SYSTEM 801

Principles of Operation

Version 2.5 November 22, 1976

IBM CONFIDENTIAL

801 Experimental Minicomputer Project
Computer Sciences Department
Thomas J. Watson Research Center
Yorktown Heights, N.Y.

Contents

1. System Architecture

Logically an 801 system consists of main storage, a central processing unit, low-speed and high-speed input/output devices, and an interrupt adapter. This structure is shown in Figure 1.1.

Main storage provides the system with directly accessible fast access storage of data. Both data and programs must be loaded into main storage (from input devices) before they can be processed.

No processing of data occurs in main storage, either implicitly or explicitly. All data must be loaded into high-speed CPU storage called registers before it can be operated upon.Between main storage and the registers, there is, on some 801 implementations, another storage level. This level, called **cache**, is split into two parts, one for data, the other for instructions.

Unlike many systems, the fetching of instructions and the fetching and storing of data are not tightly coupled. The CPU always attempts to prefetch one or more instructions. Hence, modification of an instruction by a program may not be seen when that instruction is fetched for execution, unless explicit steps have been taken to ensure that all pre-fetched instructions have been invalidated. For the purpose special instructions to control the caches have been provided.

Again, unlike many systems with caches, the data cache is not tightly coupled to the flow of data to or from input/output devices. For low-speed devices, which communicate directly with the CPU through its registers, this creates no problem. For high-speed input/output devices which may access memory directly, the program must ensure that, where necessary, updated data in cache is placed in storage prior to output, and that storage updated by input is correctly reflected in the data cache. Again, the cache control instructions can be used to guarantee correct results in these situations.

Fetching and storing of data by the CPU are only affected by any concurrent direct memory access input/output data transfer to the extent that the CPU may be stopped while awaiting memory response. When concurrent requests to a main storage location occur, access is normally granted in a predetermined sequence that assigns highest priority to input/output requests. If the first reference changes the contents, any subsequent storage fetches obtain the new contents, although, as noted above, care must be taken to ensure the synchronization of data and instruction caches where concurrent accesses are possible.

Main storage may be volatile or nonvolatile. If it is volatile, the contents of main storage are not preserved when power is turned off. If it is nonvolatile, turning power on or off does not affect the contents of main storage provided the CPU is in the stopped state and no references are made to main storage by input/output devices which can directly access storage. The organization of storage is shown in Figure 1.2.

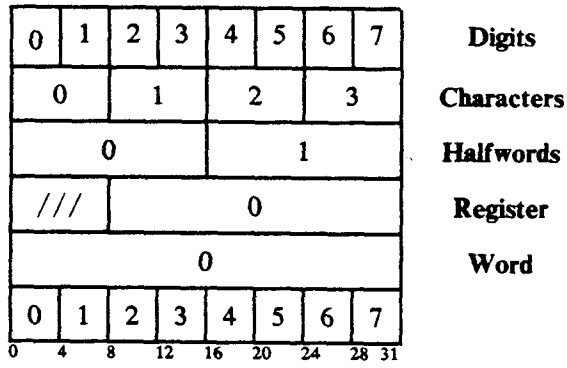


Figure 1.2. **Storage Organization**

1.1 Central Processing Unit

The central processing unit (CPU) is the controlling center of the system. It contains the sequencing and processing controls for instruction execution, interruption action, initial program loading, and other system-related functions.

The CPU, in processing instructions, attempts to achieve the greatest instruction processing rate possible. Instructions which do not require storage access may be executed while storage is being accessed for some previous instruction, and the CPU attempts to pre-fetch instructions whenever possible, overlapping such fetches with the execution of other instructions.

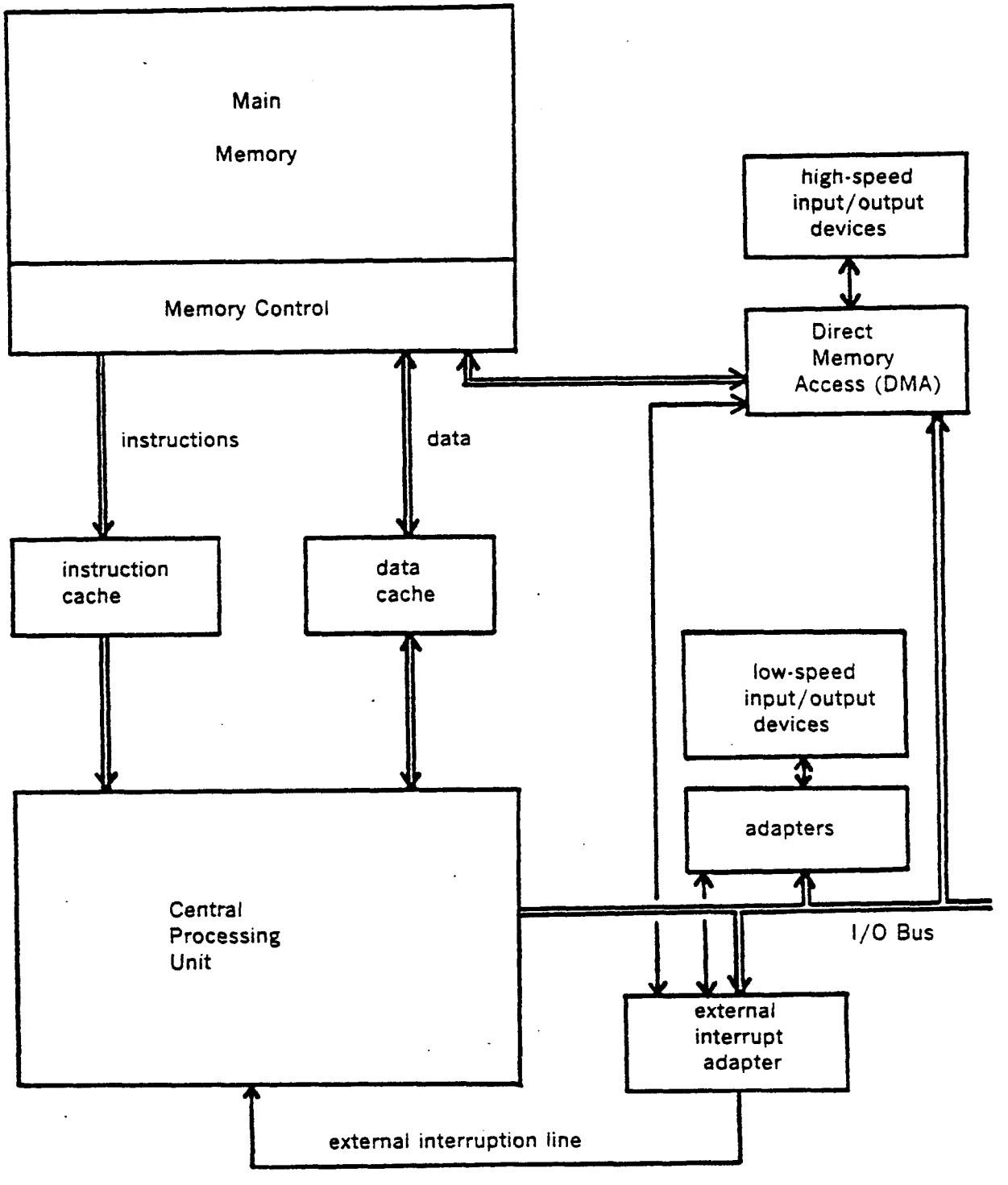


Figure 1.1. **System Organization**

Since a successful branch instruction requires the fetching of an instruction out of sequential order, means are provided to permit execution of one instruction while fetching the target instruction of the branch. Control is maintained over such pre-fetching so that instruction execution appears to take place in the order intended by the programmer.

Instructions which the CPU can execute fall into ten classes; storage access, address computation, branching, trap, move, arithmetic, logical operation, shift, system control, and input/output instructions. A separate section is devoted to each instruction class.

1.2 Register Organization

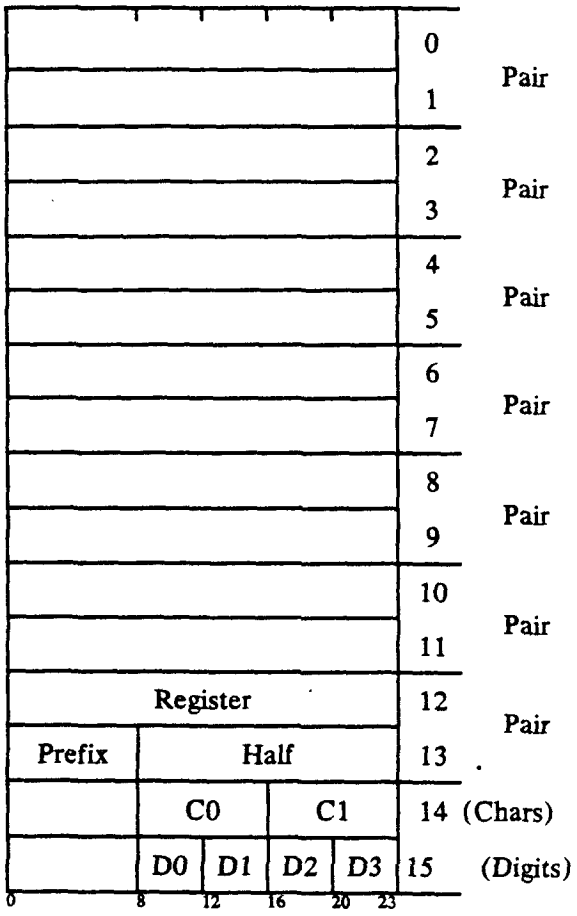
All manipulation of information is done in high-speed storage internal to the central processing unit (CPU). The principal storage internal to the CPU is a set of sixteen **registers**. Each register consists of 24 **bits**. These registers contain a **prefix** of eight bits and a **half** of sixteen bits. The half consists of two **characters, C0** and **C1**, of eight bits each. The half is also considered to consist of four **digits, D0, D1, D2,** and **D3**, each containing four bits. The register organization is shown in Figure 1.3.

To avoid the destruction of operands in certain operations, some instructions provide for the result of the operation to be placed in the **twin**, of a **pair**, of one of the source operand registers. The register twin to a given register has the name, in binary, of the given register, with the low-order bit inverted. Thus, the twin of register 5 (binary 0101) is register 4 (binary 0100), and the twin of register 14 (binary 1110) is register 15(binary 1111).

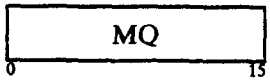
For computational purposes, registers are treated as signed algebraic quantities, unsigned positive quantities, or, where required, as unstructured logical quantities. In a register, an algebraic quantity consists of twenty-four bits in two's complement representation.

Three additional special-purpose registers exist within the CPU. These registers are known as (a) the instruction address register (IAR), (b) the MQ-register, and (c) the condition register (CR). These registers are implicitly or explicitly addressed for a particular action or operation, and they may be accessed, used, or altered by a multiplicity of instructions.

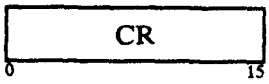
General Purpose Registers



The MQ Register



The Condition Register



Instruction Address Register

Figure 1.3. **Register Organization**

1.2.1 The Instruction Address Register

The instruction address register (IAR), as shown in Figure 1. 4, is a 24-bit register which normally contains the address of the next instruction to be executed. Since all instructions are constrained to lie on half-word boundaries, the low-order bit (bit 23) of the instruction address register is constrained to be zero.

As a rule, the content of the instruction address register is incremented by the length of the current instruction during the process of decoding that instruction. Should this instruction be a successful branch instruction, the content of the instruction address register will be changed to the address of the branch target instruction, as given by the branch instruction.

1.2.2 The MQ Register

The MQ-register (MQ) is a 16-bit register whose primary use is to provide a register extension to accommodate the product for the Multiply Step instruction and the dividend for the Divide Step instruction.

1.2.3 The Condition Register

The condition register (CR) is a 16-bit register used to reflect the effect of certain operations, to provide a mechanism for testing (and branching) on a bit or condition, and to provide a 'parity stack' to indicate which bytes of the last four half words loaded were addressed. The condition register is shown in Figure 1.4.

The first four bits of the condition register are used for the parity stack (SX). Bit 0 is known as SX0, bit 1 as SX1, bit 2 as SX2, and bit 3 as SX3. An effect of an instruction that loads the half of a register is to push the parity stack down one position. The lowest bit of the stack (SX3) is lost, while the low-order bit of the storage address is placed on the top of the stack, above the previous top three stack elements.

Bit 4 of the condition register is set by the IOR and IOW instructions. It is set to one if the I/O adapter selected by one of these instructions cannot accept the command. It is set to zero if the adapter accepts the command.

Bit 5 of the condition register is the decimal exception latch (DX). If the decimal feature is installed, this bit is set by the decimal instructions (Add and Subtract Decimal) to one or zero if an exception condition is or is not, respectively, detected. *In the initial 801 implementation, the decimal feature is not installed and this bit is reserved. It is set to zero whenever the condition register is loaded.*

Bit		
0	SX0	Parity Stack Zero
1	SX1	Parity Stack One
2	SX2	Parity Stack Two
3	SX3	Parity Stack Three
4	IO	I/O Busy
5	DX	Decimal Exception
6	EN	Previous Enable State
7	LT	Compares Less Than, Neg. Value
8	EQ	Compare Equal, Zero Value
9	GT	Greater Than or Positive
10	C0	Carry from Bit 0 or C0 =
11	C1	Carry from Bit 8 or C1 =
12	OV	Overflow
13	SO	Summary Overflow
14	PZ	Permanent Zero
15	TB	Test Bit

Figure 1.4. **Condition Register Bits**

Bit 6 of the condition register is the previous enable state latch (EN). It is set by the enable and disable instructions to reflect the enable state previous to the execution of these instructions. If the processor had been enabled, this bit is set to one; if it had been disabled, it is set to zero.

Bit 7 of the condition register is the less-than latch (LT). It is set to one by comparison operations if the first comparand is less than the second comparand. It is set to one by certain other arithmetic and logical operations if the result is negative or if the high-order bit of the result is one.

Bit 8 of the condition register is the equal latch (EQ). It is set to one by comparison operations if the first comparand equals the second comparand. It is set to one by certain other logical and arithmetic operations if the result is zero, or if all bits of the result are zeros.

Bit 9 of the condition register is the greater-than latch (GT). It is set to one by comparison operations if the first comparand is greater than the second comparand. It is set to one by certain other arithmetic and logical operations if the re-

sult is positive or if the high-order bit of a non-zero result is zero.

Bit 10 of the condition register is the carry-zero latch (C0). It is set to one by certain arithmetic instructions if the operation generates a carry out of bit position zero. It also functions as a special-purpose indicator for the Divide Step and Multiply Step instructions. This latch is set by logical compare instructions to show equality/inequality of character C0 of the comparands.

Bit 11 of the condition register is the carry-one latch (C1). It is set to one by certain arithmetic instructions if the operation generates a carry out of bit position eight. This latch is also set by logical compare instructions to show equality/inequality of character C1 of the comparands.

Bit 12 of the condition register is the overflow latch (OV), which is set to one when arithmetic and certain shift operations overflow. It also functions as a special purpose indicator for the Divide Step instruction.

Bit 13 of the condition register is the summary-overflow latch (SO). Whenever an instruction sets the overflow latch, it resets the summary-overflow latch to the OR of the overflow-latch with the old value of the summary-overflow latch.

Bit 14 of the condition register is the permanent-zero bit (PZ). It is set to zero whenever the condition register is loaded, and it cannot be reset to one. Its presence provides for a guaranteed branch in the BI format by use of the Branch On Not-Bit instruction, where the permanent zero bit is specified.

Bit 15 of the condition register is the test bit (TB). A bit may be moved to or from an arbitrary bit position in a half from or to the test bit of the condition register through use of the Move From/To Test Bit instructions.

All bits of the condition register, save those required to be zeros, can be arbitrarily set through use of the Move To Condition Register instruction. Additionally, any individual bit of the condition register may be set to an arbitrary value by use of the Insert Condition Bit Immediate instruction, except, of course, those bits required to be zero.

1.3 Instruction Formats

The five instruction formats, X, R, D, BI, and BA, are shown in Figure 1.5. For X and D instructions that refer to storage, and for I/O instructions, address calculation is according to the formulas:

X-Format (RB) + 0|(RC)

D-Format 0|(RC) + (0(bits 0-7) || I)

where 0|(RC) indicates the value 0 if RC is specified as 0, else the contents of register RC. I is treated as an unsigned 16 bit integer.

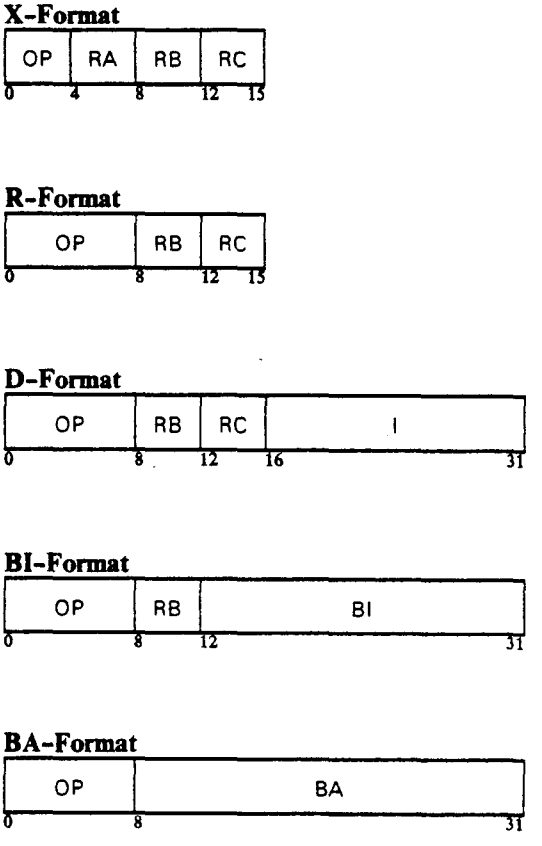


Figure 1.5. **Instruction Formats**

1.4 Interrupts

An interrupt consists of:

1. setting information about the machine state at interrupt into the Interrupt Save Byte (ISB),

2. disabling the processor,

3. storing an instruction address into a particular location in main memory, and

4. resetting the IAR to the address of a particular instruction location.

The Interrupt Save Byte (ISB) is used to save information about the machine state upon an interrupt. The first four bits (bits 0 - 3) have the same meaning for all interrupts, whereas the second four bits (bits 4 - 7) are used for interrupt codes specific to the interrupt. Bit 0 is used to store the enable state at interrupt (1 = enabled, 0 = disabled), bit 1 the length code for the instruction just executed (1 = 32 bits, 0 = 16 bits). Bit 2 is reserved for future use and bit 3 indicates whether or not the interrupted instruction was the subject instruction of a branch with execute instruction (1 = a subject instruction, 0 = not a subject instruction). The specific use of bits 4 - 7 by the different interrupts has not yet been defined. The content of the ISB can be accessed by using the Move From ISB and Condition Register instruction (MFICR) to place the value of the ISB into the prefix of a specified register, along with the value of the condition register into the half of the specified register.

The Interrupt Save Byte (ISB)

Bit	
0	Enable State
1	Instruction Length
2	Reserved
3	Subject Instruction
4	Interrupt Code 0 (IC0)
5	Interrupt Code 1 (IC1)
6	Interrupt Code 2 (IC2)
7	Interrupt Code 3 (IC3)

Figure 1.6 **Interrupt Save Byte**

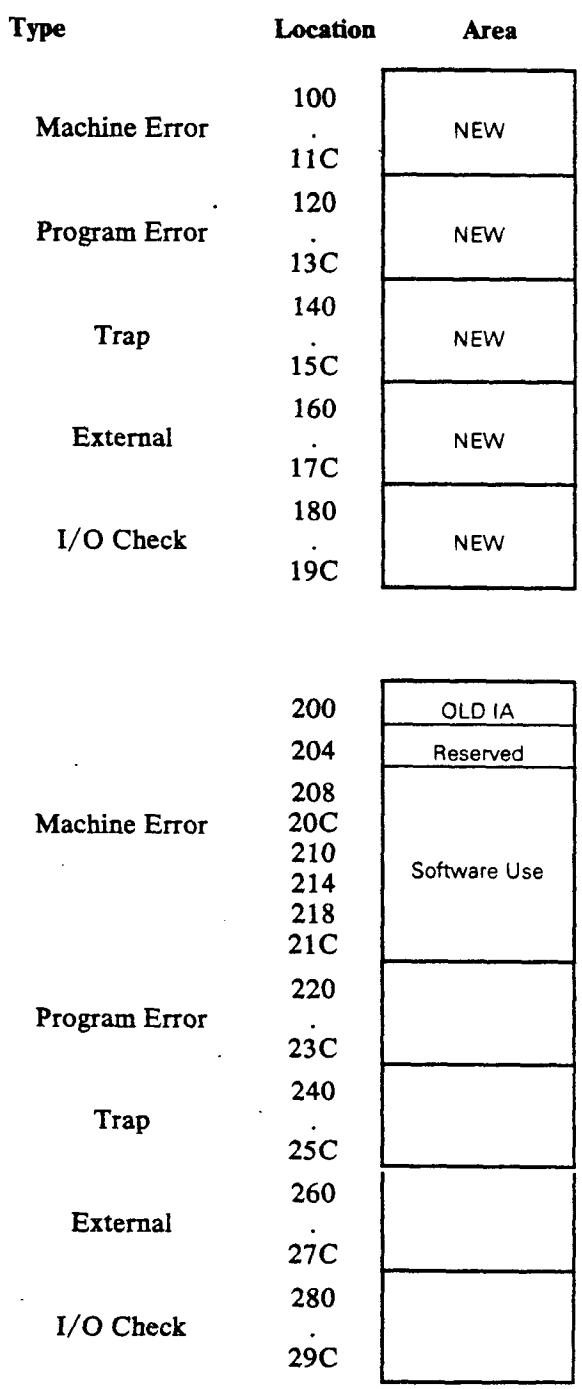


Figure 1.7. **Interrupt Areas**

Each interrupt type has an area of 32 bytes for storing the old IA and other information associated with the interrupted state, and an area of 32 bytes (8 words) to which control is transferred when the interrupt occurs. Both of these areas may or may not be in the data or instruction cache, respectively, when the interrupt occurs. The organization of these interrupt areas is shown in Figure 1.7.

The 32 byte areas called NEW are the locations to which control is passed when an interrupt occurs. Presumably, a branch instruction will be placed in an area. The word labeled OLD IA is the location into which the old IA is stored. The reserved word is saved for possible future use by the hardware to provide more information about the interrupt. The remaining six words are usable by the software.

The particular address stored in the OLD IA location depends on the instruction being executed and the type of interrupt. Normally, the address stored in the OLD IA is that of the next instruction to be executed, and the interrupted program can be resumed at that address. The exceptions to this general rule are shown in Figure 1.8. For some of these exception cases, it may be possible to reconstruct the machine state with the help of the information in the ISB.

Error	Instruction	Old IA
undef.op.	all	instruction executed
out of range data address	subject instruction	instr. following subject instruction
out of range instruction address	all	instruction executed
illegal subject instruction	illegal subject instruction	instruction executed
processor failure	all	undefined

Figure 1.8. **Old IA Exception Cases**

The following is a description of each of the interrupt types.

Machine Error - All processor and I/O machine failures are reported with this interrupt.

Program Error - Program errors are reported with this interrupt. These include:
- Out of range load/store address
- Out of range instruction address (an address 'too close' to the top of memory
- Undefined operation
- Illegal subject instruction following a branch and execute.

Trap - The trap instructions use this interrupt to report a successful comparison. The old IA will be the instruction following the trap.

External - A signal from the external interrupt adapter while the processor is enabled causes this interrupt. The interrupt will never occur between a branch and execute, and its following instruction, which are treated as a single operation.

I/O Check - A time-out or an error in the adapter interface during an I/O read or write will cause the operation to be suppressed and an I/O check interrupt to be taken. The old IA location will contain the address of the I/O instruction which failed. The interrupt will not occur between a branch and execute, and its subject instruction, which are treated as a single operation.

1.5 Input and Output

Input/Output (I/O) operations involve the transfer of information between main storage or the CPU and an I/O adapter. I/O adapters attach I/O devices to the CPU via an I/O Bus which operates at approximately 801 speed. These adapters also connect to a special adapter, called the External Interrupt Adapter, that collects all the interrupt requests from the other adapters, and presents them to the 801 through the external interrupt line. Some high-speed I/O devices also attach through Direct Memory Access (DMA) directly to main memory for the direct transfer of data at high data rates.

The 801 I/O Subsystem is described in detail in Section 12.

1.5.1 The External Interrupt Adapter

The interrupt adapter is the common link for all interrupts from the I/O adapters to the CPU. It accepts requests for service from the various attached I/O devices, and, when the CPU is enabled for external interruptions, presents them to the CPU.

The External Interrupt Adapter itself appears to the CPU like an I/O device, with various functions depending upon the particular model of the adapter.

1.5.2 Input/Output Interface

Communications between an adapter and the CPU is under program control. While all adapters, including the External Interrupt Adapter, attach to the common I/O bus, control sequences are, in general, unique to a particular adapter. These sequences, and their responses, are provided through the instructions Input-Output Read and Input-Output Write, which transmit to a specified adapter a 16 bit address/command field, and attempt to accept or transmit 16 bits of data or control information in a specified register. Hence, apart from the commonality of the I/O bus and the interrupt adapter, the interface between each device and the CPU is essentially a programmed interface.

For Direct Memory Access that attaches high-speed I/O devices directly to memory, a given control sequence is required to initiate the direct transfer of a block of data between main memory and the device. Once initiated, the data transfer proceeds without CPU intervention. The end of the data transfer is signalled to the CPU by an external interrupt from the DMA. In such circumstances the device shares memory with other active directly attached devices and the CPU.

2. Storage Access

Storage is organized as a sequence of 32-bit words, each consisting of four 8-bit bytes. Bytes in storage are consecutively numbered, left to right, starting with zero. Each number is considered the address of the corresponding byte. All addresses are computed as byte addresses. Storage addressing wraps around from the maximum byte address, 16,777,215, to address zero. If less than the maximum amount of storage is installed, an attempt to utilize a byte from a non-existent storage location will result in an address exception condition.

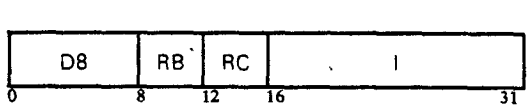
All storage accesses are for a word or multiples thereof. Accesses for a register cause the three low-order bytes of a word to be fetched or stored. Accesses for a half cause the high- or low-order half-word of a word, as required, to be fetched or stored. Accesses for instructions may require the fetching of a word, a half-word, or the low-order half-word of a word followed by the high-order half-word of the next consecutive word in storage. Half-word or word addresses are generated, respectively, by ignoring the low-order one or two bits of a byte address.

If a cache memory for data references is installed, accesses to or from cache to storage occur in multiples of words. Because instruction fetch, storage access, and register access are overlapped in the execution of load and store instructions, interrupts, such as for a bad effective address, may be imprecise.

2.1 Instructions

Load Half Algebraic, D-form

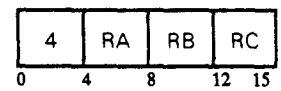
LHAD RB,RC,I



The half (chars C0 and C1) of the register specified by RB is replaced by the half word of storage addressed by 0 | (RC) + I. The resulting sign bit is extended through the prefix of register RB. The parity stack in the condition register is pushed down, and the condition register bit SX3 is lost. Condition register bit SX0 assumes the value of the low-order bit of the storage address.

Load Half Algebraic, X-form

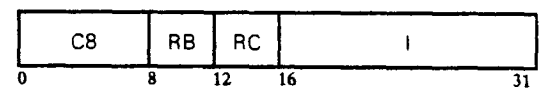
LHAX RA,RB,RC



The half (chars C0 and C1) of the register specified by RA is replaced by the half word of storage addressed by (RB) + 0 | (RC). The resulting sign bit is extended through the prefix of register RA. The parity stack in the condition register is pushed down, and the condition register bit SX3 is lost. Condition register bit SX0 assumes the value of the low-order bit of the storage address.

Load Half Zero, D-form

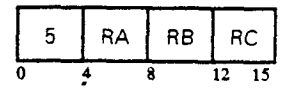
LHZD RB,RC,I



The half (chars C0 and C1) of the register specified by RB is replaced by the half word of storage addressed by 0 | (RC) + I. The prefix of register RB is set to zeros. The parity stack in the condition register is pushed down, and the condition register bit SX3 is lost. Condition register bit SX0 assumes the value of the low-order bit of the storage address.

Load Half Zero, X-form

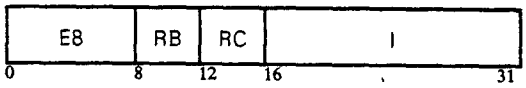
LHZX RA,RB,RC



The half (chars C0 and C1) of the register specified by RA is replaced by the half word of storage addressed by (RB) + 0 | (RC). The prefix of register RA is set to zeros. The parity stack in the condition register is pushed down, and the condition register bit SX3 is lost. Condition register bit SX0 assumes the value of the low-order bit of the storage address.

Load, D-form

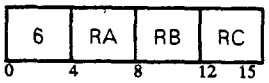
LD RB,RC,I



The content of the register specified by RB is replaced by characters 1,2 and 3 of the word in storage addressed by 0 | (RC) + I.
Note: This instruction does not affect the condition register in order to be able to preserve the machine state when processing an interrupt.

Load, X-form

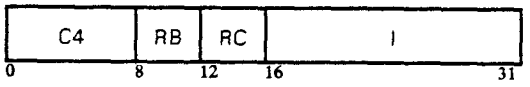
LX RA,RB,RC



The content of the register specified by RA is replaced by characters 1,2 and 3 of the word in storage addressed by (RB) + 0 | (RC).
Note: This instruction does not affect the condition register in order to be able to preserve the machine state when processing an interrupt.

Store Character, D-form

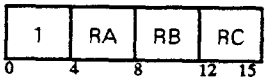
STCD RB,RC,I



The char of storage addressed by 0 | (RC) + I is replaced by char C1 of the register specified by RB.

Store Character, X-form

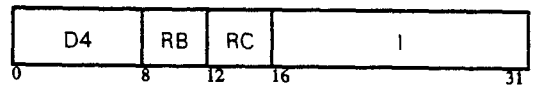
STCX RA,RB,RC



The char of storage addressed by (RB) + 0 | (RC) is replaced by char C1 of the register specified by RA.

Store Half, D-form

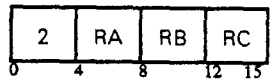
STHD RB,RC,I



The half word of storage addressed by 0 | (RC) + I is replaced by the half of the register specified by RB.

Store Half, X-form

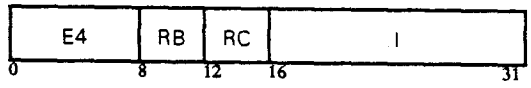
STHX RA,RB,RC



The half word of storage addressed by (RB) + 0 | (RC) is replaced by the half of the register specified by RA.

Store, D-form

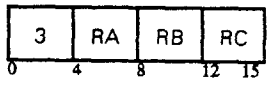
STD RB,RC,I



Chars 1, 2, and 3 of the word in storage addressed by 0 | (RC) + I is replaced by the content of the register specified by RB.

Store, X-form

STX RA,RB,RC



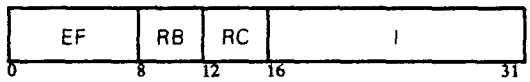
Chars 1, 2, and 3 of the word in storage addressed by (RB) + 0 | (RC) is replaced by the content of the register specified by RA.

3. Address Computation

3.1 Instructions

Compute Address, D-form

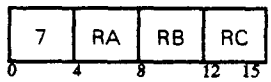
CAD RB,RC,I



The address specified by 0 | (RC) + I replaces the content of register RB. No storage references for operands occur, and the address is not inspected for address exceptions.

Compute Address, X-form

CAX RA,RB,RC



The address specified by (RB) + 0 | (RC) replaces the content of register RA. No storage references for operands occur, and the address is not inspected for address exceptions.

4. Branching

The normal sequential execution of instructions may be changed by the use of the branch instructions in order to perform subroutine linkage, decision making, and loop control.

Subroutine linkage is provided by branch and link instructions: BALA, BALAX, BALR, BALRX, BALI, and BALIX. These instructions permit not only the introduction of a new instruction address, but also preservation of the return address in an implicitly or explicitly designated register. In every case, the new instruction address, the address of the branch target instruction, is computed before the return address is saved. For the regular forms of the instruction, the return address is the address of the byte immediately following the Branch And Link instruction; for the execute forms of the instruction, the return address is the full word boundary on or preceding the location six bytes beyond the instruction immediately following the branch and link with execute instruction. In the latter case, the register containing the return address is available to the subject instruction. Note that when 0 is specified as register RC in the R-form Branch And Link instructions, the branch address is taken from register 0. A separate instruction, Move From Instruction Address Register, is provided for obtaining the current instruction address.

Facilities for decision making are provided by the branch on bit and branch on not bit conditional branch instructions: BB, BBX, BBR, BBRX, BNB, BNBX, BNBR, and BNBRX. These instructions provide the capability of branching or not according to any specified state of any bit of the condition register. Loop control can also be accomplished through use of these instructions to test the outcome of address arithmetic and counting operations.

4.1 Invalid Branch Addresses

If a branch specifies an invalid storage location as the address of the branch target instruction, the address exception condition is not set until an attempt is made to execute the branch target instruction.

4.2 Branch With Execute Instructions

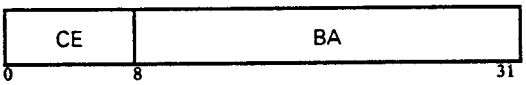
For every branch instruction, there is a corresponding branch with execute form of the instruction. The instruction immediately following a branch with execute instruction is called the subject instruction. Whether or not the branch is taken, the subject instruction is executed. The execution of the branch and of the subject instruction is considered as a single unit. If an interrupt occurs at any time during the execution of the branch and its subject instruction, the machine state will be left as if the subject instruction did not execute and the old instruction address will be that of the branch. An interruption during a branch, link, and execute, however, may or may not leave the link address in the specified register.

Certain instructions are not allowed to follow a branch and execute instruction. These are branch instructions, trap instructions, cache control instructions, and I/O instructions.

4.3 Instructions

Branch Absolute and Link

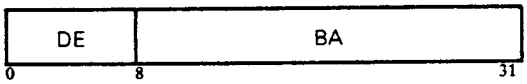
BALA BA



The content of register 15 is replaced by the updated instruction address, and then the updated instruction address is replaced by BA, with its low order bit (bit 23) forced to zero.

Branch Absolute and Link with Execute

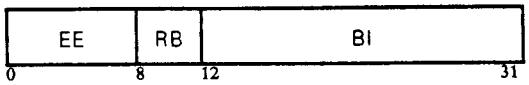
BALAX BA



The content of register 15 is replaced by the updated instruction address incremented by six and set to the preceding full word boundary, the instruction immediately following the branch instruction is executed while the the updated instruction address is replaced by BA, with its low order bit (bit 23) forced to zero.

Branch and Link, I-form

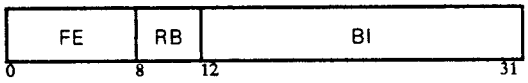
BALI RB,BI



The content of register RB is replaced by the updated instruction address, and then bits 3-22 of the updated instruction address are replaced by BI.

Branch and Link with Execute, I-form

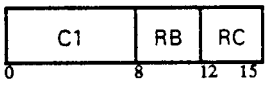
BALIX RB,BI



The content of register RB is replaced by the updated instruction address incremented by six and set to the preceding full word boundary, the instruction immediately following the branch instruction is executed while bits 3-22 of the updated instruction address are replaced by BI.

Branch and Link, R-form

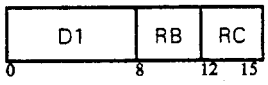
BALR RB,RC



The content of register RB is replaced by the updated instruction address. The updated instruction address is replaced by the content of register RC, with its low-order bit (bit 23) set to zero.

Branch and Link with Execute, R-form

BALRX RB,RC

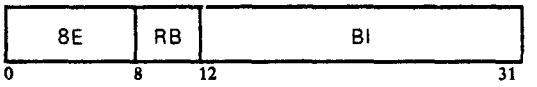


The content of register RB is replaced by the updated instruction address incremented by six and set to the preceding full word boundary. The instruction immediately following the branch

instruction is executed while the the updated instruction address is replaced by the content of register RC, with its low-order bit (bit 23) set to zero.

Branch on Bit

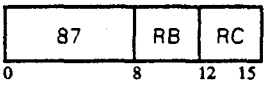
BB RB,BI



If the condition register bit specified by RB is one, bits 3-22 of the updated instruction address are replaced by BI. If the condition bit is zero, the updated instruction address is unaltered, and no branch occurs.

Branch on Bit, R-form

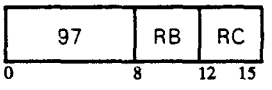
BBR RB,RC



If the condition register bit specified by RB is one, the updated instruction address is replaced by the content of the register specified by RC, and the low-order bit is forced to zero. If the condition bit is zero, the updated instruction address is unaltered, and no branch occurs.

Branch on Bit and Execute, R-form

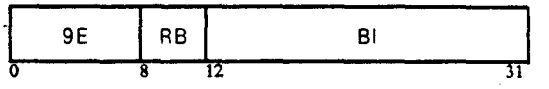
BBRX RB,RC



If the condition register bit specified by RB is one, the following instruction is executed while the updated instruction address is replaced by the content of the register specified by RC, with the low-order bit forced to zero. If the condition bit is zero, the updated instruction address is unaltered, and no branch occurs.

Branch on Bit and Execute

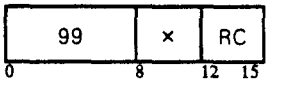
BBX RB,BI



If the condition register bit specified by RB is one, the following instruction is executed while bits 3-22 of the updated instruction address are replaced by BI. If the condition bit is zero, the updated instruction address is unaltered, and no branch occurs.

Branch, Execute and Enable

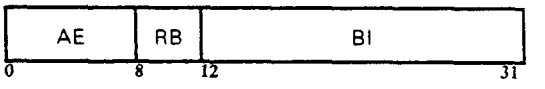
BEX RC



The instruction immediately following the BEX instruction, this following instruction called the subject instruction, is executed while the updated instruction address register is replaced by the content of the register specified by RC, with the low-order bit forced to zero. Upon completion of the subject instruction the machine becomes enabled.

Branch on Not Bit

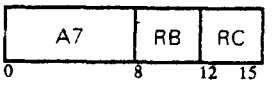
BNB RB,BI



If the condition register bit specified by RB is zero, bits 3-22 of the updated instruction address are replaced by BI. If the condition bit is one, the updated instruction address is unaltered, and no branch occurs.

Branch on Not Bit, R-form

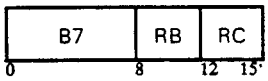
BNBR RB,RC



If the condition register bit specified by RB is zero, the updated instruction address is replaced by the content of the register specified by RC, with the low-order bit forced to zero. If the condition bit is one, the updated instruction address is unaltered, and no branch occurs.

Branch on Not Bit and Execute, R-form

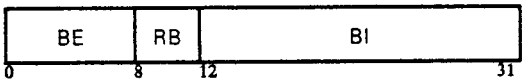
BNBRX RB,RC



If the condition register bit specified by RB is zero, the following instruction is executed while the updated instruction address is replaced by the content of the register specified by RC, with the low-order bit forced to zero. If the condition bit is one, the updated instruction address is unaltered, and no branch occurs.

Branch on Not Bit and Execute

BNBX RB,BI



If the condition register bit specified by RB is zero, the following instruction is executed while bits 3-22 of the updated instruction address are replaced by BI. If the condition bit is one, the updated instruction address is unaltered, and no branch occurs.

5. Traps

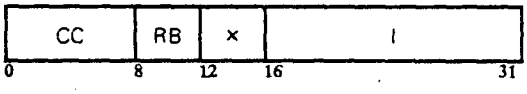
The trap instructions are provided to test for a specified set of conditions. If the conditions tested by a trap instruction are met, a trap exception condition is generated, and an interruption occurs. Control is transferred to the special interrupt area associated with trap interrupts (see section 1.4). If the tested conditions are not met, instruction execution continues with the next sequential instruction. Trap instructions may not appear as the subject instruction of a branch with execute.

The comparisons are performed on operands treated as 24 bit unsigned integers.

5.1 Instructions

Trap if Register Greater Than, Immediate

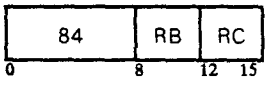
TGTI RB,I



If the content of the register specified by RB is greater than the value of the field I, extended on the left with eight zeros, a trap exception condition is generated.

Trap if Register Greater Than or Equal

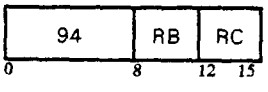
TGTE RB,RC



If the content of register RB is greater than or equal to the content of register RC, a trap exception condition is generated.

Trap if Register Less Than

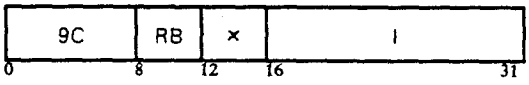
TLT RB,RC



If the content of register RB is less than the content of register RC, a trap exception condition is generated.

Trap if Register Less Than, Immediate

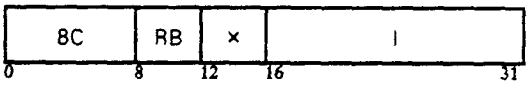
TLTI RB,I



If the content of the register specified by RB is less than the value of the field I, extended on the left with eight zeros, a trap exception condition is generated.

Trap if Register Not Equal, Immediate

TNEI RB,I



If the content of the register specified by RB is not equal to the value of the field I, extended on the left with eight zeros, a trap exception condition is generated.

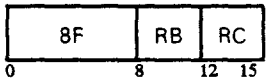
6. Moves And Inserts

This group of instructions is concerned solely with the movement of data between registers and with the insertion of data from the immediate field of an instruction into a register. Except when data is moved or inserted into the condition register, none of these instructions alter the condition register or generate exception conditions.

6.1 Instructions

Insert Condition Bit Immediate

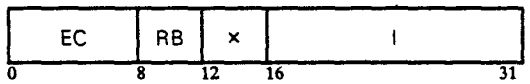
ICBI RB,RC



The condition register bit specified by RB is set equal to the low order bit of RC. If the bit specified by RB is a permanent-zero bit, the value of the bit is unchanged.

Insert Prefix Immediate

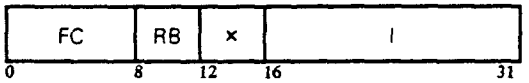
IPI RB,I



The prefix of the register specified by RB is replaced by the eight low order bits of I (bits 24-31 of the instruction.

Insert Prefix Immediate and Zero

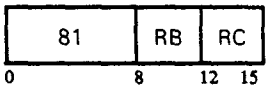
IPIZ RB,I



The prefix of the register specified by RB is replaced by the eight low order bits of I (bits 24-31 of the instruction. The half of the register specified by RB is set to zeros.

Move Character Zero from Zero

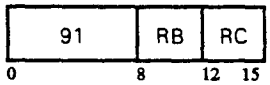
MC00 RB,RC



Char C0 of the register specified by RB is replaced by char C0 of the register specified by RC.

Move Character Zero from One

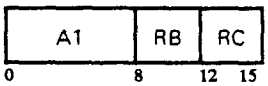
MC01 RB,RC



Char C0 of the register specified by RB is replaced by char C1 of the register specified by RC.

Move Character One from Zero

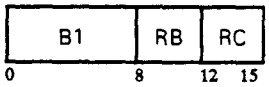
MC10 RB,RC



Char C1 of the register specified by RB is replaced by char C0 of the register specified by RC.

Move Character One from One

MC11 RB,RC



Char C1 of the register specified by RB is replaced by char C1 of the register specified by RC.

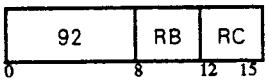
Move from Character Indexed by SX0

MFC0 RB,RC

Char C1 of the register specified by RB is replaced by char C0 or char C1 of the register specified by RC, as condition register bit SX0 is, respectively, zero or one. Char C0 of the register specified by RB is set to zero.

Move from Character Indexed by SX1

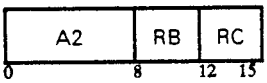
MFC1 RB,RC



Char C1 of the register specified by RB is replaced by char C0 or char C1 of the register specified by RC, as condition register bit SX1 is, respectively, zero or one. Char C0 of the register specified by RB is set to zero.

Move from Character Indexed by SX2

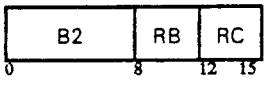
MFC2 RB,RC



Char C1 of the register specified by RB is replaced by char C0 or char C1 of the register specified by RC, as condition register bit SX2 is, respectively, zero or one. Char C0 of the register specified by RB is set to zero.

Move from Character Indexed by SX3

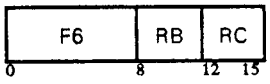
MFC3 RB,RC



Char C1 of the register specified by RB is replaced by char C0 or char C1 of the register specified by RC, as condition register bit SX3 is, respectively, zero or one. Char C0 of the register specified by RB is set to zero.

Move from Digit

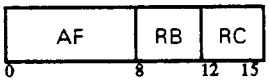
MFD RB,RC



A digit of register RB is selected by bits 22-23 of register RC. This digit is placed in digit D3 of RB and the remainder of RB is set to zero.

Move from Digit Paired

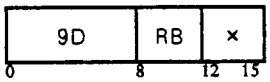
MFDP RB,RC



A digit of register RB is selected by bits 22-23 of register RC. This digit is placed in digit D3 of the twin, in a register pair, of RB and the remainder of the twin is set to zero.

Move from Instruction Address

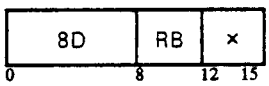
MFIA RB



The content of register RB is replaced by the value of the current instruction address, i.e., the location of this instruction.

Move from ISB and Condition Register

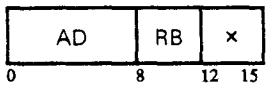
MFICR RB



The content of the prefix of register RB is replaced by the content of the interrupt save byte (ISB), and the content of the half of register RB is replaced by the content of the condition register.

Move from MQ

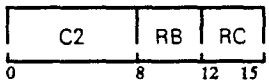
MFMQ RB



The content of the half of register RB is replaced by the content of the MQ register. The prefix of register RB is set to zeros.

Move from Prefix

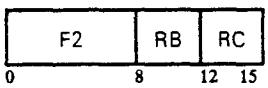
MFP RB,RC



Char C1 of the register specified by RB is replaced by the prefix of the register specified by RC. The prefix and char C0 of the register specified by RB is set to zeros.

Move from Test Bit

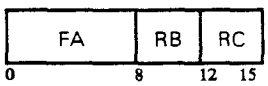
MFTB RB,RC



A particular bit of the half of register RB is set to the value of the condition register test bit. The particular bit of the half of register RB is specified by the value of digit D3 of register RC.

Move from Test Bit Immediate

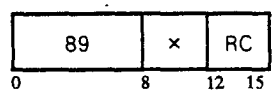
MFTBI RB,RC



The bit of the half of register RB specified by RC is set to the value of the condition register test bit.

Move to Condition Register

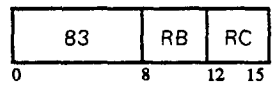
MTCR RC



Those bits of the condition register not reserved and/or required to be zero are set to the values of the corresponding bits of the half of register RC.

Move to Character Indexed by SX0

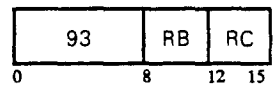
MTC0 RB,RC



Char C0 or C1 of the register specified by RB is replaced by char C1 of register RC, as condition register bit SX0 is zero or one, respectively.

Move to Character Indexed by SX1

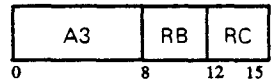
MTC1 RB,RC



Char C0 or C1 of the register specified by RB is replaced by char C1 of register RC, as condition register bit SX1 is zero or one, respectively.

Move to Character Indexed by SX2

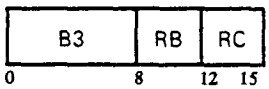
MTC2 RB,RC



Char C0 or C1 of the register specified by RB is replaced by char C1 of register RC, as condition register bit SX2 is zero or one, respectively.

Move to Character Indexed by SX3

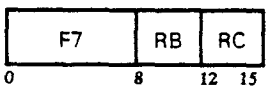
MTC3 RB,RC



Char C0 or C1 of the register specified by RB is replaced by char C1 of register RC, as condition register bit SX3 is zero or one, respectively.

Move to Digit

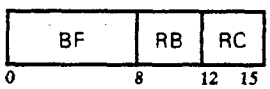
MTD RB,RC



The digit of register RB specified by bits 22-23 of register RC is replaced by digit D3 of register RB.

Move to Digit Paired

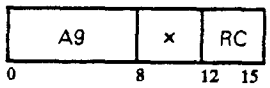
MTDP RB,RC



The digit of the twin, in a register pair, of RB selected by bits 22-23 of register RC is replaced by digit D3 of register RB.

Move to MQ

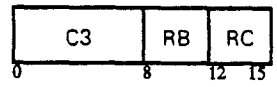
MTMQ RC



The content of the MQ register is replaced by the half of register RC.

Move to Prefix

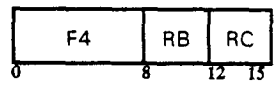
MTP RB,RC



The prefix or the register specified by RB is replaced by char C1 of the register specified by RC.

Move to Test Bit

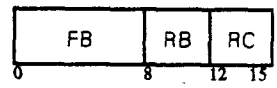
MTTB RB,RC



The condition register test bit is set to the value of a bit of the half of register RB. This bit is specified by digit D3 of register RC.

Move to Test Bit Immediate

MTTBI RB,RC



The condition register test bit is set to the value of a bit of the half of register RB, this bit specified by RC.

7. Arithmetic

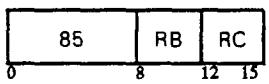
The arithmetic operations, with the exception of the decimal operations Add Decimal and Subtract Decimal, treat registers as consisting of a 24 bit quantity in two's complement representation. On these operations, the LT, EQ, and GT bits in the condition register are set to reflect the result: LT is set to one if the result has a negative sign, EQ if the result is zero, and GT if the result has a positive sign and is not zero. Condition register bit C0 is set to reflect the carry out of bit position zero and C1 the carry out of bit position eight. The overflow latch, OV, is set to one if the carry out of bit position one is not equal to the carry out of bit position zero. The summary overflow bit (SO) is set to the OR of the new value of OV with the old value of SO.

The extended operations use the value of the C1 bit to determine the result. The extended add instructions, AE and AEI, add the value of the C1 bit to the sum of the two operands to determine the result. In the extended subtract instructions, SE and SEI, the value of the first operand is added to the complement of the second operand and to this result is added the value of the C1 bit to determine the result.

7.1 Instructions

Add

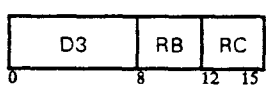
A RB,RC



The contents of registers RB and RC are added and the result placed into register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.

Absolute

ABS RB,RC

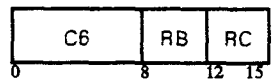


The content of register RB is replaced by the absolute value of the content of register RC. Condition bits LT, EQ, GT, OV and SO are set. Normally, only condition bit EQ or GT is set to one according to the value of the result. If register RC contains the maximum negative number, for which there is no equivalent positive number, then condition bits LT, OV and SO will be set to one.

Add Decimal

This instruction is not in the initial 801 implementation.

AD RB,RC



The two-digit decimal number in the low-order byte of the register specified by the second operand, augmented by the value of the condition register C0 bit, is added to the two-digit decimal number in the low-order byte of the register specified by the first operand. The result replaces the low-order byte of the register specified by the first operand, while the high-order bytes of this register remain unaltered.

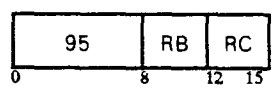
The condition register bit C0 is set to one if a carry results from the operation, otherwise it is set to zero. The condition register bit EQ is set to one if both digits of the result are zero, otherwise it is set to zero.

If any digit is an invalid decimal digit; i.e. hex'A, B, C, D, E, or F', the operation is suppressed, and the condition register decimal-exception-latch, DX, is set to one. Otherwise this latch is set to zero.

Condition register alterations:
DX, EQ, C0

Add Extended

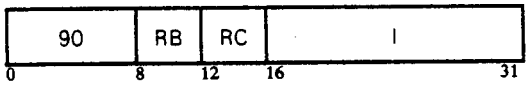
AE RB,RC



The content of register RB, the content of register RC, and the value of condition bit C1 are summed and the result placed into register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.

Add Extended Immediate

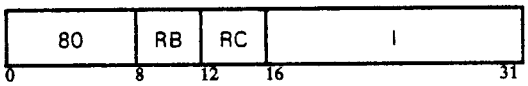
AEI RB,RC,I



The field I, extended on the left with eight zeros, the contents of register RC, and the value of condition bit C1 are summed and the result placed in register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.

Add Immediate

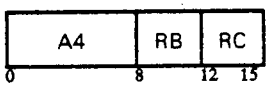
AI RB,RC,I



The field I, extended on the left with eight zeros, is added to the contents of register RC and the result placed in register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.

Compare

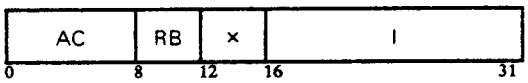
C RB,RC



The contents of registers RB and RC, both treated as 24 bit signed algebraic quantities, are compared. Condition bits LT, EQ, and GT are set according to how the value of register RB relates to that of register RC.

Compare Immediate

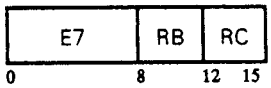
CI RB,I



The content of register RB is compared to field I, extended on the left with eight zeros. Condition bits LT, EQ, and GT are set according to how the value of register RB relates to that of field I.

Divide Step

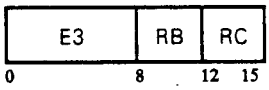
DIS RB,RC



The content of register RC is added to or subtracted from (register RB)||(bit 0 of MQ) depending on whether the signs of registers RB and RC disagree or agree. The 24 low order bits of the sum replace register RB. The MQ is shifted left one position and bit 15 of the MQ is set to 1 if and only if S, the sign of the result, equals the sign of register RC. Condition bits C0 and OV are set: C0=(RC(bit 0)=S) and OV=(RB(bit 0)=S).
Note: Condition bit SO is unaffected by this instruction.

Extend Sign

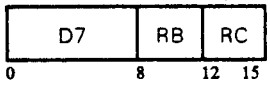
EXTS RB,RC



The content of the half of register RB is replaced by the half of register RC. Bits 0 - 7 of register RB are set to equal bit 8. Condition bits LT, EQ, and GT are set.

Multiply Step

MUS RB,RC



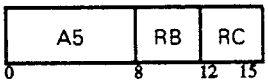
The incomplete product of register RC and bits 14-15 of the MQ are formed in (register RB)||MQ. A 26-bit sum is formed in accordance with the following table:

Conditional Bit C0	MQ Bit 14	MQ Bit 15	Algebraic Sum
0	0	0	(RB)+(RC)
0	0	1	(RB)+2(RC)
0	1	0	(RB)-(RC)
0	1	1	(RB)+0
1	0	0	(RB)+0
1	0	1	(RB)+(RC)
1	1	0	(RB)-2(RC)
1	1	1	(RB)-(RC)

The MQ is algebraically shifted right two positions, with the two low order bits of the sum replacing bits 0-1 of the MQ. Register RB is replaced with the 24 high order bits of the sum. Condition register bit C0 is set to ¬bit 14 of the MQ (before shift).

Subtract

S RB,RC

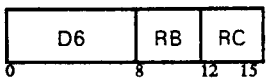


The content of register RC is subtracted from the content of register RB and the result placed into register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.
Note: if RB=RC, the content of this register is set to zero and bit C0 of the condition register is set to one.

Subtract Decimal

This instruction is not in the initial 801 implementation.

SD RB,RC



The two-digit decimal number in the low-order byte of the register specified by the second operand, augmented by the inverse of the condition register bit C0, is subtracted from the two-digit decimal number in the low-order byte of the register specified by the first operand. The result replaces the low-order byte of the register specified by the first operand, while the high-order bytes of this register are unaltered.

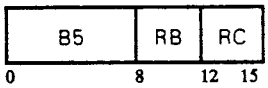
The condition register bit C0 is set to zero if a 'borrow' was required to perform the subtraction, otherwise it is set to one. The condition register equal-latch EQ is set to zero if both digits of the result are zero, otherwise it is set to one.

If any digit is an invalid decimal digit; i.e. hex'A, B, C, D, E, or F', the operation is suppressed, and the condition register decimal-exception-latch DX is set to one. Otherwise this latch is set to zero.

Condition register alterations:
DX, EQ, C0

Subtract Extended

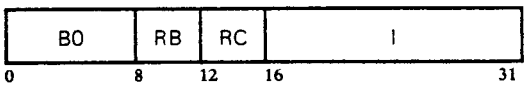
SE RB,RC



The complement of register RC is added to the content of register RB to which result is added the value of condition bit C1 and the result placed into register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.

Subtract Extended Immediate

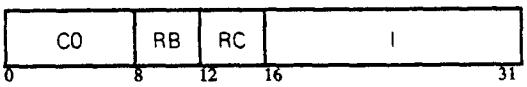
SEI RB,RC,I



The field I, extended on the left with eight zeros, is complemented and added to the content of register RC to which result is added the value of condition bit C1 and the result placed in register RB. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set.

Subtract from Immediate

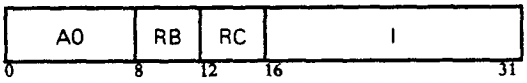
SFI RB,RC,I



The content of register RB is replaced the by content of the register specified by RC subtracted from I. For the subtraction, I is extended on the left with eight zeros. The condition bits LT, EQ, GT, C0, C1, OV, and SO are set based on the result.

Subtract Immediate

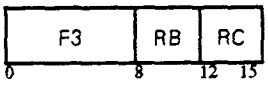
SI RB,RC,I



The content of register RB is replaced by I subtracted from the register specified by RC. For the subtraction, I was extended on the left with eight zeros. Condition bits LT, EQ, GT, C0, C1, OV, and SO are set based on the result.

Test Prefix for Overflow

TPO RB,RC



The content of register RB is replaced by the content of register RC. If any of bits 0 - 7 do not equal bit 8 (the sign bit) of register RC, then set OV and SO of the condition register to one; else set OV to zero.

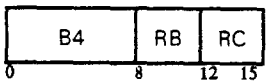
8. Logical Operations

The logical operations treat registers as 24 bit unsigned integers. The exception is the instruction Count Leading Zeros, CLZ, which is applied to the half of a register, i.e., the 16 low-order bits. The logical compares set the LT, EQ, and GT bits of the condition register according to the relative values of quantities treated as 24 bit unsigned integers. The other logical operations that set the LT, EQ, and GT bits do so according to the *algebraic* value expressed in two's complement representation. If the result is a negative value, LT is set to one; if it is zero, EQ is set to one; or if it is positive and not zero, GT is set to one.

8.1 Instructions

Compare Logical

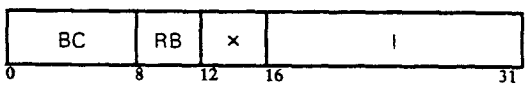
CL RB,RC



The content of register RB is compared with the content of register RC. Both comparands are treated as 24 bit unsigned quantities. Condition register bits LT, EQ, GT, C0 and C1 are set. LT, EQ, and GT are set according to how the value of register RB logically relates to that of register RC. C0 is set to one if the C0 characters in both comparands are equal, else it is set to zero. C1 is set to one if the C1 characters in both comparands are equal, else it is set to zero.

Compare Logical Immediate

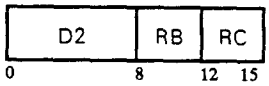
CLI RB,I



The content of register RB is compared with field I, extended to the left with 8 zeros. Both comparands are treated as 24 bit unsigned quantities. Condition register bits LT, EQ, GT, C0 and C1 are set. LT, EQ, and GT are set according to how the value of register RB logically relates to that of field I. C0 is set to one if the C0 character of register RB is equal to bits 16-23 of the instruction, else the C0 bit is set to zero. C1 is set to one if the C1 character of register RB is equal to bits 24-31 of the instruction, else it is set to zero.

Count Leading Zeros

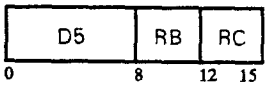
CLZ RB,RC



The content of the register specified by RB is replaced by the binary representation of the number of leading zeros in the half of the register specified by RC (i.e. The number of zeros to the left of the left-most one-bit of the half of register RC).

And

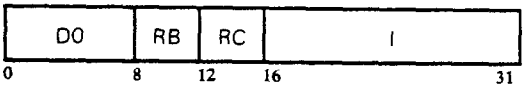
N RB,RC



The *AND* of the contents of the registers specified by RB and RC replace the content of the register specified by RB. Condition bits LT, EQ, and GT are set according to the result.

And Immediate

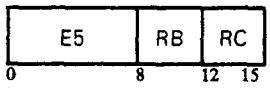
NI RB,RC,I



The *AND* of field I, extended to the left with 8 zeros, and of the content of register RC replaces the content of register RB. Condition register bits LT, EQ, and GT are set. The connective *AND* is applied bit by bit.

Or

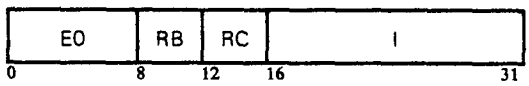
O RB,RC



The *OR* of the contents of the registers specified by RB and RC replace the content of the register specified by RB. Condition bits LT, EQ, and GT are set according to the result.

Or Immediate

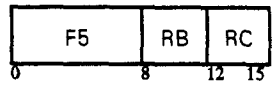
OI RB,RC,I



The *OR* of field I, extended to the left with 8 zeros, and of the content of register RC replaces the content of register RB. Condition register bits LT, EQ, and GT are set. The connective *OR* is applied bit by bit.

Exclusive Or

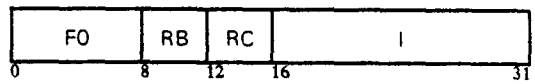
X RB,RC



The *EXCLUSIVE OR* of the contents of the registers specified by RB and RC replace the content of the register specified by RB. Condition bits LT, EQ, and GT are set according to the result.

Exclusive Or Immediate

XI RB,RC,I



The *EXCLUSIVE OR* of field I, extended to the left with 8 zeros, and of the content of register RC replaces the content of register RB. Condition register bits LT, EQ, and GT are set. The connective exclusive or is applied bit by bit.

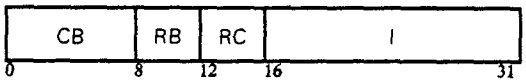
9. Shifts

Shift instructions operate either on 24 bits or on 16 bits (a half). The instructions shift a distance of from 0 to 15 bits either left or right. All shifts set the condition register to indicate if the algebraic value returned to the register is zero, positive, or negative. All except the algebraic right shift are logical in their treatment of the value shifted. On all left shifts, zeros are supplied to the vacated low order positions.

9.1 Instructions

And, then Shift Left Immediate

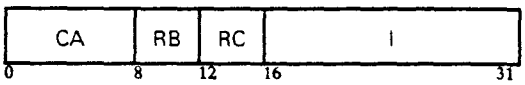
NSLI RB,RC,I



The *AND* of field I, extended to the left by eight zeros, and the contents of register RB, shifted left the number of bits specified by RC, replaces the content of register RB. Condition bits LT, EQ, and GT are set.

And, then Shift Left Paired Immediate

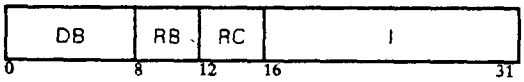
NSLPI RB,RC,I



The *AND* of field I, extended to the left by eight zeros, and the contents of register RB, shifted left the number of bits specified by RC, replaces the contents of the twin (in a pair) of register RB. Condition bits LT, EQ, and GT are set.

And, then Shift Right Immediate

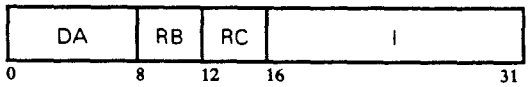
NSRI RB,RC,I



The *AND* of field I, extended to the left by eight zeros, and the contents of register RB, shifted right the number of bits specified by RC, replaces the contents of register RB. Condition bits LT, EQ, and GT are set.

And, then Shift Right Paired Immediate

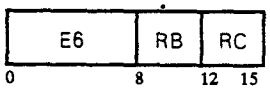
NSRPI RB,RC,I



The *AND* of field I, extended to the left by eight zeros, and the contents of register RB, shifted right the number of bits specified by RC, replaces the contents of the twin (in a pair) of register RB. Condition bits LT, EQ, and GT are set.

Shift Algebraic Right

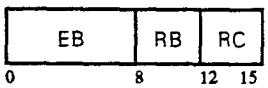
SAR RB,RC



The contents of register RB (bits 0-23) is shifted right the number of bit positions specified by digit D3 of register RC. Bits equal to the original sign bit (bit 0) are supplied to the vacated high order positions. Condition bits LT, EQ, and GT are set.

Shift Algebraic Right Immediate

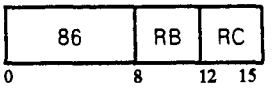
SARI RB,RC



The contents of register RB (bits 0-23) is shifted right the number of positions specified by RC. Bits equal to the original sign bit (bit 0) are supplied to the vacated high order positions. Condition bits LT, EQ, and GT are set.

Shift Half Left

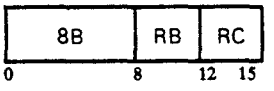
SHL RB,RC



If bit 19 of register RC is zero, then the content of the half of register RB (bits 8-23) is shifted left the number of bit positions specified by digit D3 of register RC, with zeros supplied to the vacated low order positions, and the prefix set to zero. If bit 19 of register RC is one, then register RB is set to zeros. Condition bits LT, EQ, and GT are set.

Shift Half Left Immediate

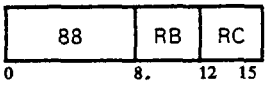
SHLI RB,RC



The content of the half of register RB (bits 8-23) is shifted left the number of bit positions specified by RC. Zeros are supplied to the vacated low order positions. The prefix of the result is set to zero. Condition bits LT, EQ, and GT are set.

Shift Half Left Paired

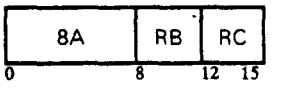
SHLP RB,RC



If bit 19 of register RC is zero, then the content of the half of register RB (bits 8-23), shifted left the number of bit positions specified by digit D3 of register RC, with zeros supplied to the vacated low order positions, and a prefix of zero, is stored in the twin, in a register pair, of RB. If bit 19 of register RC is one, then the twin of register RB is set to zeros. Condition bits LT, EQ, and GT are set.

Shift Half Left Paired Immediate

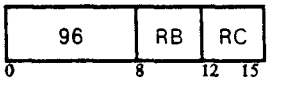
SHLPI RB,RC



The content of the half of register RB (bits 8-23), shifted left the number of bit positions specified by RC, with zeros supplied to the vacated low order positions, and a prefix of zero, is stored in the twin, in a register pair, of RB.Condition bits LT, EQ, and GT are set.

Shift Half Right

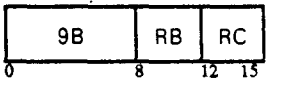
SHR RB,RC



If bit 19 of register RC is zero, then the content of the half of register RB (bits 8-23) is shifted right the number of bit positions specified by digit D3 of register RC, with zeros supplied to the vacated high order positions, and the prefix set to zero. If bit 19 of register RC is one, then register RB is set to zeros. Condition bits LT, EQ, and GT are set.

Shift Half Right Immediate

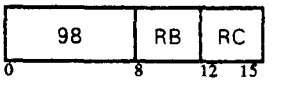
SHRI RB,RC



The content of the half of register RB (bits 8-23) is shifted right the number of bit positions specified by RC. Zeros are supplied to the vacated high order positions. The prefix is set to zero. Condition bits LT, EQ, and GT are set.

Shift Half Right Paired

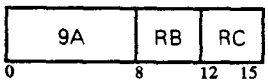
SHRP RB,RC



If bit 19 of register RC is zero, then the content of the half of register RB (bits 8-23), shifted right the number of bit positions specified by digit D3 of register RC, with zeros supplied to the vacated high order positions, and a prefix of zero, is stored in the twin, in a register pair, of RB. If bit 19 of register RC is one, then the twin of register RB is set to zeros. Condition bits LT, EQ, and GT are set.

Shift Half Right Paired Immediate

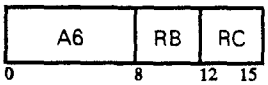
SHRPI RB,RC



The content of the half of register RB (bits 8-23), shifted right the number of bit positions specified by RC, with zeros supplied to the vacated high order positions, and a prefix of zero, is stored in the twin, in a register pair, of register RB. Condition bits LT, EQ, and GT are set.

Shift Left

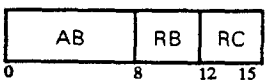
SL RB,RC



The content of register RB is shifted left the number of bit positions specified by digit D3 of register RC. Zeros are supplied to the vacated low order positions. Condition bits LT, EQ, and GT are set.

Shift Left Immediate

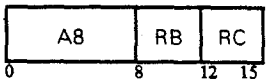
SLI RB,RC



The content of register RB is shifted left the number of bit positions specified by RC. Zeros are supplied to the vacated low order positions. Condition bits LT, EQ, and GT are set.

Shift Left Paired

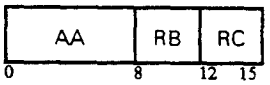
SLP RB,RC



The content of register RB, shifted left the number of bit positions specified by digit D3 of register RC and with zeros supplied to the vacated low order positions, is stored in the twin, in a register pair, of RB. Condition bits LT, EQ, and GT are set.

Shift Left Paired Immediate

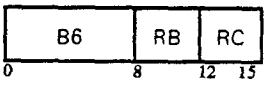
SLPI RB,RC



The content of register RB, shifted left the number of bit positions specified by RC and with zeros supplied to the vacated low order positions, is stored in the twin, in a register pair, of RB. Condition bits LT, EQ, and GT are set.

Shift Right

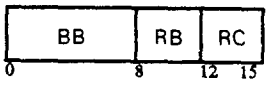
SR RB,RC



The content of register RB is shifted right the number of bit positions specified by digit D3 of register RC. Zeros are supplied to the vacated high order positions. Condition bits LT, EQ, and GT are set.

Shift Right Immediate

SRI RB,RC

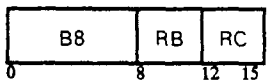


The content of register RB is shifted right the number of bit positions specified by RC. Zeros

are supplied to the vacated high order positions. Condition bits LT, EQ, and GT are set.

Shift Right Paired

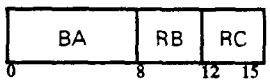
SRP RB,RC



The content of register RB, shifted right the number of bit positions specified by digit D3 of register RC and with zeros supplied to the vacated high order positions, is stored in the twin, in a register pair, of RB. Condition bits LT, EQ, and GT are set.

Shift Right Paired Immediate

SRPI RB,RC



The content of register RB, shifted right the number of bit positions specified by RC and with zeros supplied to the vacated high order positions, is stored in the twin, in a register pair, of RB. Condition bits LT, EQ, and GT are set.

10. System Control

10.1 Cache Control Operations

The 801 processor is organized to allow independent memory access for data and instructions. Each access path may be served by an independent cache. The effects of these caches on program execution (other than to improve performance) occurs only in special circumstances. Particularly, modifications to main memory by I/O paths must not be assumed to be reflected to the processor, since the areas affected may already be copied in either or both caches. Modifications to memory by the processor may not be reflected in subsequent instruction or I/O access to main memory, since the updates may be buffered in the data access cache for an indeterminate period of time. This buffering can affect both read and write I/O accesses. Reads can be changed if a buffered modification to the target area of the read is accomplished after the read has completed. Writes will transmit the wrong values if buffered modifications have not been accomplished. (It should be stressed that the 801 architecture allows for indeterminately long delays between store instructions and the actual modification of main memory.)

In systems which are as fast as the current 801 implementation, cache misses are a very important source of performance degradation. To minimize this degradation, instructions are provided which allow programs to give usage information to the cache management hardware. This will enable the system to avoid unnecessary line transfers between cache and main storage.

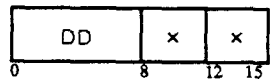
The cache control instructions are provided to allow program control of the relationship between main memory and the caches. These instructions deal with cache lines which are implementation defined. In the current implementation, lines of both data and instructions are 32 bytes long on 32 byte boundaries. The data cache does not attempt to update main memory until a line which has been changed must be removed to make room for a new line.

It is likely that details of the cache will be model dependent, and may even be changed in the prototype. Thus all cache control algorithms should be designed and packaged in a way which makes response to such changes reasonably easy. At the very least, the line sizes of the instruction and data caches should be reflected as *independent* symbolic constants in each routine which issues cache control operations or aligns data on cache line boundaries. (It is probably safe to assume that each line size is a power of two and that the lines are aligned in memory with respect to their own size.)

10.2 Instructions

Disable

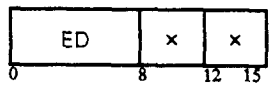
DI



The cpu is disabled for any interruptions due to an external interruption condition. The previous enable state bit (EN) of the condition register is set to one if the processor was enabled previous to this instruction, else the processor was disabled and the bit is set to zero.

Enable

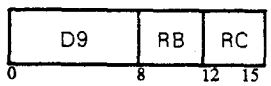
EI



The cpu is enabled for interruptions due to an external interruption condition. A pending external interrupt will cause an immediate processor interrupt after ENABLE. The previous enable state bit (EN) of the condition register is set to one if the processor was enabled previous to this instruction, else the processor was disabled and the bit is set to zero. (Engineering note: Care must be taken that an enable followed immediately by a DISABLE, IOR, or IOW instruction works correctly.)

Instruct Data Cache Line

INDCL RB,RC



Register RC contains the address of a byte in storage which resides in some line. This line is

called the subject line. RB (bits 8-11 of the instruction) instructs the cache as follows:

RB=0 Invalidate Data Cache Line
If a previously fetched copy of the line containing the byte addressed by the contents of register RC exists in the data cache, that copy is (logically) replaced by the current value of that line in main storage (no actual fetch from main storage is implied). This instruction can be used before the transmission into main storage of data from a direct memory attachment, to guarantee that the values received will be the ones seen by subsequent accesses and that pending updates are cancelled. This instruction can also be used to suppress storing dead temporary values back into main storage from the data cache.

RB=1 Store Data Cache Line
The data cache is searched to see if a pending update to the subject line exists. If a pending update is found, it is performed. The line may be retained in the cache for later use.

RB=2 Load Data Cache Line
If the subject line is not in the cache, it is loaded in the cache, just as if a data reference to the subject line had occurred.

RB=3 Set Data Cache Line
If the subject line is not in cache, a cache entry is established for it but the line is not loaded into the cache. If the subject line is in the cache, it is flagged as being unchanged. This can be used to suppress loading dead values into the data cache from main storage when new values are to be created.

The following two cache instructions are not in the initial 801 implementation. Other cache instructions, also not in the initial 801 implementation, are defined in Section 11.

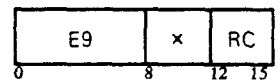
RB=4 Move to Buffer
The subject line is moved into a data cache line buffer. This instruction does not disturb any data cache entry.

RB=5 Move from Buffer
The cache line buffer is moved into the subject line. If the subject line was in cache, it is written over. If the subject line was not in cache, the line is written over in backing store -- the cache is left undisturbed.

Note. Since the buffer may be any cache line temporarily used for this purpose, MOVE TO/FROM BUFFER sequences must run disabled to ensure that the buffer contents are not modified between the sequences.

Invalidate Instruction Cache Line

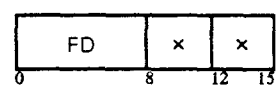
INICL RC



If the instruction cache has a copy of the line containing the word addressed by the contents of register RC, that copy is abandoned. Thus previous updates to that line which are reflected in main storage (see the Store Data Cache Line option of INDCL) will be reflected in subsequent execution. If the line indicated by this instruction is the line currently addressed by the Instruction Address Register or the line following, or if either of these lines and the lines immediately following contain consecutive ZNOP instructions, the results are unreliable. This instruction can be used in conjunction with the INDCL(RB=1) instruction to synchronize the creation or modification of program text with the subsequent execution of this new program text.

Zero Time No-op

ZNOP



No operation is performed and the next following instruction is executed, usually during the CPU cycle that would normally be taken for the execution of this instruction; hence the appearance of zero-time execution.
Note. This instruction can be used to force the IAR to a full word boundary.

11. Relocate

11.1 Relocate Facilities

The Relocate feature of the 801 is not implemented in the first prototype being developed.

The 801 without Relocate provides one real address space up to 16 megabytes. The actual size of a particular system is known to the CPU, which presents a program interrupt if an access above this limit is requested.

In addition to this real address space, the 801 Relocate feature provides up to 255 virtual address spaces, each up to 16 megabytes.

The following are some important differences between 801 Relocate and S/370 Relocate:

- An address, as presented to the storage subsystem, consists of 32 bits (8 bit Address Space ID (ASID) and 24 bit address in this address space). An ASID of 0 indicates real address space. The 24 bit address is developed normally. The 8-bit ASID is taken from one of two new registers, the Instruction Relocate Register (IRR) and the Data Relocate Register (DRR). (These may be physically in their respective caches.) The format of the Relocate Register is shown in Figure 11.1. The machine assumes that each such 32 bit address refers to a unique byte in real memory (or that the page is not in real memory). Thus shared pages are not supported by the machine in a manner guaranteed to be correct.

- Whereas the equivalent ASID in S/370 is the address of a segment table, the 801 ASID is symbolic and independent of the address of the relocation table. Thus page tables can be moved freely without invalidating cache directory entries. The format of a page table entry is shown in Figure 11.2.

- The relocation tables are one level (i.e. there are no segment tables in this architecture).

ASID: Address space
L: Length of address space (2, 4, 8, or 16 meg)
P.T. ADDRESS: Address of Page Table (on 1K boundary)

Figure 11.1 **Relocate Register Format**

Bit:

0	Reserved
1	Software Use (can be used by software to mark read-only pages - not enforced by hardware)
2	Data Reference Flag (set to one when this PTE is used by the data cache)
3	Valid Flag (1 causes a Page Fault Interrupt regardless of the value of the Real Page Address)
4-15	Real Page Address (4K pages)

Figure 11.2 **Page Table Entry Format**

- Because the 801 has a split instruction cache and data cache it is natural (and useful) to support two possibly different active address spaces, one for data and one for instructions.

(*Engineering note* : Because of the above three characteristics it is possible to run both caches "virtual", that is, to search the cache directory with the unique virtual address rather than with a real address. Thus, on cache hit, there is no performance degradation.

It is also possible now to resolve cache misses independently by accessing page tables which themselves are participating in the instruction/data cache replacement strategies.)

11.2 Interrupts

The set of interrupts already defined for the 801 remain unchanged, except for the following:

1. A new interrupt type is defined, called Page Fault. This is raised whenever one of the following occurs:

- A real address (ASID=0) exceeds the range of this machine.
- A virtual address is resolved, via a page table entry, to an address which exceeds the range of the machine.
- A virtual address is resolved to a page table entry which has an invalid flag set.
- A virtual address exceeds the size of its address space.

(Bits 4-7 of the ISB will distinguish these cases).

The Old IA for Page Fault interrupt contains the instruction whose attempted execution caused the interrupt.

(*Engineering note* : Because Page Fault interrupts are defined to be synchronous they must not be raised early (due to prefetch-buffer activity) or late (due to overlapped processing with data cache accesses).)

2. The Program interrupt for out-of-range real addresses will never be raised.

3. The execution of *all* interrupt types consists additionally in setting the Instruction and Data ASID's to zero (i.e. the first level interrupt handlers all run real).

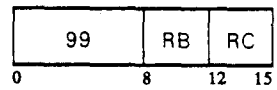
4. The reserved word following each OLD IA word in the interrupt areas is used to store the old Instruction and Data ASID's.

11.3 New or Modified Instructions

Branch, Execute and Change State

Replaces Branch, Execute and Enable

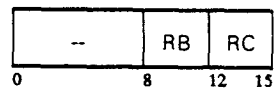
BCSX RB,RC



The instruction immediately following the BCSX instruction, called the subject instruction, is executed. Then the contents of the register specified by RB replaces the Instruction Relocate Register. Finally the machine is enabled, and the branch target is fetched (which may cause a page fault).

Branch Execute and Set ASID

BSAX RB,RC

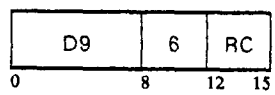


Same as above except the machine is not enabled.

Change Data Relocate Register

Option of Instruct Data Cache Line Instruction

INDCL 6,RC



The contents of the register specified by RC replace the Data Relocate Register.

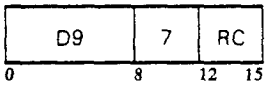
(*Engineering note* : To insure that store-back does not also require table lookup, the data

cache directory should contain the real line address in addition to the virtual address.)

Flush Data Cache Line

Option of Instruct Data Cache Line Instruction

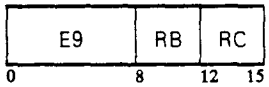
INDCL 7,RC



Same as Store DCL except that the line is not retained in the cache. (Used when a PTE is modified.)

Invalidate Instruction Cache Line

INICL RB,RC



RB contains the ASID of the line to be invalidated. Otherwise the instruction remains unchanged.

12. 801 I/O Subsystem

12.1 Introduction

Overall 801 I/O structure and the functional characteristics of certain specific I/O adapters are defined in this section. All of the I/O components described will not necessarily be in the initial 801 implementation. *Specifically, the Switch and Switch Adapter (SA) are not in the initial 801 implementation.*

12.2 I/O Structure Overview

Figure 12.1 shows the general structure of the 801 I/O subsystem. The I/O components in the figure are:

- I/O Bus is a parallel bus which moves control and data to and from the 801 under control of 801 instructions. I/O performed under control of 801 instructions is called programmed I/O. A detailed description of the bus is available in the document "801 System I/O Interface Specification" number 4322-11.

- BSA: Bus Serial Adapter is an adapter which is primarily used to transfer short blocks of data and control information to and from a control unit over a serial link.

- EIA External Interrupt Adapter collects interrupt request from all adapters and control units and presents a single external interrupt signal to the 801 when appropriate.

- Serial Link attaches a Control Unit to a BSA or DMA. The link is compatible with synchronous data link control (SDLC).

- CU A Control Unit provides the logical capabilities necessary to operate and control one or more I/O devices. A CU may be connected by a serial link to a BSA or DMA or connected locally to a Functional Adapter (FA).

- FA Functional Adapter is any adapter designed to attach a particular kind of Control Unit to the I/O Bus.

- MIO Bus The Memory I/O Bus provides access to the main memory of the 801. It is described in the document "801 Memory I/O Bus Specification" number 4322-14.

- DMA The Direct Memory Access adapter is connected to both the IO Bus and the MIO Bus. It is used for transferring data between a CU and main memory over a serial link. Once initiated the data transfer proceeds without CPU or I/O Bus involvement.

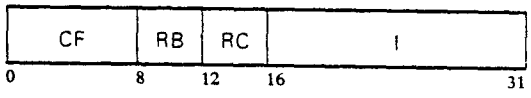
- Switch is a full duplex switch for serial links. An integral part of the switch is the Switch Adapter (SA) through which the switch is controlled.

12.3 I/O Instructions

Programmed I/O (PIO) instructions are used to transfer data between the CPU general purpose registers and I/O Bus adapters. The instructions are I/O Read (IOR) and I/O Write (IOW) and they have the D instruction format:

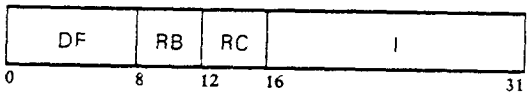
Input/Output Read

IOR RB,RC,I



Input/Output Write

IOW RB,RC,I



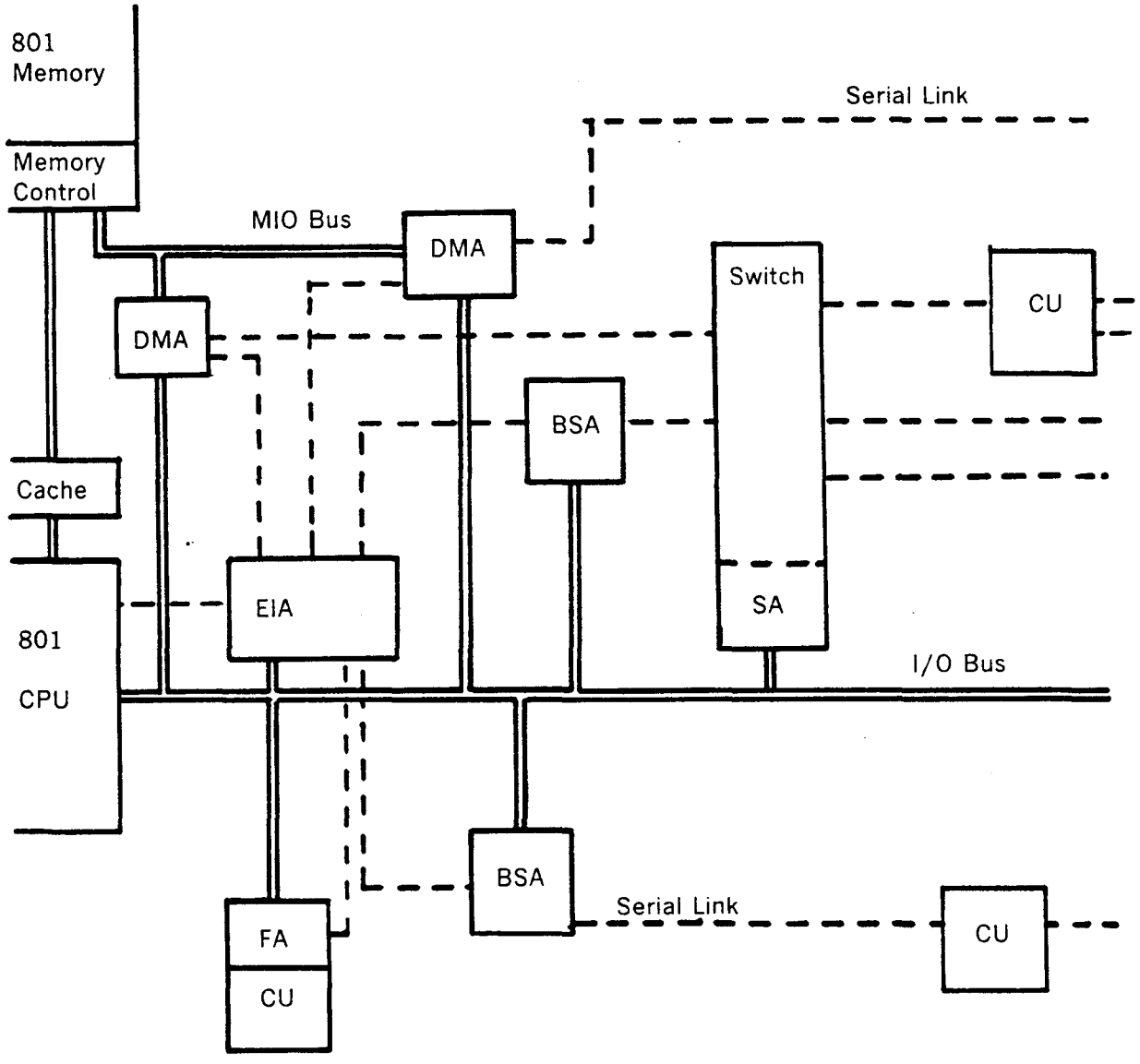
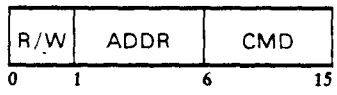


Figure 12.1. **I/O Subsystem Structure**

In the execution of both instructions, the CPU puts on the I/O Bus a 16 bit address/command field consisting of one bit indicating IOR (0) or IOW (1) followed by the lower order 15 bits of the sum of I and 0/(RC). The address/command field has the following format



The first bit distinguishes a read from a write, and ADDR is a 5 bit field used to select one adapter on the I/O Bus. The 10 bit command field CMD contains information for the selected adapter. In general the use of CMD is adapter independent.

At the time an instruction is initiated by the CPU, the selected adapter may be in a "busy" state relative to the specific instruction. If it is busy, execution of the instruction only results in setting the I/O busy bit in the condition register to 1. Otherwise, for IOW, the I/O busy bit is set to zero and the half of the source register RB are transferred to the adapter, or, for IOR, two bytes from the adapter are placed in the half of the target register RB and its prefix set to zero.

The above description of the operation of I/O instructions assumed no errors occurred during instruction execution. If an error is detected, an I/O check interrupt occurs at the completion of the instruction and the contents of RB, RC, and the value of the I/O busy bit in the condition register are unchanged.

Note: The conditions under which an instruction sets the I/O busy bit to 1 depends on the CMD field, the specific adapter, and the state of the adapter when the instruction is initiated. Adapters may be designed so that they are never busy for some or all I/O instructions. In these cases, it is not necessary for the program to test the I/O busy bit for the successful execution of the data transfer part of the I/O instruction, since it will always be set to zero.

12.4 Serial Link

Detailed information on the Serial Link is available in document 4322-12, and in "IBM Synchronous Data Link Control" GA27-3093.

For the purposes of this manual, the following properties of the Serial Link should be understood by the reader.

(1) The link is full duplex.
(2) All transmissions are delimited by SDLC flags:

    F I (CRC) F

    where F is the SDLC flag byte (01111110), I is the information in an integral number of data bytes modified by the SDLC zero insertion rule, and CRC is the standard SDLC two byte cyclic redundancy check.
(3) SDLC idle and abort are implemented by the receiver and transmitter.
(4) The rate of transmission is set by the CU data transmission clock. Control Units which are locally attached over serial links to the 801 may further control the rate by introducing a pause in the transmissions in both directions on the link by stopping the data clock for up to 100 milliseconds.

12.5 Control Units

Only Control Units which are attached to the system via serial links are covered here. A Control Unit provides the logical capabilities necessary to operate and control an I/O device. A CU accepts and executes commands sent to it and provides by means of interrupts and messages sent over the link indications concerning CU and device status. The I/O device attached to the CU participates in the execution of most types of commands. For such commands, the CU provides the detailed signaling sequences required to perform the operation.

CU commands are conveyed to a CU by a BSA or a DMA via a serial link and they have a variable length format.

The first byte sent defines the command and subsequent bytes are interpreted as data or command modifiers as appropriate to the command. All of the bytes are transmitted in a single SDLC frame.

With respect to the data transmitted by the CU, commands fall into two categories:

(1) Read type commands: These commands are requests for two or more bytes of data from the CU. If the CU is unable to provide the data, it sends a one byte CU summary status message detailing why it cannot execute the command. Should the CU detect an error condition after having initiated the transfer of the requested data, it aborts the transmission. Otherwise, the receiver of the data assumes successful execution of the command by the CU.

(2) Write type commands: One or more bytes of data are transmitted to the CU and the CU responds with its summary status byte. The CU normally sends the status byte after it receives the complete message. However, if the CU detects the end of data condition during a write or if it is in a state where it cannot execute the command, it may send the status byte before the complete message is sent.

12.5.1 Control Unit Signals

Once initiated, I/O operations are performed asynchronously relative to CPU execution. To facilitate synchronization of I/O and CPU activity certain conditions which mark the CU's and device's progress in executing a command are signaled by the CU. These signals are made available to the program either via the BSA or DMA or by CU interrupts and are as follows:

(1) Data end: the data end signal indicates that the CU has received or provided all of the data associated with the command. Data end is always signaled to the BSA or DMA, either explicitly as a bit in the summary status byte message or implicitly by sending the data requested as part of a read type command.

(2) CU end. Certain commands may keep the CU busy after the data end condition is reached. The CU end signal indicates that the CU has completed its part in executing the command. (Commands which involve an I/O device may still be in progress). CU end may be signaled concurrently with data end or it may be signaled afterwards by a CU interrupt. The CU end signal may not be generated as a signal distinct from device end by some types of Control Units.

(3) Device end. Conclusion of all commands is indicated by the device end signal. Device end may be signaled concurrently with data end or it may be signaled afterwards by a CU interrupt.

Note: For any particular CU and command data end, CU end, and device end are signaled in only one way and order. Control units which serve only one device do not distinguish between CU end and device end. Furthermore, CU end may not be distinguished from device end by some CU's even though they serve several devices. The CU generates interrupts to signal CU end and device end when these conditions occur after data end was signaled. Since an interrupt may have multiple causes, the program must sense the "status" of the CU to determine the nature of the interrupt.

12.5.2 CU States and Commands

In general, the result of initiating a CU command depends on the state of the CU when the command is initiated. Excluding the non-operational state, there are three CU states (Available, Working, and Interrupt Pending) and, exclusive of modifiers, five CU commands (Read, Write, Sense, Halt, and Control).

Available. The available state indicates that the CU is available to execute any of the five commands.

Working. The working state indicates that the CU is in the process of executing a previously initiated command. A CU leaves the working state after signaling CU-end or device-end.

Interrupt Pending. When the CU has an interrupt condition to report which has not been masked in the CU it enters the interrupt pending state. The CU leaves the interrupt pending state as a result of executing a sense command addressed to the interrupt condition register of the CU.

Note: The working state can only be entered as a result of initiating a CU command. The CU is in the available state from the time CU-end or device end occurs until either an interrupt condition which is relevant to the program is recognized by the CU or the next command is initiated. The sense command addressed to the interrupt register of the CU clears the interrupt condition.

The CU command code assignment is listed in Figure 12.2. The symbol x indicates that the bit position is ignored; m identifies a modifier bit. The meaning of the modifier bits depends on the type of control unit.

Command	Code
HALT	001 xxxxx
SENSE	000 mmmmm
READ	01 mmmmmm
WRITE	10 mmmmmm
CONTROL	11 mmmmmm

Figure 12.2. **CU Command Codes**

Read. A read operation is initiated between the CU and the BSA or DMA. The Read command and any modifiers are transmitted to the CU. If the CU is in the available state when the command is initiated, the requested bytes (at least two) are transmitted by the CU. If the CU is in the Working or Interrupt Pending State, the CU sends its status byte to the BSA or DMA. Data end is implicit with the transmission of the last byte in both cases. CU end and device end may be concurrent with or subsequent to data-end.

Write. A write operation between the CU and a BSA is initiated. The write command, any modifiers, and the data to be written are transmitted to the CU. If in the Available state, the CU responds with its status byte after receiving the complete message. If in the working or interrupt pending state, the CU may respond with its status byte immediately.

Sense. A sense operation is initiated at the CU. The sense command and any modifiers are transmitted to the CU by the BSA and the CU responds with two or more bytes of sense information, provided the CU is not in the working state when the command is initiated. When the sense command is addressed to an interrupt condition register in the CU, the register is also set to zero. If the CU is in the working state when the command is initiated, it responds with its status byte. Device end is implicitly signaled with the transmission of the last byte of sense information.

Halt. The Halt command stops the transfer of data from the CU on a read type command. The CU executes the command by sending any last data byte and the concluding CRC and flag bytes. If the transmission had not yet begun when the command is received by the CU, the CU sends the null message consisting of two flag bytes separated by the CRC bytes, and if the transmission had been completed, the command is equivalent to a NO-OP.

Control. A control operation is initiated at the CU. The control command byte with modifiers and data are sent to the CU. A control command is used to initiate an operation which does not involve transfer of data such as positioning a disk-access mechanism or preparing the CU for transfer of data with a DMA. The interpretation of the command, modifiers, and data is CU dependent. In particular, the states in which a CU accepts and executes a control command is CU dependent.

12.5.3 CU Summary Status Byte

The CU transmits its summary status byte under the following circumstances

(1) When it detects the end of a write operation.

(2) In response to a command it cannot execute because it is in the working or interrupt pending state.
(3) When it detects a transmission error in a received message.
(4) When it receives an illegal command.

The significance of the bits in the summary status byte is shown in Figure 12.3.

Bit	Designation
0	Unit Check
1	Transmission Error
2	Incorrect length
3	Working State
4	Interrupt Pending State
5	Device End
6	Control Unit End
7	Data End

Figure 12.3. **Summary Status Bits**

Unit Check Unit check indicates that the I/O device or control unit has detected an unusual condition. Detailed information is obtained by issuing a sense command. Unit check may indicate either a programming error or a hardware malfunction.

Transmission Error. The CU receiver has detected a transmission error.

Incorrect Length. This bit is set to 1 on a write operation when the CU detects the data end condition before it receives the second flag.

The significance of bits 3 to 7 is described above.

12.6 BSA

The Bus to Serial Adapter (BSA) is primarily designed to simultaneously transmit and receive from one to 16 bytes of data over the serial link. Subject to possible limitations on data rate, longer blocks of data can be handled by appropriate programming. All BSA I/O operations are initiated and controlled by PIO instructions.

12.6.1 Transmission

Data to be transmitted are loaded into the BSA transmission buffer by PIO, the flag and CRC bytes and zero insertion required by the link protocol being supplied by the BSA hardware automatically. The buffer is a first in/first out queue (FIFO) with room for 16 bytes. The program controls the loading of the queue and the time at which the transmitter starts. An over-run check condition occurs if the transmitter attempts to take a byte from an empty queue. Maskable interrupts are generated and posted in the status register when a complete frame has been sent and when the queue contains eight or fewer bytes and the transmitter is on.

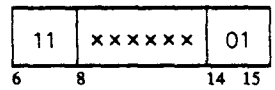
Load Two Bytes (LB2)

LB2 is an IOW with command field:

First char. C0 and then char. C1 of register RB are put on the transmission queue. If the queue does not have room for the bytes or if the last byte flag had been set by a previous instruction (*LLB2*, *LLB1*) the I/O busy bit in the condition register is set to 1.

Load Last Two Bytes (LLB2)

LLB2 is an IOW with command field:



LLB2 is identical to *LB2* in the way data are placed on the queue. In addition, the instruction starts the transmitter if it was not already started and it sets the last byte flag to 1. The transmitter resets this flag after it transmits the last byte put on the queue by this instruction and the concluding CRC and flag bytes. If the queue does not have room for the bytes or if the last byte flag is already set to 1 when the instruction is initiated, the I/O busy bit in the condition register is set to 1.

Load Last One Byte (LLB1)

LLB1 is an IOW with a command field:

Char. C0 of register RB is put on the transmission queue and the last byte flag is set to 1. The instruction starts the transmitter if it was not already started. If the queue does not have room for the byte or if the last byte flag is already set to 1 when the instruction is initiated, the I/O busy bit is set to 1. The last byte flag is reset by the transmitter after it transmits the byte stored by this instruction and the concluding CRC and flag bytes.

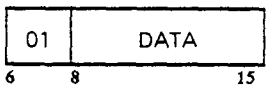
Load Two Bytes and Start Transmitter (LB2ST)

LB2ST is an IOW with a command field:

LB2ST loads the transmission queue in the same way as *LB2* and in addition starts the transmitter. If the transmitter is transmitting when the command is initiated, the I/O busy bit is set to 1.

Transmit Immediate (TRI)

TRI is an IOW with a command field:



The data portion of the command field, bits 8 to 15, is transmitted as a one byte message. The I/O busy bit is set to 1 if either the BSA is transmitting or the transmission queue is not empty when the command is initiated.

12.6.2 Receive

The BSA receiver puts the information bytes contained in each SDLC frame on a FIFO receive queue. The queue has room for 16 bytes and the program accesses the queue with IOR instructions. An overrun check condition results if either the transmitter tries to put a byte on the queue when it is full or if the queue is not empty when the first data byte of an SDLC frame is detected. The BSA has a counter accessible to the program which is equal to the number of bytes in the queue. Maskable interrupts are generated and posted in the status register when a complete frame is received and when the queue contains eight or more bytes.

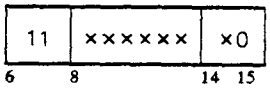
Read One Byte (RD1)

RD1 is an IOR with a command field:

RD1 transfers the first byte in the receive queue into char. C0 of the target register of the IOR. Char. C1 is set to zeros. The I/O busy bit is set to 1 if the queue is empty when the instruction is initiated.

Read Two Bytes (RD2)

RD2 is an IOR with a command field of



RD2 transfers the first byte in the receive queue into char C0 of the target register in the IOR and the next byte into char C1. The I/O busy bit is set to 1 if the queue contains less than two bytes when the command is initiated.

12.6.3 Status Register

The status register and the status register extension provide to the program the status of the BSA and the conditions under which an I/O operation was concluded. Those bits in the status register which are read-only (R/O)

are unaffected by IOW instructions addressed to the register. Some of the bits may cause the interrupt request line to be activated and these bits may be either maskable or non-maskable. The significance of the bits in the status register is shown in Figure 12.4.

Bit	Designation	Type	Interrupt	Mask
0	Interrupt Summary	R/O	Yes	No
1	Link Idle	R/O	No	-
2	*not used*	-	-	-
3	*not used*	-	-	-
4	Receive queue half-full	R/O	Yes	Yes
5	Transmit queue half-full	R/O	Yes	Yes
6	Frame received	R/W	Yes	Yes
7	Frame transmitted	R/W	Yes	Yes
8-11	Interrupt Mask	R/W	No	-
12-15	Received byte counter	R/O	No	-

Figure 12.4. **BSA Status Register**

Interrupt Summary. One or more bits in the status register extension which can cause an interrupt is set.

Link Idle. The idle condition on the link has been detected by the receiver. The bit is reset to zero when the beginning flag of the next frame is received.

Receive Queue half-full This bit is equal to one whenever there are eight or more bytes in the queue.

Transmit Queue half-full This bit is equal to one whenever there are eight or fewer bytes in the transmit queue and the transmitter is transmitting.

Frame Received. The bit is set to zero when the first flag of a frame is received and is set to 1 when the final flag is received.

Frame Transmitted. The bit is set to zero by any of the load buffer commands that start the transmitter and is set to 1 when the final flag of the frame is sent.

Interrupt Mask. The four bits selectively mask the corresponding bits in bit positions 4-7 of the status register: 1 allows the interrupt, 0 masks the interrupt.

Received Byte Counter. The four bit counter is equal to the number of bytes in the receive queue. It is incremented as each byte is received and put on the queue and decremented as bytes are removed by IOR' s.

12.6.4 Status Register Extension

The bits in the status register extension have the meaning shown in Figure 12.5.

Bit	Designation	Type	Interrupt	Mask
0	Bus Parity Error	R/W	No	-
1	Invalid Instruction	R/W	No	-
2	Abort received	R/W	Yes	No
3	Transmission Error	R/W	Yes	No
4	Receiver Overrun	R/W	Yes	No
5	Transmission Overrun	R/W	Yes	No
6	Clock Failure	R/W	Yes	No
7	Programmed Controlled Interrupt	R/W	Yes	No
8	Diagnostic Mode	R/O	No	-
9-15	*not used*	-	-	-

Figure 12.5. **BSA Status Register Extension**

Bus Parity Error. An I/O Bus parity error occurred during the execution of the current PIO instruction. An I/O check interrupt is taken by the CPU.

Invalid Instruction. The current PIO instruction is invalid. An I/O check interrupt is taken by the CPU.

Abort Received. The SDLC abort signal is detected by the receiver. The receiver is reset to the idle state.

Transmission Overrun The transmitter needs a data byte to send but the transmission queue is empty. The transmission is aborted and the transmitter reset to the idle state.

Receiver Overrun. The receiver has a data byte to put on the receive queue but the queue is full or else the receiver has received the first information byte of a frame and the queue is

nonempty. The receiver is reset to the idle state.

Transmission Error. A CRC check occurred or the number of bits received was not an integral number of bytes. The receiver is reset to the idle state.

Clock Failure The clock has not been received for more than a fixed time interval (100 milliseconds in prototype). Both transmitter and receiver are reset to their idle states.

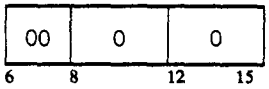
Programmed Controlled Interrupt. The program may use this bit to generate an interrupt request from the BSA.

Diagnostic Mode. This bit is set to 1 when the BSA is in diagnostic mode.

The I/O busy bit in the condition register is always set to zero by control commands.

Read Status Register

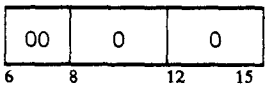
An IOR with command field:



transfers the status register into the target register of the IOR.

Write Status Register

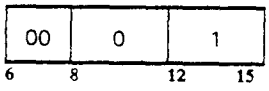
An IOW with a command field



sets the R/W bits of the status register to the value of the corresponding bits in the source register of the IOW. R/O bits are not affected.

Read Status Register Extension

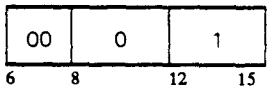
An IOR with command field:



transfers the status register extension to the target register of the IOR.

Write Status Register Extension

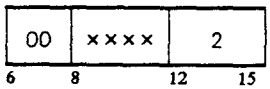
An IOW with command field:



sets the bits in the status register extension to the value of those in the source register of the IOW.

Reset Command

An IOW with command field:



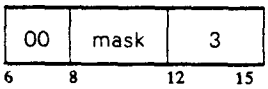
causes the BSA to perform its reset function. The I/O busy bit is always set to zero.

Reset Function. Execution of the reset function leaves the BSA in the following state:

(1) Status register and its extension all zeroes.
(2) Transmit and Receive queues empty.
(3) Transmitter in idle state, transmitting 1's.
(4) Receiver in idle state, looking for the first flag of a frame.

Write Mask

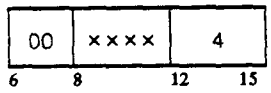
Write Mask is an IOW with a command field of



Write Mask sets the interrupt mask in the status register equal to bits 8-11 of the command field.

Reset Frame Received Bit

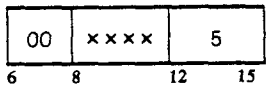
An IOW with a command field



resets the frame received bit in the status register to zero.

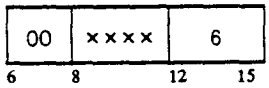
Reset Frame Transmitted Bit

An IOW with command field:



resets the frame transmitted bit in the status register to zero.

Reset Receiver - An IOW with a command field of



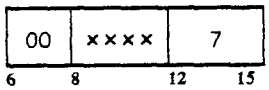
resets the receiver to the idle state and sets the received byte counter to zero.

Diagnostic Mode

In diagnostic mode, the BSA generates its own data clock signal and connects its transmitter serial output to its receiver input.

Enter Diagnostic Mode

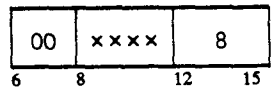
An IOW with a command field of



puts the BSA into diagnostic mode.

Leave Diagnostic Mode

An IOW with a command field of



takes the BSA out of diagnostic mode.

12.7 DMA

The Direct Memory Access adapter (DMA) is designed to transfer a block of data between main memory and a Control Unit over a serial link. Before the data transfer begins, the DMA and CU must be appropriately initialized with PIO and CU commands, respectively. A start instruction is then issued to the DMA and the data transfer proceeds to completion without CPU action. The data end condition may be recognized by either the CU or the DMA but the completion of the data transfer is always signaled to the CPU by a DMA interrupt. The CU may also signal a subsequent device end by means of an interrupt.

The DMA includes the BSA capabilities as a subset of its capabilities and except for invalid commands, the DMA and BSA are equivalent so far as programs designed for BSA operations are concerned. However, the DMA has only one transmitter and receiver.Therefore, while a DMA read or write is executing, PIO instructions which normally load the transmit queue or read the receive queue will set the I/O busy bit in the condition register to one.

12.7.1 Functional Description

In what follows, only DMA functions beyond those in a BSA are described.

Address Register

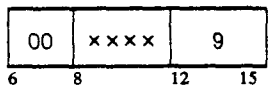
The 24 bit address register specifies the location of the first byte of data in main memory

involved in the data transfer. The low order 5 bits of the address must be zero (viz. I/O transfers must start on a cache line boundary). Subsequent bytes are used in ascending order from the original address. On read operations, the number of bytes stored in main memory is always an integral number of cache lines. When the number of bytes transferred between the CU and the DMA on a read is not divisible by 32, the remainder of the last line is padded out with zeros.

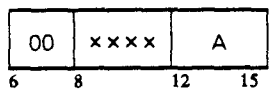
The address register is incremented by 32 each time a cache line is transferred between the DMA and main memory.

The address register may be read or written by the program.

A command field of

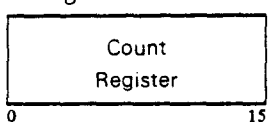


addresses the low order two bytes of the register and a command field of



addresses the high order byte of the register. The high order byte is set equal to the value of the low order byte of the source register of the IOW. An IOR addresses to the high order byte moves the byte to the low order byte of the target register of the IOR. The I/O busy bit is set to one if a previously initiated DMA operation had not yet completed when these instructions are initiated.

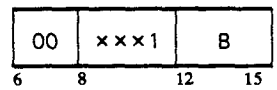
Count Register



The 16 bit count register is initialized at the beginning of a DMA operation with the number of bytes to be transferred. As each byte is transmitted to or received from the Control Unit, the count is decremented by one (but not below zero). Excluding hardware errors and the bytes that may be needed to pad out the last cache line on a read, the difference between the initial count and its value at the end of an operation is equal to the number of bytes transferred between main memory and the CU.

Set Count and Start Write

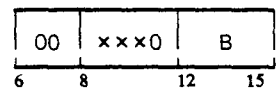
An IOW with command field:



sets the count register to the value of the data register of the IOW and initiates the transfer of data from main memory to the CU. The I/O busy bit is set to 1 if a previously initiated operation had not completed when the instruction is initiated.

Set Count and Start Read

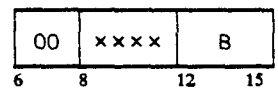
An IOW with command field:



sets the count register to the value of the data register of the IOW and initiates the DMA read operation by sending a one byte DMA read command to the CU. The CU then proceeds to send data to the DMA. The I/O busy bit is set to 1 if a previously initiated operation had not yet completed when the instruction is initiated.

Read Count

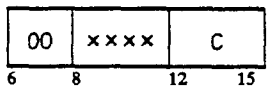
An IOR with command field:



transfers the count register into the data register of the IOR. The I/O busy bit is always set to zero by this instruction.

DMA Halt

An IOW with command field:



stops a DMA read or write operation. If the operation is a read, a HALT command is sent to the CU. Any data already received from the CU is transferred to main memory but subsequent data bytes are ignored except for CRC data checking. If the operation was a write, the transfer is concluded by completing the transmission of any partially sent data byte and then sending the concluding CRC and flag bytes. If the data transfer had already been complete or not yet been started, the instruction is equivalent to a NO-OP. The I/O busy bit is always set to zero by the instruction, and the interrupt request line is raised at the completion of the Halt operation.

12.7.2 Conclusion of DMA Operations

An interrupt request is generated at the end of all DMA read and write operations. The normal end of a read occurs when the concluding flag is received from the CU and the data has been stored in main memory. The normal end of a write occurs when the concluding flag byte has been sent to the CU and the CU summary status byte has been received. The status byte is stored in the BSA receive queue and may be read by the program.

Abnormal endings of DMA operations either generate a DMA interrupt request or an I/O check interrupt. Information detailing the cause of the abnormal end is placed in the status register extension and is available to the program as soon as the interrupt is generated.

Operations which have incorrect length (the number of bytes transferred is different from the initial value set in the count register) but which otherwise execute normally are not considered to have abnormal endings. Either the DMA or CU may recognize the data end condition. For the DMA, data end occurs when its count register goes to zero while for CU' s, data end will depend on the type of CU. There are four cases to consider

(1) DMA recognizes data end on a write: The DMA concludes its transmission and waits for the CU to respond with its summary status byte.
(2) CU recognizes data end on a read. The CU concludes its transmission and the DMA completes the transfer of the data to main memory.
(3) CU recognizes data end on a write before receiving the final flag byte: The CU sends its summary status byte with the data end and incorrect length bits set and the DMA concludes its transmission in response to the status byte.
(4) DMA recognizes data end on a read before receiving the final flag byte: The DMA sends a Halt command to the CU which concludes its transmission in response to the Halt Command.

Note: To avoid confusion between DMA and BSA interruptions, the BSA interrupt mask should be set to zeros before a DMA operation is initiated.

12.7.3 Status Register

The status register and the status register extension provide to the program the status of the DMA and the conditions under which an I/O operation was concluded. The DMA status register shown in Figure 12.6 is the same as the BSA status register except that certain bits which are not used by the BSA are utilized. The status register is accessed by the program with the same IOW and IOR instructions used for the BSA.

Incorrect Length. This bit is set to 1 on DMA read operations when either then CU sent less bytes than the initial count or else attempted to send more bytes than the initial count. The bit is unaffected during DMA write operations, incorrect length is detectable on writes by the value of the CU summary status byte sent by the CU at the conclusion of the write.

DMA Working. This bit is set to 1 by the set count and start write or set count and start read commands and is reset to zero at the end of the operation.

Bit	Designation	Type	Interrupt	Mask
0	Interrupt Summary	R/O	YES	NO
1	Link Idle	R/W	NO	-
2	Incorrect Length	R/W	NO	-
3	DMA Working	R/O	NO	-
4	Receive queue half-full	R/O	YES	YES
5	Transmit queue half-full	R/O	YES	YES
6	Frame received	R/W	YES	YES
7	Frame Transmitted	R/W	YES	YES
8-11	Interrupt Mask	R/W	NO	-
12-15	Received Byte Counter	R/O	NO	-

Figure 12.6. **DMA Status Register**

For a description of the other bits in the status register, see the BSA status register.

12.7.4 Status Register Extension

The format of the DMA status register extension is shown in Figure 12.7.

Bit	Designation	Type	Interrupt	Mask
0	Bus Parity error	R/W	NO	-
1	Invalid Instruction	R/W	NO	-
2	Abort Received	R/W	YES	NO
3	Transmission Overrun	R/W	YES	NO
4	Receiver Overrun	R/W	YES	NO
5	Transmission Error	R/W	YES	NO
6	Clock Failure	R/W	YES	NO
7	Program Controlled Interrupt	R/W	YES	NO
8	Diagnostic Mode	-	-	-
9	Memory Parity Error	R/W	NO	NO
10	Memory Overrun	R/W	NO	NO
11	Memory Sequence Error	R/W	NO	NO
12-15	*not used*			

Figure 12.7. **DMA Status Register Extension**

Bits 0 to 8 are the same as for the BSA. The register is accessed with the same IOW and IOR instructions used for the BSA.

Memory Parity Error. A parity error was detected by the DMA on data transferred on the MIO Bus. An I/O check interrupt is taken by the CPU.

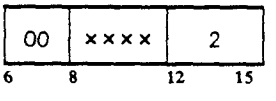
Memory Overrun. The memory system did not respond to a DMA memory request in time. An I/O check interrupt is taken by the CPU.

Memory Sequence Error. A sequence error was detected by the DMA during a MIO Bus operation. An I/O check interrupt is taken by the CPU.

When any of the errors corresponding to bits 0-6 occur, the action taken is as described for the BSA and in addition any DMA read or write in progress is terminated. For the memory errors above, the following action is taken. Any transmission in progress is aborted and both transmitter and receiver put in the idle state. Data transfer to or from main memory is terminated, the appropriate bits in the status register extension set, and the read or write operation concluded.

DMA Reset

An IOW with command field:



causes the DMA to perform its reset function. The I/O busy bit is always set to zero.

Reset Function. Execution of the reset function leaves the DMA in the following state:

(1) All registers set to zero.
(2) Transmit and Receive queues empty.
(3) Transmitter in idle state, transmitting 1' s.
(4) Receiver in idle state, looking for the first flag of a frame.

12.8 External Interrupt Adapter

External Interrupt Adapters (EIA) can be tailored for specific application environments. The following describes the EIA which will be built for the 801 prototype system.

12.8.1 Functional Description

The EIA provides for 32 interrupt sources. Twenty-seven of the 32 may be external to the EIA and five are generated internally within the IA. These internal interrupts consist of a timer interrupt and four programmed controlled interrupts (PCI). Each of the interrupt sources is associated with a unique bit in one of two 16 bit registers, IRVO and IRV1. The assignment of these 32 bit positions to interrupt sources is EIA model and configuration dependent. There is no hardware priority between the sources; priority is under program control.

Each interrupt is selectively masked by two 16 bit registers, MASK0 and MASK1; in these registers, 1 allows the interrupt and 0 masks the interrupt. The external interrupt line to the 801 is activated when and only when a non-masked interrupt source is a logical one.

The EIA has a 32 bit clock with a one microsecond resolution which can be read or written by the program. The clock is continuously compared by the EIA hardware with the value set in the interrupt time register. When the clock is greater than the value in the register the timer interrupt latch is set to 1. The interrupt latch is also set each time the clock "wraps around" from 111....1 to zero.

All programmed I/O instructions directed to the EIA set the I/O busy bit in the CPU' s condition register to zero; *ie.* the data transfer associated with I/O instructions is always completed. The first two bits in the command field of all PIO instructions are zero; only the remaining 8 bits are given below.

Interrupt Request Vector. Each interrupt source is associated with a bit position in one of the two 16 bit interrupt request vectors IRVO and IRV1. IRV0 (IRV1) may be read using a command field of X' 03' (X' 04'). The vectors are read only (the interrupt must be "reset" at the source).

Interrupt Mask. MASK0 (MASK1) is a 16 bit register which selectively masks each interrupt source in IRV0 (IRV1). MASK0 (MASK1) may be read or written using a command field of X' 05' (X' 06').

Clock. The 32 bit clock has a one microsecond resolution. The low (high) order 2 bytes may be read or written using a command field of X' 07' (X' 08'). When the low order 2 bytes are read, the high order 2 bytes are moved to the clock back up register. The back up register may be read using a command field of X' 09' .

Interrupt Time Register. The 32 bit interrupt time register is continuously compared with the clock and the timer interrupt latch is set to one when the value in clock is greater than the value in the register. The low (high) order 2 bytes of the register may be read or written using a command field of X' 0A' (X' 0B'). The write instructions reset the timer interrupt latch to zero (if the value written is less than the clock, the latch remains equal to 1).

12.8.2 Status Register

The EIA status register, shown in Figure 12.8, may be read or written using a command field of X' 00' .

Bit	Designation	Type
0-3	Programmed Controlled Interrupts	R/W
4	Timer Interrupt Latch	R/O
5	Bus Parity Error	R/W
6	Invalid Instruction	R/W
7-15	*not used*	-

Figure 12.8. **EIA Status Register**

Bits in the status register which are read only are unaffected by the IOW.

Programmed Controlled Interrupts. The program may use these bits to generate interrupt requests. The bits are wired to positions in IRV0 or IRV1.

Timer Interrupt Latch. The latch is set to 1 when the value in the clock is greater than the value in the interrupt time register or when the clock wraps around from 1111..1 to zero. The bit is wired to a position in IRV0 or IRV1. Once set to 1, it remains equal to one until reset by an IOW addressed to the interrupt time register which makes its value greater than or equal to the clock. The bit is R/O relative to an IOW addressed to the status register.

Bus Parity Error. An I/O Bus parity error occurred during the execution of the current PIO instruction. An I/O check interrupt is taken by the CPU.

Invalid Instruction. The current PIO instruction is invalid. An I/O check interrupt is taken by the CPU.

12.8.3 EIA Reset

The EIA is reset by the I/O system reset function or by an IOW with command field of X' 02' . The state after the EIA is reset is as follows.

(1) Status Register set to zero.
(2) Clock set to zero.
(3) Interrupt Masks set to zero.

12.9 Switch

The Switch allows system configurations with fewer BSAs and DMAs than Control Units. It switches full duplex serial links transparently with respect to the link protocol: bits which are received at its inlets are transmitted without modification to the outlets to which they are connected.

The design of the Switch will be specified at a later date; the Switch and Switch Adapter are not in the initial 801 implementation.

12.10 Errors in the I/O Subsystem

Errors in the I/O subsystem are classified according to where the errors are detected and how the program is alerted to the error. Errors which are detected during the execution of PIO instructions or during the transfer of data between the MIO Bus and an adapter (DMA) cause an I/O check interrupt. Information concerning the error is available to the program in the Interrupt Status Byte (ISB) and in the status register(s) in the adapter. All other errors do not result in an I/O check interrupt. Such errors may occur in an adapter, a serial link, a Control Unit or an IO device but they are only reported by an adapter or Control Unit by an interrupt request to the IA.

12.10.1 Errors in Executing PIO Instructions

An I/O check interrupt is taken by the CPU and the data transfer associated with the I/O instruction is suppressed. The value of the instruction counter stored at location 280 (hexadecimal) is equal to the address of the failing PIO instruction. Bits 0 to 3 of the ISB have their standard meaning and the remaining bits are used as follows.

Bit	Designation
4	PIO Error
5	CPU Parity Error
6	Bus Time Out
7	*not used*

PIO Error. Bit 4 is set to 1 if the I/O check interrupt was caused by a PIO instruction; it is set to zero if the error was associated with a MI/O Bus operation.

CPU Parity Error. The CPU has detected a parity error on the I/O Bus during the execution of the instruction.

Bus Timeout. No adapter has responded to the instruction within the I/O Bus timeout period. The cause of the timeout may be any of the following:

(1) No adapter recognizes the address/command field of the instruction. The address may actually be invalid or a bus error may have occurred in some bit positions of the address.
(2) An adapter is selected but it detected an IO Bus parity error in the command field or in the data (IOW).
(3) An adapter is selected but the command field is invalid.

In cases (2) and (3), bits in the status register(s) of the selected adapter are appropriately set.

12.10.2 MIO Bus Errors

An I/O check interrupt is taken by the CPU at the conclusion of the current instruction (but not between a branch and execute and its following instruction). The affected DMA terminates its data transfer with the Control Unit as described in Section 7.4 above. The value of the instruction counter stored at location 280 (hexadecimal) is equal to the address of the next instruction to be executed, and the interrupted program can be resumed at that address. Bits 0 to 3 of the ISB have their standard meaning and the remaining bits are used as follows.

Bit	Designation
4	MIO Error
5	Read/Write
6	MIO Bus Parity

MIO Error. Bit 4 is set to zero if the I/O check interrupt was caused by an MIO error; it is set to 1 if the error was associated with a PIO instruction.

Read/Write. DMA read or write operations involve transfer of data and control in both directions on the MIO Bus. Bit 5 is set to 1 if the error occurred during a transfer towards main memory and is set to zero if the transfer was towards the DMA.

MIO BUS Parity. A parity error was detected by the main memory side of the MIO bus.

DMA Detected Error. The error in the MIO Bus operation was detected by the DMA side of the MIO Bus. Information distinguishing between memory overrun, parity error or a sequence error is available in the DMA status register.

12.10.3 Errors that do not cause an I/O Check Interrupt

This class of errors is detected by IO adapters or control units. When the error is detected by an adapter, it sets its status register(s) appropriately and requests a CPU interrupt via the Interrupt Adapter (IA). If the interrupt is not masked in the EIA and the CPU is enabled, the program is alerted to the error at the conclusion of the current instruction. Otherwise, the program can only discover that an error occurred by examining the adapter' s status register(s).

When an error is detected by a Control Unit, the CU' s action may be any of the following:

(1) If the error is detected during the initiation of a CU command, it sends its summary status byte to the BSA or DMA with the appropriate bits set.
(2) If the error occurred during a read type operation bit before data end, the CU aborts the transmission.
(3) If the error occurred during a write operation but before data end, it sends its summary status byte with the appropriate bits set.
(4) If the error is detected after data end, the CU goes into the interrupt pending state.

In general, detailed information concerning the error is available to the program in CU status registers.

12.11 I/O System Reset

The I/O system reset is initiated from the operator panel by pressing system reset or power on. Each adapter on the IO Bus and each Control Unit performs its reset function. Reset causes the termination of any I/O operation in progress and the resetting of all status information and interruption conditions. The Control Units are put in the available state.

12.12 Initial Program Load (IPL)

Pressing the IPL button on the operator console initiates the following actions.

(1) The CPU is reset and both instruction and data caches are invalidated.
(2) The I/O system is reset as described in Section 12.11, above.

(3) The DMA selected for IPL by the console switches is initialized with a count of 100 hexadecimal and a read operation is initiated with the attached Control Unit.
(4) Upon completion of the read operation the CPU begins normal instruction execution starting at location zero.

CODE	MNE	FORM	PG	TYPE	INSTRUCTION
0					
1	STCX	(X)	10	strge	store char,X-form
2	STHX	(X)	10	strge	store half,X-form
3	STX	(X)	10	strge	store,X-form
4	LHAX	(X)	9	strge	load half algebraic,X-form
5	LHZX	(X)	9	strge	load half zero,X-form
6	LX	(X)	10	strge	load,X-form
7	CAX	(X)	11	adres	compute address,X-form
80	AI	(D)	21	arith	add immediate
81	MC00	(R)	16	move	move character zero from zero
82	MFC0	(R)	17	move	move from character indexed by SX0
83	MTC0	(R)	18	move	move to character indexed by SX0
84	TGTE	(R)	15	trap	trap if greater than or equal
85	A	(R)	20	arith	add
86	SHL	(R)	27	shift	shift half left
87	BBR	(R)	13	brnch	branch on bit, R-form
88	SHLP	(R)	27	shift	shift half left paired
89	MTCR	(R)	18	move	move to condition
8A	SHLPI	(R)	27	shift	shift half left paired immediate
8B	SHLI	(R)	27	shift	shift half left immediate
8C	TNEI	(D)	15	trap	trap if not equal immediate
8D	MFICR	(R)	17	move	move from ISB and condition reg.
8E	BB	(BI)	13	brnch	branch on bit
8F	ICBI	(R)	16	move	insert condition bit immediate
90	AEI	(D)	21	arith	add extended immediate
91	MC01	(R)	16	move	move character zero from one
92	MFC1	(R)	17	move	move from character indexed by SX1
93	MTC1	(R)	18	move	move to character indexed by SX1
94	TLT	(R)	15	trap	trap if less than
95	AE	(R)	20	arith	add extended
96	SHR	(R)	27	shift	shift half right
97	BBRX	(R)	13	brnch	branch on bit and execute, R-form
98	SHRP	(R)	27	shift	shift half right paired
99	BEX	(R)	13	brnch	branch,execute and enable
9A	SHRPI	(R)	28	shift	shift half right paired immediate
9B	SHRI	(R)	27	shift	shift half right immediate
9C	TLTI	(D)	15	trap	trap if less than immediate
9D	MFIA	(R)	17	move	move from instruction address
9E	BBX	(BI)	13	brnch	branch on bit and execute
9F					
A0	SI	(D)	23	arith	subtract immediate
A1	MC10	(R)	16	move	move character one from zero
A2	MFC2	(R)	17	move	move from character indexed by SX2
A3	MTC2	(R)	18	move	move to character indexed by SX2
A4	C	(R)	21	arith	compare
A5	S	(R)	22	arith	subtract
A6	SL	(R)	28	shift	shift left
A7	BNBR	(R)	13	brnch	branch on not-bit, R-form
A8	SLP	(R)	28	shift	shift left paired
A9	MTMQ	(R)	19	move	move to MQ
AA	SLPI	(R)	28	shift	shift left paired immediate
AB	SLI	(R)	28	shift	shift left immediate
AC	CI	(D)	21	arith	compare immediate
AD	MFMQ	(R)	18	move	move from MQ
AE	BNB	(BI)	13	brnch	branch on not bit

AF	MFDP	(R)	17	move	move from digit paired
B0	SEI	(D)	22	arith	subtract extended immediate
B1	MC11	(R)	16	move	move character one from one
B2	MFC3	(R)	17	move	move from character indexed by SX3
B3	MTC3	(R)	19	move	move to character indexed by SX3
B4	CL	(R)	24	logic	compare logical
B5	SE	(R)	22	arith	subtract extended
B6	SR	(R)	28	shift	shift right
B7	BNBRX	(R)	14	brnch	branch on not-bit and execute, R-form
B8	SRP	(R)	29	shift	shift right paired
B9					
BA	SRPI	(R)	29	shift	shift right paired immediate
BB	SRI	(R)	28	shift	shift right immediate
BC	CLI	(D)	24	logic	compare logical immediate
BD					
BE	BNBX	(BI)	14	brnch	branch on not bit and execute
BF	MTDP	(R)	19	move	move to digit paired
C0	SFI	(D)	23	arith	subtract from immediate
C1	BALR	(R)	12	brnch	branch and link,R-form
C2	MFP	(R)	18	move	move from prefix
C3	MTP	(R)	19	move	move to prefix
C4	STCD	(D)	10	strge	store char,D-form
C5					
C6	AD	(R)	20	arith	add decimal (not implemented)
C7					
C8	LHZD	(D)	9	strge	load half zero,D-form
C9					
CA	NSLPI	(D)	26	shift	and,then shift left paired immediate
CB	NSLI	(D)	26	shift	and,then shift left immediate
CC	TGTI	(D)	15	trap	trap if greater than immediate
CD					
CE	BALA	(BA)	12	brnch	branch and link absolute
CF	IOR	(D)	35	i-o	input-output read
D0	NI	(D)	24	logic	and immediate
D1	BALRX	(R)	12	brnch	branch and link with execute,R-form
D2	CLZ	(R)	24	logic	count leading zeros
D3	ABS	(R)	20	arith	absolute
D4	STHD	(D)	10	strge	store half,D-form
D5	N	(R)	24	logic	and
D6	SD	(R)	22	arith	subtract decimal (not implemented)
D7	MUS	(R)	21	arith	multiply step
D8	LHAD	(D)	9	strge	load half algebraic,D-form
D9	INDCL	(R)	30	sys	instruct data cache line
DA	NSRPI	(D)	26	shift	and,then shift right paired immediate
DB	NSRI	(D)	26	shift	and,then shift right immediate
DC					
DD	DI	(R)	30	sys	disable
DE	BALAX	(BA)	12	brnch	branch and link absolute with execute
DF	IOW	(D)	35	i-o	input-output write
E0	OI	(D)	25	logic	or immediate
E1					
E2					
E3	EXTS	(R)	21	arith	extend sign
E4	STD	(D)	10	strge	store,D-form
E5	O	(R)	25	logic	or
E6	SAR	(R)	26	shift	shift algebraic right
E7	DIS	(R)	21	arith	divide step

E8	LD	(D)	10	strge	load,D-form
E9	INICL	(R)	31	sys	invalidate instruction cache line
EA					
EB	SARI	(R)	26	shift	shift algebraic right immediate
EC	IPI	(D)	16	move	insert prefix immediate
ED	EI	(R)	30	sys	enable
EE	BALI	(BI)	12	brnch	branch and link,I-form
EF	CAD	(D)	11	adres	compute address,D-form
F0	XI	(D)	25	logic	exclusive or immediate
F1					
F2	MFTB	(R)	18	move	move from test bit
F3	TPO	(R)	23	arith	test prefix for overflow
F4	MTTB	(R)	19	move	move to test bit
F5	X	(R)	25	logic	exclusive or
F6	MFD	(R)	17	move	move from digit
F7	MTD	(R)	19	move	move to digit
F8					
F9					
FA	MFTBI	(R)	17	move	move from test bit immediate
FB	MTTBI	(R)	19	move	move to test bit immediate
FC	IPIZ	(D)	16	move	insert prefix immediate and zero
FD	ZNOP	(R)	31	sys	zero-time no-op
FE	BALIX	(BI)	12	brnch	branch and link with execute,I-form
FF					

MNE	CODE	FORM	PG	TYPE	INSTRUCTION
A	85	(R)	20	arith	add
ABS	D3	(R)	20	arith	absolute
AD	C6	(R)	20	arith	add decimal (not implemented)
AE	95	(R)	20	arith	add extended
AEI	90	(D)	21	arith	add extended immediate
AI	80	(D)	21	arith	add immediate
BALA	CE	(BA)	12	brnch	branch and link absolute
BALAX	DE	(BA)	12	brnch	branch and link absolute with execut
BALI	EE	(BI)	12	brnch	branch and link,I-form
BALIX	FE	(BI)	12	brnch	branch and link with execute,I-form
BALR	C1	(R)	12	brnch	branch and link,R-form
BALRX	D1	(R)	12	brnch	branch and link with execute,R-form
BB	8E	(BI)	13	brnch	branch on bit
BBR	87	(R)	13	brnch	branch on bit, R-form
BBRX	97	(R)	13	brnch	branch on bit and execute, R-form
BBX	9E	(BI)	13	brnch	branch on bit and execute
BEX	99	(R)	13	brnch	branch,execute and enable
BNB	AE	(BI)	13	brnch	branch on not bit
BNBR	A7	(R)	13	brnch	branch on not-bit, R-form
BNBRX	B7	(R)	14	brnch	branch on not-bit and execute, R-form
BNBX	BE	(BI)	14	brnch	branch on not bit and execute
C	A4	(R)	21	arith	compare
CAD	EF	(D)	11	adres	compute address,D-form
CAX	7	(X)	11	adres	compute address,X-form
CI	AC	(R)	21	arith	compare immediate
CL	B4	(D)	24	logic	compare logical
CLI	BC	(D)	24	logic	compare logical immediate
CLZ	D2	(R)	24	logic	count leading zeros
DI	DD	(R)	30	sys	disable
DIS	E7	(R)	21	arith	divide step
EI	ED	(R)	30	sys	enable
EXTS	E3	(R)	21	arith	extend sign
ICBI	8F	(R)	16	move	insert condition bit immediate
INDCL	D9	(R)	30	sys	instruct data cache line
INICL	E9	(R)	31	sys	invalidate instruction cache line
IOR	CF	(D)	35	i-o	input-output read
IOW	DF	(D)	35	i-o	input-output write
IPI	EC	(D)	16	move	insert prefix immediate
IPIZ	FC	(D)	16	move	insert prefix immediate and zero
LD	E8	(D)	10	strge	load,D-form
LHAD	D8	(D)	9	strge	load half algebraic,D-form
LHAX	4	(X)	9	strge	load half algebraic,X-form
LHZD	C8	(D)	9	strge	load half zero,D-form
LHZX	5	(X)	9	strge	load half zero,X-form
LX	6	(X)	10	strge	load,X-form
MC00	81	(R)	16	move	move character zero from zero
MC01	91	(R)	16	move	move character zero from one
MC10	A1	(R)	16	move	move character one from zero
MC11	B1	(R)	16	move	move character one from one
MFC0	82	(R)	17	move	move from character indexed by SX0
MFC1	92	(R)	17	move	move from character indexed by SX1
MFC2	A2	(R)	17	move	move from character indexed by SX2
MFC3	B2	(R)	17	move	move from character indexed by SX3
MFD	F6	(R)	17	move	move from digit
MFDP	AF	(R)	17	move	move from digit paired

MFIA	9D	(R)	17	move	move from instruction address
MFICR	8D	(R)	17	move	move from ISB and condition reg.
MFMQ	AD	(R)	18	move	move from MQ
MFP	C2	(R)	18	move	move from prefix
MFTB	F2	(R)	18	move	move from test bit
MFTBI	FA	(R)	18	move	move from test bit immediate
MUS	D7	(R)	21	arith	multiply step
MTCR	89	(R)	18	move	move to condition
MTC0	83	(R)	18	move	move to character indexed by SX0
MTC1	93	(R)	18	move	move to character indexed by SX1
MTC2	A3	(R)	18	move	move to character indexed by SX2
MTC3	B3	(R)	19	move	move to character indexed by SX3
MTD	F7	(R)	19	move	move to digit
MTDP	BF	(R)	19	move	move to digit paired
MTMQ	A9	(R)	19	move	move to MQ
MTP	C3	(R)	19	move	move to prefix
MTTB	F4	(R)	19	move	move to test bit
MTTBI	FB	(R)	19	move	move to test bit immediate
N	D5	(R)	24	logic	and
NI	D0	(D)	24	logic	and immediate
NSLI	CB	(D)	26	shift	and,then shift left immediate
NSLPI	CA	(D)	26	shift	and,then shift left paired immediate
NSRI	DB	(D)	26	shift	and,then shift right immediate
NSRPI	DA	(D)	26	shift	and,then shift right paired immediate
O	E5	(R)	25	logic	or
OI	E0	(D)	25	logic	or immediate
S	A5	(R)	22	arith	subtract
SAR	E6	(R)	27	shift	shift algebraic right
SARI	EB	(R)	27	shift	shift algebraic algebraic immediate
SD	D6	(R)	22	arith	subtract decimal (not implemented)
SE	B5	(R)	22	arith	subtract extended
SEI	B0	(D)	22	arith	subtract extended immediate
SFI	C0	(D)	23	arith	subtract from immediate
SHL	86	(R)	27	shift	shift half left
SHLI	8B	(R)	27	shift	shift half left immediate
SHLP	88	(R)	27	shift	shift half left paired
SHLPI	8A	(R)	27	shift	shift half left paired immediate
SHR	96	(R)	27	shift	shift half right
SHRI	9B	(R)	27	shift	shift half right immediate
SHRP	98	(R)	27	shift	shift half right paired
SHRPI	9A	(R)	28	shift	shift half right paired immediate
SI	A0	(D)	23	arith	subtract immediate
SL	A6	(R)	28	shift	shift left
SLI	AB	(R)	28	shift	shift left immediate
SLP	A8	(R)	28	shift	shift left paired
SLPI	AA	(R)	28	shift	shift left paired immediate
SR	B6	(R)	28	shift	shift right
SRI	BB	(R)	28	shift	shift right immediate
SRP	B8	(R)	29	shift	shift right paired
SRPI	BA	(R)	29	shift	shift right paired immediate
STCD	C4	(D)	10	strge	store char,D-form
STCX	1	(X)	10	strge	store char,X-form
STD	E4	(D)	10	strge	store,D-form
STHD	D4	(D)	10	strge	store half,D-form
STHX	2	(X)	10	strge	store half,X-form
STX	3	(X)	10	strge	store,X-form
TGTE	84	(R)	15	trap	trap if greater than or equal

TGTI	CC	(D)	15	trap	trap if greater than immediate
TLT	94	(R)	15	trap	trap if less than
TLTI	9C	(D)	15	trap	trap if less than immediate
TNEI	8C	(D)	15	trap	trap if not equal immediate
TPO	F3	(R)	23	arith	test prefix for overflow
X	F5	(R)	25	logic	exclusive or
XI	F0	(D)	25	logic	exclusive or immediate
ZNOP	FD	(R)	31	sys	zero-time no-op