University of Victoria
Department of Computer Science
CSC 355 Digital Logic and Computer Design
Lab 7: Very Small Computer System (VSCS)

The pre-lab work is to be completed and submitted **before** the deadline, as follows:
Submit source code before 4:00 p.m. on **Tuesday, November 24**, using the *Assignments* link, where you will find a link entitled PreLab7.

Introduction

The goal of this lab is to understand the design of a Simple Computer System and construct the circuit using Design Works. This Lab is entirely completed using Design Works.

A **possible architectural** block diagram of a simple computer is shown in Figure 1**.**

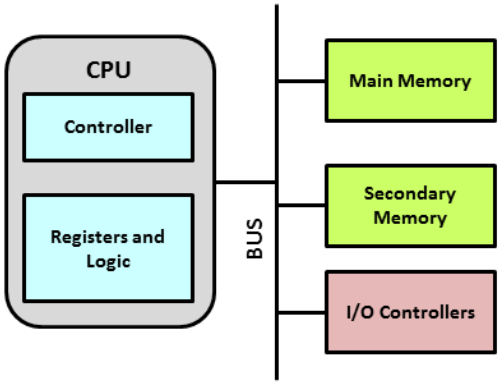


Figure 1

This lab focuses on the design of the Central Processing Unit or CPU for a VCVS. The CPU contains:

- all the internal registers;
- the arithmetic and logic circuits;
- the controller, which produces timing signals to coordinate the activities of the machine.

The workhorse of the *Registers and Logic* section of the CPU is the Arithmetic and Logic Unit (ALU). ALU circuits in larger computer systems contain additional circuitry for operations such as shifting and rotating sets of bits, multiplication and division and floating-point arithmetic. Our ALU here is rather more restricted in its selection of operations. The ALU gets all of its operands from registers. The ALU circuit to use is provided on ConneX.

The CPU controls the execution of a sequence of instructions stored in a sequence in the main memory. A special register, the program counter (PC), stores the address of the beginning of the next instruction to be fetched and executed. The PC is updated as each instruction is executed. In this lab, we do not consider the storage of instructions in memory and the program counter. (But that is an excellent extension of this design!) We simulate that unit with a set of 3 input lines called **`INSTRUCTION`** that contain the 3-bit instruction as it would come from the memory.

Specifications of the VCVS

The computer you are designing is an 8-bit accumulator architecture that takes a 3-bit instruction, decodes it and executes it. VCVS has two 8 bit registers (**`A`** and **`B`**) (You can use the built in registers in Design Works for

these), an ALU and a 3-bit instruction register as shown below in Figure 2. Note that not all control signals are shown in the diagram. The output of the ALU is always written back to the Register B (Figure 2).

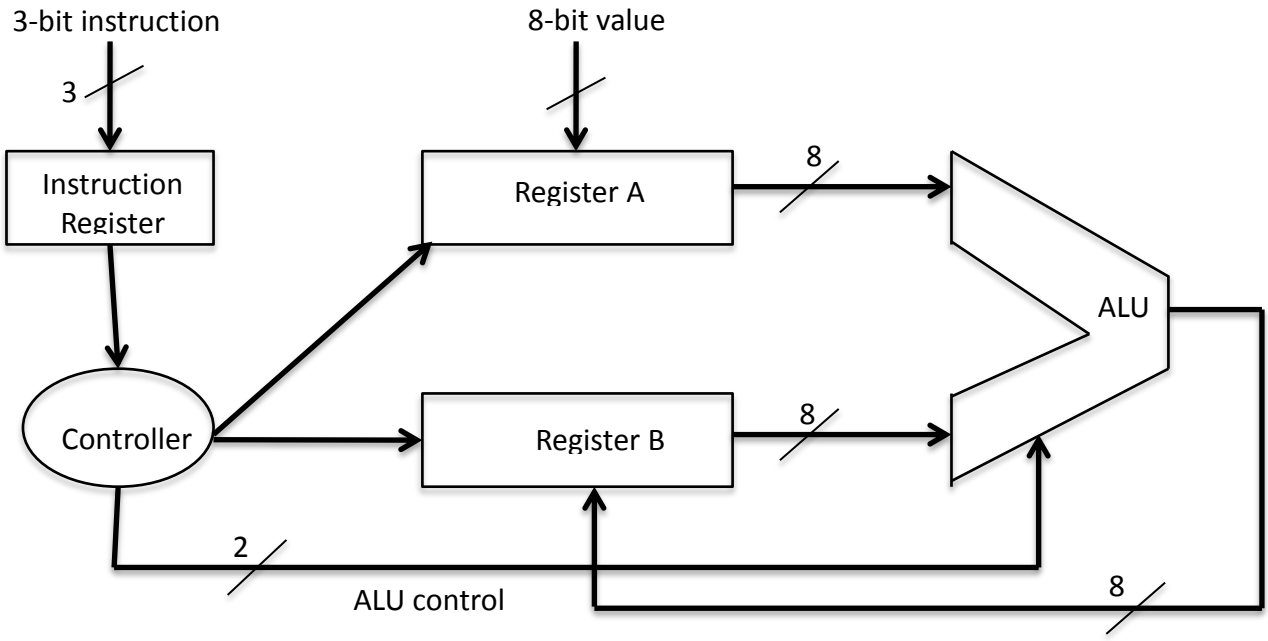


Figure 2: Block Diagram of VCVS

The VCVS operates on an external clock signal. For simplicity, assume that a switch is used to emulate the clock pulse. VCVS takes three clock cycles (pulses) for every operation.

The controller of VCVS operates as follows:

- In the first clock cycle, a 3-bit instruction is loaded (coming from binary switches) into the instruction register. A load control signal (**X**) is used for this purpose.
- In the second clock cycle, the instruction is decoded and signals to perform the appropriate logic or arithmetic operation are asserted. The signals that are asserted in this clock cycle are **Y** (to load Register **A** - use binary switches to load data) and the two bit ALU control signal, unless the instruction is **CLR**, in which case both register **A** and register **B** are cleared.
- In the third clock cycle the output of ALU is loaded back into register **B**. A load control signal **Z** is asserted to load these values into Register **B**. If the instruction is **CLR,** no action is taken during the signal **Z** and the ALU is disabled.
- The overall cycle repeats after this.

The format of the 3-bit instructions is specified as follows:

Bits	Mnemonic	Meaning
000	ADD	B ← B + A
001	SUBTR	B ← B - A
010	AND	B ← B *AND* A
011	OR	B ← B *OR* A
100	CLR	A ← 0; B ← 0

101	**NOP**	No operation
110	**NOP**	No operation
111	**NOP**	No operation

After decoding an instruction, there are three possible general categories of action:

1. If the decoded instruction is an arithmetic or logical operation (one of the first four), as in **ADD**, **SUBTR**, **AND** or **OR,** then the values that are in the two Registers (**A** and **B**) are be sent to the ALU for processing. The results from the ALU are always written back to the register **B**.
2. If the instruction is **CLR**, then registers **A** and **B** are cleared.
3. There are three opcodes, namely **101**, **110** and **111,** that are not used. These are simply specified as no operation (**NOP**) in the table. Note that when any of these instructions is present, you **MUST** disable the clock to the controller so that no operation is performed. This provides future instruction expansion room for the system.

Pre-Lab Exercise

Using the built in registers of Design Works and the ALU circuit (alu.cct), draw a complete block diagram of VSCS (based on Fig. 2) with the control signals. Note that in each clock cycle you need to generate one load signal. One approach could be to use three D flip-flops (**X,Y,Z**), driven by one clock, where:

- **X** is used as a load control for the instruction register;
- **Y** is used as load control for Register **A;**
- **Z** is used as load control for Register **B**.

The process repeats after the third clock pulse and these signals **X, Y** and **Z** will be asserted as shown in Figure 3. Submit a copy of your design.

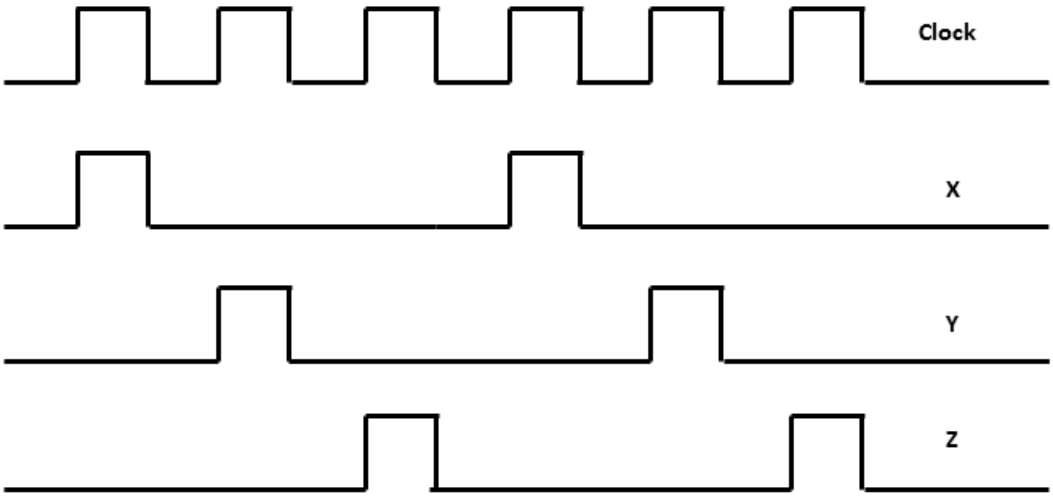
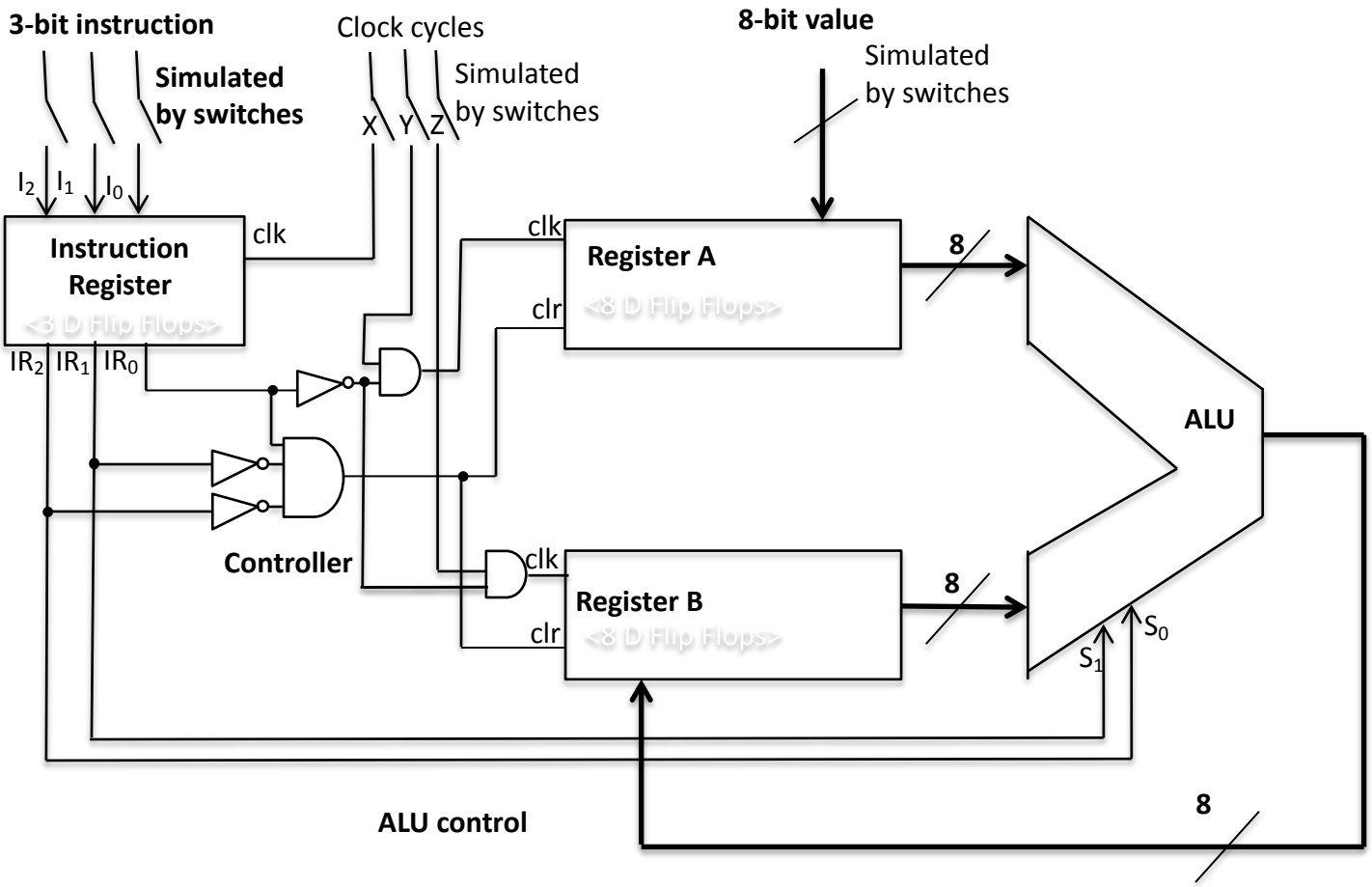


Figure 3



The 3 instruction cycle switches could be replaced by 3 D Flip Flops, with state variables X, Y and Z. D_X=X'Y', D_Y=X, D_Z=Y, all have a common clock.

In-Lab Exercises

Using DesignWorks, put together the whole VCVS according to the block diagram and including the controller logic you developed.

Test and show the operation for various instructions and register values to your lab instructor.