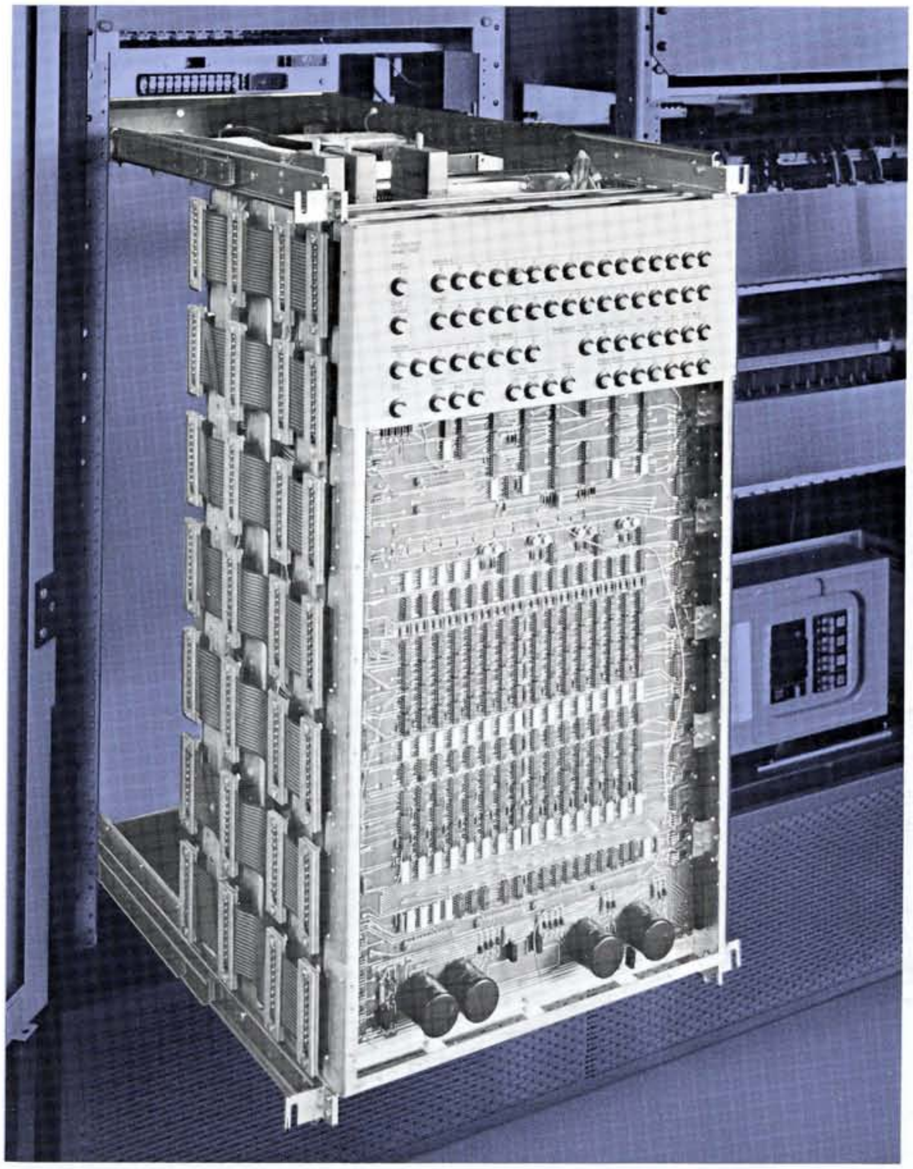
Westinghouse

P-2000 Central Processing Unit



Contents

The central processing unit (CPU) of the Prodac 2000 computer consists of 16 x 28-inch printed circuit cards mounted vertically in a slide-out rack. Four cards (see Figure 1) are required for each unit: the maintenance panel and fast access memory card (frequently called the bit card), the arithmetic and control card, at least one memory card handling 4,096 to 16,384 words, and an input/output (I/O) card. Additional memory cards (up to a total of four cards) can be added to increase core memory size to 65,536 words. The slide-out rack is large enough to accommodate all seven cards which make up the maximum system.

Card positions in the rack are interchangeable, except those of the maintenance panel and fast access memory card and the arithmetic and control card which must be used in that sequence because of interconnecting cables. All other interconnections among the cards are made with 14 ribbon siderails, installed seven on each side of the cards. Connection to the cards is made through 18-pin connectors on the siderails. All siderail signals go to all cards; it is this feature which permits the interchanging of card positions.

The maintenance panel and fast access memory card houses the maintenance control pushbuttons and the logic elements. Elimination of the cable usually required to connect the maintenance panel with the system removes a major source of noise and system failure. Fast access memory, accessible in less than 500 nanoseconds, includes all of the first 16 locations in working memory. It is also able to operate directly and quickly on any location in either fast access memory or core memory. All fast access memory locations can be addressed in the normal way by any instruction; however, the first six locations are general-purpose registers which can be addressed by name as well as by memory location. These registers include the program address register, two base or index registers used for address calculation, a shift instruction register, the accumulator and an extended accumulator.

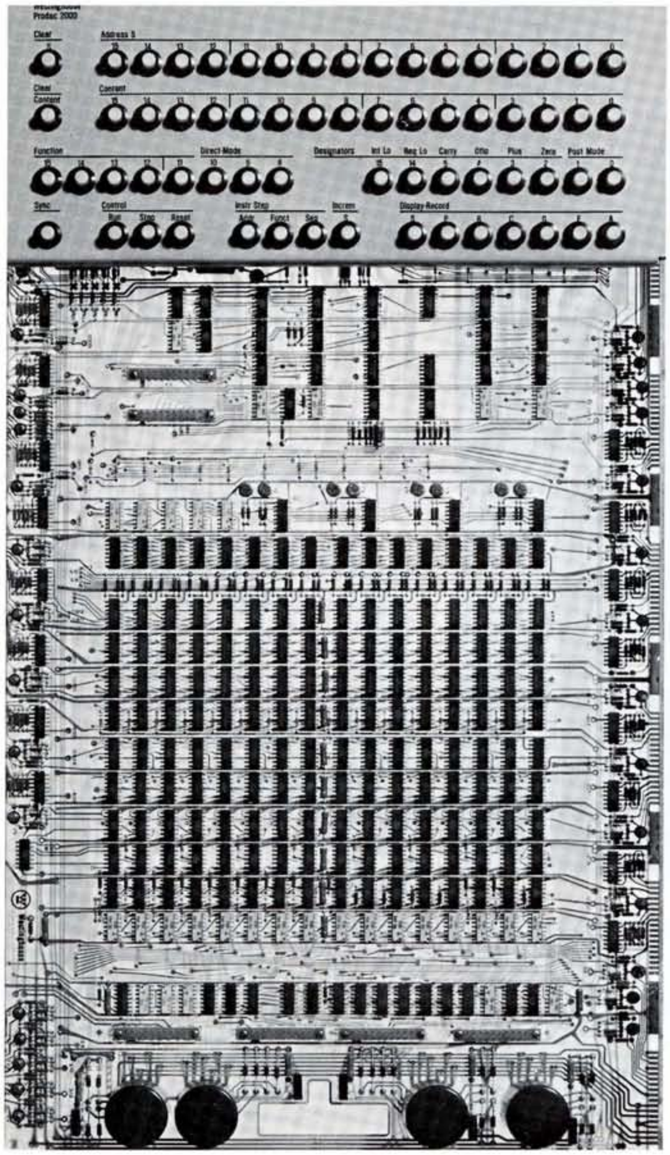
May, 1970

Westinghouse

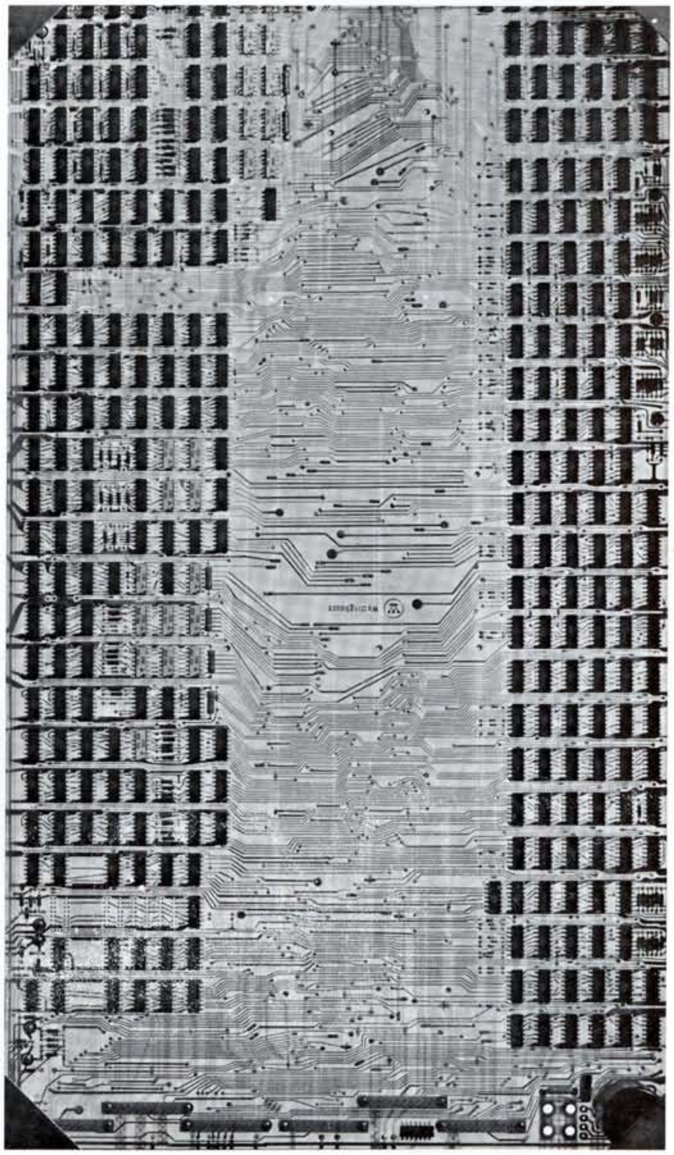
The arithmetic and control card handles double precision add and subtract instructions to permit direct arithmetic operation on 32-bit numbers. It therefore gives accurate representation over a range of −32,768 to +32,768 to the nearest unit. Both multiply and divide instructions are standard to speed program execution by eliminating the use of subroutines for these instructions.

Each memory card handles 4, 8 or 16K words of core memory. However, decoding for the full 16K is provided on each card, regardless of the amount of memory purchased. Additional memory up to the full card capability can, therefore, be field added easily without special adapters, connectors or hardware changes. Word manipulation instructions, such as shifts, increments or decrements can be performed on any location in core memory without transfer of the word to the accumulator.

Figure 1. Four Cards Make up the P−2000 CPU



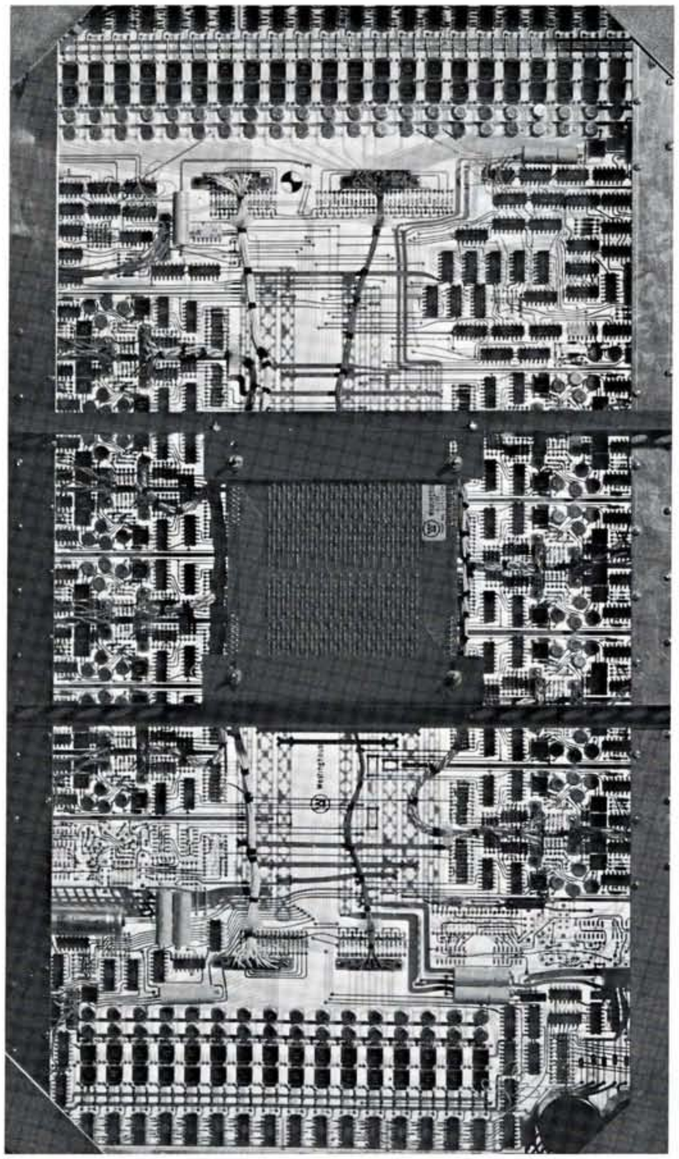
Panel 1—Maintenance Panel and Fast Access Memory

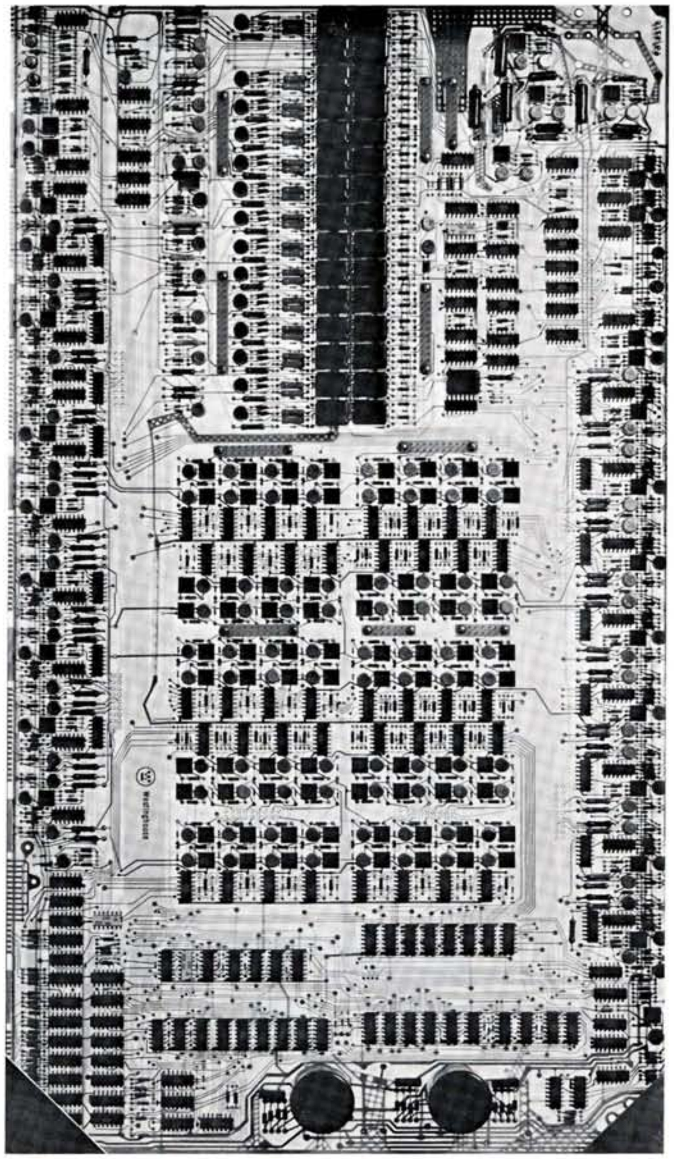


Panel 2—Arithmetic and Control Board

Any block or blocks of memory can be used as buffers for input or output operations betweeen the CPU and mass memory or the process. Buffers can be of any length and any number of buffers may be established by the program. Because data transfer is asynchronous, instruction times depend on the speed of the memory card or the I/O card. Therefore, even after a machine has been in operation, it can easily be speeded up by replacing the original memory and/or I/O card with units designed for faster operation. The functionally separate cards provide a clean break between the I/O data bus and the memory bus, permitting the I/O structure to be relatively independent.

The input/output card in the CPU handles the interface with the process I/O system —the high speed receiving and transmitting of information between the computer and machines or instruments involved in the process being controlled as well as


Panel 3—Memory


Panel 4—Input/Output Control

Westinghouse

document devices such as typewriters, teletypes, line printers, card readers and punches.

The P-2000 CPU can be supplied in several forms, depending on system requirements. The standard industrial cabinet is a double-door cabinet 87 inches high, 53 inches wide and 33 inches deep. The balance of the space in the cabinet is then used for process input/output equipment. For engineering laboratories, small control systems, and other installations where space is limited, a smaller cabinet is available—69 inches high, 24 inches wide and 30 inches deep. The small cabinet may be supplied with a solid door, a door with a glass window, or no door at all. The window is a sliding glass panel centered 46 inches above floor level so that it reveals the maintenance panel. A lock on the window prevents unauthorized access. The CPU can also be provided with no cabinet at all for OEM use.

Advantages

• Fast access memory is accessible in less than 500 nanoseconds. Sixteen flip-flop locations in working memory, including six general-purpose registers, are located on a single card.

• Field - expandable memory. Memory cards are supplied with 4K, 8K or 16K words. Decoding for 16K is supplied on all cards, so that additional memory up to the full card capacity can be added in the field.

• Double precision add and subtract is standard. Accurate representation is provided over a range of −32,768 to +32,768. Floating point hardware is available as an option.

• Hardware multiply and divide. Subroutines are not required for these instructions. Program execution is speeded up.

• Ease of troubleshooting. The functional and physical orientation of the cards helps speed up the diagnosis of trouble. Interchangeability of the cards permits close inspection of a suspect card in the machine. Cards can also be tested off line, because each contains its own timing circuits.

• Ease of maintenance. When a defective card is located, it can be replaced in minutes by pulling the siderails free and unbolting the card. Spare cards can be leased from Westinghouse, and defective cards will be repaired at the factory.

• Usable in industrial environments. The P-2000 CPU operates over a wide range of temperatures and humidity: from 0 to 55C, and from 5 to 95%, noncondensating. Voltage tolerance is ±10%. Frequency is 48 to 62 Hz.

• Powerful addressing capability. The P-2000 CPU operates in eight different modes, depending on the type of addressing used.

• Convenient instruction repertoire. The capability of the 32-instruction repertoire is extended by the flexilibity of its use. For example, a macro-instruction permits multiple loading and storing of the program registers and the designator register. Another provides eight different types of shifting from a single instruction.

Operation

The P-2000 CPU offers high-speed computation ability without loss of power in the instruction repertoire in regard to core addressing or referencing. The first 256 locations in core can be addressed absolutely by any instruction located anywhere in memory. The first sixteen locations (0 through 15) are high-speed, integrated circuit flip-flops. Six of these locations are used for the general-purpose program registers; the remaining ten are available for small, high-speed subroutines. Locations 16 through 255_{10} can be used to store system constants.

Instruction Format

The P-2000 CPU uses a 16 - bit word length. The instruction format is shown in Figure 2.

Function Code. The 5-bit F field, which represents the function of the instruction, is decoded by the CPU hardware which then executes the instruction accordingly. The 5-bit size dictates a basic instruction repertoire of 32 instructions. For programming convenience, each of these 32 instructions is assigned a mnemonic symbol which is an abbreviation of its function. These instructions are listed and described later in this bulletin.

In actual operation, however, the P-2000 computer extends this repertoire of 32 instructions in several ways:

1. The SST and EST instructions permit multiple loading and storing of the general program registers and the designator register.

2. The shift instruction can provide eight different types of shifting operations or the no-shift operation.

3. Addressing capability is flexible and efficient.

Address Mode. The 3-bit m field defines the mode of operation, telling the CPU how to determine (or calculate) the address of the operand (the location to be acted upon by the instruction). The use of this code to obtain a variety of modes of operation gives the P-2000 CPU an extremely flexible and powerful addressing structure.

Displacement Address. The 8-bit y field specifies the basic operand address or the desired channel for input/output functions. It has a total span of 256 addresses, normally from 0 through 255. However, in some instances, bit 7 is treated as a sign bit, giving the field a decimal number range from −128 through +127. For instructions which use core addressing, the number in the displacement field is calculated with other variables specified by the mode (m) bits to obtain the operand address. Addressing will be discussed in greater detail later in this bulletin.

Bit No.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Usage	Function					Mode			Displacement Address							
Symbol	F					m			y, y*							

Figure 2. P-2000 Instruction Format

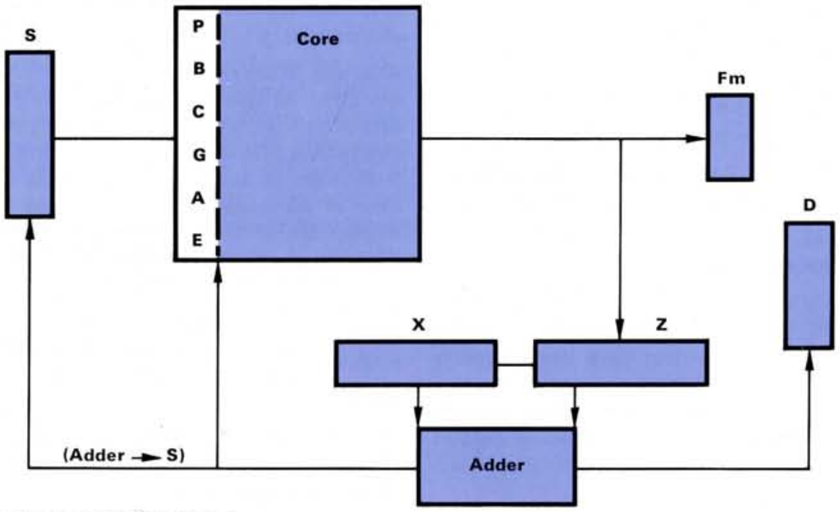


Figure 3. CPU Block Diagram

Basic CPU Hardware

A functional block diagram of the P-2000 CPU is shown in Figure 3. Working core memory contains a maximum of 65,536 (65K) words and a minimum of 4,096 (4K) words, expandable in modular increments of 4K, 8K or 16K. The first 16 locations ($0\text{-}F_{16}$) are integrated circuit flip-flops with a cycle time of less than 500 nanoseconds. The first six locations are used as general program registers. The remaining ten locations are available for any function for which the extra high speed is desirable. All 16 locations can be addressed in the normal way by any instruction.

The P-2000 CPU has the instruction ability to work directly on any memory location without disrupting the contents of the accumulator or any other of the general program registers. Shifting, decrementing, incrementing and storing zeros can be done directly on the operand location. The designator register will contain the status (zero, positive, carry and overflow) of the result of this type of instruction, because the designators are referenced in the hardware to the adder, not the accumulator.

This direct memory ability is invaluable to the programmer in interrogating flags or changing software counters while leaving the data in the general program registers intact. After such an instruction is executed, the status contained in the designators can then be tested by the jump instructions to provide logic decisions in the program.

Hardware Registers External to Memory

Registers S, Z, X, Fm, Adder and D are hardware registers external to core memory. They are used only in regulating the flow of instructions to and from the CPU and none but the D register can be directly referenced by any instruction.

S Register—Memory Address Register—Contains the 16-bit address of the memory location to be accessed for the Read or Write function.

Z Register—Memory Read Data Buffer—Contents of the addressed memory location are copied into this register. These 16 bits provide one input to the adder.

X Register—Data Buffer Register—Provides the second 16-bit input to the adder.

Fm—Instruction Function Code and Address Mode Register—Contains the 5-bit function code and the 3-bit address mode of an instruction read from core memory. The CPU decodes this register to determine the hardware sequencing required to execute an instruction.

Adder—Twos complement, 16-bit adder circuitry. Although the adder is discussed here as a register, it has no storage capacity because it contains no flip-flops. The output of the adder, which is always, instantaneously, the sum of the contents of the Z and X registers, provides the input data to core. The sum of the Z and X registers is written into the addressed core location, giving additional flexibility in the hardware restoration of that location. Although the output of the adder follows the sum of the Z and X registers, the use of the 16-bit output is controlled by hardware - determined gating signals (e.g., Adder $\longrightarrow$ S).

D—Designator Register—The 8-bit designator register, as shown in Figure 4, contains three sets of conditions:

1. The final status of the executed instruction data in the adder.

2. Lockout of external interrupts and service request interrupts. These are discussed later in this bulletin.

3. Post-indirect indexing mode.

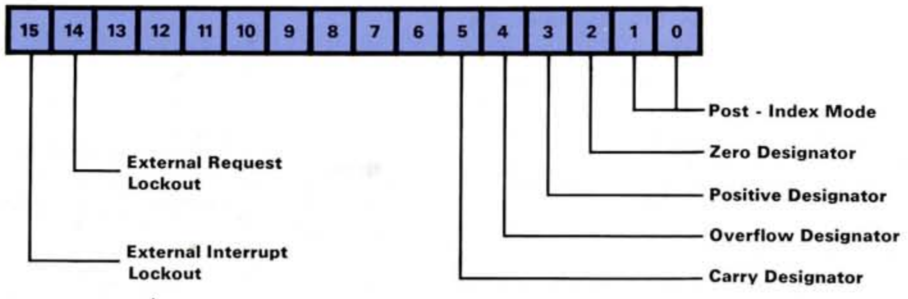


Figure 4. Format in the Designator Register

Status of data in the adder (not in the accumulator) is indicated in the low order six bits by flip-flops preset in certain positions for:

Zero—All bit positions contain 0.

Positive—The most significant bit of the word contains 0.

Carry—the adder function has generated an arithmetic carry (special for shifting).

Overflow—The most significant bit in the number in the adder is opposite to the most significant bit in the Z and X registers (positive+positive=negative; negative+negative=positive).

General Program Registers in Fast Access Memory

The general program registers are computing registers which complement the fixed CPU hardware registers to provide the powerful instruction and addressing capability of the P-2000 computer. They are located in fast access memory and can be directly addressed by the program. They form the flexible basis for instruction execution, address calculation, shifting description and the results of arithmetic operations.

P Register (location 0)—The program address register which contains the address of the instruction being executed. At the initiation of an instruction execution, the P register contains a number which is one less than the address of the instruction to be executed.

B Register (location 1)—A base or index register used for address calculation.

C Register (location 2)—A second base or index register used for address calculation.

G Register (location 3)—Describes the shift instruction.

E Register (location 4)—The extended accumulator. In double precision operations (multiply, divide, double word shifts, etc.), contains the most significant half of the double word.

A Register (location 5)—The accumulator. In double precision operations, contains the least significant half of the double word.

Address Calculation

The operand address calculation is a function of the P-2000 CPU hardware and three possible variables:

1. y—the 8-bit displacement field in the instruction word.

2. m—the 3-bit address mode in the instruction word.

3. M—the 2-bit post-index bits in the designator register.

The y and m variables are contained directly in the instruction being executed. The M variable must have been properly set by a programmed designator instruction (CDR) before execution of any instruction which is to use that variable in its calculation.

The operand address calculation can be used to combine and implement the following types of addressing:

Absolute or Relative
Pre-index
Direct or Indirect
Indirect Relative or Post-index

P-2000 address calculation is based on the relationship of the 8-bit displacement field and the maximum of 65K of memory. The 8-bit displacement field permits direct addressing of 256_{10} locations. A 16-bit number is required for direct access to 65K of memory. However, the various modes of addressing allow modification of the 8-bit number for full access to 65K and provide valuable application tools to the programmer.

Absolute Addressing. The 8-bit displacement field number can be used as a 16-bit positive number to provide direct addressing within a 256-location band starting at location 0, from an instruction located anywhere in memory. The general program registers are included in this band of locations.

Absolute addressing is accomplished in the CPU hardware by making the 8-bit displacement field number a 16-bit positive number. This is done by clearing bits 8 through 15 to 0. This number is then used as the operand address. An example of absolute addressing is shown in Figure 5.

Example: 6003 STZ y where y=7.

The instruction at 6003 will zero the contents of location 7.

Relative Addressing. The 8-bit displacement field number can be used in reference (or relative) to the address of the instruction being executed. The instruction address is used as the center of the 256-location band and the limits of the band are -128_{10} to $+127_{10}$ from that center.

Relative addressing is accomplished in the CPU hardware by treating the 8-bit displacement field number as a 7-bit number, with bit 7 as the sign of that number. When bit 7 is 0, bits 8 through 15 are cleared to 0, providing a 16-bit positive number from 0 to 127_{10}. When bit 7 is 1, bits 8 through 15 are set to 1, providing a 16-bit negative number from -1 to -128_{10}.

This 16-bit number is added to the address of the instruction being executed. The result is used as the operand address. An example of relative addressing is shown in Figure 6.

Example: 6003 STZ y where y=6.

The instruction at 6003 will zero the contents of location 6009.

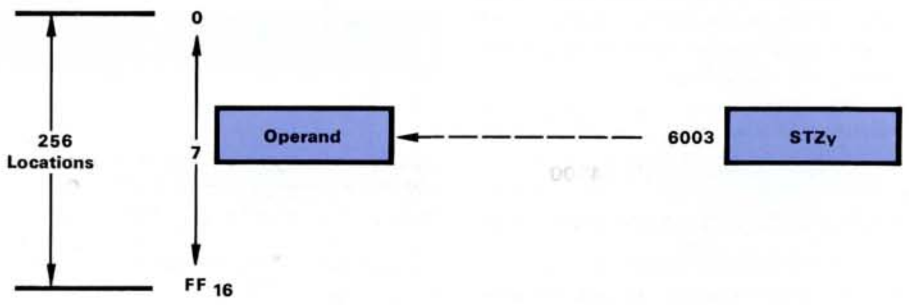


Figure 5. Absolute Addressing

Pre-index Addressing. The 8-bit displacement field number can be referenced to the contents of either the B or C index register to provide direct addressing within a 256-location band in memory, starting at the location referenced by the contents of the index register from an instruction located anywhere in core. Pre-indexing is accomplished in the CPU hardware by making the 8-bit displacement field number a 16-bit positive number (0 through 255_{10}). This number is added to the contents of the specified index register, the result being the operand address. Pre-index addressing is shown in Figure 7.

Example: 6003 y,B Where y=7 and the contents of B=3000

In this mode of addressing, the contents of the index register are often called the Base Address. Pre-index addressing is therefore sometimes referred to as base relative addressing.

Direct and Indirect Addressing. In the previous descriptions of absolute, relative and pre-index addressing, it was assumed that the operand was contained within the direct addressable band of memory. This is called direct addressing. However, the location of the operand is frequently required to be external to the 256-location band which is directly addressable by an instruction. One method of accessing such an operand is to store the operand address in a direct addressable location in the 256-location band. In that case, when the direct address is calculated (absolute, relative or pre-index), the address contained in that location is used as the operand address. This is indirect addressing.

Indirect addressing is accomplished in the CPU hardware by calculating the direct address and using the contents of that address as the operand address. An example of pre-index indirect addressing is shown in Figure 8.

Example: 6003 STX *y,B Where * indicates indirect addressing
y=7 (B)=3000 (3007)=4000

The instruction at 6003 will zero the contents of location 4000.

Post-index Addressing. If pre-index indirect addressing is used, it is also possible to modify the calculated address with the contents of either the B or C base register to accomplish post-index addressing. In this case, the CPU hardware calculates the pre-index indirect address (post-index can only be used with the pre-index indirect mode) and adds the calculated address to the contents of the specified post-index register (B or C). The result is used as the operand address. The pre-index and post-index registers (B or C) may or may not be the same register. An example of post-index addressing is shown in Figure 9, pre-indexing on regis-

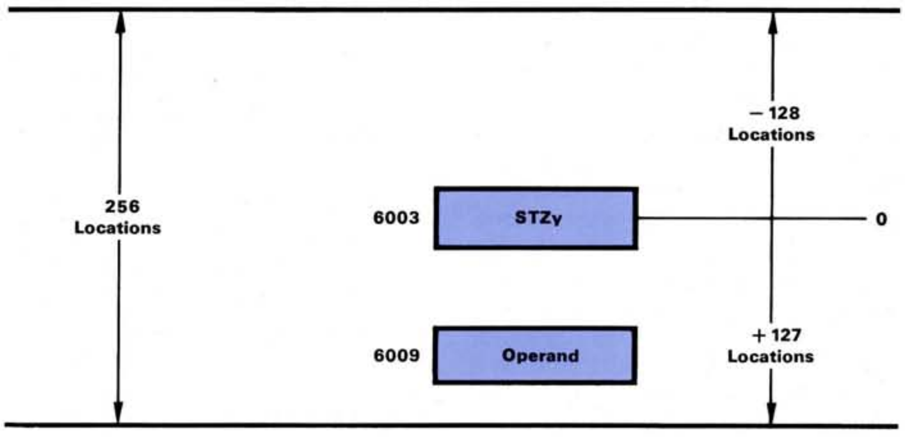


Figure 6. Relative Addressing

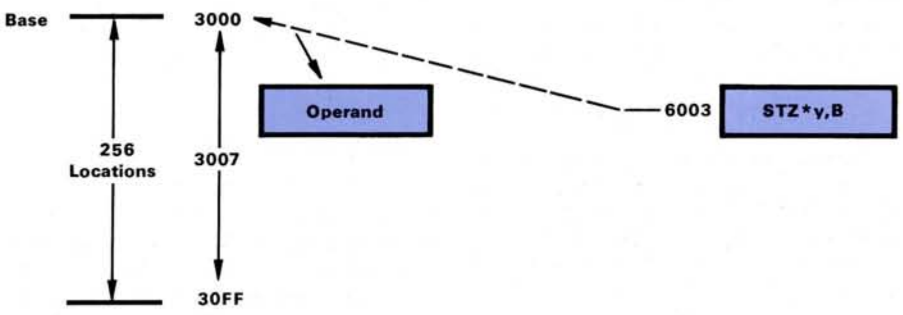


Figure 7. Pre-index Addressing

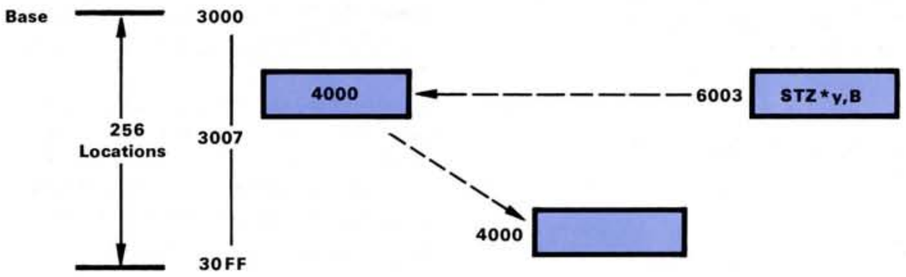


Figure 8. Pre-index Indirect Addressing

ter B and post-indexing on register C.

Example: 6003 STZ *y, B where * indicates indirect addressing
y=7 (B)=3000 (C)=3

The instruction STZ *y,B is identical with the instruction given in the example of pre-index indirect addressing. The difference is in the status of M in the designator register.

Indirect Relative Addressing. After pre-index indirect addressing, it is also possible to modify the calculated indirect address with the contents of that indirect address to accomplish indirect relative addressing. In this case, the CPU hardware calculates the pre-index indirect address (indirect relative can only be used with the pre-index mode) and adds the contents of the indirect address to the indirect address. The result is used as the operand address. Indirect relative addressing is shown in Figure 10.

Example: 6003 STZ *y,B where * indicates indirect addressing
y=2 (B)=3000 (3002)=1004

The instruction at 6003 will zero the contents of location 4006. The example used here is again identical with that used for pre-index indirect addressing and for post-index addressing. As before, the difference is determined by the status of M in the designator register.

Sequence of Address Calculation. The CPU hardware sequencing of the address calculation is:

1. Absolute or relative
2. Pre-index
3. Direct or indirect
4. Indirect relative or post-index

The line of definition between pre- and post-indexing is the point in hardware sequencing where the decision is made to use a direct or an indirect address. If indexing is done before this point, it is pre-indexing; after this point, it is post-indexing.

Specifying the Address Calculation. Instructions for calculating the address are contained in the m field and, if required, the M field of the designator register. The m field can specify the entire address calculation, except for pre-index indirect ad-

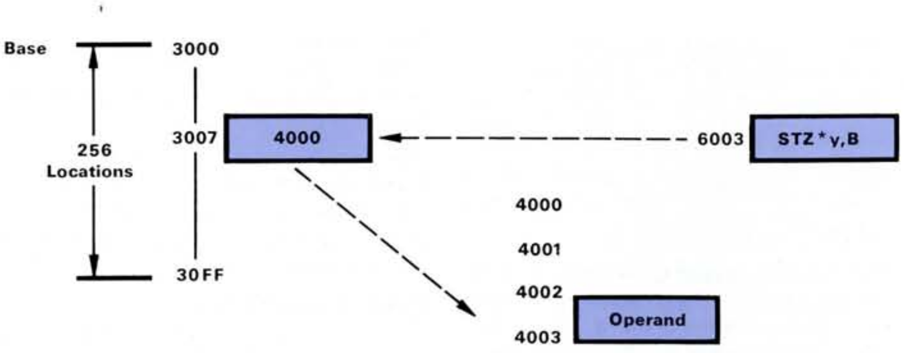


Figure 9. Post-index Addressing

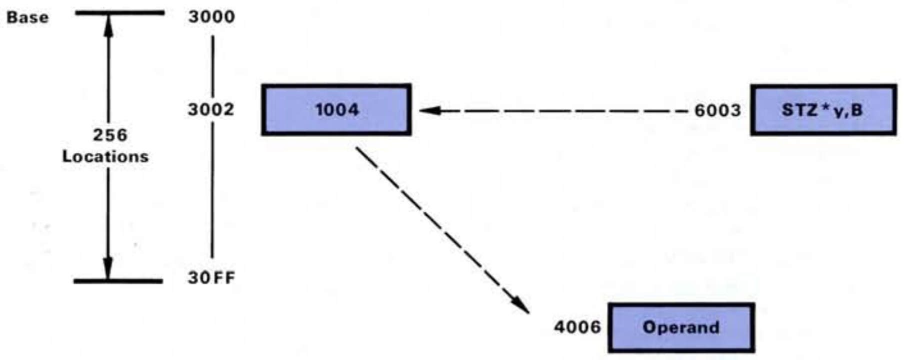


Figure 10. Indirect Relative Addressing

dressing which requires that the M field be tested. Address modifications are tabulated in Table I.

Type of Address Calculation	m	M
Absolute—Direct	000	. .
Relative—Direct	010	. .
Pre-Index B—Direct	100	. .
Pre-Index C—Direct	110	. .
Absolute—Indirect	001	. .
Relative—Indirect	011	. .
Pre-Index B—Indirect		
No Post Addressing	101	00
Indirect Relative	101	01
Post-Index B	101	10
Post-Index C	101	11
Pre-Index C—Indirect		
No Post Addressing	111	00
Indirect Relative	111	01
Post-Index B	111	10
Post-Index C	111	11

Table I. Address Modification

The absolute or relative specification is not shown in the symbolic representation of an instruction (as in the examples shown earlier in this section). The tagging (,B or ,C) indicates pre-indexing and an asterisk (*) indicates the indirect address.

Once again, it should be noted that the programmer must store the M field with the proper information if he wishes to achieve post-indexed address calculation.

Repertoire of Instructions

STP—Stop. Program execution and I/O activity is stopped pending manual restart. The stop occurs very early in the instruction sequence, when instruction bits 10 and 8 have not been considered. Therefore, regardless of their value, only the middle bit of m (bit 9) will have been acted on. Therefore, the Z register will contain y or y*, the X register will contain a number equal to 0 or to (P), and the S register will contain y or y*+(P), for m=$\phi 0 \phi$ or m=$\phi 1 \phi$, respectively. Designators are not changed.

LDA—Load Accumulator. The accumulator is cleared and loaded from the operand address location. The operand address location is not changed. The zero, and positive designators are set or cleared, depending on the resultant contents of the accumulator. Carry, lockout and overflow designators are not changed.

LDE—Load Extended Accumulator. The extended accumulator is cleared and loaded from the operand address location. The operand address location is not changed. The zero, and positive designators are set or cleared, depending on the resultant contents of the extended accumulator. The carry, lockout and overflow designators are not changed.

LDB—Load Base Register B. Base register B is cleared and loaded from the operand address location. The operand address location is not changed. The zero, and positive designators are set or cleared, depending on the resultant contents of the base register. The carry, lockout and overflow designators are not changed.

LDC—Load Base Register C. Base register C is cleared and loaded from the operand address location. The operand address location is not changed. The zero, and positive designators are set or cleared, depending on the resultant contents of base register C. Carry, lockout and overflow designators are not changed.

LDG—Load Shift Description Register—G. Shift description register G is cleared and loaded from the operand address location. The operand address location is not changed. The zero, and positive designators are set or cleared depending on the resultant contents of shift register G. The carry, lockout, and overflow designators are not changed.

CDR—Change Designator Register. The designator register is manipulated according to the bit pattern in the displacement address, y. Normal address calculation is not done, regardless of the bits in m. The instruction operates as follows:

Change Post Index, M
If bit 2 of y is 1, transfer bits 0 and 1 of y into M. If bit 2 of y is 0, do not change M. y is not changed.

Clear Overflow
If bit 3 of y is 1, clear the overflow designator to 0. If bit 3 of y is 0, do not change designator. y is not changed.

Change External Request Lockout
If bit 5 of y is 1, transfer bit 4 of y into the External Request Lockout Designator. If bit 5 of y is 0, do not change External Request Lockout. y is not changed.

Change Interrupt Lockout
If bit 7 of y is 1, transfer bit 6 of y into the Interrupt Lockout Designator. If bit 7 of y is 0, do not change Interrupt Lockout. y is not changed.

EST—Enter Status. The seven accessible registers (P, B, C, G, E, A and D) are cleared and loaded from a table starting at the operand address location and proceeding in monotonic increasing sequence, in the order stated. The table locations are not changed. The designators are left equal to the content of the final table location, with no further changes.

STA—Store Accumulator. The operand address location is cleared and loaded from the accumulator. The accumulator is not changed. The zero, and positive designators are set or cleared depending on the resultant contents of the operand address location. The carry, lockout, and overflow designators are not changed.

STE—Store Extended Accumulator. The operand address location is cleared and loaded from the extended accumulator. The extended accumulator is not changed. The zero, and positive designators are set or cleared depending on the resultant contents of the operand address location. The carry, lockout, and overflow designators are not changed.

SST—Store Status. This instruction is abnormal in its address calculation in that regardless of the content of bit 10 of the instruction, the address reference uses $m=0\phi\phi$. Thus, direct and indirect pre-indexing are suppressed. Bit 10 becomes abnormally defined, for this instruction only, as follows:

Bit $10=0$; the calculated address is taken as the table pointer for storage of the seven accessible registers (P, B, C, G, E, A, and D) and as the program pointer for the next instruction. The next instruction will be taken from the location one higher than the calculated address. The storage table will commence at the calculated address and proceed backward, clearing 7 memory locations in monotonic sequence and there storing the contents of the accessible registers in the sequence stated, with (P) being placed in the lowest numbered address of the sequence. The six registers B, C, G, E, A, and D are not changed. The designators are not changed. External interrupt action and external request action are inhibited immediately following this instruction.

Bit $10=1$; the calculated address is taken as the program pointer for the next instruction. The next instruction will be taken from the location one higher than the calculated address. The content of the B-register will be taken as the table pointer for storage of the seven accessible registers (P, B, C, G, E, A, and D). The storage table will commence at the location specified by B and proceed backward, clearing 7 memory locations in monotonic sequence and there storing the contents of the accessible registers in the sequence stated, with (P) being placed in the lowest numbered address of the sequence. At the end of this storage sequence, (B) is decreased by seven. The five registers C, G, E, A, and D are not changed. The designators are not changed. External interrupt action and external request action are inhibited immediately following this instruction.

ADD—Add to Accumulator. The operand is added to the content of the accumulator, algebraically. The result is left in the accumulator. End-around carry is not provided, and negative quantities are expressed in two's - complement notation. The zero, positive, and carry designators are set or cleared depending on the results of the adding. The overflow designator will be set if an overflow occurs; otherwise it is unchanged. The lockout designator is not changed. The content of the operand location is unchanged.

SUB—Subtract from Accumulator. The operand is subtracted from the content of the accumulator, algebraically. The result is left in the accumulator. The content of the operand location is not changed. End-around carry is not provided, and negative quantities are expressed in two's-complement notation. The zero, positive, and carry designators are set or cleared depending on the results of the subtracting. The overflow designator will be set if an overflow occurs; otherwise it is unchanged. The lockout designator is not changed.

MPY—Multiply Accumulator. The operand is multiplied by the content of the accumulator, algebraically. The result is left in the extended accumulator and the accumulator with the low order in the accumulator. Sign logic is included such that the product is a proper double-length algebraic product with the sign at the left of the extended accumulator. The content of the operand location is not changed. Negative quantities are expressed in two's-complement notation. The zero, and positive designators are set or cleared depending on the double-length product results. The carry designator is cleared and the overflow designator is not changed. The lockout designator is not changed.

DIV—Divide. The content of the extended accumulator and the accumulator is divided by the operand, algebraically. The dividend low-order portion is placed in the accumulator. The quotient appears, properly signed, in the accumulator. The remainder appears, also properly signed, in the extended accumulator. The content of the operand location is not changed.

It is presumed that at least the most significant bit of the dividend (in E) will represent dividend sign. Note that remainder sign will agree with the dividend sign as seen prior to execution. All negative quantities are expressed in two's-complement notation.

The zero, and positive designators are set or cleared depending on the quotient results. The carry designator is made equal to the sign of the remainder (i.e., set to 1 implies negative), and the overflow designator will be set if the quotient overflows the accumulator. This will happen, for instance, whenever the content of the operand location fails to exceed in magnitude twice the content of E, prior to execution.

ADA—Add Double Length to Accumulator. The double-length operand is added to the double-length accumulator, algebraically. The result is left in the double-length accumulator. End-around carry is not provided, and negative quantities are expressed in two's-complement notation. The zero, positive, and carry designators are set or cleared depending on the results of the adding. The overflow designator will be set if an overflow occurs; otherwise it is unchanged. The lockout designator is not changed. The content of the double-length operand location is unchanged.

The double-length accumulator is the end-to-end combination of registers A and E considered as a single register, with E containing sign and most significant bits. The double-length operand location is the end-to-end combination of the addressed location and that location with address larger by unity, considered as a single register. The addressed location contains sign and most significant bits.

SDA—Subtract Double Length from Accumulator. The double - length operand is subtracted from the double-length accumulator, algebraically. The result is left in the double-length accumulator. End-around carry is not provided, and negative quantities are expressed in two's-complement notation. The zero, positive, and carry designators are set or cleared depending on the results of the adding. The overflow designator will be set if an overflow occurs; otherwise it is unchanged. The lockout designator is not changed. The content of the double-length operand location is unchanged.

The double-length accumulator is the end-to-end combination of registers A and E considered as a single register, with E containing sign and most significant bits. The double-length operand location is the end-to-end combination of the addressed location and that location with address larger by unity, considered as a single register. The addressed location contains sign and most significant bits.

AND—And with Accumulator. A bit-by-bit logical product of the operand and the accumulator content is formed. The result is left in the accumulator. The content of the operand location is not changed. The zero, and positive designators are set or cleared depending on the resultant accumulator content. The carry, overflow, and lockout designators are not changed.

EOR—Exclusive OR with Accumulator. A bit-by-bit logical "exclusive OR" of the operand and the accumulator content is formed. The result is left in the accumulator. The content of the operand location is not changed. The zero, and positive designators are set or cleared depending on the resultant accumulator content. The carry, overflow, and lockout designators are not changed.

INC—Increment Location. The operand is increased by unity without involving the accumulator. The result is left in the operand location. The zero, positive, and carry designators are set or cleared depending on the results of the incrementing. The overflow designator will be set if an overflow occurs; otherwise it is unchanged. The lockout designator is not changed.

DCR—Decrement Location. The operand is decreased by unity without involving the accumulator. The result is left in the operand location. The zero, positive, and carry designators are set or cleared depending on the results of the decrementing. The overflow designator will be set if an overflow occurs; otherwise it is unchanged. The lockout designator is not changed.

SHF—Shift Location. The operand location is shifted in a manner and to an extent dictated by the content of the Shift Description Register, G. The result is left in the location(s) shifted. The zero, and positive designators are set or cleared depending on the results of the shifting. The carry designator is set equal to the last bit shifted off of either end of the single or double word being shifted. The overflow designator is set if the most significant bit of the location(s) shifted is changed during the left shifting. Otherwise, it is unchanged. The lockout designator is not changed.

Shifts provided are left and right, circular and open ended with sign or zero extension, for both single and double words. Up to 31 positions may be shifted. The Shift Description Register, G, provides type and number of places information, and is not changed by the shift command.

In double-length shifting, the operand address is the most significant word location. The least significant word location is one higher. For instance SHF, 4 with double shifting called by (G) causes (A) and (E) to shift.

JMP—Jump Unconditional. Unconditionally transfer the operand address to the P register. The designators are not changed as a result of this instruction. External interrupt action is inhibited immediately following this instruction.

PJP—Positive Jump. If the positive designator is set, transfer the operand address to the P register. Otherwise take the next instruction. The designators are not changed as a result of this instruction. External interrupt action is inhibited immediately following this instruction.

OJP—Overflow Jump. If the overflow designator is set, transfer the operand address to the P register. Otherwise take the next instruction. The designators are not changed as a result of this instruction. External interrupt action is inhibited immediately following this instruction.

CJP—Carry Jump. If the carry designator is set, transfer the operand address to the P register. Otherwise take the next instruction. The designators are not changed as a result of this instruction. External interrupt action is inhibited immediately following this instruction.

NJP—Negative Jump. If the positive designator is not set, transfer the operand address to the P register. Otherwise take the next instruction. The designators are not changed as a result of this instruction. External interrupt action is inhibited immediately following this instruction.

ZJP—Zero Jump. If the zero designator is set, transfer the operand address to the P register. Otherwise take the next instruction. The designators are not changed as a result of this instruction. External interrupt action is inhibited immediately following this instruction.

ITR—Input Transfer. The operand address location is cleared and loaded from the input data trunk and an input acknowledge signal is transmitted to the I/O subsystem. The designators are not changed.

OTR—Output Transfer. The content of the operand address location is placed on the I/O data trunk and an output acknowledge signal is transmitted to the I/O subsystem. The designators are not changed.

IOA—Input To or Output From Accumulator. The operand address is transmitted to the I/O subsystem, then either of two alternatives occurs depending on the state of bit 7 of the operand address:

Bit 7=1; The accumulator is cleared and loaded from the I/O data trunk. The zero and positive designators are set or cleared depending on the resultant content of the accumulator. The carry, lockout, and overflow designators are not changed.

Bit 7=0; The content of the accumulator is placed on the I/O data trunk. The content of the accumulator is not changed. The zero and positive designators are set or cleared depending on the resultant information placed on the I/O data trunk. The carry, lockout, and overflow designators are not changed.

Up to 128 channels may be accessed using the IOA instruction. Also, up to 64 multiplexer words may be associated with any of the channels for multiplexed format. The IOA formats are summarized below:

IOA y	where y, bit 7 is 1 for input or 0 for output and y, bits 6-0 contain the direct channel number 0-127_{10}.
IOA$^*\alpha$	where α, bit 7 is 0 for output and α, bits 13-8 contain a multiplexer word address 0-63_{10} and α, bits 6-0 contain the channel number 0-127_{10}.

Note that data output and input will be done from bit positions left justified to bit position 14.

STZ—Store Zero. The operand address location is cleared to zero without regard to the accumulator. The zero and positive desigators are set. The carry, lockout, and overflow designators are not changed.

Input/Output Channel Operation

The P-2000 CPU provides both direct channel and buffered channel operation. The input/output hardware configuration dictates what equipment will be direct and what will be buffered.

Direct channel operation requires that the program execute an IOA instruction (input to or output from the accumulator) for each word of data to be transferred into or out of the CPU. It also requires that input data from the accumulator be stored in memory before the next IOA is executed or that output data be loaded in the accumulator before the IOA is executed.

Buffered channel operation requires that the program reference the buffer (anywhere in memory), initialize the buffer size (index count), and initiate the channel operation. Once started, the data will be loaded sequentially into (input) or out of (output) the buffer without further program attention.

Channel Data Transfer Completion Signal

When a direct or buffered channel data transfer operation is executed, the CPU is free to process program instructions while awaiting the completion signal from the channel hardware. When the channel function is accomplished, the channel completion signal is generated into the CPU.

Direct Channel Data Completion. A direct channel may or may not require a completion signal. If a completion signal is required, it enters the CPU as an external interrupt. (This does not imply that external interrupts are used only for direct channel completion signals.)

The CPU reacts to the external interrupt by executing an instruction out of sequence to the running program. This instruction is contained in memory location 100_{16} which is hardware-defined as the common entry for all external interrupts. The instruction in this location initiates a special program to accomplish the desired function for the channel and then returns to the original running program.

Completion interrupt processing is controlled by the CPU hardware up to and including the execution of the out of sequence instruction. The specification of that instruction, the special program, and the return to the running program is accomplished in software.

Buffered Channel Data Completion. A completion signal for a buffered channel data transfer enters the CPU as a service request interrupt. The P-2000 computer can accommodate a maximum of 63 service request interrupts. However, four of these are committed to specific functions, leaving only 59 available for other process functions including the buffered channel data completion signals.

Each service request interrupt is hardware-assigned to two memory locations which are defined as $100_{16} + 2 \times$ SRI and $101_{16} + 2 \times$ SRI. The service request interrupt used for a buffered channel data completion signal uses the two locations as follows:

$100_{16} + 2 \times$ SRI—will contain an OTR (output) or ITR (input) instruction.

$101_{16} + 2 \times$ SRI—will contain a number which is the negative equivalent of the buffer size (index count).

When the service request interrupt which represents a completion signal is detected by the CPU, the instruction at location $100_{16} + 2 \times$ SRI is executed out of sequence to the running program (with all designator activity suppressed). Execution of this instruction includes special address calculation to determine the buffer start address, increment of the negative index count, adding of that count to the buffer start address, and finally, the data transfer to or from the location in the buffer. Return is then made immediately to the running program.

Buffered Channel-Buffer Completion Signal

When the specified buffer has been exhausted, the CPU hardware generates a buffer overflow service request interrupt (SRI 1). This situation occurs whenever a service request interrupt has incremented its negative index count to zero. The interrupt is detected and processed in the CPU hardware by the execution of the out of sequence instruction at 102_{16}. This instruction initiates a special program which scans the contents of the service request interrupt index counter for 0. After the zero index count is determined, the software accomplishes whatever function is required due to the buffer completion and then returns control to the original program.

Service request interrupts not being used should contain a positive, non-zero number in the corresponding index count to accommodate the overflow buffer scan function.

Direct Channel Address

The direct channel function requires that the CPU hardware address that channel by using the operand address as the subchannel address. This, then, requires an extension of the definition for the operand address:

1. For memory access instructions, the operand address contains the address of the location which contains the operand.

2. For channel access instructions, the operand address contains the address of the specified subchannel.

The channel may require either single or two-dimensional addressing, depending on the channel hardware. Where two-dimensional addressing is required, one coordinate is the subchannel address and the other is the word address. The term subchannel is used here to distinguish this type of addressing from direct channel and buffered channel functions.

Direct Channel Functioning

The basic tool of direct channel operation is the IOA (input to or output from the accumulator) instruction. The IOA instruction has the normal P-2000 address calculation.

IOA—absolute addressing. Absolute addressing can be used whenever single dimensional channel addressing is required. For example:

6003 IOA 009

The instruction at 6003 outputs the contents of the accumulator to channel 9. The instruction format is shown in Figure 11. Bit 7 indicates whether an input or output is to be done; if bit 7 is a 1, an input is required; if bit 7 is 0, an output is required. In the example in Figure 11, an output is called for. If an input were needed, the instruction would be:

6003 IOA 89_{16}

IOA—other than absolute addressing. Two dimensional addressing requires too many bits for it to be fully specified in the instruction word 8-bit displacement field. Therefore, absolute addressing cannot be used. Figure 12 shows relative indirect address calculation for an IOA instruction.

In this example, the instruction is:

6003 IOA *y

where y=4, m=011, and (6007)=1109. The instruction at 6003 outputs the contents of the accumulator to the channel specified by two-dimensional address word 11_{16} and subchannel 9. The instruction address calculation is relative indirect and the operand address contains the channel address. The format for the operand address (location 6007) is shown in Figure 13.

The output function is specified by bit 7 in the operand address word. A zero in bit 7 calls for an output; a one calls for an input. In the example shown, if an input were required, the operand address for the same channel address would be 1189_{16}.

Example of Direct Channel Output. (See Figure 14.) The block diagram shows the relationship of the CPU hardware configuration and the direct channel output to the I/O equipment. When the IOA instruction is executed, the following results occur:

1. The operand address is loaded into the S register. The subchannel address is decoded from the S register.

2. The contents of the accumulator are stored in the X register (via the Z register) and the Z register is zeroed. The output of the adder presents the accumulator data to the I/O equipment.

3. The data from the adder is gated into the addressed I/O equipment.

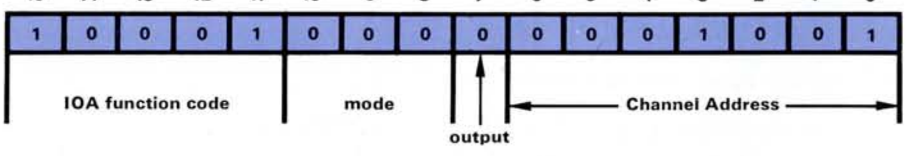


Figure 11. Instruction Format for Absolute Addressing on an IOA

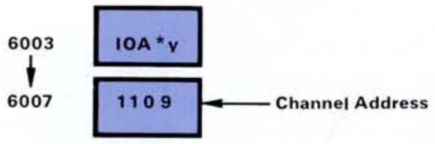


Figure 12. Relative Indirect Address Calculation for an IOA

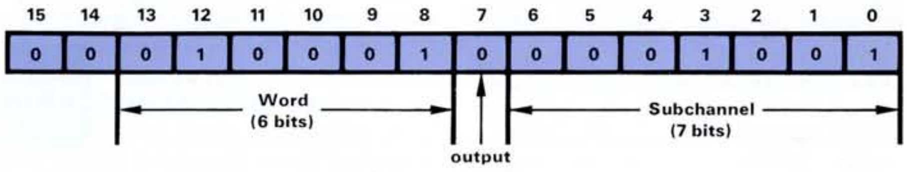


Figure 13. Operand Address Format for an IOA

Note that the completion signal through the external interrupt is not shown. This operation is described earlier in this section.

Example of Direct Channel Input. (See Figure 15.) The block diagram shows the relationship of the CPU hardware configuration and the direct channel input to the I/O equipment. When the IOA instruction is executed, the following results occur:

1. The operand address is loaded into the S register. The channel address is decoded from the S register.

2. The I/O data is stored in the X register (via gating into the Z register) and the Z register is zeroed. The output of the adder presents the I/O data to memory.

3. The I/O data is stored in the accumulator.

Note that no completion signal through the external interrupt is shown in Figure 15. This operation is described earlier in this section.

Buffered Channel Output Functioning

This operation is a mixture of hardware and software. The block diagram (Figure 16) shows the relationship of the CPU hardware and the buffered channel I/O device. In this example, it is assumed that the buffered channel completion signal is wired to service request interrupt number 6 and the contents of the 3-word buffer 5000 - 5002 are to be output.

1. Service request interrupt number 6 memory definition. The memory locations assigned to that service request interrupt are $100_{16} + 2 \times 6 = 10C_{16}$ and $101_{16} + 2 \times 6 = 10D_{16}$.

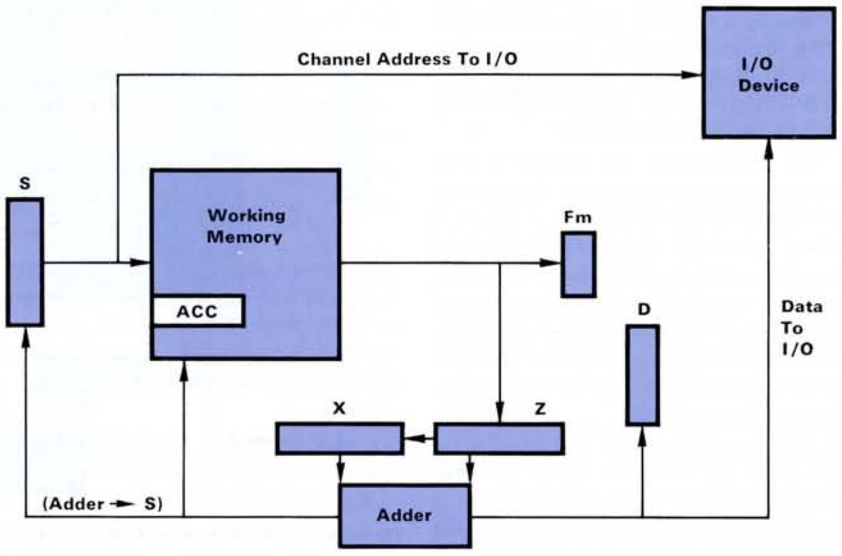


Figure 14. Block Diagram of Direct Channel Output

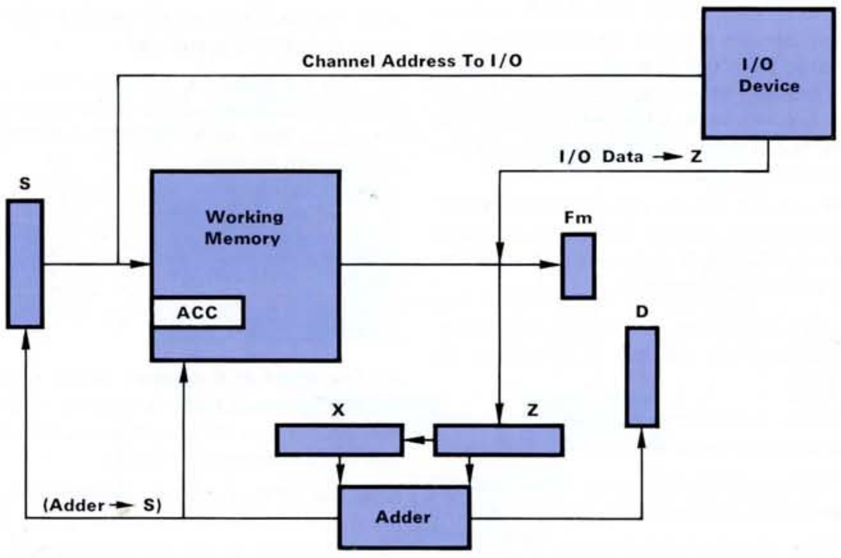


Figure 15. Block Diagram of Direct Channel Input

2. Program initialization. The programmer has arbitrarily selected location 130_{16} to contain the output buffer start address (5002).

The service request interrupt locations are loaded with:

10C OTR *80
10D DAT-3
(2's complement of the buffer size)

The address calculation for the OTR at location 10C is special, because it will be done in the buffer mode due to the service request interrupt. The address is calculated in reference to 100_{16} not 10C. This buffered channel data completion is explained earlier in this section.

3. The programmer has loaded the 3-word output buffer with data in the format required by the I/O device.

Channel Initiation. An OTR instruction executed in the program (not as an out-of-sequence execution due to a service request interrupt) initiates either an output or an input channel operation. The ITR instruction cannot be used to initiate buffered channel operation.

The CPU hardware and the service request interrupt hardware retain information as to whether or not an instruction is initiated by the detection and subsequent processing of a service request interrupt.

The initiating OTR contains the service request interrupt number 6 as its operand address. When this instruction is executed, service request interrupt number 6 is gated from the S register into the service request interrupt logic. This forces that logic to react as if the interrupt has occurred and the interrupt operation is initiated as follows:

1. When the service request interrupt logic detects the number 6 request, it stores the number 6 in a counter and begins the buffer mode of operation.

2. The instruction in location 10C is obtained and the address is calculated as follows:

a. 100_{16} is added to the instruction word displacement field. Result: 180_{16}.

b. The contents of location 180_{16} are obtained. Result: (5002).

c. The negative index count in location $10D_{16}$ is incremented (first cycle, to -2).

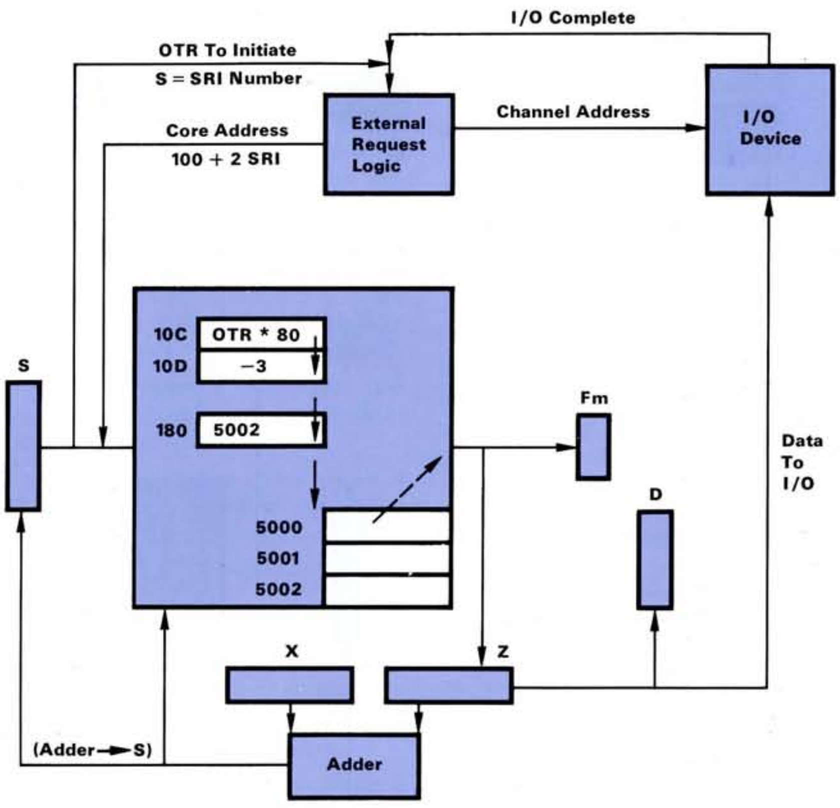


Figure 16. Block Diagram of Buffered Channel Output

If the incremented index count equals zero, the buffer overflow service request interrupt (#1) is generated.

d. The contents of location 180_{16} are added to the incremented index count and the result (first cycle, 5002 -2 = 5000) is the operand address.

3. The OTR to the calculated operand address is executed. Contents of location 5000 are copied into the X register (via the Z), and the Z register is cleared. The output of the adder presents the data to the I/O device.

4. The count of 6 retained in the service request interrupt logic is used to address the channel, causing gating of the data into the addressed I/O device.

After the OTR execution, the running program is immediately resumed. (Note that the execution of an out-of-sequence instruction due to a service request interrupt is done with designator activity suppressed). When the I/O device data transfer function is complete, the service request interrupt operation is repeated.

This recycling is continued until all the words in the buffer have been transferred. When the buffer is exhausted, the buffer overflow service request interrupt is generated and recycling stops. The buffer overflow function is explained more fully later in this bulletin.

Buffered Channel Input Functioning

Buffered channel input operation is similar to the output functioning. The same parameters are assumed as given in the output example, except that the 3-word buffer at 5000 - 5002 is loaded with data from the I/O device. The entire operation is then the same as for the output example with the following exceptions:

1. Location 10C will contain ITR *80.

2. When the ITR to the calculated operand address (at location 10C) is executed, the data is gated from the I/O device and stored directly in the buffer location (via the Z register and the adder.)

External Interrupts

External interrupts give the P-2000 system the ability to detect an event, immediately interrupt the running program, and execute an instruction out of sequence to that program. The P-2000 computer can accommodate a maximum of 64 external interrupts (number 0 through 63).

Although each external interrupt is a discrete signal, a common single entry is sent to the CPU when any external interrupt is detected. The single entry point has a hardware-assigned memory location (100_{16}). In this location, the programmer loads the SST instruction which will be executed out of sequence and will be used as the entry to an external interrupt subroutine.

External Interrupt Lockout. When an external interrupt is detected and the external interrupt lockout is not set, the CPU hardware sets the interrupt lockout flip-flop in the designator register. This prevents another external interrupt from being processed before the first is accomplished. Although the interrupt lockout prevents an external interrupt from being processed, the hardware retains any interrupt signal until it is processed.

Interrupt lockout is cleared by programming (CDR) when the need for the protection no longer exists.

The interrupt lockout can also be set by the programmer (CDR) if it is desired to protect a portion of any given program from other program execution.

External Interrupt Functioning. (See Figure 17.)

The block diagram shows the relationship of the CPU hardware to the external interrupt logic.

Individual interrupts are scanned by an oscillator, counter-decoder network contained in the external interrupt logic hardware. The scanning is completely independent of the CPU instruction execution. When the scanner detects an interrupt, it generates a common signal (HIT) to the CPU and stops the scanning, retaining the interrupt number in the counter.

Processing The Interrupt

1. The CPU detects the presence of the HIT signal. At the end of the next non-jump instruction, the HIT signal is processed as follows:

a. Forcing address 100_{16} and obtaining the instruction SST INT contained in that location. Contents of the P register are not disturbed.

b. External interrupt lockout is set.

c. The SST INT instruction is executed. This instruction saves the status of the interrupted running program and initiates a special program.

2. The special program executes an IOA FF_{16} to input the interrupt number from the counter into the accumulator. This instruction also resets the external interrupt logic counter to zero and allows scanning to be resumed.

3. The special program uses the interrupt number to transfer program control to another program specifically written for this interrupt.

4. When the interrupt program is completed, the CPU returns control to the interrupted running program and the interrupt lockout is cleared (reset).

Service Request Interrupts

Service request interrupts have already been discussed in some detail in the explanation of buffered channel operation. However, they are also used for other functions. Service request interrupts are completely independent from external interrupts.

The P-2000 computer can accommodate a maximum of 63 service request interrupts (numbered 1 through 63_{10}) for various process input/output devices. Those devices can be the type that can utilize interrupt functioning or the automatic counting feature or the buffer referencing capability inherent in this type of interrupt.

Each service request interrupt (SRI) is hardware assigned two memory locations which are defined as:

$101_{16}+2\times SRI$	Contains the index count.
$100_{16}+2\times SRI$	Used to contain the out of sequence instruction.

Processing of a service request interrupt causes the incrementing of the assigned index count if that count number is negative. If the count is already positive, the count will not be changed.

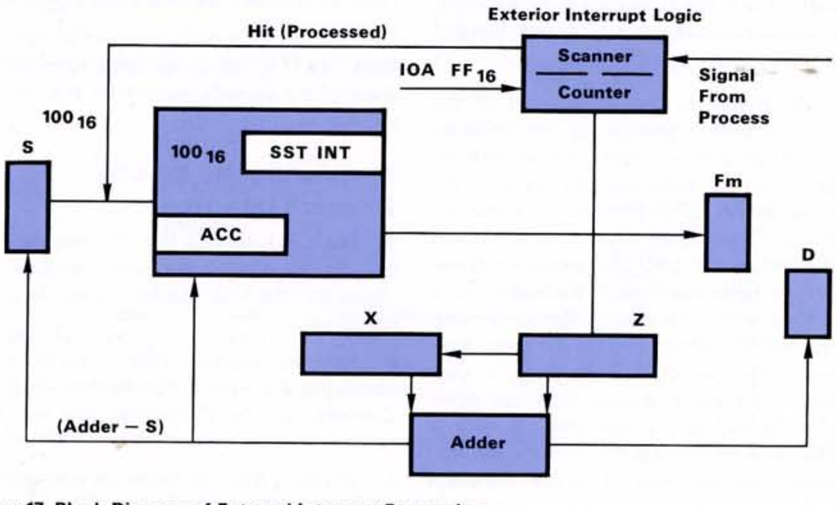


Figure 17. Block Diagram of External Interrupt Processing

Committed (Dedicated) Service Request Interrupts. In all P-2000 systems, the first four service request interrupts are committed to the following functions:

SRI No.	Function	Location
1	Buffer Overflow	102, 103
2	Power Failure	104, 105
3	Sync Int—60 CPS	106, 107
4	Sync Int—1000 CPS	108, 109

Buffer Overflow. The buffer overflow interrupt can be considered a signal internal to CPU hardware. It is processed the same as any other service request interrupt. The buffer overflow signal is generated any time another service request interrupt increments its assigned index count to zero. The signal is used to flag this condition in the hardware and to initiate a software scan of all index counts to determine which service request interrupt activity caused the overflow (buffer exhausted) condition. Appropriate action is then taken in the software.

Service Request Interrupt Operation. (See Figure 18.)

The block diagram shows the relationship of the CPU hardware to the service request interrupt logic.

Individual requests are scanned by oscillator counter-decoder hardware contained in the interrupt logic. The scanning is completely independent of the CPU instruction execution. When a request is detected, an I/O request signal is generated in the CPU, scanning is stopped, and the service request interrupt number is retained in the logic counter.

OTR Initiation. In the application of the service request interrupt for the buffered channel completion signal, the service request interrupt logic can be directly accessed by an OTR instruction executed in the program mode. The program mode is defined by the CPU and service request interrupt logic hardware. Basically, it is any time a service request interrupt out-of-sequence instruction is NOT being executed. When an OTR instruction is executed, the service request interrupt number contained in the operand address is gated from the S register into the service request interrupt logic. This will simulate the particular request and the logic will react as if it had detected the signal from the process.

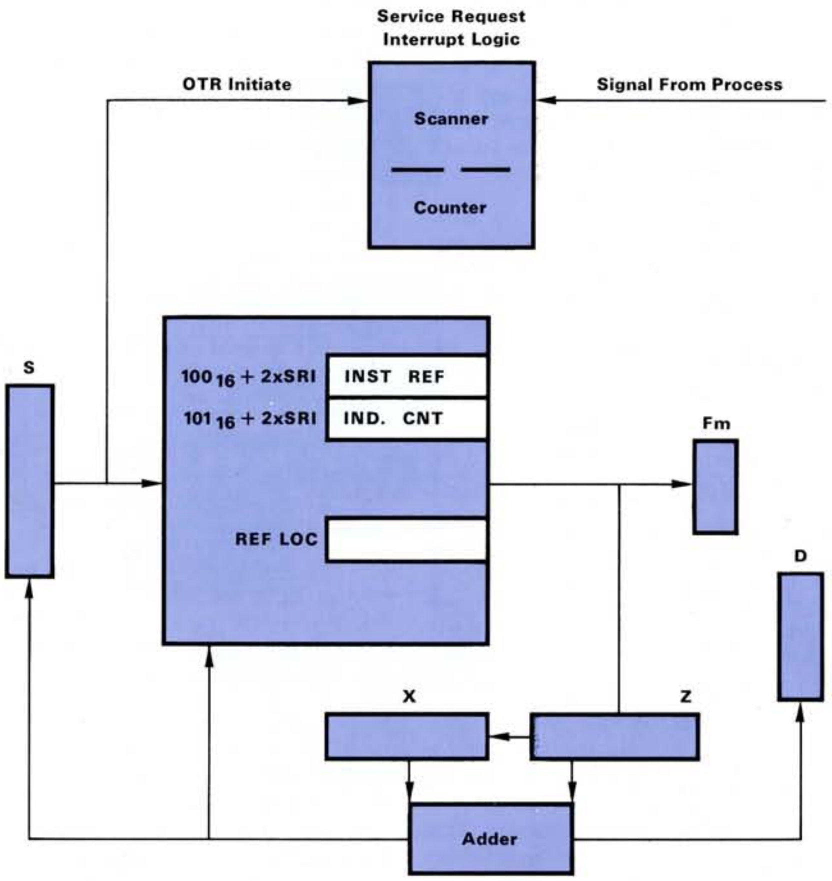


Figure 18. Block Diagram of Service Request Interrupt Operation

Note: An ITR cannot be used to initiate a channel by directly accessing the service request interrupt logic.

Processing the Service Request Interrupt

1. The CPU detects the I/O request signal. At the end of the next non-jump instruction, the I/O request signal is processed by:

a. Forcing address 100_{16}+2 x SRI and obtaining the instruction in that location. Contents of the P register are not disturbed.

b. Initiating the SRI mode of operation in the hardware.

2. The instruction at 100_{16}+2 x SRI is executed out of sequence. Designator activity is suppressed.

a. 100_{16} is added to the reference number contained in the displacement field of the instruction. Contents of the resulting address are obtained.

b. The index count is incremented if the count is a negative number. Buffer overflow is generated if the incremented number equals zero.

c. The index count is added to the contents of the address obtained in step a. The sum is the operand address for the out of sequence instruction.

3. At the completion of the out of sequence instruction, the SRI mode is completed (return to program mode). The

counter in the service request interrupt logic is reset, allowing scanning to be resumed.

Service Request Lockout

Service request lockout is never set by the CPU or service request interrupt logic as the result of processing a service request interrupt. Service request lockout must be set or cleared by programming by means of the CDR instruction. It is used at the discretion of the programmer.

Processing A Second Service Request Interrupt

A second service request interrupt is locked out by the CPU hardware until the out of sequence instruction is completed. This lockout is extended through the next instruction if that instruction is a CDR. The programmer can, therefore, use the CDR to set request lockout for subroutine protection in processing the current service request interrupt.

SST and EST Instructions

Two instructions in the P-2000 instruction repertoire (SST and EST) provide hardware multiple storing and loading of the general program registers and the designator register. These instructions are referred to as "macro-instructions," because they eliminate the need for individual instructions to accomplish this common function. Storing or loading is done in a fixed order, to or from a specified list of seven consecutive locations which may be anywhere in memory.

SST Instruction (Store Status and Jump). This instruction stores the contents of the general program registers and the designator register in seven consecutive core locations. The starting address of the list for the SST instruction is the *highest address location* and is specified in the instruction word. After storing the list, program control is transferred (jump) to the next instruction specified by the SST instruction.

For example, let us assume that the SST instruction is to be used to store the program status in a memory list which has its highest address equal to 5008. The next instruction will be at location 5009. The instruction is, therefore:

6000 SST 5008

The instruction is contained in memory at location 6000. When the register storing is complete, the list will be:

5002	(P)
5003	(B)
5004	(C)
5005	(G)
5006	(E)
5007	(A)
5008	(D)
5009	Next Instruction

The jump to the next instruction (at location 5009) is then made.

The register list is always stored in the same order. The hardware stores from the high location (starting at address 5008) in decrementing order until the contents of the last location (5002) are stored with the contents of the P register.

EST Instruction (Enter or Restore Status and Jump) This instruction enters the contents of the general program registers and the designator register from the specified list of seven consecutive memory locations. The starting address of the list for the EST instruction is the *lowest address location* and is specified in the instruction. After the contents of the list have been entered into the registers, program control is transferred to the instruction dictated by the newly entered contents of the P register. This transfer actually accomplishes the function of a jump and is inherent in the EST instruction.

For example, let us assume that the contents of seven locations from 5021 through 5027 are to be entered into the general program registers, and that the number to be stored in the P register is 6000. The instruction is:

7000 EST 5021

The result will be:

5021	6000	to P
5022	Data	to B
5023	Data	to C
5024	Data	to G
5025	Data	to E
5026	Data	to A
5027	Data	to D

After the EST instruction has entered the contents of the list into the major registers and the designator register, the P register contains the number 6000 and the CPU will automatically transfer control to the next instruction at location 6001.

The registers are always entered from the contents of the list in the same order. The hardware effectively starts from the lowest location (5021) and proceeds in incrementing order until the contents of the last location are stored in the designator register.

SST and EST Temporary Location. A core location 257_{10} (or 101_{16}) is used for temporary storage during the execution of SST and EST instructions. This location should, therefore, not be used in application programming.

SST and EST Used Together. The SST and EST instructions are completely independent and can be used individually. However, they may also be used together as reversible functions, as shown in Figure 19. In that example, a basic program (PRG

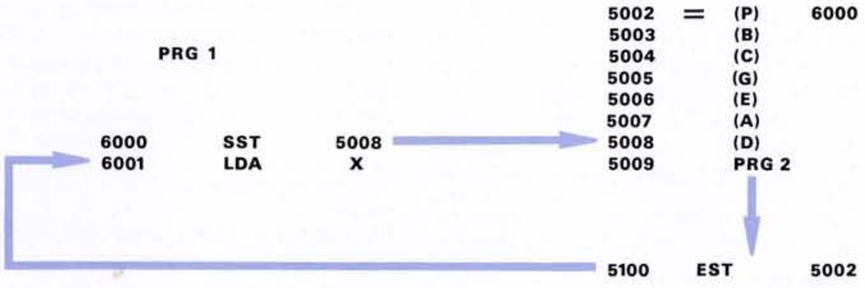


Figure 19. SST and EST Instructions Used Together

1) uses the SST to store its current status and jump to a second program (PRG 2) which can be visualized as a subroutine. At the end of the second program, an EST is executed to restore the status of the basic program and resume that program at the instruction following the SST. No address mode calculation is used in this example.

1. PRG 1—Execute instruction at 6000. When the SST instruction is executed, the current status of PRG 1 is stored at locations 5002 through 5008. The stored P (location 5002) will contain 6000.

2. PRG 1—Jump to instruction at 5009. The SST jumps to location 5009 which is the starting point of PRG 2.

3. PRG 2—Execution. The instruction list of PRG 2 is executed until the instruction at 5100 is reached.

4. PRG 2—Execute instruction at 5100. The EST instruction at 5100 enters the contents of the list at 5002 through 5008 into the major registers and the designator register. The inherent jump is performed to the next number via the number now contained in the P register (P=6000, next instruction at 6001).

5. PRG 1—Execute instruction at 6001. PRG 1 is resumed at the point where it was interrupted by the execution of the SST. Major registers (except P) and the designator register are in exactly the same state as before the interruption.

The preceding examples of SST and EST instructions were limited to simple addressing in order to keep the explanation as clear as possible. However, the SST instruction has special addressing mode capability which enables the programmer to specify the starting address of the save list through the contents of the B register as well as the calculated operand address. After the SST has been executed, the B register contains a number which is seven less than the original address. The programmer thus has a pointer which can be extremely valuable in push-down list operations (relating to compiler functions).

The EST instruction has the normal full address capability of the P-2000 computer.

Shifting

The P-2000 CPU permits direct shifting on any memory location without disturbing the contents of the general program registers. The description of the shifting operation and the number of bits to be shifted is stored in the G register before execution of the shift instruction. The format of the shift description register (G) is shown in Figure 20. The shift count can contain a number from 0 through 31_{10}.

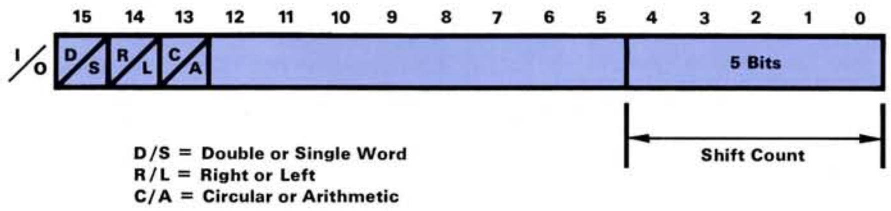


Figure 20. Shift Description Format

Single and Double Word Shifting. Single word shifting shifts the contents of the specified single, 16-bit word. Double word shifting shifts the contents of two consecutive memory locations which are treated as a single, 32-bit number. In double word shifting, the most significant half of the double word is contained in word 1, which is the operand location. Word 2 is in the operand location plus 1. For example, to shift locations 5000 and 5001 as a double word, location 5000 is word 1 and location 5001 is word 2. The operand address is 5000.

Examples of single and double word shifting are shown in Figure 21. In these examples, a 4-bit word length is used to represent a condensed version of a 16-bit word to simplify the example. The bit at the extreme left is the sign bit. In all of the examples, the initial status of the word (or words) is shown on the left and the result of the first shift on the right.

Designator Operation on Shift

Carry designator—On a right shift, the carry designator is set whenever the last shifted off bit contained a 1. On a left shift, the carry designator is set whenever the last most significant bit (single word, bit 15; double word, bit 31) shifted off contained a 1.

Overflow designator—Set during left shifting if the initial sign bit has been changed at the completion of the shift.

Zero designator—Set if the result of the shift (single word or both words of a double) are all zeros.

Positive designator—Set if the most significant bit of the result of a shift operation contains a 0 (single word, bit 15; double word, bit 31).

Sign Extension. Sign extension is used in right arithmetic shifts to continue the initial positive or negative number value. It is accomplished by copying the sign bit into the next bit (and holding the original sign bit) each time a bit shift is performed.

Zero insert. Zero insert, used in left arithmetic shifting, is accomplished by storing a 0 in bit 0 each time a bit shift is performed.

No Shift Case. When the shift instruction is used and the shift count in the G register equals 0, no shift will be performed. However, the designators will be set according to the nonshifted value. The no-shift function is valuable in interrogating the contents of any core location without changing the numerical value of those contents.

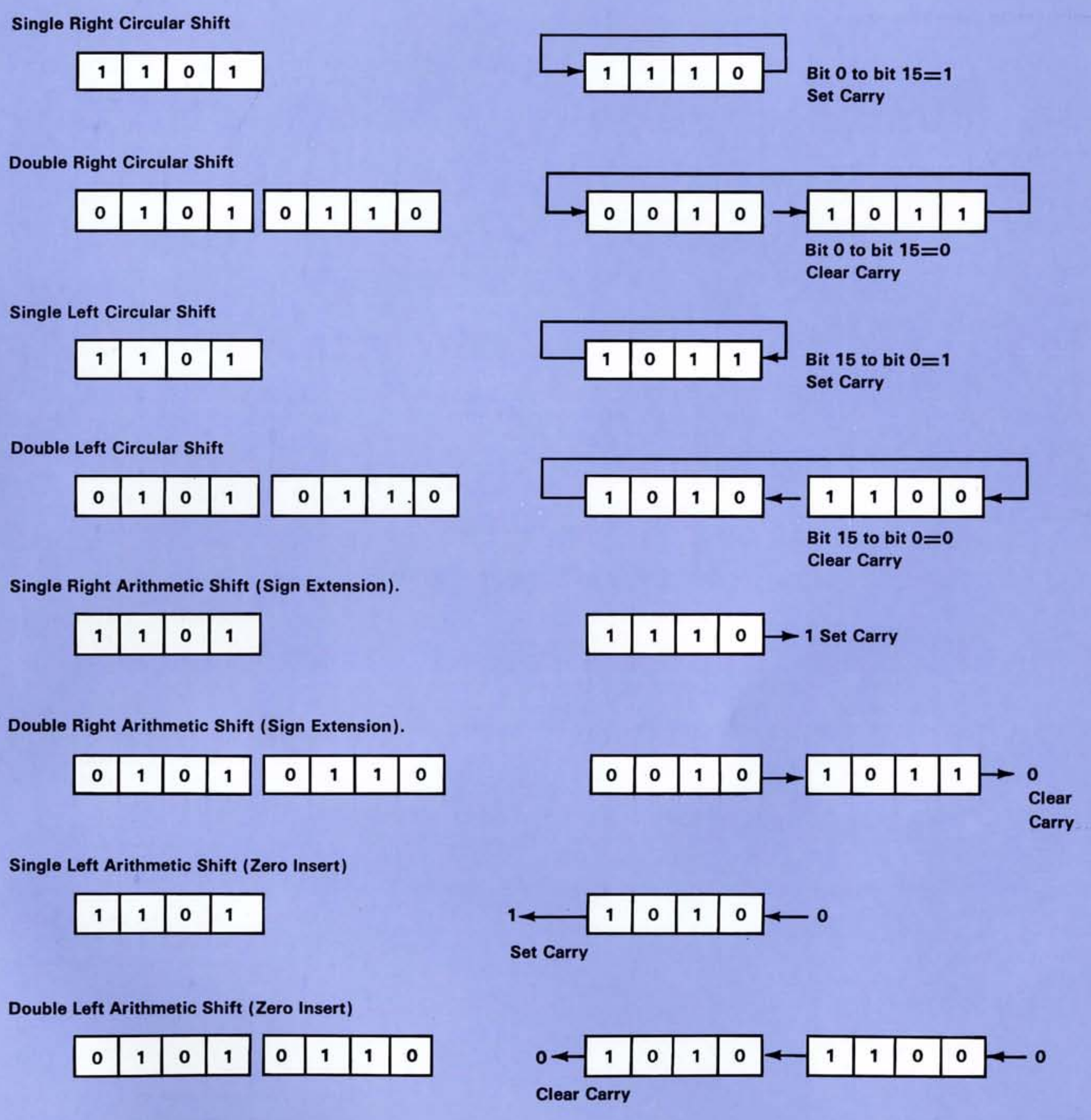


Figure 21. Examples of Shifting

P-2000 Central Processing Unit

Westinghouse Electric Corporation
Computer & Instrumentation Division, Pittsburgh, Pa. 15238
Printed in USA

102659314