The present paper considers the underlying design concepts of IBSYS/IBJOB, *an integrated programming and operating system.*

The historical background and over-all structure of the system are discussed.

Flow of jobs through the IBJOB *processor, as controlled by the monitor, is also described.*

Design of an integrated programming and operating system

Part I: System considerations and the monitor

by A. S. Noble, Jr.

total systems approach

Since the first stored-program computer was built, the importance of programming in the total computing system has grown continuously. The programming interface between user and computer—the system programs—has become a key factor in the full exploitation of technological advances in hardware design. The formulation of design principles based on systems programming experience will stimulate further advances in computing technology. From such experience an effort has been made in this set of papers to define design concepts that are independent of a particular hardware environment with the expectation that they will prove valuable for future systems development.

programming tools

To facilitate the use of stored-program computers, systems programming has developed a number of tools: *assembly programs* that translate *source programs*, written in *symbolic languages*, into *object programs* suitable for loading into computer storage; *compilers* that translate statements written in *higher-level languages*, similar to mathematics or English, into equivalent sequences of machine instructions to be assembled; *loaders* that load the assembled object programs into storage, some loaders having the capability of combining several separately-assembled *subroutines* to

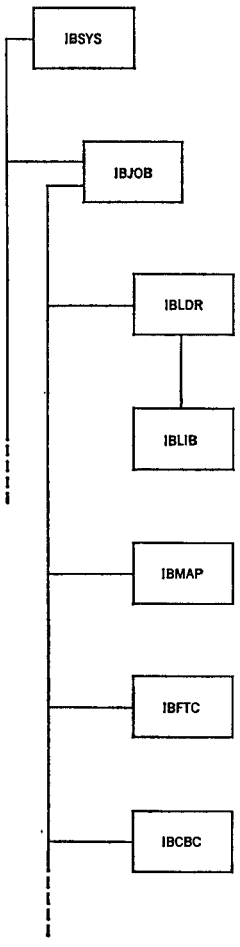
form a single program, ready for execution; *libraries* of assembled and tested subroutines that can be loaded automatically, to perform previously-defined functions; *monitors* that schedule and coordinate the execution of separate programs; *input/output control systems*, that supervise communication with I/O devices; *sorts*, *report-program generators*, and various *utility programs* that perform routine tasks.

programming, operating systems

The term, *programming system*, generally refers to the combination of a language and its associated translator, or *processor*. The processor may also include several of the other tools mentioned above. Whereas a programming system facilitates the statement of problems for computer solution, an *operating system* with its monitor automates the actual operation of the computer itself. The computer is used in the one case to assist the programmer, in the other, the operator.

The IBSYS/IBJOB system

Figure 1 IBSYS/IBJOB system



The purpose of the IBSYS operating system is to automate computer operations efficiently with *minimum* restrictions on subsystems and installation practices. The IBJOB processor, an IBSYS component subsystem, integrates several language translators within one generalized programming system.

The IBSYS/IBJOB system consists of the seven components shown in Figure 1:

- The Basic Monitor (IBSYS) contains: (a) a common, inter-system communication region that includes tables on equipment configuration and status; (b) a supervisor, directed by control cards, that changes the configuration tables and transfers control between subsystems; and (c) an editor that maintains and adapts the system to the requirements of a particular installation.
- The Monitor (IBJOB) supervises the loading of system components and regulates input/output to process a stack of jobs. Each job, or unit of work, may include any mixture of FORTRAN IV compilations, COBOL compilations, MAP assemblies, and the combined execution of binary programs from these and previous compilations and assemblies.
- The Loader (IBLDR) loads separately-assembled program segments, relocating and combining them to produce one absolute binary object program, allocates storage for common data and I/O buffers, generates initialization sequences necessary for using the input/output control system (IOCS), and provides a listing of the final storage allocation.
- The Library (IBLIB) contains IBMAP-assembled subroutines to be loaded if required by the object program.
- The Macro Assembly Program (IBMAP) processes MAP-language programs which may have been generated by the compilers or written by a programmer.

- The FORTRAN IV Compiler (IBFTC) processes programs written in the FORTRAN IV language and produces input to IBMAP.
- The COBOL Compiler (IBCBC) processes programs written in the COBOL language and produces input to IBMAP.

configuration requirements

The system was designed to operate on an IBM® 7090/7094, a binary, 32,768-word computer, with attached printer and tapes. Eight logical system I/O functions are performed: one library, one input, two output, and four utility; these may be satisfied by a minimum of seven tapes, if an off-line 1401 with punch and printer is available for processing combined output files. Optionally, a 1301 disk, or 7320 drum, may be substituted for library and utility functions.

Basically, the same over-all system design is being implemented for the IBM 7040/7044 system, although a number of differences exist. These will be examined in a later paper. In general, the ensuing discussion of the 7090/7094 system will be applicable to other versions of IBSYS/IBJOB.

To better understand the motivation behind IBSYS/IBJOB, a review of programming developments that influenced the system design should prove helpful.

Historical background of the system

Experience in the design of programming and operating systems for 704, 709, and 7090 computers dates from 1956.

FORTRAN I, II

FORTRAN, the first programming system to receive wide acceptance, was released in 1957 for the 704. A science-oriented programming system, it consisted of the FORTRAN I language and the compiler/assembler necessary to translate source programs written in this language. Machine-language object programs were punched in binary cards for loading and executing on the computer as a separate manual operation.

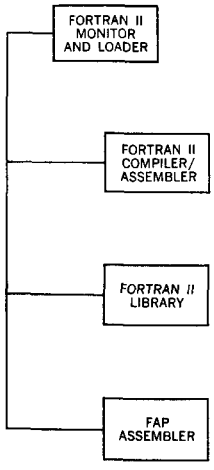
The practicality of this programming system was increased significantly by the addition of new features in 1958. With this version (FORTRAN II for the 704 and 709) one could write subprograms, which could be translated separately into relocatable binary format. The loader could then combine these separately-translated subprograms into a single absolute program for execution. This facility encouraged installations to maintain libraries of frequently-used subprograms in the form of relocatable binary decks. Eventually, the loader was modified to obtain library subprograms automatically from a library tape file, as required at load time.

FORTRAN II monitor

The most extensively used operating system today, the FORTRAN II monitor system, was developed by North American Aviation Inc., a major FORTRAN user, and introduced in 1959. It was designed to automate the operation of the 709 FORTRAN II programming system.

It accepted as input on tape a stack of jobs, each of which

Figure 2 FORTRAN II monitor



could require the system to translate source programs into relocatable binary, then combine these with previously-translated subprograms together with any required library programs, and finally execute the absolute program—all in one continuous operation. Optionally, data could follow each job on the input tape. Also, it was possible to chain together several memory loads for execution.

A symbolic machine-language programming system, FAP, with FORTRAN II-type relocatable subprogram features, was developed in 1960 at the Western Data Processing Center, UCLA, and added to the system.

The FORTRAN II monitor system, as shown in Figure 2, evolved from the accumulated practical experience of many SHARE installations. It was not planned as an integrated system; it grew as the everyday production needs of users dictated.

Planning for a completely-redesigned system, 7090 FORTRAN IV, was underway by 1960. In addition to supporting further extensions and changes in the language, it would employ modular technology in both the system organization, as well as within the components: monitor, loader, library, assembler, and compiler.

FORTRAN IV

Some of the design goals for FORTRAN IV were:

- Increased throughput in the compiler.
- Greater adaptability to environmental changes.
- Ease of system modification.
- Extended capabilities in the monitor.
- An overlay feature in the loader.
- More flexible buffering for the object program.

These goals were to be achieved within a system that would be equivalent in function to the FORTRAN II monitor system. The same goals were also consistent with the objectives later formulated for the IBJOB processor.

COMMERCIAL TRANSLATOR

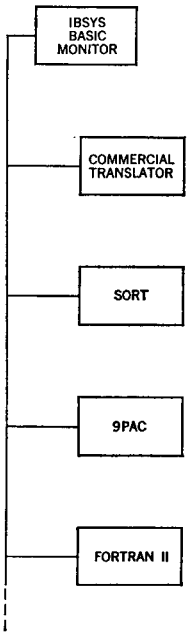
Concurrently with (but independently of) FORTRAN IV planning, COMMERCIAL TRANSLATOR, a business-oriented programming system, was being developed for the 7090. Many advanced techniques were created for this system, such as: control section concepts, direct cross-referencing, an I/O control system (IOCS), and internal text processing. In addition to its language and translator, COMMERCIAL TRANSLATOR included its own monitor, loader, and library. The relocation scheme, deck formats and operational conventions differed considerably from those of FORTRAN II.

It became apparent in the later stages of COMMERCIAL TRANSLATOR development and FORTRAN IV planning that they could share a common operational environment, particularly with respect to I/O, and system configuration requirements. Accordingly, it was decided that the IOCS developed for COMMERCIAL TRANSLATOR would also be used in FORTRAN IV and that a basic intersystem monitor (IBSYS) would be designed to define the common environment.

The IBSYS processor operating system

The IBSYS processor operating system denotes the basic monitor (IBSYS) together with any component subsystems operating under IBSYS. The initial version of IBSYS was released in conjunction with the COMMERCIAL TRANSLATOR processor for the 7090 in November 1961. Since then, previously available independent systems such as SORT, 9PAC and FORTRAN II, have been modified for operation within the IBSYS processor operating system, as shown in Figure 3. This facilitated system adaptation to new I/O devices while providing added convenience of operation.

Figure 3 IBSYS processor operating system



As mentioned earlier, one of the fundamental design criteria for the basic monitor (IBSYS) was that it should impose the fewest restrictions on subsystems operating under it. Stated simply, it was to be a framework, concerned only with environment and control between subsystems. It would maintain tables defining configuration and system assignments, and would call upon subsystems as required to perform tasks indicated by control cards in the input stream.

Some of the design goals for IBSYS were:

- Operational standards for unit assignment, system editing, and maintenance, etc.
- Minimum operator intervention for system setup.
- Assignment of system I/O functions to any suitable device available.
- Maximum flexibility for job definition and scheduling.
- Increased efficiency for peripheral operations.
- Sufficient modularity to facilitate modification of the distributed system to satisfy particular installation needs.

It was evident that all of these objectives need not be achieved at once. Minimum requirements included the intersystem communication region, or "nucleus" (IBNUC); the supervisory control program (IBSUP) to maintain configuration changes and operational continuity between subsystems; and the editor (IBEDT) to write and update the system library.

further integration

While IBSYS achieved a minimum degree of integration for previously independent subsystems, it also allowed a maximum degree of flexibility for subsystems with highly dissimilar tasks to perform. Each IBSYS subsystem contained its own intrasystem control monitor, tailored to the job processing needs peculiar to it. The possibility of further integration within programming subsystems with similar tasks became evident. Similarities of job requirements among two of the subsystems were apparent; FORTRAN II and COMMERCIAL TRANSLATOR automated the process of translating, loading, and executing programs written in particular languages. Both had similar facilities for library search, subprogram cross-referencing, and centralized I/O control.

COBOL

Many design and implementation techniques developed for COMMERCIAL TRANSLATOR were intended for use in a COBOL trans-

lator which had been planned to be implemented as soon as COMMERCIAL TRANSLATOR was completed. Also, the functional similarities between the COMMERCIAL TRANSLATOR and FORTRAN II systems applied as well to the proposed COBOL and FORTRAN IV systems.

While it was not practical to rewrite COMMERCIAL TRANSLATOR and FORTRAN II, the above considerations led to redesign of FORTRAN IV and COBOL, with the idea of integrating both translators into a single job-processing subsystem within the IBSYS processor operating system.

The IBJOB processor

Just as the IBSYS basic monitor was designed to integrate several programming systems into a common operating environment, the IBJOB processor provides a common operating environment for several language translators within a single, highly-integrated programming system.

modular components

The compilers share a common monitor, assembler, loader, and library. This makes available to the programmer, writing in the FORTRAN IV, COBOL or MAP languages, all of the features provided by these components. Improvements made to one component, such as the addition of overlay capabilities to the loader, benefit all components that depend on it. The system can be simply adapted to new hardware devices, peripheral formats, and operating procedures, with minimum disruption to installation practice. The extent of re-education required within the installation is lessened.

use of core storage

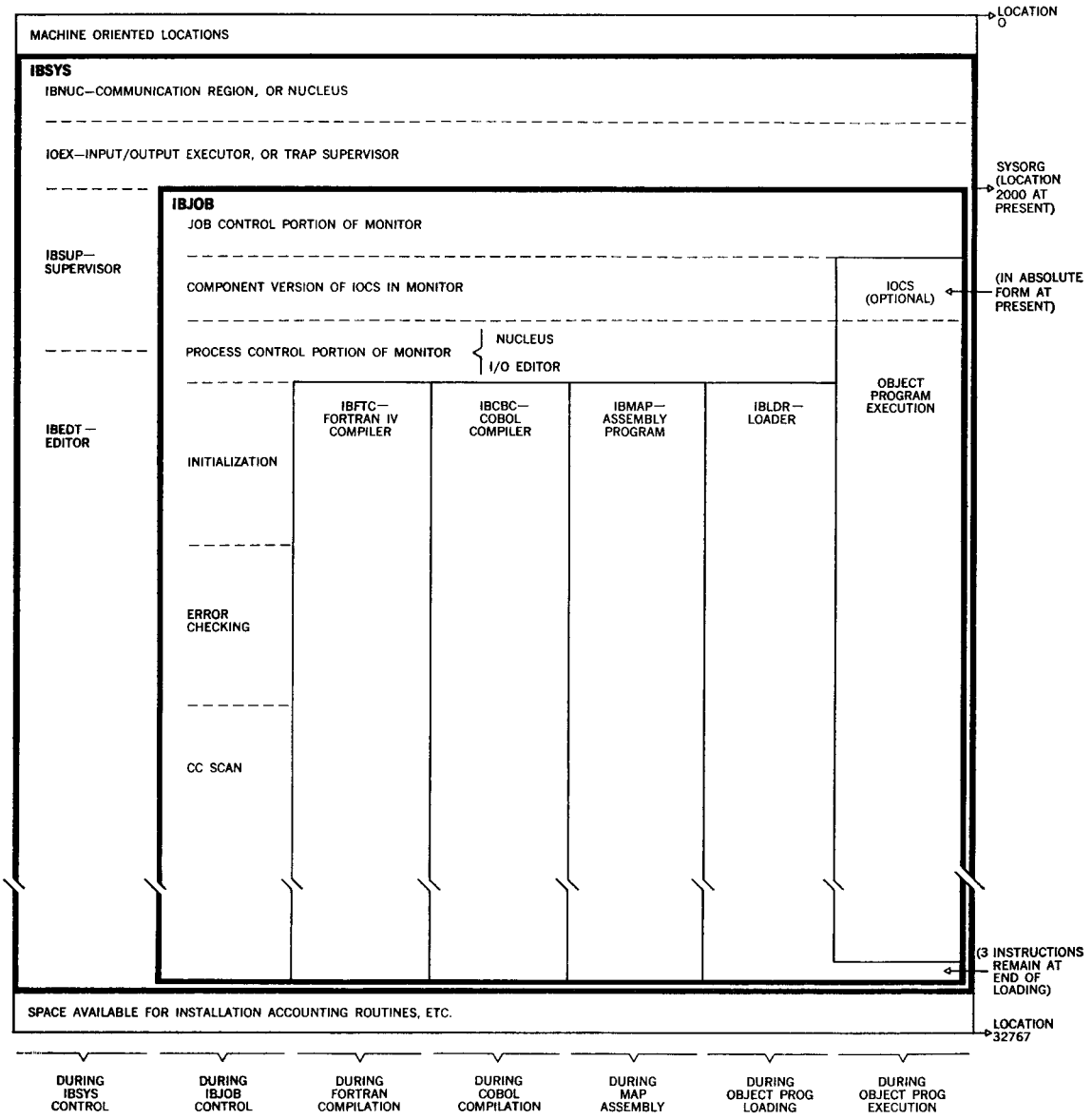
In its use of core storage, the IBJOB processor conforms to overlay structure; in fact, it was designed with the possibility in mind of processing the system itself as a MAP-relocatable object program using the overlay feature of the loader. This structure, as well as the hierarchy of control through system phases, is illustrated by the core storage chart, Figure 4. The origin of IBSYS subsystems, SYSORG, is shown on the chart below the I/O executor (IOEX). That IOEX would remain in core storage at all times was an arbitrary decision based on the assumption that all subsystems, including object programs, would make use of IOEX to preserve device independence and common error recovery procedures.

The actual minimum storage requirement for full IBSYS operation is approximately 500 locations for the communication region (IBNUC). The design of the system would permit modification to allow object programs (or subsystems) which do not require IOEX to load just below IBNUC; thus eliminating the job control portion of the monitor at object time. The chart also shows the component version of IOCS, an optional part of the object program, in its present absolute form. It would be possible to have IOCS available in the form of relocatable subroutines in the library (IBLIB).

The separate and independent functions of the supervisory

portions of IBSYS and IBJOB will be clear from the chart; IBSUP and IBJOB have the same origin. The design is also consistent with extensions to IBSYS to expand the concept of a job to include multiple subsystem and execution phases. Just as a job, or unit of work, for the IBJOB monitor can involve a sequence of phases requiring any or all of its components (IBFTC, IBCBC, IBMAP, and IBLDR); so a job, or unit of work, for the IBSYS basic monitor (really IBSUP) could involve an analogous sequence of phases between its component subsystems (IBJOB processor, SORT, etc.). From this point of view, IBJOB can be regarded as a particular

Figure 4 IBSYS/IBJOB use of core storage



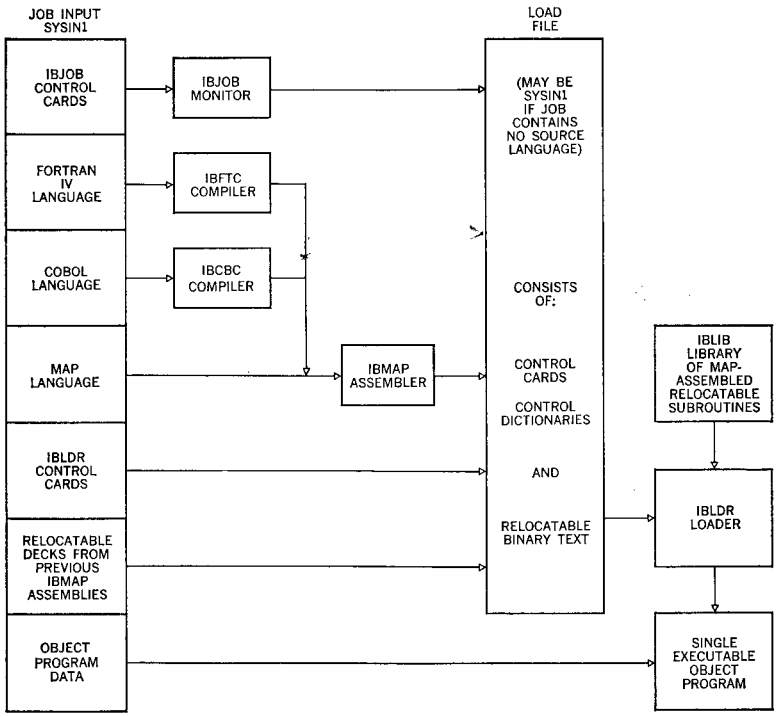
subset of one monitor, IBSYS/IBJOB. Of course, if all control functions were to be actually combined into one monitor, that monitor would have to include the monitor functions of SORT, 9PAC, FORTRAN II, and COMMERCIAL TRANSLATOR; but, a single installation would seldom require all of these. The present IBSYS concept, which allows for additions, substitutions, or deletions of components to satisfy particular installation requirements, seems most satisfactory—that is, if sufficient alternatives are present.

The IBJOB monitor

A job in the sense of a unit of work for this monitor, as for the FORTRAN II monitor, may consist of one or more compilations and assemblies, in addition to a single load-and-execution. The flow of jobs through the IBJOB processor is illustrated in Figure 5.

The monitor will call upon system components, as required, to translate any source-language programs encountered in the job deck into relocatable binary format suitable for input to the loader. A single job deck may contain several programs, each of which may be written in any one of the three languages, FORTRAN IV, COBOL, or MAP. The system I/O editor, in the monitor, will insure that the output from the compilers is fed to the assembler. All system components are serviced by the I/O editor which will provide a record of input or will accept punch or listing output,

Figure 5 Job flow through the IBJOB processor



upon request. This method of handling peripheral I/O relieves the processors of any concern with blocking, compression, or alteration of decks; with peripheral formats; or with the actual source of input or the final destination of output.

When a job requires loading, and the job deck has been completely translated into one or more relocatable binary decks, along with all necessary control cards, the monitor will call upon the loader to process this "load file" into an absolute program in memory, suitable for execution. Object program execution is terminated by return of control to the monitor, which will then proceed to process the next job.

As long as the next job falls within its scope, the IBJOB monitor retains control; in this way, system searching is held to a minimum. If the next job falls outside the scope of the IBJOB processor, control is returned to the basic monitor.

In summary, the IBSYS/IBJOB concept was intended to satisfy the diverse and changing needs of a wide variety of computer installations with an integrated programming and operating system that is, despite its generality, flexible and non-restrictive enough to adapt to particular requirements without undue loss of efficiency.

BIBLIOGRAPHY

- IBM 7090/7094 Operating Systems: Basic Monitor (IBSYS), Reference Manual C28-6248, International Business Machine Corporation, 1962.
- Programming Systems Analysis Guide, 7090/7094 IBSYS-IOEX, Programming Systems Analysis Guide C28-6299, International Business Machines Corporation, 1963.
- IBM 7090/7094 Programming Systems: IBJOB Processor, Systems Reference Library C28-6275, International Business Machines Corporation, 1963.
- IBM 709/7090 Input/Output Control System, Systems Reference Library C28-6100, International Business Machines Corporation, 1961.
- IBM 7090/7094 Programming Systems: IBJOB Processor: Overlay Feature of IBLDR, Systems Reference Library C28-6331, International Business Machines Corporation, 1963.
- IBM 7090/7094 Programming Systems: MAP (Macro Assembly Program) Language, Systems Reference Library C28-6311, International Business Machines Corporation, 1963.
- IBM 7090/7094 Programming Systems: FORTRAN IV Language, Systems Reference Library C28-6274, International Business Machines Corporation, 1963.
- IBM 7090/7094 Programming Systems: IBJOB Processor: Part 5: COBOL Compiler (IBCBC), Systems Reference Library J28-6260, International Business Machines Corporation, 1962.
- IBM 7090 Data Processing System, Reference Manual A22-6528, International Business Machines Corporation, 1959.
- IBM 7090/7094 Systems Reference Library Bibliography, A28-6306, International Business Machines Corporation, 1963.
- IBM 7040/7044 Systems Reference Library Bibliography, A28-6288, International Business Machines Corporation, 1962.
- IBM 7040/7044 Systems Reference Library Systems Summary, A28-6289, International Business Machines Corporation, 1963.
- IBM 7040/7044 Student Text, C22-6732, International Business Machines Corporation, 1963.