Exercícios Resolvidos

Exame de Época Normal, 2003/2004, Parte Prática:

1.(0,5 val.) Considere o seguinte troço de programa em linguagem Assembly do MIPS:

```
                .data 0x10010000        # segmento de dados
palavra1:       .word 13
palavra2:       .word 0x15
```

Indique, em hexadecimal, quais os valores das seguintes etiquetas:

`palavra1:` `0x10010000`

`palavra2:` `0x10010004`

Nota: também foram aceites as respostas que se referiram ao valor armazenado nas posições de memória (13 e 0x15), ou os seus equivalentes em hexadecimal/decimal, embora a resposta correcta seja a que consta desta resolução.

2.(0,5 val.) Considere o seguinte programa em Assembly do MIPS:

```
        .data
var1:   .word 30
var2:   .word -50
res:    .space 1
        .text
main:   lw $t8,var1($0)
        lw $t9,var2($0)
        and $t1,$t1,$0
        and $t0,$t0,$0
        beq $t8,$0,haz
        ori $t0,$0,1
haz:    slt $t1,$t9,$t8
wat:    and $t0,$t0,$t1
        sb $t0,res($0)
```

Qual o valor armazenado na posição de memória `res`?

1.

3. (0,2+0,3+0,5 val.) Considere o seguinte troço de programa em linguagem Assembly do MIPS:

```
        .data 0x10010000
byte1:  .byte 0x10
espaco: .space 4
byte2:  .byte 0x20
pal:    .word 10
```

3.(a) Quantos bytes foram reservados para a variável espaco?

2 bytes.

3.(b) A partir de que endereços se inicializaram:

`byte1:` `0x10010000`

`byte2: 0x10010003`

3.(c) O que acontece se o processador executar a instrução `lw $t0,pal`? Como resolver essa situação?

Dá-se um acesso desalinhado à memória, visto que se pretende carregar uma palavra (4 bytes) a partir de um endereço que não é múltiplo de 4. Uma forma possível é introduzir a directiva .align 2 no início da zona de dados.

Nota: todos os alunos tiveram a cotação total (0,5 val.) nesta pergunta, devido à gralha .space 2 do exame (deveria estar .space 4, como nesta resolução).

4. (1 val.) Complete o seguinte código: escreva um procedimento em Assembly do MIPS que percorra o vector de bytes `sala` e conte o número de bytes cujo valor é igual a 1. Armazene o resultado da contagem na variável `lixos`. (Pode assumir que a dimensão do vector `sala` é sempre 64 bytes).

```
        .data           # declara segmento de dados
sala:   .space 64       # variável que representa o conteúdo de uma sala
lixos:  .word 0         # variável que conta o número de lixos
#
# Princípio do Segmento de Texto
#
        .text   # declara segmento de código
main:
        li $t0,0        # auxiliar (lê byte i)
        li $t1,0        # conta número de lixos
        li $t2,0        # variável i do ciclo
ciclo:
        lb $t0,sala($t2)
        bne $t0,1,sala($t2)
        addi $t1,$t1,1
continua:
        addi $t2,$t2,1
        beq $t2,64,fim
        j ciclo
fim:    sb $t1,lixos
```

5. (1 val.) Considere que já se encontra implementado o procedimento `randnum`, o qual devolve um inteiro aleatório entre 0 e um dado número limite:

- argumentos: número máximo aleatório pretendido (em `$a0`)
- resultados: inteiro aleatório entre 0 e argumento `$a0` menos 1 !

Escreva um procedimento `Joga`, em Assembly, que recebe como argumento um inteiro `Aposta`. O procedimento chama o `randnum` com o argumento `10`. Caso o resultado seja igual ao argumento `Aposta`, o procedimento imprime na consola a string "Certo!". Caso contrário, imprime "Errado!".

```
joga:
        addi $sp,$sp,-4
        sw $ra,4($sp)
        move $t0,$a0   # t0 contém o argumento aposta
        li $a0,10
        addi $sp,$sp,-4
        sw $ra,4($sp)
        jal randnum
```

```
        lw $ra,4($sp)
        addi $sp,$sp,4
        beq $v0,$t0,certo      # resultado em v0
errado:
        la $a0,string_errado
        li $v0,4
        syscall
        j fim
certo:
        la $a0,string_certo
        li $v0,4
        syscall
fim:
        lw $ra,4($sp)
        addi $sp,$sp,4
        jr $ra
```

6. (1+1 val.) Pretende-se codificar um procedimento Substitui(string,x,y), em Assembly do MIPS, que dada uma string e dois caracteres, x e y, substitui nessa string todas as ocorrências do caracter x pelo caracter y.

O procedimento terá como argumentos:

- o endereço da string
- o caracter a procurar
- o caracter para a substituição
- o número de caracteres da string

Por exemplo, para a string "Sobstitoi", se o caracter a procurar for o caracter 'o', e o caracter para a substituição for o caracter 'u', a string deve ser alterada para "Substitui".

6.(a) Desenhe um fluxograma para este procedimento.

6.(b) Escreva o programa em Assembly do MIPS.

```
substitui:
        addi $sp,$sp,-4
        sw $ra,4($sp)
        li $t1,0
ciclo:
        lb $t0,($a0)
        bne $t0,$a1,continua
sub:
        sb $a2,($a0)
continua:
        addi $a0,$a0,1
        addi $t0,$t0,1
        bne $t1,$a3,ciclo
        # senão, termina
        lw $ra,4($sp)
        addi $sp,$sp,4
        jr $ra
```

7. (1 val.) Descreva o que acontece quando um procedimento é chamado usando jal.

Quando um procedimento é chamado usando `jal`, duas coisas acontecem:

- O controlo é transferido para o endereço fornecido pela instrução.
- O endereço de retorno é guardado no registo `$ra`.

8. (1 val.) Em que consistem as excepções assíncronas? Será um overflow aritmético uma excepção assíncrona?

As excepções assíncronas são as que ocorrem sem qualquer relação temporal com o programa que é executado. Pedidos de E/S, erros de memória, falhas de fornecimento de energia etc. são exemplos de excepções assíncronas. Um overflow aritmético não é uma excepção assíncrona, visto que ocorre no mesmo sítio de cada vez que um programa é executado com os mesmos dados e com a mesma alocação de memória.

Exame de Recurso, 2003/2004, Parte Prática:

1.(1 + 0,5 val.)

1.(a) Crie um programa que defina um vector de 5 palavras associado à etiqueta `vector` que comece no endereço `0x10000000` e que contenha os valores 10,11,12,13,14.

```
        .data 0x10000000
vector: .word 10,11,12,13,14
```

1.(b) O que acontece se quisermos que o vector comece no endereço `0x10000002`? Em que endereço começa realmente? Porquê?

Começa no endereço 0x10000004, por ser uma palavra de memória (4 bytes).

2. (1 val.) Considere o seguinte programa:

```
        li $t1,0
ciclo:  lb $t5,string($t1)
        beq $t5,$0,fim
        addi $t1,$t1,1
        j ciclo

fim:    li $v0,1
        move $a0,$t1
        syscall                 # chamada sistema print_int
        addi $v0,$0,10          # $v0=10 (para retornar o controlo ao SPIM)
        syscall
```

Para a cadeia de caracteres "`AC-Uma`", declarada na memória como `.asciiz`, que número aparece na consola quando este programa é executado? O que faz este programa?

Conta o número de caracteres numa dada string. Para a string "AC-Uma", aparece na consola 6.

3. (1 val.) Suponha que num determinado protocolo de comunicação de dados, é necessário introduzir caracteres de controlo ('#','/','%').

Por exemplo, se o texto a transmitir é:

```
Calvin: A realidade continua a arruinar a minha vida.
```

no destino chega:

```
Ca%lvin:# A re/alid/ade continu/a a arru#inar a minh/a vi%da.
```

Pretende-se desenvolver uma rotina que no computador destino suprima estes caracteres do texto, de forma a obter de novo o texto original. Parta do seguinte cabeçalho:

```
        .data 0xffff8888       # declara zona de memória partilhada
mensagem:
        .space 128
        .data 0x10010000       # declara segmento de dados
resultado:
#
# Princípio do Segmento de Texto
#
        .text   # declara segmento de código
main:
```

e considere que a string `mensagem` (string recebida no computador destino) termina com o caracter 0x0, e que os valores ascii dos caracteres de controlo são 0x12,0xff e 0x63 (para efeitos de comparação).

Guarde a string resultante em `resultado`.

4. (1+1 val.) Considere um vector de números em posições consecutivas de memória e cujo fim é indicado pelo valor 0. Pretende-se desenvolver uma rotina que incremente em uma unidade cada um dos números no vector. Por exemplo:

vector inicial = [1 2 3 4 5 0]

resultado = [2 3 4 5 6 0]

Considere que o endereço do vector inicial é passado para a rotina pelo registo `$a0` e que o resultado deve ficar num endereço de memória que é passado para a rotina pelo registo `$a1`.

a) Desenhe um fluxograma que realize a função pretendida.

b) Escreva a rotina em linguagem assembly do processador MIPS.

5. (1 + 0,5 val.) Considere a seguinte rotina de tratamento a um interrupção:

```
#############################################################################
# Rotina (nossa) de serviço à interrupção:
#############################################################################
ext:    li $v0 4
        la $a0 inter
        syscall

        li $v0 1
        lb $a0 in
        syscall

        sb $a0 lido($0)
continua:
        mtc0 $0, $13   # Limpar o registo Cause
```

5. a) O que acontece quando ocorre uma interrupção vinda do teclado? Uma rotina de serviço à interrupção deve ser curta? Porquê?

5. b) Considere que a etiqueta in está associada ao porto de leitura do teclado. O que faz a instrução lb $a0 in?

6. (1 val.) Qual a diferença entre o código de tratamento de uma excepção e o código de uma rotina? Uma excepção pode retornar valores? Porquê?

Enunciados de Projectos

Ano Lectivo 2003/2004 – 1º Semestre

Enunciado do Projecto:
Um Controlador para o Robot Shakeaspira

Introdução

Você e o seu colega trabalham como consultores júniores na empresa de consultadoria B., Anha & D'Akobra[2], uma empresa especializada no desenvolvimento de sistemas embebidos em tempo real (e.g. controladores industriais) para os quais é frequente programar em Assembly.

Foram ambos destacados pelo gestor de projecto para o desenvolvimento de um controlador para um robot chamado Shakeaspira. O Shakeaspira é um robot aspirador automático para a casa: movimenta-se automaticamente num quarto, evitando objectos e quaisquer obstáculos (nomeadamente as paredes) ao mesmo tempo que limpa o pó do quarto.

Figura 1: Robot aspirador para a casa com movimentação automática (modelo experimental[3]). No topo esquerdo, o robot encontra-se na estação de recarga da bateria.

O Shakeaspira possui incorporada uma bateria, que torna o robot autónomo, sem precisar de manutenção. A bateria vai descarregando ao longo do tempo, e quando atinge um determinado valor limite, o Shakeaspira dirige-se para uma localização – pré-definida – onde existe o carregador da sua bateria.

O problema é que o prazo para o desenvolvimento é curto, uma vez que a concorrência – uma empresa japonesa – já está em vias de comercializar um robot semelhante, actualmente em fase experimental (ver Figura 1). Por isso você terá de ser eficaz, criativo e inovador.

---

[2] Nome fictício, obviamente.

[3] Para mais informações sobre o robot (real) da "concorrência", visite:
`http://www.i4u.com/japanreleases/hitachirobot.htm`

Descrição do Controlador a Implementar

Por motivos de custo, o controlador será simulado em software, usando o simulador SPIM, que simula um processador MIPS R2000, a fim de testar as suas funcionalidades sem custos adicionais de hardware.

A Figura 2 ilustra o quarto onde o robot aspira. O quarto é discretizado numa grelha. Em cada posição da grelha só pode existir ou um objecto ou lixo.

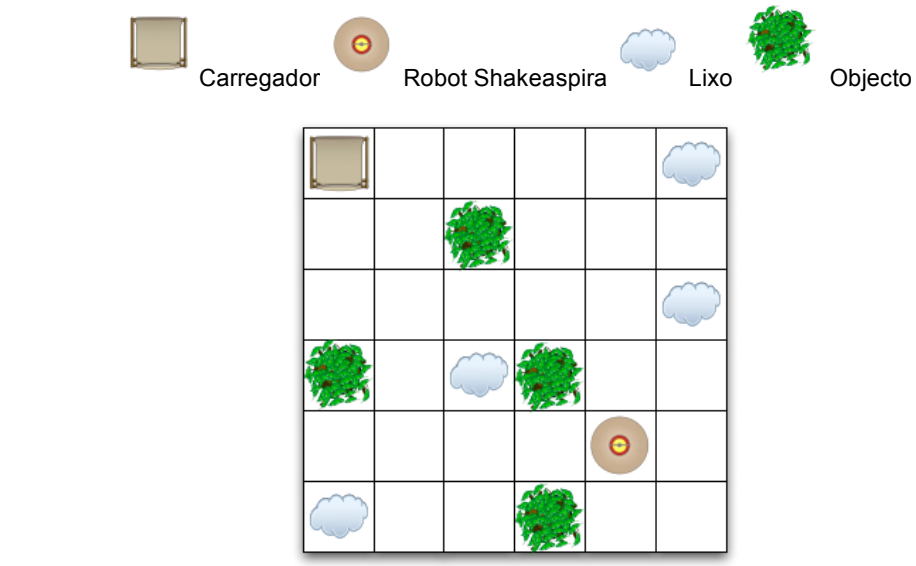

Figura 2: Ilustração de um exemplo de quarto do Shakeaspira (6×6).

No início da execução do programa, pede-se ao utilizador que introduza os seguintes parâmetros:

- Dimensão do quarto (4×4, …, 8×8)
- Quantidade de objectos (mínimo 2, máximo 6)
- Quantidade de lixo (mínimo 2, máximo 6)
- Nível inicial da bateria (mínimo 10, máximo 5000)

Não é necessário verificar se os valores introduzidos estão dentro dos limites estipulados. Os objectos e os lixos são colocados em posições aleatórias. A localização da bateria do robot é fixa: posição cuja coordenada é a origem da grelha: (0,0), pelo que nesta posição não pode ser colocado objecto ou lixo algum. A localização inicial do robot é a posição (1,0), à direita do carregador.

O robot tem dois modos de funcionamento: o **modo automático** e o **modo manual**. No **modo automático**, após a definição dos parâmetros, o robot avança à procura de lixo para recolher. O robot tem de percorrer todas as posições: se o conteúdo da posição onde se encontra possuir lixo, recolhe-o e incrementa uma variável inteira que conta o número de lixos recolhidos. Se, por outro lado, o conteúdo da posição onde se encontra possuir um objecto, o robot não o recolhe.

Sempre que o robot avança, a bateria (representada como uma variável inteira na memória) é decrementada uma unidade. Quando o valor da bateria atinge um determinado nível mínimo, o robot imprime a mensagem "Bateria Baixa!", e tem de retornar à posição base da bateria (0,0) onde recarrega a sua bateria para o valor do nível máximo (500). Cabe a si determinar um valor plausível para o nível mínimo. O robot termina a sua tarefa automática quando todo o lixo tiver sido recolhido.

O exemplo seguinte ilustra a forma como o programa deverá funcionar no **modo automático**:

```
Shakeaspira> Introduza o tamanho do quarto (4x4...8x8): 6
Shakeaspira> Introduza quantidade de objectos (2...6): 4
Shakeaspira> Introduza quantidade de lixo (2...6): 4
Shakeaspira> Introduza a bateria inicial (10...500): 100
```

Após a introdução dos parâmetros, o programa pede uma confirmação:

```
Shakeaspira> Tamanho=6, Objectos=4, Lixo=4, Bateria=100, confirma? (s=1/n=0): 1
```

No caso de o utilizador teclar `0`, o programa volta ao ciclo em que pergunta os parâmetros. Quando o utilizador teclar `2`, o programa termina. Caso o utilizador confirme, o programa mostra a sequência de acções do robot:

```
 0 1 2 3 4 5
0C O
1    R   L
2
3  O     L O
4      O
5    L   L
Shakeaspira> Lixo recolhido=0 Bateria=99
```

Respeite este formato de apresentação do conteúdo do quarto. `C` representa o carregador da bateria, `R` o robot, `O` um objecto e `L` um lixo. O programa mostra sempre quanto lixo já recolheu, assim como o nível da bateria. O programa fica à espera que o utilizador pressione a tecla `Enter`, para avançar e mostrar o estado resultante da acção seguinte[4].

```
 0 1 2 3 4 5
0C O
1      R L
2
3  O     L O
4      O
5    L   L
Shakeaspira> Lixo recolhido=0 Bateria=98

 0 1 2 3 4 5
0C O
1        R
2
3  O     L O
4      O
5    L   L
Shakeaspira> Lixo recolhido=1 Bateria=97
```

[4] Sempre que o utilizador pressionar a tecla `2`, seguida de `Enter`, o programa termina.

Neste exemplo, o robot recolheu 1 lixo na posição (4,1). O lixo (`L`) dessa posição desaparece. O programa termina quando o lixo recolhido for igual ao lixo inicialmente introduzido. O robot não pode executar saltos! Isto é, só avança entre posições contíguas, em cada uma das direcções norte,sul,este,oeste.

No **modo manual**, o funcionamento é semelhante, excepto no facto de o robot ficar parado à espera que o utilizador pressione uma das teclas direccionais já referidas.

Neste modo, o Shakeaspira é operado pelo teclado, usando as teclas `i`,`j`,`k`,`l` para especificar a direcção do seu movimento (`i`=norte, `k`=sul, `j`=este, `l`=oeste).

Para implementar este modo não pode usar a chamada de sistema `syscall`, já que esta pede sempre um `Enter` e mostra no écran a tecla digitada. Em vez disso, deve criar um novo ficheiro a partir do `trap.handler` onde associa uma rotina de tratamento das interrupções do teclado a uma rotina que move o robot numa dada direcção. Desta maneira, pode-se teclar à vontade que apenas surge na consola a "imagem" actualizada do quarto do robot.

Não é necessário, neste modo, introduzir um valor de bateria, visto que esta não é utilizada. Também não é necessário verificar se estamos a mover o robot contra uma das paredes (essa funcionalidade dá direito, contudo, a crédito extra).

Funcionalidades já implementadas

O seu gestor de projecto possui dois ficheiros que serão, provavelmente, do seu interesse, pelo que deve contactá-lo para obtê-los: um deles é um gerador de números aleatórios, que servirá para testar o Shakeaspira num quarto dispondo o lixo e os obstáculos aleatoriamente (a localização da bateria é fixa e pode assumir que o robot a conhece).

O outro é uma rotina de tratamento de interrupções adaptada da rotina original do núcleo do Sistema Operativo simulado pelo SPIM. Servirá como ponto de partida para codificar o modo de operação manual do Shakeaspira. Lembre-se que pode (e deve) reaproveitar o código do modo automático para implementar o modo manual.

Objectivos mínimos e Créditos Extra

Para obter aprovação no Projecto, é necessário que este funcione correctamente para a gama especificada de valores dos parâmetros, tanto no modo manual como no modo automático.

Existe um conjunto de funcionalidades que, a serem correctamente implementadas, conduzem a créditos extra na nota final:

- Shakeaspira que evita obstáculos: + 2 valores em relação à nota base;
- Shakeaspira em modo aleatório: + 1 valor em relação à nota base; o modo aleatório é um modo em que o robot escolhe em cada passo uma direcção aleatória, mantendo o restante comportamento já definido;
- Programa que permite colocar o Shakeaspira numa posição inicial aleatória, mantendo o restante comportamento: + 1 valor em relação à nota base.
- Alerta: no modo manual, imprime um aviso quando se move o Shakeaspira contra uma das paredes: + 1 valor.

Prazos e Critérios de Avaliação

O projecto seguirá uma entrega por fases. A nota só é dada após a discussão. No entanto é obrigatório apresentar os artefactos designados no prazo respectivo. Por cada dia de atraso na entrega, sofre uma penalização de 1 valor na nota final.

As fases e artefactos a entregar são os seguintes:

1. Fluxograma de alto nível
2. Programa a funcionar no modo automático com os objectivos mínimos
3. Versão Final do Programa + Relatório

Os **prazos** para cada fase são:

1. Dia 25 de Novembro de 2003.
2. Dia 19 de Dezembro de 2003.
3. Dia 16 de Janeiro de 2003[5].

O formato e local de entrega serão oportunamente divulgados.

Os seguintes aspectos serão avaliados:

- Fluxograma e Relatório;
- Cumprimento dos requisitos (correcto funcionamento do programa, em modo automático e em modo manual);
- Qualidade do código (e.g.: Os comentários são úteis? As etiquetas fazem sentido? Segue a convenção de utilização dos registos e procedimentos do MIPS? O código encontra-se bem estruturado?);
- Discussão do Projecto (esta componente da nota é individual e obrigatória);
- Serão valorizadas soluções que se diferenciem pela inovação, criatividade e desempenho em relação às restantes.

Projectos iguais, ou muito semelhantes, serão classificados com 0 (zero) valores.

Consulte regularmente a página Web dos laboratórios pois serão afixadas as respostas às perguntas mais frequentes dos alunos.

[5] Ainda por confirmar.