PDP-8 ISA Design

ECE586

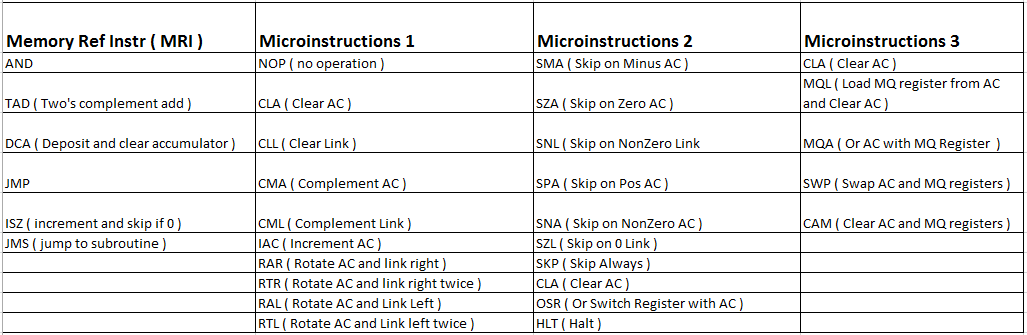
**Table of Contents…**

**Introduction:**

The PDP-8 Architecture is a 12-bit minicomputer based on simple instruction set and powerful operations in one instruction cycle. It was designed by Digital Equipment Corporation as a “Straight-8” and was later designed with CMOS microprocessors and known as the “CMOS-8s.” PDP-8’s significance is with its simplicity, low cost, and expandability. The PDP-8 registers are Accumulator (AC), Link (L, carry from AC ), Program Counter (PC), Memory address register (MAR), and Memory buffer register (MBR). It has 4K main memory with 32 pages of 128 12-bit words.

The instruction set is made of 12-bit instruction words. The 3 high order bits encode the operation. Bit 3 determines whether the address is direct or indirect. Bit 4 indicates if the page is current of page 0. Bits 5 to 11 gives the 7-bit offset address. The assembler places constant values for arithmetic in the current page. For cross-page jumps and subroutine calls, indirect addressing in the current page is used.

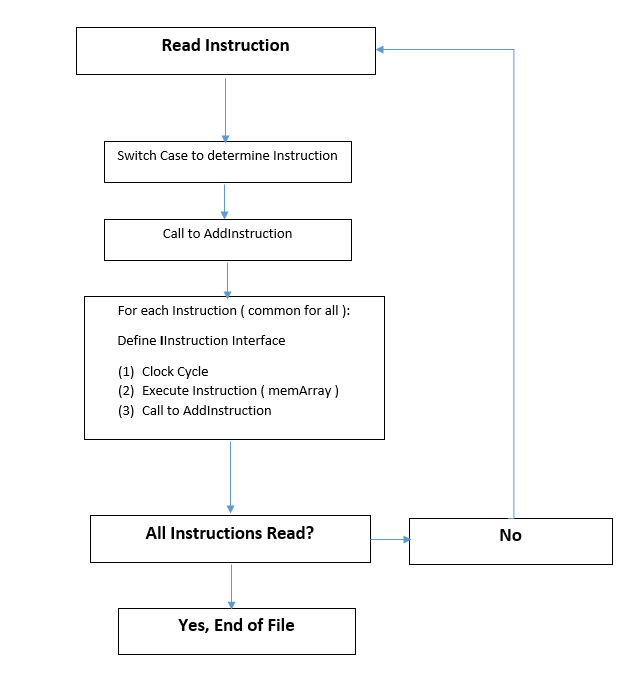
The instructions are listed below:



The objective of this project is to create a PDP-8 Instruction Set Architecture simulator to execute instructions as defined for PDP-8.

**Methods:**

Our first step was to design the framework. A layout flowchart of the framework is shown below, Fig.



The program is designed to read in instructions, determine instructions, add them into an array of instructions, and execute. This is done until the end of file is reached. For each instruction, clock cycles will be added accordingly.

**More details on exact structure ( what each class does )**

The first step was to define algorithms for each instruction. An example Algorithm for the Add instruction is shown below:

Create Algorithms for instructions….

Designed Architecture ( Source codes, outline/overview of code )….

**Testing:**

Tests for each instruction is defined in the Emulator Test Project. Multiple tests are created to determine if the expected results are produced for each instruction. An example test program is shown below:

**Conclusion:**