CS220 Computer Architecture
Digital Logic Design
Practical 10

The TkGate Logic Simulator under Linux is used to implement these practicals. Boot Linux, log in and launch the TKGate application.

In this practical, you must implement a simple 8-bit CPU circuit consisting of 4 general purpose registers, 2 data sources (simulated by 8-bit DIP switches) and an adder unit, all connected by an 8-bit bus. Tri state buffers are used so that only one source at a time is selected to use the bus.

The contents of the registers should be visible using hexadecimal displays. The registers' contents can be set from either of the two data sources via the bus. The adder must be able to add the contents of any 2 registers and place the output in a third.

Instruction coding:-
An instruction register is simulated by a third, 8-bit, dip switch to be coded as follows:
Bits 0,1 => represent the code for which source of data should be selected for register loading

- 00 means place the output from the Adder on the bus
- 01 means place Data Source A on the bus
- 10 means place Data Source B on the bus
- 11 not used

Bits 2,3 => Indicates which register will be used as source 8-bit operand A for the adder during add instructions

Bits 4,5 => Indicates which register will be used as source 8-bit operand B for the adder during add instructions. Operands to the adder always come from one of the four registers.

MSB Bits 6,7 => Indicates which destination register is to receive an 8-bit value placed on the bus whose source is indicated by bits 0,1 above. Note you will have to provide a clock signal to enable the register to sample the bus when it is selected by the instruction.

How to operate the circuit:-
So, to load register 0 from Data Source A we would use the instruction code **00**0000**01** (01H). We might define the mnemonic LOAD R0,A to represent this.

To load register 1 from Data Source B we would use the instruction code **01**0000**10** (42H). The mnemonic LOAD R1,B could represent this.

To add register 0 to register 1 and place the result in register 2 we would use the code **10010000** (90H). The mnemonic ADD R0,R1,R2 could represent this.

When your circuit is built, set data values for source A and source B via the DIP switches and set the instruction DIP switch consecutively to the codes above. This should result in register 3 receiving the contents of Data Source A + Data Source B as shown by its hexadecimal display.

Note: Every CPU circuit has its own way of encoding its particular instruction set and a set of mnemonics is defined to represent these instructions so that the assembly language programmer does not have to work with binary codes.

An example design is given here to guide you. You are required to implement the circuit and and Using phone/camera to reecord an 1 to 3 minutes video(720p or 1080p) to **demonstrate its operation. Also, make up three mnemonics of your own (either add or load instructions using your choice of sources and registers), show the opcode for them in binary and hexadecimal and demonstrate that they work.**