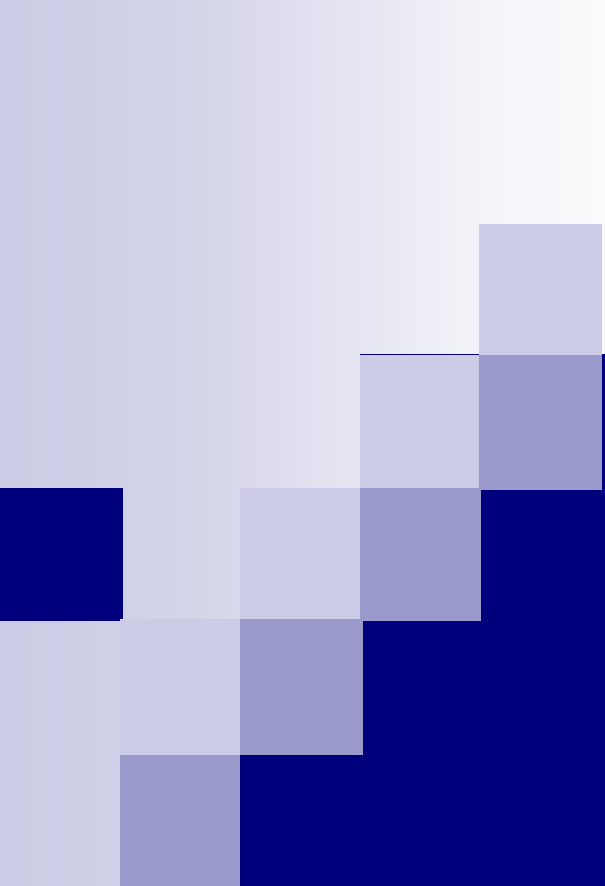




Linguagem Assembly parte 2

Prof. Maurício Braga

Modularizando o nosso código

- Em nosso último exemplo, tivemos que montar duas estruturas de repetição para fazer a mesma tarefa (imprimir uma frase na tela). Ex:

```
    ld hl, DATA
Loop2: ld a, (hl)
       cp 0
       ret z
       call CHPUT
       inc hl
       jr Loop2
```



Modularizando o nosso código

- No código anterior, se mudarmos o endereço de memória onde está a frase para um endereço onde tem outro texto, o algoritmo de impressão continuará o mesmo.
- Podemos então transformar essa parte em uma função (que chamaremos de MOSTRATXT), que chamaremos sempre que precisar.

Exercício

- Modifique no ultimo exemplo o código visto abaixo para transformá-lo em uma função MOSTRATXT. Essa função deverá receber como parâmetro de entrada em HL o endereço de memória onde se encontra o texto a ser impresso. O programa então deverá fazer uso dessa função para imprimir as duas frases (“Programa ASM” “Data: 05/11/2018”, colocando uma linha em branco entre elas.

```
ld hl, DATA
```

```
Loop2: ld a, (hl)
```

```
    cp 0
```

```
    ret z
```

```
    call CHPUT
```

```
    inc hl
```

```
    jr Loop2
```

Função MOSTRATXT

;-----

; MOSTRATXT - Função que mostra um texto terminada por 0.

; Entrada: HL - Aponta para o endereço do texto a ser exibido.

;-----

MOSTRATXT:

ld a, (hl)

cp 0

ret z

call CHPUT

inc hl

jr MOSTRATXT

Exemplo com Função MOSTRATXT

CHPUT: equ 0x00a2 ; imprime 1 caractere na tela

CLS: equ 0x00c3 ; limpa a tela

org 0xD000 ; Endereço de inicio do programa

start:

 xor a

 call CLS ; limpa a tela

Exemplo com Função MOSTRATXT

```
ld hl, NOMEDOPRG  
call MOSTRATXT  
ld hl, DATA  
call MOSTRATXT  
ret
```

; entrada da funcao: endereco em HL.

```
MOSTRATXT: ld a, (hl)  
           cp 0  
           ret z  
           call CHPUT  
           inc hl  
           jr MOSTRATXT
```

Exemplo com Função MOSTRATXT

NOMEDOPRG: DB "Programa ASM modularizado",13,10,10,0

DATA: DB "Data: 05/11/2018",13,10,10,0

end start

A instrução DJNZ

- O Z80 tem uma instrução que permite a criação de loops (estruturas de repetição) com variável contadora, a instrução DJNZ;
- A instrução DJNZ (Decrement and Jump if Not Zero), é uma instrução que decrementa o registrador B e faz um salto para o local da memória que indicarmos se B for diferente de zero.
- Vamos ver como funciona esse comando.

Funcionamento do comando DJNZ

- Precisamos definir uma etiqueta (no exemplo: LOOP) e usar um contador armazenado no registrador B.
 - LD B,X ; Indica repetição de X vezes.
 - LOOP: comando asm1
 - comando asm2
 - comando asm3...
 - djnz LOOP
- Ao chegar em DJNZ, B será decrementado em uma unidade ($B = B - 1$). Se $B \neq 0$, o loop será executado mais uma vez. Se $B=0$, a repetição termina.

Substituindo o DJNZ

- Caso o bloco de comando dentro do DJNZ cresça muito, o DJNZ precisará ser substituído por:

```
dec b
```

```
jp nz, label
```



Exercício

- Faça um programa que use a função MOSTRATXT que criamos anteriormente e imprima a frase: “Testando o loop!” 5 vezes.
- Utilize o comando DJNZ.

Solução

CHPUT: equ 0x00a2 ; imprime 1 caractere na tela
CLS: equ 0x00c3 ; limpa a tela

org 0xD000 ; Endereço de inicio do programa

start:

 xor a

 call CLS ; limpa a tela

Solução

LD B,5 ; Indica repetição de 5 vezes.

LOOP: ld hl, FRASE

call MOSTRATXT

DJNZ LOOP ; fim do laço.

ret

Solução

; entrada da funcao: endereco em HL.

MOSTRATXT: ld a, (hl)

cp 0

ret z

call CHPUT

inc hl

jr MOSTRATXT

FRASE: DB "Testando o loop com DJNZ.",13,10,10,0

end start

Loop com contador de 16 bits

- Pode ser feito da seguinte maneira:

```
LD BC,<Contador 16 bits do laço>
```

```
.loop
```

```
    ; seu código dentro do laço vem aqui
```

```
    DEC BC
```

```
    LD A,B
```

```
    OR C
```

```
    JP NZ,.loop
```



Guardando dados na memória

- Como o Z80 possui poucos registradores, precisamos em muitas situações salvar os valores que temos neles na memória, para liberar o seu uso.
- Essa necessidade costuma ocorrer também antes de chamarmos funções da BIOS, porque elas podem necessitar fazer uso dos registradores e aí irão sobrescrever os valores que tivermos colocado neles.
- Podemos então usar a memória para criar variáveis, salvando os dados lá, e recuperando-os quando precisarmos deles.

Guardando dados na memória

- Podemos reservar um espaço na memória para uma variável (variavel_1) usando o DB:
 - `VARIAVEL_1: DB 0`
- Para gravar um valor nessa variável, utilizaremos o comando load (LD):
 - `LD (VARIAVEL_1), A`
- Para recuperar o valor da memória, faremos:
 - `LD A, (VARIAVEL_1)`

Guardando dados na memória

- Outra forma de salvar dados na memória é usando as instruções **PUSH** e **POP**. Use PUSH para salvar dados na pilha (estrutura LIFO) e POP para recuperar.
- Apenas podemos usar push com AF, BC, DE, HL, IX e IY, e pop com AF, BC, DE e HL. Ex:
 - ☐ push af
 - ☐ pop af
- Útil também para copiar valores de um par de registradores de 16 bits para outro.

A função PINLIN

- A função PINLIN (endereço 0x00AE) lê uma string e a armazena em um buffer, cujo endereço – 1 byte é devolvido em HL.
- É preciso incrementar HL em uma posição para chegar a primeira letra da String.
- Vamos ver um exemplo de uso.

Exemplo

CHPUT: equ 0x00a2 ; imprime 1 letra na tela

CLS: equ 0x00c3

PINLIN: equ 0x00AE

org 0xD000

start:

 xor a

 call CLS ; limpa a tela

 ld hl, PROGRAMA

 call MostraTXT

Exemplo

ld hl, DATA
call MostraTXT

ld hl, PERGUNTA
call MostraTXT

call PINLIN ; Aguarda o usuário digitar a string e retorna em HL

inc HL ; incrementa HL porque PINLIN retorna the buffer - 1.
push HL ; salva o valor em HL

ld HL,RESP1
call MostraTXT

Exemplo

```
pop HL ; recupera o valor em HL  
call MostraTXT
```

```
ret
```

; imprime na tela. Entrada: HL (inicio do texto)
; texto deve terminar com 0 (NULL).

```
MostraTXT: ld a, (hl)
```

```
cp 0
```

```
ret z
```

```
call CHPUT
```

```
inc hl
```

```
jr MostraTXT
```

Exemplo

```
PROGRAMA:  DB "Programa le string do teclado",13,10,10,0  
DATA:      DB "Data: 06/11/2021",13,10,10,0  
PERGUNTA:  DB "Digite seu texto:",13,10,0  
RESP1:     DB 13,10,"Voce digitou: ",0
```

end start

Exercício

- Crie uma função CONV_Maiuscula. A função deverá receber uma string (através do registrador HL) e modificar todas as letras para maiúscula.

Solução

; entrada: HL - inicio do texto, saida: HL

CONV_MAIUSCULA:

ld a,(hl)

cp 0

ret z ; se encontrou 0, a string acabou, pode retornar

cp 'a'

jr c,CONV_MAIUSCULA_OK ; subtração deu negativo, não é uma
; letra minúscula, passa para a próxima letra

cp 'z'+1

jr nc,CONV_MAIUSCULA_OK ; se subtração não deu negativo, não é uma
; letra minúscula, já passou das minúsculas

sub 32 ; subtrai 32 para transformar a letra minúscula em maiúscula

ld (hl),a ; escreve a letra maiúscula por cima da minúscula

CONV_MAIUSCULA_OK: ; passa para a próxima letra

inc hl

jr CONV_MAIUSCULA

Exercício

- Modifique o último exemplo para, após perguntar um texto ao usuário e imprimir na tela o texto que o usuário digitou, imprimir também o texto digitado pelo usuário em maiúscula.

Solução

CHPUT: equ 0x00a2 ; imprime 1 letra na tela

CLS: equ 0x00c3

PINLIN equ 0x00AE

org 0xD000

start:

 xor a

 call CLS ; limpa a tela

 ld hl, PROGRAMA

 call MostraTXT

 ld hl, DATA

 call MostraTXT

Solução

ld hl, PERGUNTA
call MostraTXT

call PINLIN

inc HL ; incrementa HL porque PINLIN retorna the buffer - 1.

push HL ; salva o valor em HL

ld HL,RESP1
call MostraTXT

Solução

pop HL ; recupera o valor em HL
push HL ; salva de nov o valor em HL
call MostraTXT ; imprime o que o usuario digitou

ld HL,RESP2
call MostraTXT

pop HL
push HL ; salva de nov o valor em HL
CALL CONV_MAIUSCULA ; converte as letras para maiuscula

POP HL ; pega o endereço de HL com o texto em maiuscula
call MostraTXT
ret

Solução

; imprime na tela. Entrada: HL (inicio do texto)
; texto deve terminar com 0 (NULL).

MostraTXT: ld a, (hl)

cp 0

ret z

call CHPUT

inc hl

jr MostraTXT

Solução

; entrada: HL - inicio do texto, saida: HL

CONV_MAIUSCULA:

ld a,(hl)

cp 0

ret z ; se encontrou 0, a string acabou, pode retornar

cp 'a'

jr c,CONV_MAIUSCULA_OK ; subtração deu negativo, não é

; uma letra minúscula, passa para a próxima letra

cp 'z'+1

jr nc,CONV_MAIUSCULA_OK ; se subtração não deu negativo, não é
uma letra minúscula, já passou das minúsculas

sub 32 ; subtrai 32 para transformar a letra minúscula em maiúscula

ld (hl),a ; escreve a letra maiúscula por cima da minúscula

CONV_MAIUSCULA_OK: ; passa para a próxima letra

inc hl

jr CONV_MAIUSCULA

Solução

PROGRAMA: DB "Programa le string do teclado",13,10,10,0

DATA: DB "Data: 06/11/2021",13,10,10,0

PERGUNTA: DB "Digite seu texto:",13,10,0

RESP1: DB 13,10,"Voce digitou: ",0

RESP2: DB 13,10,"Agora em maiuscula: ",0

end start

Exercício

- Crie, utilizando como base a função CONV_Maiuscula, uma função CONV_MINUSCULA para converter um texto passado como parâmetro para letras minúsculas. A função deverá receber uma string (através do registrador HL) e modificar todas as letras para minúscula. Em seguida, modifique o ultimo exemplo para, após imprimir o que o usuário digitou, imprimir também o texto digitado pelo usuário em minúscula.

Solução

; entrada: HL - inicio do texto, saida: HL

CONV_MINUSCULA:

ld a,(hl)

cp 0

ret z ; se encontrou 0, a string acabou, pode retornar

cp 'A'

jr c,CONV_MINUSCULA_OK ; subtração deu negativo, nao eh uma letra maiuscula

cp 'Z'+1

jr nc,CONV_MINUSCULA_OK ; se subtracao nao deu negativo, nao eh uma letra maiuscula, jah passou de 'Z'

add a,32 ; adiciona 32 para transformar a letra maiuscula em minuscula

ld (hl),a ; escreve a letra maiuscula por cima da minuscula

CONV_MINUSCULA_OK: ; passa para a proxima letra

inc hl

jr CONV_MINUSCULA

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

CHPUT: equ 0x00a2 ; imprime 1 letra na tela

CLS: equ 0x00c3

PINLIN equ 0x00AE

org 0xD000

start:

xor a

call CLS ; limpa a tela

ld hl, PROGRAMA

call MostraTXT

ld hl, AUTOR

call MostraTXT

ld hl, PERGUNTA

call MostraTXT

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

call PINLIN

inc HL ; incrementa HL porque PINLIN retorna the buffer - 1.

push HL ; salva o valor em HL

ld HL,RESP1

call MostraTXT

pop HL ; recupera o valor em HL

push HL ; salva de nov o valor em HL

call MostraTXT ; imprime o que o usuario digitou

ld HL,RESP2

call MostraTXT

POP HL

push HL ; salva de nov o valor em HL

CALL CONV_MAIUSCULA ; converte as letras todas para maiuscula

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

POP HL

push HL

call MostraTXT

ld HL,RESP3

call MostraTXT

POP HL

push HL ; salva de nov o valor em HL

CALL CONV_MINUSCULA ; converte as letras todas para maiuscula

POP HL

call MostraTXT

ret

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

; imprime na tela. Entrada: HL (inicio do texto
; texto deve terminar com 0 (NULL).

MostraTXT: ld a, (hl)

cp 0

ret z

call CHPUT

inc hl

jr MostraTXT

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

; entrada: HL - inicio do texto, saida: HL

CONV_MAIUSCULA:

ld a,(hl)

cp 0

ret z ; se encontrou 0, a string acabou, pode retornar

cp 'a'

jr c,CONV_MAIUSCULA_OK ; subtração deu negativo, nao eh uma letra minuscula, passa para a proxima letra

cp 'z'+1

jr nc,CONV_MAIUSCULA_OK ; se subtracao nao deu negativo, nao eh uma letra minuscula, jah passou das minusculas

sub 32 ; subtrai 32 para transformar a letra minuscula em maiuscula

ld (hl),a ; escreve a letra maiuscula por cima da minuscula

CONV_MAIUSCULA_OK: ; passa para a proxima letra

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

```
inc hl
```

```
jr CONV_MAIUSCULA
```

; entrada: HL - inicio do texto, saida: HL

CONV_MINUSCULA:

```
ld a,(hl)
```

```
cp 0
```

```
ret z ; se encontrou 0, a string acabou, pode retornar
```

```
cp 'A'
```

```
jr c,CONV_MINUSCULA_OK ; subtração deu negativo, nao eh uma letra  
maiuscula
```

```
cp 'Z'+1
```

```
jr nc,CONV_MINUSCULA_OK ; se subtracao nao deu negativo, nao eh  
uma letra maiuscula, jah passou de 'Z'
```

```
add a,32 ; adiciona 32 para transformar a letra em minuscula
```

Exemplo função CONV_MAIUSCULA e CONV_MINUSCULA

```
    Id (hl),a      ; escreve a letra maiuscula por cima da minuscula
CONV_MINUSCULA_OK:      ; passa para a proxima letra
    inc hl
    jr CONV_MINUSCULA
```

```
PROGRAMA:    DB "Programa le string do teclado",13,10,10,0
AUTOR:       DB "Autor:desconhecido",13,10,10,0
PERGUNTA:    DB "Digite seu texto:",13,10,0
RESP1:       DB 13,10,"Voce digitou: ",0
RESP2:       DB 13,10,"Agora em maiuscula: ",0
RESP3:       DB 13,10,"Agora em minuscula: ",0
```

```
end start
```

A função CHGCOLOR

- A função CHCOLOR (endereço 0x0062) permite mudar a cor de frente e a cor de fundo da tela.
- A cor de frente é setada no endereço de memória 0xF3E9 e a cor de fundo é setada no endereço 0xF3EA.
- Vamos ver um exemplo de uso.

Exemplo

CHGCLR: EQU #0062

COR_FRENTE: EQU #F3E9

COR_FUNDO: EQU #F3EA

ORG 0xD000

start:

LD A,4

LD (COR_FRENTE),A ; seta a cor de frente como azul

LD A,15

LD (COR_FUNDO),A ; seta a cor de fundo como branca

CALL CHGCLR ; ativa a mudança de cores

RET ; encerra o programa

end start;

A função CHGET

- A função CHGET (endereço 0x009F) aguarda o usuário pressionar uma tecla.
- O código ASCII da tecla pressionada pelo usuário será retornada no acumulador A.
- Exemplo de uso:
`call CHGET`

Exercício

- Faça um programa que monte o menu abaixo e solicite o número da opção que o usuário deseja. Em seguida, o programa deverá imprimir a frase: “A opcao escolhida foi: ” seguido da tecla digitada.

Programa Menu

1 – opcao 1

2 – opcao 2

3 – opcao 3

4 – opcao 4

Sua escolha:

- Dicas: Use a função CHGET (endereço 0x009F) e as funções CLS e MOSTRATXT vistas anteriormente. Defina um espaço (ex: VAR1) na memória para guardar a tecla pressionada e use LD (VAR1),a para armazenar o valor do registrador A nesse espaço.

Solução

CHPUT: equ 0x00a2 ; imprime 1 caractere na tela

CLS: equ 0x00c3

CHGET equ 0x009F ; lê uma tecla. saída: A - código ASCII
do caractere lido

org 0xD000

start:

xor a

call CLS ; limpa a tela

ld hl, NOMEDOPRG ; inicia exibição do menu

call MOSTRATXT

ld hl, OPCAO_1

call MOSTRATXT

Solução

```
ld hl, OPCAO_2  
call MOSTRATXT  
ld hl, OPCAO_3  
call MOSTRATXT  
ld hl, OPCAO_4  
call MOSTRATXT
```

```
ld hl, PERGUNTA ; exibe a pergunta  
call MOSTRATXT
```

```
call CHGET ; recebe a tecla pressionada pelo usuário.  
LD (VAR1),A ; guarda a tecla pressionada pelo usuário.
```

Solução

```
ld hl, RESP1  
call MOSTRATXT
```

```
LD A,(VAR1)    ; recupera da memoria a tecla pressionada  
call CHPUT      ; imprime o caractere digitado pelo usuário
```

```
ret            ; encerra o programa
```

Solução

MOSTRATXT:

ld a, (hl)

cp 0

ret z

call CHPUT

inc hl

jr MOSTRATXT

NOMEDOPRG: DB "Programa Menu",13,10,10,0

OPCAO_1: DB "1 - Opcao 1",13,10,10,0

OPCAO_2: DB "2 - Opcao 2",13,10,10,0

OPCAO_3: DB "3 - Opcao 3",13,10,10,0

OPCAO_4: DB "4 - Opcao 4",13,10,10,0

Solução

PERGUNTA: DB "Sua escolha:",0

RESP1: DB 13,10,10," A opcao escolhida foi: ",0

VAR1: DB 0

end start



Exercício

- Faça um programa que pergunte ao usuário quantas vezes ele deseja imprimir uma frase na tela. Em seguida o programa deverá imprimir “texto sendo repetido.” tantas vezes quanto o usuário solicitou (o número de impressões deverá ficar entre 0 e 9).

Solução

CHPUT: equ 0x00a2 ; imprime 1 caractere na tela
CLS: equ 0x00c3 ; limpa a tela
CHGET equ 0x009F ; lê uma tecla em A - código ASCII do carac lido

org 0xD000 ; Endereço de inicio do programa

start:

```
xor a
call CLS           ; limpa a tela
; Mostra informações do programa e faz pergunta
LD HL,NOMEDOPRG ; Indica texto do nome do programa
CALL MOSTRATXT     ; Mostra texto
LD HL,AUTOR       ; Indica texto do nome do autor
CALL MOSTRATXT     ; Mostra texto
LD HL,PERGU1       ; Indica texto da pergunta
```

Solução

CALL MOSTRATXT ; Mostra texto

; Espera tecla ser pressionada e coloca seu valor em A

CALL CHGET ; recebe a tecla pressionada pelo usuário.

LD (VAR1),A ; guarda a tecla pressionada pelo usuário.

; Mostra a resposta

LD HL,RESP1 ; Indica texto da resposta

CALL MOSTRATXT ; Mostra texto

LD A,(VAR1) ; Copia para A o conteúdo de VAR1

CALL CHPUT ; exibe o valor na tela.

; Mostra a segunda resposta

LD HL,RESP2 ; Indica texto da resposta

CALL MOSTRATXT ; Mostra texto

Solução

```
LD A,(VAR1)    ; Copia para A o conteúdo de VAR1
sub 48          ; converte para um número
LD (NUM),A      ; Grava o valor do numero convertido na
                ; MEMORIA.
```

```
ld b,a
```

```
LOOP: LD HL,FRASE ; Indica Frase
      LD A,B      ; Copia em A o contador do loop
      LD (NUM),A  ; salva o contador na memória.
      CALL MOSTRATXT ; Mostra texto
      LD A,(NUM)  ; copia para A o valor do contador da memória
      LD B,A      ; Restaura o valor do contador que tinha sido
                  ; salvo na memória
      DJNZ LOOP   ; fim do laço.
      ret         ; encerra.
```

Solução

; entrada da funcao: endereco em HL.

MOSTRATXT: ld a, (hl)

 cp 0

 ret z

 call CHPUT

 inc hl

 jr MOSTRATXT

NUM: DB 0

NOMEDOPRG: DB 'Programa repeti',135,132,'o controlada pelo
usu',160,'rio',13,10,0

AUTOR: DB 'Autor: Mauricio Braga',13,10,10,0

PERGU1: DB 'Digite a quantidade de repeticoes desejada (valor de 0 a 9):
,0

Solução

RESP1: DB 13,10,10,'O valor pressionado foi: ',0

RESP2: DB 13,10,10,'O texto a seguir sera repetido entao '

VAR1: DB 0

RESP3: DB ' vezes.',13,10,10,0

FRASE: DB 'texto sendo repetido.',13,10,0

end start



ASM x linguagem alto nível

- Um programa escrito diretamente em linguagem assembly normalmente será menor e rodará mais rápido que o mesmo programa escrito em uma linguagem de alto nível.
- Apenas como um exemplo, faremos um comparativo do tamanho da versão compilada de dois programas simples escritos em C e os mesmos programas feito em ASM.
- No teste foi usado o MSX C 1.2 e o M80 da Microsoft.

ASM x linguagem alto nível

- O MSX C gera um código ASM que será compilado pelo M80, então o mesmo compilador (M80) irá produzir o resultado final tanto do código escrito direto em ASM quanto o código traduzido do C.
- Como o programa em C é gerado no formato .com, optamos por fazer um programa equivalente em ASM (formato .com).



Programas .com

- São executados a partir do interpretador de comandos do MSX-DOS.
- São carregados no endereço 100h, de onde se inicia sua execução.
- Fazem uso do BDOS.



Primeiro programa em C

```
#include <stdio.h>
```

```
main() {  
    printf("meu primeiro programa!");  
}
```

Primeiro programa em ASM

STROUT EQU 9

BDOS EQU 5

start:

LD DE,FRASE ; Indica posição do texto

LD C,STROUT ; Indica função a ser executada pelo

BDOS

CALL BDOS ; Manda o BDOS executar.

JP 0 ; Volta ao MSX-DOS

FRASE: DB 'meu primeiro programa!\$'

end



Resultado 1

- Tamanho do programa compilado escrito em C: 3467 bytes.
- Tamanho do programa compilado escrito em assembly: 37 bytes (aprox. 1% do tamanho do programa em C).
- Esse programa entretanto é muito pequeno, então só mostra o overhead máximo adicionado por um programa C. Programas maiores devem apresentar proporcionalmente um overhead menor.
- Vamos ver um exemplo envolvendo um programa ainda pequeno, mas com interação com o usuário.

Segundo programa em C

```
#include <stdio.h>
```

```
main() {
```

```
    int n = 0;
```

```
    int c = 1;
```

```
    printf("Programa 3 - otimizando a estrutura\n");
```

```
    printf("Por Fulano de tal\n\n");
```

```
    printf("Digite a quantidade de repeticoes desejada (valor de 0 a  
9): \n\n");
```

```
    scanf("%d",&n);
```

```
    printf("O valor pressionado foi: %d\n\n",n);
```

```
    printf("O texto a seguir sera repetido entao %d vezes.\n\n",n);
```

```
    while (c <=n) {
```

```
        printf("texto sendo repetido. \n");
```

```
        c = c + 1;
```

```
    }
```

```
}
```

Segundo programa em ASM

BDOS EQU 5

CONIN EQU 1

CONOUT EQU 2

STROUT EQU 9

START:

; Mostra informações do programa e faz pergunta

LD DE,NOMEDOPRG ; Indica texto do nome do programa

CALL MOSTRATXT ; Mostra texto

LD DE,AUTOR ; Indica texto do nome do autor

CALL MOSTRATXT ; Mostra texto

LD DE,PERGU1 ; Indica texto da pergunta

CALL MOSTRATXT ; Mostra texto

; Espera tecla ser pressionada e coloca seu valor em A

LD C,CONIN ; Indica a função que pega uma tecla

CALL BDOS ; Chama BDOS

Segundo programa em ASM

LD (VAR1),A ; Grava o valor da tecla na MEMORIA.

; Mostra a resposta

LD DE,RESP1 ; Indica texto da resposta

CALL MOSTRATXT ; Mostra texto

LD A,(VAR1) ; Copia para A o conteúdo de VAR1

LD E,A ; Copia para E o conteúdo de A

LD C,CONOUT ; Indica a função exibe um caractere na tela

CALL BDOS ; Chama BDOS

; Mostra a segunda resposta

LD DE,RESP2 ; Indica texto da resposta

CALL MOSTRATXT ; Mostra texto

LD A,(VAR1) ; Copia para A o conteúdo de VAR1

sub 48 ; converte para um número

LD (NUM),A ; Grava o valor do num convertido na RAM.

Segundo programa em ASM

ld b,a

LOOP: LD DE,FRASE ; Indica Frase

LD A,B ; Copia em A o contador do loop

LD (NUM),A ; salva o contador na memória.

CALL MOSTRATXT ; Mostra texto

LD A,(NUM) ; copia para A o valor do contador da RAM

LD B,A ; Restaura o valor do contador salvo na memória

DJNZ LOOP ; fim do laço.

JP 0 ; Volta ao MSX-DOS

Segundo programa em ASM

MOSTRATXT:

LD C,STROUT ; Indica função de mostrar texto do BDOS

CALL BDOS ; Manda o BDOS executar.

RET ; Retorna

NUM: DB 0

NOMEDOPRG: DB 'Programa 3 - Otimizando a estrutura',13,10,'\$'

AUTOR: DB 'Por Fulano de tal',13,10,10,'\$'

PERGU1: DB 'Digite a quantidade de repeticoes desejada (valor de 0 a 9):
\$'

RESP1: DB 13,10,10,'O valor pressionado foi: \$'

RESP2: DB 13,10,10,'O texto a seguir sera repetido entao '

VAR1: DB 0

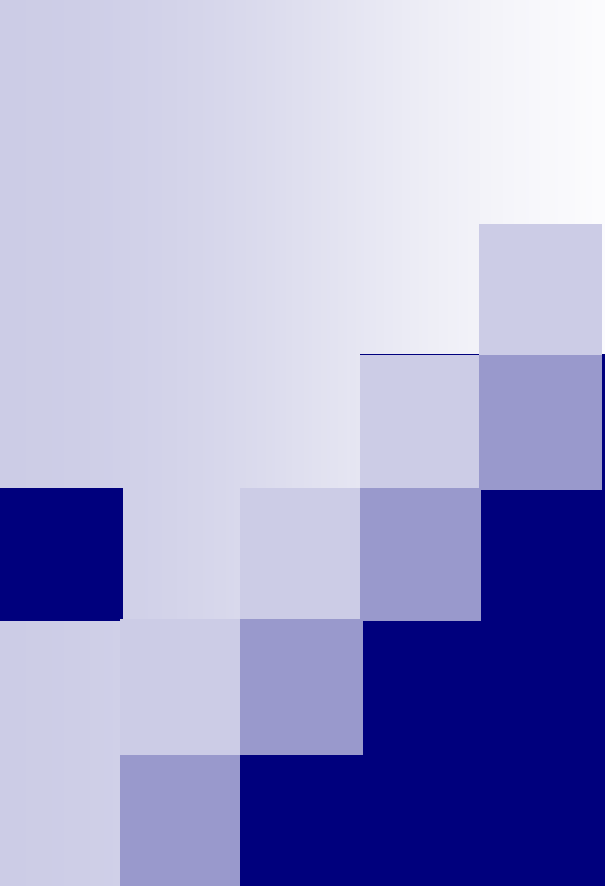
RESP3: DB ' vezes.',13,10,10,'\$'

FRASE: DB 'texto sendo repetido.',13,10,'\$'

END

Resultado 2

- Tamanho do programa compilado escrito em C: 4988 bytes.
- Tamanho do programa compilado escrito em assembly: 313 bytes (aprox. 6% do tamanho do programa em C).
- Programas maiores provavelmente apresentarão diferenças ainda menores em termos proporcionais, já que o overhead adicionado pelo uso de uma linguagem de alto nível possivelmente se manterá relativamente constante.
- Esse resultado entretanto mostra que os ganhos em memória / espaço em disco (assim como a performance) podem ser consideráveis quando se programa em ASM.



Linguagem Assembly parte 2

Prof. Maurício Braga