UFPR – Bacharelado em Ciência da Computação
CI210 – Projetos Digitais e Microprocessadores — 2011-2

Primeira Prova

1) Um *árbitro* é um circuito seqüencial com N entradas e N saídas, cuja função é determinar qual dentre N requisições será atendida.

Considere um árbitro com 3 entradas. Inicialmente todas as requisições estão inativas ($E = e_2e_1e_0 = 000$) e todas as respostas estão inativas ($S = s_2s_1s_0 = 000$). Uma requisição é sinalizada por $e_i = 1$, e é atendida quando o árbitro faz $s_i = 1$. Quando o requisitante completou sua tarefa, ele retira a requisição fazendo $e_i = 0$, ao que o árbitro responde com $s_i = 0$, tão logo quanto possível. Novas requisições que ocorram enquanto alguma está sendo atendida são ignoradas. Após o encerramento de uma tarefa, uma nova rodada de arbitragem é iniciada.

Projete um árbitro para $N = 3$. Seu projeto deve conter: (i) um circuito combinacional que decide qual das três entradas será atendida, segundo a prioridade dada por $e_3 > e_2 > e_1$; (ii) uma máquina de estados que atende e/ou nega os pedidos, conforme descrito acima, implementada com flip-flops tipo D. [15 pontos]
(iii) [5 pontos extra] Projete uma máquina de estados que atenda as requisições circularmente: se o requisitante i está sendo atendido, ou foi o último atendido, então o próximo a ser atendido é aquele de índice $(i + 1) \bmod N$.

2) Traduza para *assembly* do MIPS o trecho de programa abaixo: [15 pontos]
Para facilitar a correção indique os registradores como `ri`, `rv`, *etc.*

```
typedef struct A {
   int a;
   int b;
   int c;
   ind d;
} Atype;
```

```
Atype v[1024];
int w[1024];
int p[64];
int i;
...
for (i=0; i < 1024; i+=2) {
   w[i] = v[ p[i%64] ].a / v[ p[i%64] ].b;  // div
   w[i+1] = v[ p[i%64] ].c % v[i].d;        // mod
}
```

Segunda Prova

1) Você deve acrescentar ao processador o circuito que suporta uma nova instrução que executa a multiplicação do registrador `rs` por 2 elevado ao conteúdo de `rt`, e soma o produto ao conteúdo do registrador `rd`. O resultado é armazenado nos registradores especiais `hi` e `lo`. Esta nova instrução é chamada de *multiply-add*, `madd`, com resultado em 64 bits.
MULTIPLY-ADD: `madd rd,rs,rt  # hi&lo ← rs×2^rt + rd` (formato R)
Um dos fatores da multiplicação é sempre uma potência de dois, e por limitações da tecnologia de implementação, o tempo de propagação do circuito que efetua a operação `madd` é equivalente à soma dos tempos de propagação da ULA e do acesso à memória. Mostre como acrescentar a instrução `madd` ao conjunto de instruções do processador. Sua resposta consiste de quatro partes:

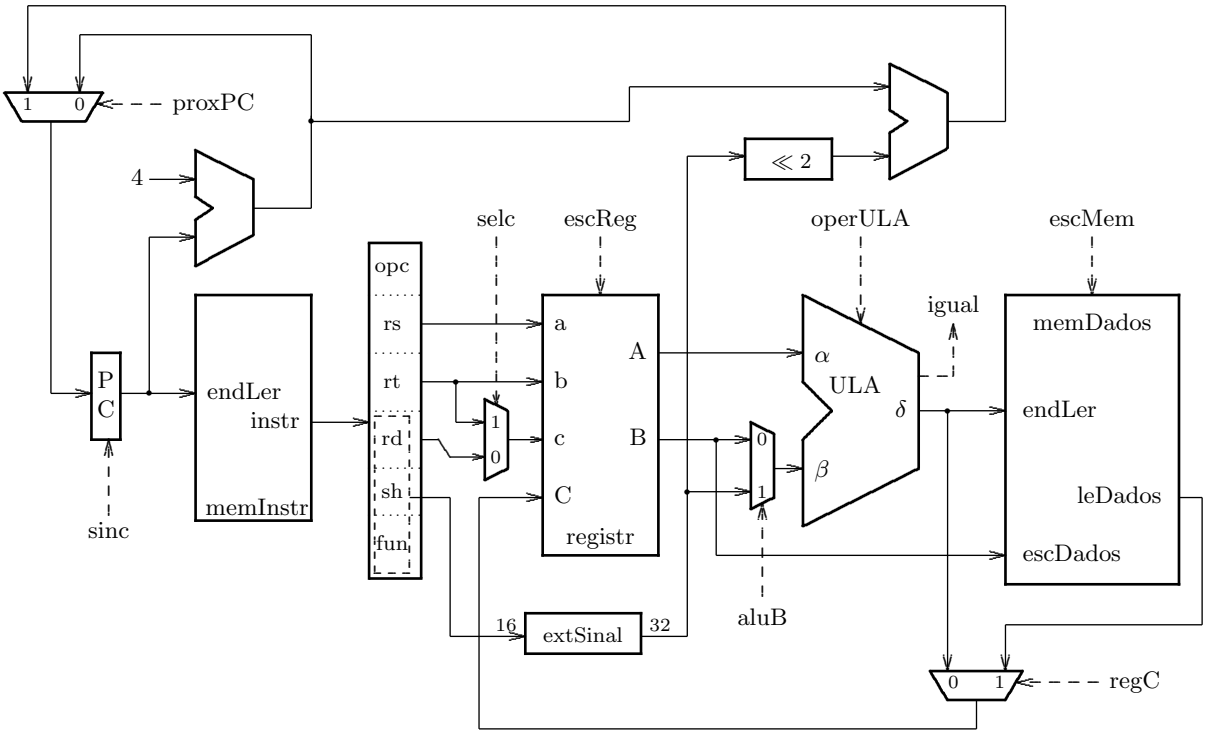
(i) um diagrama de blocos detalhado, claro, e limpo, mostrando os componentes e as ligações internas da nova unidade funcional; [5 pontos]
(ii) um desenho claro e limpo, indicando os circuitos e ligações adicionais para interligar a nova unidade funcional ao processador. *Pista:* são necessários os circuitos para as instruções `mflo` e `mfhi` *(move from {lo,hi})*; [7,5 pontos]
(iii) um diagrama de tempos com a temporização da execução de `madd`; [7,5 pontos]
(iv) traduza o programa abaixo para assembly do MIPS. [10 pontos]
Para facilitar a correção indique os registradores como `ra`, `rb`, *etc.*

```
#define i64 (long long) // inteiro de 64 bits
i64 X[1024];
int A[1024], B[1024], C[1024];

for (i=0; i<1024; i++)
   X[i] = (i64)( (A[i] * 1<<B[i]) + C[i] ); //  (1 << β) = 2^β
```

Substitutiva

1) Mostre como implementar no processador as instruções `jal` *(jump and link)*, e `jr` *(jump-register)*. Desenhe um diagrama claro e limpo em sua folha de respostas com os circuitos adicionais relevantes. [10 pontos]



2) Mostre como projetar (projeto detalhado) um registrador de deslocamento com 4 bits de largura ($Q_0Q_1Q_2Q_3$), com carga paralela (sinal `load`) nos sinais $D_0D_1D_2D_3$, inicialização assíncrona (`reset`), e deslocamento de uma posição a cada pulso do relógio em que não ocorra uma carga, a partir da entrada D_0. Escreva um modelo em código VHDL para o seu circuito. [10 pontos]

3) Traduza para *assembly* do MIPS o trecho de código C abaixo. [10 pontos]

```
// find minimum number of bits required to represent N as an unsigned binary number
int log2_ceil(int n) {
   if (n < 2)
      return 0;
   else
      return 1 + log2_ceil(n/2);
}
```

Exame Final

1) Escreva um modelo funcional em VHDL para uma unidade de lógica e aritmética com 32 bits de largura que suporte as operações de soma, subtração, *and*, *or*, *not* e *xor*. Os bits de *status* são `zero` e `negativo`. [40 pontos]

2) Mostre como implementar um contador síncrono módulo 16 com FFs do tipo D. [20 pontos]

3) Escreva um programa em C que computa a soma dos elementos da diagonal de uma matriz quadrada de NxN inteiros (N<1024). Traduza seu programa para *assembly* do MIPS. *Para facilitar a correção indique os registradores como* `ra`, `rb`, *etc.* [40 pontos]