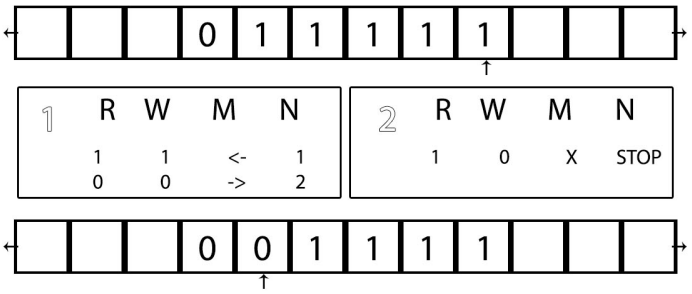
P1 ICC-1

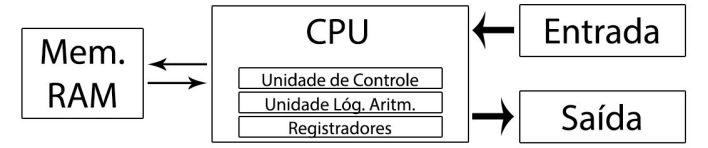
fakhoury

Máquina de Turing

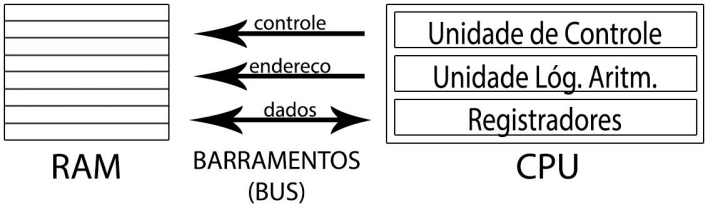


Fita infinita, obrigatóriamente lê e escreve alguma coisa

Arquitetura de Von Neumman

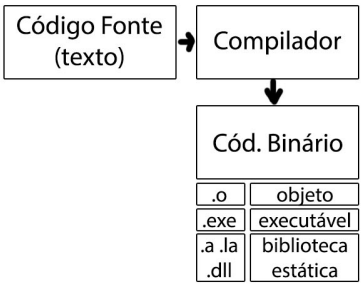


Não é uma fita - endereços



Bits por variável

char	8
int	32
float	32
double	64
long int	64
char*	64



Endereço	Memória Stack	
0x000DB	00111111	1.5 float (4 bytes)
0x000DA	11000000	
0x000D9	00000000	
0x000D8	00000000	
0x000D7	00000000	5 int (4 bytes)
0x000D6	00000000	
0x000D5	00000000	
0x000D4	00000101	
0x000D3	01000101	'E' Char (1 byte)
...	...	
0x000A3	Código	

char e char* - diferenças

char a;
char* p;

a - char
&a - endereço
p - endereço
*p - char

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
  int x = 258;
  int* end_x = &x;
  printf("endereco armazenado: %p\n", end_x);
  printf("valor nesse endereco: %d\n", *end_x);

  char* p;
  p = (char*) &x; // ou p = (char*) end_x;
  printf("endereco armazenado (em p): %p\n", p);
  printf("valor 1B nesse endereco: %d\n", *p);
  p = p + 1; // calcula o proximo endereco
  printf("endereco atual (em p): %p\n", p);
  printf("valor 1B nesse endereco: %d\n", *p);
  p = p + 1; // calcula o proximo endereco
  printf("endereco atual (em p): %p\n", p);
  printf("valor 1B nesse endereco: %d\n", *p);
  p = p + 1; // calcula o proximo endereco
  printf("endereco atual (em p): %p\n", p);
  printf("valor 1B nesse endereco: %d\n", *p);

  return 0;
}
```

RESULTADO:

```
endereco armazenado: 0x7fff5fbe9b4c
valor nesse endereco: 258
endereco armazenado (em p): 0x7fff5fbe9b4c
valor 1B nesse endereco: 2
endereco atual (em p): 0x7fff5fbe9b4d
valor 1B nesse endereco: 1
endereco atual (em p): 0x7fff5fbe9b4e
valor 1B nesse endereco: 0
endereco atual (em p): 0x7fff5fbe9b4f
valor 1B nesse endereco: 0
```

```
////////
```

```c
main() {
  int y = 254; //bits finais: 11111110
  printf("bit 1: %d\n", y & 1);
  printf("bit 2: %d\n", (y >> 1) & 1);
  printf("bit 3: %d\n", (y >> 2) & 1);
  printf("bit 4: %d\n", (y >> 3) & 1);
  printf("bit 5: %d\n", (y >> 4) & 1);
  printf("bit 6: %d\n", (y >> 5) & 1);
  printf("bit 7: %d\n", (y >> 6) & 1);
  printf("bit 8: %d\n", (y >> 7) & 1);
}
```

RESULTADO:

```
bit 1: 0
bit 2: 1
bit 3: 1
bit 4: 1
bit 5: 1
bit 6: 1
bit 7: 1
bit 8: 1
```

```c
if (condicao) {
  //faz alguma coisa
} else {
  faz outra coisa
}

while (condicao) {
  //faz alguma coisa ate
a condicao ser falsa
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void imprime_vetor(int* vet, int n) {
	int p;
	for (p=0; p < n; p++) {
		printf("[%d] %d (%p)\n", p, *(vet+p),
vet+p);
	}
	printf("\n");
}

void imprime_matriz(int* mat, int x, int y) {
	int i, j;
	for (i = 0; i < x; i++) {
		for (j = 0; j < y; j++) {
			printf("[%d,%d] %d (%p)\n", i, j,
*(mat + i * y + j), (mat + i * y + j));
		}
	}
}

void troca_var(int* a) { *a = 5; }

int main (int argc, char* argv[]) {
	int vet[4] = {1, 2, 3, 4};
	int mat[3][2] = { {10, 11}, {12, 13}, {14, 15}};

	troca_var(vet+3);

	printf("\n--Vetor--\n");
	imprime_vetor(vet, 4);

	printf("\n--Matriz--\n");
	imprime_matriz(mat, 3, 2);

	printf("\n--Outra Matriz--\n");
	int i, j;
	for (i = 0; i < 3; i++) {
		for (j = 0; j < 2; j++) {
			printf("[%d,%d] %d (%p)\n", i, j,
*(*(mat+i)+j), *(mat+i)+j);
		}
	}

	printf("\nOutras formas do vetor:\n");
	printf("1. %d\n", vet[1]);
	printf("2. %d\n", *(vet+1));
	printf("3. %d\n", *vet+1);
	printf("4. %p\n", vet);
	printf("5. %p\n", vet+1);

	printf("\nOutras formas da matriz:\n");
	printf("1. %d\n", mat[0][1]);
	printf("2. %d\n", *(*(mat+0)+1));
	printf("3. %p\n", *mat);
	printf("4. %p\n", *(mat+1));
	printf("5. %p\n", mat);
	printf("6. %p\n", mat+1);

	switch(vet[0]) { //so pra mostrar o switch
		case 1: printf("Eh 1\n");
				break;
		case 2: printf("Eh 2\n");
				break;
		default: printf("Sei la\n");
				 break;
	}

	printf("\nINT e INT*\n");
	int a = 10;
	int* ea = &a;
	int** eea = &ea;
	printf("1. %d\n", a);
	printf("2. %d\n", *ea);
	printf("3. %d\n", **eea);
	printf("4. %p\n", ea);
	printf("5. %p\n", *eea);
	printf("6. %p\n", eea);
	printf("7. %p\n", &a);
	printf("8. %p\n", &ea);
	printf("9. %p\n", &eea);

	int y = 20;
	printf("\n8 primeiros bits de %d\n", y);
	printf("bit 1: %d\n", (y >> 0) & 1);
	printf("bit 2: %d\n", (y >> 1) & 1);
	printf("bit 3: %d\n", (y >> 2) & 1);
	printf("bit 4: %d\n", (y >> 3) & 1);
	printf("bit 5: %d\n", (y >> 4) & 1);
	printf("bit 6: %d\n", (y >> 5) & 1);
	printf("bit 7: %d\n", (y >> 6) & 1);
	printf("bit 8: %d\n", (y >> 7) & 1);

	//random
	srand(time(NULL));
	int aleatorio = rand()%2+1; //[1, 2]

	if (argc == 3) {
		int inteiro = atoi(argv[1]);
		float ponto_flutuante = atof(argv[2]);
	}

	return 0;
}
```

OUTPUT

```
--Vetor--
[0] 1 (000000000062FE30)
[1] 2 (000000000062FE34)
[2] 3 (000000000062FE38)
[3] 5 (000000000062FE3C)

--Matriz--
[0,0] 10 (000000000062FE10)
[0,1] 11 (000000000062FE14)
[1,0] 12 (000000000062FE18)
[1,1] 13 (000000000062FE1C)
[2,0] 14 (000000000062FE20)
[2,1] 15 (000000000062FE24)

--Outra Matriz--
[0,0] 10 (000000000062FE10)
[0,1] 11 (000000000062FE14)
[1,0] 12 (000000000062FE18)
[1,1] 13 (000000000062FE1C)
[2,0] 14 (000000000062FE20)
[2,1] 15 (000000000062FE24)

Outras formas do vetor:
1. 2
2. 2
3. 2
4. 000000000062FE30
5. 000000000062FE34

Outras formas da matriz:
1. 11
2. 11
3. 000000000062FE10
4. 000000000062FE18
5. 000000000062FE10
6. 000000000062FE18

Eh 1

INT e INT*
1. 10
2. 10
3. 10
4. 000000000062FE0C
5. 000000000062FE0C
6. 000000000062FE00
7. 000000000062FE0C
8. 000000000062FE00
9. 000000000062FDF8

8 primeiros bits de 20
bit 1: 0
bit 2: 0
bit 3: 1
bit 4: 0
bit 5: 1
bit 6: 0
bit 7: 0
bit 8: 0
```