- Autorizada apenas a consulta da folha de consulta oficial.
- A infracção implica, no mínimo, a **ANULAÇÃO** da prova.
- Quando omissa a arquitectura, considere Linux/IA32.

Versão: A Nota mínima: **7.5/20 valores** / Duração: 120 minutos

Número:_____ **Nome:**_____

Responda aos grupos II, III, IV, V e VI 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 *cast* de um `unsigned int x` com um valor positivo para um `int` pode resultar na atribuição de um valor negativo a `x`	☐	☐
2) Em C, a avaliação de expressões que usem variáveis com e sem sinal interpreta todas as variáveis como tendo valores sem sinal	☐	☐
3) Admita um `unsigned short x` com valor `0x0123` e um valor dado por `&x` de `0x200`. Logo, o valor presente no byte `0x200` é `0x01`	☐	☐
4) Em C, se tivermos um `short x = 0x1234`, o resultado da operação `x && 0x0F0F` é `0x1030`	☐	☐
5) Em C, a adição de duas variáveis `x` e `y` do tipo `long long int` pode resultar num valor menor do que os armazenados em `x` ou `y`	☐	☐
6) Em C, a operação `u << k` tem como resultado `u * 2`^{k}, para valores inteiros de `u` com ou sem sinal e `k > 0`	☐	☐
7) Em C, admita o apontador `char *ptr`. Então, a expressão `(int*)ptr + 2` avança 8 bytes na memória	☐	☐
8) Em IA32, a instrução `pushl %eax` move para o endereço armazenado em `%esp` o valor de `%eax`, apagando o valor em `%eax`	☐	☐
9) Em IA32, o resultado da instrução `call label_xpto` depende do valor dos bits do registo EFLAGS	☐	☐
10) Admita que `%edi` e `int *ptr` armazenam o endereço do inteiro `x`. Então, `movl $1,(%edi)` é o equivalente a `*ptr = 1` em C	☐	☐
11) Em IA32, começando pelo topo da stack, encontramos primeiro os parâmetros de uma função seguidos pelo seu endereço de retorno	☐	☐
12) Em IA32, a instrução `leal (%eax,%eax,4),%eax` pode ser usada para multiplicar por quatro o valor presente em `%eax`	☐	☐
13) Em IA32, a instrução `idivb %cl` assume que o dividendo se encontra em `%ax`, deixando o quociente em `%al` e o resto em `%ah`	☐	☐
14) Admita que o valor de `%esp` é `0x1000`. A execução da instrução `call` coloca o valor de `%esp` em `0xFFC`	☐	☐
15) De acordo com a convenção usada em Linux/IA32, a responsabilidade de salvaguarda e restauro de `%ebx` é da função invocada	☐	☐
16) Admita a matriz global `short m[5][6]`. Em Assembly, acedemos ao valor de `m[2][3]` avançando 30 bytes a partir de `m`	☐	☐
17) Uma estrutura alinhada de acordo com as regras estudadas com um vector de 2 `char`, 1 `int` e 1 `short` (por esta ordem) ocupa 12 bytes	☐	☐
18) Na hierarquia de memória à medida que nos afastamos do CPU, a capacidade de armazenamento aumenta mas diminui a performance	☐	☐
19) É possível retornar como valor de saída de uma função o endereço de um bloco de dados reservado internamente pela função `malloc`	☐	☐
20) A possibilidade de existirem diversas referências para a mesma posição de memória dificulta as optimizações efectuadas pelo compilador	☐	☐

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

```
unsigned int x = 0xDEADBEEF;
unsigned short y = 0xFFFF;
int z = -1;

if (x > (short)y)
    printf("Hello");
if (x > z)
    printf("World");
```

Considere o excerto de código acima. Qual o output produzido? **Justifique a sua resposta.**

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

```
struct s1{
    short a;
    char b[3];
    long long c[3];
    union u1 *d;
    char e;
};
```

```
struct s2{
   struct s1 f;
   struct s1 *g;
   struct s2 *h;
   long long i;
   short j[3];
};
```

```
union u1 {
  int k;
  struct s2 l;
  struct s1 *m;
};
```

[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`.**

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

```
int fun1(int ap, int bp) {
    int a = ap;
    int b = bp;
    return *(&a + b);
}

int fun2(int *ap, int bp) {
    int *a = ap;
    int b = bp;
    return *(a + b);
}

int fun3(int ap, int *bp) {
    int a = ap;
    int b = *bp;
    return *(&a + b);
}
```

Indique qual das funções em C apresentadas ao lado gera o seguinte código em Assembly quando compilada. **Justifique a sua resposta.**

Nota: as funções não fazem necessariamente algo útil.

```
pushl %ebp
movl %esp,%ebp
subl $8,%esp
movl 12(%ebp),%edx
movl 8(%ebp),%eax
movl %eax,-4(%ebp)
movl (%edx),%eax
leal -4(%ebp),%edx
movl (%eax,%edx),%eax
movl %ebp,%esp
popl %ebp
ret
```

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

Admita a seguinte função em C que recebe como primeiro parâmetro um vetor de *strings* `v`, como segundo parâmetro o endereço de um inteiro `res` no qual a função armazena o resultado final e como terceiro parâmetro um valor inteiro `x` que é usado nos cálculos.

```
void func_strs(char *v[40], int *res, int x){
    int i, j;

    *res = 0;
    for(i = 0; i < 40; i++){
       if(strlen(v[i])) > ((i+1)*10))
          inc_res(res,strlen(v[i])/4*x);
       else{
          for(j=0; j < strlen(v[i]); ++j){
             inc_res(res,2*x+10*i+2*v[i][j]);
       }
}
```

```
void inc_res(int *res, int value){
   *res += value;
}
```

Reescreva a função `func_strs` 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 VI – Responda numa folha A4 separada que deve assinar e entregar no final do exame.**

```
func:
  pushl %ebp
  movl %esp,%ebp
  pushl %esi
  pushl %ebx
  movl 8(%ebp),%ebx
  movl 12(%ebp),%esi
  movw $0,%dx
  xorl %ecx,%ecx
  cmpl %ebx,%ecx
  jge .L3
.L1:
  movw (%esi,%ecx,2),%ax
  cmpw %dx,%ax
  jle .L2
  movw %ax,%dx
.L2:
  incw %dx
  incl %ecx
  cmpl %ebx,%ecx
  jl .L1
.L3:
  movw %dx,%ax
  popl %ebx
  popl %esi
  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 `n`, `a`, `i` e `x` nas expressões (*não use nomes de registos!*) (**escreva a função completa na folha A4**).

```
short func(int n, short *a){
  int i;
  short x = ____;

  for(____ = ____;____;____){

    if (____)
       ____ = ____;

    ____;
  }

  return ____;
}
```