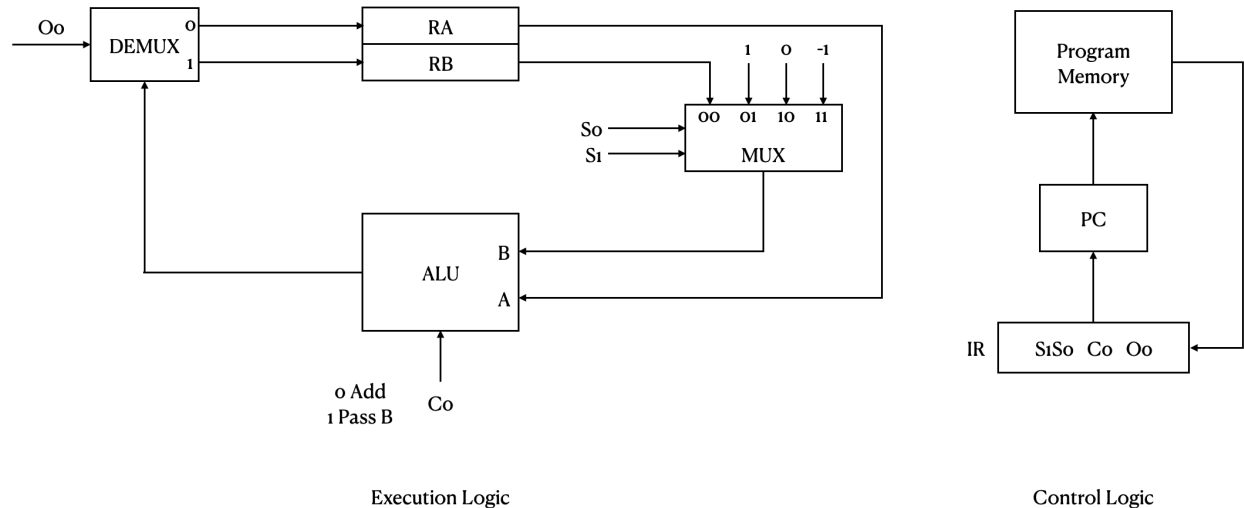


Programmable Circuits (I)

A simple circuit: Count



- We've known registers, ALU, ROM, PC, and IR. Let's build blocks, putting them together to create a simple circuit, count. Of course, you can use these components to build more complex circuits.
- This circuit is composed of two parts: execution logic and control logic.
 - **Execution logic** is to **execute an instruction**. It contains:
 - Two registers: RA and RB
 - An ALU has one control bit, which can control the two operators: Add and Pass input B to output.
 - The ALU has two inputs. Input A is from RA. Input B is one of {RB, 1, 0, -1}, which is the output of a MUX controlled by two bits: S_0 and S_1 .
 - The ALU output is either stored in RA or RB, controlled by the control bit O_0 of a DEMUX. If O_0 is 0, output of ALU is stored in RA; Otherwise, it's stored in RB.
 - **Control logic** contains a program memory (e.g., a ROM), a PC, and a IR, which is **in charge of the program execution**.
 - The **program** is a **set of instructions**. The instruction set is the space of all possible control signals. In this example, the space is 2^4 number of instructions.
 - PC tacks the address of the next instruction.
 - IR stores the current **instruction**, which is a **sequence of control bits** of the execution logic.

- This simple circuit is actually a **state machine**. Once “reset”, RA and RB have 0's to start, IR to 0's, and PC is at 0 to start. Then, the state of the circuit transit between “**fetch**” an instruction and “**execute**” the instruction.
 - At the “fetch” state, IR will read the instruction at the address stored in PC, PC's value will be incremented by 1, and the current state of the state machine will transit to “execute.”
 - At the “execute” state, execution logic will execute the instruction stored in IR, and update the current state of the state machine to “fetch.”

- A program/set of instructions stored in program memory can be:

I0: $S_1S_0 = 01$ $C_0 = 0$ $O_0 = 0$: $RA \leftarrow RA + 1$: Increment RA

I1: $S_1S_0 = 10$ $C_0 = 0$ $O_0 = 1$: $RB \leftarrow RA + 0$: Move RA to RB

I2: $S_1S_0 = 10$ $C_0 = 1$ $O_0 = 0$: $RA \leftarrow 0$: Move 0 to RA

I3: $S_1S_0 = 11$ $C_0 = 1$ $O_0 = 0$: $RA \leftarrow -1$: Move -1 to RA

I4: $S_1S_0 = 00$ $C_0 = 0$ $O_0 = 0$: $RA \leftarrow RB+RA$: Add RA to RB and put it in RA

- You can consider $I_0 \sim I_4$ as the addresses of these instructions and the binary sequence for $S_1S_0C_0O_0$ as the instructions. In a real circuit, the instructions will enable the execution logic to work. However, if you use VHDL to simulate the circuit, you will need to implement the operation corresponding to each instruction.

- A computer can do four things:

- Store data
- Move data
- Manipulate data
- Adjust control flow based on data

- This simple count can do the first three things but not the last one.
- How do we modify this machine to be able to do the last thing?