AsmROM User Manual

by Hexa Decimal

Contents

1 Introduction

Welcome to AsmROM!

AsmROM is a read only memory component for LogicWorld, with a built-in assembly language based assembler. In appearance, it is a Label component, with input pins on its left side, output pins on its right side, and a button at the bottom. It has a 16 bit address for a total of 65536 bytes of addressable memory, and an 8 bit output. The text within the label portion is accessible via the ‘x’ menu, and allows for an assembly language based program to be assembled into machine code for various different architectures.

The purpose of this manual is to describe the AsmROM’s functionality and how to use it. Since the logic interface is straight-forward, input and address and output the corresponding byte, the focus will be the text portion, where the assember operates.

2 Sections

The text is split into three sections that may be placed in any order: A configuration section denoted by ".config", An Instruction section denoted by ".instr", and a program section denoted by ".prog". The configuration and instruction sections describe how to convert the program section into machine code.

The lines within each section are further broken down into tokens separated by commas and spaces, which are interchangeable. For example "Add Ax,Bx" and "Add,Ax Bx" will function the same way. This simplifies the assember's process and allows for some variation in implementation.

3 Configuration

The configuration section includes parameters for the assembler. Parameters may be in any order.

“BigData”, case sensitive, specifies big endian mode for data. All values greater than 8 bytes, besides the instructions will be in big endian. Instructions will not be affected and default to little endian, but immediate values following instructions will be in big endian. Further explanation found in Chapter 4.

“BigInstruction”, case sensitive, specifies big endian mode for instructions. Data will not be affected and default to little endian. Further explanation found in Chapter 4.

“AddressLength”, case sensitive, specifies the number of bits in an address referred to by labels.

All lines that are not predefined parameters will be treated as enumerations. An enumeration has a name, which will be used for instruction signatures, and values, which will be used in the program. Both of which are case sensitive, must start with a letter or underscore, and contain only letters, digits, and underscores. An optional number just after the name specifies how many bits will be used in the machine code. If not present, the minimum number of bits needed to include all values will be used.

For example: “REG,2 Ax,Bx” will specify an enumeration for registers Ax and Bx. “Reg” will be used in the instruction signature, and the machine code will be two bits long. 00 for Ax, and 01 for Bx. “SREG ES,DS,SS” would also have a two bit long machine code because the largest number, two, requires two bits.

4 Instructions

The instructions section contains instruction descriptions. Each instruction has two parts: the signature, and the machine code.

Instruction signatures have a name and operand types. The name must start with a letter or underscore, and contain only letters, digits, and underscores. Operand types are either an enumeration name, or im8, im16, im32, or im64, Case Sensitive. In the machine code section, operands are referred to by a lowercase letter with ‘a’ being the first operand.

Following the signature line there may be any number of machine code lines. Each line starts with a colon, ‘:’, and contains groups of bits separated by spaces or commas. A bit is either ‘0’, ‘1’, or a lower case letter representing an operand. Multiple of the same letter may be placed in a row to represent multiple bits occupied by the operand. Though, only one is needed. Each group of bits will be padded with leading 0’s until a multiple of 8. The instruction endianness applies to the first group of bits in each line, while data endianness applies to all groups following it.

5 Program

The program section contains the actual program that will be written to the ROM. Each line can be either a label or an instruction. A label must start with a letter or underscore, contain only letters, digits, and underscores, and end with a colon. An instruction may use an enumerated value, label, or hexadecimal value as an operand. Hexadecimal values must start with 0x, and are padded with leading zeros to a multiple of 8 bits. Labels are replaced with the address it refers to. The length of all addresses is specified by the AddressLength configuration value. Enumerations may be used for naming registers and function as described in Chapter 3, Configuration.