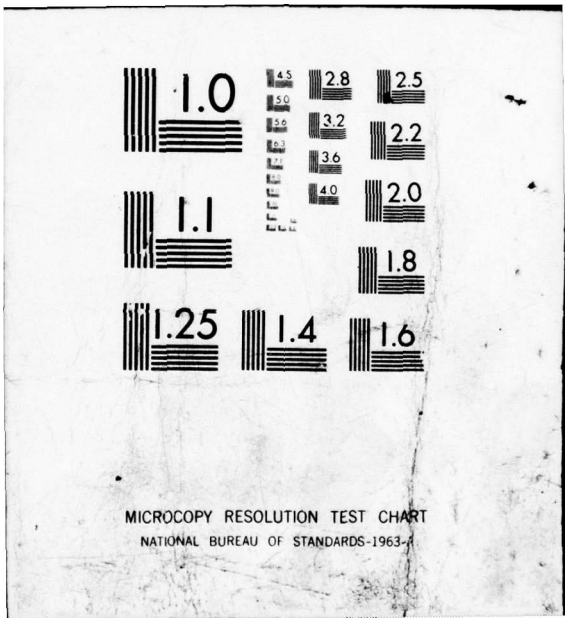
AD-A049 457 ARMY ELECTRONICS COMMAND FORT MONMOUTH N J F/G 9/2
COMPUTER FAMILY ARCHITECTURE SELECTION COMMITTEE. VOLUME II. SE--ETC(U)
SEP 77 W E BURR, S H FULLER, H STONE

UNCLASSIFIED ECOM-4527 NL

| OF |
AD
A049457

END
DATE
FILMED
3-78
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
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Research and Development Technical Report

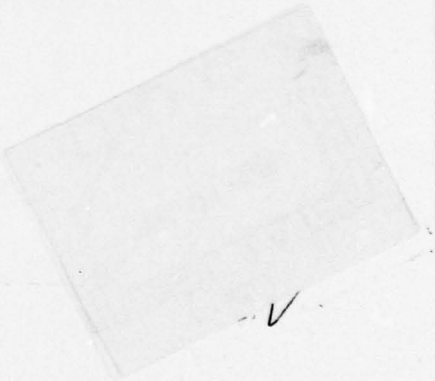
ECOM-4527

COMPUTER FAMILY ARCHITECTURE SELECTION COMMITTEE - FINAL REPORT, VOLUME II - SELECTION OF CANDIDATE ARCHITECTURES AND INITIAL SCREENING

William E. Burr
Center for Tactical Computer Sciences
Communications/Automatic Data Processing Laboratory

Samuel H. Fuller
Carnegie-Mellon University

Harold Stone
University of Massachusetts



September 1977

ECOM
US ARMY ELECTRONICS COMMAND FORT MONMOUTH, NEW JERSEY 07703

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

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: ECOM-4527 (14)	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Computer Family Architecture Selection Committee (6) Final Report Volume II. Selection of Candidate Architectures and Initial Screening.	5. TYPE OF REPORT & PERIOD COVERED: (9) Fiscal Rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) William E. Burr, Samuel H. Fuller (NRL, CMU), Harold Stone (UMASS)	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center For Tactical Computer Sciences (CENTACS), DRSEL-NL-BC, Fort Monmouth, N.J. 07703	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 1L7 62701 AH92 00109 (16)
11. CONTROLLING OFFICE NAME AND ADDRESS: (12) 54p.	12. REPORT DATE: September 1977 (11) / 13. NUMBER OF PAGES: 50 Pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): CENTACS, DRSEL-NL-BC, Fort Monmouth, N.J. 07703 (17) B1	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 consists of a summary and nine (9) volumes. It is the result of joint Army/Navy work. The complete report may be separately published by the Naval Research Laboratories.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Computer Family Architecture, data types, absolute criteria, quantitative criteria, memory structure.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The selection of initial candidate computer architectures to be evaluated by the CFA Committee was begun with letters from NRL and ECOM to 21 Navy and 15 Army commands, laboratories, system centers, and project managers, soliciting proposals for candidates architectures. These inquiries were followed by discussions at the first two CFA meetings and resulted in an initial set of nine computer architectures: →

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

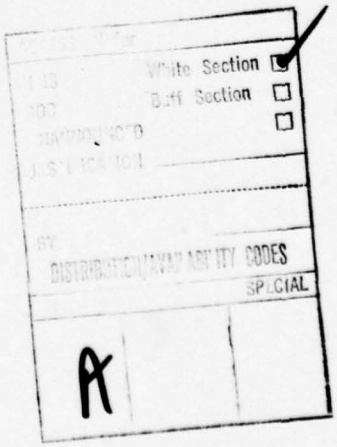
20. 1. AN/GYK-12,
3. AN/UYK-20,
5. Data General NOVA (ROLM-1664),
7. DEC PDP-11,
9. Interdata 8/32,
2. AN/UYK-7,
4. Burroughs 6700,
6. IBM System/370,
8. SEL 32,

In order to begin the evaluation process and reduce the number of candidate architectures for more intensive examination and benchmarking, a set of absolute and quantitative criteria were developed. The table below summarizes the performance of the nine initial architectures on these absolute and quantitative criteria.

ARCHITECTURE	QUANTITATIVE CRITERIA SCORE	ABSOLUTE CRITERIA
INTERDATA 8/32	1.68 (best)	problem w interrupts & traps
PDP-11	1.43	passed all
IBM S/370	1.36	passed all
AN/GYK-12	0.94	failed floating-point
ROLM/NOVA	0.92	failed virtual memory mapping
B6700	0.91	failed protection
SEL-32	0.86	failed virtual memory memory
UYK-7	0.46	failed floating point
UYK-20	0.44 (worst)	failed protection

The result of this initial ranking and screening process was to narrow the list of candidates to three machines, the Interdata 8/32, the PDP-11, and the IBM S/370, which were then more intensively investigated using test program, cost model, support software, and licensing evaluations discussed in later volumes of this report.

TABLE OF CONTENTS



1. INTRODUCTION

This volume describes how the candidate computer architectures were initially chosen and the methodology and data used in ranking these architectures according to absolute and quantitative criteria during the preliminary screening of the architectures. These measures were applied to a field of nine candidates with the objective of doing a coarse screening to determine three or four "best" candidates that would be subjected to a much more detailed analysis. At the point this procedure was formulated, it was known that time and money limitations would preclude doing a detailed analysis on all nine candidates; consequently an initial screening to limit the field was necessary.

This report covers the measures formulated for that screening, the methodology used to evaluate the architectures according to those measures, the raw data and rankings obtained from the measures, and a post screening analysis of the sensitivity of the results to the way in which the measures were combined.

In drawing up measures for a coarse screening, there were potential problems in selecting measures that discriminated for or against particular candidates, so that the results of the screening might be unreasonably biased toward or away from some candidate architectures. A good deal of judgment and collective wisdom went into the measures so as to prevent such biases from marring the selection process. Each criterion can be argued to be an important one on its own merits, and should stand up against careful inspection by knowledgeable parties. Some criteria were selected as absolute criteria in the sense that a selected candidate must satisfy each of these. In elevating a criterion to a rank of "absolute," the criterion had to be present practice in the industry as a whole, and had to contribute in some identifiable way to aiding the mission of the computer family architecture. Here, for example, such criteria as protection and virtual-to-real address translation were included. Architectures that satisfy these criteria are so commonplace today, and the characteristics of such criteria are so useful, that it is reasonable to treat these as absolute.

One criterion that might have been absolute was made a quantitative criterion to avoid biasing the selection procedure. This criterion, described later as virtualizability, has been a standard feature of the IBM 370 architecture, but of no other architecture among those considered. In dealing with this as a criterion there was a question about whether or not any of the other architectures satisfied this property, and there was a real danger of rejecting most of them on this criterion if it were given absolute status. Although virtualizability is important, it is not considered sufficiently important at this time in the development of computer technology to use as a key factor in selecting one architecture from among several, so it was not given absolute status. As computer technology develops, it may be that virtualizable machines become commonplace with concommitant development of important systems that rely on this concept. If this happens, then a CFA selection meeting held at that time might well treat virtualizability as an absolute measure.

The measures described here were developed by a working subcommittee and presented to the full Selection Committee at a meeting on 3-4 December 1975. The basic methodology involved the collection of data for each of architectures for each of the measures, which could then be normalized into some manageable form. Independently the Selection Committee organizations submitted weights for each quantitative measure that gave their views on the importance of each of these measures in meeting the objectives of missions of those organizations. The outcome of the meeting on 3-4 December was a revised set of criteria and procedures which in general adhered to the proposed methodology but had some refinement, so as to be acceptable to the voting majority (two-thirds majority) of the Committee members.

In the period that followed this meeting, nine subcommittees (one for each candidate architecture) gathered the requisite data for the quantitative and absolute measures. These data were audited for accuracy and consistency, and presented to the Selection Committee at a meeting held 17-20 February 1976. Questions in interpreting criteria that arose during the audit were presented to the full Committee for final determination. At this time, the results showed that only three architectures passed the absolute criteria screening. These were the B-6700, the PDP-11, and the IBM 370. The quantitative criteria showed that the Interdata 8/32, the PDP-11, and the IBM 370 led the other architectures by comfortable margins. Interdata's 8/32 had failed one absolute criterion on a technicality, and in fact, closer study over the next few months showed that it may satisfy this criterion and probably should have passed the absolute criteria. This question was never totally resolved by the Selection Committee. On the other hand, the B-6700 was later shown to have failed one of the absolute criteria, so it was subsequently eliminated from consideration. The screening procedures thus reduced the field to three candidates -- the Interdata 8/32, the DEC PDP-11, and the IBM S370 -- for further detailed study.

This volume records the criteria, ranking methodology, and findings of the Selection Committee. In several cases, measures for architectures were modified and refined over a period of time. In no case other than those pointed out above for the B-6700 did the refined data alter the final selections. The final section of this volume reviews the chronology of the development and collection of the measures.

2. INITIAL SELECTION OF CANDIDATE COMPUTER ARCHITECTURES

In March and April of 1975 letters were sent from NRL and ECOM to 21 Navy and 15 Army commands, laboratories, system centers, and project managers soliciting proposals for "candidate" instruction-set architectures for the CFA. Candidate architectures initially nominated were: IBM System/370, Data General NOVA/ROLM 1664, Burroughs 6700, and DEC PDP-11.

This list of four architectures was felt to be insufficient, so a meeting of NRL, NUSC, and ECOM personnel was held in July of 1975 to extend this list. As a result, the following were added to the initial list: DECsystem 10, Interdata 8/32, and Xerox Sigma Series.

At the first CFA comittee meeting on 1 and 2 October 1975, the list of candidates was reviewed, and two new architectures, the AN/UYK-7 and the AN/UYK-20, were added to the list at the request of Mr. Henry Hill of NAVSEA. The DECsystem 10 and the Sigma/Xerox were dropped from the list because of a lack of support on the Committee as to the potential of these two candidates. Subcommittees were formed to represent each of the candidate architectures.

At the second CFA committee meeting on 3 and 4 December 1975 the SEL-32 was also added to the list of candidates at the request of NSWC, Dahlgren. Before the third CFA meeting, at the request of PM, ARTADs, the AN/GYK-12 was included in this evaluation. At the third CFA meeting each architecture was examined in detail with respect to the absolute and quantitative criteria.

No additional candidate architectures were added after the AN/GYK-12. The final set of nine computer architectures that underwent the initial screening process described in the remainder of this report were:

AN/GYK-12
AN/UYK-7
AN/UYK-20
Burroughs 6700
Data General NOVA (and ROLM 1664)
DEC PDP-11
IBM System/370
Interdata 8/32
SEL 32

Although the DECsystem 10 was dropped from formal consideration at the first CFA meeting, a member of the CFA committee collected enough information so that it could be informally evaluated.

3. DEFINITION OF IMPORTANT TERMS AND RELEVANT BACKGROUND

This section provides descriptive information on terms used in succeeding sections.

a. Definition of Computer Architecture

The phrase "computer architecture" has been used to encompass a variety of concepts, but in the context of the CFA Committee it has the following specific meaning:

> Computer Architecture: The structure of the computer a programmer needs to know in order to write any machine-language program that will run correctly on the computer.

With a well specified architecture, details of data bus width, technology (core memory versus semiconductor memory, TTL versus ECL circuits), implementation speedup techniques such as cache memories and instruction lookahead buffers, physical size of computer, etc. need not be of concern to the programmer. A clear (and clean) distinction between the architecture and implementation details allows software to be transported between computers with the same architecture even though they may have very different implementations. This separation of architecture and implementation is not a radically new idea [Amdahl, et al 1964]. The IBM System/360-370 series of machines, the DEC PDP-11 series, the Data General NOVA series are just three examples of where this has already been successfully accomplished to a greater or lesser degree. The IBM System/360-370 is probably the best example. Aside from the 360 Model 20, the other couple dozen models have an excellent record for compatibility between models.[1] In fact, a good way to get a feel for what a complete computer architecture looks like, unencumbered by implementation details and "features," is to look at IBM's System/370 Principles of Operation [1973]. The IBM manual is the definitive statement of the 370 architecture. It does not come in separate versions for each of the 370 models -- it adequately describes the architecture for them all. It is significant to note that computer architectures are not as vulnerable to advances in technology as are specific implementations of the architecture. The IBM 360 architecture was announced in 1964 and now 12 years later it is still IBM's main architecture. The 360 architecture (as extended in the 370 architecture) may well last another decade before IBM finds it necessary to move to a new architecture.

b. Overview of the Components of a Computer Architecture

Data Types

A central aspect of any computer architecture is the data types that it supports. In this section we list the data types that need to be well supported in the selected architecture.

---

[1] Other computer manufacturers have also been able to build compatible implementations of the IBM S/370. The Amdahl 470/V6 is probably the best known example. National Semiconductor and Cambridge Memories have also announced plans to build and market compatible implementations of the S/370.

Bytes/Characters (8 bits). Applications: Character processing; input from transducers, e.g., 8-bit output from A to D converters. An architecture often specifically supports a particular character format, e.g., ASCII or EBCDIC, and hence the character format must be considered part of the architecture.

Integers

Small Integers (16 bits). Applications: input from high resolution transducers (rarely more than 12 bits needed here); arithmetic processing of physical signals; address calculations (addresses, array indexes, etc.); program control (loop index, conditional branching, etc.)

Large Integers (32 bits). Applications: extended precision arithmetic processing of signals and general purpose computation; random number generation; address calculation; program control.

For integers, there is the question of what representation to use: sign-magnitude, one's complement, or two's complement. (For reasons of implementation efficiency and its unique representation of zero, two's complement has become the industry standard.)

Floating Point Numbers (32 and 64 bits). Application: Floating point can be of significant assistance in arithmetic processing: they relieve the programmer of having to be concerned with scaling fixed point (i.e., integer) results. Unfortunately, the specification of floating point formats and operations are not a trivial matter. Numerical analysts have given this problem considerable attention and their results and guidelines were summarized in an early report to the CFA Committee [Fuller, 1975].

Individual Bits (Booleans). Applications: Program control; description of physical state (e.g., valves, relays, switch positions).

Variable Size Fields. Applications: Extraction and insertion of odd size fields in strings, records, and control blocks. Very useful when efficient packing of primary memory is required. (An interesting example of a byte manipulation facility is provided by the DECsystem-10 architecture.)

Decimal (4 bits per digit). Variable number of digits per decimal number. Not of primary importance in miltary tactical environment; useful in support functions -- in other words, the range of applications commonly programmed in Cobol (inventory, payoff, etc.).

It is not a requirement of an architecture that it have instructions to directly manipulate each of the above data types; only that the architecture allow the efficient manipulation of the data types. The test program evaluation process was intended, among other things, to exercise the architecture with respect to these data types.

The data types enumerated above are based on powers of 2 (i.e., 8, 16, 32, and 64). This simply reflects the dominant trend in current computer architectures. Machines that do not follow this format (e.g., DECsystem 10 and B6700) should be evaluated with comparable data types. For example, let the DECsystem 10 with its 36-bit words use 7 or 9 bit bytes, 18 and 36-bit integers, and 36 and 72-bit floating point numbers.

c. Operators for Data Types

Every architecture needs a set of operators (instructions) to process its data structures. The architecture must either include an instruction for operating on each supported data type (e.g., IBM S370) or provide a data tagging scheme (e.g., B6700). Below is a list of the main types of instructions for operating on data (other instructions for program control, I/O, etc. are discussed later):

(1) Load/Store or Move

(2) Arithmetic (add, subtract, multiply, divide, negate, absolute value, etc.)

(3) Shift, rotate, and other instructions for field alignment

(4) Logical (and, or, clear, complement, etc.)

d. Open Instruction Set

An important attribute of any architecture's instruction set is that it is relatively "open." In other words, architectures should not pack their operations so densely that there is no room for expanding the instruction set. There are two separate but important reasons to leave the instruction space fairly open:

(1) Accommodate future extensions. For example, the extended floating point instructions in the IBM S370 do not share the same format as the single and double precision formats because there was not enough room left in the instruction space, similarly for the floating point in the PDP-11/40 (note there was not room for two addresses as in other PDP-11 instructions). On the other hand, the virtual memory addition to the IBM S370 is a good example of an extension that was added well after the initial architecture but which integrated well because (among other things) space was left in the instruction space.

(2) Certain implementations of the architecture may warrant implementation of a limited (and approved) set of specialized instructions for specific applications. Room needs to be left for these specialized instructions without requiring the use of contorted formats.

e. The Effective Address Calculation Process

An aspect of computer architectures that yields to a rich variety of approaches is how addresses are constructed from the fields of an instruction. The simplest approach is to just embed the full address in an instruction. However, this is usually both inflexible and inefficient in the use of the bits in the instruction. The following techniques have been developed to make the construction of addresses from the instruction a more efficient and powerful process:

(1) Indirect addressing (single level or multi-level)

(2) Indexing

(3) Base-displacement addressing

(4) Automatic incrementing (decrementing) of address pointers during address calculation

(5) Combinations of two or more of the above techniques in a single address calculation

Well designed address calculation processes make the use of important data structures such as stacks, arrays, and linked lists very efficient. Unfortunately, it is difficult to make any absolute statements about the relative merits of address calculation schemes (e.g., multi-level indirection versus single-level indirection); a lot depends on how well the address calculation process integrates with the other features of the architecture. Therefore, we must again rely on test programs to measure the value of alternative schemes.

f. Memory Structure

We will need to distinguish between the virtual and physical address spaces of the computer architecture. The virtual address space is the set of addresses that can be generated by the effective address calculation process during instruction execution. In other words, the address space the programmer works in while accessing data structures and calculating jump addresses.

The physical address space is the set of addresses corresponding to real memory locations. In some computers, there may be the trivial one-to-one mapping between the virtual and physical address spaces. However, many other computers have been able to use the virtual/physical address space distinction to advantage. Some of the more obvious advantages are:

(1) It frees programs of explicitly needing to know the amount of physical memory at a particular installation.

(2) It provides a powerful form of memory protection.

(3) If a very large virtual address space is available in the architecture and automatic paging of physical memory is provided, the programmer is freed from the manual task of managing overlays.

(4) If the virtual address space is uncomfortably small (e.g., 64K bytes) the address translation mechanism can provide a way to support large physical address spaces.

Some computer architectures use memory addressing schemes that make it difficult to identify the size of the virtual and physical address spaces. Hence, to make the definitions more specific, let the virtual address space of an architecture be the set of addresses a program can access (via the effective address calculation process) without having to execute any privileged instructions to alter segment or page descriptors in the memory translation unit (Fig. 3-1).

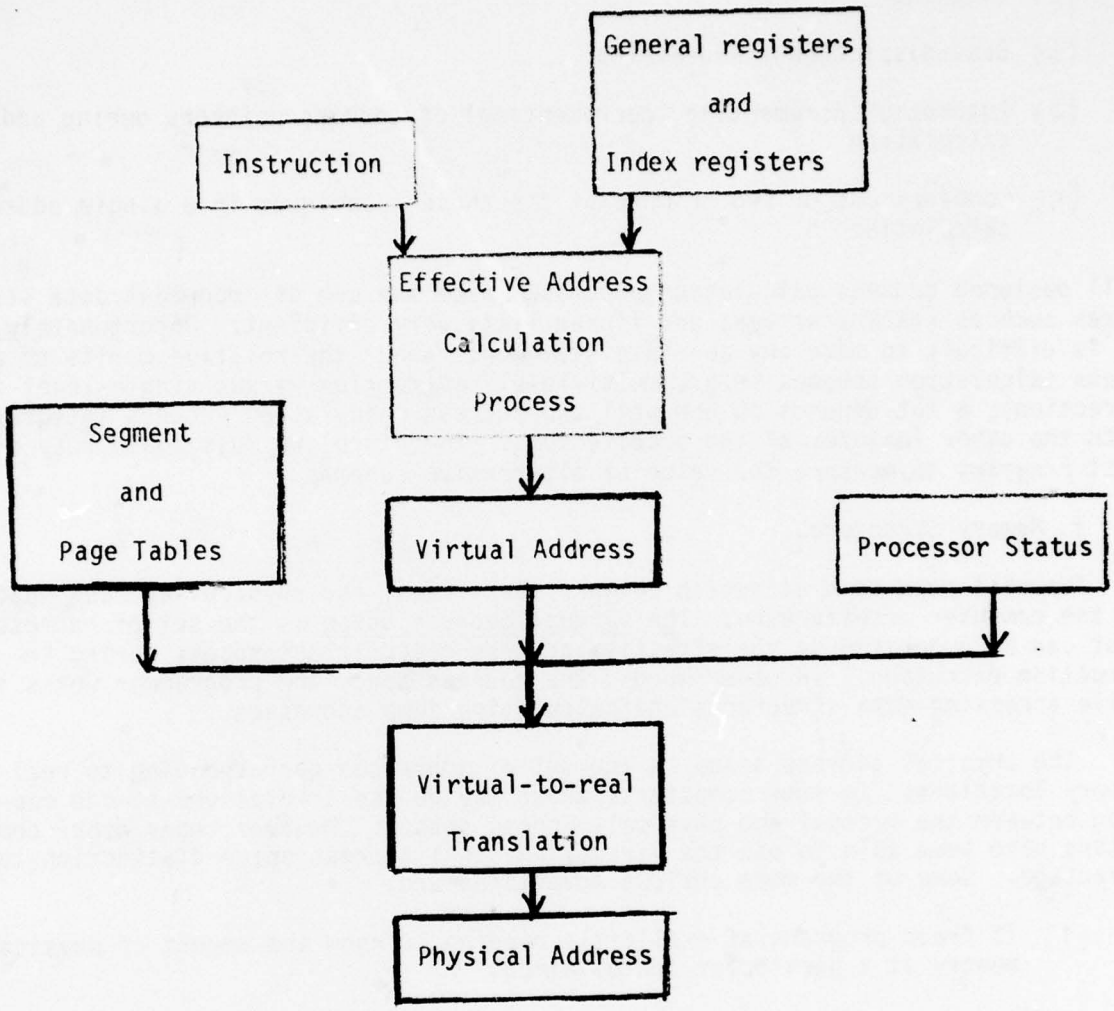


Figure 3-1: Memory Addressing Mechanism

For example, the IBM System/370 has a virtual address space of 2^{24} bytes, even though at any particular instant (because of the base-displacement addressing scheme) the processor can only access on the order of 64K bytes (i.e., 4K bytes relative to each of the 16 general purpose registers). The 370 is capable of altering its immediate addressability into the virtual address space via any number of non-privileged instructions (e.g., Load Address or Add Registers). By contrast, the DECsystem 10 is limited to 256K words of virtual memory even though many DECsystem 10 systems have more than 256K words of primary memory. In order for the DECsystem 10 to alter its addressability it must initiate a supervisor call to change the contents of a page table.

g. Control Structures and Protection

(1) Program Control

The instruction set of the architecture needs to include a complete set of operations to facilitate the flow of control in a program. Examples of instructions include: unconditional branch, conditional branch, subroutine call and return mechanisms, execute instructions, repeat instructions, and increment-jump-if-not-zero instructions. In addition, the operators described in paragraph 3.2.2 need to set condition codes for use by skip or conditional branch instructions.

Probably the point for most discussion in this section of the computer architecture is the structure of the subroutine call/return instructions. It is fairly well established by now that control stacks provide a convenient and powerful way to implement subroutine calls (e.g., the PDP-11, HP3000, B5500 and B6500). Whether control stacks are used or not, the selected architecture should facilitate the writing of reentrant, multi-level, and probably recursive procedures.

(2) Traps and Interrupts

An architecture needs well defined mechanisms to efficiently handle exceptional conditions during program execution (traps) and asynchronous signals from peripheral units (interrupts).

(3) Processor State

The state of all registers in the architecture of the processor must be clearly defined and readily saved and restored to facilitate process context switching. Some high performance implementations of processors provide two, four or even eight sets of registers that hold different processor states in order to speed context switching. An architecture should be careful not to preclude the use of such a technique (and just as importantly, the architecture should not require the use of the technique!).

For the purposes of the following evaluation process, we define the processor state as the set of bits that must be saved on either an interrupt or trap in order to resume execution of the interrupted program at some future time. Accumulators, the program counter, condition codes, and interrupt enable bits are all part of the processor state. Other registers commonly found in processors, but not part of the processor state, include the instruction register, memory address register, and memory buffer register.

(4) Protection

Protection is an important aspect of an achitecture even when the computer is not attempting to support the ubiquitous multiprogramming operating system. Examples:

Traditional: I/O Memory Management under control of supervisor programs. The supervisor is able to run in a "privileged state" that allows access to I/O devices and memory management tables.

Protection through addressability: Place Program Status Word and I/O control and status registers in virtual address space. A virtual to physical translation mechanism now provides the required protection.

Virtual machines: E.g., IBM CP67 and IBM VM operating systems.

(5) I/O Operation

The part of the computer architecture dealing with the operation of the I/O units is of critical importance in many of the real-time applications in which this proposed architecture will be used. Specification of the architecture of the I/O section is difficult because of the large amount of detail required. For example, the specific format of the byte or word a disk unit sends to the processor on an aborted access is not normally considered part of the computer's architecture yet the programmer must understand the format in detail to write the routine to handle the abnormal termination of the disk unit. If one attempted to completely define the architecture of all the I/O units of interest, the I/O section of the architecture would quickly become much larger than all the rest of the architecture combined. Moreover, for those computers not using magnetic tapes, say, the architectural specification of the magnetic tape unit is of little interest. Therefore, we will adopt the following attitude with regard to the specification of I/O units: specify only the general structure for communication with I/O units. In the PDP-11 this has been done by formalizing the concept of control and status registers. The IBM 370 has a set of I/O instructions, channel command words, and channel status words. Architecture supplements will then be issued as I/O units are developed for the base architecture. If these supplements are readily available, they can be used as guidelines for subsequent implementations of common I/O units (e.g., teletype interfaces, secondary memory controllers, processor-to-processor links).

4. ABSOLUTE CRITERIA

The criteria that were used during the initial screening of the computer architectures were divided into two types: absolute criteria that the architecture must meet and quantitative measures that were used to rank order the architectures. This section enumerates the absolute requirements and the next section presents the quantitative measures.

All the absolute criteria (with the exception of criterion 5) had to be satisfied by an implementation of the architecture which was operational by 1 January 1976. This eliminated speculative decisions based on promises or potentials that look inviting, but may not come to fruition. Failure to satisfy any absolute criterion resulted in the elimination of the architecture from further consideration. The nine absolute criteria are given below. The formal statement of each criterion is underlined, while explanations and examples are not underlined.

(1) VIRTUAL MEMORY SUPPORT. The architecture must support a virtual to physical translation mechanism. See Paragraph 3.2.5 above for a definition of virtual and physical address spaces. By vote of the selection committee, a mechanism commonly known as memory bank switching was deemed to be unsatisfactory.

(2) PROTECTION. The architecture must have the capability to add new, experimental (i.e., not fully debugged) programs that may include I/O without endangering reliable operation of existing programs. Architectures that use a privileged mode to protect vital registers and system resources generally meet this criterion.

(3) FLOATING POINT SUPPORT. The architecture must explicitly support one or more floating point data types with at least one of the formats yielding more than 10 decimal digits of significance in the mantissa.

(4) INTERRUPTS AND TRAPS. It must be possible to write a trap handler that is capable of executing a procedure to respond to any trap condition and then resume operation of the program. For example, if the processor receives a page fault trap from the address translation unit, it must be able to request the appropriate page be brought in from secondary storage and then resume execution. If resumption of execution is logically impossible (e.g., an attempt to store an operand into a read-only segment of virtual memory or fetch an instruction with a parity error) then the trap handler should be able to abort the program with an indicator of which instruction and/or operand caused the termination. A similar requirement exists for interrupts: the architecture must be defined such that it is capable of resuming execution following any interrupt (e.g., power failure, disk read error, console halt).

(5) SUBSETABILITY. Implementations of the architectures on small machines for dedicated applications must not be required to include features of the architecture intended for use on larger, multiprogrammed, multi-application configurations. At least the following components of an architecture must be able to be factored out of the full architecture:

Virtual-to-Physical Address Translation Mechanism

Floating Point Instructions and Registers (if separate from general purpose registers)

Decimal Instructions Set (if present in full architecture)

Protection Mechanism

This criterion proved to be extremely difficult to apply because the definition of what constitutes a subset machine is largely subjective.

In order to retain program compatibility across the implementations of the architecture, the following criterion must be satisfied:

> The trap mechanism of the architecture must be defined such that instructions in the full architecture, but not implemented in the subset machine, trap on the subset machine and that it be possible to write trap routines for the subset machine that allow it to interpretively execute those instructions not implemented directly in hardware (or firmware) and then resume execution. (This is an elaboration of absolute criterion 4.)

Existence of such subsets does not have to be demonstrated in an operational implementation of the architecture.

(6) MULTIPROCESSOR SUPPORT. The architecture must allow for multiprocessor configurations. Specifically, it must support some form of "test-and-set" instruction to allow the implementation of synchronization functions such as P and V.

(7) CONTROLLABILITY OF IO. A processor must be able to exercise control over any I/O Processor and/or I/O Controller (i.e., start, stop, continue, and examine state registers). This criterion was also difficult to apply because absolute control can be exercised in many different ways, some of which may not be among those intended.

(8) EXTENDIBILITY. The architecture must have enough spare capacity so that its instruction set can be extended as the needs arise. The architecture must have some method for adding instructions to the architecture consistent with existing formats. There must be at least one undefined code point in the existing opcode space of the instruction formats.

(9) READ ONLY CODE. The architecture must allow programs to be kept in a read-only section of primary memory.

5. QUANTITATIVE CRITERIA

The criteria that were used to rank the computer architectures quantitatively were:

(1) VIRTUAL ADDRESS SPACE

(a) V_1: The size of the virtual address space in bits.

(b) V_2: Number of addressable units in the virtual address space.

For example, the PDP-11 (as defined in the model 40) has a 16-bit virtual address which refers to 8-bit bytes. Hence

$$V_1 = 2^{16} \times 8 = 2^{19}$$

An item in memory is an addressable unit if it is possible to read and write that unit without altering other bits. (OR IMMEDIATE and AND IMMEDIATE were agreed by the CFA committee to be satisfactory methods for writing single bits.)

(2) PHYSICAL ADDRESS SPACE

(a) P_1: The size of the physical address space in bits.

(b) P_2: The number of addressable units in the physical address space.

Considering the PDP-11/40 again, which supports an 18-bit physical address referring to 8-bit bytes gives us:

$$P_1 - 2^{18} \times 8 = 2^{21}$$

(3) FRACTION OF INSTRUCTION SPACE UNASSIGNED

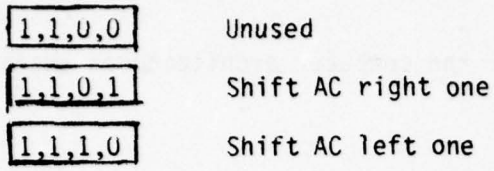
It is important to select an architecture that will allow reasonable growth over its expected lifetime. Let U be defined as the fraction of the instruction space in the architecture that is unassigned.

For example, consider the following small and primitive instruction set:

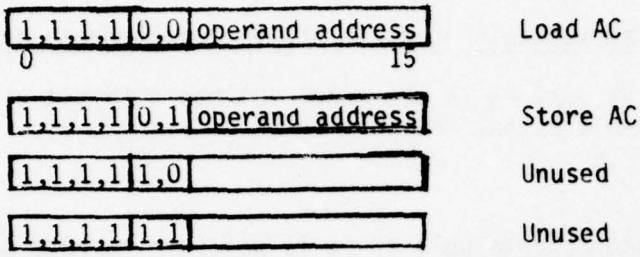
Short Instructions

0,0	Halt
0,1	Add one to AC
1,0	Clear AC

Regular Instructions



Long Instructions



The three short instructions consume 3/4 of the address space: no other instructions can be defined that begin with 00,01, or 10. The one unused regular instruction leaves 1/16 of the instruction space free and the two unused long instructions leave 1/32 of the instruction space free. Hence for this example U = 3/32.

In general, if u_i is the number of unassigned instruction of length i, then

$$U = \sum_{1 < i < \infty} u_i 2^{-i}$$

(4) SIZE OF CENTRAL PROCESSOR STATE

The amount of information that has to be stored or loaded upon interrupt and/or context swapping is clearly an important factor in the response of real time systems and in the overhead of multiprogramming systems. Let the processor state be defined as all the bits of information in a processor that must be saved in order to be able to restart an interrupted process at a later date. Processor states normally include the accumulators, index registers, program counter, condition codes, etc.

(a) C_{s1}: The number of bits in the processor state of the full architecture.

(b) C_{s2}: The number of bits in the processor state of the minimum subset of the architecture (i.e., without Floating Point, Decimal, Protection, or Address Translation Registers).

(c) C_{m1}: The number of bits that must be transferred between the processor and primary memory to first save the processor state of the full architecture upon interruption and then restore the processor state prior to resumption. This measure differs from C_{s1} above in that "register bank switching," where

provided for in the candidate architectures, may eliminate the need to save some registers in primary memory, while the instruction fetches required to save the state are included in C_{m1} but not in C_{s1}.

(d) C_{m2}: The measure analogous to C_{m1} for the minimum subset of the architecture.

To make an exact count of the number of bits in C_{s1}, include the following items:

Accumulators

Index registers

Base registers

Program counter and program status word

Protection registers

Memory mapping registers (or bank registers if these are used for virtual-to-real address transformations)

Interrupt mask register

Other registers visible to the operating system program that provide context for it or for a user program

If an architecture provides for several sets of certain registers to provide fast switching of multiple contexts, and if a program uses only one such register set when it runs in one context, then only one set of these registers is used in calculating C_{s1}. The argument is that context changing is enhanced by multiple sets of registers, so that to switch a process when all sets are filled requires only the switching of one set of context registers rather than all sets. A single context is a measure of the complexity required to run a single user. The C_{s2} measure is the same as C_{s1} with memory mapping registers, protection registers, and floating point registers omitted.

To measure C_{m1}, the bits in the processor state break down into three types. Some bits need to be stored and retrieved, some bits need only to be reset since the registers are duplicated in memory, and some bits need not be stored or reset. If context switching among multiple sets of registers is provided, then all bits in these registers contribute nothing to CM_1; they are neither stored nor set. Registers duplicated in memory are only counted once when they are read back into the processor registers. All other registers are counted twice because they must both be saved and restored. In addition to the bits in the registers, the bits in instructions required to store registers and reset them as necessary are counted in CM_1 and CM_2. Registers that are a fraction of a word in length count as a full word because storing and retrieving a part word entails a full memory cycle.

(5) VIRTUALIZABILITY

K: is unity if the architecture is virtualizable as defined in [Popek and Goldberg, 1974] otherwise, K is zero.

(6) USAGE BASE

These two measures are meant to be approximate indicators of the existing software and programmer experience base. A single individual designated by the CFA Committee chairman determined the value of these measures for all candidate architectures from a standard source. (The designated person was W. Burr.)

(a) B_1: Number of computers delivered as of the latest date for which data exists prior to 1 June 1976.

(b) B_2: Total dollar value of the installed computer base as of the latest date for which data exists prior to 1 June 1976.

(7) IO INITIATION

I: The minimum number of bits which must be transferred between main memory and any processor (central, or IO) in order to output one 8-bit byte to a standard peripheral device.

Although this measure is intended to give some insight into the responsiveness of an architecture, it is very difficult to construct an interpretation of the measure that serves this purpose well. The measure counts relatively few bits for some architectures, and this, in turn, makes the measure very sensitive to changes of a few bits. The I measure is also sensitive to several assumptions in exactly what actions are to be performed in doing the input/output operation, and where parameters for the operation are found. Unfortunately, this sensitivity made the I measure very arbitrary, and a rather inexact measure of input/output responsiveness. Consequently, the CFA committee elected to use the following guidelines to define as equitable a measure as possible.

The criterion I will measure an input/output subprogram must do the following actions:

(a) Select an input/output device whose address is given in cell 2998.

(b) Output a character on that device whose value is given in cell 2999.

(c) The first instruction of that program begins in cell 3000.

To avoid sensitivity to this particular phrasing of the problem, the following variations are permitted:

(a) The address of the character rather than the character itself may be passed as parameter.

(b) The code for the device may appear in isolation or combined with other bits to create an instruction to write on the given device. This makes it unnecessary to include instructions for creating a WRITE instruction containing the device select code passed as a parameter.

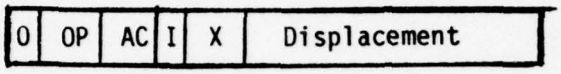
(c) Either or both parameters may be passed in a register or push-down stack instead of in the given memory locations.

No testing of device ready is necessary because the assumption is that this is representative of an interrupt process in which readiness is known.

(8) DIRECT INSTRUCTION ADDRESSABILITY

D: The maximum number of bits of primary memory which one instruction can directly address given a single base register, which may be used but not modified.

For example, the basic NOVA has the following memory reference instruction format, and addresses 16-bit words:



$D = 2^8 \times 16 = 2^{12}$ bits

(9) MAXIMUM INTERRUPT LATENCY

Let L be the maximum number of bits which may need to be transferred between memory and any processor (CP, IOC, etc.) between the time an interrupt is requested and the time that the computer starts processing that interrupt (given that interrupts are enabled). This may be interpreted as a measure of the longest non-interruptable instruction or sequence of instructions.

For example, the worst case for an IBM 370 may occur when an MVC instruction moves 256 bytes from one memory location to another. The MVC instruction itself is 6 bytes long. In this case:

$$L = (6 + 2 \times 256)\ 8 = 4144 \text{ bits}$$

(10) SUBROUTINE LINKAGE

The amount of information that has to be stored and loaded to accomplish a recursive subroutine call and return is clearly a factor in the efficiency of the architecture for structured programming. Let the user processor state be defined as all the bits of information which must be saved and restored during a subroutine call and return. This user state normally includes the accumulators, index registers, and program counter, etc.

J_1: The number of bits which must be transferred between the processor and memory to save the user state, transfer to the called routine, restore the user state, and return to the calling routine, for the full architecture. No parameters are passed.

J_2: The analogous measure to S1 above for the minimum architecture (i.e., without floating point registers).

This measure gives an indication of the size of overhead that might be encountered in doing subroutine calls in the worst case for the biggest and smallest machines

in the family. The bits counted here are related to the count in CS_1, CS_2, CM_1, and CM_2. By presumption, the bits that are stored for J_1 are exactly those for CS_1, except the protection registers, memory map registers, interrupt masks, and other registers that determine the global context for a program are not saved.

6. A COMPOSITE SCORE OF THE QUANTITATIVE CRITERIA

After developing the 17 quantitative criteria just discussed, the CFA committee had to determine how the performance of the candidate architectures on these criteria would be used to screen out all but three or four of the architectures for further consideration in the test program and software evaluation phases of the study. Clearly, the candidate architectures should be ordered relative to each of the 17 quantitative criteria and these independent orderings studied to detect weaknesses and strengths of the competing architectures. However, some unbiased scheme was ultimately needed to guide the committee in its selection of the final three or four computer architectures to undergo more intensive study. Various thresholding and weighing schemes were proposed, and the particular scheme that follows was chosen by the CFA committee at the 3 and 4 December 1975 CFA meeting. Several alternative methods were proposed at the third meeting on 28-29 February 1976, but the committee decided to continue to use the method presented below.

a. Relative Weighing of Criteria

Each voting organization of the Selection Committee was allowed a total of 100 points to be distributed among the various measures to indicate their relative importance to the command or activity. These were combined together to give weights of importance on the several measures. The measures were normalized as described below, and the final score for each architecture was the weighted sum of the normalized measures.

The weights were computed as follows: The actual value of W [x] was the number of points given to criteria x by all the committee members divided by the total number of points handed out. For example, if there were 25 committee members and a grand total of 200 points are given to V_1 by the committee members then

$$W[V_1] = 200/2500 = 0.08$$

Twenty-four of the voting CFA committee members responded to the questionnaire and these responses are given in Table 6-1. Table 6-2 shows the composite weights for Army organizations, Navy organizations, and the full Committee membership.

b. Normalization

When attempting to combine these quantitative measures we immediately faced two problems:

(1) Measures that inherently involve large magnitudes are not necessarily more important than smaller measures. For example, V_1 is on the order of 10^4 to 10^9 while K is either 0 or 1.

(2) The 17 measures are defined such that good computer architectures maximize some measures and minimize others. Specifically, the measures that a computer architecture should maximize are:

ORGANIZATION	CRITERION																
	V1	V2	P1	P2	U	CS1	CS2	CM1	CM2	K	B1	B2	I	D	L	J1	J2
NADC	4	2	4	2	5	1	1	2	2	8	0	0	19	11	19	10	10
NELC	4	2	5	2	28	2	2	2	2	3	4	1	27	4	4	4	4
SAM-D	8	10	6	8	8	6	4	4	3	8	6	2	6	8	6	3	4
SATCO	1	2	4	4	2	8	7	7	7	0	3	3	9	20	9	8	6
NAVCO	5	5	5	5	7	8	6	8	6	1	2	2	9	9	8	8	6
ARTAD	3	4	5	5	3	3	2	13	3	5	6	2	11	8	9	13	5
NUSCL	4	16	0	0	8	4	12	0	0	20	8	16	0	12	0	0	0
NUSCN	5	0	25	10	15	0	0	5	0	0	0	0	5	20	10	5	0
NTEC	2	2	15	15	5	4	3	4	3	5	2	2	10	10	10	6	2
NVLEX	7	2	7	2	2	2	2	7	12	2	1	1	7	12	15	7	12
NSWCW	0	25	0	10	0	5	0	5	0	5	0	15	15	10	0	10	0
FCDSD	10	10	5	5	0	5	0	5	0	10	5	5	10	10	10	10	0
NSWCD	3	3	10	10	5	3	3	3	3	5	0	0	15	10	15	6	6
NAVPG	0	2	0	2	0	20	3	15	5	5	10	0	12	8	3	10	5
NSRDC	5	5	13	12	2	1	1	7	1	10	4	1	5	18	5	8	2
ACSC	7	5	7	3	3	5	0	7	3	3	7	3	13	10	11	10	3
NUC	0	0	0	0	0	2	3	2	3	0	0	0	70	0	20	0	0
AVSCO	5	5	3	2	10	5	5	5	5	15	0	0	10	10	10	5	5
ECOM	2	2	2	2	4	6	15	6	15	4	6	2	6	4	6	2	16
AVLAB	2	2	2	2	4	6	15	6	15	4	6	2	6	4	6	2	16
NAVSE	4	2	13	8	10	1	1	5	1	9	2	1	4	15	8	13	3
PACMI	10	0	0	0	10	0	0	15	15	0	0	0	10	20	20	0	0
NATC	5	13	5	13	8	5	1	5	1	7	1	1	8	7	10	5	5
NRL	8	8	11	11	5	5	3	5	3	5	2	2	10	6	6	6	4

Table 6-1. Weighting of Quantitative Criteria by Member Organization

CRITERION	ARMY WEIGHTS	NAVY WEIGHTS	FULL COMMITTEE WEIGHTS
V1	.0412	.0444	.0433
V2	.0438	.0575	.0529
P1	.0425	.0706	.0612
P2	.0387	.0637	.0554
U	.0513	.0644	.0600
CS1	.0587	.0375	0466
CS2	.0675	.0219	.0371
CM1	.0700	.0544	.0596
CS2	.0713	.0319	.0450
K	.0500	.0587	.0558
B1	.0450	.0244	.0313
B2	.0200	.0281	.0254
I	.0875	.1419	.1238
D	.0912	.1081	.1025
L	.0812	.0969	.0917
J1	.0637	.0625	.0629
J2	.0762	.0331	.0475

Table 6-2. Quantitative Criteria Composite Weights

$V_1, V_2, P_1, P_2, U, K, B_1, B_2$, and D,

while the measures that should be minimized are:

$CS_1, CS_2, CM_3, CM_4, I, L, J_1$, and J_2.

Let our composite measure be a maximal measure and transform all minimal measures to maximal measures by taking the reciprocal:

$$X' = 1/X$$

To address the problem of differing scale in the measures, the following procedure was applied to develop a normalized measure.

Let the normalized X measure be denoted X. M is the number of architectures being evaluated. Normalized measures have the attractive properties that they all lie in the range (0,M); have a mean across the set of M architectures of unity; and the standard deviation of the set of normalized measures is in the interval $(o,\sqrt{m})$. Since all normalized measures have an average value of 1, no measure intrinsically dominates any other measure. We could have taken the normalization process a step further and adjusted the spread of each measure so that the measure gave a standard deviation of unity (or some other constant) across the set of architectures being evaluated. We did not do this for all measures. Some measures were better "discrimination functions" than others and we did not want in general to lose this information by further normalization. However, the committee has agreed that it is important to normalize the standard deviation of some of the measures. Specifically,

$$V_1, V_2, P_1, P_2 \text{ and } D$$

The standard deviation of these measures was normalized because they may differ by several orders of magnitude between candidate architectures, but the Selection Committee did not feel that the differences in utility, expressed by the measures, are orders of magnitude. For example, a logical address space of 2^{24} bytes was not considered to be 2^8 times more useful than a logical address space of 2^{16} bytes.

For those measures for which the committee decided to normalize the standard deviation to unity, the following procedure was applied:

$$\sigma_{\underset{\sim}{X}} = \left[\frac{1}{(m-1)} \cdot \sum_{1<m<M} (\underset{\sim}{X}_m - 1)^2 \right]^{\frac{1}{2}}$$

$$\underset{\approx}{X}_m = (\underset{\sim}{X}_m - 1)/\sigma_X + 1 = \frac{\underset{\sim}{X}_m + \sigma_X - 1}{\sigma_{\underset{\sim}{X}}}$$

c. Scaling and Composition of Measures

In order to combine the ten types of measures (17 individual measures) we used a simple, linear sum with each measure X scaled by its corresponding weighing coefficient W[X]. We used the maximal, normalized form of each measure. Let A be the composite measure. Then

$$\underset{\sim}{A} = W[V_1]\,\underset{\approx}{V}_1 + W[V_2]\,\underset{\approx}{V}_2 + W[P_1]\,\underset{\approx}{P}_1 + W[P_2]\,\underset{\approx}{P}_2 + W[D]\,\underset{\approx}{D}$$

$$+ W[U]\,\underset{\sim}{U} + W[K]\,\underset{\sim}{K} + W[B_1]\,\underset{\sim}{B}_1 + W[B_2]\,\underset{\sim}{B}_2$$

$$+ W[C_1]\,\underset{\sim}{C}'_1 + W[C_2]\,\underset{\sim}{C}'_2 + W[C_3]\,\underset{\sim}{C}'_3 + W[C_4]\,\underset{\sim}{C}'_4$$

$$+ W[I]\,\underset{\sim}{I}' + W[L]\,\underset{\sim}{L}' + W[J_1]\,\underset{\sim}{J}'_1 + W[J_2]\,\underset{\sim}{J}'_2$$

The weighing coefficients are defined so that they sum to unity and hence A is in fact a normalized measure with a mean of 1.

7. AUDITED VALUES OF CRITERIA FOR CANDIDATE ARCHITECTURES

This section reviews an audit of the absolute and quantitative measures applied in the initial screening of candidate architectures. The initial data were supplied by each of the architecture subcommittees, and were audited by H. S. Stone prior to the third CFA meeting (17-20 February 1976). Discrepancies and inconsistencies discovered in the preliminary audit were raised at the meeting of the architecture subcommittees on 17 February, at which time a complete and consistent set of data was obtained.

The data were presented to a meeting of the full CFA Committee on 18 February, with particular emphasis on data points that were subject to interpretation or whose accuracy was in question. A few changes to the data were made within the course of the next few days while the CFA committee was assembled and additional checking could be done in real time. A few more changes to the data were made even after the fourth CFA committee meeting in April 1976 because of a late audit of the AN/UYK-7 architecture data due to late submission of data on that architecture.

Tables 7-1 and 7-2 give the values for absolute and quantitative criteria for each of the nine candidate architectures.

The next section gives a composite description of how the various phases of the audit were conducted. Each of the criteria is discussed separately, with its relevance to the selection indicated. In many cases, there is an appraisal by the auditor (Stone) concerning the interpretation finally used to indicate if that interpretation results in an accurate or inaccurate measure. In a few cases, various interpretations might be applied to some criteria, with varying answers possible under different interpretations. The audit verifies that all answers are given under the same set of interpretations, having selected one of the several possible interpretations to apply uniformly to all architectures. Where an arbitrary decision was made, it was made with the concurrence of the architecture subcommittee chairmen and the Committee as a whole, and it was made so as to capture the intent of each criterion in question rather than to favor any particular candidate architecture. The resolution of these questions is given in the definition of the measure earlier in this report. Those criteria which are sensitive to arbitrary decisions are so indicated. The final rankings appear in Section 7.3.

a. Absolute Criteria

(1) Virtual memory. <u>The architecture must support a virtual to physical translation mechanism.</u>

The intent of this criterion is to take advantage of the widely used feature of many machines that is known as virtual memory. Many advantages accrue to architectures that support virtual memory, most notable of which is the ability to simplify programming by making the programmer unaware of the size of his physical memory, and by providing a mechanism for keeping active portions of a program in high-speed memory, without the need for programmer specification of data movement between high-speed memory and auxiliary memory.

The positive answers for this criterion listed in the table are not contestable, except for the AN/UYK-20 architecture. This architecture provides the page registers necessary for relocation, but does not limit the ability to change these registers to privileged programs. Some members of the committee felt that preventing user state access to the page registers was a necessary aspect of virtual memory; others disagreed. The consensus reached was that the AN/UYK-20 failed to meet the protection criterion, as well as the virtual memory criterion.

The ROLM 1664 and SEL-32 architectures were both deemed to have failed this criterion by vote of the CFA Committee on 18 February. Each of these architectures provide an address translation mechanism commonly known as "bank switching." It is different in kind from paging as implemented by most of the architectures and from the segmentation scheme used by the B-6700. In failing the ROLM 1664 and SEL-32 on this point, the CFA Committee was exercising a collective judgment that bank switching is not an acceptable alternative to a virtual memory address translation system.

We note here that the SEL-32 architecture may at some time in the future support an acceptable virtual memory mechanism. The manufacturer has made a recent statement indicating the intent to supply such a feature, and is working on it. Because this virtual memory feature did not exist on 1 January 1976, the SEL-32 architecture failed this criterion.

(2) Protection. The architecture must have the capability to add new, experimental (i.e., not full debugged) programs that may include I/O without endangering reliable operation of existing programs.

The intent of this criterion is to provide a mechanism in the hardware for aiding software development, and for preventing certain catastrophic software failures from occurring in the field. Machines without adequate protection features have hampered software developments. Presently available hardware protection features have proven to be valuable adjuncts to the architecture.

The AN/UYK-20 failed this criterion because it lacks memory protection. Any user can modify the contents of the relocation registers, and thereby read and write any word in memory.

One generic way for an architecture to fail the protection criterion is for a program to have the ability to put the machine into a noninterruptable state for an indefinite time. Since the usual situation for architectures is to permit interrupts at the completion of instructions, the most prevalent way to enter indefinitely long noninterruptable states is for an instruction to be nonterminating, and to be noninterruptable while executing. Architectures that permitted nonterminating instructions were carefully examined to identify if these were or were not interruptable.

Nonterminating instructions include instructions with infinite indirect chains and infinite chains caused by an EXECUTE instruction executing an EXECUTE instruction. Architectures with multi-level indirection include the ROLM 1664, AN/UYK-7, SEL-32, and the AN/UYK-20. The AN/UYK-20 fails the protection criterion for other reasons. The ROLM was originally considered to fail this

criterion but that decision was later reversed. The reasons for failing are rather subtle, but under pathological conditions the ROLM can indeed enter a nonterminating noninterruptable state. After 17 February it was pointed out to the auditor that the only way of entering this state has been made a privileged mode of operation on the ROLM architecture, and that a user cannot force the machine into the state with nonprivileged instructions, so the ROLM was considered to pass this criterion. The subtle problem is due to the ability to autoincrement indices used in indirect chains. If no autoincrement takes place, then an indirect chain can be interrupted and repeated after the interrupt has returned to the user. During an indirect chain, when an autoincrement takes place, no more interrupts are allowed until the chain terminates, since at that point the instruction cannot be repeated. Thus the situation creates a nonterminating noninterruptable state as part of the architecture (not just as part of this implementation of the architecture). However, autoincrementing indices during indirection is a privileged operation so no user can violate the protection criterion.

The SEL-32 passed the protection criterion at the time of the audit of 17 February on the basis that the SEL-32 Reference Manual is not specific about when interrupts may occur and when they may not, but the implementation of the machine permits infinite chains of indirect pointers to be interrupted between indirect reference. While this assertion is still accepted, there are questions that have to be investigated that cannot be answered with the present documentation. Specifically, consider the instruction BRANCH AFTER INCREMENTING WORD. This instruction increments a register and branches if the contents are nonzero. The target of the branch is computed by an effective address computation that could be nonterminating. After the register is incremented, the instruction cannot be repeated. Thus if the effective address chain is nonterminating, are interrupts locked out, and if not, how does the program recover from an interrupt correctly when the interrupt occurs during an indirect chain that terminates eventually?

Moving to the AN/UYK-7, the absence of adequate documentation and missing representation from the architecture subcommittee made it very difficult to assess whether or not the architecture met the protection criterion. During the audit it was revealed that nonterminating indirect chains are possible, and experience with real machines executing real programs has shown that the machines can enter noninterruptable states during the execution of nonterminating indirect chains. Thus AN/UYK-7 was considered to fail the protection criterion. Subsequent conversations between the auditor and Willis Hurd of Raytheon Service Organization established that the monitor clock interrupt can be honored during infinite indirect chains, and thus some form of protection is possible. This observation differs from the actual experiences related during the audit, perhaps because the monitor clock was not armed to interrupt for those cases reported. On this basis, the AN/UYK-7 is listed as having passed this criterion in Table 7-1.

The machines with EXECUTE instructions are the AN/GYK-12, AN/UYK-7, and IBM S/370. The IBM S/370 does not permit an EXECUTE instruction to execute another EXECUTE instruction. The AN/GYK-12 does permit this to occur, and presumably could have a chain of EXECUTES. During the audit procedures of 17 February, we accepted the statement that his chain is interruptable. The architecture manual is not specific on this point, so firm documentation of the point would have been required before accepting the AN/GYK-12 as a finalist.

The AN/UYK-7, by Hurd's comments, has no infinite noninterruptable chain, so even if infinite EXECUTE chains are possible, they cannot violate the protection criterion.

(3) Floating point. The architecture must explicitly support one or more floating-point data types with at least one of the formats yielding more than 10 decimal digits of significance in the mantissa.

The significance measure was determined as representative of the most stringent requirements actually encountered.

The AN/GYK-12 failed this criterion because it does not support floating point operations. The AN/UYK-7 failed because it supports a format with only 31 bits (9.2 decimal digits) of mantissa. It lacks three bits in the mantissa. Because this is so close to the borderline, one might consider more carefully the most stringent requirements on significance to determine how firm the 10 decimal digit criterion is. Had the AN/UYK-7 looked like an otherwise excellent architecture, it is likely that the committee would have reconsidered the floating point absolute criterion for it. As it turned out, however, the AN/UYK-7 did very poorly on the quantitative criteria scores (see paragraph 7.3) and the committee was not disposed to relax this requirement for an architecture with so little promise of being the final choice. It should also be pointed out here, that the AN/UYK-7 has only one floating point format, which uses 64 bits. However, the AN/UYK-7 inexplicably devotes 16 bits to its exponent (other machines do very well with only 8 bit exponents), and wastes another 16 bits which are unused. Other architectures with 64 bit floating point formats get about 16 digits of precision, while the AN/UYK-7 manages only about 9!

(4) Interrupts/traps. It must be possible to write a trap handler that is capable of executing a procedure to respond to any trap condition and then resume operation of the program.

The intent of this criterion is to permit extensions and subsets of an architecture to operate correctly so programs can be upward or downward compatible. The subsets and extensions may differ drastically in size, cost, and performance, but every program written for the native architecture can run on the subset or extension machine.

An example of a machine that fails this criterion is the PDP-11/20, the first model of the PDP-11 family, and not a representative of the PDP-11 architecture treated by this committee. Since the PDP-11/20 does not have a virtual memory facility, no traps or interrupts can occur at certain crucial points during the execution of an instruction. However, when virtual memory is added to the architecture, a trap for a nonpresent page can occur during the effective address to physical address computation. In the PDP-11/20 the program state word does not store sufficient information to retry an instruction if a page fault occurs.

Because of this problem, the state word for the architecture was modified by DEC so that the PDP-11/45 and other subsequent models can support virtual memory. This example shows how imperfections in an architecture can show up after its official release, and how architectures can evolve to correct imperfections.

Instruction retry appeared to be a real problem in the Interdata 8/32. The 8/32 has 16, 32, and 48 bit instructions. There is no way after a trap or an interrupt to tell whether the instruction which was being executed was a 16, 32, or 48 bit instruction. This is of no consequence for I/O and external interrupts, but for those traps which result from something caused by particular instructions, such as overflow or memory protection traps, it may be a problem, because it may be desirable to correct the cause of the fault, and then reexecute the instruction. The classic example of this is found in demand paging systems, where an instruction causes a memory protection trap, and the OS then decides that the trap was caused because the needed page was not in memory. The OS then loads the missing page, adjusts the contents of the memory protection and virtual to physical address translation registers, and then reexecutes the instruction which caused the trap. If there are different instruction lengths, there may be a problem in backing the location counter up to retry the instruction.

This was thought to be a problem for the 8/32. It was discovered, however, that the 8/32's location counter is not incremented after memory protection traps, so there is no need to back up. The committee was unable to determine, however, what happens to the location counter after arithmetic fault, illegal instruction, machine malfunction, and privileged instruction interrupts. The consequence of this was that the committee never was able to fully resolve whether or not the 8/32 satisfied this criterion.

(5) Subsetability. At least the following components of an architecture must be able to be factored out of the full architecture:

Virtual-to-physical address translation mechanism

Floating-point instructions and registers (if separate from general purpose registers)

Decimal instruction set (if present in full architecture)

Protection mechanism

The method for subsetting out portions of the architecture is specified to be by trap-and-emulate. Unlike other absolute criteria, this criterion did not require existence proofs of subset machines. The intent of this criterion is to permit the procurement of machines with reduced features for applications that do not use those features where there are side benefits in cost, size, and weight due to reduction of capability.

All architecture subcommittees reported that their architectures were subsetable. Because there was a great deal of disagreement in the exact measurement of subsetability, we could not challenge the replies during the audit. However, the B-6700 and the AN/UYK-7 have not been subsetted in the sense of the criterion, so that their subsetability is more speculative. In the case of the AN/GYK-12, the question of superset rather than subset was raised. What is the AN/GYK-12 architecture like with floating point?

(6) Multiprocessor support. The architecture must support some form of "test-and-set" instruction to allow for the communication and synchronization of multiple processors.

The intent of this criterion is to be sure that the basic architecture can support configurations with multiple processors. A necessary and sufficient addition to the instruction set of a typical stand-alone computer is an instruction that performs a "read-modify-write" operation in memory in such a way that the operation cannot be interrupted. The necessity of an instruction of this type has been well-known since the early 1960's, and each of the candidate architectures have at least one such instruction in their repertoires.

(7) Input/output controllability. A processor must be able to exercise absolute control over any I/O processor and/or I/O controller.

This criterion guarantees that a program retains absolute control over an input/output operation as part of the architecture definition. The interpretation of the criterion proved rather difficult. While all architectures necessarily permitted individual devices to be started and queried for status, there were varying degrees of control exercisable with respect to stopping the devices. It is reasonable to stop all input/output, or to stop selected devices. All architectures had some way of stopping a single device and stopping all devices, but how they did it varied widely in efficiency. In all cases it is efficient either to stop a single device or to stop all devices depending on the architecture, and in some cases both types of operations can be done efficiently.

Since the criterion did not spell out how to interpret input/output control, we accepted any definition of the ability to stop input/output as acceptable. All architectures passed this criterion.

(8) Extendability. The architecture must have some method for adding instructions to the architecture consistent with existing formats. There must be at least one undefined code point in the existing opcode space of the instruction formats.

The intent here is to provide for evolutionary development of the architecture. Some years in the future new operation codes or corrections to deficiencies in present operation codes may be invented, and the implementation of these can be done most readily through opcodes that are undefined today. To capture the full intent of this criterion it is most desirable for unimplemented operation codes to cause traps in the present versions of the architecture, rather than to be simple NO-OP's. However, this was not enforced during the audit because it was not spelled out explicitly in the criterion. All architectures have unused instructions, so all passed this criterion.

(9) Read-only code. It must be possible to execute programs from read-only storage.

This criterion is intended to permit an added degree of reliability of operation to be attained by permitting programs to be stored in a nonvolatile read-only memory. Discussion during the audit indicated that every program can be rewritten to be read-only on every architecture, even if that architecture does not support special types of instructions to facilitate this. The question, when phrased in absolute terms, had only one answer, namely, that every architecture supports read-only code. As such, all architectures passed this criterion. It might have been more meaningful to examine this question quantitatively.

b. Quantitative Criteria

(1) Virtual address space

V_1: The size of the virtual address space in bits.

V_2: Number of addressable units in the virtual address space.

Two aspects of these measures were open to interpretation. The audit settled on the following interpretation for treating bank switching.

The virtual address for a machine with bank switching is the address within a bank. The effect of bank switching is to increase the physical rather than the virtual address.

The second interpretation centered on the notion of "addressable unit." There are several degrees of addressability. An item may be fully addressable in the sense that it can be accessed by the address produced by an effective address computation. In descending order of power, other types of addressability include (1) the ability to produce an effective address for an item within a designated word, but if the effective address overflows that word into the next word, the addressability to the next word is not permitted, (2) addressing of an item within a word by an address designator not computed by an effective address calculation, and (3) addressing an item within a word via a mask or entity other than an address.

An example of the latter case is the OR IMMEDIATE and AND IMMEDIATE instructions of the IBM 360-370. To resolve the issue of interpretation, the subcommittees voted to accept a definition that stated the smallest addressable item is the smallest size item that can be individually written into or read from memory. This definition includes full addressability as defined above and each of the more limited addressing modes. V_1 and V_2 are equal for architectures with OR IMMEDIATE and AND IMMEDIATE instructions, or anything more powerful for accessing individual bits.

We mention here that the normalization of the virtual address space measures may be called into question. Generally speaking, there was agreement that larger virtual spaces are better than smaller ones, up to the point where the virtual address space is large enough. There was no consensus as to the relative goodness of an address space larger than necessary, nor was there a consensus on what size virtual address is large enough either for today or for the life cycle of the CFA. Under these circumstances, it is reasonable to pick some scheme that tends to favor bigger address spaces, and it is impossible to say beyond this how to normalize address spaces in the most meaningful and fairest possible way.

(2) Physical address space.

P_1: The size of the physical address space in bits.

P_2: The number of addressable units in the physical address space.

Where bank switching has been implemented, the physical address measures include the banks of memory available. For computers with virtual memory mapping, the physical address is the address resulting from the virtual-to-physical address translation. Physical address is defined apart from implementation since the physical address size need not be equal to the largest memory configuration yet delivered.

For all architectures the ratio of P_1 to P_2 is equal to the ratio of V_1 to V_2.

(3) Fraction of instruction space unassigned.

$$U = \sum_{1 \leq i < \infty} u_i 2^{-i}$$ where u_i is the number of unassigned instructions of length i.

The figures for U given in Table 7-2 have been determined from the manuals defining the architecture for each machine.

This measure is intended to show how much flexibility exists within the defined architecture to permit evolutionary changes in the future. As long as U is not equal to zero for an architecture it is possible to add new instructions, and there is no restriction on the number of new instructions that can be added. A very low value of U makes the addition of new instructions somewhat awkward, but does not prevent it. High values of U permit new instructions to be added with relative ease.

(4) The size of central processor state

CS_1: The number of bits in the processor state of the full architecture.

CS_2: The number of bits in the process state of the minimum subset of the architecture.

CM_1: The number of bits that must be transferred between the processor and primary memory to first save the processor state of the full architecture upon interruption and then restore the processor state prior to resumption.

CM_2: The measure analogous to CM_1 for the minimum subset architecture.

These measures give an approximation to the complexity of the implementation of the architectures, as well as a measure of the responsiveness of the architectures to worst-case context changes for interrupt processing. See Section 5 for a description of how this is measured.

The data for the Burroughs B-6700 is in question because it counts two registers that hold the additional mantissas required for double precision floating-point data. The B-6700 subcommittee chairman felt that these registers are not a necessary part of the architecture. However, they are indeed a part of the present architecture and are counted in this audit. Not counted are additional bits in the state due to the B-6700 display registers. There was insufficient information at the time of the audit to determine how these should be

treated. For this audit they are not counted per se, because CM_1, and CM_2 count some of the bits in the display that have to be stored to change contexts. In all likelihood a careful examination of the question of display registers would leave the numbers unchanged or increased.

(5) Virtualizability.

K: K is unity if the architecture is virtualizable in the sense of Popek and Goldberg, otherwise it is zero.

The intent of this criterion is to capture the concept of virtual machines that has shown to be of value in commercial computer systems. An architecture that supports virtual machines provides a mechanism for a privileged program to run as unprivileged and produce the results identical to those it produces as a privileged program. The importance of this idea is that an operating system can be run in user mode as a subsystem of another operating system. This permits the operating system in user mode to be protected against its own failures, and significantly enhances the ability of debugging operating systems. Moreover, with the virtual machine concept, many different operating systems can be run simultaneously in a multiprogramming fashion, thereby permitting users to run programs under the operating system that provides the facilities most desirable for each individual user.

The definition of virtual machine as provided by Popek and Goldberg in their article in CACM [Popek, 1974], is a very strict definition that guarantees that any operating system that runs on architecture X in privileged mode can also run on architecture X in nonprivileged mode. If an architecture fails this definition it may still support virtual machines in a more limited sense. For example, an operating system that uses only a restricted set of facilities of architecture X might be run in privileged and nonprivileged mode.

The audit of responses to this question accepted a positive reply only if some convincing evidence supporting the reply was offered as well. The IBM 360-370 is virtualizable, and standard operating systems run on virtual machines for this architecture. The PDP-11 is virtualizable because one such machine has been successfully demonstrated. This machine is a PDP-11/45, not a PDP-11/70 which has been the subject of the remainder of this audit. However, the differences between the PDP-11/45 and PDP-11/70 are inconsequential with respect to virtualizability, and the manufacturer expressed willingness to supply hardware modifications as nonstandard options to PDP-11/70 implementations similar to those used for the virtual PDP-11/45 implementation. On this basis, the virtualizability of the PDP-11 family was accepted by the full committee.

All other positive responses from subcommittees were changed to negative responses in Table 7-1, because they either proved to be incorrect or there was no evidence to support the positive response.

(6) Usage base.

B_1: The number of computers delivered as of the latest date for which data exists prior to 1 June 1976.

B_2: Total dollar value of the installed computer base as of the latest date for which data exists prior to 1 June 1976.

The figures for B_1 and B_2 fall into three categories, commercial end-user machines, commercial OEM machines, and military machines. The end-user machines, B-6700 and IBM 360-370, count present installations only. Machines delivered in the mid-1960's and retired and replaced since then are not counted. Contrary to this, the OEM machines count total number of deliveries. For the OEM market, the total number of deliveries is a good estimate of present installations, but is a high estimate. Figures for military machines are based on total installations. ROLM and Data General Nova sales were added together for the ROLM (NOVA) architecture.

This measure is intended to give some impression of the existing base in terms of programmers and experienced support staff (as identified by the number of installations) and in terms of other tangible items such as software base, documentation, and similar aspects proportional to the total investment in an architecture (as measured by the dollar value of the installations).

Regardless of how these measures are gathered, it is clear that the IBM 360-370, PDP-11, and ROLM (NOVA) have orders of magnitude more installations than the other architectures, and the investment in the IBM 360-370 overwhelms the sum of the investments in the other architectures.

(7) Input/output initiation.

I: The minimum number of bits which must be transferred between main memory and any processor in order to output one 8-bit byte to a standard peripheral device.

This measure was intended to give some insight into responsiveness of an architecture, but it was very difficult to construct an interpretation of the measure that would serve this purpose well. The measure counts relatively few bits for some architectures, and this in turn makes the measure very sensitive to changes of a few bits. For example, is the 8-bit byte counted in the measure or is it excluded from the measure? Since the measures are normalized to preserve ratios, an additive bias of eight bits could have a profound effect. When the eight bits are excluded a 64-bit value for I is four times as bad as a 16-bit value, but when the eight bits are included, the relative values become 72 and 24 which is only a factor of three in range. A compromise solution was worked out for this measure, and is described in Section 5.

The PDP-11 and Interdata 8/32 architectures can perform the operation with a single 16-bit instruction. The SEL-32 and ROLM computers also performed reasonably well for this measure, which is to be expected for the class of minicomputers in general. The low value recorded for the IBM 360-370 comes as a result of using the WRITE DIRECT instruction rather than the standard input/output procedures involving channel commands. It was argued successfully that the WRITE DIRECT and READ DIRECT mechanism in the architecture would be used to achieve highly responsive input/output in extreme conditions, although the original intent of these instructions is for communication between computers rather than for input/output.

(8) Direct instruction addressability.

D: The maximum number of bits of primary memory that one instruction can directly address given a single base register, which may be used but not modified.

Large D fields generally simplify programming because they reduce the need to set base registers and to maintain addressability.

Because an architecture may have several different formats, each with different D fields, the committee required that the format selected for this measure be the one used for standard LOAD and STORE operations, or the equivalent thereof. This eliminated anomalies, like the MOVE CHARACTER LONG in the IBM 370 architecture, from consideration.

(9) Maximum interrupt latency.

L: The maximum number of bits that may need to be transferred between memory and any processor between the time an interrupt is requested and the time that the computer starts processing that interrupt (given that interrupts are enabled).

The intent of this measure is to find the worst-case for responsiveness within the architecture. L measures the longest possible execution time for a single noninterruptable instruction or noninterruptable sequence of instructions.

Architectures with nonterminating noninterruptable instructions have infinite L measures and are so indicated in Table 7-2.

(10) Subroutine linkage.

J_1: The number of bits that must be transferred between the processor and memory to save the user state, transfer to the called routine, for the full architecture. No parameters are passed, and the subroutine call must be recursive.

J_2: The same number for the minimum architecture.

This measure gives an indication of the size of overhead that might be encountered in doing subroutine calls in the worst case for the biggest and smallest machines in the family. The bits counted here are related to the count in CS_1, CS_2, CM_1, and CM_2. By presumption the bits that are stored for J_1 are exactly those for CS_1, except that it is not necessary to save the protection registers, memory map registers, interrupt mask, and other registers that determine the global context for a program. Architectures with small processor states or that have LOAD/STORE MULTIPLE instructions show up well on these measures.

Architectures with independent registers for floating-point data such as IBM S370 and PDP-11 are penalized in these measures, because they include the overhead for storing and retrieving the floating-point register files, while this almost never occurs in typical subroutine calls in these architectures.

The B-6700 figures ignore the display registers since it is not known how to treat them in this measure.

c. Quantitative Scores

The weighing functions and candidate scores are given in Table 7-3. Those appearing in Table 7-3 are composite functions due to combined Army/Navy weights. Breakdowns for Army weights and Navy weights have been reported by William Burr in a memo dated 19 March 1976. Although the ordering of the architectures was slightly different for the composite and individual weighting functions, the orderings were remarkably stable. Table 7-4 shows the contribution of each criterion to the composite scores of each architecture.

An analysis of this stability for the composite weights is the subject of the following section.

#	ABSOLUTE CRITERION	CANDIDATE CFA's IBM S/370	INTER DATA 8/32	ROLM 1664	DEC PDP -11	UNIVAC AN/ UYK-7	SEL 32	BURROUGHS B6700	UNIVAC AN/UYK -20	LITTON AN/GYK -12
1	Virtual Memory	Y	Y	N	Y	Y	N	Y	N	Y
2	Protection	Y	Y	Y	Y	Y	Y?	N	N	Y?
3	Floating Point	Y	Y	Y	Y	N	Y	Y	Y	N
4	Interrupts/Traps	Y	?	Y	Y	Y	Y	Y	Y	Y
5	Subsetability	Y	Y	Y	Y	Y?	Y	Y?	Y	Y?
6	Multi Processor	Y	Y	Y	Y	Y	Y	Y	Y	Y
7	I/O Controllability	Y	Y	Y	Y	Y	Y	Y	Y	Y
8	Extensibility	Y	Y	Y	Y	Y	Y	Y	Y	Y
9	Read Only Code	Y	Y	Y	Y	Y	Y	Y	Y	Y
	SUMMARY	Y	?	N	Y	N	N	N	N	N

Y Yes, Meets Criteria
N No, Fails Criteria
Y? Conditional Yes
? Unresolved

Table 7-1. Candidate CFA Values for Absolute Criteria

#	QUANTITATIVE CRITERIA	CANDIDATE CFA's IBM S/370	INTER DATA 8/32	ROLM	DEC PDP 11	UNIVAC AN/ UYK-7	SEL 32	BURROUGHS B6700	UNIVAC AN/UYK -20	LITTON AN/GYK -12
1	V_1**	27	27	20	20	24	22	24	20	20
2	V_2**	27	27	20	19	24	22	20	17	20
3	P_1**	27	27	22***	25	23	26***	24	20	29
4	P_2**	27	27	22***	24	23	26***	20	17	29
5	U	.371	.355	.039	.043	.15	.450	.019	.125	.219
6	CS_1	1344	1632	1008	1168	992	304	306	1328	1008
7	CS_2	576	576	112	144	448	288	204	336	752
8	CM_1	3168	1120	1882	736	1472	768	408	2256	1344
9	CM_2	1312	32	544	480	1472	704	408	720	1088
10	K	1	0	0	1	0	0	0	0	0
11	B_1	17,300	185	13,800****	14,700	346	75	90	400	30
12	B_2*****	16,000	14	169	311	147	23	207	8	6
13	I	64	16	48	16	128	64	169	80	32
14	D	15	27	20	19	18	22	18	20	20
15	L	6192	560	114	112	2112	288	255	∞	1376
16	J_1	1904	2368	1360	1040	1280	960	459	1408	1344
17	J_2	1136	1280	320	400	1280	960	459	640	1088

** These values are of the form 2^x where x = indicated data except for B6700 which is of the form 3 (2^x).
*** with memory bank switching.
**** Includes NOVAS.
*****In \$ X 10^6

Table 7-2. Candidate CFA Values for Quantitative Criteria

Criterion	Weight	Architecture	Score
V_1	.04	I 8/32	1.68
V_2	.05	PDP-11	1.43
P_1	.06	IBM 370	1.36
P_2	.06	AN/GYK-12	0.94
U	.06	ROLM	0.92
CS_1	.04	B06700	0.91
CS_2	.04	SEL 32	0.86
CM_2	.05	AN/UYK-7	0.46
K	.06	AN/UYK-20	0.44
B_1	.03		
B_2	.03		
I	.12		
D	.10		
L	.09		
J_1	.06		
J_2	.05		

Table 7-3. Composite Quantitative Scores Using the Combined Army/Navy Weights

CRITERIA	IBM S/370	INTER DATA 8/32	NOVA	DEC PDP 11	UNIVAC AN/ UYK-7	SEL 32	BURROUGHS B6700	UNIVAC AN/UYK -20	LITTON AN/GYK -12
V_1	2.693	2.693	0.344	0.344	0.621	0.400	1.213	0.344	0.344
V_2	2.756	2.756	0.455	0.445	0.726	0.509	0.455	0.439	0.455
P_1	1.156	1.156	0.386	0.560	0.411	0.759	0.659	0.367	3.541
P_2	1.962	1.196	0.443	0.516	0.467	0.807	0.425	0.419	3.528
U	1.884	1.805	0.198	0.219	0.762	2.286	0.095	0.635	1.112
CS_1	0.535	0.440	0.713	0.615	0.724	2.365	2.350	0.541	0.713
CS_2	0.456	0.456	2.345	1.824	0.586	0.912	1.287	0.781	0.349
CM_1	0.325	0.921	0.548	1.402	0.701	1.344	2.530	0.457	0.768
CM_2	0.160	6.572	0.386	0.438	0.142	0.298	0.515	0.292	0.193
K	4.500	0.000	0.000	4.500	0.000	0.000	0.000	0.000	0.000
B_1	3.321	0.035	2.642	2.819	0.066	0.014	0.017	0.076	0.005
B_2	8.531	0.007	0.090	0.166	0.078	0.012	0.110	0.004	0.000
I	0.599	2.398	0.799	2.398	0.299	0.599	0.227	0.479	1.199
D	0.642	3.665	0.665	0.653	0.647	0.736	0.659	0.665	0.665
L	0.051	0.569	2.795	2.845	0.150	1.106	1.249	0.000	0.231
J_1	0.583	0.469	0.816	1.068	0.868	1.157	2.420	0.789	0.826
J_2	0.575	0.510	2.042	1.634	0.510	0.680	1.423	1.021	0.600

Table 7-4. Normalized, Unweighted Scores for Each Architecture and Quantitative Criterion

8. ROBUSTNESS OF THE EVALUATION PROCEDURE

a. Analysis

There was some concern about the role of the weighting of the measures, the normalization of the measures, and the measures themselves in the selection of finalists. In this section, we show that, given the weights applied by the committee as an indication of the importance of idealized concepts, the finalists selected are very insensitive to the exact details of the selection procedure. Almost any reasonable methodology for measuring the key concepts quantitatively would select the same finalists.

Table 7-3 gives the joint Army/Navy weights to two decimal places. (More precision is maintained by the analysis program.)

Note in Table 7-3 that the K measure (virtualizability) and B_1 and B_2 measures (present machine base) are given moderate weightings, but the sum of these weights is roughly 12% of the total weight. These measures were awarded to the IBM S/370 and PDP-11, with the NOVA/ROLM garnering some of the B_1 weights. However, these measures are accounted, they should properly advance the PDP-11 and IBM 360 about 12% ahead of the other candidate architectures, except for NOVA/ROLM which would lag about 6% behind the other two. Thus these measures in a sense present an advantage of 12% for the IBM S/370 and PDP-11 architectures that tends to select these machines if the other measures are equal for all architectures.

But the other measures are not equal for all architectures. The PDP-11 and IBM S/370 have characteristics that show up well in the quantitative measures in relation to other machines. For example, the size of virtual memory is large for the IBM S/370, and is maximum among all architectures. No matter how this measure is normalized, it should show up as an advantage for the IBM 360. Thus the IBM 360 should do well for V_1 and V_2, and fairly well for P_1 and P_2, because it has a good ability to make use of large memory. These measures have weights that total up to another 21% of the sum of the weights. When the 21% is considered in conjunction with the 12% mentioned above, we find that the IBM 360 is clearly at or near the head of the list in items that carry 1/3 of the total weight. In a field of nine candidates, this is almost sufficient to ensure its selection among the top three candidates, since it is extremely unlikely that three other candidates can gain enough weight on the remaining items so that all three can exceed the score of the IBM 360.

A similar analysis holds for the PDP-11, which, as mentioned above, properly shares 12% of the weight with the IBM 360, and to a lesser extent with ROLM, and receives additional bonuses for items that are characteristically good for minicomputers. These include a low interrupt latency and a fast input/output response. Even with noise in the I measure, as given in Table 7-2, under any reasonable measure of the input/output responsiveness, the PDP-11 must come out near the top, because this is one of the functions for which the architecture has been designed. The I and L measures carry 21% of the weight, which, together with the 12% mentioned above, comes to at least 33% of the weighted measures for which the PDP-11 should perform well. Again, using the logic mentioned above, it becomes extremely unlikely with this advantage that the PDP-11 will fall outside the first three.

The Interdata 8/32 was selected by the quantitative criteria in spite of the fact that it suffered from the 12% handicap split among IBM S/370, PDP-11, and ROLM. However, since its architecture is similar to the IBM 360 architecture, it does well in the 21% of the weighting functions involving memory for which the IBM 360 architecture does well, and because it is a minicomputer with an architecture optimized for minicomputer applications, it competes reasonably well in measures of responsiveness.

Given the relative strengths of the PDP-11, IBM 360-370, and Interdata 8/32, and the weights applied to the concepts on which the selection is to be based, there is very little freedom left to enable a different candidate to excel in sufficiently many categories to place in the top three.

Note that the PDP-11 and IBM 360-370 will do well in at least one third of the weighted categories under any normalization, and the union of these categories represents about one-half of the total weights. Some architectures do very poorly throughout these measures, and consequently have a severe handicap with regard to doing well enough in the other categories to become finalists.

For example, among the categories V_1, V_2, P_1, P_2, B_1, B_2, K and I, totaling 45% of the total weight, the ROLM architecture does moderately well in the I measure and very well in B_1, but otherwise has a consistently poor showing. Likewise, the AN/UYK-7 and SEL-32 do very poorly on all but at most one or two of these measures. The B-6700 and AN/GYK-12 do a little better on these measures, but still lag well behind the IBM 360-370 and PDP-11 under any reasonable method of scoring the functions prior to weighting. On the remaining quantities carrying 55% of the weight, if three architectures were to garner enough points to depose the PDP-11 and IBM 360-370 from being finalists, the scores would have to be split almost entirely among the three finalists with very little if any weighting awarded to the PDP-11 and IBM 360-370. Given the data gathered for these other measures, the PDP-11 and IBM 360-370 do sufficiently well on the measures to stay in as finalists for any reasonable normalization of the measured data.

The architecture whose final weighting is most likely to change under other normalizations is the Interdata 8/32, simply by virtue of the fact that it gained the most under the present normalizations. In a few categories such as the D measure and the memory measures V_1, V_2, P_1, and P_2, the Interdata 8/32 obtained a large fraction of the available weights in the present normalization. Other normalizations are unlikely to increase the points gained here by Interdata, and may reduce them. However, again the finalist selection is rather insensitive to the normalizations, because Interdata gained 60% more weight than the nearest non-finalist. Rather drastic changes in the normalization methodology would be required to drop the Interdata architecture from the top three contenders.

The results therefore are largely insensitive to the precise manner in which key characteristics are measured and normalized. There is, however, possible sensitivity due to the possibility of including or omitting a candidate architecture. A good tutorial explanation of the phenomenon that might be experienced here appears in the "Mathematical Games" section of Scientific American [Gardner, 1976]. The problem can be posed roughly as follows:

Suppose you are given three items with quantitative measures of your relative preferences. When you compare items in pairs you prefer A to B, A to C and B to C. In this comparison you naturally prefer A among the set of three objects. Yet it is possible to develop a consistent set of quantitative data that selects C (apparently the least desirable item) over A when three objects are considered as a collection.

Martin Gardner offers an example to support his statement. The example is rephrased to put it in the context of the architecture selection.

In evaluating architecture A, you give it a ranking of 3 on all criteria. Architecture B has a rank of 2 on 56% of the criteria, 4 on 22%, and 6 on 22%. Architecture C ranks 1 on 51% and 5 on 49% of the criteria. (Higher ranks indicate stronger preferences.)

In comparing items pairwise, A ranks over B on 56% of the criteria, A ranks over C on 51% of the criteria, and B ranks over C on (1 x .22) + (.22 x .51) + (.56 x .51) = .6178 of the criteria. If all criteria are weighted equally, then pairwise rankings suggest A is best and C is worst. However, if you pick an architecture by giving it credit for each criterion on which it does best and no credit otherwise, then C is best and A is worst. A is best among the three architectures on .56 x .51 = .2856 of the criteria, B is best among the three on (.44 x .51) + (.22 x .49) = .3322 of the criteria and C ranks top among the three on .49 x .78 = .3822 of the criteria. Thus it is a three-way race if we pick C over A, but if B drops out, then A is picked over C. This phenomenon is known as "Simpson's paradox" according to Gardner. Its explanation is due to the fact that in a three-way race, B receives some weighted preferences that are distributed unequally among A and C when B drops out.

In the architecture selection, if any architecture is omitted from the quantitative measures then the weights it gained are redistributed among those that remain, and the redistribution is done unequally. For example, if the SEL-32 were dropped, then the B-6700 would stand to gain inordinately on the CS_1 and CS_2 measures, since much of the weight on these measures given to the SEL-32 would revert to the B-6700.

The arguments used earlier were based on a field of nine candidates, since it is unlikely for a few of those nine candidates to do well enough on 55% of the weighted measures to catch up with the leading contenders that have done so well on the other 45% of the weighted measures. When the field is reduced to five contenders or six contenders, then we have to look at more of the weights than the 45% examined above to ascertain that race is not sensitive to the normalizations. Note that Simpson's paradox is a possibility, but not necessarily a highly probable outcome of a preference evaluation of the type conducted. Although we have not looked at all possible competitions with fewer than nine architectures, the three finalists tend to gain most of the available weight on so many criteria that it is unlikely that a nonfinalist can drop out of the race and give a sufficient quantity of weight to another nonfinalist to change the outcome. Therefore we conclude that the data appears to be robust for competitions with fewer than nine architectures, but this conclusion remains to be tested more rigorously.

b. Conclusions

The initial screening process has been conducted under reasonable technical guidelines, with the selection of the finalists determined by the merits of the architectures as measured by factors deemed to be important by the Army/Navy CFA Committee. Measures were made accurately and consistently, within the limitations due to time and availability of accurate data. The selection of three finalists was a function of the preponderance of the heavily weighted data in their favor, rather than due to idiosyncracies or biases in the measuring and weighting process.

9. CHRONOLOGY OF DEVELOPMENT OF CRITERIA

In an attempt to make the rationale behind the absolute and quantitative criteria as clear as possible, this section describes the sequence of developments that lead to the final set of nine absolute and 17 quantitative criteria described in this report.

An initial memo [Fuller and Smith, 1975] discussing selection criteria was distributed at the first CFA meeting on 1 and 2 October 1975. This memo was the result of discussions over the course of the summer of 1975 at the Naval Research Laboratory between S. Fuller, D. Parnas, D. Siewiorek, J. Shore, and W. Smith. At the first CFA meeting, a Selection Criteria subcommittee was formed to develop specific criteria the CFA committee could use in the selection of a computer architecture. The members of the Selection Criteria subcommittee were: S. Fuller (Chairman), R. Estell, L. Haynes, N. Tinkelpaugh, and D. Wise.

The Selection Criteria subcommittee met on 28 and 29 October 1975, and the results of the subcommittee were reported at the second CFA meeting on 3 and 4 December 1975 [Fuller, et al., 1975]. In the Selection Criteria subcommittee's report seven quantitative criteria were proposed: V_1, P_1, U, CS_1, CM_1, K and B_1. The S, M, and R measures of test programs were also briefly described and proposed as quantitative criteria capable of providing the most realistic measure of an architecture's capabilities. At the second CFA meeting, however, it was the consensus of the committee that it would be impractical to undertake a benchmarking effort on all nine candidate architectures, and the decision was made to expand upon the absolute and quantitative criteria developed by the Selection Criteria subcommittee, in order to use them as an initial screening procedure, prior to any test program effort. With this in mind, the CFA committee proceeded to add the following quantitative criteria to the criteria originally proposed: V_2, P_2, CS_2, CM_2, B_2, I, D, L, J_1, and J_2. Some of these quantitative measures were added because the full CFA committee felt they were important attributes of a computer architecture missed by the original set of criteria, but others were added because the absolute and quantitative criteria were now an important part of an initial screening process to be used before a benchmarking of the architectures was done. I, D, J_1, J_2, CS_2, CM_2, and, to a limited extent, V_2 and P_2 were added to try to gauge the effectiveness of the computer architecture that the committee agreed would be more completely examined in the test program phase of the evaluation process.

Each of the candidate architecture subcommittees were responsible for determining the value of their machine on the defined criteria. H. S. Stone was appointed "auditor" of the responses from the architecture subcommittees. On 17 February 1976, Harold Stone and the chairmen of the architecture subcommittees and several other members of the CFA committee met to review the responses of the architecture subcommittees, and to challenge and verify or correct the values in question. The results of this meeting, further discussion at the third CFA meeting on 18-20 February 1976, and additional work on the part of the auditor after the meeting, resulted in values for the absolute and quantitative criteria accepted by the committee and are reported in the 1976 audit by H. S. Stone [Stone, 1976].

REFERENCES

1. Amdhal, G.M., Blaauw, G.A., and Brooks, F.P., "Architecture of the IBM System/360," IBM Journal of R and D 8, 2, April 1964, 87-101.

2. Bell, C.G. and Newell, A., Computer Structures: Readings and Examples, McGraw-Hill, 1971.

3. Brent, R.P., "On the Precision Attainable with Various Floating Point Number Systems," IEEE Trans. on Computers C-22, 6 (June 1973), 601-607.

4. Fuller and Smith, "Selection Criteria for the Army/Navy Computer Family Architecture," CFA memorandum, 29 Sept. 1975. Distributed at 1-2 October 1975 CFA Selection Committee Meeting.

5. Fuller, Estell, Haynes, Tinklepaugh, and Wise, "Selection Criteria for the Army/Navy Computer Family Architecture," report of the Computer Family Architecture Committee Selection Criteria Subcommittee, 20 November 1975.

6. Gardner, Martin, "Mathematical Games," Scientific American, 234, 3, Mar. 1976, pp. 119-124.

7. IBM System/370 Principles of Operation, IBM Publication CA22-6942-2, White Plains, N.Y.

8. Stone, H.S. (editor). Introduction to Computer Architecture, SRA, Inc., Palo Alto, Calif., 1975.

9. Stone, H.S., "An Audit of the Selection Criteria for Computer Family Architecture," CFA memorandum, Jan. 1976.

10. Popek, G.J., and Goldberg, R.P., "Formal Requirements for Virtualizable Third Generation Architectures," Communications of the ACM, vol. 17, no. 7, July 1974, 412-421.

APPENDIX A: FORM FOR ARCHITECTURE EVALUATION

ARCHITECTURE EVALUATION

CANDIDATE ARCHITECTURE:_____

ABSOLUTE CRITERIA

(1) Virtual Memory Support	Yes	No
(2) Protection	Yes	No
(3) Floating Point Support	Yes	No
(4) Interrupts and Traps	Yes	No
(5) Subset ability	Yes	No
(6) Multiprocessor Support	Yes	No
(7) Controllability of IO	Yes	No
(8) Extendability	Yes	No

QUANITATIVE CRITERIA

CATEGORY	MEASURE	SCORE
Virtual Address Space		
	V_1:	_____bits
	V_2:	_____units
Physical Address Space		
	P_1:	_____bits
	P_2:	_____units
Fraction of Instruction Space Unassigned		
	U:	_____
Size of Central Processor State		
	C_1	_____bits
	C_2	_____bits
	C_3	_____bits
	C_4	_____bits

CATEGORY	MEASURE	SCORE
Virtualizability		
	K:	Yes No
IO Initiation	I:	_____bits
Direct Instruction Addressability		
	D:	_____bits
Maximum Interrupt Latency		
	L:	_____bits
Subroutine Linkage		
	J_1:	_____bits
	J_2:	_____bits

Signature of Chairman_____

Date_____

APPENDIX B: BALLOT FOR WEIGHTING QUANTITATIVE CRITERIA

CATEGORY	MEASURE	WEIGHT
VIRTUAL ADDRESS SPACE		
	V_1:	_____
	V_2:	_____
PHYSICAL ADDRESS SPACE		
	P_1:	_____
	P_2:	_____
FRACTION OF INSTRUCTION SPACE UNASSIGNED		
	U:	_____
SIZE OF CENTRAL PROCESSOR STATE		
	C_{s1}:	_____
	C_{s2}:	_____
	C_{m1}:	_____
	C_{m2}:	_____
VIRTUALIZABILITY		
	K	_____
USAGE BASE		
	B_1:	_____
	B_2:	_____
IO INITIATION		
	I:	_____
DIRECT INSTRUCTION ADDRESSABILITY		
	D:	_____
MAXIMUM INTERRUPT LATENCY		
	L:	_____

CATEGORY	MEASURE	WEIGHT
SUBROUTINE LINKAGE		
	J_1:	_____
	J_2:	_____
	TOTAL	100 POINTS

ORGANIZATION_____

SIGNATURE_____

DATE_____