DOODLE

Online Assembler - NASM Compiler IDE

```
1  global _start
2
3  extern print_str, pula_linha, read_str1, read_str2
4
5  global linha, n1_max, n1, n2_max, n2:data
6
7  ; ---------------- LISTA DE EQUIVALENCIAS ------------------
8  LF equ 0x0A     ; Salta linha
9  n1_max equ 20       ; Limite para entrada de caracteres
10 n2_max equ 20
11 ; ---------------- DADOS INICIALIZADOS --------------------
12 segment .data
13 mensini db LF,"Este programa recebe dois numeros, realiza a soma deles e converte par
14         db LF, LF
15 tam_ini equ $-mensini       ; Calcula o tamanho da string
16 pedido1 db LF, "Digite o primeiro numero: "
17 pedido1_tam equ $-pedido1
18 pedido2 db LF, "Digite o segundo numero: "
19 pedido2_tam equ $-pedido2
20 mens_conv db LF, "O valor do reg eh: "
21 tam_conv equ $-mens_conv
22 linha db LF     ; Pular uma linha
23 ; --------------- DADOS NAO INICIALIZADOS ------------------
24 segment .bss
25 n1 resb n1_max      ; Recebe as strings (buffer)
26 n2 resb n2_max  ; Eh usado tambem para os algarismos
27
28 num1 resd 1     ; Recebe o numero 1 formatado
29 num2 resd 1     ; Recebe o numero 1 formatado
30 qde1 resd 1     ; Quantidade de algarismos dos numeros
31 qde2 resd 1     ; Utilizada nos dois
32 numf resb 7     ; Recebe a string da soma (formatada)
33 qde3 resd 1     ; Quantidade de algarismos da soma
34 vt resd 1           ; Variavel de teste
35 ; ---------------------- CODIGOS --------------------------
36 segment .text
37 _start:                 ; Entrada do programa
38 ; Mostra a mensagem inicial
39     mov edx, tam_ini        ; Carrega o tamanho da str inicial
40     mov ecx, mensini        ; Aponta para a str inicial
41     call print_str      ; Procedimento que exibe a str
42 ; Entra com o primeiro numero
43 pedir1:
44     mov edx, pedido1_tam    ; Carrega o tamanho da primeira str
45     mov ecx, pedido1        ; Aponta para a primeira str
46     call print_str      ; Procedimento que exibe a str
47     call read_str1      ; Le o primeiro numero
48     dec eax
49 ; A entrada nao pode ser vazia, ou seja, conter soh o <Enter>
50     jz pedir1           ; Se for, pede para ler de novo
51 ; Segundo teste
52     cmp eax, 6          ; Limite de caracteres
53     ja pedir1           ; Se for maior, digitar novamente
54 ; Passando desse ponto significa que o numero de caracteres
55 ; eh valido
56     mov [qde1], eax     ; Salva a quantidade de caracteres
57 ; Validar os caracteres (tem que ser de algarismos)
58     mov [vt], dword 0   ; Inicializa uma variavel de teste
59     call valida     ; Procedimento que testa caracteres
60     cmp [vt], dword 0   ; Se voltar zerada nao houve erro
61     jne pedir1          ; Caso contrario, digitar novamente
62 ; O procedimento valida chama o formatador
63 ; O numero formatado retorna em EAX.
64     mov [num1], eax     ; Salva o primeiro numero
65
66 pedir2:
```

```
pedir2:
    mov edx, pedido2_tam    ; Carrega o tamanho da primeira str
    mov ecx, pedido2        ; Aponta para a primeira str
    call print_str      ; Procedimento que exibe a str
    call read_str2      ; Le o primeiro numero
    dec eax
; A entrada nao pode ser vazia, ou seja, conter soh o <Enter>
    jz pedir2           ; Se for, pede para ler de novo
; Segundo teste
    cmp eax, 6          ; Limite de caracteres
    ja pedir2           ; Se for maior, digitar novamente
; Passando desse ponto significa que o numero de caracteres
; eh valido
    mov [qde2], eax     ; Salva a quantidade de caracteres
; Validar os caracteres (tem que ser de algarismos)
    mov [vt], dword 0   ; Inicializa uma variavel de teste
    call valida2        ; Procedimento que testa caracteres
    cmp [vt], dword 0   ; Se voltar zerada nao houve erro
    jne pedir2          ; Caso contrario, digitar novamente
; O procedimento valida chama o formatador
; O numero formatado retorna em EAX.a
    mov [num2], eax     ; Salva o primeiro numero
;-----------------------------------------
    add eax, [num1]
    call format_num
; Mensagem da conv
    mov edx, tam_conv
    mov ecx, mens_conv
    call print_str
;mostrar conteudo memoria num1

    mov edx, [qde3]
    mov ecx, numf
    call print_str
call pula_linha
call pula_linha
fim:
    mov eax, 1          ; Servico exit
    int 0x80            ; Encerra o programa
; ----------------- AREA DOS PROCEDIMENTOS -----------------

valida:
; Varre a string supostamente numerica e testa cada caracter
    mov esi, 0          ; Inicializa ESI
                    ; (indice de varredura)
                    ; A base eh o ponteiro, n1
val_car1:               ; Inicio do loop
    mov al, [n1 + esi]  ; Busca o caracter atual
    sub al, 48          ; Tenta converter caracter em
                    ; algarismo
    cmp al, 9           ; Se for algarismo vai de 0 a 9
    ja erro         ; Se for acima de 9 nao eh
                    ; algarismo, acusa o erro para
                    ; comecar de novo
    mov [n1 + esi], al  ; Coloca o algarismo na posicao
                    ; ESI (antigo caractere)
                    ; do mesmo buffer (n1)
    inc esi         ; Aponta para o proximo caratere
    cmp esi, [qde1]     ; Verifica se jah terminou a
                    ; varredura
    jne val_car1        ; Se ainda nao, volta para o inicio
                    ; do laco de repeticao
; Ao sair vetor de caracteres virou vetor de algarismos
; Nao teve erro
    call format_num1        ; Procedimento chamando
                    ; procedimento
; Retorna um numero formatado em EAX
    jmp sai_valida      ; Contorna o bloco de atualizacao
                    ; da vt (porque nao teve erro)
erro:
    mov [vt], dword 1
sai_valida:
```

```
138     ret
139 ; ---------------------------------------------------------
140 valida2:
141 ; Varre a string supostamente numerica e testa cada caracter
142     mov esi, 0          ; Inicializa ESI
143                         ; (indice de varredura)
144                         ; A base eh o ponteiro, n1
145 val_car2:               ; Inicio do loop
146     mov al, [n2 + esi]  ; Busca o caracter atual
147     sub al, 48          ; Tenta converter caracter em
148                         ; algarismo
149     cmp al, 9           ; Se for algarismo vai de 0 a 9
150     ja erro2            ; Se for acima de 9 nao eh
151                         ; algarismo, acusa o erro para
152                         ; comecar de novo
153     mov [n2 + esi], al  ; Coloca o algarismo na posicao
154                         ; ESI (antigo caractere)
155                         ; do mesmo buffer (n1)
156     inc esi             ; Aponta para o proximo caratere
157     cmp esi, [qde2]     ; Verifica se jah terminou a
158                         ; varredura
159     jne val_car2        ; Se ainda nao, volta para o inicio
160                         ; do laco de repeticao
161 ; Ao sair vetor de caracteres virou vetor de algarismos
162 ; Nao teve erro
163     call format_num2        ; Procedimento chamando
164                         ; procedimento
165 ; Retorna um numero formatado em EAX
166     jmp sai_valida2     ; Contorna o bloco de atualizacao
167                         ; da vt (porque nao teve erro)
168 erro2:
169     mov [vt], dword 1
170 sai_valida2:
171     ret
172 ; ---------------------------------------------------------
173 format_num1:
174 ; Formata um numero a partir dos algarismos no buffer n1
175 ; Retorna o numero formatado em EAX
176     mov esi, 0          ; Inicializa o indice
177     mov eax, 0          ; Inicializa o acumulador
178     mov ebx, 10     ; Base decimal, multiplicador
179 ini_loop1:
180     mul ebx         ; EAX * EBX = EDX:EAX.
181                         ; O resultado fica em EAX (cabe)
182     mov edx, 0          ; DL vai ser usado para receber o
183                         ; proximo algarismo
184     mov dl, [n1 + esi]  ; DL recebe o algarismo atual
185     add eax, edx        ; O acumulador recebe o valor
186                         ; anterior mais o algarismo atual
187 ; Algoritimo da formatacao: multiplica por 10
188 ; e soma com o proximo
189     inc esi         ; Aponta para o proximo digito
190     cmp esi, [qde1]     ; Verifica se jah chegou ao
191     je sai_fornum       ; final do buffer
192     jmp ini_loop1       ; Se ainda nao, proximo digito
193 sai_fornum:
194 ; Numero formatado em EAX
195     ret
196
197 ;-------------------------
198 format_num2:
199 ; Formata um numero a partir dos algarismos no buffer n1
200 ; Retorna o numero formatado em EAX
201     mov esi, 0          ; Inicializa o indice
202     mov eax, 0          ; Inicializa o acumulador
203     mov ebx, 10     ; Base decimal, multiplicador
204 ini_loop2:
205     mul ebx         ; EAX * EBX = EDX:EAX.
206                         ; O resultado fica em EAX (cabe)
207     mov edx, 0          ; DL vai ser usado para receber o
208                         : proximo algarismo
```

```
    mov dl, [n2 + esi]  ; DL recebe o algarismo atual
    add eax, edx        ; O acumulador recebe o valor
                        ; anterior mais o algarismo atual
; Algoritimo da formatacao: multiplica por 10
; e soma com o proximo
    inc esi         ; Aponta para o proximo digito
    cmp esi, [qde2]     ; Verifica se jah chegou ao
    je sai_fornum2      ; final do buffer
    jmp ini_loop2       ; Se ainda nao, proximo digito
sai_fornum2:
; Numero formatado em EAX
    ret

;-------------------------
format_num:
; Constroi uma string com o numero em EAX
; Algoritmo: divide sucessivamento por 10
; Tomam-se os restos na ordem inversa
; Converte-se em caracteres
    mov ebx, 10     ; Divisor, base destino
    mov esi, 0          ; Inicializa o contador de restos
ida:                    ; Inicio da formatacao
    mov edx, 0          ; Interfere na divisao
    div ebx         ; Divide o numero por 10
                    ; O quociente fica em EAX
                    ; e resto em EDX
    push rdx            ; Armazena na pilha o valor de DL
                    ; Corresponde a um algarismo
    inc esi         ; Aponta para a proxima posicao
    cmp eax, 0          ; Quando o quociente eh 0
    je sai          ; Termina a "ida"
                    ; (ESI - 1) restos foram empilhados
; Se ainda nao terminou
    jmp ida         ; Retorna ao inicio do laco
sai:
    mov edi, 0          ; Indice do buffer da soma

    mov [qde3], esi     ; Salva a quantidade
                ; Vai ser usada como tamanho da
                    ; string ao exibir a soma
volta:
    pop rax         ; Recuperando os restos
                    ; na ordem inversa
    add al, 48          ; Formatando caracter atual
    mov [numf + edi], al    ; Colocando no buffer
    inc edi         ; Aponta para a proxima posicao

    dec esi         ; Desconta o ESI
    jnz volta           ; Ateh zerar
    ret
;-------------------------------------------------------------
```

Execute Mode, Version, Inputs & Arguments

Stdin Inputs

Result