CMU-CS-78-106

# Harp:

## A Low-Cost 25 MIPS Digital Processor

Stan Kriz
Raj Reddy
Brian Rosen
Steven Rubin
Steve Saunders

Computer Science Department
Carnegie-Mellon University

February 25, 1976

Reissued for general distribution
February 17, 1978

Harp is an ultra-high performance processor motivated by the low-level processing requirements of machine perception tasks. It is designed for extreme speed, low cost, easy producibility, simplicity of structure and programming, and complete diagnosability. This paper discusses the design philosophy, architecture, instruction set, and performance of Harp.

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense under contract number F44620-73-C-0074 and is monitored by the Air Force Office of Scientific Research.

Preface

The design of Harp was completed in 1975; construction of a prototype proceeded far enough to validate the engineering aspects of the design and assure us that the speed goals had been met, but not to a complete running Harp system. We nevertheless feel that this description of its design and philosophy and the performance results of careful simulation may be of interest to others.

R.R. & S.S.

## I. Introduction

The real-time operation of algorithms that arise in speech and vision research is often limited by the speed of the low-level signal handling algorithms, which in turn are typically limited by the computation time of a critical inner loop. To speed up the execution times of these algorithms, dedicated special purpose hardware or very fast auxilliary processors have been used to implement the critical code. In signal processing problems, and especially with the FFT, good results have been obtained with a special machine architecture relying heavily on instruction-cycle overlap (pipelining) and on multiple parallel arithmetic units capable of performing certain complex operations, notably the FFT "butterfly" multiply, very efficiently [FDP] [SPS-41] [AP-120B]. These processors are almost invariably both expensive to build and difficult to program, due to their complex structure and parallelism. Their computational power on suitable tasks has nevertheless made them useful.

At Carnegie-Mellon University we are investigating artificial intelligence algorithms which, although they deal directly with input or generated signals, do not fall entirely within the realm of conventional "signal processing". Floating point and complex arithmetic are not usually required, and while the FFT is used in some problems, most algorithms cannot consistently utilize such special features as butterfly-multiply hardware efficiently. Low precision integers may be used for most computations; often only small bit fields are manipulated, as in vision tasks where pixels may be as small as 4 bits. When signal processing is done, multiple precision is as satisfactory as floating point. Considerable logical manipulation and decision-making capability are also called for. We have designed a new processor, Harp, which suits the needs of our research and is considerably more general-purpose in its orientation than the signal processors.

Harp has the following design goals, some of which are in keeping with the signal-processor approach, some not:

A. Extremely high speed (under 40 ns cycle).

B. Low cost ($5000-$10,000 parts).

C. Programming simplicity: transparent pipeline and parallelism.

D. Large number of high-speed registers.

E. Separate instruction and data memories for overlapped access.

F. General-purpose capability (minicomputer-like instruction repertoire).

G. 16-bit data and instructions for minicomputer compatibility.

H. Large high speed second-level memory (4K - 64K).

I. Extensive diagnostic capability.

Software tools were used extensively during the design of Harp. A simulator, debugger, and assembler, were built very early in the design phase of each proposed architecture. These tools were used to code 8 representative algorithms (summarized in Section VII) to give the designers programming experience and performance information. A number of complete design iterations resulted before Harp was committed to hardware.

## II. Architecture

Harp is an auxiliary processor to a host PDP-11. It is a "pure" 16-bit machine: instructions, data paths, and ALU are all 16 bits wide; there are no "byte" instructions or packed data representations.

The Harp processor operates from two small, very high speed memories--the data and instruction working stores. There are no data registers or accumulators in

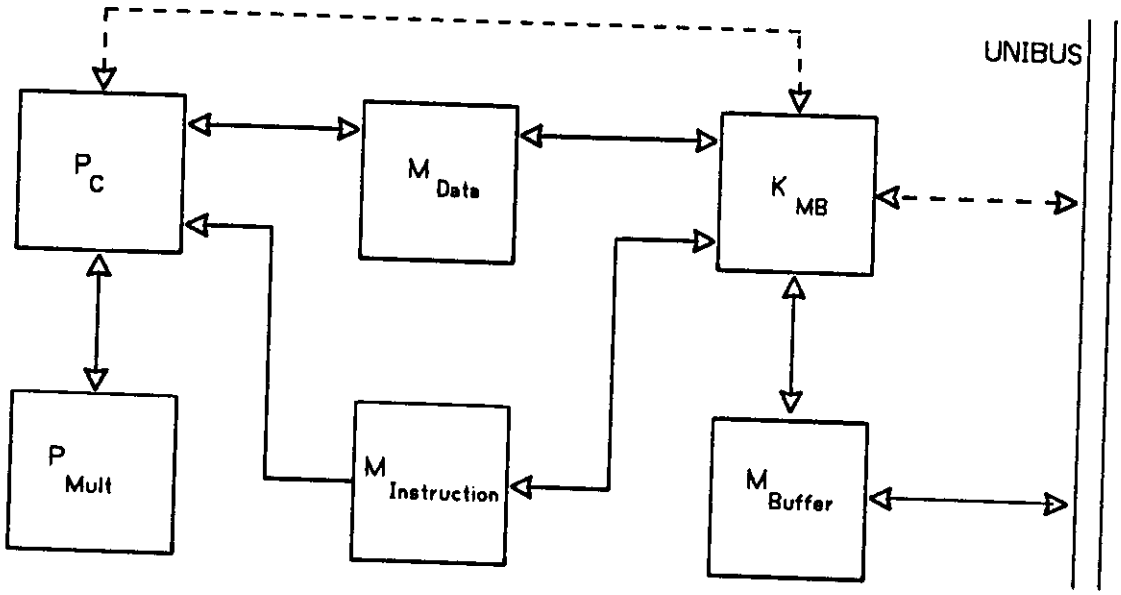


Figure 1: PMS Diagram of Harp

the conventional sense. The data working stores have a 13 ns cycle time, providing the processor with a 1.2 Gbps (billion bits per second) data bandwidth.

The working store sizes -- 64 words of data, 64 to 256 words of instructions -- were determined by the number of address bits available in an instruction and by physical size limitations caused by signal propagation delays. Because Harp is designed to run small code fragments these small working stores are usually satisfactory. The enforced separation of data from instructions, though it seldom occurs in general-purpose machines, does not impact programming style, and gains considerable speed by overlapping code and data memory accesses.

The processor itself operates in the usual pipelined instruction overlap fashion (see Figure 2). Instruction execution is divided into four stages: 1) instruction fetch 2) operand fetch 3) execution 4) result store. Total execution time is 133 ns, with the four stages overlapped to permit a new instruction fetch every 40 ns. Thus the instruction rate is 25 Mips (million instructions per second).

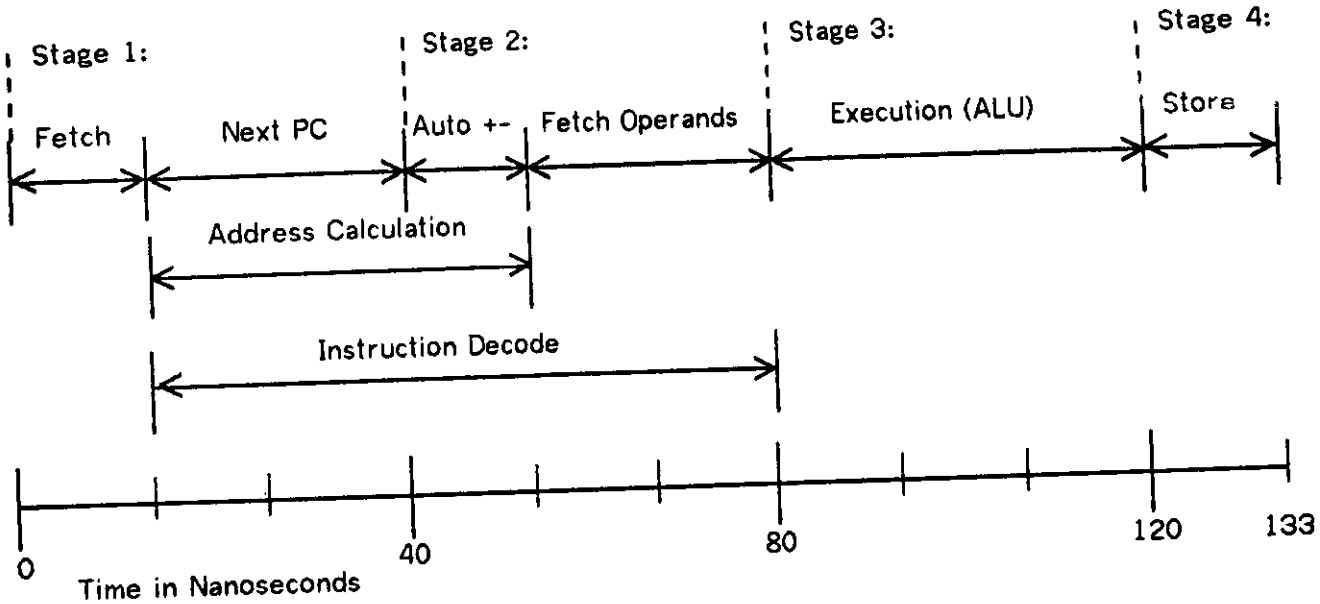


Figure 2: Harp Pipeline Timing

Cost and generality constrained the design of the Harp arithmetic unit. It is limited to a 16 bit two's complement general purpose function generator and a single, separate (and slower) multiplier. The algorithms listed in Section VII show that execution speed is not limited by the instruction set.

The working store contents can be transferred to or from a large buffer memory via a block transfer mechanism. The transfer rate of 0.8 Gbps (20 ns per word) avoids bottlenecking the fast processing of Harp. Buffer memory is expandable from 4K to 64K and is double-ported, one port permitting high speed transfers to the Harp working stores, the other compatible with a PDP-11 UNIBUS.

No single-word direct access to buffer memory is provided for several reasons: 1) this memory is interleaved, and the delay for initiating access is five times greater than the average transfer time; 2) allowing access an a cycle-to-cycle basis would have slowed Harp's execution rate considerably; 3) two words of working store are taken up by addresses for each reference to buffer memory.

The block transfer machanism is used to overlay programs too large to fit in the available instruction memory. Block transfers are fast enough to make frequent overlays acceptable in many situations: a 64-word transfer takes less than 1.5 us.

Since Harp is intended to operate in conjunction with a host computer, no input or output capability is provided other than the buffer memory connection to the UNIBUS. The UNIBUS bandwidth is sufficient for most I/O to real-time devices and mass storage. Since the relatively slow PDP-11 coordinates these transfers to the buffer memory, no hardware interrupt capability is included in Harp.

## III. Instruction Set

Two-address instructions are advantageous in high speed processing, since one such instruction can accomplish as much as two or three single-address instructions. Including two-address or "2-op" instructions in the 16 bit Harp instruction words, however, forced a tightly-packed coding of address and opcode fields. Careful consideration of the algorithms to be implemented on Harp led to two helpful observations: 1) only a few kinds of double operand instructions are used in a given loop, although the instructions themselves vary from algorithm to algorithm (see details in Figure 4); 2) indexed addressing with some kind of auto-incrementation is very useful for the vector-oriented operations. These considerations led to the unconventional instruction format shown in Figure 3.

The upper two bits of the instruction word, if nonzero, indicate a two-op instruction. The three possible nonzero combinations select one of three op-code registers: OPA, OPB or OPC. The contents of the selected register then determines the particular two-op instruction from among 19 implemented functions. Thus a

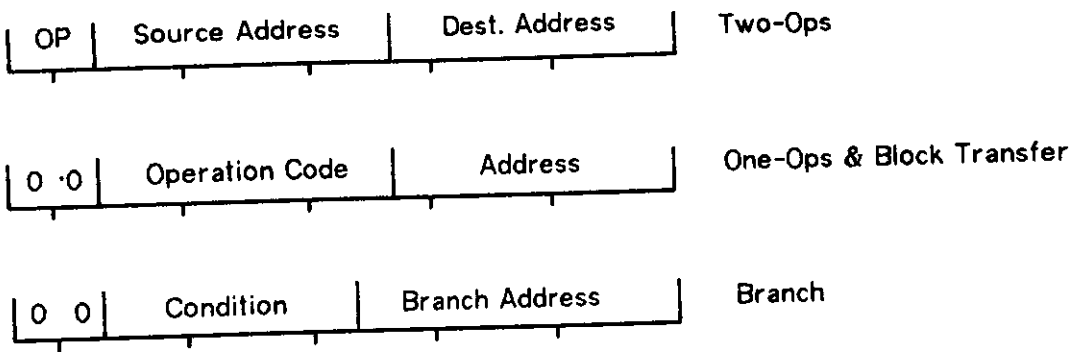


Figure 3: Harp Instruction Formats

program is limited to only three double-op instruction types at a time, but the programmer can select which three by loading the op-code registers.

The operand address fields contain 7 bits: 6 to directly address the 64 locations in the data working store, and one addressing mode bit. The mode bit selects one of a pair of available modes as determined by the ADRM register (see Appendix B). There are two index registers, X1 and X2, with auto-increment and auto-decrement capability. These registers may also be modified by some of the branch instructions for simplified loop control.

If the upper instruction word bits are zero, Harp decodes the instruction to determine whether it is a single-op, branch, register load/store, or block transfer. The 16 single-op instructions are similar to PDP-11 instructions. Arithmetic and logical operations set the N, Z, C, and V bits in the PS register which are used by the conditional branch instructions, again following the PDP-11 conventions. There are instructions for loading and storing the 8 state registers (OPA, OPB, OPC, ADRM, X1, X2, PS, PC) including a literal load.

The block transfer instructions move data between the buffer memory and the working stores and optionally interrupt the PDP-11. A two word descriptor in working

store provides a buffer memory start address, a working store start address, and a block length.

A special set of instructions dispose of the output of the separate multiplication processor, which performs 16x16 bit multiplications and retains the 32-bit product. Instructions are provided to store the two halves of this result in data memory and to add them to data memory (to accumulate a double-precision inner product). The multiplication processor operates in parallel with the central processor but receives instructions and operands from it. The 16x16 multiply takes 80 ns to complete, so a program must execute at least one instruction after the MUL before accessing the product.

Though Harp is a pipelined machine, the pipe is invisible to all instructions except explicit state-register references. Branch lookahead hardware avoids delay while allowing natural instruction sequencing. Programmers need not be concerned with such complications as "clearing the pipe on a branch" which plague most signal processors.

## IV. Engineering Considerations

Harp is implemented with 10K series ECL integrated circuits in the processors, working stores and their control [MECL]. Approximately 400 ECL chips are mounted on 8 double-sided 9 x 12 inch PC boards. The "backplane" is square shaped instead of flat, with sockets for two PC boards on each of the four sides; the boards plug into this "core" in a cross configuration so that the component side of each board is accessible at all times. This radial construction technique keeps inter-board communication delay very short, and during servicing provides access to all chips

simultaneously without the need for extender boards. Although the cross consumes considerable packaging volume, it is rackmountable at 17 x 17 x 12 inches.

Double-sided PC construction was chosen over wire-wrap because of cost, signal fidelity, and reproducibility. Multilayer PC, with its higher prototype turn-around time and expense is not justified by the present circuit density.

Memory chips of sufficient size and speed for the buffer memory are presently least expensive in TTL, so this memory and the PDP-11 interface are implemented with TTL. The buffer memory chips and their control circuitry are mounted on PC boards adjoining the cross. There is room to expand the memory to 64K words (16 boards) while keeping the memory bus length to the ECL converters under 12 inches. Cables connect to the PDP-11 interface and control circuitry. This 100 chip TTL circuit is wire-wrapped in the prototype.

The prospect of having several Harps in our environment encouraged diagnostic capability to ease the hardware maintance burden. Since most of the machine failures are expected to be hard static faults rather than high speed timing problems, considerable work has gone into a hardware diagnostic system and its software support package. All of these involve the PDP-11 as an interrogater and data collector. The machine clock may be stopped and Harp may be single-stepped by PDP-11 software. Between cycles, all registers and inter-board data and control signals may be examined by the PDP-11 without affecting the state of Harp; the working stores can be accesed through the Harp processor while Harp is halted.

Lower-level diagnostics exercise the buffer memory, working stores, processor, and control at levels sucessively deeper within Harp. Combined with the accessibility of inter-board signals, these should allow location of faults at the register level.

## V. Software

Harp is fully supported by an extensive collection of system software, including a symbolic assembler, an unusual graphics-based debugging package, a complete simulator, and a set of PDP-11 routines for Harp control.

The Harp display-oriented debugger runs on the PDP-11 and uses a graphic display terminal to maintain a picture of the instruction memory (decoded), the data memory, and the state registers; it has facilities for tracing programs and modifying memory in Harp. The debugger can be used with either the simulator or the actual hardware.

A package of BLISS-11 routines and macros allows the PDP-11 program to access all Harp memories and to control execution of Harp. These routines are the basic facility for inner loop execution on Harp.

## VI. General-Purpose Capabilities

The designers of Harp began with a view of creating a "functionally specialized architecture" for the problems encountered in speech and vision research. As we progressed it became clear that the only relevant common characteristics of the tasks are 1) fairly large amount of processing on each datum, and 2) small items, usually 4 to 12 bits of precision. The first observation is expressed in the size of data working store -- 64 words is large if considered as "registers", but small for a "main memory"; as "working store", it is well matched to Harp's tasks. The second led to the choice of small (16 bit) integers as the data type. The resulting design looks not at all like a "specialized" processor, but more like a clean vertically microcoded minicomputer with residual control. Notably, at least one recent signal processor design is billed by its builders as basically a "high performance minicomputer" [LDVT].

We expect Harp to be so cost-effective that it will be attractive whenever more processing power is needed on a PDP-11 system or similar minicomputer.

## VII. Performance

The simulated performance of Harp on 8 representative tasks is shown in Figure 5, compared to the performance of two conventional computers on the same tasks. The effectiveness of the Harp architecture and instruction set design can be seen in Figure 4 which shows frequencies of use of various machine features.

| ALGORITHM | TIME | XFER TIME | % XFER TIME | INST SPACE USED | BITS OF PRECISION |
|---|---|---|---|---|---|
| Histogram | 204 | 75 | 33 | 41 | 8 & 19 |
| Dragon | 271 | 90 | 37 | 64 | 16 |
| FFT | 36 | 11 | 33 | 63 | 16 & 32 |
| Temp. Match | 145 | 22 | 31 | 26 | 12 & 28 |
| Fit Boxes | 22 | 5 | 15 | 47 | 10 & 16 & 26 |
| Auto Corr. | 1332 | 201 | 15 | 55 | 12 & 32 |
| Edge Sup. | 230 | 21 | 9 | 63 | 8 |
| Smoothing | 116 | 33 | 28 | 34 | 8 |
| TOTAL | 2356 | 458 | 19 | 393 | |
| TOTAL W/O Auto Corr. | 1024 | 257 | 25 | | |

| ALGORITHM | TWO-OPS USED | DYN % TWO-OPS | STAT % TWO-OPS | DYN % MULT TIME | DYN % BRANCHES |
|---|---|---|---|---|---|
| Histogram | MUL,ADD,AND | 40 | 29 | 8 | 4 |
| Dragon | MOV,ADD,CMP | 45 | 48 | 0 | 12 |
| FFT | MOV,MUL,ADD | 16 | 24 | 10 | 5 |
| Temp. Match | MOV,MUL | 18 | 19 | 36 | 19 |
| Fit Boxes | MOV,MUL,CMP | 23 | 19 | 26 | 17 |
| Auto Corr. | MOV,MUL,ADD | 11 | 24 | 22 | 32 |
| Edge Sup. | MOV,SUB,CMP | 32 | 24 | 0 | 27 |
| Smoothing | MUL,ADD,SUB | 66 | 29 | 0 | 15 |
| TOTAL | | 22 | 35 | 16 | 25 |
| TOTAL W/O Auto Corr. | | 37 | | 8 | 16 |

Figure 4: Harp Statistics

| Algorithm | PDP-10 SAIL | PDP-11 ASM | HARP |
|---|---|---|---|
| Histogramming: | 7.62 | 5.99 | 0.172 |
| Dragon (Speech Understanding): | 12.39 | 9.84 | 0.219 |
| Butterfly Multiply for FFT: | 1.49 | 2.50 | 0.029 |
| Template Matching (14x40 Multiply): | 25.04 | 17.96 | 0.114 |
| Autocorrelation with Hamming Window: | 126.36 | 130.79 | 1.054 |
| Edge Suppression: | 14.93 | 6.73 | 0.178 |
| Picture Smoothing or Texture: | 20.66 | 9.11 | 0.103 |
| Matrix Multiplication for 3-D Graphics: | 5.97 | 5.32 | 0.018 |

Figure 5: Comparative Times

Histogramming is an image understanding algorithm that examines an entire picture and yields the total number of points at each intensity level. Two data types are used: picture elements which are 8 bits, and 256 "buckets" for counting picture elements. The buckets must be at least 19 bits for pictures which are 480 by 640 points.

Dragon is a speech recognition system that uses probability networks to understand speech. The inner loop involves calculating the probabilities of transitioning through the network at each time interval. All values are represented as logarithms so that no multiplication is necessary. Sixteen-bit precision is adequate throughout all calculations.

Template matching, a speech understanding algorithm, compares the parametric representations of two signals. It multiplies a 1x14 array with a 14x40 array to give a 1x40 array. Maximum input precision is 12 bits, so 28 bits are adequate for calculations and output.

The graphics algorithm Box Fitting retrieves points in a sparsely represented three-dimensional space. The space is broken down into as many as 8 arbitrarily rotated and located rectangular boxes, each completely described by a 4x4

transformation matrix. Given a point in 3-space, the list of boxes is searched to find out whether the point is in a box. If successful, a search is done on 8 sub-boxes, etc. The 3-space points are 10 bits and the transformation matrices 16 bits but the intermediate results need 26 bits.

Autocorrelation is a speech algorithm for examining waveforms. The Harp algorithm performs a Hamming window normalization on the input data prior to the autocorrelation. This consists of multiplying 12-bit signal values by a window function to give a 12-bit signal. The algorithm produces correlation figures for 15 different period intervals along an input waveform. The output values, multiplied and summed across 200 samples, must be at least 32 bits.

Smoothing is an image understanding algorithm that examines every point in a picture and smoothes it in relation to its context. The 8 points surrounding the point to be smoothed are summed; if the sum exceeds a threshold, the point is smoothed out. Another image understanding algorithm is Edge Suppression, used for thinning out the edges in a picture by removing extraneous picture elements on an edge. Each scan line is examined, and only local maximum points are retained. Both of these image algorithms work on picture elements of up to 8 bits in size.

Fast Fourier Transform is the standard signal processing algorithm that transforms from a time domain to a frequency domain. This particular implementation does a bit-reversal (re-organizing of data) and a transform of 1024 complex values using 16 bit integers.

## VIII. Acknowledgements

Raj Reddy saw the need for Harp and provided direction to the project. Together, the authors developed the architecture and instruction set. Stan Kriz designed the ECL logic; Brian Rosen designed the buffer memory. Steven Rubin and Steve Saunders wrote the simulator, Steven Rubin the debugger, David King the Harp assembler. Steven Rubin and Robert Watkins wrote the test algorithms. Greg Lawson assisted with the diagnostics. The authors would also like to thank Bill Broadley and Jim Teter for helpful discussions and Mark Firley, Nancy Whitaker, Eric Ostrom and Eric Woudenberg for technical assistance.

# APPENDIX A

## Instructions

| Mnemonic | Description |
|---|---|
| **Two-Op Instructions** | |
| ADD | Add |
| ADDC | Add with carry |
| SUB | Subtract |
| SUBC | Subtract with carry |
| RSUB | Reverse subtract |
| RSUBC | Reverse subtract with carry |
| MUL | Multiply |
| MOV | Move |
| MOVN | Move negative |
| AND | Logical AND |
| BIS | Logical OR |
| NAND | Logical NAND (Not-And) |
| NOR | Logical NOR (Not-Or) |
| XOR | Logical Exclusive Or |
| EQV | Logical Equivalence |
| BIC | Bit Clear |
| IMP | Logical Implication |
| BIT | Test bits |
| CMP | Compare |
| **One-Op Instructions** | |
| NOP | No operation |
| HALTI | Halt and interrupt |
| TST | Test |
| CLR | Clear |
| COM | Complement |
| NEG | Negate |
| INC | Increment |
| DEC | Decrement |
| ROR | Rotate right |
| ROL | Rotate left |
| ASR | Arithmetic shift right |
| ASL | Arithmetic shift left |
| ADC | Add carry |
| SBC | Subtract carry |
| SXT | Sign extend |

Block Transfer Instructions

| | |
|---|---|
| DATAI | Read into data memory |
| DATAO | Write from data memory |
| INSTI | Read into instruction memory |
| INSTO | Write from instruction memory |
| INTDI | Read into data memory and interrupt |
| INTDO | Write from data memory and interrupt |
| INTII | Read into instruction memory and interrupt |
| INTIO | Write from instruction memory and interrupt |

Branch Instructions

| | |
|---|---|
| BR | Branch |
| BNE | Branch if not equal |
| BEQ | Branch if equal |
| BGE | Branch if greater than or equal |
| BLT | Branch if less than or equal |
| BGT | Branch if greater than zero |
| BLE | Branch if less than or equal |
| BPL | Branch if positive |
| BMI | Branch if negative |
| BHI | Branch if higher (unsigned) |
| BLOS | Branch if lower or same (unsigned) |
| BVC | Branch if overflow clear |
| BVS | Branch if overflow set |
| BHIS/BCC | Branch if higher or same (unsigned) / carry set |
| BLO/BCS | Branch if lower (unsigned) / carry set |

State Register Instructions

| | |
|---|---|
| STn | Store register n into data memory |
| LDnD | Load register n from data memory |
| LDn | Load register n from literal |

Multiply-fetch Instructions

| | |
|---|---|
| MACH | Add high part of product |
| MACL | Add low part of product |
| MSTH | Store high part of product |
| MSTL | Store low part of product |

Loop Control Instructions

| | |
|---|---|
| AOB1 | Increment X1 and branch if non-zero |
| AOB2 | Increment X2 and branch if non-zero |
| SOB1 | Decrement X1 and branch if non-zero |
| SOB2 | Decrement X2 and branch if non-zero |

## APPENDIX B

## Addressing Modes

The following table lists the meaning of the Addressing Mode Register (ADRM). This register is divided into two 4-bit fields called the source mode and the destination mode. Each field selects a pair of addressing modes that a source or destination operand can choose from. The high bit of the instruction address field selects one of the pair.

| | |
|---|---|
| D | The low 6 bits directly address the operand. |
| X1 | THe low 6 bits are added to index register 1 to obtain the address. |
| X2 | The low 6 bits are indexed with index register 2. |
| X1+ | The low 6 bits are indexed with X1, which is then incremented. |
| X2+ | The low 6 bits are indexed with X2, which is then incremented. |
| X1- | The low 6 bits are indexed with X1, which is then decremented. |
| X2- | The low 6 bits are indexed with X2, which is then decremented. |

## Bibliography

[AP-120B] Floating Point Systems Inc., "AP-120B Floating-Point Array Transform Processor," July 1975.

[BLISS] Digital Equipment Corporation, "BLISS-11 Programmer's manual," December 1972.

[FDP] B. Gold et al., "The FDP, a Fast Programmable Signal Processor," IEEE Trans Computers, vol. C-20, pp. 33-38, January 1971.

[LDVT] P. Blankenship et al., "The Lincoln Digital Voice Terminal System," Lincoln Laboratory Technical Note 1975-52, August 1975.

[MECL] W. Blood, MECL System Design Handbook, Motorola Inc., 1971.

[SPS-41] B. Eross and J. Fischer, "SPS User's Manual," Signal Processing Systems, February 1974.