Burroughs

BSP

OVERVIEW, PERSPECTIVE, ARCHITECTURE

BSP

BURROUGHS SCIENTIFIC PROCESSOR

OVERVIEW, PERSPECTIVE, ARCHITECTURE

CONTENTS

1. INTRODUCTION

One of the most exciting developments in large-scale scientific computing is the announcement of the Burroughs Scientific Processor (BSP). This system, capable of delivering up to 50 million answers per second, is intended to solve the very largest problems in engineering and science.

The BSP is one of the so-called "supercomputers." As such, it is designed to deliver at least one and in most instances several orders of magnitude more processing power than the largest general-purpose computers.

Supercomputer design and utilization is a subject of much more than academic interest. A number of application areas, addressable only by supercomputers, can be linked directly to our progress and survival. These areas include numerical weather prediction, structural analysis, linear programming, natural resource exploration, and nuclear technology. Associated with each application is at least one critical issue, as indicated below.

Application	Critical Issues
Numerical weather prediction	Agricultural production and flood control
Structural analysis	More energy-efficient, safer automobiles
	Safer, more economical buildings, bridges, roads

Linear programming	Application of limited resources to maximize or minimize a specified objective
Nuclear terminology	More cost-effective, safer sources of energy

Consider numerical weather prediction. At the present time, supercomputers are being used extensively by atmospheric research institutions around the world as key tools in understanding and predicting the weather. Assume it were possible to compute regional forecasts accurately several months in advance. Imagine how this would benefit food production. Given an accurate, long-range forecast, a country could take a major step toward predicting its crop yields and could plan to ensure that it had an adequate food supply. At the present time, it is conceivable that only a "super" supercomputer could deliver the computing power necessary to achieve this goal.

It has been argued in some quarters that all large computers (including the super-scale systems discussed here) will soon be superseded by collections of mini-computers or ensembles of thousands of microprocessors. The rationale behind this argument is that the era of truly inexpensive hardware is at hand; and that it ought to be possible to have (in some aggregate form, at least) several orders of magnitude more processing power at a much lower cost in the ensemble of micro-processors or the collection of minicomputers.

Unfortunately, no one has yet determined a method of controlling or utilizing the power available in the ensemble, nor how to partition a large problem currently soluable only by a supercomputer onto a collection of smaller machines in order to obtain a timely solution to the problem. Unquestionably, inexpensive hardware will be exploited in the future, but some fundamental problems in control will have to be solved first.

2. ARCHITECTURAL PHILOSOPHY

Parallelism is the architectural philosophy underlying the design of the BSP. It is synonymous with concurrency and simultaneity, namely, many things going on at once. It can be defined as the employment of multiple computing resources to increase the throughput of a system, and can be understood and utilized in terms of the two basic parameters that characterize all computers: space and time.

Spatial parallelism is exploited by employing replicated units doing identical tasks simultaneously. Temporal parallelism is exploited by equipping a single unit with the capability to perform different tasks simultaneously.

Given these definitions, it is easy to see that parallelism is not a new idea in computer design. It has been extensively employed in general purpose data processing systems via multiprocessing (replicated CPUs) and multiprogramming (where the I/O requirements of one job are balanced against the processing requirements of another job). The principal objective in the general-purpose system is to maximize the throughput of a mix of jobs (Figure 1); but in the context of very large-scale scientific processing, parallelism is defined with a different end in mind. It is the application of multiple computing resources to the solution of a single problem (Figure 2).

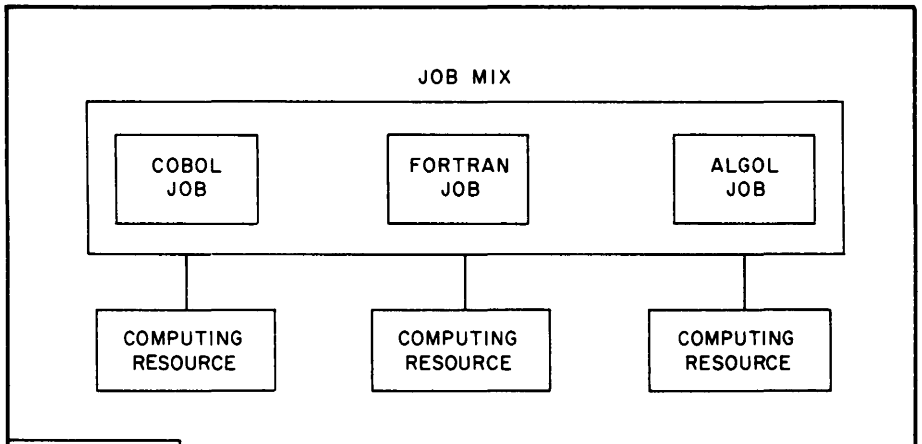

FIGURE 1.

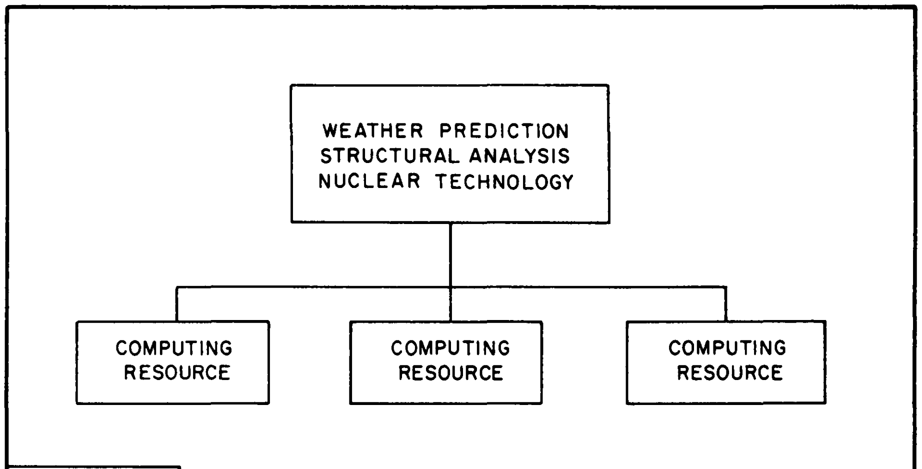

FIGURE 2.

3. PARALLELISM RATIONALE

The applications that require the power of a supercomputer are quite distinct from one another in that they address different natural phenomena and use different mathematical techniques. But they do have one common characteristic: massive amounts of computation. In fact, the number of arithmetic operations needed to solve some problems is now in the trillions.

This situation is not likely to change — for problem requirements continue to grow. Computerized models of natural phenomena are quite simple by nature's standards. Scientists are constantly striving to perfect their models by making them more accurate and by exercising them with more and more data (Figure 3).

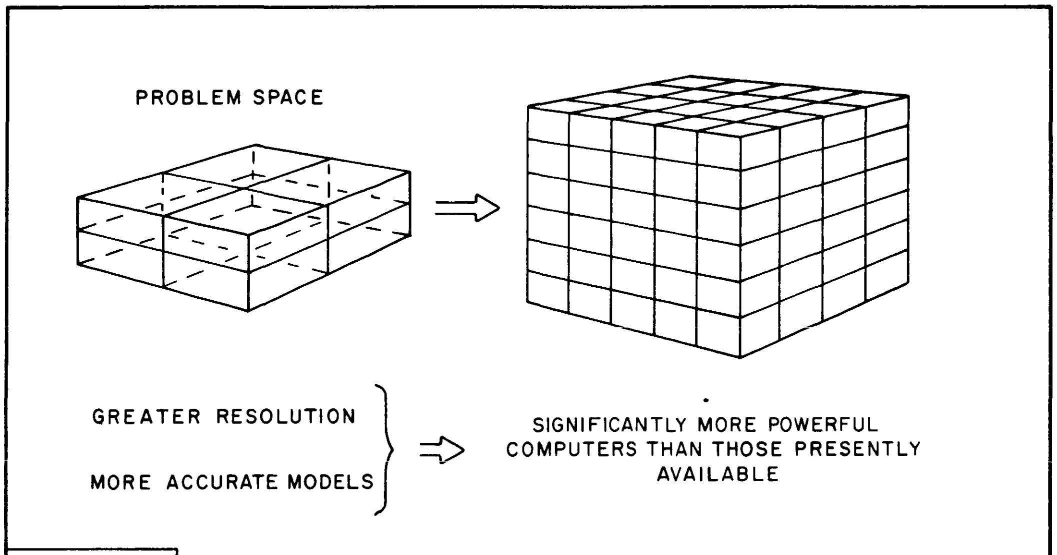
PROBLEM SPACE
GREATER RESOLUTION
MORE ACCURATE MODELS
SIGNIFICANTLY MORE POWERFUL COMPUTERS THAN THOSE PRESENTLY AVAILABLE

FIGURE 3.

The amount of computation required by more sophisticated models places enormous burdens on the computing systems which support them. The burden is especially heavy if the computer is sequentially organized (Figure 4), that is, if all arithmetic operations must be done one at a time. The reason is that sequential organizations are now running into the limitations of the so-far immutable law of physics which dictates that it is not possible to transfer information from one point to another faster than the speed of light.

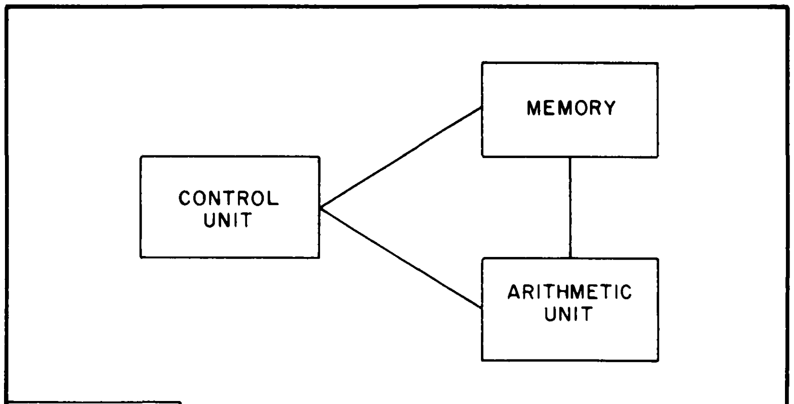

FIGURE 4.

Traditionally, serial machines have demonstrated performance gains by little more than a repackaging of the basic organization of Figure 4 in faster and faster hardware. That is, computer technology has advanced from vacuum tubes to transistors to integrated circuits, with corresponding increases in the number of operations per second (tens of thousands, hundreds of thousands, and millions of operations per second respectively).

While it is expected that "hardware only" based improvements will continue, they cannot be expected to continue at the pace that has enabled computer designers to see an order of magnitude increase in performance every three to five years. Thus, to guarantee the levels of performance needed by superscale problems, the conclusion is inescapable: some additional component is necessary in the basic architecture of a computer system. That component is parallelism.

4. PARALLELISM USEFULNESS

It is natural to ask if parallelism is a sufficiently general concept to be useful in computer design. Parallelism turns out to be extremely useful because our perception of nature is highly susceptible to the types of parallelism that can be built into a computer.

Our perception of natural phenomena begins typically with a description in terms of continuous mathematics, which is then translated into a description in terms of finite mathematics. The discretization process is suggested in Figure 5.

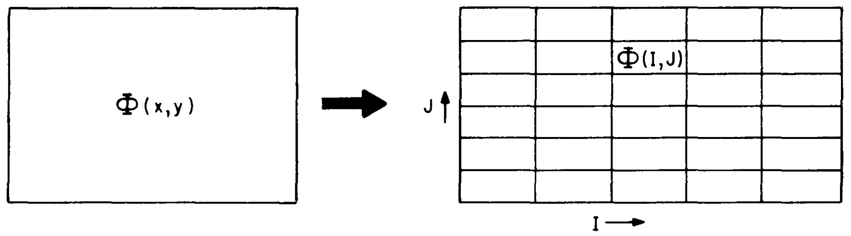

FIGURE 5.

Suppose the quantity of interest is a function called Φ (x, y). It might be a measure of temperature or charge distribution. It is to be computed over the surface of a slab by means of solving a differential equation. If the equation were exactly soluble, Φ (x, y) could be determined for any point on the slab. However, in many instances, the equation is not exactly soluable. One must, therefore, use a finite approximation to the differential equation and be content with computing the finite equivalent Φ(I, J) at a finite number of points on the slab.

Two points should be understood about the computer solution of the Φ(I, J) on a sequential computer. First, all Φ (I, J)s are computed one at a time. Second, the total amount of computation time is proportional to the number of grid points and to the solution time per grid point.

However, in many instances there is nothing in the mathematics which dictates that the Φ (I, J) be computed one at a time. In fact, many models have the property that Φ (I+1, J) depends only on Φ (I, J). This means that a number of Φ (I+1, J)s can be computed simultaneously implying a substantial increase in performance (Figure 6).

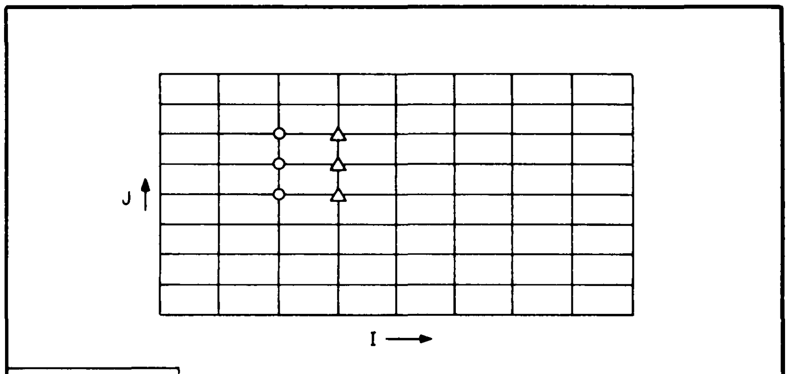

FIGURE 6.

Simultaneous computation suggests parallelism. Parallel or simultaneous computation in turn suggests that there may be an entity more suitable to an architecture based on parallel technology than the single operand which is associated (conceptually, at least), with a sequential or serial architecture.

The basic quantity susceptible to parallelism is the linear vector. In this context, a vector is defined as a set of operands upon which some sequence of arithmetic operations is to be performed. A linear vector is a vector whose elements are mapped into the memory of a computer in a linear fashion, i. e., the addresses of the elements differ by a constant (Figure 7).

Simple manipulations of linear vectors correspond to looping structures in FORTRAN. For example, if A and B are defined as vectors with 100 elements each, then the vector statement:

$$\overline{C} = \overline{A} + \overline{B}$$

is equivalent to:

```
      DO 10 I = 1, 100        (1)
   10 C(I) = A(I) + B(I).
```

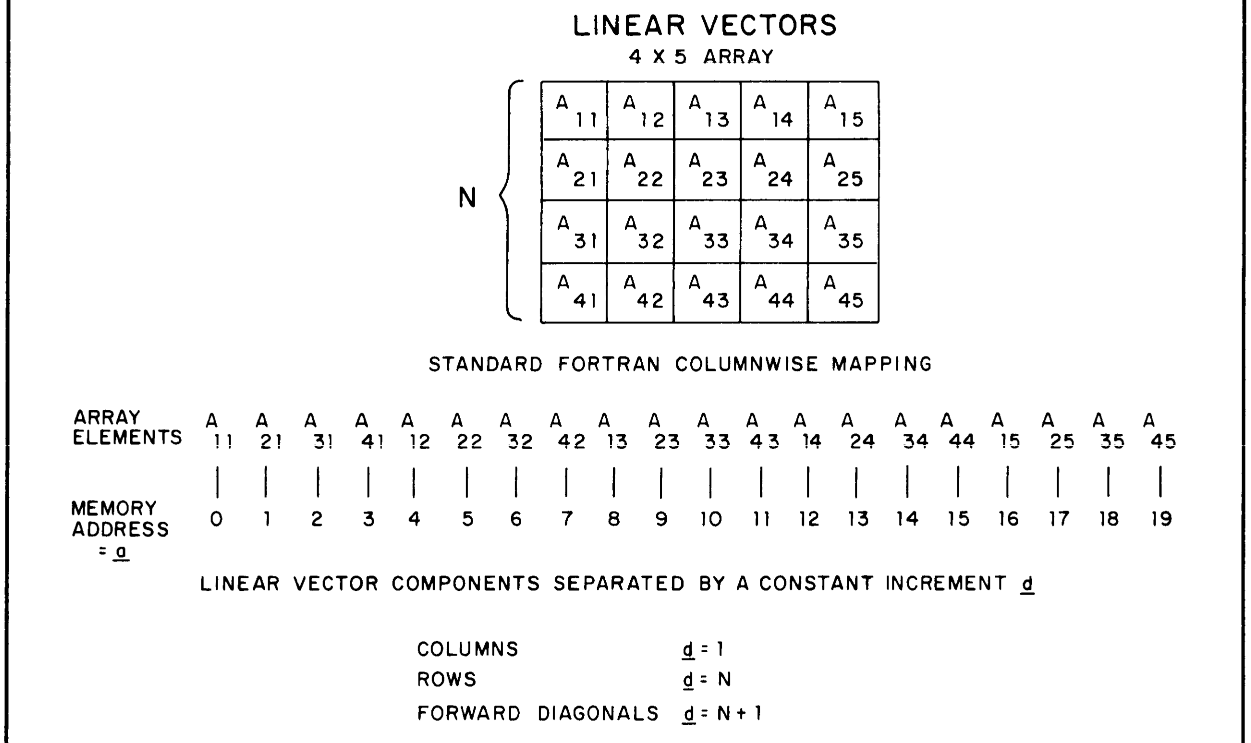
LINEAR VECTORS
4 X 5 ARRAY
N
A_{11} A_{12} A_{13} A_{14} A_{15}
A_{21} A_{22} A_{23} A_{24} A_{25}
A_{31} A_{32} A_{33} A_{34} A_{35}
A_{41} A_{42} A_{43} A_{44} A_{45}
STANDARD FORTRAN COLUMNWISE MAPPING
ARRAY ELEMENTS
A_{11} A_{21} A_{31} A_{41} A_{12} A_{22} A_{32} A_{42} A_{13} A_{23} A_{33} A_{43} A_{14} A_{24} A_{34} A_{44} A_{15} A_{25} A_{35} A_{45}
MEMORY ADDRESS = $\underline{a}$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
LINEAR VECTOR COMPONENTS SEPARATED BY A CONSTANT INCREMENT $\underline{d}$
COLUMNS $\underline{d} = 1$
ROWS $\underline{d} = N$
FORWARD DIAGONALS $\underline{d} = N + 1$

FIGURE 7.

5. PARALLELISM IN SUPERCOMPUTERS

Manipulation of linear vectors in supercomputers can utilize both the spatial and temporal aspects of parallelism. Spatially parallel supercomputers are called "array" processors; temporarily parallel machines are called "pipelined" processors.

The classical array processor consists of a number of replicated arithmetic elements operating in "locked step" under direction of a single control unit. Under ideal conditions, an array processor with N processing elements should realize an N-fold increase in performance over a serial computer with one processing element (assuming that the processor element characteristics of both systems are identical).

This can be illustrated by the example of vector addition (equation 1 on page 8). Suppose the sequential computer of Figure 4 requires four clocks (or machine cycles) to complete a single floating-point addition. The total time necessary to calculate the loop of equation 1 is:

$$t_{\text{sequential}} = (100) * 4 \text{ clocks} = 400 \text{ clocks.}$$

Suppose the array processor of Figure 8 has sixteen processing elements (PEs), each of which requires four clocks to complete a floating-point add. Because the PEs operate simultaneously, the example additions of equation 1 would be done sixteen at a time. Thus, the total time to complete the vector addition would be:

$$t_{\text{array}} = \left\lceil \frac{100}{16} \right\rceil * 4 \text{ clocks} = 28 \text{ clocks}$$

where the symbol $\lceil\ \rceil$ defines the so called "ceiling" operation, implying an integer division and a "round up".

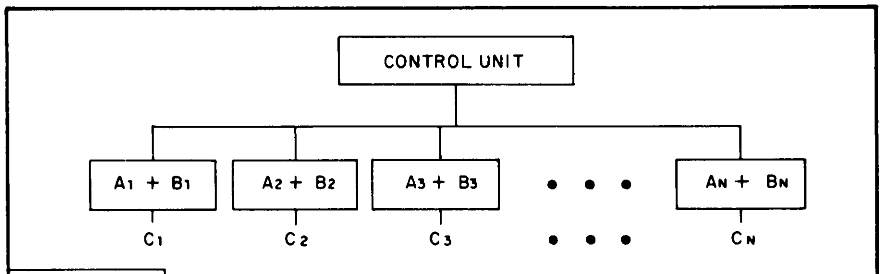

FIGURE 8.

In a pipelined computer, individual arithmetic operations are broken into stages. The output of one stage becomes the input for the next stage (Figure 9). This is contrasted with a nonpipelined unit in which one entire arithmetic operation must be completed before a new operand pair can be sent in. The advantage of a pipelined processor is that once the first pair of operands has been processed, results appear at the end of the pipeline every clock (instead of every four clocks).

The advantage of a pipeline processor over a sequential processor can also be illustrated by the simple example of vector addition. The time required by the simple four-stage pipeline of Figure 9 is:

$$t_{pipe} = 1 * 4 \text{ clocks} + 99 * 1 \text{ clock} = 103 \text{ clocks.}$$

Figure 10 illustrates a microscopic pipeline profile for vector addition.

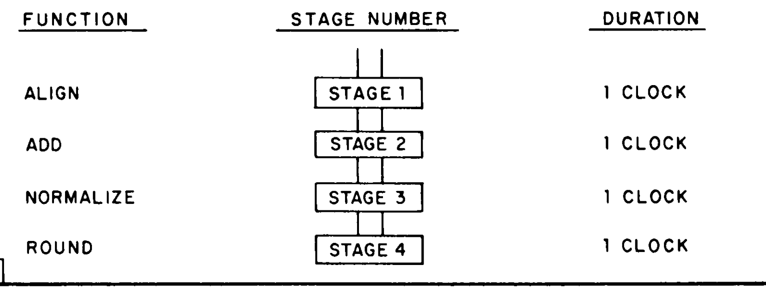

FIGURE 9.

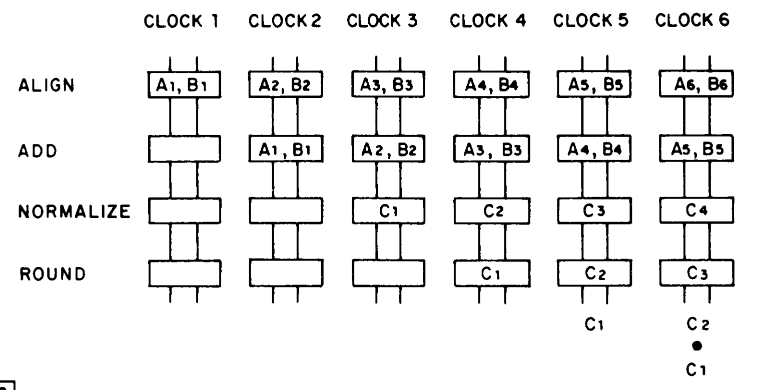

FIGURE 10.

6. PARALLELISM IMPLEMENTATION IN THE BSP

The architecture of the Burroughs Scientific Processor (Figure 11) exploits both spatial and temporal parallelism in that it is a synthesis of both array and pipelined processing. The BSP exploits spatial parallelism because its arithmetic engine consists of sixteen arithmetic elements operating in locked step. What is unusual about the BSP is the way in which temporal parallelism is used. The BSP is pipelined — but not in the traditional "microscopic" sense (where what is segmented is the elementary arithmetic operation). In the BSP, a more "macroscopic" point of view is adopted, and the focus of segmentation and overlap are the processes necessary to complete a memory-to-memory floating-point operation within the parallel processor.

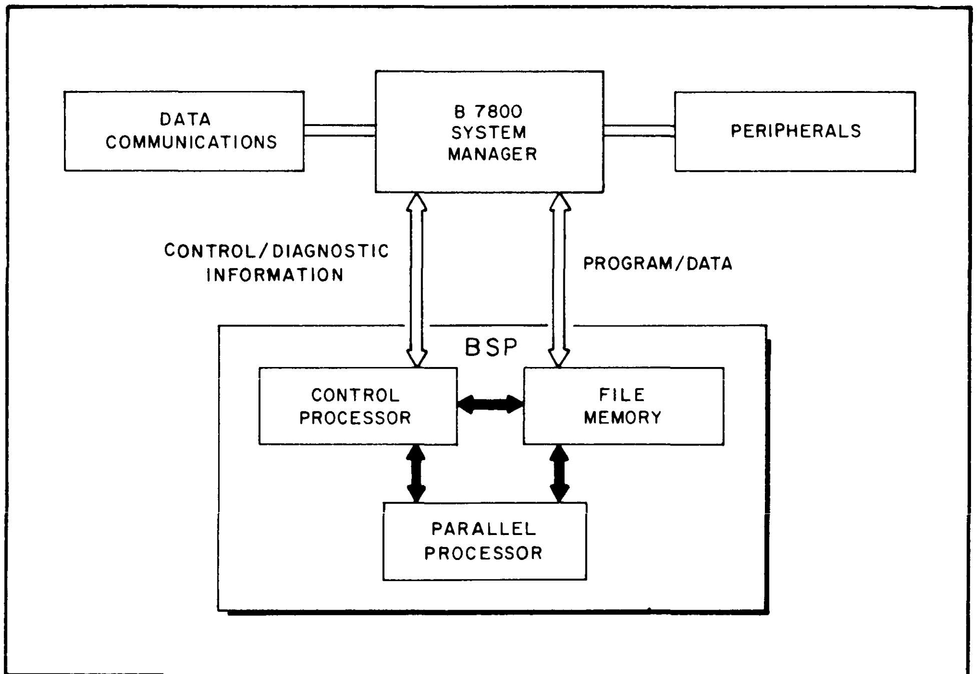

FIGURE 11.

How the synthesis is achieved can be understood by focusing on the organization of the parallel processor (Figure 12), and considering the sequence of steps necessary to complete a memory-to-memory operation. First, sixteen operands are fetched from the banks of the parallel memory, routed via the input alignment network into the architecture elements for processing, and rerouted via the output alignment network into the parallel memory for storage. These steps constitute a type of five-stage macroscopic pipe:

FETCH

ALIGN

PROCESS

ALIGN

STORE

In the BSP, these stages are completely overlapped (Figure 13), meaning that once an operation has started, processing in the arithmetic elements is essentially continuous.

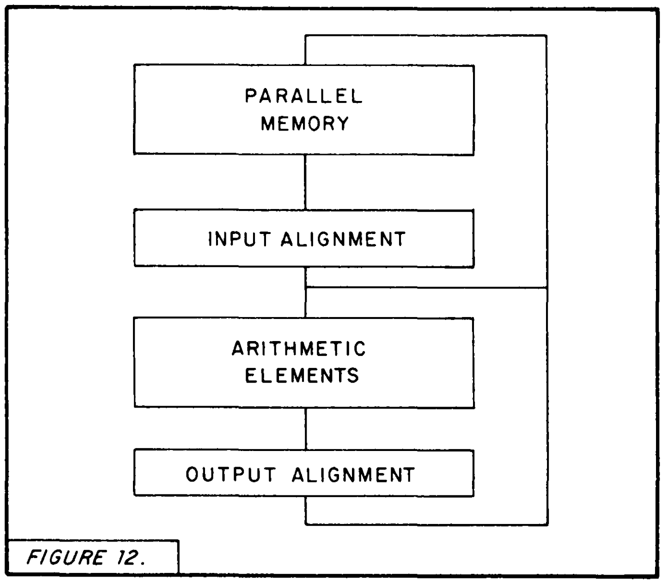

FIGURE 12.

CLOCK 1	FETCH		
CLOCK 2	ALIGN	FETCH	
CLOCK 3	PROCESS	ALIGN	FETCH
CLOCK 4	ALIGN	PROCESS	ALIGN
CLOCK 5	STORE	ALIGN	PROCESS
CLOCK 6		STORE	ALIGN
CLOCK 7			STORE

FIGURE 13.

7. SUMMARY

With parallelism such an integral part of its architecture, the BSP is ideally suited for high-speed scientific problem-solving. Its parallel architecture, complemented by conflict-free memory access, very large memory sizes, and exceptionally high-speed secondary storage transfer rates, can sustain computing rates in excess of 40 million answers per second.

With this type of power available, scientists and engineers can now solve the problems of today more cost effectively. They can also attempt the problems of tomorrow — which simply could not be addressed by less powerful systems.