# SIMPLE RISC PROCESSOR MICROARCHITECTURE

**v.2.0**

1 December 2025

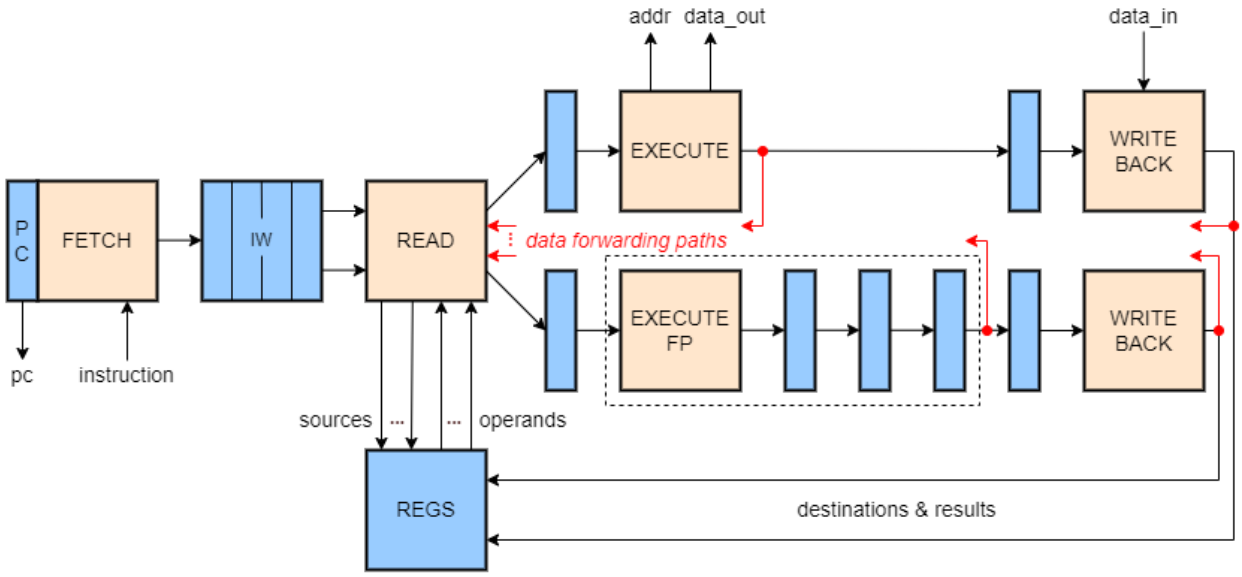


Figure 1
Simple RISC Processor top diagram.
Registers are shaded in blue.

## 1. Processor pipeline

The Simple RISC Processor block diagram is sketched in Figure 1. It has the following pipeline stages:

- The fetch stage (F) keeps and updates the program counter (PC) and reads instructions from the program memory. At the end of the fetch cycle, the fetched instruction is saved in the instruction window (IW).
- The read stage (R) selects from the instruction window a ready instruction (one whose all sources have valid values), reads its operands from the register set, and dispatches it to the appropriate execution unit (E for integer arithmetic, logic, branch and load/store instructions; EF for floating point arithmetic instructions), writing it in the corresponding pipeline register. The read stage may dispatch two instructions at a time, if both are ready, one to the E unit, the other to the EF unit. The Instruction Window (IW) stores up to 4 instructions which are not ready for execution.
- The E execution stage does the actual computation for the integer arithmetic and logic instructions, delivers the memory address (addr) for load/store instructions, delivers also data to be sent to the memory (data out) for store instructions, and outputs the new pc value for branch instructions.
- The EF execution unit computes the result for the floating-point instructions (ADDF and SUBF). It is a 4-stages pipeline.

- The last stage, write back (W), writes the result into the destination register. For the load instruction the result comes from the memory output (data in). If the results from both execution units are available, the W stage writes them simultaneously into the register file.

## 2. Instruction window (IW) and read stage (R)

The read stage manages data and control dependencies.
At the end of the fetch cycle any instruction is saved in one location of the instruction window (except when IW is full and no instruction is dispatched to execution), which acts, for that instruction, as the pipeline register in front of the read stage. The instruction stays there as long as it is in the read stage. When the instruction is selected to be dispatched to execution, another fetched instruction may replace it, or the location state is changed to free.
The newest instruction in IW has the following restrictions that eliminate any WAR, WAW or control hazards:

1. If it is a branch instruction, no newer instruction can enter the IW until the branch instruction is dispatched to execution. This prevents instructions from a wrong branch to be executed before the branch instruction and thus writing the register file.
2. If its destination is the same as the destination of any other instruction inside the processor, it cannot be dispatched to execution stage and no newer instruction can enter the IW until this WAW dependency is cleared.
3. If its destination is the same as the source of any other instruction inside the IW, it cannot be dispatched to execution stage and no newer instruction can enter the IW until this WAR dependency is cleared. That prevents this instruction from being executed out-of-order and to overwrite a source register than an older instruction, blocked for some reason in IW, hasn't yet chance to read it.

In any of the above 3 cases, the newest instruction is kept in the IW and the fetch stage is blocked too (it's PC is not updated). The older instructions present in the IW are not subject to the above restrictions (they are supposed to be solved earlier).
Any instruction from the IW may be dispatched to execution if the following conditions are met:

- There is not any older instruction inside the IW or the EF (in its first 3 pipeline stages) whose destination is the same as one of its sources;
- It has no WAW or WAR dependency if it is the newest instruction in IW (see conditions 2 and 3 above);

If two or more instructions may be dispatched to the same execution unit, always the older one is selected.

## 3. Branch instruction execution

After a branch instruction is fetched, by default the next instructions to be fetched are from the next locations in the program memory.
When the unconditional branch instruction is executed, or the conditional branch instruction condition is evaluated to true at execution, the target instruction address is saved into the PC, the newest instruction in the IW is annulled (changed to NOP) and the fetched instruction is also changed to NOP.

## 4. Data forwarding

If an instruction waits in IW for an operand, because one of its source registers is the same as the destination of an older instruction still inside the processor (RAW dependency), the result of that older instruction may be forwarded from that older instruction using the data forwarding paths (shown in red) when the older instruction passes through the W stage, the E stage or the last stage of the EF pipeline. The forwarding path will be used only if the waiting instruction is dispatched to the execution stage.
The older instruction may write its destination before the waiting instruction is dispatched. In that case the RAW dependency ends and the waiting instruction (which is no longer waiting for that operand) will read the operand directly from its source register.

## 5. Register file

The register file has 4 read ports, to allow simultaneous dispatch of 2 instructions, and 2 write ports to allow simultaneous writes from the write back stage of both the integer pipeline and the floating-point pipeline. It is assumed that the writes are always to different registers, since the WAW dependency is resolved during the read stage by blocking in the IW the newest instruction with the same destination as the one of an older instruction in processing.