TOY/2 - a minimalist 16-bit CPU

TOY/2 is a minimal 16-bit processor inspired by the TOY CPU described in [1]. The architecture was improved in many areas:

- TOY/2 was implemented using pipelining - instructions are executed while the next instruction is fetched.
- The program counter is incremented by the main ALU while the operand is read. This reduced complexity. The accumulator and the program counter switch places for this. [3] also does this.
- The instruction set was improved. Some redundant instructions were dropped, others added. Control logic is minimal (max. 4 levels of logic).
- TOY/2 can address a full 64K word program and memory address space.

TOY/2 was implemented in 3µm CMOS (Philips / Faselec SACMOS), and takes up a whopping 3300 transistors (excluding I/O pads). It would fit into a small corner of a FPGA today. It was designed by Pascal Dornier and Stephan Paschedag at ETH Zürich in 1988 as part of a course in chip design.

The Data Path

TOY/2 has a simple 16-bit data path based on the ALU described in [2].

The programmer sees the following registers:

- A accumulator
- PC program counter
- T temporary register for indirect store

and the following flags:

- C carry
- Z zero

The Instruction Set

All instructions are 16-bit: 4 bits for the opcode, and 12 bits for the direct address.

Opcode	Mnemonic	Description	Function
0	`JMP vec`	Indirect jump	PC:=vec
1	`ADC src`	Addition	A:=A+[src]
2	`XOR src`	Exclusive or	A:=A xor [src]
3	`SBC src`	Subtract	A:=A - [src] - C
4	`ROR`	Rotate right through carry	A,C:=A,C ror 1
5	`TAT`	Transfer to T	T:=A
6	`OR src`	Logical or	A:=A or [src]

7	222	illegal	undefined
8	AND src	Logical and	A:=A and [src]
9	LDC src	Load A, clear carry	A:=[src]; C:=0
A	BCC vec	Indirect jump if carry clear	IF C=1 THEN PC:=[vec]
B	BNE vec	Indirect jump if not zero	IF Z=0 THEN PC:=[vec]
C	LDI	Load A indirect	A:=[A]
D	STT	Store T indirect	[A]:=T
E	LDA src	Load A, don't clear carry	A:=[src]
F	STA dest	Store A	[dest]:=A

TOY/2 does not implement a stack, call instructions (which can be implemented using indirect jumps) or interrupts. The focus was on a minimalist design that could be implemented in the time available (and that was so simple that the assistants would let us do it).

The architecture does not make very effective use of code space. It should still do a bit better than a Turing machine...

Performance was still projected to be quite respectable. For example, the prime number sieve benchmark would take about 1.2s at 4 MHz compared to 0.7s on a 80286 (10 MHz, no wait states) or 0.35s on a 68020 (16 MHz, 1 wait state, cache enabled).

References

[1] Microcoded versus Hard-wired Control, Phil Koopman, Byte, 1/87

[2] The Architecture of Microprocessors, Francois Anceau, Addison-Wesley, 1986

[3] Strip Architecture Fits Microcomputer Into Less Silicon, Electronics, 1/27/1981