

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 “unsigned short x=0xABFF;” e “char *p=(char)&x;”. “printf(“%hhX”, *p);” imprime o valor “AB”..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 2) Em C, admita “int x = 0xCFC7;” e “short y = (short)x;”. Logo, “int z = (int)y;” atribui o valor 0xCFC7 a z..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 3) Em C, admita “short x = 0x1234;”. Logo, “unsigned short y = (x && 0x00FF);” atribui o valor 0x34 a y..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 4) Em C, usamos memória dinâmica porque a <i>heap</i> é uma zona de memória com um tempo de acesso menor do que a <i>stack</i> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 5) Em C, admita a variável “int x;” à qual é atribuída um valor positivo. Logo, “short y=(short)x*2;” será sempre positivo..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 6) Em C, o maior valor positivo que é possível armazenar na variável “char x;” é $2^8 - 1$ | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 7) Em C, admita que ptr é uma variável do tipo char*. Então, a expressão (short*)ptr + 7 avança 14 bytes na memória..... | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 8) O compilador 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 checked="" type="checkbox"/> | <input type="checkbox"/> |
| 9) Em IA32, é possível usar a instrução “leal (%eax, %eax, 4), %eax” para multiplicar por 50 o valor presente em %eax..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 10) Em IA32, reservamos espaço para as variáveis locais de uma função somando o número de bytes necessários ao valor atual de %esp..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 11) Em IA32, é possível retornar de um bloco de código com a instrução ret quando a sua invocação/salto foi efetuada com a instrução jmp..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 12) Em IA32, são usados registos para suportar a passagem do valor de retorno de uma função invocada à função invocadora..... | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 13) Admita o vetor global “int a[10];” em C. “movl \$2, %ecx” seguido de “movl a(, %ecx, 4), %eax” coloca a[4] em %eax..... | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| 14) Em IA32, podemos substituir “popl %eax” por “movl (%esp), %eax” seguido de “addl \$4, %esp”..... | <input checked="" 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 checked="" type="checkbox"/> |
| 16) O tamanho de uma <i>union</i> é o menor possível se declararmos os seus campos por ordem decrescente de tamanho do tipo de dados..... | <input type="checkbox"/> | <input checked="" 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 checked="" type="checkbox"/> | <input type="checkbox"/> |
| 18) Na hierarquia de memória à medida que nos afastamos do processador a capacidade de armazenamento aumenta e diminui a performance.... | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 19) A localidade espacial indica a probabilidade de acesso a dados e instruções em endereços próximos daqueles acedidos recentemente..... | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 20) Uma das otimizações efetuadas pelos compiladores é a substituição da invocação de uma função pelo seu código..... | <input checked="" type="checkbox"/> | <input type="checkbox"/> |

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

[1v] a) Escreva em C o código da função unsigned int replace_byte(unsigned int x, int i, unsigned char b) que deverá retornar um valor em que o byte i do parâmetro x foi substituído pelo byte b. Os bytes são numerados de 0 (menos significativo) a 3 (mais significativo). Alguns exemplos de como a função deve operar:

```
replace_byte(0x12345678, 2, 0xAB) --> 0x12AB5678
replace_byte(0x12345678, 0, 0xAB) --> 0x123456AB
```

```
unsigned int replace_byte (unsigned int x, int i, unsigned char b){
    int nbits = i << 3;
    unsigned mask = ~(0xFF << nbits);
    return (x & mask) | (b << nbits);
}
```

[1v] b) Preencha a seguinte tabela mostrando o efeito das instruções seguintes, quer em termos de localização dos resultados (registo ou endereço de memória), quer dos respetivos valores. (cada instrução não é afetada pela execução das instruções anteriores)

Endereço	Valor
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

Registo	Valor
%eax	0x100
%ecx	0x1
%edx	0x3
%ebx	0x4

Instrução	Destino	Valor
addl %ecx, (%eax)	0x100	0x100
subl %edx, 4(%eax)	0x104	0xA8
shll \$4, (%eax, %edx, 4)	0x10C	0x110
incl 8(%eax)	0x108	0x14
subl %ebx, %eax	%EAX	0xFC

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

Considere as seguintes declarações:

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

[1.5v] **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.**

```
struct s1{
    int *a;          0x100: 4 bytes
    struct s2 b;     0x104: 20 bytes
    struct s1 *c;    0x118: 4 bytes
    char d;          0x11C: 1 byte
    [gap]            0x11D: 3 bytes
};
```

Tamanho da estrutura: 32 bytes

[1.5v] **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.

Não. Os campos da estrutura já estão indicados por ordem decrescente de tamanho de tipo de dados. Além disso, nesta estrutura, independentemente da ordem que indicarmos os campos, a restrição de alinhamento que obriga a que o tamanho total da estrutura seja múltiplo da maior restrição de alinhamento dos seus campos (neste caso $K=4$) faz com que a estrutura tenha sempre 20 bytes.

[2v] **c)** Considere a seguinte função em C:

```
void s1_init(struct s1 *p){
    p->b.d = p->b.e.h;
    p->a = &(p->b.d);
    p->c = p->b->f;
}
```

O GCC gerou o código Assembly ao lado para `s1_init`. Com base nesse código, preencha as expressões em falta no código C. **Comente o seu código.**

```
s1_init:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl 8(%eax),%edx
    movl %edx,4(%eax)
    leal 4(%eax),%edx
    movl %edx,(%eax)
    movl 12(%eax),%edx
    movl %edx,24(%eax)
    movl %ebp,%esp
    popl %ebp
    ret
```

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

Considerando o código Assembly à esquerda, preencha os espaços em branco no código correspondente em C. Tenha em conta a reordenação de instruções introduzida pelo GCC. (escreva a função completa na folha A4)

```
proc1:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl 12(%ebp),%edx
    shr1 %eax
    jnc .L2
    movl 8(%ebp),%eax
    cmpl %edx,%eax
    jle .L3
    imull %edx,%eax
    jmp .L4
.L3:
    leal (%edx,%eax),%eax
    jmp .L4
.L2:
    movl 8(%ebp),%eax
    cmpl $20,%eax
    jg .L5
    leal (%eax,%edx,4),%eax
    jmp .L4
.L5:
    subl %edx,%eax
.L4:
    movl %ebp,%esp
    popl %ebp
    ret
```

```
int proc1(int x, int y){
    int val = x;

    if(x % 2 != 0){
        if(x > y){
            val = x * y;
        }else{
            val = x + y;
        }
    }else{
        if(x <= 20){
            val = x + 4*y;
        }
        else{
            val = x - y;
        }
    }

    return val;
}
```

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

Considere a seguinte função em C que permite multiplicar duas matrizes de inteiros. Apresente uma nova versão da função `matrix_mult` com a mesma funcionalidade, mas melhor desempenho. Admita que o compilador que é usado não efetua nenhuma otimização. Considere que as matrizes são quadradas. A constante `N` indica o número de linhas e colunas das matrizes. **Indique claramente cada uma das otimizações usadas sob a forma de comentário no código.**

```
void matrix_mult(int a[N][N], int b[N][N], int c[N][N]){
    int i,j,k;

    for (i = 0; i < N; i++){
        for (j = 0; j < N; j++){
            c[i][j] = 0;
            for (k = 0; k < N; k++)
                c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

```
void matrix_mult(int a[N][N], int b[N][N], int c[N][N]){
    int i,j,k;
    int *aRow , *bCol, *bElement, *cRow, tmp = 0;
    aRow = &a[0][0];
    cRow = &c[0][0];
    for (i = 0; i < N; i++){
        bCol = &b[0][0];
        for (j = 0; j < N; j++){
            tmp = 0;
            bElement = bCol;
            for (k = 0; k < N; k++){
                // Reduzir custo de operações de cálculo dos endereços
                tmp += aRow[k] * *bElement;
                bElement += N;
            }
            bCol ++;
            // Reduzir escritas na memória
            cRow[j] = tmp;
        }
        aRow += N;
        cRow += N;
    }
}
```