SIMPLE RISC PROCESSOR MICROARCHITECTURE

v.1.0

20 December 2016

1. Processor pipeline

The Simple RISC Processor block diagram is sketched in Figure 1.

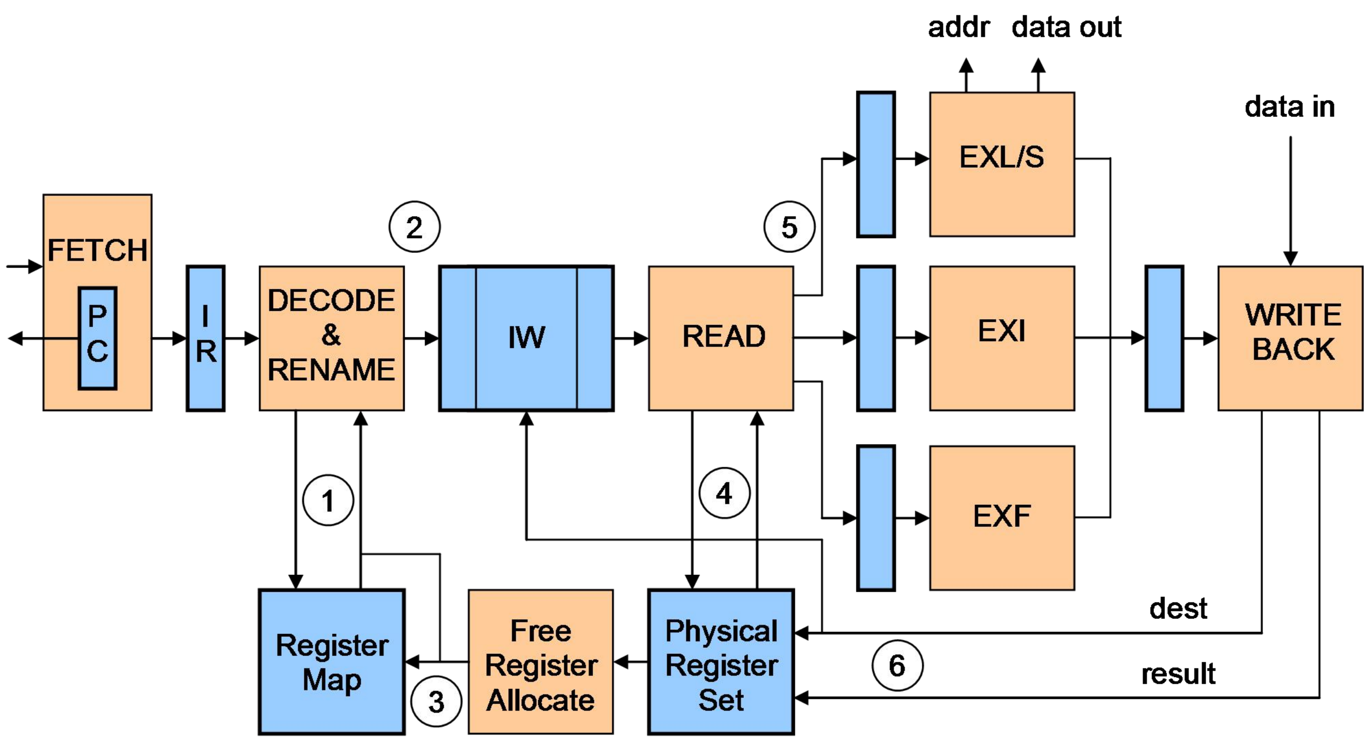

Figure 1
Simple RISC Processor main diagram.
Registers are shaded in blue.

The processor pipeline is at least 5 stages deep.

- The fetch stage (FETCH) keeps and updates the program counter (PC) and reads instructions from the program memory. Its output may be assimilated with the instruction register (IR).
- The decode and rename stage (DECODE&RENAME) decodes all instruction fields and maps the source and destination fields values to the physical registers addresses. If successful, this stage ends with the renamed instruction entering the Instruction Window (IW) and proceeding to the the read stage.
- The read stage (READ) selects 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 (EXI for integer arithmetic, logic and control instructions; EXLS for load/store instructions; EXF for floating point arithmetic instructions), writing it in the corresponding pipeline register.
- The Instruction Window (IW) keeps track of all instructions that have passed the renaming stage and have not yet finished processing.

- The execution stage (EXI, EXLS or EXF) does the actual computation for the arithmetic and logic instructions, computes the necessary address for the branch instructions ,and delivers the memory address (addr) for load/store instructions. It delivers also the data to be sent to the memory (data out) for the store instruction.
- The last stage, write back (WRITE BACK), sends back the result to the register set and broadcasts the destination address to all instructions in the instruction window. For the load instruction the result comes from the memory output (data in).

2. Instruction window and physycal register set

The Simple RISC Processor has a superscalar microarchitecture with a centralized dynamic instruction scheduling. The Instruction Window manages data and control dependencies, keeping relevant data for all instructions that are in the read, execute or write-back stages.
Each entry of the Instruction Window (Figure 2) keeps all instruction fields necessary for further instruction processing: *operation* (op), result *destination* address (dest) and operand *sources* addresses (src1 and src2). It has additional fields required for dynamic scheduling: instruction's *age* (age) and *valid* operand bits (v1 and v2).

age	op	dest	src1	v1	src2	v2

Figure 2
Instruction Window register fields.

Each register from the register set has, beside its *value*, two supplementary fields used for register renaming management (Figure 3): a *valid* bit indicating that the register value is ready to use and a *counter* (cnt) field which indicates how many instructions that needs this register value have not yet read it.

value	v	cnt

Figure 3
Physical register fields.

3. ModifiedThornton algorithm

Instruction issue may be out of order (O-o-O) if, while it stays in the read stage, other newer instructions proceed to the execution stage. If two ore more instructions in the read stage are ready for execution, a round robin mechanism selects the instructions to be issued for execution. It is dispatched to the proper execution unit according to the instruction operation.
If two or all execution unit finish simultaneously their operations only one result is allowed to proceed to the last stage (the oldest), the other one being stalled in the execution stage. The stalled execution unit will also block the instruction issue to its input, but has no effect on the instruction issue to the unstalled execution unit.
Write back may also be out of order, as a result of both out of order issue and different execution unit latencies (EXF has at least 4 internal stages).

3.1 instruction decode and rename stage

The architectural addresses of the destination and source registers are mapped to physical addresses according to the Register map table ① (see Figure 1 for this and next circled numbers).

The decoded instruction with the renamed destination and sources fields are pushed into the instruction window ②, together with the valid bits as they are read from the register set. (It is possible that while the instruction is promoted to the read stage its source register is written at that same moment. The valid bit of the src field must be set simultaneously!).
If the destination physical register is not free (because there is at least one instruction in the read stage that needs its value), the Free Register Allocate circuit finds a free physical register to be used immediately as the new renamed destination and updates the Register map table ③.
If the Instruction Window is full and no instruction exits it (no instruction in the write-back stage), or if there is no any free physical register (but only if the instruction needs a destination), then the instruction is stalled at the decode&rename stage (and the fetch stage is stalled too), waiting for the structural dependency to be cleared.

3.2 instruction read (issue and dispatch) stage
An instruction is entitled to issue only if all its operands are valid (some instructions need only one operand or none at all) and the execution unit is ready. If not, the instruction stays in the read stage untill all the aforementioned conditions are met.
If two or more instructions are ready for execution, the oldest one is selected to be issued. It will read its operands ④ and will be dispatched to the proper execution unit ⑤.

3.3 write back stage
At the end of the write back stage the instruction writes its result into the destination register and validates the register value ⑥. Also, the destination address is sent to the Instruction Window in order for all instructions awaiting this result to validate the corresponding source fields. After that the instruction exits the Instruction Window (its entry is set free).