

## **CHAPTER 4. INSTITUTE OF PRECISION MECHANICS AND COMPUTER TECHNOLOGY**

### **4.1 Introduction**

In this chapter we examine developments at the Institute of Precision Mechanics and Computer Technology (ITMVT), primarily within the division devoted to the El'brus multiprocessors. ITMVT has been without question the most active and prominent R&D facility for Soviet high-performance computing for four decades. Pursuing the goal of developing the fastest machines possible, ITMVT has played a decisive role in shaping the high-performance computing landscape, and the broader computing industry as well.

The El'brus computers have been the most powerful systems manufactured in the Soviet Union and have been principal tools of that country's most demanding computer users. The third generation of this family is now nearing completion. In tracing the development of this series, we learn much about the factors—technical, social, political, and economic—which have shaped their evolution in the Soviet context. These systems give us considerable insight into the process of developing large-scale advanced technologies within the Soviet system. We will examine the impact of the reform-related changes that have had on the technology and the organizational context within which they were developed, and discuss the prospects for large-scale development in the future.

### **4.2 History of ITMVT Research**

The Institute of Precision Mechanics and Computer Technology was established in 1948 in Moscow under the USSR Academy of Sciences through the merger of the Department of Precision Mechanics of the Institute of Machine Studies, the Electric Simulation Laboratory of the Institute of Energy, and the Department of Approximate Calcula-

tions of the Institute of Mathematics [Golo88, 24; Itmv90; Crow93]. The first director, from 1948-1950 was N. G. Bruyevich.

The Institute's role in Soviet digital computing began in 1950 when academician M. A. Lavrent'yev became director and established a laboratory for the development of electronic digital computers [Bard87; Bard88]. He invited S. A. Lebedev, the father of Soviet digital computing, to Moscow as its head. Lebedev had built the Soviet Union's first electronic, digital, stored-program computer, the MESM, in Kiev between 1947-1951 at the Institute of Electrical Engineering. He had had the strong support of Lavrent'yev who was the vice-president of the Ukrainian Academy of Sciences from 1945-1948 [Itmv90; Crow93].

We discussed the contributions of S. A. Lebedev in chapter 3. Throughout the 1960s and 1970s he established ITMVT as the leading R&D facility for Soviet high-performance computing and cultivated a strong tradition of building fast machines with original architecture for use in real-world applications. He pioneered advances in all aspects of computing, and initiated a tradition of working as closely as possible with industrial design bureaus to shorten development cycles and ease the transition to series production.

Besides these computers, ITMVT workers built a number of special-purpose systems, primarily for the military, about which almost nothing has been written in the open literature.<sup>1</sup>

During the early 1970s, following the introduction of the CDC 7600, Soviet leadership became increasingly concerned about the "computer gap" with the West in the area of high-performance computing. Heated discussions on the topic were held at the highest levels of the Military-Industrial Commission (VPK), the Academy of Sciences, and the

---

<sup>1</sup>These include the 5E92B (1964), 5E51 (1967), 5E65 (1968), 5E67 (1970), 5E26 (1974) [Laza91b].

ministries involved in computing. These discussions coincided with those about who should succeed S. A. Lebedev as the director of ITMVT, who stepped down in 1973 and died in 1974 [Glus78b; Bard87]. The leading candidates for the position were V. A. Mel'nikov, the chief engineer of the successful BESM-6, and V. S. Burtsev, the chief engineer for some real-time special-purpose machines for the military. Mel'nikov was proposing the BESM-10, a general-purpose multiprocessor successor to the BESM-6 built with integrated circuits. Burtsev proposed the El'brus, named after the highest mountain in Europe. This family is discussed at length below.

Both machines could not be supported. Financing could have been allocated easily by government decree, but manufacturing capacity was more difficult to create. At that time Minradioprom did not have sufficient facilities to manufacture both. The Moscow SAM Plant was busy with mass production of the BESM-6 and anticipated production of the AS-6. A number of other major Minradioprom factories, such as those in Minsk, Penza and Kazan', were being converted to the manufacture of ES mainframes. The Zagorsk Electro-Mechanical Factory (ZEMZ) was to be the primary facility for the next generation ITMVT machine, but it did not have the capacity to carry out two major projects at once.

The discussions and maneuvering for support deeply divided ITMVT, the user communities for ITMVT machines, and the Soviet industrial leadership. Burtsev was selected to replace Lebedev because he proved more successful in establishing a base of support. Mel'nikov had the support of a large portion of the BESM-6 user community, in the Academy of Sciences and among certain groups of military users such as nuclear weapons designers. This included the Father of the Soviet space program, Academician M. V. Kel'dysh who was the head of the Institute of Applied Mathematics. These users had an investment in code which ran well on the BESM-6 that they wanted to preserve.

Furthermore, many potential users were alarmed at the complexity of the machine Burtsev was proposing. Burtsev, on the other hand, was able to enlist the support of the influential Academician G. I. Marchuk as well as large segments of the military user community, such as the rocket-builders who were most interested in real-time applications. Burtsev also had the all-important support of the Minister and Deputy-Minister of Minradioprom, V. D. Kalmykov and V. S. Semenikhin [Grea83; Kalm72].

When Lebedev resigned his position in 1973, a year before his death, Burtsev was appointed director, ensuring that the El'brus program would live. Shortly thereafter, Mel'nikov left ITMVT and founded a new organization, the Delta Scientific Production Association in the Ministry of the Electronics Industry, taking a number of the BESM-6 systems programmers like V. P. Ivannikov with him. Other BESM-6 engineers, mainly those who were working on the AS-6, remained at ITMVT.

### 4.3 El'brus-1 and El'brus-2

#### 4.3.1 Requirements

To a large extent, the requirements which most strongly shaped the El'brus design were dictated by military applications. Most of the customers for ITMVT machines, and virtually all customers for the El'brus machines, were in the military-industrial sector.<sup>2</sup> Their applications included controlling space missions, operating anti-ballistic missile and real-time radar installations, and running atomic energy stations. The El'brus were also designed for large-scale, computationally intensive scientific applications including weapons design. These applications demanded above all high performance, and high reli-

---

<sup>2</sup>In 1991, G. G. Ryabov stated that over 80% of the customers for [all] ITMVT machines (not just the El'brus) were in the military industrial complex [Laza91b]. No figures are available for the early 1970s, but the percentage was undoubtedly quite high then as well.

ability. The latter was necessary both in real-time applications where system failure could cost lives, and in scientific computation where a system that crashes midway through a lengthy computation is not useful. Reliability requirements were to have a particularly strong influence on the design, since designers had to incorporate many fault-tolerant features into the architecture to compensate for unreliable components provided by the Ministry of the Electronics Industry (Minelektronprom).

The principal customers for the EI'brus also had a strong need to be able to develop real-time software quickly and effectively. Ease and efficiency of programming became a third requirement which powerfully shaped the EI'brus hardware and software design. Finally, the EI'brus was also to serve at the heart of large-scale, distributed data processing centers. Extensive I/O and data transmission facilities were important.

In a 1975 article, Burtsev summed up the major trends in computer development which were incorporated into the EI'brus [Burt75]:

- high reliability;
- ease of software development;
- inter-generational software compatibility;
- increased main and peripheral storage;
- the ability to link geographically distributed equipment into a central data processing center.

In particular, he noted the trend in computer languages away from physical addresses, making software development easier. At the systems level this leads to the concept of virtual memory. Within processors, this principle can be reflected in not using explicit addresses for registers but dynamic allocation. As we shall see, this concept has significant consequences for the EI'brus.

#### 4.3.2 Design Antecedents

The ideas for how to design a machine to meet these requirements came from a variety of sources. An early ITMVT machine called the 5E92B, developed in 1964, was able to detect and correct all single-bit errors in hardware. It established a precedent for placing significant control in hardware [Laza91]. The principle of hardware control was taken to new heights in the El'brus-1 and -2. Not only were fault-tolerant features incorporated into the hardware, but also support for operating systems and high-level languages, and advanced instruction scheduling features as well.

Although some of the other ideas implemented in the El'brus—modular architecture, machine support for high-level languages, tagged architecture, etc.—had been implemented to some degree in earlier Soviet machines, one of the strongest influences on the thinking of the El'brus designers came from the writing of J. K. Iliffe and the Burroughs 500 and 700 Systems which incorporated many of Iliffe's ideas.<sup>3</sup>

In [Ilif68], Iliffe defines a computer system from a programmer's point of view. Rather than view a system as a linear store of both instructions and data (classical von Neumann model), Iliffe sought to define a non-linear system structure which reflected the structure of programs in a multiprogramming environment. He proposed using a tree structure for program and data segments and a hierarchy of processes. Such an arrangement would support dynamic control over data structures and processes, and support interacting, parallel processes [Ilif68, 1-15]. To further simplify programming and enhance the ability of the system itself to monitor processes, Iliffe proposed incorporating a means of indicating, at the hardware level, the type of individual data elements [Ilif68, 11-12]. If the machine could distinguish between addresses and instructions, and dynamically interpret operands, the number of instruction/data type combinations specified in the in-

---

<sup>3</sup>Iliffe developed and helped implement many of his ideas before the publication of [Ilif68].

struction set could be reduced significantly, and programmers could be insulated from local optimization of the address and instruction codes [Ilif68, 33]. To implement data typing in hardware, Iliffe proposed using hardware tags [Ilif68, 34-35].

In 1961, Burroughs introduced the first commercial computer using a tree structure, the B 5000 [Ilif68, 25; Orga73, vii]. This was one of the first machines to incorporate a push-down stack for operands. A stack readily lends itself to representing the structure of block-structured, procedure-oriented languages—of the Algol family, for example—which are characterized by nested blocks that define the scope of an algorithm’s variables and identifiers, and allocation and deallocation of dynamic resources. A stack is a useful structure for representing the execution state of structured programs. A program block’s data and instructions can be pushed onto the stack as the block is entered, and popped off when control exits the block. The stack reflects the context of the active block.

The B 5000 was followed by the B 5500 and B 6500 in 1969 and, in 1970, Burroughs introduced the B 5700, B 6700, and B 7700 mainframes. The latter systems refined and expanded the ideas pioneered in the B 5000. Key design goals for the Burroughs machines, as for the El’brus, were reliability, high speed, and ease of programming. Some of the key design features used to achieve these goals were:

- a modular structure consisting of multiple CPU, memory, I/O, and data transmission modules which were treated as shared resources in a single, integrated system;
- multiprocessing and multiprogramming as a normal mode of operation;
- dynamic allocation of system resources, including CPUs, memory, I/O and data transmission processors;
- hardware oriented towards efficient compilation and execution of programs written in high-level languages;

- stack-based complex instruction set computer (CISC) central processing units;
- hardware tags;
- software compatibility between machine generations.

Each of these principles and many implementation details were adopted by the El'brus developers. In implementing them, El'brus designers were guided by a basic philosophy that the machine should have an integrated hardware/software design, as had been advocated by Iliffe. Consequently, they felt that as much control as possible should be implemented in hardware to simplify programming of systems and applications software, and all programming without exception should be done in a high-level language. During the design phase Burtsev, Babayan and their coworkers examined several Western systems (including the CDC 6600, the MU5 project at Manchester University, and the Multics operating system) to see which would best support such an integrated design. They felt that the Burroughs machines offered the best opportunity.

The philosophy of an integrated design was held to a non-trivial degree by the Burroughs designers, but as we shall see, the El'brus designers carried them to new levels. In particular, the El'brus designers decided to implement a single language, later called El'-76, in place of multiple languages oriented towards separate programming functions such as systems programming, applications programming, job control, etc. Another basic difference was the implementation of a more advanced form of virtual memory, described below. These decisions led to the development of an instruction set, programming language, and compilers which differ significantly from that of the Burroughs machines.

Two core design principles of the El'brus which were hardly implemented in the Burroughs machines were achieving high performance through:

- multiple functional units;

- dynamic instructions scheduling performed by hardware.

### 4.3.3 Burroughs/El'brus Comparison

In this section we describe the architecture of the El'brus computers and compare and contrast it with that of the Burroughs B 6700.<sup>4</sup> A detailed description of the B 6700 and El'brus is beyond the scope of this study; interested readers can find more information in [Orga73; Burr72; Zamo85; Baba90]. We focus on some of the principal features of the machines, highlighting points of similarity and difference on both the conceptual and implementation levels. Such an analysis will reveal the degree to which the Soviets adopted not only specific architectural features, but also important guiding principles which were to shape the development of this family in future years, even after the Burroughs architecture had been largely discarded. We will also be able to determine how advanced the Soviet work was, relative to the world-wide state-of-the-art.

While the El'brus and Burroughs machines have considerable similarities to one another, they have both quantitative and qualitative differences. Quantitative differences are those in which the essential nature of an architectural feature is the same in two systems, but the degree of implementation (number of units, volume, size, etc.) differs. Quantitative changes in one part of a system frequently necessitate qualitative changes in another part of the system. There are a number of examples of such patterns in the El'brus vis-à-vis the Burroughs machines. The qualitative differences in a number of cases reflect indigenous innovations and Soviet contributions to the computing field.

---

<sup>4</sup>In this section we limit our discussion to the El'brus-1 and El'brus-2 which have essentially the same architectures. Unless otherwise stated, comments referring to the El'brus computers in this section refer to both.

#### 4.3.3.1 System Organization

Table 4-1 provides a comparison of the features of the Burroughs 6700 and El'brus computers which differ quantitatively. Like the Burroughs machine, the El'brus computers consisted of multiple central processing units linked by a crossbar switch to shared main memory modules. Independent I/O processors have direct access to the memory modules as well, and relieve the CPUs of much I/O overhead. Data Transmission Processors linked to the I/O processors are also independent digital computers responsible for interfacing with a wide variety of peripheral devices employing a wide array of telecommunications line disciplines. The architecture of the El'brus computers is shown in Figure 4-1.

#### 4.3.3.2 CPU

The central processing units of the B 6700 and the El'brus computers are stack-based, using a zero address CISC instruction set and reverse Polish notation. By using a stack-based architecture and a number of other features, Burroughs and El'brus designers developed machines whose hardware reflected the structure of the software sufficiently well that most, if not all, code for the system could be written in a high-level language.

The object code of a compiled program consists of a set of segments. A segment generally corresponds to a single procedure or block in the source code. When program execution is started, two portions of memory are allocated: one for the stack and another for the segment dictionary which is used to reference the multiple program segments. Each entry in the segment dictionary points to a single segment indicating whether or not that segment is located in main memory.

The stack structure consists of procedure activation records. Each record contains memory allocated for a procedure's variables and descriptors, pointing to a data structure

	<b>El'brus-1</b>	<b>El'brus-2</b>	<b>B 6700</b>
<b>Word Length</b> (bits)	64+8	64+8	48+4
<b>Clock Period</b> (nsec)	260	47	200
<b>CPUs</b>	1-10	1-10	1-3
<b>Memory Modules</b>	4-32	4-32	1-32
Size of Module	256 Kbytes	4.5 Mbytes	96 Kbytes (16K words) 384 Kbytes (64K words)
Min main memory	1 Mbyte	18 Mbytes	384 Kbytes
Max main memory	8 Mbytes	144 Mbytes	6 Mbytes
Max memory exchange rate per processor	20.7 Mbytes/s	180 Mbytes/s	183 Mbytes/s
<b>I/O Processors</b>	1-4	1-4	1-3
Max peripherals per I/O processor	256	256	128
Max peripherals in configuration	1024	1024	256
Max throughput per I/O processor	3.6 Mbytes/s	30 Mbytes/s	1.67 Mbytes/s
<b>Data Transmission Processors</b>	1-16	1-16	1-12
Max communications lines serviced in system	2560	2560	2048
<b>Performance</b> (nsec, (cycles))			
Addition (single precision)	520 (2)	141 (3)	200 (1)
Multiplication (s. precision)	1300 (5)	235 (5)	2000 (10)
Division (single precision)		1081 (23)	10800 (54)

Table 4-1 Comparison of El'brus and B 6700 System Characteristics  
Sources:[Dpro71; Burr72; Dpro77; Golo80; Mvke80;  
Timo81; Zamo85; Baba90]

such as an array. Memory portions for code segments and arrays are allocated by the operating system on demand.

Descriptors provide a level of addressing indirection which facilitates access to the same data or code structure by multiple tasks, re-enterability of code, and an economical use of stack storage. Both the Burroughs and El'brus computers were designed as multi-

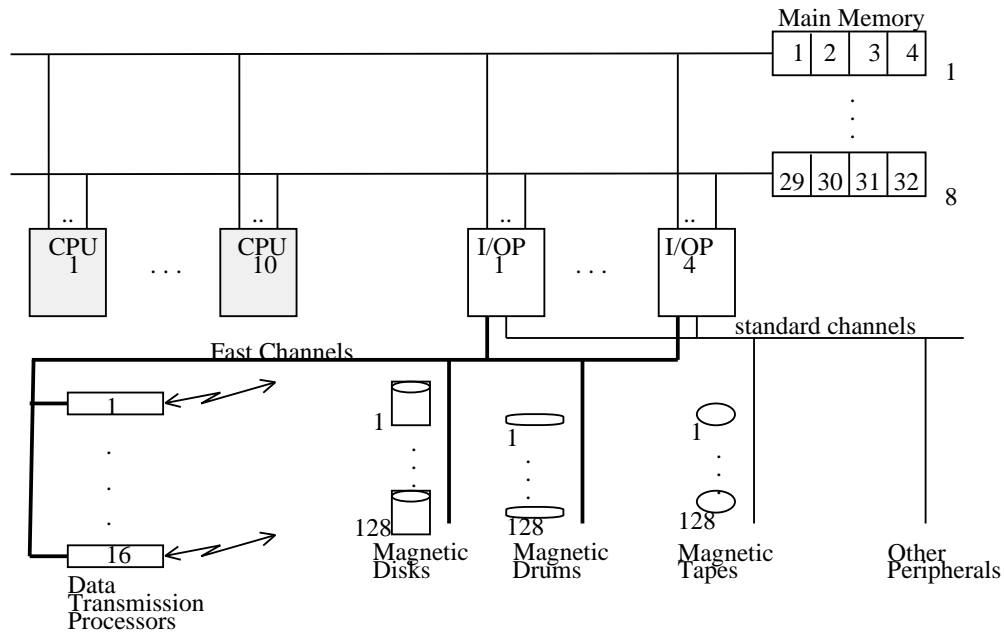


Figure 4-1 El'bus Structure  
Source: [Baba90, 46]

processing, multitasking systems. Separating the code portion from the data portion makes it possible for multiple tasks to access and execute identical code, but each with its own data. Each task therefore has its own data stack and segment dictionary stack, but code can easily be shared among tasks.

The B 6700 and El'bus machines use very similar stack disciplines. The above description applies to both. A more detailed examination reveals that both use comparable special-purpose registers and control words to manage the stack. For example, each system has base-of-stack, stack limit, and top-of-stack registers. Mark stack control words placed at the start of the addressing space for each entered procedure are linked together to provide a dynamic history of procedure entry. Thirty-two display registers point to the mark stack control words, collectively defining the blocks whose addressing spaces are

global to the procedure (block) currently being entered. A return control word points to the calling procedure, indicating a return address when a block is exited.

While the Burroughs and El'brus computers have very similar approaches in their treatment of tasks and stacks, they differ a great deal in CPU construction and the underlying dynamics of execution. The B 6700 CPU consists of a 48-bit adder (arithmetic unit), an address processing unit, seven functional controllers (program, arithmetic, string, stack adjust, interrupt, transfer, and memory), and register sets. The latter include four 51-bit data registers which hold the top two stack elements (single or double precision), one current program instruction word register, one scratch register, and 48 20-bit registers which include the 32 display registers plus eight base and eight index registers [Burr72, 5-1:3]. In addition to these functional resources, the CPU has ten so-called Operator Family Controllers which group related operators into families to minimize the logic required in the processor. The families include arithmetic operators, logical operators, subroutine operators, and others [Burr72, 5-1].

The Program Controller controls the program flow. It controls the transfer of an instruction word to the Current Program Instruction Word register, decodes the syllable to be executed, and selects the appropriate Operator Family Controller to execute the instruction. A key feature is that instructions are executed sequentially in the order dictated by the compiler. There can be no over-lapped execution of arithmetic instructions because the CPU has only one adder.

In designing their CPU, the El'brus developers sought to increase performance by increasing the amount of parallelism within the CPU. The El'brus machines have ten functional units: an adder, a multiplier, a divider, a logic unit, a decimal-coded conversion unit, an operand call unit, an operand write unit, a string processing unit, a subroutine execution unit, and an indexation unit. There is some functional overlap between these

units and the controllers of the B 6700, but important differences exist. In particular, the El'brus has four times as many arithmetic units. Multiple arithmetic units had been implemented in other machines—notably the CDC 6600 in 1964 which had eight arithmetic functional units—but not in a stack-based architecture.

Stack-based architectures are not well suited to feed multiple, parallel functional units. Zero address operations assume that the necessary operands are located at the top of the stack. Clearly, only one zero address operation at a time can access operands correctly, essentially precluding the simultaneous operation of multiple functional units. In the B 6700, the stack-based approach worked well because instructions were executed sequentially; in practice, this meant that while one controller was executing an instruction the others remained idle.

The challenge for the El'brus designers was how to achieve efficient, parallel operation of the functional units within the context of a stack-based architecture. The key features of the solution to this problem were a) an internal, register-based (non-stack based) representation of the top of the stack, and b) dynamic, sequential/parallel scheduling of instructions [Baba90, 63-64, 73-75].

A so-called instruction block decodes instructions sequentially, in the order specified in the object code generated by the compiler, and places them in an instruction buffer. Rather than feeding these instructions directly into the functional units, however, the control unit converts the instructions from a zero address representation to an internal, addressed, register format. The instructions now incorporate explicit references to registers which will contain the necessary operands. While the B 6700 used two registers to store the two top elements of the stack, the El'brus computers have a 32-word buffer for the top 32 stack elements. The conversion from zero address to register-based instructions permits access to other than the top elements in the stack.

The control unit issues the instructions to the appropriate functional units at a maximum rate of two instructions per clock cycle.

A unique feature of the El'brus is that the instructions can be issued to the functional units before all the necessary operands are available. Hardware controls indicate the availability of operands, and the functional units simply wait until all operands are available before executing instructions. In effect, execution takes place in a data-flow manner with the exact order of execution depending on the order in which operands become available.

To the compiler, the El'brus looks in every way like a stack-based machine, even though the underlying implementation is not a stack.

The El'brus designers placed considerable value on ease of software development and placed a great deal of control in the hardware. In particular, all scheduling of functional units and all decisions about parallel execution of instructions are handled by the hardware. In contemporary terminology, this is called a superscalar approach to instruction-level parallelism. A distinguishing feature of a superscalar processor is that it is presented with a sequential program and tries to execute as many instructions as possible in parallel, with all scheduling decisions being handled in hardware [Fish91, 1236-1237]. Although some of the underlying ideas have been developed since the 1960s (the IBM System/360 Model 91, for example), superscalar approaches are currently very much at the forefront of RISC processor development.

A further difference between the Burroughs and El'brus approaches is the treatment of arrays. In the B 6700 all elements of arrays are accessed (indirectly) by indexing through an array descriptor. Because of this extra memory access, an extra memory cycle is often unavoidable [Orga73, 85]. There is no special hardware support for vectors [Orga73, 91].

In the El'brus, the hardware is designed to detect operations on arrays (vectors) and provide pre-fetching of array elements into local cache memory [Pent82, 57; Baba90, 56]. In the index unit there is associative memory which stores the address of the current element together with the step in memory. Only the first element must be accessed through the array descriptor; all others can be accessed directly. The associative memory can store information on up to six arrays, and the element address computation in a loop takes only one clock cycle [Baba90, 56]. Array elements for up to five loop iterations can be fetched in advance [Baba90, 14]. For this reason, implementation of vector operation in the El'brus is considerably more efficient than that in the B 6700.

The cache memory in each processor plays an important role not only in how the each El'brus CPU handles operations on arrays (vectors), but also in the multiprocessor operation of the system as a whole. The B 6700 had no cache, only a local set of special-purpose registers. In the El'brus machines, the cache consists of four distinct sections (size given for El'brus-2):

- instruction buffer (512 words) for storing instructions executed by the program. Subsequently, if instructions are executed multiple times, access time is shortened;
- stack buffer (256 words) for holding the most active (topmost) portion of the stack, which otherwise is stored in main memory;
- array buffer (256 words) for storing array elements which are processed in loops;
- associative memory for globals (1024 words) for data other than that stored in the other buffers. This includes global program variables, data descriptors, and local procedure data which does not fit in the stack buffer [Kriu80, 66-67; Baba90, 13-14].

This cache organization made it possible to incorporate a relatively large number of processors effectively into a shared-memory configuration. Cache memory basically contains copies of data or instructions stored in main memory, and in multiprocessor systems it is possible that multiple processors can each have a local copy of the same data element simultaneously. Some means are needed to make sure that all copies are identical. As the number of processors accessing main memory concurrently increases, the overhead required to maintain cache coherency can become prohibitive, limiting the number of processors which can be used effectively to a small number. In part for this reason, configurations of IBM multiprocessors such as the IBM 3033 and the IBM 3084 incorporated only up to four processors. The IBM dual-processor 3033 (introduced in 1978) used a simple store-through design in which data changed in the cache is immediately changed in main storage. The 3084 model (introduced in 1982) employed a more advanced, store-in, cache coherence scheme in which transfers to main memory could be delayed until cache elements were to be overwritten, or another processor accessed the corresponding data elements in main memory [Pras89, 217-218].

Cache coherence in the El'brus was maintained through the use of the segmented cache together with the notion of a "critical section" in a program. Portions of a program which access resources (data, files, peripherals) which are shared by multiple processors must be written by the programmer as critical sections in a manner which regulates simultaneous access. In the case of the El'brus, the programmer uses semaphores to synchronize access [Baba90, 17, 113-119]. A segmented cache and the use of critical sections make it possible to limit significantly the amount of overhead in achieving coherence. First, on the average, critical sections constitute only about 1% of the execution time of El'brus programs [Baba90, 17]. In other words, in the remaining 99% of the time, a given data element will not exist in more than one cache simultaneously and cache co-

herence is not a problem. Instructions in the instruction buffer are static, so copies in multiple caches will remain identical. In the El'brus, the array buffer will, as a rule, be empty during operations in critical sections. The only data elements for which incoherence is a serious problem are those contained in the associative memory for globals. Therefore, measures to reconcile the cache of multiple processors are taken only when absolutely necessary. This is one of the reasons the El'brus configuration can accommodate up to 10 processors.

Cache memory was not a new concept when the El'brus was designed. The IBM System/360 model 195 (announced in 1969) had 32 Kbytes of cache, for example. The El'brus represent one of the earlier examples of a segmented cache, however. Three of the buffer types used in the El'brus (stack buffer, instruction buffer, and associative memory buffer) were implemented on a more limited scale in the B 7700 (introduced in 1976), but the El'brus-1 design and much of the construction would have been completed by the time information about the B 7700 became available in Russia [Dpro77, 11c]. The El'brus cache coherence solution is significant because it is one of the earliest uses of a mechanism that would support a relatively large number (10) of processors in a shared memory configuration. The El'brus was one of the first general-purpose shared-memory system in the world with this number of processors to reach series production.<sup>5</sup>

#### 4.3.3.3 Tags

Like the Burroughs machines, the El'brus use hardware tags to enable hardware to identify specific types of data and instructions. The Burroughs machines used three-bit tags to identify single/double precision operands, data descriptors, and a number of con-

<sup>5</sup>Others which share this honor were some bus-connected minisupercomputers with multiple processors sharing multiple memory units which appeared during the early-mid 1980s. For example, the Sequent Balance 8000, with up to 12 processing elements was first delivered in 1984 [Hock88, 46]. This pre-dates the El'brus-2, but is later than the El'brus-1.

trol words. In their zeal to place as much control in hardware as possible, El'brus designers developed an elaborate set of hardware tags. Using six-bit tags, they were able to distinguish between half/single/double precision operands, whole/real numbers, empty/full words, labels (including such specialized labels as "privileged label without block of external interrupts" and "normal label without block of address information write"), semaphores, control words, and others [Zamo85, 129].

One major purpose of the tags was simplification of programming. If the functional units could distinguish between real and integer operands, they could be designed to adapt themselves to computation on either. There would be no need for separate scalar and floating-point units. The El'brus could in effect implement dynamic typing of data which is useful in, for example, building operating systems in which the precise type of the operands may not be known ahead of time. This capability was one of the reasons why the entire El'brus operating system could be written in a high-level language.<sup>6</sup>

Another purpose of the tags was error detection. Hardware could detect such errors as attempted arithmetic operations on control words, for example. Tags could also be used for memory protection, for restricting writes to specific types of data [Baba90, 11, 54-55].

Tags were not an invention of the El'brus designers. They had been a key element in Iliffe's Basic Machine [Ilif68] and the Burroughs machines which it inspired. The El'brus pushed these ideas to a new level of detail and complexity.

---

<sup>6</sup>Other hardware features played a role as well. One of the main problems was being able to use high-level languages for specific parts of the system software, such as memory allocation and process switching. To do this, code transparency and object code predictability are required. The El'brus uses special very-high-level hardware to accomplish this, e.g., process switching can be programmed as a sequence of assignment statements performing clearly defined actions on special hardware registers. The operating system mechanisms were defined first, and the hardware design and instruction set were tailored accordingly.

#### 4.3.3.4 Memory

As in the Burroughs machines, El'brus main memory is shared among all processors and is modular in design. The B 6700 main memory was initially implemented as 1-64 memory modules of 16K words (48+4 bits) each for a maximum memory size of 1024K words. Later versions of this machine incorporated a mix of 16K and 64K word modules, although the maximum memory size remained 1024K words [Dpro71, 11b; Burr72, 1-8; Dpro77, 11b]. Data could be interleaved across modules.

The El'brus contained significantly more memory. The El'brus-1 contains 4-32 modules of 256 Kbytes each [Zamo85, 145], while the El'brus-2 contains modules of 2M words (4.5 Mbytes) for a total of 16M words (144 Mbytes) [Baba90, 47,78]. The El'brus memory is organized as a hierarchy, with a memory section (stored in a single cabinet) consisting of four memory modules; each memory module consists of up to 32 blocks, each containing 16K words. Interleaving is possible at multiple levels: between sections, between modules within a section, and within the individual modules. Up to four words can be read from each memory module in one cycle. The maximum throughput per section in the El'brus-2 is 450 Mbytes/s, although the maximum data exchange rate with each processor is 180 Mbytes/s [Baba90, 78].

The memory management schemes of the B 6700 and El'brus are, at the general level, very similar. Both employ segmentation. Memory is organized into variable length segments which reflected the logical divisions of a program determined by the compiler. Corresponding to the logical division of a program, segments can be coded independently, given different levels of protection, and shared among processes.

In the B 6700, segments were moved between main and virtual storage as complete segments [Orga73, 90]. Arrays were an exception. They could be stored in main memory in groups of 256 words each, bounded on both ends by memory link words [Burr72, 5-7].

The El'brus treats program segments differently from data and array segments. The former are treated just as in the B 6700 and moved in and out of main memory in one piece. Data and arrays of constants, on the other hand, are organized into pages of 512 words each [Zamo85, 122]. This approach is similar to that of the B 6700, except the paging principle is applied more broadly, to data, as well as to arrays. In both cases, this allows the processor to handle data sets which exceed the amount of physical memory available to a process. However, the El'brus approach uses memory more efficiently and allows faster process swapping.<sup>7</sup>

The El'brus employ a more contemporary type of virtual memory. In the Burroughs computers, addressing was limited to 20 bits, or  $2^{20}$  words, the maximum size of physical memory in both the B 6700 and B 7700. Segments were moved between main and secondary storage and their presence in main memory was indicated by a "presence bit" in their descriptor which remained in main memory during the run of the process. There was no concept of a true virtual memory space which was larger than the total amount of physical memory; the descriptors contained only physical addresses [Burr72, 1-8; Orga73, 17].

The El'brus machines use a similar, 20-bit addressing scheme for program segments, but 32-bit addressing was used for segments of data and arrays of constants. This provided a virtual memory space of  $2^{32}$  bytes (4 Gigabytes). These segments were moved between virtual and physical storage using a paging mechanism which used paging tables stored in an associative page memory unit to convert between virtual and physical addresses. Virtual addresses consist of a page number and an offset within the page [Zamo85, 119, 122; Baba90, 66]

---

<sup>7</sup>In the B 6700 process swapping required scanning the process stack for all descriptors to arrays used by the current process.

#### 4.3.4 Reliability

Building computers for a variety of applications, including real-time, both the Burroughs and El'brus engineers placed high priority on system reliability. This was particularly true for the El'brus engineers who had to struggle with a notoriously unreliable component base. The modular design of both machines provided redundancy of all system resources, and the ability for each unit (CPU, memory module, I/O processor, etc.) to operate independently of the others. Within this modular framework, hardware and software mechanisms to increase reliability were implemented on a number of levels.

Both the Burroughs and El'brus computers incorporated fail-soft features in which systems routines regularly check systems modules and, when a module fails, automatically remove the troublesome device from operation and reassign its functions to other modules without operator intervention [Dpro77, 11e; Zamo85, 117; Baba90, 97]. In the El'brus computers there are actually two levels of recovery—"soft restart" and "hard restart". In the first, processes are interrupted and restarted on processors; in the second, a unit is logically removed from the configuration [Burt87, 20].

To improve the reliability of individual modules, the El'brus computers use Hamming code error detection and correction in memory modules and re-read or re-write to memory [Burt87, 19; Baba90, 80]. Automatic instruction retry (up to 16 times) is used in the CPU. The B 6700 did not have the latter feature, which was introduced in the B 7700 [Dpro77, 11m]. In spite of such features, the reliability of individual modules remained low in comparison with machines such as the Cray which had a mean time between failure of hundreds or thousands of hours. An official analysis of the El'brus-2 reliability made at the time of state testing of the El'brus-2 in 1985 gives the figures shown in Table 4-2.

Device	Mean Time Between Failures (hours)	Mean Time to Repair (hours)
CPU	92	0.6
Main Memory	1263	0.29
I/O Processor	565	0.3

Table 4-2 El'brus-2 Reliability Characteristics  
Sources:[Burt87, 18]

It is significant to note, however, that although the mean time to failure is low, especially for the CPUs, the mean time to repair is also well under an hour. As a rule, therefore, an El'brus configuration under constant surveillance by trained technicians could be kept running for long periods of time. As modules failed, they would be switched out by the system itself, quickly repaired by technicians, and switched back into operation. El'brus operation was, as a result, very labor intensive. El'brus computers at the most important installations (such as at the All-Union Scientific Research Institute of Experimental Physics (VNIIEF) in Arzamas-16) did have skilled on-site technicians who could keep the machine running nearly all the time. Installations which did not have a skilled maintenance crew could not expect to keep the system up for long periods of time.

If a CPU failed, a process was restarted from a checkpoint on another processor. The practical impact of this was that jobs which had long execution times between checkpoints (i.e. if extensive computations were carried out on a very large data set), the danger that the process would fail in the middle and have to be restarted was very real. According to one user, "running jobs on the machine was nerve-wracking."

One of the main reasons for the lack of reliability were the multi-chip modules (described below) which were used in the El'brus-2 processor from 1985-1989. When gate

arrays replaced multi-chip modules in the El'brus-2 construction in 1989, reliability improved dramatically. According to some reports, the reliability of the CPU increased to 240-500 hours mean time between failures.

#### 4.3.5 Performance

The performance figures given for the El'brus computers are nearly always the same: In full configurations, 12-15 MIPS for the El'brus-1, and 120-125 MIPS for the El'brus-2 [Tass78; Golo80; Mvke80; Ivan87; Baba90, 47]. These figures represent El'brus performance on a Gibson-3 mix (a mix of a variety of instructions used to measure performance on IBM and ES mainframe computers), rather than the theoretical peak performance [Baba90, 50]. According to V. S. Burtsev, the theoretical peak performance figures were never emphasized (or published in Russia, to our knowledge) because he personally did not believe in using peak performance figures to advertise his machines. Besides a personal reluctance to use "unrealistic" theoretical peak performance figures (an opinion voiced by many in the West), Burtsev was using a means of quantifying performance traditional for the ES series (and for IBM machines). The real competition to the El'brus computers came from the high-end ES mainframes which were the only real alternative for organizations seeking to acquire significant general-purpose computing power. Seeking a basis for comparison, policy makers perhaps preferred to have performance of the two families of machines based on the same kinds of tests. Both a single-processor El'brus-2 and an ES-1066 have official performances of 12.5 MIPS on a Gibson-3 mix [Dani84; Ecot85; Vdnk86]. In a direct test on a large physics problem, however, the single-processor El'brus-2 ran 2.5 times faster than the ES-1066 on 32-bit operands, and 2.8 times faster on 64-bit operands [Baba90, 15]. The theoretical peak performance (TPP) of the El'brus-2 was calculated independently and published in [Doro92]. Taking into account the number of clock cycles to compute results in each of the functional units which

could perform floating-point operations, the authors computed a theoretical peak performance of 9.4 Mflops per processor, or 94 Mflops for a 10-processor configuration Doro92[Doro92, 5].

A few other performance reports (not independently verified) have been published. In 1988, S. V. Kalin ran the 24 Livermore Fortran Kernels on a single El'brus-2 CPU and measured a harmonic mean<sup>8</sup> of 2.7 Mflops [Baba90, 50]. In comparison, Pfeiffer, et al. report a harmonic mean of 15.26 Mflops for the LFK on a single-processor Cray X-MP with a clock cycle of 9.5 nsec and a theoretical peak performance of 210 Mflops [Pfei90, 140].

These figures appear to point to a strength of the El'brus design: average performance on a variety of applications. Although the peak performance of the Cray X-MP processor is over 20 times that of the El'brus-2 CPU, the harmonic mean is only 5.7 times greater, a ratio only slightly larger than the ratio of the clock periods of the two machines. The El'brus-2 performance on nicely vectorizable problems is significantly lower than that of the Cray X-MP, but it performs rather well, relative to its clock period, when a variety of programs—not all of them vectorizable—are run.

In retrospect, however, the designers of the El'brus underestimated the importance of vector-pipelining for achieving high performance on vectorizable problems. If an El'brus processor were designed on vector-pipeline principles, producing two results (i.e., add and multiply as in a Cray) it would have had a theoretical peak performance of 42.5

---

<sup>8</sup>The harmonic mean (unweighted in this case) is computed as

$$\frac{1}{\binom{1}{I} \sum \frac{1}{R_i}}$$

where  $I$  = number of programs, and  $R_i$  = execution rate of program  $i$ . The harmonic mean, taking into account the proportion of a task completed at a given rate, gives a “truer” indication of average performance than an arithmetic mean [Worl84, 124-125].

Mflops. In a 10-processor configuration, the TPP would be 425 Mflops. A possible reason for the lack of vector-pipelining is that the basic design work on the machine was carried out before the impact of the Cray-1 introduction on supercomputer design was felt. It is also possible that the relatively good performance of the El'brus-2 on mixed tasks made the issue less pressing.

There are two basic bottlenecks in El'brus-2 performance. The first is the lack of pipelining in the functional units so that each functional unit uses 3+ cycles to generate a result. Second, the instruction issue mechanism is only able to issue two or fewer operations per cycle. Given the rate at which the functional units operate, this is not necessarily a bottleneck in real applications. It could be, if the functional units were pipelined.

#### 4.3.6 Differences Between the El'brus-1 and El'brus-2

The El'brus-1 and El'brus-2 are virtually identical in their basic design. They differ by an order of magnitude in their Gibson-3 (not theoretical peak) performance, however. The primary reason for this was the difference in the component technology. First, the El'brus-1 uses TTL technology, while the El'brus-2 uses ECL components. The clock period could therefore be decreased by a factor of more than five. This alone does not explain the factor of ten difference in performance. Performance also depended on the rate at which data could be moved from main memory to the CPUs. While the El'brus-1 used ferrite core memory, the El'brus-2 uses semiconductor memory [Golo80; Mvke80; Baba90, 78]. As a result, the data throughput from main memory to each processor was nine times greater in the El'brus-2, making it a more balanced system.

Another difference was the treatment of arrays. As has been mentioned, the El'brus-2 has a hardware mechanism to prefetch array elements, enabling it to perform considerably better on vector computations than the El'brus-1.

As a result of these and other changes, the instruction sets of the El'brus-1 and El'brus-2 were slightly different. In computers where much systems software is coded in assembler, differences in the instruction set typically mean that machines are not software compatible. Porting code from one machine to another requires not only recompilation, but modification of the source code. The decision to code all El'brus software, including the operating system, in a high-level language meant that all software could be moved from one platform to another simply by recompiling code. No modifications were necessary. This feature, the lack of assembly-level code to "fix" the instruction set was to give the El'brus designers considerable freedom of movement in later years. It proved beneficial in the transition from the El'brus-1 to the El'brus-2, and would in later years make it possible to expand the El'brus line into a variety of architectural approaches (mentioned below, and in chapter 3) which bore little resemblance to stack-based machines.

#### **4.4 The El'brus in the Soviet context**

##### **4.4.1 The Long Road from Conception to Production**

The El'brus project was begun around 1970. Most of the design work was done between 1970 and 1973, when the draft design (*ekskiznyy proyekt*) was completed [Pent82, 10]. During these years, designers had access to information on the B 6700, but only about the instruction set and the system structure block diagrams. During 1975-1976 they obtained more detailed functional descriptions of the Burroughs which led to some modifications of the design of the hardware and programming language. In 1977, a B 6700 was sold to the oil industry and ITMVT designers were able to examine it in detail. Because it had never been a tradition at ITMVT to develop functional duplicates of Western machines (à la the ES program) and much information about the B 6700 hardware was lacking at crucial design stages, the El'brus hardware was designed from scratch.

The first El'brus-1 prototype became operational around the end of 1977, and the first parts of the operating system were run on it at the end of 1978 [Golo80; Golo86]. It underwent state testing in 1979, and was accepted by a state commission in 1980 [Golo86, 87].

El'brus software was the topic of a conference held in Novosibirsk in 1976, and in 1978 the machine was touted in a prominent article in Pravda [Baba77; Burt78].

The El'brus-2 was developed in parallel with the El'brus-1, but on a very different component base. The draft design was completed in 1978. The first El'brus-2 prototypes were running in 1984 and a two-processor version underwent state testing in 1985. In February, 1986 a 10-processor unit was brought on-line. Full series production began in 1987 and continued through the end of 1992.

The El'brus computers had a long and painful birth. Shortly after the Pravda announcement, the Estonian Academy of Sciences announced that it would receive an El'brus-1 by the end of 1980 to be used in a collective-use computer center to service a number of Academy institutes [Vyrk78]. The Institute of Cybernetics was to house the machine and develop a time-sharing system to give other institutes access [Vyrk79]. These estimates proved to be wildly optimistic. By the end of 1981, some, but not all, portions of the El'brus-1 had arrived in Tallinn [Favo81; Gudi82]. By 1982, the complete machine was expected by 1984 [Aben82]. A dual-processor machine was finally running in 1986, thanks to the purchase of a cooling station from the Finns. Once installed, it was equipped only with about a dozen disk drives and drums (with a total capacity of about 70+ Mbytes (!)), had low reliability (especially when multiple user jobs were running) and was used rather infrequently.

While there were rumors that many other customers for the El'brus-1 had their orders canceled without explanation [Harv83] and that the El'brus program as a whole was in

trouble, the Estonian machine was perhaps an extreme case. Primary El'brus customers with good maintenance support report using the El'brus-1 with some success. The Estonian experience did nothing to help the reputation of the El'brus program, however. According to G. G. Ryabov, this machine was shipped to Tallinn "practically untested" and without the equipment and support necessary to get it running quickly. This bad experience, coupled with the long delays in getting the machine into production and the general lack of involvement of the broader Academic computing community gave the program a bad reputation. Although the machine was praised publicly, in closed discussions the machine was criticized by the Academic and industrial communities. Growing criticism over the progress being made, complicated by personality conflicts within ITMVT and disagreements over the future course of the program, brought matters to a head in 1984. Burtsev was removed as director of ITMVT and replaced by G. G. Ryabov.

Burtsev's removal did not necessarily dampen the criticism of the machine. Only during the late 1980s were significant number of El'brus-2 processors manufactured. Between 1985 and 1989, approximately 100 El'brus-2 processors were reportedly manufactured. Of these, half were used in five 10-processor installations. By 1992, over 200 processors had been manufactured.

Even then the machine, which had been developed principally for military customers, was inaccessible to large segments of the traditional (BESM-6) user community. Nearly all installations were highly restricted. These included weapons designers at VNIIEF in the closed city of Arzamas-16, Mission Control in Moscow, rocket designers at the Energiya Scientific-Production Association, and others [Laza91].

In 1989 some efforts were made to remedy this situation through the creation of the Collective Use Computing Center of the USSR Academy of Sciences, housed in the new

Presidium of the Academy of Sciences building in Moscow, which currently contains an eight-processor El'brus-2 configuration [Veli89]. This was, however, too little, too late.

In comparison to the BESM-6, which was very reliable by Soviet standards and well understood by those who used it, the El'brus was considered very unreliable and hard to maintain, a fact which has been acknowledged by the designers themselves. This was especially true for those users who did not have strong maintenance support. Only in 1989 when gate arrays replaced the unreliable multi-chip modules did the machine become acceptably reliable.

#### 4.4.2 The Role of the El'brus in the Soviet Computer Industry

Approximately fifteen years elapsed from the start of El'brus-2 design work to its introduction into series production. Modest numbers of El'brus-1 computers were built in the interim, but the real goal, from the start, was the ECL-based El'brus-2. We observe that very long development times are a common occurrence in Soviet high-performance computing, but why is this so? In the case of the El'brus, many of the delaying factors can be traced to the nature of the technology, the structure of the infrastructure supporting research and development, the quality of inputs from, in particular, Minelektronprom, and, to a lesser degree, the relationship between ITMVT and the El'brus factories.

The El'brus computers were a driving force for the entire Soviet computer industry. The mission of ITMVT, both held by ITMVT engineers and imposed by policy makers in Minradioprom and the VPK, was to build the fastest machine possible given the available technology. Development involved a balancing act. On the one hand, a working machine had to be produced, so the capabilities of the supporting industries had to be taken into close account. On the other hand, high requirements were needed to force the rest of the industry to raise their technological level. Once production of a new technology (such

as components) was assimilated, it was often incorporated into other computers, such as the ES mainframes.

Building the “fastest machine possible” generally implied “at the world level.” Building machines at the “world level” required the development of new components, cables, power supplies, cooling systems, printed-circuit boards, connectors, computer-aided design tools, manufacturing facilities, etc. for each generation. The El’brus computers pushed the boundaries of many technologies simultaneously. Of the over one-hundred million rubles spent annually on high-performance projects spearheaded by ITMVT, only 25-30% stayed at ITMVT. The rest went to fund development of supporting technologies in other institutes, some of which were in other ministries. All together, the El’brus computers involved hundreds of enterprises which manufactured everything from cabinets to glass bulbs to printed-circuit boards to components. In most cases, new products had to be developed and manufactured.

It is perhaps the case that without a specific, high-profile project like the El’brus there would have been little impetus for the supporting industries to improve their technologies. Pushing many technological boundaries simultaneously had negative effects on the time needed to develop the end product, however. First, a great amount of time and effort was needed to get each factory to assimilate production of new items. Second, so many parts of the computer were little more than prototypes themselves that the debugging process was greatly extended.

Getting each of the factories to produce what was needed was “an absolute nightmare.” First, dealing with each factory involved a long bureaucratic trail. The director of the ITMVT, Burtsev and later Ryabov, would have to negotiate at each level of the economic management structure, from the factory up through ministerial section heads to the minister himself and, on many occasions, to the Central Committee of the Communist

Party. The greater the ‘‘administrative distance’’ between ITMVT and a particular factory, the more people were involved in this chain, the longer the negotiations took, and the weaker the feedback and accountability between ITMVT and the factory. Particularly time consuming was the interaction with entities in other ministries, in Minelektronprom, in particular, but even within Minradioprom negotiating the hierarchy was problematic. At each level one had to deal with individuals who had a monopoly position, and their own interests [Burt92, 9].

Second, although the factories were subordinated to the ministries, they did have considerable *de facto* influence over production. Factories were typically heavily loaded with orders and often used this as an excuse for not completing a given order on time. Under these conditions, factories tended to favor manufacture of simpler products already assimilated into production. Re-tooling for a new product cost much time, effort, and required the termination of some existing production, often at the expense of missing other production targets. Because each factory also had to depend on many other suppliers, assimilation of new production was very stressful. If the factory would not get any more money for a new product than for an old product for which there were still orders, it had little incentive to go through the trouble. Furthermore, since the Plan indicators often specified number of items to manufacture, quality was a secondary concern [Berl76]. The low quality of parts was a continuing thorn in the side of the El’brus designers; although they expended a great amount of effort and hardware designing the machine to be reliable, they were never able to completely compensate for the low reliability of the components.

For each new product, ITMVT faced factories disinclined to assimilate new technologies. Overcoming this resistance involved working the higher levels of bureaucracy to bring sufficient pressure on the factories that they would develop and incorporate El’brus

parts into production. The monopolistic nature of the economy further complicated matters. A very high percentage of the parts of an El'brus are single-source; only one factory manufactures them. El'brus designers and manufacturers had little choice in the selection of specific factories for specific products. If a factory was slow in producing a given item, there were no alternatives to waiting, or putting pressure on the factory to move more quickly.

In principle, the Military-Industrial Commission (VPK), which had inter-ministerial oversight over computing, should have been in a position to facilitate the interaction. In practice, according to some intimately involved with policy making, the VPK would at one time push for speeded development of one system, then later for another. The El'brus had to compete for attention and, when pressure was applied to speed up development of something else, El'brus-related efforts languished.

#### 4.4.3 The El'brus Component Base

While each new item which had to be assimilated into production and manufactured played a role in lengthening the development time, components were the most problematic. From the beginning of the El'brus program around 1970, the plans were to build the El'brus using ECL 100 series (later called the IS-100) components. These were functional duplicates of Motorola's MECL 10K chips. The first experimental units became available to designers in 1972-1973, but series production was far enough in the future that designers had to change their plans. The El'brus-1, therefore, was built using low TTL chips—the Logika-2 and 133 series, reportedly functional duplicates of Texas Instruments chips [Shaw85]—rather than ECL [Grub80; Mvke80].

In 1972, ITMVT initiated the development of a multi-chip module technology and pushed Minelektronprom to work on implementation as well. The modules consisted of 8-10 IS-100 chips on a single substrate. Theoretically, this would make it possible to in-

crease the speed of connections to individual chips. The chips themselves were manufactured by Minelektronprom, but ITMVT established a facility to place multiple chips on a single carrier. This was later done at the Minelektronprom plant. The first units became available in 1976, and development of the El'brus-2 processor began. Approximately half its logic was implemented as individual IS-100 series chips and the other half as multi-chip modules. This unit was built during 1979-1980 and debugging began at ITMVT around 1981.<sup>9</sup>

Both the IS-100 and multi-chip modules had reliability problems which compounded debugging difficulties and compromised the over-all reliability of the El'brus systems. By one estimate, 1-1.5 years were added to the debugging stage solely because powering a processor on and off had a tendency to break down the chips. Another source states that the system debugging was stretched out 2-3 times because of the poor components. On the order of five multi-chips modules reportedly were replaced per day at the beginning. Resistor chips used in the initial models were also quite unreliable. Several tens of El'brus-2 computers were built using the multi-chip modules. Given their low reliability, however, designers tried to switch to gate arrays as quickly as possible, eventually phasing out the multi-chip modules.

ECL gate arrays with 100-200 gates per chip were developed in 1983-1984 for the El'brus-2 and the first El'brus-2 processor using these (experimental) chips was completed in 1986. It took several years for Minelektronprom to assimilate mass production of the chips at a satisfactory level, however. Several El'brus-2 computers were manufactured at ZEMZ during 1985, but they were unworkable. Reportedly, the ceramic frames of the gate arrays would crack because the cooling system did not adequately dissipate the heat the chips generated. Efforts to improve the design of the gate arrays stretched

---

<sup>9</sup>Another informed source states that the first multi-chip modules became available in 1977-1978.

out development time by a year at least. Delivery of both the first experimental El'brus-2 (1984-1986) and second workable El'brus-2 based on gate arrays (1986-1987) were both impacted by the delay. Only in 1989 did the gate arrays replace the multi-chip modules in series produced El'brus-2 processors.

ITMVT served as a driving force for the electronics industry in more ways than one. First, development of the IS-100, multi-chip modules, and gate arrays was initiated by ITMVT for its high performance computers. Once manufacturing had been assimilated, the chips were used more broadly in the computer industry, in the ES mainframes in particular. For example, the ES-1035, ES-1036, ES-1061, ES-1065, ES-1066 and a number of parallel processors based on ES technology used ECL chips called the IS-500 or K500 series which were essentially the same as the IS-100 [Anto81; K86; Lomo87; Torg88]. The main difference, reportedly, is that the IS-100 have through-hole pinouts, while the IS-500 are surface mount. The I300B gate arrays, developed for the MKP and El'brus-3 (described below), have been used in a re-engineering of the ES-1066, called the ES-1087. ITMVT also used the components for its line of BESM-6 compatible systems.

Second, to get the needed chips, ITMVT felt it necessary to push Minelektronprom by actively participating in the design of the chips themselves. A rather close working relationship was established between ITMVT and engineers at Minelektronprom design facilities during the mid-1980s, especially after Ryabov became director of ITMVT [Kova87, 2]. ITMVT had a division devoted to the development of CAD systems for chip and printed-circuit board design, and between 1984 and 1988 ITMVT even developed a special-purpose system for testing chips for Minelektronprom. ITMVT engineers would take the basic technical parameters of the chips and design them using a CAD system which, while developed at ITMVT, was used by Minelektronprom. Simulation tests were also carried out at ITMVT. During this time there would be frequent interaction between

ITMVT and Minelektronprom. Close cooperation and exchange of technology was necessary to ensure that the design could be manufactured by the factory. The design, together with a set of tests developed at ITMVT, would be sent to Minelektronprom which would manufacture a chip tested jointly by ITMVT and Minelektronprom. ITMVT had to approve the first chip, which would then be put into series production.

The fact that ITMVT and chip designers in Minelektronprom had good relationships did not mean that it was easy to get the chip manufacturing factories (a different group of decision-makers) to produce the chips, for reasons mentioned above. Minelektronprom was notorious for manufacturing a few prototype chips which were approved by the state commissions, but then have great difficulties manufacturing large quantities of reliable chips. One El'brus factory representative has commented that the battle to get Minelektronprom to manufacture all the chips they needed was an annual affair, often reaching the level of the ministers who would sign an agreement among themselves indicating the amount of chips individual factories would receive.

Thus, the development of computers drove the development of the component base rather than vice-versa, as has become the predominant practice in the West, particularly during the 1980s and 1990s.

The complexity of the El'brus architecture compounded the problems with the component base. Early in the design phase many expressed concern that the machine was too complex for the available technology. These fears were realized. The complexity of the logic required much specialized hardware and levels of IC integration which placed great demands on the component base. Whether the architecture was too complex or the component base too weak is a largely academic question; the fact is that there was a mismatch between the architecture and what could be supported by the component base which aggravated the reliability problems.

This lesson was not lost on ITMVT developers, however. The El'brus-3 and MKP appear to have designs which, in relation to the capabilities of the components, are relatively less complex than the El'brus-2 [Baba89b, 130; Supe91, 17]. Whether they are simple enough for the components at hand is still an open question, however.

#### 4.4.4 Peripheral Storage

One area in which ITMVT has not been an industry driver is external storage. The El'brus machines have been equipped with magnetic drums designed for the BESM-6, but the magnetic disks and tapes have nearly always been those designed for the ES mainframes, using the IBM-compatible data channels. The El'brus-1 could accommodate up to 32 disks and 32 drums per I/O processor, with any four of them operating concurrently (on the four channels making up the fast channel block) [Golo80; Kriu80; Mvke80]. The maximum exchange rate with peripheral storage was 4 Mbytes/s per I/O processor. The ES-5056 magnetic disks had a storage capacity of 7.25 Mbytes each [Mvke80; Es76; Kezl86]. As disks with greater capacity were developed for the ES mainframes, they were incorporated into El'brus configurations. Although there is every reason to suspect that the most important El'brus users received the best drives Soviet and Eastern European (chiefly Bulgarian) industry had to offer, disk drives advanced slowly. The largest capacity disks manufactured in series production by Eastern Bloc industry, the 317.5 Mbyte ES-5063, did not become available until 1984-1986 or later [Dani84; Dani86]. With a data transfer rate of 1.198 Mbytes/s, the disks underutilized the El'brus-2 fast data channels which each had a transmission rate of 4 Mbytes/s [Baba90, 90]. Following the breakdown of trade relations with Eastern Bloc countries in 1991 (with Bulgaria in particular) shortages of disks became a severe problem for many users. As late as 1991 there were El'brus installations with (significant numbers of) 100 Mbyte disks only.

Although ITMVT did reportedly on occasion try to press Minradioprom for the development of higher-capacity disks, its efforts were unsuccessful.

#### 4.4.5 Relationship with Factories

ITMVT has had long-standing relationships with the small number of factories manufacturing its machines. The Moscow Calculating-Analytic Machines (SAM) Plant manufactured the general-purpose systems designed for mass-use. These include the BESM-6 and related machines such as the AS-6, the SVS-1, and El'brus-B. The MKP, developed by many of the same individuals who worked on the BESM-6 and AS-6 has also been constructed here. The El'brus computers were manufactured primarily at the Zagorsk Electro-Mechanical Factory (ZEMZ) outside of Moscow, with the I/O processors, telecommunications processors, and some disk drives manufactured at SAM [Supe91b, 15]. Some El'brus subsystems such as the memory modules were manufactured at the Penza Computational Electronic Machines Plant (VEM). During the 1980s a factory in Tashkent was also retooled to manufacture El'brus machines.

ITMVT worked closely with the factories, especially with their associated design bureaus. In general, re-tooling a factory was a very slow process for reasons mentioned above. By establishing a close working relationship early on in a given project, ITMVT could improve communication with the factory and help ensure that the machines were being constructed in a manner which suited the plant's capabilities.

ITMVT and its few primary production factories were dependent on each other. ITMVT depended on the factories' manufacturing facilities. The factories in turn depended on ITMVT for the development of new technologies: the machines, CAD systems, documentation standards, etc. By design, the factory and ITMVT worked with much the same technology. It was therefore possible, when necessary, for the factory to make small changes—redesign a PCB, for example—without the involvement of ITMVT.

The factory was also free to use the technology for development and production of products which had little to do with ITMVT. At the same time, the factory had little hope of advancing technologically without ITMVT's help. One exception to this symmetry, however, was printed-circuit board manufacturing. Around 1988, ITMVT had been able to acquire technology to manufacture 20-layer PCBs through a joint venture with the Swiss firm Rode, Inc. The same technology was to have been installed at the factories, but government hard-currency allocations were reduced before this could be done. Reportedly, the SAM plant has recently acquired equipment capable of manufacturing 20-layer PCBs (those used in the MKP and the El'brus-3) through a joint effort involving a German firm.

As soon as the work on concrete logic circuits began, ITMVT incorporated specialists from the factory design bureau into the development process. They served as liaisons to the series production factory. Interaction between ITMVT and the design bureau engineers took place nearly on a daily basis. The latter were intimately involved in prototype construction and the development of the series production documentation. Two basic tasks were accomplished concurrently: construction of the prototype, and creation of new production technology for the series production plant. The latter was nearly always necessary, since ITMVT machines were continually pushing existing technological boundaries.

Although a close working relationship with the design bureau greatly facilitated getting machines into production, it did not smooth the ITMVT-factory relationship completely. The design bureaus and the series production plants viewed a new machine differently. The engineers involved in prototype construction were usually enthusiastic about the work which they considered very interesting and personally challenging. For the series production plant, however, the introduction of new projects was a process filled

with uncertainty and delay. Before *perestroika*, the factories had more than enough orders to occupy existing capacity; assimilating new projects tied up such capacity and reduced the total output of the plant.

The technology itself complicated the relationship. The El'brus computers were difficult to build. They were also a moving target. As prototypes were built, the design and construction were frequently changed. Even after the prototypes were completed, modifications (such as the use of new kinds of chips) were made, requiring changes to the series production facilities. Furthermore, these changes had to be incorporated into machines which had already been manufactured, as well as to subsequent units. Although ITMVT directors were usually able to get additional funding allocated to the principal customers to cover this cost, the series production factories found such changes undesirable.

#### 4.4.6 ITMVT Structure

ITMVT had a traditional institute-division-subdivision-laboratory structure. The basic hierarchy and the corresponding titles, pay levels, and administrative responsibilities were well defined by regulations in effect for applied-science institutes throughout the Soviet Union. The specific structure of ITMVT reflected the technical tasks associated with building ITMVT systems. Divisions were devoted to specific machines, and the structures within a division were devoted to the constituent hardware and software tasks, etc. Thanks to the long-term and stable nature of systems development, ITMVT's structure remained rather constant throughout the 1970s and 1980s.

It was difficult for people to be transferred from one part of the organization to another, but communication throughout the institute was quite fluid. One of the traditions established when S. A. Lebedev was director was that the various teams should have the freedom to interact with each other and share ideas. Lebedev created a collegial work en-

vironment. The hierarchy rather strictly determined which individuals could write articles about the machines, however.

Prior to 1985, ITMVT incorporated a number of technical divisions. Two of these were devoted to the high-performance computing systems. The first, headed by A. A. Sokolov, worked on the MKP. This is the same division and, to a large extent, the same team that worked on the BESM-6 and the AS-6. This division also had a subdivision working on BESM-6 compatible systems called the SVS-1 and the El'brus-B. The El'brus computers were developed in a second division. This division consisted of hardware and software subdivisions.

ITMVT's research strategy was to develop two lines of high-performance systems concurrently. This practice originated in the 1960s when one team developed general-purpose systems (BESM-6 and its predecessors) and another developed special-purpose systems for the military. When the El'brus program was initiated, both teams worked on general-purpose systems. The strategy served two basic purposes. First, it increased the chances that one system or another would be put into series production. Second, it created some in-house competition for resources and recognition between teams which encouraged greater productivity and creativity.

Prior to 1985, Babayan was formally the head of the El'brus software subdivision, although in fact he was in charge of both subdivisions.<sup>10</sup> Functionally, the software subdivision had two components: software development tools, and operating systems. Compil-

---

<sup>10</sup>There were a number of instances in the El'brus division in which according to the formal structure (titles) two or more individuals were at peer levels (heads of laboratories, heads of subdivisions, etc.). In practice, however, one was the superior of the others. Such situations arose because of the regulations regarding wages, which specified how much money an individual with a given title could earn. An individual could not be given a raise beyond that specified in the regulations without giving him a new title. So, to increase an individual's wages, he would be officially made the head of an organizational unit at the same level as his immediate boss; unofficially, however, he remained subordinate to his boss. In other cases, a single subdivision could be broken into two pieces, effectively reducing the influence of the original division head. The reasons for such a change could just as easily be political as technical.

ers and other tools were incorporated into a subdivision separate from the operating systems subdivision because programming-language related features played an important part in the development of systems software. The Novosibirsk subsidiary of ITMVT which developed conventional language compilers like FORTRAN was actually subordinated to the software development tools subdivision, however.

A third division worked on small-scale special-purpose and embedded systems for the military.

A fourth division, under Ye. A. Krivosheyev, worked on a specialized vector-pipelined processor for El'brus-2 configurations [Burt85].

Two other divisions supported the systems divisions. G. G. Ryabov headed the division of computer-aided design (CAD) systems; F. P. Galetskiy headed the multilayer PCBs and subassemblies division. The latter had worked on the multi-chip modules. Another division worked on main memory systems.

The main memory and PCBs divisions reflected V. S. Burtsev's efforts to compensate for an insufficiently responsive Ministry of the Electronics Industry. He hoped that by creating divisions to work on the design and development of chips and printed-circuit boards that he could provide his systems developers with components earlier in the systems life-cycle, and get them into production in Minelektronprom more quickly.

When G. G. Ryabov became director, he made a number of changes to the organizational structure. First, B. A. Babayan was made the formal head of the El'brus division. Second, Galetskiy's PCB technology division was strengthened through increases in staff and equipment. Third, although he did not terminate the division working on the vector processor (discussed in section 3.2.3.1), he did not support its work and eventually the division dissolved and the vector processor project ended. Fourth, he transformed the memory and electronic components divisions into small units which were to serve as a

bridge between ITMVT and Minelektronprom, hoping that they would lead to improved relationships between ITMVT and Minelektronprom. As we have described, these changes did improve the interaction between ITMVT and some of the design groups within Minelektronprom, but they still did not result in the timely delivery of quality components by the electronics ministry.

In addition to the technical divisions, ITMVT had two administrative councils. One council oversaw dissertation defenses and the awarding of Candidate of Science and Doctor of Science degrees, and made recommendations of appointments to scientific positions. The scientific-technical council, consisting of the director, his deputies, the division heads, and a Party representative, advised the directorate (which consisted of the director and his deputies) on technical matters. Several specialized committees such as the architecture committee, the software committee, etc. were subordinate to this council. This council could make recommendations, but the real decisions were made by the directorate, and the division heads.

## **4.5 El'brus-3**

### **4.5.1 El'brus-3 Origins**

As the first El'brus-2 processors were being completed in 1984-1985, the El'brus team under B. A. Babayan began preliminary work on the next generation of machines, and in 1985 ITMVT received a state order for the design and construction of a machine with a theoretical peak performance of 10 Gflops. The basic requirements for the El'brus-3 remained the same as those for the El'brus-1, and -2. High-performance on both scientific and general-purpose computation, reliability, and software compatibility with earlier El'brus models were particularly important.

There were a number of problems with the El'brus-1, -2 design which made it unsuitable as a basis for a machine with the performance called for in the state order. In addition to the architectural features limiting performance discussed in section 4.3.5, much information about instruction and data dependencies that can be determined from the source code was not available to the dynamic scheduler at run-time. The scheduler could only look ahead, in the best case, 32 instructions (the total number of buffer stations holding instructions and operands/operand addresses at each functional unit). This was frequently insufficient, especially in the case of conditional control transfers. Also, the dynamic scheduling made diagnostics very difficult. It was impossible to determine statically the exact order in which instructions were executed. The changes in scheduling from one run to another produced performance figures that also varied. Babayan says that it was very uncomfortable for him when demonstrating the system's performance before government commissions not to be able to duplicate precisely the performance claims! For these reasons, designers decided use pipelined functional units and look at static instruction scheduling.

To gain more control over execution and use more of the data and instruction dependency information contained in a program, developers began considering the possibility of giving the translator knowledge of, and control over, each execution cycle. In other words, the compiler would have sufficient knowledge of instruction execution times, memory access times, and transmission delays that it could schedule execution at a very fine-grain level. Developers were initially not sure that such a machine could be built, but were inspired by the experience of Floating-Point Systems, Inc. (FPS), which has built attached-array processors in which programmers are given very low-level control over the scheduling of each functional unit.

Thanks in large part to the design efforts and persistence of ITMVT, development of I300B ECL gate arrays with 1500 gates/chip<sup>11</sup> had progressed far enough in the Ministry of the Electronics (Minelektronprom) by 1984-1985 that computer engineers could begin designing machines which would incorporate them. The chips, with a minimum and average gate delay of 400 and 800-900 picoseconds, would support the design of a processor with a clock period of 10 nsec.

The first draft design of the El'brus-3 was a pure, traditional vector-pipeline machine. Even with a 10 nsec clock period, a conventional vector-pipeline processor producing two results per clock period would have a theoretical peak performance of only 200 Mflops. To achieve 10 Gflops, 50 such processors would have to be combined into a single system! Greater numbers of pipes could be incorporated into a single processor, but the types of problems which could effectively use a greater number of pipes is limited; it was a strong requirement that the El'brus-3 have high performance not only on scientific, but also on general-purpose applications. For this reason a pure vector-pipeline approach was rejected [Doro92, 16-17].

To satisfy the El'brus-3 requirements and achieve greater control over execution, designers adopted a very-long-instruction-word (VLIW) architecture. VLIW is a one form of instruction-level parallelism [Fish91]. Others are superscalar and data flow. The three vary in how dependencies between instructions are specified and which part of the system (programmer or compiler vs. hardware) specifies the dependencies and makes scheduling decisions. VLIW places this burden on the compiler, while a superscalar approach places it on the hardware (as in the El'brus-2). In VLIW terminology, basic units of computation such as addition, memory load, and branch are called operations. These corre-

---

<sup>11</sup>In comparison, the early Cray X-MP (prototyped in 1982) used 16-gate array circuits. The Cray Y-MP (1987) uses 2,500-gate array circuits, and the new C90 (1991) uses 10,000-gate array circuits. All these gate arrays are built using ECL technology.

spond to instructions in traditional sequential architectures. A VLIW instruction is a set of operations that are to be executed simultaneously. The compiler schedules a program by forming instructions out of operations that can be executed simultaneously.

Although they abandoned the complex, zero-address instructions for stack processing, and dynamic instruction scheduling of the El'brus-2, designers continued to adhere to several other El'brus design principles. Coarse-grain multiprocessing, modularity, shared-memory, hardware support for high-level languages and the operating system, multiple functional units, and hardware data tags were all incorporated into the El'brus-3. The design requirements which had given rise to these principles in the first place were still in effect. Furthermore, the requirements that software be portable between generations of El'brus computers was no longer just a good philosophy; it was a practical necessity since a considerable amount of systems and applications software now existed for these machines. The early decision to code all El'brus software in a high-level language gave designers the freedom to alter the architecture of individual processors radically. At the same time, the procedure-oriented nature of the El'-76 programming language and process-scheduling on the El'brus-2 made it necessary to preserve a procedure-orientation in the El'brus-3. As we shall see, this requirement was a principal cause of the difference between the El'brus and Western VLIW approaches.

Following a brief description of the El'brus-3 architecture, we will compare the El'brus-3 with two Western commercial VLIW machines, the Cydra 5 from Cydrome, Inc. and the Trace computers from Multiflow, Inc.

#### 4.5.2 System Organization

To reach the goal of 10 Gflops, Babayan and others designed a tightly-coupled multiprocessor, shown in Figure 4-2 and described in more detail in [Baba90; Doro92], consisting of 16 processors, each with seven pipelined functional units (five arithmetic)

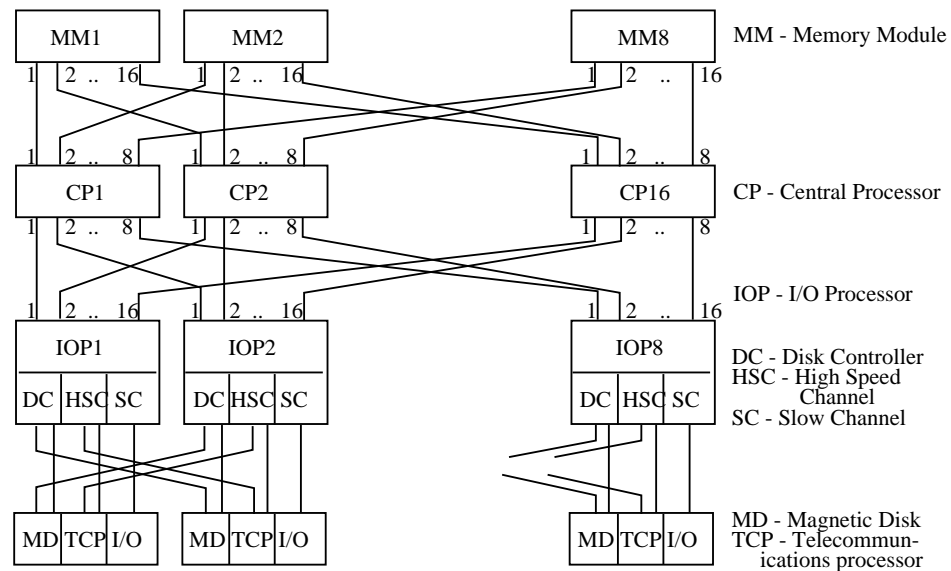


Figure 4-2 El'brus-3 Structure  
 Source:[Baba90, 38]

which could generate a result in each 10 nsec clock period.<sup>12</sup> The results of gate-level simulation performed after wire routing showed that a 12.5 nsec clock period was needed to allow the necessary data transfer at each stage, forcing designers in 1990 to change the clock period [Doro92, 7; Supe91b, 20]. Each processor therefore has a theoretical peak performance of 560 Mflops, or 8.96 Gflops for the full configuration [Doro92, 7]. The configuration also contains eight shared main memory sections, eight I/O processors, 16 telecommunications processors, magnetic disks, and other I/O devices. There are 18 Mbytes (2-megawords) of local memory in each CPU, interleaved on 32 banks. Each main memory section has a 288 Mbyte (32-megawords) capacity, with a cycle time of

<sup>12</sup>Each of the two logic units could perform a compare operation on floating-point operands.

400 nsec. The 16-ported main memory section is interleaved on 128 banks with a cycle time of 35 clock periods.

Data are transferred between the CPUs buffers, local and main memories, and the I/O processors via an I/O buffer in each CPU and input and output crossbar switches. Each of the eight I/O processors has a 200 Mbytes/s data transfer rate to and from the CPUs for an aggregate bandwidth of 1600 Mbytes/s.

#### 4.5.3 CPU

The CPU consists of nine functional units: two adders, two multipliers, one divider, two load/store units, and two logic units. To reduce the complexity of the interconnect structure, the divider and one of the logical units are each combined with a load/store unit to form two compound units. Thus, there are seven functional units interconnected by a full 15x16 crossbar switch. All functional units are pipelined. The CPU also includes an Instruction Unit, an Indexation Unit, a Cache/Memory Management Unit, Local Processor Memory, and a Buffer Memory. The latter includes a 1024-word stack buffer and a 512-word array element buffer [Baba89b, 125]. The El'brus-3 employs the same cache-coherence mechanisms as the El'brus-2 [Baba89b, 125-126].

Operands for the functional units can come from

- seven functional unit outputs;
- seven synchronous history result buffers;
- a multiported stack buffer;
- a multiported array element buffer;
- literals from the Instruction Unit.

#### 4.5.4 Influences on El'brus-3 Design

The El'brus-3 designers were strongly influenced by Western work on horizontal architectures. In 1981, FPS introduced the AP-120B attached-array processor. This system contained two floating-point arithmetic units which received operands via multiple data paths [Hock88, 209-224]. Instructions, issued every clock period, had fields which controlled the operation of all units in the computer during that clock period. These units included the two arithmetic units, an ALU for computing addresses, loop counts and indices, I/O ports, memory banks, etc. Programming the AP-120B required a very detailed understanding of the low-level timing and operations of the machine; as a result, very few people (chiefly those at FPS) programmed the system directly. Most users limited themselves to using the subroutine library provided by FPS. In short, given the technology and construction, the machine provided good performance, but it was quite inflexible and difficult to program. In general, an attached-array processor is more difficult to use and less efficient than a stand-alone machine because the programmer must manage two machines and explicitly move data back and forth between them. The data transfer path is slow relative to the processing speed of the attached processor, making it suitable only for problems in which the ratio of computation to data is high.

During the early 1980s Joseph Fisher worked on compilers for horizontal architectures which incorporated trace scheduling [Fish84]. Trace scheduling is a global compaction technique originally developed for generating long instructions of microcode from a sequential source (horizontal microcode). When using trace scheduling, the compiler “guesses” at the runtime control flow of a program so that sequences of code can be executed in advance, in parallel. Code can be reorganized when scheduling the Trace. To preserve the correct state of the machine to the external world if the compiler has made an incorrect guess and must backtrack, additional code (“compensation code”) is in-

served to recover the proper state. Obviously, the system works best when the compiler makes correct guesses. An advantage of Fisher's compilers was that they performed global optimization of a program. The compiler analyzed the entire program and in principle had the freedom to rearrange code throughout the program, not preserving the integrity of the program's basic blocks, at least not at the scheduling level. Much of Fisher's work was incorporated into the Trace series of VLIW computers developed by Multiflow, Inc. during the late 1980s.

The work of FPS and Fisher in particular influenced El'brus designers. The latter did not seek to copy the Western work, but studied it thoroughly to understand the strengths and weaknesses. In particular, they did not like the difficulty of programming and inflexibility of the FPS attached-array processors, or the way that Fisher's compensation code greatly increased the size of the executable (and slowed down execution when an incorrect "guess" was made). The most important influence of the Western work was that it demonstrated that a VLIW approach was possible, giving El'brus designers the confidence to proceed.

A third body of Western VLIW work resulted in the construction of the Cydra 5 departmental supercomputer at Cydrome, Inc. [Rau89]. As we shall see, there is significant overlap between many architectural features of this machine and the El'brus-3. However, the El'brus-3 had been nearly completely designed before 1989, when El'brus engineers first learned of the Cydra 5 [Rau89]. The El'brus engineers made many of the same design solutions, but independently of the Cydra work. This is an excellent example of the parallel evolution of technology.

#### 4.5.5 Comparison of El'brus-3 with Western VLIW Machines

##### 4.5.5.1 Scheduling

Like the Trace and Cydra 5, the El'brus-3 performs a global analysis of a source program. Like the Trace, it analyzes one procedure or module at a time, looking for the critical path of computation. The compiler builds data dependency, control dependency, and procedure call graphs and tries to discover hidden dependencies between addresses to be computed at run time.

In spite of its advantages in achieving greater optimization, however, a global scheduling approach could not be used by the El'brus designers. Because the El'brus compiler must use the same code for all calls to a given procedure, it uses a procedure-oriented static scheduling (PSS) model in which code motion is restricted to within procedure boundaries. Consequently, the El'brus-3 designers used methods of reducing control dependencies in conditional branches and procedure calls that differ from those used in the Trace and Cydra 5.

The fundamental reasons for the PSS lie in the El'brus design goals and development environment, which differ from those of the Trace and Cydra 5. In contrast to the American computers, the El'brus-3, like the El'brus-1, and -2, was developed almost exclusively for military applications. The two primary requirements were high performance and portability of software. Cost, either in rubles or volume of hardware, was not a strong constraint.

Under these circumstances designers decided to duplicate a set of functional units, to implement an expensive full crossbar switch and distributed register memory for intermediate results, to use multiported register files, and so on. The availability of a sufficient number of functional units made it possible to execute multiple operations from alterna-

tive basic blocks in parallel, rather than “guessing” at the correct path, as in the Trace. The El’brus-3 compiler does not select the most likely path and does not use compensation code to restore program correctness if some predictions were wrong. Instead, multiple possible paths are executed simultaneously. To branch means to take the results produced by the true path. Further details of scheduling of branches are discussed in [Doro92, 15-16]. There it is shown that the El’brus-3 and Cydra 5 treatments of memory access operations in alternative execution paths are equivalent (although slightly different in implementation).

To preserve the portability of software, the El’brus-3 had to support El’-76, the base language of all models of El’brus computers. A procedure is a key object in the language. It is both a building block of the program and a basic computation unit. Designers were obliged to preserve the ability to build a program from separately scheduled procedures without any code duplication. Code motion is therefore limited by procedure boundaries and the same procedure code is used in all calls to a procedure. This contrasts with the in-line procedure substitution used in the Trace and Cydra 5.

#### 4.5.5.2 Loops

One of the goals of the El’brus-3 design was to achieve vector supercomputer performance on scientific applications while providing high performance on a broader set of applications. The Cydra 5 designers had the same goal, building their machine to handle all the workload of a typical department, not just the numerical tasks [Rau89, 12-13]. Designing a machine which could process loops with recurrences and conditional branches was key. Such loops are typically not vectorizable on traditional vector-pipeline machines and therefore do not utilize the computing potential of those machines well.

As is described in [Doro92, 16-19], designers of both machines independently implemented very similar solutions. The approach of both machines is based on the ability to

store and access multiple loop iteration contexts during loop processing. For each loop iteration a set of registers, called an iteration frame, is dynamically allocated within the circular array element buffer. All variables to be processed during a loop iteration are placed within its iteration frame. Such variables can be both array or vector elements and scalars. They are accessed by means of base loop registers, each of which holds an iteration pointer and the iteration size. Any array element buffer is the sum of an instruction's displacement and the pointer.

Fast random access to vector elements has two primary advantages. On the one hand, the use of such a register file makes it possible to attain vector processing performance comparable to that of a vector supercomputer. In comparison with the latter, the El'brus-3 has additional overhead costs from the adding of the displacement to the loop base register.

On the other hand, random access to vector elements is much more flexible than in the case of vector registers. First, the size of an iteration frame and the location of any loop variable within the frame are known at compile time. Second, multiple iteration frames can be accessed during one clock period. As a result, any loop variable within the frames placed in the array element buffer can be accessed. The iteration frame approach is superior to vector registers in processing loops with recurrences.

In addition to providing mechanisms to support operations on multiple iterations of one loop, the El'brus designers implemented hardware mechanisms to help handle nested loops. In contrast, no architectural support for loops exists in the Trace, where loop unrolling in software is used. In the Cydra 5, hardware support similar to that of the El'brus-3 is provided for allocating registers in the array element buffer for each iteration, storing an iteration frame history, simultaneously accessing multiple iteration frames, and look-

ahead loading of data for the next iteration. It does not have hardware to handle nested loops.

#### 4.5.5.3 Instructions

El'brus-3 instructions consist of a set of operations and an instruction format. Operations can be both half-single (32-bit data and 4-bit tag) and single precision (64-bit data and 8-bit tag). There are also operations to work with any word of double-precision (128-bit) data. The El'brus-3 approach to instruction encoding is the same as for the Trace: a packed variable-length memory representation of unpacked fixed-length machine instructions. The packed El'brus-3 instruction can consist of one to four 72-bit words. Locations of some operations within the instruction are fixed and others are variable. The unpacked instruction is 504 bits wide; it has a fixed location for each operation.<sup>13</sup> The Cydra 5 also uses an instruction format in which six operations to the six functional units could be initiated by one instruction. The MultiOp format consists of seven partitions, one for each operation and an additional one to control the instruction unit and other miscellaneous operations. It is 256 bits long [Rau89, 24]. The Cydra 5 has an additional format, the UniOp, which allows only a single operation to be initiated per instruction, making it possible to fit multiple UniOp instructions in each 256 bit instruction word. This was done to handle portions of code with little parallelism more effectively.

#### 4.5.5.4 Synchronization and Exception Handling

Thanks to the shared-memory, multiprocessor nature of the El'brus configurations, there are some cases in which the static schedule of an El'brus computation can be altered at run time: overlapped execution for two independently scheduled procedures,

<sup>13</sup>Only 456 bits are used, however. [Baba89b, 123] states that the instruction word was 320 bits long. The size of the word was changed in 1989, after that article was written.

communication with asynchronous memory, and exception handling. The first case is specific to the El'brus, arising from the need to support separate compilation and execution of procedures, potentially using the same executable image; the others are traditional challenges for any VLIW architecture. In contrast to the Trace and Cydra 5, the El'brus-3 features the use of common, shared memory that is scheduled non-statically. The designers had no choice, because static scheduling of memory is not possible for multiprocessors with shared memory. The specific solutions to these problems are given in [Doro92, 21-22].

#### 4.5.6 Performance

Since a fully operational El'brus-3 processor has not been built at the time of this writing, it is impossible to state what the actual performance will be like. A simulation of a (10 nsec) single-processor El'brus-3 running the Livermore FORTRAN Kernels was run on an El'brus-2.<sup>14</sup>

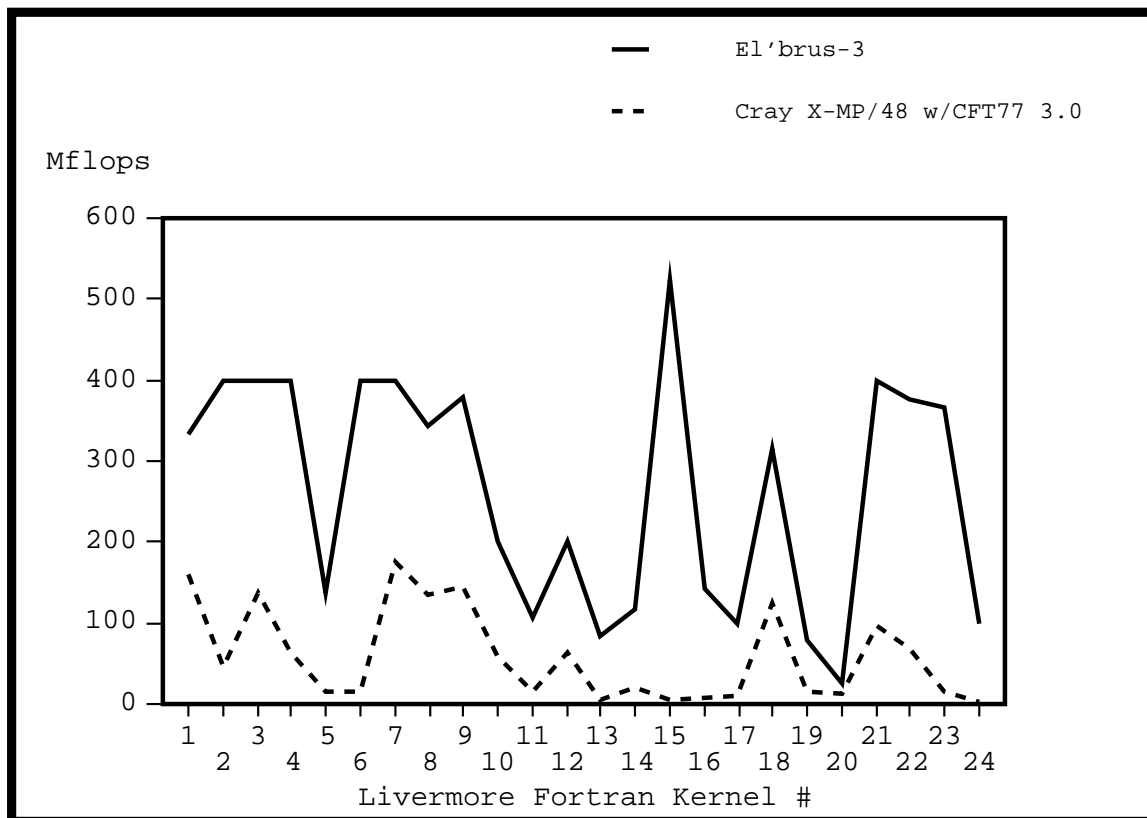
Figure 4-3 compares these results with real Cray X-MP/48 (single processor configuration) performance on the same benchmarks.

Although the meaningfulness of the simulation results can be debated,<sup>15</sup> they seem to indicate that the El'brus-3 is able to execute considerably more operations per clock period than the Cray X-MP, and, relative to the theoretical peak performance, has a more

---

<sup>14</sup>Jack Dongarra, the well-known author of the widely-used LINPACK benchmarks has remarked, "I never believe simulations."

<sup>15</sup>In particular, why should the performance on Loop 15 be the highest of all? It is vectorizable (although the Cray's CFT 77 compiler failed to take advantage of this), but not more vectorizable than other Loops.



	<b>El'brus-3</b>	<b>Cray X-MP/48</b>
Harmonic Mean	151.06	15.26
Maximum	523	175
Minimum	26	3.11
CPUs (in this configuration)	1	1
Clock Period (nsec)	10	9.5
Theoretical Peak Performance, Mflops (one CPU)	700	210

Figure 4-3 El'brus-3, Cray X-MP/48 Performance on LFK  
Sources:[Pfei90]

consistent performance across the spectrum of applications (as measured by the harmonic mean) [Supe91b, 19].

#### 4.5.7 Status

A mock-up of a multiplier was completed in 1990. In 1991, the manufacturing documentation was completed and several prototype processors were under construction at the Zagorsk Electro-Mechanical Factory outside of Moscow [Supe91b, 20]. Assembly suffered some delays while engineers waited for the necessary hardware from supplier factories. The El'brus-3, at the time of this writing, was undergoing hardware debugging. Financing remained stable, although marginally adequate, through 1992.

### 4.6 El'brus Microprocessors

During the early-mid 1980s as the El'brus-2 project was winding down, ITMVT engineers made important decisions about the future directions of the work. At this time they were strongly influenced by research strategies at IBM and DEC to provide a family of software compatible computers, covering a broad range of performances. At low end, DEC was pursuing the MicroVAX and VAXstation systems and IBM, the Micro370 desktop machines.

The El'brus approach was well suited for the creation of such a family. We have already discussed how the lack of assembler language programming enable ITMVT engineers to design a high-end, El'brus-2 compatible system with a radically different architecture. For the same reasons, engineers could build El'brus-2 compatible microproces-

sors having very different architectures.<sup>16</sup> Two lines of development were pursued, one based on the El'brus-2 architecture, and one based on the El'brus-3 VLIW approach.

#### 4.6.1 El'-90

The 32-bit El'-90 microprocessor project was begun in 1986 by a team headed by V. M. Pentkovskiy (the author of the El'-76 language). The technical statement of work was created in 1987. There were few commercial RISC processors at this time, but El'-90 designers carefully studied the available literature and those microprocessors which had been marketed, such as Fairchild's Clipper, introduced in 1985.

The El'-90 architecture reflects a combination of RISC and El'brus-2 ideas. The uniqueness of the microprocessor arises from the combination of basic RISC concepts such as simple control and a simple instruction set with features traditionally part of large-scale, mainframe systems: multiple functional units, multi-level memory hierarchy (including sufficient cache), and multiprocessing support [Pent90]. During the mid-1980s, few microprocessor designers in the world had clear ideas about the best way to proceed. For this reason, many of the design decisions were indigenous.

Because providing full hardware control like the El'brus-2 would have required very extensive and complex logic in the integrated circuit, the El'-90 designers chose a RISC approach here, simplifying hardware control and placing more of the burden on the software. They used a simplified instruction set, the majority of which could be executed in

---

<sup>16</sup>An interesting historical parallel is the development of the Alpha AXP 64-bit microprocessor at DEC. Like the El'brus microprocessors, the Alpha AXP was a radical architectural departure from an established line, the VAX, and was designed for use in multiprocessor configurations. To port the Open VMS operating system and other VAX-based user applications which had significant assembler-level code, Alpha engineers had to use a sophisticated set of "privileged architecture library" software routines to hide low-level hardware dependent functions, and develop a binary translator to run existing VAX binary images. This required much greater sophistication and effort than porting El'brus software to a new platform [Site93; Kron93; Thac93].

one cycle, multiple, highly-pipelined functional units, a wide control word visible to the compiler, multi-level memory hierarchy including large multi-port multi-window register files, and special instructions to support tightly-coupled multiprocessor systems. The El'-90 was also the first microprocessor implemented with a full-scaled tagged architecture to support dynamic, hardware typing, and ease of programming and debugging. Specifics about the construction of the cache—the cache's miss ratio, the write-to-memory policies, line replacement policies, etc.—were made with the goal of supporting a 10-processor configuration. Naturally, the experience of the El'brus-2 played an important role in these determinations. More detail of the El'-90 architecture can be found in [Pent90]. The "El'brus Micro", a multiprocessor designed to be based on the El'-90 is described in [Cher86].

The scale of the project (half a million transistors) required more extensive contacts between ITMVT and Minelektronprom than had been needed before. Thanks to G. G. Ryabov's efforts, a temporary technical collective (VNTK) was created to bridge the gap between ITMVT divisions and between ITMVT and Minelektronprom. This organization form had become possible during the mid 1980s and is discussed at greater length in chapter 6. Ultimately, the VNTK was transformed into a bona fide ITMVT division. The issue was made more urgent by the fact that the El'-90 was to be constructed using CMOS technology, rather than ECL which had been traditional at ITMVT. ITMVT engineers lacked the tools and experience to develop such technology alone. The VNTK incorporated both ITMVT engineers—V. Fel'dman and A. Zaitsev—and industry specialists, principally S. Kovalenko from Minelektronprom, in the same organizational unit. The first El'-90 prototypes were built in 1990. Because the electronics factory participating in the project, the Scientific-Research Institute of Precision Technology (NIITT) was unable

to achieve reliable production at the 1.5 micron level, no working chips were ever produced, although chips executing parts of the instruction set were developed.

The EI'-91S is the successor to the EI'-90. To a large extent, it preserves the design and technical decisions incorporated into the EI'-90. This project did not pass the design stage.

#### 4.6.2 EI'-95

In 1989-1990 when the EI'brus-3 architecture had been rather fully developed, B. A. Babayan initiated a second microprocessor project based on the VLIW architecture. The original intent was to implement the EI'brus-3 architecture and structure in several chips. This proved to be very difficult given the microelectronics technology available. To simplify the implementation, the notion of hardware tags was rejected and the number of functional units decreased. The subsequent design contained three arithmetic channels, a 12-port register file, variable length instruction words, loop processing techniques, and static instruction scheduling plus speculative execution of branch alternatives.

During 1990, ITMVT began very actively looking for potential Western partners for their work. Ryabov and Babayan spoke with many American, European, and Asian companies during the early 1990s, including Hewlett-Packard, Siemens, Hyundai and others. Their work on the EI'-95 microprocessor in particular impressed Bill Joy of Sun Microsystems, with whom they met in November, 1990 [Mark92]. Sun gave ITMVT the SPARC instruction set in 1991 and Babayan's team worked on the design of a microprocessor incorporating the EI'brus(-3) ideas in a manner compatible with the SPARC processors. Representatives of the two companies met a number of times during 1991 and early 1992, and in March, 1992 signed a contract for the design of a microprocessor and the development of a compiler for it [Mark92].

## 4.7 A Period of Change

### 4.7.1 Demand for ITMVT Computers

The El'brus computers were very large and very expensive, both to purchase and to operate. They cost millions of rubles,<sup>17</sup> needed extensive and complex cooling systems, and required a significant staff of operators to keep the system running. Although many (primarily military) users had been willing to pay large sums to achieve this computing power, not all potential users could afford these systems. This was particularly true during the early 1990s. Volume production of reasonably reliable El'brus-2 processors was finally achieved, just as the reforms introduced under *perestroika* started to take effect. As a result of the Law on State Enterprises (Associations) and the associated *khozraschet* accounting principles in particular, individual enterprises and institutes were granted significantly greater flexibility and responsibility for their own finances and became more careful with their expenditures. As government spending and investment in the military and scientific research and development were scaled back, organizations had less money to spend, and became less able to purchase expensive hardware. For example, the Institute of Applied Mathematics, which had received the first series-production BESM-6 and developed much systems software for that machine, decided to purchase a transputer-based machine rather than an El'brus-2, largely for financial reasons. Uncertainty about the future aggravated a worsening economic situation; unsure of what the future would hold, potential customers became reluctant to make large purchases.

---

<sup>17</sup>As price controls were lifted and negotiated prices became more common the late 1980s and 1990s, inflation rose to very high levels. Prices during these years are difficult to compare without specific information about price levels throughout the economy. The El'brus computers cost millions of rubles before inflation rose to significant levels.

Although the reforms gave enterprises the opportunity to purchase products directly from each other, nearly 100% of EI'brus-2 processors were manufactured under state orders. As government spending decreased, economic uncertainty increased, and the financial state of enterprises and institutes worsened, orders for the EI'brus-2 dropped to nearly zero by the end of 1992. Some individual processors were being manufactured for existing customers who wanted to upgrade a configuration or replace a processor, but no new installations were being created. A contributing factor, although probably not the dominant one, was the fact that the EI'brus-2 had become obsolete in the eyes of many.

The new machines, the EI'brus-3 and MKP, also had few customers, largely for the same reasons. Since these machines were still under development, there were few organizations outside the set of primary customers who had funded the development who wished to spend millions of rubles on unproven technology.

Ryabov has tried to find foreign customers for the machines among the countries of South-East Asia, but without success [Laza91b]. In short, by 1992, a viable market for the large machines had ceased to exist.

#### 4.7.2 Relationships with Suppliers

By 1990, Minelektronprom enterprises felt acutely the need to manufacture products for which there was a market. The weak demand for new ITMVT machines such as the MKP and the EI'brus-3 and the declining demand for the EI'brus-2 meant that the demand for the chips used in these machines was also slack. By 1991 Minelektronprom had already assimilated production of the I300B gate arrays so factories would have welcomed orders for them, if volumes were high enough. This was not the case, and economic considerations were pushing Minelektronprom factories in the direction of simple, consumer products such as ICs for watches and electronic games which do not require state-of-the-art manufacturing technology and which could potentially be sold for hard

currency on the world market. At times, such production was being pursued at the expense of the few orders from ITMVT. It was also becoming difficult for ITMVT to get chips which required development work from Minelektronprom, even when money was available.

The situation worsened in 1992. As orders for El'brus-2 processors declined, the loss of a single order had a significant impact on the volume of chips that need to be ordered. When orders are sporadic, the chip manufacturers have to shut down production for a time, then start it up again. As a result, the chips which are manufactured after a lull are often defective.

The USSR disintegrated into individual states in December, 1991, and the political divisions did have some effect on ITMVT activities. The factory in Tashkent ceased production of El'brus-2s, but this reportedly was, for ITMVT, not an unwelcome event, since the factory had required much hand-holding. Specialized power supplies were also manufactured in Uzbekistan, and an alternative producer had to be found. Some contacts for the El'brus-3 were manufactured only in Yerevan, Armenia. When supplies of these were cut off as a result of the turmoil there, El'brus designers had to try to find replacements. Other plants have not been able to manufacture parts with the necessary heat-tolerance, however [Gig1921217, 3].

The political divisions have had much less of an effect on ITMVT operations than economic factors, however, since most of the suppliers are in Russia. As relationships became based on economics rather than administration, enterprises became less willing to manufacture complex goods with a limited market. It became more difficult to acquire not only chips, but power-supplies, cables, connectors, etc. In some cases, suppliers were themselves unable to get the necessary inputs. In either case, production of many El'brus

parts became sporadic, at best, and El'brus development and manufacture suffered correspondingly.

The underlying reality is that computers like the El'brus require an extensive infrastructure of up-stream industries. Failure by a few enterprises to supply the necessary parts can stall the entire production. Given the low level of redundancy among these industries, finding replacements is very difficult. In recent years, this infrastructure became more and more unreliable.

#### 4.7.3 Relationships with the Factories

Traditionally, ITMVT and the factories were “glued together” by their mutual interdependence and the centralized planning mechanisms which stated that ITMVT computers were to be manufactured at specific factories for specific customers. Unlike entities in a scientific-production association, ITMVT and its factories were not linked administratively, except at higher levels within Minradioprom.

As individual enterprises were given greater freedom and responsibility for their own activities, the relationship between ITMVT and the principal factories gradually changed. The changes were not abrupt. ITMVT and the factory were still dependent on each other and both desired to continue to work closely together. Financing was still available for the factories, via ITMVT. State orders continued for the El'brus-2 and other systems through 1992. Funding for the El'brus-3 and MKP development continued—largely because the principal customers were reluctant to lose the large amounts they had already invested in these projects—although in real terms it declined, because of the high inflation levels.

Such funding was proving inadequate to support the over-all operations of the factories, however. ZEMZ and the Moscow SAM Plant no longer enjoyed the guaranteed El'brus market of earlier years.

Re-tooling for the manufacture of new machines involved considerable resources, and a market had to exist to recover those expenditures. While ITMVT directors had been able to get funding to cover the factory's cost over-runs in the past when El'brus-2 machine designs were changed, customers were now less tolerant. They began demanding that the factory cover such costs itself. Given the lack of a market, and the cost associated with high-performance computers, the series production factories were reluctant to establish a production line for them. The factories had to consider alternatives to large ITMVT computers.

Both factories started manufacturing goods for which there was a greater demand, including the assembly of personal computers, the manufacture of audio equipment, chemical products, watches, etc. The factories were also forced to become more attentive to the specific needs of customers. They began asking customers for their specific requirements and building configurations to meet those needs.

#### 4.7.4 Relationship with the Ministry

Traditionally, the ministry played a very important role in the the life of ITMVT. It had a dominant voice in creating the Plan for ITMVT and the associated factories. All financing passed through the ministry's hands, even though it came from a customer who commissioned the R&D. The ministry also selected or removed directors. In short, the ministry played a dominant administrative and financial role.

In the years following the Law on State Enterprises (Associations), the influence of the ministry waned. ITMVT was able to deal more directly with customers, and financing

for R&D began to come directly from them. The Law on State Enterprises (Associations) gave employees of enterprises the right to elect their own leadership. One leading individual at ITMVT stated in 1991, “Now I don’t know what the role of the ministry is.”

## **4.8 The Response to Change**

### 4.8.1 Changes in Structure

The five technical divisions have remained largely intact since the start of *perestroika*. Each of them is fundamental to ITMVT’s viability in high-performance computing (HPC). As long as the ITMVT’s mission is to build the fastest machines possible, it is felt that a structured, integrated set of divisions oriented around the major projects and supporting technologies is essential.

However, the pressures to provide sufficient wages to keep ITMVT employees from leaving the institute forced some experimentation with alternative organizational forms. These new forms have not taken the place of the existing structures, but are add-ons. The 1987 Law on State Enterprises (Associations) made it possible to create cooperatives under the auspices of enterprises [Prav870701]. These small organizations had few restriction on wage levels, and could negotiate prices for the services performed. In practice, the Law on Cooperatives provided a mechanism whereby some of the restrictions on wages and financing in state enterprises could be skirted. Rather than have work done by its employees in their capacity as ITMVT employees, ITMVT could contract out work to a cooperative at a price which would allow a relatively high wage for the cooperative’s employees. The end result was that the same people would do the same work, but via a legal mechanism which allowed them to be paid more for it. The cooperative also was a mechanism by which an enterprise could convert accounting rubles (*beznalichnyy*) into cash (*nalichnyy*). Such a cooperative was created at ITMVT around 1989. Research

work, or work under the institute's Plan could not be carried out in a cooperative, but much of the supporting work, or work done under contract for specific customers (installing new disks, upgrading a computer center, etc.) could.

As new laws on organizations were passed, ITMVT gradually took advantage of them. Shortly after the 1990 Law on Small Enterprises was passed, ITMVT created a single small enterprise for essentially the same reasons as the cooperative. The number of workers at ITMVT dropped from on the order of 2500 in the late 1980s to approximately 1700 by the end of 1992, mostly because of people who accepted an early retirement offer. This also eased the burden of wages.

During the late 1980s, ITMVT made use of an older law allowing the creation of temporary scientific-technical collectives [Ntr85]. This 1983 law made it easier to form temporary structures which incorporated people from multiple industries, organizations, and divisions within a given organization. They were another effort to reduce the large gap which often existed among enterprises involved in various industrial branches. Changes in legislation made such arrangements possible, but economic and technical factors made them desirable. We have mentioned the VNTK created to support work on the EI'-90. As customers became more careful with how they spent their money, ITMVT felt the need to work more productively. Such arrangements reportedly improved the efficiency of the teams. Another VNTK was formed to work on parts of the MKP project 1989. After the Law on Small Enterprises was passed, some collectives were established as small enterprises.

The introduction of new organizational forms within ITMVT could help increase wages of some individuals, but could not solve a basic difficulty with financing: the creation of large, complex systems required large amounts of financial and material support. The activities of collectives and small enterprises cannot generate nearly enough money

to support large-scale projects. ITMVT and the factories are administratively (and financially) distinct, so ITMVT receives no money from the sale of computers by the factories. In other words, the sale of current generation machines generates no income which can be used to fund development of the next generation. Traditionally, funding for large-scale R&D has come from a few principal customers who, in turn, had been allocated funds to support development by their ministries. As *khozraschet* took effect and the economic situation of these (military) customers worsened, they became reluctant and/or unable to spend the large amounts required to build new high-end machines. ITMVT had few options besides relying on government support.

The new structures, by nature small-scale and rather independent, were not well suited for the large-scale, integrated work required for large-scale systems. They were created to work on small, largely independent tasks which were funded independently of each other. Large-scale systems require considerable long-term funding from a single, or small number of sources. By the end of 1992, ITMVT had at least two types of organization. Approximately ten small enterprises had been formed, employing a few hundred people. They have their own bank accounts, the right to set their own pay scales, enter into contracts independently of ITMVT leadership, and make their own personnel decisions. They all exist within the framework of ITMVT, which gives them name recognition, technical assistance, etc. The largest of these is the SPARC Center, which forms the basis for a contract between ITMVT and Sun Microsystems [Mark92]. The SPARC Center drew in individuals from a number of ITMVT divisions. Most of the workers had been part of the El'brus-3 project in Babayan's division. Babayan was also able to attract individuals from the CAD division who had worked on tools for the El'brus-3, and programmers from the division working on military embedded systems.

The core functions of ITMVT—the capability necessary to develop large-scale systems—retained the traditional integrated division structure. These functions were not a part of a small enterprise and remained under the leadership of ITMVT. Ryabov saw to it that within each division, the essential teams of engineers had enough funding to keep them together, and projects to keep them active.

In March, 1992, the SPARC Center signed a contract with Sun Microsystems to work on developing a microprocessor and the associated compiler [Mark92]. The following September, the SPARC Center signed a second agreement with SunPro, a subsidiary of Sun Microsystems, for software development for existing Sun products. This contract involved 33 people located in Moscow, at the (former) ITMVT subsidiary in Novosibirsk, and the Computer Center of the former Leningrad State University [Mark92b].

#### 4.8.2 Changes in Technology

There have been few changes to the high-performance computing technology which can be directly attributed to the reform efforts. The technical characteristics of the MKP and the El'brus-3 reflect the requirements and design goals established when the projects were initiated. To our knowledge, there have been no changes to the El'brus-3 design, architecture, or construction which reflect attempts to make the system more viable under current conditions.

There are three main reasons for this. First, the El'brus-3 is close enough to completion that to alter its architecture, design, or construction significantly would seriously compromise it. Initiated just before the start of *perestroika*, the El'brus-3 was designed to be a very large, very powerful system, the most powerful system under development in the Soviet Union. Cost was not a strong constraint, as we have mentioned. By the time the effects of the deteriorating economy and strained development infrastructure were felt acutely in the early 1990s, the El'brus-3 had progressed to the final stages of devel-

opment. Changes at this point would have serious costs in time, effort, and money for ITMVT, its principal customers, ZEMZ, and upstream industries. Second, the principal customers still needed the most powerful computers they could obtain, and had already funded the project for half a decade. Changing the direction of the project at this point would jeopardize much of that investment. As a result, they continued to support the project. Third, ITMVT leadership felt that not following through on the El'brus-3 (and the MKP) would compromise the institute's position and capabilities in the high-performance computing sector. In Ryabov's words [Ryab91, 5]:

As for the larger machines, there have not been any sharp turns. We must absolutely carry through the work to the creation of a prototype. This is not simply connected with economics; it is necessary for maintaining the level of development. To be fully convinced of one's calculations, of one's achievements. As concerns the El'brus-3, it is simply a matter of implementing this long-instruction word architecture, of obtaining a working instrument. It is difficult to overestimate the feedback loop. You can't do everything on paper.

#### 4.8.3 Preserving Capability

The high-performance computing industry depends on the triad of developers, manufacturers and supporting industries, and users. The industry cannot survive unless each of these is active. Since ITMVT had been a leader and industry driver for decades, its challenge was not only preserving itself, but the entire industry of users, manufacturers, and supporting industries as well.

We have already mentioned that Ryabov's strategy for preserving in-house capability at ITMVT included securing sufficient financing for the El'brus-3 and MKP to keep the development teams intact, and preserving the division-oriented structure necessary for the development of large systems. In addition, he sought to find tasks which, while not nec-

essarily internationally competitive, would enable engineers to continue practicing their professional skills until funding for better projects became available. Such tasks were necessary for those involved in microelectronics and CAD development in particular.

To the extent possible, Ryabov worked to “secure an income” for the Minelektronprom factories developing El’brus chips. This included funding development out of discretionary ITMVT funds, trying to acquire government funding for development, and finding customers for machines built with those chips. To encourage the factories to continue production of the large machines, ITMVT was compelled to subsidize construction of the production technology and find customers for them, i.e. do much of the factories’ “marketing” [Supe91, 18].

To support the user community and expand the user base, Ryabov searched for ways to grant access to users who could not afford to purchase the machines, and include users outside the circle of traditional primary customers in the development process. One mechanism he plans to use is the creation of a supercomputing center located at Moscow State University. Since Soviet computer users are less able than before to purchase large, expensive systems themselves, a supercomputer center would allow multiple organizations to pool their resources and acquire a machine to use jointly. Ryabov is spearheading such an effort together with V. M. Repin, director of the Scientific-Research Computing Center at Moscow State University.

The centers would play an important role in “breaking in” new models, such as the MKP. The two or three years following state testing of the initial prototype have always involved considerable debugging and “modernization,” improvement of systems software and occasional upgrading of hardware elements. In the past, this has always been done at the installations of the primary customers, those who paid for the R&D. Ryabov is eager to place some of the first units into the hands of universities and centers such as

that just described to promote much broader involvement in the “breaking in,” and the more rapid development and perfecting of software. In an effort to avoid the criticism that accompanied the introduction of the El’brus line, “[w]e will try to see to it that each stage is controlled by our users.’”

Ryabov’s conviction (shared by many others) that all parts of the high-performance computing sector had to work together to survive was a principal factor in the creation of the Supercomputer Association of Users, Developers, and Manufacturers of High-Performance Systems on January 11, 1991. As indicated by its name, the association was formed by organizations representing a wide spectrum of high-performance computing activities. Members include ITMVT, the Moscow SAM Plant, the Kurchatov Institute of Atomic Energy, Moscow State University, several space-related research institutes, and integrated-circuit manufacturers from Zelenograd [Usdi91, 30]. In all, twenty-six organizations contributed to the start-up fund of the association, but considerably more have participated in the association’s activities [Supe91, 20].

Many of the participants in the Supercomputer Association were involved in a user-group for the BESM-6 which existed from 1968 to 1976 [Supe91b, 36]. This group, consisting of users and developers, held formal conferences every year and a half, and regular seminars. Like the BESM-6 users’ group, the Supercomputer Association was designed to draw together individuals from the entire spectrum of the HPC community to exchange information, expertise, and opinions [Supe91b, 36]. Among other purposes, such as the coordination and support of promising small-scale projects, the Association serves as a mouthpiece of the HPC community. Given the state of the nation’s economy, the sector had little hope of long-term viability if the government did not support it. The Supercomputer Association became one of the chief lobbying groups [Supe91, 6].

In October, 1991, the Supercomputer Association organized the conference “Problems of the Development of High-Performance Computing Systems,” held at a resort complex in Nepetsino, outside of Moscow. It drew approximately 210 individuals from 63 organizations [Supe91b, 34]. The conference sought to provide a forum in which four basic issues could be discussed: 1) the state of computer technology in the USSR; 2) the large problems—technical, economic, social, and political—facing the industry; 3) issues relevant to the user community; 4) issues regarding the future activities of the Supercomputer Association itself.

Many presentations and talks were given, but the underlying theme that “united we stand; divided we fall” was prevalent. No longer was it safe to assume that the government would sustain the sector. There were those within the government who argued (and continue to argue) that the questions of providing food, housing, and consumer goods to the populace and averting a breakdown of the social fabric are more important than the support of high-performance computing. The HPC community needed to make an equally compelling case that the future of science in the former Soviet Union depends to no small degree on the continuity of the scientific community through these troubled times, and that high-performance computing was not just one such branch of science, but was an important enabling technology for the rest. Many presentations, including a dubbed broadcast of an episode of “Adam Smith’s Money World” dealing with the Western supercomputing industry were used to stress this point. In short, the conference was designed to convince the HPC community to work together, and the government officials present that HPC deserved on-going support. The conference also gave ITMVT the opportunity to make a sales pitch for its own machines.

Was the conference successful? As G. G. Ryabov said late in 1992, “[t]he principal achievement of the conference is that the projects of ITMVT have not died” [Ryab92, 1].

The conference was probably not solely responsible for this fact, but did give ITMVT and other organizations a forum for making their case. HPC continues to be supported, albeit at a level which is hardly adequate for future development.

As 1992 drew to a close, HPC capability still existed, thanks to the persistence of ITMVT, although it was hanging by a thread. Although the factories had shifted much of their production to consumer goods, they still maintained, at a minimum level, the capability to manufacture high-performance computers.

#### **4.9 Discussion**

Since 1950 when S. A. Lebedev moved from Kiev to Moscow, ITMVT has been the unquestioned leader in Soviet high-performance computer research and development. Over the course of more than four decades engineers here have built over a half-dozen generations of high-performance computing systems for industrial use, amassing in the process considerable experience in all facets of computer development. In the process, the institute has been a driving force for the Soviet computing industry as a whole, sometimes pioneering the development of new technology, and always providing a high-profile project to serve as the focal point for technological advance in upstream industries.

The El'brus computers have been the most powerful built by Soviet industry, designed and developed for the most demanding users in the country. While reflecting the influence of foreign work, they are indigenous products which embody a host of original design decisions. Some, such as the particular dynamic scheduling and cache segmentation mechanisms are original to the El'brus. Others, like the use of hardware tags, were not original in the context of world-wide development, but were quantitative or qualitative variations of existing ideas. The evolution of the El'brus computers has been long and difficult. The R&D cycles regularly have stretched across ten years or more. As a

result, design decisions which were quite advanced at the time they were conceived, often were less so by the time the machines entered series production when measured against the international state-of-the-art.

There has been a great deal of continuity between the generations of this system, even during the turbulent *perestroika* years. Basic features of the design have remained qualitatively unchanged from one generation to another, although quantitatively the difference between El'brus generations is often greater than between consecutive models in Western supercomputer lines. Nevertheless, there are some highly significant qualitative differences which make the El'brus line a useful case for studying technological advance.

In this section we summarize the main factors which have shaped the complex evolution of the El'brus computers and the organization within which they were developed in light of the conceptual framework discussed in chapter 2. We highlight the changes in these factors since the start of the *perestroika* reforms and analyze their impact. We will set the stage for an examination of technological paradigms and trajectories in this context, and identify possible contributions of the El'brus story to contingency theories. In conclusion, we discuss the prospects for these machines and ITMVT more generally.

#### 4.9.1 The Technology

In table 4-3 we summarize some of the key factors influencing the evolution of the El'brus computers. Items in bold font indicate changes since 1985 from items in the preceding line. Within this extensive set are both elements of stability and continuity, and drastic change.

The El'brus computers evolved within the context of a set of guiding principles which, with some exceptions, has remained quite consistent over the last two and a half decades in particular. Each principle has been shaped by the development context, strate-

<p><u>Environment</u>  Directive form of economic management  <b>Increasingly market driven</b>  Monopolistic infrastructure  Extensive set up upstream industries  Contrary incentive structure for enterprises  <b>incentives based on market signals</b>  Requirements of principal users:  high performance on broad spectrum of tasks,  real-time capability, high reliability, ease of  programming, upward compatibility between  generations  Relatively close links with supporting industry,  based on mutual interest and administrative  ties  <b>weakening links with supporting industry,</b>  <b>based on mutual interest and persuasion</b>  Invasive regulations regarding organizational  structure and operation  <b>much greater autonomy to institute and</b>  <b>sub-institute organizational structures</b>  Strong market  <b>very weak market</b></p>	<p><u>Technology</u>  Modular,  Procedure-oriented,  Multiprocessing,  shared memory,  Hardware tags,  Segmented cache, etc.  Stack-based architecture  <b>Very-long-instruction-word architecture</b>  Complex instruction set computer (CISC)  <b>Reduced instruction set (RISC-like)</b></p>
<p><u>Technological availability</u>  Examples and ideas from West: Iliffe, Burroughs,  horizontal architectures, etc.  <b>Expanded opportunities for contact with</b>  <b>West and use of Western products</b>  Heavy reliance on advanced, newly developed  components, subsystems, tools. Often un-  reliable  Extensive experience in computer development</p>	<p><u>Organizational structure</u>  Traditional division-oriented structure  <b>hybrid, more flexible structure</b>  Structure oriented around basic HPC tasks,  subsystems, supporting technologies  <b>structures oriented towards new tasks</b></p> <p><u>Beliefs (design principles)</u>  Design fastest machines possible  Have high average performance  Develop computers for real users  Place as much control as possible in hardware  <b>Place much control in software</b>  Develop integrated hardware/software system  Develop all software in high-level language</p>
<p><u>Organizational slack</u>  High levels of funding  <b>Inadequate funding</b>  Integrated funding stream oriented towards large-  scale projects  <b>Greater reliance on small-scale</b>  <b>contract/project work</b></p>	<p><u>Strategy</u>  Serve as driver for supporting industries  Develop close relationships with factories  Cultivate in-house capabilities in variety  of supporting technologies  Use modular architecture, hardware support for  high-level language, general-purpose architec-  ture, etc.</p>

Table 4-3 Factors Influencing El'brus Evolution

gies, and technology, and in turn has shaped the R&D strategies employed and, ultimately, the technologies themselves.

A foundational principle at ITMVT since its inception has been that this institute would work to design and develop the fastest computers possible. Since high-performance computers (generally categorized by their performance rates) have often been viewed as a symbol of a country's technical achievement, the target speed for a new computer has always been made with an eye towards what has been achieved elsewhere in the world. A companion principle for the El'brus systems is that they deliver high average performance, not just theoretical peak performance. This mission has always been tempered by a second foundational principle, that ITMVT would be in the business of developing real computers for real users, not systems whose primary purpose is to demonstrate an interesting idea or feature. While the first principle has caused ITMVT to set its sights high, the second has forced developers to take a realistic view of the capabilities of the supporting industries, both at the time of design and in the near future. It has also compelled ITMVT engineers to consider all the parts of industrial computer systems. They cannot concentrate exclusively on developing a fast processor; they must consider each subsystem in the context of a complete configuration consisting of processors, memory, I/O capabilities, peripheral devices, interconnect systems, and software.

A key development strategy arises at the junction of these two principles. ITMVT is and should be, it is felt, a driver of the Soviet computing industry. The desire to build the fastest machine possible has caused ITMVT to make efforts to raise the capability of the supporting industries by being a demanding customer for advanced technology, actively participating in R&D in many fields, including microelectronics and printed-circuit boards, and playing an active role as spokesman and lobbyist for the entire high-performance computing sector. Moreover, ITMVT's leadership has felt—correctly, in our view—that if it had not play this leading role, the upstream industries would have advanced much more slowly than they did.

Taken together with the realities of developing advanced technology in the Soviet Union, the above create a Catch-22 for ITMVT. We discussed the impact that being an industry driver had on El'brus development times in section 4.4.2. Factors in ITMVT's environment, such as the nature of industrial structure and management and the corresponding disinclination of factories to assimilate new technology made the acquisition of the necessary inputs an extremely arduous task. The use of a wide array of newly-developed, unproven technologies made development difficult and time consuming. If ITMVT were not an industry driver, the supporting industries would have advanced more slowly. Because ITMVT was a driver for so many industries, however, development cycles were extremely long, by world standards. In order to remain a world-class player, ITMVT had to increase the performance of a new generation system not by 2-4 times over its predecessor like Cray, but by one or two orders of magnitude. This non-incremental approach, in turn, placed additional stress on the supporting industries, helping to ensure that development times would remain long and difficult.

Strategic decisions were made to address this issue. The first, dating back to Lebedev, was to cultivate close ties with the factories and other upstream organizations. While development cycles remained long, they would have been even longer had this strategy not been pursued. A second decision was to build in-house capability in a variety of supporting technologies.

At a more system specific level, the guiding design principles and strategies were shaped by such environmental factors as the principal users' requirements and the nature of the available technology, machine architecture ideas from the West, and the design philosophies of the engineers.

The requirement to provide high performance on a wide variety of applications reinforced the existing ITMVT mission. The reliability requirement forced engineers to con-

sider specific design features to compensate for low-quality inputs. The ease-of-programming requirement forced a departure from the traditional ITMVT approach under S. A. Lebedev.

The strategy employed to meet these requirements included the following elements. First, the machine would be modular in nature, providing redundancy in CPUs, memory, I/O processors, and data communications processors. This would facilitate maintenance, improve performance, and allow a variety of configuration sizes to suit different users. Second, high-level language and systems software constructs would be supported in hardware. Third, the system would have a general-purpose architecture using coarse-grain parallelism and shared memory. This provided the least problematic and most certain route to achieving high performance on a variety of applications.

Supporting these strategies were two design philosophies strongly held by Babayan, Burtsev and others. First, as much control as possible—over instruction scheduling, configuration control, error detection and management—should be placed in hardware. The machine should be designed to ease not only applications development, but also systems software such as operating systems, compilers, and utilities. Second, the machine should be an integrated hardware/software system.

Many of these philosophies and strategies had been forming at ITMVT during the 1960s, but the ideas put forth by Iliffe and their concrete implementation in the Burroughs machines served as a catalyst, inspiring and refining the thinking of the El'brus engineers. Similarly, the El'brus-3 evolution was shaped significantly by information about Western developments in horizontal architectures.

As we have seen, many of the requirements, strategies, and design philosophies remained constant between El'brus generations. The El'brus-2 and El'brus-3 have much in common in the basic requirements, the development environment, and the architecture on

the systems level. In the underlying CPU architectures, however, they are very different. These two machines and the El'brus microprocessors provide an excellent example of the impact of one generation of technology on the next, and the impact of shifts in certain basic design philosophies.

The El'brus-3 was developed within the context of the same basic system requirements as the El'brus-2. The latter had been a design goal of the early El'brus machines [Burt75], but was a strong requirement for the El'brus-3, since a considerable volume of systems and applications had been built. The design team of the El'brus-3 was to a large extent the same team that designed the El'brus-2; their experience and many of their design philosophies were applied directly to the new machine. The modular system structure and coarse-grain parallelism on processors with shared memory remained in the El'brus-3, as did multiple functional units, hardware tags and the corresponding support for high-level language constructs.

The existing software, requiring compatibility at the level of the El'-76 programming language, provided both opportunities for change and constraints. The designers had great freedom to alter the underlying processor design because El'-76 did not directly reflect the low-level structure of the El'brus-2 processors. The compiler provided the transformation from a rather abstract (high-level) program representation to the executable representation which ran on the hardware. Changing the hardware required changing the compiler, not the program being compiled. In contrast, a computer's assembly language generally reflects the structure of the underlying hardware explicitly and is very sensitive to changes in it. When systems software is written (at least in part) in assembly language, the need for compatibility becomes a constraint which forces a great deal of continuity of the hardware design from one generation to the next.

Given this freedom, El'brus-3 designers could implement a VLIW architecture which reflected not only a change in instruction set, number of functional units or registers, etc., but a fundamental change in one of the design principles. Control over instruction scheduling—handled dynamically by hardware in the El'brus-2—is handled statically by software in the El'brus-3. This change in philosophy was necessitated by the requirements for performance and the perceived inadequacies of the El'brus-2, and encouraged by the experience of Western horizontal architecture efforts known El'brus-3 designers.

While they did not greatly constrain the design of hardware, the operational characteristics of El'-76 did impact some of the mechanisms which were migrated into software in the El'brus-3. The primary example is the nature of the scheduling mechanism. El'-76 is a procedure-oriented language, and the need to support re-enterable, independently compilable procedures in an El'brus configuration forced designers to develop the procedure-static scheduling model described in section 4.5.5.1.

The level of financing, or organizational slack, available for the El'brus projects also had a significant influence over the machines' design. For much of the project's history, relatively high levels of funding were provided for systems development. Resources were not unlimited, of course, but the El'brus systems were among the highest priority computing projects. Resources were available to build expensive components and subsystems, or provide additional systems resources. For examples, designers of the El'brus-3 incorporated duplicate sets of functional units, a full crossbar switch, and multiported registers to improve performance, even though they were expensive and increased the volume of hardware. The directors of ITMVT were also able to acquire additional funds to cover cost over-runs at the development factory.

The changes in ITMVT's environment since 1985 have had little impact on the design, architecture, or construction of the El'brus computers. As discussed in section 4.8.2,

basic requirements and guiding principles remained quite constant, the cost of significantly changing a project became higher as the project advanced, and the principal customers continued to fund development. The main impact of reform-related changes was to the infrastructure needed to support such a project. The relationship between ITMVT, the factories, and the supporting industries became less administratively-oriented and more based on economics as decision-making and financial authority grew more decentralized. ITMVT made a great effort to sustain the funding and preserve the industry infrastructure and capability necessary to carry the El'brus through to completion. This task grew more difficult through the 1990s and the El'brus project suffered delays as a result, but the core infrastructure and the basic goals and strategies for the technology remained intact through 1992. The prospects for the technology are discussed at greater length below.

The El'brus-1, -2, and -3 lie along a "technological trajectory" which has been quite consistent for over 20 years. In each generation, designers sought to increase performance through some combination of faster and improved components, reduced clock periods, greater volumes of primary and secondary storage, greater numbers of processors and functional units within processors, and improved processor architecture. Improved performance did not come at the expense of loss of generality however; the El'brus machines performed well on a wide spectrum of problems. Basic systems characteristics—coarse-grained parallelism through a moderate number of powerful processors with shared main memory, modularity, multiprocessing, independent I/O and data transmission processors, hardware support for high-level language constructs, software compatibility with previous generations—remained very similar throughout the generations. There are many points of consistency in the lower-level design: the use of a segmented cache, hardware tags, etc.

One of the few points of sharp discontinuity in the technological trajectory was the design of the individual processors, as the stack-based architecture of the El'brus-1, -2 processors was replaced by a VLIW approach in the El'brus-3. Does this reflect a shift in the technological paradigm of the El'brus? We will examine this question in greater detail in chapter 8.

#### 4.9.2 The Organization

We described the structure of ITMVT and, in particular, the El'brus division in sections 4.4.6 and 4.8.1. Table 4-4 outlines some key influences on organizational structure. The basic structure, a traditional partitioning into division, subdivisions, and laboratories, was defined largely by the regulatory environment surrounding ITMVT and other applied research institutes throughout the USSR.

The specific structure, the number of structural units at each level and the specific tasks were determined primarily by the nature of the tasks they were to address and, occasionally, by political considerations. Because the research focus of ITMVT remained quite constant for decades, its structure remained reasonably stable. Structural changes reflected the changing nature of the research, or the desire to expand research in a particular direction. The creation of divisions to develop components, PCBs, and memory systems reflect a desire to try to compensate for weaknesses in the electronics ministry, for example.

The changes to ITMVT structure over the last decade reflect the interaction between the pressures and opportunities brought by reform processes, and the need for certain types of structures to support the institute's mission. The most significant factor creating a need for structural change has been the worsening economic climate. As financial conditions worsened, changes had to be made which would either increase the amount of organizational slack and/or improve productivity to make better use of that which was

<p><u>Environment</u> Laws establishing norms for job titles, wage levels Involvement of high-level officials in approving changes to organizational structures <b>Legislation allowing alternative organizational forms</b> <b>Legislation giving individual institutes the authority to determine their own structure</b> <b>Legislation implementing <i>khozraschet</i> principles at institute and sub-institute levels</b> Strong market for ITMVT products <b>Declining market for ITMVT computers</b></p>	<p><u>Technology</u> Functional division of tasks needed for El'brus development</p> <hr/> <p><u>Organizational structure</u> Traditional division, laboratory structure <b>More flexible, autonomous organizational forms: cooperatives, VNTK, small enterprises</b></p>
<p><u>Technological availability</u> <b>Examples of alternative organizational structures at other institutes</b></p>	<p><u>Beliefs</u> <b>Key to survival is maintenance of integrated structures</b> <b>Key to survival is retaining core development personnel</b></p>
<p><u>Organizational slack</u> Stable funding for HPC development <b>Decreasing government funding</b> <b>Weak, but existent funding for core R&amp;D teams</b> <b>Payment of higher wages through non-traditional organizational units</b> <b>Foreign investment (Sun Microsystems)</b></p>	<p><u>Strategy</u> <b>Maintain integration of core capability</b> <b>Creat flexible autonomous organizational structures</b> <b>Seek foreign partners</b></p>

Table 4-4 Factors Influencing Organizational Structure within ITMVT

available. “Organizational slack” was increased in two ways, by efforts to generate increased revenues, or to convert existing resources into a more flexible form, i.e. the conversion of accounting rubles (*beznaichnyye*) into cash (*naichnyye*). A basic goal was finding a way to pay workers wages that would sustain them and keep them at ITMVT.

The creation of cooperatives, small enterprises, and temporary collectives served each of these purposes. They provided a way to skirt legislative restrictions on wages, to enter into contracts with negotiated (i.e. higher) prices, to convert accounting rubles into cash to pay the workers, and bring together individuals best suited to carrying out a particular task in an efficient, timely manner.

Such organizational structures would not have been possible without changes to the laws regarding institutes and enterprises which made such structures possible and gave institute leaders the authority to create them. Although not a highly significant factor, ITMVT benefited from the experience of other organizations which had pioneered organizational forms soon after the laws were changed. In other words, the “technological opportunity” or know-how available in society helped encourage ITMVT leaders to make these changes.

Whole-scale conversion to small organizational units would have had a highly detrimental effect on the operations of the institute as a whole. ITMVT’s divisions had been designed to complement each other, and to work towards common goals. The high-performance computing divisions could not have been successful under the Soviet conditions without the contribution of the electronics and CAD divisions, for example. The various systems built at ITMVT had much overlap in their component base and construction. A non-trivial measure of centralized management and coordination was needed to integrate the efforts of the divisions.

The cooperatives and small enterprises offered new ways of increasing the wages of workers and improving certain types of work, but also decentralized the management of the institute as a whole. The small enterprises in particular were given considerable responsibility for finding their own contracts.

To maintain the institute’s overall capability and counteract the tendency towards decentralization and smaller-scale tasks, ITMVT has retained core capabilities in the traditional structures under the coordinating management of the directorate. Maintaining this structure requires financial and material support for the projects, however. To date, the government and principal customers for the large machines have continued funding at

levels which have kept the teams largely intact, but which are inadequate for comprehensive projects.

#### 4.9.3 Prospects

The reform process has brought a number of changes with positive implications for innovation at ITMVT, but many negative consequences that threaten the institute's ability to carry out the R&D of large-scale systems that has been its primary activity for decades.

Idea generation is directly related to amount of "cross-fertilization" of ideas which comes about through the intensive interaction of developers with other individuals or organizations who are working in areas related to HPC. In recent years, the some necessary preconditions for fruitful cross-fertilization have been established, although the short-term effect on the development of large-scale systems will not necessarily be great. We have seen through the activities of Ryabov and the Supercomputer Association that as the centralized support and coordinating mechanisms for HPC development have grown less certain, the developers, manufacturers, and users have become more genuinely interested in working together to save the industry. Developers at ITMVT must compete for customer finances and the goal of developing a product to suit a customer's needs and requirements has become more pressing. This is particularly evident in the activities of the small enterprises which must secure contracts for themselves to survive. Close interaction with a demanding customer base is crucial for generating ideas for future development. However, if the contracts are not for leading-edge systems which demand the generation and refinement of new ideas, work will stagnate.

As restrictions for interaction with Western organizations have been relaxed, cross-fertilization can potentially come about through a greater flow of ideas to and from the West. The SPARC Center is likely to have a high-level of idea generation, although not as high as it could be. Working under contract with Sun Microsystems, the center has

close interaction with a very demanding customer who is pushing the Center to generation, refine, and implement state-of-the-art ideas at a greater rate than would have been the case otherwise. Sun is supporting the work financially and materially, but beyond the basic information the SPARC Center needs to fulfill its contract, the flow of knowledge is largely uni-directional. Because of export control restrictions, Sun is forbidden to share with SPARC Center workers its cutting-edge ideas, even when the Center's work is on a par with Sun's.

“Coalition-building”—the securing of support from high-level authorities and the infrastructure needed to supply financing, materials, tools, and know-how—has become much more problematic for ITMVT as a whole, but easier, in some respects, for Babayan's division. In a nutshell, maintenance of the extensive infrastructure necessary to support large-scale, cutting edge systems development now depends on the good-will participation of the hundreds of upstream factories and institutes. The centralized administrative coordination no longer serves this purpose.

The infrastructure now depends more on horizontal, economic links than vertical, administrative ones. While a positive development in the long term, the short term effects are disheartening. The two levers now available to keep the infrastructure together are economic self-interest and persuasion. Given the worsening state of the economy, the cut-back on government funding, and the corresponding low demand for high-performance systems, neither level is operating very effectively. Complicating matters is the lack of redundancy in ITMVT's existing infrastructure. The failure of a small number of factories to deliver certain parts can easily cause the delay or failure of a project.

A fundamental problem faced by ITMVT is that it currently has few alternatives to seeking direct funding of R&D from outside sponsors, either industrial, foreign, or government. ITMVT is not administratively integrated with any organization which sells a

product which can generate in-house revenue for R&D. It does not receive an portion of a factory's profit from the sale of ITMVT high-performance systems, personal computers, watches or anything else. The small enterprises help generate small-scale revenue, but not enough to support large-scale projects. Without the possibility of directly generating revenue from its work, it will be very difficult for ITMVT to find investors other than those who are paying directly for research results. In the current economic climate, ITMVT is likely to be funded more for many, small-scale projects than for a few large-scale projects; there are now no indications that ITMVT will receive support for the development of large-scale successors to the El'brus-3 and MKP. Small-scale funding, although sometimes designed to keep core teams intact, will have a tendency to reduce the level of integration of ITMVT as a whole, making it more difficult to re-establish large projects in the future.

Babayan and his team at the SPARC Center are in a more favorable position. First, their work is being sponsored at a level adequate to the task. Coalition building for Babayan in this work is easier because he is no longer relying on a vast network of Soviet organizations. The Center's "product" is software, and a microprocessor design. Implementation and manufacturing will done in the West by Sun, a company with all the facilities necessary to bring the SPARC Center's results to market. The SPARC Center will also generate revenue through participation in the distribution channel for Sun products in the former Soviet Union. Thanks to the prestige of being hired by Sun Microsystems, Babayan's team should be in a good position to find additional support from the West, should this be necessary. If the relationship with Sun were to sour badly, however, this might not be the case. While the SPARC Center is certainly maintaining capability in machine architecture and software, the work is being done for Sun. The links between this

team and others at ITMVT which used to support the El'brus division's work are likely to grow weaker over time.

ITMVT's ability to construct prototype of large-scale systems has also been compromised by several factors. The issues of "technological availability"—the availability of the tools, components, and manufacturing technologies—remains particularly acute for the ITMVT machines, since they have always been very dependent on advanced, customized technology. Unlike other projects which used existing, proven technology, the El'brus and MKP systems demanded the development of new technologies. We have discussed the difficulties ITMVT has had acquire the necessary inputs.

Are the prospects any better today? Could Western technology substitute for Soviet technology in its machines? The possibilities are greater now, but the success of such an approach is questionable. In principle, ITMVT can now use non-Soviet parts in its computers. National policies in the past have stated that ITMVT computer had to use completely indigenous parts. This restriction has now been lifted. If it had the necessary financial resources (a non-trivial problem), ITMVT could acquire Western integrated circuits, CAD systems and workstations, disk drives, and many other pieces of commodity, or near commodity, technology. The borders between East and West have now become quite permeable to this kind of trade.

However, systems of the scale and complexity of the high-end ITMVT machines cannot be built solely out of commodity parts. They use customized chips, printed-circuit boards, cooling systems, cabling, etc. The development of the architecture is intimately linked with the development of these elements. ITMVT could reduce its dependence on some of the upstream Soviet industries, but would not be able to build machines solely out of Western parts. First, although CoCom restrictions have eased in recent years, they are still very much in force for advanced computing and the technologies—particularly

manufacturing technologies—which are necessary for their development. Second, the customized or special-purpose components and subsystems are very expensive in the West and could easily cost more than ITMVT or any customer would be willing to pay at present. Third, the logistical problems of trying to deal with vendors of specialized technology in the West would at best delay development and increase costs considerably.

Although the creation of flexible teams to address specific, temporary tasks may increase the institute's productivity in specific areas, the problem of prototype development is completely overshadowed by the difficulty of acquiring or developing the necessary inputs.

The path for ‘‘innovation diffusion’’ of ITMVT's machines has been well defined since the early 1960s. The relationships with the factories and associated special-design bureaus have been quite stable. ITMVT designed its machines for industrial use. Since the late 1980s, however, ITMVT's ability to get machines into series production has deteriorated as the market for the large systems collapsed and the relationships with the factories moved towards a greater economic basis. On the other hand, ITMVT is paying closer attention to what customers are willing to purchase may in the future develop systems for which a greater market exists.

What are the possible scenarios for ITMVT in the future? Will it be able to continue to develop large supercomputers? In the short term, the answer is most likely, no. The projects nearing completion—the El'brus-3 and MKP—will probably be completed. This task is difficult, but not impossible given that the necessary inputs have already been developed. The prospects for funding for a new large-scale project are currently very slim, and even with funding, the challenge of getting the cooperation of the upstream industries is very formidable. Without a steady stream of large-scale funding and support, there will not be a new generation of high-end systems.

For the intermediate term, ITMVT leaders are concentrating on keeping the core teams intact in anticipation of improved circumstances in the future. So far, they have succeeded. In the absence of a single, over-arching project to keep them integrated and pursuing a common goal, the work of teams is likely to drift apart over time. Even if each team works on state-of-the-art projects (such as at the SPARC Center), incompatibilities in technology, orientation, or culture will gradually arise between them. The more time passes, the more difficult it will be to reunite the pockets of capability into a single large-scale project.

The most likely scenario in our opinion is that ITMVT will continue to be transformed from a single, focused institute to a collection of smaller organizations pursuing their own projects. ITMVT will continue to lose its ability to organize and support the upstream industries and will have to become more of technology follower than driver. In other words, the upstream industries (some domestic, some foreign) will evolve in their own fashion, and ITMVT will increasingly have to accept the technology available and build what it can using it. Although the ability to develop large-scale systems will atrophy, its ability to build quality systems on a smaller scale may very well be enhanced.