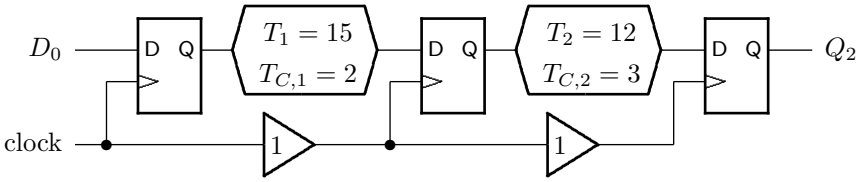
UFPR – Bacharelado em Ciência da Computação
CI1210 – Projetos Digitais e Microprocessadores — 2019-2

Primeira Prova

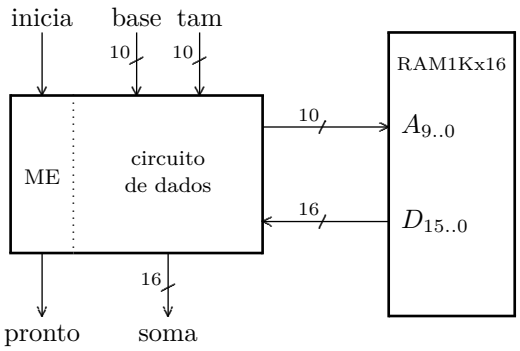
1. Calcule o período mínimo de operação adequada para o circuito abaixo. Os circuitos combinacionais têm os parâmetros de temporização indicados. Mostre todas as contas.
Os tempos são em nanosegundos. [5 pontos]



Flip-flops:
$T_{prop} = 4$, $T_{cont} = 2$,
$T_{su} = 4$, e $T_h = 2$.

2. Implemente um multiplexador de duas entradas utilizando transistores CMOS. [5 pontos]

3. Você deve projetar um sistema com as interfaces mostradas na figura abaixo. O sistema recebe um endereço inicial (`base`) e um tamanho (`tam`). Quando o sinal `inicia` é ativado, a memória RAM é percorrida desde o endereço `base` até o endereço `base+tam-1`, e os conteúdos da RAM são somados e acumulados em `soma`. Quando todas as `tam` palavras foram adicionadas, o sinal `pronto` é ativado e o valor da soma é exibido no sinal `soma`. Seu sistema deve se comportar como o trecho de código mostrado abaixo. A forma como a memória RAM é preenchida é irrelevante para o seu projeto. O período do relógio é suficientemente longo.




```
#define P 1024

short RAM[P];
short i, base, tam, soma;

soma = 0;
i = 0;
while ( i < tam ) {
   soma = soma + RAM[ base+i ];
   i = i + 1;
}
```

O projeto do circuito de dados vale 10 pontos e o projeto da máquina de estados que o controla vale 5 pontos – indique somente o diagrama de estados. Você pode usar quaisquer dos componentes estudados neste semestre **desde que a resposta mostre claramente o que cada componente é ou faz**. "Uma caixa com rabiscos" **sem uma definição ou especificação clara e completa** será considerada um erro grave. Nenhuma adivinhação favorável será empregada quando da correção.

Segunda Prova — 2019-2

4. A função `checksum()` computa a soma módulo-1 de todos os elementos de um vetor. Seus argumentos são o endereço inicial do vetor, o número de elementos, e o valor inicial da "soma", se zero ou `0xffff.ffff`. Traduza para *assembly* do MIPS o trecho de programa ao lado. Seu código *assembly* deve empregar as convenções de programação do MIPS. [10 pontos]

Para facilitar a correção indique os registradores que não são usados na convenção de chamada de funções como `ri`, `rs`, *etc.*

Use as folhas internas para escrever o programa em *assembly*.

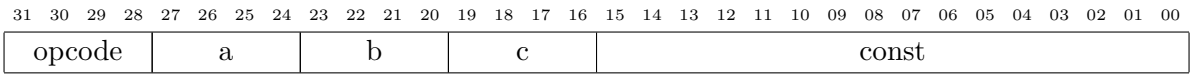
```
    ...
    int X[1024];
    ...
    s = checksum(&(X[64]), 256, -1);
    ...

int checksum(int vet[], int size,
             int init)
{
    int s, i;

    s = init;
    i = 0;
    while (i < size) {
      s = s XOR vet[i]; // XOR == ^
      i = i + 1;
    }
    return(s);
}
```

5. A codificação das instruções do Mico XII é indicada abaixo. Sua tarefa é mostrar como implementar, no Mico XII, as instruções (i) *"move from lo"* – copia o conteúdo do registrador especial LO para um registrador de uso comum; (ii) *"move from hi"* – copia o conteúdo do registrador especial HI para um registrador de uso comum; e (iii) *shift logical left any* – que produz uma cópia deslocada de N posições de `R(a)` em `R(c)` e o deslocamento é especificado por `R(b)`, assim definidas:

(i) a instrução **`mflo`** (*move from lo*) faz: `R(c)` ← `LO`; (opcode α)
(i) a instrução **`mfhi`** (*move from hi*) faz: `R(c)` ← `HI`; (opcode β)
(iii) a instrução **`slla`** (*shift logical left any*) faz: `R(c)` ← `R(a)` << `R(b)`; (opcode γ)



Suponha que existam opcodes disponíveis para as novas instruções. `hi` e `lo` contém o produto, e quociente e resto nas multiplicações e divisões. Sua resposta deve conter: [15 pontos]

(a) a codificação das novas instruções;
(b) restrições na codificação de `slla`;
(c) alterações no circuito do Mico XII – você pode usar a folha de perguntas para indicar as modificações; não esqueça de colocar seu nome na folha;
(d) o conteúdo da memória de controle para as novas instruções.

Exame Final

6. Projete um circuito combinacional que computa a maioria dentre 5 bits. Seu circuito deve produzir saída em 1 quando a maioria absoluta dos bits no quinteto é 1 (3/5, 4/5 ou 5/5). [30 pontos]

7. Suponha que seu circuito de maioria, da questão anterior, é alimentado por um registrador de cinco bits, e a resposta é coletada num registrador de um bit (um *flip-flop*). Com base nos tempos de propagação na tabela ao lado, compute o período mínimo do relógio do circuito da maioria e verifique se as condições de *setup* e *hold* são satisfeitas. [30 pontos]

	T_{prop}	T_{cont}	T_{su}	T_h
FF/reg	10	2	8	4
inv	2	1		
and-2	3	1		
and-3	4	2		
and-4	5	3		
and-5	6	4		
or-2	3	1		
or-3	4	2		
or-4	5	3		
or-5	6	4		

8. Traduza para *assembly* do MIPS o trecho de programa ao lado. Seu código *assembly* deve empregar as convenções de programação do MIPS. [40 pontos]

Para facilitar a correção indique os registradores que não são usados na convenção de chamada de funções como `rf`, `rn`, *etc.*

Use as folhas internas para escrever o programa em *assembly*.

```
   ...
   s = fib(6);
   ...
int fib(int n) {
   int c, fst, snd, nxt;

   fst = 0
   snd = 1;
   for (c = 0; c < n; c++) {
      if ( c <= 1 ) {
         nxt = c;
      } else {
         nxt = fst + snd;
         fst = snd;
         snd = nxt;
      }
   }
   return(nxt);
}
```