Computer Organization and Architecture Lab

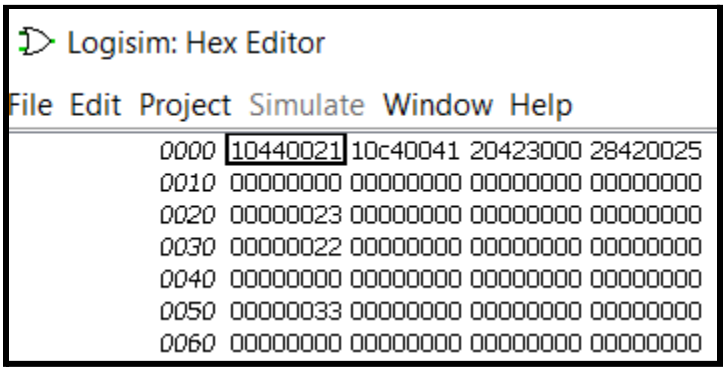
Project: Design a 32 - bit RISC Processor

Name: Harsh Singh Jadon

Roll Number: 19CS01061

How to use the 32 - bit RISC processor?

1. Open **Instruction Encoding document,** in order to convert each instruction into corresponding hexadecimal value.
2. Load instructions into memory - As told in the encoding format document, instruction inside memory is represented in hexadecimal format. Program counter starts at address 0000 of the memory. Therefore, load instruction from 0000 into memory.
3. Right-click on RAM and select **"Edit Contents"** for inserting into memory. (Enter Hexadecimal values). It should look like this -

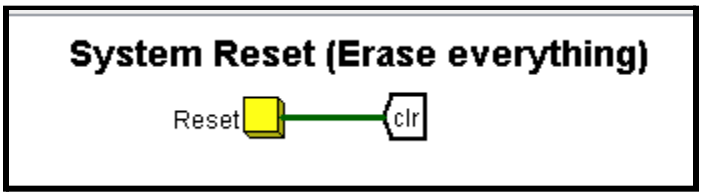


4. Press the System clock to start executing the instructions. You will have to press manually, every time when you want to change the current state of executing the instruction. The changes are always reflected at the rising edge of the clock cycle.
5. Press 10 times the clock, in order to completely execute the instruction inside the five-stage pipeline.
6. At every stage, using the **"Show simulation hierarchy"** option, you can check the state of execution which means what is the state of the Register File, which control signals are generated, which operation is being executed by the ALU and so on.
7. After 10 times pressing the system clock, your Program Counter will get incremented.

Kudos !! You have successfully executed your first instruction present at the 0000 address location. Want to execute more instructions?

If yes, then repeat the same process of pressing the clock 10 times and see the changes.

8. After you have successfully executed all the instructions, if you want to run a different program, then you will have to use the **Memory clear and Reset button** before loading a new program.

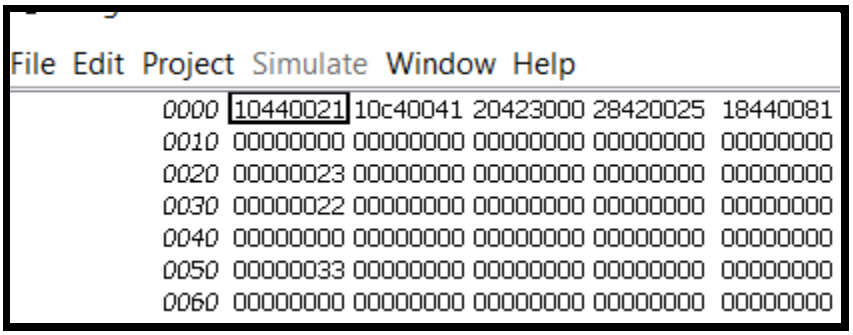


Want to try a dummy program and check how the CPU works? Load this sample program given in the lab manual inside the memory and execute it.

Program (A = B + C - Immediate): -
00000000: Load R1, X(R2) ; Loads B
00000004: Load R3, Y(R2) ; Loads C
00000008: Add R1, R1, R3 ; Adds B+C
0000000C: Sui R1, R1, #Immediate ; Subtracts Immediate from (B+C)
00000010: Store R1, Z(R2) ; Stores result in A
00000014: HLT ; Halts execution

Note - a) Assume, R2 contains the base address of the data and R2 is loaded with 0x10(start).
b) B takes a value of 0x23 at 0x20 address.
c) C takes a value of 0x22 at 0x30 address.
d) Immediate value takes the value of 0x12.
e) Final answer is stored at memory location 0x50, which is equal to 0x33 (0x23 + 0x22 - 0x12).

This is how you should load into memory.

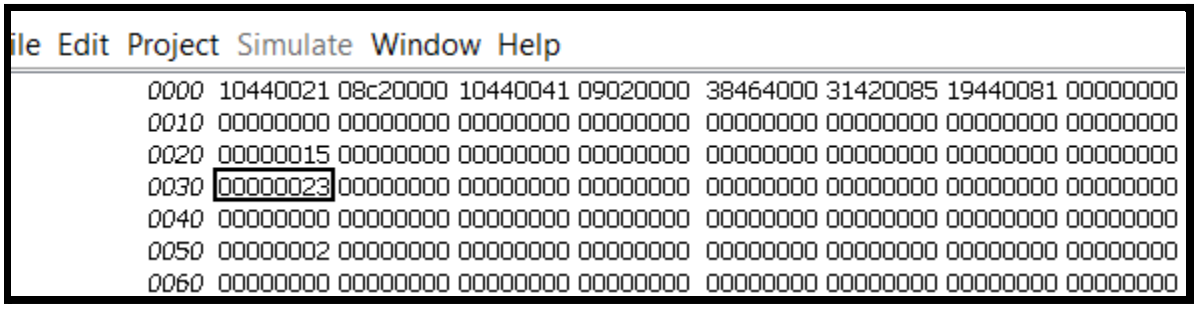


Want to try one more dummy program and check how the CPU works? Load this sample program given in the lab manual inside the memory and execute it.

Program (A = (B OR C) AND Immediate): -
00000000: Load R1, X(R2); Loads B
00000004: Move R3, R1; Moves R1 to R3
00000008: Loads R1, Y(R2); Loads C
0000000C: Move R4, R1, Moves R1 to R4
00000010: OR R1, R3, R4; Performs OR of R3 and R4
00000014: ANI R5, R1, #Immediate; Performs AND with Immediate
00000018: Store R5, Z(R2); Stores the result in A
0000001c: HLT ; Halts execution

Note - a) Assume, R2 contains the base address of the data and R2 is loaded with 0x10(start).
b) B takes a value of 0x15 at 0x20 address.
c) C takes a value of 0x23 at 0x30 address.
d) Immediate value takes the value of 0x42.
e) Final answer is stored at memory location 0x50, which is equal to 0x02 = (0x15 OR 0x23) AND 0x42.

This is how you should load into memory.



Conclusion:

a) I have made use of tunnels and probes, instead of heavy wiring and have names subsections properly, in order to clearly understand the circuit.
b) RAM size: 256 KB
c) This is a 32-bit RISC processor with an address bit width of 16 and a data bit width of 32.