1. Classificar

Qual o maior número inteiro, sem sinal, representável num registo de 32 bits?

- 2^{32}
- $2^{32}+1$
- $2^{31}+1$
- $2^{32}-1$
- $2^{31}-1$
- 2^{31}

(Cotação: 1.0 pontos)

2. Classificar

Represente o número -48 em binário (complemento para 2) num registo de 8 bits. (Represente o resultado com o prefixo `0b` , por exemplo `0b00000000`)

(Cotação: 1.0 pontos)

3. Classificar

O número de 32 bits `0x12345678` representado como uma sequência de bytes com ordenação **little endian** é

- `0x87654321`
- `0x78` , `0x56` , `0x34` , `0x12`
- `0x12` , `0x34` , `0x56` , `0x78`
- `0x21` , `0x43` , `0x65` , `0x87`
- `0x87` , `0x65` , `0x43` , `0x21`

(Cotação: 1.0 pontos)

4. Classificar

Sabe-se que estão presentes em memória, a partir do endereço `0x10008000`, a seguinte sequência de bytes `0xff`, `0xff`, `0xff`, `0x00`, `0x00`, `0x00`, `0x00`, `0xff` Sabendo que este conjunto bytes representa uma sequência de números inteiros de 32 bits, com sinal e ordenação **big endian**, indique qual das seguintes opções é a sequência correcta (as respostas estão em décimal)

- Os números são inválidos
- 16777215, -16777214
- 15, 15, 15, 0, 0, 0, 0, 15
- -256, 255
- 15, -15

(Cotação: 1.0 pontos)

5. Classificar

Num processador MIPS, suponha que o registo `$t0` contém o valor `1` quando se executam as instruções

```
sll $t1, $t0, 3
add $t1, $t1, $t0
```

Qual o valor do registo `$t1` após a execução destas instruções?

- -8
- Infinito
- 7
- 9
- 8

(Cotação: 1.0 pontos)

6. Classificar

Suponha que `0x41840000` representa um número em vírgula flutuante no formato IEEE 754, precisão simples. Represente este número em vírgula flutuante, em decimal. Um exemplo de uma resposta válida é `4.5e-1`. Note que `0.13` ou `15.78` não são números em vírgula flutuante regulares, pois à esquerda da vírgula tem de estar um único algarismo e ser diferente de zero.

(Cotação: 1.0 pontos)

7. Classificar

Considerando que `0x00000000` representa um número em vírgula flutuante no formato IEEE754, precisão simples, indique que número é este.

- Inf
- -0.0
- 0.0
- -Nan
- Nan
- -Inf
- Não é um número válido no formato IEEE754

(Cotação: 1.0 pontos)

8. Classificar

Represente o número `-7.3125` em vírgula flutuante no formato IEEE 754, precisão simples. Um exemplo de uma resposta válida é `0x12345678` . Não se esqueça de incluir o prefixo `0x` .

(Cotação: 1.0 pontos)

9. Classificar

Considere o seguinte ciclo:

```
float x = 0.0;
while (x < 0.9)
    x = x + 0.3;
```

- O número 0.3 tem dizima infinita quando convertido para binário.
- Se a representação dos números fosse exacta, o ciclo iria executar 2 vezes.
- Como o número 0.3 não tem representação exacta em binário, o número de vezes que o ciclo é executado pode não coincidir com a solução exacta.
- O número 0.9 não é representável exactamente em binário.

- [ ] Se a representação dos números fosse exacta, o ciclo iria executar 3 vezes.
- [ ] Se a representação dos números fosse exacta, o ciclo iria executar 4 vezes.

(Cotação: 1.0 pontos)

10. Classificar

Qual das opções implementa a pseudo-instrução `ble $t0, $t1, L` ?

```
slt $at, $t1, $t0
bne $at, $zero, L
```

```
slt $at, $t1, $t0
beq $at, $zero, L
```

```
slt $at, $t0, $t1
beq $at, $zero, L
```

```
slt $at, $t0, $t1
bne $at, $zero, L
```

(Cotação: 1.0 pontos)

11. Classificar

Considere o troço de código

```
        li $t0, 0x000100ff
L:      bge $t0, $zero, L
        addi $t0, $t0, -1
```

Indique quais das seguintes afirmações são verdadeiras (podem existir várias).

- [ ] O ciclo é executado menos de dez mil vezes.
- [ ] O código acima contém 2 pseudo-instruções e 1 instrução real.
- [ ] A pseudo-instrução `li` pode ser traduzida numa única instrução real.
- [ ] A instrução `addi` é sempre executada independentemente do branch ser tomado ou não.
- [ ] O códido termina com t0 igual a -2.
- [ ] O códido termina com t0 igual a -1.

- [ ] O programa não termina pois tem um ciclo infinito.
- [ ] O código máquina ocupa 12 bytes.
- [ ] O ciclo é executado mais de vinte mil vezes.
- [ ] A tradução deste código em instruções reais produz 5 instruções.

(Cotação: 1.0 pontos)

12. Classificar

Durante a execução de uma função, esta pode modificar livremente a maior parte dos registos. No entanto, a função terá de repor os valores originais de alguns dos registos antes de terminar (registos preservados). Indique quais são os registos que **não têm de ser preservados** por uma função.

- [ ] v0-v1
- [ ] s0-s7
- [ ] t0-t9
- [ ] a0-a3
- [ ] sp
- [ ] gp
- [ ] at

(Cotação: 1.0 pontos)

13. Classificar

Para guardar a string `"hello"` na pilha é necessário

- ( ) Decrementar o registo `$sp` de 8 bytes.
- ( ) A string não cabe na pilha pois o registo `$sp` só tem 32 bits.
- ( ) Decrementar o registo `$sp` de 6 bytes.
- ( ) Decrementar o registo `$sp` de 5 bytes.

Ajuda

(Cotação: 1.0 pontos)

14. Missão (quase) Impossível - T01E01 Classificar

Bom dia Jim. Uma equipa de dois programadores trabalharam num projecto para converter uma string arbitrária para maiúsculas. Um dos programadores implementou uma função "converte_char" que recebe como argumento um carácter e devolve-o convertido para maiúsculas (números, símbolos e letras maiúsculas ficam inalterados). O segundo programador ficou incumbido de fazer a função principal "converte_string". Esta recebe uma string como argumento e usa a função "converte_char" para converter cada um dos carácteres originais para maiúsculas. Infelizmente, o segundo programador desapareceu misteriosamente e o projecto não ficou terminado. O código seguinte foi tudo o que se encontrou no seu portátil:

```
converte_string:
        move $s0, $a0
L1:
        lb $a0, 0($s0)
        jal converte_char
        nop
        sb $v0, 0($s0)
        bne $v0, $zero, L1
        addi $s0, $s0, 1
        jr $ra
        nop
```

Jim, a sua missão, caso decida aceitá-la, é terminar o código e pô-lo a funcionar. Introduza esta função completa no espaço em baixo (não inclua a outra função "converte_char"). Como habitual, esta mensagem irá autodestruir-se assim que submeter o teste. Boa sorte.

Ajuda

(Cotação: 1.0 pontos)

15. Classificar

Considere a função `foo` em assembly MIPS

```
foo:
    addi $v0, $a0, 0
L:  lb $t0, 0($a0)
    bne $t0, $zero, L
    addi $a0, $a0, 1
    jr $ra
    sub $v0, $a0, $v0
```

O que faz esta função?

- Procura o carácter nulo numa string e devolve o seu endereço.
- Converte uma string de minúsculas para maiúsculas.
- Calcula o espaço ocupado por uma string, sem contar com o carácter nulo.
- Converte uma string de maiúsculas para minúsculas.
- Calcula o espaço ocupado por uma string, contando com o carácter nulo.

(Cotação: 1.0 pontos)

16. Classificar

Considere a função `foo` em assembly MIPS

```
foo:
L:  lb $t0, 0($a0)
    bne $t0, $zero, L
    addi $a0, $a0, 1
    jr $ra
    addi $v0, $a0, -1
```

O que faz esta função?

- Procura o carácter nulo numa string e devolve o seu endereço.
- Converte uma string de minúsculas para maiúsculas.
- Calcula o espaço ocupado por uma string, sem contar com o carácter nulo.
- Converte uma string de maiúsculas para minúsculas.
- Calcula o espaço ocupado por uma string, contando com o carácter nulo.

(Cotação: 1.0 pontos)

17. Classificar

Tendo em consideração que o código seguinte é executado sequencialmente em modo não privilegiado (*user mode*), indique quais as instruções que produzem excepções.

```
lui $a0, 0x8000
addi $a0, $a0, -1
lw $t0, 2($zero)
lb $t0, 2($zero)
lui $t1, 0xffff
ori $t1, $t1, 0xffff
addi $t1, $t1, 1
```

- [ ] `lui $a0, 0x8000`
- [ ] `addi $a0, $a0, -1`
- [ ] `lw $t0, 2($zero)`
- [ ] `lb $t0, 2($zero)`
- [ ] `lui $t1, 0xffff`
- [ ] `ori $t1, $t1, 0xffff`
- [ ] `addi $t1, $t1, 1`

Ajuda

(Cotação: 1.0 pontos)

Código máquina de algumas instruções MIPS:

Opcode	Rs	Rt	Rd	SA	Funct	Assembly
000000	Rs	Rt	Rd	00000	100000	add Rd, Rs, Rt
000000	Rs	Rt	Rd	00000	100010	sub Rd, Rs, Rt
000000	00000	Rt	Rd	amount	000000	sll Rd, Rt, 31
000000	Rs	00000	00000	00000	001000	jr Rs
001000	Rs	Rt	16 bit sign-extend			addi Rt, Rs, -32768
001101	Rs	Rt	16 bit zero-extend			ori Rt, Rs, 0x1234
000100	Rs	Rt	16 bit sign-extend			beq Rt, Rs, -32768
000101	Rs	Rt	16 bit sign-extend			bne Rt, Rs, -32768
100011	Rs	Rt	16 bit sign-extend			lw Rt, -32768(Rs)
101011	Rs	Rt	16 bit sign-extend			sw Rt, -32768(Rs)
000010	26 bit					j address
000011	26 bit					jal address

Numeração dos registos:

zero	at	v0 - v1	a0 - a3	t0 - t7	s0 - s7	t8 - t9	k0 - k1	gp	sp	fp	ra
0	1	2 - 3	4 - 7	8 - 15	16 - 23	24 - 25	26 - 27	28	29	30	31

20. Classificar

Descodifique o código máquina `0x2108ffff` .

- `add $t0, $t0, $at`
- `lw $t0, -1($t0)`
- `addi $t0, $t0, -1`
- `ori $t0, $zero, 0xffff`

(Cotação: 1.0 pontos)

21. Classificar

Determine o código máquina da instrução `add $t0, $t1, $t2` . Represente em hexadecimal e não se esqueça do prefixo `0x` .

(Cotação: 1.0 pontos)

22. Classificar

Determine o código máquina da instrução `lw $t0, 4($t1)` . Represente em hexadecimal e não se esqueça do prefixo `0x` .

(Cotação: 1.0 pontos)

Submeter teste