COMP262-01
CPU SIM Project: a PEP8 Virtual Machine SIMULATION

PROJECT BACKGROUND

Virtual machines in computer science are used both for teaching concepts and for implementing algorithms
The most widespread machine for implementing algorithms in industry is probably the Java virtual machine.
Pep/8 is a virtual machine for teaching computer systems concepts developed by Warford(et al) and based on the seven levels of abstraction popularized by Tanenbaum: application, high-order language, assembly, operating system, instruction set architecture (ISA), microcode, and logic gate.
For a number of years we, at CSUCI, have used an assembler/simulator for the Pep/8 in the Computer Systems courses I & II (COMP162/262), to provide students with a hands-on experience at the high-order language, assembly, and ISA levels.

The PEP/8 Virtual Machine is a classical 16-bit von Neumann computer with an accumulator (A), an index register (X), a program counter (PC), a stack pointer (SP), and an instruction register (IR).
It has eight addressing modes: immediate, direct, indirect, stack-relative, stack-relative deferred, indexed, stack-indexed, and stack-indexed deferred.
The instruction set is based on an expanding opcode yielding a total of 39 instructions, which come in two flavors – unary and nonunary.
The unary instructions consist of a single 8-bit instruction specifier(InsSp), while the nonunary instructions have the instruction specifier followed by a 16-bit operand specifier (OpSp).
For the nonunary instructions, the addressing modes determine the operand location from the operand specifier as follows:

Addressing mode	Operand
Immediate	OpSp
Direct Mem	[OpSp]
Indirect Mem	[Mem [OpSp]]
Stack-relative	Mem [SP + OpSp]
Stack-relative deferred Mem	[Mem [SP + OpSp]]
Indexed	Mem [OpSp + X]
Stack-indexed	Mem [SP + OpSp + X]
Stack-indexed deferred Mem	[Mem [SP + OpSp] + X]

Most virtual machines used for teaching, have software support that provide students with hands-on experience in programming at the assembly level, and Pep/8 is no exception.
In the Computer Systems I course (COMP162), students are given complete ‘C’ programs and required to translate them as a compiler would.
They have access to an assembler that translates their programs to the ISA level, which they can then test/run on a Virtual Machine simulator. The Computer Systems I course is a prerequisite for the Computer Systems II (COMP262) course. So, students in Computer Systems II already have a working knowledge of the high-order language, assembly language, and ISA levels of a von Neumann machine, based on an implementation of the Pep/8 computer.

PROJECT DESCRIPTION

The purpose of this project is to provide the Computer Systems II (COMP262) students with the opportunity to analyze, design and apply the concepts covered in class and the textbook material by designing and implementing an HLL program (an App) that Simulates the Execution Phase (running of a program) on the PEP8 Virtual Machine.
These applied analysis and design tasks will parallel the study of the general computer architecture and organization concepts covered in the course and serves to reinforce the application of the material studied.
A purely theoretical analysis of computer systems, at the structural and Instruction Set Architecture (ISA) level may be adequate for understanding the overall execution of a machine language program, but it is no match for the in-depth understanding obtained from Reverse Engineering, Designing and Implementing a software simulation of such process.

PROJECT GUIDELINES

Students are expected to complete the different parts of the project, which must be submitted by the due dates. For all phases, constraints and guidelines will be provided (in each of the more detailed documents) and a submission mechanism made available.
The final phase consists of the completed Simulation program, which produces the SAME results, when 'Running' a PEP8 machine language program, as the Warford Virtual PEP8 app does.

PROJECT DELIVERABLES

The student should learn as much from working on the project as they would from class sessions and/or the homework and the textbook presentation.
IF the student actually works on the project, applying a steady gradual effort, as opposed to 'cramming' it hours before the due date, the project should be both easy and enjoyable.
The project has cumulative "Phases/Parts/Stages/Sections" so that the student **DOES NOT** fall behind/fail as it is likely to happen IF the whole project was due all at one time.