

Recent Developments in Computers and Their Implication for Reactor Calculations

W. J. Worlton and E. A. Voorhees
Los Alamos National Laboratory

Argonne National Laboratory Report ANL-7050
Proceedings of the Conference on the Applications to Reactor Problems
May 17-19, 1965

Quoting from the bottom of page 17:

2.3 Increase in Computing-capacity Requirements

The increase in computing-capacity requirements over a long period of time in a scientific research center can be gauged quantitatively from Fig. 2. This figure shows the history of the computing capacity, measured in 7090 units, installed at Los Alamos Scientific Laboratory since 1952. The step function shows actual capacity, and the straight line is a "rule of thumb" line showing the growth in requirements. This curve, approximated by $C=2^{0.5t}$, indicates that there is a long-term increase in computing-capacity requirements of about a factor of two every 2 yr. Selengut⁽⁹⁾ has suggested that the speed of available computers increases by about an order of magnitude (a factor of ten) every 5 yr. This would correspond to an increase of $C=2^{0.67t}$ rather than $C=2^{0.5t}$. However, this value was suggested for short-term increases rather than the long-term experience shown in Fig. 2. the cuve is not meant literally, of course, but has provided an accurate guide for anticipating computing requirements for many years.

The prediction from Selengut (1959) is that computer speeds increase by an order of magnitude every 10 years. This is approximately the same as "Moore's Law" which predicts that computing speed doubles every 18 months.

RECENT DEVELOPMENTS IN COMPUTERS AND THEIR IMPLICATIONS FOR REACTOR CALCULATIONS*

W. J. Worlton and E. A. Voorhees
Los Alamos Scientific Laboratory
University of California
Los Alamos, New Mexico

(Paper presented by W. J. Worlton)

I. INTRODUCTION

The word symbiosis has a Greek origin meaning "to live together" and usually implies that two or more forms of life live together for their mutual benefit. The concept can be generalized to mean any mutually beneficial relationship, including different areas of science. It would be difficult to conceive of physics without mathematics, medicine without chemistry, or engineering without physics. The present high level of development of the sciences could hardly have been achieved without the very effective mutual assistance which they provide each other.

This paper, and indeed this conference, is concerned with the symbiotic relationship between the Nuclear Sciences and the Computer Sciences. Although conceived earlier, both of these sciences were born in the 1940's, and have matured together. The Nuclear Sciences have provided motivation and direction for the Computer Sciences, and the Computer Sciences have provided the means for solving problems in the Nuclear Sciences which could not have been attacked in any other way. The need for scientific computation was felt so strongly by the National Laboratories and other laboratories of the AEC contractors that they pioneered the development of stored-program computers in the early 1950's, e.g., the ORACLE at Oak Ridge National Laboratory, the AVIDAC at Argonne National Laboratory, and the MANIAC-I at Los Alamos Scientific Laboratory. It would be impossible to write a "Who's Who" of the early development of computing without including a large number of nuclear scientists whose influence covered both fields: von Neumann of the Atomic Energy Commission, Householder at Oak Ridge, Ulam and Metropolis at Los Alamos, Flanders at Argonne, and Ehrlich and Hurwitz at Knolls.⁽¹⁾ Although computing now has a much broader base of support than just the Nuclear Sciences, there remains a strong and important relationship between these disciplines.

In this paper we will analyze the effects of recent changes in computing technology on this continuing relationship, including the growing capabilities of computers, the changing price structure of computing, and the requirements imposed on the computing industry by reactor calculations.

*Work performed under the auspices of the U. S. Atomic Energy Commission.

We have confined our attention to the ultra-high-speed computers rather than to the whole range of computers which might possibly be used in such calculations.

2. THE REQUIREMENTS OF REACTOR CALCULATIONS

2.1 Classification of Reactor Calculations

Sangren,⁽¹⁻⁷⁾ following the work of Radkowsky and Brodsky,⁽⁸⁾ has been largely responsible for a unified classification of the wide range of reactor calculations. This conceptual model, shown in Fig. 1, notes the four major areas of reactor calculations and the important approaches in each area. The classification can be carried further to show code types and methods of solutions, but soon loses its conceptual value as it gains completeness. The model is intended to provide only a general insight and is an intentional oversimplification of a very complex and overlapping field.

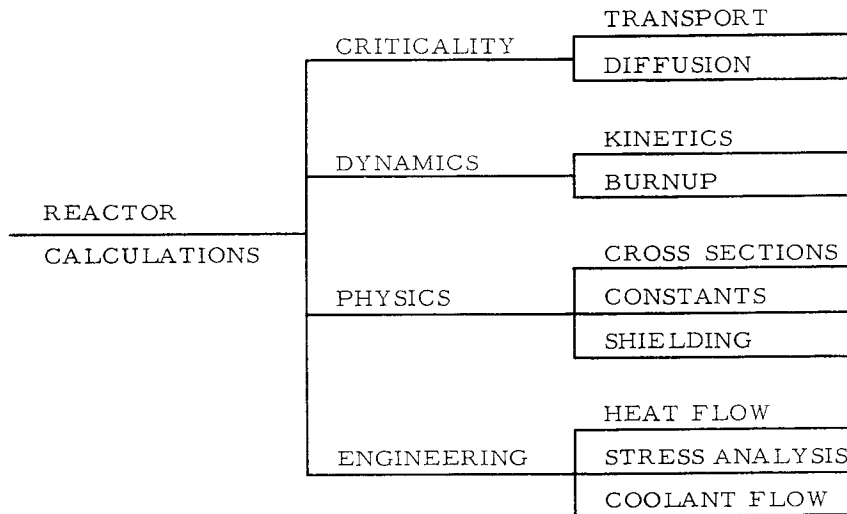


Fig. 1. Classification of Reactor Calculations

2.2 The Limitations on Reactor Calculations

Problems in the analysis of nuclear systems, together with the problems in the aerospace industry and weather analysis, belong to that difficult class of scientific calculations for which there has never been a computer with enough speed, storage, or adequate input-output equipment to satisfy the demand. This has implied that crude, inexact methods were used rather than the more elegant, but more expensive, methods. It has meant that one-dimensional codes were used when two and three dimensions were really needed to provide complete and accurate information. It has meant that diffusion theory was used where transport theory was needed.

It has meant using a series of time-independent calculations, with hand calculations in between, instead of time-dependent calculations. It has meant running Monte Carlo calculations to low confidence limits. It has meant restricting the number of calculations which have gone into parameter studies for optimization purposes. As the capabilities of computers have increased, some of the restrictions imposed earlier have been removed. But as the computing time for a particular type of calculation is reduced, the calculation is then used as part of a parameter study, thus imposing new quantitative requirements.

The most important restriction, however, has been qualitative in nature. There has always been a class of problems for which the available computers were inadequate--problems which were simply out of the question, either because of excessive storage requirements or excessive running time. Calculations which take a full 8-hr shift to complete border on the impractical, since a very large fraction of the total computing capacity is committed to them, and they thereby exclude other calculations. Thus, both quantitative and qualitative boundary conditions have been imposed on reactor calculations by the speed of the computers available.

Storage capacity has also imposed restrictions on the class of possible reactor calculations. An attempt has been made to bypass the storage restrictions by segmenting problems to allow the use of an external storage medium, usually magnetic tape or disk, as an extension of main storage. The effect of this has been to increase the code preparation and debugging time (since the control algorithms are more complex), increase the probability of error (since no external storage device is as reliable as main storage), and increase the running time (since access to any auxiliary storage device is relatively expensive in time).

2.3 Increase in Computing-capacity Requirements

The increase in computing-capacity requirements over a long period of time in a scientific research center can be gauged quantitatively from Fig. 2. This figure shows the history of the computing capacity, measured in 7090 units, installed at Los Alamos Scientific Laboratory since 1952. The step function shows actual capacity, and the straight line is a "rule of thumb" line showing the growth in requirements. This curve, approximated by $C = 2^{0.5t}$, indicates that there is a long-term increase in computing-capacity requirements of about a factor of two every 2 yr. Selengut⁽⁹⁾ has suggested that the speed of available computers increases by about an order of magnitude (a factor of ten) every 5 yr. This would correspond to an increase of $C = 2^{0.67t}$ rather than $C = 2^{0.5t}$. However, this value was suggested for short-term increases rather than the long-term experience shown in Fig. 2. The curve is not meant literally, of course, but has provided an accurate guide for anticipating computing requirements for many years.

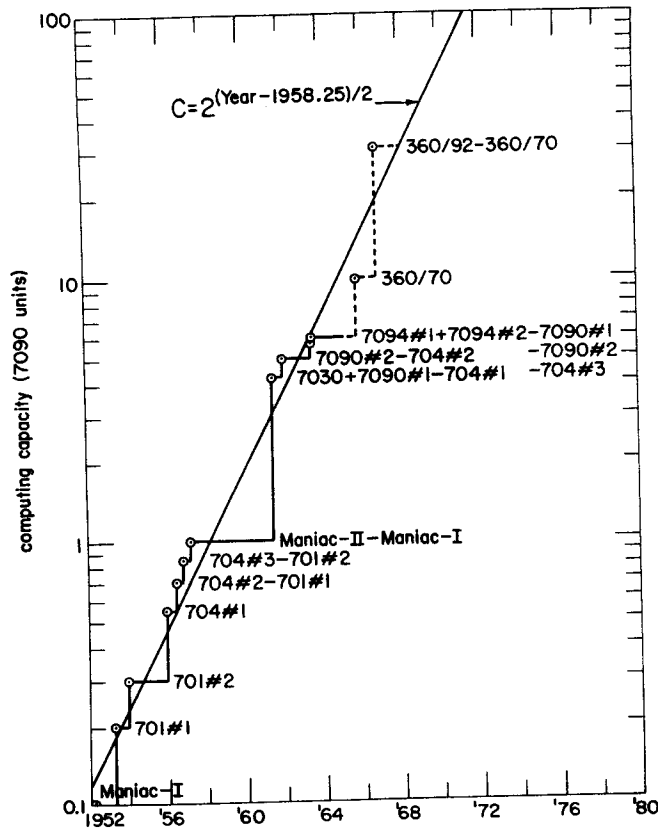


Fig. 2
Growth in Computing-
capacity Requirements

2.4 Some Specific Requirements of Reactor Calculations

Each problem type in Fig. 1 has its own peculiarities and needs. For example, Monte Carlo calculations have only modest storage requirements, but have long running times; diffusion-theory codes have less stringent requirements for running time, but use large blocks of storage; S_n calculations use both large amounts of storage and large amounts of running time. Since the S_n calculations present some of the more difficult problems, these requirements will be examined in detail.

Average running times for S_n calculations of varying dimensionality are shown in Table 1, with the 7030 used as a reference computer. From this table it is possible to gauge the increase in computer speed needed to reduce multidimensional S_n -calculation time so these calculations can be used as part of an optimization study. This is a point that is easily overlooked: the single-calculation time does not provide a correct estimate of the usefulness of a code, since dozens or even hundreds of these calculations are used in the optimization of a nuclear system. Note that each dimensional increase costs about a factor of twenty in running time over the lower dimensions. This includes a factor of two in the number of angular rays to be evaluated, and a factor of about ten for the number of space points used.

Table 1

REPRESENTATIVE RUNNING TIMES FOR S_n CALCULATIONS

Computer Speed X 7030	Execution Time for S_n Calculations		
	1-D, min	2-D, min	3-D, hr
1	10.0	200	67.0
5	2.0	40	13.0
10	1.0	20	6.7
20	0.5	10	3.3
100	0.1	2	0.7

It would require a computer which was twenty times as fast as the 7030 to run two-dimensional problems at the same rate at which we now run one-dimensional problems. This provides us with a guide for estimating the computing power needed for S_n calculations: we gain a dimension when the speed of computers increases by a factor of twenty.

The extremely wide range of running times and storage requirements for S_n calculations is indicated by Table 2. Both absolute and relative numbers of mesh points for which the angular flux must be evaluated are indicated as a function of n , the S_n order, and d , the number of space dimensions. The number of angular rays is given by $M = 2^d n(n+4)/8$; the number of space points is assumed to be 10^{d+1} . Thus the number of mesh points for each energy group is given by

$$P_{n,d} = 10^{d+1} 2^d n(n+4)/8.$$

The calculation time does not increase precisely with the number of mesh points, however, since the additional mesh points increase the convergence rate. The total number of mesh points for which the angular flux must be calculated is given by $GP_{n,d}$, where G is the number of energy groups. For twenty-five energy groups the angular flux array can range in size up to 40,000,000 points! This storage requirement is usually reduced by telescoping the storage, i.e., each new value of the angular flux is stored in the same storage cell as some previous value, so that one of the space variables is estimated and fluxes retained only on the boundaries.

Several conclusions can be drawn from this analysis. First, it is clearly impossible to store the angular flux array in main storage for multi-dimensional S_n problems. This implies that the use of these calculations is solidly tied to efficient auxiliary storage devices with low cost-per-word, fast access time, and high transmission rate. If we were to conjure up such a device, it would have a storage capacity of fifty-million words, an access time of a few milliseconds, a transmission rate approximating the main storage, and a cost of a few cents per word. All of these characteristics have been achieved in separate devices, but not in a single device.

Table 2
 REPRESENTATIVE MESH-POINT REQUIREMENTS
 FOR S_n CALCULATIONS

n	d	M	$P_{n,d}$	$R_{n,d} = P_{n,d}/P_{2,1}$
2	1	3	300	1.00
4	1	8	800	2.67
8	1	24	2,400	8.00
16	1	40	4,000	13.33
2	2	6	6,000	20.00
4	2	16	16,000	53.33
8	2	48	48,000	160.00
16	2	80	80,000	266.67
2	3	12	120,000	400.00
4	3	32	320,000	1066.67
8	3	96	960,000	3200.00
16	3	160	1,600,000	5333.33

A second conclusion from Table 2 is that the general use of three-dimensional S_n calculations awaits the development of a computer which is several hundred times as fast as the computers which are presently available. Although the next generation of computers will reduce the calculation time to where a few three-dimensional calculations of high interest can be done, they will still consume a large fraction of the total computing capacity.

Although the above requirements are sufficiently stringent that they will give job security to computer designers for many years to come, they by no means exhaust the field. Reactor dynamics, for example, present a compounding of the difficulties encountered in criticality calculations. A burnup code is a means of running a controlled series of criticality calculations, with a nuclide-depletion calculation added.

Kinetics calculations have all of the difficulties of criticality calculations, but with the effects of time dependence added. It would seem unlikely that the computing industry could produce a computer in the next 20 yr which would exhaust the requirements of reactor calculations.

3. RECENT DEVELOPMENTS IN COMPUTER DESIGN

3.1 The New Computers

The computers considered here will be those whose first delivery date is scheduled in 1964 through 1967 and whose internal speed is at least equal to the 7030. Nine computers fall into this category. Table 3 summarizes some of the pertinent information about these computers.

Table 3

THE NEW HIGH-SPEED COMPUTERS

Manufacturer	Computer	Approximate Monthly Rental, \$ ^{a)}	Approximate Speed X 7030 ^{b)}	Approximate Delivery (first)
CDC	3800	50,000	1	1/66
CDC	6600	80,000	6	9/64
CDC	6800	85,000	20	7/67
GE	635	55,000	1	11/64
IBM	360/62	58,000	1	9/65
IBM	360/70	80,000	2	9/65
IBM	360/92	142,000	20	9/66
PHILCO	213	78,000	2	9/65
UNIVAC	1108	45,000	2	8/65

^{a)}Rentals vary by a factor of two, depending on configuration. These values are taken from Adams Associates, "Computer Characteristics Quarterly (Mar)."

^{b)}These speeds are taken from LASL tests.

The speeds given in the table are subject to several qualifications: (1) they are based on large scientific calculations; (2) there is considerable variation in relative computer speeds even within this class of calculations; (3) except for the 6600, the values given are manufacturer's estimates; (4) the values given are for single-processor systems, even where multi-processor systems are offered.

The first-delivery dates in Table 3 are also approximate, as some organizations have found the hard way!

Several facts of interest for reactor calculations are immediately apparent from this table. First, computers will soon be available which exceed present computer speeds by a factor of twenty. This is a rather important number in view of the discussion in Section 2.4, since this means that we will gain a dimension in reactor calculations with the delivery of these faster systems. Second, computer rentals and purchase prices have not significantly increased; if anything, they have decreased. Since computer speeds have increased, this means that more problems can be run for the same amount of money that was possible previously, or, put another way, the cost per problem will be considerably lowered by the new computers. Third, IBM is getting a significant amount of competition in the high-speed computer market. Since IBM is not continuing the 7090-series of computers, and the 7030 sales were fairly small, the IBM hold on the high-speed market has been broken. 7094 users are faced with reprogramming, regardless of whether they stay with IBM or go to another manufacturer, so many users are for the first time taking a hard look at other manufacturers. This probably accounts to some extent for the improved price-performance structure being offered to the customers for the new computers.

3.2 The Trend in Computer Speeds

Figure 3 shows computer speeds measured in 7030 units plotted against the first delivery date of the computer. There is a double trend in this development. The lower line shows the general speed trend for the IBM-7090-series, as well as its predecessors and competitors. The upper line is an ultra-high-speed development, started by the 7030, which will apparently cause the termination of the lower line. The cluster of computers at the end of the lower line are all capable of multiprocessing operation; where this option is exercised, these computers fall even closer to the upper line. Since ultra-high speed is crucial to many types of reactor calculations, we can all view this accelerated development of faster computers with a great deal of satisfaction. In fact, the research organizations represented here today probably had a considerable influence on the development.

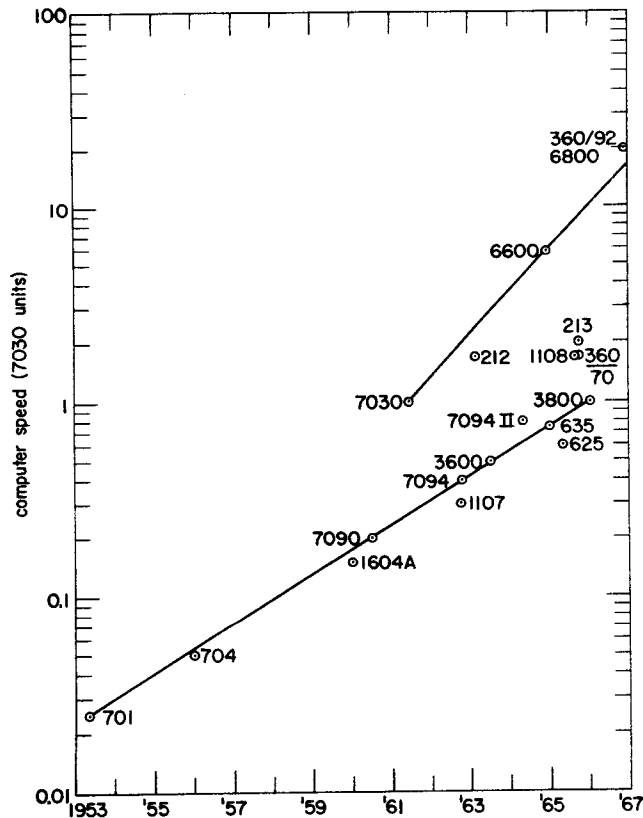


Fig. 3
Increase in Available
Computing Speed

These trends cannot be projected indefinitely into the future, of course, although it will be interesting to see how far and at what rate the development will go. The length of the transmission lines in computers is already becoming a crucial factor in achieving speeds; after all, nature will send information no faster than about one foot per nanosecond. One way to overcome this limitation is to design smaller computers, so that the signals need not travel so far. Another is the use of parallel circuits, such that many signals are being transmitted at once, i.e., to break the single transmission time into many shorter, parallel transmission times. The

first of these methods is being employed in the microminiaturization-packaging techniques employed in integrated-circuit technology. The second is being employed in multiprocessing designs. The combination of smaller circuits with multiprocessing systems will allow faster computers to be developed for some time.

3.3 The Trend in Computer Costs

Figure 4 shows a 12-yr trend in the cost of running a problem. As a figure of merit for problem cost, we have taken the monthly rental divided by the speed of the computer. Thus, if rentals are equal but one computer is twice as fast as another, twice as many problems can be run for the same amount of money on one computer as compared to the other. This implies that the cost per problem is cut in half. This figure of merit needs to be qualified by a factor which determines the percent of the computing capacity which can be profitably used. The figure of merit used here is most meaningful to users who can saturate the capacity of a computer with useful work. Absolute costs become important for those organizations which can use only a fraction of the time they have available on a computer. For example, a 360/70 has a cost per problem which is about four times the cost per problem for the 360/92; if the capacity of the 360/92 can be fully occupied, then the 360/92 is a much better buy. However, if the 360/70 will satisfy the requirements, the absolute cost becomes dominant, and a factor of two can be saved in problem cost.

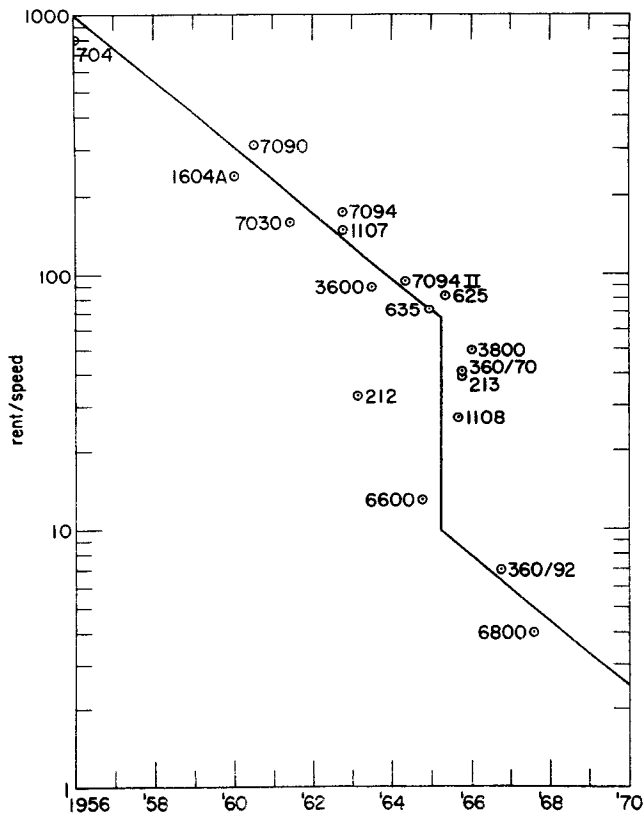


Fig. 4
Trend in Problem Cost

Several startling facts are apparent from Fig. 4. The first is that it takes a three-cycle semilog plot to show the decrease in computer costs in a 12-yr period. Let's hope that trend continues! The second is that the year 1965 will see a sharp discontinuity in the trend of computer costs. For whatever reasons, we will suddenly be able to buy computing power in 1965 and 1966 which is cheaper by a factor of ten than it was 2-3 yr ago. We have attempted to discover the reason for this by discussing the new price structure with some leading manufacturers, and they give two main reasons for this dramatic development: competition and integrated circuits. The competition has developed largely because IBM has had to let go of a market of several hundred large-scale users in the 7090-series, and their competitors have done some fine work in producing competitive machines. Integrated-circuit technology could be seen on the horizon as early as 1959, but is just now making its impact felt on the industry. These circuits are cheaper to manufacture by about a factor of four, they are more reliable, and because their manufacturing processes can be automated, they allow more firms to compete for this market. The ease with which a product can be produced strongly influences the competition.

A question which is frequently considered about computer systems concerns the relative merit of several small systems compared to one large, fast system. Figure 5 is a log-log plot of cost per problem versus the speed of the computer, which can be used in the analysis of this question. This plot shows that it is in general true that the faster the computer is, the less it costs to run a problem on it. As with Fig. 4, this needs to be qualified with the assumption of total use. A least-squares fit of this data gives

$$C = 65S^{-0.83},$$

where C is cost per problem in arbitrary units, and S is the speed of the computer in 7030 units. In other words, the cost of running a problem on a computer is inversely proportional to the 0.8 power of the speed of the computer. For example, the 360/92 is about nine times as fast as the 360/70, but the rental is only higher by a factor of two; thus, it would cost about four to five times as much to use nine 360/70's to do the work of the 360/92, if questions other than cost are assumed to be equivalent for the two systems.

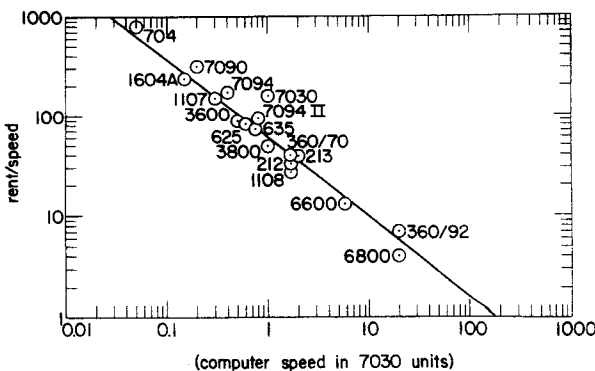


Fig. 5
Problem Cost vs.
Computer Speed

The trend to faster computers, then, carries with it the inherent implication of lower cost per problem. It is surprising to find that a problem which cost \$800 to run on the 704 can be run on modern computers for \$8!

3.4 The Trends in Computer Storage Systems

Computer storage systems can be classified in several ways; in Table 4 they are classified by their access mechanism as being (1) random access (e.g., magnetic cores), (2) cyclic access (e.g., magnetic disks), and (3) serial access (e.g., magnetic tape). The figures given in Table 4 are area prices only and do not reflect any particular systems. Classified by their functions, storage systems are used for (1) main storage (which communicates directly with the Central Processing Unit) and (2) auxiliary storage (which serves both as an extension of main storage and for input-output communications).

Table 4
COMPARISON OF STORAGE SYSTEMS

Storage System	Approximate Cost/Word, \$	Approximate Access Time	Approximate Transmission Rate, kc	Reliability	Removable?
Random access					
· Fast	20.00	1/2-1 μ sec	10,000	Very high	No
· Bulk	2.00	5-10 μ sec	1,000	Very high	No
Cyclic access					
· Drum	0.20	5-20 msec	1,000	High	No
· Disk	0.02	5-200 msec	1,000	High	No a)
Serial access					
· 1-in. tape	0.00002 b)	5 msec-5 min	200	Medium	Yes
· 1/2-in. tape	0.00002 b)	5 msec-5 min	100	Low	Yes

a)Some manufacturers have developed removable disk packs.

b)This price does not include the cost of the transport.

Main storage, which is the most critical in the computer's hierarchy of storage, has imposed restrictions on reactor calculations largely because the standard capacity has been limited to 32K (K = 1024). This restriction will be lifted considerably by the new computers, which have a standard capacity of 128K, or 131,072 words. The addressing capability of the new computers will allow expansion of directly addressed storage to twice that amount, or 262,144 words, and some computers can address up to two-million words. However, the cost of computer storage systems is inversely proportional to their access time, and million-word main storages are not yet practical, even if they are possible: at \$20 per word, a million-word main storage would not normally pay for itself.

Since the delivery of the first 704, in January 1956, the cycle time of computer main storage has decreased by a factor of 50--from 12 μ sec in the 704 to 250 msec in the 6800. The effective access time of storage

systems has decreased even further, however, since all modern main-storage systems consist of many independent boxes which have interlaced addresses, so that the probability of a conflict in access requests is considerably decreased. The 6800, for example, has 32 independent boxes of storage, so the probability of random requests going to the same box is rather small.

For many years there has been a technology gap between the devices used for main storage and those used for auxiliary storage--a gap of three orders of magnitude in access time and a similar, but inverse, gap in cost. This gap is now being filled by the introduction of a type of random-access storage which has speed and cost which are lower by a factor of ten than the main storage. This is usually referred to as "bulk storage." The advantage of bulk storage is that it can be addressed as an extension of main storage; the disadvantages include its cost (100 times the cost of disk) and its limited capacity compared to disk (defined both by its cost and the addressing limitation). For very large storage requirements, bulk storage at \$1 or \$2 per word is clearly too expensive to pay its way. Its usefulness will depend on those applications which require an intermediate-size storage, say 500,000 words, for which the access time rather than the transmission rate is crucial. This is a specialized requirement, and the applications of bulk storage are not yet well developed.

One of the most far-reaching developments in storage systems is the design of high-performance, low-cost disks. Whereas tapes have been used previously for auxiliary storage, they are rapidly being replaced by disks, both in software and in applications. Disks have several advantages over tapes: (1) they are much more reliable, (2) they have larger storage capacity, (3) they have higher transmission rates, and (4) they have lower positioning times. These advantages have led to entirely new concepts in operating systems as well as the ability to handle efficiently very large data arrays in scientific applications. There is a world of difference between a computer with 32K of main storage using tapes for auxiliary storage, and a computer with 128K of main storage using disks for auxiliary storage. This combination will allow large multidimensional reactor calculations to be programmed with very little more difficulty than was previously attached to one-dimensional calculations.

Associative storage, also called "content addressable storage," is a new approach to the addressing problem. Whereas the traditional method of referring to the contents of main storage is by an explicit address, associative storage is referred to by its contents. The reference quantity, called a key, is simultaneously compared with the contents of every cell in the storage, and the cell which matches the key is then read out. In the most general case, the key can refer to any part of the storage word. The most obvious application for associative storage is for applications which depend on "table look-up" procedures. An identifier is stored with the

data, and it is then not necessary to search for the data, since readout is instantaneous upon presenting the identifier to the storage. Tables of the form $[\theta, f(\theta)]$ could be stored, and instantaneous access could be obtained from $f(\theta)$ upon presenting the argument θ . This could have important applications in Monte Carlo codes which require functions of random numbers. Associative stores are several times as expensive as directly addressed stores, and the need for the associative store is specialized, so very large capacities of this type will probably not be used. However, a few thousand words of associative storage for tables would be useful in many reactor calculations.

3.5 Trends in Computer Organization

Demands for increased speed and reliability in computers have led to several forms of parallel computer designs, called "multiprocessing." The term multiprocessing implies that more than one computer unit is active at any given instant in time. We will use the term explicit multiprocessing to refer to those designs in which the Central Processing Unit (CPU), the Storage Units, and the Input-output Control Units are separable--physically, logically, and electronically. Figure 6 illustrates this design. In these designs, there may be several CPU's, several completely independent Storage Units, several Input-output Control Units, and more of each can be added to meet computing, storage, and input-output requirements. The advantages of the explicit multiprocessing designs lie in their modularity and reliability. The computing, storage, and input-output capacity of the machine can be tailored to fit the needs very precisely, and if the needs increase, more units can be added. Computers of this design almost never go off the air completely, since all units are duplicated. The GE-600 series computers, the Philco 213, and the Univac 1108 are examples of computers using this design.

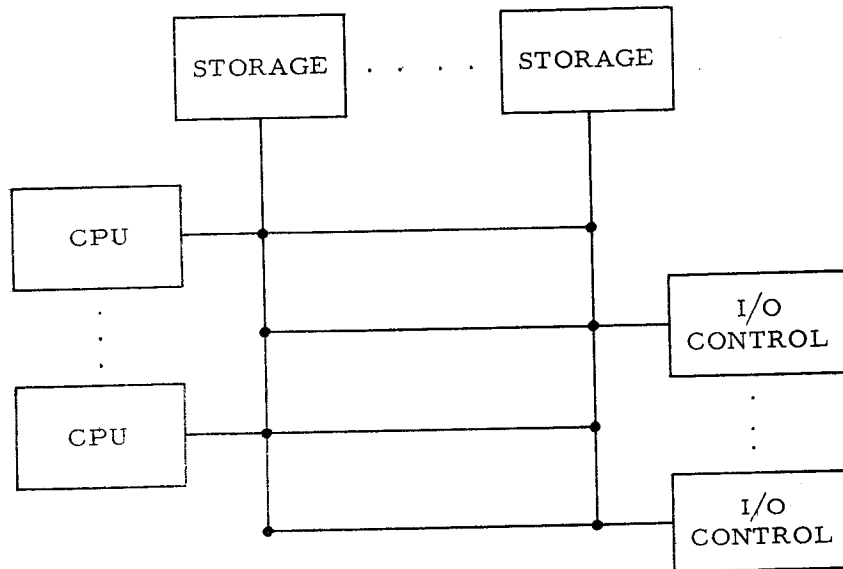


Fig. 6. Explicit Multiprocessing Design

Multiprocessing is used in another form in the CDC-6000 series and the IBM-360/92 (see Figs. 7 and 8); we will refer to this form as implicit multiprocessing. In these designs there is only a single CPU, but there are many independent logical units within the CPU which operate concurrently. Instead of a single arithmetic unit, for example, there are many independent arithmetic units within the CPU. Instead of a single instruction stack, there may be an instruction stack for floating-point, another for fixed-point, and another for branching instructions, and another for store instructions--and each of these may have their own operand registers. It appears to be easier to obtain high speeds with implicit multiprogramming than with explicit multiprogramming; accompanying the higher speeds is an inherent lower cost per problem.

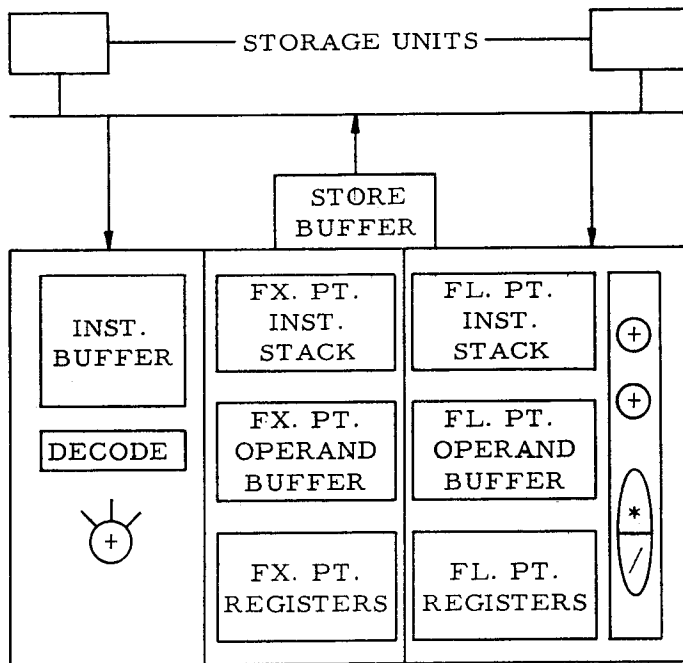


Fig. 7. Implicit Multiprocessing Design

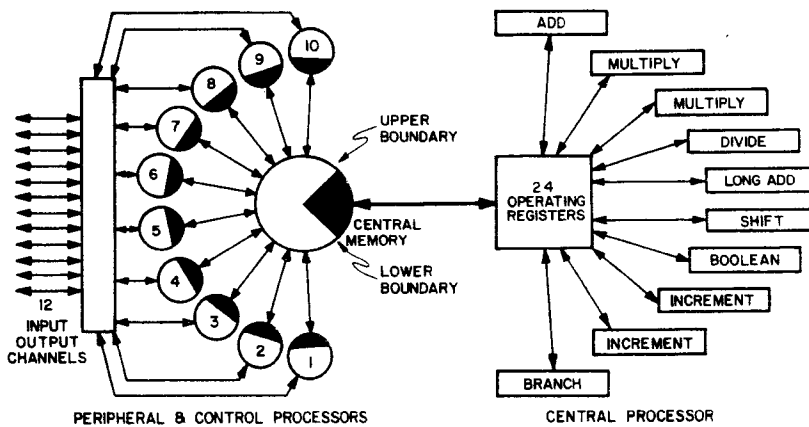


Fig. 8. Block Diagram of 6600

The logical extension of multiprocessing is found in highly parallel designs, such as the Westinghouse Solomon-II. Instead of having just two or three CPU's, this design offers up to 1024 CPU's, each with their own storage and arithmetic unit. The arithmetic units are relatively simple, and individually are not particularly fast, but when 1000 add-instructions are executed simultaneously, the effective instruction time is lowered by three orders of magnitude! The difficulty in this design lies in the programming. In order to obtain high performance with this design, it is necessary to keep a large number of the CPU's busy, and not every mathematical method is amenable to simultaneous operations on all mesh points. For those methods which can take advantage of the design, the gain is very great; the particle-in-cell method in hydrodynamics and elliptic partial differential equations in general are the most immediate areas of application. This design, however, falls into the category of special-purpose computers, and is unlikely to replace conventional designs.

3.6 Trends in Computer Software

Any discussion of software which pretends to be at least partially complete must divide software into two general categories: 1) programming languages, and 2) operating systems.

Programming languages are designed to make life easier for the programmer--at least in theory. There can be little question that great strides have been made in this area since the early days of internally stored-programmed computers when programmers worked in octal machine codes stored in absolute locations. The next development was relative and regional assembly programs which allowed operation mnemonics and symbolic data references. It is difficult to imagine the immense amount of manpower which would be required to program the entirety of today's volume of problem codes by such techniques.

The advent of FORTRAN and other automatic-coding languages provided a breakthrough in allowing the problem proposer to develop his own problem codes without requiring the services of an expert. In terms of computer efficiency, however, this is not an unmixed blessing. An appreciable amount of machine time is used in compiling, retrying incorrect codes, and simple misuse of the language itself. Often the language is not capable of permitting efficiency even though the programmer is aware of shortcuts and other methods of obtaining efficiency. For this reason, some compilers allow supplemental use of "longhand" coding. There are drawbacks to this technique, however, if one wishes to run his code on an incompatible computer. Longhand coding still is the only efficient coding technique if one is interested in maximum computing speed and flexibility. The question is one of programmer efficiency versus computer efficiency.

Continuing efforts are being made to improve the efficiency and flexibility of programming languages. It is somewhat surprising, however, that there are only a few small, scattered efforts being made in developing languages which are more convenient for the programmer. There are only trivial technical obstacles which prevent such refinements as superscripts, subscripts, "displayed" division, and mathematical function symbolism being used without artificial, cumbersome, and "unnatural" notational devices.

At present there seems to be a philosophical contest between one general all-encompassing language and numerous more-or-less specialized languages. In the latter case we have, for example, FORTRAN, ALGOL, and others for scientific users; COBOL, COMTRAN, and others for business users; and APT for industrial users. It is not clear how this contest will end, but it appears that computer manufacturers are making every effort to reduce the number of languages to one, if possible.

IBM has for some time been developing NPL, New Programming Language, as a possible future successor to FORTRAN. Many programmers are questioning the desirability of such a replacement in the immediate future. It would appear that the final answer to this question will depend in large measure upon the compiling speed and object code efficiency achievable with NPL. There are some who doubt this will ever be possible. Many do not expect this to occur within the next three to five years. The reaction to an administrative decision to discontinue FORTRAN within the near future would likely be most interesting to witness!

The development of elegant(?) debugging techniques over the last decade has not kept pace with the relatively elegant development of the programming languages. The feasibility of remote consoles and multiprogrammed operating systems will doubtless accelerate developmental efforts in more convenient, faster response-time debugging methods. To some degree, the success of such methods will depend upon the user's ability not to waste excessive amounts of computer time.

Programming languages for use with multiprocessor computers would not need to be significantly different from uniprocessor languages. If, however, one program used more than one processor, it would be necessary to augment the language for control purposes. The maximum output of a multiprocessor system would normally be obtained if each problem run limited itself to one computer module.

Parallel processors of the SOLOMON type would pose difficulties in problem formulation in many cases due to boundary conditions, activating and de-activating various processors, conditional branching, and so forth. Such difficulties would give rise to special language statements. For many types of problems, new mathematical formulations would, in addition, be required.

COMPUTER IMPLICATIONS FOR REACTOR CALCULATIONS

With regard to operating systems there appears to be a definite trend towards multiprogramming (or multitasking) supervisors. In a multiprogramming system, one computation may be interrupted in order to give the computer another calculation while input-output is being performed for the first problem or because the second problem is of higher priority or because an allotted amount of time has expired which is cyclically given to a set of problems. Improved computer efficiency should therefore result unless the "overhead" is too high. Another advantage is that the computer is available to more problems "simultaneously"--some of which might be debugging runs which would otherwise have to wait for problems to be completed.

A trend which has been apparent for some time and which greatly improves "turnaround" time is that of parallel execution of input-output on-line. At the present time a single 1100-line-per-minute printer attached to our 7030 usually has a problem's output available within 5 min (rarely as long as 15 min) after a problem's completion. Before the acquisition of this printer, output was written on a tape to be printed on a 1401, and the total delay would usually be from 1-2 hr. It is anticipated that our next computer will have three printers--two of which will be used in this fashion. Parallel input methods also reduce turnaround time.

The problem of higher volumes of output from future faster computers is a question of no little concern. The assistance of the programmer will be required to reduce the amount of output per unit of computation if reasonable and practical amounts are to be generated. One technique which would appear feasible would be for the user to store detailed results on a large storage device and request only "summary" type results to be printed. If, then, within 24-48 hr (or ?), the user needs some of this stored information, it could be provided. Following that, the stored information could be destroyed or transferred to tape or removable disk. The aim would be to print only that information which would actually be used.

It also appears that there is a trend towards greater use of graphical and plotting devices. The information content per page with such devices can often be extremely high. A variety of such graphical devices, as well as film viewers and printers, are currently available.

Software trends, in general, would tend to indicate greater convenience in the area of programming languages. On the other hand, it would appear that some amount of inconvenience may be imposed on the programmer because of more complicated operating systems which attempt to maximize computer efficiency. This, however, should result in improved service and turnaround time.

4. CONCLUSION

To understand any rapidly developing technology it is important to analyze trends rather than current status. This is especially true for those who are responsible for influencing the selection and design of computing equipment. The trends toward higher speeds, lower costs, parallel organization, and larger capacity memories are all beneficial to the scientific users of computing equipment. There is a parallel trend, however, which should be viewed with some concern: The combining of commercial and scientific computing requirements into a single design. This could be beneficial since it might contribute to lower overall costs for the manufacturers and users. The scientific user must beware, however, that scientific requirements do not get diluted in the mixture, such that his needs are overlooked. Scientific users should analyze and document their requirements, and then transmit these to the manufacturers in order to influence the design of future computers. It may be of sufficient importance for the Reactor Mathematics and Computation Division to establish a committee on computer design, whose purpose would be to summarize the continuing requirements of reactor calculations and present these to the manufacturers.

References

1. Roos, B. W. and W. Sangren, Advances in Nuclear Science and Technology, "Some Aspects of the Use of Digital Computers in Nuclear Reactor Design," Academic Press, New York (1964).
2. Sangren, W., Digital Computers and Nuclear Reactor Calculations, John Wiley and Sons, New York (1960).
3. Sangren, W., The Role of Digital Computers in Nuclear Design, Nucleonics, p. 56 (May 1957).
4. Nather, V. and W. Sangren, Codes for Reactor Computations, Nucleonics (Nov 1961).
5. Roos, B. W. and W. Sangren, Codes for Reactor Computations, Nucleonics (Aug 1962).
6. Nather, V. and W. Sangren, Abstracts--Nuclear Reactor Codes, Commun. Assoc. Computing Mach. (Jan 1959).
7. Nather, V. and W. Sangren, Abstracts--Nuclear Reactor Codes, Commun. Assoc. Computing Mach. (Jan 1960).
8. Radkowsky, A. and R. Brodsky, A Bibliography of Available Digital Computer Codes for Nuclear Reactor Problems, AECU-3078 (1955).
9. Selengut, D. S., Digital Computers and Nuclear Design, HW-59679 (1959).

ANL-7050
Mathematics and
Computers
(TID-4500, 43rd Ed.)
AEC Research and
Development Report

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60440

PROCEEDINGS OF THE
CONFERENCE ON THE
APPLICATION OF COMPUTING METHODS
TO REACTOR PROBLEMS

May 17-19, 1965

Margaret Butler
General Chairman

Sponsored by
Argonne National Laboratory
European Nuclear Energy Agency
Mathematics and Computation Division,
American Nuclear Society

August 1965

Operated by The University of Chicago
under
Contract W-31-109-eng-38
with the
U. S. Atomic Energy Commission