# Pilha e Subrotinas

## Pilha

A pilha é uma estrutura lógica (LIFO – *last in first out* – o último a entrar é o primeiro a sair) que fornece regras bem claras de entrada e saída de dados da memória. Um acesso comum à memória, necessita que se coloque um endereço, já um acesso à pilha se faz sem a necessidade de colocar um endereço (porém este endereço deve ser informado inicialmente antes de começar a executar as operações na pilha).

A pilha é uma estrutura tão importante nos sistemas microprocessados que existe um registrador no 8085 especialmente criado capaz de trabalhar com a pilha. Este registrador é o SP (*stack pointer* – ponteiro de pilha) que tem 16 bits e armazena o endereço do topo da pilha e deve ser inicializado antes de fazer as operações de pilha.

## Operações da pilha

As duas operações de acesso à pilha são tradicionalmente chamadas de PUSH (empilhar) e POP (desempilhar) e assim também são chamadas no 8085. Cada operação trabalha com 2 bytes de uma só vez e trabalham com os pares de registradores: B/C, D/E, H/L e A/PSW. A pilha cresce no sentido decrescente dos endereços. Isto é interessante para restringir um programa dentro de uma área de memória com o código no início, a pilha no fim e no meio o espaço para as variáveis.

Exemplo de funcionamento da instrução PUSH B (empilha os registradores B e C):

**SP = SP -1**

**Armazena B no endereço de memória dado por SP**

**SP = SP -1**

**Armazena C no endereço de memóriadado por SP**

Exemplo de funcionamento da instrução POP B (desempilha nos registradores B e C):

**Carrega C com o conteúdo do endereço dado por SP**

**SP = SP + 1**

**Carrega B com o conteúdo do endereço dado por SP**

**SP =SP + 1**

Temos as duas instruções completamente complementares no seu funcionamento.

Veja no arquivo notas de aula do 8085 um exemplo de funcionamento da pilha (página 51). Os exemplos a seguir serão explicados em vídeo com o simulador sim8085 para que fique bem claro o funcionamento da pilha.

Exemplo 1

```
LXI SP, 08D0H
MVI B, BBH
MVI C, CCH
MVI D, DDH
MVI E, EEH
PUSH B
PUSH D
LXI B, 0000H
LXI D, 0000H
POP D
POP B
HLT
```

Neste exemplo a instrução LXI indica uma cópia imediata como o MVI, porém de 2 bytes, já que o SP tem 16 bits. Esta instrução inicializa o valor do SP para que a pilha não fique em local aleatório. Da mesma forma o LXI B corresponde a MVI B e MVI C. Economizamos assim um byte no programa pois duas instruções MVI equivalem a 4 bytes e um LXI é uma instrução de 3 bytes. O LXI B e LXI D servem para zerar os registradores  para quando for fazer o POP ficar claro que os valores dos registradores foram modificados. O mais interessante deste exemplo é ver o funcionamento junto com os registradores e a memória

Exemplo 2

```
LXI SP, 08D0H
LXI B, BBCCH
LXI D, DDEEH
PUSH B
PUSH D
POP B
POP D
```

Neste exemplo usamos a pilha para trocar os valores dos registradores B com D e C com E

# Subrotinas

São trechos de código separados do programa principal que podem ser executados a partir de qualquer chamada a elas. Em C as subrotinas são as funções, bastando chamar a função por seu nome e o código daquela função será executado. Uma função das subrotinas é a organização de programas em trechos de código homogêneos, desta forma pode-se separar partes do programa que fazem uma determinada tarefa para dar melhor legibilidade. Outra função é de poder ser chamada quantas vezes for necessário, o que reduz a quantidade de código e consequente uso de memória do programa quando temos trechos de código que podem ser executados várias vezes no programa.

Em assembly do 8085 temos duas instruções para trabalhar com as subrotinas: CALL e RET

CALL endereço

Esta instrução desvia a execução para o endereço indicado e salva o endereço da próxima instrução sequencial na pilha. Este endereço armazenado volta a ser carregado no PC quando a subrotina acaba e o comando RET é executado. Note que podemos usar rótulos como usamos com qualquer instrução de salto.

O exemplo abaixo mostra como funciona chamadas a subrotinas. Veja o vídeo deste exemplo para acompanhar os detalhes no simulador do 8085.

```
            LXI SP, 08D0H
            MVI A, 0
            MVI B, 3
            MVI C, 4
            CALL mult
            HLT
mult:       ADD B
            DCR C
            JNZ mult
            RET
```

Este programa faz a multiplicação de B por C e coloca o resultado em A (exemplo já foi dado anteriormente) porém usando uma subrotina. Como as instruções CALL e RET usam a pilha, SP foi inicializado no início do programa (basta que seja antes da chamada a CALL ou RET). Vejam que o rótulo mult serve tanto como entrada para a subrotina como para o uso do loop de multiplicação.