Sistemas de Microprocessadores

Mestrado Integrado em Eng. Electrotécnica e Computadores

– Mini Teste –

10 de Abril de 2013
Duração: 60 min. + 10 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, conta 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
A	A	A	D	A	A	C	A	C	D	A	D

1. Considere o programa 'myprog.exe' que é chamado da seguinte forma '`myprog b c`'. Os parâmetros de entrada da função `main` são:

a. `argc=3` e `argv[]={"myprog", "b", "c"}`
b. `argc=2` e `argv[]={"b", "c"}`
c. `argc=2` e `argv[]={"myprog", "b", "c"}`
d. Nenhuma das anteriores.

2. Qual das seguintes afirmações, relativas ao `gcc` e `gdb` que utilizou nas aulas, é VERDADEIRA:

a. O uso da flag `-g` aquando da compilação permite ao `gdb` relacionar o código executável com o código fonte para fins de *debugging*.
b. O uso da flag `-c` com o compilador `gcc` produz um ficheiro em código C.
c. O uso da flag `-o` com o compilador `gcc` produz um ficheiro objeto.
d. O `gdb` pode receber como entrada o ficheiro em código C de forma a fazer o *debug* de erros de sintaxe.

3. Relativamente ao código C em anexo diga qual das afirmações é VERDADEIRA.

a. Após a execução do código, os valores armazenados na memória que é alocada na *heap* são {-1,1}.
b. O código causa um *"segmentation fault"*.

```
int main(){
 int num, num2;
 int *ptr,**handle;

 num = 1;
 ptr = (int *)malloc(2 * sizeof(int));
 handle = &ptr;
 **handle = num2 + 1;
 *(*handle+1) = num;
 *ptr =ptr[0]-num2-2;}
```

c. A variável `ptr` está armazenada na memória dinâmica (*heap*).

d. Os valores armazenados na memória que é alocada na *heap* são indeterminados após a execução do código.

4. **Nas aulas práticas utilizou a estrutura `vector_t` declarada ao lado. Considere o código em anexo em que a função `main` chama a função `transforma`. O que é impresso no ecrãn?**

a. `Value: 0`
b. `Value: 1`
c. `Value: 2`
d. Não é possível antecipar o valor que vai ser impresso

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

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

int transforma(vector_t *ptr){
 int vec2x1[2]={0,0};
 free(ptr->data);
 ptr->data=vec2x1;
}
```

5. **Considere o seguinte programa em C. Indique qual a declaração correta das variáveis `p` e `h`.**

a. `int *p,**h;`
b. `int *p,*h;`
c. `int *p, h;`
d. `int **p,*h;`

```
p = (int *)malloc(sizeof(int));
*p =10;
h = &p;
```

6. **Qual dos segmentos de código em C reproduz mais fielmente o ciclo em assembly mostrado em baixo**

```
loop:
 addiu     $s0,$s0,-1
 slt       $t0,$s1,$s0
 slt       $t1,$s0,$s2
 add       $t0,$t0,$t1
 bne       $t0,$0,out
 j         loop
out:
```

a. `do{$s0=$s0-1;} while (($s1 >= $s0) && ($s0 >= $s2))`
b. `do{$s0=$s0-1;} while (($s1 >= $s0) || ($s0 >= $s2))`
c. `do{$s0=$s0-1;} while (($s1 < $s0) || ($s0 < $s2))`
d. `do{$s0=$s0-1;} while (($s1 < $s0) && ($s0 < $s2))`

7. **Considere a programação de uma função em *assembly* do MIPS. Indique qual das seguintes afirmações é FALSA:**

a. Os registos `$a0` a `$a3` são utilizados para passar parâmetros para dentro da função.
b. A função pode devolver um `double` utilizando os registos `$v0` e `$v1`
c. O registo `$ra` tem que ser sempre salvaguardado na pilha.
d. A pilha tem que ser devolvida como foi encontrada.

8. **O código em C em baixo refere-se a uma rotina "Max" cujo objectivo é determinar o máximo numa tabela de inteiros. Será que alguma das versões em assembly faz exatamente a mesma coisa?**

```
int Max(int* ptr,int num_elements){
 int i, result;
 result=*ptr ;
 for (i=1; i<num_elements; i++)
  if (ptr[i]>result)
   result=ptr[i] ;
 return(result) ;}
```

(A)

```
Max:  lw    $v0, 0($a0)
      addiu $t0,$0,4
      mult  $a1,$t0
      mflo  $a1
      addu  $a1,$a0,$a1
loop: addiu $a0,$a0,4
      sltu  $t0,$a0,$a1
      beq   $t0,$0,out
      lw    $t1,0($a0)
      slt   $t0,$v0,$t1
      beq   $t0,$0,loop
      addiu $v0,$0,$t1
      j     loop
out:  jr    $ra
```

(B)

```
Max:  lw    $v0, 0($a0)
      addiu $t0,$0,4
      mult  $a1,$t0
      mflo  $a1
loop: addiu $a0,$a0,4
      slt   $t0,$a1,$a0
      bne   $t0,$0,out
      lw    $t1,0($a0)
      sltu  $t0,$t1,$v0
      beq   $t0,$0,loop
      addiu $v0,$0,$t1
      j     loop
out:  jr    $ra
```

(C)

```
Max:  lb    $v0, 0($a0)
      addiu $t0,$0,1
      mult  $a1,$t0
      mflo  $a1
      addu  $a1,$a0,$a1
loop: sltu  $t0,$a1,$a0
      bne   $t0,$0,out
      lb    $t1,0($a0)
      slt   $t0,$t1,$v0
      beq   $t0,$0,jump
      addiu $v0,$0,$t1
jump: addiu $a0,$a0,1
      j     loop
out:  jr    $ra
```

(D) Nenhuma das opções acima é correta

9. **Considere a função func cujo código em assembly para o MIPS é mostrado ao lado. Qual dos seguintes códigos em C é equivalente?**

```
.text

.globl func

func: sll $t1, $a1, 2
      add $t1, $a0, $t1
      lw  $t0, 0($t1)
      lw  $t2, -4($t1)
      sw  $t2, 0($t1)
      sw  $t0, -4($t1)
      jr  $ra
```

(A)

```
void func (int v[], int k){
      int temp;
      temp = v[k];
      v[k] = v[k+1];
      v[k+1] = temp;}
```

(B)

```
void func (int v[], int k){
      int temp;
      temp = v[k];
      v[k] = v[k+2];
      v[k+2] = temp;}
```

(C)

```
void func(int v[], int k){
      int temp;
      temp = v[k];
      v[k] = v[k-1];
      v[k-1] = temp;}
```

(D)

```
void func(int v[], int k){
      int temp;
      temp = v[4*k];
      v[4*k] = v[4*(k-1)];
      v[4*(k-1)] = temp;}
```

10. **Considere a função "func" que faz a soma de dois parâmetros passados através de $a0 e $a1. Qual é a afirmação VERDADEIRA?**

a. A função utiliza registos proibidos.

b. A função não vai regressar porque o valor de $ra não foi reposto.

c. A função não respeita a convenção de registos.

d. A função não deixa a pilha como a encontrou.

```
func:
      addiu $sp, $sp, 4
      sw    $ra, 0($sp)
      addu  $v0, $a0, $a1
      addiu $sp, $sp, -4
      jr    $ra
```