LSI-11 WRITABLE CONTROL STORE ENHANCEMENTS TO U.C.S.D. PASCAL

Gordon Smith and Roger Anderson
Lawrence Livermore Laboratory
University of California
P.O. Box 5507
Biomedical Sciences Division
Livermore, California 94550

ABSTRACT

The DEC KUV11-AA Writable Control Store was used to implement selected portions of the U.C.S.D. Pascal P-machine in firmware. The frequency and execution speed of P-machine instructions were measured in a battery of test programs to guide the selection process. A 32 to 46% reduction in execution time has been obtained for these test programs.

INTRODUCTION

The recent release of the KUV11-AA Writable Control Store (WCS)[1] has permitted user access to a level of the LSI-11 previously restricted to a select few. A number of interesting projects are possible with this product such as the application described here, the microprogramming of portions of Ken Bowles' U.C.S.D. Pascal P-machine.[2]

The LSI-11 Microprocessor and Writable Control Store

The LSI-11 is a microprocessor microprogrammed to emulate the PDP-11 instruction set. The microprocessor has 26 eight-bit registers that are addressed by a combination of direct and indirect means. Its instruction set is highly vertical, i.e., it is not unlike a conventional, albeit primitive, machine language. Control is exercised by a Translation Array, consisting of four programmed logic arrays, that examines the fetched PDP-11 level machine instruction and determines where microprogram execution is to begin. The Translation Array may continue to exert control by generating new inputs to the location counter as a function of the current value of the location counter, interrupt signals, and other control inputs.

The memory address space is 2K words. It is divided into four 512 word pages. Half of this address space, or two pages, is used to emulate the PDP-11. The EIS/FIS chip is optional and adds a third page of microcode that emulates an extended PDP-11 instruction set. These additional instructions include integer multiply and divide plus a battery of floating point instructions. The fourth page is left unused.

The WCS contains a 1K, or two page, random access memory that is primarily intended to be used as the third and fourth page of the microprocessor memory. Use of the WCS as the third page of memory is restricted by the Translation Array. Normally the EIS/FIS code resides on this page. To facilitate its execution the Translation Array is programmed to perform various control functions when execution reaches specific memory locations on the page. User microprograms must avoid these locations. Complications can be avoided entirely by loading the EIS/FIS code into one page of the WCS and restricting new microprograms to the other.

DEC has allocated opcodes 76700-76777 for user microprogramming. When the Translation Array encounters an opcode in this range, it directs control to a single address in the fourth page of memory. The user is then responsible for decoding the individual opcodes. Use of the Translation Array to facilitate execution of user microprograms is not supported.

The U.C.S.D. Pascal P-machine

The U.C.S.D. Pascal system is a complete stand-alone system designed to run on micro- and minicomputers. One of its most impressive features is its use of an underlying P-machine. The P-machine is a stack-oriented pseudocomputer that exists as an interpreter written in the assembly language of the host computer. Pascal source code is compiled to an intermediate P-code that is, in effect, the assembly language of the P-machine. This design makes the system highly portable. The operating system itself is written in Pascal. Only the relatively small native code interpreter must be written to transfer it to a new host. To date there have been successful implementations on the PDP-11 series and the 8080 family of microprocessors, including the 8080A and 8085. Other advantages of this type of implementation include the efficient use of small memories and fast compilation speed.

PROJECT DEFINITION

As described above, the execution of the U.C.S.D. Pascal P-machine is a two-level process. The LSI-11 microprocessor emulates a PDP-11 computer, which in turn simulates the P-machine. We are exploring the possibility of having the LSI-11 microprocessor emulate the P-machine directly. The primary advantage will be an increase in program execution speed.

One of our initial observations was that there is not enough room in the WCS to implement the entire

P-machine. On the average, it requires more microcode than macrocode to implement a P-machine instruction. One measure put the ratio at 1.7 words of microcode for every word of macrocode. The current LSI-11 macro-level interpreter requires 2.4K words. A microprogrammed P-machine, we estimate, will require 3 to 4K words of microcode without the use of the Translation Array.

For this project we used one of the pages of the WCS to implement portions of the P-machine and the other page to contain the EIS/FIS code. Our task was to select the best portions to microcode.

SELECTION OF PORTIONS OF THE P-MACHINE TO MICROCODE

Control Structure

To execute a P-machine instruction, an opcode must be read from macro-level memory, control transferred to the appropriate routine for execution, and control returned for the next instruction fetch. This process will be called the interpreter fetch sequence. In the macro-level P-machine interpreter, this takes approximately 25 µs for most instructions and 30 to 45% of the total program execution time. This data makes the interpreter fetch sequence an excellent candidate for microprogramming.

The interpreter fetch sequence in our micro/macro interpreter uses a variation of the scheme used in the macro-level interpreter. In this scheme the P-machine opcode is used to index a table of macro-addresses for the respective routines. This same table is used by the mcro/macro interpreter with the difference that micro-addresses are also stored in the table. The addresses are differentiated by having the high order four bits of the words containing the 11-bit micro-addresses set to ones. Macro-addresses never have these bits set because the high end of memory is normally reserved for I/O devices.

The microcoded interpreter fetch sequence has two entry points. First, its execution can be initiated from macrocode by the opcode 76704. Second, after a microcoded P-machine instruction is executed, a jump is made directly into the interpreter fetch routine. The direct entry from microcode is faster than from macrocode.

The speed of the microcoded interpreter fetch sequence averages approximately 14.5 µs for most instructions. This is somewhat disappointing, but nevertheless is our most successful single microcoded routine. It alone can reduce program execution time by 12 to 19%.

The micro/macro control structure can handle microcode instructions in addition to 76704. Useful instructions include general purpose instructions such as a block move. Other useful ones are specialized instructions that execute the frequently used parts of a P-machine instruction, leaving the logic for handling special cases or error conditions in macrocode. The scheme for supporting the non-76704 instructions centers around the use of the microstruction, Modify Instruction (MI). This technique is discribed in detail in the WCS Users Guide.[1] Briefly, the MI instruction uses the fetched macro-level instruction to index a table of jump instructions in the microprocessor memory.

Another key element of the control structure is the handling of interrupts. Periodically, during the execution of long microcode routines a check is made to see if interrupts are pending. If an interrupt has occurred, execution is aborted and control returns to macrocode to service the interrupt. After the interrupt has been serviced, the micro-routine is restarted from the beginning. Microcoded routines that may be aborted must be careful to postpone making permanent changes until after the last interrupt check.

P-machine Instructions

A series of tests were conducted to determine the best P-machine instructions to microcode. For these tests a DEC KWV11-A programmable real-time clock was used to maintain a running count of the number of microseconds spent executing each P-machine instruction. Also, each instruction was counted as it was executed. From this data the average execution speed, percent of program execution time, and percent of instruction execution frequency were calculated for each instruction. Instructions with a high percentage of program execution time and/or a high frequency of execution are prime targets for microprogramming. Frequency of execution is important because of the faster direct entry from microcode to the interpreter fetch sequence.

Unfortunately, there does not exist a typical program that can be tested. Instead a battery of test programs was assembled to gain insight into commonly used programs. These test programs are as follows:

Compilations	-	Six programs totalling 3341 lines of source code were compiled. The programs were selected at random. Two were written by one of the authors of this paper, one was WHETSTONE (see below), and three (XREF, CALC, & RT11TOEDIT) were selected from the software distributed by U.C.S.D..
WHETSTONE	-	This program is a synthetic benchmark developed by H. Curnow and B. Wichmann[3]. It exercises a computer in a manner considered typical of scientific programs. Specifically, it includes array manipulation, conditional jumps, procedure calls, integer arithmetic, and trigonometric and other standard functions using real numbers.
Sorts	-	Three programs, Quicksort (recursive), Quicksort (nonrecursive), and Heapsort, were used to sort an identical array of 3,000 reasonably random

integers. The algorithms used were based on those given by N. Wirth[4].

Miscellaneous exploratory programs — Two programs were run in the hopes of gaining special insights into the behavior of the P-machine. The first creates a cross-reference of a Pascal source program. This XREF program was written at Sperry-Univac. The second (BTSI) builds a balanced tree in the heap and conducts searches of that tree. Again, the algorithm was based on one by N. Wirth[4].

An example of these test results is presented in Table 1.

With this test data, we are able to select portions of the P-machine to microprogram, code those portions, and then evaluate the resulting performance. The ultimate basis for the evaluation is the percent reduction in program execution time derived per word of WCS used and the consistency of the improvement across the spectrum of test programs.

RESULTS

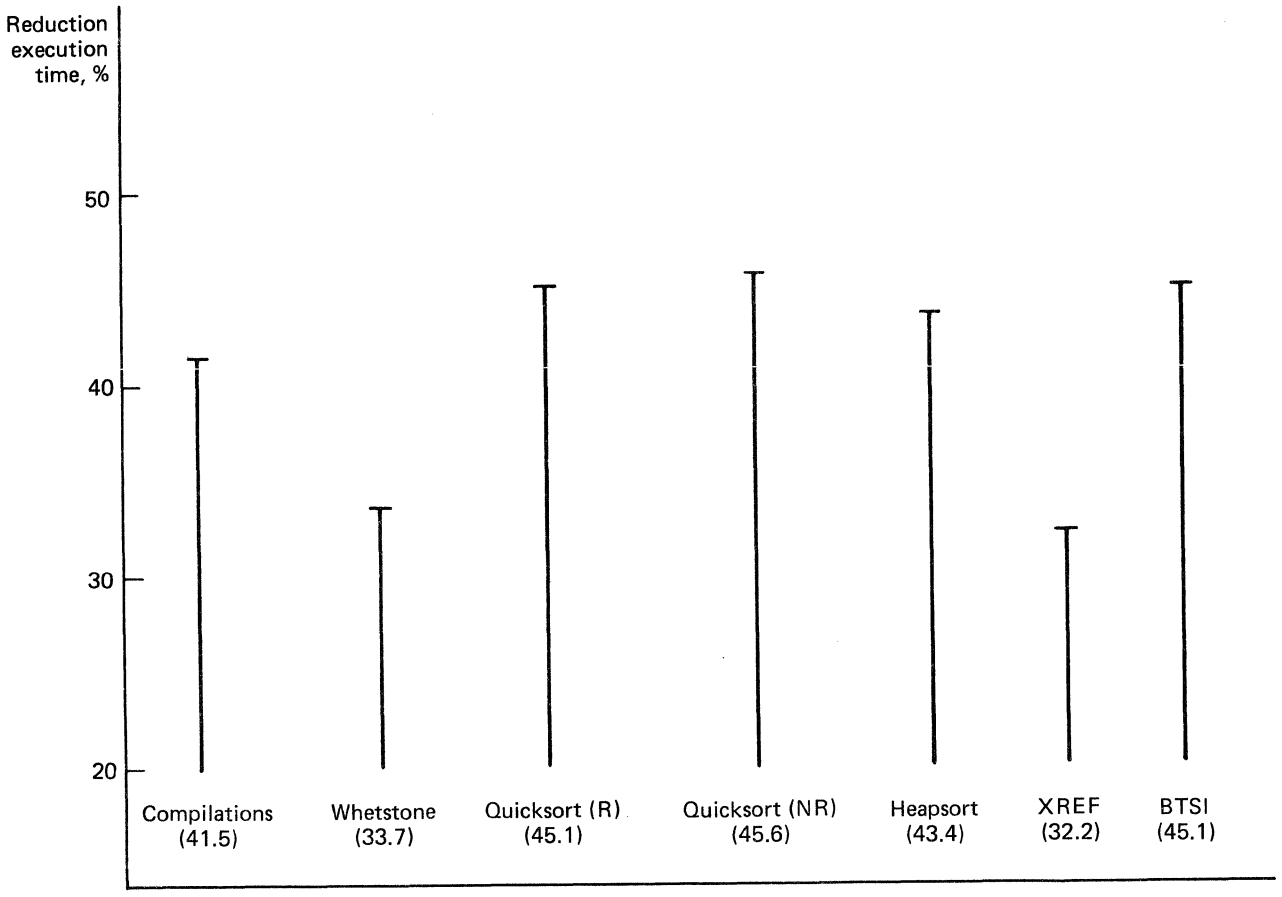
To date a page of microcode has been coded. All of this code was written before the test series described above was completed, although preliminary test results were available. Tests using a line time clock have measured a 32 to 46% reduction in execution time for the test programs when compared to the macro-level LSI-11 interpreter. (Note, both interpreters use the EIS/FIS code). These results are shown in Fig. 1. The page of microcode contains 19 P-machine instructions and four general purpose instructions. The speed improvements

Table 1. Execution speed, percent of execution time, and execution frequency of individual P-machine instructions coded in LSI-11 assembler language. These results were obtained from the compilation of 3341 lines of source code. The average execution speeds are adjusted to eliminate the time for the interpreter fetch sequence. The percent of program execution time includes the interpreter fetch sequence time in the calculation. The interpreter fetch sequence and the single instruction SLDC are in microcode.

Mnemonic	Instruction	Average execution speed in μs	Percentage of program execution time	Percentage of execution frequency
CIP	Call Intermediate Procedure	632	21.6	2.0
CSP	Call Standard Procedure	1186	17.5	0.9
FJP	False Jump	25	3.7	8.7
RNP	Return From Non-base Procedure	75	3.1	2.5
SRO	Store Global Word	31	2.5	4.7
SLDO	Short Load Global Word, total	12	2.5	12.7
INN	Set Inclusion	114	2.1	1.1
SLDL	Short Load Local Word, total	12	1.7	8.4
LDO	Load Global Word	32	1.6	3.0
CLP	Call Local Procedure	205	1.6	0.5
EQUI	Integer Comparison, =	20	1.6	4.5
UJP	Unconditional Jump	22	1.5	3.9
LDM	Load Multiple Words	68	1.4	1.2
STL	Store Local Word	31	1.3	2.5
CXP	Call External Procedure	487	1.1	0.1
XJP	Case Statement	63	1.1	1.0
ADI	Add Integer	10	1.0	6.0
UNI	Set Union	92	0.8	0.5
LDB	Load Byte	21	0.8	2.3
LAO	Load Global Address	31	0.7	1.4
SIND	Short Index and Load Word, total	17	0.7	2.4
LLA	Load Local Address	31	0.7	1.3
SLDO12	Short Load Global Word, offset 12	12	0.7	3.4
SLDC	Short Load Word Constant, total	3	0.6	15.3
IXA	Index Array	75	0.5	0.4
SLDO3	Short Load Global Word, offset 3	12	0.5	2.6

The remaining Instructions have percents of program execution time of less than 0.5% and percents of total frequency of execution of less than 2.3%.

Fig. 1. Percent reductions in program execution time obtained by the micro/macro interpreter when compared to the macro-level interpreter.



obtained for the P-machine instructions and the number of words of WCS required to code them are given in Table 2. The general purpose instruction are:

An instruction to retrieve "BIG" operands. These operands may be either one or two bytes long, depending on whether the sign bit of the first byte is set.

An instruction to traverse down the static links of the P-machine stack n levels.

A block move instruction that increments the source and destination addresses as each word is moved.

A block move instruction that decrements the source and destination addresses as each word is moved.

The microcode produced to date and the detailed test results and procedures are available from the authors on request. Test were run using the LSI-11/2 (KD11-HA) with the MSV11-DD 32K memory.

WHAT CAN BE DONE

Work is still in progress. Both the selection of routines to microcode and the density of the microcode can be improved. The current reduction in execution time, as previously mentioned, is 32 to 46%. We expect that this figure can be improved by several percentage points. A final figure of roughly 45 to 55% seems to be possible. When it is completed, this code could be programmed into read-only memory and made available at a price considerably lower than the WCS board. Although such a product may not be of widespread interest, some installations may find it worthwhile.

An exciting possibility is the complete conversion of the microprocessor to a P-machine. In particular, if the Translation Array could be used for the interpreter fetch sequence and other control functions, speed improvements should far outstrip anything that can be accomplished with a single page of WCS. Space will still be a major problem even with the Translation Array. If this problem can be overcome speed improvements by a factor of four or more do not seem unrealistic.

ACKNOWLEDGEMENT

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48.

Table 2. Microcoded P-machine instruction execution times and the number of WCS words used. The timing results were obtained from a test consisting of two compilations, the Whetstone and Quicksort (nonrecursive). Execution speeds are adjusted to eliminate the time for the interpreter fetch sequence. Similarly, microcode for the interpreter fetch sequence is not counted in the "words of WCS" figures. The number of "Words of WCS shared" with other routines are included in the number of "Total words of WCS".

*Figure is an estimate.

Mnemonic	Instruction	Micro time, μs	Macro time, μs	Total words of WCS	Words of WCS shared
AND	Logical And	5	17	6	0
CHK	Check Subrange Bounds	22	26	19	0
CIP	Call Intermediate Procedure	188	542	29	0
CLP	Call Local Procedure	75	189	113	0
FJP	False Jump	4	24	20	13
GRTI	Integer Comparison, $>$	10	23	17	9
LAO	Load Global Address	10	38	22	13
LDCI	Load Constant Word	5	22	8	0
LDM	Load Multiple Word	22	56	24	0
LDO	Load Global Word	10*	32	24	13
LEQI	Integer Comparison, $\leqslant$	11	21	17	9
LLA	Load Local Address	9*	31	22	13
NEQI	Integer Comparison, $\neq$	8	21	12	9
RNP	Return From Non-base Procedure	24	74	49	0
SLDC	Short Load Word Constant	3	6*	4	0
SRO	Store Global Word	10*	31	23	13
STL	Store Local Word	9	31	23	13
UJP	Unconditional Jump	6	24	14	13
XJP	Case Statement	16	63	28	0

*Figure is an estimate.

REFERENCES

(1) LSI-11 WCS Users Guide, EK-KUV11-TM-001, Digital Equipment Corporation, Maynard, Mass. (1978).

(2) UCSD (Mini-Micro Computer) Pascal, Release Version I.4, January 1978, Institute for Information Systems, University of California at San Diego, La Jolla, Ca., Ken Bowles director.

(3) H.J. Curnow and B.A. Wichmann, "A Synthetic Benchmark", The Computer Journal, Vol 19, No 1, February 1976, pp 43-49.

(4) N. Wirth, Algorithms + Data Structures = Programs, Prentice Hall, New Jersey, 1976.