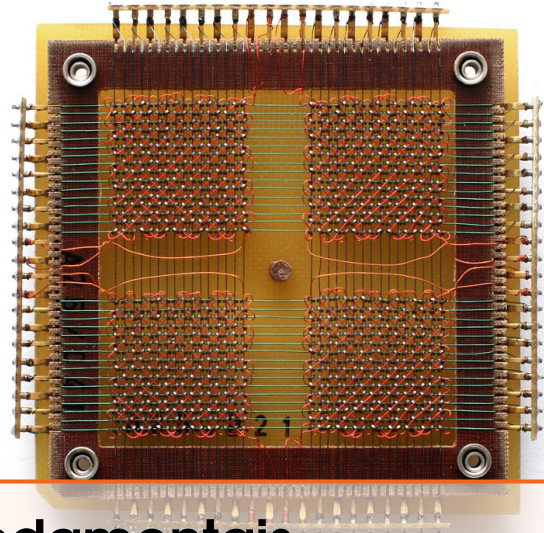


Sumário

1	Conceitos fundamentais	3
1.1	Linguagens de programação	3
1.2	Sistemas operacionais	3
1.3	O que são compiladores	3
1.4	Ambientes de desenvolvimento	3
2	Visão Geral da linguagem C	5
2.1	Origens da linguagem	5
2.2	Porque aprender C?	5
2.3	Onde C é usado?	5
2.4	Estrutura de um programa em C	5
2.5	Meu primeiro programa em C	5
2.6	Compilando meu programa	5
2.7	Executando	6
2.8	Entendendo meu programa	6
3	Variáveis, Tipos de Dados e Expressões Aritméticas	7
3.1	Tipo de dados básicos	7

3.2	Variáveis	7
3.3	Modificadores	7
3.4	Constantes	7
3.5	Expressões Aritméticas	7
3.5.1	Operação de resto (%)	7
3.6	Conversão de tipo de dados	7
4	Estruturas de Controle, Operadores Logicos e Relacionais	9
4.1	Operadores relacionais	9
4.2	Operadores lógicos	9
4.3	O comando if	9
4.3.1	O comando else	9
4.3.2	O comando if-else-if	9
4.3.3	Ifs aninhados	9
4.4	O comando switch	9
4.5	Operador ternário(?)	9
5	Estruturas de repetição (Loops)	11
5.1	Comando for	11
5.2	Comando while	11
5.3	Comando do-while	11
5.4	Controle de loops	11
5.4.1	O comando break	11
5.4.2	O comando continue	11
5.5	Loops aninhados	11
5.6	Loops infinitos	11
6	Vetores (Arrays)	13
6.1	Trabalhando com vetores	13
6.1.1	Declarando vetores	13
6.1.2	Inicializando vetores	13

6.1.3	Acessando vetores	13
6.2	Vetor de caracteres	13
6.3	Aplicacoes com vetores	13
6.3.1	Imprimir os elementos	13
6.3.2	Soma dos elementos	13
6.3.3	Reverter o vetor	13
6.3.4	Ordenacao de vetor	13
6.4	Vetores Multidimensionais	13
7	Ponteiros	15
7.1	Armazenamento Primário	15
7.1.1	Memória Principal	16
7.2	Usando Ponteiros	16
7.3	Usando Vetores	16
7.4	Vetores NÃO são Ponteiros	16



1. Conceitos fundamentais

Conceitos

- item

1.1 Linguagens de programação

Linguagens de programação.

1.2 Sistemas operacionais

Linux.

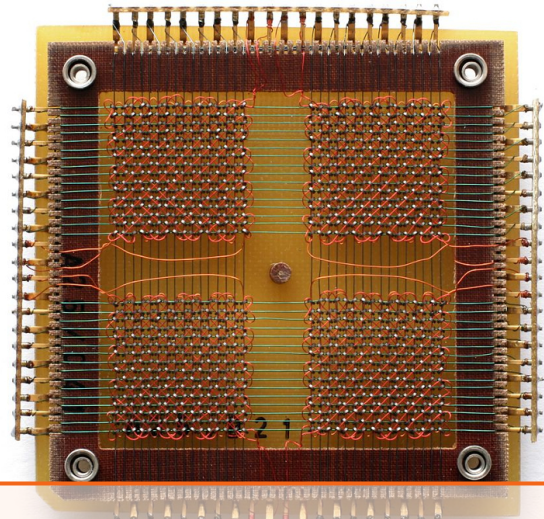
1.3 O que são compiladores

Compiladores.

1.4 Ambientes de desenvolvimento

Ambientes de desenvolvimento.

Origens da linguagem
Porque aprender C?
Onde C é usado?
Estrutura de um programa em C
Meu primeiro programa em C
Compilando meu programa
Executando
Entendendo meu programa



2. Visão Geral da linguagem C

blz

2.1 Origens da linguagem

blz

- oi

2.2 Porque aprender C?

blz

2.3 Onde C é usado?

blz

2.4 Estrutura de um programa em C

blz

2.5 Meu primeiro programa em C

blz

2.6 Compilando meu programa

blz

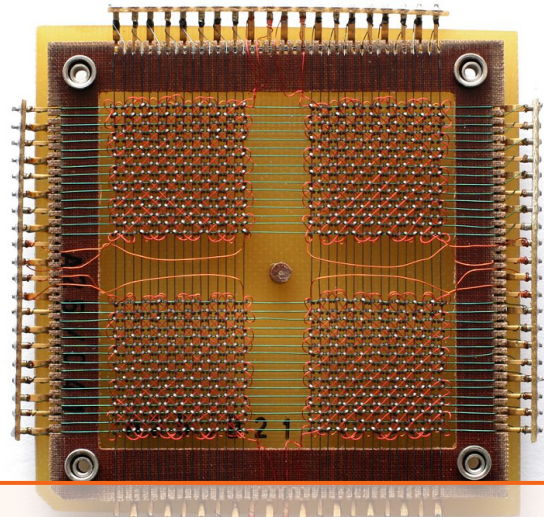
2.7 Executando

blz

2.8 Entendendo meu programa

blz

Tipo de dados básicos
Variáveis
Modificadores
Constantes
Expressões Aritméticas
 Operação de resto (%)
Conversão de tipo de dados



3. Variáveis, Tipos de Dados e Expressões Ar

Conceitos

- item

3.1 Tipo de dados básicos

3.2 Variáveis

3.3 Modificadores

3.4 Constantes

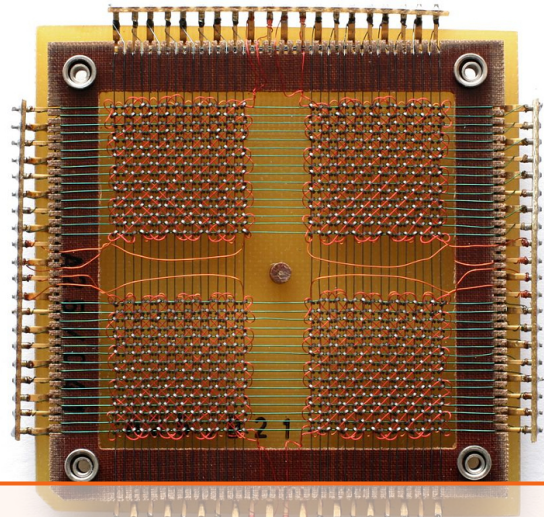
3.5 Expressões Aritméticas

3.5.1 Operação de resto (%)

3.6 Conversão de tipo de dados

Conversao.

Operadores relacionais
Operadores lógicos
O comando if
 O comando else
 O comando if-else-if
 ifs aninhados
O comando switch
Operador ternário(?)



4. Estruturas de Controle, Operadores Lógicos

Conceitos

- item

- 4.1 Operadores relacionais**
- 4.2 Operadores lógicos**
- 4.3 O comando if**
 - 4.3.1 O comando else**
 - 4.3.2 O comando if-else-if**
 - 4.3.3 ifs aninhados**
- 4.4 O comando switch**
- 4.5 Operador ternário(?)**

Conversao.

Comando for
Comando while
Comando do-while
Controle de loops
 ○ comando break
 ○ comando continue
Loops aninhados
Loops infinitos



5. Estruturas de repetição (Loops)

Conceitos

- item

5.1 Comando for

5.2 Comando while

5.3 Comando do-while

5.4 Controle de loops

5.4.1 O comando break

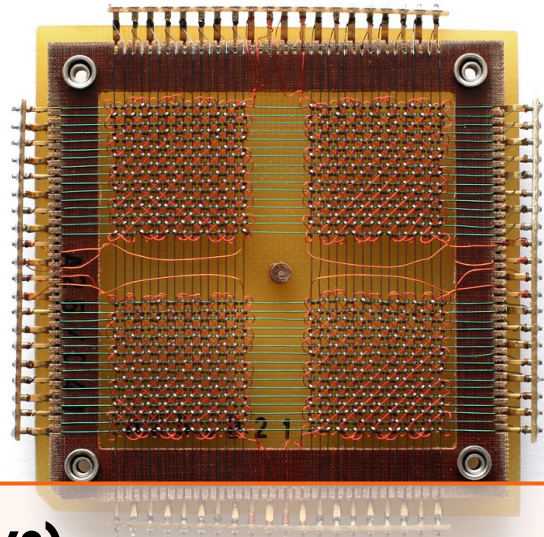
5.4.2 O comando continue

5.5 Loops aninhados

5.6 Loops infinitos

Conversao.

Trabalhando com vetores
Declarando vetores
Inicializando vetores
Acessando vetores
Vetor de caracteres
Aplicacoes com vetores
Imprimir os elementos
Soma dos elementos
Reverter o vetor
Ordenacao de vetor
Vetores Multidimensionais



6. Vetores (Arrays)

Conceitos

- item

6.1 Trabalhando com vetores

6.1.1 Declarando vetores

6.1.2 Inicializando vetores

6.1.3 Acessando vetores

6.2 Vetor de caracteres

6.3 Aplicacoes com vetores

6.3.1 Imprimir os elementos

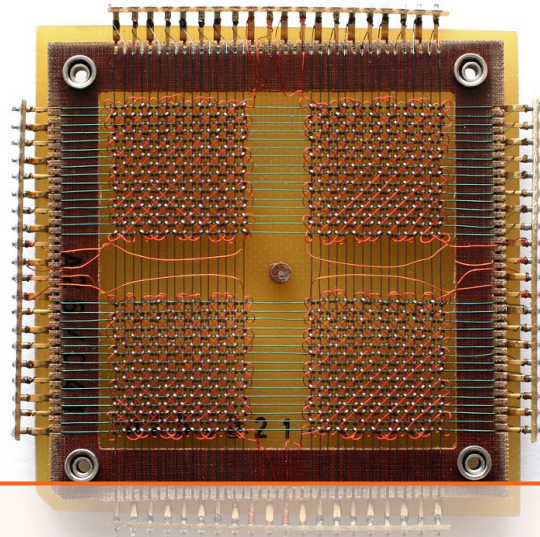
6.3.2 Soma dos elementos

6.3.3 Reverter o vetor

6.3.4 Ordenacao de vetor

6.4 Vetores Multidimensionais

Conversao.



7. Ponteiros

7.1 Armazenamento Primário

Em um computador atual (2014), existem três tipos principais de armazenamento primário: **registradores do processador**; **cache do processador**, e; **memória principal**.

Registradores são pequenos locais de armazenamento, com tamanho estático, contidos no processador. Por fazerem parte do *ISA (Instruction Set Architecture)*, variam com a arquitetura (x86, x86_64, MIPS etc). Um exemplo de registradores de uso geral da arquitetura x86_64 é: *rax, rbx, rcx, rdx*.

A cache do processador é um contêiner de dados intermediário e de acesso aleatório com maior capacidade que os registradores. Ela se situa entre a memória principal e o próprio processador com o intuito de diminuir o tempo médio de acesso às informações; podendo ser subdividida em cache de instruções (lida apenas com a leitura da memória), cache de dados (trata da leitura e escrita da memória) e *TLB - Translation lookaside buffer* (com a finalidade de agilizar a tradução da memória virtual x física).

O termo “Memória Principal” é comumente usado como referência à memória RAM (*Random Access Memory*), uma vez que nesta reside a grande porção de dados utilizados antes e/ou após o processamento. Essa referência se dá, também, pela transparência que os registradores e a cache do processador tem em comparação à memória principal, tendo em vista a utilização direta dos dados contidos nela quando no desenvolvimento de software.

Há alguns detalhes importantes a serem ressaltados a respeito desses três repositórios primários de dados.

1. A memória principal (RAM), distintamente dos registradores e da cache, não se encontra no processador mas sim conectada a ele através do barramento de memória que, por sua vez, subdivide-se em dois: barramento de endereço e barramento de dados [Figura 7.1].
2. O gerenciamento da memória cache, mesmo que possível através de instruções como *INVD* e *WBINVD* na arquitetura x86, é e deve, por segurança, ser deixado ao encargo do próprio processador, salvo em casos realmente justificáveis.
3. Os dados de todos eles são obtidos por endereçamento, ou seja, por indexação, mas o que os torna diferentes, à visão do programador, é o fato de os registradores e a cache

só serem acessados dessa forma pelo microcódigo da arquitetura, enquanto a memória principal pode o ser por endereçamento via código de máquina (programa residente na própria memória).

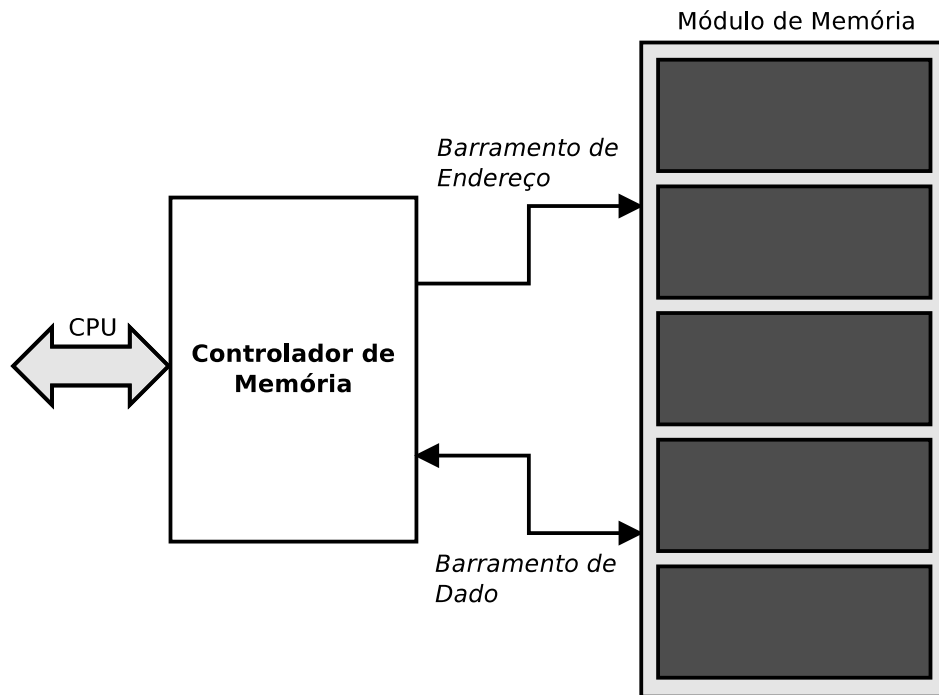


Figura 7.1: Barramentos de Endereço e de Dados

7.1.1 Memória Principal

Sobre a memória principal.

7.2 Usando Ponteiros

Como usar ponteiros.

7.3 Usando Vetores

Como usar vetores.

7.4 Vetores NÃO são Ponteiros

Vetores não são ponteiros.