Report

The Tiny Register Machine (TRM)

Author(s):
Wirth, Niklaus

Publication Date:
2009

Permanent Link:

https://doi.org/10.3929/ethz-a-006819428 →

Technical Memo 25.1.2009, rev. 2.10.2009

The Tiny Register Machine (TRM)

Niklaus Wirth

Introduction

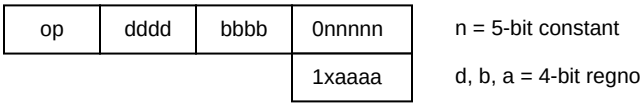
The TRM is a straight forward RISC architecture implemented on a Field Programmable Gate Array (FPGA). The TRM was designed as part of a project to develop a computer for practical experimentation with multiprograms on a genuine multiprocessor device. The project is implemented on a standard development board. The FPGA contains 12 TRMs connected by a network with 8-bit data lines. In addition, the board contains a large memory and several other features. Essential, however, is considered the setup of 12 identical processors with local program and data memories. Therefore, the design embodies the paradigm of *process cooperation by message passing*.

1. The Architecture

The processor contains an arithmetic-logical unit (ALU) and a shifter. The 32-bit operands and results are stored in a bank of 16 registers. The local data memory consists of 2048 words of 32 bits. The local program memory consists of 4096 instructions with 18 bits.

2. The Instruction Set and its Encoding

There are 16 instructions. Instruction have 4 fields, called *op, d, b, a*, each consisting of 4 bits, except *a* which consists of 6 bits. *op* defines the operation, *d, b*, and *a* designate a register. The latter may instead also denote a constant. The format is:



2.1. Register operations

code	operation	
0	AND	R.d := R.b & R.a (in place of R.a may stand the literal n)
1	BIC	R.d := R.b & ~R.a
2	OR	R.d := R.b | R.a
3	XOR	R.d := R.b % R.a
4	ADD	R.d := R.b + R.a
5	SUB	R.d := R.b - R.a
6	MUL	R.d := R.b * R.a
7	NEG	R.d := ~R.a
8	LIT	R.d := n (10-bit literal)
9	ROR	R.d := R.b rotated by R.a or by n

2.2. Load and Store instructions

12	LD	R.d := Mem[R.b + n]
12	LD	R.d := Mem[R.b + R.a]
13	ST	Mem[R.b + n] := R.d
13	ST	Mem[R.b + R.a] := R.d

LD	dddd	bbbb	0nnnnn	n = 5-bit offset
LD	dddd	bbbb	10aaaa	
ST	dddd	bbbb	0nnnnn	
ST	dddd	bbbb	10aaaa	

2.3. Branch instructions

14	Bc	conditional jump PC := PC + n
15	BL	branch and link R15 := PC; PC := PC + n

Bc	cccc	nnnnnnnnnn	n = 10-bit offset
BL	nnnnnnnnnnnnnn		n = 14-bit offset

10	BLR	conditional branch and link reg R15 := PC; PC := R.a
11	BR	branch reg PC := R.a (return from procedure)

BLR	cccc	xxxx	10aaaa
BR	xxxx	xxxx	10aaaa

2.4. Special instructions

BR	xxxx	xxxx	111111	return from interrupt
BR	xxxx	xxxx	0000me	set mode
BLR	cccc	xxxx	10aaaa	trap tp a

3. Condition Codes

The condition register consists of 4 bits. N and Z are set by all register instructions, C and V by the ADD and SUB instructions only. The ROR instruction sets C equal to bit 0 of the rotated result.

N = bit 31 of result	C = carry out
Z = all 32 bits are zero	V = overflow

Branch instructions contain 4 condition field. Its value determines, which condition is tested. S stands for N xor V.

0000	Z	EQ	1000	HI	~(~C | Z) uns.
0001	~Z	NE	1001	LS	~C | Z uns.
0010	C		1010	GE	~S
0011	~C		1011	LS	S
0100	N		1100	GT	~(S | Z)
0101	~N		1101	LE	S | Z
0110	V		1110		flase (jump always)
0111	~V		1111		true

4. Interrupts and Traps

The TRM interface has 3 interrupt signals *irq0 – irq2*. If any of them is high, the processor is interrupted. The processor executes the following steps similar to a BL instruction:

1. It switches from normal mode to interrupt mode. Each mode uses a separate bank of 16 registers.
2. The PC and the condition code register are deposited in register 15.
3. The processor branches to location 2 for *irq0*, 3 for *irq 1*, and 4 for *irq2*.
4. Interrupts ar disabled.

The processor is returned to the normal mode by a special BR instruction with a = 15. This instruction causes the following actions:

1. The PC and the condition codes are reloaded from register 15.
2. The processor switches to normal mode.
3. Interrupts are enabled.

The processor has two 1-bit status registers:

intEnb	interrupts are disabled (0) or enabled (1)
mode	the processor is in normal (0) or interrupt (1) mode

These registers can be set/reset by the special instruction, setting the mode register to *m*, and the enable register to *e*. When the processor is started (reset) both registers have the value 0, i.e. interrupts are disabled, the processor is in normal mode, and PC is set to 0.

For traps a BLR instruction with a literal address is used. This branch is condition, making it useful as a trap for array bound and overflow checks and for type guards. n is the address of the trap destination.

Interrupt handlers in TRM-Oberon

An interrupt or trap handler is specified in Oberon using a special feature of the TRM compiler, as shown by the following example, where interrupts are assumed to be caused by a timer at source *irq2*.

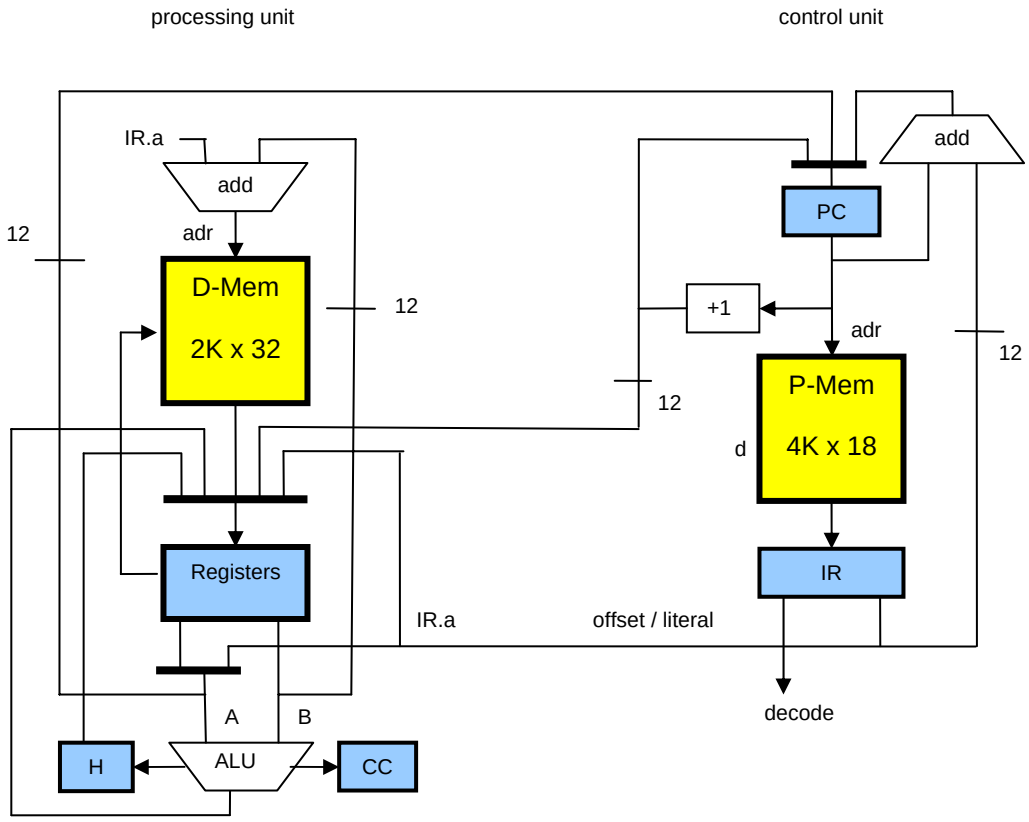
```
MODULE M;
  VAR cnt, blink: INTEGER;

  PROCEDURE* Interrupt @4;
  BEGIN INC(cnt); PUT(0F07H, 0);  (*reset request*)
    IF cnt = 500 THEN blink := 1 - blink; PUT(0F0CH, blink); cnt := 0 END ;
  END Interrupt;

BEGIN cnt := 0;
  SETPSR(2);  (*enter interrupt mode*)
  SETSP(400H); (*set stack pointer in interrupt mode to 400H*)
  SETPSR(1);  (*return to normal mode*)
  REPEAT UNTIL FALSE
END M.
```

The asterisk following the symbol PROCEDURE causes the special return branch to be used at the end of the procedure. The @n causes a branch instruction to the procedure to be placed in loaction *n*. The statement PUT(0F07H, 0) resets the timer interrupt request. Before entering interrupt mode, the stack pointer of the interrupt register bank must be initialized. This is done in the module's body by the statement SETSP(400H) . Statement SETPSR(2) causes the processor to use the register bank of the interrupt mode, and SETPSR(1) switches back to normal mode and enables interrupts. Statement PUT(0F0CH, blink) outputs the value *blink* to a set of 10 LEDs.

5. Block diagram of the TRM



6. The TRM Interface

The hardware interface of the TRM processor consists of 8 input and 3 output signals. The inputs are:

clk	the processor clock (97 MHz)
clk2	a clock shifted by half a cycle
clk3	double frequency clock for the multiplier
rst	reset, active low, sets PC to 0
irq0, irq1, irq2	interrupt signals, active high
inbus	32-bit input bus

The output signals are

ioadr	5-bit I/O address
iowr	write enable
outbus	a 32-bit bus