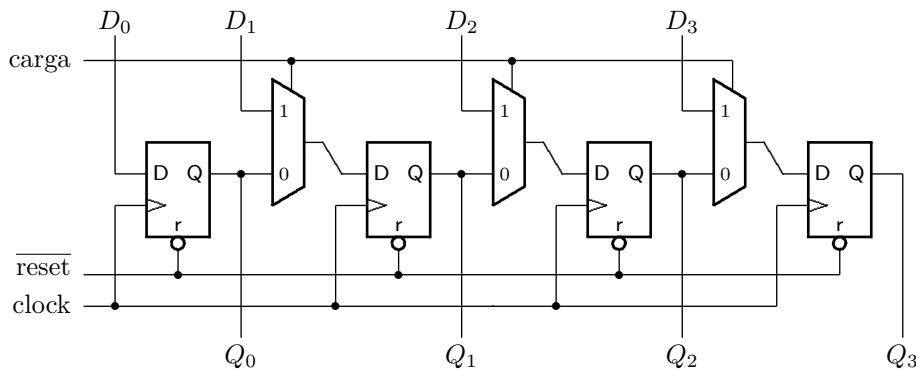


### Primeira Prova

- 1) Projete um circuito combinacional que produz em sua saída uma versão deslocada da entrada, conforme a especificado na Equação 1. Sua resposta deve mostrar claramente todos os passos da solução. [10 pontos]

$$\begin{aligned}
 E &: \mathbb{B}_4 && \text{entrada} \\
 S &: \mathbb{B}_{13} && \text{saída} \\
 d &: \mathbb{B}_2 && \text{controle} \\
 \text{circ} &: (\mathbb{B}_4 \times \mathbb{B}_2) \mapsto \mathbb{B}_{13} \\
 \text{circ}(E, S, d) &\equiv S = E \times 2^d, \quad 2^d \in \{1, 8, 64, 512\}
 \end{aligned} \tag{1}$$

- 2) Analise o circuito abaixo e desenhe um diagrama de tempo com os sinais *clock*, *reset*, *carga*,  $D_{0..3}$  e  $Q_{0..3}$ , considerando que  $D_{0..3} = 1001$  ( $D_{0..3}$  permanece estável) e que ocorre um pulso em *carga* com largura de um período do relógio, meio ciclo após o sinal *reset* ficar inativo. [10 pts]



- 3) Projete um circuito sequencial síncrono que produz em sua saída a sequência mostrada abaixo. Seu projeto deve empregar FFs tipo T, e a resposta deve mostrar claramente todos os passos da solução. [10 pontos]

000 → 001 → 011 → 010 → 110 → 111 → 101 → 100 → 000 → ...

### Segunda Prova

- 2) 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*. [10 pontos]

```

...
x = power(y,z);
...

int power(int n,int exp) {
    if (exp > 1)
        return (n * power(n,exp-1));
    else
        return (n);
}

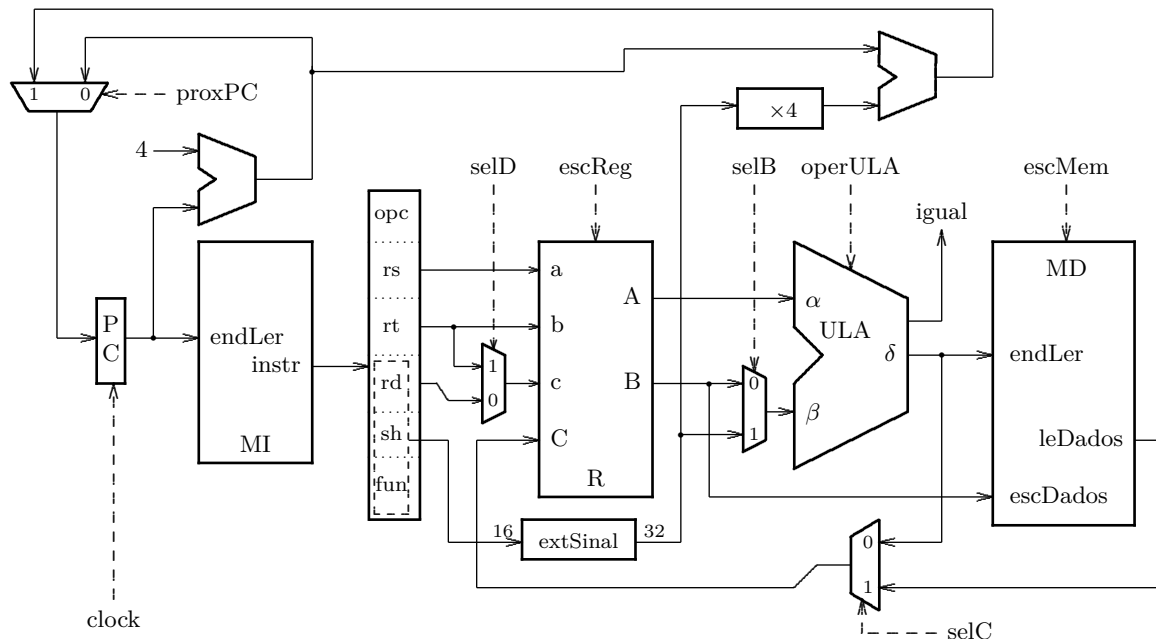
```

- 1) Mostre como implementar a instrução JUMP THROUGH MEMORY definida abaixo. Sua resposta deve conter: (i) indicação clara de quaisquer modificações necessárias ao circuito do processador, que podem ser desenhadas na folha de perguntas; (ii) uma tabela de sinais de controle indicando

os sinais ativos durante a execução desta instrução; e (iii) um diagrama de tempos completo da execução desta instrução.

```
jtm desl(rs)      # PC←MD[ desl + regs[rs] ]
```

(formato I)



# Exame Final

- 1) Mostre como implementar um circuito que multiplica dois números de quatro bits. Sua resposta deve conter um diagrama de blocos do circuito de dados e um diagrama para a máquina de estados que controla o multiplicador. [25 pontos]
- 2) Mostre como projetar 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  $D_0$ . Sua resposta deve conter um diagrama de blocos com *flip-flops* e outros componentes que sejam necessários. [25 pontos]
- 3) Escreva um programa em C que computa a soma dos elementos da diagonal de uma matriz quadrada de  $N \times N$  inteiros ( $N < 1024$ ). Traduza seu programa para *assembly* do MIPS. Para facilitar a correção indique os registradores como ra, rb, etc. [25 pontos]
- 4) Mostre como implementar a instrução BRANCH-AND-LINK definida abaixo. Sua resposta deve conter: [25 pontos]
  - (i) indicação clara das modificações no circuito (*pode ser desenhado na folha de perguntas*);
  - (ii) a tabela com os sinais de controle;
  - (iii) um diagrama de tempos completo da execução desta instrução.

```
bal ender    # regs[31] ← PC+8 , PC ← (PC+4) + (extSinal(ender) × 4)
```

(formato I)