

**Lista de Exercício E02**  
**2021**

**2016AV02Q01:**

Quanto e quais são os registradores visíveis de propósito geral das máquinas com arquitetura de 64 bits? Quais são as possíveis maneiras (com relação ao tamanho das palavras) de acessar os registradores classificados como Gerais<sup>1</sup>? É possível criar mais registradores de propósito geral, além dos citados na resposta, durante a escrita de um código em assembly?

nota(s) da questão:

<sup>1</sup> Durante uma aula da etapa 02 de LM, aluno R.B. pediu se poderia criar outro registrador, porque não gostava dos nomes atuais.

**2016AV02Q02:**

Considere que você implementou um novo editor de planilhas para o libre office no arquivo excellibre.asm<sup>1</sup> que contém um código escrito em assembly com sintaxe Intel. Considere também que este código foi escrito para uma máquina de arquitetura de 64 bits e o montador nasm está instalado nesta. O código faz uso somente de chamadas para as funções printf() e scanf() do C. Quais são os comandos necessários, em terminal linux, para montar e linkar (ligar) esta aplicação?

nota(s) da questão:

<sup>1</sup> Durante uma aula de LM, aluno J.V. pediu se tabela de um exercício poderiam ser criada no Excel do libreoffice.

**2016AV02Q03: (não aplicável - ignorar)**

Quanto e quais são os registradores da FPU (ponto Flutuante)? Quais são as instruções que transferem os números reais da memória para estes registradores e para memória destes registradores?

**2016AV02Q04: (questão adaptada para SSE e AVX, originalmente solicitava MMX)**

Quanto e quais são os registradores SSE2? Cite 2 exemplos de instruções aritméticas e 1 exemplo de instrução lógica que opere nestes registradores.

**2016AV02Q05:**

Considere que uma grande empresa de produção de aeromodelos contratou você, aluno de CC da Unioeste – Cascavel, para transcrever um código em C, utilizado no mais recente modelo de Yellowcóptero<sup>1</sup>, lançado especialmente para os cheaters<sup>2</sup> de pokemon<sup>2</sup>, e apresentado no Quadro 2016AV02Q05, para Assembly (Sintaxe Intel).

```
1. #include<stdio.h>
2. #define flap 20
3.
4. unsigned int dflap(unsigned int n, unsigned int exp){
5.     return n*exp;
6. }
7.
8. int main(){
9.     printf("rotação em: %d\n", dflap(flap,2));
10.    return 0;
11. }
```



Quadro 2016AV02Q05. Código para cálculo do rotor traseiro do Yellowcóptero.

**Observações:**

- O código pode ser transcrito para arquitetura x64 ou x86 e obrigatoriamente obedecer o protocolo de chamada de funções Unix-like da arquitetura escolhida.
- Caso algum dos passos do protocolo de chamada de subprogramas não seja necessário para completude da transcrição, é importante que este seja descrito e comentado para avaliação desta questão.
- Os valores de n e exp são sempre positivos e  $(n * exp) < +4.294.967.295$

nota(s) da questão:

<sup>1</sup> Durante uma aula de LM, o aluno G.M. solicitou ao professor o que “era um pontinho amarelo no céu?”

<sup>2</sup> Durante outra aula de LM, o aluno J.V. solicitou se poderia deixar para fazer a prática depois porque “tinha um pokemon que ele queria pegar hoje usando VPN”.

**2016AV02Q06:** (não aplicável - ignorar)

Descreva as principais diferenças entre os protocolos de chamada de sub-programas das arquiteturas x64 e x86.

*Gotta Catch 'Em All, depois da prova!*

**2017AV02Q01:** (questão adaptada para SSE e AVX, originalmente solicitava FPU)

Quantos e quais são os registradores visíveis de propósito específico para ponto-flutuante SSE/AVX? Qual destes registradores é considerado o topo da pilha de registradores de ponto-flutuante? Responda com os nomes dos registradores usados para acesso aos dados.

**2017AV02Q02:** (questão adaptada para SSE e AVX, originalmente solicitava FPU)

Quais são as instruções que movem dados escalares entre a memória e um registrador de ponto-flutuante SSE/AVX? Exemplifique o uso destas instruções.

**2016AV02Q03:**

Considere que você implementou um programa para arquitetura x86\_64 (64bits), que calcula a nota final dos alunos de Linguagem de Montagem no arquivo `ImFinais.asm`. Este programa implementa chamadas às funções `printf()` e `scanf()`. Qual é a sequência de comandos, em terminal Linux, para montar, ligar e executar o programa `ImFinais.asm`?

**2016AV02Q04:** (não aplicável - ignorar)

Considere o código do Quadro 2016AV02Q04 abaixo:

```
1. section .data
2.     var1: dw 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
3.     var2: dw 1, 2, 3, 4, 5, 6, 7, 8, 0, 9
4. section .bss
5.     var3: resw 10
6.     section .text
7. global _start
8. _start:
9.     movq    mm0, [var1]
10.    movq    [var3], mm0
11.    movq    mm1, [var2]
12.    paddw   mm0, mm1
13.    movq    [var3], mm0
14. fim:
15.    mov rax, 1
16.    mov rbx, 0
17.    int 0x80
```

Quadro 2016AV02Q04. código completo e funcional para questão Q04.

Qual é o valor contido em `var3` ao executarmos este código até o label `fim`?

(2016AV02Q05 na próxima página)

**2016AV02Q05:**

Considere o código escrito em linguagem C, completo e funcional, contido no Quadro 2016AV02Q05. Traduza<sup>1</sup> os códigos das funções powerOf3 e isMulPossible para Linguagem de Montagem utilizando sintaxe Intel e tendo como arquitetura alvo a x86\_64 (64 bits<sup>2</sup>)

<sup>1</sup>Traduzir: não otimize o código

<sup>2</sup>64 bits : respeite o protocolo e comente passos, inclusive os não utilizados!

```
1. #include<stdio.h>
2. /* def: Função que Calcula n ao cubo
3. * param: n -> ponteiro para o número n
4. * return: n^3
5. */
6. void powerOf3(int *n){
7.     int p = *n;
8.     p = p*p*p;
9.     *n = p;
10.    return *n;
11. }
12.
13. /* def: Função que Calcula o número de multiplicações usadas
14. * em uma função de multiplicação de matrizes tradicional
15. * param: c1 -> número de colunas da Matriz M1
16. *         l2 -> número de linhas da Matriz M2
17. * return: Se possível, retorna o número de multiplicações usadas
18. *         Caso contrário, retorna 0
19. */
20. int isMulPossible(int c1, int l2){
21.     int multiplicacoes = 0;
22.     if (c1 == l2){
23.         multiplicacoes = c1;
24.         powerOf3(&multiplicacoes);
25.     }
26.     return multiplicacoes;
27. }
28.
29. /* def: Função principal
30. * executa a chamada à função isMulPossible e imprime resultado
31. */
32. int main(){
33.     int Mc1 = 5;
34.     int Ml2 = 5;
35.     printf("Número de Multiplicações: %d\n", isMulPossible(Mc1,Ml2));
36.     return 0;
37. }
```

Quadro 2016AV02Q05. Código completo e funcional em C para Questão 05.