

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 correcta).**

- | | V F |
|--|---|
| 1) Em C, o maior valor que podemos armazenar numa variável do tipo <code>int</code> é 2^{32} | <input type="checkbox"/> <input type="checkbox"/> |
| 2) Admita a variável <code>unsigned int a = 0xFFFFFFFF</code> em C. Então, a variável <code>unsigned int b = a + 1</code> tem o valor zero | <input type="checkbox"/> <input type="checkbox"/> |
| 3) Admita a variável <code>short a = 0x0123</code> e um valor dado por <code>&a</code> de <code>0x200</code> em C. Então, o valor presente no byte <code>0x201</code> é <code>0x01</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 4) Em C, ao truncarmos uma variável do tipo <code>int</code> que armazena um valor positivo para um <code>short</code> podemos ficar com um valor negativo | <input type="checkbox"/> <input type="checkbox"/> |
| 5) Em C, o operador lógico <code> </code> (OR) termina a avaliação da expressão se encontrar uma condição que seja avaliada como verdade | <input type="checkbox"/> <input type="checkbox"/> |
| 6) Em C, as divisões de inteiros usando deslocamentos podem exigir a alteração do dividendo para que o arredondamento seja correto | <input type="checkbox"/> <input type="checkbox"/> |
| 7) Admita que <code>int *ptr</code> armazena endereço inicial de um vetor de <code>int</code> . Logo, <code>(short*)ptr + 8</code> aponta para o quinto elemento | <input type="checkbox"/> <input type="checkbox"/> |
| 8) Admita que declara a variável <code>int x</code> como variável global em C. Logo, os 4 bytes são reservados na <i>stack</i> | <input type="checkbox"/> <input type="checkbox"/> |
| 9) Em C, é possível retornar como valor de saída de uma função o endereço de um vetor <code>short *vec = (short*)malloc(20)</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 10) Em Assembly, a instrução <code>popl %eax</code> é equivalente a executar <code>movl (%esp), %eax</code> seguido de <code>addl \$4, %esp</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 11) Em Assembly, a instrução <code>leal (%eax, %eax, 4), %eax</code> pode ser usada para multiplicar por 5 o valor em <code>%eax</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 12) Se <code>%ecx</code> for 4 e <code>%esi</code> o endereço inicial de um vetor de <code>short</code> , <code>movw 6(%esi, %ecx, 2), %ax</code> coloca em <code>%ax</code> o oitavo elemento | <input type="checkbox"/> <input type="checkbox"/> |
| 13) Em IA32, a <i>stack</i> é usada para suportar a passagem do valor de retorno de uma função invocada à função invocadora | <input type="checkbox"/> <input type="checkbox"/> |
| 14) Admita que o valor de <code>%esp</code> é <code>0x1004</code> . A execução da instrução <code>ret</code> coloca o valor de <code>%esp</code> em <code>0x1000</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 15) Em IA32, reservamos espaço para as variáveis locais de uma função subtraindo o número de bytes necessários ao valor atual de <code>%ebp</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 16) Admita a matriz global <code>int m[4][5]</code> . Em Assembly, acedemos ao endereço de <code>m[i]</code> calculando <code>m + i*20</code> | <input type="checkbox"/> <input type="checkbox"/> |
| 17) Uma estrutura, alinhada de acordo com as regras estudadas, com 1 <code>char</code> , 1 <code>double</code> e um 1 <code>char*</code> (por esta ordem) ocupa 16 bytes | <input type="checkbox"/> <input type="checkbox"/> |
| 18) O tamanho de uma estrutura é garantidamente menor se indicarmos os seus campos por ordem decrescente do seu tamanho | <input type="checkbox"/> <input type="checkbox"/> |
| 19) A fragmentação interna dos blocos reservados na <i>heap</i> é originada pelas regras de alinhamento e <i>overhead</i> da gestão dos blocos | <input type="checkbox"/> <input type="checkbox"/> |
| 20) A possibilidade de existirem diversas referências para a mesma posição de memória em C dificulta a otimização efetuada pelo compilador | <input type="checkbox"/> <input type="checkbox"/> |

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

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 */

int arith(int x, int y) {
    int result = 0;
    result = x*M + y/N;
    return result;
}
```

Admita agora que a função foi compilada para valores específicos de `M` e `N`. O compilador optimizou a multiplicação e divisão pelas constantes `M` e `N` usando deslocamentos, tal como estudado nas

aulas. O código seguinte em C é uma tradução do código Assembly gerado:

```
int optarith(int x, int y) {
    int t = x;
    x <<= 5;
    x -= t;
    if (y < 0) y += 7;
    y >>= 3;
    return x+y;
}
```

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

[5v] **Grupo III – Responda numa folha A4 separada que deve assinar e entregar no final do exame.**

Considere as seguintes declarações:

```
typedef struct {
    short x;
    int y;
} structA;
```

```
typedef struct {
    structA a;
    structA *b;
    int x;
    char c;
    int y;
    char e[3];
    short z;
} structB;
```

[1.5v] **a)** Indique o alinhamento dos campos de uma estrutura do tipo `structB`. 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.**

[1.5v] **b)** Se definirmos os campos da estrutura `structB` por outra ordem é possível reduzir o número de bytes necessários para o seu armazenamento? **Justifique a sua resposta.** Indique, 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 as seguintes funções em C, respeitando as declarações iniciais das estruturas:

<pre>short fun1(structB *s){ return s->a.x; } short* fun2(structB *s){ return &s->z; }</pre>	<pre>short fun3(structB *s){ return s->z; } short fun4(structB *s){ return s->b->x; }</pre>
---	---

Indique a que funções (`fun1`, `fun2`, `fun3` ou `fun4`) correspondem os seguintes excertos de código em Assembly. **Escreva as funções completas na folha A4.**

<pre>_____: pushl %ebp movl %esp,%ebp movl 8(%ebp),%eax addl \$28,%eax popl %ebp ret _____: pushl %ebp movl %esp,%ebp movl 8(%ebp),%eax movl 8(%eax),%eax movswl (%eax),%eax popl %ebp ret</pre>	<pre>_____: pushl %ebp movl %esp,%ebp movl 8(%ebp),%eax movl 28(%eax),%eax popl %ebp ret _____: pushl %ebp movl %esp,%ebp movl 8(%ebp),%eax movswl (%eax),%eax popl %ebp ret</pre>
--	---

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

Admita a seguinte função em C que recebe como primeiro parâmetro o endereço de uma *string* `str` e como segundo parâmetro o endereço de um inteiro `hash` no qual a função armazena o resultado.

<pre>void calcHash(char *str, int *hash){ int i, j; *hash = 0; for(i = 0; i < strlen(str); i++){ j = strlen(str) / 2; *hash += secret(str,i) * 32 + j); } }</pre>	<pre>int secret(char *str, int pos){ return str[pos] % 26; }</pre>
---	--

Apresente uma segunda versão da função `calcHash` em C com a mesma funcionalidade, mas melhor desempenho. Admita que o compilador que é usado não efetua nenhuma otimização. **Indique claramente cada uma das otimizações usadas sob a forma de comentário no código.**

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

<pre>fun: pushl %ebp movl %esp,%ebp movl 16(%ebp),%ecx movl 12(%ebp),%eax movl 8(%ebp),%edx cmpl %ecx,%edx jl .L1 .L2: addl %edx,%eax decl %edx cmpl %ecx,%edx jge .L2 .L1: movl %ebp,%esp popl %ebp ret</pre>	<p>Com base no código Assembly à esquerda, preencha os espaços em branco no código correspondente em C. Apenas pode usar as variáveis <code>x</code>, <code>y</code>, <code>z</code>, <code>i</code> e <code>result</code> nas expressões (<i>não use nomes de registos!</i>) (escreva a função completa na folha A4).</p> <pre>int fun(int x, int y, int z){ int i, result = _____; for(i=____; ____; ____){ result = _____; } return _____; }</pre>
--	---