

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, admita a variável “char x = -1;”. Logo, o valor armazenado em “char y = (unsigned)x >> 1;” é 127..... | <input type="checkbox"/> <input type="checkbox"/> |
| 2) Em C, a operação u << k tem sempre como resultado u * 2 ^k , para valores inteiros de u com ou sem sinal e 0 < k <= 31..... | <input type="checkbox"/> <input type="checkbox"/> |
| 3) Em C, admita a variável “unsigned int x=0x12345678;” cujo endereço é 0x100. Logo, o valor presente no byte 0x102 é 0x34.... | <input type="checkbox"/> <input type="checkbox"/> |
| 4) Em C, a função malloc permite-nos reservar blocos de memória na stack em tempo de execução que podem ser depois redimensionados..... | <input type="checkbox"/> <input type="checkbox"/> |
| 5) Em C, quando a soma aritmética de duas variáveis u e v do tipo int é superior a 2 ³¹ o valor obtido é equivalente a u + v - 2 ³¹ | <input type="checkbox"/> <input type="checkbox"/> |
| 6) Em C, a divisão correta de um inteiro negativo x por 2 ^k através de um deslocamento deve ser obtida com “(x+(1<<k)-1)>>k”..... | <input type="checkbox"/> <input type="checkbox"/> |
| 7) Em C, admita um vetor “int vec[10];” e um apontador “short *ptr = (short*) vec”. Então, ptr + 4 avança para vec[2]..... | <input type="checkbox"/> <input type="checkbox"/> |
| 8) Em C, é correto retornar como valor de saída de uma função o endereço de um bloco de memória reservado na heap dentro da função..... | <input type="checkbox"/> <input type="checkbox"/> |
| 9) Em Assembly, o equivalente a “*ptr1 = *ptr2”, apontadores do tipo int* em C, pode ser obtido com “movl (%eax), (%ebx)”..... | <input type="checkbox"/> <input type="checkbox"/> |
| 10) Em Assembly, a instrução “popl %eax” é equivalente a “movl (%esp), %eax” seguido de “addl \$4, %esp”..... | <input type="checkbox"/> <input type="checkbox"/> |
| 11) Em Assembly, as operações de multiplicação e divisão de inteiros têm instruções diferentes para valores com e sem sinal..... | <input type="checkbox"/> <input type="checkbox"/> |
| 12) A adição de dois bytes com sinal com valores \$127 e \$10 deixa as flags do registo EFLAGS com os valores ZF=0, SF=1, CF=0, OF=1..... | <input type="checkbox"/> <input type="checkbox"/> |
| 13) Em IA32, a stack é usada para suportar o retorno do valor de saída de uma função, tal como acontece com o controlo de fluxo..... | <input type="checkbox"/> <input type="checkbox"/> |
| 14) Em IA32, admita que o valor de %esp é 0x1004. A execução da instrução “call func” coloca o valor de %esp em 0x1000..... | <input type="checkbox"/> <input type="checkbox"/> |
| 15) De acordo com a convenção usada em Linux/IA32, a responsabilidade da salvaguarda e restauro de %edx é da função invocadora..... | <input type="checkbox"/> <input type="checkbox"/> |
| 16) Admita a matriz global int m[10][3]. Em Assembly, acedemos ao valor de m[3][1] avançando 40 bytes a partir de m..... | <input type="checkbox"/> <input type="checkbox"/> |
| 17) Uma estrutura, alinhada de acordo com as regras estudadas, com 2 char, um vetor de 5 int e 1 short (por esta ordem) ocupa 24 bytes.... | <input type="checkbox"/> <input type="checkbox"/> |
| 18) O tamanho de uma estrutura sujeita a alinhamento tem de ser múltiplo da menor restrição de alinhamento dos seus campos..... | <input type="checkbox"/> <input type="checkbox"/> |
| 19) O tamanho de um bloco reservado na heap pode ser maior do que o número de bytes passados por parâmetro na função malloc..... | <input type="checkbox"/> <input type="checkbox"/> |
| 20) A invocação de funções introduz overhead e limita as possibilidades de optimização dos programas por parte do compilador..... | <input type="checkbox"/> <input type="checkbox"/> |

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

Considere o seguinte código em C em que num_t é um tipo de dados declarado através de typedef:

```
void product(num_t *dest, unsigned int x, num_t y){
    *dest = x * y;
}
```

Admita que o GCC gerou o bloco de código em Assembly descrito ao lado, correspondente ao corpo da função.

```
movl 12(%ebp), %eax
movl 20(%ebp), %ecx
imull %eax, %ecx
mull 16(%ebp)
leal (%ecx, %edx), %edx
movl 8(%ebp), %ecx
movl %eax, (%ecx)
movl %edx, 4(%ecx)
```

[1v] a) Qual o tipo de dados de num_t? Justifique a sua resposta.

[1v] b) Descreva o algoritmo usado na multiplicação. Argumente porque está correto.

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

Considere as seguintes declarações:

```
typedef struct {
    short int a[3];
    char b;
    long long int c;
    int d;
    unionB ub;
    char e;
}structA;
```

```
typedef union {
    int a;
    char b;
    short c;
    long int d;
}unionB;
```

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

```
char return_unionB_b(structA **matrix, int i, int j){
    return matrix[i][j].ub.b;
}
```

Reescreva a função `return_unionB_b` em Assembly. Na sua resolução tenha em consideração que `matrix` é uma matriz de estruturas criada dinamicamente na heap através da função `malloc`. Respeite a declaração inicial 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.**

Considere o seguinte código em Assembly:

```
func:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp), %eax
    movl 12(%ebp), %ecx
    movl 16(%ebp), %edx
    cmpl %ecx, %edx
    jle .L2
    movl 8(%eax), %eax

.L1:
    shrw $2, 4(%eax)
    incl %ecx
    cmpl %ecx, %edx
    jg .L1

.L2:
    movl %esp, %ebp
    popl %ebp
    ret
```

Com base no código Assembly à esquerda, preencha os espaços em branco no código correspondente em C. **(escreva a estrutura e a função completas na folha A4).**

```
typedef struct node{
    int x;
    _____ y;
    struct node *next;
    struct node *prev;
}node_t;

void func(_____, int a, int b){
    node_t *m;

    m = _____;

    while(_____) {
        m->y = _____;
        a++;
    }
}
```

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

Admita o seguinte excerto de código em C. A função `calc_matrix` recebe como primeiro parâmetro o endereço de uma estrutura, onde está armazenado o endereço de uma matriz dinâmica de inteiros e o seu tamanho atual, e como segundo parâmetro o endereço de um inteiro `res` no qual a função armazena o resultado computado.

```
typedef struct{
    int lines;
    int columns;
    int **m;
}data_t;

void calc_matrix(data_t *matrix, int *res){
    int i, j;

    *res = 0;

    for(j= 0; j < num_columns(matrix); j++){
        for(i = 0; i < num_lines(matrix); i++){
            *res += 16*i + get_element(matrix,i,j);
        }
    }
}
```

```
int num_lines(data_t *matrix){
    return matrix->lines;
}

int num_columns(data_t *matrix){
    return matrix->columns;
}

int get_element(data_t *matrix, int i, int j){
    return matrix->m[i][j];
}
```

Apresente uma segunda versão da função `calc_matrix` 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.**