Practical Assignment – IPC006 Processoren

Designing the RUN1718 CPU

Hand in before Friday January 19th 2018 at 23:59, by email to `kbrink@cs.ru.nl`.

In this practical assignment, you will create a design for the RUN1718 CPU based on the specifications in the Practicum Manual, and you will implement this design in Hades. You should read the entire assignment before starting with Hades.

Assignment

0. Create a high-level design for your CPU (we recommend doing this on paper, not in Hades). Make sure that you have data flow diagrams for each phase of the instruction cycle, in particular for the Decode+Execute phase of each of the 7 instruction formats listed in Appendix B of the Practicum Manual. Use these diagrams to decide on the input and output interfaces for each component listed in section 3.2 of the manual. Decide which inputs require a multiplexer, and how these will be controlled. Incorporate the feedback you received from your TA on the homework of week 4 ("Data Path") into your design. If you did not yet do the homework of week 4, you should do so now!

   Note that it is easier to change the interfaces to your components at this stage than later on, so it is worth spending time getting this right.

1. Implement the RUN1718 CPU in Hades, according to the description in the Practicum Manual. You should base your implementation on the template files provided on the "Practicum" page on OCW. The Timer and the Register Bank are given, and *should not be changed*. We suggest starting with the Flag Register Bank, the Tester, and the ALU; the Instruction Decoder is often considered more difficult.

   Your design should satisfy the following criteria:

   - meaningful names for components and input- and output ports
   - a neat and sensible layout
   - "Manhattan layout" (only horizontal and vertical wires with 90-degree angles)
   - no unnecessary wire-crossings

   You should thoroughly test each individual component by putting test values on the inputs, and checking that the outputs are as expected. Once you are confident that the individual components are working correctly, connect them according to your data flow diagrams. You can test the implementation of the combined CPU by putting machine codes on the DATABUS_IN input and clicking through execution phases of the CPU, as demonstrated in the lecture. For instructions that read from memory, put a suitable value on DATABUS_IN during the Decode+Execute phase.

2. When your implementation of the CPU is mostly complete, you can also test its interaction with the rest of the computer. The file `computer.hds` provides an implementation of a full computer, with all the elements of the Von Neumann-architecture: a data- and address bus, main memory, and I/O devices. You can include your CPU as a subcomponent of this design.

   You may optionally create a short assembly language program that demonstrates the functionality of your CPU, preferably with a loop or other nontrivial control flow. This can be translated to machine code using the assembler that will be published on OCW; the resulting `.rom` file can then be loaded into the the RAM component of the computer. However, you should not spend time on this until the basic functionality of your CPU is working properly!

What to hand in

The practical assignment *must* be done in pairs. You should hand in the following items before the deadline:

1. The Hades implementation of your CPU (`.hds` and `.sym` files).
   - use descriptive filenames
   - do not include unnecessary files (e.g. `hades.jar`, Hades backup files `.hds_0`, etc.)
   - put all files in the same directory (no subdirectories)
   - your Hades files should not contain absolute filenames (`C:/Processoren/RUN1718/ALU.hds`). Change these to relative filenames (`./ALU.hds`) before handing in. You can either do this manually with a text editor, or use the script on the "Hades" page on OCW.
2. A `README` file in plain text or PDF (not `.docx`), containing:
   - your names and student numbers
   - a description of any unimplemented functionality or bugs in your CPU that you are aware of. If you can clearly explain why a bug exits and what could be done about it, you may still receive points for the corresponding part.
   - (optional) if you think it is necessary to explain certain design choices you made, you can do so briefly here.

   Please keep the `README` as short as possbile.
3. (Optional) If you tested your CPU with an assembly program running on the complete computer, include the commented program. Give a clear description of the behaviour of your program in the comments.

Put all of the files above into a single `.zip` file, named according to your names and student numbers, like this: `s1234567_s7654321_Janssen_deVries.zip`. Send the zipfile by email to `kbrink@cs.ru.nl` before Friday January 19th 2018 at 23:59. Include the phrase "Practicum Processoren" in the subject.

Tips

- Using hierarchical designs (i.e. subcomponents) is encouraged. However, you should store all files in the same directory (this avoids problems when opening your files on a different computer).
- The bit rotation unit in the ALU (for the assembly instructions `ROL` and `ROR`) can be implemented using no more than 5 copies of the component `rtlib.arith.RotateLeft`.
- For all instructions using a 10-bit constant $c10$, this constant is converted to a 32-bit value using sign extension.
- A list of recommended (and *not* recommended) components, and general Tips & Tricks for using Hades, can be found on the "Hades" page on OCW.