Nº: _____ Nome:_____

As duas perguntas seguintes destinam-se a ajudar à avaliação dos trabalhos 1 e 2. Se não está a realizar a avaliação laboratorial este ano, por já ter obtido uma nota laboratorial em ano anterior, não deve responder a estas duas perguntas, mas escrever neste espaço "Não estou a realizar a parte laboratorial em 2020/21" e rubricar.

Se responder a estas perguntas, a nota final do trabalho será ajustada de acordo com o descrito nas regras de avaliação da cadeira no CLIP

QTPC1

a) No script Shell que escreveu para o TPC1 deverá ter usado um comando parecido com o seguinte:

```
find $1 -type f -exec sha1sum {} \; | sort -k 2,2 > temp
```

Explique o que faz este comando. Se o trabalho que entregou não usou uma solução parecida com a indicada, explique como conseguiu um efeito equivalente ao obtido pelo comando *find* acima.

b) Considere o seguinte programa em C que usa a biblioteca *stdio* para copiar um ficheiro de texto para o *stdout*. Insira o código que falta

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLINE 128
int main( int argc, char *argv[] ){
       FILE *f1;
       char buf[MAXLINE];

       if( argc != 2 ){
                    printf("Usage: %s file_to_copy\n", argv[0]);
                    exit(2);
       }
       f1 = fopen( _______________________________);
       if( f1 == NULL){
              perror("file_to_copy");
              exit(1);
       }
       while (fgets( buf, MAXLINE, f1 ) != NULL){



       };
       fclose(f1);
        return 0;
}
```

QTPC2

Para o TPC2 foi-lhe proposto que completasse o código de um simulador (escrito na linguagem C) de uma arquitetura composta por um CPU muito simples e uma memória.

Instruções máquina do processador:

O processador usa palavras de 32 bits e tem 16 registos (r0 a r15). Tem ainda um registo como *Program Counter* (PC) e outro para a instrução a executar (IR). A memória está organizada em palavras de 32 bits por endereço, sendo cada endereço de 12 bits. Há também duas *flags*

- ZERO contém um valor diferente de zero quando a última operação aritmética deu zero e 0 quando a última operação aritmética não deu zero
- POSITIVO contêm um valor diferente de zero quando a última operação aritmética deu um resultado >= 0 e zero quando a última operação aritmética deu um resultado < 0

As instruções têm tamanho fixo de 32 bits, sendo os 8 mais significativos para o código de operação, 12 bits para especificar os registos que contêm os dois operandos (*registo1* e *registo2*) e o resultado de instruções aritméticas (*registo3*) e os restantes 12 para o endereço, se existir.

31 — 24	23 — 20	19 — 16	15 — 12	11 — 0
Código da instrução	Registo 1	Registo 2	Registo 3	Endereço

Pretende-se acrescentar uma instrução máquina com o código 0x0F em que

31 — 24	23 — 20	19 — 16	15 — 12	11 — 0
00001111	Registo 1	Registo 2	Registo 3	ignorado

em que o conteúdo do registo indicado nos bits 23 a 20 é multiplicado pelo conteúdo do registo especificado nos bits 19 a 16, sendo o resultado guardado no registo especificado nos bits 15 a 12. Os bits 11 a 0 são ignorados

```
#include <stdio.h>
#define NREGS 16

extern int Mem[];
extern int Regs[];
void dorun(){
unsigned int pc;  // program counter or intruction pointer
unsigned int ir;  // instruction register

unsigned int opcode;
unsigned int reg1;
unsigned int reg2;
unsigned int reg3;
int val;
unsigned int address;
unsigned char zero;
unsigned char positivo;

pc = 0;
while( 1 ) {
       ir = Mem[pc];                  // FETCH
       opcode = ir >> 24;             // DECODE
       reg1 = (ir & 0x00f00000) >> 20;
       reg2 = (ir & 0x000f0000) >> 16;
       reg3 =   (ir & 0x0000f000) >> 12;
       address =  ir & 0x00000fff;
       val = signExtension20To32(ir & 0x00ffffff);

      switch( opcode ){              // EXECUTE
            case 0x00:   /* HALT */
                printf("HALT instruction executed\n");
                return;

            // Omitido o Código C que implementa as outras instruções
```

```
            // …
            // acrescentar aqui o código que simula a instrução descrita acima









        default:
            printf("Invalid instruction!\n");
            return;
    }
  }
}
```