INSTRUCTION STREAM MONITORING OF THE PDP-11

by

Charles J. Neuhauser

May 1979

TECHNICAL NOTE NO. 156

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

The work described herein was supported in part by the Army Research Office - Durham under contract no. DAAG29-78-0205.

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

TECHNICAL NOTE No. 156

May 1979

INSTRUCTION STREAM MONITORING OF THE PDP-11

by

Charles J. Neuhauser

ABSTRACT

An instruction stream monitoring system based on the Stanford Emulation Laboratory facilities is described. The particular target machine analyzed is the PDP-11. Because the emulation facility is efficient at the emulation task it is possible to support monitoring an interactive system, such as UNIX, without unreasonable performance degradation. Raw data from the monitor, in the form of "event counts", is further processed by an off-line analysis program. The analysis methods are described and an example of the results is given.

The work described herein was supported in part by the Army Research Office - Durham under contract no. DAAG29-78-0205.

1. Introduction

1.1 Objectives

In this report we will describe a system currently in use at the Stanford Emulation Laboratory which is capable of measuring limited aspects of a dynamic PDP-11 instruction stream. Rather than measure a "hard" realization of a PDP-11, an emulation based technique is used in which the image machine is nominally equivalent to a PDP-11/05 CPU. Our purpose in this report is to describe the capability of our current measurement system and to assess its scope and limitations. The measurement system operates in conjunction with an analysis system which summarizes the measured data. The methods of analysis will be discussed in detail, but an in depth interpretation of the resulting data will be left as the subject of a future report.

One of our objectives in the Stanford Emulation Laboratory is to make data such as that described here available to a large population of researchers and computer architects. We expect that in the near future a tape library will be generated containing data from a series of carefully selected experiments representative of the PDP-11 environment. At a later point in time comparable data tapes for other architectures (e.g. S/360, NOVA etc.) will be available.

1.2 General Description of the Measurement System

Measurements of the PDP-11 are made using an emulation technique based on the resources of the Stanford Emulation Laboratory [1,2]. Figure 1-1 shows a highly schematized drawing of the laboratory which consists of two sub-sections: the Emmy processor system and the PDP-11 processor system. In operation the Emmy processor, its micro-store and its main memory provide the emulation base for the target (or image) processor, in this case the PDP-11. Emulation of the PDP-11 has been described in detail in an earlier report [3].

Peripherals of the target system are emulated cooperatively by the Emmy processor and the PDP-11 processor system. The PDP-11 system supports the UNIX operating system. Operationally, the PDP-11 acts as an I/O channel and handles device access requests on behalf of the emulated target machines [4]. Peripheral devices are never accessed directly by the Emmy processor. Note: even though the particular target machine under discussion here is a PDP-11, the fact that the laboratory support processor is a PDP-11 has no material effect on the discussion. There is no particular system advantage to having identical support and target processors, although in practice some operational steps, such as establishing the emulated system, are expedited.

Figure 1-2 shows the overall flow of data in the measurement and analysis system. During emulation of the target machine the emulator

based in the Emmy processor maintains event counters in the microstore. These counters tabulate various events which occur at the level of target machine instruction execution, for example type of instruction executed. At the end of a test run the event counters are frozen and dumped to a file. This file is analyzed by a Fortran program which is run under an emulated 370 system on the Emmy. An example of the output of the analysis program is shown in the appendix.

1.3 Assessment of the Approach

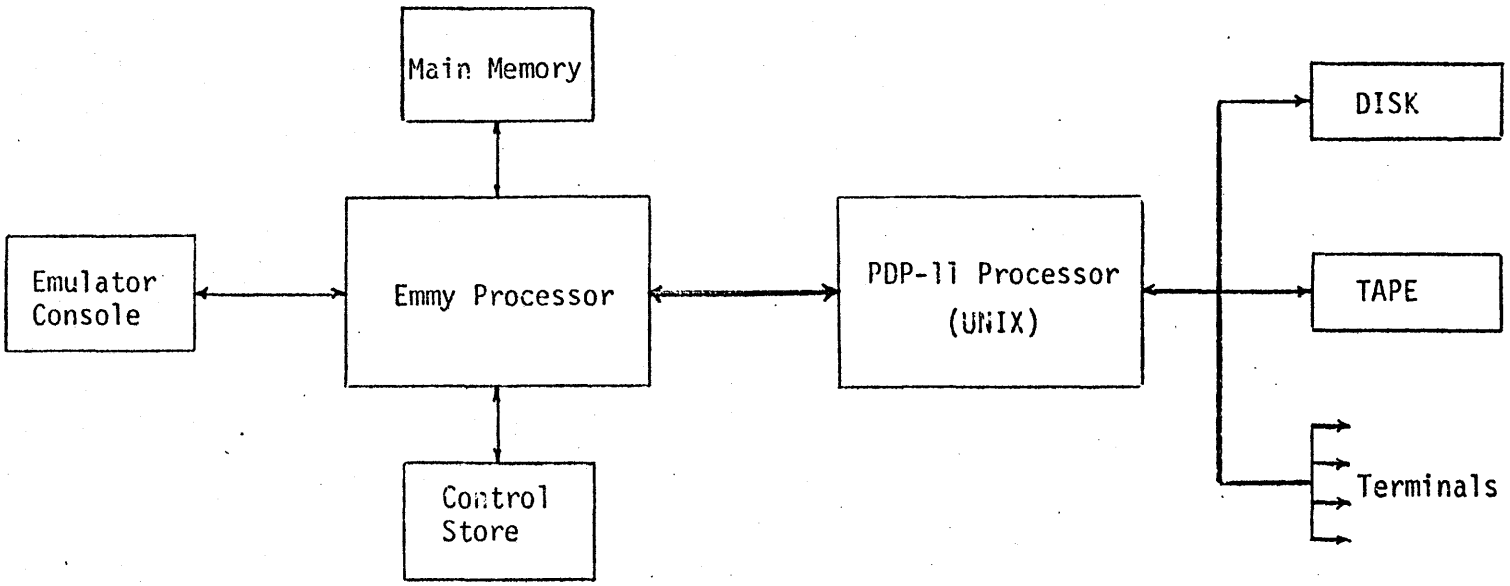
There are, of course, a wide spectrum of approaches by which the data presented here might have been gathered. For example, hardware monitors might be used, or alternatively a software monitor might be employed to step the target program and examine the results at each point. Over the range of monitoring methods many trade-offs must be made. The most important of these involve the time and space consumed by the method. At one extreme is hardware monitoring, which has little or no effect on image machine execution time, but has a high cost in terms of space if one considers the fast, expensive hardware registers and control necessary to carry out the measurement function. At the other end of the spectrum is the software monitor which may be relatively inexpensive in terms of resources but may decrease target machine execution rate by factors of 1000 or more [5].

The emulation approach we have implemented lies, in a subjective sense, between the two approaches mentioned above. In terms of time the emulation monitor impacts the emulation of the target machine by about 20%, although one must bear in mind that the use of an emulated target has already slowed execution by a factor of five in the case of the PDP-11. In terms of space, the monitor mechanism consumes about half of the available control store in the Emmy processor (8K out of the available 16K bytes).

Aside from space and time there are also other limiting factors involved in the use of a monitoring mechanism. In a general sense, the purpose of a monitor is to capture some aspect of the target machine state at each execution of a target machine instruction. Hardware monitors can only retain a small part of the image machine state. They are also limited in what fraction of the machine state they may access. In contrast, the software monitor, which is already heavily burdened in time can easily afford to capture the entire state change brought about by target instruction execution. In fact, the limiting case is the generation of a "trace tape" which is an exact record of program execution. With the emulation approach the entire state transition of the target machine is available for examination, but only a small part of the state may be retained.

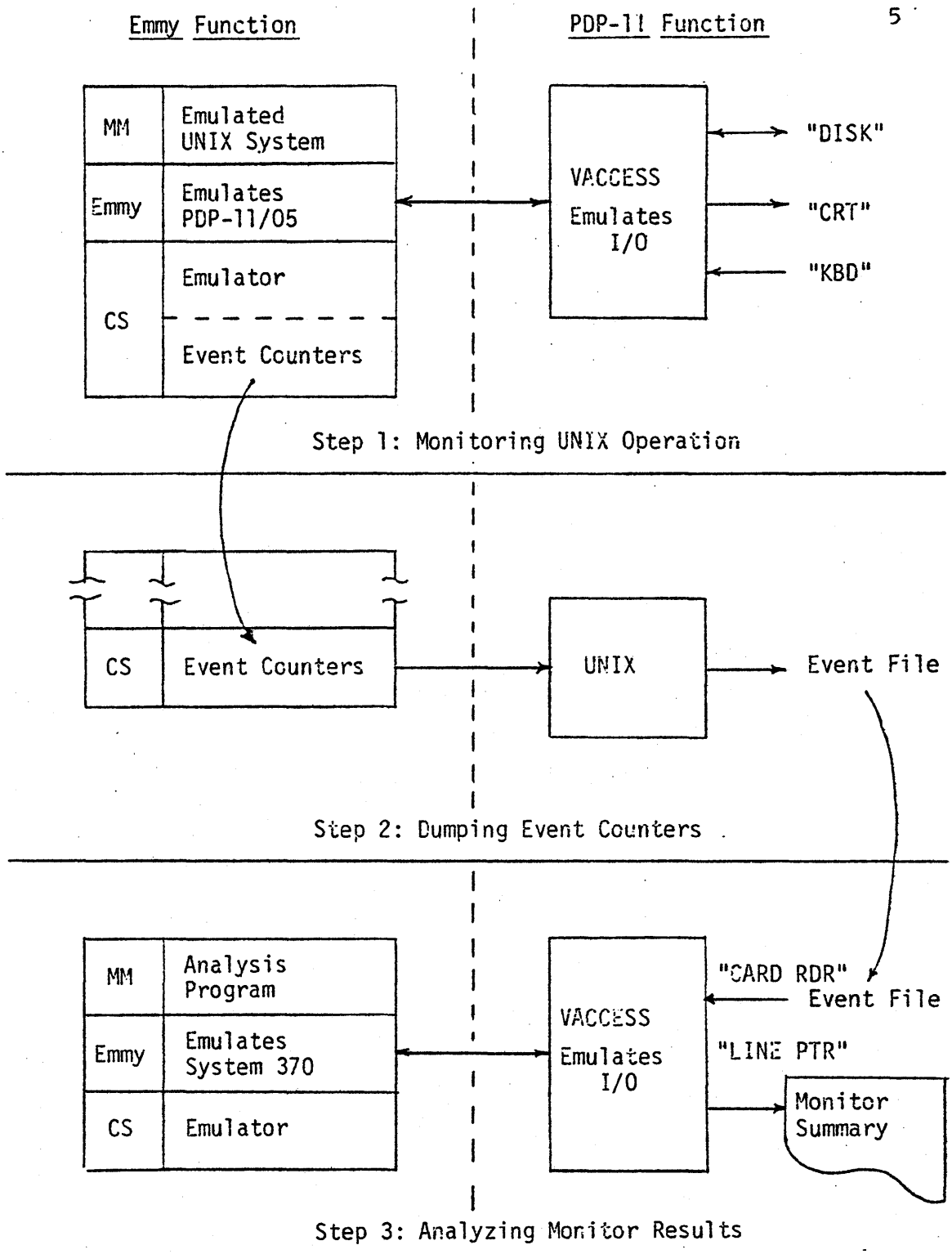
When a software base monitor generating a trace tape is used all analysis of the instruction stream may be deferred until the program

test run is complete. Other types of monitors, because they have limited space available, must perform some data reduction during the data collection operation. This means that one must make decisions before the fact concerning the data to be produced by the experiment. Generally, most of the information concerning target machine state will be lost during the data collection phase. This happens in a way which prevents its recovery later. Fortunately, most of the results we are interested in are statistical in nature and are preserved through the data collection process.



Schematic Organization of the Laboratory

Figure 1-1



Sequencing of a Monitored Experiment

Figure 1-2

2. Description of the Monitoring Technique

In this section we will discuss, in some detail, the techniques used by the PDP-11 emulator and its associated monitor. We are interested in two aspects of the monitor; first, the general approach used and second, the special problems which arise in monitoring the PDP-11. Many of these problems are related to monitor control and the handling of emulated ("soft") interrupts by the system.

2.1 Basic Emulator Structure

The PDP-11 emulator used in the monitoring experiments has a structure somewhat different from that described in a previous report [3], in that the emulator described here is more capable and has a more complex I/O system. Figure 2-1 shows the basic structure of the PDP-11 emulator and I/O system. Code emulation is straightforward and involves a fetch/decode phase followed by an operand formation phase, an execute phase and finally a phase in which data is written (if required).

I/O in the emulator is handled somewhat independently of the code emulation by a combination of I/O emulation routines in the Emmy and an access program ("VACCESS") running on the PDP-11 under UNIX. When data must be read or written during code emulation the I/O routines are called into play. These routines emulate specific PDP-11 devices, such as TTY and disk, and operate so that the code emulator functions as if it were communicating with real PDP-11 peripherals. In response to code emulator I/O read and write operations the I/O emulator formulates a read/write request and issues it to the VACCESS program via a system of mailboxes. In essence, the requests placed in the mailboxes are I/O channel commands and the VACCESS program acts as a simple I/O channel.

Accesses to devices are completed by the VACCESS program through normal UNIX device interfaces. When the requested operation is completed VACCESS places a status indication in the mailbox and signals the Emmy via an interrupt. A note: the actual data transfer requested may take place directly to the Emmy main store or indirectly through the mailbox.

"Hard" interrupts to Emmy, that is interrupts directly from the PDP-11 or the Emmy timer, are captured by the interrupt handler. This routine sorts out the interrupts and signals the I/O emulator. After examining the interrupt source and cause, the I/O emulator may issue a "soft" interrupt to the code emulator, in effect allowing the emulated PDP-11 to see a device interrupt. In other cases, the I/O emulator may simply use the interrupt to sequence internally without notifying the code emulator.

2.2 Basic Monitor Structure

To monitor instruction execution micro-code is added to the PDP-11 emulator as shown in Figure 2-2. The additional code may be divided into two parts: the monitor itself and "capture points". During emulation of the PDP-11 the current instruction is broken down in successive stages to select the appropriate emulator action. As each piece of decoded information is produced it is saved by micro-code at the appropriate capture point. Actually, all the information gathered by the capture points is saved in one 32 bit Emmy register. In a sense, the information in this register is an encoding of the actual instruction which was executed.

The emulator monitor (if enabled) is entered before the next instruction is executed. Here the encoded information from the capture points is used to control the updating of the event counters. For example, an event counter is maintained for each PDP-11 opcode. Everytime the monitor is entered the appropriate opcode event counter is incremented by one.

Event counters are maintained in control store which because of its fast access time helps to minimize the impact of monitoring on emulator operation. This also means that main memory is freed for the exclusive use of the target machine, although this is not very important in the PDP-11 emulation, which uses only half of the available physical Emmy main memory. Actually, in the current version of the monitor about 480 operand counters are held in an unused area of Emmy main memory. These counters are associated with floating point operations and are thus lightly used.

At the point that the monitor is entered the last PDP-11 instruction has been completely decoded and executed. The information from the capture points is in a form that is very efficient for the monitor to decode and analyze. This makes the monitor action relatively simple and reduces its impact on emulator performance. Furthermore, the analysis programs can be simple since much of the instruction decoding has already been done by the emulator, running on the Emmy which is specifically designed for such tasks.

2.3 Encoded Instruction Format

As instruction emulation proceeds the information gathered at the capture points is encoded and saved in an Emmy register designated here as the "information register". Figure 2-3 illustrates the format of this 32 bit register. There is a different format for each basic PDP-11 instruction format, which may be one of the following:

Format	Examples
Two operand	MOV, ADD, BITB
One operand	INC, DECB, TST
Branch	BR, BPL, BEQ, SOB
CCOP	CLN, SEN, NOP
Other	RTS, EMT, Interrupt

The low twelve bits of the information register are common for all formats and contain the control store address of the opcode counter for the given instruction. This address is also used by the monitor as a base to access other information such as the information register format and secondary event counter addresses. Remaining fields of the format (except for BOF) are taken directly from the target instruction:

Field	Instruction Bits
Source Mode/Register	11-06
Destination Mode/Register	05-00
Branch Offset	07-00
CCOP Qualifier	03-00

One field, BOF (for Branch Outcome Flag) is used to encode dynamic information related to instruction execution. The BOF flag is set if the branch (conditional, unconditional or SOB) is successful and is cleared otherwise. Excluding generated data and addresses, the BOF flag is the only information generated by PDP-11 instruction execution which is not explictly defined by the instruction itself.

2.4 Monitor Control

Enabling of the emulation monitor is controlled in two ways, externally or by the execution of selected PDP-11 instructions. External control is exercised by the PDP-11 console based on the Datapoint 2200. This console may activate or deactivate the monitor. Provisions have also been made to reset the monitor event counters from the console.

Although the console may activate the monitor in the sense of "arming" it, the monitor will not operate until a selected instruction stream event occurs. For monitor control during emulated PDP-11 execution of UNIX we have elected to use bit 8 of the PDP-11 PS (Processor Status). When this bit is set (i.e. "1") monitoring is disabled even though it may be marked as "active" by the console. This feature allows the monitoring of selected sections of the UNIX operating system. Note: bit 8 of the PS is undefined in the PDP-1/05 and thus may be use for monitor control in the emulator.

Currently, our most important use of dynamic monitor control is to disable monitoring whenever an "illegal" PDP-11 instruction is

encountered. The basic PDP-11 emulator is an exact copy of the PDP-11/05 [6]. Because much of the UNIX system code was written for other PDP-11 models, especially the 11/34, many "illegal" instructions are encountered. Illegal instructions cause processor traps, and special UNIX code is entered to emulate the function of these instructions. This process is, of course, very expensive in terms of time. By disabling the monitor when illegal instructions are being emulated we may measure code execution on various PDP-11 models even though the emulator supports only the model 11/05. Ideally, of course, these illegal instructions would be implemented directly in micro-code. However, using the PDP-11 based emulator code has allowed us to execute code containing floating point operations immediately and avoid a long period of micro-code construction and testing. The penalty, unfortunately, is reduced performance of the PDP-11 emualtor by a factor of two (for simple extended instructions like SXT and ASHC) to ten (for floating point operations).

2.5 Details of Monitor Control

Figure 2-4 shows some of the details of the PDP-11 emulator monitoring scheme. Almost all aspects of instruction monitoring can be handled in a straight forward manner as described above. However, events and instruction which cause a change to the processor status require special attention since these events may cause bit 8 of the PS to change and thus change the enabling of the monitor.

Actually, there are two "monitors" in the PDP-11 emulator. Located before the code emulator is the instruction monitor, which increments event counters associated with explicitly executed instructions. In addition, located within the interrupt resolution logic, is the "interrupt monitor" which increments the event counter for interrupts. Interrupts counted include interrupts due to device status changes and to execution of illegal instructions. The interrupt resolution logic is entered anytime there is a potential change to the PDP-11 status since such status changes may produce interrupts by lowering the PDP-11 priority level. Potential sources of status changes are:

1) Ordinary instructions (e.g. MOV, INC etc.) which modify device registers and cause an immediate interrupt. This might happen when an interrupt enable bit is turned on.

2) Ordinary instructions which modify the PS register.

3) Soft interrupts from the I/O emulator which may, depending on the current priority, turn into real emulator interrupts.

4) RTI instruction execution.

5) Programmed interrupts from EMT, TRAP, BPT or IOT.

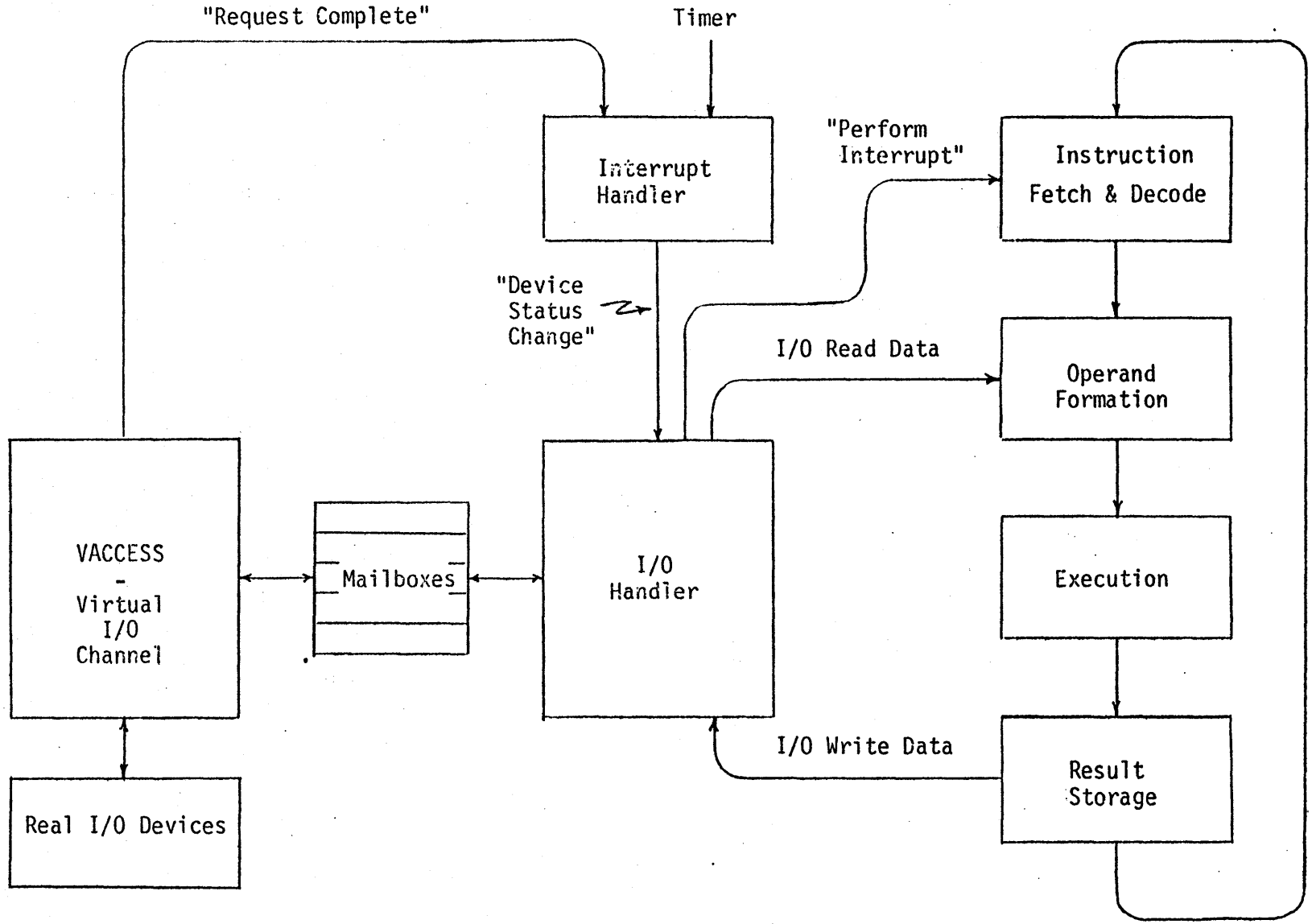
6) Illegal instruction interrupts.

Emulator action in the interrupt resolution logic evaluates each interrupt or status change request and decides whether or not to execute an interrupt. If an interrupt is executed the interrupt resolution logic is entered recursively to check for possible status changes which might generate new interrupts. Once the interrupt resolution logic reaches a quiescent state, code emulation is resumed by entering either the monitor or the code emulator. Usually events which might cause an interrupt save the current monitor enable state before entering the interrupt resolution logic, then on exit this bit is checked to see whether the instruction which caused the interrupt should be tabulated. Only interrupts which arise from "soft interrupts" bypass the instruction monitoring since they will eventually be counted by the interrupt monitor when they become "real" interrupts.

2.6 Modifications to UNIX to Support Dynamic Monitoring

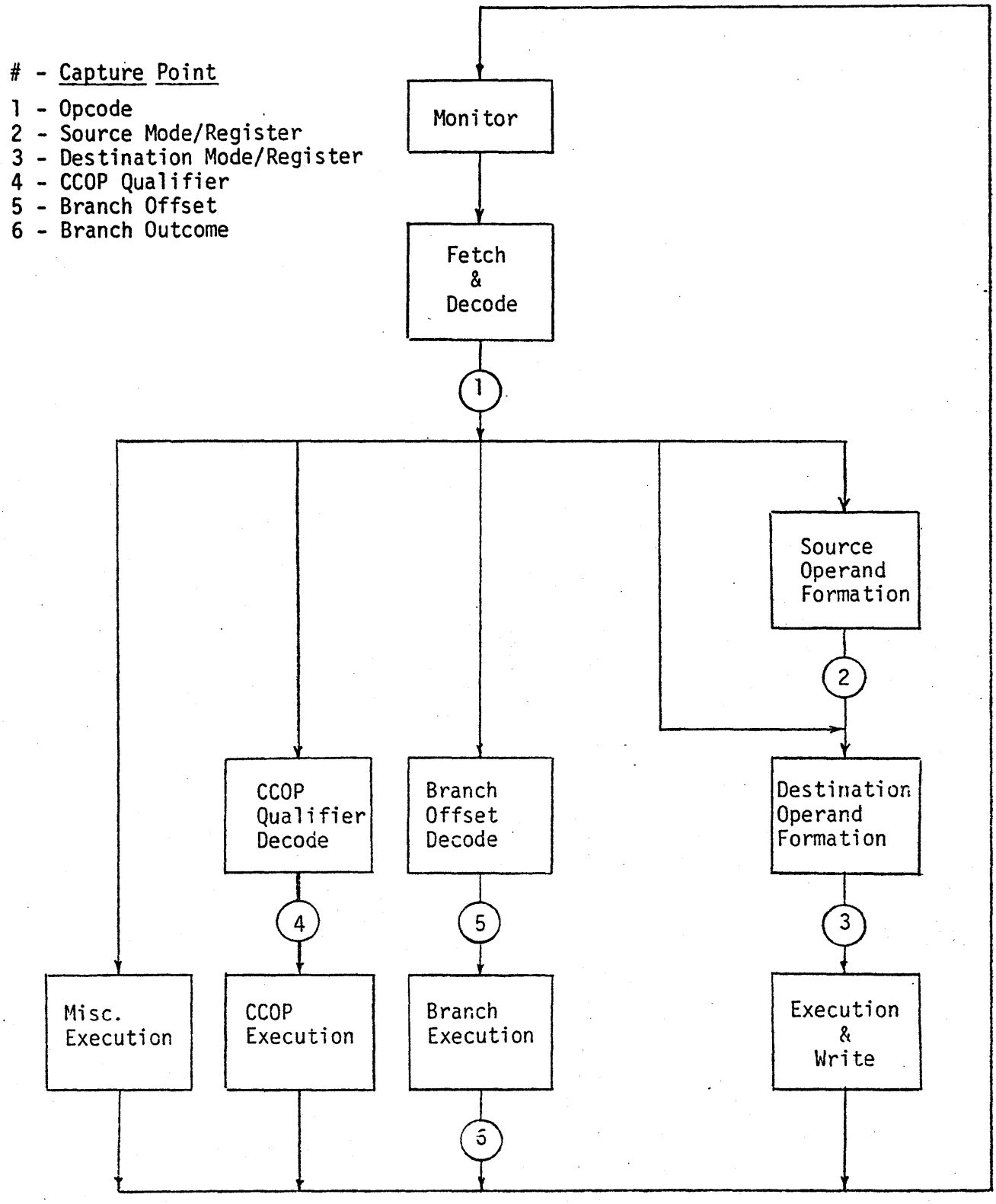
Two minor modifications have been made to MINI-UNIX to facilitate monitor control. First, the PS word which is loaded on CPU interrupts is modified so that bit 8 of the new PS will be set. This inhibits the monitoring of "illegal" instruction emulation by the target machine. Second, a new instruction, "RETTRAP", is added (code 000007). This instruction is operationally equivalent to an RTI instruction (and is counted as such), but it copies bit 8 of the current PS word to the old PS word. The RETTRAP instruction replaces a single RTI instruction used only in the UNIX machine exception handler.

In UNIX a machine exception (e.g. illegal instruction, system TRAP etc.) may cause execution of user supplied code which handles the exception. This code is normally run whith the same PS as the code which caused the exception and is entered by execution of an RTI instruction operating with the old user PS and a user supplied PC. By replacing the RTI with an RETTRAP user supplied code for illegal instruction exceptions (this is always a floating point unit emulator) operates with the monitor enabling specified by the illegal instruction exception. This means that only floating point instructions execution is counted and not the user supplied emulator code executing on the target machine.



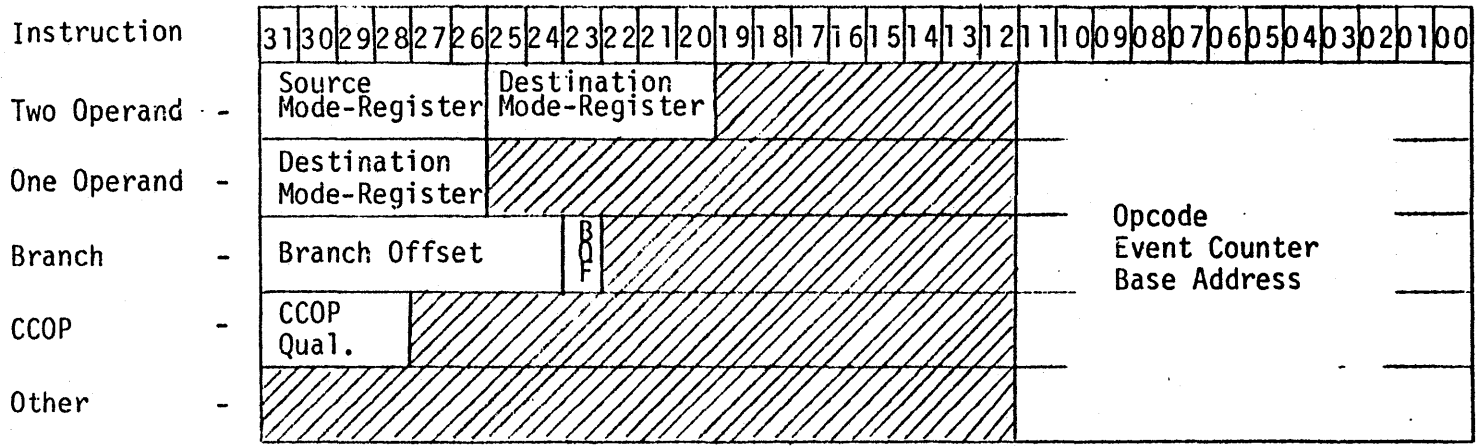
Basic Emulator Structure

Figure 2-1



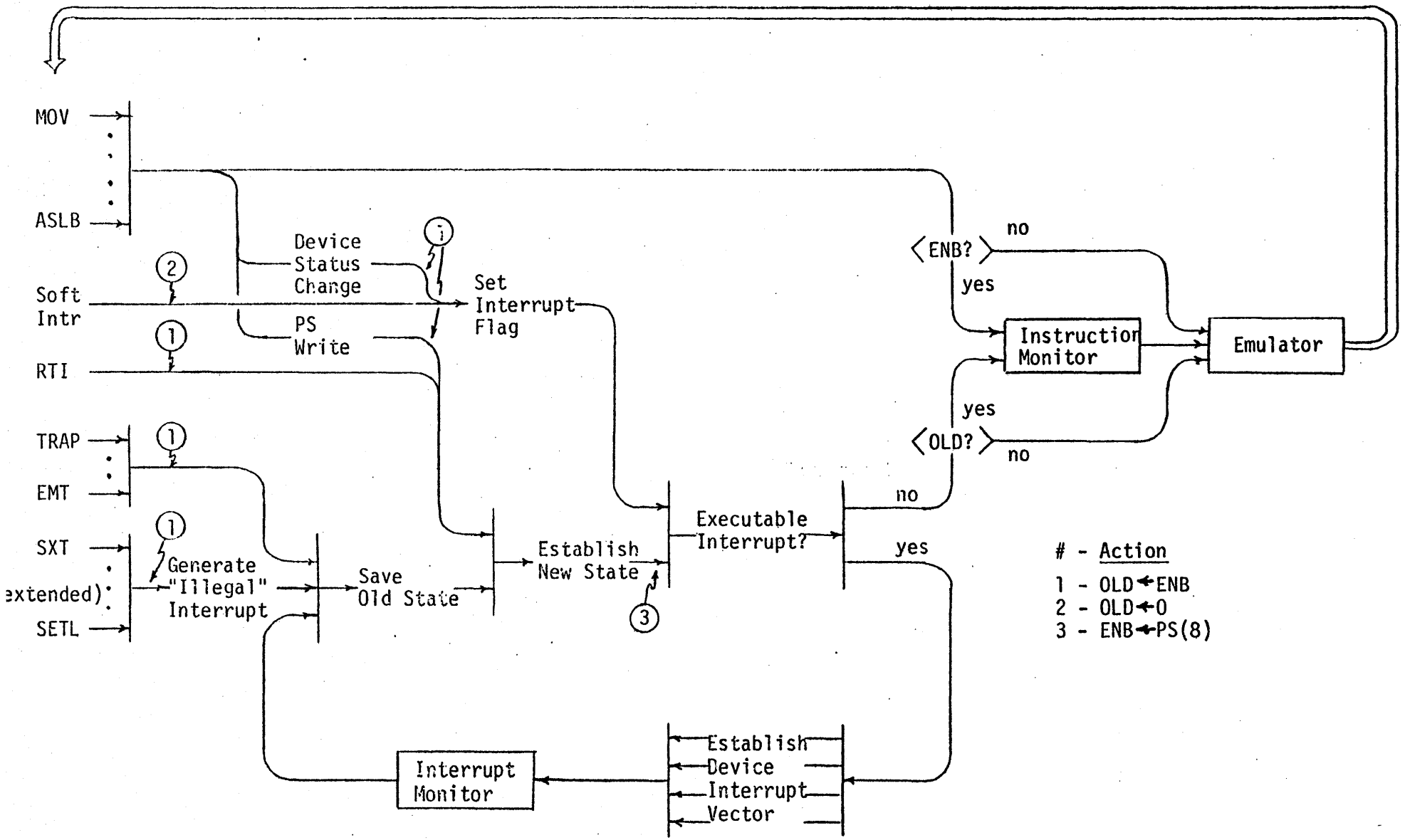
Structure of Code Emulator with Monitor

Figure 2-2



Information Register Format

Figure 2-3



Interrupt Resolution Logic

Figure 2-4

3. Description of Raw Monitor Data

At the termination of a monitored program run the Emmy control store (and possibly main store) contain counters which have recorded the occurrence of various events. In this section the semantics of the event counters are discussed. Event counters are grouped as follows:

Counter Type	Number	Size	Total
Opcode	97	1	97
Operand mode/register	78	24	1872
CCOP qualifier	2	16	32
Branch offset	256	1	256
Branch outcome	16	4	64
			2321

3.1 Opcode Counters

There are 97 event counters used to tabulate the opcodes encountered in the instruction stream. Usually there is an event counter for each unique PDP-11 opcode, but there are a few exceptions. First, double and single precision floating point operations are not differentiated. Thus, for example, ADDF and ADDD are counted together. Second, WAIT, HALT and RESET are counted as one instruction, called miscellaneous or MISC. Any unexpected event such as an I/O error or non-emulated illegal instruction code is counted as MISC also. A WAIT instruction is counted only when first encountered so that the actual time spent waiting is not visible to the monitor. All CCOP instructions are counted by one of two counters: CCLR or CSET.

Interrupts to the emulator (i.e. "soft" interrupts which cause breaks in the PDP-11 instruction stream) are counted as instructions by the INTR event counter.

3.2 Operand Mode/Register Counters

Many PDP-11 instructions, such as MOV, ADD, and DEC, specify a mode and register to be used in operand formation. Mode/register information is specified by a six bit field (i.e. three mode bits and three register bits). For instructions which operate explictly on two operands there are two mode/register fields: one for the source and one for the destination.

For each instruction which has mode/register fields the monitor tabulates information pertaining to the field usage. Where an instruction specifies both source and destination fields information is tabulated independently for each field. Because a mode/register

field is six bits wide, 64 mode/register combinations are possible. Approximately 70 instructions specify at least one mode/register field. Tabulating all 64 possible mode/register combinations for each such instruction would require an excessive amount of control store. Therefore, register specifications are mapped into three counter groups:

GR - General Register - R0, R1, R2, R3 and R4
SP - Stack Pointers - R5 and R6
PC - Program Counter - R7

This mapping reflects the underlying UNIX register usage scheme in which registers R0 through R4 are user working registers (actually R0 and R1 are usually employed as temporaries) and R5 and R6 are used to reference the system stack which contains dynamically allocated data, subroutine arguments and generated temporaries. The net result of the mapping above is to reduce the mode/register information for each instruction operand to 24 counters (i.e. three register groupts times eight modes). Preliminary examination of UNIX monitor output shows that R5 and R6 are used in much the same way and that grouping these two register accesses together is reasonable.

3.3 CCOP Qualifier Counters

CCOP instructions are used to set or clear selected bits of the PS condition codes. Although there are many CCOP mnemonic codes (e.g. CLV, SEN, NOP etc.) there are actually only two CCOP instructions when the low four bit field (bits 03-00) of the instruction is considered as a modifier describing the condition code bits to be manipulated. The two general CCOP codes are CSET (set selected bits) and CCLR (clear selected bits). For each of these instruction codes one of sixteen qualifier fields may be specified (i.e. any combination of the four condition code bits may be manipulated). Qualifier field usage is tabulated separately for CCLR and CSET, thus consuming a total of 32 counters.

3.4 Branch Qualifier (Offset) Counters

Branch instructions (conditional or unconditional) use an eight bit field to specify the direction and distance of the branch target relative to the current PC. If a branch is successful (i.e. the PC is to be modified by the offset field) the monitor will increment a counter corresponding to the offset field. There are 256 branch qualifiers which are used to tabulate the offset fields for all branch instructions including SOB.

3.5 Branch Outcome Counters

For each branch instruction (BR, conditional branches and SOB) four event counters are maintained relating to branch direction and outcome:

Counter	Direction	Outcome
0	Forward	Ignored
1	Forward	Taken
2	Backward	Ignored
3	Backward	Taken

The branch direction is determined from the high bit of the branch offset field (zero indicates a forward branch). There are 16 groups of four counters each used in tabulating branch outcome information for a total of 64 counters.

4. Output of Analysis Program

The analysis program described below operates on the raw monitor data collected from one experimental program run. This program serves two purposes: first, it organizes the raw input data and presents it in a readable form. Second, it combines specific information about the instruction syntax and semantics of the PDP-11 with the raw data to evaluate simple aspects of the PDP-11 architecture. For example, the analysis program determines the average length of a PDP-11 instruction from opcode counts and instruction syntax.

In the sections below the output of the analysis program is explained and factors considered in generating this output are given. The appendix gives the full output from a single analysis run. Page numbers given with each section header refer to page numbers of the appendix. Various tables following the current main section give the syntactic and semantic information assumed about t e PDP-11 instruction set.

4.1 Opcode Usage Summary (page 1-1 and 1-2)

There are currently 97 opcode groups tabulated by the monitor. As shown in table 2-1 some groups contain more than one PDP-11 opcode mnemonic. The opcode usage summary gives the number of usages and percent usage for each of the 97 opcodes. Opcodes are sorted by count and presented in descending order. To augment the presentation percent usage is shown as a histogram with a scale of 0% to 30%.

All PDP-11 opcodes have been placed in one of three groups which reflect their primary usage [7]:

1) Functional - Makes an explicit transformation of data
2) Memory - Moves data or clears a storage cell
3) Procedural - Cause a potential break in the instruction stream or modifies processor state.

Obviously, this classification is somewhat subjective. In classifying an instruction only the opcode is considered; operand dependent semantics are ignored. For example, an ADD can be used in place of a TST instruction if the source operand is set to zero, but the ADD is still counted as a functional instruction. CLR and CLRB are counted as a "memory" type instructions since their function is to move an implicit zero to a storage cell.

Several ratios between class counts are tabulated:

$$\text{Memory ratio} = \frac{\#\ \text{Memory Instructions}}{\#\ \text{Functional Instructions}}$$

$$\text{Procedural ratio} = \frac{\#\text{ Procedural Instructions}}{\#\text{ Functional Instructions}}$$

$$\text{Non-Functional ratio} = \frac{\#\text{ Memory and Procedural Instructions}}{\#\text{ Functional Instructions}}$$

These ratios provide a simple evaluation of architectural effectiveness in that larger ratios imply excessive consumption of machine resources by instructions which do not transform data, the primary goal of computation. "I eal" architectures have been proposed which minimize or eliminate the usage of explicit Memory and Procedural type instructions [8].

4.2 Instruction Breakdown and Opcode Size (page 1-3)

By considering both opcode and operand usage counts it is possible to estimate the length of an average PDP-11 instruction and also the average allocation of bits within an instruction which specify the various instruction related functions. We consider each PDP-11 instruction (excluding interrupts) to be composed of two parts: a 16 bit required "base" and optional "extension" fields of one or two 16 bit words. The instruction base may specify several pieces of information:

1) Opcode - the type and form of the operation
2) Operand - the method of forming instruction operands
3) Qualifier - either a branch offset or CCOP code mask

Table 1 summarizes the allocation of bits to these functions for every tabulated opcode. Average bit usage for each component is calculated from the opcode usage data and this syntactic information.

In many cases a PDP-11 instruction which requires an operand will use extension fields to either specify the operand or its address. Extension field usage is a function of the mode/register specification a follows:

Extension Usage	Mode	Registers
Index	6	R0 - R6
Index Deferred	7	R0 - R6
Immediate	2	R7
Absolute	3	R7
Relative	6	R7
Relative Deferred	7	R7

The opcode size summary tabulates the number of instructions encountered which have a given opcode size. Interrupts (INTR) are

included and assigned an opcode size of zero. In terms of information theory, an ideally constructed machine would have its shortest opcodes most frequently used [9]. The opcode size summary gives a subjective estimate of the coding efficiency of the PDP-11 instruction set.

4.3 Register and Memory Usage Summary (page 1-4)

Data presentation in the "Register and Memory Usage" summary is a assessment of the accessing burden placed on the register and memory resources by the program being measured. Our approach is similar to that used in the CFA study [10], except that in our analysis only explicitly specified accesses are counted rather than all register accesses required by the hardware to perform an instruction. For example, we do not count the register accesses required to perform auto-increment/decrement while the CFA approach does.

Figure 4-1 illustrates in a schematic manner the relationship between the operator and operands specified by an instruction and the storage resources of the processor for a two operand instructions such as ADD. In forming an operand read accesses to storage resources will be required. These accesses may be data for use by the operator or addresses to be used in accessing the actual data (i.e. indirecting). After the specified operation is complete the result is written to storage resources under control of the generated destination address. There are, of course, a wide range of operators in the PDP-11. Some use only one source operand (e.g. INC), others have no destination operand write (e.g. CMP) and so forth. In computing the accesses to memory and registers we must consider both the operator and the actual operand mode/register specification together.

4.3.1 Operand Specified Accesses

Below we define the access classes used in the storage usage summary. This is done primarily on the basis of operand specification field of the instruction although for some instructions (e.g. EMT, TRAP etc.) certain storage accesses are implied. Except for "Instruction" and "Displacement" accesses, both memory and register accesses are defined for each class.

Instruction - One memory access for each opcode encountered.

Displacement - One memory read access for every mode 6 or mode 7 operand specification encountered.

Data Read - The word read from a cell is used directly as data by an operator. Specifically:

Register Data Read - Mode 0 only

Memory Data Read - Modes 1 through 7

Data Write - The result of an operation is written to a cell. Modes are counted as in "Data Read" above.

Address - Data read from a cell is used as an address in operand formation. Specifically:

Register Address Read - Modes 1 through 7
Memory Address Read - Modes 3, 5 or 7

Misc. Read - A cell is read implicitly because of the opcode specification.

Misc. Write - A cell is written implicitly because of the opcode specification.

The category "All Reads" includes read accesses related to Instruction, Displacement, Data Read, Address and Miscellaneous Read accesses. The remaining two catagories, Data Write and Miscellaneous Write comprise the "All Write" catagory.

4.3.2 Opcode Specified Accesses

In the above section we specified how the operand specification in the mode/register field of an instruction should be interpreted in counting potential storage accesses. This interpretation must be further modified based on the operator part (i.e. opcode) of the instruction. For example, for a CMP instruction no result is written so accesses specified by the destination field should be counted as reads only. All PDP-11 instructions have been placed into one of 26 catagories related to their "form", that is the way in which the instruction uses its operands. Form numbers are shown in Table 2.

In Table 1 instruction forms are related to the access catagories. There are two sections to this table: operand specified accesses and opcode specified accesses. Accesses marked in the operand specified section will be counted only if the operand specification so designates. For example, if an "Address" access is marked an access will be tabulated only if an operand field of the instruction indicates an address access (e.g. register address access if the mode is 1 through 7; memory address access if the mode is 3, 5 or 7).

Opcode specified accesses are counted independently of the operand specification since they are implicit with respect to the opcode used. Usually opcode specified acceses relate to state change instructions, such as EMT, TRAP, RTI, or to extended and floating

point instructions where one operand is always a register.

4.3.3 Limitations in the Usage Summary

It was not possible to make access tabulation precise due to limitations in the emulator/monitor micro-code and in the size of the analysis program. The essential limitations to our method are:

1) Accesses by byte oriented istructions are counted as full word accesses.

2) All floating point accesses are counted as one word regardless of the size (i.e. floating or double).

3) Double register accesses related to certain extended operations are counted as single word accesses.

In general, we have measured the access burden on storage resources due to the opcode/operand specification of the instruction independently of the data type (e.g. byte, word, floating or double).

4.4 Operand Usage Data (pages 1-5 through 1-14)

Raw data giving operand usage for each instruction which specifies a mode/register field is presented by the "Operand Usage" summary in a readable form. Where an instruction, such as MOV, has two operands (source and destination) the usage of each operand is tabulated independently. For a given operand, usage with respect to mode and register is given as a percent of the total occurrence of the instruction. Marginal percentages are calculated by mode and also by register group. Register group data reflects that of the raw data in that all references to certain registers are considered in common:

Group	Includes
GR - General Registers	R0, R1, R2, R3 and R4
SP - Stack Pointers	R5 and R6
PC - Program Counter	PC

This grouping reflects the basic usage of registers in code generated by the "C" compiler and applies to nearly all user state code in UNIX.

Operand usages are presented only if an instruction actually occurs. The order in which the statistics appear results from the internal organization of the opcode counters.

4.5 Operand Summary For Selected Instructions (page 1-15)

Operand usage information for frequently used instructions has been summarized in one place by this analysis. Ten instruction categories (word operand only) have been defined. Taken together these catagories comprise a majority (usually 75% or more) of the instructions encountered. The instructions in each catagory are as follows:

Catagory	Instructions	Operands
Move	MOV	2
Clear	CLR	1
Compare	CMP	2
Test	TST	1
Arith2	ADD,SUB	2
Arith1	INC,DEC,NEG,ADC,SBC	1
Logic2	BIS,BIC	2
Logic1	COM,ROL,ROR,ASL,ASR,SWAB	1
Jump	JMP	1
Call	JSR	1

Percentages are given on a per operand basis and are exactly the same as those presented in the operand usage summary.

4.6 Register Usage Cross Comparison (page 1-16)

In the operand usage summary described above marginal percentages for register usage were calculated. A simple graphical method has been used to present this data on a per operand basis. The percent usage for general registers (GR) and stack pointers (SP) is presented on an X-Y plot where the X-axis represents stack pointer usage and the Y-axis represents general register usage. By implication 45 degree lines from upper left to bottom right may be drawn to indicate PC usage. The lower left corner of the graph would then represent 100% PC related usage. Marginal values for register usage are calculated for all modes for which the register is specified so the location of the actual operand is not known precisely (e.g. the register might have been used indirectly).

A data point has been placed on the plot for each operand (source or destination) used. A different symbol has been used to represent each combination of usage (source and destination) and instruction form (one or two operand). Instruction names associated with a given symbol are printed to the right of the plot in an order corresponding to SP percent usage. Unfortunately, the line printer used to produce the output quantizes the symbol position so that some information is lost. The coordinates of a given point can be determined accurately, if desired, directly from the operand usage data summary.

4.7 CCOP Qualifier Analysis (page 1-17)

For each of the two CCOP instructions (CSET and CCLR) the usage of the qualifier is given. Percentages are calculated on the basis of total CCOP usage (i.e. the sum of CSET and CCLR usage). Marginal sums are given for usage by instruction and by code mask.

4.8 Summary of Branch Instructions (page 1-18)

In this summary all 14 conditional branches (except SOB) are considered as one group. PDP-11 conditional branches are symetric, that is each branch for the occurrence of a given condition has a corresponding branch for its non-occurrence. In this presentation branches have been paired on the basis of condition codes tested. The column headed "0" corresponds to the left most conditional branch named and to a test outcome of "0" or false. Marginal percentages have been computed on the basis of test outcome and conditions tested.

Branches have been counted on the basis of their occurrence in the I-stream without consideration given to their actual outcome.

4.9 Summary of I-stream Breaks (page 1-18)

All instructions (including interrupts) which may cause an explicit break in the PDP-11 instruction stream are tabulated in this summary. Instructions in which the ocodes do not specify a break but in which operand usage does are not considered. An example of such an instruction would be a move immediate to register R7. In the data presentation the occurrence of each potential break instruction is shown as a percentage of all instructions with break potential. Conditional and SOB instructions may or may not cause a break depending on their ourcome. Actual breaks to the instruction stream are calculted by considering the outcome of these instructions.

The average length of an instruction run between breaks in the I-stream is calculated by dividing the total number of instructions executed by the number of instructions which are potential (or actual) breaks.

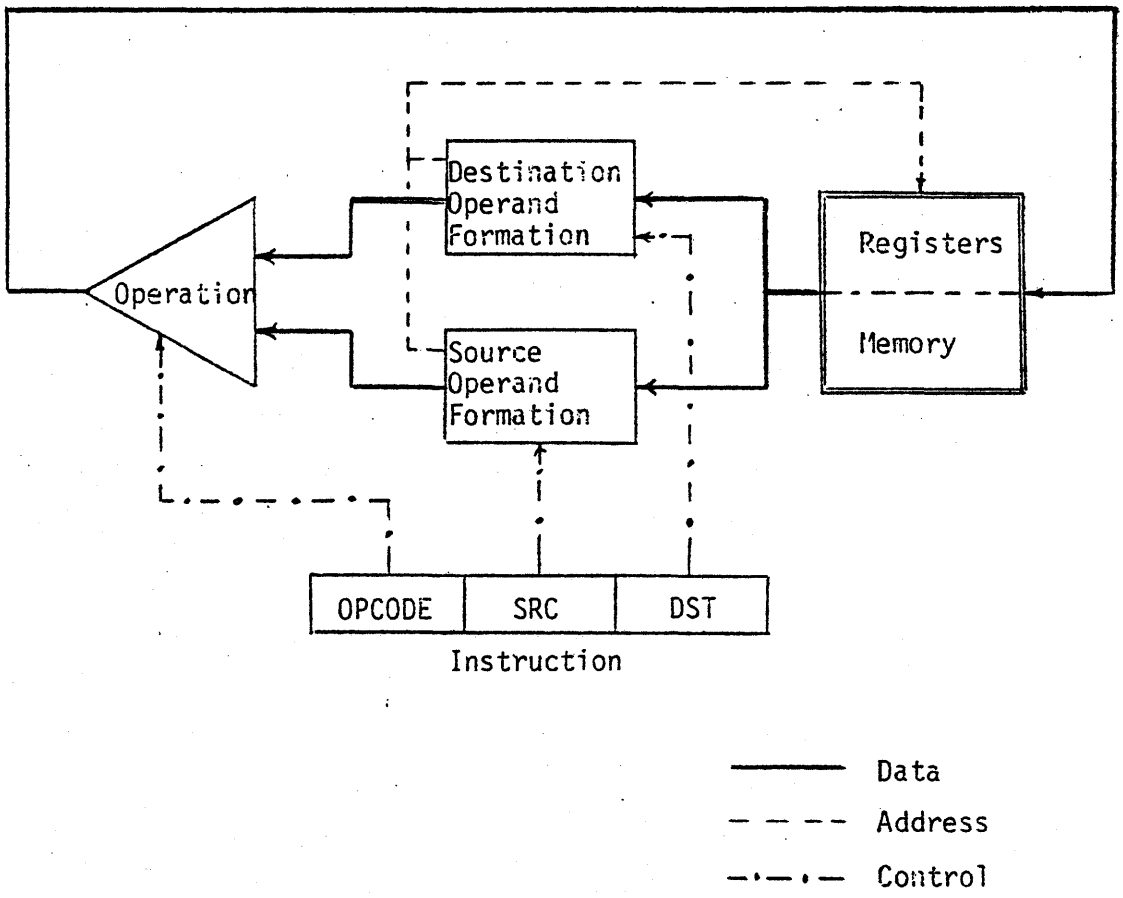
Note: a small anomaly results from the monitor control mechanism in that illegal instructions in the 11/05 implementation (such as ASHC) are counted twice; once at the actual occurrence and again as an illegal instruction interrupt (INTR). Furthermore, when the monitor is setup to ignore emulation of these instructions the illegal interrupt will be counted on entry but the associated RTI will not be counted on exit. This artifact of monitoring will make the usage of INTR and RTI appear unbalanced.

4.10 Branch Qualifier Statistics (page 1-19)

For each branch (conditional, unconditional or SOB) which is taken an event counter associated with the branch offset value (one out of 256) is incremented. In presenting this data branch offsets (in words) are grouped logrithmicly, for example, 2-3, 4-7, 8-15 and so forth. Percentages are calculated on the basis of all branches taken and presented graphically as a histogram.

4.11 Conditional Branch Outcomes (page 1-20)

All branch instructions (conditional, unconditional and SOB) are examined by the monitor which tabulates branch outcome and direction. This information is presented on a per instruction basis. Percentages are calculated on the basis of all branch type instructions encountered, whether they were taken or not. Conditional branches have been presented in pairs reflecting their use of particular condition code bits. For each branch outcome (taken or ignored) the percentages associated with branch direction have been summed. Further, the outcome percentages have been summed (i.e. the column marked "TOTAL") to give a distribution of branch types for all branches encountered during emulation.



Operator and Operand Storage Accessing

Figure 4-1

5. Example of a Monitored Program

The appendix to this report gives the complete analysis of a single PDP-11 program run, the compilation of a 500 line "C" program. The complete program consisted of about 8 million instructions and consumed about 85 seconds of real time on the Emmy CPU. On the actual PDP-11 this program consumed about 520 seconds. The real time clock in the emulated PDP-11 has been slowed down by a factor of six so that clock related system activity consumes about the same fraction of system resources as it would on the actual PDP-11. To UNIX on the emulated system real time appears to pass at one-sixth the rate of actual real time. With this modification the "time" reports from both PDP-11 systems when running the example are comparable:

	Actual	Emulated
User	1:02	1:16
System	:25	:18
Real Time	1:27	1:34

6. References

[1] C. Neuhauser, "Emmy System Processor -- Principles of Operation", Technical Note No. 114, Computer Systems Laboratory, Stanford University, May 1977.

[2] C. Neuhauser, "Emmy Peripherals -- Principles of Operation", Technical Note No. 77, Computer Systems Laboratory, Stanford University, December 1975.

[3] C. Neuhauser, "An Emmy Based PDP-11/20 Emulator", Technical Note No. 110, Computer Systems Laboratory, March 1977.

[4] J. Huck, "A Virtual I/O System for the Stanford Emmy - VACCESS", Technical Note No. 144, Computer Systems Laboratory, Stanford University, May 1979.

[5] G. Rose, "Performance Evaluation Under UNIX and a Study of PDP-11 Instruction Usage", Operating Systems Review, Vol. 12, No.3, July 1978.

[6] PDP-11 04/05/10/35/40/45 Processor Handbook, Digital Equipment Corporation, 1975.

[7] M. Flynn, "Trends and Problems in Computer Architecture", Proceedings of IFIP Congress 1974, North Holland Publishing Co.

[8] L. Hoevel and M. Flynn, "The Structure of Directly Executed Languages: A New Theory of Interpretive System Design", Technical Report No. 130, Computer Systems Laboratory, Stanford University, March 1977.

[9] C. Foster, R. Gonter and E. Riseman, "Measures of Op-Code Utilization", IEEE Transactions on Computers, May 1971.

[10] S. Fuller and W. Burr, "Measurement and Evaluation of Alternative Computer Architectures", Computer, Vol. 10, No.10, October 1977.

Table 1 - Instruction Characteristics

Key: Oc - Size of opcode in bits
Op - Size of operand fields in bits
Qa - Size of qualifier fields in bits
CN - Counter number

Name	Type	Oc	Op	Qa	Form	CN	Includes
MISC	P	16	0	0	1	0	WAIT, HALT, RESET
MOV	M	4	12	0	5	1	---
CMP	P	4	12	0	7	2	---
BIT	P	4	12	0	7	3	---
BIC	F	4	12	0	6	4	---
BIS	F	4	12	0	6	5	---
ADD	F	4	12	0	6	6	---
SUB	F	4	12	0	6	7	---
JMP	P	10	6	0	8	8	---
SWAB	F	10	6	0	2	9	---
JSR	P	7	9	0	9	10	---
CLR	M	10	6	0	4	11	---
COM	F	10	6	0	2	12	---
INC	F	10	6	0	2	13	---
DEC	F	10	6	0	2	14	---
NEG	F	10	6	0	2	15	---
ADC	F	10	6	0	2	16	---
SBC	F	10	6	0	2	17	---
TST	F	10	6	0	3	18	---
ROR	F	10	6	0	2	19	---
ROL	F	10	6	0	2	20	---
ASR	F	10	6	0	2	21	---
ASL	F	10	6	0	2	22	---
MOVB	F	4	12	0	5	23	---
CMPB	P	4	12	0	7	24	---
BITB	P	4	12	0	7	25	---
BICB	F	4	12	0	6	26	---
BISB	F	4	12	0	6	27	---
CLRB	M	10	6	0	4	28	---
COMB	F	10	6	0	2	29	---
INCB	F	10	6	0	2	30	---
DECB	F	10	6	0	2	31	---

Table 1 - continued

Name	Type	Oc	Op	Qa	Form	CN	Includes
NEGB	F	10	6	0	2	32	---
ADCB	F	10	6	0	2	33	---
SBCB	F	10	6	0	2	34	---
TSTB	P	10	6	0	3	35	---
RORB	F	10	6	0	2	36	---
ROLB	F	10	6	0	2	37	---
ASRB	F	10	6	0	2	38	---
ASLB	F	10	6	0	2	39	---
RTI	P	16	0	0	13	40	---
BPT	P	16	0	0	10	41	---
IOT	P	16	0	0	10	42	---
INTR	P	0	0	0	12	43	Any interrupt
EMT	P	16	0	0	10	44	---
TRAP	P	16	0	0	10	45	---
RTS	P	13	3	0	11	46	---
CCLR	P	12	0	4	14	47	CLN, CLZ, CLV, CLC
CSET	P	12	0	4	14	48	SLN, SLZ, SLV, SLC
BR	P	8	0	8	15	49	---
BNE	P	8	0	8	15	50	---
BEQ	P	8	0	8	15	51	---
BGE	P	8	0	8	15	52	---
BLT	P	8	0	8	15	53	---
BGT	P	8	0	8	15	54	---
BLE	P	8	0	8	15	55	---
BPL	P	8	0	8	15	56	---
BMI	P	8	0	8	15	57	---
BHI	P	8	0	8	15	58	---
BLOS	P	8	0	8	15	59	---
BVC	P	8	0	8	15	60	---
BVS	P	8	0	8	15	61	---
BHIS	P	8	0	8	15	62	---
BLO	P	8	0	8	15	63	---
SXT	F	10	6	0	2	64	---
MUL	F	10	6	0	16	65	---
DIV	F	10	6	0	16	66	---
ASH	F	10	6	0	16	67	---
ASHC	F	10	6	0	17	68	---
XOR	F	10	6	0	16	69	---
SOB	P	7	3	6	18	70	---
FMUL	F	8	8	0	25	71	MULF, MULD

Table 1 - continued

Name	Type	Oc	Op	Qa	Form	CN	Includes
FMOD	F	8	8	0	25	72	MODF, MODD
FADD	F	8	8	0	25	73	ADDF, ADDD
FLD	M	8	8	0	23	74	LDF, LDD
FSUB	F	8	8	0	24	75	SUBF, SUBD
FCMP	P	8	8	0	25	76	CMPF, CMPD
FST	M	8	8	0	24	77	STF, STD
FDIV	F	8	8	0	25	78	DIVF, DIVD
STEX	M	8	8	0	24	79	STEXP
STCC	M	8	8	0	24	80	STCFI, STCFL, STCDI, STCDL
STCF	M	8	8	0	24	81	STCFD, STCDF
LDEX	M	8	8	0	23	82	LDEXP
LDCC	M	8	8	0	23	83	LDCIF, LDCLF, LDCID, LDCLD
LDCF	M	8	8	0	23	84	LDCDF, LDCFD
LDFP	P	8	8	0	23	85	LDFPS
STFP	P	8	8	0	23	86	STFPS
STST	P	8	8	0	24	87	---
FCLR	M	10	6	0	22	88	CLRF, CLRD
FTST	P	10	6	0	21	89	TSTF, TSTD
FABS	F	10	6	0	20	90	ABSF, ABSD
FNEG	F	10	6	0	20	91	NEGF, NEGD
CFCC	P	16	0	0	19	92	---
SETF	P	16	0	0	19	93	---
SETI	P	16	0	0	19	94	---
SETD	P	16	0	0	19	95	---
SETL	P	16	0	0	19	96	---

Table 2 - Instruction Form Definitions

Operand Specified:

RDR - Register Data Read
RDW - Register Data Write

RMR - Register Misc. Read
RMW - Register Misc. Write

MMR - Memory Misc. Read
MMW - Memory Misc. Write

Opcode Specified:

SR - Source Read
SA - Source Address

DR - Destination Read
DW - Destination Write
DA - Destination Address

Form	Examples	RDR	RDW	RMR	RMW	MMR	MMW	SSR	SSA	DDR	DDW	DDA
1	MISC	-	-	-	-	-	-	0	0	0	0	0
2	INC, DEC	-	-	-	-	-	-	0	0	1	1	1
3	TST	-	-	-	-	-	-	0	0	1	0	1
4	CLR	-	-	-	-	-	-	0	0	0	1	1
5	MOV	-	-	-	-	-	-	1	1	0	1	1
6	BIC, ADD	-	-	-	-	-	-	1	1	1	1	1
7	CMP, BIT	-	-	-	-	-	-	1	1	1	0	1
8	JMP	-	-	-	-	-	-	0	0	0	0	1
9	JSR	-	-	1	1	0	1	0	0	0	0	1
10	EMT, BPT	-	-	0	0	2	2	-	-	-	-	-
11	RTS	-	-	1	1	1	0	-	-	-	-	-
12	INTR	-	-	0	0	2	2	-	-	-	-	-
13	RTI	-	-	0	0	2	0	-	-	-	-	-
14	CCLR, CSET	-	-	-	-	-	-	-	-	-	-	-
15	BR, BNE	-	-	-	-	-	-	-	-	-	-	-
16	MUL, DIV	1	1	-	-	-	-	1	1	0	0	0
17	ASHC	2	2	-	-	-	-	1	1	0	0	0
18	SOB	1	1	-	-	-	-	0	0	0	0	0
19	SETF	-	-	-	-	-	-	0	0	0	0	0
20	FABS, FNEG	-	-	-	-	-	-	0	0	1	1	1
21	FTST	-	-	-	-	-	-	1	1	0	0	0
22	FCLR	-	-	-	-	-	-	0	0	0	1	1
23	LDEX, LDCF	0	1	-	-	-	-	1	1	0	0	0
24	STST, STEX	1	0	-	-	-	-	0	0	0	1	1
25	FADD, FMUL	1	1	-	-	-	-	1	1	0	0	0
26	FCMP	1	0	-	-	-	-	1	1	0	0	0

OPCODE FREQUENCY SUMMARY

NAME	COUNT	PCT	CUMM	0% 5% 10% 15% 20% 25% 30%
MOV	2482286	32.55	32.55	IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
CMP	688342	9.03	41.57	IPPPPPPPPPPPPPPPPPPP
BNE	539653	7.08	48.65	IPPPPPPPPPPPPPPP
JSR	389146	5.10	53.75	IPPPPPPPPPPP
BEQ	368300	4.83	58.58	IPPPPPPPPPP
TST	331729	4.35	62.93	IPPPPPPPPP
DEC	311561	4.09	67.02	IFFFFFFFFF
CLR	264353	3.47	70.48	IMMMMMMM
RTS	260960	3.42	73.90	IPPPPPPP
BR	253089	3.32	77.22	IPPPPPPP
JMP	251612	3.30	80.52	IPPPPPPP
ADD	194727	2.55	83.08	IFFFFFF
MOVB	179998	2.36	85.44	IMMMMM
BGT	124757	1.64	87.07	IPPPP
BLOS	124315	1.63	88.70	IPPPP
INC	116297	1.52	90.23	IFFFF
BGE	93684	1.23	91.45	IPPP
BIC	62584	0.82	92.28	IFF
BLT	59208	0.78	93.05	IPP
TSTB	54198	0.71	93.76	IPP
SUB	53594	0.70	94.46	IFF
ASL	53516	0.70	95.17	IFF
BHI	50933	0.67	95.83	IPP
BLO	38589	0.51	96.34	IPP
BIT	37399	0.49	96.83	IP
BIS	30872	0.40	97.24	IF
CMPB	29330	0.38	97.62	IP
BLE	26850	0.35	97.97	IP
BISB	20735	0.27	98.24	IF
SOB	16225	0.21	98.46	IP
DIV	15742	0.21	98.66	IF
BHIS	15136	0.20	98.86	IP
ASR	14030	0.18	99.05	IF
CLRB	10966	0.14	99.19	IM
XOR	7749	0.10	99.29	IF
RTI	7220	0.09	99.39	IP
SWAB	6476	0.08	99.47	IF
INTR	6460	0.08	99.56	IP
ASH	6169	0.08	99.64	IF
MUL	4349	0.06	99.69	IF
BITB	4332	0.06	99.75	IP
NEG	3283	0.04	99.79	IF
ASHC	2962	0.04	99.83	IF
CCLR	2135	0.03	99.86	IP
BPL	2016	0.03	99.89	IP
ADC	1856	0.02	99.91	IF
SXT	1531	0.02	99.93	IF
INCB	1123	0.01	99.95	IF
MISC	896	0.01	99.96	IP
ROL	824	0.01	99.97	IF
TRAP	759	0.01	99.98	IP
SBC	600	0.01	99.99	IF

OPCODE FREQUENCY SUMMARY

NAME	COUNT	PCT	CUMM	0% 5% 10% 15% 20% 25% 30%
BICB	410	0.01	99.99	IF
BVS	335	0.00	100.00	IP
DECB	244	0.00	100.00	IF
BMI	67	0.00	100.00	IP
COM	15	0.00	100.00	IF
CSET	3	0.00	100.00	IP
COMB	0	0.	100.00	I
NEGB	0	0.	100.00	I
ADCB	0	0.	100.00	I
SBCB	0	0.	100.00	I
RORB	0	0.	100.00	I
ROLB	0	0.	100.00	I
ASRB	0	0.	100.00	I
ASLB	0	0.	100.00	I
BPT	0	0.	100.00	I
IOT	0	0.	100.00	I
EMT	0	0.	100.00	I
BVC	0	0.	100.00	I
XOR	0	0.	100.00	I
FMUL	0	0.	100.00	I
FMOD	0	0.	100.00	I
FADD	0	0.	100.00	I
FLD	0	0.	100.00	I
FSUB	0	0.	100.00	I
FCMP	0	0.	100.00	I
FST	0	0.	100.00	I
FDIV	0	0.	100.00	I
STEX	0	0.	100.00	I
STCC	0	0.	100.00	I
STCF	0	0.	100.00	I
LDEX	0	0.	100.00	I
LDCC	0	0.	100.00	I
LDCF	0	0.	100.00	I
LDFP	0	0.	100.00	I
STFP	0	0.	100.00	I
STST	0	0.	100.00	I
FCLR	0	0.	100.00	I
FTST	0	0.	100.00	I
FABS	0	0.	100.00	I
FNEG	0	0.	100.00	I
CFCC	0	0.	100.00	I
SETF	0	0.	100.00	I
SETI	0	0.	100.00	I
SETD	0	0.	100.00	I
SETL	0	0.	100.00	I

FUNCTIONAL	TOTAL=	911249	% FUNCTIONAL=	11.9	
MEMORY	TOTAL=	2937603	% MEMORY =	38.5	M-RATIO= 3.22
PROCEDURAL	TOTAL=	3777678	% PROCEDURAL=	49.5	P-RATIO= 4.15
GRAND	TOTAL=	7626530	PERCENT =	100.0	NF-RATIO= 7.37

INSTRUCTION BREAKDOWN (NOT INCLUDING INTERRUPTS)

	COMPONENT	BITS
BASE	OPCODE	6.51
	OPERAND	7.69
	QUALIFIER	1.80
		16.00
EXTENSION	INDEX	1.43
	INDEX DEFERRED	0.24
	IMMEDIATE	1.79
	ABSOLUTE	0.32
	RELATIVE	1.88
	RELATIVE DEF	0.16
		5.82

AVERAGE INSTRUCTION LENGTH = 21.82 BITS

OPCODE SIZE SUMMARY (INCLUDING INTERRUPTS)

BITS	NUMBER	PERCENT
0	6460	0.08
4	3784609	49.62
7	434593	5.70
8	1696932	22.25
10	1431963	18.78
12	2138	0.03
13	260960	3.42
16	8675	0.12
	7626530	100.00

REGISTER AND MEMORY USAGE (INCLUDING INTERRUPTS)

ACCESS	REGISTER NUMBER	REGISTER PER INST	MEMORY NUMBER	MEMORY PER INST
INSTRUCTION			7626530	1.000
DISPLACEMENT			1766650	0.232
DATA READ	2545185	0.334	3344412	0.439
DATA WRITE	2283027	0.299	1585012	0.208
ADDRESS	5233237	0.686	353575	0.046
MISC READ	650106	0.085	289838	0.038
MISC WRITE	650106	0.085	403584	0.053
ALL READS	8428528	1.105	13381005	1.755
ALL WRITES	2933133	0.385	1988596	0.261
TOTAL	11361661	1.490	15369601	2.015

DATA READ/WRITE RATIOS

		WRITES REG	WRITES MEM
READ	REG	1.11	1.61
	MEM	1.46	2.11

OPERAND USAGE SUMMARY

MOV - SRC	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	28.11	1.00	7.49	0.	12.33	0.	2.73	0.15	!	51.81
SP	20.89	0.03	8.57	0.	0.	0.	7.53	0.22	!	37.24
PC	0.00	0.	5.29	0.14	0.	0.	4.77	0.75	!	10.95
	48.99	1.03	21.35	0.14	12.33	0.	15.03	1.12		100.00
MOV - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	45.33	0.14	6.79	0.44	0.14	0.	1.27	0.	!	54.11
SP	16.49	4.62	0.08	0.	19.44	0.	1.53	0.06	!	42.22
PC	0.	0.	0.	0.22	0.	0.	2.96	0.49	!	3.68
	61.82	4.75	6.88	0.66	19.58	0.	5.76	0.55		100.00
CMP - SRC	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	39.36	0.41	12.35	0.	0.01	0.	1.28	0.	!	53.40
SP	0.02	0.89	3.23	0.	0.	0.	3.41	0.	!	7.55
PC	0.	0.43	36.75	0.	0.	0.	1.87	0.	!	39.05
	39.38	1.73	52.32	0.	0.01	0.	6.56	0.		100.00
CMP - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	49.76	3.42	10.56	0.	5.10	0.	4.72	0.16	!	73.71
SP	0.20	0.80	3.77	0.	0.	0.	3.65	0.32	!	8.74
PC	0.	0.	15.39	0.06	0.	0.	2.10	0.	!	17.55
	49.96	4.22	29.72	0.06	5.10	0.	10.47	0.48		100.00
BIT - SRC	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.02	0.	0.	0.	0.	0.	0.	0.	!	0.02
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	0.	99.98	0.	0.	0.	0.	0.	!	99.98
	0.02	0.	99.98	0.	0.	0.	0.	0.		100.00
BIT - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	29.49	24.27	0.	0.	0.	0.	20.73	0.	!	74.49
SP	0.	0.	0.	0.	0.	0.	12.25	0.	!	12.25
PC	0.	0.	0.	0.	0.	0.	13.27	0.	!	13.27
	29.49	24.27	0.	0.	0.	0.	46.24	0.		100.00

OPERAND USAGE SUMMARY

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
BIC	- SRC										
	GR	0.01	0.	0.	0.	0.	0.	2.46	0.	!	2.47
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	97.53	0.	0.	0.	0.	0.	!	97.53
		0.01	0.	97.53	0.	0.	0.	2.46	0.		100.00
BIC	- DST										
	GR	69.76	3.88	0.	0.	0.	0.	0.04	0.	!	73.68
	SP	0.	8.61	0.	0.	0.	0.	3.06	0.	!	11.66
	PC	0.	0.	0.	0.	0.	0.	14.65	0.	!	14.65
		69.76	12.48	0.	0.	0.	0.	17.75	0.		100.00
BIS	- SRC										
	GR	14.05	0.	0.	0.	0.	0.	0.	0.	!	14.05
	SP	0.	0.	17.76	0.	0.	0.	0.24	0.	!	18.00
	PC	0.	0.	65.70	0.	0.	0.	2.25	0.	!	67.95
		14.05	0.	83.46	0.	0.	0.	2.49	0.		100.00
BIS	- DST										
	GR	31.97	10.32	0.	0.	0.18	0.	0.03	0.	!	42.50
	SP	0.	12.73	0.	0.	0.	0.	2.51	0.32	!	15.57
	PC	0.	0.	0.	0.	0.	0.	41.93	0.	!	41.93
		31.97	23.05	0.	0.	0.18	0.	44.47	0.32		100.00
ADD	- SRC										
	GR	15.91	0.	0.	0.	0.	0.	0.16	0.	!	16.06
	SP	0.37	0.	0.78	0.	0.	0.	3.20	0.	!	4.36
	PC	0.	0.01	77.49	0.	0.	0.	2.08	0.	!	79.58
		16.28	0.01	78.27	0.	0.	0.	5.45	0.		100.00
ADD	- DST										
	GR	64.66	1.91	0.	0.	0.01	0.	0.59	0.	!	67.17
	SP	3.02	7.18	0.	0.	0.	0.	5.95	0.01	!	16.16
	PC	0.	0.	0.	0.	0.	0.	15.63	1.04	!	16.67
		67.68	9.09	0.	0.	0.01	0.	22.17	1.05		100.00

OPERAND USAGE SUMMARY

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
SUB - SRC	GR	14.10	0.10	0.	0.	0.	0.	0.00	0.	!	14.20
	SP	0.	0.	0.	0.	0.	0.	6.26	0.	!	6.26
	PC	0.	0.	76.97	0.	0.	0.	2.57	0.	!	79.54
		14.10	0.10	76.97	0.	0.	0.	8.83	0.		100.00
SUB - DST	GR	49.51	0.00	0.	0.	0.	0.	7.73	0.	!	57.24
	SP	26.54	3.25	0.	0.	0.	0.	2.30	0.	!	32.09
	PC	0.	0.	0.	0.	0.	0.	10.67	0.	!	10.67
		76.05	3.25	0.	0.	0.	0.	20.70	0.		100.00
JMP - DST	GR	0.	30.68	0.	0.01	0.	0.	0.	14.65	!	45.34
	SP	0.	0.	0.	0.	0.	0.	0.	1.52	!	1.52
	PC	0.	0.	0.	9.62	0.	0.	43.52	0.	!	53.14
		0.	30.68	0.	9.63	0.	0.	43.52	16.17		100.00
SWAB - DST	GR	100.00	0.	0.	0.	0.	0.	0.	0.	!	100.00
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		100.00	0.	0.	0.	0.	0.	0.	0.		100.00
JSR - DST	GR	0.	6.34	0.	0.44	0.	0.	0.	0.00	!	6.78
	SP	0.	0.	0.	0.	0.	0.	0.	0.20	!	0.20
	PC	0.	0.	0.	30.25	0.	0.	62.77	0.	!	93.01
		0.	6.34	0.	30.68	0.	0.	62.77	0.21		100.00
CLR - DST	GR	21.72	0.07	48.46	0.	1.93	0.08	0.92	0.	!	73.18
	SP	0.25	1.05	0.	0.	2.85	0.	2.12	0.01	!	6.28
	PC	0.	0.	0.	0.	0.	0.	20.54	0.	!	20.54
		21.97	1.12	48.46	0.	4.78	0.08	23.59	0.01		100.00

OPERAND USAGE SUMMARY

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
COM	- DST										
	GR	100.00	0.	0.	0.	0.	0.	0.	0.	!	100.00
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		100.00	0.	0.	0.	0.	0.	0.	0.		100.00
INC	- DST										
	GR	7.42	0.84	0.	0.	0.	0.	34.42	0.	!	42.68
	SP	0.	0.06	0.	0.	0.	0.	10.43	0.03	!	10.53
	PC	0.	0.	0.	0.	0.	0.	46.79	0.	!	46.79
		7.42	0.90	0.	0.	0.	0.	91.65	0.03		100.00
DEC	- DST										
	GR	64.08	0.00	0.01	0.	0.	0.	15.21	0.31	!	79.62
	SP	1.46	0.01	0.	0.	0.	0.	2.78	0.	!	4.25
	PC	0.	0.	0.	0.	0.	0.	16.13	0.	!	16.13
		65.55	0.01	0.01	0.	0.	0.	34.12	0.31		100.00
NEG	- DST										
	GR	97.81	0.	0.	0.	0.	0.	0.	0.	!	97.81
	SP	0.	2.19	0.	0.	0.	0.	0.	0.	!	2.19
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		97.81	2.19	0.	0.	0.	0.	0.	0.		100.00
ADC	- DST										
	GR	0.	55.60	0.	0.	0.	0.	0.	0.	!	55.60
	SP	0.	44.40	0.	0.	0.	0.	0.	0.	!	44.40
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		0.	100.00	0.	0.	0.	0.	0.	0.		100.00
SBC	- DST										
	GR	100.00	0.	0.	0.	0.	0.	0.	0.	!	100.00
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		100.00	0.	0.	0.	0.	0.	0.	0.		100.00

OPERAND USAGE SUMMARY

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
TST	- DST										
	GR	12.31	2.33	23.57	0.	0.06	0.	2.89	0.	!	41.16
	SP	0.	0.25	18.90	0.	24.24	0.	2.10	0.63	!	46.12
	PC	0.	0.	0.	0.23	0.	0.	12.49	0.	!	12.72
		12.31	2.58	42.47	0.23	24.30	0.	17.48	0.63		100.00
ROR	- DST										
	GR	100.00	0.	0.	0.	0.	0.	0.	0.	!	100.00
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		100.00	0.	0.	0.	0.	0.	0.	0.		100.00
ROL	- DST										
	GR	100.00	0.	0.	0.	0.	0.	0.	0.	!	100.00
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		100.00	0.	0.	0.	0.	0.	0.	0.		100.00
ASR	- DST										
	GR	90.42	0.	0.	0.	0.	0.	0.	0.	!	90.42
	SP	0.	9.58	0.	0.	0.	0.	0.	0.	!	9.58
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		90.42	9.58	0.	0.	0.	0.	0.	0.		100.00
ASL	- DST										
	GR	99.74	0.	0.	0.	0.	0.	0.	0.	!	99.74
	SP	0.	0.	0.	0.	0.	0.	0.26	0.	!	0.26
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		99.74	0.	0.	0.	0.	0.	0.26	0.		100.00

OPERAND USAGE SUMMARY

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
MOVB - SRC	GR	13.59	3.91	20.23	0.	0.	0.	11.43	0.01	!	49.17
	SP	0.	0.	0.	0.	0.	0.	23.86	6.25	!	30.10
	PC	0.	0.00	1.87	0.	0.	0.	8.08	10.78	!	20.73
		13.59	3.91	22.10	0.	0.	0.	43.37	17.03		100.00
MOVB - DST		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
	GR	53.87	0.86	18.25	0.	0.00	0.	1.38	14.54	!	88.90
	SP	0.	2.87	0.	0.	2.30	0.	2.54	0.	!	7.71
	PC	0.	0.	0.	0.	0.	0.	2.27	1.12	!	3.39
		53.87	3.73	18.25	0.	2.30	0.	6.19	15.66		100.00
CMPB - SRC		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
	GR	0.	19.73	12.48	0.	0.	0.	0.48	0.	!	32.69
	SP	0.	0.82	0.	0.	0.	0.	0.	0.00	!	0.82
	PC	0.	0.88	56.63	0.	0.	0.	2.04	6.93	!	66.49
		0.	21.43	69.11	0.	0.	0.	2.52	6.93		100.00
CMPB - DST		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
	GR	0.15	33.86	23.56	0.	0.05	0.	22.07	0.	!	79.69
	SP	0.	0.	0.	0.	0.	0.	1.48	3.91	!	5.39
	PC	0.	0.	7.75	0.	0.	0.	7.13	0.03	!	14.92
		0.15	33.86	31.31	0.	0.05	0.	30.68	3.94		100.00
BITB - SRC		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
	GR	10.18	0.	0.	0.	0.	0.	0.	0.	!	10.18
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	14.66	75.16	0.	0.	0.	0.	0.	!	89.82
		10.18	14.66	75.16	0.	0.	0.	0.	0.		100.00
BITB - DST		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
	GR	0.	31.74	0.	0.	0.	0.	31.00	0.	!	62.74
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	37.26	0.	!	37.26
		0.	31.74	0.	0.	0.	0.	68.26	0.		100.00

OPERAND USAGE SUMMARY

BICB - SRC	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	2.44	97.56	0.	0.	0.	0.	0.	!	100.00
	0.	2.44	97.56	0.	0.	0.	0.	0.		100.00

BICB - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.	84.63	0.	0.	0.	0.	15.37	0.	!	100.00
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	0.	84.63	0.	0.	0.	0.	15.37	0.		100.00

BISB - SRC	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.35	0.	0.	0.	0.	0.	0.	65.28	!	65.62
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	0.18	3.51	0.	0.	0.	1.06	29.64	!	34.38
	0.35	0.18	3.51	0.	0.	0.	1.06	94.91		100.00

BISB - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	94.91	4.15	0.	0.	0.	0.	0.64	0.	!	99.70
SP	0.	0.	0.	0.	0.	0.	0.	0.30	!	0.30
PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	94.91	4.15	0.	0.	0.	0.	0.64	0.30		100.00

CLRB - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.	1.77	28.13	0.	0.	0.	2.65	0.	!	32.56
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	0.	0.	0.	0.	0.	63.55	3.89	!	67.44
	0.	1.77	28.13	0.	0.	0.	66.20	3.89		100.00

INCB - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.	0.18	0.	0.	0.	0.	26.54	0.	!	26.71
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	0.	0.	0.	0.	0.	73.29	0.	!	73.29
	0.	0.18	0.	0.	0.	0.	99.82	0.		100.00

OPERAND USAGE SUMMARY

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
DECB -	DST										
	GR	0.	0.	0.	0.	0.	0.	99.18	0.	!	99.18
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.82	0.	!	0.82
		0.	0.	0.	0.	0.	0.	100.00	0.		100.00

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
TSTB -	DST										
	GR	0.	7.46	6.90	0.	0.	0.	21.37	0.	!	35.73
	SP	0.	0.	0.	0.	0.	0.	0.	2.49	!	2.49
	PC	0.	0.	0.	0.	0.	0.	37.18	24.60	!	61.78
		0.	7.46	6.90	0.	0.	0.	58.55	27.09		100.00

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
SXT -	DST										
	GR	100.00	0.	0.	0.	0.	0.	0.	0.	!	100.00
	SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
	PC	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
		100.00	0.	0.	0.	0.	0.	0.	0.		100.00

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
MUL -	DST										
	GR	1.22	0.	0.	0.	0.	0.	0.	0.	!	1.22
	SP	0.	0.	0.16	0.	0.	0.	0.25	0.	!	0.41
	PC	0.	0.	90.80	0.	0.	0.	7.56	0.	!	98.37
		1.22	0.	90.96	0.	0.	0.	7.82	0.		100.00

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
DIV -	DST										
	GR	0.10	0.	0.	0.	0.	0.	0.	0.	!	0.10
	SP	0.	0.	0.03	0.	0.	0.	0.06	0.	!	0.10
	PC	0.	0.	96.28	0.	0.	0.	3.53	0.	!	99.80
		0.10	0.	96.31	0.	0.	0.	3.59	0.		100.00

		R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
ASH -	DST										
	GR	0.02	0.	0.	0.	0.	0.	0.	0.	!	0.02
	SP	0.	0.	1.22	0.	0.	0.	0.	0.	!	1.22
	PC	0.	0.	98.77	0.	0.	0.	0.	0.	!	98.77
		0.02	0.	99.98	0.	0.	0.	0.	0.		100.00

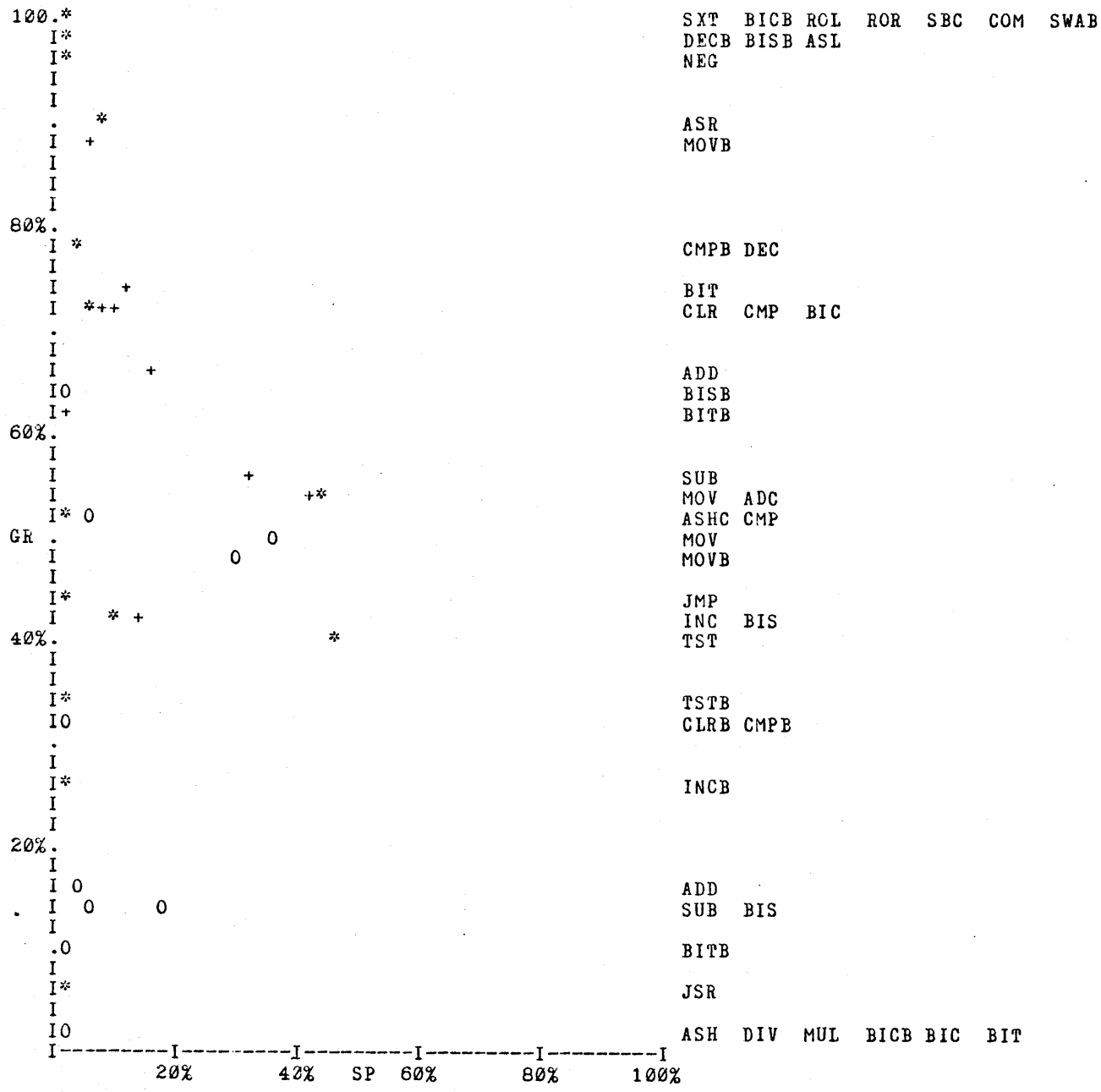
OPERAND USAGE SUMMARY

ASHC - DST	R	@R	(R)+	@(R)+	-(R)	@-(R)	X(R)	@X(R)		
GR	0.	52.06	0.	0.	0.	0.	0.	0.	!	52.06
SP	0.	0.	0.	0.	0.	0.	0.	0.	!	0.
PC	0.	0.	47.94	0.	0.	0.	0.	0.	!	47.94
	0.	52.06	47.94	0.	0.	0.	0.	0.		100.00

OPERAND SUMMARY FOR SELECTED INSTRUCTIONS

	MOVE		CLEAR	COMPARE		TEST	ARITH2		ARITH1	LOGIC2		LOGIC1	JUMP	CALL
	SRC	DST	DST	SRC	DST	DST	SRC	DST	DST	SRC	DST	DST	DST	DST
GR	28.11	45.33	21.72	39.36	49.76	12.31	15.52	61.39	48.92	4.65	57.28	98.20	0.	0.
(GR)	1.00	0.14	0.07	0.41	3.42	2.33	0.02	1.50	0.46	0.	6.01	0.	30.68	6.34
(GR)+	7.49	6.79	48.46	12.35	10.56	23.57	0.	0.	0.01	0.	0.	0.	0.	0.
@(GR)+	0.	0.44	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.01	0.44
-(GR)	12.33	0.14	1.93	0.01	5.10	0.06	0.	0.00	0.	0.	0.06	0.	0.	0.
@-(GR)	0.	0.	0.08	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
X(GR)	2.73	1.27	0.92	1.28	4.72	2.89	0.12	2.13	20.16	1.65	0.04	0.	0.	0.
@X(GR)	0.15	0.	0.	0.	0.16	0.	0.	0.	0.23	0.	0.	0.	14.65	0.00
SP	20.89	16.49	0.25	0.02	0.20	0.	0.29	8.10	1.05	0.	0.	0.	0.	0.
(SP)	0.03	4.62	1.05	0.89	0.80	0.25	0.	6.33	0.23	0.	9.97	1.63	0.	0.
(SP)+	8.57	0.08	0.	3.23	3.77	18.90	0.61	0.	0.	5.87	0.	0.	0.	0.
@(SP)+	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
-(SP)	0.	19.44	2.85	0.	0.	24.24	0.	0.	0.	0.	0.	0.	0.	0.
@-(SP)	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
X(SP)	7.53	1.53	2.12	3.41	3.65	2.10	3.86	5.16	4.79	0.08	2.88	0.17	0.	0.
@X(SP)	0.22	0.06	0.01	0.	0.32	0.63	0.	0.01	0.01	0.	0.11	0.	1.52	0.20
--	0.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
--	0.	0.	0.	0.43	0.	0.	0.00	0.	0.	0.	0.	0.	0.	0.
#N	5.29	0.	0.	36.75	15.39	0.	77.38	0.	0.	87.01	0.	0.	0.	0.
@#A	0.14	0.22	0.	0.	0.06	0.23	0.	0.	0.	0.	0.	0.	9.62	30.25
--	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
--	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
A	4.77	2.96	20.54	1.87	2.10	12.49	2.19	14.56	24.14	0.74	23.66	0.	43.52	62.77
@A	0.75	0.49	0.	0.	0.	0.	0.	0.82	0.	0.	0.	0.	0.	0.
	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

REGISTER TYPE CROSS COMPARISON - GR VS. SP
* = ONE OPERAND - DST
+ = TWO OPERAND - DST
0 = TWO OPERAND - SRC



CCOP QUALIFIER ANALYSIS

CODE	CCLR	CSET	SUM
NONE	0.	0.	0.
---C	99.86	0.	99.86
--V-	0.	0.14	0.14
--VC	0.	0.	0.
-Z--	0.	0.	0.
-Z-C	0.	0.	0.
-ZV-	0.	0.	0.
-ZVC	0.	0.	0.
N---	0.	0.	0.
N--C	0.	0.	0.
N-V-	0.	0.	0.
N-VC	0.	0.	0.
NZ--	0.	0.	0.
NZ-C	0.	0.	0.
NZV-	0.	0.	0.
NZVC	0.	0.	0.
	99.86	0.14	100.00

SUMMARY OF BRANCH INSTRUCTIONS

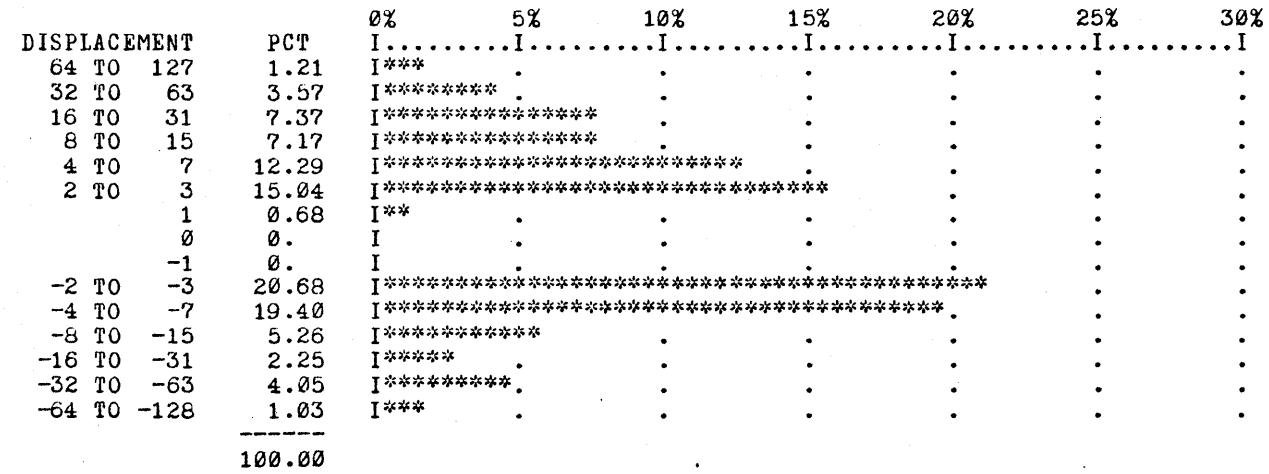
CONDITIONAL BRANCH SUMMARY (% OF CONDITIONALS ENCOUNTERED)

BRANCHES		TEST	0	1	
BPL	BMI	N---	0.14	0.00	0.14
BNE	BEQ	-Z--	37.38	25.51	62.88
BVC	BVS	--V-	0.	0.02	0.02
BHIS	BLO	---C	1.05	2.67	3.72
BGE	BLT	N-V-	6.49	4.10	10.59
BGT	BLE	NZV-	8.64	1.86	10.50
BHI	BLOS	-Z-C	3.53	8.61	12.14
			57.22	42.78	100.00

SUMMARY OF I-STREAM BREAKS

INSTRUCTION	POTENTIAL	ACTUAL	
B--	54.91	42.88	
SOB	0.62	0.60	
BRN	9.63	12.23	
JMP	9.57	12.16	
JSR	14.80	18.81	
RTS	9.93	12.61	
TRAP	0.03	0.04	
EMT	0.	0.	
BPT	0.	0.	
IOT	0.	0.	
INTR	0.25	0.31	
RTI	0.27	0.35	
	100.00	100.00	
INSTRUCTION RUNS	2.90	3.69	INSTRUCTIONS

BRANCH QUALIFIER STATISTICS FOR BRANCHES TAKEN



DISPLACEMENT	PCT
64 TO 127	1.21
32 TO 63	3.57
16 TO 31	7.37
8 TO 15	7.17
4 TO 7	12.29
2 TO 3	15.04
1	0.68
0	0.
-1	0.
-2 TO -3	20.68
-4 TO -7	19.40
-8 TO -15	5.26
-16 TO -31	2.25
-32 TO -63	4.05
-64 TO -128	1.03
	100.00

CONDITIONAL BRANCH OUTCOMES

	!---------- IGNORED ----------!			!---------- TAKEN ----------!			
TYPE	FORWARD	BACKWARD	SUM	FORWARD	BACKWARD	SUM	TOTAL
BR	0.	0.	0.	4.61	10.16	14.77	14.77
BPL	0.08	0.00	0.09	0.03	0.	0.03	0.12
BMI	0.00	0.	0.00	0.00	0.	0.00	0.00
BNE	3.10	2.58	5.67	8.36	17.47	25.83	31.50
BEQ	11.21	2.50	13.71	7.07	0.72	7.79	21.50
BVC	0.	0.	0.	0.	0.	0.	0.
BVS	0.02	0.	0.02	0.00	0.	0.00	0.02
BHIS	0.54	0.05	0.59	0.24	0.05	0.29	0.88
BLO	0.62	0.02	0.64	1.46	0.16	1.62	2.25
BGE	0.19	0.03	0.22	5.03	0.22	5.25	5.47
BLT	2.37	0.69	3.06	0.39	0.00	0.40	3.46
BGT	0.48	0.31	0.78	2.47	4.02	6.50	7.28
BLE	0.88	0.02	0.90	0.63	0.04	0.67	1.57
BHI	0.28	0.14	0.42	0.81	1.73	2.55	2.97
BLOS	6.37	0.02	6.40	0.73	0.13	0.86	7.26
SOB	0.	0.22	0.22	0.	0.73	0.73	0.95
	26.14	6.58	32.72	31.84	35.43	67.28	100.00