

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 <code>cast</code> de uma variável do tipo <code>int</code> para uma do tipo <code>float</code> altera o padrão de bits da variável..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 2) Em C, o <code>cast</code> implícito em determinadas situações de variáveis com sinal para valores sem sinal pode levar a <i>bugs</i> no programa..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 3) Admita um <code>int x</code> com valor <code>0x01234567</code> e um valor dado por <code>&x</code> de <code>0x100</code> . Logo, o valor presente no byte <code>0x100</code> é <code>0x67</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 4) Em C, se tivermos uma variável <code>x</code> do tipo <code>short</code> com o valor <code>0x1234</code> , o valor <code>-0x1234</code> pode ser obtido através de <code>~x + 1</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 5) Em C, a adição de duas variáveis <code>u</code> e <code>v</code> do tipo <code>int</code> tem como resultado $(u+v) \bmod 32$ | <input type="checkbox"/> | <input type="checkbox"/> |
| 6) Em C, é garantido que o resultado de uma divisão inteira por 2^k , obtida através de <code>u >> k</code> , é correctamente arredondado se <code>u < 0</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 7) Admita que <code>ptr</code> é uma variável do tipo <code>char*</code> . Então, a expressão <code>(int*)ptr + 7</code> avança 28 bytes na memória..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 8) Em Assembly, a instrução <code>movb (%esi), (%edi)</code> permite copiar um byte para uma nova posição de memória numa única instrução | <input type="checkbox"/> | <input type="checkbox"/> |
| 9) Em Assembly, o resultado das instruções de salto condicional depende do valor dos bits do registo EFLAGS..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 10) Admita que <code>%edi</code> e <code>int *ptr</code> armazenam o endereço do inteiro <code>x</code> . Então, <code>movl \$1, (%edi)</code> é o equivalente a <code>*ptr = 1</code> em C | <input type="checkbox"/> | <input type="checkbox"/> |
| 11) Os parâmetros de uma função não podem ser acedidos usando o registo <code>%esp</code> em vez do <code>%ebp</code> como base do endereçamento | <input type="checkbox"/> | <input type="checkbox"/> |
| 12) Admita <code>0xF000</code> e <code>0x0100</code> em <code>%edx</code> e <code>%ecx</code> , respetivamente. <code>leal (%edx, %ecx, 4), %esi</code> armazena em <code>%esi</code> o valor <code>0xF400</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 13) Em IA32 é usada a <i>stack</i> para armazenar o valor de retorno de uma função, à semelhança do que acontece com o seu endereço de retorno..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 14) Admita que o valor de <code>%esp</code> é <code>0x100C</code> . A execução da instrução <code>ret</code> coloca o valor de <code>%esp</code> em <code>0x1010</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 15) Os registo <code>%eax</code> é local a cada uma das funções, o que dispensa qualquer cuidado no seu uso entre invocações de funções..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 16) Admita a matriz global <code>short int m[5][3]</code> . Em Assembly, acedemos ao valor de <code>m[3][0]</code> avançando 18 bytes a partir de <code>m</code> | <input type="checkbox"/> | <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 12 bytes.. | <input type="checkbox"/> | <input type="checkbox"/> |
| 18) É <u>sempre</u> possível diminuir o tamanho de um estrutura alinhada alterando a ordem dos seus campos..... | <input type="checkbox"/> | <input type="checkbox"/> |
| 19) É possível redimensionar, com a função <code>realloc</code> , o tamanho um vetor de inteiros <code>vec</code> declarado estaticamente com <code>int vec[10]</code> | <input type="checkbox"/> | <input type="checkbox"/> |
| 20) A possibilidade de existirem diversas referências para a mesma posição de memória dificulta a optimização efectuada 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.**

Considere o código da função `sum` ao lado que pretende somar os elementos de um vetor `a`. O número de elementos do vetor é passado no parâmetro `unsigned int length`.

Quando invocada com o valor 0 no argumento `length`, a função deveria retornar 0.0. No entanto, é gerado um erro de acesso à memória.

```
float sum(float a[], unsigned int length){
    int i;
    float result = 0.0;

    for(i=0; i<= length-1; i++)
        result += a[i];
    return result;
}
```

[1v] a) Explique detalhadamente porque o erro acontece.

[1v] b) Demonstre como o código poderia ser corrigido. **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 int code;
    long int start;
    char raw[3];
    double data;
} OldSensor;
```

```
typedef struct {
    short int code;
    short int start;
    char raw[5];
    short int sense;
    short int ext;
    double data;
} NewSensor;
```

[1.5v] **a)** Indique o alinhamento dos campos de uma estrutura do tipo `OldSensor`. 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 `OldSensor` 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 o seguinte fragmento de código em C, respeitando as declarações iniciais das estruturas.

```
void xpto(OldSensor *oldData){
    NewSensor *newData;

    /* zeros out all the space of oldData */
    bzero((void *)oldData, sizeof(OldSensor));

    oldData->code = 0x104f;
    oldData->start = 0x80501ab8;
    oldData->raw[0] = 0xe1;
    oldData->raw[1] = 0xe2;
    oldData->raw[2] = 0x8f;
    oldData->data = 1.5;

    newData = (NewSensor *) oldData;
    ...
}
```

Admita que após estas linhas de código começamos a aceder aos campos da estrutura `NewSensor` através da variável `newData`. Indique, em hexadecimal, o valor de cada um dos campos de `newData` indicados a seguir. Tenha em atenção a ordenação dos bytes em memória em Linux/IA32!

- a) `newData->code` = 0x_____
- b) `newData->raw[0]` = 0x_____
- c) `newData->raw[2]` = 0x_____
- d) `newData->raw[4]` = 0x_____
- e) `newData->sense` = 0x_____

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

Admita a existência de um vetor `v` preenchido com um número arbitrário de inteiros positivos e cuja última posição preenchida tem o valor -1. O código seguinte em C determina a soma de todos os seus valores positivos.

```
void sum_elements(int *v, int *sum){
    int i, val;
    *sum = 0;
    for(i = 0; i < vec_length(v); i++){
        get_element(v, i, &val);
        *sum += val;
    }
}
```

```
void get_element(int *v, int i, int *val){
    *val = v[i];
}

int vec_length(int *v){
    int i=0, length=0;
    while(v[i++] != -1)
        length++;
    return length;
}
```

Reescreva a função `sum_elements` em C usando as técnicas de optimização estudadas nas aulas. **Indique claramente cada uma das optimizações 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.**

Considere o seguinte código Assembly:

```
loop_func:
    pushl %ebp
    movl %esp, %ebp
    pushl %ebx
    movl 8(%ebp), %ecx
    movl 12(%ebp), %ebx
    movl $1, %eax
    cmpl %ecx, %ebx
    jle .L4
.L6:
    leal (%ebx, %ecx), %edx
    imull %edx, %eax
    shll %ecx
    cmpl %ecx, %ebx
    jg .L6
.L4:
    movl $0, %edx
    idivl %ebx
    popl %ebx
    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 `a`, `b` e `result` nas expressões (*não use nomes de registos!*) (**escreva a função completa na folha A4**).

```
int loop_func(int a, int b){
    int result = _____;

    while(_____) {
        _____;
        _____;
    }

    _____;

    return result;
}
```