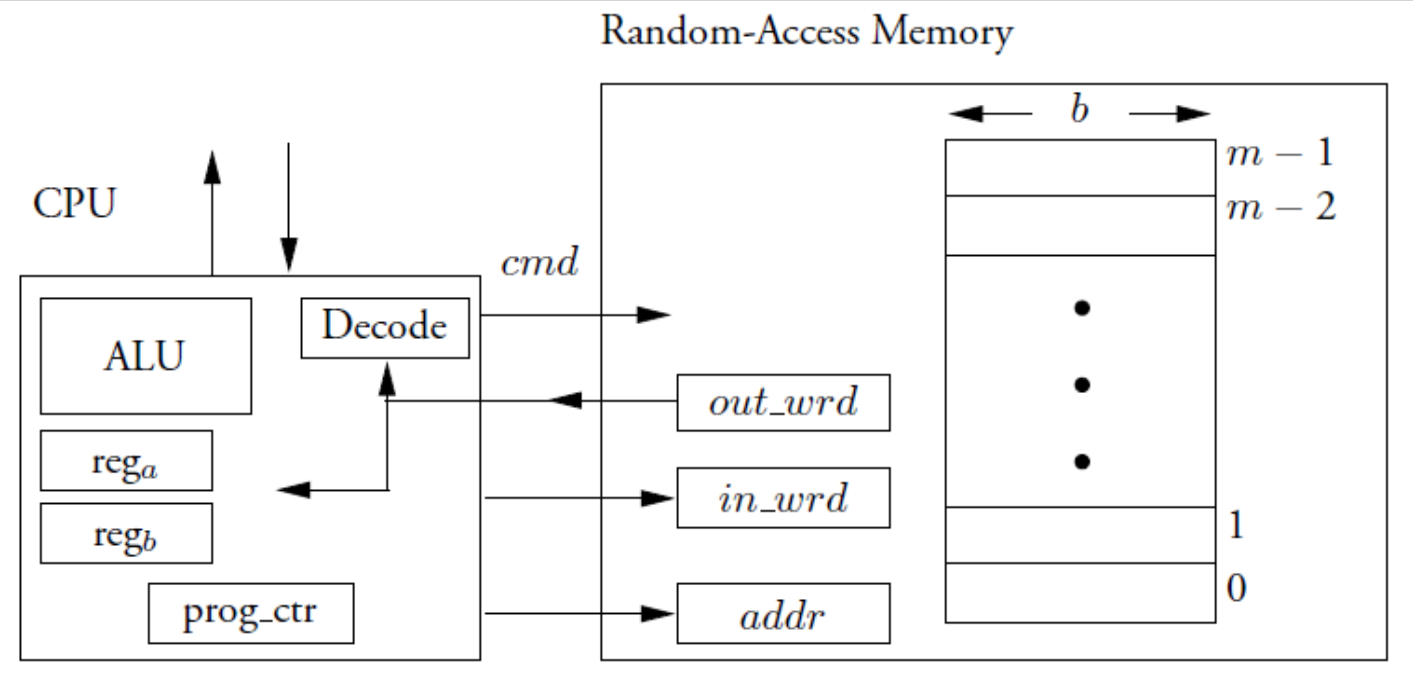
- Principais elementos:
  - *memória* (RAM, random access memory): armazena os dados e programas durante o processamento.
  - *processador* (CPU, central processing unit): realiza operações que mudam o estado da memória (*i.e.* fazem as contas).
  - *dispositivos de armazenamento permanente*: armazena dados e programas mesmo quando a energia é desligada (discos rígidos (HDD), memórias de estado-sólido (SSD) etc.)
  - *dispositivos de entrada e saída*: teclado, mouse, câmera, vídeo, wi-fi, bluetooth etc.


CPU
RAM
Entrada
teclado
mouse
rede
microfone
sensores
Saída
monitor
rede
auto-falante
atuadores
Armazenamento
permanente

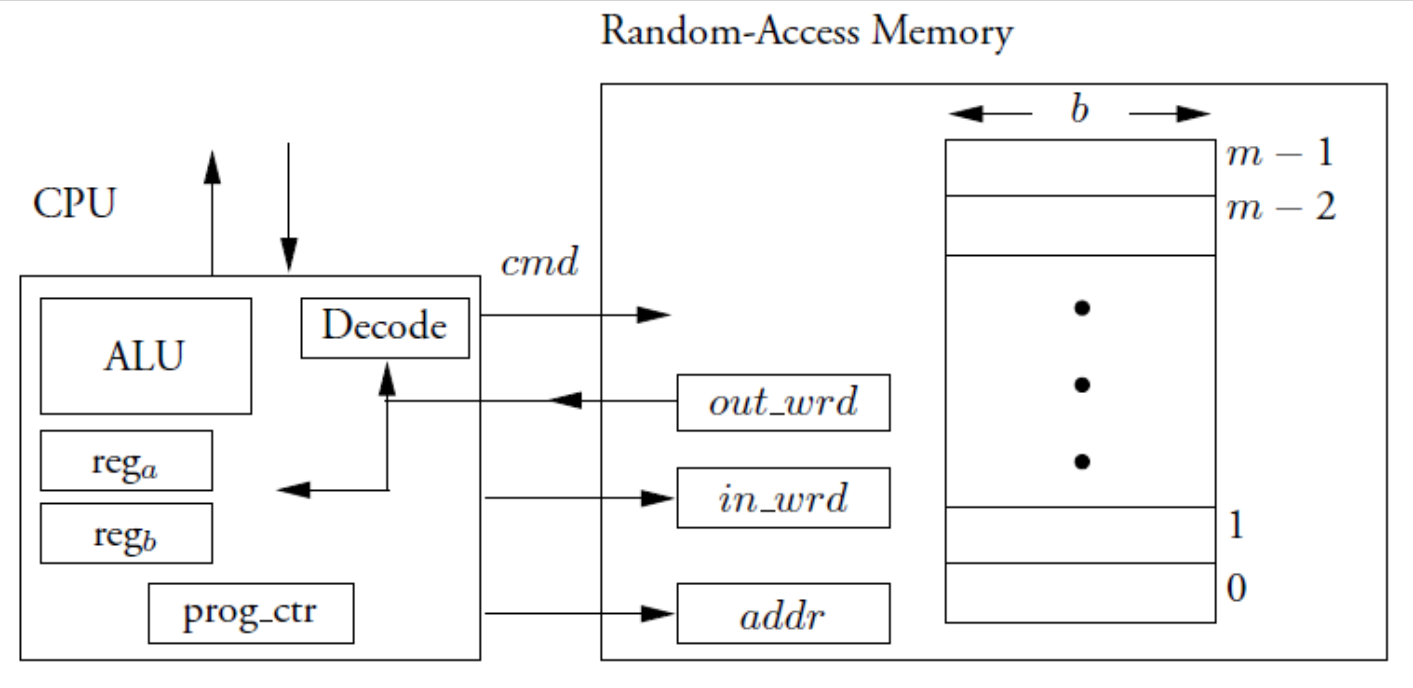
- A memória RAM também é chamada de memória principal ou memória primária.
- O armazenamento permanente também é chamado de memória secundária.



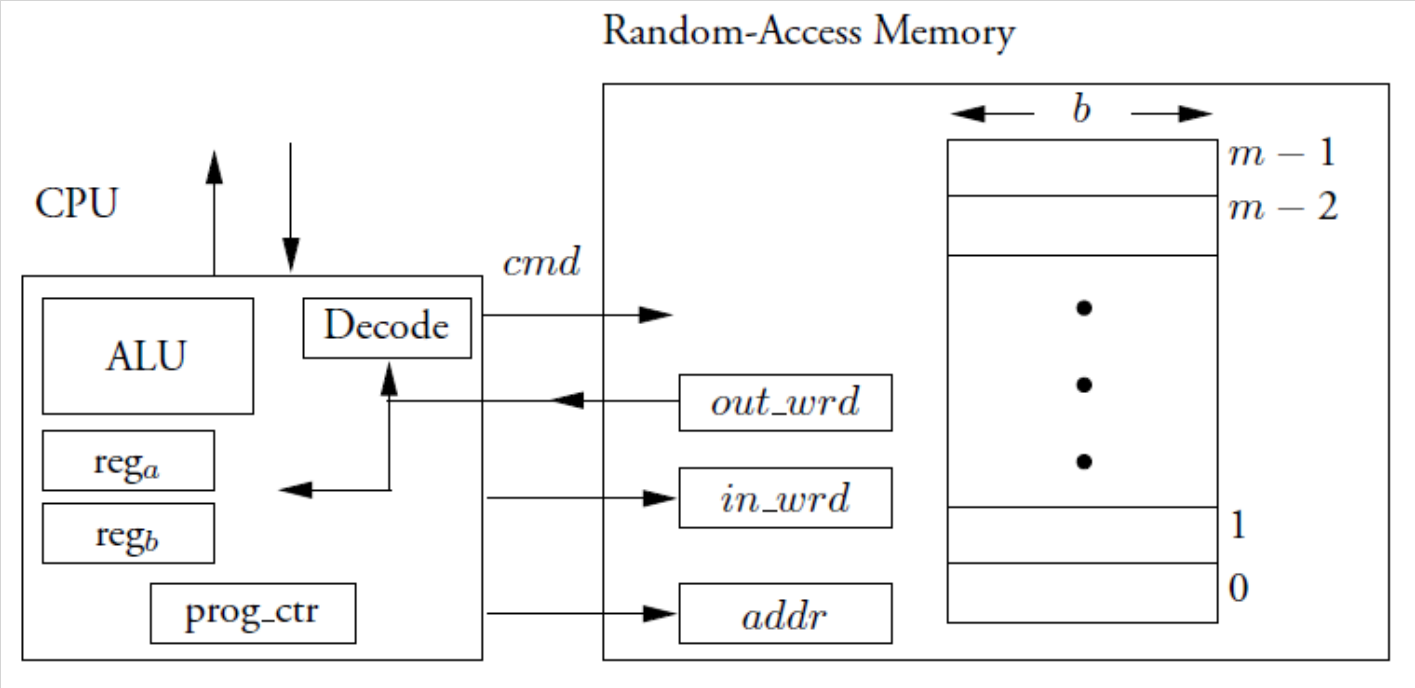
[Savage, Models of Computation, 1998]



- A memória tem n posições indexadas $0, 1, 2, ..., n-1$ que podem armazenar um número fixo de bits b cada.
- Cada instrução de um programa ou dado ocupa uma ou mais posições consecutivas de memória.



- A CPU tem registradores de b bits cada, que armazenam operandos.
- A CPU tem uma unidade lógica e aritmética (ALU) que realiza operações relacionais e aritméticas entre dados nos registradores.
- A CPU começa com um programa armazenado a partir da posição x da memória e com o endereço x armazenado no *contador de programa*.



- A cada passo de computação, a CPU traz uma instrução da memória, decodifica essa instrução (i.e. identifica a instrução), executa a instrução e modifica o contador de programa para que ele tenha o endereço da próxima instrução.

- As instruções são relativamente simples.
  - ler uma posição da memória para um registrador,
  - escrever de um registrador em uma posição da memória,
  - fazer operações aritméticas envolvendo valores nos registradores,
  - fazer testes relacionais envolvendo valores nos registradores,
  - escolher a próxima instrução que vai ser executada com base em alguma condição envolvendo valores (if/while).

- Muitas linguagens de programação têm uma estrutura que coincide com a forma de operação das CPUs.
- Os programas são construídos em torno de variáveis, testes e repetições.
- Essas linguagens são chamadas imperativas e incluem Python e C.

- Esse modelo básico de computador é útil para entender bem programas escritos em C.
- CPUs mais modernas têm vários mecanismos adicionais para permitir que elas executem muitas instruções por segundo, mas em essência todas operam no ciclo leitura-decodificação-execução.

Hierarquia de memórias

- O armazenamento de dados em um computador é organizado como uma hierarquia de memórias: registradores, cache nível 1, cache nível 2, cache nível 3, RAM, SSD, HD.
- Ao longo da hierarquia as memórias se tornam maiores e mais lentas.
- Tipicamente, o tempo necessário para transferir um único byte ou um bloco de bytes consecutivos entre dois níveis da hierarquia é o mesmo.
- A CPU gerencia a transferência de blocos de dados ao longo da hierarquia.

- A hierarquia de memórias tem impacto nas estruturas de dados:
  - operar dados seqüencialmente quase sempre é mais rápido que operá-los de forma não-seqüencial.
  - operar o mesmo dado várias vezes quase sempre é mais rápido que operar vários dados distintos.

Evento	Latência	Latência em escala humana aprox.
Acesso a registrador de CPU a 3.3GHz	0.3 ns	1 s
Acesso a cache nível 1	0.9 ns	3 s
Acesso a cache nível 2	2.8 ns	9 s
Acesso a cache nível 3	12.9 ns	43 s
Acesso a memória principal (DRAM)	120 ns	6 m
SSD I/O	50-150 μs	2-6 dias
HDD I/O	1-10 ms	1-12 meses
Internet: SF a NY	40 ms	4 anos
Internet: SF a Londres	81 ms	8 anos
Internet: SF a Sydney	183 ms	19 anos

[Systems Performance: Enterprise and the Cloud, Brendan Gregg, 2013]

Software

- Software são os programas que podem ser executados em um computador.
- Costumamos pensar no software em duas partes:
  - *sistema operacional*, p.ex. Windows, Linux, Android etc.
  - *aplicações de usuários*: são os nossos programas, nossas apps. Exemplos são jogos, office, navegadores para internet, programas de controle de um robô industrial, os trabalhos de MC202 etc.

Sistema operacional

- Um sistema operacional é um conjunto de programas que realiza várias tarefas, dentre elas:
  - Faz a carga de cada programa que vai ser executado, *i.e.* recupera o programa do armazenamento permanente e coloca o programa na memória adequadamente.
  - Faz o gerenciamento e alocação dos componentes (CPU, rede, memória, armazenamento etc.) dentre todos os programas que estão sendo executados ao mesmo tempo pelo computador.
  - Protege cada programa e os dados dele de outros programas.
  - Detecta erros durante a execução de programas e durante a interação entre os componentes.

- Quando um programa nosso é executado ele interage com vários componentes do sistema operacional.
- A parte que a gente vê (o sistema de janelas ou linha-de-comandos) vem junto com o sistema operacional mas também é considerada uma aplicação de usuário.

Recursos

- Apesar de serem rápidos, qualquer operação que um computador realiza leva tempo.
- Os dados que um programa lê ou produz ocupam parte da memória.
- Programas também podem usar recursos como rede ou outros dispositivos.
- Quando temos muitos dados ou problemas complexos, o tempo de execução ou a demanda por recursos de um programa pode se tornar crítico.

Programar

- Programar um computador é especificar instruções para ele.
- A programação de um computador é feita em uma *linguagem de programação*.
- Um programa em uma linguagem de programação é um texto para ser lido por pessoas.

Compilação, interpretação

- O computador não executa o programa diretamente, é preciso haver um mapeamento entre a linguagem de programação e instruções da CPU.
- Esse mapeamento pode pode ser através de compilação, interpretação, ou ambos.

- Compilação: traduz o programa em instruções da CPU. O programa traduzido pode ser executado diretamente pelo computador (C).
- Interpretação: o programa é executado por outro programa, o interpretador. Não há tradução.
  O interpretador pode ser visto como um programa que simula um computador capaz de executar um programa naquela linguagem de programação.
- Híbridos: o programa é traduzido em um programa em uma linguagem intermediária. O programa na linguagem intermediária é executado por um interpretador (Python).

- Uma vantagem da compilação é permitir verificações e otimizações do código antes do programa ser executado. Uma desvantagem é que a compilação de sistemas grandes com otimizações sofisticadas é demorada.
- Tipicamente, linguagens interpretadas têm uma estrutura mais flexível. Uma vantagem da interpretação é permitir a execução interativa do programa. Uma desvantagem é que o desempenho é pior.
- Os híbridos buscam um compromisso entre desempenho e flexibilidade.

conceitos relacionados a programas

Programa, expressões, sentenças

- Um programa é uma seqüência de sentenças.
- Uma expressão é uma combinação de variáveis, constantes e chamadas de função conectadas por operadores.
- Uma sentença é uma combinação de palavras reservadas, expressões e operadores que realiza uma ação.

Variável

- Uma ou mais posições de memória associadas a um nome e a um tipo e que tem um valor.
- O valor da variável é a informação que está armazenada nessas posições de memória.
- O valor da variável pode ser modificado, mas o nome não pode.

Tipo

- Define o significado dos bits que compõem uma variável e quais operadores podem ser aplicados a ela.

Função

- Um bloco delimitado de sentenças de uma linguagem de programação que tem um nome.
- Pode haver definição de variáveis dentro do bloco.
- Uma função recebe parâmetros. Dentro da função, um parâmetro se comporta como uma variável.
- O nome da função é usado para chamar a função.
- A chamada da função causa a execução dos comandos que a compõem.
- O ponto de entrada da função é único.
- Outros nomes são sub-rotina, sub-programa, procedimento.