Kronos: Processor Family for High-Level Languages

Dmitry N. Kuznetsov, Alexey E. Nedorya, Eugene V. Tarassov, Vladimir E. Philippov, Marina Ya Philippova

Kronos: Processor Family for High-Level Languages

Dmitry N. Kuznetsov[1], Alexey E. Nedorya[2], Eugene V. Tarassov[3], Vladimir E. Philippov[4], and Marina Ya. Philippova[5]

[1] California, USA; leo.kuznetsov@gmail.com
[2] Veberi Ltd., Yaroslavl, Russia; a.nedoria@webaby.ru
[3] Intel (Wind River), Inc.; eugene@largest.net
[4] Ershov Institute of Information Systems, SB RAS, Novosibirsk, Russia; fil@iis.nsk.su
[5] xTech Ltd., Novosibirsk, Russia; flm@xtech.ru

Abstract. The article describes the history of the Kronos project that took place in the middle of the 1980s in Akademgorodok, Novosibirsk, and was devoted to creating the first Russian 32-bit processor, Kronos. The basic principles of high-level languages oriented architecture design are presented by the example of the Kronos processors family architecture. The article contains a short review of the original multiuser multitask operation system, Excelsior, that was designed and implemented within the project and used for the Kronos family processors.

Keywords: Kronos, Modula-2, 32-bit processors, high-level languages oriented architecture, OS Excelsior

1 Background

Kronos is a generic name of a family of 32-bit processors for creating micro- and mini-computers. The Kronos processors architecture was designed to support high-level programming languages (C, Modula-2, Oberon, Pascal, Occam, etc.) that allowed implementing the newest ideas in software development and computer usage.

The Kronos family of processors was developed in the Novosibirsk Computing Center (SB AS USSR) within the framework of the MARS project ("Modulnye Asinkhronnye Razvivayemye Sistemy" – Modular Asynchronous Open Systems) by the Kronos research team of the START research and technology group (RTG) (1985-1988). The team leader was Dr. Vadim Kotov; the principal developers were Dmitry (Leo) Kuznetsov, Alexey Nedorya, Eugene Tarassov, and Vladimir Philippov. In 1990, START was transformed into the Institute of Informatics Systems named after A.P. Ershov SB RAS.[1]

Kronos processors were manufactured as a pilot series mainly for building development computers designed for software development. The Kronos 2.6 processors were used for small-lot production of the Kronos-2.6WS Workstation. The first prototype of the workstation was demonstrated at the "Science-88" exhibition that was held in Moscow in 1988. Kronos-2.6WS workstations were used as development computers in a number of projects in the defense industry of the USSR.

[1] http://www.kronos.ru/

In particular, the Applied Mechanics Research used Kronos for developing satellite software and Production Association named after M.F. Reshetnyev (NPO PM, Krasnoyarsk-26).

One of the remaining Kronos-2.6WS workstations can be found in the Science Museum, London, UK, the Polytechnic Museum in Moscow, the Museum of Siberian Branch of the Russian Academy of Science (SB RAS, Novosibirsk), Novosibirsk State University (NGU), and IIS SB RAS in Novosibirsk.

2 The Kronos Project

The goal of the Kronos project was to create a multi-purpose 32-bit processor with hardware support of high-level languages. Kronos processors were used in open-architecture embedded microcomputers and single and multi processor workstations.

Modula-2, created by N. Wirth in 1975 based on Pascal (and later Oberon language), has been chosen as the main project language [1]. With the inheritance of the best qualities of Pascal, Modula-2 nevertheless had a number of distinguished features, making it most suitable for developing and implementing the software that fully met the requirements of its time. Such features included:

- Modularity (introduced the concept of module and possibility of its decomposition into definition and implementation parts);
- Well-developed data and control structures;
- Static type checking;
- Procedure types that allow dynamic parameterization of code by external actions;
- Low-level programming helper functions that made the control less rigid and enabled structural data memory mapping.

Architecture of Kronos is loosely based on the Lilith personal computer developed by N. Wirth at the ETH, Zurich. However, Kronos team made many independent decisions and moved to 32-bit architecture. For instance, while the sets of RAM value access commands, arithmetical and logical operations, and control structures remained unaltered, there were substantial changes in the architectures of process interaction, interrupt subsystem, addressing and working with peripheral devices. Many simplifications were achieved due to the 32-bit architecture.

The 32-bit word format allowed the use of the Kronos processors for solving computationally extensive problems. Implementation of some AI (Artificial Intelligence) prototype systems has been made possible due to huge 32-bit address space (up to 4 billion words). Hardware support of event interrupts, processes synchronization and inter-process communication as well as compact code made Kronos processors the preferred choice in some real-time automation systems.

3 Kronos Processors Architecture

Kronos architecture was different from the traditional ones and had the following differentiating features:

1) Expression calculation took place on a fast hardware stack of small fixed depth, preserving stack content over function calls and switching process (threads);
2) The code and data of any processes were separated which resulted in ability to re-entry all programs and even their separate parts (modules);
3) Position independent code segment (PIC) with displacement table for starting a procedure addresses simplified the function call instructions and removed necessity for static linking;
4) Position independent data (PID) addressing scheme allowed a mapping of object-oriented concepts of modern programming languages. Independent base registers for local, global and external objects streamlined compilation process for high level languages;
5) Cross module base register addressing scheme enabled dynamic program loading – binding - execution;
6) Kronos architecture has been extended for operations with complex data structures.

To describe specific features of the Kronos processors architecture it is necessary to define some concepts. The program that is currently executed by a processor is a process. A process includes the following components:

a) Tables of currently downloaded code segments of separately compiled modules [DFT];
b) Global data field [G];
c) Code segment [F];
d) String constant fields [STRINGS];
e) Procedure stack (P-stack) [P];

In brackets [] there are base registers for the component address evaluation. Their meanings are as follows.

DFT (data frame table) – Supports dynamic binding of the separately compiled modules.
P – Stack has additional marking with the following indicators:
L – Base register that keeps the address of the beginning of the local data of the current procedure;
S – Base register that keeps address of the beginning of the free top part of the P-stack;
H – P-stack limiter (the limit of S extension).
P – Stack overlap (S >= H) is checked by hardware throwing interrupt exception.

The Kronos-architecture supports the following address schemes: local, global, external, intermediate and indirect. The values of all structural variables (arrays and records) were represented indirectly with reference to the beginning of the selected memory zone. Structural components of such values were located directly inside them. Thus, it was possible to limit the maximum size of a local data section of procedure to 256 words for fast addressing; that is, in terms of Modula-2 a procedure fast address first 256 local variables and takes more cpu cycles if the number of local variable is greater than 256.

Expressions were calculated on the register based fast arithmetic stack – the A-stack. The compiler statically traced the A-stack overlap and if necessary saved the content of A-stack to memory and restored it as needed.

M-code is a sequence of bytes without any bit tags. The first byte of the instruction determines how many (if any) bytes of immediate operand will follow. Most of the commands are in single-byte format. Extracting commands and operands from the sequence was carried out in parallel with incrementing PC (program counter) control register. M-code had four modifications of the length of an immediate operand: half-byte operand (when the last four command bits were interpreted as an operand), byte, 2-byte and 4-byte. Operands were represented by a hex, single byte, 2 bytes, or 4 bytes following the command code.

The command system was built in such a way that the most frequent operations had a shorter code, which increased the speed and reduced the code size. For instance, stack loading of the expression of the 0,1,...,15 numbers was executed by the LI0, LI1, ... , LI15, which were “load immediate” commands that together with the operand took one byte of the code. Similarly, the stack loading of the number 153 required the LIB, which was the “load immediate byte” command that together with the operand required two bytes of the code.

Procedures were represented with their own number from the 0...255 range. The code segment contained a “procedure displacements table” relative to procedures start. Therefore, the procedure call commands also had a length of one (for the procedures with 0...15 numbers) or two bytes. The mechanism of transferring parameters to the procedures varied. If a procedure had fewer than seven parameters, they were transferred via an arithmetic stack that reduced the costs of parameters processing in the call point. All arithmetic and logical operations worked on one or two upper stack elements and put the result on the top of the stack after popping up arguments.

The process descriptor at the beginning of the P-stack contains a snapshot of all registers at the moment of the last switching to/from a process. There is a process switching command that saves the current state of registers in the descriptor of a suspended process and restores the state of the registers of a resuming process, switching to it. This is similar to co-routine switching. It is very powerful way of hardware support for multi-tasking and synchronization. All interrupts were processed as process switching, indexed by interrupt number at the beginning of memory known as interrupt table.

4 Kronos 2.X Processor Family

The Kronos 2.X Processor Family included three models: 2.2, 2.5, and 2.6. All processors implemented the same system of commands. Processors differentiated by hardware implementation, cpu clock speed, and minor external devices variations.

The logic of functioning of all processor units was implemented in the micro-program control block. Two data buses connected arithmetic and logic unit, registers block, fast hardware seven words A-stack, command decoder, and communication buffer with the in-out bus. The two-bus internal processor structure enabled to execute binary operations on the stack (arithmetic, logical and other operations) in a single CPU cycle, which considerably increased processor efficiency. Micro-program control simplified processors and allowed implementation of complex commands. All processor hardware implementation is based on the Russian TTL (ТТЛ) and TTLSh (ТТЛШ) medium-level integrated components of widely adapted 155, 555, 531, 1802, 589, and 556 series.

The Kronos 2.2 was the first implementation of the above architecture concept. The processor was produced as a plug in board for "Elektronika-60" computer (similar to DEC PDP-11). The Kronos 2.2 was fully compatible with all devices that supported data communication protocol on Q-bus 22. The processor operated with 32-bit words. As the bus was 16-bits wide, the access to a word in memory required two Q-bus 22 cycles. The arithmetic and logic unit (ALU) for the Kronos 2.2 was 20-bit. Address arithmetic was processed in a single cycle, and data in two cycles. Such inner organization resulted in placing the whole processor on a single board using the element base of the time. The volume of directly addressed RAM of the processor was up to 4 MB; the clock rate was 4 MHz; the performance was 600 thousands operations per second on the stack arguments.

Unlike the 2.2 model, the Kronos 2.5 was fully 32-bits processor. It was produced in form of two boards. The interface with external devices was through the Multibus-1. Kronos 2.5 has up to 2 MB of local on-board memory. The CPU clock rate was 3 MHz; the performance was one million basic stack operations per second.

The Kronos 2.6 represented a further advancement of the 2.5 processor. It had an implementation and flexible reconfiguration designed to enable variety of usages. The processor could be embedded into a separate mini-computer as well as into a multi-processor complex.

The Kronos 2.6 was compatible with Euromekhanika bus and device specfication. The board size was 233.3 x 220 mm (E2). The basic set included the processing section board (ALU, stack, registers), a micro-program control board, a local memory board (min 512KB upto 2 MB), and the adapter board for the in/out bus. All devices were communicating via local synchronized 32-bit bus. The processor did not depend on a particular in/out bus. Compatibility with I/O bus was based on the specific set of external bus adaptors. Additional memory, inter-processor communication adaptor, local network controller, storage device on magnetic disks, code memory board (for separating code and data), bitmap-display, arithmetic calculator and other devices that expand processor capabilities has been implemented as a separate components on the local bus. The clock rate of the processor was 3 MHz; the performance was 1.5 million operations per second on stack operands.

5 Excelsior Operating System

Computers with the Kronos 2.2, 2.5, or 2.6 processors were controlled by the operating system Excelsior (OS Excelsior) that was designed as a general purpose operating system. In designing Excelsior OS, developers followed the basic principles of openness system interface, scalability, modularity, ease of integration, and user-friendly interface.

5.1 Open System Interface and Scalability

The OS design required to provide ways to accommodate wide variety of hardware configurations and allow the system to have only the components needed for specific usage scenario. Modular design of the OS and dynamic linking of all the components at run time accomplished it. The set of components presented at the particular installation of OS depended on the specific of the application.

5.2 Modularity

The qualities of Modula-2 programming language were the corner stone of the software modularity. In turn, software modularity created a basis for dynamic loading. Dynamic loading eliminated the stage static linking from the “edit - compile – launch” cycle and, therefore, considerably improved productivity of software engineers. Dynamic loading also meant there was no need to keep multiple copies of the same code – e.g. the frequently used modules, for instance, libraries. The absence of code redundancy, in its turn, saved external storage (in comparison with “linking” to the standard libraries in traditional systems).

5.3 Ease of Integration

Ease of integration was an attribute of all software tools of OS Excelsior that were built as a modular stratified hierarchical system. Practical application of Kronos computers as programmer’s workstations in various projects showed that in most cases the developed software was not only independent software products but also sets of libraries that implemented major project abstractions. Those libraries were suitable for use with other software; they continually expanded the “construction kit” that Kronos programmers used for creating new systems.

5.4 User-Friendly Interface

Excelsior OS implemented user-friendly interface via well developed command line shell and binary file manipulation utilities. Kronos OS also included software

engineer friendly screen text editor integrated with compiler and underlying shell execution environment.

6 Conclusion

Excelsior OS was based on Object Oriented ideas and abstract data types. Specific of devices were encapsulated in various implementations of driver modules that supported interface for all logical instances with a class of devices. Therefore, the system as well as application software communicated with devices only via well-specified logical interface.

Finally, it should be noted that all Excelsior OS software tools were oriented towards multi-process, multi-task, and multi-user applications. Due to its modular architecture, the Kronos processor addressed a wide range of applications, from embedded real-time systems to super-mini-computers. The most remarkable attributes of quality and programmability made Kronos indispensable in applications that required supporting continuous software development process.

7 Afterword

In September and October of 2005, Niklaus Wirth visited Russia. On September 21 at Moscow's Polytechnic Museum, Prof. Wirth has met Kronos research group (Vladimir and Marina Philippov, and Aleksey Nedorya). On behalf of the Institute of Informatics Systems SB RAN, V. Philippov presented to the Polytechnic Museum a functioning Kronos-2.6WS workstation. This workstation has been granted by Applied Mechanics Research and Production Association named after M.F. Reshetnyev (Krasnoyarsk-26 or, currently, Zheleznogorsk), where it was used in the 1990s as a space satellite software development workstation.

References

1. Kuznetsov, D., A. Nedorya, A. Osipov, E. Tarassov. Processor Kronos in a multiprocessor system. // Vychislitel'nye sistemy i programmnoe obespechenie. Comp. Center SB AS USSR, Novosibirsk, 1986, pp. 13-19. (In Russian)
2. Kuznetsov, D., A. Nedorya. Design of symbol tables for languages with complicated visibility rules // Metody translyatsii i konstruirovaniya programm. Comp. Center SB AS USSR, Novosibirsk, 1984, pp. 153-158. (In Russian)
3. Kuznetsov, D., A. Nedorya. Symbol files as the operating system interface. // Informatika. Tekhnologicheskie aspekty. Comp. Center SB AS USSR, Novosibirsk, 1987, pp. 68-75. (In Russian)
4. Kuznetsov, D., A. Nedorya, E. Tarassov, V. Philippov. Kronos, a workstation for a professional programmer. // Avtomatizirovannoye rabochee mesto programmista. Novosibirsk, 1988, (In Russian)
5. Kuznetsov, D., A. Nedorya, E. Tarassov, V. Philippov. Kronos: high-level languages processor family. // Mikroprocessornye sredstva i sistemy, 1989. (In Russian)

6. Vasyokin, A., D. Kuznetsov, A. Nedorya, E. Tarassov. Kronos: hard- and software support for high-level programming languages. Prelim. Proc. of All-Russian methodological conference on using computers in student studies and research, Novosibirsk, 1986, pp. 92-94. (In Russian)
7. Wirth, N., *Programming in Modula-2*, Third, Corrected Edition. Springer-Verlag. Berlin, Heidelberg, New York, 1985.
8. Wirth, N., *A Fast and Compact Compiler for Modula-2*, ETH, Zurich, 1985.