21 (2 val.) Implemente a seguinte função C que concatena a string *str* ao final da string *dst* e retorna o novo comprimento da string *dst*. Assuma que *dst* tem espaço suficiente para guardar a concatenação das duas strings. Na sua implementação **não pode** utilizar as funções *strcat* e *strcpy* da biblioteca standard do C. **Pode** utilizar outras, como *strlen*.

```
int strconcat(char* dst,  char* str) {

        int dst_size = strlen(dst);
        int str_size = strlen(str);

         for (int i = 0; i <= str_size; i++)
                dst[i + dst_size] = str[i];

         return str_size + dst_size;

}
```

22 (2 val.) Complete a implementação do seguinte programa em assembly para Linux IA-32 que coloca no registo EAX o **maior** valor que faz parte do vetor *values*.

```
.data
values:  .int …                  # Um vetor de valores inteiros
LEN = ( . - values)/4
.text
.global _start
_start:

    mov $0, %ecx
    cmp $LEN, %ecx
    je end_loop   # just to ensure that the vector has values, was not mandatory

    mov (values), %eax # first element

my_loop:
    inc %ecx
    cmp $LEN, %ecx
    je end_loop
    mov values( , %ecx, 4), %edx
    cmp %eax, %edx
    jl my_loop
    mov %edx, %eax
    jmp  my_loop

end_loop:

    mov   $1,   %ebx          # Chamada ao sistema EXIT
    mov   $1,   %eax
    int     $0x80
```

Arquitetura de Computadores - Teste 1

30/4/2022 Duração 1h45 Sem consulta

As perguntas com múltiplas respostas têm de ser respondidas na folha de respostas anexa.
Respostas erradas descontam 1/5 da cotação da pergunta.

→ Considere um CPU que oferece um ISA com 30 instruções, oito registos de uso geral com 20 bits, entre outros registos, como o IP (or PC) e o IR, também de 20 bits.

1 (0.8 val) Quantos bits precisa para codificar o conjunto de instruções do CPU?
A) 5 **B)** 6 **C)** 7 **D)** 24 **E)** 23

2 (0.8 val) Qual é o maior valor sem sinal e o menor valor com sinal que pode guardar nos registos do CPU?
A) 2^{19}-1 e -2^{19} **B)** 2^{20} -1 e -2^{20} **C)** 2^{19} e -2^{19} **D)** 2^{20}-1 e -2^{19} **E)** 2^{19}-1 e -2^{20}

3 (0.8 val) Considerando uma memória endereçável ao byte (cada byte tem o seu endereço), qual é o tamanho máximo para a memória endereçável por um programa em execução neste CPU?
A) 2^{20} Bytes **B)** 2^{32} Bytes **C)** 2^{20} GBytes **D)** 2^{32} KBytes **E)** 2^{32} MBytes

4 (0.8 val) Assuma que cada instrução é codificada numa palavra de 20 bits e que os bits 17 e 16 (marcados com a letra **B** na palavra que se segue: XX**BB** XXXX XXXX XXXX XXXX) são utilizados para definir qual é o tipo da instrução: ALU, Mov, FPU ou Interrupt. Qual das seguintes expressões permite obter o tipo da instrução a partir do registo IR?
A) IR | 0x30000 **B)** IR >> 16 **C)** (IR >> 16) | 0x3 **D)** IR & 0x30000 **E)** (IR >> 16) & 0x3

→ Considere uma unidade aritmética e lógica (ALU) com operações ADD e SUB sobre inteiros de 4 bits (negativos representados na norma *complemento para dois*). Indique qual o valor correto das flags ZF (*zero flag*), CF (*carry flag*), OF (*overflow flag*) e SF (*sign flag*) quando são executadas as seguintes operações:

5 (0.8 val) -4 + 7
A) ZF = 0, CF = 1, OF = 0, SF=0 **B)** ZF = 0, CF = 1, OF = 1, SF=0 **C)** ZF = 0, CF = 0, OF = 1, SF=0
D) ZF = 0, CF = 1, OF = 0, SF=1 **E)** ZF = 0, CF = 1, OF = 1, SF=1

6 (0.8 val) -5 - 7
A) ZF = 0, CF = 1, OF = 0, SF=0 **B)** ZF = 0, CF = 0, OF = 1, SF=0 **C)** ZF = 0, CF = 1, OF = 1, SF=0
D) ZF = 0, CF = 1, OF = 1, SF=1 **E)** ZF = 1, CF = 1, OF = 1, SF=0

→ Considere a representação IEEE-754 32 bits (1 bit para o sinal, 8 para o expoente e 23 para a mantissa) para valores em vírgula flutuante.

7 (0.8 val) Qual é a representação IEEE-754 32 bits para o valor -21.0?
A) 10000010001010000000000000000000 **B)** 01000001101010000000000000000000
C) 10000010001010000000000000000000 **D)** 11000001101010000000000000000000

8 (0.8 val) Que valor está codificado na seguinte representação IEEE-754 32 bits 01000000111010000000000000000000?
A) 7.25 **B)** 5.4004 **C)** 4.0625 **D)** 7.5 **E)** 6.25

9) (0.8 val) Considere duas variáveis — ***int*** *i* e ***float*** *f* — e que a representação binária do valor atribuído a ambas é 10000000000000000000000000000010. Qual das seguintes expressões é verdadeira?
A) i == f **B)** i < f **C)** i > f **D)** i >> f **E)** i == (**int**) f

10 (0.8 val) Complete a afirmação seguinte de forma a que seja correta. Aquando da execução de programa….

A) os dados e o código (o programa) são guardados na memória. O código é guardado em binário e os dados são guardados em diferentes formatos. Por exemplo, os valores inteiros são representados em binário e as letras são representadas pelos seus próprios símbolos, como 'a'.

B) os dados e o código (o programa) são guardados em binário na memória

C) os dados e o código (o programa) são guardados na memória. O código é guardado em hexadecimal e os dados são guardados em binário.

D) os dados são guardados na memória e o código vai sendo lido do disco para os registos do CPU.

E) os dados e o código (o programa) são guardados em binário na memória. O código é guardado em binário e os dados são guardados em hexadecimal.

11 (0.8) val. Para efetuar uma operação sobre um dispositivo de entrada/saída, como um teclado ou um disco, um programa em execução

A) tem de interagir diretamente com o dispositivo efetuando uma chamada ao dispositivo.

B) tem de chamar o Sistema Operativo efetuando uma chamada ao sistema, se o dispositivo for um dispositivo de entrada, como um teclado, mas pode interagir diretamente com o dispositivo, se este for um dispositivo de saída, como o monitor.

C) tem de chamar o Sistema Operativo efetuando uma chamada ao sistema, se o dispositivo for um dispositivo de saída, como um monitor, mas pode interagir diretamente com o dispositivo, se este for um dispositivo de entrada, como o teclado.

D) pode escolher se faz uma chamada ao sistema ou se interage diretamente com o dispositivo.

E) tem de chamar o Sistema Operativo efetuando uma chamada ao sistema.

→ Considere um programa com a seguinte sequência de linhas de código C:

```
1.  int v[] = { 9, 6, 3 };      4.  int **f = &a;        7.  int c = *b;
2.  int *a = v;                 5.  *b = *b + 1;         8.  int d = *a + 2;
3.  int *b = a;                 6.  a = a+1;             9.  a--;
```

12 (0.8 val) Qual é o valor das variáveis **c** e **d** depois de executada a linha 9?

A) c = 10, d = 8 **B)** c = 9, d = 8 **C)** c = 9, d = 6 **D)** c = 10, d = 6 **E)** c = 10, d = 5

13 (0.8 val) Qual é o valor das expressões ***a** e ****f** depois de executada a linha 9?

A) *a = 9, **f = 6 **B)** *a = 9, **f = 9 **C)** *a = 10, **f = 10 **D)** *a = 10, **f = 6 **E)** *a = 10, **f = 9

14 (0.8 val) Considere o seguinte extrato de código C (o código ASCII de 0 é 48):

```
float f1 = 6.45;
int i1 = (int)f1;
float f2 = (float)i1;
char c = i1+'0';
printf ("f1=%.2f, i1=%d, f2=%.2f, c=%c\n", f1, i1, f2, c);  // %.2f -float com 2 casas decimais,
                                                            // %d - inteiro decimal, %c - caracter
```

O que é que aparece escrito na consola?

A) f1=6.45, i1=6, f2=6.00, c=6 **B)** f1=6.45, i1=6, f2=6.45, c=6 **C)** f1=6.45, i1=6, f2=6.00, c=54
D) f1=6.45, i1=1087268454, f2=6.00, c=54 **E)** f1=6.45, i1=1087268454, f2=1.00, c=6

15 (0.8 val) Como é que é guardado na memória de uma arquitetura Little Endian o valor $130_{(10)}$ a 16 bits?

A) 01000001 00000000 **B)** 00000000 01000001 **C)** 00000000 10000010
D) 10000010 00000000 **E)** 00000010 01000000

→ Considere a seguinte declaração de dados em assembly do IA-32 para ambiente Linux.

```
msg:    .ascii "Hello"
```

16 (0.8 val) Escolha a opção que compara o primeiro elemento de msg com '\n'.

A)
```
mov $msg,%ebx
mov (%ebx),%eax
cmp $'\n', %eax
```

B)
```
mov $msg,%ebx
mov (%ebx),%al
cmp $'\n', %al
```

C)
```
mov $msg,%ebx
mov (%ebx),%al
cmp %al, $'\n'
```

D)
```
mov msg,%ebx
mov (%ebx),%eax
cmp $'\n', %eax
```

E)
```
mov msg,%ebx
mov (%ebx),%al
cmp $'\n', %al
```

17 (0.8 val) Qual dos seguintes extratos de código pode ser usado como base para percorrer o vetor?

A)
```
l1: mov $msg, %ebx
    … # teste fim do vetor
    … # aceder a (%ebx)
    inc  %ebx
    jmp l1
```

B)
```
    mov $0, %ecx
l1: … # teste  fim do vetor
    … # aceder a msg(, %ecx, 1)
    inc %ecx
    jmp l1
```

C)
```
    mov $0, %ecx
l1: … # teste  fim do vetor
    … # aceder a (, %ecx, msg)
    inc %ecx
    jmp l1
```

D)
```
    mov $msg, %ecx
l1: … # teste fim do vetor
    … #  aceder  a msg(, %ecx, 1)
    inc %ecx
    jmp l1
```

E)
```
    mov $0, %ecx
l1: … # teste  fim do vetor
    … # aceder  a msg(, %ecx, 4)
    inc %ecx
    jmp l1
```

→ Considere a seguinte subrotina em assembly do IA-32 para ambiente Linux.

```
sub: push %ebp
     mov %esp, %ebp
     mov 8(%ebp), %eax
     cmp 16(%ebp), %eax
     jg fim
     mov 12(%ebp), %eax
fim: pop %ebp
     ret
```

18 (0.8 val) Qual é o código correspondente à chamada C sub(14, 34, 4)?

A)
```
pushl $4
pushl $34
pushl $14
call sub
add $12, %eax
```

B)
```
pushl $14
pushl $34
pushl $4
call sub
add $12, %esp
```

C)
```
pushl $14
pushl $34
pushl $4
call sub
sub $12, %esp
```

D)
```
pushl $4
pushl $34
pushl $14
call sub
sub $12, %esp
```

E)
```
pushl $4
pushl $34
pushl $14
call sub
add $12, %esp
```

19 (0.8 val) Qual é será o valor do registo EAX após a chamada a sub(14, 34, 4)?

A) 34 **B)** 3 **C)** 4 **D)** 14 **E)** 0

20 (0.8 val) Considere a seguinte subrotina em assembly do IA-32 para ambiente Linux.

```
1.  sub: push %ebp
2.       mov %esp, %ebp
3.       push %ebx
4.       mov $0, %eax
5.       pop %ebp
6.       ret
```

Qual é o resultado de executar call sub?

A) The EAX register is set to 0 and there is no runtime error.
B) A runtime error, as the stack does not have the correct contents when executing the instruction at line 5. We should add instruction mov %esp, %ebp between lines 4 and 5.
C) A runtime error, as the stack does not have the correct contents when executing the instruction at line 5. The instruction pop %ebp should be replaced by pop %ebx
D) A runtime error, as the stack does not have the correct contents when executing the instruction at line 5. We should add instruction pop %ebx between lines 4 and 5.
E) A runtime error, as the stack does not have the correct contents when executing the instruction at line 5. We should add instruction pop %ebx between lines 5 and 6.