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 `int x` com um valor negativo para um `unsigned int` resulta sempre na atribuição de um valor positivo a `x`	☐	☐
2) Em C, as variáveis com sinal usam mais um bit para armazenar se o valor é positivo ou negativo do que as suas equivalentes sem sinal	☐	☐
3) Em C, admita as variáveis "`int x=0xABCD;`" e "`char *ptr=&x`". Logo, "`printf("%hhX",*(ptr+1));`" imprime o valor `0xCD`	☐	☐
4) Em C, se tivermos um `short x = 0x1234`, o resultado da operação `x && 0x0F0F` é `0x0204`	☐	☐
5) Em C, a adição de duas variáveis `x` e `y` do tipo `unsigned int` pode resultar num valor menor do que os armazenados em `x` ou `y`	☐	☐
6) Em C, "`x >> 2`" aplica um deslocamento aritmético para a direita se `x` for do tipo `unsigned int` e um lógico se `x` for do tipo `int`	☐	☐
7) Em C, admita o vetor "`short vec[5];`". A função `realloc` permite alterar o tamanho de `vec` para armazenar mais elementos	☐	☐
8) Em IA32, a instrução `pushl %eax` move para o endereço armazenado em `%esp` o valor de `%eax`, mantendo inalterado o valor de `%esp`	☐	☐
9) Em IA32, o valor final de `%ebx` após a instrução `cmovg %eax,%ebx` depende do valor dos bits do registo EFLAGS	☐	☐
10) Em IA32, "`testl $-1,%ecx`" seguido de "`jz xpto`" permite saltar para a etiqueta `xpto` se o valor de `%ecx` for zero	☐	☐
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 cinco o valor presente em `%eax`	☐	☐
13) Em IA32, é possível usar "`shll $3, %eax`" seguido de "`negl %eax`" para multiplicar por -8 o valor de `%eax`	☐	☐
14) Em IA32, a instrução "`pushl %eax`" é equivalente a "`subl $4,%esp`" seguido de "`movl %eax,(%esp)`"	☐	☐
15) Em IA32, a *stack* é usada para suportar o retorno de um valor de saída de 64 bits de uma função	☐	☐
16) O sistema operativo executa periodicamente uma desfragmentação da *heap* para melhorar o desempenho dos programas em C	☐	☐
17) A fragmentação interna dos blocos reservados na *heap* é consequência das regras de alinhamento e *overhead* da gestão dos blocos	☐	☐
18) Na hierarquia de memória à medida que nos afastamos do CPU, a capacidade de armazenamento aumenta, mas diminui a performance	☐	☐
19) O bloco de código "`for(i=0;i<N;i++)for(j=0;j<M;j++)sum+=m[i][j];`" não exibe boa localidade espacial nem temporal	☐	☐
20) A possibilidade de existirem diversas referências para a mesma posição de memória dificulta as otimizações efetuadas pelo compilador	☐	☐

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

[1.5v] **a)** Considere o seguinte excerto de código. Qual o output produzido? **Justifique a sua resposta.**

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

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

[1.5v] **b)** Considere o seguinte excerto de código. Em que zonas do espaço de endereçamento do processo são armazenados os bytes necessários para acomodar as variáveis `i`, `j` e `k`? **Justifique a sua resposta.**

```
int i;
int main(){
   int j;
   int *k = (int *)malloc(sizeof(int));
}
```

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

```
struct s1{
    short a;
    struct s2 b;
    long long c[3];
    union u1 d;
    char e;
};
```

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

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

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`.**

[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
```

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

```
fun:
  pushl %ebp
  movl %esp,%ebp
  movl 16(%ebp),%ecx
  movl 12(%ebp),%eax
  movl 8(%ebp),%edx
  cmpl %ecx,%edx
  jl .L1
.L2:
  addl %edx,%eax
  decl %edx
  cmpl %ecx,%edx
  jge .L2
.L1:
  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`, `z`, `i` e `result` nas expressões (*não use nomes de registos!*) (**escreva a função completa na folha A4**).

```
int fun(int x, int y, int z){

  int i, result = __________;

  for(i=____; ____; ____){
    result = ________;
  }

  return ________;
}
```

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

Admita a seguinte função em C que recebe no primeiro parâmetro o endereço dum vetor de *strings* `v`, no segundo parâmetro `size` o número de elementos armazenados nesse vetor e no terceiro parâmetro `res` o endereço dum inteiro no qual a função armazena o resultado final.

```
void func_strs(char *v, int size, int *res){
    int i, j;

    *res = 0;
    for(i = 0; i < size; i++){
       if(strlen(v[i])) > ((i+1)*8){
          for(j=0; j < strlen(v[i]); ++j){
             *res += inc_res(2*size,v[i][j]);
          }
       }
   }
}
```

```
int inc_res(int delta, int value){
   return (10*value)/(5*delta);
}
```

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