Chapter 22

Design of the B 5000 system[1]

William Lonergan / Paul King

Computing systems have conventionally been designed via the 'hardware' route. Subsequent to design, these systems have been handed over to programming systems people for the development of a programming package to facilitate the use of the hardware. In contrast to this, the B 5000 system was designed from the start as a total hardware-software system. The assumption was made that higher level programming languages, such as ALGOL, should be used to the virtual exclusion of machine language programming, and that the system should largely be used to control its own operation. A hardware-free notation was utilized to design a processor with the desired word and symbol manipulative capabilities. Subsequently this model was translated into hardware specifications at which time cost constraints were considered.

Design objectives

The fundamental design objective of the B 5000 system was the reduction of total problem through-put time. A second major objective was facilitation of changes both in programs and system configurations. Toward these objectives the following aspects of the total computer utilization problem were considered:

Statement of problems in higher-level machine-independent languages; efficiency of compilation of machine language; speed of compilation of machine language; program debugging in higher-level languages; problem set-up and load time; efficiency of system operation; ease of maintaining and making changes in existing programs, and ease of reprogramming when changes are made in a system configuration.

Design criteria

Early in the design phase of the B 5000 system the following principles were established and adopted:

Program should be independent of its location and unmodified as stored at object time; data should be independent of its location; addressing of memory within a program should take advantage of contextual addressing schemes to reduce redundancy; provisions should be made for the generalized handling of indexing and subroutines; a full complement of logical, relational and control operators should be provided to enable efficient translation of higher-level source languages such as ALGOL and COBOL; program syntax should permit an almost mechanical translation from source languages into efficient machine code; facilities should be provided to permit the system to largely control its own operation; input-output operations should be divorced from processing and should be handled by an operating system; multi-programming and true parallel processing (requires multiple processors) should be facilitated, and changes in system configuration (within certain broad limitations) should not require reprogramming.

[1]*Datamation,* vol. 7, no. 5, pp. 28–32, May, 1961.

System organization

The B 5000 system achieves its unique physical and operational modularity through the use of electronic switches which function logically like telephone crossbar switches. Figure 1 depicts the basic organization of the system as well as showing a maximum system.

Master control program

A master control program will be provided with the B 5000 system. It will be stored on a portion of the magnetic drum. During normal operations, a small portion of the MCP will be contained in core memory. This portion will handle a large percentage of recurrent system operations. Other segments of the MCP will be called in from the magnetic drum, from time to time, as they are required to handle less frequently-occurring events, or system situations. Whenever the system is executing the master control program, it is said to be in the Control State. All entries to the Control State are made via 'interrupts.' A special operation is provided, which can only be executed when the system is in the Control State, to permit control to return to the object program it was executing at the time the 'interrupt' occurred.

The following are a few typical occurrences which cause an automatic 'interrupt' in the system: An input-output channel is

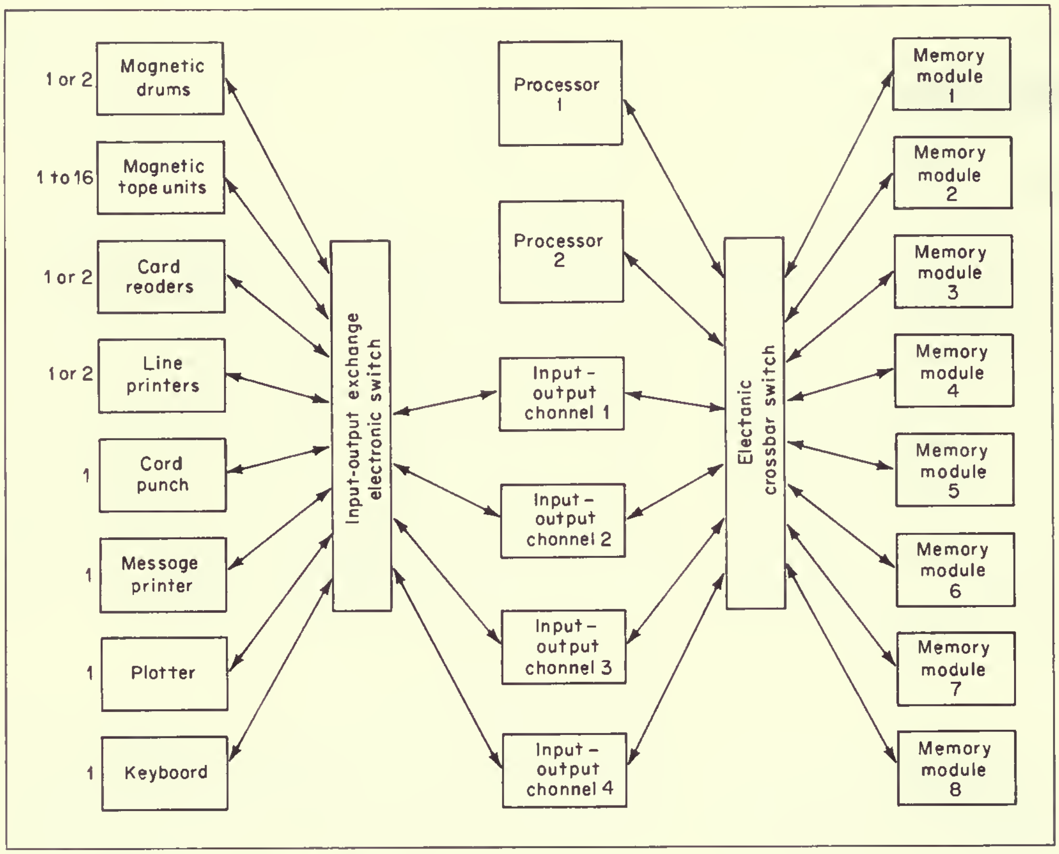


Fig. 1. Organization of the B5000 system.

available, an input-output operation has been completed or an indexing operation was attempted which violated the storage protection features built into the system.

In addition to processing interrupt conditions, the master control program handles fundamental parts of the total system operation such as the initiation of all input-output operations, tanking of input-output areas when required, file control, allocation of memory, scheduling of jobs (priority ratings, system requirements of each object program, and the present system configuration are considered), maintenance of an operations log and maintenance of a system description.

Operating modes

The B 5000 can either operate with fixed-length words or with variable-length fields. These two modes of operation are called the word mode and the character mode. For certain operations, a processor operating on words is most desirable and for other operations, a variable field length mode of operation is most desirable. By combining both abilities in one processor, a processor can operate in the mode most desirable for the operation at hand. In a B 5000 system, it is even possible for one processor to be operating in the word mode and the other in the character mode.

When operating in the word mode, a standard format for the data word is used as illustrated in Fig. 2.

Note that the standard word is an octal floating point word. However, the mantissa is treated as an integer rather than as a fraction (heretofore the reverse has been common practice). This provides two benefits: first, an integer has the same internal representation as its unnormalized floating point correspondent; and, second, the range of numbers that can be expressed, rather than being from 8^{+64} to 8^{-63}, is 8^{+76} to 8^{-51}. The first feature eliminates

F	SE	EXPONENT	SO	Integer Part

F—Flag (1 bit)
SE—Sign of Exponent (1 bit)
Exponent (6 bits)
SO—Sign of Operand (1 bit)
Integer Part (39 bits)

Fig. 2. Data word — word mode.

the need for fixed-to-floating point conversion; integers and floating point numbers can be mixed in arithmetic calculations. The second expands the range where trouble with range is most often encountered, namely, in numbers with extremely large magnitude.

The flag serves a dual purpose. The function of the flag depends on how the program references the data word. If the data word is a single variable and not an element of an array, the flag identifies the word as being operand, that is, a data word. If the word is an element of an array, the flag may be used to identify this particular element as an element of data which is not to be processed by the normal program (for example, a boundary point in mesh calculations).

When operating in the character mode, each data word consists of eight alphanumeric characters as illustrated in Fig. 3. Programs in the character mode can address any character in a word. Fields can start at any position in a word. A processor in a single operation can operate on fields of any length up to 63 characters long; operations on fields of greater length can easily be programmed. For example, two 57 character fields could be compared in a single operation.

There are two instances when the character mode operates with words of the type used in the word mode. Operations are provided in the character mode for converting numeric information in the alphanumeric representation to the standard word type of the word mode and vice versa. In both of these instances, the length of the alphanumeric fields being converted to or from the word mode type of word can be no greater than eight characters long. Again, conversion of fields of greater length can easily be programmed.

The purpose of the word mode is to provide the advantages of high-speed parallel operations, floating-point abilities and the inherent information density possible in a binary machine. In the first case, it is economically feasible to provide parallel operations in a word machine; the cost of parallel operations on variable length fields would be prohibitive. In the last case, a given size memory can contain over twenty percent more numeric information if that information is expressed in binary rather than binary-coded decimal, and over eighty percent more information than can be expressed in six-bit alphanumeric representation.

The purpose of the character mode is to provide editing, scanning, comparison and data manipulative abilities (although addition and subtraction are also provided). The type of editing facilities provided obviate the need for the artificial "add-shift-extract-store" type of editing. For example, operations are provided for generalized insertion of editing symbols (such as blanks, decimal points, floating dollar signs, etc.) and for the substitution or suppression of any unwanted characters. For those interested in the new area of Information Processing Languages, the character mode is particularly well suited to list structures.

Program organization

Programs in the B 5000 are composed of strings of syllables. A syllable is the basic unit of the program and is twelve bits in length. The term "syllable" is used rather than instruction to distinguish it from conventional single-address or multi-address instructions. Each program word contains four syllables and they are executed sequentially in a left-to-right order within the program word, and sequentially by word. Branching is allowed to any syllable within a word. Before delving into some of the details of the internal operation of the B 5000 processor, it is necessary to discuss stacks, Polish notation, and the Program Reference Table.

The stack

The internal organization of single-address computers forces the wasting of both programming and running time for the storage and recall of the intermediate results in the sequence of computation. The data must be placed into the proper registers and memory cells before the operation can be executed, and their contents must often be completely rearranged before the next operation can be performed. Multi-address computers are constructed to make the execution of a few selected operations more efficient, but at the expense of building inefficiencies into all the rest. Automatic programming aids attack this problem indirectly: they relieve the programmer of the need to laboriously code his

First Character	Second Character	Third Character	Fourth Character	Fifth Character	Sixth Character	Seventh Character	Eighth Character

Fig. 3. Data word — character mode.

way around machine design, but they still must provide object coding to accomplish the storage and recall functions. In brief, conventionally designed computers, with or without automatic programming aids, require the wasteful expenditure of programming effort, memory capacity, and running time to overcome the limitations of their internal organization.

The problem is attacked directly in the B 5000 by incorporation of a "pushdown" stack, which completely eliminates the need for instructions (coded or compiled) to store or recall intermediate results.

In a B 5000 processor, the stack is composed of a pair of registers, the A and B registers, and a memory area. As operands are picked up by the programs, they are placed in the A register. If the A register already contains a word of information, that word is transferred to the B register prior to loading the operand into the A register. If the B register is also occupied by information, then the word in B is stored in a memory area defined by an address register S. Then the word in A can be transferred to B and the operand brought into the A register. The new word coming into the stack has pushed down the information previously held in the registers. As each pushdown occurs, the address in the S register is automatically increased by one. The information contained in the registers is the last information entered into the stack; the stack operates on a "last in–first out" principle. As information is operated on in the stack, operands are eliminated from the stack and results of operations are returned to the stack. As information in the stack is used up by operations being performed, it is possible to cause "pushups," i.e., a word is brought from the memory area addressed by the S register, and the address in the S register is decreased by one.

To eliminate unnecessary pushdowns and pushups, the A and B registers both have indicators used for remembering whether the registers contain information or are empty. When an operand is to be placed in the stack and either of the registers is empty, no pushdown into memory occurs. Also, when an operation leaves one or both of the registers empty, no automatic pushup occurs.

Polish notation

The Polish logician, J. Lukasiewicz, developed a notation which allows the writing of algebraic or logical expressions which do not require grouping symbols and operator precedence conventions. For example, parentheses are necessary as grouping symbols in the expression $A(B+C)$ to convey the desired interpretation of the expression. In the expression $A+B/C$, the normal interpretation is $A+(B/C)$, rather than $(A+B)/C$, because of the convention that the / operator is of higher precedence than the + operator. The right-hand Polish notation used in the B 5000 is based on placing the operators to the right of their operands: $A + B$ becomes $AB+$ in Polish notation. $A+B+C$ can be written either as $AB+C+$, or as $ABC++$. In the expression $ABC++$, the first + operator says to add the operands B and C. The second + operator says to add A to the sum of B and C. Returning to the first examples above, $A(B+C)$ can be written as $BC+A\times$ or $ABC+\times$ in Polish. The second example is written as $BC/A+$ or $ABC/+$. The extension of Polish notation to handle equations is shown in the following example:

Conventional notation $Z=A(B-C)/(D+E)$

Polish notation $ABC-\times DE+/Z=$

The stack in use

To illustrate the functioning of the stack, two simple examples are shown in Figs. 4 and 5. In the examples, the letters P, Q and R represent syllables in the program that cause the operands P, Q, and R to be picked up and placed in the stack. The symbols + and × represent syllables that cause the add and multiply operations to occur. The two examples represent different ways of writing $P(Q+R)$ in Polish notation. The first example in Fig. 4 does not require pushdowns or pushups. The second example, shown in Fig. 5, requires a pushdown in the execution of the syllable R, and a pushup in the execution of the syllable ×. The columns in the table represent the contents of the various registers after execution of the syllable listed in the first column.

Independence of addressing

One of the goals set in the design of the B 5000 was to make the programs independent of the actual memory locations of both the program itself and the data, in order to provide really automatic

Polish Notation QR+P×

Syllable Executed	*Contents of*	
	Register A	*Register B*
Q	Q	Empty
R	R	Q
+	Empty	R+Q
P	P	R+Q
×	Empty	P(R+Q)

Fig. 4

Polish Notation PQR+×

Syllable Executed		*Contents of*			
		Register A	*Register B*	*Register S*	*Cell 101*
P		P	Empty	100	–
Q		Q	P	100	–
R	Pushdown	Empty	Q	101	P
	Execute	R	Q	101	P
+		Empty	Q–R	101	P
×	Pushup	Q–R	P	100	–
	Execute	Empty	P(Q–R)	100	–

Fig. 5

program segmentation. Through automatic program segmentation, it is possible to have program size practically independent of the size of core memory. The systems analyst or programmer intending to do multi-processing is then no longer faced with the difficult task of planning what jobs are to be run together in order that system storage capacities are not exceeded.

In achieving independence of addressing, a solution requiring large contiguous areas of memory was not deemed satisfactory. Each segment of the program and each data area should be completely relocatable without modification to the program. It is then possible to load all the segments of a program or programs onto the drum at load time and call in the segments to any available space in core memory as needed during run time. If some segment of a program is overlaid by a subsequent segment of a program, the segment of the program destroyed in core memory is still available on the drum to be called in again if needed.

Due to the very high program densities in the B 5000, the availability of high capacity drum storage on every system and automatic segmentation, a minimum B 5000 system has the capacity for a program or programs equivalent to approximately 40,000 to 60,000 single address instructions. Of course, if an installation normally ran such large programs, the system would very likely not be a minimum system. However, the installation having an occasional need to run very large programs is not prevented from doing so by storage capacity.

Processing speed now becomes a function of the size of core memory. If large programs are run in a system with small core memory, time will be consumed in recalling program segments from drum to core. If the core memory is expanded, less time will be spent in such activity and the program or programs will be speeded up, and no reprogramming is required.

Program reference table

The means of achieving independence of addressing in the B 5000 is called a Program Reference Table (PRT). The PRT is a 1,025 word relocatable area in memory used primarily for storing control words that locate data areas or program segments. There are also control words for describing input-output operations. These control words, called descriptors, contain the base address and size of data areas, program segments and input-output areas. A descriptor specifying an input-output operation also contains the designation of the unit to be used and the type of operation to be performed. Operands may also be stored in the PRT, providing direct access to single values such as indices, counts, control totals, etc.

In the word mode of the B 5000, every item of data is considered to be either a single value or an element of an array of data. If it is a single value, it will be obtained directly by indexing a descriptor contained in the PRT.

Program segments are described by program descriptors. In addition to core base address, the program descriptor contains the location in drum storage of the program segment and an indication if the program segment is currently in core memory starting at the address specified in the descriptor. Entry to a program segment is made via its program descriptor contained in the PRT. If the program segment is in core memory, entry will be made to the program segment. However, when entry is attempted to a program segment whose descriptor indicates that the segment is not in core memory, automatic entry to the Master Control Program will occur and the desired segment will then be brought in from the drum. Notice that in moving from one segment to another, it is not necessary to know whether the segment to be entered is currently in core memory. Branching within a program segment is self-relative, i.e., the distance to jump either forward or backward is specified, not the address to be jumped to.

As a result of keeping all actual addresses of data and program in the PRT, the program itself does not contain any addresses, but only references to the PRT. To specify one of the 1,024 positions in the PRT requires only 10 bits which contributes greatly to the high program density achieved in the B 5000. Since the PRT is relocatable, references to the PRT contained in the program are to relative locations, thus completely freeing the program from any dependence whatsoever on actual memory locations.

Word mode program

The word mode of the B 5000 processor has four types of syllables. The syllable type is distinguished by the two high-order bits of each 12-bit syllable. The types of syllable and the identification bits are:

00—Operator Syllable
01—Literal Syllable
10—Operand Call Syllable
11—Descriptor Call Syllable

The first of these, the operator syllable, causes operations to be performed. The remaining ten bits of the operator syllable are the operation codes. There are approximately sixty different operations in the word mode. For those operations requiring an operand or operands, the processor checks for sufficient operands in the registers; if they are not there, pushups from the stack in memory occur automatically.

The literal syllable is used for placing constants in the stack to be used as operands. The ten bits of the literal syllable are transferred to the stack. This allows the program to contain integers less than 1,024 as constants.

The operand call syllable, and the descriptor call syllable address locations in the program reference table. The purpose of the operand call syllable is to place an operand in the stack; the purpose of the descriptor call syllable is to place the address of an operand, a descriptor, in the stack. There are four situations that arise, depending on the word read from the program reference table.

1 The word is an operand.
2 The word is a descriptor containing the address of the operand.
3 The word is a descriptor containing the base address of the data area in which the operand resides.
4 The word is a program descriptor containing the base address of a subroutine.

For (1), the operand call syllable has completed its action by placing an operand in the stack. The descriptor call syllable will cause the construction of a descriptor of the operand, replacing the operand by the constructed descriptor.

For (2), the operand call syllable then reads the operand from the cell addressed. The descriptor call syllable has completed its action.

For (3), indexing of the descriptor by the item that is now the second item in the stack occurs. For an operand call syllable, the operand is obtained from the indexed address; for the descriptor call syllable, action is complete after the indexing.

In the case of (4), subroutine entry occurs to the subroutine addressed. A word of the three previous types may be left in the registers upon return from the subroutine, in which instance the actions described above will take place, depending upon the type of syllable which initiated the subroutine.

Essentially, the four types of action that occur for an operand call syllable are obtaining an operand directly, indirectly, from an array, or by computation. Sometimes in the use of the call syllables, it is not known which type of action will occur for a particular syllable when the program is created. This is particularly true for call syllables in subroutines.

Programs in the word mode consist of strings of syllables which follow the rules of Polish notation. Variable length strings of call syllables and literal syllables, which place items of information in the stack, are followed by operator syllables which perform their operations on information in the stack.

The indexing features of the B 5000 allow generalized indexing and at the same time provide complete storage protection. Data areas and program segments of different programs may be intermingled, but a program is prevented from storing outside of its data areas. The method of indexing allows any of the 1,024 words of the program reference table to be considered index registers. Multilevel indexing is provided, i.e., indices of arrays can themselves be elements of arrays.

The subroutine control provided in the B 5000 allows nesting of subroutines—even recursive nesting (a subroutine is a subroutine of itself)—arbitrarily deep. Dynamic allocation of storage for parameter lists and temporary working storage simplify the use of subroutines. Storage is automatically allocated and deallocated as required.

Character mode program

In the character mode of the B 5000 Processor, there is only one type of syllable, called the operator syllable. Program segments in the character mode are constructed of strings of these syllables. The character mode is designed to provide editing, formatting, comparison, and other forms of data manipulation. In doing so, the processor uses two areas of memory—the source and destination areas. When a program switches from word mode to character mode, two descriptors containing the base addresses of these areas are supplied. The source area or destination area may be changed at any time during character mode so that the program may act on several areas.

The character mode operator syllable is split into two 6-bit parts; the last part specifies the operation to be performed and the first part specifies the number of times the operation is to be performed. Operations are provided for the transferring, deletion, comparison, and insertion of characters or bits. Also, there are operations which allow the repetition of syllable strings. This is quite useful for complex table look-up operations and for editing information which contains repeated patterns.

Conclusion

The Burroughs B 5000 system has been designed as an integrated hardware-software package which offers such benefits as savings in the memory space required to store equivalent object programs; multi-processing and parallel processing; and running identical programs on systems with different size memories and different system configurations with no loss in individual system efficiency.

References

LoneW61; BartR61; BockR63; CarlC63; MaheR61

Chapter 9

Design of the B 5000 System[1]

William Lonergan / Paul King

Computing systems have conventionally been designed via the "hardware" route. Subsequent to design, these systems have been handed over to programming systems people for the development of a programming package to facilitate the use of the hardware. In contrast to this, the B 5000 system was designed from the start as a total hardware-software system. The assumption was made that higher level programming languages, such as ALGOL, should be used to the virtual exclusion of machine language programming, and that the system should largely be used to control its own operation. A hardware-free notation was utilized to design a processor with the desired word and symbol manipulative capabilities. Subsequently this model was translated into hardware specifications at which time cost constraints were considered.

Design Objectives

The fundamental design objective of the B 5000 system was the reduction of total problem through-put time. A second major objective was facilitation of changes both in programs and system configurations. Toward these objectives the following aspects of the total computer utilization problem were considered:

Statement of problems in higher-level machine-independent languages; efficiency of compilation of machine language; speed of compilation of machine language; program debugging in higher-level languages; problem set-up and load time; efficiency of system operation; ease of maintaining and making changes in existing programs, and ease of reprogramming when changes are made in a system configuration.

Design Criteria

Early in the design phase of the B 5000 system the following principles were established and adopted:

Program should be independent of its location and unmodified as stored at object time; data should be independent of its location; addressing of memory within a program should take advantage of contextual addressing schemes to reduce redundancy; provisions should be made for the generalized handling of indexing and subroutines; a full complement of logical, relational and control operators should be provided to enable efficient translation of higher-level source languages such as ALGOL and COBOL; program syntax should permit an almost mechanical translation from source languages into efficient machine code; facilities should be provided to permit the system to largely control its own operation; input-output operations should be divorced from processing and should be handled by an operating system; multi-programming and true parallel processing (requires multiple processors) should be facilitated, and changes in system configuration (within certain broad limitations) should not require reprogramming.

System Organization

The B 5000 system achieves its unique physical and operational modularity through the use of electronic switches which function logically like telephone crossbar switches. Figure 1 depicts the basic organization of the system as well as showing a maximum system.

Master Control Program

A master control program will be provided with the B 5000 system. It will be stored on a portion of the magnetic drum. During normal operations, a small portion of the MCP will be contained in core memory. This portion will handle a large percentage of recurrent system operations. Other segments of the MCP will be called in from the magnetic drum, from time to time, as they are required to handle less frequently-occurring events, or system situations. Whenever the system is executing the master control program, it is said to be in the Control State. All entries to the Control State are made via "interrupts." A special operation is provided, which can only be executed when the system is in the Control State, to permit control to return to the object program it was executing at the time the "interrupt" occurred.

The following are a few typical occurrences which cause an automatic "interrupt" in the system: An input-output channel is available, an input-output operation has been completed or an indexing operation was attempted which violated the storage protection features built into the system.

In addition to processing interrupt conditions, the master control program handles fundamental parts of the total system operation such as the initiation of all input-output operations, linking of input-output areas when required, file control, allocation of memory, scheduling of jobs (priority ratings, system requirements of each object program, and the present system configuration are considered), maintenance of an operations log and maintenance of a system description.

[1]*Datamation*, vol. 7, no. 5, May 1961, pp. 28–32.

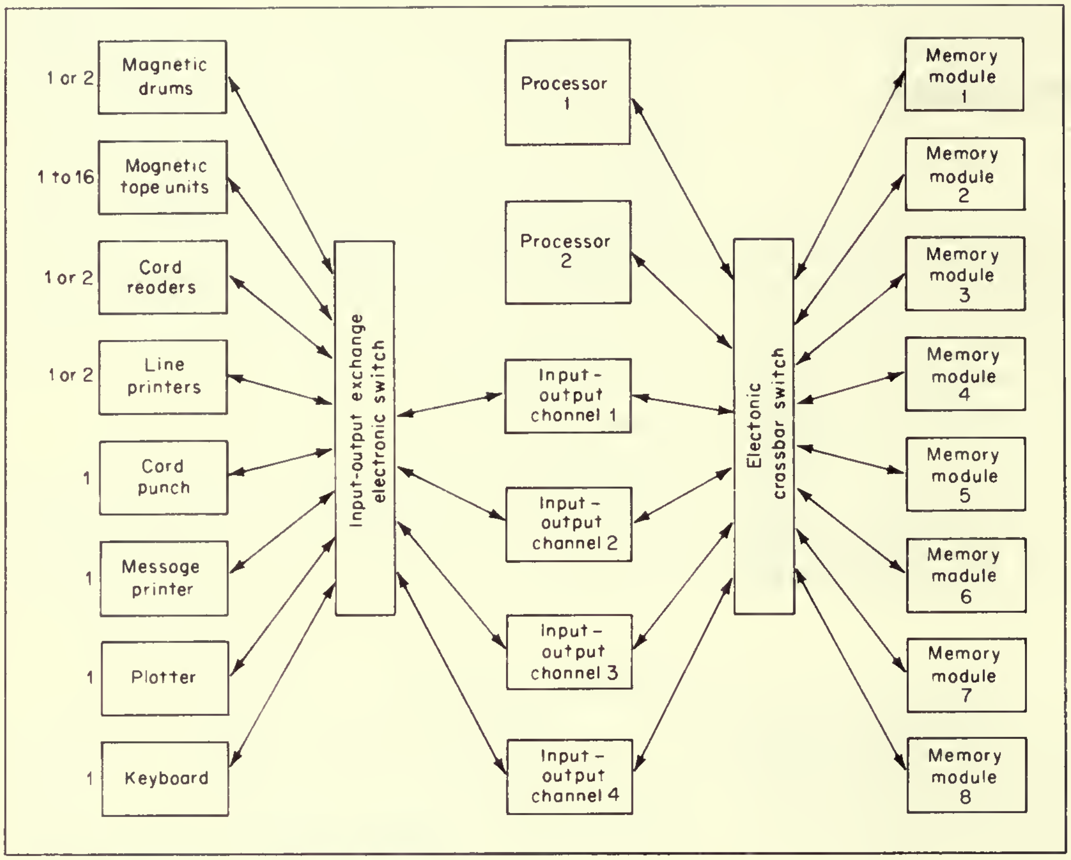


Fig. 1. Organization of the B 5000 system.

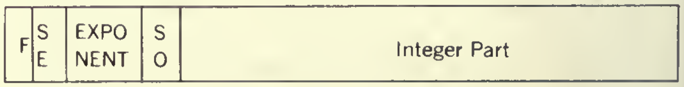
Operating Modes

The B 5000 can either operate with fixed-length words or with variable-length fields. These two modes of operation are called the word mode and the character mode. For certain operations, a processor operating on words is most desirable, and for other operations, a variable field length mode of operation is most desirable. By combining both abilities in one processor, a processor can operate in the mode most desirable for the operation at hand. In a B 5000 system, it is even possible for one processor to be operating in the word mode and the other in the character mode.

When operating in the word mode, a standard format for the data word is used as illustrated in Fig. 2.

Note that the standard word is an octal floating point word. However, the mantissa is treated as an integer rather than as a fraction (heretofore the reverse has been common practice). This provides two benefits: first, an integer has the same internal representation as its unnormalized floating point correspondent; and, second, the range of numbers that can be expressed, rather than being from 8^{+64} to 8^{-63}, is 8^{+76} to 8^{-51}. The first feature eliminates the need for fixed-to-floating point conversion; integers and floating point numbers can be mixed in arithmetic calculations. The second expands the range where trouble with range is most often encountered, namely, in numbers with extremely large magnitude.

The flag serves a dual purpose. The function of the flag depends



F—Flag (1 bit)
SE—Sign of Exponent (1 bit)
Exponent (6 bits)
SO—Sign of Operand (1 bit)
Integer Part (39 bits)

Fig. 2. Data word—word mode.

on how the program references the data word. If the data word is a single variable and not an element of an array, the flag identifies the word as being operand, that is, a data word. If the word is an element of an array, the flag may be used to identify this particular element as an element of data which is not to be processed by the normal program (for example, a boundary point in mesh calculations).

When operating in the character mode, each data word consists of eight alphanumeric characters as illustrated in Fig. 3. Programs in the character mode can address any character in a word. Fields can start at any position in a word. A processor in a single operation can operate on fields of any length up to 63 characters long; operations on fields of greater length can easily be programmed. For example, two 57 character fields could be compared in a single operation.

There are two instances when the character mode operates with words of the type used in the word mode. Operations are provided in the character mode for converting numeric information in the alphanumeric representation to the standard word type of the word mode and vice versa. In both of these instances, the length of the alphanumeric fields being converted to or from the word mode type of word can be no greater than eight characters long. Again, conversion of fields of greater length can easily be programmed.

The purpose of the word mode is to provide the advantages of high-speed parallel operations, floating-point abilities and the inherent information density possible in a binary machine. In the first case, it is economically feasible to provide parallel operations in a word machine; the cost of parallel operations on variable length fields would be prohibitive. In the last case, a given size memory can contain over twenty percent more numeric information if that information is expressed in binary rather than binary-coded decimal, and over eighty percent more information than can be expressed in six-bit alphanumeric representation.

The purpose of the character mode is to provide editing, scanning, comparison and data manipulative abilities (although addition and subtraction are also provided). The type of editing facilities provided obviate the need for the artificial "add-shift-extract-store" type of editing. For example, operations are provided for generalized insertion of editing symbols (such as blanks, decimal points, floating dollar signs, etc.) and for the substitution or suppression of any unwanted characters. For those interested in the new area of Information Processing Languages, the character mode is particularly well suited to list structures.

First Character	Second Character	Third Character	Fourth Character	Fifth Character	Sixth Character	Seventh Character	Eighth Character

Fig. 3. Data word—character mode.

Program Organization

Programs in the B 5000 are composed of strings of syllables. A syllable is the basic unit of the program and is twelve bits in length. The term "syllable" is used rather than instruction to distinguish it from conventional single-address or multi-address instructions. Each program word contains four syllables and they are executed sequentially in a left-to-right order within the program word, and sequentially by word. Branching is allowed to any syllable within a word. Before delving into some of the details of the internal operation of the B 5000 processor, it is necessary to discuss stacks, Polish notation, and the Program Reference Table.

The Stack

The internal organization of single-address computers forces the wasting of both programming and running time for the storage and recall of the intermediate results in the sequence of computation. The data must be placed into the proper registers and memory cells before the operation can be executed, and their contents must often be completely rearranged before the next operation can be performed. Multi-address computers are constructed to make the execution of a few selected operations more efficient, but at the expense of building inefficiencies into all the rest. Automatic programming aids attack this problem indirectly: they relieve the programmer of the need to laboriously code his way around machine design, but they still must provide object coding to accomplish the storage and recall functions. In brief, conventionally designed computers, with or without automatic programming aids, require the wasteful expenditure of programming effort, memory capacity, and running time to overcome the limitations of their internal organization.

The problem is attacked directly in the B 5000 by incorporation of a "pushdown" stack, which completely eliminates the need for instructions (coded or compiled) to store or recall intermediate results.

In a B 5000 processor, the stack is composed of a pair of registers, the A and B registers, and a memory area. As operands are picked up by the programs, they are placed in the A register. If the A register already contains a word of information, that word is transferred to the B register prior to loading the operand into the A register. If the B register is also occupied by information, then the word in B is stored in a memory area defined by an address register S. Then the word A can be transferred to B and the operand brought into the A register. The new word coming into the stack has pushed down the information previously held in the registers. As each pushdown occurs, the address in the S register is automatically increased by one. The information contained in the registers is the last information entered into the stack; the stack operates on a "last in–first out" principle. As information is operated on in the stack, operands are eliminated from the stack and results of operations are returned to the stack. As information in the stack is used up by operations being performed, it is possible to cause "pushups," i.e., a word is brought from the memory area addressed by the S register, and the address in the S register is decreased by one.

To eliminate unnecessary pushdowns and pushups, the A and B registers both have indicators used for remembering whether the registers contain information or are empty. When an operand is to be placed in the stack and either of the registers is empty, no pushdown into memory occurs. Also, when an operation leaves one or both of the registers empty, no automatic pushup occurs.

Polish Notation

The Polish logician, J. Lukasiewicz, developed a notation which allows the writing of algebraic or logical expressions which do not require grouping symbols and operator precedence conventions. For example, parentheses are necessary as grouping symbols in the expression A(B + C) to convey the desired interpretation of the expression. In the expression A + B/C, the normal interpretation is A+(B/C), rather than (A+B)/C, because of the convention that the / operator is of higher precedence than the + operator. The right-hand Polish notation used in the B 5000 is based on placing the operators to the right of their operands: A + B becomes AB + in Polish notation. A+B+C can be written either as AB+C+, or as ABC++. In the expression ABC+ +, the first + operator says to add the operands B and C. The second + operator says to add A to the sum of B and C. Returning to the first examples above, A(B+C) can be written as BC+A× or ABC+× in Polish. The second example is written as BC/A+ or ABC/+. The extension of Polish notation to handle equations is shown in the following example:

Conventional notation Z=A(B−C)/(D+E)
Polish notation ABC−×DE+/Z=

The Stack in Use

To illustrate the functioning of the stack, two simple examples are shown in Figs. 4 and 5. In the examples, the letters P, Q, and R represent syllables in the program that cause the operands P, Q, and R to be picked up and placed in the stack. The symbols + and × represent syllables that cause the add and multiply operations to occur. The two examples represent different ways of writing P(Q+R) in Polish notation. The first example in Fig. 4 does not require pushdowns or pushups. The second example, shown in Fig. 5, requires a pushdown in the execution of the syllable R, and a pushup in the execution of the syllable ×. The columns in the table represent the contents of the various registers after execution of the syllable listed in the first column.

Polish Notation QR+P×

Syllable Executed	*Contents of*	
	Register A	*Register B*
Q	Q	Empty
R	R	Q
+	Empty	R+Q
P	P	R+Q
×	Empty	P(R+Q)

Fig. 4

Independence of Addressing

One of the goals set in the design of the B 5000 was to make the programs independent of the actual memory locations of both the program itself and the data, in order to provide really automatic program segmentation. Through automatic program segmentation, it is possible to have program size practically independent of the size of core memory. The systems analyst or programmer intending to do multi-processing is then no longer faced with the difficult task of planning what jobs are to be run together in order that system storage capacities are not exceeded.

In achieving independence of addressing, a solution requiring large contiguous areas of memory was not deemed satisfactory. Each segment of the program and each data area should be completely relocatable without modification to the program. It is

Polish Notation PQR+×

Syllable Executed		*Contents of*			
		Register A	*Register B*	*Register S*	*Cell 101*
P		P	Empty	100	–
Q		Q	P	100	–
R	Pushdown	Empty	Q	101	P
	Execute	R	Q	101	P
+		Empty	Q−R	101	P
×	Pushup	Q−R	P	100	–
	Execute	Empty	P(Q−R)	100	–

Fig. 5

then possible to load all the segments of a program or programs onto the drum at load time and call in the segments to any available space in core memory as needed during run time. If some segment of a program is overlaid by a subsequent segment of a program, the segment of the program destroyed in core memory is still available on the drum to be called in again if needed.

Due to the very high program densities in the B 5000, the availability of high capacity drum storage on every system and automatic segmentation, a minimum B 5000 system has the capacity for a program or programs equivalent to approximately 40,000 to 60,000 single address instructions. Of course, if an installation normally ran such large programs, the system would very likely not be a minimum system. However, the installation having an occasional need to run very large programs is not prevented from doing so by storage capacity.

Processing speed now becomes a function of the size of core memory. If large programs are run in a system with small core memory, time will be consumed in recalling program segments from drum to core. If the core memory is expanded, less time will be spent in such activity and the program or programs will be speeded up, and no reprogramming is required.

Program Reference Table

The means of achieving independence of addressing in the B 5000 is called a Program Reference Table (PRT). The PRT is a 1,024 word relocatable area in memory used primarily for storing control words that locate data areas or program segments. There are also control words for describing input-output operations. These control words, called descriptors, contain the base address and size of data areas, program segments and input-output operation areas. A descriptor specifying an input-output operation also contains the designation of the unit to be used and the type of operation to be performed. Operands may also be stored in the PRT, providing direct access to single values such as indices, counts, control totals, etc.

In the word mode of the B 5000, every item of data is considered to be either a single value or an element of an array of data. If it is a single value, it will be obtained directly by indexing a descriptor contained in the PRT.

Program segments are described by program descriptors. In addition to core base address, the program descriptor contains the location in drum storage of the program segment and an indication if the program segment is currently in core memory starting at the address specified in the descriptor. Entry to a program segment is made via its program descriptor contained in the PRT. If the program segment is in core memory, entry will be made to the program segment. However, when entry is attempted to a program segment whose descriptor indicates that the segment is not in core memory, automatic entry to the Master Control Program will occur and the desired segment will then be brought in from the drum. Notice that in moving from one segment to another, it is not necessary to know whether the segment to be entered is currently in core memory. Branching within a program segment is self-relative, i.e., the distance to jump either forward or backward is specified, not the address to be jumped to.

As a result of keeping all actual addresses of data and program in the PRT, the program itself does not contain any addresses, but only references to the PRT. To specify one of the 1,024 positions in the PRT requires only 10 bits which contributes greatly to the high program density achieved in the B 5000. Since the PRT is relocatable, references to the PRT contained in the program are to relative locations, thus completely freeing the program from any dependence whatsoever on actual memory locations.

The Word Mode Program

The word mode of the B 5000 processor has four types of syllables. The syllable type is distinguished by the two high-order bits of each 12-bit syllable. The types of syllable and the identification bits are:

00—	Operator Syllable
01—	Literal Syllable
10—	Operand Call Syllable
11—	Descriptor Call Syllable

The first of these, the operator syllable, causes operations to be performed. The remaining ten bits of the operator syllable are the operation codes. There are approximately sixty different operations in the word mode. For those operations requiring an operand or operands, the processor checks for sufficient operands in the registers; if they are not there, pushups from the stack in memory occur automatically.

The literal syllable is used for placing constants in the stack to be used as operands. The ten bits of the literal syllable are transferred to the stack. This allows the program to contain integers less than 1,024 as constants.

The operand call syllable, and the descriptor call syllable address locations in the program reference table. The purpose of the operand call syllable is to place an operand in the stack; the purpose of the descriptor call syllable is to place the address of an operand, a descriptor, in the stack. There are four situations that arise, depending on the word read from the program reference table.

1 The word is an operand.

2 The word is a descriptor containing the address of the operand.

3 The word is a descriptor containing the base address of the data area in which the operand resides.

4 The word is a program descriptor containing the base address of a subroutine.

For (1), the operand call syllable has completed its action by placing an operand in the stack. The descriptor call syllable will cause the construction of a descriptor of the operand, replacing the operand by the constructed descriptor.

For (2), the operand call syllable then reads the operand from the cell addressed. The descriptor call syllable has completed its action.

For (3), indexing of the descriptor by the item that is now the second item in the stack occurs. For an operand call syllable, the operand is obtained from the indexed address; for the descriptor call syllable, action is complete after the indexing.

In the case of (4), subroutine entry occurs to the subroutine addressed. A word of the three previous types may be left in the registers upon return from the subroutine, in which instance the actions described above will take place, depending upon the type of syllable which initiated the subroutine.

Essentially, the four types of action that occur for an operand call syllable are obtaining an operand directly, indirectly, from an array, or by computation. Sometimes in the use of the call syllables, it is not known which type of action will occur for a particular syllable when the program is created. This is particularly true for call syllables in subroutines.

Programs in the word mode consist of strings of syllables which follow the rules of Polish notation. Variable length strings of call syllables and literal syllables, which place items of information in the stack, are followed by operator syllables which perform their operations on information in the stack.

The indexing features of the B 5000 allow generalized indexing and at the same time provide complete storage protection. Data areas and program segments of different programs may be intermingled, but a program is prevented from storing outside of its data areas. The method of indexing allows any of the 1,024 words of the program reference table to be considered index registers. Multilevel indexing is provided, i.e., indices of arrays can themselves be elements of arrays.

The subroutine control provided in the B 5000 allows nesting of subroutines—even recursive nesting (a subroutine is a subroutine of itself) arbitrarily deep. Dynamic allocation of storage for parameter lists and temporary working storage simplify the use of subroutines. Storage is automatically allocated and deallocated as required.

Character Mode Program

In the character mode of the B 5000 Processor, there is only one type of syllable, called the operator syllable. Program segments in the character mode are constructed of strings of these syllables. The character mode is designed to provide editing, formatting, comparison, and other forms of data manipulation. In doing so, the processor uses two areas of memory—the source and destination areas. When a program switches from word mode to character mode, two descriptors containing the base addresses of these areas are supplied. The source area or destination area may be changed at any time during character mode so that the program may act on several areas.

The character mode operator syllable is split into two 6-bit parts; the last part specifies the operation to be performed and the first part specifies the number of times the operation is to be performed. Operations are provided for the transferring, deletion, comparison, and insertion of characters or bits. Also, there are operations which allow the repetition of syllable strings. This is quite useful for complex table look-up operations and for editing information which contains repeated patterns.

Conclusion

The Burroughs B 5000 system has been designed as an integrated hardware-software package which offers such benefits as savings in the memory space required to store equivalent object programs; multi-processing and parallel processing; and running identical programs on systems with different size memories and different system configurations with no loss in individual system efficiency.

References

Lonergan and King [1961]; Barton [1961]; Bock [1963]; Carlson [1963]; Maher [1961].