Arquitectura de Computadores

ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

– Exame de Recurso –

5 de Julho de 2019

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 imediata, 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 corretamente respondida vale cinco pontos; cada resposta errada desconta dois pontos; e cada pergunta não respondida vale zero pontos. Uma nota final abaixo de zero pontos, vale zero valores.

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

1ª Frequência (5 valores) – Duração: 60 minutos + 10 minutos de tolerância

1	2	3	4	5	6	7	8	9	10	11	12

2ª Frequência (10 valores) – Duração: 85 minutos + 20 minutos de tolerância

10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Exame de Recurso (15 valores) – Duração: 100 minutos + 20 minutos de tolerância

5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

1. **Considerando o programa em C ao lado, indique qual das seguintes opções é VERDADEIRA.**
   - **a.** O programa termina com um erro do tipo *Segmentation Fault*.
   - **b.** É apresentado no ecrã o valor `0x06`.
   - **c.** É apresentado no ecrã o valor `0x08`.
   - **d.** É apresentado no ecrã o valor `0x09`.

```
int main(){
  int a[] ={0,2,4,6,8,10,12};
  int *p1, *p2, **p3;
  p1 = a+1;
  p2 = p1+3;
  p3 = &p1;
  printf("%#X\n",*(*p3+1)+(*p2)-3;
  return 0;
}
```

2. **Dado o excerto de código em *assembly* (MIPS) ao lado, cuja função é guardar no registo $t3 a soma de todos os elementos do vetor num cujo valor seja inferior a 5. Escolha a opção que representa as instruções em falta assinalada com <...> na caixa em baixo:**
   - **a.** lw $t2, 0($t0) e addiu $t0,$t0,1
   - **b.** lw $t2, 0($t0) e addiu $t0,$t0,4
   - **c.** lb $t2, 0($t0) e addiu $t0,$t0,4
   - **d.** Nenhuma das opções anteriores está correcta.

```
        .data
num:    .word   1,5,14,3,8,2,4,6,9,0
        .text
        la      $t0, num
        li      $t1, 10
        add     $t3,$zero,$zero
        li      $t4,5
ciclo:
        beqz    $t1, out
        <. . .>
        addi    $t1,$t1,-1
        bge     $t2,$t4,nao_soma
        add     $t3,$t3,$t2
nao_soma:
        j       ciclo
out:
```

3. **No desenvolvimento de um programa e teste com *debugger*, qual das seguintes sequências de comandos é válida. Note que o ponto e vírgula permite separar dois comandos consecutivos:**
   - **a.** gcc -g f1.c f2.s -O -o main; gdb main
   - **b.** gcc -c f1.c f2.s -O main; gdc main
   - **c.** gcc -g -c f1.c f2.s -o main; gdb main
   - **d.** gdb -g f1.c f2.s -o main; gcc main

4. **Relativamente ao datapath de uma CPU do tipo multiple-cycle com cinco etapas e com os seguintes tempos máximos de execução por etapa: 1 – instruction fetch (400 ps); 2 – instruction decode (320 ps); 3 – ALU (320 ps); 4 – memory access (420 ps); 5 – register write (320 ps), indique qual das seguintes afirmações é VERDADEIRA?**
   - **a.** O período do relógio da CPU é definido pela velocidade máxima da ALU.
   - **b.** O período do relógio é dependente do tempo mínimo de execução de todas as etapas do *datapath*.
   - **c.** O período do relógio da CPU é definido pelo maior tempo de execução de todas as etapas.
   - **d.** O período do relógio da CPU é definido pela média aritmética dos tempos de execução de todas as etapas.

5. **Após a execução do seguinte excerto de código *assembly* (MIPS) diga quais são os valores que estão no array *num***
   - **a.** 0x00,0x22,0x33,0x44
   - **b.** 0x00,0x22,0x00,0x00
   - **c.** 0x00,0x00,0x00,0x00
   - **d.** Nenhuma das opções anteriores está correcta

```
          .data
num:      .byte
0x11,0x22,0x33,0x44

          .text

          la        $t0, num
          lw        $t1, 0($t0)
          andi      $t1,$t1,0xFF00
          sw        $t1,0($t0)
```

6. **Diga qual das afirmações é FALSA:**
   - **a.** Uma operação do tipo *branch* não permite mover o PC (*Program Counter*) para qualquer zona da memória
   - **b.** Se usar uma instrução do tipo *jal* para chamar uma função, não é necessário calcular manualmente o *return address*.
   - **c.** Numa instrução do tipo *branch*, o campo *immediate* pode ter um valor negativo.
   - **d.** O endereçamento relativo do PC (*Program Counter*) não permite codificar instruções do tipo *branch* em 32-bit

7. **Tendo em conta o *datapath* de um processador MIPS e a normal execução de uma instrução, indique qual das seguintes afirmações é VERDADEIRA:**
   - **a.** Na execução de uma instrução do tipo *jump*, o processador está *idle* nas etapas 4 e 5.
   - **b.** Na execução de uma instrução do tipo *set*, o processador está ativo em todas as etapas do *datapath*.
   - **c.** Não existe nenhuma instrução em cuja execução o processador esteja *idle* em mais do que 1 etapa do *datapath*.
   - **d.** Na execução de uma instrução do tipo *store*, o processador está ativo na etapa 5 *datapath*.

8. Considere um computador equipado com memória cache em que o acesso à memória demora 90 ns, enquanto o acesso à cache demora 10 ns. Qual o *speedup* que se consegue obter no sistema se apenas 10% dos acessos à memória resultarem numa *cache miss*.

a. 1.1 **c.** 5
b. 50 **d.** 11

9. Considere uma hierarquia de memória (memória principal + memória cache do tipo *Direct Mapped*) tal que na estrutura de endereçamento, o campo *Index* ocupa 10 bits do endereço. Se, para o mesmo tamanho da memória principal, memória cache e tamanho de bloco da memória, a memória cache fosse do tipo *4-way set-associative*, quantos sets tinha a memória?

a. 512 sets. **b.** 1024 sets. **c.** 128 sets. **d.** 256 sets.

10. 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
       <...>
ciclo:
       blez $t0,fim_ciclo
       mul     $t1,$t1,$t0
       <...>
       addi    $t0,$t0,-1
       b       ciclo
fim_ciclo:
       ...
```

C
```
$t1=1;
for ($t0=20;$t0>0;$t0--)
{
       $t1=$t1*$t0;
       if ($t1>1000)
              break;
}
```

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

a. `la $t0,20` e `blt $t1,$t2,fim_ciclo`

b. `li $t0,20` e `bgt $t1,$t2,fim_ciclo`

c. `li $t0,20` e `blt $t1,$t2,fim_ciclo`

d. `la $t0,20` e `nop`

11. Relativamente ao Assembly do MIPS, diga qual das afirmações é FALSA:

a. Durante a execução de um programa os registos *$a0-$a3* podem ser alterados.
b. Convencionalmente, o registo *$ra* é usado como endereço de retorno de uma função.
c. Os registos *$t0-$t1* são usados para devolver valores em funções.
d. O registo *$0* tem valor nulo e permanece sempre assim ao longo da execução do programa.

12. Considere o excerto de código em *Assembly* do MIPS apresentado na caixa ao lado. Indique qual das opções representa o valor armazenado no registo $t3 após a execução deste excerto de código em Assembly do MIPS:

a. $t3=2 **c.** $t3=3
b. $t3=1 **d.** $t3=4

```
       .data
tab:   .word
1,20,13,24,5,16,87,68,999,1010
       .text
       la      $t0,tab
       lw      $t1,4($t0)
       lw      $t2,8($t0)
       addi    $t3, $0, 1
       slt     $t3, $t2,$t1
       bgtz    $t3, cond2
       addi    $t3,$t3,1
cond2:
       addi    $t3,$t3,1
fim_ciclo:
```

13. **Considere uma memória cache do tipo "8-way associative cache" com um tamanho total de 2 MB e blocos de 512 bytes. Numa arquitetura de 32 bits, um endereço de memória de decompõe-se em:**
    a. "Tag" de 17 bits, "index" de 6 bits e "offset" de 9 bits.
    b. "Tag" de 15 bits, "index" de 8 bits e "offset" de 9 bits.
    c. "Tag" de 16 bits, "index" de 7 bits e "offset" de 9 bits.
    d. "Tag" de 14 bits, "index" de 9 bits e "offset" de 9 bits.

14. **Considere o excerto de código Assembly do MIPS listado abaixo. Relativamente às duas intruções** `bne`**, indique qual terá de ser o valor do imediato em cada uma das instruções** `bne` **de forma que o código salte para as labels** `L1` **e** `L2`**.**

```
          ...
      addi $t1,$0,20
L1:
      addi  $t4,$0,0
      sll   $t5,$t3,8
      add   $t5,$t5,$t2
L2:
      sw    $t3,0($t5)
      addi  $t5,$t5,4
      addi  $t4,$t4,1
      bne   $t4,$t1,L2
      addi  $t3,$t3,1
      bne   $t3,$t1,L1
EXIT:
          ...
```

    a. No primeiro `bne` o valor do imediato deverá ser `-3`, enquanto no segundo `bne` deverá ser `-8`.
    b. No primeiro `bne` o valor do imediato deverá ser `-4`, enquanto no segundo `bne` deverá ser `-9`
    c. No primeiro `bne` o valor do imediato deverá ser `4`, enquanto no segundo `bne` deverá ser `9`.
    d. No primeiro `bne` o valor do imediato deverá ser `-6`, enquanto no segundo `bne` deverá ser `-8`.

15. **Relativamente ao excerto de código abaixo. Sabendo que o "*label*"** `Loop` **indica o endereço da memória de instruções** `0x0000F400`**, qual dos seguintes códigos hexadecimais corresponde à codificação da instrução** `beq $t0,$0,Loop`**.**

```
Loop: addi $s0,$s0,0x10000
      addi $s1,$s1,-1
      slti $t0,$s1,1
      beq $t0,$0,Loop
```

    a. `0x1100FFFA`
    b. `0x1100FFFC`
    c. `0x1100F400`
    d. `0x1100000C`

16. **Considere a instrução em Assembly do MIPS dada pelo seguinte código hexadecimal** `0x00A45024`**. Sabendo que os registos** `$a0`**,** `$a1` **e** `$t2` **são os registos** `#4`**,** `#5` **e** `#10`**, respectivamente, indique qual das seguintes instruções representa a descodificação da instrução anterior:**

    a. `and $a0,$t2,$a1`
    b. `or $t2,$a1,$a0`
    c. `and $t2,$a1,$a0`
    d. `add $t2,$a0,$a1`

17. **Considere o seguinte excerto de código em Assembly do MIPS. Diga que valores irão ficar armazenados nos registos** `$t1` **e** `$t2`**.**

```
addi  $t0,$0,-12
sll   $t1,$t0,2
sra   $t2,$t1,1
```

    a. O registo `$t1` irá ficar com o valor -48 e o registo `$t2` com o valor -6.
    b. O registo `$t1` irá ficar com o valor 48 e o registo `$t2` com o valor -24.
    c. O registo `$t1` irá ficar com o valor -24 e o registo `$t2` com o valor -12.
    d. O registo `$t1` irá ficar com o valor -48 e o registo `$t2` com o valor -24.

18. Considere o seguinte código em Assembly do MIPS em que o programa salta para outra zona do código definida pela "label" `func`. São mostradas simultaneamente as instruções, bem como o seu endereço na memória.

Endereço	Instrução
	...
0x00400000	add $a0,$t0,$t1
0x00400004	jal func
...	...

Indique qual o endereço que será guardado no registo `$ra` após a execução da instrução "jump and link".

a. 0x00400004
b. 0x00400008
c. 0x00400000
d. 0x00400005

19. Assumindo que a "label" `array` se refere a uma tabela de inteiros armazenada no endereço de memória `0x10000448`, e que as "label" `funcA` corresponde a uma referência externa ao ficheiro, indique quantas entradas na tabela de realocação gerará o seguinte código Assembly do MIPS?

```
loop:
      lw    $t0,10($sp)
      lw    $t1,14($sp)
      addu  $t2,$t1,$t0
      sw    $t2,18($sp)
      blt   $t0,50, loop
      la    $a0,array
      lw    $t3,24($a0)
      jal   func
      move  $v0,$0
      lw    $ra,8($sp)
      addiu $sp,$sp,24
            ...
```

a. 3 entradas na tabela de realocação.
b. 2 entradas na tabela de realocação.
c. 4 entradas na tabela de realocação.
d. 5 entradas na tabela de realocação.

20. Indique qual a instrução em Assembly do MIPS correspondente ao incremento de 3 vezes de um ponteiro para inteiros (`int *ptr`) em C, isto é, equivalente a fazer `ptr=ptr+3` em C. Assuma que o registo `$a0` contém o ponteiro `ptr`.

a. `addi $a0, $a0, 3`
b. `addi $a0, $0, 6`
c. `addi $a0, $a0, 12`
d. `addi $a0, $a0, -3`

21. Considerando o trecho de programa indicado ao lado, indique qual das afirmações é <u>VERDADEIRA</u>:

```
char str[] = "Uma STRING";

void main(){
 int i;
 char *temp;

 temp=(char *)malloc(strlen(str)+1);

 for(i=0;i<=strlen(str);i++)
    temp[i]=str[i];
}
```

a. A variável ***str*** vai ser armazenada na zona de dados estáticos, a variável ***i*** na pilha e a variável ***temp*** é armazenada no *heap*.
b. As variáveis ***str*** e ***temp*** vão ser armazenadas no *heap* porque são tabelas.
c. As variáveis ***str, i*** e ***temp*** vão ser armazenadas na pilha, sendo que ***str*** e ***temp*** apontam para zonas de memória no *heap*.
d. A variável ***str*** vai ser armazenada na zona de dados estáticos do programa, a variável ***temp*** e ***i*** na pilha. A variável ***temp*** vai conter um endereço localizado no *heap*.

22. Das instruções indicadas a seguir, indique qual a única que é uma instrução TAL:

a. `andi $4,$5,0x0000FF00`
b. `mul $a0,$t0,$t1`
c. `bge $t1,$t2,salto`
d. `ori $4,$5,0x30002024`

23. Considere um sistema baseado num processador com um valor de CPI <u>REAL</u> igual a 4.0. O sistema possuí duas caches, uma para instruções e outra para dados. O CPI <u>IDEAL</u> deste sistema é igual a 2. A hit rate da cache de instruções é de 90%. A miss penalty para a memória de instruções é igual a 10 ciclos de relógio e a da memória de dados é igual a 20 ciclos de relógio. Sabendo que apenas 25% das instruções envolvem um acesso à memória de dados indique qual a *hit rate* da cache de dados?

a. 90% **b.** 80% **c.** 85% **d.** 20%

24. Considere o excerto de código na caixa seguinte. Indique qual das seguintes afirmações é <u>VERDADEIRA</u>:

```
I1:   add   $t1, $t1, $t4
I2:   sw    $t4, 0($t1)
I3:   add   $t4, $t4, $t5
I4:   andi  $t5,$t1,0x000000ff
I5:   sw    $t5,0($t1)
```

- **a.** As instruções I2 e I5 evidenciam uma situação de conflitos estrutural.
- **b.** As instruções I3 e I4 evidenciam uma situação de conflitos de dados.
- **c.** As instruções I1 e I4 evidenciam uma situação de conflitos estrutural.
- **d.** Todas as afirmações são verdadeiras.

25. Como se decompõe a instrução em Assembly do MIPS `ori $t1,$t0,0x201130FF` em instruções TAL?

```
a)  lui $at, 0x2011
    andi $at, $at, 0x30FF
    ori $t1, $t0, $at
```

```
b) lui $at, 0x2011
    ori $at, $at, 0x30FF
    ori $t1, $t0, $at
```

```
c)  lui $at, 0x30FF
    ori $at, $at, 0x2011
    ori $t1, $t0, $at
```

```
d)  lui $at, 0x30FF
    andi $at, $at, 0x2011
    ori $t1, $t0, $at
```

26. Assumindo que o registo `$a0` contém o valor `0xAABBCCDD`, indique qual é o valor armazenado no registo `$t0` após a execução do excerto de código *Assembly* do MIPS listado abaixo:

- **a.** `0x0000FFCC`.
- **b.** `0xAAFFCCDD`.
- **c.** `0xFFFF0000`.
- **d.** `0xAABBCCDD`.

```
ori  $t0,$a0,0x00FF0000
sll  $t0,$t0,8
andi $t0,$t0,0xFFFF0000
srl  $t0,$t0,16
```