⚠ al é o registo de 8 bits do ~~edx~~ eax

Nº A92846	**Nome:** Carlos Miguel Passos Ferreira	**Turma:** Turma 1

Resolução dos exercícios (deve ser redigido manualmente)

1.

a) Teste do programa

Escreva aqui o que apareceu no monitor desde que começou a execução do código, incluindo os caracteres que tiver introduzido e o resultado da execução do código.

```
[In]  ./m-contaN   ← (execução do programa)
[Out] Introduza a cadeia de caracteres -->
[In]  1239aaswe67899
[out] Qual a posição inicial na cadeia de caracteres -->
[In]  2
[Out] O somatório dos digitos na cadeia e --> 51
```

⊛ Neste ponto, é feita a soma do dígito e é acumulado no somatório que corresponde ao resultado

b) Código desmontado da função `contaN` (com o `gdb`, em *assembly*)

Escreva aqui os comandos que usou para obter o código desmontado da função.

```
$ gdb m-contaN
$ disass contaN
```

c) Anotação do código *assembly* e resposta às questões colocadas no enunciado.

Mostre aqui o código desmontado pedido na alínea anterior, mas devidamente anotado. Não esquecer que anotação de código deve corresponder a uma explicação do que faz cada instrução numa perspetiva **(i)** de gestão da *stack frame* ou **(ii)** de código C da função.

```
0x080483c0  <contaN+0>  : push %ebp                  # começa a função
c1          <contaN+1>  : mov  %esp,%ebp             # cria um novo stack pointer (ebp = esp)
c3          <contaN+3>  : push %esi                  } salvaguarda dos registos %esi e %ebx
c4          <contaN+4>  : push %ebx                  }
c5          <contaN+5>  : mov  0x8(%ebp),%esi        # indica que a cadeia começa em %esi
c8          <contaN+8>  : mov  0xc(%ebp),%ecx        # ecx será a posição inicial da cadeia
cb          <contaN+11> : mov  (%ecx,%esi,1),%dl     # dl é o primeiro char que se vai ler
ce          <contaN+14> : xor  %ebx,%ebx             # inicia o resultado (apaga o resultado antigo do ebx)
d0          <contaN+16> : test %dl,%dl               # testa se dl = 0
d2          <contaN+18> : je   0x80483ec <contaN+44> # if dl = 0, estamos no fim da cadeia e vai para o fim (passa a ASCII)
d4          <contaN+20> : lea  0xffffffd0(%edx),%eax # al = dl - 30
d7          <contaN+23> : cmp  $0x9,%al              # compara al com o 9
d9          <contaN+25> : ja   0x80483e2 <contaN+34> # se al for > 9, salta porque não é dígito
db          <contaN+27> : movsbl %dl,%eax            # extensão do dl, que tinha 8 bits e passou para o eax com 32 bits
de          <contaN+30> : lea  0xffffffd0(%eax,%ebx,1),%ebx # ebx = eax + ebx - 30 ⊛
e2          <contaN+34> : inc  %ecx                  # avança na cadeia
e3          <contaN+35> : mov  (%ecx,%esi,1),%al     # coloca em al o próximo char a ser lido
e6          <contaN+38> : test %al,%al               # testa se al é 0 (se é "null" ou não em ASCII)
e8          <contaN+40> : mov  %al,%dl               # dl = al
ea          <contaN+42> : jne  0x80483d4 <contaN+20> # se al ≠ 0, então volta ao ciclo (volta ao somatório)
ec          <contaN+44> : mov  %ebx,%eax             # ebx é o resutado --> que será retornado
ee          <contaN+46> : pop  %ebx                  # termino da função
```

↓ Assinale no código em cima a resposta a cada uma das 6 questões colocadas no enunciado.
Se precisar de acrescentar alguns esclarecimentos adicionais, faça-o a seguir a este texto.

+ termino

```
ef <contaN+47> : pop  %esi
f0 <contaN+48> : leave
f1 <contaN+49> : ret
f2 <contaN+50> : nop
f3 <contaN+51> : nop
```

b) Execute de novo o mesmo programa através do `gdb`. Use os comandos disponíveis para examinar código, de forma a visualizar o código simbólico ("desmontado" ou *disassembled*) correspondente à função (e apenas este). **Registe** o código *assembly* obtido.

c) Desconfia-se que a estrutura da função que estava em `contaN.c` seja do tipo:

```
int i;
int result;
???
for ( ??? ; s[i]!= ??? ; ???)
   if (s[i] >= '0' && ??? )
      result += ??? ;
return result;
```

Anote cuidadosamente o código visualizado na alínea anterior tendo em consideração que o resultado da função é devolvido no registo `%eax`.

Identifique no código:

- os registos que são atribuídos às variáveis locais `result` (%ebx) e `i` (%ecx)
- os registos que são usados com os argumentos da função (que chama) %esi e %ecx ???
- a condição de teste do ciclo `for` s[i] != 0
- o modo como a variável `i` é atualizada i ++ ou i = i + 1
- o código decimal correspondentes aos dígitos representados em *ASCII* subtrair ao valor 0x30
- a expressão em C que atualiza o valor de `result` no ciclo result += s[i]

d) Com base no resultado das alíneas anteriores, **recupere** o ficheiro `contaN.c`.
(Para fazer depois da sessão laboratorial)

0xffffffd0
↓
está em complemento para 2
↓
resultado da conversão: -30