SPECIFICATION

Instruction Set Modifier Register

Description

This invention relates generally to a processor for a data processing system, and, in particular, to an instruction set modifier register which is loadable under program control and which permits the execution of more than one instruction set by the processor, utilizing the same instruction decoding circuitry and instruction execution control logic circuitry.

In a data processing system the basic logical and arithmetic computational operations are performed by the processor. For this purpose there is provided within the processor a number of registers and logic circuits. The processor registers are used to receive, hold, and transmit information (data and instructions) used by the processor. Several different types of registers are provided within the typical processor. For example, an accumulator register temporarily stores data and accumulates the results of logical or arithmetic operations. A program counter stores the address of the next instruction in memory to be executed. An instruction register stores the instruction code (also known as the operation code) portion of the instruction which is currently being executed by the processor, and an address register or data counter stores the operand portion of the currently executing instruction. Throughout the specification the term "instruction" will hereafter be used to refer specifically to the instruction code or operation code portion of an instruction.

To enable the processor to perform all of the intended arithmetic and logic operations which are desired to be performed, the processor is provided with the capability of executing a repertory of individual instructions collectively known as an instruction set. Individual instructions are executed by the processor to perform such operations as loading information into a register, transferring information between registers or between registers and memory, comparing the contents of two registers, and so forth. Such instructions may be thought of as "macroinstructions" since the execution of one such instruction by the processor comprises a number of sub-operations or "microinstructions" by the circuitry making up the instruction execution control logic portion of the processor. During the execution of a single instruction many different logic gates in the instruction execution control logic circuitry may be opened and closed in a precise sequence in order to implement the particular macro-operation called for by the instruction. The opening or closing of each gate may be individually viewed as a single microinstruction.

It is very useful for a processor to have the capability of executing an instruction set for a different computer in addition to its own instruction set. Normally the processor is endowed with a unique instruction set comprising a plurality of individual operation code words, each of which comprises a distinctive combination of 1's and 0's, in response to which the instruction decoding circuitry and instruction execution control circuitry perform all of the individual microinstructions necessary to carry out the particular instruction. A known prior art emulation technique, referred to as microprogramming, allows a processor to execute instruction sets from several different computers. For example, the IBM (Registered Trade Mark) System/360 and System/370, as well as other known computer systems, utilize a special program called a "microprogram" to execute a plurality of individual microinstructions which together comprise a basic macroinstruction. In response to a given macroinstruction, a special processor memory known as a control store is accessed and a microprogram corresponding to the macroinstruction is executed, with each of the individual microinstructions, serving to control the desired operation of the instruction execution control logic circuitry. In this manner, an IBM 1401, for example, can have its instructions emulated on an IBM System/360 or System/370.

It will be appreciated that the capability of a processor to emulate another processor by executing the instruction set of the other processor adds a great deal of flexibility and versatility to a computer system, which can result ultimately in substantial savings to the user of such system.

The advantages to the user of a data processing system having emulation capabilities include the ability to execute software written for more than one type of computer system thereby avoiding the necessity of expensive and time-consuming software conversion efforts.

While the technique of microprogramming has been utilized in a number of mainframe and minicomputer systems, it does not lend itself well to microcomputer systems, where the space necessary for a control store is difficult to justify on the limited silicon area of a large-scale-integrated (LSI) circuit device. There is therefore a need for a mechanism in a processor permitting the processor to execute two or more instruction sets without requiring the extensive consumption of silicon space in an LSI circuit device.

According to the invention there is provided a processor comprising: a data bus for transmitting and receiving data, and a plurality of registers and logic circuits including an instruction register coupled to said data bus for temporarily storing an individual one of a plurality of instructions from two or more instruction sets, said instruction sets sharing in common at least one instruction representing an operation to be performed by said processor, said at least one instruction being represented by a different instruction word in each of said instruction sets, said instruction register generating a first set of binary signals in response to storing a first instruction word representation of said at least one instruction and generating a second set of binary signals in

response to storing a second instruction word representation of said at least one instruction, an instruction set modifier register having a plurality of states each corresponding to one of said instruction sets, said instruction set modifier register generating a third set of binary signals representing a first one of said plurality of states and generating a fourth set of binary signals representing a second one of said plurality of states, instruction decoding logic comprising a first plurality of conductors responsive to said instruction register, a second plurality of conductors responsive to said instruction set modifier register, a third plurality of conductors intersecting said first and second plurality of conductors, a fourth plurality of conductors intersecting said first and second plurality of conductors, first means responsive to said first and third sets of binary signals for generating a first group of binary signals over said third plurality of conductors, second means responsive to said second and fourth sets of binary signals for generating a second group of binary signals over said fourth plurality of conductors, and, instruction execution control logic having a plurality of control lines and controlling the operation of said plurality of registers and logic circuits by activating selected ones of said control lines, said instruction execution control logic being responsive to said third and fourth plurality of conductors and comprising means for activating the same ones of said control lines in response to either of said first and second group of signals.

The invention will now be described by way of example only with particular reference to the accompanying drawings wherein:

Figure 1 is a block diagram of a preferred embodiment of the invention and includes the internal bus structure, programmable registers, and logic blocks and

Figure 2 shows a detailed logic diagram of a portion of the block diagram shown in Figure 1.

In Figure 1, a block diagram of a processor, including the internal bus structure, programmable registers, and logic blocks, is shown. The block diagram of Figure 1 represents a modification of the architecture of the Motorola 6800 microprocessor in accordance with the teachings of the present invention. For a complete description of the Motorola 6800 microprocessor reference may be had to United States Patent No. 4,037,204, and to the related patents and patent applications referenced therein.

A number of control function blocks are shown in Figure 1, including bus control circuit 11, halt logic 13, timing generator 21, restart logic 35, instruction decode and control circuit 41, and interrupt logic circuitry 45. The microprocessor architecture depicted in Figure 1 represents the basic functional connections between the registers, counters, arithmetic logic unit (ALU), control function blocks, internal address busses, and internal data busses.

The microprocessor device 10 shown in Figure 1 has four 16-bit registers and four 8-bit registers available for use by the programmer. The program counter PCL, PCH indicated by reference numerals 58 and 66, respectively, is a two-byte register which points to the current program address. Its contents are updated after execution of an instruction. The incrementor register INC, which includes INCL 36 and INCH 40, updates the current program address during the execution of a current instruction by incrementing or decrementing its current contents. It may load its contents into the program counter independently of the address bus and may function as an auxiliary register or as an up-down counter or incrementor/decrementor of other registers connected to the address bus or data bus. The stack pointer SP includes SPL 70 and SPH 76 and is a two-byte register that contains the address of the next available location in an external pushdown/pop-up stack.

The temporary register T 38 is used for temporary storage and may store the higher order address bits. The index register IX includes IXL 65 and IXH 69 and is a two-byte register used to store a 16-bit memory address in the indexed mode of memory addressing.

Two 8-bit accumulators ACCA 82 and ACCB 77 are used to hold operands and results from the ALU 88. The condition code register 85 indicates six condition flags: Negative Number, Zero, Overflow, Carry from Bit 7, Carry from Bit 3, and Interrupt Mask. These six bits of the condition code register are used as testable conditions for conditional branch instructions. The condition code register 85 also contains two unassigned bits.

With specific reference now to the instruction register IR 53, instruction set modifier register ISMR 100, and instruction decode and control circuit 41, the features unique to the present invention will be described. Instruction register IR 53 is an 8-bit register which latches up and holds information in the form of individual instructions applied to it from data bus 57. Instruction decode and control circuit 41 generates a plurality of control signals over a plurality of conductors shown generally as control signal bus 43, which conductors are understood to be dispersed throughout the processor to the registers, ALU circuitry, busses, control function blocks, and the various coupling and buffer circuits associated therewith in order to control in proper sequence the necessary information transfers during execution of a given instruction held within instruction register IR 53. Reference may be had to the aforementioned United States Patent No. 4,037,204, and in particular to Figures 3A—3N, 3P—3Z, 3AA—3NN, and 3PP—3TT for the specific interconnections between the instruction decode and control circuit 41 and the various processor elements controlled, with the exception of the interconnections between the instruction decoding control circuit 41 and the instruction set modifier register ISMR 100. The latter specific

interconnections are discussed hereinafter with respect to Figure 2.

Still with reference to Figure 1, it will be seen that the instruction set modifier register ISMR 100 may receive information over data bus 57 via branch 95 thereof. Such information serves to load ISMR 100 with a binary number representing a particular instruction set. For each unique number stored in ISMR 100 it is possible for the instruction decode and control circuit 41 to interpret a given instructiom stored in instruction register IR 53 as a member of a different instruction set. In its simplest form, ISMR 100 may be a 1-bit flipflop, one state of which indicates that instructions received and stored by instruction register IR 53 are to be interpreted as belonging to a first instruction set and the other state of which indicates that instructions received are to be interpreted as belonging to a second instruction set.

The particular state to which ISMR 100 is set is transmitted as a signal over bus 101 to instruction decode and control circuit 41. In turn, ISMR 100 may receive control signals from instruction decode and control circuit 41 over bus 102.

With reference now to Figure 2 the structure and operation of the instruction set modifier register ISMR 100 as it relates to the instruction register IR 53 and the instruction decode and control circuit 41 will now be described. Instruction decode and control circuit 41 is shown in dashed outline in Figure 2 and comprises instruction decoding logic 111 and instruction execution control logic 112. A plurality of register and ALU control lines 43, 102, and 23, two sets of IR output lines 141—148 and 151—158, a pair of ISMR output lines 140 and 150, a timing signal bus 25, a restart logic control line 37, a non-maskable interrupt (NMI) line 47, and an interrupt request ($\overline{IRQ}$) line 51 interface the instruction decode and control circuit 41 to the other processor elements.

Within the instruction decoding logic 111 a plurality of lines 116 intersect the ISMR output lines 140 and 150 and the IR output lines 141—148 and 151—158 and lead into logic circuitry within the instruction execution control logic 112, which logic circuitry generates a sequence of outputs over register and ALU control lines 43, 23, and 102 to control the operation of the various registers and other control circuitry within the processor in response to the contents of the IR and ISMR. Lines 116 are connected to lines 140—148 and 150—158 at the locations, for example, shown by X's in Figure 2, each representing a transistor which is rendered conductive by the application of a binary 1 signal over one of lines 140—148 or 150—158 to its gate electrode and which is rendered nonconductive when a binary 0 signal is applied to its gate electrode.

A representative portion of the instruction execution control logic circuitry 112 of instruction decode and control circuit 41 is shown in dashed outline 103 and comprises AND gate 107, inverter 108, and NAND gate 109. It should be understood that logic circuit portion 103 is shown for illustrative purposes only and that the instruction execution control logic circuitry of a typical processor instruction decode and control circuit will typically have many more logic gates between the instruction decoding logic 111 and the output of the instruction decode and control circuit 41. Reference may be had, for example, to the aforementioned United States Patent No. 4,037,204 (Figure 3) for a disclosure of a typical instruction decode and control circuit.

IR 53 is implemented in the form of a plurality of bistable latches or flip-flops, one for each bit of an instruction word received over input lines 121—128. IR 53 receives as control inputs thereto an interrupt response signal over line 47' as well as control signals from instruction decode and control circuit 41 over lines not shown. In response to the binary value received over each of IR input lines 121—128, IR 53 generates a corresponding binary value over output lines 141—148 and the complement of such value over lines 151—158, the complement being produced by inverters 131—138.

ISMR 100 may also be implemented as one or more bistable latches or flip-flops. Only one such flip-flop is required to provide the capability of executing one additional instruction set. However, the capability of executing a third and fourth instruction set can readily be provided by using an additional flip-flop, and so forth. In response to a binary signal received over ISMR input line 120, ISMR 100 generates a signal having the same binary value over line 140 and the complement of such signal over line 150 after inversion by inverter 130. The dashed line inputs 166—168 to ISMR 100 represent additional possible input lines, each giving the capability of executing twice as many additional instruction sets. Thus N input lines allow 2^N instruction sets to be executed.

The operation of a preferred embodiment of the invention will now be described with reference to Figure 2. The output lines 141—148 and 151—158 of IR 53 are connected to various ones of lines 116, as mentioned earlier, to provide the desired decoding of any given instruction within a given instruction set. By way of example, assume for the moment that ISMR 100 is in a first state representing a first instruction set to be executed, in response to, for example, a binary 0 input over line 120, and that IR 53 is storing a given 8-bit instruction word 11011001 (line 128 inputs the most significant bit and line 121 inputs the least significant bit). Because ISMR 100 is assumed to be storing a 0, line 140 is low and line 150 is high. The high output over line 150 of ISMR 100 turns on transistors 170, 173, 175, and 177 whose control gates are responsive to the high signal on line 150. Because IR 53 is assumed to be holding instruction word 11011001, lines 141, 152, 153, 144, 145, 156, 147, and 148 are all high, whereas the remaining outputs from IR 53 are low. The transistors whose

control gates are coupled to those lines of IR 53 which are in a high state are turned on, whereas those transistors whose control gates are coupled to those lines in a low state remain turned off. Conductive paths are formed along particular ones of lines 116 to the instruction execution control logic 112, provided that all of the connecting transistors along such lines have been enabled by an appropriate combination of high level outputs from IR 53. For example, a conductive path is formed along line 163, since transistors 175, 178 and 179 are all turned on. OR gate 183 accordingly generates a high input into AND gate 107. Similarly, line 165 is rendered conductive, since transistors 177, 190, 191, and 192 are all turned on so that OR gate 184 generates a high input into AND gate 107. AND gate 107 accordingly generates a high input into NAND gate 109. Since neither transistor 194 or 195 on lines 160 and 161, respectively, is turned on, OR gate 182 generates a low input to inverter 108, which in turn generates a high input to NAND gate 109. NAND gate 109 thus generates a low output for the instruction word 11011001 of the first instruction set.

If it is desired to achieve the same logical output from the represented portion 103 of the instruction execution control logic circuit 112, using an altogether different instruction word from a second instruction set, the state of ISMR is switched to a logical "1" so that output line 140 is high and line 150 is low. Assuming that 01110001 is the instruction word in the second instruction set which corresponds to instruction word 11011001 in thefirst instruction set, appropriate connecting transistors are provided along lines 160, 162, and 164, which serve as alternate inputs into OR gates 182—184, respectively, in order to generate the identical logical result at the output of logic circuit portion 103 for both instruction words. Line 164 goes high since transistors 176 and 193 are turned on, and consequently OR gate 184 is also turned on, transmitting a high level input into AND gate 107. Line 162 also becomes conductive when transistors 174 and 180 are turned on, so that OR gate 183 transmits a high level input into AND gate 107. Consequently AND gate 107 is turned on and generates a high level signal to NAND gate 109. Line 160 does not become conductive, since neither transistor 194 nor transistor 195 is enabled for this particular instruction word. As a result, OR gate 182 generates a low level input to inverter 108, which generates a high level input to NAND gate 109. NAND gate 109 thus generates a low level output, since both its inputs are high. Thus the ultimate output of logic circuit segment 103 is the same when instruction word 11011001 of the first instruction set or instruction word 01110001 of the second instruction set is present in IR 53. To ensure that identical results from each of the other logic circuit portions (not shown) of instruction execution control logic circuit 112 are achieved for these two different instruction words, appropriate connections are made between lines 116 and lines 140—148 and 150—158 from ISMR and IR 53 in a manner analogous to that just described with refernece to logic portion 103.

An OR gate 180—184 is provided for each input into the instruction execution control logic 112 for which it is desired to generate the identical control signal sequence over register and ALU control lines 23, 102, and 43 for different instruction words from different instruction sets. To the input of each such OR gate is connected a separate line corresponding to each different instruction set, the instructions of which one wishes to execute. In the example shown in Figure 2 two different instruction sets may be executed, depending upon the logical state of ISMR output lines 140 and 150. Accordingly, there are two lines input to each OR gate 180—184. If it were desired to execute four different instruction sets, then a maximum of four lines 116 would be input into each OR gate 180—184. It should be understood that an OR gate need not be provided for every input line into instruction execution and control logic 112, since for certain instructions from two or more instruction sets several bit positions may be shared in common, and those portions of instruction execution control logic 112 whose operation is initiated by binary 1's in such bits will be made operative whenever binary 1's are present in such bit positions, whether in an instruction word from one instruction set or in an instruction word from another instruction set. For example, line 167 will become conductive whenever a binary 1 is received over lines 121 and 126 as inputs to IR 53. Since it is desired that line 167 become conductive whenever transistors 198 and 199 are turned on, regardless of the states of the remaining bits in the instruction word, there is no need to provide an OR gate between the instruction decoding logic 111 and the instruction execution control logic 112 or to provide an alternative input line to such OR gate.

ISMR 100 may have its contents altered, indicating a different instruction set is to be executed, either by an appropriate signal transmitted over control line 102 from instruction execution control logic 112, in response to an appropriate instruction loaded into IR 53, or directly from data bus 57, as a result of a memory access, for example, again under program control in response to a given instruction in IR 53. Alternatively, ISMR 100 may be loaded by any appropriate alternative means, such as a console switch.

Claims

1. A processor comprising a data bus for transmitting and receiving data, and a plurality of registers and logic circuits including an instruction register coupled to said data bus for temporarily storing an individual one of a plurality of instructions from two or more instruction sets, said instruction sets sharing in common at least one instruction representing an operation to be

control gates are coupled to those lines of IR 53 which are in a high state are turned on, whereas those transistors whose control gates are coupled to those lines in a low state remain turned off. Conductive paths are formed along particular ones of lines 116 to the instruction execution control logic 112, provided that all of the connecting transistors along such lines have been enabled by an appropriate combination of high level outputs from IR 53. For example, a conductive path is formed along line 163, since transistors 175, 178 and 179 are all turned on. OR gate 183 accordingly generates a high input into AND gate 107. Similarly, line 165 is rendered conductive, since transistors 177, 190, 191, and 192 are all turned on so that OR gate 184 generates a high input into AND gate 107. AND gate 107 accordingly generates a high input into NAND gate 109. Since neither transistor 194 or 195 on lines 160 and 161, respectively, is turned on, OR gate 182 generates a low input to inverter 108, which in turn generates a high input to NAND gate 109. NAND gate109 thus generates a low output for the instruction word 11011001 of the first instruction set.

If it is desired to achieve the same logical output from the represented portion 103 of the instruction execution control logic circuit 112, using an altogether different instruction word from a second instruction set, the state of ISMR is switched to a logical "1" so that output line 140 is high and line 150 is low. Assuming that 01110001 is the instruction word in the second instruction set which corresponds to instruction word 11011001 in thefirst instruction set, appropriate connecting transistors are provided along lines 160, 162, and 164, which serve as alternate inputs into OR gates 182—184, respectively, in order to generate the identical logical result at the output of logic circuit portion 103 for both instruction words. Line 164 goes high since transistors 176 and 193 are turned on, and consequently OR gate 184 is also turned on, transmitting a high level input into AND gate 107. Line 162 also becomes conductive when transistors 174 and 180 are turned on, so that OR gate 183 transmits a high level input into AND gate 107. Consequently AND gate 107 is turned on and generates a high level signal to NAND gate 109. Line 160 does not become conductive, since neither transistor 194 nor transistor 195 is enabled for this particular instruction word. As a result, OR gate 182 generates a low level input to inverter 108, which generates a high level input to NAND gate 109. NAND gate 109 thus generates a low level output, since both its inputs are high. Thus the ultimate output of logic circuit segment 103 is the same when instruction word 11011001 of the first instruction set or instruction word 01110001 of the second instruction set is present in IR 53. To ensure that identical results from each of the other logic circuit portions (not shown) of instruction execution control logic circuit 112 are achieved for these two different instruction words, appropriate connections are made between lines 116 and lines 140—148 and 150—158 from ISMR and IR 53 in a manner analogous to that just described with refernece to logic portion 103.

An OR gate 180—184 is provided for each input into the instruction execution control logic 112 for which it is desired to generate the identical control signal sequence over register and ALU control lines 23, 102, and 43 for different instruction words from different instruction sets. To the input of each such OR gate is connected a separate line corresponding to each different instruction set, the instructions of which one wishes to execute. In the example shown in Figure 2 two different instruction sets may be executed, depending upon the logical state of ISMR output lines 140 and 150. Accordingly, there are two lines input to each OR gate 180—184. If it were desired to execute four different instruction sets, then a maximum of four lines 116 would be input into each OR gate 180—184. It should be understood that an OR gate need not be provided for every input line into instruction execution and control logic 112, since for certain instructions from two or more instruction sets several bit positions may be shared in common, and those portions of instruction execution control logic 112 whose operation is initiated by binary 1's in such bits will be made operative whenever binary 1's are present in such bit positions, whether in an instruction word from one instruction set or in an instruction word from another instruction set. For example, line 167 will become conductive whenever a binary 1 is received over lines 121 and 126 as inputs to IR 53. Since it is desired that line 167 become conductive whenever transistors 198 and 199 are turned on, regardless of the states of the remaining bits in the instruction word, there is no need to provide an OR gate between the instruction decoding logic 111 and the instruction execution control logic 112 or to provide an alternative input line to such OR gate.

ISMR 100 may have its contents altered, indicating a different instruction set is to be executed, either by an appropriate signal transmitted over control line 102 from instruction execution control logic 112, in response to an appropriate instruction loaded into IR 53, or directly from data bus 57, as a result of a memory access, for example, again under program control in response to a given instruction in IR 53. Alternatively, ISMR 100 may be loaded by any appropriate alternative means, such as a console switch.

Claims

1. A processor comprising a data bus for transmitting and receiving data, and a plurality of registers and logic circuits including an instruction register coupled to said data bus for temporarily storing an individual one of a plurality of instructions from two or more instruction sets, said instruction sets sharing in common at least one instruction representing an operation to be

performed by said processor, said at least one instruction being represented by a different instruction word in each of said instruction sets, said instruction register generating a first set of binary signals in response to storing a first instruction word representation of said at least one instruction and generating a second set of binary signals in response to storing a second instruction word representation of said at least one instruction, an instruction set modifier register having a plurality of states each corresponding to one of said instruction sets, said instruction set modifier register generating a third set of binary signals representing a first one of said plurality of states and generating a fourth set of binary signals representing a second one of said plurality of states, instruction decoding logic comprising a first plurality of conductors responsive to said instruction register, a second plurality of conductors responsive to said instruction set modifier register, a third plurality of conductors intersecting said first and second plurality of conductors, a fourth plurality of conductors intersecting said first and second plurality of conductors, first means responsive to said first and third sets of binary signals for generating a first group of binary signals over said third plurality of conductors, second means responsive to said second and fourth sets of binary signals for generating a second group of binary signals over said fourth plurality of conductors, and instruction execution control logic having a plurality of control lines and controlling the operation of said plurality of registers and logic circuits by activating selected ones of said control lines, said instruction execution control logic being responsive to said third and fourth plurality of conductors and comprising means for activating the same ones of said control lines in response to either of said first and second group of signals.

2. A processor as claimed in claim 1 wherein said instruction set modifier register is coupled to said data bus and to one of said control lines and wherein the state of said instruction set modifier register is set in accordance with the execution of a predetermined instruction by said processor.

3. A processor as claimed in claim 1 wherein said instruction set modifier register comprises at least one bistable logic means.

4. A processor substantially as hereinbefore described and as shown in the accompanying drawings.

Printed for Her Majesty's Stationery Office by the Courier Press, Leamington Spa, 1982. Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.