SIMPLE RISC PROCESSOR MICROARCHITECTURE

v.3.0

20 dec 2019

1. Execution stage:

The Simple RISC Processor main blocks are sketched in figure 1.

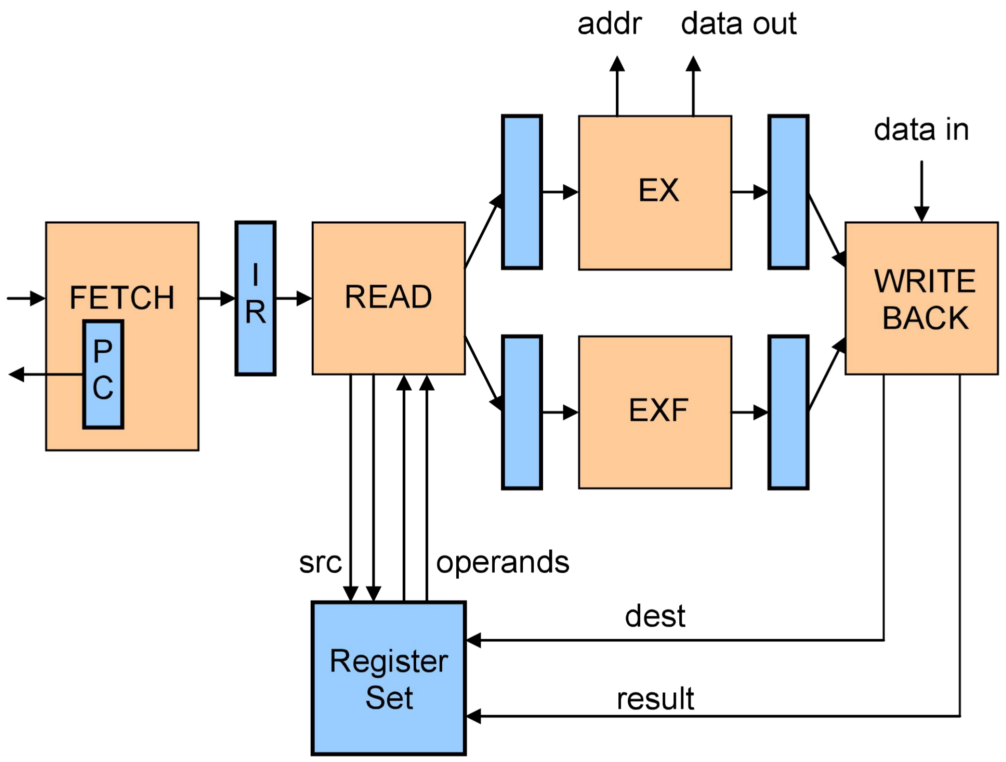

Figure 1
Simple RISC Processor Pipeline.
ISA registers and pipeline registers are shaded in blue.

There are two execution units, EXF for floating point instructions (ADDF and SUBF), and EX for all other instructions (integer arithmetic, logic, load/store and control).
The EX execution unit is exactly the same as in the previous microarchitecture version.
The EXF execution unit has a latency of four clock cycles. It is implemented as a four stage pipeline, each stage corresponding to a particular phase of the floating point addition algorithm (comparison, alignment, addition, renormalization).

2. Read stage

At the end of the read stage, the processor sends the current instruction to the proper execution unit based on its type (opcode). The floating point instructions are sent to the pipeline register in front of the EXF execution unit, the other instructions are sent to the pipeline register in front of the EX execution unit. At each clock cycle the current instruction is sent, together with its operands, to the corresponding execution unit for further processing, and a NOP is sent to the other execution unit. If the instruction must be sent to the EX execution unit but that execution unit is backpressured from the write-back stage, the current instruction remains in the read stage and a NOP is sent to the EXF execution unit.
If an operand is not available (because it is the result of a floating point instruction that has not yet reached its final execution stage, or because it is the data read from memory by an immediately preceding LOAD instruction) the current instruction is kept in the read stage for one more cycle and NOPs are sent to the execution units.

3. Write-back stage

The write-back stage selects a result from the pipeline registers at the execution units output and sends it to its destination in the register set.
There are four possible cases:

- EXF and EX deliver NOPs - no result is sent to the register set;
- EXF has a useful result and EX has a NOP - the EXF result is selected;
- EX has a useful result and EXF has a NOP - the EX result is selected;
- EX and EXF have useful results - the EXF result, which is from an older instruction, is selected;

The last case implies that the unselected result (from EX) is kept into the pipeline register at the output of the execution stage for one more cycle, which in its turn backpressures the pipeline - the instruction that resides in the pipeline register in front of that execution unit is also stalled for one clock cycle.
A special case is when EXF result is selected and the instruction stalled in the write-back stage is a LOAD, whose result is right then coming from the data memory (as `data in`). In this case the `data in` value must be stored internally such that when the LOAD is finally selected to write its result into the register set this stored `data in` value is used (the `data in` input may change because of a newer LOAD that is executed while the older LOAD is kept in write-back stage waiting for selection).

4. Data forwarding

With two execution units, there are six possible data forwarding pathways for each operand read in the Read stage.

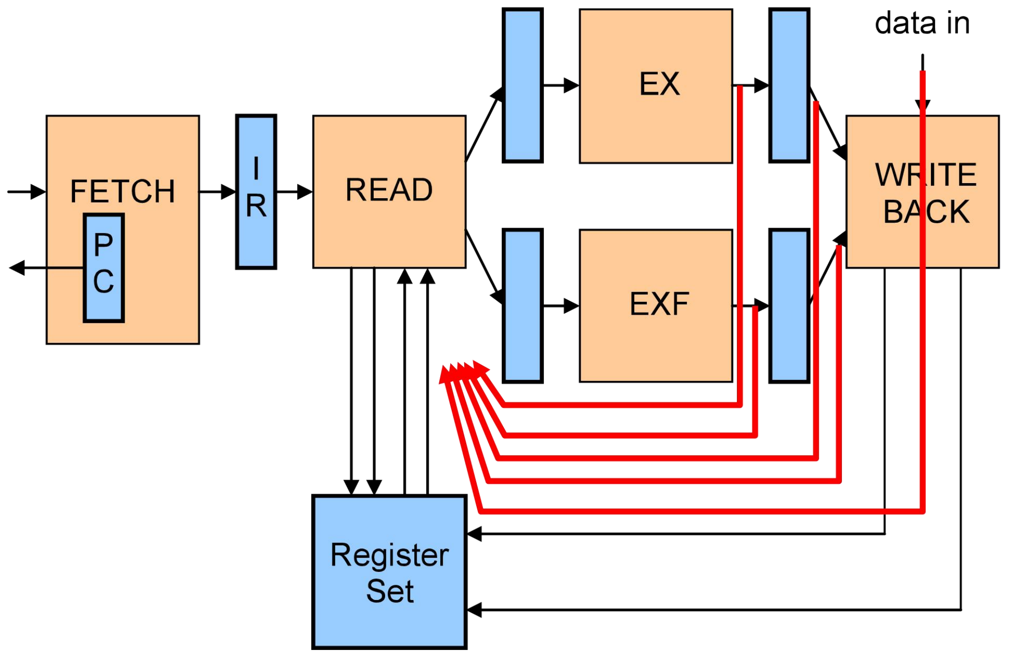

Figure 2
Data forwarding pathways

If two or more previous instructions have the same destination as the source of the operand for the instruction in the read stage, the newest result is forwarded. Because the EXF execution unit has a latency of four clock cycles, the priority for forwarding is always the following (starting with the newest result):

1. EX current output
2. EX registered result, or `data in`, or stored `data in` value
3. EXF current output
4. EXF registered result

Case 2 pathway is determined by the instruction type. If it is a LOAD instruction, the `data in` value is selected. If that LOAD instruction has been stalled in the write-back stage (for at least one clock cycle), the stored `data in` value is selected. For all other instruction types the EX registered result is selected.

5. Jump control

If an uncoditional jump (JMP, JMPR) is executed or a conditional jump (JMPc, JMPRc) is being validated, the PC is load with the target value, and the pipeline registers between Fetch and Read stages, and between Read and Execution stages are loaded with NOPs. The jump instruction itself either advances to the write-back stage (where it does nothing) or, if the write-back stage backpressures the EX execution unit, it dissapears completely (having being replaced with a NOP in the pipeline register in front of the EX execution unit).