INTRODUCTION

The principal classifying feature of the PB 440 is its microprogrammed character. Like other so-called “general purpose” computers, it derives control from a program of instructions or orders which has been previously prepared and stored within the computer memory. Unlike most general purpose computers, however, the PB 440 programmer is permitted to direct the computer at a more basic level of control by logically manipulating the contents of individual registers and flip flops.

Compared to the more sophisticated (and more restrictive) manipulative capabilities of conventional digital computers, the PB 440 instructions may properly be termed “micro-orders”. In accomplishing a given data processing operation, an appropriately programmed series of these micro-orders would be executed by the PB 440. A more conventional computer would achieve the same result by activating a succession of logical circuits in a fixed sequence which had been determined by the computer designer. Thus, an important consequence of this micro-programming (or “stored logic”) feature is the relative freedom of the programmer to select or devise (macro) instruction sets which are appropriate to his particular problem, rather than being limited to a single set which was compromised by the designer in order to accommodate a wide variety of applications. The logical organization of a computer which provides this feature and to a lesser extent, its programming implications, are the topics of this paper.

Before elaborating on the design, several comments might be made for the benefit of the serious student of microprogramming. The authors’ use of the term “microprogramming” appears to agree with the definitions introduced and employed by Wilkes and Glantz2 and with the one pulse time criterion referred to by Mercer4 in discussing the elementary nature of micro-orders. A review of the work of earlier authors also indicates that the non-destructively read memory module, used in the PB 440 for the storage of micro-order sequences, replaces the earlier concept of a wired microprogram memory.1 It provides the capability sought by Glantz5, for his sub-command Sequence Memory, and later by Graselli6, for the Control memory in his “conventional microprogrammed control unit.” Finally, it should be stated that, while the virtues and merits of a microprogrammed philosophy of machine organization, expressed by previous workers, influenced the decision to organize the PB 440 control unit in this fashion, following the adoption of the basic philosophy the details of implementation developed naturally and independently.

FUNCTIONAL CHARACTERISTICS

The principal characteristics of the PB 440 are introduced in Table 1. A maximum of 64
TABLE 1
PB 440 CHARACTERISTICS

General Characteristics
Computer type: micro-order, stored program
Data handling mode: parallel
Internal number system: binary
Timing: synchronous, 1 megacycle logic clock rate

Programming Characteristics
Instruction format: variable, determined by micro-program (64 micro-orders)
Word lengths: micro-orders—12 bits
data word length and format—determined by micro-program
Index registers: as required by instruction format
Programmable registers: 4 processing unit registers functionally defined by micro-program

Memory Characteristics
Main Memory: 4096 words, 24 bits, random access, 5 microsecond cycle time, expandable to 28,672 words in modules of 4096 words each
Fast Memory: 256 words, 24 bits; random access non-destructive read, 1 microsecond read cycle time, expandable in modules of 256 words each

Input-Output Equipment Options
Typewriter, 15 characters/sec.
Paper tape reader, 500 characters/sec.
Paper tape punch, 110 characters/sec.
Card reader, 800 cards/minute
Card punch, 250 cards/minute
Magnetic tape units 15 KC to 83.3 KC (IBM compatible)
Line printer, 1000 lines/minute
Also provision for:
A-D converters, plotters, display equipment, communication links, hybrid system linkage, mass storage devices, other computers

distinct micro-order types are decoded by the PB 440 control unit for manipulative and control purposes. In addition to an identifying 6-bit micro-order code, a 6-bit modifier field is contained in each micro-order. This basic format is illustrated in Figure 1, where two micro-orders are shown occupying a 24-bit memory word. The modifier fields have been further divided into two 3-bit fields designated R1 and R2. Although the composite 6-bit field has a different significance for some micro-orders, generally, the R1 and R2 octal digits each refer to one of seven hardware registers whose contents may be manipulated in a manner specified by the associated micro code.

The ability to perform logical manipulations on the contents of selected registers permits operations to be conducted on data whose length and number form are determined by the programmer. It was recognized, however, that the execution times resulting from the use of such general capabilities can be improved upon for one or more common types of data formats, if special micro-orders tailored to these formats are included in the design. Character manipulation and arithmetic operations on signed numbers and floating point numbers are three cases which occur so frequently that this type of special treatment is in order. The data formats illustrated in Fig. 1 were selected for these purposes, and it will be noted below during an examination of the micro-order repertoire that operations on data expressed in one of these formats have been facilitated.

A representative number of PB 440 micro-orders have been grouped in Table 2 according to their function. A few general remarks will serve to illustrate the use of each group. The register manipulative operations of the first group may be applied to any of the addressable registers, permitting the role of a register to vary from operation to operation as determined by the programmer. Identified in this group is the ability to manipulate particular fields of data which facilitate the programming of operations on numbers expressed in either the sign-magnitude or floating point formats presented earlier. Like the register manipula-

![Figure 1. PB 440 Basic Word Formats.](From the collection of the Computer History Museum (www.computerhistory.org))
The memory operations further reflect the microprogrammed nature of the computer in that any of the addressable registers may be used to supply address information and to receive or supply data, as required by the operation. Also, eight working storage locations may be addressed directly by micro-order without reference to an address register. The Load Special micro-order (LDS) permits an entry in one of several 64-word tables to be obtained from memory on the basis of a 6-bit partial address in the D register. This special purpose order facilitates the interpretation of pseudo instruction operation codes as micro-routine starting addresses. It also permits the rapid translation of one alphanumeric code to another during input/output operations.

The arithmetic operations are generally self-explanatory. As was the case with the register manipulations above, the significance of particular fields of data in the sign-magnitude and floating point formats has been recognized and accommodated. The need to accomplish the important operations of multiplication and division rapidly led to the specialized Multiply Step and Divide Step micro-orders.

The group of logical operations may be applied to any pair of registers and may be conducted in conjunction with a data transfer. The operations of the shifting group provide a very general shifting capability which can be applied to any register. The shift commands include micro-orders for the purpose of normalizing and equalizing numbers during floating point processing.

The arithmetic operations are generally self-explanatory. As was the case with the register manipulations above, the significance of particular fields of data in the sign-magnitude and floating point formats has been recognized and accommodated. The need to accomplish the important operations of multiplication and division rapidly led to the specialized Multiply Step and Divide Step micro-orders.
In the category Skip and Jump commands, several micro-orders have been collected under a group name which reflects their common feature, that of directing program control to a new sequence of micro-orders. It will be seen that the ability to specify the P register in the R2 field of a register “copy” operation provides the most general means for achieving a jump in the program sequence. The relative transfer operations (FTR, BTR) of this group accomplish a similar result by modifying the program counter (P register) contents rather than substituting a completely new address. The test micro-orders (TZO, TNZ, TCT, TCF) permit the programmer to test a variety of logical conditions and skip a following micro-order on the basis of either condition met or condition not met.

A final group of micro-orders, not included in Table 2, permits programmed control over input/output equipment. They provide a means for issuing commands to selected external devices, controlling the operation of an I/O channel commutator, and conducting data transfer operations with previously activated devices. The variety of data transfer modes which may be programmed using these micro-orders include:

1. single character input/output
   (a) wait for device ready
   (b) test for device ready
   (c) interrupt program when device ready
2. data block transfer
   (a) uninterrupted transfer at device data rate
   (b) buffered transfer to or from a pre-assigned area of storage on an “interrupt program when device ready” basis.

PROCESSING UNIT

The processing unit consists of four 24-bit registers, two 15-bit registers and a single 8-bit register. For purposes of register transfers and other manipulations, the registers are aligned logically as indicated in Fig. 2. In addition to their general programmed use, two registers have fixed roles in internal operations:

1. The P-register serves as the microprogram counter, being incremented automatically and submitting its contents as a memory address during each micro-order-pair access cycle.
2. The N-register serves as the repeat counter for timing shift operations, and multiply-step and divide-step micro-orders.

It will be noted in Fig. 2 that the N-register is located in the bit positions corresponding to the exponent field of a floating point format word; and that the L-register, being of maximum address length can be used to good effect as a macro-instruction location counter.

The Processing Unit Block Diagram, Fig. 3, illustrates the manner in which the addressable registers of the computer may be connected to a binary full adder and other manipulative logic by use of the processing unit bus structure. Under the control of decoded micro-orders, the registers may be selectively gated onto the logic input busses (1 and 2), the desired manipulative logic enabled, and the result gated from the appropriate logic output bus (3 or 4) into the desired register. Thus, the addition of the contents of registers A and B may be accomplished by logically connecting A to bus 1, B to bus 2, enabling the full add logic and connecting bus 3 (carrying the binary sum) to the input logic of register B. Similarly, the 1's complement of the contents of register L may be obtained by connecting the outputs of L to bus 2, selecting the inversion logic input to bus 4, and connecting bus 4 to the register L input logic. Data transfers and register exchanges are facilitated by a direct connection between busses 2 and 4 and by the ability to disable the bus 2 input to the full add logic.*

* The PB 440 bus structure is reminiscent of the busses of Kampe's arithmetic unit (reference 5) but gains significant advantages from placing manipulative logic between register output and input busses.

Figure 2. Processing Unit Registers.
It should be noted on Fig. 3 that buses 1, 2, and 4 and their associated gating logic also provide the means for communication with the memory system, the input/output devices, and the operator's console. During the execution of a micro-order requiring memory operation, for example, the register designated for supplying the address will be gated onto bus 1 and bus 1 will be selected as the input to the memory address bus. For a memory "read" operation, the contents of the memory data output bus will be gated onto bus 4 and thence to the selected destination register. For a memory "write" operation, the data source register will be selected as an input to bus 2 which in turn will supply the memory data input bus.

During an input operation, bus 4 may be connected to the computer input bus which carries both data and status information from external devices. Similarly, data and control information may be gated from the register designated by an output micro-order to an external device via bus 2 and the computer output bus.

Access to the memory system and the addressable registers of the processing unit, from the operator's console, may be accomplished efficiently by making use of existing bus structure logic. Bus 4 may be monitored as a common point for displaying both memory data and register contents. Manual memory interrogations may be conducted from the console by supplying address data as an input to the memory address bus and by selecting the memory data out bus as an input to bus 4. For storage operations, data from a "register" of switches at the console may be gated onto bus 2 and thence to the memory data input bus. The contents of processing unit registers may be displayed by use of bus 2 and the direct input to bus 4. The same logical path may be used to alter the contents of a specified register by selecting the console data switches as an input to bus 2, and bus 4 as an input to the designated register.

The logical details of the processing unit bus structure may now be considered. Illustrated from left to right in Fig. 4 are:

1. the X-register and its associated logic for decoding the micro-order type and the register codes (R1, R2) of the micro-order which is to be executed at the next logic clock;
2. the general bit position, i, of each of the processing unit registers which may be gated onto bus 1 and/or bus 2 as dictated by control signals derived from the micro-order type and register codes;
(3) the principal types of arithmetic and manipulative logic which may be applied selectively to the register output busses; and

(4) the register input busses (3 and 4), which may be selectively gated into the general bit position, i, of each of the processing unit registers; again, under the control of the current micro-order type and register code.

Not indicated on Fig. 4, but important to an understanding of computer operation, is the logic for several other processing unit functions. These include:

(1) a variation on the binary addition logic which permits data on bus 1 to be incremented or decremented;

(2) logic to increment the contents of the P-register (program counter);

(3) logic to decrement the contents of the N-register (repeat counter);

(4) double length shifting logic on registers A and B; and

(5) the logic for applying a variety of tests to the data being transmitted over the general bus structure.

Apparent on Fig. 4 are several features which are significant in the mechanization of the processing unit logic described above. It will be noted, for example, that the flip-flops of the 7 programmable registers need to supply only normal (or true) outputs to the register selection logic at the inputs to busses 1 and 2. The complement form of the selected register's contents is made available to the manipulative logic by inverting amplifiers on the busses. The bus 2 amplifiers supply not only the logic associated with the full adder, but also the variety of functions which are mechanized at the input to bus 4.

An equally important feature is the mechanization of the flip-flop input logic which is permitted by the use of single input, DELAY flip-flops. The logical properties of the "D" flip-flop require a "one" input at the beginning of each clock interval during which a "one" output is desired, and cause a return to the zero state if no input is present. This eliminates the need for constructing the "reset" portion of conventional set/reset logic, and further eliminates the requirement that register input busses 3 and 4 be implemented to carry complement information.
Finally, as an indication of required logic circuit performance, Fig. 4 identifies the delay producing elements in both data and control paths. The accumulated effect of these individual time delays is to establish the upper limit of the logic clock frequency. More specifically, a 12-bit micro-order will be inserted into the left half of the X-register (left side of Fig. 4) at one logic clock time, and the voltage levels produced by the register input logic (right side of Fig. 4) will establish the new states of the register flip-flops at the next logic clock time. In a typical operation (ADD, e.g.), the delays encountered are, in order of occurrence:

1. decoding and driver delay for micro-order type and register selection controls;
2. logic circuit delay at bus 1 and 2 input gating;
3. bus 1 and 2 driver delays;
4. carry propagation time through full add logic;
5. bus 3 driver delay; and
6. logic circuit delay at register input gating.

The delay times experienced with PB 440 production circuitry show an average delay of 25-40 nanoseconds through AND-OR-AND-OR diode logic and a bus amplifier, and 300 nanoseconds maximum carry propagation time in the full adder. A logic clock frequency of 1 megacycle was found to provide for worst case decoding and signal paths with adequate margin and still allow a required 200 nanoseconds settling time for flip-flop input circuitry.

As a comment on the basic nature of the micro-order repertoire, all non-repetitive micro-orders require a single clock time for their execution. That is, a binary addition of two 24-bit registers is accomplished in one microsecond with the result replacing one of the operands.

MEMORY SYSTEM

The use of memory input, output and address busses in processing unit communications with the memory system was introduced in Fig. 3 above. The technique used in connecting individual memory modules to these busses is illustrated in Fig. 5, and will be described further following some general remarks concerning this method of memory communication.

One of the most significant features of the bus type memory communication is that it facilitates the modular expansion of the memory system. As indicated in Fig. 5, up to eight modules of 4096 words each may be accommodated without altering the logic of the basic computer design. Memory control logic, which treats each module as a separate asynchronous device, is a second important feature which has the following advantages:

1. Memory modules of different access and cycle times may be accommodated in the system without complicating the control logic or introducing problems into the process of program storage allocation.
2. The ability to initiate a memory read operation in one module while a rewrite (restore) operation is being conducted in a previously addressed module creates the opportunity to store program instructions or data strings alternately in the two modules and, as a consequence, to realize a significant reduction in program execution time.
3. As memory modules of improved performance are developed they can be used to replace the slower original equipment, thus prolonging the competitive life of the computer.

Item (1) is particularly important to the dual-speed memory concept employed in the
PB 440. It permits the combined use of a fast, nondestructively read memory unit containing microprogrammed stored logic, and a larger capacity core memory containing macro-instructions, operands and constants. In addition to allowing overlapped operation of core modules, item (2) permits several accesses to the fast memory and the continued execution of the microprogram during the rewrite portion of each main memory cycle.

The connection of the Exchange Register and Address Register of the individual modules to the memory busses is enabled by address selection and control logic, as shown in Fig. 5. The memory address bus and Read/Write control lines appear there as inputs to each memory module. The three most significant bits of the 15-bit address designate one of eight possible modules and the remaining 12 bits specify a particular one of its 4096 words.

One code is assigned to Fast memory and the remaining codes are used in designating up to 7 magnetic core memory modules. Module selection requires that in addition to the presence of a Read or Write request and the appropriate module code, the indicated module must be in a “not busy” condition. In requesting a memory operation, the computer control logic maintains the address information on the bus until an acknowledgement is received which indicates acceptance. In the event that the designated module is currently conducting a write cycle, the busy indication prevents the initiation of the requested operation until the cycle is completed.

Referring to Fig. 5, the output of the module selection logic gates the address data into the module address register and, in the case of a write operation, also connects the memory data input bus to the module exchange register. At this point in the write cycle timing a “Write Release” signal is sent to the computer control logic which permits continued (or renewed) program execution. At the conclusion of a read cycle a “Read Complete” signal is generated which connects the module exchange register to the memory data output bus and indicates to the processing unit that data is ready.

In order to insure that the data is not sampled while in transition between logic levels, some form of synchronization with the logic clock is required. This interlock is achieved by a “Data Ready” flip-flop which responds to the completion signal sufficiently in advance of the next logic clock to guarantee that the accompanying memory data will be settled at clock time. The Data Ready flip-flop also provides an echo to the selected module and a release signal to the computer control logic. The computer is thus not permitted to initiate a memory read operation before receiving the data ready signal terminating a previous read operation. At the left of Fig. 5, two possible paths are shown for memory output data. The direct path to the micro-order decoding register is enabled at the time in each computer cycle when the next micro-order pair is presented to the control circuitry. The second path, which uses bus 4, is typically employed to gate the results of an operand access into the register specified by the micro-order currently being executed.

The PB 440 standard memory module (4096 words of 25 bits) is designed to be connected in a tandem arrangement to the memory busses of a single computer. Certain classes of applications however require that one or more modules which comprise the memory system of the computer be shared with other devices. These applications can be accommodated by employing a Memory Interchange Unit to sample Read/Write requests from the several devices and to connect the device’s address and data busses to those of the memory module(s) being shared for the duration of a single operation. From 1-8 memory modules may thus be shared by a maximum of 4 devices as shown in Fig. 8. In order to permit two or more PB 440’s to share a common portion of their memory as described above, it is only necessary to connect the “Remote Memory” output jacks of each of the individual computers to a set of the Memory Interchange input jacks.

INPUT/OUTPUT SYSTEM

The input/output system and its relationship to the units already described is illustrated in the Input/Output System Block Diagram of Fig. 6.

Communication with the peripheral equipments is achieved by controlling their connec-
The types of devices which may be so connected to the computer appear in Fig. 6 as individual blocks, distributed along the length of the input/output bus. The electronic connection of these equipments to the bus is accomplished by the use of a device controller which is unique for each equipment type and which permits the connection of up to 4 devices. A controller contains sufficient logic:

1. to sample input/output bus control signals and determine when one of its devices is being "addressed" by the computer;
2. to connect the device to the bus for the purpose of receiving commands or transferring data; and
3. to maintain the device in a previously commanded mode of operation until either further commands are received or a predetermined condition occurs.

A controller also provides character or word buffering, appropriate to the device, and accomplishes any necessary error detection. The PB 440 input/output system is expandable to 64 controllers, each with 4 devices.

The basic control techniques employed in the logical mechanization of the input/output bus permit a variety of communication modes to be programmed. These include the ability to conduct up to four independent data block transfer operations, with assignable priorities, on a program interrupt basis. External devices with data rates on the order of 80,000 characters/second can be accommodated by this means. The uninterrupted transfer of data to or from one program-selected device can be carried on at character rates up to 400 KC.

The details of implementing external device communications are presented in Fig. 7 which identifies the data and control lines that compose the input/output bus. Also, shown schematically, are the data connections to busses 2 and 4 and the logic required by the program interrupt feature. On the input side of the bus, the 24 data or status lines are gated on to bus 4, under the control of an input type micro-order, for distribution to the program selected register. The transfer of a command or data word is similarly conducted on the output side of the bus by gating the information from bus 2 under the control of an output type micro-order.

The four control lines, shown in the central portion of Fig. 7, serve to coordinate the transfer of information on the bus. The presence of a Command Transfer Control Signal identifies the information currently on the output lines as a command word. In addition to specifying a controller and a device number, a command word contains a buffered channel assignment.
to be used by the selected device in requesting subsequent data transfers. In responding to such a command, a selected device energizes the Data/Command Transfer Echo line and places 24 bits of status information on the input lines.

If, at the time a device is activated, the programmer elects to be interrupted when the device is ready for a transfer of data, processing continues until a signal is recognized on one of the buffered channel request lines. The buffered channel commutator halts automatically at this line, the program is interrupted, and control passes to a stored logic sequence which prepares to receive or transmit data into or from a preassigned memory location. Under stored logic control the locked position of the commutator is transmitted as a buffered channel enable signal to the interrupting device. A Data Transfer Control Signal simultaneously clocks data out of the computer for an output transfer, or in the case of an input transfer, indicates the computer's state of readiness. The transfer is terminated by the computer upon the recognition of a Data Transfer Echo, the commutator is unlocked (or reset) and control is returned to the interrupted program.

In contrast to the somewhat automatic features of the above described data transfer, at least two other modes of input/output communication are possible. The programmer may elect to devote the computer on a full time, uninterrupted basis to a data block transfer with a selected device. In this mode, the commutator is locked on the desired buffered channel under program control and a data transfer control signal is issued. The computer is thereby halted until the completion of the data transfer is indicated by the receipt of a data transfer echo. This process is repeated until the specified number of words has been transferred.

A third mode makes use of interrupt masking flip-flops, which may be set and tested under program control. They provide a means of masking buffered channel request signals at the input to the commutator but at the same time permit the request lines to be tested by the program for possible action. Thus, the readiness of the external device may be tested before executing a data transfer command and setting the input/output interlock.

A final input/output feature, denoted “special interrupt” in Fig. 7, permits the computer to respond to external events that were anticipated but not initiated by the program. This type of interrupt may correspond to the pressing of a typewriter key by the operator or the occurrence of some singular real time event. It may be tested periodically as a basis for transferring control to a program which will determine the cause of the alarm by interrogating each of the devices known to communicate over the special interrupt line. A data transfer can then be scheduled by assigning a buffered channel to the interrupting device in the manner described above.

These basic input/output capabilities can be extended by the use of an I/O adapter unit which permits direct buffering of data in and out of memory at character rates up to 800 KC. In Fig. 8, the address and data registers of such a direct channel are shown connected to a memory module which is shared with the computer. During system assembly this adapter is connected between the computer I/O bus and the controller for the selected device (a separate adapter is required for each device). The adapter transmits I/O commands from the computer to the controller and returns status response to the computer as required by the basic I/O communications procedures.

![Figure 8. PB 440 Shared Memory and Hi-Speed I/O Options.](From the collection of the Computer History Museum (www.computerhistory.org))
In addition, it contains a word counter and a current storage address counter which are filled by the programmer at the outset of a data block transfer between the device and the designated area of shared memory. The adapter thereafter functions to translate the controller's requests for inputs or outputs into Memory Read or Write signals, respectively, and conducts the corresponding data transmissions directly from or into memory. The computer may thus continue processing without interruption until such time as the desired number of words shall have been transferred and will then be directed to a micro-order sequence which will terminate the transmission.

The data transfer rates which can be accommodated if one memory module is used with the above described Memory Interchange and I/O Adapter options are determined by the memory module cycle time (5 microseconds) and the degree of competition among the devices connected to a Memory Interchange. The resulting data rate capabilities are tabulated below for the conditions of 1-4 active channels. In addition to a 24-bit word rate, a character rate is quoted which is applicable to those character oriented devices whose I/O controller is equipped with an assembly/disassembly register for 6-bit characters. Also, the 24-bit word rate is quoted as a bit rate to facilitate comparison with other systems.

### SHARED MEMORY DATA RATES

<table>
<thead>
<tr>
<th>No. Simultaneously Active Channels</th>
<th>24-Bit Word Rate</th>
<th>6-Bit Char. Rate</th>
<th>Bit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200 KC</td>
<td>800 KC</td>
<td>4.8 MC</td>
</tr>
<tr>
<td>*2</td>
<td>100 KC</td>
<td>400 KC</td>
<td>2.4 MC</td>
</tr>
<tr>
<td>3</td>
<td>67 KC</td>
<td>268 KC</td>
<td>1.6 MC</td>
</tr>
<tr>
<td>4</td>
<td>50 KC</td>
<td>200 KC</td>
<td>1.2 MC</td>
</tr>
</tbody>
</table>

*If two memory modules are interlaced the following speeds apply:

| 2                                 | 400 KC          | 1600 KC         | 9.6 MC   |

### PROGRAMMING FEATURES

The microprogrammed character of the PB 440 has been developed above and its effect on the logical organization of the computer was discussed in some detail. A consideration of the programming features implied by this design approach will provide additional examples of microprogramming philosophies.

It has been emphasized that the PB 440 does not have a set of instructions in the same sense that conventional computers have. Instead, its basic design provides the programmer with the option of describing, in terms of elementary micro-orders, a sequence of steps which define an instruction. For clarity, we might refer to an instruction described in this way as a "macro-instruction". A complete instruction set consists of a "control" sequence, to determine which of the various macro-instructions is indicated, and a set of "function" sequences, each of which defines a macro-instruction. This instruction set is normally stored in fast memory although micro-orders can be executed from main memory if so desired.†

The design of an instruction set thus consists of three tasks. First, a macro-instruction format is chosen. This format need not be restricted to a word length of 24 bits, nor to single address instructions; it may, in fact, be a format used by another computer. Indexing, indirect and relative addressing, and other special functions may be included.

A control or interpretive sequence of micro-orders is next devised which fetches and identifi-

† Grasselli's use of a Main Memory and a Control Memory in his "conventional microprogrammed control unit" (Ref. 6) resembles the PB 440 control concept if his fixed logic decoding of a macro-instruction is instead accomplished by a microprogrammed "interpretive sequence" which itself determines the starting address of a microprogrammed "function sequence."
flies successive macro-instruction, performs indicated address modifications and obtains operands as required.

Lastly, the function sequences, each defining an instruction, are written. Here the programmer-designer reduces an instruction to its basic logical ingredients and, thus, mechanizes each operation he wishes to include in the instruction set. The last micro-order of each function sequence returns control to the interpretive sequence.

It should be noted that the amount of time required to interpret a macro-instruction should be kept as short as possible since it must be added to the “execution” time of each macro-instruction. It may vary from less than 5 microseconds, for a relatively simple format, to more than 25 microseconds, where indirect addressing, indexing, and other mode indicators must all be satisfied.

To offset this interpretive “overhead”, a distinct advantage appears in the mechanization of the relatively more complex functions which are usually offered as subroutines on conventional computers.† The ability to efficiently construct the desired function by listing an appropriate sequence of logical manipulations (micro-orders) should be contrasted with the preparation of a subroutine of conventional instructions whose inflexible execution usually includes time for steps not required in the task at hand. The need for register exchange and other housekeeping operations required by fixed logical structures is also obviated during microprogrammed manipulations of functionally unassigned registers. In the time required to perform ten conventional “short” instructions of 10 microseconds duration, a PB 440 sequence of 100 micro-orders can be executed. As a performance indication, a microprogrammed, Arctangent sequence requires approximately 160 microseconds.

A “library” of macro-operations, programmed in micro-order sequences, has been expressed in several useful macro-instruction formats. This permits the user, who is neither prepared to think in terms of micro-orders, nor inclined to place a compiler between himself and the computer, to use a more familiar instruction word format in preparing his programs.

To the casual programmer who has occasion to use only those programming aids and instruction sets that have been devised and provided for him by others, the microprogrammable feature means a larger and more varied instruction repertoire and generally improved performance. To the more senior programmer, this feature represents both a benefit and a challenge. He benefits from the ability to modify and extend the instructions which have been microprogrammed previously, and he is challenged to optimize critical applications by creative programming at the basic micro-order level.

CONCLUSIONS

A particular approach to the design of a microprogrammable computer has been implemented and described above. Its advantages may be summarized by stating that:

1. the ability to replace specialized logical circuitry by a sequence of stored micro-orders leads to efficiencies in the design and mechanization of the computer,

2. the microprogrammed decoding or interpretation of pseudo-instructions allows variable instruction word formats but adds to execution times,

3. the execution time of functions which are typically provided as subroutines are significantly improved by microprogramming implementation,

4. the inherent flexibility of the approach permits the manufacturer to adapt the computer to individual customer requirements, and the customer to adapt the computer to a variety of tasks, and

5. the micro-order control of basic logical operations facilitates the simulation of other computers making possible the use of existing software.

ACKNOWLEDGEMENTS

The authors wish to express their appreciation to Mr. Lowell Amdahl for his valuable ad-

† This conviction is shared with Glantz who presents a clear discussion on p. 78 of reference 2.
vice and consulting assistance during the preliminary design phase of PB 440 development. Congratulations are in order to Mr. J. Roy Willson for his outstanding work on all aspects of computer circuit design and to Messrs. Bill Nickell and Scott Nelson in the perfection of the PB 440 memory units. Thanks are also expressed to Messrs. Stanley Groves and James Lusk for painstaking efforts expended during the course of detailed logical design and equipment checkout.

REFERENCES


