

Versão: A

Nota mínima: 7.5/20 valores / Duração: 120 minutos

Número: _____

Nome: _____

Responda aos grupos II, III, IV e V em folhas A4 separadas.

[8v] Grupo I - Assinale no seguinte grupo se as frases são verdadeiras ou falsas (uma resposta errada desconta 50% de uma correta).

- | | V | F |
|---|--------------------------|--------------------------|
| 1) Em C, considere “unsigned char x=-1; short y=10;”. À variável “short z=x+y;” é atribuído o valor 265..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 2) Em C, os tipos de dados com sinal usam mais um bit para armazenar se o valor é positivo ou negativo do que os seus equivalentes sem sinal. | <input type="checkbox"/> | <input type="checkbox"/> |
| 3) Em C, considere “int x=0xA0B0F0CC;”. À variável “short y=(short)x;” é atribuído um valor interpretado como negativo..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 4) Em C, o operador lógico (OR) termina a avaliação da expressão logo que encontre uma condição que seja avaliada como verdade..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 5) Em C, admita “short v[]={0xAABB, 0xCCDD}; int x=(int*)v;”. Então, no inteiro x fica armazenado o valor 0xCCDDAABB. | <input type="checkbox"/> | <input type="checkbox"/> |
| 6) Em C, “x>>2” aplica um deslocamento aritmético para a direita se x for do tipo unsigned int e um lógico se x for do tipo int..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 7) Em C, é seguro retornar como valor de saída de uma função o endereço de um vetor “short *vec=(short*)malloc(20)”..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 8) Em x86-64, a instrução “pushq %rax” é o equivalente a “subq \$8,%rsp” seguido de “movq %rax, (%rsp)”..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 9) Em x86-64, o valor final de %rbx após a instrução “cmovg %rax,%rbx” depende do valor dos bits do registo RFLAGS..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 10) Em x86-64, “testl \$1,%ecx” seguido de “jz xpto” permite saltar para a etiqueta xpto se o valor de %ecx for 1..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 11) Em x86-64, a stack nunca é usada para passar parâmetros a uma função..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 12) Em x86-64, a instrução “leaq (%rax,%rax,4), %rax” pode ser usada para multiplicar por cinco o valor presente em %rax..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 13) Em x86-64, é possível usar “shll \$3, %eax” seguido de “notl %eax” para multiplicar por -8 o valor de %eax..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 14) Em x86-64, admita que o valor de %rsp é 0x1008. A execução da instrução call coloca o valor de %rsp em 0x1000..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 15) Em x86-64, qualquer instrução que altere os 4 bytes menos significativos de um registo coloca a zero os 4 bytes mais significativos..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 16) Em x86-64, de acordo com a convenção de salvaguarda e restauro de registos estudada nas aulas, %r12 é um registo callee saved..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 17) O sistema operativo executa periodicamente uma desfragmentação da heap para melhorar o desempenho dos programas em C..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 18) A fragmentação interna dos blocos reservados na heap é consequência das regras de alinhamento e overhead da gestão dos blocos..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 19) Na hierarquia de memória à medida que nos afastamos do CPU, a capacidade de armazenamento aumenta, mas diminui a performance..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 20) O bloco de código “for (i=0; i<N; i++) for (j=0; j<M; j++) sum+=m[j][i];” exhibe boa localidade espacial e temporal..... | <input type="checkbox"/> | <input type="checkbox"/> |

[2v] Grupo II – Responda numa folha A4 separada que deve assinar e entregar no final do exame.

[1v] a) A instrução leaq pode ser usada para realizar operações do tipo $(A \ll K) + B$, em que K é 0, 1, 2, ou 3. Por exemplo, podemos calcular $3 \cdot A$ como $(A \ll 1) + A$, invocando leaq (%rax,%rax,2), %rax. Considerando apenas os casos $B = 0$ ou $B = A$, e para todos os valores possíveis de K, que múltiplos de A podem ser calculados com uma única invocação da função leaq?

[1v] b) Para cada um dos valores de K indicados, como podemos calcular $X \cdot K$ usando apenas o número indicado de operações (deslocamentos e somas/subtrações)?

K	Deslocamentos	Somas/Subtrações	Expressão
7	1	1	
30	4	3	
28	2	1	
55	2	2	

[4v] Grupo III – Responda numa folha A4 separada que deve assinar e entregar no final do exame.

<pre>struct s1{ short a; char b; struct s2 c; union u1 d; long e; };</pre>	<pre>struct s2{ long *f; struct s1 *g; struct s2 *h; int i; union u1 *j[3]; };</pre>	<pre>union u1 { int k; char l; long m; struct s1 *n; };</pre>
--	--	---

[1v] a) Indique o alinhamento dos campos de uma estrutura do tipo **struct s1**. Indique claramente, para cada campo, o seu endereço, bem como as partes alocadas, mas não usadas, para satisfazer as restrições de alinhamento. Indique o tamanho total da estrutura. Admita que a estrutura está colocada a partir do endereço 0x100.

[1v] **b)** Se definirmos os campos da estrutura **struct s2** por outra ordem é possível reduzir o número de bytes necessários para o seu armazenamento? **Justifique a sua resposta** indicando, em caso afirmativo, qual a ordem dos campos que garante o menor tamanho, o novo endereço de cada campo e das partes alocadas mas não usadas, bem como o novo tamanho total da estrutura.

[2v] **c)** Considere o seguinte fragmento de código em C:

```
short return_s2_c_i(struct s2 **matrix, int i, int j){
    return matrix[i][j].g->c.i;
}
```

Reescreva a função `return_s2_c_i` em Assembly. Na sua resolução tenha em consideração que `matrix` é uma matriz dinâmica de estruturas do tipo `struct s2`. Assuma que os valores de `i` e `j` estão dentro dos limites reservados. Respeite a declaração da estrutura usada na alínea a). **Comente o seu código.**

[3v] **Grupo IV – Responda numa folha A4 separada que deve assinar e entregar no final do exame.**

[1,5v] **a)** No seguinte excerto de código em C foram omitidos os valores das constantes `M` e `N`:

```
#define M      /* Número mistério 1 */
#define N      /* Número mistério 2 */

long P[M][N];
long Q[N][M];

long sum_element(long i, long j){
    return P[i][j] + Q[j][i];
}
```

Admita que a função foi compilada para valores específicos de `M` e `N` e o compilador gerou o seguinte código em Assembly:

```
sum_element:
    leaq 0(,%rdi,8), %rdx
    subq %rdi, %rdx
    addq %rsi, %rdx
    leaq (%rsi,%rsi,4), %rax
    addq %rax, %rdi
    leaq Q(%rip), %r8
    leaq P(%rip), %r9
    movq (%r8,%rdi,8), %rax
    addq (%r9,%rdx,8), %rax
    ret
```

Quais os valores de `M` e `N`? **Justifique a sua resposta**

[1,5v] **b)** Admita os seguintes endereços e conteúdo da memória:

Endereço	Conteúdo
0x1000	0x1018
0x1004	0x1014
0x1008	0x1010
0x100C	0x100C
0x1010	0x1008
0x1014	0x1004
0x1018	0x1000

Admita que o endereço do vetor `vec` é `0x1000` e são executadas as seguintes instruções:

```
leaq vec(%rip), %rdx
movl $3, %ecx
leaq (%rdx, %rcx, 4), %rax
movl (%rax, %rcx, 4), %eax
```

No final, que valor (em hexadecimal) fica em `%eax`? **Justifique a sua resposta.**

[3v] **Grupo V – Responda numa folha A4 separada que deve assinar e entregar no final do exame.**

```
/* increment values by k */
void incrk(int *v, int *z, int k){
    *v += k;
    *z += k;
}

/* compute x + 3 + y + 3 */
int fun(int x, int y){
    int localx = x;
    int localy = y;
    incrk(&localx, &localy, 3);
    return localx + localy;
}
```

Com base no código C acima, preencha os espaços em branco no código correspondente em Assembly ao lado. **(escreva a função completa na folha A4).**

```
fun:
    pushq %rbp
    movq %rsp,%rbp
    _____
    _____
    _____
    _____
    call incrk
    _____
    _____

    movq %rbp,%rsp
    popq %rbp
    ret
```