TD2-6: Building the β machine

October 6th - November 17th , 2021 INFO0012 – Computation Structures

In the last session, you built an ALU (arithmetic–logic unit) and a counter within the LOGISIM simulation environment. These are two elements of our home-made computer, the β-machine. The next practical sessions will lead you to a fully working computer, made from logic gates up (we are using Logisim blocks, but virtually all elements are explained from logic gates up in the theoretical sessions).

Carefully follow the steps below and save regularly. Do not forget to regularly submit the intermediate steps (on eCampus, in the appropriate space)! Taking a fully working machine and then submit restricted versions as intermediate steps will not be accepted.

Each practical session is associated with one deliverable. You can work from home and you can work faster. Support is only provided by the assistant, each week, on the university premises. You cannot be more than one week late, that is, the work planned for one session has to be delivered on the day (midnight) of the next session. This allows to work from home for one session, ask details to the assistant the next week, and fix a few details. But be careful not to accumulate delays. If you are slow or late, it is expected that you work at home to compensate.

1. Take your ALU from previous session, create a subcircuit with it, and create a nice shape for future inclusion in the beta machine.

2. Create a subcircuit with the register file, as shown in the theoretical sessions on slide 7-8. Only implement registers `r0` to `r4`, as well as `r29` to `r31`. Check that it is working properly by setting values, e.g., 1, 2, 3, 4, into registers 1, 2, 3, 4, respectively, and then outputting registers 4 and 2. Also check register 31.

3. Create a subcircuit with the instruction memory. Use a 1 kB (ROM) memory, and simply ignore bits outside the range. Also notice that the instruction memory outputs 32 bit words, so use a memory of 256 times 32 bits. The 2 least significant bits of address, and the 22 most significant bits are thus ignored (use a splitter to select the appropriate 8 bits).

4. In the main circuit create the program counter (you can refer to the previous session). You are free to create subcircuits for +4 and the register, for aesthetics, but it is not mandatory.

5. Connect the program counter to the instruction memory. Load the sample program (on eCampus) and check that at each clock tick, the output of the instruction memory is the next instruction of the program (of course, branching is not yet working).

6. Save and submit your Logisim file. This is the deliverable for Week 1.

7. Create a subcircuit for the control logic, and use a ROM, addressed by the 6 bits of the opcode, outputting 12 bits of control.

8. Connect the above circuits so that the ADD operation with three registers work as expected (see Slide 7-12). Program the control logic for ADD. Check by putting by hand the values 6 and 3

in registers r1 and r2, set an ADD instruction in the instruction memory that would compute the sum of r1 and r2 into r3, check the solution.

9. Save and submit your Logisim file. This is the deliverable for Week 2.
10. Implement ADDC (see Slide 7-15).
11. Logisim allows to save/load ROM images. Reverse engineer the format and create a script that will create a suitable ROM file for the control logic for all the ALU instructions.
12. Check that it is working properly.
13. Save and submit your Logisim file. This is the deliverable for Week 3.
14. Create a circuit for the data memory. Again, 1 kB is largely enough. It is advised to restrict to word addresses only (that is, assume the address will always be a multiple of 4). It has a 32 bits output. Use a logisim RAM with separate load/store ports.
15. Integrate it to you circuit (see Slide 7-21), modify your script above to augment your control logic with LD/ST operations, and check that these operations work as expected.
16. Save and submit your Logisim file. This is the deliverable for Week 4.
17. Implement JMP, BEQ/BNE. Modify your script to support all operations.
18. Check that your machine works as expected, using the sample program provided (on eCampus).
19. Save and submit your Logisim file. This is the deliverable for Week 5 (last deliverable).