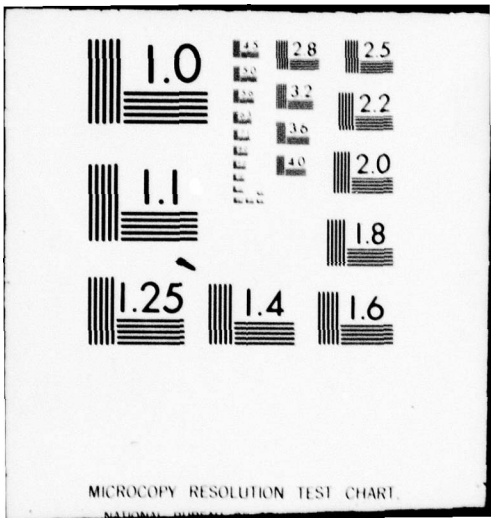
AD-A060 944 ARMY COMMUNICATIONS RESEARCH AND DEVELOPMENT COMMAND --ETC F/G 9/2
A COMPARISON OF THE EXISTING SUPPORT SOFTWARE BASES OF THE AN/G--ETC(U)
JUL 78 J WAGNER, H STONE

UNCLASSIFIED CORADCOM-78-10 NL

1 OF 1
AD A060 944

END
DATE FILMED
1-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

LEVEL II

(9) RESEARCH AND DEVELOPMENT TECHNICAL REPORT,
(14) CORADCOM- 78-10

(6) A COMPARISON OF THE EXISTING SUPPORT SOFTWARE BASES OF THE AN/GYQ-21 ARCHITECTURE AND THE CURRENT MILITARY ARCHITECTURES.

(10) James Wagner, H. Stone
CENTER FOR TACTICAL COMPUTER SYSTEMS
H. Stone
UNIVERSITY OF CALIFORNIA

(12) 79p.

(11) July 1978

DISTRIBUTION STATEMENT
Approved for public release;
distribution unlimited.

(16) 1L162701AH92

(11) B1

CORADCOM
US ARMY COMMUNICATION RESEARCH & DEVELOPMENT COMMAND
FORT MONMOUTH, NEW JERSEY 07703

410 489

NOTICES

Disclaimers

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CORADCOM-78-10	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Comparison of the Existing Support Software Bases of the AN/GYQ-21 Architecture and the Current Military Architectures		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James Wagner (CORADCOM) Dr. H. Stone (Univ of Mass)		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Tactical Computer Systems (CENTACS) DRDCO-TCS-BC		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1L1 62701 AH92R11701
11. CONTROLLING OFFICE NAME AND ADDRESS CENTACS DRDCO-TCS-BC Fort Monmouth, NJ 07703		12. REPORT DATE July 1978 13. NUMBER OF PAGES 76
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This report is Volume III of three Volumes: Volume I- Final Report Life Cycle Cost Analysis of Instruction-Set Architecture Standardization for Military Computer-Based Systems: Volume II - Phase II Comparative Evaluation of the MCF Computer Architectures.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Support Software, Instruction-Set Architecture, Software base, Software Tools

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A comparison of the support software bases of the four currently utilized military architectures and the AN/GYQ-21 (DEC PDP-11) is described. Approach taken in this evaluation is to model the software development process, hypothesize an ideal support software structure for that model and then apply that structure against each of the architectures in turn. The support software available for each architecture can then be listed and by estimating the development cost of each tool, the total software bases and deficiencies in → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

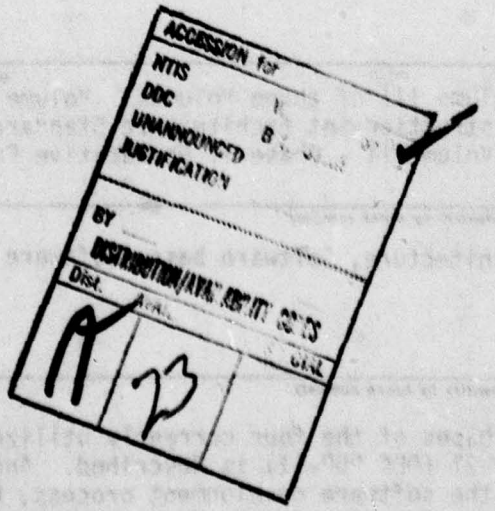
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. dollars of each architecture determined. The table below depicts the results of the evaluation:

	BASE	DEFICIENCY
AN/UYK-7	$13,325K	$10,570K
AN/UYK-20	$ 5,105K	$18,790
AN/UYK-19	$ 9,665K	$14,230K
AN/GYQ-21	$14,865K	$ 9,030K
AN/GYK-12	$ 4,445K	$19,450K

This evaluation is one of the major inputs to a life cycle cost analysis, reported separately, of 78 Army/Navy military computer based systems which attempts to determine the cost-effective approach to standardization of computer architectures.



CONTENTS

TABLES

1. INTRODUCTION

a. Scope. This report details the results of a study which compares the availability of support software for the AN/GYQ-21 architecture (DEC PDP-11) to that available for the current military architectures (AN/UYK-7, AN/UYK-19, AN/UYK-20 and AN/GYK-12). This report is Volume III of three Volumes: Volume I - Life Cycle Cost Analysis of Instruction-Set Architecture Standardization for Military Computer Based Systems: Volume II - Phase II Comparative Evaluation Of The MCF Computer Architectures, concerned with measuring the economic effects of standardization of computer instruction-set architectures on military computer-based systems. For the purpose of this report, support software is defined to include all system software and support software tools which aid in the development or maintenance of tactical software sytems. The results of this evaluation are one important input into a life cycle cost analysis, reported separately,[1] which determines the relative life-cycle cost of 78 Army/Navy military computer-based systems as a function of two basic architecture standardization scenarios: (1) adopting a single computer family architecture as a standard architecture versus (2) having at least four architecture families in the inventory. For Scenario (1) each of the military architectures and the commercial architecture DEC PDP-11, is taken, in turn, to be the architecture standard.

The method we used to compare the support software bases of the architectures was to develop a model of the software development process and from that, to structure a hypothetically complete support software environment. Based upon that we then designed an evaluation methodology and applied it to the architectures. The evaluation methodology is detailed in Section 2 and the results of the evaluation are summarized in Section 3 and detailed in Section 6. Section 4 details the software development model and Section 5 the support software structure.

b. Background

This report covers work that is an extension of work performed in 1975/1976 for the Army/Navy Computer Family Architecture (CFA) Selection Committee[2] that was evaluating which of the currently available architectures would be the best choice for a standard military computer family architecture. As part of that evaluation, a comparison of the software bases of the three finalist architectures, IBM 360, Interdata 7/32 and DEC PDP-11 was made.[3] The results of that study were one important input into two complementary life cycle cost models which contributed to the committee's recommendations of the DEC PDP-11 architecture as the best choice for a standard military computer family architecture. That effort may be considered as Phase I of the ongoing CFA selection process.

In 1977, Phase II of the CFA selection process was initiated to conduct a detailed life cycle cost analysis of the four Army/Navy military architectures versus the AN/GYQ-21 (DEC PDP-11). One important input into this life cycle analysis is the comparison of the support software bases of the five architectures. That comparison is the subject of this report. The overall results of the life cycle analysis are reported separately.[1]

2. EVALUATION METHODOLOGY

a. Overview. The evaluation methodology utilized herein is essentially identical to that utilized on the Phase I evaluation of the software bases of the three commercial finalist architectures. First, a model of the tactical software development is developed. Next, within the context of that development model, a complete set of support software tools is delineated along with an essential set of requirements for each tool. Then,the applicability of the tools is determined. The availability of each of these tools for the five architectures is then determined and a quantitative assessment of the software bases is made.

b. Software Development Model. In order to determine the universe of tools which in any way could be applied to the development of tactical software systems, a model of the software development process is developed. In particular, the process is partitioned into the following major activities: (1) Analyze Requirements, (2) Design Software, (3) Build System Tests, (4) Build and Unit Test Software (5) Integrate and System Test, and (6) Maintain System.

c. Support Software Structure. Within the context of the software development model defined above, tools are determined which support the activities of the model. Existing tools as well as tools in the R&D stage of development are delineated and requirements (essential characteristics) for each tool are compiled.

d. Desirable Support Software Base. Next,the applicability of the tools to the tactical software development process is determined by having CFA committee members, who represent a broad spectrum of Army/Navy project managers and laboratories, vote on the applicability of the tool to their activities. Those tools which fall below a predetermined threshold are then discarded from further consideration. The remaining tools constitute the desirable or target support software base.

e. Software Base Value. Next,the availability of each tool on each architecture is determined by querying the manufacturers and users of the architecture. The development cost for each tool is quantified by estimating the size of the tool in source lines of code and by converting source line to development cost, utilizing a programmer productivity figure. The sum of the development costs constitutes the present value of the support software base.

3. SUMMARY OF RESULTS

a. Overview. Two significant results are generated as a result of this study: first, which tools are considered important by tactical software developers, that is, the desirable support software base and, secondly, what is the comparative value of the existing support software base of each architecture under consideration.

b. Desirable Support Software Base. The desirable support software base was determined by balloting performed in the original CFA evaluation in 1976. Table 3-1 lists the most desirable tools in descending order of applicability.

TABLE 3-1. SUMMARY: DESIRABLE SUPPORT SOFTWARE BASE

Compilers
Macro Assembler
Interactive Source Language Editors
Interactive Symbolic Debuggers
Extended Overlay Linker
Test Case Design Advisor
Integrated Library
Text Processing System
Data Base Management System
GP System Simulator
Time Sharing OS + Virtual Machine Monitor
Language Independent Monitors
Test Data Generator
Non-Interactive Symbolic Debugger
Computer System Simulator
Batch Source Language Editor
Language Dependent Monitor
Time Sharing OS + Multiprocessing OS + Virtual Machine Monitor
Basic Assembler
Real Time OS
Test Instrumenters and Analyzers
Automatic SW Production and Test
Basic Linker
Standards Enforcers
Reformatters
Test Data Auditor
Simple Overlay Linker
Data Base Design Aid

c. Comparative Software Base Values. Table 3-2 depicts the major results of the evaluation of the software bases of the four military architectures and AN/GYQ-21 architecture.

TABLE 3-2. COMPARATIVE SOFTWARE BASE VALUES

	BASE	DEFICIENCY
AN/UYK-7	$13,325K	$10,570K
AN/UYK-20	$ 5,105K	$18,790K
AN/UYK-19	$ 9,665K	$14,230K
AN/GYQ-21	$14,865K	$ 9,030K
AN/GYK-12	$ 8,285K	$15,450K

The base figure for each architecture represents the total estimated cost of developing its existing support software while the deficiency figure represents the estimated cost of developing those tools which are in the desirable software base but which are non-existent on the architecture.

4. SOFTWARE DEVELOPMENT MODEL

a. General. The technical approach taken here is substantially that which was utilized for the comparison of the CFA finalist architectures support software. First a model of the tactical software development will be presented that provides a structure for the delineation of support software tools. This structure will then be utilized in the quantitative assessment of the architectures support software.

b. Software Development Activities. The software development process is partitioned into the following major activities: (1) Analyze Requirements, (2) Design Software, (3) Build System Tests, (4) Build and Unit Test Software, (5) Integrate and System Test, and (6) Maintain System. The following paragraphs describe these activities.

(1) Analyze Requirements (Activity 1). "Analyze Requirements" performs a decomposition of the user needs into the functions of the required system. Following decomposition and the development of a functional model, functions are allocated to hardware, software, firmware, and people. The results are then used to search a descriptive catalog of existing systems to locate suitable candidates for reuse or modification. The systems (if any) resulting from this search and any new functions that must be developed may be simulated to determine their gross performance characteristics. This activity is controlled by the analysts' knowledge of the current state of the art and the available budget for the proposed new system. If a decision is taken to proceed with development, a software functional design specification and a project schedule are produced which are used to control the "Design Software" activity.

(2) Design Software (Activity 2). During this activity, the software functional design specification is used to produce the implementation specification. A library of "proven algorithms" is available to assist in design. The "Design Software" activity may respond to the "Analyze Requirements" activity with "can't design" or "can't meet schedule". The output of this activity is the implementation specification which is used to control the software unit build and integration activities (Activities 4 and 5).

(3) Build System Tests (Activity 3). "Build System Tests" is the activity that results in the design and construction of system acceptance tests. Note that it is controlled by the same set of functional specifications that control the "Design Software" activity, and that it is unconstrained by and, therefore, may proceed in parallel with "Design Software" and "Build and Unit Test Software". A library of previously constructed tests that are presumably tied to sub-systems is available for reuse as directed by Activity 1. The output of this activity is the set of system test scenarios, drivers and monitors that will control Activity 5.

(4) Build and Unit Test Software (Activity 4). During this activity, the implementation specifications produced by Activity 2 are used to produce unit tested software modules. A library of previously constructed modules is available for reuse (with or without modification).

(5) Integrate and System Test (Activity 5). During this activity, the modules produced by Activity 4 and the interface and subsystem specifications produced by Activity 3 are used to bind the system components into their final form. The system is then exercised to validate the system using the test scenarios and monitors provided by Activity 3. The final output of this activity is the completed system released to the maintenance, distribution and configuration control activity (Activity 6). Integration and/or test failures are reflected back to the design, test, production or software production activities.

(6) Maintain System Activity (Activity 6). The final development activity, "Maintain System" is primarily a clearing house and control center for the reception, evaluation, and control of engineering change requests. These requests are routed to the appropriate activity for implementation, design, or analysis. The maintenance function distributes configuration controlled systems to the users, and releases the results of the development effort into the available technology data base.

5. SUPPORT SOFTWARE STRUCTURE.

a. General.

The software base will be structured, in part, by partitioning the software tool types according to the specific development activities they support. However, this approach may ignore that part of the software base that supports the operation of such tools and does not clearly indicate those tools that support more than one activity. To make such software visible, the software base will be structured further through a layered approach that will provide insight into the relationship among software base components.

There are at least five distinct virtual layers associated with an operational computer system and three layers of software that support the software development process. Layer 0, the innermost layer, represents the bare computer hardware including items such as processors, channels, main storage, mass storage, bulk I/O, archival storage, hardware monitors, terminals, sensors, and communications interface devices. Layers 1 through 3 "reside" on the hardware and collectively provide the virtual machine capability that is necessary to support layer 4, the applications software. In the following paragraphs, layers 1 through 3 are described along with short descriptions of the tools that reside in these layers and the relationship of such tools to the various development activities.

b. Layer 3: Functional Support Tools. Layer 3 contains those tools that provide direct support to the software development activities. These are the tools with which the applications software developer has the greatest interaction. Layer 3 tools will be related to the specific development activites they support.

(1) Layer 3 Tool Types That Support Activity 1 (Analyze Requirements). The types of tools that are directly applicable to requirement analysis are listed below:

(a) General Purpose System Simulators. Allows a user to construct a computer model of a real or proposed system and to perform simulation experiments to determine the behavior of the model under various operational conditions.

(b) System Description Languages and Analyzers. Assist system analysts in describing the functional characteristics of a system and in validating the consistency and completeness of a functional decomposition.

(2) Layer 3 Tool Types That Support Activity 2 (Design Software). The types of tools that are directly applicable to software design are listed below:

(a) Computer System Simulators. Similar in nature to the general purpose simulator except that its basic building blocks represent real computer system components whose modeled behavior approximates the throughputs, capacities, and access times achievable on the modeled equipments.

(b) Data Base Design Aids. Assist data base designers in grouping data elements into logical record classes and in determining the relationships among logical record classes implicit in either the nature of the data or the usage of the data.

(c) Data Dictionary Systems. Assist data base designers in managing the data definition activities.

(3) Layer 3 Tool Types That Support Activity 3 (Build System Tests). The types of tools required to support system test construction are listed below:

(a) Test Data Generators. Create data files for testing and validating programs.

(b) Test Data Auditors. Compare data files against specification and produce reports of discrepancies and/or compliance.

(c) Test Case Design Advisors. Analyze programs written in a high level language and present the results of that analysis in a form suitable to assist test case designers in the selection of test data.

(d) Test Instruments and Analyzers. Instrument modules under test so as to collect data characterizing and behavior of the module.

(4) Layer 3 Tool Types That Support Activity 4 (Build and Unit Test Software). The types of tools that are required to support the program development and unit test activity are listed below:

(a) Assemblers. Allow programs to be coded in a symbolic language in which statements generally correspond to a single machine instruction. Specific tools include Basic Assemblers and Macro Assemblers. (hosted and self-hosted)

(b) Compilers. Translate programs written in a high level language into either relocatable object code acceptable to a Linker or assembly language acceptable to an Assembler. (hosted and self-hosted)

(c) Instruction Simulators. Simulates the instruction set of the target machine on a host machine. Provides trace, snapshot and dump capabilities.

(d) Linkers. Combine the text produced by separate invocations of Compilers and Assemblers ("object modules") into executable code strings ("load modules" or "core images") that can be loaded into the computer's main storage and executed without further processing. Specific tools are Basic Linkers, Simple Overlay Linker, and Extended Overlay Linker.

(e) Debugging Aids. Assist the programmer in locating the sources of program

errors that have been discovered during unit testing, usually by giving him some control over the execution of the module under test that is external to the normal program code. Specific tools are Interactive Symbolic Debuggers, Non-Interactive Symbolic Debuggers, Interactive Absolute Debuggers and Non-Interactive Absolute Debuggers.

(f) Module Libraries and Change Control Systems. Provide computer controlled maintenance of groups of related source modules (programs), object modules (the output of Assemblers and Compilers), and load modules (the output of Linkers). Specific tools are Basic Libraries, Integrated Libraries, and Automatic Software Production and Test Systems.

(g) Performance Monitors. Assist the programmer in quantifying the resource consumption characteristics of a program and in isolating performance-critical areas. Specific tools are Language Dependent Monitors and Language Independent Monitors.

(h) Standards Enforcers. Allow source programs to be examined automatically and checked for conformance to installation-defined standards of format, content, and usage.

(i) Preprocessors and Reformatters. Assist programmers in producing well-structured and readable programs by allowing the introduction of structured programming constructs into source programs for languages that do not have them, and by automatically controlling indentation, the placement of comments, etc., to produce readable listings.

(5) Layer 3 Tool Types That Support Activity 5(Integrate and System Test) and Activity 6 (Maintain System). There are no unique layer 3 tools that exist to support these activities. The tools that were listed for activities 1 through 4 are generally applicable to activities 5 and 6 at layer 3. Most of the tools used in practice that are specifically oriented to activity 5 are special-purpose, e.g., test environment tools (emulators, hot benches, system integration lab support, virtual machines), test drivers and special performance monitors.

c. Layer 2: General Support Services. The primary function of layer 2 tools is to provide a framework of common services that will allow the outputs of third layer functions to be stored, retrieved and inter-communicated. Second layer functions should be usable for common purposes across different third layer functions, and should serve to hide (where possible) differences between first layer and third layer functions. Layer 2 tool types provide general support to all of the software development activities. These tool types are summarized as follows:

(1) Data Base Management Systems. Allow the user of a computer system to define the contents of and the logical relationships between collections of data items that represent some useful abstraction of a real-world phenomenon (tactical command and control system, the modules and documentation of a system of computer programs) without being concerned with the physical mechanics of storing, locating, and retrieving items or groups of items.

(2) PERT/CPM Systems. Assist managers in planning and controlling project activities.

(3) Project Estimation Systems. Assist in the development of work breakdown structures and related performance standards for use in estimating project resource requirements.

(4) Documentation Aids. Assist in the preparation and maintenance of documentation about the modules of a system. Specific tools are Text Processing Systems, Flowchart Construction Languages and Automatic Flowcharters.

(5) Data Manipulation Utilities. Allow the system user to alter the format and content of data files independent of the logical significance of the data fields involved. Specific tools are Sort/Merge Programs and Editors (Interactive Source Language Editors, Interactive Object Module Editors, Batch Source Language Editors, and Batch Object Module Editors).

(6) Information Retrieval Systems. General purpose application programs operating either on-line (interactively) or in batch that interpret user requests to locate and display information that is stored either within a structured data base or within separate files. Specific tools are Query Language Systems and Report Writers.

d. Layer 1: Operating System Services. Layer 1 implements the operating system services that present a "virtual machine" interface to the services/tools at layers 2 and 3 and manage the real system hardware. The layer 1 tool types are generally applicable across all of the software development activities. Layer 1 tool types/capabilities are listed below:

(1) Basic Operating Systems (BOS). Runs single user processes from initiation to termination. May or may not overlap I/O with execution. Provides basic I/O support that allows user to refer to files symbolically and to read and write them without knowing the hardware details of the I/O Interface. Provides basic batch supervisor services that control normal and abnormal job termination, job to job transition, and operator communication. Provides a minimum base for program development by supporting at least one language translator and/or linker/loader.

(2) Multiprogramming Operating System (MOS). Provides all of the services of the Basic Operating System. Supports the concurrent execution of two or more user jobs by allowing the execution of any job to be suspended while another is executed without any special programming considerations in the user job. Prevents concurrently executing user jobs from accidentally or intentionally destroying each other or the supervisor.

(3) Multiprocessor Operating System (MPOS). Allows the computing load to be spread across more than one processor based on automatic (programmed) load-leveling algorithms or operator control, but does not require special case programming

in the user job. Multiprocessor Operating Systems include the shared storage, loosely coupled, and networked types.

(4) Virtual Machine Monitor (VMM). The operating system presents an interface to the user program that makes it appear that the program is executing on a real computing system.

(5) Time-Sharing Operating System (TSOS). This is a variant of the multiprogramming operating system in which system resources are allocated to user jobs in such a way that all jobs appear to progress at the same rate. In addition, users are allowed to "interact" with and receive output from their jobs via terminals. Such systems are optimized for response rather than throughput.

(6) Real-Time Operating Systems (RTOS). Allows user jobs to be executed within specified short time limits.

5. RESULTS

a. General. In order to compare the architectures, a procedure was required which could quantify their support software investment. The items deemed to be identifiable, obtainable and translatable into financial data were applicability and availability.

b. Applicability. Applicability was determined by utilizing the results obtained by the balloting of the CFA members in the Spring of 1976. Applicability, then, involved the determination of the functional relevance of a given software base component (tool type) to the development of military tactical software systems. Applicability was not intended to be a binary criterion but was to be a measure of the potential importance of the component, ranging from "not applicable" to "essential". Factors that were to be considered in determining the importance of a tool type, in addition to essentiality, included spectrum coverage, economic impact, software size, and the number of different instances of the same tool type for a given CFA. Committee members were sent a list of support software tools delineated in the previous section, were allotted 5,000 points, and were asked to distribute these points over the tools thereby indicating the relative applicability/importance of each tool to their activities. The guidance given to the members was that if a tool was essential for military computer software development then it should deserve more weight than one that is only nice to have. Consideration was also to be given to spectrum coverage, i.e., the utility of the component across development activities as well as across development disciplines. Consideration was to be given to economic impact, the potential cost savings that may be realized through use of the tool type.

The results of this applicability balloting were compiled and the tool list was ordered in terms of points received. A predetermined threshold of 1000 points was then applied against the list. In other words, any tool which received less than that threshold would no longer be considered. The justification for this was that DOD could not afford to build tools which a representative spectrum of system developers determined were not very applicable.

The applicability results eliminated approximately half of the tools delineated in the previous section leaving 28 tools to be utilized in the availability phase of the evaluation. These tools are delineated in Table 6-1.

c. Availability. The availability phase of the evaluation consisted of visiting or contacting, for each of the architectures, manufacturers, project managers, government laboratories and any other known user of the architecture in order to determine what software tools existed. The annex of this document entitled "Independent Audit of the Availability of Support Software for the AN/GYQ-21 Architecture and Current Military Architectures" details the results of the availability phase of this evaluation.

TABLE 6-1. COMPILED RESULTS OF THE APPLICABILITY BALLOTING

9708	Compilers
4405	Macro Assemblers
4317	Interactive Source Language Editors
3752	Interactive Symbolic Debuggers
3367	Extended Overlay Linker
3010	Test Case Design Advisors
2969	Integrated Library
2687	Text Processing System
2677	DBMS
2380	GP System Simulator
2270	TSOS + VMM
1872	Language Independent Monitors
1688	Test Data Generator
1645	Non-Interactive Symbolic Debugger
1623	Computer System Simulator
1535	Batch Source Language Editors
1418	Language Dependent Monitors
1400	TSOS + MPOS + VMM
1390	Basic Assembler
1310	RTOS + TSOC
1217	Test Instrumenters and Analyzers
1187	Automatic SW Production and Test
1158	Basic Linker
1140	Standards Enforcers
1110	Reformatters
1095	Test Data Auditor
1008	Simple Overlay Linker
1006	Data Base Design Aid
975	Sort/Merge
880	Preprocessors
855	Basic or Loosely Coupled Library
834	Project Estimation System
808	System Description Language and Analyzer
795	RTOS
792	PERT/CPM
760	TSOS
757	MOS
705	BOS
675	Report Writers
658	Interactive Object Module Editors
642	Interactive Absolute Debugger

TABLE 6-1. COMPILED RESULTS OF THE APPLICABILITY BALLOTING (CONTD)

622	Data Dictionary System
615	RTOS + MPOS
607	Query Language Systems
600	PDL
600	Batch Object Module Editors
600	Automatic Flowcharters
558	Flowchart Construction Languages
513	Non-Interactive Absolute Debugger
410	MPOS
365	TSOS + MPOS
350	BOS + VMM
315	MOS + VMM
305	RTOS + MOS
250	MPOS + VMM

c. Consolidation of Results. Enough data was now available to determine the relative current software dollar investment as well as the current software dollar deficiency of the architectures. In order to do this, a development cost for each tool was needed. It was known that the architecture manufacturers and software vendors considered such information to be proprietary. Therefore, the following approach was taken: First each manufacturer and project manager was requested to provide the source code size (disregarding comments) for his available tools as well as the language which the tool was written in, and the object code size in instructions. Second, a productivity figure was needed. F. Brooks, "The Mythical Man-month" [4] was considered to be the best source since it compiled productivity figures from IBM's OS/360 development as well as Bell Laboratories ESS software development. Brooks cites 600 lines of code per man-year for operating system development and 2,000 lines of code per man-year for other software development. It was felt that the state-of-the-art in operating systems had improved significantly since his data was obtained (nearly ten years ago) and thus a figure of 1,000 and 2,000 lines of code per man-year for operating system and other support software, respectively, was decided upon. Third, a fully loaded price per man-year of $70,000 was assumed. Table 6-2 depicts the source code data and estimated development cost for each of the tools. The next section delineates the final results of the evaluation process.

d. Non-Target Oriented Tasks. During the process of this evaluation, it occurred to the investigators that certain of the tools of Table 6-1 were of such a general non-target oriented nature that if one tool of the type were available on some machine then it would probably not be developed by the MCF program and therefore, credit should be given to all architectures for this type tool. Tools in this class (above the threshold line of Table 6-1) are:

(1) General Purpose System Simulator

(2) Test Data Generator

(3) Test Data Auditor

(4) Integrated Library

(5) Automatic Software Production and Test

(6) Standards Enforcers

(7) Text Processing Systems

(8) Interactive Source Language Editors

(9) Batch Source Language Editors

All of the evaluations detailed in this report are based upon this assumption.

TABLE 6-2. ESTIMATES OF TOOLS SIZE BY SOURCE LINES OF CODE AND ESTIMATED TOOL COST

	IBM (Source Lines)	UNIVAC (Source Lines)	Interdata (Source Lines)	UNIVAC 1100 (Source Lines)	Litton (Source Lines)	DOLI (Source Lines)	Military AYK-7 (Source Lines)	Gov't Estimate ($)	Est. ($)
1.1.1 General Purpose System Simulator	20,000 Assembly								700K
1.2.1 Computer System Simulator								420K	420K
1.2.2 Data Base Design Aid								1150K	1150K
1.3.1 Test Data Generator								350K	350K
1.3.2 Test Data Auditor								140K	140K
1.3.3 Test Case Design Advisors								2100K (for all lan-guages)	2100K (for all languages)
1.3.3.1 FORTRAN									
1.3.3.2 COBOL									
1.3.3.3 CMS-2									
1.3.3.4 JOVIAL									
1.3.3.5 TACPOL									
1.3.4 Test Instruments & Analyzers								1400K (for all lan-guages)	1400K (for all languages)
1.3.4.1 FORTRAN									
1.3.4.2 COBOL									
1.3.4.3 JOVIAL									
1.3.4.5 TACPOL									

TABLE 6-2. ESTIMATES OF TOOLS SIZE BY SOURCE LINES OF CODE AND ESTIMATED TOOL COST (CONTINUED)

	IBM (Source Lines)	DEC (Source Lines)	Interdata (Source Lines)	UNIVAC UYK-20 (Source Lines)	Litton (Source Lines)	ROLM (Source Lines)	UNIVAC UYK-7 (Source Lines)	Gov't Estimate ($)	Est. ($)
1.4.1.1 Basic Assembler						3870			135K
1.4.1.2 Macro Assembler	40,000 Assembly			6,000		11,000 Assembly	32,000		800K
1.4.2 Compilers									
1.4.2.1 FORTRAN	75,000 FORTRAN & Assembly			14,000		9,500 14,000			1000K
1.4.2.1 COBOL	300,000 Assembly					60,000			6,300K
1.4.2.3 CMS-2				90,000 ($300K)			175,000		3,300K
1.4.2.4 JOVIAL							45,000		1,600K
1.4.2.5 TACPOL								1000K	1,000K
1.4.3.1 Basic Linker				1800 Assembly	5000	4400			130K
1.4.3.2 Simple Overlay Linker								210K	210K

TABLE 6.2. ESTIMATES OF TOOLS SIZE BY SOURCE LINES OF CODE AND ESTIMATED TOOL COST (CONTINUED)

	IBM (Source Lines)	DEC (Source Lines)	Interdata (Source Lines)	[illegible] (Source Lines)	Litton (Source Lines)	POL™ (Source Lines)	UNIVAC UYK-7 (Source Lines)	Gov't Estimate ($)	Est ($)
1.4.3.3 Extended Overlay Linker	15,000 Assembly								500K
1.4.4.1 Interactive Symbolic Debugger									
1.4.4.1.1 ASSEMBLY						1024			
1.4.4.1.2 FORTRAN	15,000 HOL								300K per lan guage
1.4.4.1.3 COBOL									
1.4.4.1.4 CMS-2									
1.4.4.1.5 JOVIAL									
1.4.4.1.6 TACPOL									
1.4.4.3 Non-Interactive Symbolic Debugger								200K (for all lan-guages)	200K (for all lan-guages)
1.4.4.3.1 ASSEMBLY									
1.4.4.3.2 FORTRAN									
1.4.4.3.3 COBOL									
1.4.4.3.4 CMS-2									
1.4.4.3.5 JOVIAL									
1.4.4.3.6 TACPOL									

TABLE 6-2. ESTIMATES OF TOOLS SIZE BY SOURCE LINES OF CODE AND ESTIMATED TOOL COST (CONTINUED)

	IBM (Source Lines)	DEC (Source Lines)	Interdata (Source Lines)	UNIVAC UYK-20 (Source Lines)	Litton (Source Lines)	ROLM (Source Lines)	UNIVAC UYK-7 (Source Lines)	Gov't Estimate ($)	Est. ($)
1.4.5.2 Integrated Library					2600				100K
1.4.5.3 Automatic Software Production & Test								1000K	1000K
1.4.6.1 Language Dependent Monitors						200			50K per language
1.4.6.1.1 ASSEMBLY									
1.4.6.1.2 FORTRAN									
1.4.6.1.3 COBOL									
1.4.6.1.4 CMS-2									
1.4.6.1.5 JOVIAL									
1.4.6.1.6 TACPOL									
1.4.6.2 Language Independent Monitor								210K	210K
1.4.7 Standard Enforcers								420K (for all languages)	420K (for all languages)
1.4.7.1 FORTRAN									
1.4.7.2 COBOL									
1.4.7.3 CMS-2									
1.4.7.4 JOVIAL									
1.4.7.5 TACPOL									

TABLE 6-2. ESTIMATES OF TOOLS SIZE BY SOURCE LINES OF CODE AND ESTIMATED TOOL COST (CONTINUED)

		IBM (Source Lines)	DEC (Source Lines)	Interdata (Source Lines)	UNIVAC UYK-20 (Source Lines)	Litton (Source Lines)	ROLM (Source Lines)	UNIVAC UYK-7 (Source Lines)	Gov't Estimate ($)	Est. ($)
1.4.8	Reformatters								560K (for all languages)	560K (for all languages)
	1.4.8.1 FORTRAN									
	1.4.8.2 COBOL									
	1.4.8.3 CMS-2									
	1.4.8.4 JOVIAL									
	1.4.8.5 TACPOL									
2.1	Data Base Management System	119K Assembly Language 32K Run-time Words								4200K
2.3.1	Text Processing System								630K	630K
2.4.2.1	Interactive Source Language Editor						3700			130K
2.4.2.3	Batch Source Language Editor				2100 Assembly		580 FORTRAN			100K

TABLE 6-2. ESTIMATES OF TOOLS SIZE BY SOURCE LINES OF CODE AND ESTIMATED TOOL COST (CONTINUED)

	IBM (Source Lines)	DEC (Source Lines)	Interdata (Source Lines)	UNIVAC UYK-20 (Source Lines)	Litton (Source Lines)	ROLM (Source Lines)	UNIVAC UYK-7 (Source Lines)	Gov't Estimate ($)	Est. ($)
3.9 RTOS + TSOS	48K Assembly Language								3500K
3.13 TSOS + VM1								2800K	2800K
3.14 TSOS + MPOS + VM1								1400K	1400K
Instruction Simulator				23,000	1600	4000			350K
Cross-Assembler				24,000					340K
Cross-Compiler								1000K per language	1000K per language
FORTRAN COBOL CMS-2 JOVIAL TACPOL									

f. Results Assuming All Languages Required. Tables 6-3 to 6-7 depict the software bases and individual tools estimated cost for each of the military architectures and the AN/GYQ-21 assuming that a compiler and associated language dependent tools are necessary for all of the languages delineated below:

(1) COBOL

(2) FORTRAN

(3) CMS-2

(4) JOVIAL

(5) TACPOL

Tables 6-8 to 6-12 list the tools deficient for each of the architectures making an assumption identical to that utilized above. Table 6-13 depicts a summary in dollars for all of the architectures.

g. Results Assuming One Language Plus DOD-1 Required And Unequal Compiler Costs.

Tables 6-14 to 6-24 represent the results of an analysis based upon a slightly different, but probably more realistic assumption. First it was assumed that the tools required for each architecture are the language independent tools, plus those tools that are language dependent and support those languages currently used extensively or likely to be used in the future. For example, we only require language dependent tools for CMS-2 for the AN/UYK-7 and AN/UYK-20, TACPOL for the AN/GYK-12 and FORTRAN for the AN/UYK-19 and AN/GYQ-21. Secondly, since the DOD Common Higher Order Language is likely to be required for all military computers in the future, we assumed that each architecture is required to have a DOD-1 compiler and all other DOD-1 dependent support tools.

Specifically, Tables 6-14 to 6-18 represent the tools available for each architecture and the estimated cost for each tool. Tables 6-19 to 6-23 represent the tools available for each architecture and the estimated cost for each tool. Tables 6-19 to 6-23 represent the deficiencies of each architecture and Table 6-24 depicts a summary of the total bases and deficiencies in dollars for each architecture.

A comment should be made concerning the estimated compiler cost for the DOD Common Higher Order Language (DOD-1). At first glance, the estimated cost of 600K seems to be inconsistent with the estimates used for compilers of other languages. However, the DOD-1 compiler will be implemented with a common root front end and will be developed independently of the MCF Program by ARPA. All that will be necessary if one wants to generate a compiler for a particular machine is to write a code generator for that machine. Therefore, it was felt that the estimated cost of 600K for such a compiler was realistic.

h. Results Assuming One Language Plus DOD-1 Required and Equal Compiler Costs. Table 6-24 represents the result of an analysis based upon assuming unequal compiler development costs. While these compiler development values reflect estimates based upon actual data or from source lines of code in the compiler, it is probably unrealistic to estimate different development costs for different languages. Therefore, Table 6-25 represents a similar analysis to that of Table 6-24, but the development cost of each compiler for the current languages has been estimated at $1,000,000.

TABLE 6-3. AN/UYK-7 SOFTWARE BASE

1.	Test Instrumenters & Analyzers:	280K
	FORTRAN	
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compilers	5600K
	FORTRAN CMS-2 JOVIAL	
5.	Basic Linker	130K
6.	Interactive Debugging Aids	800K
	Assembler CMS-2	
7.	Non-Interactive Debugging Aids	50K
	CMS-2	
8.	Language Independent Monitor	210K
9.	Reformatter	110K
	FORTRAN	
10.	Text Processing System	630K
11.	Interactive Source Editor	130K
12.	Batch Source Editor	100K
13.	RTOS + TSOS	3500K
14.	Instruction Simulator	350K
15.	Data Base Management System	4200K

TABLE 6-3. AN/UYK-7 SOFTWARE BASE (CONTD)

16. General Purpose System Simulator	700K
17. Test Data Generator	350K
18. Test Data Auditor	140K
19. Integrated Library	100K
TOTAL	$18,315K

TABLE 6-4. AN/UYK-20 SOFTWARE BASE

1.	Test Instrumenter & Analyzer	560K
	FORTRAN CMS-2	
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compilers	4300K
	FORTRAN CMS-2	
5.	Basic Linker	130K
6.	Language Dependent Monitor Assembler	50K
7.	Language Independent Monitor	210K
8.	Reformatter	110K
	FORTRAN	
9.	Batch Source Editor	130K
10.	Instruction Simulator	350K
11.	General Purpose System Simulator	700K
12.	Test Data Generator	350K
13.	Test Data Auditor	140K
14.	Integrated Library	100K
15.	Text Processing System	630K
16.	Interactive Source Editor	130K
	Total	$8,825K

TABLE 6-5. AN/UYK-19 SOFTWARE BASE

1.	Test Instrumenter and Analyzer:	
	FORTRAN	280K
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compiler	10,600K
	FORTRAN - COBOL - CMS-2	
5.	Basic Linker	130K
6.	Simple Overlay Linker	210K
7.	Interactive Debugging Aid	600K
	Assembly - COBOL	
8.	Reformatter	
	FORTRAN	110K
9.	Data Base Management Syster	4,200K
10.	Interactive Source Editor	120K
11.	Batch Source Editor	100K
12.	Instruction Simulator	350K
13.	General Purpose System Simulator	700K
14.	Test Data Generator	350K
15.	Test Data Auditor	140K
16.	Integrated Library	100K
17.	Text Processing Syster	630K
	TOTAL	$19,565K

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

TABLE 6-6. AN/GYQ-21 SOFTWARE BASE

1.	Data Base Design Aid	1150K
2.	Test Instrumentor & Analyzer	
	FORTRAN	280K
3.	Assembler	135K
4.	Macro Assembler	800K
5.	Compilers:	7300K
	FORTRAN COBOL	
6.	Basic Linker	130K
7.	Simple Overlay Linker	210K
8.	Extended Overlay Linker	500K
9.	Interactive Debugging Aids:	600K
	Assembly COBOL	
10.	Non-interactive Debugging Aid:	100K
	FORTRAN COBOL	
11.	Integrated Library	100K
12.	Reformatters:	
	FORTRAN	110K
13.	Data Base Management System	4200K
14.	Text Processing System	630K
15.	Interactive Source Editor	130K

TABLE 6-6. AN/GYQ-21 SOFTWARE BASE (CONTD)

16.	Batch Source Editor	100K
17.	RTOS + TSOS	3,500K
18.	Instruction Simulator	350K
19.	General Purpose System Simulator	700K
20.	Test Data Generator	350K
21.	Test Data Auditor	140K
	TOTAL	21,515K

TABLE 6-7. AN/GYK-12 SOFTWARE BASE

1.	Test Instrumenter and Analyzer	
	TACPOL	280K
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compilers	
	TACPOL	1,000K
5.	Basic Linker	130K
6.	Non-interactive Debugging Aids	50K
	TACPOL	
7.	Batch Source Editor	100K
8.	General Purpose System Simulator	700K
9.	Test Data Generator	350K
10.	Test Data Auditor	140K
11.	Text Processing System	620K
12.	Interactive Source Editor	120K
	TOTAL	$4,445K

TABLE 6-8. AN/UYK-7 SOFTWARE DEFICIENCIES

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisor	2100K
	FORTRAN	
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
4.	Test Instrumenter & Analyzer	1100K
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
5.	Compilers	7300K
	COBOL	
	TACPOL	
6.	Simple Overlay Linker	210K
7.	Extended Overlay Linker	500K
8.	Interactive Debugging Aids	1000K
	COBOL	
	FORTRAN	
	JOVIAL	
	TACPOL	
9.	Non-Interactive Debugging Aids	260K
	Assembly	
	FORTRAN	
	COBOL	
	JOVIAL	
	TACPOL	
10.	Automatic Software Production Test	1000K

TABLE 6-8. AN/UYK-7 SOFTWARE DEFICIENCIES (CONTD)

11.	Language Dependent Monitor	300K
	Assembly FORTRAN COBOL CMS-2 JOVIAL TACPOL	
12.	Standards Enforcer	420K
	FORTRAN COBOL CMS-2 JOVIAL TACPOL	
13.	Reformatter	450K
	COBOL CMS-2 JOVIAL TACPOL	
14.	RTOS + TSOS + VMM	2,800K
15.	TSOS + MPOS + VMM	1,400K
	TOTAL	$20,410K

TABLE 6-9. AN/UYK-20 SOFTWARE DEFICIENCIES

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisor	1200K
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
4.	Test Instrumenter & Analyzer	1100K
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
5.	Compiler	8900K
	COBOL	
	JOVIAL	
	TACPOL	
6.	Simple Overlay Linker	210K
7.	Extended Overlay Linker	500K
8.	Interactive Debugging Aids	1800K
	Assembly	
	COBOL	
	FORTRAN	
	CMS-2	
	JOVIAL	
	TACPOL	
9.	Non-Interactive Debugging Aids	280K
	Assembly	
	COBOL	
	FORTRAN	
	CMS-2	
	JOVIAL	
	TACPOL	

TABLE 6-9. AN/UYK-20 SOFTWARE DEFICIENCIES (CONTD)

10.	Automatic Software Production and Test	1000K
11.	Language Dependent Monitors	250K
	COBOL FORTRAN CMS-2 JOVIAL TACPOL	
12.	Standards Enforcers	420K
	FORTRAN COBOL CMS-2 JOVIAL TACPOL	
13.	Reformattors	450K
	COBOL CMS-2 JOVIAL TACPOL	
14.	Data Base Management System	4200K
15.	RTOS + TSOS	3500K
16.	RTOS + TSOS + VMM	2800K
17.	TSOS + MPOS + VMM	1400K
	TOTAL	$29,580K

TABLE 6-10. AN/UYK-19 SOFTWARE DEFICIENCIES

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisors	2100K
	FORTRAN	
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
4.	Test Instrumenters & Analyzers	1100K
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
5.	Compilers	2600K
	JOVIAL	
	TACPOL	
6.	Extended Overlay Linker	500K
7.	Interactive Debugging Aids	1200K
	FORTRAN	
	CMS-2	
	JOVIAL	
	TACPOL	
8.	Non-Interactive Debugging Aids	240K
	FORTRAN	
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
	Assembly	

TABLE 6-10. AN/UYK-19 SOFTWARE DEFICIENCIES (CONTD)

9.	Automatic Software Production and Test	1,000K
10.	Language Independent Monitor	210K
11.	Language Dependent Monitor	300K
	Assembly FORTRAN COBOL CMS-2 JOVIAL TACPOL	
12.	Standards Enforcers	420K
	FORTRAN COBOL CMS-2 JOVIAL TACPOL	
13.	Reformatter	450K
	COBOL CMS-2 JOVIAL TACPOL	
14.	RTOS + TSOS	3,500K
15.	RTOS + TSOS + VMM	2,800K
16.	TSOS + MPOS + VMM	1,400K
	TOTAL ...	$19,390K

TABLE 6-11. AN/GYQ-21 SOFTWARE DEFICIENCIES

1.	Computer System Simulator	420K
2.	Test Case Design Advisors	2100K
	FORTRAN	
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
3.	Test Instrumenters and Analyzers	1100K
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	
4.	Compilers	5900K
	CMS-2	
	JOVIAL	
	TACPOL	
5.	Interactive Debugging Aids	1200K
	FORTRAN	
	CMS-2	
	JOVIAL	
	TACPOL	
6.	Non-Interactive Debugging Aids	200K
	Assembly	
	CMS-2	
	JOVIAL	
	TACPOL	
7.	Automatic Software Production and Test	1000K
8.	Language Dependent Monitor	300K
	Assembly	
	FORTRAN	
	COBOL	
	CMS-2	
	JOVIAL	
	TACPOL	

TABLE 6-11. AN/GYQ-21 SOFTWARE DEFICIENCIES (CONTD)

9.	Language Independent Monitor	210K
10.	Standards Enforcers	420K
	FORTRAN CMS-2 COBOL JOVIAL TACPOL	
11.	Reformatter	450K
	COBOL CMS-2 JOVIAL TACPOL	
12.	RTOS + TSOS + VMM	2,800K
13.	TSOS + MPOS + VMM	1,400K
	TOTAL	$17,500K

TABLE 6-12. AN/GYK-12 SOFTWARE DEFICIENCIES

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisor	2100K
	FORTRAN COBOL CMS-2 JOVIAL TACPOL	
4.	Test Instrumenter & Analyzer	1100K
	FORTRAN COBOL CMS-2 JOVIAL	
5.	Compilers	12,200K
	FORTRAN COBOL JOVIAL CMS-2	
6.	Simple Overlay Linker	210K
7.	Extended Overlay Linker	500K
8.	Interactive Debugging Aids	1800K
	FORTRAN COBOL CMS-2 JOVIAL TACPOL Assembly	
9.	Non-Interactive Debugging Aids	250K
	Assembly FORTRAN COBOL CMS-2 JOVIAL	

TABLE 6-12. AN/GYK-12 SOFTWARE DEFICIENCIES (CONTD)

10.	Automatic Software Production and Test	1,000K
11.	Language Dependent Monitor	300K
	Assembly FORTRAN COBOL CMS-2 JOVIAL TACPOL	
12.	Language Independent Monitor	210K
13.	Reformatters	560K
	FORTRAN COBOL CMS-2 JOVIAL TACPOL	
14.	Data Base Management System	4,200K
15.	RTOS + TSOS	3,500K
16.	RTOS + TSOS + VMM	2,800K
17.	TSOS + MPOS + VMM	1,400K
18.	Instruction Simulator	350K
19.	Standards Enforcer	420K
	COBOL FORTRAN CMS-2 JOVIAL TACPOL	
21.	Integrated Librarian	100K
	TOTAL	$34,570K

TABLE 6-13. SOFTWARE BASE AND DEFICIENCY COMPARISON

	BASE	DEFICIENCY
AN/UYK-7	$18,315K	$20,410K
AN/UYK-20	$ 8,825K	$29,580K
AN/UYK-19	$19,565K	$19,390K
AN/GYQ-21	$21,515K	$17,500K
AN/GYK-12	$ 4,445K	$34,570K

TABLE 6-14. AN/UYK-7 SOFTWARE BASE

(Assuming only DOD-1 and One Currently Utilized Language Required Plus Associated Tools)

1.	Assembler	135K
2.	Macro Assembler	800K
3.	Compilers	3300K
	CMS-2	
4.	Basic Linker	130K
5.	Interactive Debugging Aids	800K
	Assembler CMS-2	
6.	Non-Interactive Debugging Aids	50K
	CMS-2	
7.	Language Independent Monitor	210K
8.	Text Processing System	630K
9.	Interactive Source Editor	130K
10.	Batch Source Editor	100K
11.	RTOS + TSOS	3500K
12.	Instruction Simulator	350K
13.	Data Base Management System	4200K
14.	General Purpose System Simulator	700K
15.	Test Data Generator	350K
16.	Test Data Auditor	140K
17.	Integrated Library	100K
	Total	$15,625K

TABLE 6-15. AN/UYK-20 SOFTWARE BASE

(Assuming only DOD-1 and One Currently Utilized Language Required plus Associated Tools)

1.	Test Instrumenter & Analyzer CMS-2	280K
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compilers CMS-2	3300K
5.	Basic Linker	130K
6.	Language Dependent Monitor Assembler	50K
7.	Language Independent Monitor	210K
8.	Batch Source Editor	100K
9.	Instruction Simulator	350K
10.	General Purpose System Simulator	700K
11.	Test Data Generator	350K
12.	Test Data Auditor	140K
13.	Integrated Library	100K
14.	Text Processing System	630K
15.	Interactive Source Editor	130K
	Total:	7405K

TABLE 6-16. AN/UYS-19 SOFTWARE BASE

(Assuming only JOVIAL and One Currently Utilized Language Required Plus Associated Tools)

1.	Test Instrumenter & Analyzer FORTRAN	280K
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compiler FORTRAN	1000K
5.	Basic Linker	130K
6.	Simple Overlay Linker	210K
7.	Interactive Debugging Aid Assembly	300K
8.	Reformatter FORTRAN	110K
9.	Data Base Management System	4200K
10.	Interactive Source Editor	130K
11.	Batch Source Editor	100K
12.	Instruction Simulator	350K
13.	General Purpose System Simulator	700K
14.	Test Data Generator	350K
15.	Test Data Auditor	140K
16.	Integrated Library	100K
17.	Text Processing System	630K
	Total:	$9665K

TABLE 6-17. AN/GYQ-21 SOFTWARE BASE

(Assuming only DOD-1 and One Currently Utilized Language Required plus Associated Tools)

1.	Data Base Design Aid	1,150K
2.	Test Instrumenter and Analyzer	280K
	FORTRAN	
3.	Assembler	135K
4.	Macro Assembler	800K
5.	Compilers	1,000K
	FORTRAN	
6.	Basic Linker	130K
7.	Simple Overlay Linker	210K
8.	Extended Overlay Linker	500K
9.	Interactive Debugging Aids	300K
	Assembly	
10.	Non-Interactive Debugging Aid	500K
	FORTRAN	
11.	Integrated Library	100K
12.	Reformatters	110K
	FORTRAN	
13.	Data Base Management System	4,200K
14.	Text Processing System	630K

TABLE 6-17. AN/GYQ-21 SOFTWARE BASE (CONTD)

15.	Interactive Source Editor	130K
16.	Batch Source Editor	100K
17.	RTOS + TSOS	3,500K
18.	Instruction Sirulator	350K
19.	General Purpose System Simulator	700K
20.	Test Data Generator	350K
21.	Test Data Auditor	140K
	TOTAL	14,865K

TABLE 6-18. AN/GYK-12 SOFTWARE BASE

(Assuming only DOD-1 and One Currently Utilized Language Required Plus Associated Tools)

1.	Test Instrumenter & Analyzer TACPOL	280K
2.	Assembler	135K
3.	Macro Assembler	800K
4.	Compilers TACPOL	1000K
5.	Basic Linker	130K
6.	Noninteractive Debugging Aids TACPOL	50K
7.	Batch Source Editor	100K
8.	General Purpose System Simulator	700K
9.	Test Data Generator	350K
10.	Test Data Auditor	140K
11.	Text Processing System	630K
12.	Interactive Source Editor	130K
	Total:	$4445K

TABLE 6-19. AN/UYK-7 SOFTWARE DEFICIENCIES

(Assuming only DOD-1 And One Currently Utilized Language Required Plus Associated Tools)

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisor	800K
	DOD-1 CMS-2	
4.	Test Instrumenter & Analyzer	560K
	DOD-1 CMS-2	
5.	Compilers	600K
	DOD-1	
6.	Simple Overlay Linker	210K
7.	Extended Overlay Linker	500K
8.	Interactive Debugging Aids	500K
	DOD-1	
9.	Non-Interactive Debugging Aids	80K
	Assembly DOD-1	
10.	Automatic Software Production Test	1000K
11.	Language Dependent Monitor	150K
	Assembly CMS-2 DOD-1	
12.	Standards Enforcer	180K
	CMS-2 DOD-1	

TABLE 6-19. AN/UYK-7 SOFTWARE DEFICIENCIES (CONTD)

13.	Reformatter CMS-2 DOD-1	220K
14.	RTOS + TSOS + VMM	2800K
15.	TSOS + MPOS + VMM	1400K
	Total	$10,570K

TABLE 6-20. AN/UYK-20 SOFTWARE DEFICIENCIES

(Assuming only DOD-1 And One Currently Utilized Language Required Plus Associated Tools)

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisor	800K
	CMS-2 DOD-1	
4.	Test Instrumenter & Analyzer	280K
	DOD-1	
5.	Compiler	600K
	DOD-1	
6.	Simple Overlay Linker	210K
7.	Extended Overlay Linker	500K
8.	Interactive Debugging Aids	1300K
	Assembly CMS-2 DOD-1	
9.	Non-Interactive Debugging Aids	130K
	Assembly CMS-2 DOD-1	
10.	Automatic Software Production and Test	1000K
11.	Language Dependent Monitors	100K
	CMS-2 DOD-1	
12.	Standards Enforcers	180K
	CMS-2 DOD-1	

TABLE 6-20. AN/UYK-20 SOFTWARE DEFICIENCIES (CONTD)

13.	Reformatters CMS-2 DOD-1	220K
14.	Data Base Management System	4200K
15.	RTOS + TSOS	3500K
16.	RTOS + TSOS + VMM	2800K
17.	TSOS + MPOS + VMM	1400K
	Total	$18,790K

TABLE 6-21. AN/UYK-19 SOFTWARE DEFICIENCIES

(Assuming only DOD-1 and One Currently Utilized Language Required Plus Associated Tools)

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisors	800K
	FORTRAN DOD-1	
4.	Test Instrumenters & Analyzers	280K
	DOD-1	
5.	Compilers	600K
	DOD-1	
6.	Extended Overlay Linker	500K
7.	Interactive Debugging Aids	1000K
	FORTRAN DOD-1	
8.	Non-Interactive Debugging Aids	130K
	FORTRAN Assembly DOD-1	
9.	Automatic Software Production and Test	1000K
10.	Language Independent Monitor	210K
11.	Language Dependent Monitor	150K
	Assembly FORTRAN DOD-1	
12.	Standards Enforcers	180K
	FORTRAN DOD-1	

TABLE 6-21. AN/UYK-19 SOFTWARE DEFICIENCIES (CONTD)

13.	Reformatter	110K
	DOD-1	
14.	RTOS + TSOS	3500K
15.	RTOS + TSOS + VMM	2800K
16.	TSOS + MPOS + VMM	1400K
	Total	$14,230K

TABLE 6-22. AN/GYQ-21 SOFTWARE DEFICIENCIES

(Assuming only DOD-1 and One Currently Utilized Language Required Plus Associated Tools)

1.	Computer System Simulator	420K
2.	Test Case Design Advisors	800K
	FORTRAN DOD-1	
3.	Test Instrumenters and Analyzers	280K
	DOD-1	
4.	Compilers	600K
	DOD-1	
5.	Interactive Debugging Aids	1000K
	FORTRAN DOD-1	
6.	Non-Interactive Debugging Aids	80K
	Assembly DOD-1	
7.	Automatic Software Production and Test	1000K
8.	Language Dependent Monitor	150K
	Assembly FORTRAN DOD-1	
9.	Language Independent Monitor	210K
10.	Standards Enforcers	180K
	FORTRAN DOD-1	
11.	Reformatters	110K
	DOD-1	

TABLE 6-22. AN/GYQ-21 SOFTWARE DEFICIENCIES (CONTD)

12.	RTOS + TSOS + VMM	2800K
13.	TSOS + MPOS + VMM	1400K
	Total	$9,030K

TABLE E-23. AN/GYK-12 SOFTWARE DEFICIENCIES

(Assuming only DOD-1 and One Currently Utilized Language Required plus Associated Tools)

1.	Computer System Simulator	420K
2.	Data Base Design Aid	1150K
3.	Test Case Design Advisor	900K
	TACPOL DOD-1	
4.	Test Instrumenter & Analyzer	280K
	DOD-1	
5.	Compilers	600K
	DOD-1	
6.	Simple Overlay Linker	210K
7.	Extended Overlay Linker	500K
8.	Interactive Debugging Aids	1300K
	TACPOL Assembly DOD-1	
9.	Non-Interactive Debugging Aids	80K
	Assembly DOD-1	
10.	Automatic Software Production and Test	1000K
11.	Language Dependent Monitor	150K
	Assembly TACPOL DOD-1	
12.	Language Independent Monitor	210K

TABLE 6-23. AN/GYK-12 SOFTWARE DEFICIENCIES (CONTD)

13.	Reformatters TACPOL DOD-1	220K
14.	Data Base Management System	4200K
15.	RTOS + TSOS	3500K
16.	RTOS + TSOS + VMM	2800K
17.	TSOS + MPOS + VMM	1400K
18.	Instruction Simulator	350K
19.	Standards Enforcer DOD-1 TACPOL	180K
20.	Integrated Library	100K
	Total	$19,450K

TABLE 6-24. SOFTWARE BASE AND DEFICIENCY COMPARISON

(Assuming only DOD-1 and One Currently Utilized Language Required Plus Associated Tools)

	BASE	DEFICIENCY
AN/UYK-7	$15,625K	$10,570K
AN/UYK-20	$ 7,405K	$18,790K
AN/UYK-19	$ 9,665K	$14,230K
AN/GYO-21	$14,865	$ 9,030K
AN/GYK-12	$ 4,445K	$19,450K

TABLE 6-25. SOFTWARE BASE AND DEFICIENCY COMPARISON

(Assuming Equal Compiler Costs for all Existing Languages)

	BASE	DEFICIENCY
AN/UYK-7	$13,325K	$10,570K
AN/UYK-20	$ 5,105K	$18,790K
AN/UYK-19	$ 9,665K	$14,230K
AN/GYO-21	$14,865K	$ 9,030K
AN/GYK-12	$ 4,445K	$19,450K

BIBLIOGRAPHY

1. Stone, H, "Final Report Life Cycle Cost Analysis of Instruction-Set Architecture Standardization for Military Computer-Based Systems", CORADCOM Preliminary Report, January 1978.

2. Burr, W., Coleman, A., and Smith, W., "Summary Report of the Army/Navy Computer Family Architecture Selection Committee", ECOM Technical Memorandum 4545, September 1977.

3. Wagner, J., Lieblein, E., Rodriguez, A., and Stone, H., "Procedure for and Results of the Evaluation of the Software Bases of the Candidate Architectures for the Military Computer Family", ECOM Technical Report 4549, September 1977.

4. Brooks, F., "The Mythical Man-Month", 1975

APPENDIX A

INDEPENDENT AUDIT OF THE AVAILABILITY OF SUPPORT SOFTWARE FOR THE AN/GYQ-21 ARCHITECTURE AND CURRENT MILITARY ARCHITECTURE

BY: DR HAROLD S. STONE, University of Massachusetts
Department of Electrical Engineering
Amherst, MA 01003

I. INTRODUCTION

This report contains a description of the existing software base for the military computers AN/UYK-7, AN/UYK-20, AN/GYK-12, and AN/UYK-19. For completeness, this report includes a compendium of the software base for the AN/UYK-21 (the military designation for the PDP-11 architecture) that was prepared in the summer of 1976 (Wagner et. al, 1976). The software study treated the set of software tools that were selected by a committee of Army/Navy representatives during the selection of a candidate architecture for the standard military computer family. A list of the tools and the availability of each for the candidate architectures appears in this report. A full description of each tool appears in Chapter 6, Section 6.2.2 of (Burr, Coleman, and Smith, 1976).

The method used to assess the software base for each computer was similar to that used in the previous study. James Wagner and Harold Stone visited the manufacturer of each computer system and principal users and queried the individuals about the available software. Where questions arose, the questions were answered by examining reference documentation. A tool was deemed to be available for a specific architecture if

(a) The tool was a genuine released item supported by user documentation,

(b) The tool had to be available for use as of 1 January 1977, and not merely nearing release, and

(c) the tool had to satisfy the characteristics for that tool as described in (Burr, Coleman, and Smith, 1976, Section 6.2.2).

In evaluating the software base, we have enlarged the study to include software systems that are nonself-hosted. This gives rise to some interpretation as to what tools should be counted in the software base of a particular architecture and what tools should be treated as missing. In several instances, the existence of a software tool for one system makes that tool available to all architectures since the tool itself need not be self-hosted. For example, one tool in the base is a General Purpose Systems Simulator of which GPSS for the IBM System/370 is but one example. Since this tool is unlikely to be deployed in tactical systems but is more likely to be used at

software development centers it does not make sense to insist that it run on a particular computer family provided that some computer that supports the tool is available at each development center. In the case of the simulator, there is a GPSS system or equivalent for every major commercial computer system, and we deem it extremely likely that at least one commercial computer will be accessible by support software staff. Consequently, the availability of GPSS counts as a simulator in the software base of each candidate architecture.

We selected GPSS for this example since its function is completely independent of the architecture of the computers involved in this study. Other tools that share this characteristic are so indicated. Examples of software tools that do not have this characteristic are compilers, assemblers, and operating systems since each of these are dependent in some essential way on the target computer architecture.

The inclusion of nonself-hosted software in this study has been done by considering each tool in two forms--self-hosted and nonself-hosted. The listings are compiled separately. Some tools simply cannot be used except in self-hosted form, so these do not appear in the tabulation of nonself-hosted software. Among these tools are operating systems and language independent monitors since both of these depend crucially on the execution of the tool on the host architecture in real time. Other tools omitted from the nonself-hosted list have been omitted for similar reasons.

In examing the software base of the military computers the following sources were used:

AN/UYK-7: Univac, Litton Data Systems, FCDSSA, NAVSEA

AN/UYK-20: Univac, NAVECS

AN/GYK-12: Litton Data Systems

AN/UYK-19: ROLM Corp. and Data General (for NOVA software)

AN/UYK-21: Digital Equipment Corp. (for PDP-11)

The software base for the AN/AYK-14 is essentially that of the AN/UYK-20 since the AYK-14 is upward compatible from the UYK-20, and the group in charge of the UYK-20 software is currently making the few modifications necessary to move the entire base to the AYK-14. Although the AYK-14 was not included in this study, very little error is introduced by treating its base to be equal to the UYK-20 base.

This report is organized into three additional sections. Section II treats the self-hosted software, and Section III treats the nonself-hosted software. Section IV contains a summary of the principle data. Although this report describes only the availability of each tool, a companion report to be issued shortly by James Wagner places a dollar value on each tool and gives an estimate of the total value of the base for each computer architecture. These data will be used in the life cycle cost analysis for the MCF computer project.

II. SELF-HOSTED SOFTWARE

Table 1 gives the pertinent results of the software base study for self-hosted software. Details of the table are explained in this section. We list only the tools that exist for at least one architecture.

Tool 1.1.2 Data Base Design Aid. This tool is used for the construction of data base systems. For the UYK-21 the appropriate tool is the SCHEMA Compiler which is part of the IDMS-11 data base management system for the PDP-11, and is available from Digital Equipment Corporation.

Tool 1.3.4 Test Instruments and Analyzers. This tool allows a programmer to instrument his source program so as to obtain counts of the number of times designated portions of his program are executed. He can also discover which portions have not been executed at all. Since the tool is specific to particular source languages, and since it may itself be written in a machine independent source language, it may be available for architectures if it is available for one. Indeed, this is the case for a FORTRAN analyzer. Either RXVP-1 or TAP may be used; both are commercially available. They are written in FORTRAN and can be moved to any computer system with a FORTRAN compiler. Thus they count in the base of all architectures except the GYK/12 since it does not have a FORTRAN compiler in its base. The GYK-12 has a TACPOL compiler with the test instrumentation built into the language, so that it gets credit for this tool in the TACPOL base.

Tool 1.4.1.1 Basic assembler. All architectures have assemblers available. In several cases there are more than one assembler. Since macro assemblers satisfy the basic assembler need, and all architectures have macro assemblers, we list the source of the tool under the macro assembler heading in the next paragraph.

Tool 1.4.1.2 Macro Assembler. The macro assemblers available for the several computers are the following:

UYK-7	MACRO/32 (from Univac)
UYK-20	Level 2 Assembler and MACRO/20 (from Univac and NAVECS)
GYK-12	L3050 Macro Assembler (from Litton)
UYK-19	Macro Assembler (from ROLM)
UYK-21	MACRO/11 (from DEC)

Tool 1.4.2.1 FORTRAN Compiler. All architectures except the GYK-12 have a FORTRAN compiler available. Some have more than one. Univac and NAVECS are sources for the UYK-7 and UYK-20 compilers while ROLM and DEC supply compilers for the UYK-19 and UYK-21, respectively.

Tool 1.4.2.2 COBOL compiler. The two architectures that are compatible with commercial computers have COBOL compilers while the purely military architectures do not. The COBOL for the UYK-19 is called IPI COBOL available from the Flordia-based company IPI. DEC supplies the COBOL for the UYK-21.

Tool 1.4.2.3 CMS-2 Compiler. For the purposes of this study we do not distinguish among the several different levels of CMS-2. For the architectures that are considered here, if a CMS-2 compiler exists, then the study found it exists for at least two levels of the language definition. Univac and NAVECS can source the CMS-2 compilers for the UYK-7 and UYK-20 architectures. Command Control Communications Corporation of San Pedro, California supplies the CMS-2 for the UYK-19.

Tool 1.4.2.7 JOVIAL compiler. Univac has written a JOVIAL for the UYK-7 and can supply this as a product.

Tool 1.4.2.8 TACPOL compiler. Litton's TACPOL compiler for the GYK-12 is called TACPOL-B and is available from litton.

Tool 1.4.3.1 Basic Linker. Architectures that had simple overlay linkers were credited with having basic linkers as well. Linking capability exists for all architectures at least at the basic level and in several instances at the simple overlay level. The linkers credited to each architecture are as follows:

UYK-7	Linker (from Univac)
UYK-20	Linking loader (from Univac and NAVECS)
GYK-12	Embedded in PSS and TOS operating systems (from Litton)
UYK-19	RLDR (from Data General and ROLM)
UYK-21	LINK-11 (from DEC)

Tool 1.4.3.2 Simple Overlay Linker. This tool performs basic linking of modules, plus has the capability to construct overlay trees so that overlay calls from module to module can be implemented. The programs and their respective sources are given as follows:

UYK-19	RLDR (from Data General and ROLM)
UYK-21	RSX TASK BUILDER (from DEC)

Tool 1.4.3.3 Extended overlay linker. This tool has all of the capabilities of a simple overlay linker plus the ability to manage memory dynamically so as to take advantage of available memory. There is some concern as to the applicability of this tool in a system with virtual memory since the virtual memory achieves the same function in a different way. Only the UYK-21

has such a linker, namely in the RSX TASK BUILDER available from DEC. The tool known as DMR for Dynamic Memory Restructurer available for the UYK-7 has some of the capability of an extended overlay linker in that it can restructure memory assignments dynamically in a system in which one or memory modules have failed. However, its user interface does not appear to be adequate for the intended purposes of the extended overlay linker since the main function of DMR is reconfiguration for reliability purposes.

Tool 1.4.4.1 Interactive Debugger for Assembler. This tool is deemed to be similar to debugging systems similar to ODT and DDT for the PDP-10 and PDP-11 computer systems. The debugging aids given in the table are OED for the UYK-7 (from Litton), DEBUG for the UYK-19 (from ROLM) and DDT for UYK-21 (from Carnegie-Mellon University and DEC).

Tool 1.4.4.1.3 Symbolic debugger for COBOL. This tool is the COBOL equivalent of the previous tool in that one has to be able to debug interactively at the source language level with the aid of this tool. This presupposes that COBOL exists on the architecture. The debugging aids are built-in features of the two architectures that have COBOL compilers. For the UYK-19, the program is available from Data General and for the UYK-21 it is available from DEC.

Tool 1.4.4.3.2 Noninteractive Symbolic Debugger for FORTRAN. Of the architectures for which FORTRAN is available only the UYK-21 architecture has a FORTRAN with built-in symbolic debugging aids for batch (noninteractive) debugging of FORTRANS. The source of the software is DEC and it is contained in PDP-11 FORTRAN.

Tool 1.4.4.3.3 Noninteractive Symbolic Debugger for COBOL. This is the same situation as for the FORTRAN batch debugging aid. Only the UYK-21 architecture has such software available, and it is part of the COBOL package for the PDP-11 available from DEC.

Tool 1.4.4.3.4 Noninteractive Symbolic Debugger for CMS-2. The CMS-2 compiler available from Univac for the UYK-7 has debugging facilities for noninteractive debugging at a symbolic level.

Tool 1.4.4.3.6 Noninteractive Symbolic Debugger for TACPOL. The TACPOL language as implemented by Litton for the GYK-12 contains debugging directives that satisfy this requirement.

Tool 1.4.4.2 Integrated Library. This tool is a comprehensive library change control system. It exists for the UYK-21 architecture in the form of the Modification Request Control System and Source Code Control System that is part of the Programmer's Work Bench System available from DEC for the PDP-11. For the GYK-12, Litton has developed a tool known as SPS Librarian. Other architectures apparently do not have software that meets the specifications for this tool.

Tool 1.4.6.1.1 Language Dependent Monitor for Assembler. There exists a tool known as STV for System Test Vehicle available from Univac and NAVECS that appears to meet the tool specifications. This runs on the UYK-20.

Tool 1.4.6.1.2 Language Independent Monitor. STV for the UYK-20 satisfies this tool requirement and is available from Univac and NAVECS. A monitor for the UYK-7 is available from Litton as part of ITAWDS software system written to support the LHA project.

Tool 1.4.8.2 Reformatters. All architectures that have FORTRAN compilers are deemed to have this tool since it is available as a FORTRAN source to operate on FORTRAN as a preprocessing step. The tool is called IFTRAN-2 FORTRAN Preprocessor for Structured Programming. Since the GYK-12 does not have a FORTRAN compiler, it is missing this tool. However, if a FORTRAN compiler becomes available for the GYK-12, then this tool becomes available automatically. Thus the absence of this tool does not penalize the GYK-12's software base.

Tool 2.1 Data Base Management System. The tool known as ITAWDS mentioned above for the UYK-7 for language independent monitors is a comprehensive software system that contains within it a database management system. It is available from Litton. The two architectures that have commercial counterparts also have data base management systems. For the UYK-19, the system is known as MIDAS, and is available from Boeing Computer Services. For the UYK-21, the tool is IDMS-11, available from DEC for the PDP-11.

Tool 2.3.1 Text Processing System. The text-processing system function is to prepare final copy of publishable materials, and it has many built-in features that assist this process. The text processor listed for the UYK-7 is available from FCDSSA. The tool listed for the UYK-21 is TYPESET-11 and is available from DEC for the PDP-11.

Tool 2.4.2.1 Interactive Source Language Editor. The editor for the UYK-19 architecture is known as SPEED, and is available from ROLM. The UYK-7 editor is a subsystem of SHARE 7 and is available from Univac. Any one of a number of editors for the UYK-21 architecture meet this criterion. Among them is the editor that runs under IAS available from DEC for the PDP-11.

Tool 2.4.2.3 Batch Source Language Editor. All architectures have batch editors available according to the list below:

UYK-7	Subsystem of Share 7 (from Univac)
UYK-20	Subsystem of Level 2 Librarian system (from Univac)
UYK-19	BEDIT (from ROLM)
GYK-12	Subsystem of SPS Librarian (from Litton)
UYK-19	SLIPER (from DEC)

Tool 3.9 Real-Time Operating System plus Time-sharing. The UYK-7 version of this tool is the ITAWDS system available from Litton. The UYK-21 software system is called IAS and is available from DEC.

Tool 3.13 Time-sharing plus virtual machine monitor. Of the five architectures only the UYK-21 was given credit for being virtualizable in the sense of being able to support a virtual machine monitor as of January 1, 1976. The architectures have not evolved since that time to the point where any others have been shown to be virtualizable. However, not even the UYK-21 was given credit for this tool since the only instance of a virtual machine monitor for the UYK-21 exists in a research environment, and cannot said to be a releasable piece of software. For similar reasons, no architecture was credited with Tool 3.14, which is a superset of this tool.

III NONSELF-HOSTED SOFTWARE

Software that can be hosted on architectures other than the candidate architectures must necessarily not depend on realtime responses of the candidate architectures. This eliminates operating systems, performance monitors, and data base management systems from the list of tools that can be self-hosted. Similar reasoning reduces the list of possible nonself-hosted tools to the list given in Table II. Since some of the tools fall in the category of tools that are available for all architectures if available for one, they are credited to all architectures. The text in this section explains all such cases.

Tool 1.1.1 General Purpose System Simulator. This program exists on the IBM 370 and Univac 1100 series computers among others. It is probably not necessary to mount a special development to create a running version on a military computer system.

Tool 1.4 Cross assembler. All architectures have at least one cross assembler that executes on a foreign architecture producing object code for the native architecture. In some cases the cross assembler is written in a machine independent language such as FORTRAN so that it can run on any one of several different computers. The list of cross assemblers available is as follows:

UYK-7	Available for Univac 1100 series computers from Univac.
UYK-20	Runs on six different computers since it is written in compatible FORTRAN. (from NAVECS)
UYK-19	Available for IBM 370, Univac 1100, and CDC 6000 series computers (from Data General)
UYK-21	Available for IBM 370 and CDC 6000 computers from Computer Associates and First Data
GYK-12	Available for IBM 370

All assemblers that run on foreign hardware are macro assemblers.

Tool 1.4.5.1 FORTRAN Compiler. The UYK-20 architecture has a FORTRAN compiler for it written in FORTRAN that has successfully executed on CDC 6000, Univac 1100, and IBM 370 series computers. The AN/UYK-21 has a FORTRAN corss compiler for it that runs on the GE 6000 computers, and is available from GE.

Tool 1.4.5.2 COBOL Compiler. A COBOL compiler that runs on the IBM 370 and generates code for the UYK-19 has been written and is in operation by the Navy.

Tool 1.4.5.3 CMS-2 Compiler. The CMS-2 Compiler for the UYK-19 mentioned above is written in FORTRAN and runs on several different computers. Similarly the CMS-2 compiler for the UYK-20 is written in FORTRAN and runs on several different computers. There is a UYK-7 CMS-2 compiler that runs on Univac 1100 computer systems, and the nonself-hosting methodology is the normal mode of operation with CMS-2 program development.

Tool 1.4.5.4 JOVIAL compiler. JOVIAL is written in itself and can run on several different computers while producing object code for a specific computer. Since there is self-hosted JOVIAL compiler for the UYK-7, this compiler can be and has been run successfully on other computers that have JOVIAL.

Tool 1.4.5.5 TACPOL. TACPOL-A is the TACPOL compiler for GYK-12 that executes on the IBM 370 series computers.

Tool 1.4.6 Instruction simulators. All but the UYK-21 have reported nonself-hosted instruction simulators. We suspect that there exist such simulators for the UYK-21 as well, and are currently investigating this possibility. The ISP compiler at Carnegie-Mellon University does simulate the PDP-11 correctly at instruction level, but it is not a piece of software with extensive outside release and use, so it is not counted here. A summary of the instruction-level simulators appears below:

UYK-7 Runs on the Univac 1100 series computers (from Univac)

UYK-20 FORTRAN based program for UYK-20 runs everywhere (from Univac)

GYK-12 Available for IBM 370 (from Litton)

UYK-21 Available for IBM 370 (from First Data and Computer Associates)

UYK-19 Available for IBM 370 (from ROLM)

IV FINAL SUMMARY

A word of caution concerning the accuracy of the data and its validity for use in cost models is worth while stating here. There may be some tools that missed our inventory for one reason or another. We sought several sources where possible for an architecture and we achieved good agreement among the sources. It is rather unlikely that anything significant was missed, and highly probably that all data given are correct as of this date.

The tables contain a single bit of information about each item, that is, whether it exists in releaseable form or not. But two different software systems that ostensibly perform the same function may have vastly different characteristics, utility to the user, and procurement costs. Given the limited time and resources at our disposal, the data presented are all that can be reasonably be done within the constraints. When these data are put to use in cost models it is well to take into account the potential differences in values and costs of supposedly identical software, if this can be done in a reasonable manner.

TABLE I. SELF-HOSTED TOOL AVAILABILITY

		UYK 7	UYK 20	UYK 19	UYK 21	GYK 12
1.1.1	General Purpose System Simulator	N	N	N	N	N
1.2.1	Computer System Simulator	N	N	N	N	N
1.2.2	Data Base Design Aid	N	N	N	Y	N
1.3.1	Test Data Generator	N	N	N	N	N
1.3.2	Test Data Auditor	N	N	N	N	N
1.3.3	Test Case Design Advisors (for each of five standard languages)	N	N	N	N	N

TABLE I. SELF-HOSTED TOOL AVAILABILITY (CONTINUED)

		UYK 7	UYK 20	UYK 19	UYK 21	GYK 12
1.3.4	Test Instruments and Analyzers:					
	FORTRAN	Y	Y	Y	Y	N
	TACPOL	N	N	N	N	Y
	(for each of three other standard languages)	N	N	N	N	N
1.4.1	Assemblers:					
	Basic	Y	Y	Y	Y	Y
	Macro	Y	Y	Y	Y	Y
1.4.2	Compilers:					
	FORTRAN	Y	Y	Y	Y	N
	COBOL	N	N	Y	Y	N
	CMS-2	Y	Y	Y	N	N
	JOVIAL	Y	N	N	N	N
	TACPOL	N	N	N	N	Y
1.4.3	Linkers:					
	Basic	Y	Y	Y	Y	Y
	Simple Overlay	N	N	Y	Y	N
	Extended Overlay	N	N	N	Y	N

TABLE I. SELF-HOSTED TOOL AVAILABILITY (CONTINUED)

		UYK-7	UYK-20	UYK-19	UYK-21	GYK-12
1.4.4.1	Interactive Debugging Aids:					
	Assembler	Y	N	Y	Y	N
	COBOL	N	N	Y	Y	N
	CMS-2	Y	N	N	N	N
	(for three other standard languages)	N	N	N	N	N
1.4.4.3	Noninteractive Debugging Aids:					
	Assembler	N	N	N	N	N
	FORTRAN	N	N	N	Y	N
	COBOL	N	N	N	Y	N
	CMS-2	Y	N	N	N	N
	JOVIAL	N	N	N	N	N
	TACPOL	N	N	N	N	Y
1.4.5.2	Integrated Library	N	N	N	Y	Y
1.4.5.3	Automatic Software Production Test	N	N	N	N	N
1.4.6.1	Language Dependent Monitors:					
	Assembler	N	Y	N	N	N
	(for five standard languages)	N	N	N	N	N
1.4.6.2	Language Independent Monitor	Y	Y	N	N	N
1.4.7	Standards Enforcers:					
	(for five standard languages)	N	N	N	N	N

TABLE I. SELF-HOSTED TOOL AVAILABILITY (CONTINUED)

		UYK 7	UYK 20	UYK 19	UYK 21	GYK 12
1.4.8	Preprocessors, reformattors:					
	FORTRAN	Y	Y	Y	Y	N
	(for four other standard languages)	N	N	N	N	N
2.1	Data Base Management System	Y	N	Y	Y	N
2.3.1	Text Processing System	Y	N	N	Y	N
2.4.2	Editors:					
	Interactive	Y	N	Y	Y	N
	Batch	Y	Y	Y	Y	Y
3	Operating systems:					
	RTOS + TSOS	Y	N	N	Y	Y
	RTOS + TSOS + VMM	N	N	N	N	N
	TSOS + MPOS + VMM	N	N	N	N	N

TABLE II. NONSELF-HOSTED SOFTWARE TOOLS

		UYK-7	UYK-20	UYK-19	UYK-21	GYK-12
1.4.4	Cross Assemblers	Y	Y	Y	Y	Y
1.4.5	Cross Compilers:					
	FORTRAN	N	Y	N	Y	N
	COBOL	N	N	Y	N	N
	CMS-2	Y	Y	Y	N	N
	JOVIAL	Y	N	N	N	N
	TACPOL	N	N	N	N	Y
1.4.6	Instruction Simulator	Y	Y	Y	Y	N

REFERENCES

Burr, W., A. Coleman, and W. Smith, 1976. "Summary of the Final Report of the Army/Navy Computer Family Architecture Selection Committee," 1 December 1976.

Wagner, James, E. Lieblein, Jorge Rodriguez, and H. S. Stone, 1976. "Evaluation of the Software bases of the candidate architectures for the military computer family," interim report issued 1 August, 1976, contained in Final Report of the Army/Navy Computer Family Architecture Selection Committee as Chapter VI, Section II.