STARAN parallel processor system software

by EDWARD W. DAVIS

Goodyear Aerospace Corporation
Akron, Ohio

INTRODUCTION

This paper is concerned with the features and concepts of system software for a parallel associative array processor—STARAN.* Definitions of parallel processors have appeared often. Essentially they are machines with a large number of processing elements. They have the capability to operate on multiple data streams with a single instruction stream. STARAN is a line of parallel processors with a variable number of processing elements.

Along with the multiple processing elements, STARAN has a memory organization that allows access either by location or association. That is the address of a memory word can be used explicitly, or words can be selected by association based on their content. Processing elements can operate on data selected associatively, making the machine an associative processor.

An alignment, or permutation, network in the machine provides a flexible interconnection between processing elements. This network is used to align data in the memory with the processing elements requiring the data and to provide communication between processors. This results in an array organization, making the machine an array processor. STARAN is thus a true parallel, associative, array processor.

It is expected that one might be curious about the use of this machine: the operating system, language processing software, user program development, and execution control aids. This paper gives a brief description of software for all these purposes. Some parts will be recognizable as fundamental members of the software for other general purpose computing systems. Special development was required, however, to handle features unique to the parallel organization.

The programming language is new. It includes declarations for defining storage in the arrays and instructions for using the parallel and associative properties of the machine. Interactive execution control software has been written. It simplifies development and debugging of user programs. This software differs from conventional debugging tools by the extensions related to the array memory organization. Discussion of the language and control software, plus methods of interfacing STARAN to other machines, are the major points of the paper.

* TM, Goodyear Aerospace Corporation, Akron, Ohio.

STARAN SYSTEMS

STARAN is an operational computing system. The hardware architecture is described in a companion paper presented at this conference[1] and in other literature.[2,3] A particular installation and its potential use is described in a companion paper.[4] This paper is concerned with a description of the existing system software. There are two modes of operation. First, STARAN can be operated as a stand-alone parallel processing system. Peripherals for this mode typically include a card reader, line printer, paper tape reader and punch, and cartridge type disk unit. Second, STARAN and a cooperating, or host, machine can be operated in an integrated fashion. This means that: (1) commands to the STARAN disk operating system can originate in the other machine, (2) the storage system of the host is available to STARAN users for program or data storage, and (3) a single task can use both machines to satisfy its processing requirements. All peripherals belonging to a stand-alone STARAN and to the host are available when the machines are integrated.

This paper describes the software for the STARAN stand-alone mode of operation, then covers the additional software used with the integrated mode.

Since the STARAN processor architecture is detailed in a companion paper[1] only a basic diagram is given in Figure 1. The multi-dimensional access associative arrays and their controls are the main architectural features. The sequential control, a Digital Equipment Corporation (DEC) PDP-11 minicomputer, has a minor role in the architecture but is important for software considerations. Other architectural features are mentioned later in the paper.

SOFTWARE FOR STARAN STAND-ALONE MODE

Software for the STARAN stand-alone mode of operation can be discussed from the standpoints of the operating system, language processing, and execution control procedures.

Batch disk operation system

In this paper, an operating system means the collection of routines that give the user appropriate control of the com-

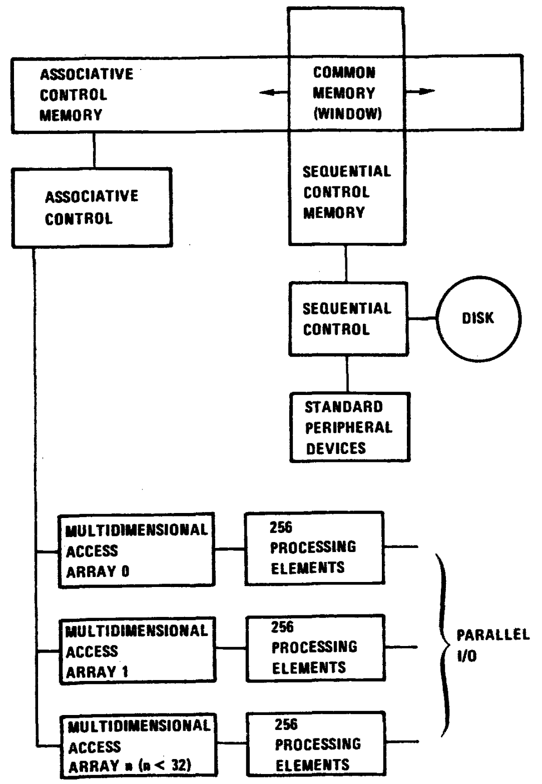
ASSOCIATIVE CONTROL MEMORY
COMMON MEMORY (WINDOW)
ASSOCIATIVE CONTROL
SEQUENTIAL CONTROL MEMORY
SEQUENTIAL CONTROL
DISK
STANDARD PERIPHERAL DEVICES
MULTIDIMENSIONAL ACCESS ARRAY 0
256 PROCESSING ELEMENTS
MULTIDIMENSIONAL ACCESS ARRAY 1
256 PROCESSING ELEMENTS
MULTIDIMENSIONAL ACCESS ARRAY n (n < 32)
256 PROCESSING ELEMENTS
PARALLEL I/O

Figure 1—STARAN block diagram

puting system, inform him of system status, provide input/output (I/O) facilities, and provide access to system programs. STARAN features a disk operating system (DOS) and has a batch processing capability. The batch command stream can be assigned to any character input device, allowing control to originate at the control console or from a user's file on the batch device.

The disk is a file structured bulk storage medium. All system software is resident on the device for easy, rapid access by the user.

Listed below are the standard programs supplied with the DEC PDP-11 batch system:

Program Name	Function
MACRO	Macro-assembler
LINK	Linker
LIBR	Librarian
PIP	File utility package
EDIT	Text editor
ODT	On-line debugging package
FORTRAN	Fortran compiler

These programs are not discussed further since primary emphasis in this paper is on the STARAN-related software that has been added to the above list to build the STARAN disk operating system.

One general rule used in software development was to avoid changes to the basic DEC batch system. This rule was intended to simplify any future change to a new DEC release.

Language processing

APPLE—Programs for STARAN are written in the APPLE* assembly language (Associative **P**rocessor **P**rogramming Languag**E**).[5] This language has some mnemonics that generate one machine language instruction and others that generate a sequence of machine instructions. The one-to-many mnemonics generally implement a parallel algorithm for arithmetic or search operations using the arrays. Thus, APPLE is at a higher level than sequential machine assembly languages.

APPLE produces relocatable or absolute program sections and has a conditional assembly capability. Groups of instructions in the language are listed below:

1. Assembler directives
2. Branch instructions
3. Register load and store
4. Array instructions
   a. Loads
   b. Stores
   c. Associative searches
   d. Parallel moves
   e. Parallel arithmetic operations
5. Control and test instructions
6. Input/output (I/O) instructions

Most of these groups of instructions resemble those of other typical assemblers. The unique group—array instructions—deals with operations on the multi-dimensional access arrays and the registers in their processing elements (PE). Some general comments apply to all the array instructions listed above. Operations take place only on arrays enabled by the array select register.[2] Fields are of variable length within each array word and are defined for various instructions by field pointers and length counters. The common register, a part of associative control, can contain an operand, which is used in common by all selected array words.

More detail is presented below on the array instructions; i.e., loads, stores, associative searches, parallel moves, and parallel arithmetic operations.

The "load" array instructions load the processing element (PE) registers or the common register with data from arrays. Logical operations may be performed between the current PE register contents and the array data. The language has mnemonics for the common logical operations, while the machine supports all 16 functions of two logical variables.

* TM, Goodyear Aerospace Corporation, Akron, Ohio

A given load instruction can increment, decrement, or leave as is an array field pointer. Thus, a single one of these instructions can load registers, perform logic, and change pointer values. Operations to set, clear, or rotate the PE registers are included in this group.

The "store" array instructions are used to move PE or common register data into the arrays. A mask feature is provided that allows writing only in mask enabled array words. As with the load instructions, logical operations may be performed between the current PE registers contents and the array data. Also, the array field pointer can be incremented, or left unchanged.

The "associative search" array instructions allow the programmer to search for particular conditions in the arrays. Only those words enabled by the mask register take part in the searches. Searches can be performed that compare a value in the common register with a value in a field of all array words. Another variety of search compares one field of a word with a second field of the same word for all array words. Comparisons can be made for such conditions as equal, not equal, greater than, greater than or equal, etc. Maximum and minimum searches also can be performed. Combinations of searches yield such functions as between limits and next higher. Additional mnemonics in this group are provided to resolve multiple responders to the searches.

The "parallel move" instructions are provided to move an array memory field to another field within the same array word. As with searches, a word is active for this instruction only when enabled by the mask register. Types of moves are direct, complement the field, increment or decrement the field, and move the absolute value.

The "parallel arithmetic" array instructions allow the programmer to perform parallel operations in the arrays. These operations are subject to mask register word enabling. Arithmetic can use a value in the common register as one operand and a value in a field of all array words as the parallel operand. Alternatively, one field of a word can be arithmetically combined with a second field of the same word for all array words. Operations supplied by APPLE are add, subtract, multiply, divide, and square root.

Macro—A macro language is provided to increase the user's flexibility at assembly time.[6] The macro language has a large set of arithmetic, logical, relational, and string manipulation operators. Adding macro variable symbol handling, conditional expansion capability, and ability to nest macro calls make it possible to write powerful macro instructions. System and user macro libraries have been implemented.

Benefits to the user are the ability to define new mnemonics, redefine existing mnemonics, and conveniently generate standard instruction sequences.

Mnemonics have been added to the basic APPLE language by including macros in the system library. Primarily, the added mnemonics are floating point instructions. They are fixed field length operations in both single and double precision.

Building Load Modules—Software used to convert source language programs into executable load modules includes an APPLE assembler, macro-preprocessor, and relocating linker. Figure 2 shows this software and the flow of programs or modules through it.

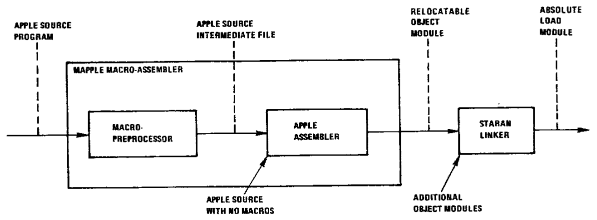

Figure 2—Language processing software

Building load modules begins with the original program written in APPLE. This source program may contain macro instructions. Translation of the source into a machine language object module is by MAPPLE (APPLE assembler with Macro-preprocessor on the front end). If it is known that the source program does not contain macro instructions, it is possible to input the source directly to the APPLE assembler.

A relocatable object module is converted to an absolute load module by the STARAN linker. Multiple object modules may be input to the linker since it has the function of resolving symbols defined across object module boundaries (global symbols) as well as adjusting addresses for relocation.

Use of the language processing software is fully described in the STARAN User's Guide.[7]

Execution control

Execution control software is discussed below, covering loading, executing, and debugging programs on STARAN. Four modules are involved: the loader, STARAN program supervisor, debug module, and control module.

Loader—Output of the STARAN linker is shown in Figure 2 as an absolute load module. The loader has the straightforward task of moving a load module into STARAN control memory beginning at the address specified in a text block. Options on loading are to load and not execute or to load and begin execution either at an address given with the load module or at one given with the load command. The load module is accessible from a user program to enable calling for a load from an executing program. This means that overlay modules can be brought in dynamically.

STARAN Program Supervisor (*SPS*)—The SPS is the software interface between the associative and sequential portions of STARAN. This module has services for system users when programming in APPLE and when programming a PDP-11 routine to interact with an APPLE program.

For the APPLE program, SPS makes the I/O instructions of the disk operating system (DOS) available, provides a program overlay capability, and provides a programmable interrupt to a PDP-11 routine. The PDP-11 routine inter-

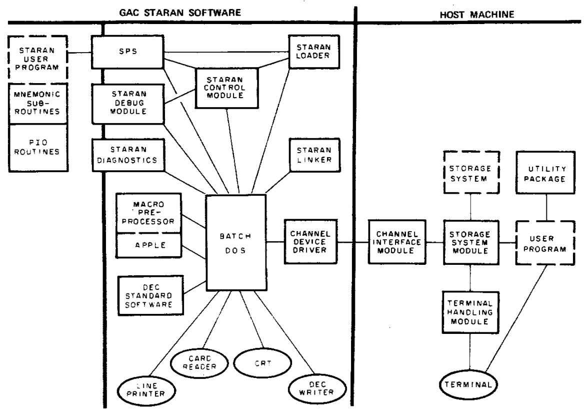

Figure 3—System software diagram

acts through a software link, which receives the APPLE interrupts, and through the issuing of control information to the associative control logic.

In addition, SPS supplies interface services. It transfers data between associative and sequential memory through the common memory window (Figure 1). SPS also fields associative processor error interrupts.

Concurrent execution of associative and sequential routines, with interaction, is made possible by SPS.

STARAN Debug Module (SDM)—The SDM helps the user debug APPLE programs by giving him control of the execution of the program being debugged, and access to memory and registers. Such features as single step, trace, and breakpoint provide good execution control. Dumps of all memory areas can be taken, with both word slice and bit slice available for the multi-dimensional access arrays. All memory locations also can be modified.

STARAN Control Module (SCM)—This final operational module is the interface between the user and execution of a STARAN program. By running SCM, the user enters a mode in which STARAN related commands are recognized. Such commands as start, halt, and continue execution are processed directly by SCM. When the load command is used, SCM passes control to the loader for that function. If debug aids are needed, a simple command adds all debug module features to SCM.

All the operational software modules are described more fully in the STARAN User's Guide.[7]

SOFTWARE FOR THE INTEGRATED MODE

General

The integrated use of the STARAN parallel processor and a host sequential computer makes additional software necessary. One major concern is the interface between the computers; this requires a software module in both machines. A second concern involves reasonable ease of use for the integrated mode; procedure packages are added as needed to satisfy this concern.

Figure 3 is a block diagram of the software modules in STARAN and a typical host machine. Interface software can be seen as the channel device driver in STARAN and the channel interface module in the host. Routines that might be added to simplify operation in the integrated mode are the storage system module to provide access to the host's storage, a terminal handling module to provide smooth interaction with a terminal user, and a set of utilities.

The STARAN/HIS-645 software

Figure 4 shows the relationship between software modules in STARAN and the HIS-645, which runs under the Multics time-shared operating system.[8] This facility exists at Rome Air Development Center (RADC), N. Y. and is described in a companion paper presented at this conference.[4] As indicated, Multics contains three categories of software: command level, user process, and system related. Command level software is brought into execution by user-supplied commands, as from a Multics terminal. User process software consists essentially of subroutines called from a user program. System-related software is the collection of routines that support use of the system, such as handling input and output, and are usually called indirectly by the user program.

Additional details on the design and use of this software are described in the STARAN/HIS-645 User's Guide.[9]

Interface Modules—The two modules for the interface, shown in Figure 4, are the 645 device driver in the STARAN

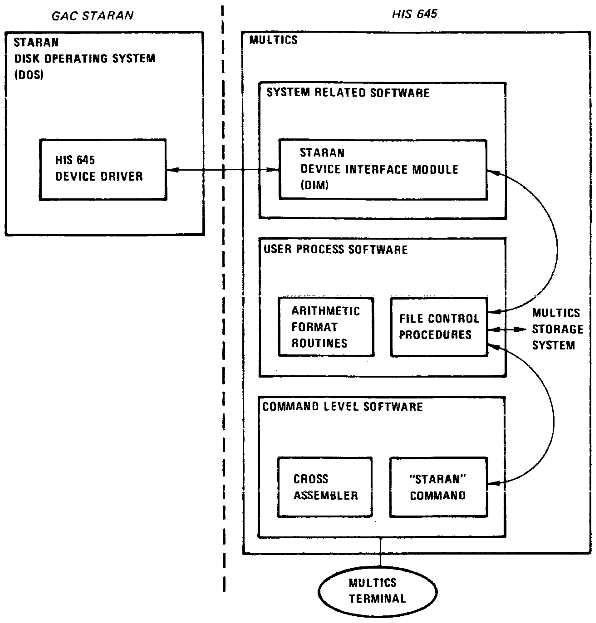

Figure 4—STARAN/645 system software relationship

batch disk operating system (DOS) and the STARAN device interface module (DIM).

The 645 device driver provides the interface between the DOS monitor and the 645 computer. It communicates with the monitor as do other device drivers for standard peripherals. If the device looks like an input for character information, then batch commands can come from it. The batch stream can be assigned to the device. This is the significance, for Multics, of the batch feature on the DOS.

In reality, the device treated by the 645 driver is used for much more than character input. The 645 appears as three logical devices.

One device looks like the disk, logically. The driver supports both ASCII and binary transfer modes, both formatted and unformatted. At any one time, up to 14 data-sets may be open on this device.

A second device looks like a card reader, logically. It is a read-only device with an ASCII transfer mode. This unit serves as the batch command stream input so a Multics user can control the system.

The third device looks like a paper tape punch, logically. It is a write-only device with ASCII and binary transfer modes. Job log output, in the integrated mode, is always assigned to this unit.

STARAN DIM—In Multics terminology, a device interface module (DIM) coordinates communications with a particular physical device. Data manipulation by the STARAN DIM assumes all Multics data is in character form. It converts characters into the form needed for output to STARAN and converts data received from STARAN into Multics character form. This means, for example, that Multics arithmetic data must be converted to a character form prior to output, and from characters following input. The conversion is done by a procedure superior to the DIM. The DIM also handles retransmission of bad data and reports a failure to its caller after a specific number of unsuccessful tries on the same data.

In the Multics software structure, the DIM is located in a position inferior to the file control procedures, shown in Figure 4 and described in the next part of this paper.

System Use Modules—The file control procedures (FCP) greatly simplify operation of STARAN from Multics. It enables a Multics user process (program) to interact with STARAN by initializing the interface, handling communication between the machines, and terminating the interface. The FCP also makes the necessary calls to the DIM to initialize and terminate the interface.

With FCP, a user process, executing in the 645, can call for STARAN, and it can pass commands, programs, and data to STARAN. The FCP raises the point at which the user becomes involved from sequences of calls to the DIM to a more symbolic call to FCP routines from the user process.

User involvement in the interface to STARAN is raised still higher from the user process to the Multics command level by a "STARAN" module. Essentially, this module is a supplied user process that passes parameters used in the terminal command to the FCP. The parameters identify the STARAN batch command stream input and output devices. The module calls appropriate FCP routines to establish interaction with STARAN.

In typical operation of STARAN from a terminal, this Multics command is used with STARAN commands also coming from the terminal. Initializing and terminating the interface are not a concern of the user. The Multics terminal becomes very similar to the STARAN control console when this module is used.

STARAN and the 645 differ in the lengths of their data representations. STARAN has a 32-bit control memory, while the 645 has a 36-bit word length. Arithmetic format routines are provided to convert either integer or floating point data between the 645 format and the format used by the DIM for transmission to STARAN.

A cross assembler has been written in PL/1. This is a functionally equivalent version of the MAPPLE assembler to be run in Multics. It is available to terminal users on the time-shared basis. It accepts APPLE and macro statements and produces STARAN object code in the Multics character format required by the DIM for transmission to STARAN.

STARAN/Σ5 integrated mode

A second method of interfacing STARAN with a host machine has been implemented in the Evaluation and Test Facility at Goodyear Aerospace Corporation. This facility has an XDS Σ5 as the host. The direct memory access capability of STARAN has been used to allow an 8K area of Σ5 memory to be used as STARAN control memory. Either programs or data may be stored here with control provided by interrupts between the machines. Software for this system is a communications library package with subroutines callable from FORTRAN or machine language in the Σ5.

CONCLUSION

A brief description has been given of software packages that compose the system for the operational STARAN parallel associative array processor. Also described is the additional software that makes STARAN operational when integrated with HIS-645 or XDS Σ5 sequential computers. The goal of all the software is to provide tools to use STARAN in the stand-alone and integrated modes. The tools are intended to increase convenience for the user and improve total system throughput.

Many modules have been discussed. Some of these are essentially transparent to the user, some may not be needed by certain users, and some may be required by all users. For stand-alone STARAN operation, the programmer must know APPLE and the use of the assembler and linker. He must be able to run the control module and load programs.

He will probably be interested in the debug module. The STARAN program superivsor is transparent for most users. It is not necessary to know any of the sequential control programs or languages.

REFERENCES

1. Batcher, K. E., *STARAN Parallel Processor System Hardware,* GER-15996, Goodyear Aerospace Corporation, 19 November 1973.
2. Rudolph, J. A., "A Production Implementation of an Associative Array Processor—STARAN," *1972 Fall Joint Computer Conference Proceedings* December 1972, pp. 229–241.
3. *STARAN Reference Manual,* GER-15636A, Goodyear Aerospace Corporation, September 1973.
4. Feldman, J. D. and L. C. Fulmer, *RADCAP: An Operational Parallel Processing Facility,* GER-15946B, Goodyear Aerospace Corporation, 21 December 1973.
5. *STARAN APPLE Programming Manual,* GER-15637A, Goodyear Aerospace Corporation, September 1973.
6. *STARAN MACRO Programming Manual,* GER-15643, Goodyear Aerospace Corporation, September 1973.
7. *STARAN User's Guide,* GER-15644, Goodyear Aerospace Corporation, September 1973.
8. Organick, E. I., *The Multics System,* MIT Press, 1972.
9. *STARAN/HIS-645 User's Guide,* GER-15641, Goodyear Aerospace Corporation, September 1973.