Atividade 8

O objetivo desta atividade é permitir que o aluno seja capaz de criar um programa em linguagem assembly que utilize procedimentos e manipule a pilha.

Parte 1 – Procedimentos

A declaração de procedimentos em linguagem assembly é feita utilizando a diretiva **PROC** para indicar o início do procedimento e a diretiva **ENDP** para indicar o fim do procedimento:

```
NOME PROC
  ... Código do procedimento ...
NOME ENDP
```

Onde:

NOME: é o nome do procedimento.

A chamada de procedimentos é feita utilizando a instrução **CALL** seguida do nome do procedimento. Assim, para chamar o procedimento NOME mostrado anteriormente deve-se executar a instrução:

```
CALL NOME
```

O retorno de um procedimento é feito utilizando a instrução **RET**. Assim, no procedimento NOME mostrado anteriormente deve-se adicionar a instrução RET no ponto (ou nos pontos) onde deseja-se retornar da chamada do procedimento:

```
NOME PROC
  ... Código do procedimento ...
  RET
NOME ENDP
```

Quando vários procedimentos são definidos em um programa, é necessário especificar qual procedimento será o primeiro a ser chamado quando o programa for executado. Isso é feito colocando-se o nome do procedimento em frente à diretiva END no final do programa. No exemplo mostrado a seguir, são definidos três procedimentos P1 e MAIN, sendo que o procedimento MAIN será chamado quando o programa for executado:

```
TITLE Procedimentos
.MODEL SMALL
.STACK 100h
.CODE
P1 PROC
    ... Código do procedimento P1 ...
    RET
P1 ENDP
MAIN PROC
    ... Código do procedimento MAIN ...
    CALL P1
    ...
    MOV AH,4Ch
    INT 21h
MAIN ENDP
END MAIN
```

A declaração de procedimentos é extremamente útil quando desejamos modularizar um programa, permitindo a reutilização de um trecho de código sem que ele tenha que ser replicado ao longo do programa várias vezes.

Suponha, por exemplo, que em um determinado programa temos a necessidade de pular linha várias vezes. Podemos então declarar um procedimento chamado PLINHA e chamar esse procedimento toda vez que desejarmos pular linha no programa.

Ex:

```
TITLE PulaLinha
.MODEL SMALL
.STACK 100h
.DATA
   MSG1  DB  “Primeira linha.$”
   MSG2  DB  “Segunda linha.$”
   MSG3  DB  “Terceira linha.$”
.CODE

PLINHA PROC
   MOV AH,2
   MOV DL,10
   INT 21h
   MOV DL,13
   INT 21h

   RET
PLINHA ENDP

MAIN PROC
   MOV AX,@DATA
   MOV DS,AX

   MOV AH,9
   LEA DX,MSG1
   INT 21h

   CALL PLINHA

   MOV AH,9
   LEA DX,MSG2
   INT 21h

   CALL PLINHA

   MOV AH,9
   LEA DX,MSG3
   INT 21h

   CALL PLINHA

   MOV AH,4Ch
   INT 21h
MAIN ENDP
END MAIN
```

Parte 2 – Pilha

A pilha é uma região de memória que pode ser utilizada para armazenar informações no formato LIFO (Last In First Out), ou seja, o último elemento armazenado na pilha é sempre o primeiro a ser removido.

Para inserir um elemento na pilha devemos utilizar a instrução PUSH, como mostrado a seguir:

```
PUSH FONTE
```

Onde:

FONTE: deve ser um registrador ou uma variável de **16 bits** que contém o valor que será armazenado na pilha.

Para remover um elemento da pilha devemos utilizar a instrução POP, como mostrado a seguir:

```
POP DESTINO
```

Onde:

DESTINO: deve ser um registrador ou uma variável de **16 bits** onde será armazenado o valor removido da pilha.

Suponha, por exemplo, que em um determinado programa temos a necessidade de ler três caracteres e exibi-los em ordem invertida. Podemos então utilizar a pilha para armazenar os caracteres lidos e em seguida remove-los em ordem invertida para a exibição, como no exemplo mostrado a seguir.

Ex:

```
TITLE Pilha
.MODEL SMALL
.STACK 100h
.CODE
   MOV AX,@DATA
   MOV DS,AX

   MOV CX,3
LER:
   MOV AH,1
   INT 21h
   PUSH AX       ; armazena o caracter contido em AL na pilha
   LOOP LER

   MOV CX,3
EXIBIR:
   MOV AH,2
   POP DX        ; remove o caracter da pilha e armazena em DL
   INT 21h
   LOOP EXIBIR

   MOV AH,4Ch
   INT 21h
END
```

Atividade para entrega

Crie um programa em linguagem assembly chamado **ATIV8.ASM** que lê uma frase do teclado e exibe na linha seguinte a frase invertida, **utilizando a pilha para armazenar os caracteres** digitados pelo usuário. **A leitura e exibição dos caracteres devem ser implementadas dentro de um procedimento chamado INVERTE que será chamado pelo procedimento principal chamado MAIN.**

Exemplo:

C:\> ATIV8.EXE

Digite uma frase: **Organizacao de Computadores**

Frase invertida: **serodatupmoC ed oacazinagrO**

ENTREGA

Cada aluno deve:

1) Criar uma pasta em seu escaninho no AVA com o nome **Atividade8.**
2) Postar o arquivo **ATIV8.ASM** dentro da pasta **Atividade8**.