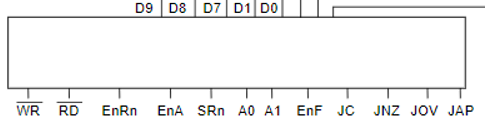


LEIM

**Computação Física – Trabalho Prático 2**

Docente – Prof. Carlos Carvalho



**Trabalho realizado por:**

- Diogo Coito Nº 50029

- Guilherme Gouveia Nº 49740

- Tomás de Matos Nº 49711

*20/05/2022*

Índice:

[Exercício 1 1](#_Toc100948642)

[A) 1](#_Toc100948643)

[B) 1](#_Toc100948644)

[C) 3](#_Toc100948645)

[D) 3](#_Toc100948646)

[E) 4](#_Toc100948647)

[Exercício 2 7](#_Toc100948648)

[A) 7](#_Toc100948649)

[B) 8](#_Toc100948650)

[C) 8](#_Toc100948651)

[D) 11](#_Toc100948652)

[Código: 12](#_Toc100948653)

[Exercício 3 18](#_Toc100948654)

[A) 18](#_Toc100948655)

[C) 20](#_Toc100948656)

[D) 20](#_Toc100948657)

[Função de estado seguinte: 22](#_Toc100948658)

[Função de saídas: 22](#_Toc100948659)

[E) 22](#_Toc100948660)

[F) 23](#_Toc100948661)

[Código: 23](#_Toc100948662)

Indice de Tabelas e Imagens:

[Tabela de codificação das instruções 3](#_Toc103946663)

[Modelo de entradas e saídas do contador: 3](#_Toc103946664)

[Modelo de Moore-Mealey: 4](#_Toc103946665)

[ASM: 5](#_Toc103946666)

[Tabela de verdade do contador: 6](#_Toc103946667)

[Mapa 1 da função estado seguinte: 7](#_Toc103946668)

[Mapa 2 da função estado seguinte: 7](#_Toc103946669)

[Mapa 1 da função de saída: 7](#_Toc103946670)

[Mapa 2 da função de saída: 7](#_Toc103946671)

[Com o auxílio de tudo o que foi demonstrado nas alíneas anteriores, chegamos assim ao seguinte circuito lógico simplificado que demonstra com mais detalhes o funcionamento do circuito: 8](#_Toc103946672)

[Circuito lógico simplificado Ex3: 8](#_Toc103946673)

[Circuito 3: 12](#_Toc103946674)

Exercício 1

**1) Registos e os respetivos bits**

Registos:

* Registo A, Rn (R0 e R1), PC, Cy/Bw, Ov e Z.

Memória de Dados:

* Adress Bus, Data Bus de M.D

Memória de Código:

* Adress Bus, Data Bus de M.C
  1. **Resolução:**

**Registo A**: 5 bits;

**Registo Rn**: 2 registos de 5 bits (R0 e R1);

**Registo PC**: 7 bits;

**Registo Cy/Bw**: 1 bit;

**Registo Ov**: 1 bit;

**Registo Z**: 1 bit;

O registo Rn tem 5 bits já que é dito no enunciado que será igual a um valor de 5 bits. Por essa razão, também o A terá 5 bits já que será igualado a Rn. PC terá 7 bits, que é o maior valor a que é igualado (end7). Cy/Bw, Ov e Z terão todos 1 bit por serem valores booleanos.

Exercício 2

**2) Memória de Dados e Código e os seus bits**

**2.1) Memória de Dados:**

* Adress Bus: 5 bits;
* Data Bus: 5 bits;

Adress Bus e Data Bus terão ambos 5 bits já que o registo A e o registo Rn são de 5 bits, respetivamente.

**2.2) Memória de Código:**

* Adress Bus: 7 bits;
* Data Bus: 10 bits;

Adress Bus terá 7 bits já que será o valor de PC que tem também 7 bits. Por sua vez, o Data Bus possui 10 bits que será o valor de saída (D9 até D0).

Exercício 3

**3) Codificação de Instruções**

**3.1) Codificação**

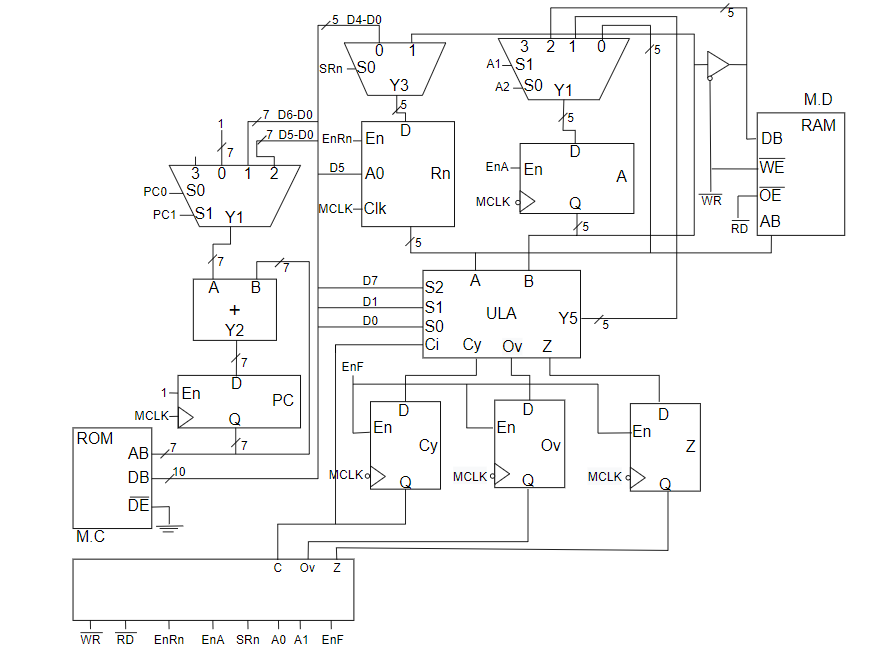
Para este projeto, decidimos usar 10 bits para codificar as instruções. Podíamos ter usado 9 bits, mas teríamos de aumentar o número de bits de dependência e preferimos manter os 10 bits.

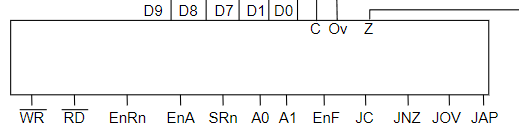
Os nossos bits de dependência são o D9, D8, D7, D1 e D0 que variam entre 0-1. A primeira instrução tem como instrução 000------- e a última 111------- sendo desta maneira os extremos.

Uma imagem com mesa

Descrição gerada automaticamente

Tabela de codificação das instruções





Uma imagem com mesa

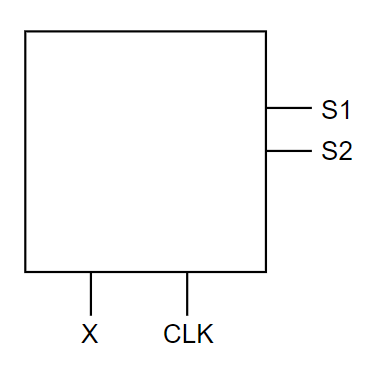
Descrição gerada automaticamente

Uma imagem com mesa

Descrição gerada automaticamente

A)

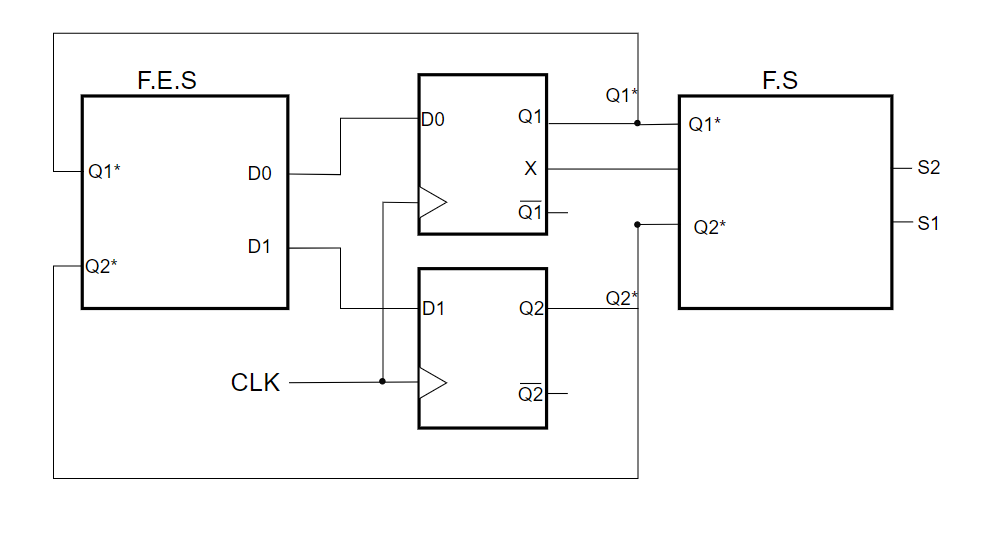
Modelo de entradas e saídas do contador:



No último exercício começámos por desenhar o modelo de entradas e saídas. O CLK representa o clock que vai ser dado ao circuito para subir ou descer de valor. O X dará a subida ou a descida, e o S2 e S1 darão o segundo e primeiro dígitos, respetivamente, do número resultante.

B)

Modelo de Moore-Mealey:

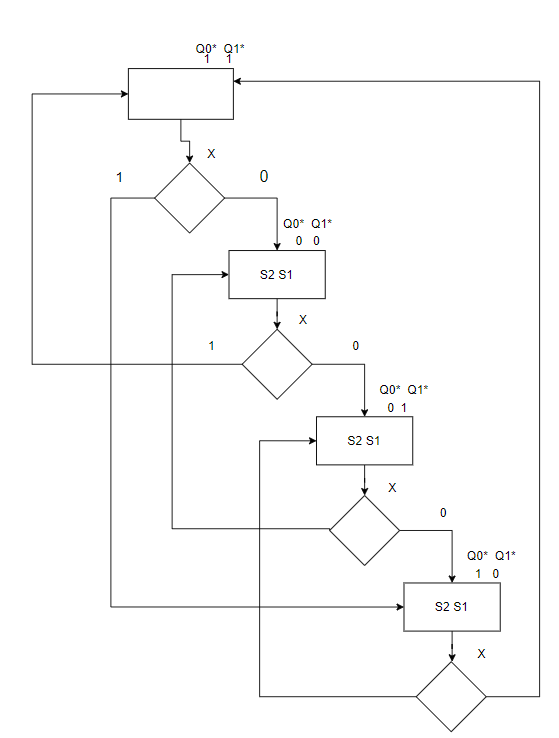


Na alínea B, desenhámos o diagrama de blocos do circuito, tendo colocado variáveis já mencionadas, acrescentando Q1\* e Q2\*, Q1 e Q2 e D0 e D1. Q1\* e Q2\* serão os dígitos 1 e 2, respetivamente, do número no estado atual. D0 e D1 serão os dígitos do número no próximo estado, tal como Q1 e Q2. À esquerda podemos observar a função estado seguinte que definirá qual será o próximo estado. No meio as funções que decidirão se se passa para o próximo estado e à direita a função de saída dependente de Q1\* e Q2\*.

C)

Na figura abaixo podemos observar os diferentes estados possiveis no circuito:

ASM:



D)

Na alínea D) realizámos uma tabela de verdade apresentada em baixo onde X representa a soma (0) ou subtração (1), Q1\* e Q0\* representam os dígitos de 1 e 0, respetivamente, do número no presente estado. D1 e D0 representam os dígitos do próximo estado e S1 e S0 os dígitos do número resultante:

Tabela de verdade do contador:

Uma imagem com mesa

Descrição gerada automaticamente

Em baixo, podem ser vistos com mais detalhe os mapas das diferentes funções presentes no nosso circuito:

Mapa 1 da função estado seguinte:

Uma imagem com mesa

Descrição gerada automaticamente

Mapa 2 da função estado seguinte:

Uma imagem com mesa

Descrição gerada automaticamente

Mapa 1 da função de saída:

Uma imagem com mesa

Descrição gerada automaticamente

Mapa 2 da função de saída:

Uma imagem com mesa

Descrição gerada automaticamente

Com os mapas, chegámos então às seguintes expressões lógicas:

Função de estado seguinte:

D0 = Q0\*

D1 = Q1\*⊕Q0\*

Função de saídas:

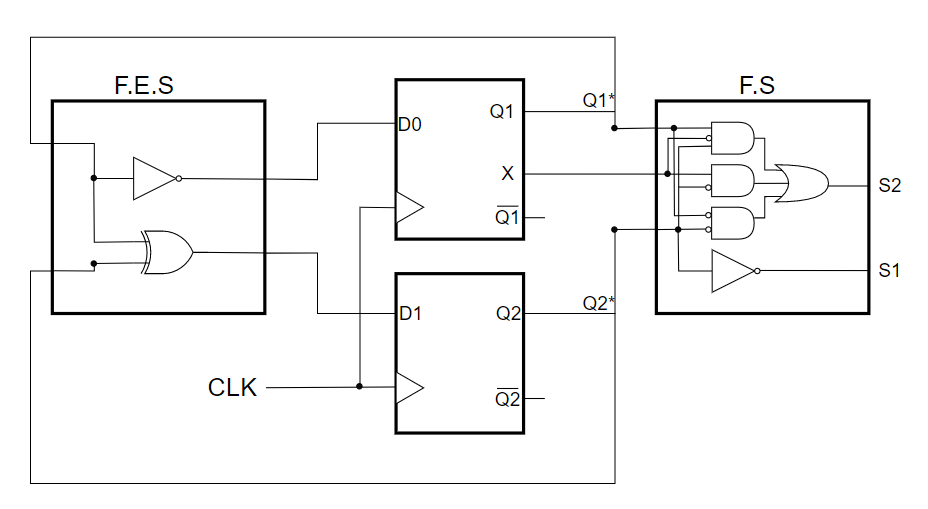
S0 = Q0\*

S1 = Q1\*.Q0\* + X.Q1 + X.Q1.Q0

E)

Com o auxílio de tudo o que foi demonstrado nas alíneas anteriores, chegamos assim ao seguinte circuito lógico simplificado que demonstra com mais detalhes o funcionamento do circuito:

Circuito lógico simplificado Ex3:



F)

Ao contrário da montagem do exercicio anterior, esta foi bastante mais fácil e pequena. Por outro lado, o código demorou-nos mais tempo a realizar devido, principalmente, à dificuldade na compreensão do que era pedido no enunciado. Assim que compreendemos a 100% o que nos era pedido, o código desenrolou-se bastante facil e rapidamente. Como nos outros exercicios já possuiamos toda a lógica necessária antes sequer de começar o código e a montagem. Esta lógica é breve e tornou-se mais fácil ainda quando compreendemos que o primeiro e o último dígitos seriam sempre “1”. Com esta informação só nos precisávamos de preocupar em encontrar os dois digitos do meio do número binário resultante. Depois de realizarmos toda a montagem com o código no Tinkercad passámos então para a montagem real no arduino e encontrámos algumas dificuldades. Uma destas dificuldades é que ao tirar e colocar o fio para realizar o clock, existia uma espécie de bounce e, com o auxilio do prof. Carlos, colocámos uma função que gera um clock chamada clockGenerator(byte, float). A par das 3 funções habituais, colocámos uma adicional chamada estadoSeguinte() que define o próximo estado. Na função returnValores() onde os valores obtidos são colocados nos LEDs, colocámos também os LEDs do primeiro e último digitos sempre aceso, já que estes digitos são sempre “1”.

Código:

int saida1 = A0; // digito 1 de saida

int saida2 = 2; // digito 2 de saida

int LED3 = 3; // LED do digito resultante no indice 3

int LED2 = 5; // LED do digito resultante no indice 2

int LED1 = 6; // LED do digito resultante no indice 1

int LED0 = 9; // LED do digito resultante no indice 0

unsigned long t1, t0; // longs para o tempo

volatile bool x;

bool a; // a=0 => subir ; a=1 => descer

bool Q0; // estado anterior no digito 1

bool Q1; // estado anterior no digito 2

bool D0; // estado seguinte no digito 2

bool D1; // estado seguinte no digito 1

bool S0; // resultado do digito 1

bool S1; // resultado do digito 2

void setup() {

pinMode(LED3, OUTPUT);

pinMode(LED2, OUTPUT);

pinMode(LED1, OUTPUT);

pinMode(LED\_BUILTIN, OUTPUT);

pinMode(LED0, OUTPUT);

pinMode(saida1, INPUT\_PULLUP);

pinMode(saida2, INPUT\_PULLUP);

attachInterrupt(0, CLK, RISING);

interrupts();

Serial.begin(9600);

}

void clockGenerator(byte pino, float f){ // função que gera um clock

static boolean state = LOW;

static unsigned long t1, t0;

t1 = millis();

if (t1 - t0 >= 500. / f){

digitalWrite(pino, state = !state);

t0 = t1;

}

}

void CLK () { // função responsável pelo clock

Q1 = flip\_flop\_D(D1);

Q0 = flip\_flop\_D(D0);

}

bool flip\_flop\_D(bool D) { // função responsável pelo flip\_flop

return D;

}

void loop() {

clockGenerator(LED\_BUILTIN, 2); // função que gera um clock para atenuar o bounce

checkInterruptor(); // função que lê o interruptor

estadoSeguinte(); // função que define o estado seguinte

logica(); // função responsável pela lógica

returnValores(); // função que retorna os valores nos LEDs

}

void checkInterruptor() {

a = digitalRead(saida1);

}

void estadoSeguinte() {

D1 = Q1 ^ Q0;

D0 = !Q0;

}

void logica() {

S1 = ((!Q1)&(!Q0))|(a&(!Q1))|((!a)&Q1&Q0);

S0 = !Q0;

}

void returnValores() {

digitalWrite(LED3, 1);

digitalWrite(LED2, S1);

digitalWrite(LED1, S0);

digitalWrite(LED0, 1);

}

Aqui está um exemplo da montagem no Tinkercad:

Circuito 3:

Uma imagem com texto, eletrónica, circuito

Descrição gerada automaticamente

Output do contador no ficheiro zip.