



Arquitetura de Computadores

ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

– Exame da Época Normal –

6 de Janeiro de 2023

Duração: 90 minutos + 30 minutos de tolerância

Nome: _____ Número: _____

Notas Importantes:

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 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)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. Considerando o programa em C ao lado, indique qual das seguintes opções é **VERDADEIRA**.

- O programa termina com um erro do tipo *Segmentation Fault*.
- É apresentado no ecrã o valor 0x04.
- É apresentado no ecrã o valor 0x05.
- É apresentado no ecrã o valor 0x06.

```
int main(){
    int a[] = {0,1,2,3,4,5,6};
    int *p1, *p2, **p3;
    p1 = a+1;
    p2 = p1+3;
    p3 = &p1;
    printf("%#X\n", *((*p3+1)+(*p2)-1);
    return 0;
}
```

2. Considere um certo programa que deve executar ordenadamente as operações OP1, OP2, OP3, OP4, OP5 sobre um conjunto de 10 dados distintos. Considere que a operação mais rápida demora 2ns a executar e a mais lenta demora 5ns a executar. Qual o tempo de execução global assumindo o modo de operação em pipeline (sem stalls de paragem) com pipeline vazia no início da execução do programa?

- a. 28 ns b. 20 ns c. 50 ns d. 70 ns

3. Qual dos segmentos de código em C reproduz mais fielmente o ciclo em *assembly* indicado em baixo:

- do{\$s4=\$s4-1;} while ((\$s3 < \$s4) || (\$s4 < \$s5))
- do{\$s4=\$s4-1;} while ((\$s3 >= \$s4) && (\$s4 >= \$s5))
- do{\$s4=\$s4-1;} while ((\$s3 >= \$s4) || (\$s4 >= \$s5))
- do{\$s4=\$s4-1;} while ((\$s3 < \$s4) && (\$s4 < \$s5))

```
loop:
    addiu $s4,$s4,-1
    slt   $t0,$s3,$s4
    slt   $t1,$s4,$s5
    add   $t0,$t0,$t1
    bne   $t0,$0,out
    j     loop
out:
```

4. Quantas vezes deve uma memória principal ser mais lenta do que a memória cache por forma a garantir que o tempo médio de acesso à memória seja de 12ns, assumindo que o tempo de acesso à memória cache é de 6ns e a hit rate na cache é igual a 75%.

- 4 vezes mais lenta.
- 5 vezes mais lenta.
- 6 vezes mais lenta.
- 8 vezes mais lenta.

5. Com base no ficheiro Makefile apresentado à direita, diga qual das afirmações abaixo está correta?

(Note-se que os números são apenas indicativos da ordem das linhas e não fazem parte do ficheiro.)

```
1 meuprog: meumain.o funcoes.o
2     gcc -g meumain.o funcoes.o -o meuprog
3
4 meumain.o: meumain.c funcoes.h
5     gcc -c meumain.c
6
7 funcoes.o: funcoes.s
8     gcc -c funcoes.s
```

- a. A inclusão do ficheiro *funcoes.h* na linha 4 do ficheiro Makefile não serve para nada pois estamos a compilar apenas o ficheiro *meumain.c*
- b. O ficheiro *funcoes.c* é gerado automaticamente a partir do *funcoes.s* e depois usado para gerar o ficheiro objeto correspondente.
- c. Basta executar o comando “*make*” sem quaisquer argumentos para compilar o programa, porque *meuprog* é o primeiro “*target*” declarado.
- d. Executando o comando “*make meuprog*” irá efetuar sempre todas as etapas de compilação dos vários ficheiros.

6. Considere um datapath com cinco etapas: 1 – instruction fetch; 2 – instruction decode; 3 – ALU; 4 – memory access; 5 – register write. Indique qual das seguintes instruções em Assembly do MIPS está inactiva na etapa 5 do datapath.

- a. `slti $t1,$t0,1`
- b. `sb $t1,2($t0)`
- c. `sll $t1,$t0,2`
- d. `lbu $t1,3($t0)`

7. 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. `lbu $t2, 0($t0)` e `addiu $t0,$t0,1`
- d. Nenhuma das opções anteriores está correcta

```
num: .data      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:
```

8. Considere o seguinte código em Assembly do MIPS relativo a uma *string* armazenada em memória. Indique qual das seguintes instruções será a mais adequada para ler para o registo \$t0 o carácter “!”.

- a. `lh $t0,16($a0)`
- b. `lw $t0,28($a0)`
- c. `lb $t0,7($a0)`
- d. `lbu $t0,7($a0)`

```
.data
frase: .asciiz "Bom dia!"

.text
la $a0,frase
<instrução em falta>
```

9. Considere que tem uma memória cache com um tamanho total de 256 kB, organizada em blocos de 512 bytes, numa arquitectura de 32 bits e que o endereçamento é efectuado ao nível do byte. Tendo em conta que a estrutura de endereço da memória se decompõe num “*Tag*” de 16 bits, num “*index/set*” de 7 bits e um “*offset*” de 9 bits, estamos perante uma memória cache do tipo:

- a. 2-way set-associative cache.
- b. 4-way set-associative cache.
- c. Direct mapped cache.
- d. Fully associative cache.

10. Quando falamos sobre a forma como o processador pode comunicar com dispositivos externos, sabemos que podem ser utilizados dois mecanismos diferentes: I/O programado e *Memory-mapped I/O*. Relativamente a este assunto, indique qual das seguintes afirmações é **VERDADEIRA**:

- a. O I/O programado é mais rápido do que o *Memory-mapped I/O*.
- b. O *Memory-mapped I/O* requer instruções em *assembly* especiais para poder funcionar.
- c. O *Memory-mapped I/O* requer mais linhas no barramento de dados da CPU.
- d. O *Memory-mapped I/O* simplifica o hardware porque partilha a interface da memória.

11. 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=14
- b. \$t3=6
- c. \$t3=16
- d. \$t3=8

```
.data
tab: .word 2,4,6,8,10,12,14,16,18,20
.text
la    $t0,tab
lw    $t1,4($t0)
lw    $t2,12($t0)
addi  $t3,$0,2
slt   $t4,$t2,$t1
bne   $t4,$0,cond2
add   $t3,$t3,$t1

cond2:
      add   $t3,$t3,$t2
fim_ciclo:
```

12. Relativamente ao excerto de código ao lado indique qual dos seguintes códigos hexadecimais corresponde à codificação da instrução `beq $t0, $0, Loop`.

- a. 0x1100FFF6
- b. 0x1100FFFA
- c. 0x1100FFF8
- d. 0x11000008

```
Loop: lw    $s0,0($a0)
      addi  $s0,$s0,0x40000
      addi  $a0,$a0,4
      andi  $s0,$s0,0xFF00
      slti  $t0,$s0,1500
      beq   $t0,$0,Loop
```

13. 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. `and $t2, $a1, $a0`
- c. `or $t2, $a1, $a0`
- d. `add $t2, $a0, $a1`

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

- a. `subi $4, $5, 0x0000FF00`
- b. `slti $a0, $a1, 0x00001000`
- c. `bge $t1, $t2, salto`
- d. `andi $4, $5, 0x30002024`

15. No terceiro trabalho prático usou-se um teclado cuja leitura é feita por varrimento, escrevendo-se no endereço 0xFFFF0012 um valor que seleciona a linha de teclas a testar e lendo-se no endereço 0xFFFF014 se há alguma tecla premida nessa linha. Se não houver nenhuma tecla premida o valor devolvido é 0. Se houver uma tecla premida, o valor devolvido é composto por duas partes: o número da linha no *nibble* menos significativo e o número da coluna no mais significativo.

Considerando as sequências de pares de valores separados por vírgulas abaixo, onde o primeiro é o valor enviado e o segundo a resposta. Qual das sequências abaixo corresponderá à situação em que a tecla 'b' está a ser premida:

- a. 0x01 -> 0x00, 0x02->0x00, 0x04->0x84, 0x08->0x00
- b. 0x01 -> 0x00, 0x02->0x00, 0x04->0x43, 0x08->0x00
- c. 0x01 -> 0x00, 0x02->0x00, 0x04->0x0B, 0x08->0x00
- d. 0x01 -> 0x00, 0x02->0xB0, 0x04->0x00, 0x08->0x00

16. Considere o trecho de código apresentado ao lado. Assumindo que a “label” `tabNums` se refere a uma tabela de inteiros armazenada no endereço de memória `0x20004008`, e que as “label” `funcCalc` e `funcMin` correspondem a referências externas ao ficheiro com o código, indique quantas entradas na tabela de realocação gerará o seguinte código Assembly do MIPS?

```

la      $a0, tabNums
loop:   lw      $t1, 0($a0)
        add     $a0, $t1, $t1
        jal     funcCalc
        beq     $v0, $0, end
        la      $a0, tabNums
        jal     funcMin
        move    $a0, $v0
        bne     $a0, $0, loop
end:

```

- a. 3 entradas na tabela de realocação.
- b. 4 entradas na tabela de realocação.
- c. 5 entradas na tabela de realocação.
- d. 6 entradas na tabela de realocação.

17. Considere um sistema baseado num processador com um valor de CPI REAL igual a 5.35. O sistema possui duas caches, uma para instruções e outra para dados. O CPI IDEAL deste sistema é igual a 3. A hit rate da cache de instruções é de 95%. A miss penalty para a memória de instruções é igual a 15 ciclos de relógio e a da memória de dados é igual a 20 ciclos de relógio. Sabendo que apenas 40% das instruções envolvem um acesso à memória de dados indique qual a *hit rate* da cache de dados?

- a. 80%
- b. 90%
- c. 85%
- d. 10%

18. Como se decompõe a instrução em Assembly do MIPS `addi $t1, $t0, 0x00003040` em instruções TAL?

a) `lui $at, 0x1000`
`andi $at, $at, 0x3040`
`add $t1, $t0, $at`

b) `lui $at, 0x0000`
`ori $at, $at, 0x3040`
`add $t1, $t0, $at`

c) `lui $at, 0x3040`
`ori $at, $at, 0x0000`
`add $t1, $t0, $at`

d) Não se decompõe porque já é uma instrução TAL

19. Assumindo que o registo `$a0` contém o valor `0xAABBCCDD`, indique qual é o valor armazenado no registo `$v0` após a execução do excerto de código Assembly do MIPS listado ao lado:

```


srl $t0, $a0, 16
sll $t1, $a0, 16
or $t0, $a0, $t0
andi $v0, $t0, 0xFF00FFFF
ori $v0, $v0, 0x00EE0000

```

- a. `0xDDEEAABB`
- b. `0xDDCCBBAA`
- c. `0xCCEEAABB`
- d. `0xBBAAEECC`

20. Relativamente ao registo `$sp` (“*stack pointer*”) do Assembly do MIPS, diga qual das afirmações é VERDADEIRA:

- a. O registo `$sp` guarda o endereço correspondente ao topo da pilha e contém sempre o endereço mais elevado dos elementos da pilha.
- b. O registo `$sp` guarda o endereço correspondente ao topo da pilha e contém sempre o endereço mais baixo dos elementos da pilha.
- c. O registo `$sp` deverá sempre permanecer constante ao longo da execução de um programa.
- d. O registo `$sp` não deverá manter o mesmo valor antes e depois de uma instrução do tipo `jal`.

	<p align="center">Arquitetura de Computadores ENGENHARIA INFORMÁTICA FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE DE COIMBRA</p> <p align="center">– Exame da Época Normal –</p> <p align="center">6 de Janeiro de 2023</p> <p align="center">Parte II</p>	(a preencher pelo docente)

Nome: _____ Número: _____

Considere a representação “*pipelined*” do programa apresentada abaixo (F=“*Instruction Fetch*”, D=“*Decode*”, A=“*Execute*” ou “*Arithmetic*”, M=“*Memory Access*”, R=“*Register Write Back*”):

#1 addi \$a0, \$0, 1	F D A M R
#2 lw \$t4, 4(\$a1)	F D A M R
#3 addi \$t4, \$t4, 4	F D A M R
#4 lw \$t5, 0(\$t4)	F D A M R
#5 add \$t5, \$t5, \$a0	F D A M R
#6 lw \$t3, 0(\$a1)	F D A M R
#7 sw \$t3, 8(\$a1)	F D A M R
#8 add \$t0, \$t3, \$a0	F D A M R

- i) Desenhe setas para indicar a dependência de dados entre os diferentes níveis. A seta deve começar no nível em que os dados ficam pela primeira vez disponíveis e terminar onde os dados são absolutamente necessários. Indique também todos os conflitos detetados. Assuma que o processador usa todos os mecanismos disponíveis para resolução de conflitos.
- ii) Resolva todos os conflitos detetados de forma a tornar o código o mais eficiente possível e determine quantos ciclos de relógio são necessários para a execução total do código (assuma que o pipeline se encontra VAZIO no momento da execução do código apresentado).