

Versão: A

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

Número: _____

Nome: _____

Responda aos grupos III, IV e V em folhas A4 separadas. No final, deverá entregar 4 folhas assinadas, indicando a versão do enunciado.

[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 <code>cast</code> de uma variável do tipo <code>int</code> para uma do tipo <code>unsigned int</code> altera o padrão de bits da variável..... | <input type="checkbox"/> |
| 2) Em C, numa expressão com variáveis do tipo <code>int</code> com e sem sinal, todas as variáveis são convertidas para valores sem sinal | <input type="checkbox"/> |
| 3) Em C, se tivermos um <code>char</code> com representação binária de 10011010, o <code>cast</code> para um <code>short</code> resulta no valor 1111111110011010 | <input type="checkbox"/> |
| 4) Admita um inteiro <code>x</code> com valor 0x01234567 e um valor dado por <code>&x</code> de 0x100. Logo, o valor presente no byte 0x101 é 0x45..... | <input type="checkbox"/> |
| 5) Em C, a adição de duas variáveis do tipo <code>int</code> com valores positivos nunca pode resultar num valor negativo | <input type="checkbox"/> |
| 6) Em C, a operação <code>u<<k</code> resulta em <code>u*2^k</code> , independentemente da variável <code>u</code> ter ou não sinal | <input type="checkbox"/> |
| 7) Em Assembly, a instrução <code>movl %eax, (%esp)</code> pode ser usada para escrever o valor de <code>%eax</code> no topo da <code>stack</code> | <input type="checkbox"/> |
| 8) Em Assembly, a instrução <code>cmp</code> não altera os seus parâmetros nem os bits do registo <code>EFLAGS</code> | <input type="checkbox"/> |
| 9) Admita que o endereço de uma variável <code>x</code> está armazenado em <code>%edi</code> . É possível alterar o valor de <code>x</code> executando <code>movl \$15, %edi</code> | <input type="checkbox"/> |
| 10) A instrução <code>movl 4(%ebp), %eax</code> armazena em <code>%eax</code> o valor antigo de <code>%ebp</code> numa função que inicia com o prólogo estudado | <input type="checkbox"/> |
| 11) Admita que o endereço de um vetor do tipo <code>int</code> está armazenado em <code>%esi</code> e que o valor de <code>%ecx</code> é 2. A instrução <code>leal (%esi, %ecx, 4), %esi</code> armazena em <code>%esi</code> o endereço do terceiro elemento do vetor..... | <input type="checkbox"/> |
| 12) A execução da instrução <code>call</code> não implica a alteração do valor do registo <code>%esp</code> | <input type="checkbox"/> |
| 13) Não é possível passar o endereço de uma variável local para outra função chamada pela primeira usando a <code>stack</code> | <input type="checkbox"/> |
| 14) Depois de um <code>call</code> de uma função com 3 parâmetros inteiros podemos executar <code>subl \$12, %esp</code> para retirar os parâmetros da <code>stack</code> | <input type="checkbox"/> |
| 15) A convenção de salvaguarda de registos indica que o registo <code>%esi</code> deve ser gerido pela função invocada..... | <input type="checkbox"/> |
| 16) Admita a declaração da matriz <code>int m[4][5]</code> . Em Assembly, o endereço do elemento <code>m[i][j]</code> é dado pela expressão <code>m+20*i+4*j</code> | <input type="checkbox"/> |
| 17) Uma estrutura, alinhada de acordo com as regras estudadas, com um vector de 2 <code>char</code> , 1 <code>int</code> e 1 <code>short</code> (por esta ordem) ocupa 8 bytes.... | <input type="checkbox"/> |
| 18) O espaço ocupado por uma estrutura alinhada é sempre o mesmo, independentemente da ordem dos seus campos..... | <input type="checkbox"/> |
| 19) O espaço ocupado por uma <code>union</code> é sempre o mesmo, independentemente da ordem dos seus campos..... | <input type="checkbox"/> |
| 20) O compilador não tem qualquer dificuldade em mover a invocação de funções para outro local do programa com vista à sua optimização | <input type="checkbox"/> |

[2v] **Grupo II – Para cada uma das expressões seguintes em C indique se a expressão é sempre verdadeira (uma resposta errada desconta 50% de uma correcta).**

Considere as seguintes declarações:

```
int x = f(); /* Retorna valor em [0, INT_MAX] */
int y = f(); /* Retorna valor em [0, INT_MAX] */
```

```
/* Conversão para outros formatos */
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
double dx = (double) x;
double dy = (double) y;
```

Expressão	Sempre verdade?
<code>x*x >= 0</code>	<input type="checkbox"/>
<code>x+y == uy+ux</code>	<input type="checkbox"/>
<code>x == (int)(float)x</code>	<input type="checkbox"/>
<code>(double)(float)x == (double)ux</code>	<input type="checkbox"/>
<code>dx + dy == (double)(y+x)</code>	<input type="checkbox"/>
<code>(dx + dy) - dx == dy</code>	<input type="checkbox"/>

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

Considere as seguintes declarações:

```
typedef struct {
    char c;
    double *p;
    int i;
    double d;
    short s;
} struct1;
```

```
typedef struct {
    float x;
    struct1 *s1;
    char e;
    int v[4];
} struct2;
```

[1.5v] **a)** Indique o alinhamento dos campos de uma estrutura do tipo `struct1`. Indique claramente, para cada campo, o seu endereço, bem como as partes alocadas mas não usadas para satisfazer as restrições de alinhamento. **Admita que a estrutura está colocada a partir do endereço 0x100.**

[1.5v] **b)** Se definirmos os campos da estrutura `struct1` 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.

[2v] **c)** Para cada sequência de código em Assembly à esquerda, complete a função correspondente em C à direita, considerando as declarações iniciais das estruturas (**escreva a função completa na folha A4**).

<pre>f1: pushl %ebp movl %esp, %ebp movl 8(%ebp), %eax movl 16(%eax), %eax movl %ebp, %esp popl %ebp ret f2: pushl %ebp movl %esp, %ebp movl 8(%ebp), %eax movl 4(%eax), %eax movl 8(%eax), %eax movl %ebp, %esp popl %ebp ret</pre>	<pre>__ f1(struct2 *s2) { return _____; } __ f2(struct2 *s2) { return _____; }</pre>
---	---

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

O código seguinte em C preenche a diagonal de uma matriz $N \times N$ com o valor passado no argumento `val`.

```
void fix_set_diag(int m[N][N], int val){
    int i;
    for(i = 0; i < N; i++)
        m[i][i] = val;
}
```

Quando compilado para Assembly, o GCC gera o seguinte código para um valor de $N = 4$

```
fix_set_diag:
    pushl %ebp
    movl %esp, %ebp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    movl $0, %eax
.L14:
    movl %edx, (%ecx,%eax)
    addl $20, %eax
    cmpl $80, %eax
    jne .L14
    ret
```

Reescreva a função `fix_set_diag` em C usando uma optimização similar à demonstrada no código Assembly. Use expressões que recorram à variável N em vez de valores inteiros constantes, para que o seu código continue a funcionar se o valor de N for alterado.

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

Considere o seguinte código Assembly:

```
loop:
    pushl %ebp
    movl %esp, %ebp
    movl 8(%ebp), %ecx
    movl 12(%ebp), %edx
    xorl %eax, %eax
    cmpl %edx, %ecx
    jle .L4
.L6:
    decl %ecx
    incl %edx
    incl %eax
    cmpl %edx, %ecx
    jg .L6
.L4:
    incl %eax
    movl %ebp, %esp
    popl %ebp
    ret
```

Com base no código Assembly à esquerda, preencha os espaços em branco no código correspondente em C. Apenas pode usar as variáveis `x`, `y` e `result` nas expressões (*não use nomes de registos!*) (**escreva a função completa na folha A4**).

```
int loop(int x, int y){
    int result;

    for(_____; _____; result++){
        _____;
        _____;
    }

    _____;

    return result;
}
```