Arquitetura de Computadores

PROF. ISAAC

Array

Array

- Arranjo (array).
- Conjunto finito de elementos homogêneos.
  - É possível identificar o primeiro, o segundo, ..., o nésimo elemento do arranjo.
  - Homogeneidade: todos os elementos do arranjo são do mesmo tipo.
- Ocupa um grupo de posições de memória adjacentes e identificadas pelo mesmo nome.

Array

- Arranjo de UMA dimensão, no qual as posições de memória são logicamente ocupadas em ordem sequencial crescente.
- Exemplo: vetor de inteiros

Valores	**2**	**2**	**1**	**2**	**2**	**0**	**8**	**4**	**7**	**3**	**5**
Endereço	20h	21h	22h	23h	24h	25h	26h	27h	28h	29h	2Ah

Exemplo de Array

- Vamos armazenar 10 valores na RAM em endereços sequenciais.
- Mostrar esses valores na mesma ordem usando P1 como saída.
- Implementar de forma que a quantidade de valores possa ser alterada.

Exemplo de Array

Em uma estrutura de array devemos definer o endereço inicial e o tamanho do array.

```
MOV 02H, #10        ; quantidade de escritas
MOV 03H, #020H      ; endereço inicial
```

Exemplo de Array

Então o programa principal informa o endereço do array, tamanho e chama as sub-rotinas de escrita e leitura.

```
Principal:
    MOV 20h, #020h      ; endereço inicial
    MOV 21h, #10        ; quantidade de escritas
    ACALL WRITE         ; chama a sub-rotina de escrita
    ACALL READ          ; chama a sub-rotina de read
    MOV P1, #0FFh       ; apaga os leds no edsim51
    SJMP $              ; fica preso nessa linha
```

Exemplo de Array - escrita

Agora criaremos uma sub-rotina que **escreve** no array.

```
WRITE:
        MOV A, #0FFH        ; valor para escrever no array
        MOV R0, 20H         ; R0 possui o inicio do array
        MOV R1, 21H         ; qtd que falta escrever
    ROT:
        MOV @R0,A           ; escreve A no endereço @R0 do vetor
        DEC A               ; novo valor de A
        INC R0              ; próximo endereço de escrita
        DJNZ R1, ROT        ; atualiza quantidade que falta
        RET
```

Exemplo de Array - leitura

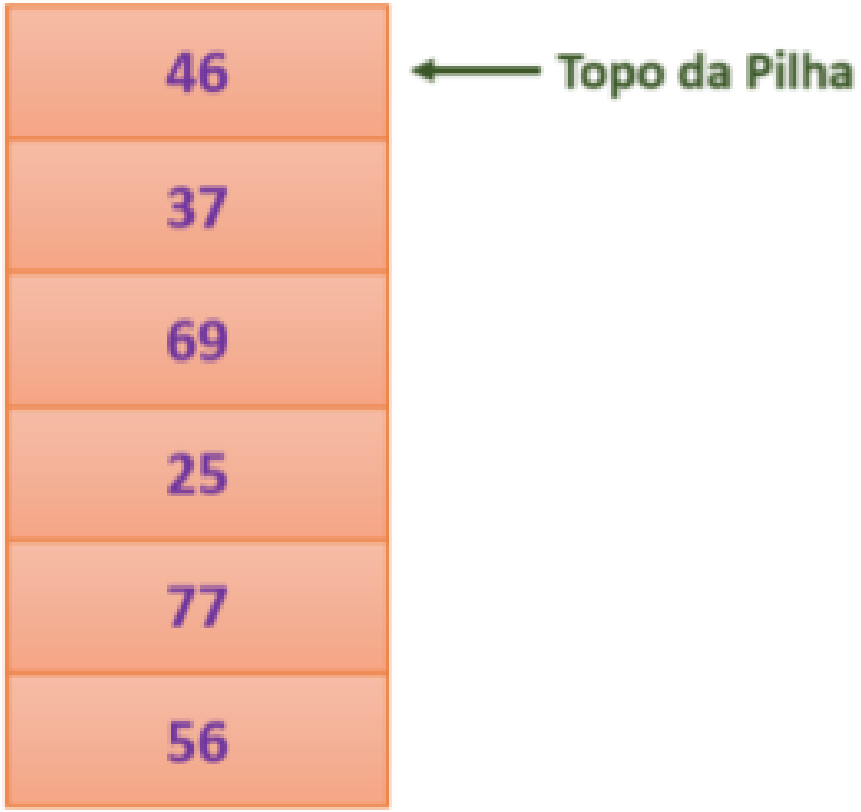
Agora criaremos uma sub-rotina que **lê** o array.

```
READ:
        MOV R0, 20h         ; endereço inicial
        MOV R1, 21h         ; qtd de leituras (tamanho do vetor)
ROT2:
        MOV A, @R0          ; realizando a leitura por endereçamento indireto
        MOV P1, A           ; escreve na porta P1
        INC R0                ; próximo endereço de leitura
        DJNZ R1, ROT2         ; atualiza quantidade que falta
        RET                   ; Retorna da sub-rotina
```

Estrutura de Pilha

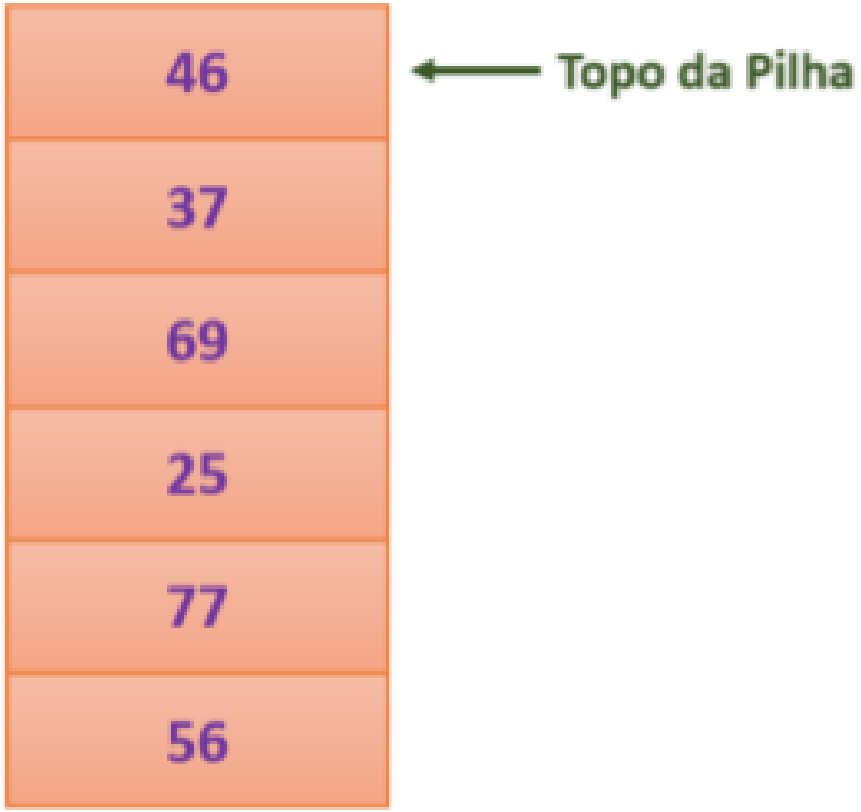
Pilha

- O elemento removido é o mais recente;
- Estrutura do tipo LIFO (last in first out).

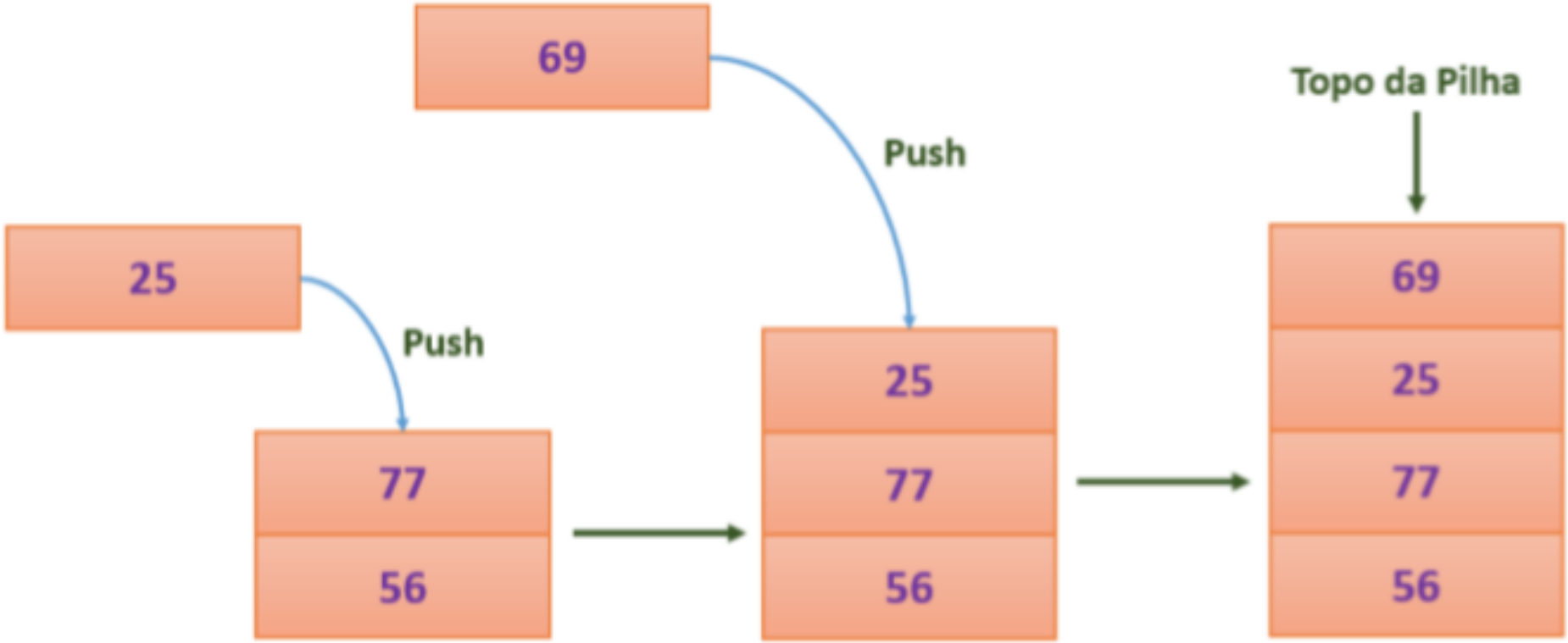


Operação Push

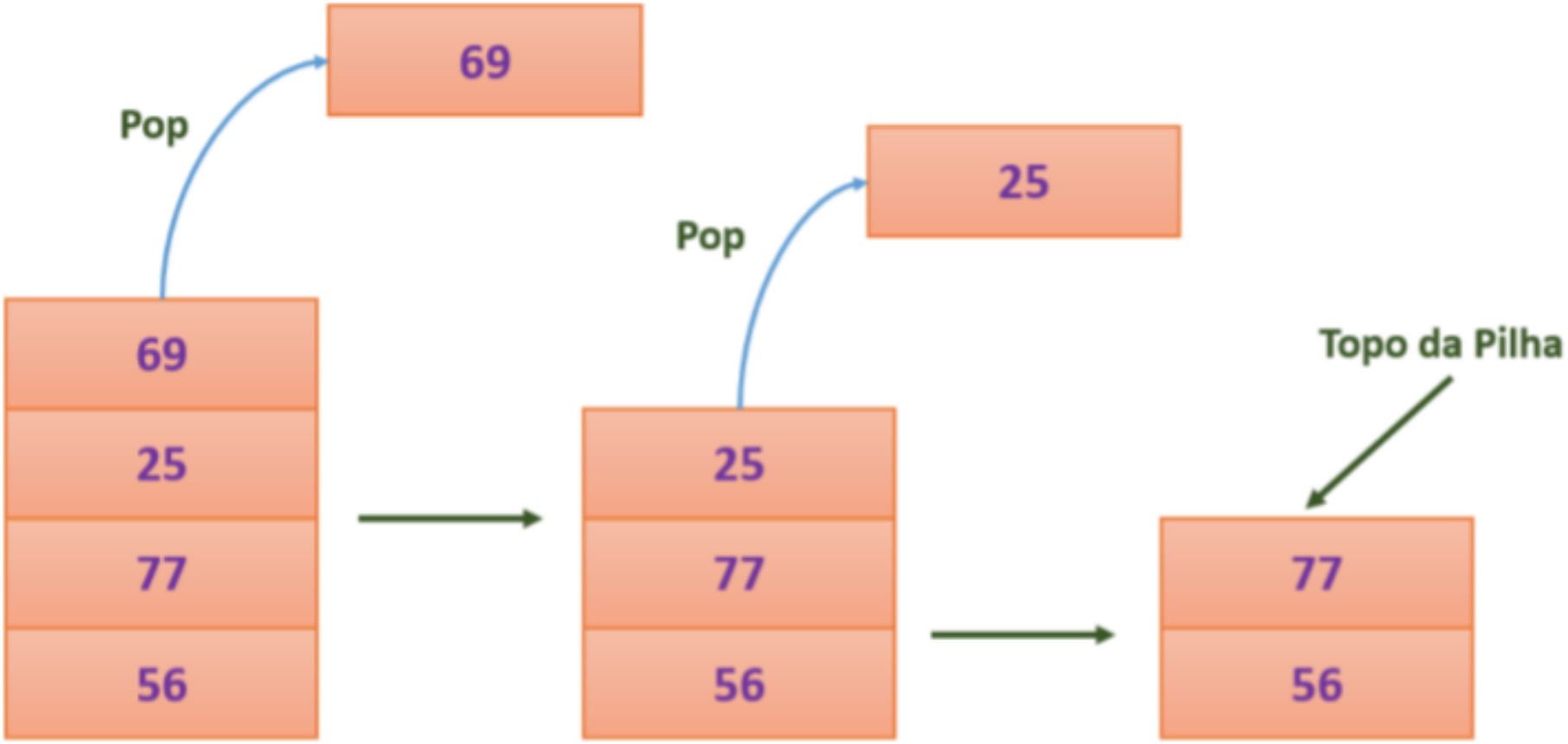
- As operações acontecem sempre na mesma extremidade da estrutura.



Operação Push



Operação Pop



Program Counter (PC)

- Todos os microprocessadores e microcontroladores possuem um PC;
- Armazena o endereço da próxima instrução que deve ser executada;
- No 8051, inicializa em 0000H;
- Incrementa conforme a execução do programa;

Stack Pointer (SP)

- Uma pilha (stack) endereçada por 8 bits;

- Utilizada geralmente para armazenar informações temporárias, como endereços em chamadas (jumps).

Instruções do MSC-51

Tipo	Quantidade
Aritméticas	24
Lógicas	25
Transferência (cópia) de Dados	28
Booleanas	17
Saltos	17

Instruções aritméticas: envolvem operações do tipo soma, subtração, multiplicação, divisão, incremento e decremento.

Instruções lógicas: fazem operações bit a bit com registradores e também rotações.

Instruções do MSC-51

Instruções de transferência (cópia) de dados: copiam bytes entre os diversos registradores e a RAM interna.

Instruções booleanas: essas instruções são denominadas booleanas porque trabalham com variáveis lógicas (variável de 1 bit). Como o próprio nome sugere, elas são talhadas para resolver expressões booleanas.

Instruções de desvio: desviam o fluxo de execução do programa, chamam subrotinas, fazem desvios condicionais e executam laços de repetição.

Instruções do 8051

Lógica	Aritmética	Memória	Outros
ANL ✓	ADD ✓	MOV ✓	NOP ✓
ORL ✓	ADDC ✓	MOVC ✓	RET e RETI ✓
XRL ✓	SUBB ✓	MOVX ✓	ACALL e LCALL ✓
CLR ✓	MUL ✓	PUSH	JMP ✓
CPL ✓	DIV ✓	POP	AJMP ✓
RL ✓	INC ✓	XCH ✓	LJMP ✓
RLC ✓	DEC ✓	XCHD ✓	SJMP ✓
RR ✓	DA		JB e JNB ✓
RRC ✓			JZ e JNZ ✓
SWAP ✓			JC e JNC ✓
SETB ✓			JBC ✓
			DJNZ ✓
			CJNE ✓

Instrução - PUSH

Operação: PUSH

Função: Insere valor na pilha

Sintaxe: PUSH *endereço_iram*

Descrição : PUSH armazena o valor especificado em *endereço* na pilha. Primeiramente, o valor do SP é incrementado de 1 e então o valor é armazenado.

Exemplo:

- PUSH Acc

Instrução - POP

Operação: POP

Função: Retira um valor da pilha

Sintaxe: POP *endereço_iram*

Descrição : POP retira o ultimo valor armazenado na pilha e coloca o mesmo em *endereço_iram*. O SP é então decrementado de 1.

Exemplo:

- POP 34h

Instruções PUSH e POP

		bytes	MC	Op1	Op2
PUSH	end8	2	2	C0	end8
POP	end8	2	2	D0	end8

Instruções para operar com a pilha.

Instruções do 8051

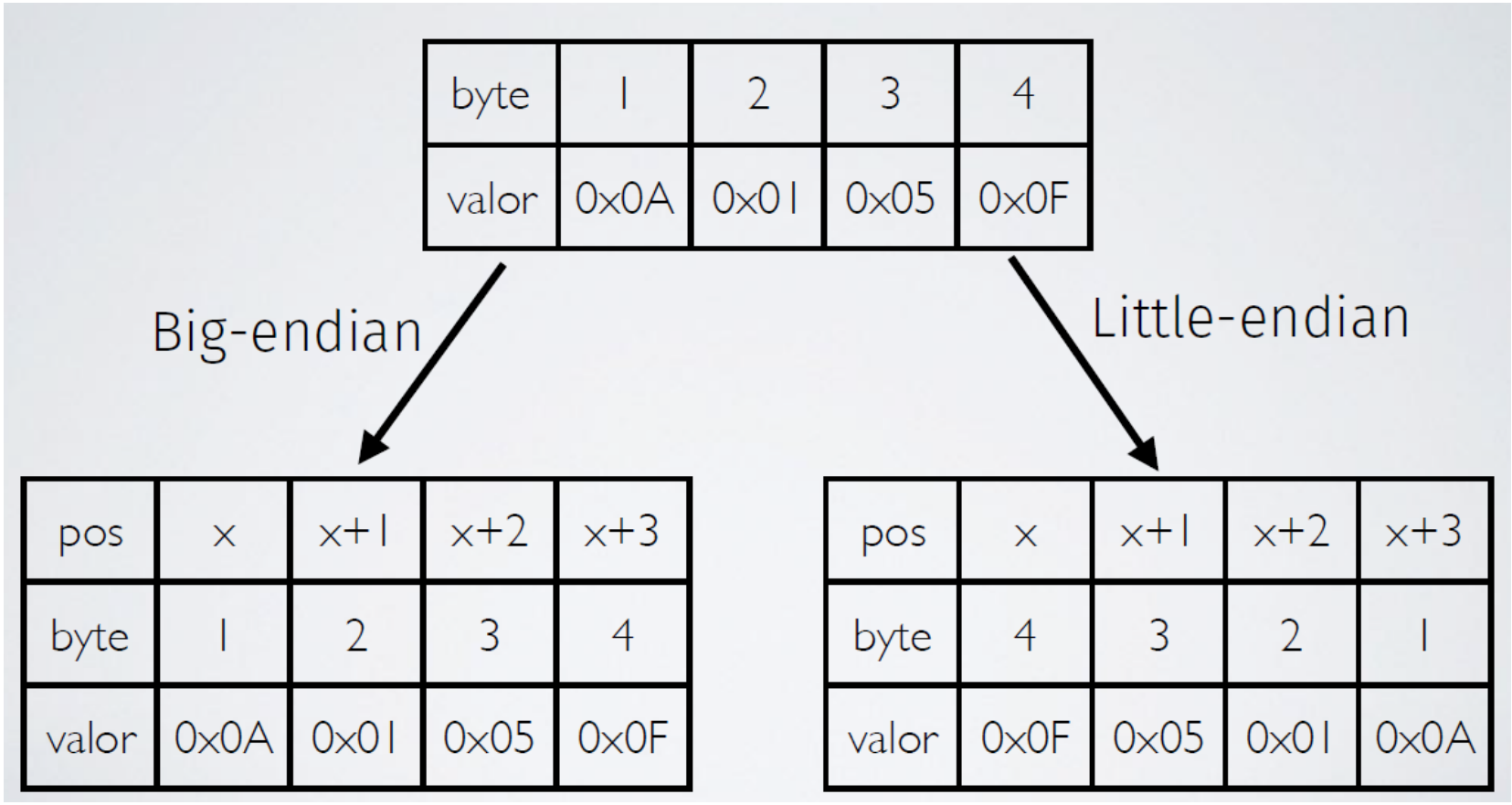
Lógica	Aritmética	Memória	Outros
ANL ✓	ADD ✓	MOV ✓	NOP ✓
ORL ✓	ADDC ✓	MOVC ✓	RET e RETI ✓
XRL ✓	SUBB ✓	MOVX ✓	ACALL e LCALL ✓
CLR ✓	MUL ✓	PUSH ✓	JMP ✓
CPL ✓	DIV ✓	POP ✓	AJMP ✓
RL ✓	INC ✓	XCH ✓	LJMP ✓
RLC ✓	DEC ✓	XCHD ✓	SJMP ✓
RR ✓	DA ✓		JB e JNB ✓
RRC ✓			JZ e JNZ ✓
SWAP ✓			JC e JNC ✓
SETB ✓			JBC ✓
			DJNZ ✓
			CJNE ✓

Armazenamento dos dados

Armazenamento dos dados

- Grande parte dos dados são constituídos de mais do que 1 byte;
- É necessário armazenar todos os bytes de forma sequencial na memória;

Armazenamento dos dados



Armazenamento dos dados

- Os bytes são guardados por ordem crescente do seu "peso numérico" em endereços sucessivos da memória (extremidade menor primeiro ou little-endian).
- Os bytes são guardados por ordem decrescente do seu "peso numérico" em endereços sucessivos da memória (extremidade maior primeiro ou big-endian).

Armazenamento dos dados

Atualmente:

- Redes: big-endian
- Processadores: little-endian
- Sistema de arquivo: depende…

Exemplo em C

```c
#include <stdio.h>

int main() {
    int valor = 0x12345678; // Valor em hexadecimal
    unsigned char *ponteiro; // Ponteiro para unsigned char (1 byte)

    ponteiro = (unsigned char *)&valor;// Aponta para o primeiro byte da variável 'valor'

    printf("Valor da variável: 0x%x\n", valor);

    // Como um int geralmente tem 4 bytes (dependendo do sistema e compilador),
    // nós iteramos por cada byte e imprimimos seu valor e endereço.
    for (int i = 0; i < sizeof(int); i++) {
        printf("Endereço do byte %d: %p, valor: 0x%x\n", i, (ponteiro + i), *(ponteiro + i));
    }

    return 0;
}
```

Armazenamento dos dados

Isso é Big-endian ou Little-endian?

```
> make -s
> ./main
Valor da variável: 0x12345678
Endereço do byte 0: 0x7ffc423f0794, valor: 0x78
Endereço do byte 1: 0x7ffc423f0795, valor: 0x56
Endereço do byte 2: 0x7ffc423f0796, valor: 0x34
Endereço do byte 3: 0x7ffc423f0797, valor: 0x12
>
```

Armazenamento dos dados

```
> make -s
> ./main
Valor da variável: 0x12345678
Endereço do byte 0: 0x7ffc423f0794, valor: 0x78
Endereço do byte 1: 0x7ffc423f0795, valor: 0x56
Endereço do byte 2: 0x7ffc423f0796, valor: 0x34
Endereço do byte 3: 0x7ffc423f0797, valor: 0x12
>
```

Endereço	Valor
184	12
185	34
186	56
187	78

Big-endian

Endereço	Valor
184	78
185	56
186	34
187	12

Little-endian

Exemplos

Exemplo 1

Teste e verifique o que faz esse programa.

```
        MOV A, #0Fh
ROT:
        PUSH Acc
        DEC A
        JNZ ROT
```

Exemplo 2

Teste e verifique o que faz esse programa.

```
        MOV A, #0Fh
        MOV 030h, #0FFh
ROT:
        PUSH 030h
        DEC A
        JNZ ROT
```

Exemplo 3

Teste e verifique o que faz esse programa.

```
        MOV  A, #0Fh
ROT:
        PUSH Acc
        DEC   A
        JNZ   ROT

        MOV   R2, #0Fh
        MOV   R0, #60h
ROT2:
        POP    Acc
        MOV   @R0, A
        INC    R0
        DJNZ  R2, ROT2
```

Exemplo 4

Teste e verifique o que faz esse programa.

```
        MOV  A, #0Fh
        MOV  50h, #0FFh
        MOV  SP, #2Fh
ROT:
        PUSH  50h
        DEC    A
        JNZ    ROT
        PUSH 50h
```

Exemplo 5

Teste e verifique o que faz esse programa.

```
        MOV  R2, #0Fh
        MOV  50h, #0FFh
        MOV  SP, #2Fh
ROT:
        PUSH 50h
        DJNZ  R2, ROT
        PUSH 50h
```

```
        MOV  R2, #0Fh
        MOV  R0, #70h
ROT2:
        POP   Acc
        MOV @R0, A
        INC    R0
        DJNZ  R2, ROT2
        POP   Acc
        MOV @R0, A
```

Uso da Pilha para acesso e retorno das Funções

Teste e verifique o que faz esse programa, observe o valor do SP e o que está sendo armazenado na Pilha.

```
Principal:
        MOV   A, #01BH
        ACALL FUNC01
        NOP

org 0150h

FUNC01:
        MOV   30H, #0FFH
        ACALL FUNC02
        NOP
        RET
```

```
org 0270h

FUNC02:
        MOV   40H, #0AAH
        ACALL FUNC03
        NOP
        RET

org 03B0h

FUNC03:
        MOV  40H, #11H
        RET
```

Exercícios

Exercício 1.

Carregue na memória números inteiro positivo como um vetor.

Exemplo: [10, 32, 04, 01, 76, 20]

Endereço : **valor**

20h : 10
21h : 32
22h : 04
23h : 01
24h : 76
25h : 20

E programe uma sub-rotina que encontre um valor desejado no vetor e sinalize na porta P1 que o valor foi encontrado.

Bibliografia

ZELENOVSKY, R.; MENDONÇA, A. Microcontroladores Programação e Projeto com a Família 8051. MZ Editora, RJ, 2005.

Gimenez, Salvador P. Microcontroladores 8051 - Teoria e Prática, Editora Érica, 2010.