Arquitectura de Computadores

ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

– 1ª Frequência –

9 de Abril de 2019
Duração: 75 min. + 15 min. de tolerância

Nome:_____ **Número:**_____

Notas Importantes:

A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior. Não serão admitidas quaisquer tentativas de fraude, levando qualquer tentativa detectada à reprovação imediata, tanto do facilitador como do prevaricador.

Durante a prova pode consultar a bibliografia da disciplina (slides, livros, enunciados e materiais de apoio aos trabalhos práticos). No entanto, não é permitido o uso de computadores/máquinas de calcular e a consulta de exercícios previamente resolvidos.

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 pergunta errada desconta dois pontos; cada pergunta não respondida vale zero pontos. Um total abaixo de zero, 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

1. Após a execução do seguinte programa em C, que valores serão impressos no ecrã?

```
int main(){

  char a[]="135";

  char *p1, **p2;

  p1 = a+1;
  p2 = &p1;

  printf("%c, %c, %c \n",a[1], *p1,*(*(p2)+1));
  return 0;
}
```

a. 1, 3, 5.
b. 3, 3, 5.
c. 3, 4, 5.
d. 3, 3, 4.

2. Considere um processador que possui uma CPI real igual a 4 ciclos de relógio e uma CPI igual a 2 ciclos de relógio quando todos os acessos à memória envolvem apenas a *cache*. Assuma a existência de duas *caches*, uma para instruções e outra para dados. A *hit rate* na cache de instruções é de 85% e a *miss rate* na *cache* de dados é igual a 20%. A *miss penalty* é de 8 ciclos de relógio na cache de instruções e de 10 ciclos de relógio na cache de dados. Qual a percentagem de instruções que envolvem um acesso à memória de dados?

a. 20% **b.** 30% **c.** 40% **d.** 50%

3. Considere o excerto de código na caixa seguinte. Indique qual das opções representa o valor correcto do registo `$t0`, após a execução deste código.

```
li    $t0,0x80402010
li    $t1,0x00FFFF00
and   $t0,$t0,$t1
ori   $t0,$t0,0x0040
srl   $t0,$t0,8
```

a. `$t0=0x00004020`
b. `$t0=0x00402000`
c. `$t0=0x00804000`
d. `$t0=0x00FFFF00`

4. **Relativamente à convenção de registos no MIPS, diga qual das afirmações é <u>VERDADEIRA</u>:**
   - **a.** Os registos `$a0` a `$a3` são usados para passar parâmetros para funções, o que limita a 4 o número máximo de argumentos que uma função pode ter.
   - **b.** O ponteiro `$sp` permanece estático ao longo da execução de um programa.
   - **c.** Os registos `$t0-$t3` podem ser usados para passagem de parâmetros para uma função, desde que o seu valor seja salvaguardado na pilha.
   - **d.** O valor do registo `$ra` deve ser manualmente preservado sempre que se entra numa função que vai chamar outras funções.

5. **Considere as seguintes linhas de código em linguagem Assembly do MIPS. Indique qual o valor que estará armazenado na posição de memória dada por my_ptr no final da execução do código.**

```
.data
my_ptr:     .word   100

.text
la $t0, my_ptr
lw $t2,0($t0)
slti $t3,$t2,10
sw $t3,0($t0)
```

   - **a.** Na posição de memória *my_ptr* passará a estar o valor 1.
   - **b.** Na posição de memória *my_ptr* passará a estar o valor 0.
   - **c.** Na posição de memória *my_ptr* continuará a estar o valor 100.
   - **d.** Nenhuma das restantes afirmações é verdadeira.

6. **Relativamente à gestão da memória em C, diga qual das afirmações é FALSA.**
   - **a.** Quando uma determinada função de um programa termina a sua execução, a sua "*stack frame*" não é logo descartada, permanecendo activa na pilha até que o programa chegue ao fim.
   - **b.** O espaço de memória ocupado por uma variável global fica alocado durante toda a execução do programa.
   - **c.** A memória da "*heap*" pode ser alocada/desalocada em qualquer instante.
   - **d.** Quando uma determinada função termina a sua execução, as suas variáveis locais são destruídas.

7. **Considerando o código indicado ao lado, indique qual das seguintes afirmações é <u>VERDADEIRA</u>:**

```
double media = 0;

void main(){
 int i,num;
 double *tab;

 scanf("%d",&num);
 tab=(double*)malloc(num*sizeof(double));

 for(i=0;i<num;i++){
  scanf("%f",&tab[i]);
  media+=tab[i];
 }
 media/=num;
 …
}
```

   - **a.** A variável ***tab*** vai ser armazenada no *heap* porque é uma tabela.
   - **b.** As variáveis ***media, i*** e ***num*** vão ser armazenadas na pilha e a variável ***tab*** vai ser armazenada no *heap*.
   - **c.** A variável ***media*** vai ser armazenada na zona de dados estáticos do programa, a variável ***num, tab*** e ***i*** na pilha. A variável ***tab*** contém um endereço localizado no *heap*.
   - **d.** A variável ***media*** vai ser armazenada na zona de dados estáticos, a variável ***i*** e ***num*** na pilha e a variável ***tab*** é armazenada no *heap*.

8. No trabalho prático 3 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 virgulas abaixo, onde o primeiro é o valor enviado e o segundo a resposta. Qual das sequências abaixo corresponderá à correta leitura da tecla 5 premida (que é a segunda da segunda linha)

a. 0x01 -> 0x00, 0x02->0x00, 0x04->0x55, 0x08->0x00
b. 0x01 -> 0x00, 0x02->0x22, 0x04->0x55, 0x08->0x00
c. 0x01 -> 0x00, 0x02->0x22, 0x04->0x35, 0x08->0x00
d. 0x01 -> 0x00, 0x02->0x22, 0x04->0x00, 0x08->0x00

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 15 bits, num "*index/set*" de 8 bits e um "*offset*" de 9 bits, estamos perante uma memória cache do tipo:

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

10. Considere o código em *Assembly* do MIPS apresentado à direita e que manipula o vetor de inteiros `tab[]` **armazenado em memória. Após a execução do código, que valores ficarão armazenados no vetor** `tab[]`**?**

a. tab = {4,8,12,16,20,24,28,32,36,40};
b. tab = {2,4,6,8,10,12,14,16,18,20};
c. tab = {1,2,3,4,5,6,7,8,9,10};
d. Nenhuma das anteriores.

```
      .data
tam: .word  10
tab: .word  1,2,3,4,5,6,7,8,9,10
      .text
      la    $t0,tab
      la    $t1,tam
      lw    $t2,0($t1)
      add   $t3,$0,$0
ciclo:
      beq   $t2,$t3,out
      lw    $t4,0($t0)
      sll   $t4,$t4,2
      sw    $t4,0($t0)
      addi  $t3,$t3,1
      addi  $t0,$t0,4

      j ciclo
out:
```

11. Qual dos segmentos de código em C reproduz mais fielmente o ciclo em *assembly* indicado ao lado?

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

a.
```
do{$s4=$s4-1;}
while (($s3 >= $s4) || ($s4 >= $s5))
```
b.
```
do{$s4=$s4-1;}
while (($s3 < $s4) || ($s4 < $s5))
```
c. `do{$s4=$s4-1;} while (($s3 < $s4) || ($s4 >= $s5))`
d. `do{$s4=$s4-1;} while (($s3 < $s4) && ($s4 >= $s5))`

12. Considere o seguinte programa em C em que a tabela *tab* é armazenada no endereço 0x7F7FE580. Com base nisso, que valores serão impressos no ecrã?

```
#include <stdio.h>

int main(){
  int tab[]={1,3,5,7,9,11,13,15,17,19};
  int *ptr;
  ptr=tab+3;
  printf("%#X, %#X, %#X, %#X.\n",ptr,*ptr,*ptr-2,*(ptr-2));
  return 0;
}
```

a. 0x7F7FE58C, 0x7, 0x3, 0x3
b. 0x7F7FE580, 0x7F7F83, 0x3,0x1
c. 0x7F7FE584, 0x7, 0x5, 0x3
d. 0x7F7FE58C, 0x7, 0x5, 0x3

13. 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)`

14. Determine a hit rate de um sistema com um tempo médio de acesso de 15ns, em que o acesso à memória principal requer 50ns e o acesso à cache é 5 vezes mais rápido:

a. 90%
b. 10%
c. 20%
d. 80%

15. 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.