Technical Report CS75014-R

VIRTUAL PROGRAMMING INSTRUMENT EXTENDED HEWLETT-PACKARD [VPI/EHP]

G. W. Gorsline*
G. B. Gorsline**
Phil Scanlon***

May 1975

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

*Professor of Computer Science, VPI&SU, Blacksburg, VA 24061
**Student of Computer Science, VPI&SU, Blacksburg, VA 24061
***Computer Programmer, Department of Computer Science, VPI&SU, Blacksburg, VA 24061

Context of Project

A definition and implementation effort is currently underway to provide an augmented HP2100A processor for use by students and faculty. The main restrictions affecting the design are:

- preservation of the ability to operate in the present hardware/firmware machine mode
- preservation of the ability to operate TSB, DOS-M, DOS-3, and stand-alone software as furnished and maintained by the vendor

Augmentation will include:

- availability of two index registers and a page register (implying both real and virtual pages of size 512 words
- availability of software interrupts in addition to hardware interrupts, both serviced through an extended relocatable equipment table and user created I/O control blocks
- availability of input/output devices as logical references to relocatable, reentrant, pure code device drivers (including appropriate non-device connected interrupt server routines).
- availability of all software modules and associated data areas as completely relocatable, reentrant, recursive, pure code pages.
- the complete separation of logical program addresses from hardware addresses with a firmware translation routine allowing programmatic modes ranging from uniprogramming to virtual memory paging
- programmatic invokation of an alternate WCS module 0' through a VPI&SU hardware/WCS augmentation
- availability of communication to and from the central University computer system (IBM/370: 158-HASP-158) in two modes:
  1. CMS (asynchronous)
  2. HASP quences (making the RJE HP21MX and the advanced 370 SPOOLing abilities available for I/O) (bisynchronous)
- availability of a cross-assembler, a macro-preprocessor, and a cross-micro-assembler for the dynamically alterable and augmented HP machine (executing on the central University system).

- dynamically alterable user defined instruction repertoire through a user micro-instruction mapping table

It is a major design requirement that the following be true:

- the present hardware/firmware definition be either
  1. restorable during normal power up, or
  2. restorable by setting a single manual toggle switch, or
  3. restorable by invoking an I/O instruction supported by special hardware (installation defined, produced, and supported).

- pure, reentrant, and recursive code possible

- unprogramming , multiprogramming (MFT or MVT types), and virtual memory uniform size paging systems possible

HARDWARE - no changes are contemplated (beyond the programmatic WCS module 0 ⇔ 0' switch) although augmentation is always possible (if funding is available).

MICROPROGRAMMING - among the many micro-procedures change needed, the following three have major implications:

1. a new jump-subroutine instruction supporting pure code/re-entrant code/recursive calls (via save state stacks)

2. a new central hardware/software interrupt fielding micro-procedure

3. a replacement instruction fetch phase micro-procedure allowing calculation of the effective hardware address of the operand including logical program indirect addressing, base page, and indexing (pre and/or post). Appropriate calculation of the succeeding instruction address will also be accomplished.

These additions and changes imply at least minor changes to the basic instruction set coding. The operational implications are:

1. power up the HP2100A processor with control store module 0 (ROM) in control;

2. load, using module 0 and the VPI/EHP systems loader, volatile WCS module 0' with the new fetch micro-procedure, new central interrupt fielder, new jump-subroutine instruction, and the altered basic instruction set;

3. the loader will invoke the I/O instruction that programmatically controls the switch making module 0' act as module 0; and

4. then will finally initialize the equipment table and load the user furnished nucleus of the desired software system.

SOFTWARE - All I/O drivers and interrupt servers will be written as pure, reentrant, non-recursive routines completely relocatable as desired by the systems user (usually an operating systems designer). All tables, stacks, queues, buffers, etc. will also be defined as relocatable in a similar manner. Thus, the software contents and their relative locations in primary memory will be completely under operating systems software control. The I/O drivers and interrupt servers may gradually be transferred to micro-code in WCS as the need arises and talent is available. Various programming systems will be needed including cross-assemblers, cross-compilers, primary memory managers, secondary memory access methods, etc.

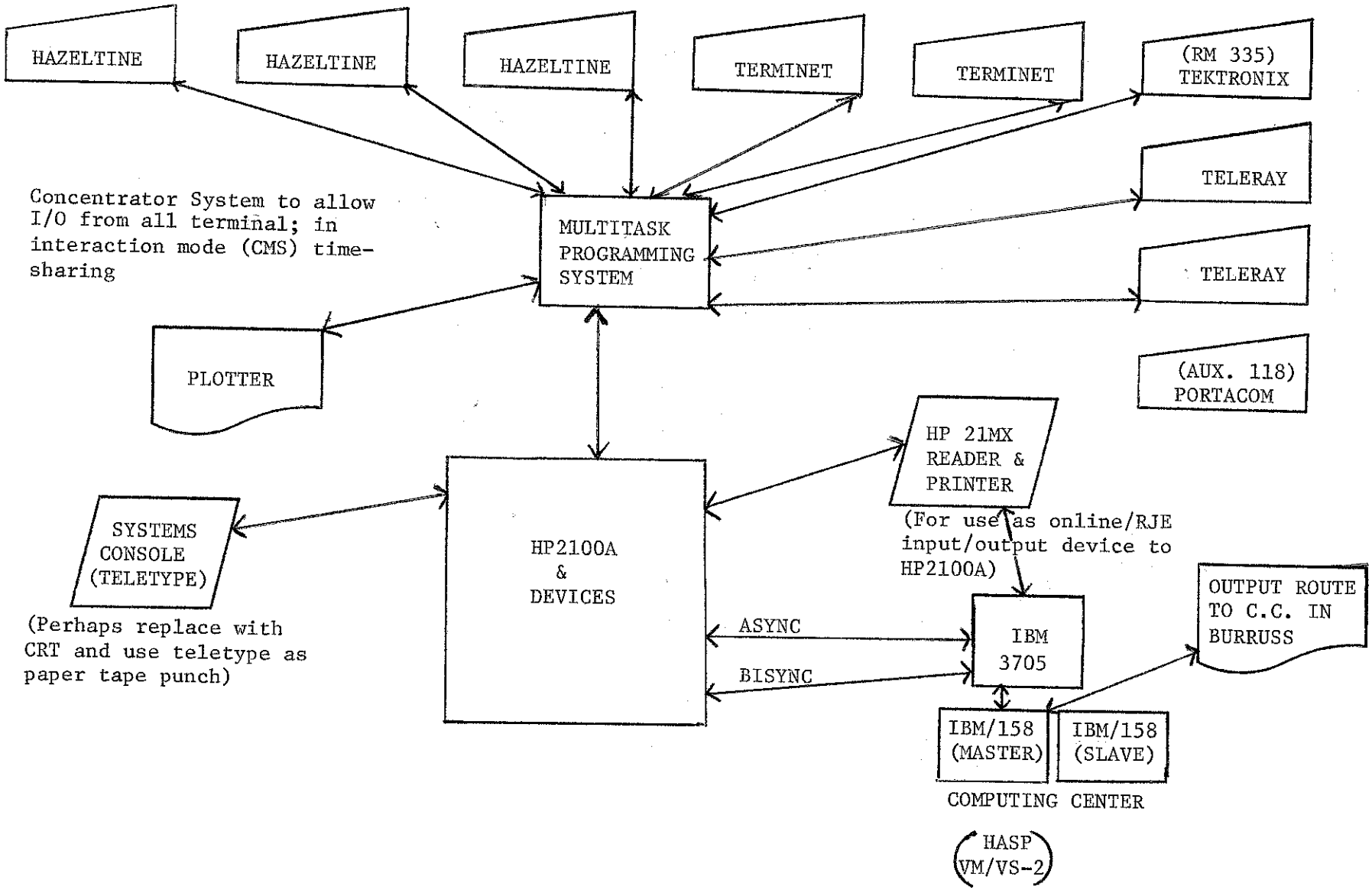
Introduction

The Virtual Programming Instrument/Extended Hewlett Packard is a micro-programmed HP2100A computer with 16K of 16 bit words primary memory as a portion of a standard HP2000E system. The presently planned configuration is specified in figure 1.

The host machine available for microprogramming to emulate the VPI/EHP is graphically depicted in figure 2. 1,024 words of microprogram control storage is available in four modules of 256 words each. Module 0 is Read Only Memory and contains the HP2000E standard basic instruction set as defined and furnished by the vendor. Module 1 is also ROM and contains the 6 floating point instructions as defined and furnished by the vendor. The standard operating systems (TSB, DOS-M, DOS-3, etc) and their associated language processors (BASIC, FORTRAN, ALGOL, etc.) as well as support packages such as utilities (tape to disk, etc.) are supported by the machine as defined by these two modules.

The other two modules of Writable Control Storage (512 words) are available for user microprogramming in a dynamically alterable mode for the definition of new or alternate instructions. These two modules of WCS are being employed to implement an alternate instruction set to emulate a basically different hardware machine. This concept is different from retaining the instructions of modules 0 and 1 and supplementing them with additional instructions. [This concept of supplementation is also a valid and interesting area of investigation]. Rather, the VPI/EHP replaces the instructions of modules 0 with an entirely different set, the instructions of module 1 (floating point) remains available and many prove valuable.

Figure 1: PROPOSED HP2100A SYSTEMS CONFIGURATION



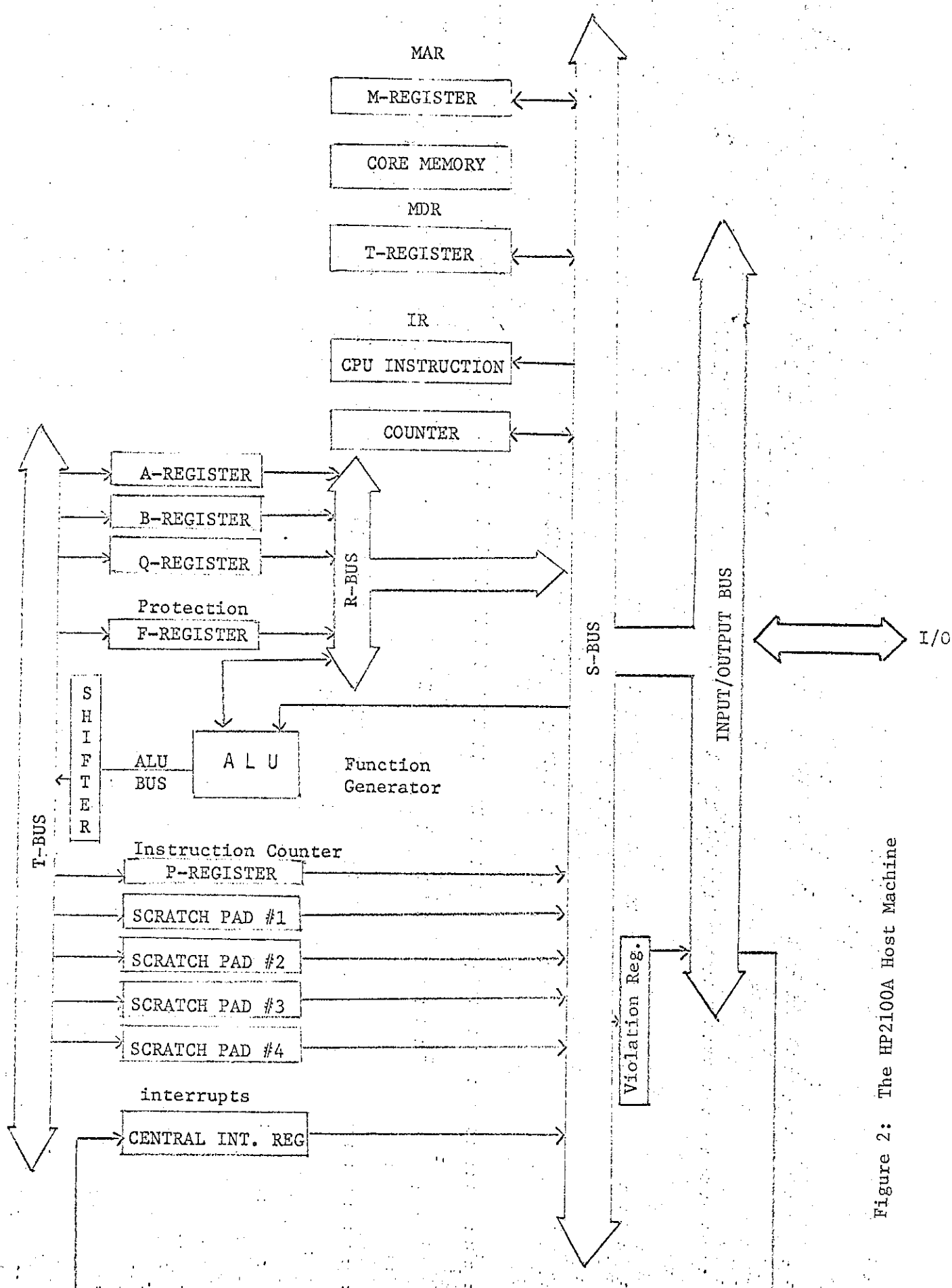


Figure 2: The HP2100A Host Machine

The emulated machine (the VPI/EHP) is paged (equal size) with logical program address space (segment: page: displacement) mapped to real memory space through a microprogrammed instruction fetch phase micro-procedure. Page faults cause an interrupt within the fetch micro-procedure and result in page fetch from secondary memory. The replacement algorithm, I/O call, and all portions of the operating system are left undefined for future development, investigation, and research by interested faculty and/or students.

The raison d'etre of this definition/implementation effort is two fold:

1. to investigate the factors involved in microprogramming a machine design on a host computer not particularly suited for the concept.

2. to provide a vehicle for advanced instruction and for research into virtual memory operating and programming systems as well as the instructions needed for efficiency.

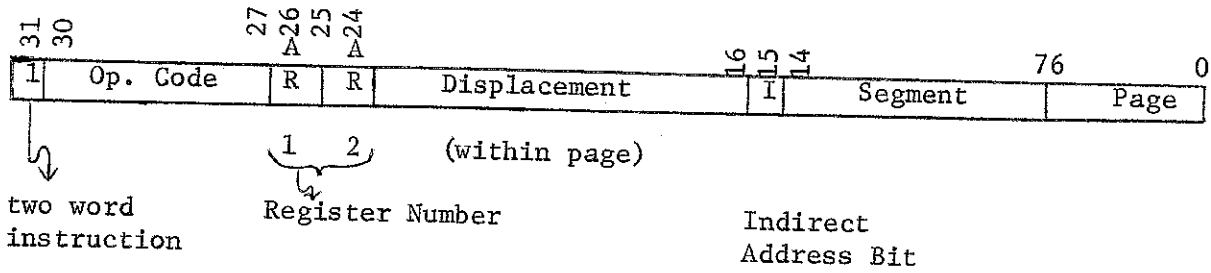
An Overview

The Extended HP as a Virtual Programming Instrument is a microprogrammed HP2100A CPU as a portion of a standard HP2000E system. The VPI/EHP mode is entered by loading WCS modules 0' and 2 with the requisite instructions using the I/O instructions of ROM module 0 and then enabling WCS module 0' as the basic instruction set. ROM module 0 (the standard basic instruction set) is not usuable in the VPI/EHP mode. No permanent changes to the CPU are involved and ROM module 0 can always be enabled by a programmatic or manual switch.

The VPI/EHP is a logically addressable machine possessing several program memories of 16,777,216 words of 16 bits each (program address space = 2^{24} words. Logical program (virtual) memory is implemented as a series of uniform sized pages

(512 words = 2^9) organized as segments distributed under control of an alterable algorithm between the 16,384 words (2^{14}) of randomly addressable core memory (read/write time = 980 manoseconds) and the two direct access movable head disks. Thus, a two level (segment/page) virtual memory is available for each of several programs in a multiprogrammed mode. Note that a maximum of 128 pages are allowed on each of a maximum of 256 segments for each program--or a total of 32,768 pages per program (each of size 512 words).

The format for instructions referencing memory is (OP Codes 0010 through 1111):



If bit 31 is zero, the instruction is one word in length and includes bits 16 to 31 only. In this case the operand is on the current segment/page. Indirect addressing is restricted to two word instructions and is performed after all indexing (pre-indexing), although indexing may also be specified at the succeeding level (post-indexing). Indirect addressing is limited to a chain of length sixteen.

Address constants use the same format as the memory referencing instructions except that bits 27 and 28 refer to index registers B and A respectively. The non-memory referencing instructions and their one word format are exactly the same for the VPI/EHP as for the HP2100A except that all SKIP instructions are changed to JUMP instructions.

Major design features of the VPI/EHP, not available with the HP2000E, are the ability to program in pure code, the availability of reentrant procedures, and the support of recursive external procedure calls. In particular, all interrupt service routines including I/O device drivers are written as fully reentrant pure code. All calls to external procedures result in the automatic stacking of the machine state by firmware.

The relocatable VPI/EHP control stack for each program is a doubly linked list (cell size of 32 words), with the format as given in figure 3. The location is under the control of the operating system software. The current cell of the control stack will contain the psuedo-registers for the VPI/EHP. They are:

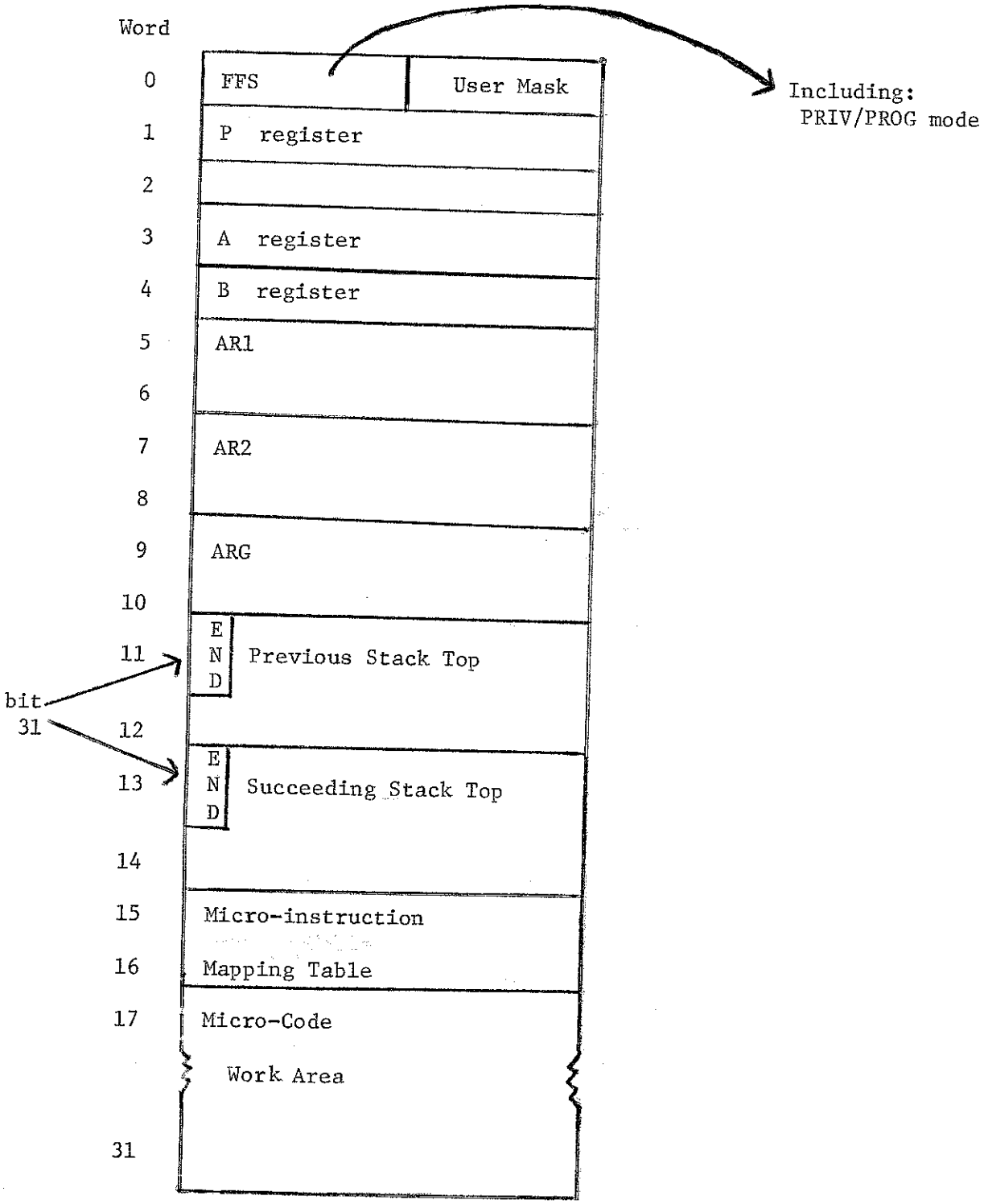
P - register	- 24 bits - Program Counter
A - register	- 16 bits - 'A' Accumulator
B - register	- 16 bits - 'B' Accumulator
AR1	- 24 bits - Address Register 1
AR2	- 24 bits - Address Register 2
ARG	- 24 bits - Argument List Pointer

Note that this results in automatic saving of all psuedo-registers in the control stack at procedure call time as the initiated procedure will cause invocation of a new cell containing new psuedo-registers.

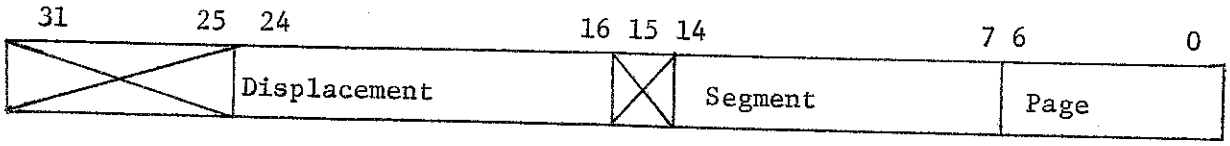
A relocatable interrupt address table is provided to contain the pointers to the service procedures. Three pointers are provided:

1. the address of the appropriate stack
2. the address of the appropriate entry point
3. the address of the appropriate I/O block or work area

Figure 3. Control Stack Format.



The pointer format is that of a double-word instruction with bits 15 and 25-31 being unused:



Each entry requires three double-words of primary memory to contain these three pointers with the exception of the 32 SVC entries which are only two words long. The table is 106 entries long and thus occupies 508 words of memory. The following sections are defined (and constitute a priority of service order):

Entries 1 - 64:	6 words:	Equipment Service Interrupts
Entries 65 - 74:	6 words:	System Faults Interrupts
Entries 75 - 106:	2 words:	Supervisor Services (or software interrupts)

The ten systems faults interrupts are defined as:

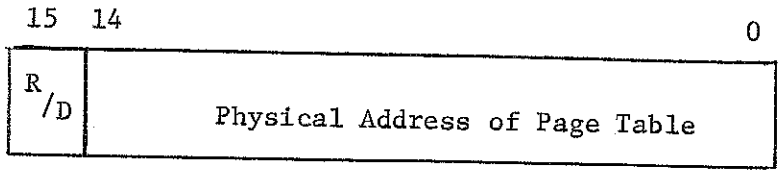
Undefined operation
Priviledged operation
Memory protection violation
Excess Indirect Addressing Count
Page Table Not Resident
Page Not Resident
Bottom of Stack (end)
Top of Stack (return)
Instruction Not in WCS
Anything else (weird conditions)

Note that 32 distinct supervisor service request calls (software interrupts) are provided. The priority order is linear by interrupt number. These supervisor service request calls are completely definable by the operating system designer. The somewhat generous number should allow ample opportunity for investigations concerning types, parsimony, and effects on systems software implementation of variations in software interrupts.

The first 64 entries (equipment service interrupts) are defined by the hardware design and, thus, are not subject to change or priority reordering at software design time. They correspond directly to the hardware I/O 'slots' (or boards) of the HP2000E system. It should be again noted that interrupt service priority is defined by the numerical ordering of this--the I/O interrupt and Fault Entry Table.

Real memory addresses 0 to 127 are reserved for a Micro-Vector Table to: 1) allow fast transfer of control to the various relocatable tables; 2) to provide work space, constants, and pointers for the various micro-procedures that implement the instructions of the VPI/EHP; and 3) to collect statistics on memory referencing. A diagram of these locations is given in figure 4.

A relocatable real memory resident Segment Table must be provided by the operating system designer. Each entry (of 256) of each segment table refers to a page table. The format is:



where the R/D bit in position 15 indicates that the page table is in real memory (=1) or on disk (=0). If the page table is on disk and is referenced, a page table fault will result. The subsequent interrupt will initiate the transfer of the page-table to real memory and the change of the R/D bit to 1 and the 15 bit address field to the appropriate real memory address. During the entire period that a program is active, its segment table must be resident in real memory. Thus, the number of resident segment tables equals the number of active programs (including the operating system). It is possible to define partial segment tables, if desirable.

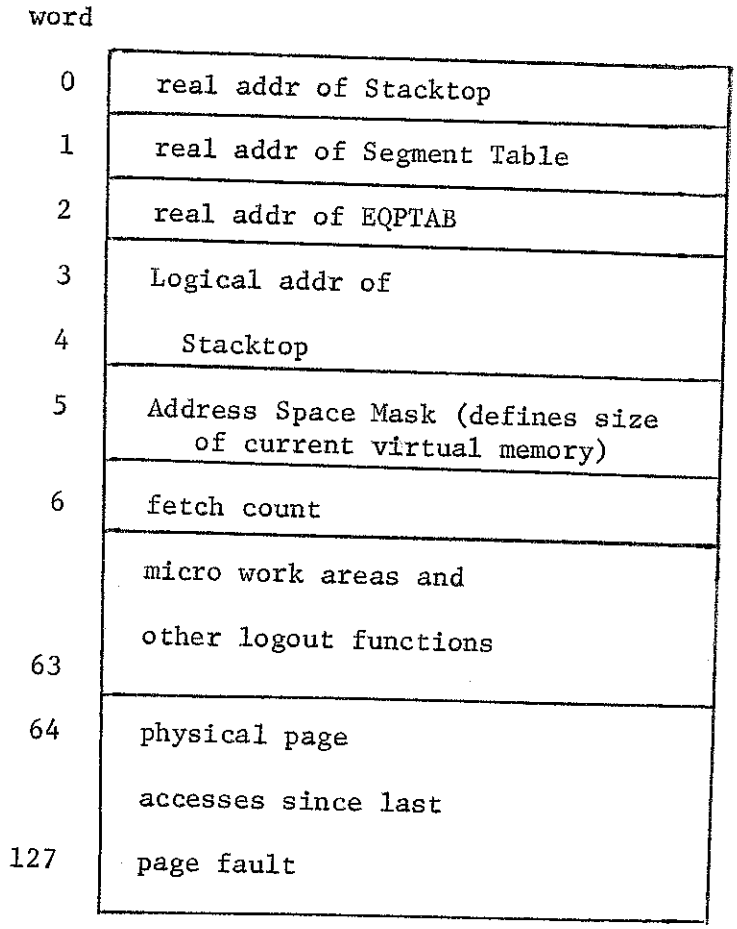
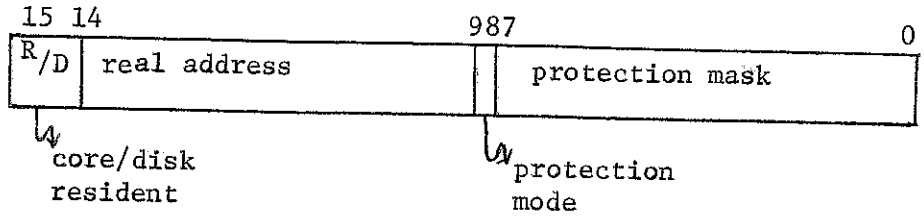


Figure 4. Physical layout of the micro-vector table. Note that this table occupies real locations 0 though 127 while the other tables (the control stack and the I/O interrupt and fault entry table) are relocable at the whim of the software designer.

On the contrary, each page table can be transient in real memory. Four page tables will fit on a page. With 16K real memory and a page size of 512 words, 32 real pages are possible. Real pages 0 to 30 are usable for systems and user programs space; real page 31 is reserved for a binary loader (BDBL) and for an EHP/HP interface systems loader. Six bits are provided for real page identification of which the least significant five bits are concatinated with the nine bit displacement field of the instruction to derive the fourteen

bit real memory address (thus real memory expansion to 32K is possible). If the page is not in real memory, the entire fifteen bits constitute a disk address. The format of each entry of a page table is:



where:

bits 0-7: user protection mask, (only in program mode) tested on each access for legality of access.

bit 8: protection mode (only if protection mask corresponds to user mask)
0 = not accessible
1 = read only

bits 9-14: real memory page address as a power of two over 512

bits 0-14: physical disk address of page if not resident

bit 15: like this same bit in the segment table, indicates if the page is in real memory (=1) or on disk (=0)

INSTRUCTIONS

Six types of instructions are provided by the VPI/EHP computational system:

1. Extended HP Group - These seven instructions are those that use the table definitions that constitutes the extended system. They are:

CSS	Call a Supervisor Service Procedure
CALL	Jump to a Procedure
ARG	Get Address of an Argument
RET	Return from a Procedure or Supervisor Service Procedure
XSTK	Exchange Control Stacks
LP	Load a Pointer
STP	Store a Pointer

2. Memory Reference Group - These twenty-four instructions are divided into three groups.

   A. Single-Register Subgroup - These fourteen instructions include a virtual memory address. If this address is within the currently used page, a one word instruction is possible; if not, a second word contains the segment/page reference. Thirteen of these instructions correspond in action to those of the HP2100A. The other is a subtract (replacing the Jump Sub). The instructions are:

LDA	Load A
LDB	Load B
STA	Store A
STB	Store B
ADA	Add to A
ADB	Add to B
SUB	Subtract from A (two's compliment)
AND	And with A
XOR	Exclusive OR with A
IOR	Inclusive OR with A
JMP	Jump (absolute)
ISZ	Increment memory/jump on zero
CPA	Compare to A/jump if not equal
CPB	Compare to B/jump if not equal

   B. Double-Register Subgroup - These four instructions also include a virtual memory address using the same approach as for the single-register subgroup. These instructions correspond in action to those of the HP2100A. They are:

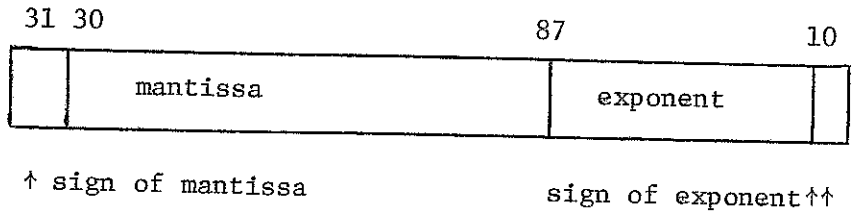
MPY	Multiply by A/result in B + A

DIV	Divides B + A by memory/result in A/remainder in B
DLD	Loads A and B with memory and memory +1
DST	Store A and B with memory and memory +1

C. Floating Point Subgroup - These six instructions also include a virtual memory address as above and correspond to those of the HP2100A. They are:

FAD	Floating Point Add
FSD	Floating Point Subtract
FMP	Floating Point Multiply
FDV	Floating Point Divide
FIX	Convert Floating Point to Integer
FLT	Convert Integer to Floating Point

The floating point datum format is two words in size and corresponds to that of the HP2100A:



3. Register Reference Group - The forty-five register reference instructions are divided into two subgroups: the shift/rotate subgroup and the alter/jump subgroup.

A. Shift/Rotate Subgroup - Of these twenty-two instructions, sixteen operate on a single register while six involve a datum contained in a double-register.

Shift Group

ALS	Left Shift A-register one bit arithmetic
BLS	Left Shift B-register one bit arithmetic
ARS	Right Shift A-register one bit arithmetic
BRS	Right Shift B-register one bit arithmetic
ALR	Left Shift A-register one bit and clear sign bit
BLR	Left Shift B-register one bit and clear sign bit
ASR	Arithmetic Right Shift N bits/double register
ASL	Arithmetic Left Shift N bits/double register
LSR	Logical Right Shift N bits/double register
LSL	Logical Left Shift N bits/double register

Rotate Group

RAL	Left Rotate A-register one bit logical
RBL	Left Rotate B-register one bit logical

RAR	Right Rotate A-register one bit logical
RBR	Right Rotate B-register one bit logical
ELA	Left Rotate A- and E-registers one bit logical
ELB	Left Rotate B- and E-registers one bit logical
ERA	Right Rotate A- and E-registers one bit logical
ERB	Right Rotate B- and E-registers one bit logical
ALF	Left Rotate A-register four bits logical
BLF	Left Rotate B-register four bits logical
RRR	Right Rotate B- and A-registers N bits logical
RRL	Left Rotate B- and A-registers N bits logical

B. Alter/Jump Subgroup - Of these twenty-three instructions, eleven are set/clear registers; two are increment registers; nine are conditional jump; and one is a reverse jump condition (or absolute jump).

Clear Register Group

CLA	Set A-register to zero
CLB	Set B-register to zero
CLE	Set E-register to zero
CMA	Ones Compliment A-register
CMB	Ones Compliment B-register
CME	Ones Compliment E-register
CCA	Set A-register to all ones
CCB	Set B-register to all ones
CCE	Set E-register to all ones
STO	Set Overflow Flip/Flop to one
CLO	Clear Overflow Flip/Flop to zero

Increment Register Group

INA	Add one to A-register
INB	Add one to B-register

Conditional Jump Group

JZA	Jump if A-register is zero
JZB	Jump if B-register is zero
JEZ	Jump if E-register is zero
JSA	Jump if A-register is positive
JSB	Jump is B-register is positive
JLA	Jump is A-register is even
JLB	Jump if B-register is even
JOS	Jump if Overflow Flip/Flop is one
JOC	Jump if Overflow Flip/Flop is zero

Reverse Condition Test

RJC	Used in conjunction with the conditional jump group to reverse the test--absolute jump when used alone.

4. Input/Output Group - This group of instructions operates to transfer information from external devices and either:

   the A- or B- registers, or
   Direct to Memory (DMA).

   The use of registers for I/O is limited to one word or byte of data The use of DMA allows transfer of data in any size block (maximum = 32 K). Both types of I/O use a subset of the same basic twelve instructions.

STF	Set Flag of I/O channel or function
CLF	Clear Flag of I/O channel or function
JFS	Jump if I/O Flag is set
JFC	Jump if I/O Flag is clear
MIA	IOR I/O buffer of device with A-register
MIB	IOR I/O buffer of device with B-register
LIA	Load I/O buffer of device into A-register
LIB	Load I/O buffer of device into B-register
OTA	Load A-register into I/O buffer of device
OTB	Load B-register into I/O buffer of device
STC	Set control bit of I/O channel or device
CLC	Clear control bit of I/O channel or device (turns device off-00 is all devices)

5. Systems Control Group - The two instructions in this group are:

NOP	No Operation (16 bits zero) Allows space to be saved for instruction alteration or data psuedo-ops
HLT	Halts the System. Has the same effect as the console HALT push button. (priviledged)

6. MAC Group - As the HP2100A processor was designed to maintain software and device compatability with the predecessor HP 2114, 2115, and 2116 processors, some of the controls are hardware generated and some are micro-programmed. The VPI/EHP allows user microprogramming of instructions supplemental to those described above. Space exists in WCS for nine additional instructions which can be expanded by a fairly large factor if necessary (through jump tables).

Appendix 1

VPI/EHP Instructions

1. CSS: Call Supervisor Service

   CSS <number>, <data>

105140B	HP MAC GROUP
	<number>
address constant	<data>

   <number>: the left 5 bits are used as the requested service number.

   <data> : address of the argument list. Loaded into ARG register upon entry to CSS instruction. Enters CSS in priviledged state, mask = 'ff'.

2. CALL: Call a procedure

   CALL <label> <data>

105141B	HP MAC GROUP
addr con	<label>
addr con	<data>

   <label>: the address of where to start executing the procedure

   <data> : address of the Argument List

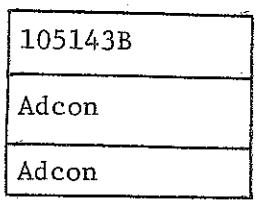
3. RET: Return from a procedure or supervisor service

   RET

105142B	HP MAC GROUP

4. XSTK: Exchange Control Stacks

XSTK <Data 1>, <Data 2>

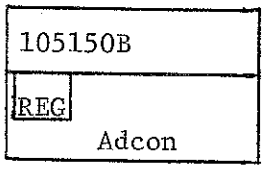


HP MAC GROUP
<Data 1>
<Data 2>

A priviledge instruction that stores the current stack pointer register at <Data 1> and loads <Data 2> as the new stack pointer register.

5. LP: Load Pointer

LP <REG>, <Data>



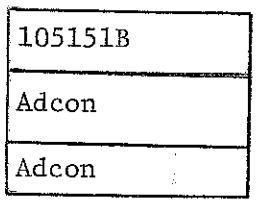
HP MAC GROUP
<Data>

The two bits not used in the Adcon specify which register is to be loaded with <Data> as below:

00:A left truncated to 16 bits
01:B left truncated to 16 bits
10:AR1
11:AR2

6. STP: Store Pointer

STP <Data 1>, <Data 2>

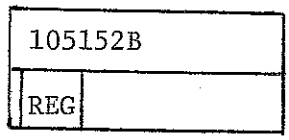


HP MAC GROUP
<Data 1>
<Data 2>

Stores <Data 1> at <Data 2>

7. ARG: Get Argument Address

ARG <REG> <Data>



HP MAC GROUP

<Data>

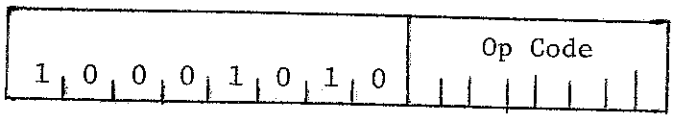
The 16 bit <data> field contains the argument number. If the datum is 0, the real address of the argument list is loaded into the register. If the datum is 1, 2, 3, ..., 256; the real address of the corresponding argument is loaded into the specified register. Bit 30 of the ADCON is 'on' for the last argument and 'off' for all others.

Appendix 2

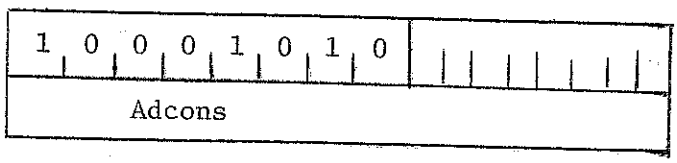
Instruction Formats

1. EHPG Extended HP Group

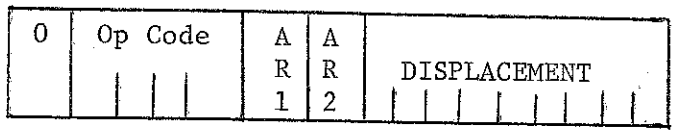
a.



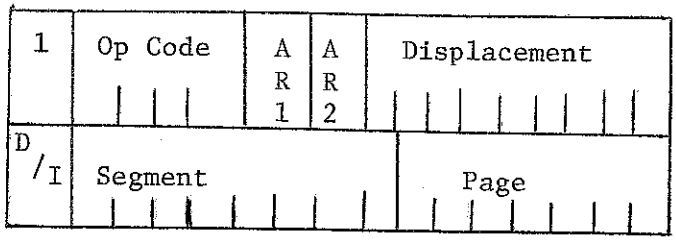
b.



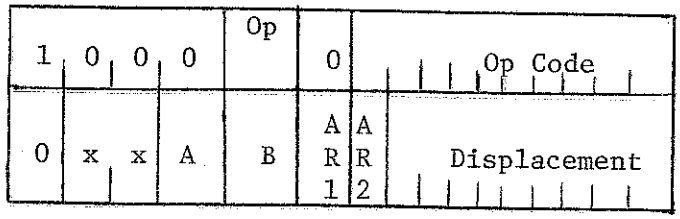
2A 1. MRG Memory Reference Group (Single Register)



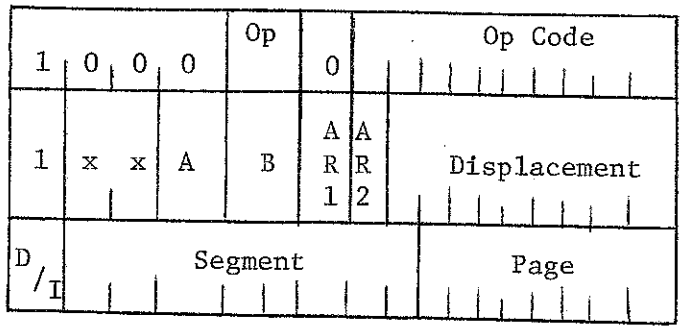
2.



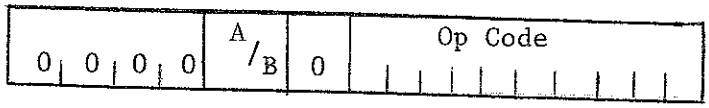
2B 1. MRG Memory Reference Group (Double Register)



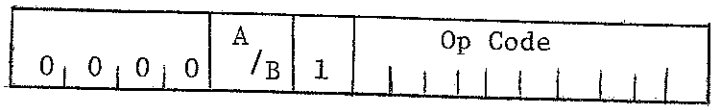
2.



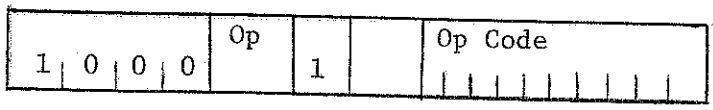
3A. RRG Register Reference Group (Shift/Rotate)



3B. RRG Register Reference Group (Alter/Jump)

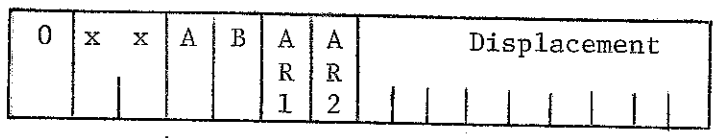


4. IOG Input/Output Group

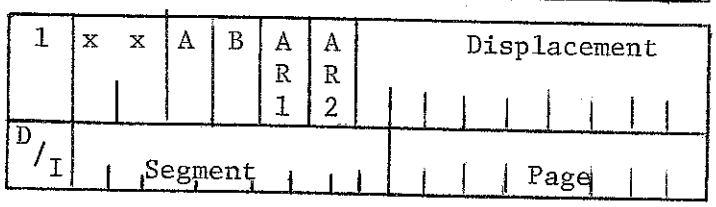


5. Adcon

a.



b.



The opcodes are identical to those listed in the HP2100A reference guide except on page 18 the JSB opcode is a two's compliment subtract from the A register. The extended system group is supplemental to the HP2100A system.

SUMMARY

A design is presented for the VPI/EHP--an augmentation of the HP2100A through microprogramming to produce a Virtual Programming Instrument/Extended HP. Writable Control Store module 0' will contain the eighty-three normal instructions of the HP2100A except that the jump-sub instruction will be replaced by a two's-compliment subtract memory from the A-register. In addition, seven new instructions support pure reentrant code and recursive subprogram calls. Thus ninety instructions are defined. Thirty-two software interrupts (supervisor calls) are made available in addition to sixty-four device and ten system faults interrupts.

The firmware implemented architecture is designed to support a multiprogramming operating system in the virtual memory philosophy. A uniform sized page with a resident segment table allows each "active" program to possess an addressing space of 16,777,216 words or 33,554,432 bytes. Each page is 512 words (1024 bytes) in size; each segment contains up to 128 pages (65,536 words=131,072 bytes); each program can have up to 256 segments (16 megawords=33 megabytes). A new instruction fetch micro-procedure calculates all effective hardware addresses, including indirect addressing and both pre-indexing and/or post-indexing. Subprocedure calls result in automatic stacking of the machine state while a return restores the previous machine state. In effect, this VPI/EHP instruction fetch micro-procedure is a micro-programmed dynamic address translation (DAT BOX) plus page fault and page replacement algorithm (with the replacement plan alterable by the operating system).

Communications to and from the central University IBM/370 158-HASP-158 system is provided to CMS via dial-up asynchronous communication and to the HASP queues via dial-up bisynchronous communication. The local HP 21MX based

HASP RJE station is thus available for I/O by the VPI/EHP as are all the other peripherals of the central system and the associated network.

The purpose of this design and implementation is to provide an architecture suitable for experimentation in modern programming systems. Needed software includes:

Device Drivers (physical and logical)

Data Management Systems

File Management Procedures

Compilers and Assemblers (VPI/EHP resident and cross)

Operating Systems