

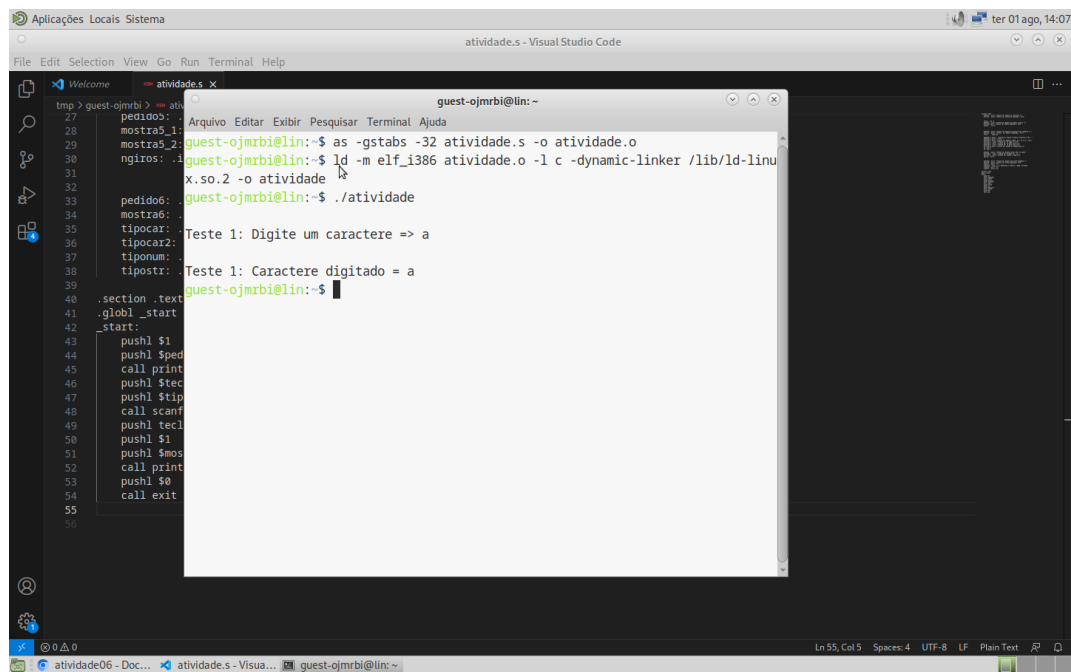
## Prática 06 - Partes 1, 2 e 3

RA117306 Felipe Gabriel Comin Scheffel

RA117741 Douglas Kenji Sakakibara

1)

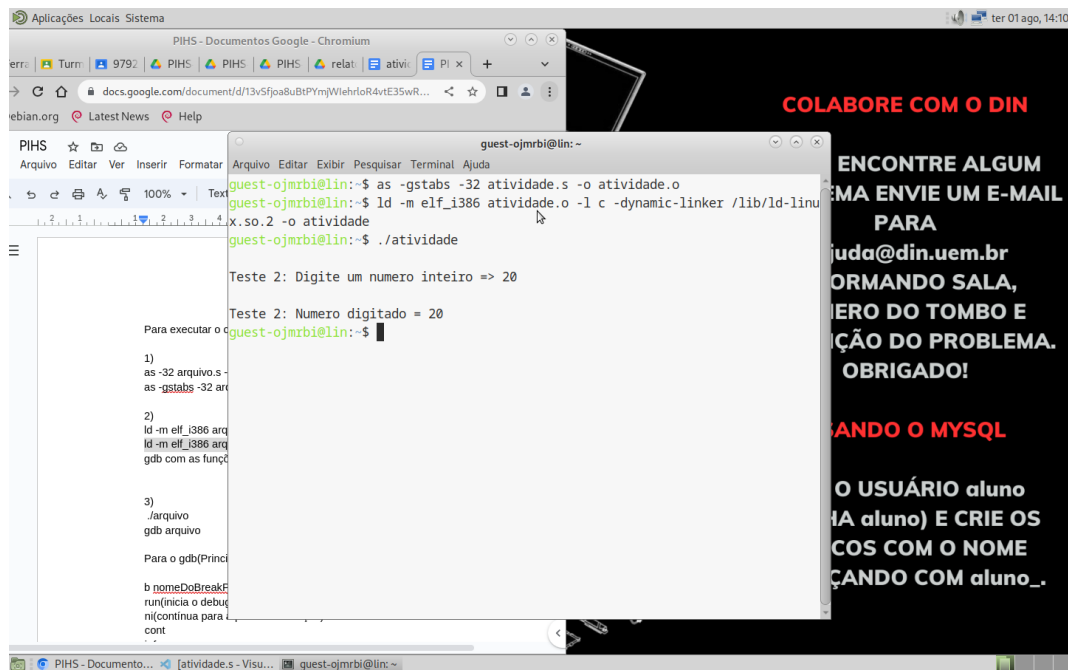
### Teste 01



```
Aplicações Locais Sistema
atividade.s - Visual Studio Code
File Edit Selection View Go Run Terminal Help

guest-ojmrbi@lin: ~
27 tmp > guest-ojmrbi > as -o atividade.o atividade.s
28 mostra5_1:
29 guest-ojmrbi@lin:~$ as -gstabs -32 atividade.s -o atividade.o
30 mostra5_2:
31 guest-ojmrbi@lin:~$ ld -m elf_i386 atividade.o -l c -dynamic-linker /lib/ld-linux
32 x.so.2 -o atividade
33 guest-ojmrbi@lin:~$ ./atividade
34 Teste 1: Digite um caractere => a
35 tipocar2:
36 tipocar2:
37 tiponum:
38 Teste 1: Caractere digitado = a
39 tipostr:
40 guest-ojmrbi@lin:~$
41
42 .section .text
43 .globl _start
44 _start:
45     pushl $1
46     pushl $ped
47     call print
48     pushl $tec
49     pushl $tip
50     call scanf
51     pushl $tecl
52     pushl $1
53     pushl $mos
54     call print
55     pushl $0
56     call exit
```

### Teste 02



2)

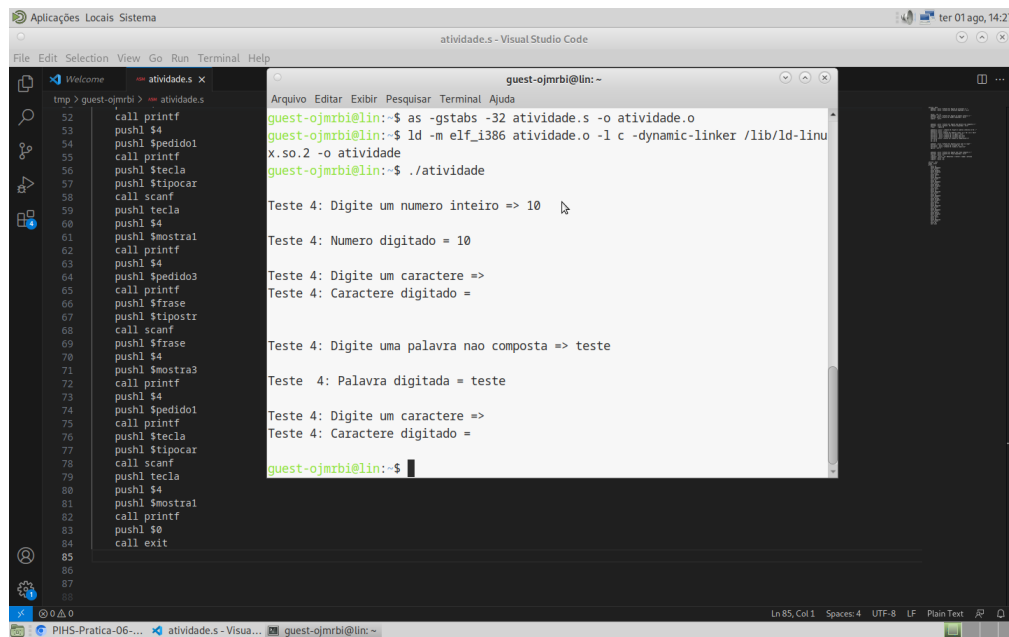
## Teste 03



Pois o endereço empilhado para armazenar a string deve ser o endereço inicial de uma região da memória. Diferente dos testes 1 e 2 que trabalham com valores do tipo int e caracter.

3)

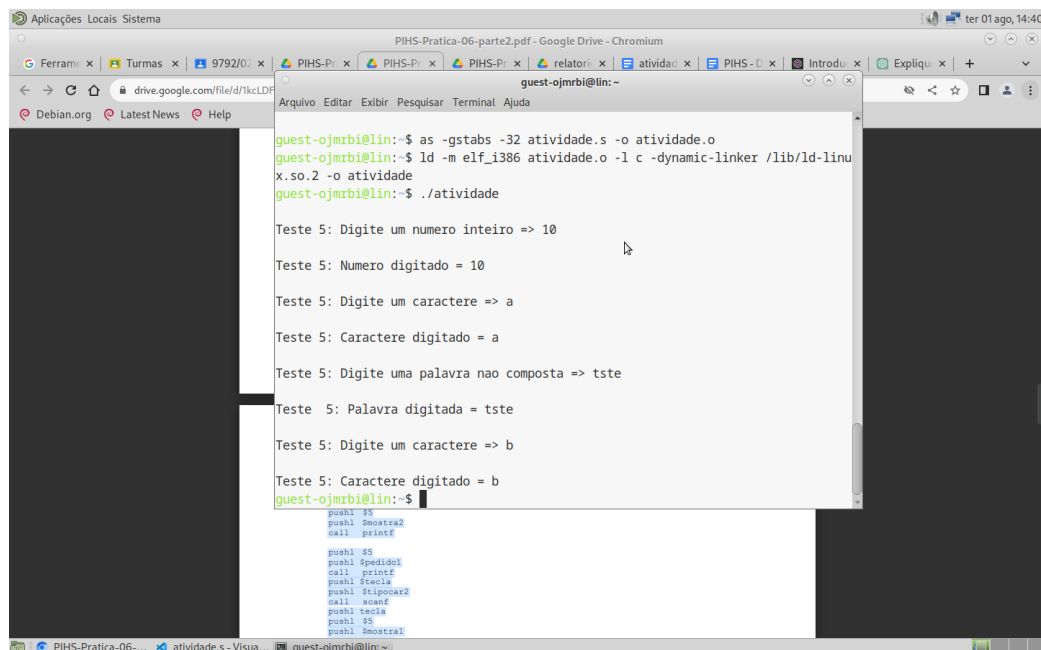
## Teste 04



```
guest-ojmrbi@lin:~  
guest-ojmrbi@lin:~$ as -gstabs -32 atividade.s -o atividade.o  
guest-ojmrbi@lin:~$ ld -m elf_i386 atividade.o -l c -dynamic-linker /lib/ld-linux  
X.so.2 -o atividade  
guest-ojmrbi@lin:~$ ./atividade  
  
Teste 4: Digite um numero inteiro => 10  
  
Teste 4: Numero digitado = 10  
  
Teste 4: Digite um caractere =>  
Teste 4: Caractere digitado =  
  
Teste 4: Digite uma palavra nao composta => teste  
  
Teste 4: Palavra digitada = teste  
  
Teste 4: Digite um caractere =>  
Teste 4: Caractere digitado =  
  
guest-ojmrbi@lin:~$
```

A função scanf para a leitura ao encontrar um espaço em branco ou nova linha, no segundo e no último caso, que são pedidos um caractere de entrada, ele lerá o caractere de nova linha deixado no buffer de entrada pelo Enter pressionado anteriormente. Isso fará com que o segundo caractere seja '\n'.

## Teste 05



```
guest-ojmrbi@lin:~$ as -gstabs -32 atividade.s -o atividade.o  
guest-ojmrbi@lin:~$ ld -m elf_i386 atividade.o -l c -dynamic-linker /lib/ld-linux  
X.so.2 -o atividade  
guest-ojmrbi@lin:~$ ./atividade  
  
Teste 5: Digite um numero inteiro => 10  
  
Teste 5: Numero digitado = 10  
  
Teste 5: Digite um caractere => a  
Teste 5: Caractere digitado = a  
  
Teste 5: Digite uma palavra nao composta => tste  
  
Teste 5: Palavra digitada = tste  
  
Teste 5: Digite um caractere => b  
Teste 5: Caractere digitado = b  
guest-ojmrbi@lin:~$
```

## 4) Teste 06

```
guest-ojmrbi@lin:~$ as -gstabs -32 atividade.s -o atividade.o
guest-ojmrbi@lin:~$ ld -m elf_i386 atividade.o -l c -dynamic-linker /lib/ld-linux.so.2 -o atividade
guest-ojmrbi@lin:~$ ./atividade

Teste 6: Digite 2 numeros inteiros:
N1 = 5
N2 = 10

Teste 6: Numeros lidos: n1 = 5 e n2 = 10

Teste 6: n2 maior que n1

Teste 6: Acabou as comparacoes!
guest-ojmrbi@lin:~$
```

```
7) Teste 6: Acrescente as instruções de comparação e salto condicional.

pushl $6
pushl $pedido4_1
call printf
pushl $n1
pushl $iponum
call scanf

pushl $6
pushl $pedido4_2
call printf
pushl $n2
pushl $iponum
call scanf

pushl n2
pushl n1
pushl $6
pushl $mostra4_1
call printf

movl n2, %eax # %eax e %ecx não são al
je %eax # aqui também serve
jl n2menorn1
jmp n1menorn2

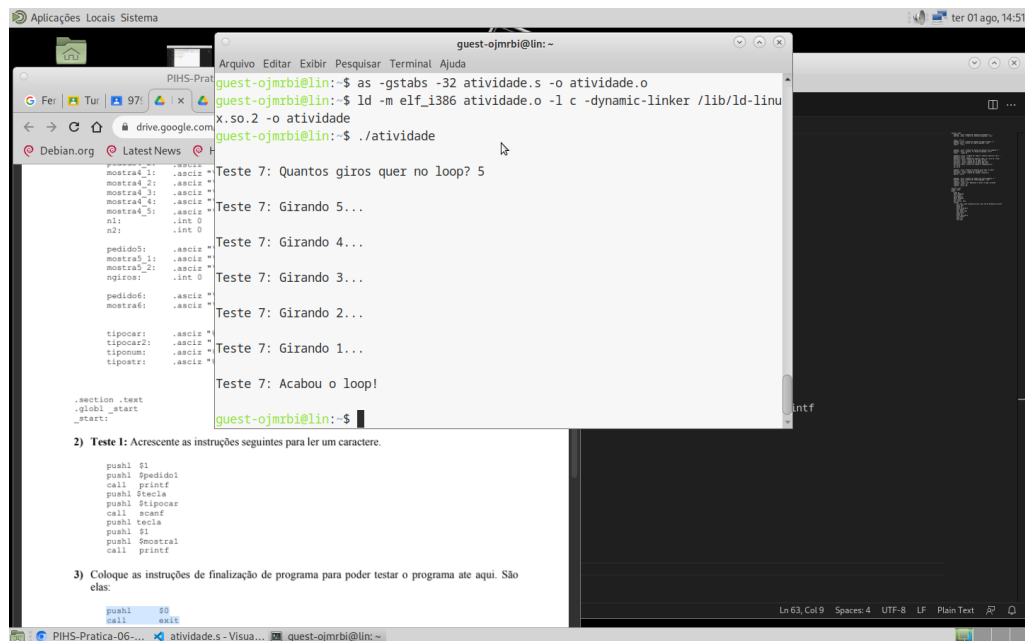
saiguais:
pushl $6
pushl $mostra4_5
call printf
pushl $0
call exit
```

A instrução `cmpl` compara os dois operandos de 32 bits e o resultado lógico dessa comparação é colocado no registrador de flag da arquitetura (EFLAGS). As instruções `je`, `jl` e `jmp`, são utilizadas para definir qual caminho é direcionado a partir do resultado lógico do `cmpl`.

A instrução não pode ser substituída, pois na instrução `cmpl n1, %eax` o valor de `n1` está sendo comparado com o valor contido em `eax`, já na instrução `cmpl n1, n2` está sendo comparado com o valor de `n2`.

5)

Teste 07



```
guest-ojmrbi@lin:~$ as -gstabs -32 atividade.s -o atividade.o
guest-ojmrbi@lin:~$ ld -m elf_i386 atividade.o -l c -dynamic-linker /lib/ld-linux
x.so.2 -o atividade
guest-ojmrbi@lin:~$ ./atividade
Teste 7: Quantos giros quer no loop? 5
Teste 7: Girando 5...
Teste 7: Girando 4...
Teste 7: Girando 3...
Teste 7: Girando 2...
Teste 7: Girando 1...
Teste 7: Acabou o loop!
guest-ojmrbi@lin:~$
```

2) **Teste 1:** Acrescente as instruções seguintes para ler um caractere.

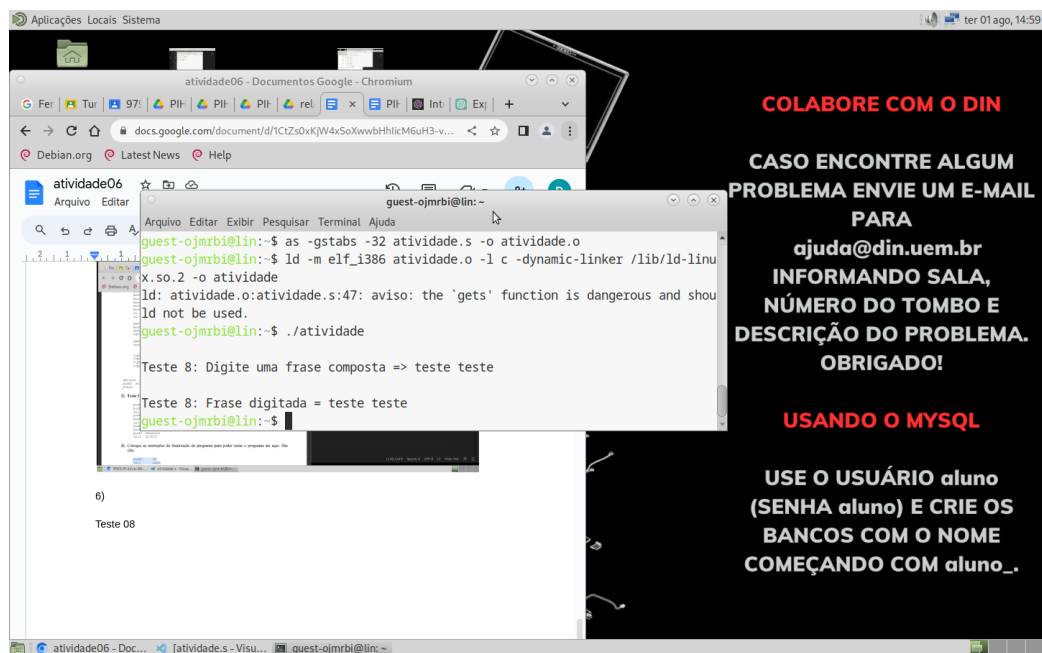
```
pushl $1
pushl $pedido1
call printf
pushl $tecla
pushl $tipocar
call scanf
pushl $tecla
pushl $1
pushl $mostra1
call printf
```

3) Coloque as instruções de finalização de programa para poder testar o programa ate aqui. São elas:

```
pushl $0
call exit
```

6)

## Teste 08



**COLABORE COM O DIN**

**CASO ENCONTRE ALGUM PROBLEMA ENVIE UM E-MAIL PARA ajuda@din.uem.br INFORMANDO SALA, NÚMERO DO TOMBO E DESCRIÇÃO DO PROBLEMA. OBRIGADO!**

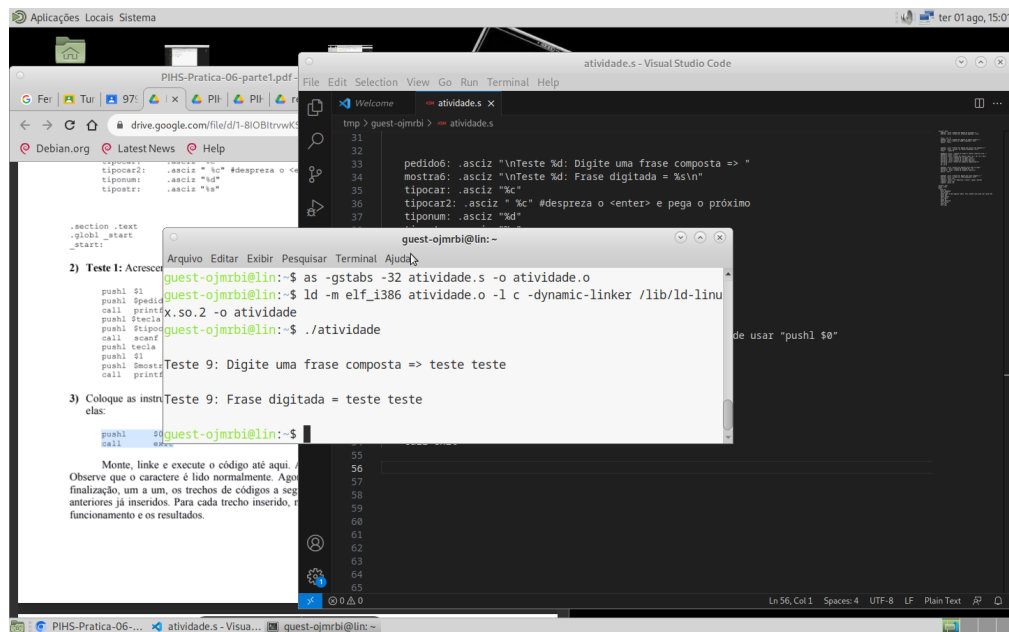
**USANDO O MYSQL**

**USE O USUÁRIO aluno (SENHA aluno) E CRIE OS BANCOS COM O NOME COMEÇANDO COM aluno\_.**

```
guest-ojmrbi@lin:~$ as -gstabs -32 atividade.s -o atividade.o
guest-ojmrbi@lin:~$ ld -m elf_i386 atividade.o -l c -dynamic-linker /lib/ld-linux
x.so.2 -o atividade
ld: atividade.o:atividade.s:47: aviso: the 'gets' function is dangerous and should not be used.
guest-ojmrbi@lin:~$ ./atividade
Teste 8: Digite uma frase composta => teste teste
Teste 8: Frase digitada = teste teste
guest-ojmrbi@lin:~$
```

7)

## Teste 09



Em comparação com a função `gets`, o `fgets` permite que seja passado como parâmetro o número máximo de caracteres da string que vai ser armazenada, prevenindo um possível “estouro” do tamanho do buffer alocado.