# Documentação EP3 - Álgebra Booleana

Gabriel de Russo e Carmo 9298041
Germano Hüning Neuenfeld 9298340
Luis Gustavo Bitencourt de Almeida 9298207
Victor Wichmann Raposo 9298020

Junho de 2016

## 1 Introdução

Este documento visa explicar como funciona nosso ciclo de instrução, terceira parte do projeto de MAC0329 - Álgebra Booleana e Circuitos Digitais, ministrada pelo professor Junior Barrera.
A modularização desta parte do projeto, quando comparada com a primeira, foi bem menor pois os túneis do Logisim não funcionam entre dois circuitos diferentes.
Todos os valores numéricos estão representados em hexadecimal (instruções, endereços de memória, etc). As instruções são identificadas conforme a tabela abaixo

| Código | Instrução | Descrição |
|---|---|---|
| 0x0B | {LDA} AB | Carrega dado do endereço AB no acumulador |
| 0x0C | {STA} AB | Carrega acumulador no endereço AB |
| 0x32 | {NOP} | Sem operação |
| 0x15 | {ADD} AB | Soma o valor de ACC com o conteúdo do endereço AB |
| 0x16 | {SUB} AB | Subtrai o conteúdo do endereço AB do valor de ACC |
| 0x17 | {MUL} AB | Produto do valor de ACC com o conteúdo do endereço AB |
| 0x18 | {DIV} AB | Quociente da divisão do valor de ACC pelo conteúdo do endereço AB |
| 0x19 | {REM} AB | Resto da divisão do valor de ACC pelo conteúdo do endereço AB |
| 0x33 | {JMP} AB | Salto não condicional para o endereço AB |
| 0x46 | {STP} | Fim da execução |
| 0x34 | {JLE} AB | Salto condicional para AB se o valor de ACC é menor ou igual a zero |
| 0x35 | {JDZ} AB | Salto condicional para AB se o valor de ACC é diferente de zero |
| 0x36 | {JGT} AB | Salto condicional para AB se o valor de ACC é maior que zero |
| 0x37 | {JEQ} AB | Salto condicional para AB se o valor de ACC é igual a zero |
| 0x38 | {JLT} AB | Salto condicional para AB se o valor de ACC é menor que zero |
| 0x49 | {JGE} AB | Salto condicional para AB se o valor de ACC é maior ou igual a zero |
| 0x1F | {INN} AB | Lê da entrada e armazena na posição AB |
| 0x29 | {PRN} AB | Escreve na saída o conteúdo na posição AB |

**Observações:** Instruções aritméticas armazenam o resultado descrito no acumulador. Instruções inválidas causam comportamento inesperado.

# 2 Módulos

## 2.1 Main

Módulo principal que implementa praticamente tudo. Vários túneis são usados visando melhor clareza do circuito. Descreveremos em detalhes cada parte.

### 2.1.1 Controlador

Responsável pela decodificação da instrução. Faz isso com auxílio de módulos auxiliares. Uma vez entendida, seus túneis são responsáveis pelo comportamento de todo o resto do circuito.

### 2.1.2 PC

O PC é um counter e sua função é guardar qual posição de memória será processada. É incrementado sempre em uma unidade, com exceção da instrução JMP, que muda o valor de PC para um endereço arbitrário.

### 2.1.3 Acumulador

O acumulador é um registrador de 16 bits que carrega o endereço proveniente da memória quando a instrução LDA for requisitada ou o valor de saída da ALU quando alguma operação aritmética for requisitada.

### 2.1.4 Memória

Para implementar a memória, utilizamos o bloco RAM do Logisim com 16 bits de dados e endereçamento de 8 bits. Por padrão, o endereço acessado é o que vem de PC, mas no caso de uma instrução que requer o acesso à memória - denominada de Ram Access no circuito - que são as de I/O (LDA, STA, INN e PRN) e as de operações aritméticas (ADD, SUB, MUL, DIV, REM), o endereço varia conforme especificado. Sempre tem como saída o dado do endereço atual.

### 2.1.5 IR

O IR é um registrador de 16 bits que recebe o dado da memória ou zero. É importante que IR tenha valor zero durante uma instrução para o bom funcionamento da lógica sequencial. Para garantir que isto aconteça, usamos um multiplexer. A saída é dividida em duas partes: a instrução e o possível endereço correspondente.

### 2.1.6 Clock

Clock geral do circuito. O clock funciona desde o início da simulação até que a instrução STP seja requisitada ou um overflow aritmético aconteça. Nestes casos, com o auxílio de um flip-flop JK, o clock para.

### 2.1.7 Entrada

Apenas as intruções STA e INN, tem a permissão de escrever em uma posição de memória. Quando uma delas ocorre um valor - que está no acumulador no caso da STA ou que o foi inserido pelo usuário para a INN - será armazenado em um endereço específico da memória RAM do nosso circuito.

#### 2.1.8 Saída

A saída é um registrador que inicialmente começa com o valor zero. Quando uma instrução PRN é requisitada, o registrador recebe um valor da memória RAM (do endereço que acompanha a instrução).

#### 2.1.9 ALU

Responsável pelas operações aritméticas. Recebe o valor do acumulador e um valor da memória RAM e devolve o resultado da expressão aritmética correspondente (Acc OP RAM out). Também indica overflow quando este acontecer. Para seleção da operação, conta com um módulo auxiliar que converte a instrução da tabela na primeira página para uma instrução de 3 bits da ALU, como especificado a seguir.

#### 2.1.10 Jump condicional

Reponsável por decidir se um desvio condicional deve ou não acontecer. Em caso negativo, apenas chama a instrução NOP. Em caso positivo, com o auxílio de alguns módulos auxiliares (que serão descritos a seguir), chama a instrução JMP com mesmo endereço.

### 2.2 isABC

Módulos auxiliares que comparam se o valor de entrada é igual ao valor da instrução (veja tabela da primeira página). Para tal, verificam se a entrada XOR instrução é zero.

### 2.3 is[OP|JMPC]

Módulos auxiliares que comparam se o valor de entrada é igual ao valor de alguma instrução aritmética ou de desvio condicional. Para tal, utilizam vários módulos do tipo isABC, descritos anteriormente.

### 2.4 isZERO

Verifica se a entrada de 8 bits é zero.

### 2.5 ALU

Unidade lógica aritmética. Recebe dois números de 16 bits e um seletor de 3 bits (soma, subtração, multiplicação, divisão ou resto), devolvendo o resultado da operação e um indicador de overflow.

### 2.6 InstToAlu

Converte a instrução aritmética da tabela da primeira página para o seletor da ALU.

### 2.7 MSK

Módulo auxiliar para o desvio condicional. Recebe na entrada um número de 16 bits (proveniente do acumulador) e devolve uma máscara de 6 bits, onde cada bit é uma flag que diz se a condição foi satisfeita. Os bits menos e mais significativos indicam sucesso nos JLE e JGE, respectivamente. A ordem das flags segue a tabela da primeira página.

## 2.8 GO?

Módulo auxiliar para o desvio condicional. Recebe uma máscara de 6 bits contendo flags para as condições satisfeitas e a instrução do desvio condicional. Por meio de um left shift e um AND, devolve 1 se o desvio deve ser efetuado e 0 caso contrário.

## 2.9 OprFix

Implementam overflow na aritmética do Logisim.

## 2.10 ABS

Devolve o absoluto de um número de 16 bits (complemento de dois).