∇ TRISC

TRISC-16 Datasheet

Features

- 16-bit RISC CPU with 8 general purpose registers.
- 14 simple and fast instructions that can perform simple procedures when put together.
- Three-stage instruction execution cycle: fetch, decode and execute.
- Can address up to 64 KB of program and data memory.

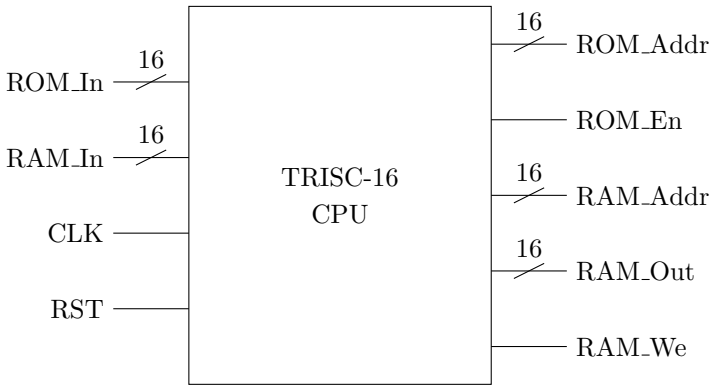

Figure 1: CPU Pinout

Applications

- Learning and teaching computer engineering.
- Very simple computations.

General Description

The TRISC (Training-RISC) was made to facilitate the learning process of computer engineering subjects, like Computer Architecture and Organization, Embedded Systems, Assembly Programming and Digital Systems Design, decreasing the learning curve of these subjects.

Design choices were made to maximize ease of learning, such as a RISC design philosophy and using pure Harvard architecture. Some common instructions have been removed to simplify the instruction set and CPU diagram as much as possible. This CPU was also made in VHDL to widen the learning capabilities.

Even so, this CPU can perform some of the important operations that more complex processors can also perform, such as: logical, arithmetic and memory access operations.

TRISC Architecture

The TRISC 16-bit architecture, developed for the TRISC-16 processor, focus on an organization that facilitates the understanding of the processor, having 4 modulated and integrated units: **Control Unit** (controls the processor's other units), **Datapath** (perform the CPU operations), **ROM** (the program memory) and **RAM** (the data memory). The functional block diagram of the processor can be seen in the figure bellow.

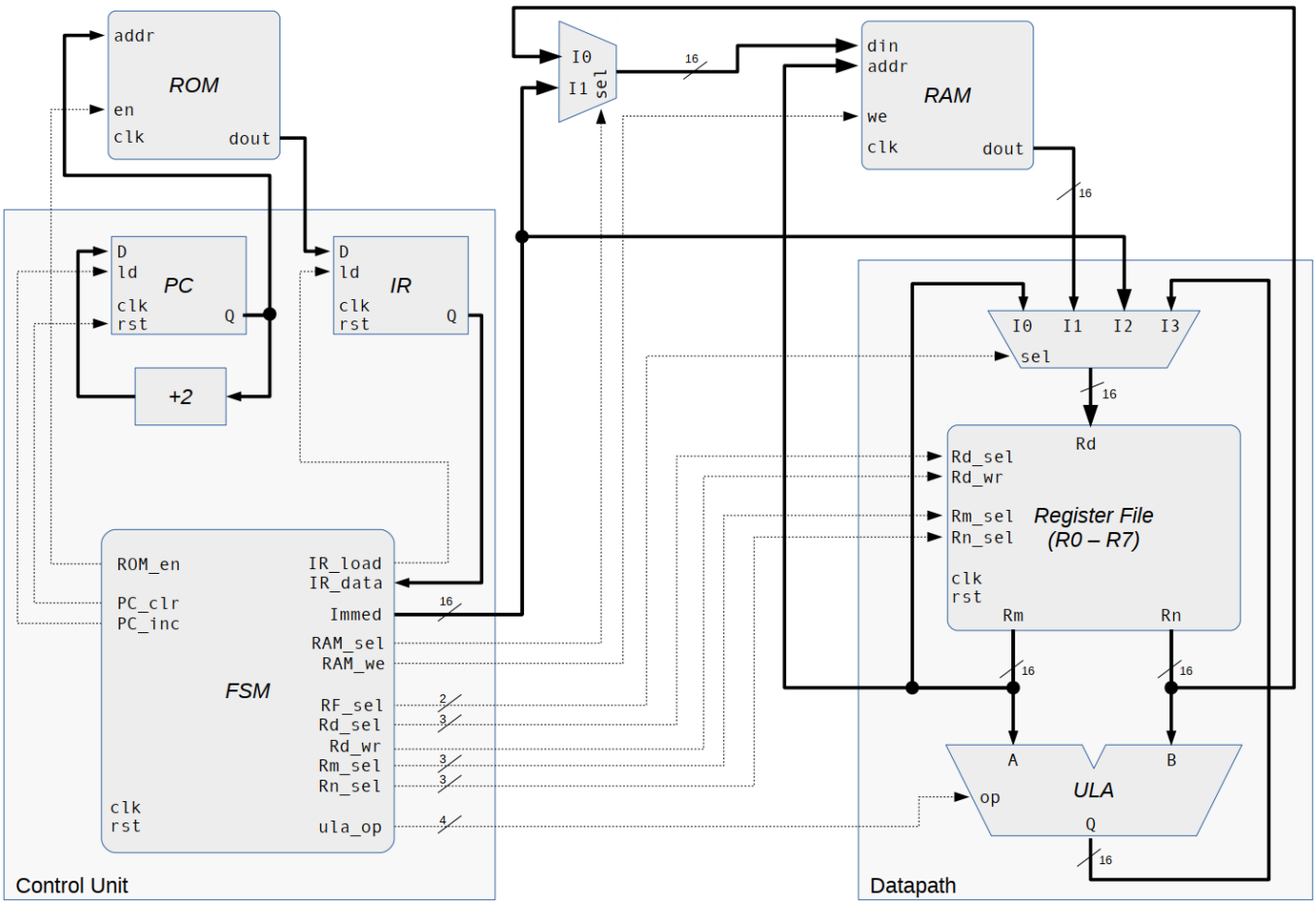

Figure 2: Processor's Block Diagram

Being a 16-bit CPU (not large and complicated as 32-bits, but not insuficient as 8-bits) it can work with 16-bit internal and external data and address buses. It has 8 general purpose 16-bit registers.

The two main units are connected, where the Control Unit contains a Finite State Machine-based controller, that issues diverse control signals to the system (such as RAM_we, that indicates whereas the operation is a read or write operation). It also has internal registers, such as the PC (program counter) and IR (instruction register).

Technical Details

Bellow are the summarized the technical details of the processor, being the project decisions that brought the TRISC-16 to a state of easier comprehension.

- 16-bit processor made in VHDL.
- 8 general purpose 16-bit registers (R0-R7).
- RISC design.
- Pure Harvard Architecture.
- Up to 64 KB of program and data memory.

Instruction Set

The Instruction Set Architecture (ISA) of the TRISC-16 is shown bellow, having 14 simple, but fast and usable instructions.

Instruction	**Operation**	**Type**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NOP	nop	NOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MOV Rd, Rm	Rd = Rm	MOV	0	0	0	1	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	-	-	-	-	-
MOV Rd, #Im	Rd = #Im	MOV	0	0	0	1	1	Rd2	Rd1	Rd0	Im7	Im6	Im5	Im4	Im3	Im2	Im1	Im0
STR [Rm], Rn	[Rm] = Rn	STORE	0	0	1	0	0	-	-	-	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
STR [Rm], #Im	[Rm] = #Im	STORE	0	0	1	0	1	Im7	Im6	Im5	Rm2	Rm1	Rm0	Im4	Im3	Im2	Im1	Im0
LDR Rd, [Rm]	Rd = [Rm]	LOAD	0	0	1	1	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	-	-	-	-	-
ADD Rd, Rm, Rn	Rd = Rm + Rn	ALU	0	1	0	0	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
SUB Rd, Rm, Rn	Rd = Rm Rn	ALU	0	1	0	1	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
MUL Rd, Rm, Rn	Rd = Rm * Rn	ALU	0	1	1	0	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
AND Rd, Rm, Rn	Rd = Rm AND Rn	ALU	0	1	1	1	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
ORR Rd, Rm, Rn	Rd = Rm OR Rn	ALU	1	0	0	0	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
NOT Rd, Rm	Rd = ¬Rm	ALU	1	0	0	1	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	-	-	-	-	-
XOR Rd, Rm, Rn	Rd = Rm XOR Rn	ALU	1	0	1	0	0	Rd2	Rd1	Rd0	Rm2	Rm1	Rm0	Rn2	Rn1	Rn0	-	-
HALT	halt	HALT	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 1: Processor Instruction Set

The instruction set was made to teach the inner workings of a CPU, not to process any meaningful data. There are simple instructions with unfilled "bit slots", which anyone using can implement more complex functionalities (in VHDL), add new instructions or replace existing ones.

Instruction Cycles

There are 3 main states for the CPU execution: **fetch** (instruction brought from the ROM to the IR), **decode** (find out which instruction will be executed, that is, the group, and which operands will be used) and **execution** (executes the specific instruction). There are 6 instructions groups with similar instructions.

Additionally, there is a initial state, **init**, to setup the correct signals in the reset event.

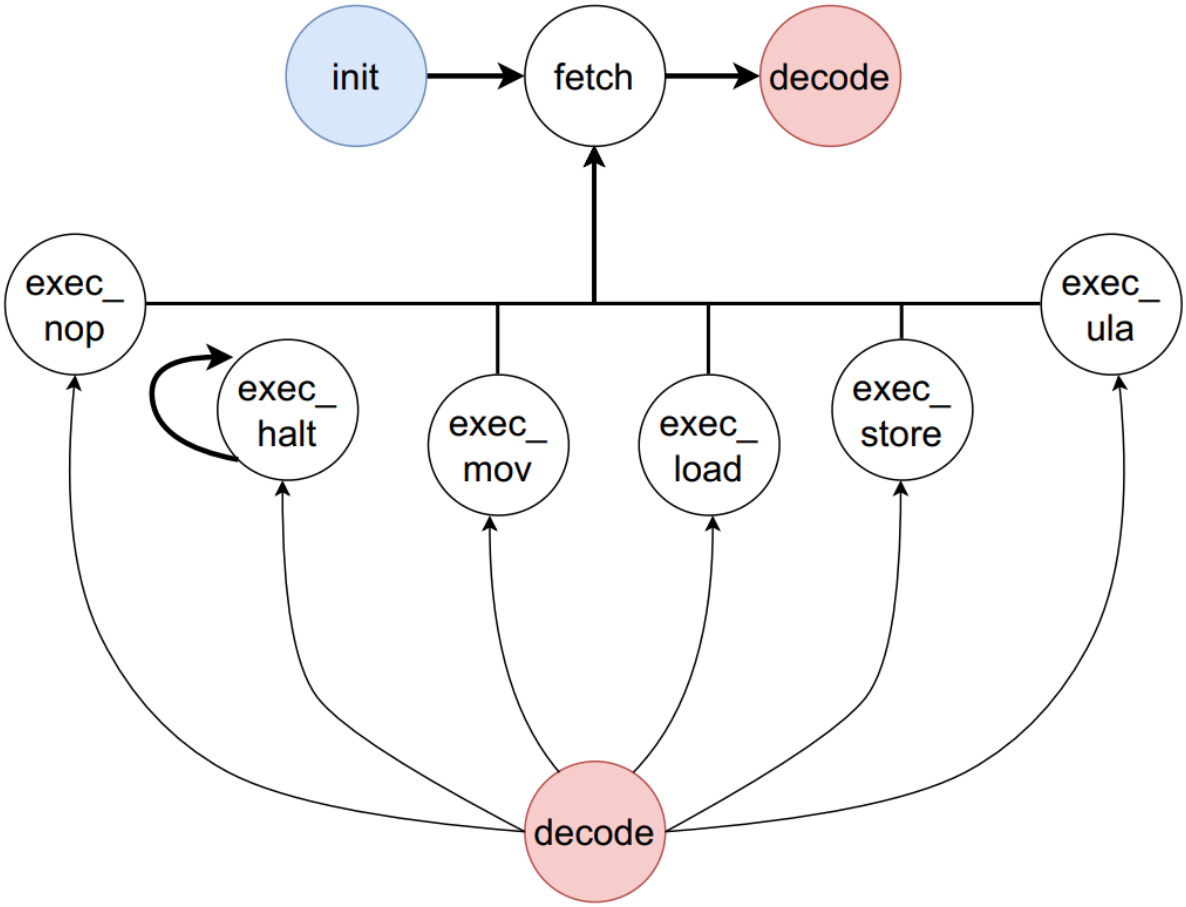

Figure 3: Control Unit Execution Cycles

Control Unit Signals

The control signals that the FSM emits are shown bellow:

Signals	Init	Fetch	Decode	Exec HALT	Exec MOV	Exec LOAD	Exec STORE	Exec ALU
PC_clr	1	0	-	-	-	-	-	-
PC_Inc	0	1	0	-	-	-	-	-
ROM_en	0	1	0	-	-	-	-	-
IR_load	0	1	0	-	-	-	-	-
Immed	0x0000	-	-	-	-	-	-	-
RAM_sel	0	0	-	-	-	-	1 / 0	-
RAM_we	0	0	-	-	-	-	1	-
RF_sel	00	-	-	-	10 / 00	01	-	11
Rd_sel	000	-	-	-	-	-	-	-
Rd_wr	0	0	-	-	1	1	-	1
Rm_sel	00	-	-	-	-	-	-	-
Rn_sel	000	-	-	-	-	-	-	-
alu_op	0000	-	-	-	-	-	-	-

Table 2: Processor Control Signals

These signals are emitted from the control unit to the datapath, ROM and RAM modules.

There are some control signals issued conditionally to execute a instruction. If the bit 11 of the instruction is 1, than the MOV and STR instructions works with immediates. If so, for the store instruction the **RAM_sel** must be 1 to select the immediate, or 0 otherwise, and for the mov instruction the **RF_sel** must be 0b10 to select the immed to enter the register file, and 0b00 to select Rm.