

Nº a98280 Nome: Miguel Ângelo Machado Turma: PL 5

Código otimizado em assembly (tal como está no ficheiro while-loop.s)

segunda a
alínea (b)
↓

- .text
- .p2align 2,,3
- globl while-loop
 - type while-loop, @function

while-loop:

[grande antigo
base pointer]

[atualiza o base-
pointer]

pushl %ebp
movl %esp, %ebp
movl 16(%ebp), %edx

~~pushl~~

testl %edx, %edx

pushl %ebp

movl 12(%ebp), %eax

movl 8(%ebp), %ebp →

Ananque

jle .L3 ~~41b59256n sua função~~

movl %edx, %edx

sall \$4, %edx

cmpl %edx, %eax → [compara y com 16*n]

jge .L3 → [Se y < 16*n sair do ciclo]

• .p2align 2,,3

• L6:

addl %edx, %ebx → $x = n$

imull %edx, %ebx → $y = n$

| Registo | Variável | Atribui- ção inicial |
|---------|----------|----------------------------|
| %ebx | x | 4 |
| %eax | y | 2 |
| %edx | n | 3 |
| %edx | n | $3 \rightarrow 16^*n$ |

Variável
temporária



(2)

* dec $\% \text{edx}$ $\rightarrow [n--]$
 subl \$16, $\% \text{eax}$
 testl $\% \text{edx}, \% \text{edx}$ $\rightarrow [\text{Testa se } n > 0]$
 jle .L3
 empl $\% \text{eax}, \% \text{eax}$ $\} \quad \text{NPF}$
 jl .L6 $\} \rightarrow$
 Tal como anteriormente (na pág. anterior),
 vê se $y < 16^n$ e se for,
 sair do ciclo

L3:

Término

movl $\% \text{ebx}, \% \text{eax}$
 popl $\% \text{ebx}$ $\} \quad$ quando o valor de x no
 leave endereço de y
 ret $\} \quad$ libera o espaço que o endereço de
 x ocupava previamente

Lfe1:

- size while-loop, - Lfe1 - while-loop
- Section .note:GNU-stack, "", @mbytes

→ Escreva aqui o código de um prog. simples (main) que usa a função while-loop

```

int #while-loop (int x, int y, int n);
int main()
{
  int x, y, n;
  x=4;
  y=2;
  n=3;
  while-loop(x,y,n);
  return 0;
}
  
```

(2)

→ Conteúdo dos registos que receberam os argumentos

08048310 <while-loop>

| | |
|----------|----------|
| 8048310: | 55 |
| 8048311: | 89 e5 |
| 8048313: | 8b 55 10 |
| 8048316: | 85 d2 |
| 8048318: | 53 |
| 8048319: | 8b 45 0e |
| 804831d: | 8b 50 08 |
| 804831f: | 78 1e |
| 8048321: | 89 d1 |
| 8048323: | e1 e1 04 |
| 8048326: | 39 e8 |
| 8048328: | 7d 13 |
| 804832a: | 89 f6 |
| 804832e: | 01 d3 |
| 804832e: | 0f a1 e2 |
| 8048331: | 4a |
| 8048332: | 83 e9 10 |
| 8048335: | 85 a2 |
| 8048337: | 72 04 |
| 8048339: | 39 08 |
| 804833b: | 7e ef |
| 804833d: | 89 d8 |
| 804833f: | 5b |
| 8048340: | e9 |
| 8048341: | e3 |
| 8048342: | 89 f6 |

| | |
|--------------------------------|--|
| pushl %ebp | |
| movl %esp,%ebp | |
| movl 0x10(%ebp),%edx | |
| testl %edx,%edx | |
| pushl %ebx | |
| movl 0xe(%ebp),%eax | |
| movl 0x8(%ebp),%ebx | |
| imul 804833d<while-loop+0x2015 | |
| movl %edx,%ecx | |
| shll \$0x4,%eax | |
| cmp %eax,%eax | |
| jge 804833d<while-loop+0x2015 | |
| movl %esi,%esi | |
| addl %edx,%ebx | |
| imul %edx,%eax | |
| dec %edx | |
| subl \$0x10,%eax | |
| testl %eax,%edx | |
| imul 804833d<while-loop+0x2015 | |
| cmp %eax,%eax | |
| jlt 804832e+0x2015 | |
| movl %ebx,%eax | |
| popl %ebx | |
| leave | |
| ret | |
| movl %esi,%esi | |

08048344 < main >;

| | | |
|----------|----------------|----------------------------|
| 8048344: | 55 | push %ebp |
| 8048345: | 89 e5 | mov %esp, %ebp |
| 8048347: | 83 ee 08 | sub \$0x8, %esp |
| 8048349: | 83 e4 f0 | and \$0xffffffff0, %esp |
| 804834d: | 50 | push %eax |
| 804834e: | 6a 03 | push \$0x3 |
| 8048350: | 6a 02 | push \$0x2 |
| 8048352: | 6a 04 | push \$0x4 |
| 8048354: | 18 b7 ff ff ff | call 8048310 < while loops |
| 8048359: | 31 e0 | xor %eax, %eax |
| 804835b: | ec | leave |
| 804835e: | c3 | ret |
| 804835d: | 90 | nop |
| 804835f: | 90 | nop |
| 804835f: | 90 | nop |

→ Endereços das instruções onde insino os breakpoints

→ 8048316

→ 804831f

junti 2 seguidos
804831e
804831f

Escrevendo 50 0
2º

→ 804832e

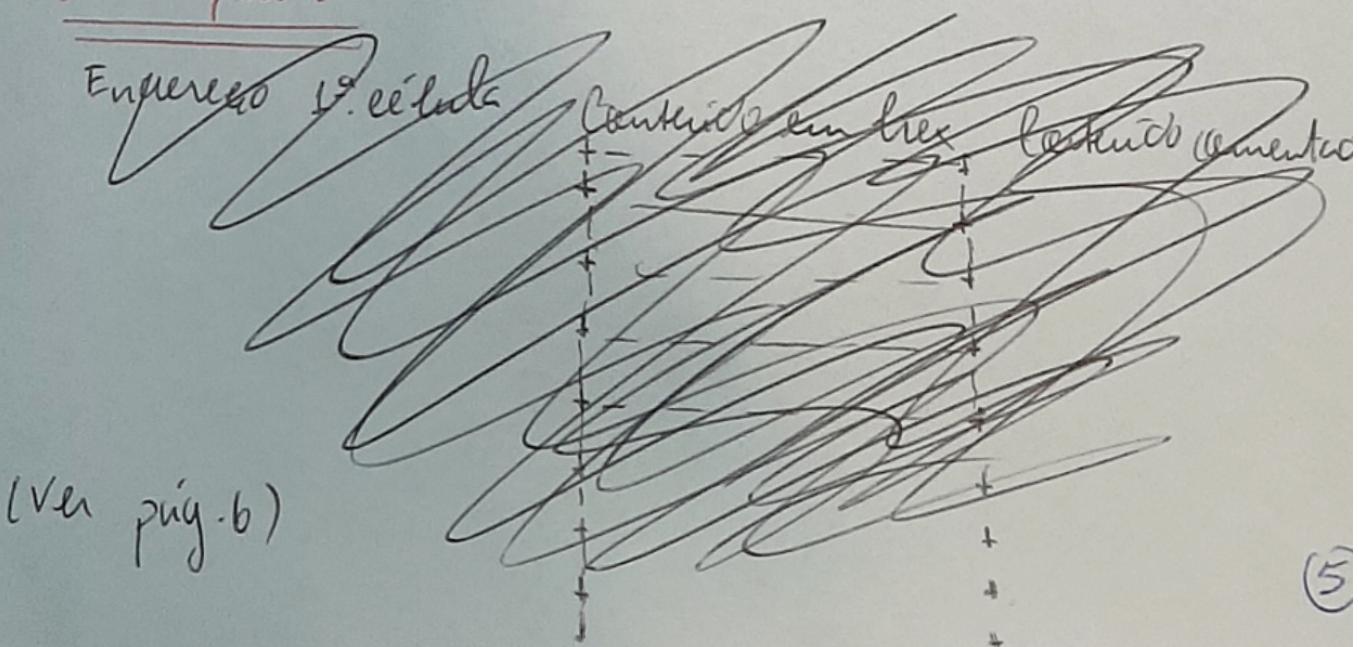
→ Estimation

| Registro | Variável | Break 1 | Break_ | Break_ | Break_ | Break_ |
|----------|----------|---------|--------|--------|--------|--------|
| %ebx | x | | 4 | 7 | 7 | 9 |
| %eax | y | 1 | 2 | 6 | 6 | 12 |
| %edx | n | 3 | 3 | 2 | 2 | 1 |

→ Após execução do código

| Registro | Variável | Break_1 | Break_2 | Break_3 | Break_4 | Break_5 |
|----------|----------|---------|---------|---------|---------|---------|
| %ebx | x | | 4 | 4 | 7 | 9 |
| %eax | y | 1 | 2 | 2 | 6 | 12 |
| %edx | n | 3 | 3 | 3 | 2 | 5 |
| %ecx | | | 48 | 32 | 16 | |
| | | | | | | |

→ stack frame



Endereço 1^º célula

0xbffffe394

Conteúdo em hex

| | | | |
|-----------|---------------|---|---------------------|
| + - - - - | 0x08048348 | + | SP |
| + - - - - | 0x000 00030 | + | 16*n |
| + - - - - | 0x000 00003 | + | n |
| + - - - - | 0x 000 00002 | + | Y |
| + - - - - | 0x 080 000 04 | + | Z |
| + - - - - | 0x08048325 | + | Endereço de retorno |
| + - - - - | 0xbffffe394 | + | |
| + - - - - | 0xbffffe398 | + | |
| + - - - - | 0xbffffe39e | + | |

Conteúdo comentado