UFPR – Bacharelado em Ciência da Computação
CI210 – Projetos Digitais e Microprocessadores — 2013-2

Primeira Prova

1. Projete um circuito que efetua a multiplicação de dois números positivos de 4 bits, como especificado na Equação 1. O circuito deve implementar a multiplicação por adições repetidas, *e não com somas e deslocamentos.*

Seu projeto deve conter o circuito de dados e a máquina de estados que controla o circuito de dados.

Os dois operandos de 4 bits devem ser inicialmente carregados nos registradores *mndo* e *mdor*. Um sinal de entrada da interface externa ao multiplicador *inicia* carrega os operandos e dispara a multiplicação. O produto é disponibilizado no registrador *prod* quando o sinal *pronto* for ativado pela máquina de estados que controla o multiplicador. [15 pontos]

A função $num : B^n \mapsto \mathbb{N}$ retorna o inteiro correspondente ao seu argumento, que é um número representado em binário.

$$
\begin{array}{l}
inicia, pronto : \mathbb{B} \\
mndo, mdor : \mathbb{B}^4 \\
prod : \mathbb{B}^8 \\
prod4x4 : \mathbb{B} \times (\mathbb{B}^4 \times \mathbb{B}^4) \mapsto \mathbb{B}^8 \times \mathbb{B} \\
prod4x4(inicia, mndo, mdor, prod, pronto) \ \equiv \\
\qquad inicia \ \Rightarrow \ pronto \wedge [\, num(prod) = num(mndo) \times num(mdor) \,]
\end{array}
\tag{1}
$$

2. Projete um circuito sequencial síncrono que produz em sua saída a sequência mostrada ao lado. Seu projeto deve empregar quatro FFs tipo T (T de "tatu"), e a resposta deve mostrar claramente todos os passos da solução. [10 pontos]

0000 → 1100 → 1001 → 0110 → 0011 → 1111 → 0000 (repete)

3. Projete um circuito combinacional que é um deslocador com 8 bits na entrada ($e_i, i \in \{7..0\}$) e que permite deslocamentos de nenhuma, uma, ou duas posições para a esquerda. Dependendo do sinal de controle, o bit de saída s_i pode mostrar qualquer um dentre $\{e_i, e_{i+1}, e_{i+2}\}$.
(i) Sua resposta deve conter o circuito que efetua os deslocamentos do bit s_i, bem como a composição de 8 destes circuitos para implementar o deslocador de 8 bits. Use multiplexadores. Atenção com os bits de saída s_7 e s_0.
(ii) Estenda o circuito de forma a manter o sinal de números representados em complemento de dois nos deslocamentos. [10 pontos]

Segunda Prova

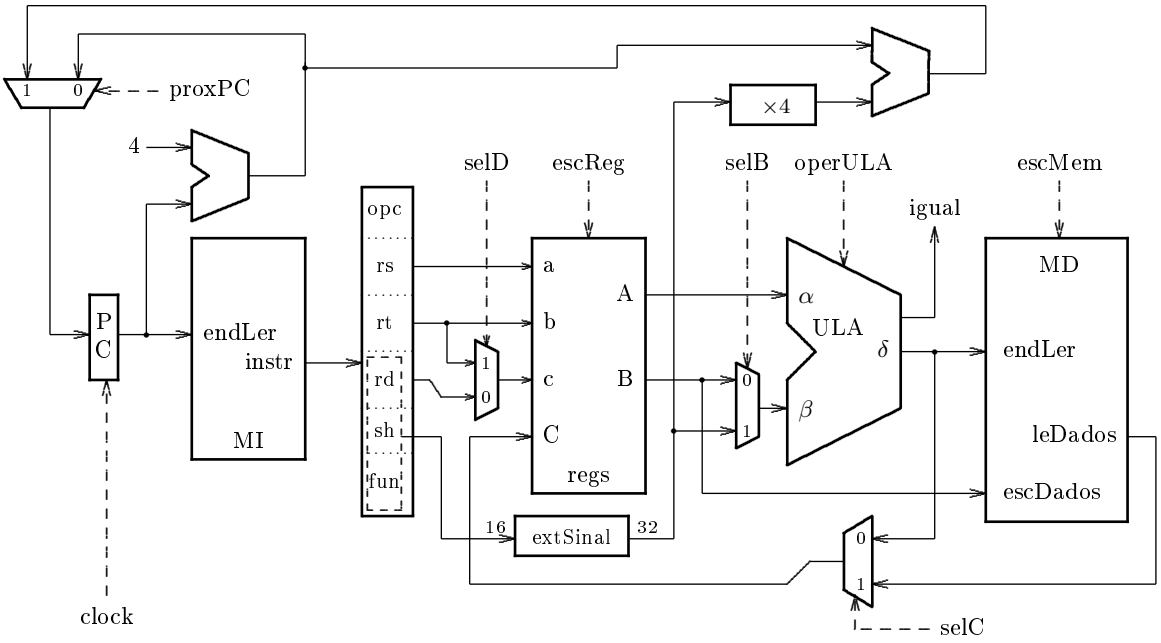
4. Mostre como implementar a instrução LOAD-WORD-SCALED (`lws`), que emprega um novo modo de endereçamento, chamado de *base-deslocamento escalado* e definido como

endereço efetivo = regBase + $2^{\text{escala}} \times$ desloc .

O registrador base (R[rs]) e o deslocamento são os mesmos campos que na instrução LOAD-WORD. A *escala* é obtida do registrador R[rt], e este registrador é sobrescrito ao final da instrução. Sua resposta deve conter:
(i) indicação clara da implementação do circuito combinacional do deslocador; [5 pontos]
(ii) indicação clara das modificações ao circuito do processador (pode ser respondido na folha de perguntas); [5 pontos]

(iii) a tabela com os sinais de controle para a intrução `lws`; [5 pontos]
(iv) um diagrama de tempos completo da execução desta instrução. [5 pontos]

`lws rt,rs(desl)` `# R[rt] ← mem[ R[rs] + ( 2^R[rt] × extS(desl) ) ]` (formato I)



5. Traduza para *assembly* do MIPS o trecho de programa ao lado. Seu código *assembly* deve empregar as convenções de programação do MIPS. *Para facilitar a correção indique os registradores que não são usados na convenção de chamada de funções como* `re`, `rn`, *etc. Use uma folha inteira para escrever seu programa em assembly.* [15 pontos]

```
int a, i;
int x[2048], y[256];
...
i=0;
a=0;
while (i < 1024) {
   a = a + x[i] + y[ (x[i] % 256) ]; // MOD
   i = i + 1;
}
```

Exame Final

6. Traduza para *assembly* do MIPS o trecho de programa ao lado. Seu código *assembly* deve empregar as convenções de programação do MIPS. *Para facilitar a correção indique os registradores que não são usados na convenção de chamada de funções como* `re`, `rn`, *etc. Use uma folha inteira para escrever seu programa em assembly.* [40 pontos]

```
int v; int X[1024], Y[1024];
...
int reduz(int *v, int vSz, int *w, int wSz) {
   int i=1; int a=0;
    while (i < vSz) {
      a = a + v[i] + w[ (v[i] % wSz) ]; // MOD
      i = i*2;
   }
   return a;
}
...   v = reduz(X, 1024, Y, 1024); ...
```

7. Projete um circuito sequencial síncrono que produz em sua saída a sequência mostrada ao lado. Seu projeto deve empregar FFs tipo T, e a resposta deve mostrar claramente todos os passos da solução. [30 pontos]

$0000 \rightarrow 1000 \rightarrow 1100 \rightarrow 0110 \rightarrow 0011 \rightarrow 0001 \rightarrow 0000 \rightarrow \cdots$

8. Mostre como implementar a instrução JUMP-AND-LINK REGISTER (`jalr`). A vírgula indica "execução concorrente".

`jalr rt,rs  # R[rt]←PC+4 , PC←R[rs]  jump-and-link-register` (formato R)

Sua resposta deve conter:
(a) indicação clara das modificações ao circuito do processador; [10 pontos]
(b) a tabela com os sinais de controle para a intrução `jalr`; [10 pontos]
(c) um diagrama de tempos completo da execução desta instrução. [10 pontos]