UFPB – DEPARTAMENTO DE INFORMÁTICA
Prof. José Antônio Gomes de Lima
Alunos: **Fábio Alexandre E. Melo -** 11508361
Thuane Mikaella - 11325835
Luigge Lena - 11512972

RELATÓRIO DE CIRCUITOS LÓGICOS

(período: 2016.2)

Índice

Introdução

Esse projeto da disciplina de Circuitos lógicos é composto da criação de quatro programas em código simplificado baseado em Assembly, com o auxílio do programa *Montador Java,* que os converte para a linguagem de descrição verilog. tais códigos são executados em uma simulação de CPU simples, para a resolução das problemáticas propostas na especificação do projeto, e de suas respectivas execuções e simulações de forma de onda utilizando a IDE Altera Quartus II..

1.Características do Circuito

Unidade Central de Processamento (UCP)

UCP					
Chip	**Device**	**Input Pins**	**Output Pins**	**LCs**	**Frequência máxima de operação**
Cyclone	EP1C4F324C6	2	146	436	10 MHZ
Comentários :					
Descrição: Uma Unidade Central de Processamento (CPU) constituída por oito registradores (REM, RDM, R1,R2,R3,R4,R5 e R6) e uma unidade lógica aritimética (ULA) de 8 bits, com cinco multiplexadores e treze possíveis operações e relacionada à uma memória SRAM síncrona de 16x8.					

2. Programa que calcula a sequência de Fibonacci

Listagem do Programa fonte

```
/* Fibonacci */

MOV REM,0;
MOV RDM,0; /* movendo 0 para o primeiro registrador */
WRITE MEM;

READ R1; /* movendo 0 para R1 e R2 - Primeiro Digito do Fibonacci */
READ R2;

MOV RDM,1;    /* movendo 1 para R1 e R3  - Segundo Digito do Fibonacci */
WRITE MEM;
READ R1;
READ R3;

LOOP:
ADD R2,R3; /* somando o digito anterior com o proximo, enviando para R1 */
MOV R1,R2;

ADD R3,R2; /* somando o digito próximo com o anterior, enviando para R1 */
MOV R1,R3;

JUMP LOOP;
```

3. Programa que calcula o fatorial do número 5

Listagem do Programa fonte

```
/* Fatorial */

/*    MOVE 5 PARA R2 E R3
      MOVE 1 PARA R4 E R6 */

MOV REM,1;
MOV RDM,5;
WRITE MEM;
READ R2;
READ R3;

MOV RDM,1;
WRITE MEM;
READ R4;
READ R6;


LOOP:
      SUBA R3,R4; /* SUBTRAI 1 DE R3, PARA DECRESCER */
      MUL R2,R3; /* MULTIPLICA O VALOR DE R3 COM O ACUMULADO EM R2 */
      COMP R3,R6; /* VERIFICA SE R3 EH 1 */
      JUMP PRINT,IG; /* SE FOR 1, TERMINA O LOOP E PRINTA */
      JUMP LOOP;

PRINT:
      MOV R1,R2; /* PRINTA O RESULTADO EM R1 */
```

4. Programa que calcula a equação: X/17 + 3 = 0

```
/* X/17+3 = 0 */


/* COMO SÓ PODEMOS MOVER NUMEROS DE 0 A 15,
ARMAZENAMOS O 17 NO REGISTRADOR SOMANDO 14+3 */

MOV REM,1;
MOV RDM,14;
WRITE MEM;
READ R2;

MOV RDM,3;
WRITE MEM;
READ R4;


/*    SIMPLIFICANDO X/17+3=0 CHEGAMOS EM X=(-3)*17 */

ADD R2,R3; /* TRANSFORMA O 14 NO 17 */

SUBB R3,R4; /* 3-3 = 0 */
SUBB R3,R4; /* 0-3 = -3 */

/* AGORA QUE TEMOS OS VALORES '17' NO R2 E '-3' NO R3,
VAMOS CALCULAR  17*(-3) */

MUL R2,R3;
MOV R1,R2; /* ARMAZENA A RESPOSTA NO R1 */
```

5. Programa que calcula os números primos à partir de 2

```
/* Contagem de Números Primos */

/*    Inicialização */

      MOV REM,0;
      MOV RDM,0; /* valor zero para comparação */
      WRITE MEM;

      MOV REM,1;
      MOV RDM,1;
      WRITE MEM;

      READ R5; /* Variavel de Contagem (1) */
      READ R4;

      MOV REM,2;
      MOV RDM,2;
      WRITE MEM;

      READ R1; /* Envia o primeiro Primo (2) */
      READ R2; /* Variável de Testes Crescente (2) */
      READ R6; /* Variavel de Subtração (2) */

NEXT:                    /* Início do Loop */

      ADD R2,R5; /* Adiciona 1 no R2 */

      MOV REM,3; /* Salva o valor do numero de testes */
      MOV RDM,R2; /* na posição 3 da RAM */
      WRITE MEM;
      MOV R3,R2;

PARITY:           /* Etapa de Verificação de Paridade */

      SUBA R3,R6; /* Subtrai 2 do R3 */
      COMP R3,R4;
      JUMP CONT,ig; /* continua, se igual a 1 */
      COMP R3,R6;
      JUMP NEXT,ig; /* proximo numero, se for igual a 2 */
      JUMP PARITY;
```

```
CONT:
      READ R3; /* Restaura o valor de R3 */

/* Se chegou aqui, nao é par: verifica se é primo */

SKIP:
      /* restaura R4 para (1) */
      MOV REM,1;
      READ R4;


TEST:
      SUBA R3,R4;
      COMP R3,R6; /* se o valor decrescido for 2, é primo */
      JUMP PRINT,ig;

      MOV REM,4;   /* salva o valor de N-1 na RAM */
      MOV RDM,R3;
      WRITE MEM;

/* Modulo */

      MOV R4,R3; /* copia o R3 temporariamente para R4 */

      DIVB R3,R2;    /* (a/n) */
      MUL  R3,R4;    /* n * (a/n) */
      SUBA R3,R2;    /* n * (a/n) - n */

      /* temos o modulo salvo em R3 */

      /* movendo '0' para R4, para que possamos testar o modulo */
      MOV REM,0;
      MOV RDM,0;
      READ R4;

      COMP R3,R4;

      JUMP NEXT,ig; /* se modulo é zero, não é primo */

      /* Restaura o valor de R3 e testa novamente */
      MOV REM,4;
      READ R3;

      JUMP CONT;

PRINT:   /* Imprime o Resultado em R1 */
      MOV R1,R2;
      JUMP NEXT;
```

Conclusão

Com esse projeto, podemos aprender que, além das instruções de funcionamento básico de uma CPU, mesmo sem a disponibilidade de funções avançadas de linguagens de alto nível e com uma quantidade pequena de instruções e registradores, é possível resolver e representar operações lógicas e matemáticas complexas.