MICROPROCESSADORES E MICROCONTROLADORES

PROJETO 01

201840601017 – IAGO COSTA DAS FLORES

201740601024 – JULIANA BATISTA DA SILVA

201740601025 – LEYRISVAN DA COSTA NASCIMENTO

Marabá – PA

2021

UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ
INSTITUTO DE GEOCIÊNCIAS E ENGENHARIAS
FACULDADE DE COMPUTAÇÃO E ENGENHARIA ELÉTRICA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

IAGO COSTA DAS FLORES
JULIANA BATISTA DA SILVA
LEYRISVAN DA COSTA NASCIMENTO

EXPERIMENTOS 05

Relatório referente ao Projeto 01. Este relatório é um critério de atividade avaliativa da disciplina MICROPROCESSADORES E MICROCONTROLADORES.
Ministrada pelo professor: Dr. Elton Rafael Alves

Marabá – PA

2021

Sumário

01. Objetivo

1. Desenvolva uma calculadora em Assembly, com a seguintes condições:
- ✓ Operações de apenas 1 digito.
- ✓ Operação realizadas: SOMA, SUBTRAÇÃO, MULTIPLICAÇÃO E DIVISÃO.
- ✓ A calculadora deve apresentar valores negativos.
- ✓ O usuário deve digitar os valores e o programa exibir no final o resultado de acordo com a operação utilizada.

2. Desenvolver um programa de computador que leia um valor numérico entre 1 e 12 e apresente por extenso o nome do mês correspondente ao valor entrado. Caso sejam fornecidos valores menores que 1 e maiores que 12, o programa deve apresentar a mensagem" Valor invalido".

02. Atividade 01

edit: C:\Users\LEYRISVAN\Desktop\atividade01.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
001 TITLE CALCULADORA
002
003 #MAKE_EXE# ; tipo de arquivo para ser gerado
004
005 DADOS SEGMENT 'DATA' ; define segmento de dados
006 m1 DB 0Dh, 0Ah, 'Valor 1: ', 24h
007 m2 DB 0Dh, 0Ah, 'Valor 2: ', 24h
008 m3 DB 0Dh, 0Ah, 'Result: ', 24h
009 msg4 DB 0Dh, 0Ah, 'Valor invalido', 24h
010
011 escolha_1 DB 0Dh, 0Ah, 'Escolha o numero:', 24h
012 escolha_2 DB 0Dh, 0Ah, '1 - multiplicacao', 24h
013 escolha_3 DB 0Dh, 0Ah, '2 - divisao', 24h
014 escolha_4 DB 0Dh, 0Ah, '3 - soma', 24h
015 escolha_5 DB 0Dh, 0Ah, '4 - subtracao', 24h
016 escolha_6 DB 0Dh, 0Ah, '5 - finalizar', 24h
017 quebra_linha DB 0Dh, 0Ah, 24h
018
019 DADOS ENDS
020
021 PILHA SEGMENT STACK 'STACK' ; define segmento de pilha
022 DW 0100h DUP(?) ; define tamanho da pilha
023 PILHA ENDS
024
025 CODIGO SEGMENT 'CODE' ; define segmento de codigo
026 ASSUME CS:CODIGO, DS:DADOS, SS:PILHA ; declara cada segmento aos
027 ;registradores
028
029 fim MACRO ; define macro de finalizacao
030 MOV AH, 4ch ;encerramento do programa
031 INT 21h ;Controle do SO
032 ENDM ; finaliza macro fim
033
034 msg MACRO ; define macro para imprimir mensagem na tela
035 MOV AH, 09h
036 INT 21h ; apresentacao da mensagem
037 ENDM ; finaliza macro msg
```

```
038
039 INICIO_SCRIPT PROC FAR ; inicia procedimento INICIO do tipo FAR
040 MOV AX, DADOS ; acesso do segmento de codigo ao segmento de dados
041 MOV DS, AX ; movendo AX para DS
042 MOV ES, AX ; movendo AX para ES
043
044 escolha_opcao:
045 MOV DX, OFFSET escolha_1
046 MSG ; chama macro que imprime caractere na tela
047 MOV DX, OFFSET escolha_2
048 MSG ; chama macro que imprime caractere na tela
049 MOV DX, OFFSET escolha_3
050 MSG ; chama macro que imprime caractere na tela
051 MOV DX, OFFSET escolha_4
052 MSG ; chama macro que imprime caractere na tela
053 MOV DX, OFFSET escolha_5
054 MSG ; chama macro que imprime caractere na tela
055 MOV DX, OFFSET escolha_6
056 MSG ; chama macro que imprime caractere na tela
057 MOV DX, OFFSET quebra_linha
058 MSG ; chama macro que imprime caractere na tela
059 CALL input ; entrada de valor
060 CMP AL, 31h ; se AL igual a 1 decimal faz
061 JE multiplicacao ; se igual chama rotina
062 CMP AL, 32h ; se AL igual a 2 decimal faz
063 JE divisao ; se igual chama rotina
064 CMP AL, 33h ; se AL igual a 3 decimal faz
065 JE soma ; se igual chama rotina
066 CMP AL, 34h ; se AL igual a 4 decimal faz
067 JE subtracao ; se igual chama rotina
068 CMP AL, 35h ; se AL igual a 5 decimal faz
069 JE finaliza ; se igual chama rotina
```

```
070
071 divisao:
072 SUB AL, 30h ; subtrai o valor zero decimal do caractere
073 MOV DX, OFFSET m1
074 MSG ; chama macro que imprime caractere na tela
075 CALL input ; chama rotina input para receber caractere
076 SUB AL, 30h ; subtrai o valor zero decimal do caractere
077 MOV BH, AL ; movimenta primeiro valor de AL para BH
078 MOV DX, OFFSET m2
079 MSG ; chama macro que imprime caractere na tela
080 CALL input ; chama rotina input para receber caractere
081 SUB AL, 30h ; subtrai o valor zero decimal do caractere
082 MOV BL, AL ; move segundo valor para BL
083 MOV AL, BH ; retorna primeiro valor para AL
084 MOV AH, 0d ; zera AH para efetuar a divisao
085 DIV BL ; efetua a divisao
086 MOV BL, AL ; guarda resultado da divisao em BL
087 MOV DX, OFFSET m3
088 MSG ; chama macro que imprime caractere na tela
089 JMP resultado_divisao ; jmp para imprimir resultado na tela
090
091 multiplicacao:
092 MOV DX, OFFSET m1
093 MSG ; chama macro que imprime caractere na tela
094 CALL input ; chama rotina input para receber caractere
095 SUB AL, 30h ; subtrai o valor zero decimal do caractere
096 MOV BL, AL ; movimenta AL para BL
097 MOV DX, OFFSET m2
098 MSG ; chama macro que imprime caractere na tela
099 CALL input ; chama rotina input para receber caractere
100 SUB AL, 30h ; subtrai o valor zero decimal do caractere
101 MUL BL ; efetua a multiplicacao
102 MOV BX, AX ; resultado movido para BX
103 MOV DX, OFFSET m3
104 MSG ; chama macro que imprime caractere na tela
105 JMP resultado_multiplicacao ; jmp para imprimir resultado na tela
106
```

```
subtracao: ; rotina para subtracao de dois numeros
SUB AL, 30h ; subtrai o valor zero decimal do caractere
MOV DX, OFFSET m1
MSG ; chama macro que imprime caractere na tela
CALL input ; chama rotina input para receber caractere
MOV BH, AL ; guardando primeiro valor em BH
MOV DX, OFFSET m2
MSG ; chama macro que imprime caractere na tela
CALL input ; chama rotina input para receber caractere
MOV BL, AL ; guardando primeiro valor em BL
MOV DX, OFFSET m3
MSG ; chama macro que imprime caractere na tela
SUB BH, BL ; subtrai BH-BL para BH
JGE valor_positivo ; se BH>=BL pula para rotina valor_positivo
JL valor_negativo  ; se BH<BL pula para rotina valor_negativo

soma: ;rotina para soma de dois numeros
SUB AL, 30h ; subtrai o valor zero decimal do caractere
MOV DX, OFFSET m1
MSG ; chama macro que imprime caractere na tela
CALL input ; chama rotina input para receber caractere
MOV BH, AL ; guardando primeiro valor em BH
MOV DX, OFFSET m2
MSG ; chama macro que imprime caractere na tela
CALL input ; chama rotina input para receber caractere
MOV BL, AL ; guardando segundo valor em BL
MOV DX, OFFSET m3
MSG ; chama macro que imprime caractere na tela
ADD BH, BL ; soma BH+BL para BH
SUB BH, 30h ; limpando buffer 0 decimal
SUB BH, 30h ; limpando buffer 0 decimal
JMP resultado ; chama rotina para resultado da subtracao ou soma

valor_positivo: ; rotina valor positivo
JMP resultado ; chama rotina para resultado da subtracao ou soma

valor_negativo:
NEG BH ; transforma o valor negativo (complemento de dois) em seu equivalente positivo
MOV AL, 2Dh ; movimenta o valor 2Dh para o AL (ASCII 2Dh = -)
MOV AH, 0Eh ; chama comando da interrupcao
INT 10h ; mostra na tela o caractere negativo
JMP resultado ; chama rotina para resultado da subtracao

resultado_divisao: ; resultado funciona para divisao
MOV AL, BL ; movimenta resultado de BL para AL
ADD AL, 30h ; adiciona o 0 decimal em ascII ao caractere
MOV AH, 0Eh ; comando para interrupcao
INT 10h ; mostra caractere na tela
JMP escolha_opcao ; volta para selecao de opcao

resultado_multiplicacao: ; resultado_2 funciona para multiplicacao
MOV AX, BX ; movimenta resultado de BX para AX
ADD AX, 30h ; adiciona o 0 decimal em ascII ao caractere
MOV AH, 0Eh ; comando para interrupcao
INT 10h ; mostra caractere na tela
JMP escolha_opcao ; volta para selecao de opcao

resultado: ; resultado que funciona para soma e subtracao
MOV AL, BH ; movimenta resultado de BH para AL
ADD AL, 30h ; adiciona o 0 decimal em ascII ao caractere
MOV AH, 0Eh ; comando para interrupcao
INT 10h ; mostra caractere na tela
JMP escolha_opcao ; volta para selecao de opcao

finaliza: ;finaliza o script geral
FIM

INICIO_SCRIPT ENDP ; finaliza procedimento inicio_script

input PROC NEAR ; inicio procedimento input
MOV AH, 01h ; entrada do caractere pelo teclado
INT 21h ; chama interrupcao para receber caractere
CMP AL, 30h ; compara o valor recebido ao decimal em ascII
JL erro ; salta se caracteres recebido menor que zero --> 30h
CMP AL, 40h ; compara valor recebido ao decimal em ascII
JGE erro ; salta se caractere recebido maior ou igual a 40h
JMP fim_validacao ; salta para rotina fim_validacao

erro: ; rotina erro
MOV DX, OFFSET msg4 ; obtem endereco da variavel msg4
MSG ; chama macro para imprimir mensagem
FIM ; finaliza programa com macro

fim_validacao: ; rotina fiM_validacao
RET ; retorna da rotina
input ENDP ; fim do procedimento input

CODIGO ENDS ; finaliza o segmento de codigo

END INICIO_SCRIPT
```

Figura 01:código fonte do Atividade01

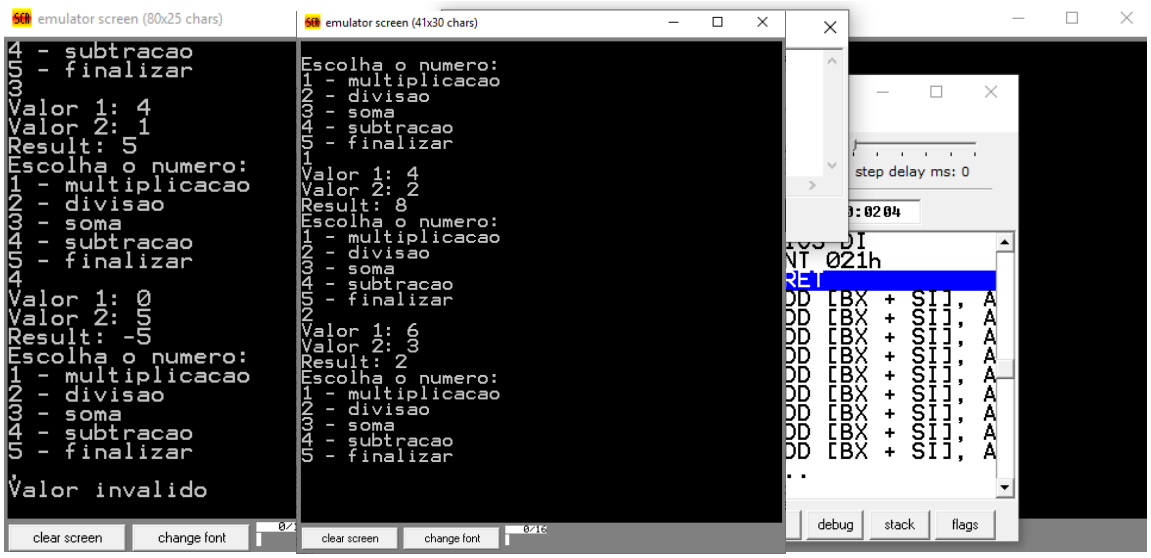

Figura 02: execução código fonte da Atividade01

Resultado: Para desenvolver o código foi utilizado a experiencia de todas as atividades e experimentos estudados. As instruções serviram para o foco de poder abstrair a ideia de como fazer todas as operações que obedeça aos requisitos da atividade como estar descrito nos comentários no código fonte da **Figura 01**. O resultado foi obtido com êxito atendendo as operações de multiplicação, divisão, adição e subtração escolhendo a operação desejada de acordo como é demonstrado na **Figura 02**. E assim também uma condição foi colocada pra quando um valor não valido for digitado e também uma opção para finalizar e sair das operações.

03. Atividade 02

edit: C:\Users\LEYRISVAN\Desktop\escolha_mes.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
;*******************************
;* Programa: ESCOLHA_MES.ASM *;
;*******************************

INCLUDE 'emu8086.inc' ; chamada da biblioteca emu8086.inc

org 100h

.DATA
valor DW 0
mensagem DB 0Dh, 0Ah,'DIGITE UM VALOR DO MES (1-12):',0 ; valor da variavel e 0
mes1 DB 0Dh, 0Ah,'MES 1 E JANEIRO',0
mes2 DB 0Dh, 0Ah,'MES 2 E FEVEREIRO',0
mes3 DB 0Dh, 0Ah,'MES 3 E MARCO',0
mes4 DB 0Dh, 0Ah,'MES 4 E ABRIL',0
mes5 DB 0Dh, 0Ah,'MES 5 E MAIO',0
mes6 DB 0Dh, 0Ah,'MES 6 E JUNHO',0
mes7 DB 0Dh, 0Ah,'MES 7 E JULHO',0
mes8 DB 0Dh, 0Ah,'MES 8 E AGOSTO',0
mes9 DB 0Dh, 0Ah,'MES 9 E SETEMBRO',0
mes10 DB 0Dh, 0Ah,'MES 10 E OUTUBRO',0
mes11 DB 0Dh, 0Ah,'MES 11 E NOVEMBRO',0
mes12 DB 0Dh, 0Ah,'MES 12 E DEZEMBRO',0
invalido DB 0Dh, 0Ah,'VALOR INVALIDO',0


.CODE
LEA SI,mensagem ; aponta para SI(acesso ao conteudo da string)
                ;o endereco de memoria da variavelmensagem
CALL PRINT_STRING ; acessa o conteudo do registrador SI
CALL SCAN_NUM ; efetua a leitura do teclado
MOV valor, CX ; armazena o valor do teclado em CX
;PUTC 13d
;PUTC 10d

CMP valor,1D ; compara o conteudo do valor com 1
JE case1    ;salta pra case1
```

```
039 CMP valor,2D ; compara o conteudo do valor com 2
040 JE case2    ;salta pra case2
041
042 CMP valor,3D ; compara o conteudo do valor com 3
043 JE case3
044
045 CMP valor,4D ; compara o conteudo do valor com 4
046 JE case4    ;salta pra case4
047
048 CMP valor, 5D; compara o conteudo do valor com 5
049 JE case5
050
051 CMP valor, 6D; compara o conteudo do valor com 6
052 JE case6
053
054 CMP valor, 7D; compara o conteudo do valor com 7
055 JE case7
056
057 CMP valor, 8D; compara o conteudo do valor com 8
058 JE case8
059
060 CMP valor, 9D; compara o conteudo do valor com 9
061 JE case9
062
063 CMP valor, 10D; compara o conteudo do valor com 10
064 JE case10
065
066 CMP valor, 11D; compara o conteudo do valor com 11
067 JE case11
068
069 CMP valor, 12D; compara o conteudo do valor com 12
070 JE case12
071
072  JMP default
073
074   case1:
075   LEA SI, mes1      ;apresenta a mensagem de mes1
076   CALL PRINT_STRING ;chamada da subrotina pra imprimir a string
077   JMP saida         ;salto para saida
078
079   case2:
080   LEA SI, mes2      ;apresenta a mensagem de mes
081   CALL PRINT_STRING;chamada da subrotina pra imprimir a string
082   JMP saida         ;salto para saida
083
084   case3:
085   LEA SI, mes3
086   CALL PRINT_STRING
087   JMP saida
088
089   case4:
090   LEA SI, mes4
091   CALL PRINT_STRING
092   JMP saida
093
094     case5:
095   LEA SI, mes5
096   CALL PRINT_STRING
097   JMP saida
098
099
100   case6:
101   LEA SI, mes6
102   CALL PRINT_STRING
103   JMP saida
104
105   case7:
106   LEA SI, mes7
107   CALL PRINT_STRING
109
110   case8:
111   LEA SI, mes8
112   CALL PRINT_STRING
113   JMP saida
114
115   case9:
116   LEA SI, mes9
117   CALL PRINT_STRING
118   JMP saida
119
120   case10:
121   LEA SI, mes10
122   CALL PRINT_STRING
123   JMP saida
124
125   case11:
126   LEA SI, mes11
127   CALL PRINT_STRING
128   JMP saida
129
130   case12:
131   LEA SI, mes12
132   CALL PRINT_STRING
133   JMP saida
134
135    default:
136    LEA SI,invalido
137    CALL PRINT_STRING
138    JMP saida
```

```
139
140
141 DEFINE_PRINT_STRING ; biblioteca emu8086.inc
142 DEFINE_SCAN_NUM ; biblioteca emu8086.inc
143
144  saida:
145     mov ax, 4c00h ; exit to operating system.
146     int 21h
147 END
148
149 end start ; set entry point and stop the assembler.
```

Figura 03:código fonte da Atividade02

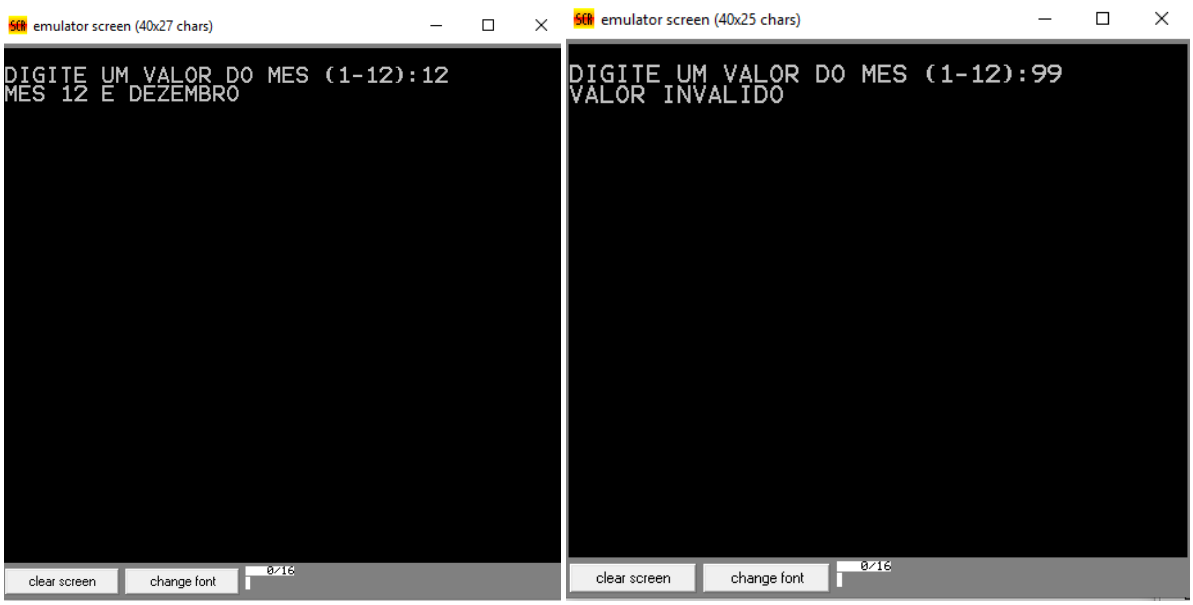

Figura 04: execução do código fonte da Atividade02

Resultado: Com base do experimento 19 foi possível idealizar como incrementar valores fazer comparações para que assim as condições sejam atendidas. No código fonte como mostra a **Figura 03**, podemos observar a simplicidade do código bem como as variáveis inicializado com zero para assim termos precisão do resultado. O INCLUDE 'emu8086.inc' faz chamada da biblioteca emu8086.inc para escrever a string da variável ao qual for apontada e logo após é incrementada pelo registrador SI e feito a chamada da sub-rotina CALL PRINT_STRING que é atendida pelo DEFINE_PRINT_STRING terminado a impressão do conteúdo segue a próxima instrução que é o CALL SCAN_NUM que é atendida pelo DEFINE_ SCAN_NUM onde faz a leitura do numero pelo teclado, em seguido o numero é colocado num contador CX, que em seguida irá fazer a comparação e a instrução que a seguir , dando-se o salto para o caso que lhe atenda e assim a mensagem é apresentada ao registrador SI e a mensagem é impressa na tela pela chamada de sub-rotina DEFINE_PRINT_STRING que é atendida DEFINE_PRINT_STRING ; biblioteca emu8086.inc. como mostra a **Figura 04** e o programa é finalizado pelo salto de JMP saída.

05. Referências

https://seguranca-informatica.pt/introducao-ao-assembly/#.YNIeCOhKjIU
Acesso em 08/07/2021.

http://www.facom.ufu.br/~gustavo/OC1/Apresentacoes/Assembly.pdf

Acesso em 08/07/2021

http://www.inf.furb.br/~maw/arquitetura/aula16.pdf Acesso em 08/07/2021