

Pontifícia Universidade Católica de Goiás

Curso: Arquitetura de computadores 1 (CMP1057) Turma: C011

Data: 28 de novembro de 2020

Alunos: Alexssander Pereira de Moraes 2019.1.0028.0066-6

Felipe Camilo Alves

2019.1.0028.0017-8

Avaliação de N2 Laboratório

Código "Principal.asm":



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 2 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:
```

```

67     mov edx, pedido2_tam    ; Carrega o tamanho da primeira str
68     mov ecx, pedido2       ; Aponta para a primeira str
69     call print_str          ; Procedimento que exibe a str
70     call read_str2          ; Le o primeiro numero
71     dec eax
72 ; A entrada nao pode ser vazia, ou seja, conter soh o <Enter>
73     jz pedir2               ; Se for, pede para ler de novo
74 ; Segundo teste
75     cmp eax, 6               ; Limite de caracteres
76     ja pedir2               ; Se for maior, digitar novamente
77 ; Passando desse ponto significa que o numero de caracteres
78 ; eh valido
79     mov [qde2], eax          ; Salva a quantidade de caracteres
80 ; Validar os caracteres (tem que ser de algarismos)
81     mov [vt], dword 0        ; Inicializa uma variavel de teste
82     call valida2             ; Procedimento que testa caracteres
83     cmp [vt], dword 0        ; Se voltar zerada nao houve erro
84     jne pedir2              ; Caso contrario, digitar novamente
85 ; O procedimento valida chama o formatador
86 ; O numero formatado retorna em EAX.a
87     mov [num2], eax          ; Salva o primeiro numero
88 ;-----
89     add eax, [num1]
90     call format_num
91 ; Mensagem da conv
92     mov edx, tam_conv
93     mov ecx, mens_conv
94     call print_str
95 ;mostrar conteudo memoria num1
96
97     mov edx, [qde3]
98     mov ecx, numf
99     call print_str
100 call pula_linha
101 call pula_linha
102 fim:
103     mov eax, 1               ; Servico exit
104     int 0x80                 ; Encerra o programa
105 ; ----- AREA DOS PROCEDIMENTOS -----
106
107 valida:
108 ; Varre a string supostamente numerica e testa cada caracter
109     mov esi, 0               ; Inicializa ESI
110                                ; (indice de varredura)
111                                ; A base eh o ponteiro, n1
112 val_car1:                    ; Inicio do loop
113     mov al, [n1 + esi]        ; Busca o caracter atual
114     sub al, 48                ; Tenta converter caracter em
115                                ; algarismo
116     cmp al, 9                 ; Se for algarismo vai de 0 a 9
117     ja erro                   ; Se for acima de 9 nao eh
118                                ; algarismo, acusa o erro para
119                                ; comecar de novo
120     mov [n1 + esi], al        ; Coloca o algarismo na posicao
121                                ; ESI (antigo caractere)
122                                ; do mesmo buffer (n1)
123     inc esi                   ; Aponta para o proximo caractere
124     cmp esi, [qde1]           ; Verifica se jah terminou a
125                                ; varredura
126     jne val_car1              ; Se ainda nao, volta para o inicio
127                                ; do laco de repeticao
128 ; Ao sair vetor de caracteres virou vetor de algarismos
129 ; Nao teve erro
130     call format_num1          ; Procedimento chamando
131                                ; procedimento
132 ; Retorna um numero formatado em EAX
133     jmp sai_valida            ; Contorna o bloco de atualizacao
134                                ; da vt (porque nao teve erro)
135 erro:
136     mov [vt], dword 1
137 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 caractere
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 ; Algoritmo 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

```

```

209     mov dl, [n2 + esi] ; DL recebe o algarismo atual
210     add eax, edx       ; O acumulador recebe o valor
211                       ; anterior mais o algarismo atual
212 ; Algoritmo da formatacao: multiplica por 10
213 ; e soma com o proximo
214     inc esi           ; Aponta para o proximo digito
215     cmp esi, [qde2]   ; Verifica se jah chegou ao
216     je sai_fornum2    ; final do buffer
217     jmp ini_loop2     ; Se ainda nao, proximo digito
218 sai_fornum2:
219 ; Numero formatado em EAX
220     ret
221
222 ;-----
223 format_num:
224 ; Constroi uma string com o numero em EAX
225 ; Algoritmo: divide sucessivamente por 10
226 ; Tomam-se os restos na ordem inversa
227 ; Converte-se em caracteres
228     mov ebx, 10       ; Divisor, base destino
229     mov esi, 0        ; Inicializa o contador de restos
230 ida:
231     mov edx, 0        ; Interfere na divisao
232     div ebx           ; Divide o numero por 10
233                       ; O quociente fica em EAX
234                       ; e resto em EDX
235     push rdx          ; Armazena na pilha o valor de DL
236                       ; Corresponde a um algarismo
237     inc esi           ; Aponta para a proxima posicao
238     cmp eax, 0        ; Quando o quociente eh 0
239     je sai            ; Termina a "ida"
240                       ; (ESI - 1) restos foram empilhados
241 ; Se ainda nao terminou
242     jmp ida           ; Retorna ao inicio do laco
243 sai:
244     mov edi, 0        ; Indice do buffer da soma
245
246     mov [qde3], esi   ; Salva a quantidade
247                       ; Vai ser usada como tamanho da
248                       ; string ao exibir a soma
249 volta:
250     pop rax           ; Recuperando os restos
251                       ; na ordem inversa
252     add al, 48        ; Formatando caracter atual
253     mov [numf + edi], al ; Colocando no buffer
254     inc edi           ; Aponta para a proxima posicao
255
256     dec esi           ; Desconta o ESI
257     jnz volta         ; Ateh zerar
258     ret
259 ;-----

```

Código “segundario.asm”:

[UUUUUL](#)

Online Assembler - NASM Compiler IDE

```
1 global print_str, pula_linha, read_str1, read_str2
2
3 extern linha, n1_max, n1, n2_max, n2
4
5 section .text
6 ; ----- AREA DOS PROCEDIMENTOS -----
7 print_str:
8 ; Exibe uma string no monitor
9 ; Chamada: EDX = tamanho; ECX = ponteiro
10 mov ebx, 1 ; Descritor (monitor = 1)
11 mov eax, 4 ; Servico (print)
12 int 0x80 ; Executa (exibe)
13 ret ; Devolve ao chamador
14 ;-----
15
16 pula_linha:
17 ; Salta uma linha na tela
18 mov edx, 1 ; Tamanho da string
19 mov ecx, linha ; Ponteiro da string
20 call print_str
21 ret
22
23 ;-----
24
25 read_str1:
26 ; Le uma string do teclado
27 ; Chamada: nenhum (universal)
28 mov edx, n1_max ; Limita a quantidade aceita
29 mov ecx, n1 ; Destino da str lida
30 mov ebx, 0 ; Descritor (teclado = 0)
31 mov eax, 3 ; Servico (read)
32 int 0x80 ; Executa a leitura
33 ret
34 ; No final EAX recebe o tamanho real do numero mais o <Enter>
35 ;-----
36
37
38 read_str2:
39 ; Le uma string do teclado
40 ; Chamada: nenhum (universal)
41 mov edx, n2_max ; Limita a quantidade aceita
42 mov ecx, n2 ; Destino da str lida
43 mov ebx, 0 ; Descritor (teclado = 0)
44 mov eax, 3 ; Servico (read)
45 int 0x80 ; Executa a leitura
46 ret
47 ; No final EAX recebe o tamanho real do numero mais o <Enter>
48 ;-----
```

Resultados:

```
(alexssander@ DESKTOP-BIGFMFH) ~/TrabalhoAsm
$ ./Principal
Este programa recebe dois numeros, realiza a soma deles e converte para um valor em registrador

Digite o primeiro numero: 12345A
Digite o primeiro numero: a11111
Digite o primeiro numero: 1234567
Digite o primeiro numero: 123456
Digite o segundo numero: aaaa
Digite o segundo numero: 1234567
Digite o segundo numero: 123450
O valor do reg eh: 246906
(alexssander@ DESKTOP-BIGFMFH) ~/TrabalhoAsm
$
```

Comandos para execução:

```
nasm -f elf64 -o principal.o principal.asm
```

```
nasm -f elf64 -o secundario.o secundario.o
```

```
ld principal.o secundario.o -o principal
```

Arquivos em anexo.