The control data STAR-100—Performance measurements

by CHARLES J. PURCELL

Control Data Corporation
St. Paul, Minnesota

INTRODUCTION

The CONTROL DATA STAR-100 (STring-ARray) Computer is a very large, general purpose, high speed computing system. The STAR-100 computer utilizes integrated circuitry, ferrite core memory, 400 hz power and freon cooling in the hardware implementation. The logical design of the computer combines stream processing, virtual addressing, hardware macro instructions, segmented (pipeline) arithmetic units and a 256-word high speed register file to perform arithmetic and logical operation on discrete or structured data elements (Figure 1).

Several of the STAR-100 Systems are in final test and checkout at this time preparatory to delivery. The STAR-100 is extending the state of computer art in terms of the memory bandwidth, the arithmetic pipeline performance, hardware aids to the operating system and built-in sub-routines for the programmer. As with any substantial improvement in computing, we at CONTROL DATA have had to learn a great deal about the use of this computer before it could be completed. The particular areas of substantial interest are the hardware design, potential performance and the program language environment.

HARDWARE DESIGN

The initial design goal of the STAR-100 required the production of at least 40 million additions per second on arrays of data formatted in 64-bit floating point words. This requirement in the STAR-100 is met by the execution of a three-address memory to memory array instruction. The source arrays and the result array must be carefully located in contiguous cells of memory in order to provide the logic designer with the power to completely overlap more than 80 million loads, 40 million adds and 40 million stores every second. The instruction repertoire of the STAR-100 was carefully selected to match the logic design with the program. One can visualize the demands on the programmer by considering the STAR-100 as a simplified hardware implementation of a programming language.[1]

Array operations are provided in the STAR-100 on either 32-bit or 64-bit floating point operands. Consideration of the total computational requirements suggested that the rate of array operations would require support by a wide variety of textual operations on byte strings as well as logical operations on bit strings. The 256 word register file is used to contain scalars, descriptors or index quantities used in support of the array or string operations. Branch and test operations are available on bits, bytes indices, scalars, arrays, strings or conditions. In all, some 230 instructions are provided in the STAR-100.

A physical design evolved from these considerations (Figure 2) whereby the complete computer is assembled in one large monolithic structure. The minimum STAR-100 computer contains 4 million (8-bit) bytes of memory, 4 I/O channels, 1 direct access channel, the Central Processor and a Maintenance Control Unit. The Central Processor contains the Storage Access Control unit (SAC), the Stream Control unit, and the floating point pipelines.

The Stream Control unit of the Central Processor contains all streaming and instruction control, and operand alignment, buffering and addressing. The Stream Control unit also contains the register file of 256 64-bit words used for instruction and operand addressing, indexing, storing constants, field length counts, and as source and destination for conventional register to register instructions in 3-address format. A microcode memory in the Stream Control unit controls the initiation, interrupt (if necessary) and termination of all array and string instructions.

The register file utilizes a twenty nanosecond read orwrite cycle time semi-conductor memory component. The microcode memory utilizes an eighty nanosecond, two phase, semiconductor memory. Main memory of the STAR-100 is built of thirty-two banks each of 1.28 microsecond full cycle time, 512 bit word, ferrite core memory. At this time, the checkout process is focusing on the verification of each individual instruction to take over control of the data streams, correctly operate and form the results in the appropriate location. Then each and every instruction combination must also be verified as to the correct control and results.

The Central Processor of the STAR-100 System is aided by a number of auxiliary processing input/output stations (Figure 3). These stations provide intelligent control of the input/output requirements of the STAR-100 by use of control messages from the central monitor program. These con-

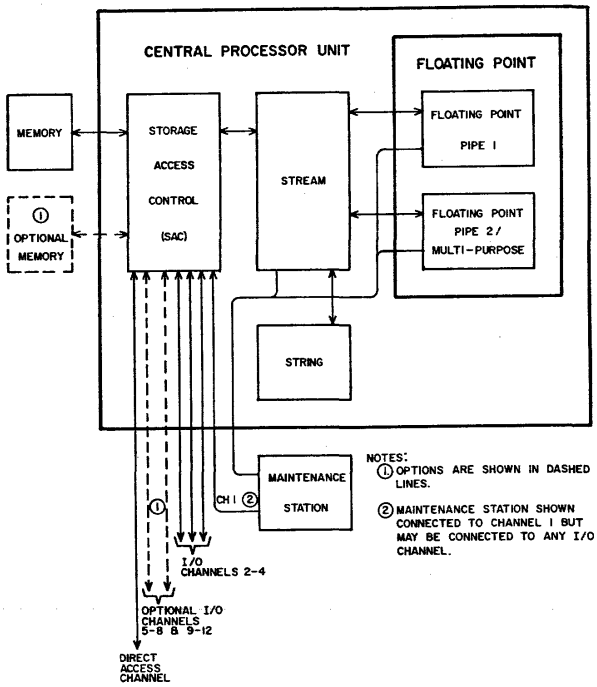

Figure 1—Basic CDC STAR-100 configuration

trol messages are processed by the station control unit (processor) while data streams move into and out of the station buffer unit.[2] A typical STAR-100 system is shown in Figure 4. Peripheral systems of this type have been in use for several years in our laboratory.

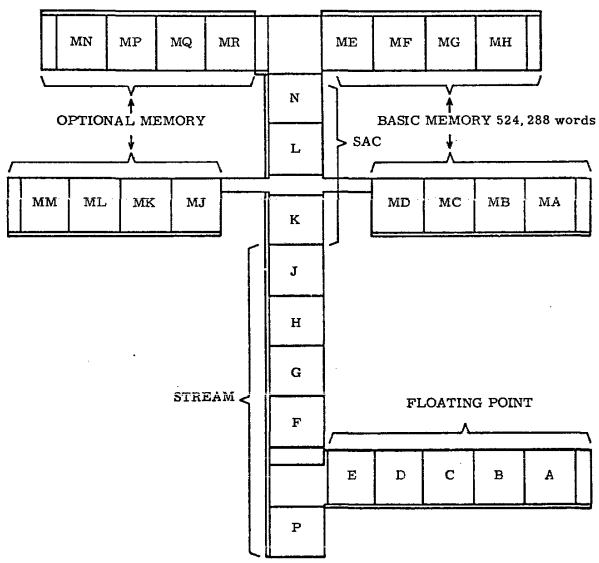

Figure 2—STAR-100 section identification

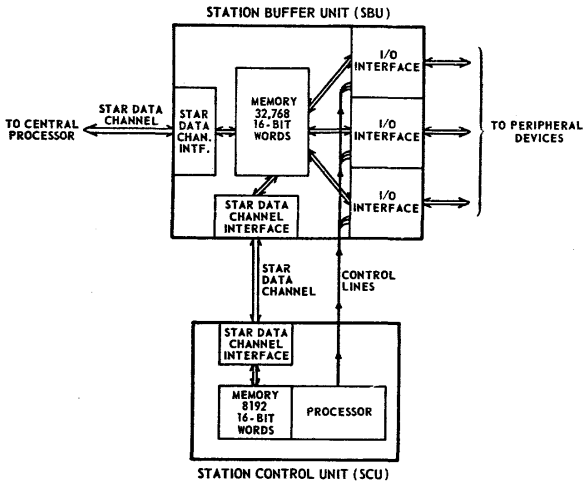

Figure 3—Basic station

PERFORMANCE MEASUREMENTS

As of December 1973 about ⅔ of the instruction vocabulary of the STAR-100 has been verified for correct performance on either of two central processors. This verification includes the extensive range of interrupt possibilities as well as most of the instruction combinations. As a result of these tests we have verified the raw performance potential of the STAR-100 via the contract SAMPLE PROBLEM. This test is representative of the best use of the STAR-100 in forming arithmetic results on packed arrays of data in either 32-bit or 64-bit floating pointing format. Some 32 billion intermediate results are correctly formed in 625 seconds yielding a rate of greater

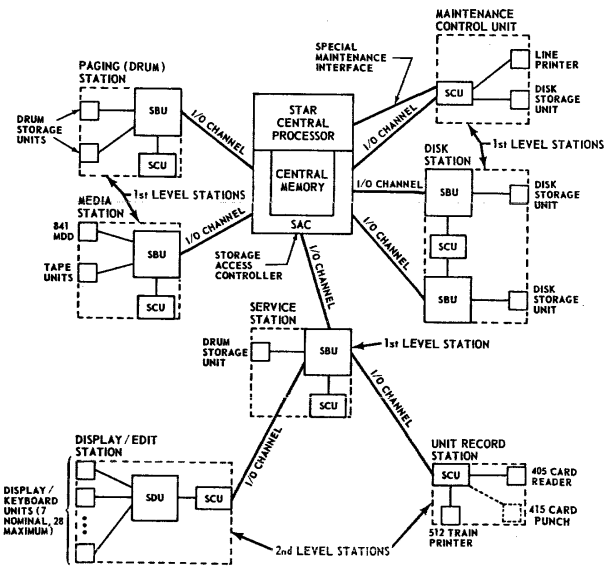

Figure 4—Typical STAR-100 system

than 50 million results per second. Other meaningful performance measurements await the installation of the STAR Operating System and appropriate compilers, interpreters and assemblers on the STAR-100 (MARCH-MAY 1974).

The performance potential of the peripheral system has been reported in the previous NCC.[2]

SOFTWARE REQUIREMENTS

The general purpose instruction repertoire and control facilities of the STAR-100 were designed to facilitate the development of dynamic recursive languages such as PL/1, ALGOL and APL. Users, in general, have continued to use FORTRAN practices in code development as well as for execution time service routines. Thus the STAR-100 System will require the implementation of FORTRAN (ANSI 1973) with substantial enhancements before the official release date in 1974. The planned enhancements to FORTRAN can be characterized as explicit APL-like monadic and dyadic functions in a static pre-defined environment. The expansions include:

1. Subscripting (K, I, L, M, N, J, S)
   implied DO loop (K, I, L, M, J, BEGIN:END: DELTA)
   cross section (K, I, *, M, N, J, S)
   offset (K, I, L, M, N, J, S; LENGTH)
   where any construct that specifies operations on an ARRAY will form a VECTOR instruction; other constructs will be executed in the conventional instruction sequence.
2. New Data Types
   BIT one bit per bit of memory
   SPARSE an order vector BIT mapping of an ARRAY
   DESCRIPTOR a scalar containing the length and base address of an ARRAY or CHARACTER
3. ARRAY INTRINSICS and BASICS
4. ARRAY FUNCTIONS
   where an ARRAY argument is processed to return an ARRAY result.
5. Selection Operations
   controlled store C=B. CNTRL. A
   compress C=B. COMPR. A
   expand C=B. EXPAND. A
   gather C=I. COMPR. A
   scatter C=I. EXPAND. A
   where A is an ARRAY, B is BIT, C is the result, and I is an Indexlist.
6. ARRAY ARITHMETIC
7. ARRAY TESTS
   compare B=A. RELATIONAL. C
   search I=A. RELATIONAL. C
   select

The general theme of these expansions is to produce a language super-set which could be compiled and executed on a conventional computer having the suggested language constructs in its compiler. FORTRAN remains a language fixed at compile time with an even larger set of reserved words.

Code optimization for the STAR-100 computer is a rich and rewarding field in terms of potential performance improvements. The compiler must allocate the use of the register file, eliminate common sub-expressions, minimize redundant address calculations and exploit the special hardware facilities of the STAR-100. In many cases a conventional FORTRAN program must be de-compiled in order to determine the intent of the programmer (INNER PRODUCT, INITIALIZATION, etc.).

The same expansions to FORTRAN would be required for the STAR implementation of PL/1 or ALGOL. APL makes full use of the STAR-100 via APL*STAR.

CONCLUSION

This project required the services of many divisions in CONTROL DATA, particularly the Advanced Development Laboratory. Many of the concepts utilized in the STAR-100 have been supplied by the programming community in general. These concepts were built into the STAR-100 in response to the question: "Just what is it that these programmers do?"

REFERENCES

1. Iverson, K. E., *A Programming Language*, Chapter 1, pp. 1-40, Wiley, 1962.
2. Hohn, W. C., and P. D. Jones, "The CONTROL DATA STAR—100 Paging Station," Vol. 42 N.C.C. Proceedings 1973, pp. 421-426.
3. Control Data, *STAR—100 Features Manual*, Pub. No. 60418100 October 1973, Control Data Corp., St. Paul, Minn. 55112.