

Samuel Machado 2020219391

| | | | | |
|---|---|------------------------------|----------|-----------|
|  <p>FCTUC</p> | Arquitectura de Computadores ENGENHARIA INFORMÁTICA FACULDADE DE CIÉNCIAS E TECNOLOGIA UNIVERSIDADE DE COIMBRA – Exame de Época Normal – Parte I 1X de Junho de 201X Duração: 90 min. + 10 min. de tolerância | (a preencher pelo professor) | | |
| | | C | E | NR |
| | | | | |

Nome: _____ Número: _____

Notas Importantes:

A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante de ensino superior. Não serão admitidas eventuais tentativas de fraude, que provocarão a reprovação immediata, tanto do facilitador como do prevaricador.

Durante a prova pode consultar a bibliografia da disciplina (slides, livros, enunciados e material de apoio a trabalhos práticos). No entanto, não é permitido o uso de computadores, calculadoras ou qualquer outro dispositivo electrónico.

Este é um teste de escolha múltipla e deverá assinalar sem ambiguidades as respostas na tabela apresentada a baixo. Cada pergunta correctamente respondida vale cinco pontos; cada pergunta errada desconta dois pontos; e cada pergunta não respondida vale zero pontos. Uma nota total abaixo de zero pontos, conta como zero valores.

Respostas: (indicar resposta A, B, C ou D, debaixo do número da questão)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| B | C | A | A | A | D | B | B | B | D | D | B | C | D | D | A | C | A |

1. Suponha que deseja usar a função MIPS abaixo indicada para devolver a divisão inteira por 2^{32} do resultado da multiplicação de dois números, passados por parâmetro. Qual das seguintes instruções deveria substituir a linha representada por <...>?

- a. Uma instrução apenas não é suficiente para o efeito.
- b.** mfh \$v0
- c. mflo \$v0
- d. div \$a0, \$a1

| | |
|-------------|---|
| MIPS | <pre>multu \$a0, \$a1 <...> jr \$ra</pre> |
|-------------|---|

2. Considere o datapath de um processador MIPS e a execução de uma instrução. Indique qual das seguintes afirmações é VERDADEIRA:

- a. Se o valor imediato de uma instrução load for igual a zero (p. ex. lw \$r3, 0(\$r1)), então a instrução está *idle* na execução da etapa ALU.
- b. A instrução nop faz com que o processador esteja *idle* nas 5 etapas do ciclo de execução.
- c.** Na execução de uma instrução de jump incondicional o processador está *idle* nas etapas *Memory* e *Register Write*. * *rainha no exame 2022*
- d. Não existe nenhuma instrução em cuja execução o processador esteja *idle* em mais do que 1 etapa.

3. Considere a instrução em Assembly do MIPS dada por 'lw \$a0, 0(\$t5)'. Sabendo que os registos \$a0 e \$t5 são os registos #4 e #13, respectivamente, indique qual das seguintes instruções representa a codificação da instrução anterior:

- a.** 0x8DA40000
- b. 0x8D840000
- c. 0x8C8D0000
- X** 0x8C04000D *

loads e stores não são tipo I ($6+5+5+16$)

6 | 5 | 5 | 16
 100011 | 01101 | 00100 | 0x0000
 8 D A 4 0 0 0 0 *

4. Considere o seguinte código em Assembly do MIPS, que pretende implementar a seguinte instrução equivalente em linguagem C:

```

MIPS
...
li      $t1,1
li      $t2,1000
<...>
lw      $t0,0($t3)
ciclo:
<...>
    mul   $t1,$t1,$t0
    bgt  $t1,$t2,fim_ciclo
    addi  $t0,$t0,-1
    b     ciclo
fim_ciclo:
...

```

C
 $\text{fica enquanto } > 0$
 $\$t1=1;$
 $\text{for } (\$t0=A; \$t0>0; \$t0--)$
 $\{$
 $\quad \$t1=\$t1*\$t0;$
 $\quad \text{if } (\$t1>1000)$
 $\quad \quad \text{break};$
 $\}$

sai quando
 ≤ 0 *

Escolha das opções disponíveis aquela que correctamente representa o par de instruções <...> em falta no código MIPS

- a. 'la \$t3,A' e 'blez \$t0,fim_ciclo' *
 - ~~b.~~ 'li \$t3,A' e 'blz \$t0,fim_ciclo'
 - c. 'la \$t3,A' e 'blz \$t0,fim_ciclo'
 - d. 'la \$t3,A' e 'nop'
5. Imagine que tinha um sistema cuja memória demora 30ns para aceder aos dados desejados. Assumindo que tinha como objectivo mínimo obter um hit rate de 70% e um ganho total de desempenho de 3, quantas vezes mais rápida teria de ser a cache em comparação com a memória principal?

- a. Nenhuma das outras respostas.*
- b. 15x
- c. 10x
- d. 5x

$$\begin{aligned} \text{HR} &= 0,7 \\ \text{Su} &= 3 \end{aligned}$$

$$3 = \frac{1}{0,3 + \frac{0,7}{6e}} \Rightarrow 1 = 0,9 + \frac{2,1}{6e} \Rightarrow 0,1 = \frac{2,1}{6e} \Rightarrow 6e = 21$$

6. Considere o excerto de código na caixa seguinte. Indique qual das opções representa o valor correcto do registo \$a0, após a execução deste código.

- ~~a.~~ 100
- b. 0
- c. Nenhuma das outras respostas.
- d. 1*

```

MIPS
tab: .data
      .word 200, 20, 220, 40, 200, 140, 120, 20, 100, 10
      .text $t0
      la    $t0,tab
      lw    $t1,4($t0)
      lw    $t2,8($t0)
      slt  $a0,$t1,$t2
      =1

```

$20 < 220$
true

7. Considere um processador MIPS. Suponha que quer dividir um número por 8. Considere que a declaração correspondente em C a esse número seria `int num;`. Escolha qual das instruções seguintes correctamente divide o número por 8. Nota: recorde-se que o número pode ser negativo.

- a. srl \$a0,\$a0,3
- b.** sra \$a0,\$a0,3 * → *não faz extensão de sinal (srl não)*
- ~~c.~~ sra \$a0,\$a0,8
- ~~d.~~ div \$a0,\$a0,3

8. Considere um processador MIPS. Suponha que o registo \$v0 contém a leitura de um *encoder* de um motor, que utiliza um formato composto pelos três campos seguintes (do mais para o menos significativo): *NotUsed* (10 bits), *MotorPosition* (16 bits), *MotorID* (6 bits). Indique qual dos seguintes excertos de código permite ler o valor da posição do motor e guardá-lo no registo \$s0. Nota: o campo com a posição do motor é o campo *MotorPosition*.

MIPS 1

MIPS 2

...
andi \$s0,\$v0,0x003FFF00
srl \$s0,\$s0,6

minimizar

deletar comentários

MIPS 3

MIPS 4

- a.** MIPS 1
 - b.** MIPS 2
 - c.** MIPS 3
 - d.** MIPS 4



9. Considere o excerto de código na caixa seguinte. Indique qual das opções representa o valor correcto dos registos, após a execução deste excerto.

- a. $\$t3 = -5$ e $\$t4 = 0xFFFFFFF$ A
 - b.** $\$t3 = -5$ e $\$t4 = -5$ *
 - c. $\$t3 = -5$ e $\$t4 = 5$
 - d. $\$t3 = 5$ e $\$t4 = 5$

| MIPS | | $t_1 = 10$ |
|------|----------------|---------------------|
| ... | | |
| li | \$t1, 10 | $t_2 = 5$ |
| li | \$t2, 5 | $t_3 = 5 - 10 = -5$ |
| sub | \$t3,\$t2,\$t1 | |
| subu | \$t4,\$t2,\$t1 | $t_4 = 5 - 10 = -5$ |
| ... | | |

10. Considere o seguinte excerto de código MIPS. Sabendo que a instrução que está em falta, assinalada com < . . . >, salta para a label 'label1', indique qual das seguintes opções é a mais correcta para essa instrução, se a intenção for a de imprimir o valor 105.

- a.** jal label1
 - b.** jr label1
 - c.** jal \$ra
 - d.** jal label1 *

O jardim guarda o sra

```
MIPS
    li      $a0,100
    <...>
#label0:
    #print integer
    li      $v0,1
    syscall

    ...
#label1:
    addi   $a0,$a0,5
    jr     $ra
```

→ com o gal
dá jump para *

11. Considere o excerto de código na caixa seguinte. Indique qual das opções representa a *string* impressa no ecrã, após a execução deste excerto.

- a. Nenhuma das respostas está correcta.
 - b. Arquitectura
 - c. Computadores
 - d. Arqui**

encontro 0 de termos negativos c
meio

```
MIPS
.data
seq: .byte 0x41, 0x72, 0x71, 0x75, 0x69, 0
      0x74, 0x65, 0x63, 0x74, 0x75, 0x72, 0x61, 0

...
la      $t0, seq

# print string
li      $v0, 4
move   $a0,$t0
syscall
...
```

12. Dos tipos padrão de ciclos em C seguintes qual necessita apenas de uma (1) instrução de salto numa implementação equivalente em MIPS?

- a. 'while'
- b.** 'do-while' *não volta para trás*
- c. 'for'
- d. Todas necessitam de mais do que uma instrução de salto na implementação MIPS.

13. Considere o excerto de código na caixa seguinte. Indique qual das opções representa a string impressa no ecrã, após a execução deste excerto.

- a. Nenhuma das outras respostas.
- b. AC no DEEC
- c.** AB no DEEC *
- d. BC no DEEC

```

MIPS
    .data
str:   .asciiz "AC no DEEC\n"

    ...
la     $t0, str
lbu   $t1, 0($t0)
addi  $t1,$t1,1
sb    $t1, 1($t0) t1 = 'A'
*      t1 = 'B'
*      store de 'B' com 1 de offset *

#print string
li    $v0,4
move $a0,$t0
syscall
...

```

14. Nas aulas práticas utilizou a estrutura vector_t declarada ao lado. Considere o código indicado em que a função main chama a função prod(). Que valor é impresso no ecrã?

- a. Valor: 1
- b. Valor: 3
- c. Valor: 6
- d.** Valor: 2 *

```

struct vector_t {
    size_t size;
    int *data;
};

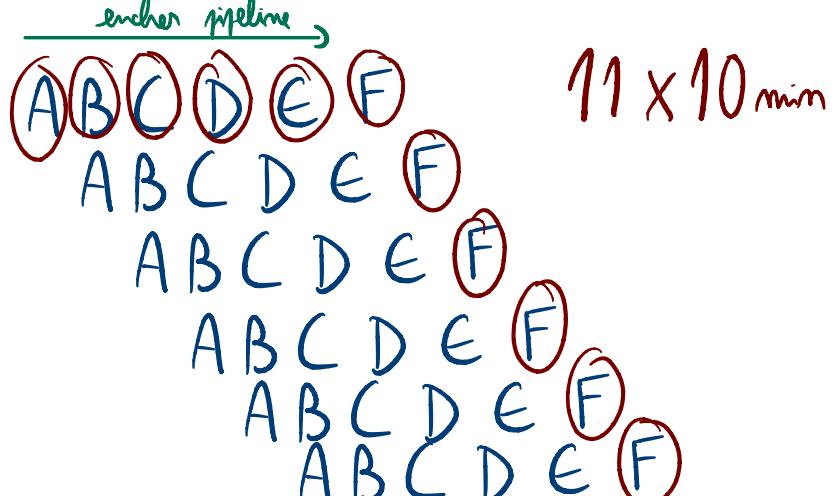
int Prod(vector_t *ptr) {
    *(pt->data)=*(pt->data)+3;
}

int main(){
vector_t *vec;
vec=(vector_t *)malloc(sizeof(vector_t));
vec->size=2;
vec->data=(int *)malloc(vec->size*sizeof(int));
vec->data[0]=1;
vec->data[1]=2;
Prod(vec);
printf("Valor: %d \n", *(vec->data)+1);
}

```

15. Assuma que para completar uma determinada função deve executar ordenadamente as seguintes operações A, B, C, D, E e F sobre um conjunto de 6 dados distintos, e que cada operação demora 10 minutos a executar. Qual o tempo de execução global assumindo agora o modo de operação em pipeline (sem stalls de paragem)?

- a. 60 minutos
- b. 100 minutos
- c. Outro valor
- d.** 110 minutos *



16. O caso mais comum numa compilação de um programa C para assembly usando o gcc é o uso do registo \$v0 para armazenar resultados de operações. Qual das razões indicadas a seguir para essa prática é VERDADEIRA?

- a. O objectivo mais comum de uma função é retornar um valor representando o resultado de uma operação; logo, como todas as rotinas em C são funções, faz sentido colocar por omissão o resultado de quaisquer operações em \$v0, porque este é o registo base de retorno. *
- b. Na arquitectura MIPS, o *hardware* que suporta as operações de aritmética e lógica estão optimizadas para o uso do registo \$v0 como registo destino.
- c. Algun registo teria de ser usado por omissão para esta tarefa, e calhou \$v0 ser o registo escolhido para esse efeito pelos criadores deste compilador.
- d. Nenhuma das afirmações acima é verdadeira.

17. A instrução ori tem uma valor imediato de 16 bits. Como decomporia a instrução assembly do MIPS ori \$s4,\$s2, 0xFFA7E2E6 numa instrução válida?

- a. lui \$at, 0xE2E6
ori \$at, \$at, 0xFFA7
ori \$s4, \$s2, \$at
 - b. lui \$at, 0xFFA7
~~ori \$at, \$at, E2E6~~
ori \$s4, \$s2, \$at
 - c. lui \$at, 0xFFA7 *
ori \$at, \$at, 0xE2E6
ori \$s4, \$s2, \$at
 - d. Outro conjunto de instruções.
- lui é load upper immediate*
o lui tem de ter o 0xFFA7

18. Assuma:

```
int a[10],*pa;  
pa = &a[0];
```

array are not assignable

Qual das seguintes afirmações é FALSA?

- a. Tanto **a = pa** como a++ são operações legais * **F**
- b. &a[i] "equivale a" a + i **V**
- c. pa[i] "equivale a" *(pa + i) **V**
- d. a[i] "equivale a" *(a + i) **V**