

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 “unsigned short x=0xABFF;”. O valor armazenado em “char y = (char)x   0x0;” é -1.....                             | <input type="checkbox"/> <input type="checkbox"/> |
| 2) Em C, qualquer que seja o valor atribuído à variável “int y;”, a atribuição “int x = -y;” é equivalente a “int x = ~y + 1;”.....          | <input type="checkbox"/> <input type="checkbox"/> |
| 3) Em C, admita as variáveis “unsigned char x,y;”. Se x for par, “y = (x 1) & 0xFF;” atribui um valor ímpar a y .....                        | <input type="checkbox"/> <input type="checkbox"/> |
| 4) Em C, a função realloc permite-nos redimensionar em tempo de execução blocos de memória reservados na stack .....                         | <input type="checkbox"/> <input type="checkbox"/> |
| 5) Em C, admita a variável “short x;” à qual é atribuída um valor negativo. Logo, “short y = x*2;” será sempre menor do que zero ....        | <input type="checkbox"/> <input type="checkbox"/> |
| 6) Em C, o maior valor positivo que é possível armazenar na variável “unsigned short x;” é $2^{16} - 1$ .....                                | <input type="checkbox"/> <input type="checkbox"/> |
| 7) Em C, admita que “int *ptr=(int*)malloc(20);” é declarado numa função. É correto terminar a função com “return ptr;” .....                | <input type="checkbox"/> <input type="checkbox"/> |
| 8) O Assembler é o programa que recebe como <i>input</i> código escrito numa linguagem de alto nível como o C e o traduz para Assembly ..... | <input type="checkbox"/> <input type="checkbox"/> |
| 9) Em IA32, a instrução “leal (%eax,%eax,4),%eax” multiplica por 5 o valor presente em %eax .....  | <input type="checkbox"/> <input type="checkbox"/> |
| 10) Em IA32, a instrução “idivb %cl” assume que o dividendo se encontra em %ax, deixando o quociente em %al e o resto em %ah.....            | <input type="checkbox"/> <input type="checkbox"/> |
| 11) Em IA32, se pretendermos dividir valores inteiros sem sinal podemos usar as instruções div ou idiv obtendo sempre o mesmo resultado..    | <input type="checkbox"/> <input type="checkbox"/> |
| 12) Em IA32, “shrl %eax” seguido de “jnc xpto” permite saltar para a linha xpto se o valor presente em %eax for par.....                     | <input type="checkbox"/> <input type="checkbox"/> |
| 13) Admita o vetor global “short a[10];” em C. “movl \$2,%ecx” seguido de “movw a(,%ecx,4),%ax” coloca a [4] em %ax.....                     | <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 0x1008 .....                 | <input type="checkbox"/> <input type="checkbox"/> |
| 15) De acordo com a convenção usada em Linux/IA32, a responsabilidade da salvaguarda e restauro de %ebp é apenas da função invocadora....    | <input type="checkbox"/> <input type="checkbox"/> |
| 16) Admita que M e N são valores grandes. “for (j=0;j<N;j++) for (i=0;i<M;i++) sum+=m[i][j];” terá a melhor performance.....                 | <input type="checkbox"/> <input type="checkbox"/> |
| 17) O tempo de acesso a um setor num disco é dominado pelo tempo de pesquisa e latência de rotação da cabeça de leitura .....                | <input type="checkbox"/> <input type="checkbox"/> |
| 18) Na hierarquia de memória à medida que nos afastamos do processador, a capacidade de armazenamento aumenta bem como a performance .       | <input type="checkbox"/> <input type="checkbox"/> |
| 19) Diz-se que um bloco de código possui boa localidade espacial quando não existem intervalos de alinhamento entre as variáveis usadas..... | <input type="checkbox"/> <input type="checkbox"/> |
| 20) A invocação de funções introduz <i>overhead</i> e limita as possibilidades de otimização 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.**

[1v] **a)** Considere o seguinte excerto de código. Qual dos outputs indicados do lado direito é produzido? **Justifique a sua resposta.**

```
int x = 0x15213F10 >> 4;
char y = (char)x;
unsigned char z = (unsigned char)x;
printf("y = %d, z = %u", y, z);
```

- (a) y = -241, z = 15
- (b) y = -15, z = 241
- (c) y = -241, z = 241
- (d) y = -15, z = 15

[1v] **b)** Considere o seguinte excerto de código. Qual dos outputs indicados do lado direito é produzido, assumindo que o utilizador insere corretamente um inteiro? **Justifique a sua resposta.**

```
int x;
printf("Please input an integer:");
scanf("%d",&x);
printf("%d", (!x)<<31);
```

- (a) 0, qualquer que seja o inteiro lido do teclado
- (b) INT\_MIN, qualquer que seja o inteiro lido do teclado
- (c) 0 ou INT\_MIN, dependendo do inteiro lido do teclado
- (d) Um valor dependente do inteiro lido do teclado

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

Considere as seguintes declarações:

```
struct s1{
    char a[3];
    union u1 *b;
    int c;
};
```

```
struct s2{
    struct s1 d;
    struct s1 *e;
    struct s2 *f;
    long long int g;
    short h[3];
};
```

```
union u1 {
    int i;
    struct s2 j;
    struct s1 *k;
};
```

[1.5v] **a)** Indique o alinhamento dos campos de uma estrutura do tipo `struct s2`. 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 `struct s1` 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:

```
int return_ul_i(struct s2 matrix[10][20], int i, int j){
    return matrix[i][j].e->b->i;
}
```

Reescreva a função `return_ul_i` em Assembly. Na sua resolução tenha em consideração que `matrix` é uma matriz de estruturas estática. 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.**

Considerando as declarações do exercício anterior, preencha os espaços em branco nas funções à direita, considerando o código correspondente à esquerda. **(escreva as funções completas na folha A4)**

```
proc1:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl 4(%eax),%eax
    movw 32(%eax),%ax
    movl %ebp, %esp
    popl %ebp
    ret

int proc2(struct s2 *x){
    return x->e->c;
}

proc3:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl (%eax),%eax
    movl 8(%eax),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```

```
short proc1(struct s1 *x){
    return x->_____ ;
}
```

```
proc2:
    pushl %ebp
    movl %esp,%ebp
    _____
    _____
    movl %ebp,%esp
    popl %ebp
    ret
```

```
int proc3(union ul *x){
    return x->_____ ;
}
```

[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 para manipulação de uma lista simplesmente ligada não circular com medições de temperaturas em graus Fahrenheit. Assuma que o campo `next` com o valor `NULL` indica o fim da lista e que existe uma função `int length(List *p)` para determinar o número de elementos da lista. A função `count_positives_celsius` recebe em `p` o endereço de uma lista e armazena no endereço dado por `k` o número de elementos dessa lista cujas temperaturas em Celsius são maiores do que 0.

```
typedef struct LIST{
    int data;
    struct LIST *next;
}List;
```

```
void count_positives_celsius(List *p, int *k){
    int i;
    *k = 0;
    for (i = 0; i < length(p); i++){
        if ((p->data-32)*5/8 > 0)
            (*k)++;
        p = p->next;
    }
}
```

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