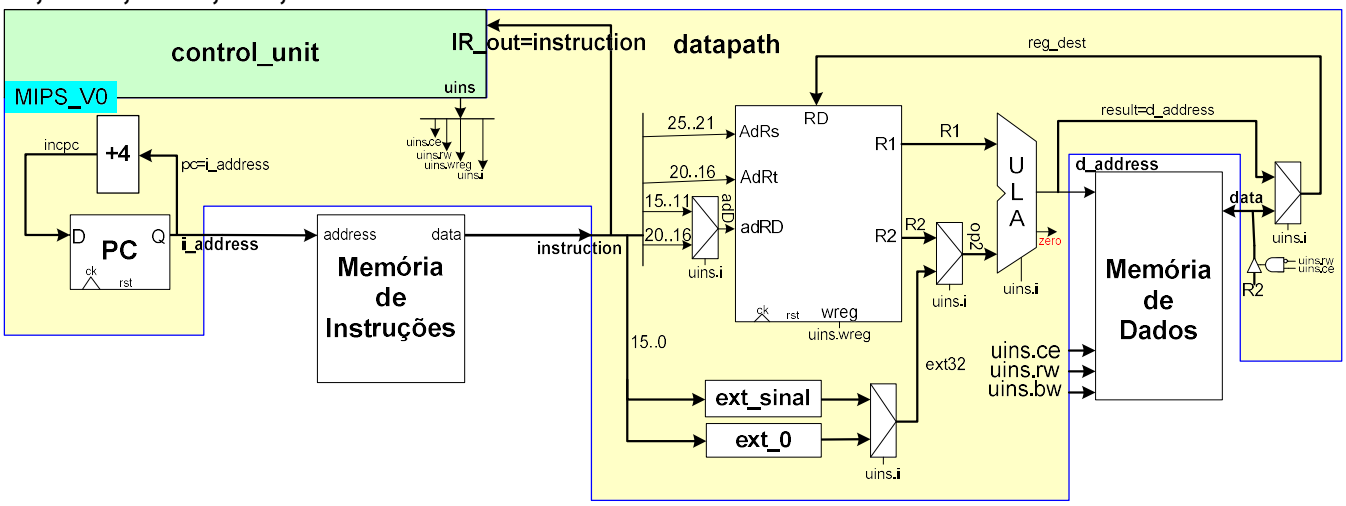
Prova P2 **Disciplina: Organização de Computadores**

Professor: Ney Laert Vilar Calazans

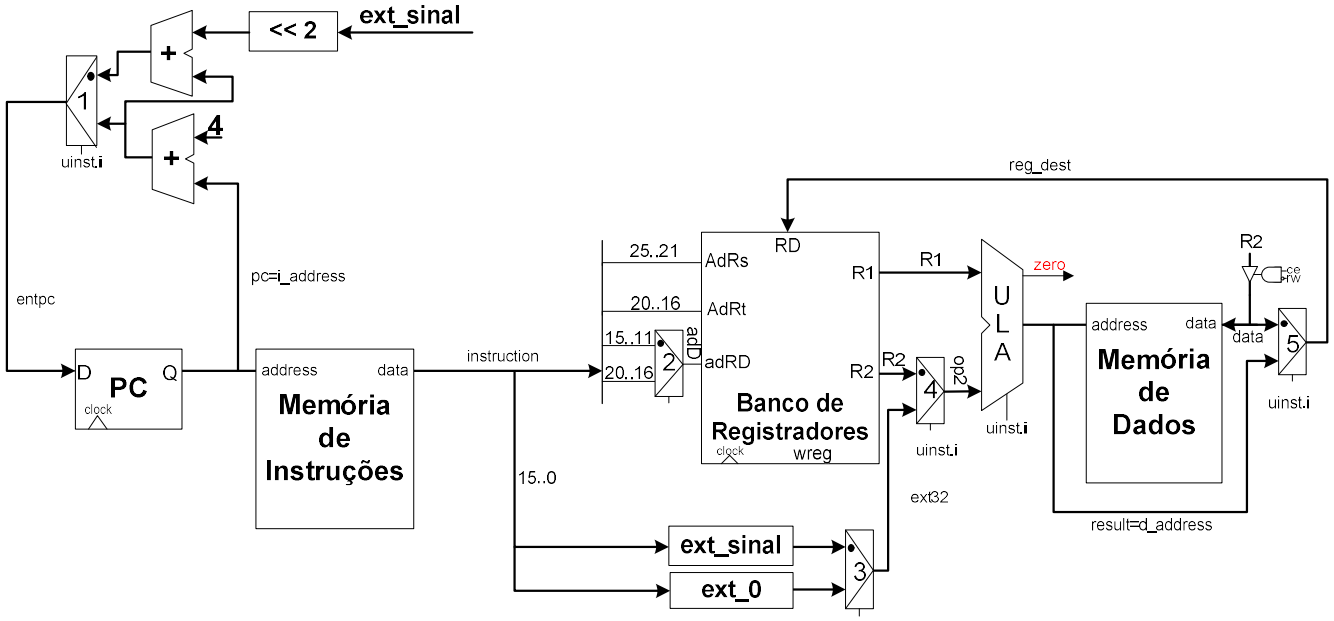
Aluno:

04/novembro/2015

Para realizar a prova, refira-se à proposta de organização MIPS monociclo vista em aula. O desenho da versão monociclo aparece abaixo, com detalhamento do Bloco de Dados. Assuma que as instruções às quais esta organização monociclo dá suporte de execução são apenas as seguintes (exceto quando a questão particular especificar de outra forma): **ADDU, SUBU, AND, OR, XOR, NOR, LW, SW e ORI**.



1. [3,0 pontos] Considere o bloco de dados monociclo apresentado abaixo. Ele resultou da transformação do bloco de dados visto em aula para dar suporte às novas instruções **BEQ** e **LUI**. O ponto nos multiplexadores indica condição verdadeira (ou, equivalentemente, sinal de controle do mux em 1). Assuma que as instruções às quais este processador dá suporte de execução são apenas as seguintes: **ADDU, SUBU, AND, OR, XOR, NOR, LW, SW, ORI, BEQ e LUI.**



a) Suponha que existe uma única falha, que afeta apenas o multiplexador que escolhe o valor a ser escrito no banco de registradores (Mux 5). A falha é que o sinal de controle deste multiplexador está grudado eternamente no valor 0. Diga quais instruções ainda podem ser executadas corretamente, justificando para cada uma delas.

b) Observe na documentação do MIPS a funcionalidade da instrução ADDIU e diga se e como o hardware acima deveria ser alterado para ser capaz de executar esta instrução. Que operação deve ser realizada na ULA? Quais condições deveriam ser fixadas nos 5 multiplexadores para executar esta instrução? Diga se alguma condição de multiplexador é irrelevante.

2. (3,0 pontos) Considere as linhas de código VHDL abaixo, que são partes da descrição do MIPS monociclo (MIPS_V0). Este trecho contém a definição dos sinais **`uins.ce`**, **`uins.rw`** e **`uins.wreg`** do bloco de dados, que correspondem aos três sinais de controle que comandam o acesso à memória e a escrita no banco de registradores. Assuma como verdadeiro o fato que as 9 instruções da MIPS monociclo são codificadas em 4 bits com valores na faixa de 0000 a 1000, em ordem crescente de valores de acordo com a sequência: (ADDU, SUBU, AND, OR, XOR, NOR, LW, SW, ORI). Pede-se: (a) Defina os códigos associados às instruções mencionadas no VHDL; (b) Usando os códigos definidos no item (a), desenhe um diagrama esquemático que use apenas portas lógicas e componentes tais como multiplexadores, demultiplexadores e decodificadores, e que corresponda a um circuito correto e completo que pode resultar da conversão do código VHDL dado para hardware.

Dicas: (*i*) Assuma que as entradas do esquemático são todos os sinais não atribuídos no trecho VHDL dado.

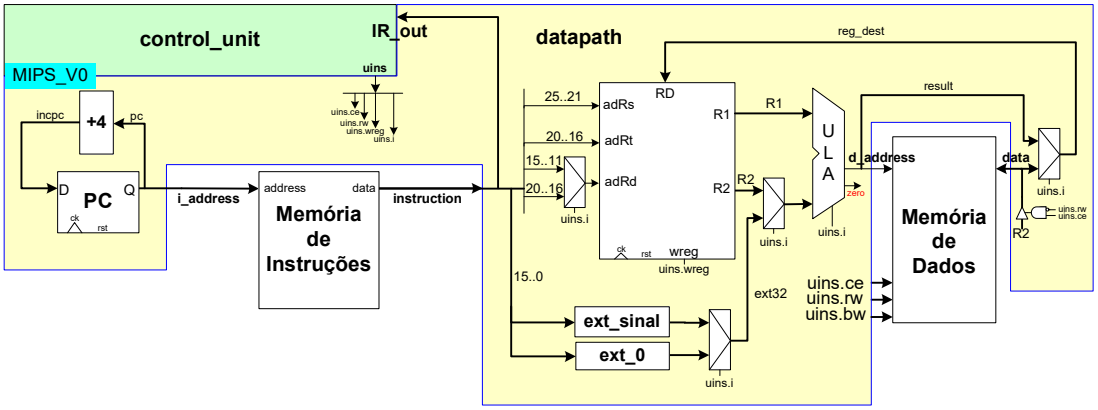
```
1. uins.ce    <= '1' when i=SW  or i=LW else '0';
2. uins.rw    <= '0' when i=SW  else '1';
3. uins.wreg  <= '0' when i=SW  else '1';
```

3. [4,0 pontos] Assuma uma frequência de operação de 800 MHz para o processador **MIPS monociclo** e assuma que a organização original foi alterada para dar suporte a executar todas as instruções do programa abaixo, mantendo a característica monociclo. Calcule:
   a) (1,5 pontos) O número de ciclos de relógio que leva p/ executar o programa, com a área de dados fornecida;
   b) (1 ponto) O tempo de execução do programa em segundos (1ns=10^{-9} segundos);
   c) (1 ponto) O que faz este programa, do ponto de vista semântico?
   d) (0,5 pontos) Este programa possui subrotina(s)? Se sim, onde esta(s) se encontra(m)? Defina o intervalo de linhas.

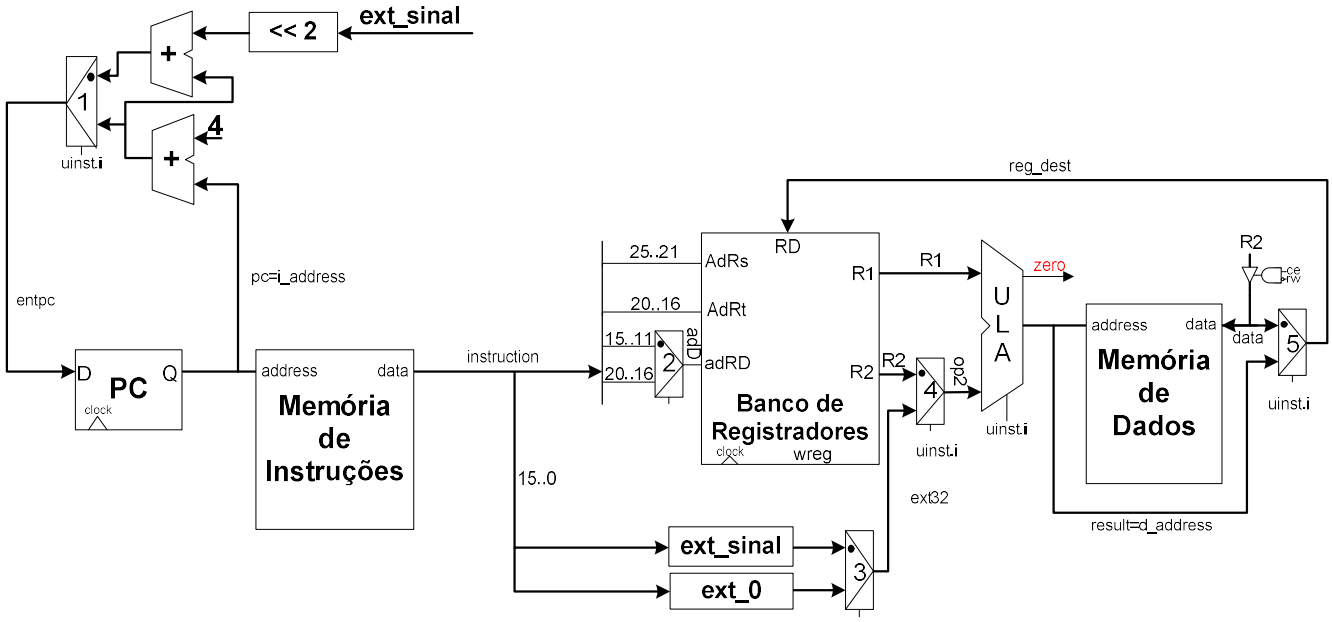
```
1.          .data
2.  n:      .word  10
3.  r:      .word  0
4.          .text
5.          .globl main
6.  main:   la     $t0,n
7.          lw     $t0,0($t0)
8.          addiu  $sp,$sp,-8
9.          sw     $ra,0($sp)
10.         sw     $t0,4($sp)
11.         jal    d2
12.         lw     $ra,0($sp)
13.         lw     $t0,4($sp)
14.         addiu  $sp,$sp,8
15.         la     $t1,r
16.         sw     $t0,0($t1)
17.         li     $v0,10
18.         syscall
19. d2:     lw     $t0,4($sp)
20.         slti   $t1,$t0,2
21.         beq    $t1,$zero,nm2
22.         sw     $zero,4($sp)
23.         jr     $ra
24. nm2:    addiu  $t1,$t0,-2
25.         bne    $t1,$zero,n2
26.         addiu  $t1,$zero,1
27.         sw     $t1,4($sp)
28.         jr     $ra
29. n2:     lw     $t0,4($sp)
30.         addiu  $t0,$t0,-2
31.         addiu  $sp,$sp,-8
32.         sw     $ra,0($sp)
33.         sw     $t0,4($sp)
34.         jal    d2
35.         lw     $ra,0($sp)
36.         lw     $t0,4($sp)
37.         addiu  $sp,$sp,8
38.         addiu  $t0,$t0,1
39.         sw     $t0,4($sp)
40.         jr     $ra
```

Gabarito

Para realizar a prova, refiram-se as propostas de organização MIPS monociclo e multiciclo vistas em aula. O desenho da versão monociclo aparece abaixo, com detalhamento do Bloco de Dados. O Bloco de Dados da versão multiciclo encontra-se no verso. Assuma que as instruções às quais a organização monociclo dá suporte de execução são apenas as seguintes, exceto se a questão particular especificar de outra forma: **ADDU, SUBU, AND, OR, XOR, NOR, LW, SW e ORI**.



1. [3,0 pontos] Considere o bloco de dados monociclo apresentado abaixo. Ele resultou da transformação do bloco de dados visto em aula para dar suporte às novas instruções **BEQ** e **LUI**. O ponto nos multiplexadores indica condição verdadeira (ou, equivalentemente, sinal de controle do mux em 1). Assuma que as instruções às quais este processador dá suporte de execução são apenas as seguintes: **ADDU, SUBU, AND, OR, XOR, NOR, LW, SW, ORI, BEQ e LUI.**



a) Suponha que existe uma única falha, que afeta apenas o multiplexador que escolhe o valor a ser escrito no banco de registradores (Mux 5). A falha é que o sinal de controle deste multiplexador está grudado eternamente no valor 0. Diga quais instruções ainda podem ser executadas corretamente, justificando para cada uma delas.

b) Observe na documentação do MIPS a funcionalidade da instrução ADDIU e diga se e como o hardware acima deveria ser alterado para ser capaz de executar esta instrução. Que operação deve ser realizada na ULA? Quais condições deveriam ser fixadas nos 5 multiplexadores para executar esta instrução? Diga se alguma condição de multiplexador é irrelevante.

Solução:

a) O problema afeta apenas as instruções que necessitam trazer um valor da Memória de Dados para dentro do processador, pois o único caminho que não pode ser mais trilhado é da saída desta para a

linha reg_dest (entrada do Banco de Registradores). Assim, a única instrução que não funciona mais é a **LW**. Nenhuma das demais instruções usa o caminho que foi bloqueado.

b) A instrução **ADDIU** usa no bloco de dados os mesmos caminhos que a instrução ORI, com exceção da passagem pelo bloco de extensão, onde se deve usar extensão de sinal e não extensão de 0. Outra diferença em relação ao ORI é que para a ADDIU a ULA deve executar uma operação de SOMA bit a bit de suas entradas op1 e op2. No bloco de controle, deve-se acrescentar a lógica de decodificação do ADDIU e acrescentar o nome desta no tipo inst_type. No bloco de dados absolutamente nada muda. Nem a ULA é alterada, visto que sua operação *default* é a de soma. As condições dos muxes são as seguintes:

- Controle do Mux 1 = 0 (incrementa o PC);
- Controle do Mux 2 = 0 (instruction(20 downto 16) define o registrador destino);
- Controle do Mux 3 = 1 (deixa passar extensão de sinal);
- Controle do Mux 4 = 0 (deixa passar extensão de sinal para a entrada op2 da ULA);
- Controle do Mux 5 = 0 (saída da ULA, result, passa para reg_dest).

Não existe sinal de controle irrelevante em nenhum mux.

2. (3,0 pontos) Considere as linhas de código VHDL abaixo, que são partes da descrição do MIPS monociclo (MIPS_V0). Este trecho contém a definição dos sinais **`uins.ce`**, **`uins.rw`** e **`uins.wreg`** do bloco de dados, que correspondem aos três sinais de controle que comandam o acesso à memória e a escrita no banco de registradores. Assuma como verdadeiro o fato que as 9 instruções da MIPS monociclo são codificadas em 4 bits com valores na faixa de 0000 a 1000, em ordem crescente de valores de acordo com a sequência: (ADDU, SUBU, AND, OR, XOR, NOR, LW, SW, ORI). Pede-se: (a) Defina os códigos associados às instruções mencionadas no VHDL; (b) Usando os códigos definidos no item (a), desenhe um diagrama esquemático que use apenas portas lógicas e componentes tais como multiplexadores, demultiplexadores e decodificadores, e que corresponda a um circuito correto e completo que pode resultar da conversão do código VHDL dado para hardware.

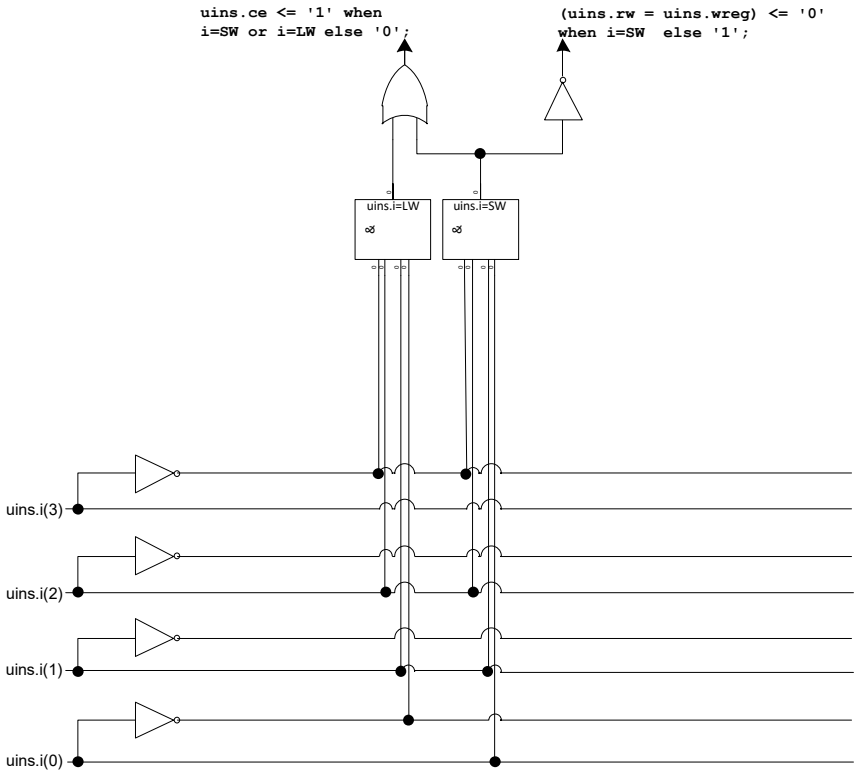
Dicas: (*i*) Assuma que as entradas do esquemático são todos os sinais não atribuídos no trecho VHDL dado.

```
1. uins.ce    <= '1' when i=SW  or i=LW else '0';
2. uins.rw    <= '0' when i=SW  else '1';
3. uins.wreg  <= '0' when i=SW  else '1';
```

Solução:

a) A solução desta parte da questão é trivial. Dado o pressuposto, a codificação é óbvia: ADDU=0000, SUBU=0001, AND=0010, OR=0011, XOR=0100, NOR=0101, LW=0110, SW=0111 e ORI=1000.

b) Usando as dicas fornecidas, a solução desta questão é o diagrama de esquemáticos abaixo:



3. [4,0 pontos] Assuma uma frequência de operação de 800 MHz para o processador **MIPS monociclo** e assuma que a organização original foi alterada para dar suporte a executar todas as instruções do programa abaixo, mantendo a característica monociclo. Calcule:
   a) (1,5 pontos) O número de ciclos de relógio que leva p/ executar o programa, com a área de dados fornecida;
   b) (1 ponto) O tempo de execução do programa em segundos (1ns=10^{-9} segundos);
   c) (1 ponto) O que faz este programa, do ponto de vista semântico?
   d) (0,5 pontos) Este programa possui subrotina(s)? Se sim, onde esta(s) se encontra(m)? Defina o intervalo de linhas.

```
1.          .data               # 0 ciclos - diretiva
2.  n:      .word  10           # 0 ciclos - definição de dados
3.  r:      .word  0            # 0 ciclos - definição de dados
4.          .text               # 0 ciclos - definição de dados
5.          .globl main
6.  main:   la     $t0,n        # 2 ciclos - pseudo-instrução tranformada em lui+ori
7.          lw     $t0,0($t0)   # 1 ciclo - coloca valor de n em $t0
8.          addiu  $sp,$sp,-8   # 1 ciclo - aloca duas posições na pilha
9.          sw     $ra,0($sp)   # 1 ciclo - empilha end retorno para o SO
10.         sw     $t0,4($sp)   # 1 ciclo - empilha valor original de n
11.         jal    d2           # 1 ciclo - chama d2(n)
12.         lw     $ra,0($sp)   # 1 ciclo - recupera endereço de retorno para o SO
13.         lw     $t0,4($sp)   # 1 ciclo - recupera resultado da avaliação de d2(n)
14.         addiu  $sp,$sp,8    # 1 ciclo - desaloca pilha
15.         la     $t1,r        # 2 ciclos - pseudo-instrução tranformada em lui+ori
16.         sw     $t0,0($t1)   # 1 ciclo - escreve resultado da chamada d2(n) em r
17.         li     $v0,10       # 1 ciclo - pseudo-instrução tranformada em addiu
18.         syscall             # 1 ciclo - Cai fora do programa
19. d2:     lw     $t0,4($sp)   # 1 ciclo - recupera argumento n da pilha
20.         slti   $t1,$t0,2    # 1 ciclo - testa se n < 2
21.         beq    $t1,$zero,nm2# 1 ciclo - se n >= 2, salta
22.         sw     $zero,4($sp) # 1 ciclo - senão coloca 0 abaixo do topo da pilha
23.         jr     $ra          # 1 ciclo - e retorna
24. nm2:    addiu  $t1,$t0,-2   # 1 ciclo - n >= 2, então subtrai 2 de n
25.         bne    $t1,$zero,n2 # 1 ciclo - Se não chegou a zero, salta
26.         addiu  $t1,$zero,1  # 1 ciclo - Se chegou a 0 gera 1 em $t1
27.         sw     $t1,4($sp)   # 1 ciclo - Coloca resultado 1 abaixo do topo da pilha
28.         jr     $ra          # 1 ciclo - e retorna
29. n2:     lw     $t0,4($sp)   # 1 ciclo - n-2 > 0. Pega de novo valor de n da pilha
30.         addiu  $t0,$t0,-2   # 1 ciclo - subtrai 2 de n
31.         addiu  $sp,$sp,-8   # 1 ciclo - Prepara chamada recursiva, alocando pilha
```

```
32.       sw     $ra,0($sp)    # 1 ciclo - guarda endereço de retorno a quem chamou esta
   instância de d2
33.       sw     $t0,4($sp)    # 1 ciclo - guarda n atualizado
34.       jal    d2            # 1 ciclo - Faz chamada recursiva
35.       lw     $ra,0($sp)    # 1 ciclo - Na volta, recupera end. De retorno para quem
   chamou
36.       lw     $t0,4($sp)    # 1 ciclo - recupera n calculado na recursão
37.       addiu  $sp,$sp,8     # 1 ciclo - desaloca pilha da chamada anterior
38.       addiu  $t0,$t0,1     # 1 ciclo - incrementa n
39.       sw     $t0,4($sp)    # 1 ciclo - guarda na pilha de quem chamou e
40.       jr     $ra           # 1 ciclo - retorna
```

Solução:

a) O rótulo main identifica um programa principal (entre linhas 6 a 18) que chama uma subrotina e executa exatamente uma vez (sem laços). Logo o tempo de execução do programa principal é de exatamente **15 ciclos** (2 pseudos de 2 ciclos, uma pseudo de 1 ciclo e 10 instruções)
A subrotina que inicia no rótulo d2 contém o restante do código (linhas 19 a 40). Esta subrotina pode retornar ao programa principal por um de 3 pontos (instruções jr $ra nas linhas 23, 28 e 40). A subrotina é recursiva, pois como se vê na linha 34 (jal d2), contém uma chamada para si mesma dentro do seu código. O retorno pela linha 23 acontece apenas quando a subrotina recebe um parâmetro com valor 0 ou 1. O retorno pela linha 28 apenas acontece quando a subrotina recebe n=2 e o retorno pela linha 40 sempre ocorre depois de uma recursão:
   i. Com os dados fornecidos (n=10) a subrotina recursiva vai entrar em recursão e a cada nível desta recursão n é decrementado de 2;
   ii. Assim, há 5 níveis de recursão, com n=10, 8, 6, 4 e 2. Retorna-se pela linha 28 no último nível da recursão e pela linha 40 nos níveis 1 a 4.
   iii. A partir de d2, executa-se 3 ciclos e salta-se a nm2. De nm2 executam-se 2 ciclos e salta-se para n2. Chegando-se em n2 sempre se executam 12 ciclos (excluindo-se os ciclos da recursão para níveis inferiores.
   iv. De iii acima, conclui-se que cada nível de recursão de 1 a 4 gasta 3+2+12=**17 ciclos** de relógio.
   v. O nível 5 sai pela linha 28, depois de executar por 3+5=**8 ciclos** de relógio

   Assim, o número total de ciclos gastos para executar o programa é 15+17*4+8=**91 ciclos**.

b) Como o relógio é de 800MHz, o período deste é $1/800*10^6=1{,}25x10^{-9}$s. Logo, o tempo de execução é $91+1{,}25x10^{-9}$s=$1{,}1375x10^{-7}$s.

c) Este programa computa o resultado de dividir n por 2 de forma recursiva.

d) Sim, este programa possui uma subrotina recursiva, d2, localizada entre as linhas 19 e 40 do programa.