Signetics 2650 User Manual

Browse >

GENERAL FEATURES

Contents | < Browse | Browse >

The 2650 processor is a general purpose, single chip, fixed instruction set, parallel 8-bit binary processor. A general purpose processor can perform any data manipulations through execution of a stored sequence of machine instructions. The processor has been designed to closely resemble conventional binary computers, but executes variable length instructions of one to three bytes in length. BCD arithmetic is made possible through use of a special DAR machine instruction.

The 2650 is manufactured using Signetics' N-channel silicon gate MOS technology. N-channel provides high carrier mobility for increased speed and also allows the use of a single 5-volt power supply. Silicon gate provides for better density and speed. Standard 40-pin dual in-line packages are used for the processor.

The 2650 contains a total of seven general purpose registers, each eight bits long. They may be used as source or destination for arithmetic operations, as index registers, and for I/O transfers.

The processor can address up to 32768 bytes of memory in four pages of 8192 bytes each. The processor instructions are one, two or three bytes long, depending on the instruction. Variable length instructions tend to conserve memory space since a one- or two-byte instruction may often be used rather than a three-byte instruction. The first byte of each instruction always specifies the operation to be performed and the addressing mode to be used. Most instructions use six of the first eight bits for this purpose, with the remaining two bits forming the register field. Some instructions use the full eight bits as an operation code.

The most complex direct instruction is three bytes long and takes 9.6

microseconds to execute. This figure assumes that the processor is running at its maximum clock rate, and has an associated memory with cycle and access times of one microsecond or less. The fastest instruction executes in 4.8 microseconds.

The clock input to the processor is a single phase pulse train and uses only one interface pin. It requires a normal TTL voltage swing, so no special clock driver is required.

The Data Bus and Address signals are tri-state to provide convenience in system design. Memory and I/O interface signals are asynchronous so that Direct Memory Access (DMA) and multiprocessor operations are easy to implement.

The 2650 has a versatile set of addressing modes used for locating operands for operations. They are described in detail here.

The interrupt mechanism is implemented as a single level, address vectoring type. Address vectoring means that an interrupting device can force the processor to execute code at a device-determined location in memory.

INTERNAL REGISTERS

Contents | < Browse | Browse >

In order for the processor to execute an instruction, it performs the following general steps:

1. The Instruction Address Register (IAR) provides an address for memory.
2. The first byte of an instruction is fetched from memory and stored in the Instruction Register (IR).
3. The Instruction Register (IR) is decoded to determine the type of instruction and the addressing mode.
4. If an operand from memory is required, the operand address is resolved and loaded in the Operand Address Register (OAR).
5. The operand is fetched from memory and the operation is executed.
6. The first byte of the next instruction is fetched.

The Instruction Register (IR) holds the first byte of each instruction and directs the subsequent operations required to execute each instruction. The IR contents are decoded and used in conjunction with the timing information to control the activation and sequencing of all the other elements on the chip. The Holding Register (HR) is used in some multiple-byte instructions to contain further instruction information and partial absolute addresses.

The Arithmetic Logic Unit (ALU) is used to perform all of the data manipulation operations, including Load, Store, Add, Subtract, And, Inclusive Or, Exclusive Or, Compare, Rotate, Increment and Decrement. It contains and controls the Carry (C) bit, the Overflow (OVF) bit, the Interdigit Carry (IDC) bit and the Condition Code (CC) register.

The Register Stack (RS) contains six registers that are organized into two banks of three registers each. The Register Select (RS) bit picks one of the two banks to be accessed by instructions. In order to accommodate

the register-to-register instructions, register zero (R0) is outside the array. Thus, register zero is always available along with one set of three registers.

The Address Adder (AA) is used to increment the instruction address and to calculate relative and indexed addresses.

The Instruction Address Register (IAR) holds the address of the next instruction byte to be accessed. The Operand Address Register (OAR) stores operand addresses and sometimes contains intermediate results during effective address calculations.

The Return Address Stack (RAS) is an eight level, Last In, First Out (LIFO) storage which receives the return address whenever a Branch-to-Subroutine instruction is executed. When a Return instruction is executed, the RAS provides the last return address for the processor's IAR. The stack contains eight levels of storage so that subroutines may be nested up to eight levels deep. The Stack Pointer (SP) is a three-bit wraparound counter that indicates the next available level in the stack. It always points to the current address.

PROGRAM STATUS WORD

Contents | < Browse | Browse >

The Program Status Word (PSW) is a special purpose register within the processor that contains status and control bits. It is 16 bits long and is divided into two bytes called the Program Status Upper (PSU) and the Program Status Lower (PSL).

The PSW bits may be tested, loaded, stored, preset or cleared using the instructions which affect the PSW. The sense bit, however, cannot be set or cleared because it is directly connected to pin #1.

	7	6	5	4	3	2	1	0
PSU	S	F	II	Not used		SP2	SP1	SP0

S: Sense
F: Flag
II: Interrupt Inhibit
SP2: Stack Pointer 2
SP1: Stack Pointer 1
SP0: Stack Pointer 0

	7	6	5	4	3	2	1	0
PSL	CC1	CC0	IDC	RS	WC	OVF	COM	C

CC1: Condition Code 1
CC0: Condition Code 0
IDC: Interdigit Carry
RS: Register Bank Select
WC: With/Without Carry
OVF: Overflow
COM: Logical/Arithmetic Compare
C: Carry/Borrow

SENSE (S)
The Sense bit in the PSU reflects the logic state of the sense input to the processor at pin #1. The sense bit is not affected by the LPSU, PPSU, or CPSU instructions. When the PSU is tested (TPSU) or stored into register zero (SPSU), bit #7 reflects the state of the sense pin at the time of the instruction execution.

FLAG (F)
The Flag bit is a simple latch that drives the Flag output (pin #40) on the processor.

INTERRUPT INHIBIT (II)
When the Interrupt Inhibit (II) bit is set, the processor will not recognize an incoming interrupt. When interrupts are enabled (II=0), and an interrupt signal occurs, the inhibit bit in the PSU is then automatically set. When a RETE instruction is executed, the inhibit bit is automatically cleared.

STACK POINTER (SP)
The three Stack Pointer bits are used to address locations in the Return Address Stack (RAS). The SP designates the stack level which contains the current return address. The three SP bits are organized as a binary counter which is automatically incremented with execution of Branch-to-Subroutine instructions, and decremented with execution of Return instructions.

CONDITION CODE (CC)
The Condition Code is a two-bit register which is set by the processor whenever a general purpose register is loaded or modified by the execution of an instruction. Additionally, the CC is set to reflect the relative value of two bytes whenever a compare instruction is executed.
The following table indicates the setting of the CC whenever data is set into a general purpose register. The data byte is interpreted as an 8-bit, two's complement number.

Register Contents	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

For compare instructions the following table summarizes the setting of the CC. The data is compared as two 8-bit absolute numbers if bit #1, the COM bit, of the Program Status Lower (PSL) byte is set to indicate "logical" compare (COM=1). If the COM bit indicates "arithmetic" (COM=0), the comparison instructions interpret the data bytes as two 8-bit two's complement binary numbers.

Register to Storage Compare Instruction	Register to Register Compare Instruction	CC1	CC0
Reg X Greater than Storage	Reg #0 Greater than Reg X	0	1
Reg X Equal to Storage	Reg #0 Equal to Reg X	0	0
Reg X Less than Storage	Reg #0 Less than Reg X	1	0

The CC is never set to %11 by normal processor operations, but it may be explicitly set to %11 through LPSL or PPSL instruction execution.

INTERDIGIT CARRY (IDC)

For BCD arithmetic operations it is sometimes essential to know if there was a carry from bit #3 to bit #4 during the execution of an arithmetic instruction.

The IDC reflects the value of the Interdigit Carry from the previous add or subtract instruction. After any add or subtract instruction execution, the IDC contains the carry or borrow out of bit #3.

The IDC is also set upon execution of Rotate instructions when the WC bit in the PSW is set. The IDC will reflect the same information as bit #5 of the operand register after the rotate is executed.

REGISTER SELECT (RS)

There are two banks of general purpose registers with three registers in each bank. The register select bit is used to specify which set of three general purpose registers will be currently used. Register zero is common and is always available to the program. An individual instruction may address only four registers, but the bank select feature effectively expands the available on-chip registers to seven. When the Register Select bit is 0, registers #1, #2 and #3 in register bank #0 will be accessible, and when the bit is 1, registers #1, #2 and #3 in register bank #1 will be accessible.

WITH/WITHOUT CARRY (WC)

This bit controls the execution of the add, the subtract and the rotate instructions.

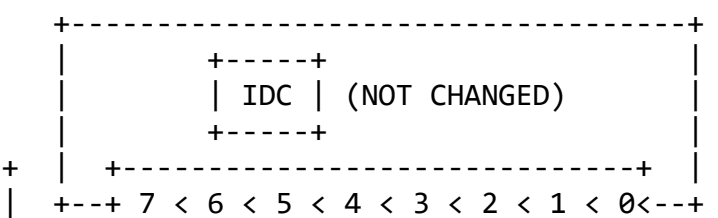
Whenever an add or a subtract instruction executes, the following bits are either set or cleared: Carry/Borrow (C), Overflow (OVF), and Interdigit Carry (IDC). These bits are set or reset without regard to the value of the WC bit. However, when WC=1, the final value of the carry bit affects the result of an add or subtract instruction, ie. the carry bit is either added (add instruction) or subtracted (subtract instruction) from the ALU.

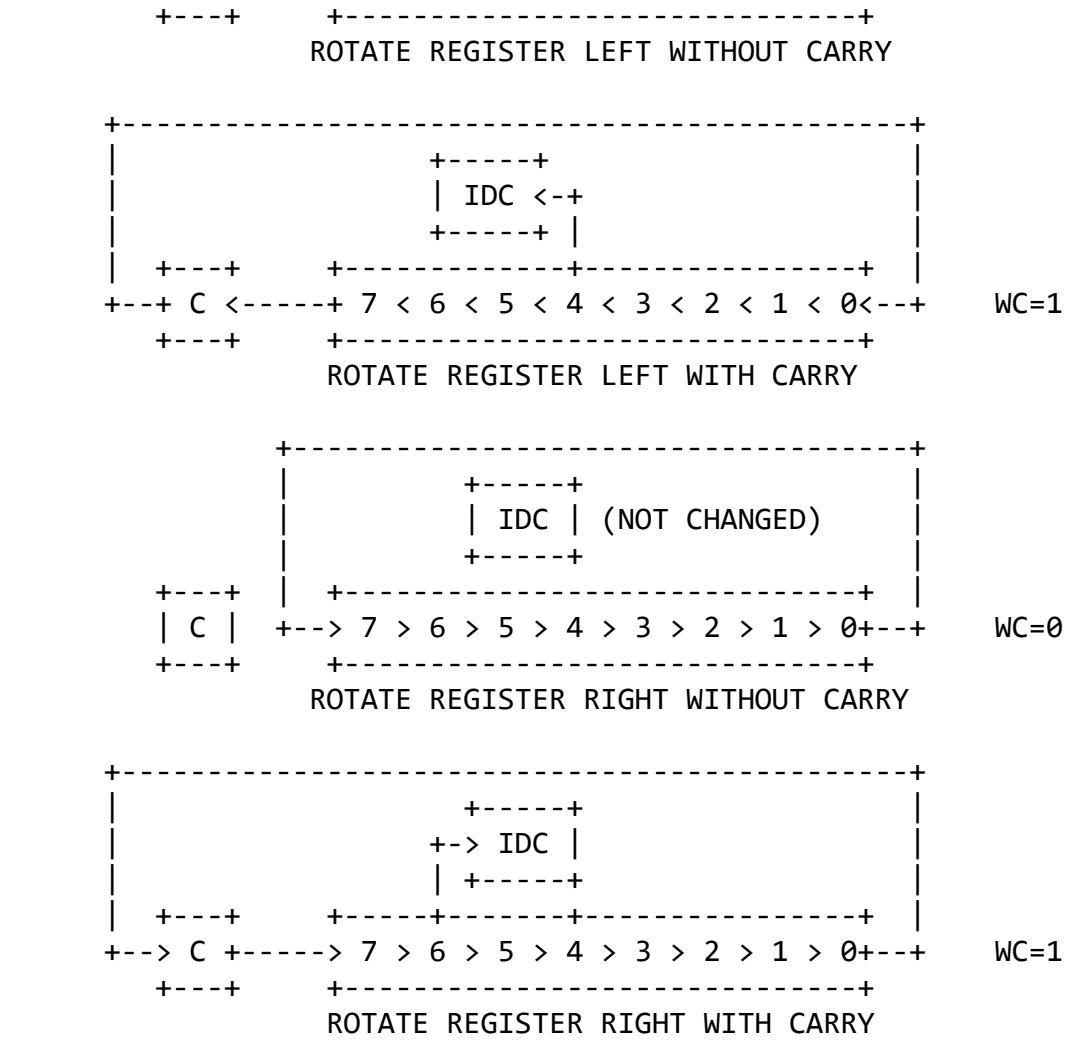
Whenever a rotate instruction executes with WC=0, only the eight bits of the rotated register are affected. However, when WC=1, the following bits are also affected: Carry/Borrow (C), Overflow (OVF) and Interdigit Carry (IDC). The carry/borrow bit is combined with the 8-bit register to make a nine-bit rotate. The overflow bit is set whenever the sign bit (bit #7) of the rotated register changes its value, ie. from a zero (0) to a one (1) or from a one (1) to a zero (0). The interdigit carry bit is set to the new value of bit #5 of the rotated register (!).

OVERFLOW (OVF)

The overflow bit is set during add or subtract instruction executions whenever the two initial operands have the same sign but the result has a different sign. Operands with different signs cannot cause overflow. For example: A binary +124 (%01111100) added to a binary +64 (%01000000) produces a result of %10111100 which is interpreted in two's complement form as a -68. The true answer would be 188, but that answer cannot be contained in the set of 8-bit, two's complement numbers used by the processor, so the OVF bit is set.

Rotate instructions also cause OVF to be set whenever the sign of the rotated byte changes.





COMPARE (COM)

The compare control bit determines the type of comparison that is executed with the Compare instructions. Either logical or arithmetic comparisons may be made. The arithmetic compare assumes that the comparison is between 8-bit, two's complement numbers. The logical compare assumes that the comparison is between 8-bit positive binary numbers. When COM is set to 1, the comparisons will be logical, and when COM is set to 0, the comparisons will be arithmetic.

CARRY (C)

The Carry bit is set by the execution of any add or subtract instruction that results in a carry or borrow out of the high order bit of the ALU. The carry bit is set to 1 by an add instruction that generates a carry, and a subtract instruction that does *not* generate a borrow. Inversely, an add that does not generate a carry causes the C bit to be cleared, and a subtract instruction that generates a borrow also clears the carry bit.

Even though a borrow is indicated by a zero in the Carry bit, the processor will correctly interpret the zero during subtract with borrow operations as in the following table.

Low Order bit Minuend	Low Order bit Subtrahend	Carry bit Borrow bit	Low Order Bit Result
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1

1	1	0	1
1	1	1	0

The carry bit may also be set or cleared by rotate instructions as described earlier under "With/Without Carry".

To perform an Add with Carry or a Subtract with Borrow, the WC bit must be set.

MEMORY ORGANIZATION

Contents | < Browse | Browse >

The 2650 has a maximum memory addressing capability of 0-32767 locations. Most direct addressing instructions have thirteen bits allocated for the direct address. Since thirteen bits can only address locations 0-8191, a paging system was implemented to accommodate the entire address range.

The memory may be thought of as being divided into four pages of 8192 bytes each. The addresses in each page range as in the following chart:

	Start Address	End Address	
page 0	%0000000000000000	%0011111111111111	0- 8191
page 1	%0100000000000000	%0111111111111111	8192-16383
page 2	%1000000000000000	%1011111111111111	16384-24575
page 3	%1100000000000000	%1111111111111111	24576-32767

The low order 13 bits in every page range through the same set of numbers. These 13 bits are the same 13 bits addressed by non-branch instructions and are also the same 13 bits which are brought out of the 2650 on the address lines ADR0-ADR12.

The high order two bits of the 15-bit address are known as the page bits. The page bits when examined by themselves also represent, in binary, the number of the memory page. Thus, the address %0100000001101101 is known as address location 109 in page 1. The page bits, corresponding to ADR13 and AD14, are brought out of the 2650 on pins 19 and 18. These bits may be used for memory access when more than 8192 bytes of memory are connected.

There are no instructions to explicitly set the page bits. They may be set through execution of direct or indirect branch or branch-to-subroutine instructions. It may be seen that these instructions have 15 bits allocated for the address and when such an instruction is executed, the two high order address bits are set into the page bit latches in the 2650 processor and will appear on ADR13 and ADR14 during memory accesses until they are specifically changed.

For memory access from non-branch instructions, the 13-bit direct address will address the corresponding location within the current page only. However, the non-branch memory access instruction may access any byte in any page through indirect addressing which provides the full 15-bit address. In the case of non-branch instructions, the page bits are only temporarily changed to correspond to the high order two bits of the 15-bit indirect address used to fetch the argument byte. Immediately after the memory access, ADR13 and ADR14 will revert to their previous value.

The consequences of this page address system may be summarized by the following statements.

1. The RESET signal clears both page latches, ie. ADR13 and ADR14 are

cleared to zero.

2. All non-branch direct memory access instructions address memory within the current page.

3. All non-branch memory access instructions may access any byte of addressable memory through use of indirect addressing which temporarily changes the page bits for the argument access, but which revert back to their previous state immediately following instruction execution.

4. All direct and indirect addressing branch instructions set the page bits to correspond to the high order two bits of the 15 bit address.

5. Programs may *not* flow across page boundaries, they must branch to set the page bits.

6. Interrupts always drive the processor to page zero.

SUBROUTINE LINKAGE

Contents | < Browse | Browse >

The on-chip stack, along with the Branch-to-Subroutine and Return instructions, provide the facility to transfer control to a subroutine. The subroutine can return control to the program that branched to it via a Return instruction.

The stack is eight levels deep which means that a routine may branch to a subroutine, which may branch to another subroutine, etc., eight times before any Return instructions are executed.

When designing a system that utilitizes interrupts, it should be remembered that the processor jams a ZBSR into the IAR and then executes it. This will cause an entry to be pushed into the on-chip stack like any other Branch-to-Subroutine instruction and may limit the stack depth available in certain programs.

When branching to a subroutine, the following sequence of events occurs:

1. The address in the IAR is used to fetch the Branch-to-Subroutine instruction and is then incremented in the Address Adder so that it points to the instruction following the subroutine branch.

2. The Stack Pointer (SP) is incremented by one so that it points to the next Return Address Stack (RAS) location.

3. The contents of the IAR are stored in the stack at the location designated by the Stack Pointer (SP).

4. The operand address contained in the Branch-to-Subroutine instruction (the address of the first instruction of the subroutine) is inserted into the IAR.

When returning from a subroutine, this sequence of events occurs:

1. The address in the IAR is used to fetch the return (RETC, RETE) instruction from memory.

2. When the return instruction is recognized by the processor, the contents of the stack entry pointed to by the Stack Pointer (SP) is placed into the IAR.

3. The Stack Pointer (SP) is decremented by one.

4. Instruction execution continues at the address now in the IAR.

CONDITION CODE USAGE

Contents | < Browse | Browse >

The two-bit register, called the Condition Code (CC), is incorporated in the Program Status Word (PSW). It may be seen in the description of the 2650 instructions that the CC is specifically set by every instruction that causes data to be transferred into a general purpose register and it is also set by compare instructions.

The reason for this design feature is that after an instruction executes, the CC contains a modest amount of information about the byte of data that has just been manipulated. For example, a program loads register #1 with a byte of unknown data and the CC setting indicates that the byte is positive, negative or zero. The negative indication implies that bit #7 is set to one.

Consequently, a data manipulation operation when followed by a conditional branch is often sufficient to determine desired information without resorting to a specific test, thus saving instructions and memory space.

In the following example, the CC is used to test the parity of a byte of data which is stored at symbolic memory location CHAR.

```
    EQ      EQU     0           ;the equal condition code
    CHAR    DATA    2           ;unknown data byte
    WC      EQU     $4          ;the With Carry (WC) bit
    LT      EQU     2           ;CC mask
            CPSL    WC          ;clear With Carry (WC) bit
            LODI,r2 -8          ;set up counter
            SUBZ    r0          ;clear reg #0
            LODR,r1 CHAR        ;get the character (CC is set)
    LOOP:   BCFR,LT  G01        ;if not set, don't count (CC is tested)
            ADDI,r0 +1          ;count the bit
    G01:    RRL,r1              ;move bits left (CC is set)
            BIRR,r2 LOOP        ;loop till done

    ;Finished, test if reg #0 has a one in low order
    ;If bit #0 = 1, odd parity. If bit #0 = 0, then even.

            TMI,r0  $1
            BCTR,EQ ODD
    EVEN:   HALT
    ODD:    HALT
```

ADDRESSING MODES

Contents | < Browse | Browse >

An addressing mode is a method the processor uses for developing argument addresses for machine instructions.
  The 2650 processor can develop addresses in eight ways:

* Register addressing
* Immediate addressing
* Relative addressing
* Relative, indexed addressing
* Absolute addressing
* Absolute, indirect addressing
* Absolute, indexed addressing
* Absolute, indirect, indexed addressing

However, of these eight addressing modes, only four of them are basic. The others are variations due to indexing and indirection. The following text describes how effective addresses are developed by the processor.

REGISTER ADDRESSING

All register-to-register instructions are one byte in length. Instructions utilizing this addressing mode appear in this general format.

```
    oooooorr
    76543210
```

o: operation code
r: register

Since there are only two bits designated to specify a register, register zero always contains one of the operands while the other operand is in one of the three registers in the currently selected bank. Register zero may also be specified as the explicit operand giving instructions such as: EORZ r0.
  In one byte register addressing instructions which have just one operand, any of the currently selected general purpose registers or register zero may be specified, eg. RRL,r0.

IMMEDIATE ADDRESSING

All immediate addressing instructions are two bytes in length. The first byte contains the operation code and register designation, while the second byte contains data used as the argument during instruction execution.

```
    oooooorr    vvvvvvvv
    76543210    76543210
     Byte 0      Byte 1
```

o: operation code
r: register
v: two's complement binary number or 8-bit logic mask

The second byte, the data byte, may contain a binary number or a logic

mask depending on the particular instruction being executed. Any register may be designated in the first byte.

RELATIVE ADDRESSING

Relative addressing instructions are all two bytes in length and are memory reference instructions. One argument of the instruction is a register and the other argument is the contents of a memory location. The format of relative addressing instructions is:

oooooorr	iaaaaaaa
76543210	76543210
Byte 0	Byte 1

o: operation code
r: register
i: indirect addressing flag
a: relative displacement

The first byte contains the operation code and register designation, while the second byte contains the relative address. Bits #0-#6, byte #1, contain a 7-bit two's complement binary number which can range from -64 to +63. This number is used by the processor to calculate the effective address. The effective address is calculated by adding the address of the first byte following a relative addressing instruction to the relative displacement in the second byte of the instruction.

If bit #7, byte #1, is set to "1", the processor will enter an indirect addressing cycle, where the actual operand address will be accessed from the effective address location. See Indirect Addressing.

Two of the branch instructions (ZBSR, ZBRR) allow addressing relative to page zero, byte #0, of memory. In this case, values up to +63 reference the first 63 bytes of page zero and values up to -64 reference the last 64 bytes of page zero.

ABSOLUTE ADDRESSING FOR NON-BRANCH INSTRUCTIONS

Absolute addressing instructions are all three bytes in length and are memory reference instructions. One argument of the instruction is a register, designated in bits #1 and #0, byte #0; the other argument is the contents of a memory location. The format of absolute addressing instructions is:

oooooorr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210
Byte 0	Byte 1	Byte 2

o: operation code
r: index register or argument register
i: indirect addressing flag
c: index control bits
a: address

Bits #4-#0, byte #1, and #7-#0, byte #2, contain the absolute address and can address any byte within the same page that the instruction appears in.

The index control bits, bits #6 and #5, byte #1, determine how the effective address will be calculated and possibly which register will be the argument during instruction execution. The index control bits have the following interpretation:

Index control Bit #6	Bit #5	Meaning
0	0	Non-indexed address
0	1	Indexed with auto-increment
1	0	Indexed with auto-decrement
1	1	Indexed only

When the index control bits are 0 and 0, bits #1 and #0 in byte #0 contain the argument register designation and bits #0 to #4, byte #1, and bits #0 to #7, byte #2, contain the effective address. Indirect addressing may be specified by setting bit #7, byte #1, to a one.

When the index control bits are 1 and 1, bits #1 and #0 in byte #0 designate the index register and the argument register implicitly becomes register zero. The effective address is calculated by adding the contents of the index register (8-bit absolute integer) to the address field. If indirect addressing is specified, the indirect address is accessed and then the value in the index register is added to the indirect address. This is commonly called post indexing.

When the index control bits are 0 and 1, the address is calculated exactly as when the control bits contain 1 and 1 *except* a binary 1 is added to the contents of the selected index register *before* the calculation of the effective address proceeds. Similarly, when the index control bits contain 1 and 0, a binary 1 is subtracted from the contents of the selected index register *before* the effective address is calculated.

ABSOLUTE ADDRESSING FOR BRANCH INSTRUCTIONS

The three-byte absolute addressing branch instructions deviate slightly in format from ordinary absolute addressing instructions as shown below:

oooooorr	ippaaaaa	aaaaaaaa
76543210	76543210	76543210
Byte 0	Byte 1	Byte 2

o: operation code
r: register or condition code mask
i: indirect addressing flag
p: page
a: address

The notable difference is that bits #6 and #5, byte #1, are no longer interpreted as Index Control bits, but instead are interpreted as the high order bits of the address field. This means that there is no indexing allowed on most absolute addressing branch instructions, but indexed branches are possible through use of the BXA and BSXA instructions. The bits #6 and #5, byte #1, are used to set the current page register, thus enabling programs to directly transfer control to another page.

See the Memory Organization, BXA and BSXA instructions, and Indirect Addressing.

INDIRECT ADDRESSING

Indirect addressing means that the argument address of an instruction is not specified by the instruction itself, but rather the argument address will be found in the two bytes pointed to by the address field or relative address field, of absolute or relative addressing instructions. In the case of absolute addressing, the value of the index register is added to

the indirect address, *not* to the value in the address field of the instruction. In both cases, the processor will enter the indirect addressing state when the bit designated "I" is set to one. Entering the indirect addressing sequence adds two cycles (6 clock periods) to the execution time of an instruction.

Indirect addresses are 15-bit addresses stored right justified in two contiguous bytes of memory. As such, an indirect address may specify any location in addressable memory (0-32767). The high order bit of the two-byte indirect address is not used by the processor.

Only single level indirect addressing is implemented. The following examples demonstrate indirect addressing.

```
%00001110    %10000000    %01010001    LODA,r2 *$51
Address $10  Address $11  Address $12

             %00000001    %00101000    ACON    $128
             Address $51  Address $52

             %01100111                 DATA    $67
             Address $128
```

The LODA instruction in memory locations $10, $11 and $12 specifies indirect addressing (bit #7, byte #1, is set). Therefore, when the instruction is executed, the processor takes the address field value, $51, and uses it to access the two-byte indirect address at $51 and $52. Then using the contents of $51 and $52 as the effective address, the data byte containing $67 is loaded into register #2.

```
%00001010    %10000101                 LODR,r2 *$17
Address $10  Address $11

             %00000001    %00101000    ACON    $128
             Address $17  Address $18

             %01100111                 DATA    $67
             Address $128
```

In a fashion similar to the previous example, the relative address is used to access the indirect address which points to the data byte. When the LODR instruction is execute, the data byte contents, $67, will be loaded into register #2.

INSTRUCTION FORMAT EXCEPTIONS

There are several instructions which are detect by decoding the entire 8 bits of the first byte of the instruction. These instructions are unique and may be noticed in the instruction descriptions. Examples are: HALT, CPSU, CPSL.

Of this type of instruction, two operation codes were taken from otherwise complete sets thus eliminating certain possible operations. The cases are as follows:

(NOT OKAY)	STRZ r0	Storing register zero into register zero is not
(OKAY)	NOP	implemented, the operation code is used for NOP (no operation).
(NOT OKAY)	ANDZ r0	AND of register zero with register zero is not
(OKAY)	HALT	implemented, the operation code is used for HALT.

INSTRUCTION FORMATS

Contents | < Browse | Browse >

```
(Z) REGISTER ADDRESSING      oooooorr

o: operation code
r: register number/value or condition

(I) IMMEDIATE ADDRESSING     oooooorr    vvvvvvvv

o: operation code
r: register number
v: data mask or binary value

(R) RELATIVE ADDRESSING      oooooorr    iaaaaaaa

o: operation code
r: register number/value or condition
i: indirect bit
a: relative displacement (-64 <= displacement <= +63)

(A) ABSOLUTE ADDRESSING      oooooorr    iccaaaaa    aaaaaaaa
    (NON-BRANCH INSTRUCTIONS)

o: operation code
r: register number/index register number
i: indirect bit
c: index control bits
a: absolute address

(B) ABSOLUTE ADDRESSING      oooooorr    ippaaaaa    aaaaaaaa
    (BRANCH INSTRUCTIONS)

o: operation code
r: register number/value or condition
i: indirect bit
p: page
a: absolute address

    INDIRECT ADDRESSING      uppaaaaa    aaaaaaaa

u: unused
p: page
a: absolute address

(E) MISCELLANEOUS            oooooooo
    INSTRUCTIONS

o: operation code

Index control:
    %00 = non-indexed
    %01 = indexed with auto-increment
```

```
%10 = indexed with auto-decrement
%11 = indexed only
```

ALPHABETIC LISTING

Contents | < Browse | Browse >

ADDA	ADDI	ADDR	ADDZ
ANDA	ANDI	ANDR	ANDZ
BCFA	BCFR		
BCTA	BCTR		
BDRA	BDRR		
BIRA	BIRR		
BRNA	BRNR		
BSNA	BSNR		
BSFA	BSFR		
BSTA	BSTR		
BSXA			
BXA			
COMA	COMI	COMR	COMZ
CPSL	CPSU		
DAR			
EORA	EORI	EORR	EORZ
HALT			
IORA	IORI	IORR	IORZ
LODA	LODI	LODR	LODZ
LDPL *	LPSL	LPSU	
NOP			
PPSL	PPSU		
REDC	REDD	REDE	
RETC	RETE		
RRL	RRR		
SPSL	SPSU	STPL *	
STRA	STRR	STRZ	
SUBA	SUBI	SUBR	SUBZ
TMI			
TPSL	TPSU		
WRTC	WRTD	WRTE	
ZBRR			
ZBSR			

NUMERIC LISTING

Contents | < Browse | Browse >

$00	Indeterminate	$80	ADDZ r0
$01	LODZ r1	$81	ADDZ r1
$02	LODZ r2	$82	ADDZ r2
$03	LODZ r3	$83	ADDZ r3
$04	LODI,r0	$84	ADDI,r0
$05	LODI,r1	$85	ADDI,r1
$06	LODI,r2	$86	ADDI,r2

$07	LODI,r3	$87	ADDI,r3
$08	LODR,r0	$88	ADDR,r0
$09	LODR,r1	$89	ADDR,r1
$0A	LODR,r2	$8A	ADDR,r2
$0B	LODR,r3	$8B	ADDR,r3
$0C	LODA,r0	$8C	ADDA,r0
$0D	LODA,r1	$8D	ADDA,r1
$0E	LODA,r2	$8E	ADDA,r2
$0F	LODA,r3	$8F	ADDA,r3
$10	LDPL *	$90	Undefined
$11	STPL *	$91	Undefined
$12	SPSU	$92	LPSU
$13	SPSL	$93	LPSL
$14	RETC,eq	$94	DAR,r0
$15	RETC,gt	$95	DAR,r1
$16	RETC,lt	$96	DAR,r2
$17	RETC,un	$97	DAR,r3
$18	BCTR,eq	$98	BCFR,eq
$19	BCTR,gt	$99	BCFR,gt
$1A	BCTR,lt	$9A	BCFR,lt
$1B	BCTR,un	$9B	ZBRR
$1C	BCTA,eq	$9C	BCFA,eq
$1D	BCTA,gt	$9D	BCFA,gt
$1E	BCTA,lt	$9E	BCFA,lt
$1F	BCTA,un	$9F	BXA,r3
$20	EORZ r0	$A0	SUBZ r0
$21	EORZ r1	$A1	SUBZ r1
$22	EORZ r2	$A2	SUBZ r2
$23	EORZ r3	$A3	SUBZ r3
$24	EORI,r0	$A4	SUBI,r0
$25	EORI,r1	$A5	SUBI,r1
$26	EORI,r2	$A6	SUBI,r2
$27	EORI,r3	$A7	SUBI,r3
$28	EORR,r0	$A8	SUBR,r0
$29	EORR,r1	$A9	SUBR,r1
$2A	EORR,r2	$AA	SUBR,r2
$2B	EORR,r3	$AB	SUBR,r3
$2C	EORA,r0	$AC	SUBA,r0
$2D	EORA,r1	$AD	SUBA,r1
$2E	EORA,r2	$AE	SUBA,r2
$2F	EORA,r3	$AF	SUBA,r3
$30	REDC,r0	$B0	WRTC,r0
$31	REDC,r1	$B1	WRTC,r1
$32	REDC,r2	$B2	WRTC,r3
$33	REDC,r3	$B3	WRTC,r3
$34	RETE,eq	$B4	TPSU
$35	RETE,gt	$B5	TPSL
$36	RETE,lt	$B6	Undefined
$37	RETE,un	$B7	Undefined
$38	BSTR,eq	$B8	BSFR,eq
$39	BSTR,gt	$B9	BSFR,gt
$3A	BSTR,lt	$BA	BSFR,lt
$3B	BSTR,un	$BB	ZBSR
$3C	BSTA,eq	$BC	BSFA,eq
$3D	BSTA,gt	$BD	BSFA,gt
$3E	BSTA,lt	$BE	BSFA,lt
$3F	BSTA,un	$BF	BSXA,r3
$40	HALT	$C0	NOP

$41	ANDZ r1	$C1	STRZ r1
$42	ANDZ r2	$C2	STRZ r2
$43	ANDZ r3	$C3	STRZ r3
$44	ANDI,r0	$C4	Undefined
$45	ANDI,r1	$C5	Undefined
$46	ANDI,r2	$C6	Undefined
$47	ANDI,r3	$C7	Undefined
$48	ANDR,r0	$C8	STRR,r0
$49	ANDR,r1	$C9	STRR,r1
$4A	ANDR,r2	$CA	STRR,r2
$4B	ANDR,r3	$CB	STRR,r3
$4C	ANDA,r0	$CC	STRA,r0
$4D	ANDA,r1	$CD	STRA,r1
$4E	ANDA,r2	$CE	STRA,r2
$4F	ANDA,r3	$CF	STRA,r3
$50	RRR,r0	$D0	RRL,r0
$51	RRR,r1	$D1	RRL,r1
$52	RRR,r2	$D2	RRL,r2
$53	RRR,r3	$D3	RRL,r3
$54	REDE,r0	$D4	WRTE,r0
$55	REDE,r1	$D5	WRTE,r1
$56	REDE,r2	$D6	WRTE,r2
$57	REDE,r3	$D7	WRTE,r3
$58	BRNR,r0	$D8	BIRR,r0
$59	BRNR,r1	$D9	BIRR,r1
$5A	BRNR,r2	$DA	BIRR,r2
$5B	BRNR,r3	$DB	BIRR,r3
$5C	BRNA,r0	$DC	BIRA,r0
$5D	BRNA,r1	$DD	BIRA,r1
$5E	BRNA,r2	$DE	BIRA,r2
$5F	BRNA,r3	$DF	BIRA,r3
$60	IORZ r0	$E0	COMZ,r0
$61	IORZ r1	$E1	COMZ,r1
$62	IORZ r2	$E2	COMZ,r2
$63	IORZ r3	$E3	COMZ,r3
$64	IORI,r0	$E4	COMI,r0
$65	IORI,r1	$E5	COMI,r1
$66	IORI,r2	$E6	COMI,r2
$67	IORI,r3	$E7	COMI,r3
$68	IORR,r0	$E8	COMR,r0
$69	IORR,r1	$E9	COMR,r1
$6A	IORR,r2	$EA	COMR,r2
$6B	IORR,r3	$EB	COMR,r3
$6C	IORA,r0	$EC	COMA,r0
$6D	IORA,r1	$ED	COMA,r1
$6E	IORA,r2	$EE	COMA,r2
$6F	IORA,r3	$EF	COMA,r3
$70	REDD,r0	$F0	WRTD,r0
$71	REDD,r1	$F1	WRTD,r1
$72	REDD,r2	$F2	WRTD,r2
$73	REDD,r3	$F3	WRTD,r3
$74	CPSU	$F4	TMI,r0
$75	CPSL	$F5	TMI,r1
$76	PPSU	$F6	TMI,r2
$77	PPSL	$F7	TMI,r3
$78	BSNR,r0	$F8	BDRR,r0
$79	BSNR,r1	$F9	BDRR,r1
$7A	BSNR,r2	$FA	BDRR,r2

$7B	BSNR,r3	$FB	BDRR,r3
$7C	BSNA,r0	$FC	BDRA,r0
$7D	BSNA,r1	$FD	BDRA,r1
$7E	BSNA,r2	$FE	BDRA,r2
$7F	BSNA,r3	$FF	BDRA,r3

* = 2650-B only

ORGANIZED BY FUNCTION

Contents | < Browse | Browse >

LOAD/STORE
- LODZ
- LODI
- LODR
- LODA
- STRZ
- STRR
- STRA

ARITHMETIC
- ADDZ
- ADDI
- ADDR
- ADDA
- SUBZ
- SUBI
- SUBR
- SUBA

LOGICAL
- IORZ
- IORI
- IORR
- IORA
- EORZ
- EORI
- EORR
- EORA
- ANDZ
- ANDI
- ANDR
- ANDA

BRANCH
- BCTR
- BCTA
- BCFR
- BCFA
- BRNR
- BRNA
- BIRR
- BIRA
- BDRR

BDRA
BXA
ZBRR

SUBROUTINE BRANCH
BSTR
BSTA
BSFR
BSFA
BSNR
BSNA
BSXA
ZBSR

SUBROUTINE RETURN
RETC
RETE

COMPARISON
COMZ
COMI
COMR
COMA

INPUT/OUTPUT
REDC
REDD
WRTC
WRTD
REDE
WRTE

PROGRAM STATUS MANIPULATION
LDPL*
LPSU
LPSL
SPSU
SPSL
STPL*
CPSU
CPSL
PPSU
PPSL
TPSU
TPSL

ROTATE
RRR
RRL

MISCELLANEOUS
NOP
HALT
TMI
DAR

* = 2650-B only

ADDA

Contents | < Browse | Browse >

ADD ABSOLUTE

Addressing Mode:	Absolute
Size:	3 bytes
Signetics Mnemonic:	ADDA,r (*)a(,X)
CALM Mnemonic:	ADD|ADDC reg,abs
	ADD|ADDC reg,@abs
	ADD|ADDC A,(reg)+abs
	ADD|ADDC A,(reg)+@abs
	ADD|ADDC A,(+reg)+abs
	ADD|ADDC A,(+reg)+@abs
	ADD|ADDC A,(-reg)+abs
	ADD|ADDC A,(-reg)+@abs
IEEE-964 Mnemonic:	ADD|ADDC reg,/abs
	ADD|ADDC reg,/@abs
	ADD|ADDC .0,/abs(reg)
	ADD|ADDC .0,/@abs(reg)
	ADD|ADDC .0,/abs(+reg)
	ADD|ADDC .0,/@abs(+reg)
	ADD|ADDC .0,/abs(-reg)
	ADD|ADDC .0,/@abs(-reg)
Binary Code:	$8C-$8F

100011rr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time:	4 cycles (12 clock periods)
Processor Registers Affected:	C, CC, IDC, OVF

Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This three-byte instruction causes the contents of register r and the contents of the byte of memory pointed to by the effective address to be added together in a true binary adder. The eight-bit sum replaces the contents of register r.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

Note:

Add with Carry may be effected. See Carry (C) bit.

Pseudocode:

```
    ADDA,rn abs          ;rn += *(abs);                          ;4,3
    ADDA,r0 abs,rn       ;r0 += *(abs + rn);                     ;4,3
    ADDA,r0 abs,rn+      ;r0 += *(abs + ++rn);                   ;4,3
                         ;or, rn++; r0 += *(abs + rn);
    ADDA,r0 abs,rn-      ;r0 += *(abs + --rn);                   ;4,3
                         ;or, rn--; r0 += *(abs + rn);
    ADDA,rn *abs         ;rn += *(*(abs));                       ;6,3
    ADDA,r0 *abs,rn      ;r0 += *(*(abs) + rn);                  ;6,3
    ADDA,r0 *abs,rn+     ;r0 += *(*(abs) + ++rn);                ;6,3
                         ;or, rn++; r0 += *(*(abs) + rn);
    ADDA,r0 *abs,rn-     ;r0 += *(*(abs) + --rn);                ;6,3
                         ;or, rn--; r0 += *(*(abs) + rn);
```

ADDI

Contents | < Browse | Browse >

ADD IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: ADDI,r v
CALM Mnemonic: ADD|ADDC reg,#imm
IEEE-694 Mnemonic ADD|ADDC reg,#imm
Binary Code: $84-$87

100001rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r and the contents of the second byte of this instruction to be added together in a true binary adder. The eight-bit sum replaces the contents of register r.

Note:
Add with Carry may be effected. See Carry (C) bit.

Pseudocode:

```
    ADDI,rn imm          ;rn += imm;                             ;2,2
```

ADDR

Contents | < Browse | Browse >

ADD RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: ADDR,r (*)a
CALM Mnemonic: ADD|ADDC reg,.+rel
ADD|ADDC reg,@.+rel
IEEE-694 Mnemonic: ADD|ADDC reg,$rel
ADD|ADDC reg,$@rel
Binary Code: $88-$8B

100010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r and the contents of the byte of memory pointed to by the effective address to be added together in a true binary adder. The eight-bit sum replaces the contents of register r.
Indirect addressing may be specified.

Note:
Add with Carry may be effected. See Carry (C) bit.

Pseudocode:

```
ADDR,rn rel          ;rn += *(rel);                    ;3,2
ADDR,rn *rel         ;rn += *(*(rel));                 ;5,2
```

ADDZ

Contents | < Browse | Browse >

ADD TO REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: ADDZ r
CALM Mnemonic: ADD|ADDC A,reg
IEEE-694 Mnemonic ADD|ADDC .0,reg
Binary Code: $80-$83

```
    100000rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the contents of the specified register, r, and the contents of register zero to be added together in a true binary adder. The eight-bit sum of the addition replaces the contents of register zero. The contents of register r remain unchanged.

Note:
Add with Carry may be effected. See Carry (C) bit.

Pseudocode:

```
    ADDZ    rn          ;r0 += rn;                          ;2,1
```

ANDA

Contents | < Browse | Browse >

AND ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: ANDA,r (*)a(,X)
CALM Mnemonic:
AND reg,abs
AND reg,@abs
AND A,(reg)+abs
AND A,(reg)+@abs
AND A,(+reg)+abs
AND A,(+reg)+@abs
AND A,(-reg)+abs
AND A,(-reg)+@abs
IEEE-964 Mnemonic:
AND reg,/abs
AND reg,/@abs
AND .0,/abs(reg)

```
                                AND    .0,/@abs(reg)
                                AND    .0,/abs(+reg)
                                AND    .0,/@abs(+reg)
                                AND    .0,/abs(-reg)
                                AND    .0,/@abs(-reg)
```

Binary Code: $4C-$4F

010011rr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This three-byte instruction causes the contents of register r to be logically ANDed with the contents of the memory byte pointed to by the effective address. The result of the operation replaces the contents of register r.

The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

Pseudocode:

```
ANDA,rn abs          ;r0 &= *(abs);                          ;4,3
ANDA,r0 abs,rn       ;r0 &= *(abs + rn);                     ;4,3
ANDA,r0 abs,rn+      ;r0 &= *(abs + ++rn);                   ;4,3
                     ;or, rn++; r0 &= *(abs + rn);
ANDA,r0 abs,rn-      ;r0 &= *(abs + --rn);                   ;4,3
                     ;or, rn--; r0 &= *(abs + rn);
ANDA,rn *abs         ;rn &= *(*(abs));                       ;6,3
ANDA,r0 *abs,rn      ;r0 &= *(*(abs) + rn);                  ;6,3
ANDA,r0 *abs,rn+     ;r0 &= *(*(abs) + ++rn);                ;6,3
                     ;or, rn++; r0 &= *(*(abs) + rn);
ANDA,r0 *abs,rn-     ;r0 &= *(*(abs) + --rn);                ;6,3
                     ;or, rn--; r0 &= *(*(abs) + rn);
```

ANDI

Contents | < Browse | Browse >

AND IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: ANDI,r v
CALM Mnemonic: AND reg,#imm
IEEE-694 Mnemonic AND reg,#imm
Binary Code: $44-$47

010001rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r to be logically ANDed with the contents of the second byte of this instruction. The result of this operation replaces the contents of register r.

The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Pseudocode:

```
ANDI,rn imm          ;rn &= imm;                              ;2,2
```

ANDR

Contents | < Browse | Browse >

AND RELATIVE

Addressing Mode: Relative

Size: 2 bytes
Signetics Mnemonic: ANDR,r (*)a
CALM Mnemonic:
```
AND    reg,.+rel
AND    reg,@.+rel
```
IEEE-694 Mnemonic:
```
AND    reg,$rel
AND    reg,$@rel
```
Binary Code: $48-$4B

010010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r to be logically ANDed with the contents of the memory byte pointed to by the effective address. The result of this operation replaces the contents of register r.
Indirect indexing may be specified.
The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Pseudocode:

```
ANDR,rn rel          ;rn &= *(rel);                          ;3,2
ANDR,rn *rel         ;rn &= *(*(rel));                       ;5,2
```

ANDZ

Contents | < Browse | Browse >

AND TO REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: ANDZ r
CALM Mnemonic: AND A,reg
IEEE-694 Mnemonic: AND .0,reg

Binary Code: $41-$43

```
    010000rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the contents of the specified register, r, to be logically ANDED with the contents of register zero. The result of the operation replaces the contents of register zero. The contents of register r remain unchanged.
The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Note:
Register r may not be specified as zero. This operation code, %01000000, is reserved for HALT.

Pseudocode:

```
    ANDZ    rn          ;r0 &= rn;                          ;2,1
```

BCFA

Contents | < Browse | Browse >

BRANCH ON CONDITION FALSE, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BCFA,v (*)a
CALM Mnemonic: JUMP,NE abs
JUMP,NE @abs
JUMP,LE abs
JUMP,LE @abs
JUMP,GE abs
JUMP,GE @abs
IEEE-694 Mnemonic: BNE /abs
BNE /@abs

```
                                BLE     /abs
                                BLE     /@abs
                                BGE     /abs
                                BGE     /@abs
```

Binary Code: $9C-$9E

```
    100111vv    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210
```

v: condition (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Synonyms:

```
    BCFA,eq = BNEA = Branch if Not Equal, Absolute       after compare
    BCFA,gt = BNHA = Branch if Not Higher, Absolute      after compare
    BCFA,lt = BNLA = Branch if Not Lower, Absolute       after compare

    BCFA,eq = BNZA = Branch if Not Zero, Absolute        after load/arithmetic
    BCFA,gt = BNPA = Branch if Not Positive, Absolute    after load/arithmetic
    BCFA,lt = BNLA = Branch if Not Minus, Absolute       after load/arithmetic
```

Description:

  This three-byte conditional branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field does not match the current Condition Code (CC) field in the Program Status Word. If there is no match, the contents of the Instruction Address Register (IAR) are replaced by the effective address.

  If the v field and CC field match, the next instruction is fetched from the location following the third byte of this instruction.

  Indirect addressing may be specified.

  The v field may not be set to $3 as this bit combination (%10011111) is used for the BXA operation code.

Pseudocode:

```
    BCFA,eq abs         ;if != goto abs;                    ;3,3
    BCFA,gt abs         ;if <= goto abs;                    ;3,3
    BCFA,lt abs         ;if >= goto abs;                    ;3,3
    BCFA,eq *abs        ;if != goto *(abs);                 ;3+2,3
    BCFA,gt *abs        ;if <= goto *(abs);                 ;3+2,3
    BCFA,lt *abs        ;if >= goto *(abs);                 ;3+2,3
```

BCFR

Contents | < Browse | Browse >

BRANCH ON CONDITION FALSE, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: BCFR,v (*)a
CALM Mnemonic:
```
JUMP,NE .+rel
JUMP,NE @.+rel
JUMP,LE .+rel
JUMP,LE @.+rel
JUMP,GE .+rel
JUMP,GE @.+rel
```
IEEE-694 Mnemonic:
```
BNE     $rel
BNE     $@rel
BLE     $rel
BLE     $@rel
BGE     $rel
BGE     $@rel
```
Binary Code: $98-$9A

```
    100110vv    iaaaaaaa
    76543210    76543210
```

v: condition (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Synonyms:

```
    BCFR,eq = BNER = Branch if Not Equal, Relative       after compare
    BCFR,gt = BNHR = Branch if Not Higher, Relative      after compare
    BCFR,lt = BNLR = Branch if Not Lower, Relative       after compare

    BCFR,eq = BNZR = Branch if Not Zero, Relative        after load/arithmetic
    BCFR,gt = BNPR = Branch if Not Positive, Relative    after load/arithmetic
    BCFR,lt = BNLR = Branch if Not Minus, Relative       after load/arithmetic
```

Description:
This two-byte conditional branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field does not match the current Condition Code (CC) field in the Program Status Word. If there is no match, the contents of the Instruction Address Register (IAR) are replaced by the effective address.

If the v field and CC field match, the next instruction is fetched from the location following the second byte of this instruction.

Indirect addressing may be specified.

The v field may not be set to $3 as this bit combination (%10011011) is used for the ZBRR operation code.

Pseudocode:

```
    BCFR,eq rel         ;if != goto rel;                      ;3,2
    BCFR,gt rel         ;if <= goto rel;                      ;3,2
    BCFR,lt rel         ;if >= goto rel;                      ;3,2
    BCFR,eq *rel        ;if != goto *(rel);                 ;3+2,2
```

```
    BCFR,gt *rel         ;if <= goto *(rel);                    ;3+2,2
    BCFR,lt *rel         ;if >= goto *(rel);                    ;3+2,2
```

BCTA

Contents | < Browse | Browse >

BRANCH ON CONDITION TRUE, ABSOLUTE

Addressing Mode:	Absolute
Size:	3 bytes
Signetics Mnemonic:	BCTA,v (*)a
CALM Mnemonic:	JUMP,EQ abs or JUMP,AO abs JUMP,EQ @abs or JUMP,AO @abs JUMP,GT abs JUMP,GT @abs JUMP,LT abs or JUMP,NO abs JUMP,LT @abs or JUMP,NO @abs JUMP abs JUMP @abs
IEEE-694 Mnemonic:	BEQ /abs BEQ /@abs BGT /abs BGT /@abs BLT /abs BLT /@abs BR /abs BR /@abs
Binary Code:	$1C-$1F

```
    000111vv    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210
```

v: condition (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time:	3 cycles (9 clock periods)
Processor Registers Affected:	None
Condition Code Setting:	N/A

Synonyms:

```
    BCTA,eq = BEA = Branch if Equal, Absolute       after compare
    BCTA,gt = BHA = Branch if Higher, Absolute      after compare
    BCTA,lt = BLA = Branch if Lower, Absolute       after compare

    BCTA,eq = BZA = Branch if Zero, Absolute        after load/arithmetic
    BCTA,gt = BPA = Branch if Positive, Absolute    after load/arithmetic
    BCTA,lt = BLA = Branch if Minus, Absolute       after load/arithmetic

    BCTA,eq = BOA = Branch if Ones, Absolute        after test
    BCTA,lt = BMA = Branch if Mixed, Absolute       after test
```

```
    BCTA,un = BA  = Branch Absolute                anytime
```

Description:

This three-byte conditional branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field matches the current Condition Code (CC) field in the Program Status Word. If there is a match, the contents of the Instruction Address Register (IAR) are replaced by the effective address.

If the v field and CC field do not match, the next instruction is fetched from the location following the third byte of this instruction.

Indirect addressing may be specified.

If the v field is set to $3, an unconditional branch is effected.

Pseudocode:

```
    BCTA,eq abs         ;if == goto abs;                    ;3,3
    BCTA,gt abs         ;if > goto abs;                     ;3,3
    BCTA,lt abs         ;if < goto abs;                     ;3,3
    BCTA,un abs         ;goto abs;                          ;3,3
    BCTA,eq *abs        ;if == goto *(abs);                ;3+2,3
    BCTA,gt *abs        ;if > goto *(abs);                 ;3+2,3
    BCTA,lt *abs        ;if < goto *(abs);                 ;3+2,3
    BCTA,un *abs        ;goto *(abs);                       ;5,3
```

BCTR

Contents | < Browse | Browse >

BRANCH ON CONDITION TRUE, RELATIVE

Addressing Mode:	Relative
Size:	2 bytes
Signetics Mnemonic:	BCTR,v (*)a
CALM Mnemonic:	JUMP,EQ .+rel or JUMP,AO .+rel
	JUMP,EQ @.+rel or JUMP,AO @.+rel
	JUMP,GT .+rel
	JUMP,GT @.+rel
	JUMP,LT .+rel or JUMP,NO .+rel
	JUMP,LT @.+rel or JUMP,NO @.+rel
	JUMP .+rel
	JUMP @.+rel
IEEE-694 Mnemonic:	BEQ $rel
	BEQ $@rel
	BGT $rel
	BGT $@rel
	BLT $rel
	BLT $@rel
	BR $rel
	BR $@rel
Binary Code:	$18-$1B

```
    000110vv    iaaaaaaa
    76543210    76543210
```

v: condition (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Synonyms:

```
    BCTR,eq = BER = Branch if Equal, Relative       after compare
    BCTR,gt = BHR = Branch if Higher, Relative      after compare
    BCTR,lt = BLR = Branch if Lower, Relative       after compare

    BCTR,eq = BZR = Branch if Zero, Relative        after load/arithmetic
    BCTR,gt = BPR = Branch if Positive, Relative    after load/arithmetic
    BCTR,lt = BLR = Branch if Minus, Relative       after load/arithmetic

    BCTR,eq = BOR = Branch if Ones, Relative        after test
    BCTR,lt = BMR = Branch if Mixed, Relative       after test

    BCTR,un = BR  = Branch Relative                 anytime
```

Description:
This two-byte conditional branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field matches the current Condition Code (CC) field in the Program Status Word. If there is a match, the contents of the Instruction Address Register (IAR) are replaced by the effective address.
If the v field and CC field do not match, the next instruction is fetched from the location following the second byte of this instruction.
Indirect addressing may be specified.
If the v field is set to $3, an unconditional branch is effected.

Pseudocode:

```
    BCTR,eq rel         ;if == goto rel;                    ;3,2
    BCTR,gt rel         ;if > goto rel;                     ;3,2
    BCTR,lt rel         ;if < goto rel;                     ;3,2
    BCTR,un rel         ;goto rel;                          ;3,2
    BCTR,eq *rel        ;if == goto *(rel);                 ;3+2,2
    BCTR,gt *rel        ;if > goto *(rel);                  ;3+2,2
    BCTR,lt *rel        ;if < goto *(rel);                  ;3+2,2
    BCTR,un *rel        ;goto *(rel);                       ;5,2
```

BDRA

Contents | < Browse | Browse >

BRANCH ON DECREMENTING REGISTER, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BDRA,r (*)a

CALM Mnemonic: DECJ,NE reg,abs
DECJ,NE reg,@abs
DEC reg
IEEE-694 Mnemonic: DBNZ reg,/abs
DBNZ reg,/@abs
DEC reg
Binary Code: $FC-$FF

111111rr	ippaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This three-byte branch instruction causes the processor to decrement the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, ie. the effective address replaces the previous contents of the Instruction Address Register (IAR). If the new value in register r is zero, the next instruction to be executed follows the second byte of this instruction.
Indirect addressing may be specified.

Pseudocode:

```
BDRA,rn abs          ;if (--rn != 0) goto abs;                ;3,3
                     ;or, rn--; if (rn != 0) goto abs;
BDRA,rn *abs         ;if (--rn != 0) goto *(abs);           ;3+2,3
                     ;or, rn--; if (rn != 0) goto *(abs);
```

BDRR

Contents | < Browse | Browse >

BRANCH ON DECREMENTING REGISTER, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: BDRR,r (*)a
CALM Mnemonic: DECJ,NE reg,.+rel
DECJ,NE reg,@.+rel
DEC reg
IEEE-694 Mnemonic: DBNZ reg,$rel
DBNZ reg,$@rel
DEC reg
Binary Code: $F8-$FB

111110rr	iaaaaaaa

```
    76543210    76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:

This two-byte branch instruction causes the processor to decrement the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, ie. the effective address replaces the previous contents of the Instruction Address Register (IAR). If the new value in register r is zero, the next instruction to be executed follows the second byte of this instruction.

Indirect addressing may be specified.

Pseudocode:

```
    BDRR,rn abs         ;if (--rn != 0) goto abs;                ;3,2
                        ;or, rn--; if (rn != 0) goto abs;
    BDRR,rn *abs        ;if (--rn != 0) goto *(abs);           ;3+2,2
                        ;or, rn--; if (rn != 0) goto *(abs);
```

BIRA

Contents | < Browse | Browse >

BRANCH ON INCREMENTING REGISTER, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BIRA,r (*)a
CALM Mnemonic:
```
INCJ,NE reg,abs
INCJ,NE reg,@abs
INC     reg
```
IEEE-694 Mnemonic:
```
IBNZ    reg,/abs
IBNZ    reg,/@abs
INC     reg
```
Binary Code: $DC-$DF

```
    110111rr    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
  This three-byte branch instruction causes the processor to increment the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, ie. the effective address replaces the previous contents of the Instruction Address Register (IAR). If the new value in register r is zero, the next instruction to be executed follows the second byte of this instruction.
  Indirect addressing may be specified.

Pseudocode:

```
    BIRA,rn abs          ;if (++rn != 0) goto abs;                 ;3,3
                         ;or, rn++; if (rn != 0) goto abs;
    BIRA,rn *abs         ;if (++rn != 0) goto *(abs);            ;3+2,3
                         ;or, rn++; if (rn != 0) goto *(abs);
```

BIRR

Contents | < Browse | Browse >

BRANCH ON INCREMENTING REGISTER, RELATIVE

Addressing Mode:	Relative
Size:	2 bytes
Signetics Mnemonic:	BIRR,r (*)a
CALM Mnemonic:	INCJ,NE reg,.+rel
	INCJ,NE reg,@.+rel
	INC reg
IEEE-694 Mnemonic:	IBNZ reg,$rel
	IBNZ reg,$@rel
	INC reg
Binary Code:	$D8-$DB

```
    110110rr    iaaaaaaa
    76543210    76543210
```

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time:	3 cycles (9 clock periods)
Processor Registers Affected:	None
Condition Code Setting:	N/A

Description:
  This two-byte branch instruction causes the processor to increment the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, ie. the effective address replaces the previous contents of the Instruction Address Register (IAR). If the new value in register r is zero, the next instruction to be executed follows the second byte of this instruction.
  Indirect addressing may be specified.

Pseudocode:

```
    BIRR,rn abs          ;if (++rn != 0) goto abs;                 ;3,2
                         ;or, rn++; if (rn != 0) goto abs;
    BIRR,rn *abs         ;if (++rn != 0) goto *(abs);            ;3+2,2
                         ;or, rn++; if (rn != 0) goto *(abs);
```

BRNA

Contents | < Browse | Browse >

BRANCH ON REGISTER NON-ZERO, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BRNA,r (*)a
CALM Mnemonic: JUMP,regNE abs
JUMP,regNE @abs
IEEE-694 Mnemonic: BNZ reg,/abs
BNZ /@abs
Binary Code: $5C-$5F

```
    010111rr    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210
```

r: register (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This three-byte branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address, ie. the effective address replaces the contents of the Instruction Address Register (IAR).
If the specified register contains a zero value the next instruction is fetched from the location following the third byte of this instruction.
Indirect addressing may be specified.

Pseudocode:

```
    BRNA,rn abs          ;if (rn != 0) goto abs;                   ;3,3
    BRNA,rn *abs         ;if (rn != 0) goto *(abs);              ;3+2,3
```

BRNR

Contents | < Browse | Browse >

BRANCH ON REGISTER NON-ZERO, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: BRNR,r (*)a
CALM Mnemonic: JUMP,regNE .+rel
JUMP regNE @.+rel
IEEE-694 Mnemonic: BNZ reg,$rel
BNZ $@rel
Binary Code: $58-$5B

010110rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This two-byte branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address, ie. the effective address replaces the contents of the Instruction Address Register (IAR).
If the specified register contains a zero value the next instruction is fetched from the location following the second byte of this instruction.
Indirect addressing may be specified.

Pseudocode:

```
BRNR,rn rel          ;if (rn != 0) goto rel;                ;3,2
BRNR,rn *rel         ;if (rn != 0) goto *(rel);           ;3+2,2
```

BSFA

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE ON CONDITION FALSE, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BSFA,v (*)a
CALM Mnemonic: CALL,NE abs
CALL,NE @abs
CALL,LE abs
CALL,LE @abs
CALL,GE abs
CALL,GE @abs
IEEE-694 Mnemonic: CALLNE /abs

```
                                CALLNE  /@abs
                                CALLLE  /abs
                                CALLLE  /@abs
                                CALLGE  /abs
                                CALLGE  /@abs
Binary Code:                    $BC-$BE

    101111vv    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210

v: condition (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time:                 3 cycles (9 clock periods)
Processor Registers Affected:   SP
Condition Code Setting:         N/A
```

Description:

This three-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch only if the two-bit v field does not match the current Condition Code (CC) field in the Program Status Word (PSW). If the fields do not match, the Stack Pointer (SP) is incremented by one and the current contents of the Instruction Address Register (IAR), which points to the location following the third byte of this instruction, is pushed into the Return Address Stack (RAS). The effective address replaces the previous contents of the IAR.

If the v field and CC field match, the next instruction is fetched from the location following the third byte of this instruction and the Stack Pointer (SP) is unaffected.

Indirect addressing may be specified.

If the v field may not be coded as $3 as this bit combination (%10111111) is used for the BSXA operation code.

Pseudocode:

```
    BSFA,eq abs         ;if != gosub abs;                    ;3,3
    BSFA,gt abs         ;if <= gosub abs;                    ;3,3
    BSFA,lt abs         ;if >= gosub abs;                    ;3,3
    BSFA,eq *abs        ;if != gosub *(abs);                ;3+2,3
    BSFA,gt *abs        ;if <= gosub *(abs);                ;3+2,3
    BSFA,lt *abs        ;if >= gosub *(abs);                ;3+2,3
```

BSFR

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE ON CONDITION FALSE, RELATIVE

```
Addressing Mode:                Relative
Size:                           2 bytes
Signetics Mnemonic:             BSFR,v  (*)a
CALM Mnemonic:                  CALL,NE .+rel
                                CALL,NE @.+rel
```

```
                                CALL,LE .+rel
                                CALL,LE @.+rel
                                CALL,GE .+rel
                                CALL,GE @.+rel
IEEE-694 Mnemonic:              CALLNE  $rel
                                CALLNE  $@rel
                                CALLLE  $rel
                                CALLLE  $@rel
                                CALLGE  $rel
                                CALLGE  $@rel
Binary Code:                    $B8-$BA

    101110vv    iaaaaaaa
    76543210    76543210

v: condition (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time:                 3 cycles (9 clock periods)
Processor Registers Affected:   SP
Condition Code Setting:         N/A
```

Description:

This two-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch only if the two-bit v field does not match the current Condition Code (CC) field in the Program Status Word (PSW). If the fields do not match, the Stack Pointer (SP) is incremented by one and the current contents of the Instruction Address Register (IAR), which points to the location following the second byte of this instruction, is pushed into the Return Address Stack (RAS). The effective address replaces the previous contents of the IAR.

If the v field and CC field match, the next instruction is fetched from the location following the second byte of this instruction and the Stack Pointer (SP) is unaffected.

Indirect addressing may be specified.

The v field may not be coded as $3 as this bit combination (%10111011) is used for the ZBSR operation code.

Pseudocode:

```
    BSFR,eq rel         ;if != gosub rel;                   ;3,2
    BSFR,gt rel         ;if <= gosub rel;                   ;3,2
    BSFR,lt rel         ;if >= gosub rel;                   ;3,2
    BSFR,eq *rel        ;if != gosub *(rel);               ;3+2,2
    BSFR,gt *rel        ;if <= gosub *(rel);               ;3+2,2
    BSFR,lt *rel        ;if >= gosub *(rel);               ;3+2,2
```

BSNA

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE ON REGISTER NON-ZERO, ABSOLUTE

```
Addressing Mode:                Absolute
```

Size: 3 bytes
Signetics Mnemonic: BSNA,r (*)a
CALM Mnemonic: CALL,regNE abs
CALL,regNE @abs
IEEE-694 Mnemonic: CALLNZ reg,/abs
CALLNZ reg,/@abs
Binary Code: $7C-$7F

011111rr	ippaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: SP
Condition Code Setting: N/A

Description:
This three-byte subroutine branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address. Before replacing the current contents of the Instruction Address Register (IAR) with the effective address, the Stack Pointer (SP) is incremented by one and the address of the byte following the third byte of the instruction is pushed into the Return Address Stack (RAS).
If the specified register contains a zero value the next instruction is fetched from the location following the third byte of this instruction.
Indirect addressing may be specified.

Pseudocode:

```
BSNA,rn abs          ;if (rn != 0) gosub abs;                ;3,3
BSNA,rn *abs         ;if (rn != 0) gosub *(abs);           ;3+2,3
```

BSNR

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE ON REGISTER NON-ZERO, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: BSNR,r (*)a
CALM Mnemonic: CALL,regNE .+rel
CALL,regNE @.+rel
IEEE-694 Mnemonic: CALLNZ reg,$rel
CALLNZ reg,$@rel
Binary Code: $78-$7B

011110rr	iaaaaaaa
76543210	76543210

```
r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: SP
Condition Code Setting: N/A

Description:
This two-byte subroutine branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address. Before replacing the current contents of the Instruction Address Register (IAR) with the effective address, the Stack Pointer (SP) is incremented by one and the address of the byte following the second byte of the instruction is pushed into the Return Address Stack (RAS).
If the specified register contains a zero value the next instruction is fetched from the location following the second byte of this instruction.
Indirect addressing may be specified.

Pseudocode:

```
    BSNR,rn rel          ;if (rn != 0) gosub rel;                ;3,2
    BSNR,rn *rel         ;if (rn != 0) gosub *(rel);           ;3+2,2
```

BSTA

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE ON CONDITION TRUE, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BSTA,v (*)a
CALM Mnemonic:
CALL,EQ abs or CALL,AO abs
CALL,EQ @abs or CALL,AO @abs
CALL,GT abs
CALL,GT @abs
CALL,LT abs or CALL,NO abs
CALL,LT @abs or CALL,NO @abs
CALL abs
CALL @abs
IEEE-694 Mnemonic:
CALLEQ /abs
CALLEQ /@abs
CALLGT /abs
CALLGT /@abs
CALLLT /abs
CALLLT /@abs
CALL /abs
CALL /@abs
Binary Code: $3C-$3F

```
    001111vv    ippaaaaa    aaaaaaaa
```

```
    76543210    76543210    76543210

v: condition (2 bits)
i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: SP
Condition Code Setting: N/A

Synonyms:

```
    BSTA,un = BSA = Branch to Subroutine, Absolute
```

Description:

This three-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch only if the two-bit v field matches the current Condition Code (CC) field in the Program Status Word (PSW). If the fields match, the Stack Pointer (SP) is incremented by one and the current contents of the Instruction Address Register (IAR), which points to the byte following the third byte of this instruction, is pushed into the Return Address Stack (RAS). The effective address replaces the previous contents of the IAR.

If the v field and CC field do not match, the next instruction is fetched from the location following the third byte of this instruction and the Stack Pointer (SP) is unaffected.

Indirect addressing may be specified.

If the v field is set to $3, an unconditional subroutine branch is effected.

Pseudocode:

```
    BSTA,eq abs         ;if == gosub abs;                    ;3,3
    BSTA,gt abs         ;if > gosub abs;                     ;3,3
    BSTA,lt abs         ;if < gosub abs;                     ;3,3
    BSTA,un abs         ;gosub abs;                          ;3,3
    BSTA,eq *abs        ;if == gosub *(abs);                ;3+2,3
    BSTA,gt *abs        ;if > gosub *(abs);                 ;3+2,3
    BSTA,lt *abs        ;if < gosub *(abs);                 ;3+2,3
    BSTA,un *abs        ;gosub *(abs);                       ;5,3
```

BSTR

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE ON CONDITION TRUE, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: BSTR,v (*)a
CALM Mnemonic: CALL,EQ .+rel or CALL,AO .+rel
CALL,EQ @.+rel or CALL,AO @.+rel
CALL,GT .+rel
CALL,GT @.+rel

```
                                CALL,LT .+rel or CALL,NO .+rel
                                CALL,LT @.+rel or CALL,NO @.+rel
                                CALL    .+rel
                                CALL    @.+rel
CALM Mnemonic:                  CALLEQ  $rel
                                CALLEQ  $@rel
                                CALLGT  $rel
                                CALLGT  $@rel
                                CALLLT  $rel
                                CALLLT  $@rel
                                CALL    $rel
                                CALL    $@rel
Binary Code:                    $38-$3B

    001110vv    iaaaaaaa
    76543210    76543210

v: condition (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time:                 3 cycles (9 clock periods)
Processor Registers Affected:   SP
Condition Code Setting:         N/A

Synonyms:

    BSTR,un = BSR = Branch to Subroutine, Relative

Description:
  This two-byte conditional subroutine branch instruction causes the
processor to perform a subroutine branch only if the two-bit v field
matches the current Condition Code (CC) field in the Program Status Word
(PSW). If the fields match, the Stack Pointer (SP) is incremented by one
and the current contents of the Instruction Address Register (IAR), which
points to the byte following the second byte of this instruction, is
pushed into the Return Address Stack (RAS). The effective address replaces
the previous contents of the IAR.
  If the v field and CC field do not match, the next instruction is
fetched from the location following the second byte of this instruction
and the Stack Pointer (SP) is unaffected.
  Indirect addressing may be specified.
  If the v field is set to $3, an unconditional subroutine branch is
effected.

Pseudocode:

    BSTR,eq rel         ;if == gosub rel;                       ;3,2
    BSTR,gt rel         ;if > gosub rel;                        ;3,2
    BSTR,lt rel         ;if < gosub rel;                        ;3,2
    BSTR,un rel         ;gosub rel;                             ;3,2
    BSTR,eq *rel        ;if == gosub *(rel);                  ;3+2,2
    BSTR,gt *rel        ;if > gosub *(rel);                   ;3+2,2
    BSTR,lt *rel        ;if < gosub *(rel);                   ;3+2,2
    BSTR,un *rel        ;gosub *(rel);                          ;5,2
```

BSXA

Contents | < Browse | Browse >

BRANCH TO SUBROUTINE INDEXED, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BSXA,r3 (*)a,x
CALM Mnemonic: CALL (D)+abs
CALL (D)+@abs
IEEE-694 Mnemonic: CALL /abs(.3)
CALL /@abs(.3)
Binary Code: $BF

10111111	ippaaaaa	aaaaaaaa
76543210	76543210	76543210

i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: SP
Condition Code Setting: N/A

Description:
This three-byte branch instruction causes the processor to perform an unconditional subroutine branch. Indexing is required and register #3 must be specified as the index register because the entire first byte of this instruction is decoded by the processor.

Execution of this instruction causes the Stack Pointer (SP) to be incremented by one, the address of the byte following the third byte of this instruction is pushed into the Return Address Stack (RAS), and the effective address replaces the contents of the Instruction Address Register (IAR).

If indirect addressing is specified, the value in the index register is added to the indirect address to calculate the effective branch address.

Pseudocode:

```
BSXA    abs,r3      ;gosub abs + r3;                    ;3,3
BSXA    *abs,r3     ;gosub *(abs) + r3;                 ;5,3
```

BXA

Contents | < Browse | Browse >

BRANCH INDEXED, ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: BXA,r3 (*)a,x
CALM Mnemonic: JUMP (D)+abs

```
                                JUMP    (D)+@abs
IEEE-694 Mnemonic:              BR      /abs(.3)
                                BR      /@abs(.3)
Binary Code:                    $9F

    10011111    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210

i: indirect addressing flag (1 bit)
p: page (2 bits)
a: absolute address (13 bits)

Execution Time:                 3 cycles (9 clock periods)
Processor Registers Affected:   None
Condition Code Setting:         N/A
```

Description:

This three-byte branch instruction causes the processor to perform an unconditional branch. Indexing is required and register #3 must be specified as the index register because the entire first byte of this instruction is decoded by the processor. When executed, the content of the Instruction Address Register (IAR) is replaced by the effective address.

If indirect addressing is specified, the value in the index register is added to the indirect address to calculate the effective branch address.

Pseudocode:

```
    BXA     abs,r3      ;goto abs + r3;                     ;3,3
    BXA     *abs,r3     ;goto *(abs) + r3;                  ;5,3
```

COMA

Contents | < Browse | Browse >

COMPARE ABSOLUTE

```
Addressing Mode:                Absolute
Size:                           3 bytes
Signetics Mnemonic:             COMA,r (*)a(,X)
CALM Mnemonic:                  COMP    reg,abs
                                COMP    reg,@abs
                                COMP    A,(reg)+abs
                                COMP    A,(reg)+@abs
                                COMP    A,(+reg)+abs
                                COMP    A,(+reg)+@abs
                                COMP    A,(-reg)+abs
                                COMP    A,(-reg)+@abs
IEEE-694 Mnemonic:              CMP     reg,/abs
                                CMP     reg,/@abs
                                CMP     A,(reg)+/abs
                                CMP     A,(reg)+/@abs
                                CMP     A,(+reg)+/abs
                                CMP     A,(+reg)+/@abs
                                CMP     A,(-reg)+/abs
```

CMP A,(-reg)+/@abs

Binary Code: $EC-$EF

111011rr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

	CC1	CC0
Register r greater than memory byte	0	1
Register r equal to memory byte	0	0
Register r less than memory byte	1	0

Description:
This three-byte instruction causes the contents of register r to be compared to the contents of the memory byte pointed to by the effective address. The comparison will be performed in either "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word (PSW).

Where COM=1 (logical mode), the values will be treated as 8-bit, positive binary numbers; when COM=0 (arithmetic mode), the values will be treated as 8-bit, two's complement numbers.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

The execution of this instruction *only* causes the Condition Code (CC) to be set as in the preceding table.

Pseudocode:

```
COMA,rn abs          ;compare rn against *(abs);              ;4,3
COMA,r0 abs,rn       ;compare r0 against *(abs + rn);         ;4,3
COMA,r0 abs,rn+      ;compare r0 against *(abs + ++rn);       ;4,3
                     ;or, rn++; compare r0 against *(abs + rn);
COMA,r0 abs,rn-      ;compare r0 against *(abs + --rn);       ;4,3
                     ;or, rn--; compare r0 against *(abs + rn);
COMA,rn *abs         ;compare rn against *(*(abs));           ;6,3
COMA,r0 *abs,rn      ;compare r0 against *(*(abs) + rn);      ;6,3
COMA,r0 *abs,rn+     ;compare r0 against *(*(abs) + ++rn);    ;6,3
                     ;or, rn++; compare r0 against *(*(abs) + rn);
COMA,r0 *abs,rn-     ;compare r0 against *(*(abs) + --rn);    ;6,3
                     ;or, rn--; compare r0 against *(*(abs) + rn);
```

COMI

Contents | < Browse | Browse >

COMPARE IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: COMI,r v
CALM Mnemonic: COMP reg,#imm
IEEE-694 Mnemonic: CMP reg,#imm
Binary Code: $E4-$E7

111001rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

	CC1	CC0
Register r greater than v	0	1
Register r equal to v	0	0
Register r less than v	1	0

Description:
This two-byte instruction causes the contents of register r to be compared to the contents of the second byte of this instruction. The comparison will be performed in either "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word (PSW).
Where COM=1 (logical mode), the values will be treated as 8-bit, positive binary numbers; when COM=0 (arithmetic mode), the values will be treated as 8-bit, two's complement numbers.
The execution of this instruction *only* causes the Condition Code (CC) to be set as in the preceding table.

Pseudocode:

```
COMI,rn imm          ;compare rn against imm;                ;2,2
```

COMR

Contents | < Browse | Browse >

COMPARE RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: COMR,r (*)a
CALM Mnemonic:
COMP reg,.+rel
COMP reg,@.+rel
COMP A,(reg)+.+rel
COMP A,(reg)+@.+rel
COMP A,(+reg)+.+rel
COMP A,(+reg)+@·+rel
COMP A,(-reg)+.+rel
COMP A,(-reg)+@.+rel

IEEE-694 Mnemonic:

```
CMP    reg,$rel
CMP    reg,$@rel
CMP    .0,(reg)+$rel
CMP    .0,(reg)+$@rel
CMP    .0,(+reg)+$rel
CMP    .0,(+reg)+$@rel
CMP    .0,(-reg)+$rel
CMP    .0,(-reg)+$@rel
```

Binary Code: $E8-$EB

111010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

	CC1	CC0
Register r greater than memory byte	0	1
Register r equal to memory byte	0	0
Register r less than memory byte	1	0

Description:

This two-byte instruction causes the contents of register r to be compared to the contents of the memory byte pointed to by the effective address. The comparison will be performed in either "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word (PSW).

Where COM=1 (logical mode), the values will be treated as 8-bit, positive binary numbers; when COM=0 (arithmetic mode), the values will be treated as 8-bit, two's complement numbers.

The execution of this instruction *only* causes the Condition Code (CC) to be set as in the preceding table.

Pseudocode:

```
COMR,rn abs          ;compare rn against *(abs);              ;3,2
COMR,rn *abs         ;compare rn against *(*(abs));           ;5,2
```

COMZ

Contents | < Browse | Browse >

COMPARE TO REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: COMZ r
CALM Mnemonic: COMP A,reg
IEEE-694 Mnemonic: CMP .0,reg
Binary Code: $E0-$E3

```
    111000rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

	CC1	CC0
Register zero greater than register r	0	1
Register zero equal to register r	0	0
Register zero less than register r	1	0

Description:
This one-byte instruction causes the contents of the specified register, r, to be compared to the contents of register zero. The comparison will be performed in either "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word (PSW).
Where COM=1 (logical mode), the values will be treated as 8-bit, positive binary numbers; when COM=0 (arithmetic mode), the values will be treated as 8-bit, two's complement numbers.
The execution of this instruction *only* causes the Condition Code (CC) to be set as in the preceding table.

Pseudocode: (COMZ r0)

```
    COMZ    r0          ;CC = EQ;                               ;2,1
```

Pseudocode: (all)

```
    COMZ    rn          ;compare r0 against rn;                 ;2,1
```

CPSL

Contents | < Browse | Browse >

CLEAR PROGRAM STATUS LOWER, SELECTIVE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: CPSL v
CALM Mnemonic: BIC L,#imm
CLR CARRY or CLRC
CLR LOGICOMP
CLR OVERFLOW or CLRV
CLR WITHCARRY
CLR BANK or CLR BANK1
IEEE-694 Mnemonic: AND L,#~imm
CLRC
CLRV
Binary Code: $75

```
    01110101    vvvvvvvv
```

```
    76543210    76543210
```

v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC, IDC, RS, WC, OVF, COM, C
Condition Code Setting:
The CC bits may be cleared by the execution of this instruction.

Description:
This two-byte instruction causes individual bits in the Lower Program Status Byte to be selectively cleared. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is cleared to zero. Any bits in the status byte which are not selected are not modified.

Pseudocode:

```
    CPSL    imm          ;PSL &= ~(imm);                    ;3,2
```

CPSU

Contents | < Browse | Browse >

CLEAR PROGRAM STATUS UPPER, SELECTIVE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: CPSU v
CALM Mnemonic: BIC U,#imm
CLR STACK
CLR IOF or ION
CLR OUTPUT
CLR INPUT
IEEE-694 Mnemonic: AND .U,#~imm
EI
Binary Code: $74

```
    01110100    vvvvvvvv
    76543210    76543210
```

v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: F, II, SP
Condition Code Setting: N/A

Description:
This two-byte instruction causes individual bits in the Upper Program Status Byte to be selectively cleared. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is cleared to zero. Any bits in the status byte which are not selected are not modified.

Pseudocode:

```
    CPSU    imm          ;PSU &= ~(imm & %01100111);              ;3,2
```

DAR

Contents | < Browse | Browse >

DECIMAL ADJUST REGISTER

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: DAR,r
CALM Mnemonic: DA reg
IEEE-694 Mnemonic: ADJ reg
Binary Code: $94-$97

```
    100101rr
    76543210
```

r: register (2 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:
The Condition Code (CC) is set to a meaningless value.

Description:
This one-byte instruction conditionally adds a decimal ten (two's complement negative six in a four-bit binary number system) to either the high order 4 bits and/or the low order 4 bits of the specified register r.

The truth table below indicates the logical operation performed. The operation proceeds based on the contents of the Carry (C) and Interdigit Carry (IDC) bits in the Program Status Word. The C and IDC remain unchanged by the execution of this instruction.

This instruction allows BCD sign magnitude arithmetic to be performed on packed digits by the following procedure.

BCD Addition:
1. add $66 to augend
2. perform addition of addend and augend
3. perform DAR instruction

BCD Subtraction:
1. perform subtraction (2's complement of subtrahend is added to the minuend)
2. perform DAR instruction

Since this operation is on sign-magnitude numbers, it is necessary to establish the sign of the result prior to executing in order to properly control the definition of the subtrahend and minuend.

Carry	Interdigit Carry	Added to Register r
0	0	$AA
0	1	$A0
1	1	$00

1	0	$0A

Pseudocode:

```
    DAR,rn              ;rn = BCD(rn);                          ;3,1
```

EORA

Contents | < Browse | Browse >

EXCLUSIVE OR ABSOLUTE

Addressing Mode:	Absolute
Size:	3 bytes
Signetics Mnemonic:	EORA,r (*)a(,X)
CALM Mnemonic:	XOR reg,abs
	XOR reg,@abs
	XOR A,(reg)+abs
	XOR A,(reg)+@abs
	XOR A,(+reg)+abs
	XOR A,(+reg)+@abs
	XOR A,(-reg)+abs
	XOR A,(-reg)+@abs
IEEE-964 Mnemonic:	XOR reg,/abs
	XOR reg,/@abs
	XOR .0,/abs(reg)
	XOR .0,/@abs(reg)
	XOR .0,/abs(+reg)
	XOR .0,/@abs(+reg)
	XOR .0,/abs(-reg)
	XOR .0,/@abs(-reg)
Binary Code:	$2C-$2F

001011rr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time:	4 cycles (12 clock periods)
Processor Registers Affected:	CC
Condition Code Setting:	

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This three-byte instruction causes the contents of register r to be logically Exclusive ORed with the contents of the memory byte pointed to by the effective address. The result of the operation replaces the previous contents of register r.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

Pseudocode:

```
EORA,rn abs          ;rn ^= *(abs);                          ;4,3
EORA,r0 abs,rn       ;r0 ^= *(abs + rn);                     ;4,3
EORA,r0 abs,rn+      ;r0 ^= *(abs + ++rn);                   ;4,3
                     ;or, rn++; r0 ^= *(abs + rn);
EORA,r0 abs,rn-      ;r0 ^= *(abs + --rn);                   ;4,3
                     ;or, rn--; r0 ^= *(abs + rn);
EORA,rn *abs         ;rn ^= *(*(abs));                       ;6,3
EORA,r0 *abs,rn      ;r0 ^= *(*(abs) + rn);                  ;6,3
EORA,r0 *abs,rn+     ;r0 ^= *(*(abs) + ++rn);                ;6,3
                     ;or, rn++; r0 ^= *(*(abs) + rn);
EORA,r0 *abs,rn-     ;r0 ^= *(*(abs) + --rn);                ;6,3
                     ;or, rn--; r0 ^= *(*(abs) + rn);
```

EORI

Contents | < Browse | Browse >

EXCLUSIVE OR IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: EORI,r v
CALM Mnemonic:
```
XOR    reg,#imm
NOT    reg
```
IEEE-964 Mnemonic:
```
XOR    reg,#imm
NOT    reg
```
Binary Code: $24-$27

011001rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1

Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r to be logically Exclusive ORed with the contents of the second byte of this instruction. The result of this operation replaces the contents of register r.
The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Pseudocode:

```
EORI,rn imm          ;rn ^= imm;                              ;2,2
```

EORR

Contents | < Browse | Browse >

EXCLUSIVE OR RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: EORR,r (*)a
CALM Mnemonic: XOR reg,.+rel
XOR reg,@.+rel
IEEE-964 Mnemonic: XOR reg,$rel
XOR reg,$@rel
Binary Code: $28-$2B

001010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r to be logically Exclusive ORed with the contents of the memory byte pointed to

by the effective address. The result of this operation replaces the previous contents of register r.

Indirect indexing may be specified.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Pseudocode:

```
EORR,rn abs          ;rn ^= *(abs);                          ;3,2
EORR,rn *abs         ;rn ^= *(*(abs));                       ;5,2
```

EORZ

Contents | < Browse | Browse >

EXCLUSIVE OR TO REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: EORZ r
CALM Mnemonic: XOR A,reg
CLR A
IEEE-964 Mnemonic: XOR .0,reg
CLR .0
Binary Code: $20-$23

001000rr
76543210

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This one-byte instruction causes the contents of the specified register, r, to be logically Exclusive ORed with the contents of register zero. The result of the operation replaces the contents of register zero. The contents of register r remain unchanged.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result

```
        0           0           0
        0           1           1
        1           1           0
        1           0           1
```

Pseudocode:

```
    EORZ    rn          ;r0 ^= rn;                          ;2,1
```

HALT

Contents | < Browse | Browse >

HALT, ENTER WAIT STATE

Addressing Mode: Implicit
Size: 1 byte
Signetics Mnemonic: HALT
CALM Mnemonic: WAIT
IEEE-964 Mnemonic: HALT or WAIT
Binary Code: $40

```
    01000000
    76543210
```

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
  This one-byte instruction causes the processor to stop executing instructions and enter the WAIT state. The RUN/WAIT line is set to the WAIT state.
  The only way to enter the RUN state after a HALT has been executed is to reset the 2650 or to interrupt the processor.

Pseudocode:

```
    HALT                ;for (;;);                          ;2,1
```

IORA

Contents | < Browse | Browse >

INCLUSIVE OR ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: IORA,r (*)a(,X)
CALM Mnemonic:
```
OR      reg,abs
OR      reg,@abs
OR      A,(reg)+abs
```

```
                                OR      A,(reg)+@abs
                                OR      A,(+reg)+abs
                                OR      A,(+reg)+@abs
                                OR      A,(-reg)+abs
                                OR      A,(-reg)+@abs
IEEE-964 Mnemonic:              OR      reg,/abs
                                OR      reg,/@abs
                                OR      .0,/abs(reg)
                                OR      .0,/@abs(reg)
                                OR      .0,/abs(+reg)
                                OR      .0,/@abs(+reg)
                                OR      .0,/abs(-reg)
                                OR      .0,/@abs(-reg)
Binary Code:                    $6C-$6F
```

011011rr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This three-byte instruction causes the contents of register r to be logically Inclusive ORed with the contents of the memory byte pointed to by the effective address. The result of the operation replaces the previous contents of register r.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

Pseudocode:

```
IORA,rn abs         ;rn |= *(abs);                          ;4,3
IORA,r0 abs,rn      ;r0 |= *(abs + rn);                     ;4,3
IORA,r0 abs,rn+     ;r0 |= *(abs + ++rn);                   ;4,3
                    ;or, rn++; r0 |= *(abs + rn);
IORA,r0 abs,rn-     ;r0 |= *(abs + --rn);                   ;4,3
                    ;or, rn--; r0 |= *(abs + rn);
```

```
    IORA,rn *abs          ;rn |= *(*(abs));                          ;6,3
    IORA,r0 *abs,rn       ;r0 |= *(*(abs) + rn);                     ;6,3
    IORA,r0 *abs,rn+      ;r0 |= *(*(abs) + ++rn);                   ;6,3
                          ;or, rn++; r0 |= *(*(abs) + rn);
    IORA,r0 *abs,rn-      ;r0 |= *(*(abs) + --rn);                   ;6,3
                          ;or, rn--; r0 |= *(*(abs) + rn);
```

IORI

Contents | < Browse | Browse >

INCLUSIVE OR IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: IORI,r v
CALM Mnemonic: OR reg,#imm
IEEE-694 Mnemonic: OR reg,#imm
Binary Code: $64-$67

011001rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r to be logically Inclusive ORed with the contents of the second byte of this instruction. The result of this operation replaces the contents of register r.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Pseudocode:

```
    IORI,rn imm           ;rn |= imm;                                 ;2,2
```

IORR

Contents | < Browse | Browse >

INCLUSIVE OR RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: IORR,r (*)a
CALM Mnemonic:
```
OR      reg,.+rel
OR      reg,@.+rel
```
IEEE-694 Mnemonic:
```
OR      reg,$rel
OR      reg,$@rel
```
Binary Code: $68-$6B

011010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction causes the contents of register r to be logically Inclusive ORed with the contents of the memory byte pointed to by the effective address. The result of this operation replaces the previous contents of register r.

Indirect indexing may be specified.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Pseudocode:

```
IORR,rn rel          ;rn |= *(rel);                          ;3,2
IORR,rn *rel         ;rn |= *(*(rel));                      ;5,2
```

IORZ

Contents | < Browse | Browse >

INCLUSIVE OR TO REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: IORZ r
CALM Mnemonic: OR A,reg
IEEE-694 Mnemonic: OR .0,reg
Binary Code: $60-$63

```
011000rr
76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the contents of the specified register, r, to be logically Inclusive ORed with the contents of register zero. The result of the operation replaces the contents of register zero. The contents of register r remain unchanged.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Pseudocode:

```
IORZ    rn          ;r0 |= rn;                          ;2,1
```

LDPL

Contents | < Browse | Browse >

LOAD PROGRAM STATUS, LOWER from memory (2650-B only)

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: LDPL (*)a
CALM Mnemonic: LOAD L,abs
LOAD L,@abs
IEEE-694 Mnemonic: LD L,/abs
LD L,/@abs

Binary Code: $10

```
    00010000    ippaaaaa    aaaaaaaa
    76543210    76543210    76543210
     Byte 0      Byte 1      Byte 2
```

i: indirect addressing flag
p: page
a: address

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: CC, IDC, RS, WC, OVF, COM, C
Condition Code Setting:
The CC will take on the value in bits #7 and #6 of the byte pointed to by the effective address.

Description:
This three-byte instruction causes the current contents of the Lower Program Status Byte to be replaced with the contents of the byte of memory pointed to by the effective address.
See Program Status Word description for bit assignments.

Pseudocode:

```
    LDPL    abs         ;PSL = *(abs);                  ;4,3
    LDPL    *abs        ;PSL = *(*(abs));               ;4,3
```

LODA

Contents | < Browse | Browse >

LOAD ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: LODA,r (*)a(,X)
CALM Mnemonic:

```
LOAD   reg,abs
LOAD   reg,@abs
LOAD   A,(reg)+abs
LOAD   A,(reg)+@abs
LOAD   A,(+reg)+abs
LOAD   A,(+reg)+@abs
LOAD   A,(-reg)+abs
LOAD   A,(-reg)+@abs
```

IEEE-964 Mnemonic:

```
LD     reg,/abs
LD     reg,/@abs
LD     .0,/abs(reg)
LD     .0,/@abs(reg)
LD     .0,/abs(+reg)
LD     .0,/@abs(+reg)
LD     .0,/abs(-reg)
LD     .0,/@abs(-reg)
```

Binary Code: $0C-$0F

```
    000011rr    iccaaaaa    aaaaaaaa
```

```
    76543210    76543210    76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)
```

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This three-byte instruction transfers a byte of data from memory into the specified register, r. The data byte is found at the effective address. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero. The previous contents of register r are lost.
Indirect addressing and/or indexing may be specified.

Pseudocode:

```
    LODA,rn abs          ;rn = *(abs);                          ;4,3
    LODA,r0 abs,rn       ;r0 = *(abs + rn);                     ;4,3
    LODA,r0 abs,rn+      ;r0 = *(abs + ++rn);                   ;4,3
                         ;or, rn++; r0 = *(abs + rn);
    LODA,r0 abs,rn-      ;r0 = *(abs + --rn);                   ;4,3
                         ;or, rn--; r0 = *(abs + rn);
    LODA,rn *abs         ;rn = *(*(abs));                       ;6,3
    LODA,r0 *abs,rn      ;r0 = *(*(abs) + rn);                  ;6,3
    LODA,r0 *abs,rn+     ;r0 = *(*(abs) + ++rn);                ;6,3
                         ;or, rn++; r0 = *(*(abs) + rn);
    LODA,r0 *abs,rn-     ;r0 = *(*(abs) + --rn);                ;6,3
                         ;or, rn--; r0 = *(*(abs) + rn);
```

LODI

Contents | < Browse | Browse >

LOAD IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: LODI,r v
CALM Mnemonic: LOAD reg,#imm
IEEE-694 Mnemonic: LD reg,#imm
Binary Code: $04-$07

```
    000001rr    vvvvvvvv
    76543210    76543210
```

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction transfers the second byte of the instruction, v, into the specified register, r. The previous contents of r are lost.

Pseudocode:

```
LODI,rn imm          ;rn = imm;                              ;2,2
```

LODR

Contents | < Browse | Browse >

LOAD RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: LODR,r (*)a
CALM Mnemonic: LOAD reg,.+rel
LOAD reg,@.+rel
IEEE-694 Mnemonic: LD reg,$rel
LD reg,$@rel
Binary Code: $08-$0B

000010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte instruction transfers a byte of data from memory into the specified register, r. The data byte is found at the effective address formed by the addition of the a field and the address of the byte

following this instruction. The previous contents of register r are lost. Indirect addressing may be specified.

Pseudocode:

```
    LODR,rn rel          ;rn = *(rel);                          ;3,2
    LODR,rn *rel         ;rn = *(*(rel));                       ;5,2
```

LODZ

Contents | < Browse | Browse >

LOAD REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: LODZ r
CALM Mnemonic: LOAD A,reg
IEEE-694 Mnemonic: LD .0,reg
Binary Code: $01-$03

```
    000000rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction transfers the contents of the specified register, r, into register zero. The previous contents of register zero are lost. The contents of register r remain unchanged.
When the specified register, r, equals 0, the operation code is changed to $60 by the assembler. The instruction, %00000000, yields indeterminate results.

Pseudocode:

```
    LODZ    rn           ;r0 = rn;                              ;2,1
```

LPSL

Contents | < Browse | Browse >

LOAD PROGRAM STATUS, LOWER from register zero

Addressing Mode: Immediate
Size: 1 byte
Signetics Mnemonic: LPSL
CALM Mnemonic: LOAD L,A
IEEE-694 Mnemonic: MOV .L,.0
Binary Code: $93

```
    10010011
    76543210
```

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC, IDC, RS, WC, OVF, COM, C
Condition Code Setting:
The CC will take on the value in bits #7 and #6 of register zero.

Description:
This one-byte instruction causes the current contents of the Lower Program Status Byte to be replaced with the contents of register zero.
See Program Status Word description for bit assignments.

Pseudocode:

```
    LPSL                ;PSL = r0;                          ;2,1
```

LPSU

Contents | < Browse | Browse >

LOAD PROGRAM STATUS, UPPER from register zero

Addressing Mode: Immediate
Size: 1 byte
Signetics Mnemonic: LPSU
CALM Mnemonic: LOAD U,A
IEEE-694 Mnemonic: MOV .U,.0
Binary Code: $92

```
    10010010
    76543210
```

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: F, II, SP
Condition Code Setting: N/A

Description:
This one-byte instruction causes the current contents of the Upper Program Status Byte to be replaced with the contents of register zero.
See Program Status Word description for bit assignments. Bits #4 and #3 of the PSU are unassigned and will always be regarded as containing zeroes.

Pseudocode:

```
    LPSU                ;PSU = (r0 & %01100111);            ;2,1
```

NOP

Contents | < Browse | Browse >

NO OPERATION

Addressing Mode: Implicit
Size: 1 byte
Signetics Mnemonic: NOP
CALM Mnemonic: NOP
IEEE-694 Mnemonic: NOP
Binary Code: $C0

```
    11000000
    76543210
```

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This one-byte instruction causes the processor to take no action upon decoding it. No registers are changed, but fetching and executing a NOP instruction requires two processor cycles.

Pseudocode:

```
    NOP                 ;;                                  ;2,1
```

PPSL

Contents | < Browse | Browse >

PRESET PROGRAM STATUS LOWER, SELECTIVE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: PPSL v
CALM Mnemonic: OR L,#imm
SET CARRY or SETC
SET LOGICOMP
SET OVERFLOW or SETV
SET WITHCARRY
SET BANK or SET BANK1
IEEE-694 Mnemonic: OR .L,#imm
SETC
SETV
Binary Code: $77

```
    01110111    vvvvvvvv
    76543210    76543210
```

v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC, IDC, RS, WC, OVF, COM, C
Condition Code Setting:
The CC bits may be set by the execution of this instruction.

Description:
This two-byte instruction causes individual bits in the Lower Program Status Byte to be selectively set to binary one. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is set to binary one. Any bits in the status byte which are not selected are not modified.

Pseudocode:

```
    PPSL    imm          ;PSL |= imm;                          ;3,2
```

PPSU

Contents | < Browse | Browse >

PRESET PROGRAM STATUS UPPER, SELECTIVE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: PPSU v
CALM Mnemonic: OR U,#n
SET STACK
SET IOF or IOF
SET OUTPUT
SET INPUT
IEEE-694 Mnemonic: OR .U,#imm
DI
Binary Code: $76

01110110	vvvvvvvv
76543210	76543210

v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: F, II, SP
Condition Code Setting: N/A

Description:
This two-byte instruction causes individual bits in the Upper Program Status Byte to be selectively set to binary one. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is set to binary one. Any bits in the status byte which are not selected are not modified.

Pseudocode:

```
    PPSU    imm          ;PSU |= (imm & %01100111);              ;3,2
```

REDC

Contents | < Browse | Browse >

READ CONTROL

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: REDC,r
CALM Mnemonic: LOAD reg,$CTRL
IEEE-694 Mnemonic: IN reg,CTRL
Binary Code: $30-$33

```
    001100rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte input instruction causes a byte of data to be transferred from the data bus into register r.

Signals on the data bus are considered to be true signals, ie. a high level will be set into the register as a one.

When executing this instruction, the processor raises the Operation Request (OPREQ) line, simultaneously switching the M/IO line to IO, the R/W line to R (Read), the D/C line to C (Control), and the E/NE line to NE (Non-extended).

Pseudocode:

```
    REDC,rn          ;rn = *(CONTROLBUS);                    ;3,2
```

REDD

Contents | < Browse | Browse >

READ DATA

Addressing Mode: Register

Size: 1 byte
Signetics Mnemonic: REDD,r
CALM Mnemonic: LOAD reg,$DATA
IEEE-694 Mnemonic: IN reg,DATA
Binary Code: $70-$73

011100rr
76543210

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte input instruction causes a byte of data to be transferred from the data bus into register r. Signals on the data bus are considered to be true signals, ie. a high level will be set into the register as a one.
When executing this instruction, the processor raises the Operation Request (OPREQ) line, simultaneously switching the M/IO line to IO and the R/W line to R (Read). Also, during the OPREQ signal, the D/C line switches to D (Data), and the E/NE line to NE (Non-extended).

Pseudocode:

```
REDD,rn             ;rn = *(DATABUS);                    ;3,2
```

REDE

Contents | < Browse | Browse >

READ EXTENDED

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: REDE,r v
CALM Mnemonic: LOAD reg,$port
IEEE-694 Mnemonic: IN reg,port
Binary Code: $54-$57

010101rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC

Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This two-byte input instruction causes a byte of data to be transferred from the data bus into register r. During the execution of this instruction, the content of the second byte of this instruction is made available on the address bus. Signals on the data bus are true signals, ie. a high levels is interpreted as a one.
During execution, the processor raises the Operation Request (OPREQ) line, simultaneously placing the contents of the second byte of the instruction on the address bus. During the OPREQ signal, the M/IO line is switched to IO, the R/W line to R (Read), the E/NE line to E (Extended).

Pseudocode:

```
REDE,rn imm          ;*(ADDRESSBUS) = imm; rn = *(DATABUS);   ;3,2
```

RETC

Contents | < Browse | Browse >

RETURN FROM SUBROUTINE, CONDITIONAL

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: RETC,v
CALM Mnemonic: RET,EQ or RET,AO
RET,GT
RET,LT or RET,NO
RET
IEEE-694 Mnemonic: RETEQ
RETGT
RETLT
RET
Binary Code: $14-$17

```
000101vv
76543210
```

v: condition (2 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: SP
Condition Code Setting: N/A

Synonyms:

```
RETC,un = RET = Return from Subroutine
```

Description:

This one-byte instruction is used by a subroutine to conditionally effect a return of control to the program which last issued a subroutine branch instruction.

If the two-bit v field in the instruction matches the Condition Code (CC) field in the Program Status Word, the following action is taken: The address contained in the top of the Return Address Stack (RAS) replaces the previous contents of the Instruction Address Register (IAR), and the Stack Pointer (SP) is decremented by one.

If the v field does not match CC, the return is not effected and the next instruction to be executed is taken from the location following this instruction.

If v is specified as $3, the return is executed unconditionally.

Pseudocode:

```
RETC,eq             ;if == return;                    ;3,1
RETC,gt             ;if > return;                     ;3,1
RETC,lt             ;if < return;                     ;3,1
RETC,un             ;return;                          ;3,1
```

RETE

Contents | < Browse | Browse >

RETURN FROM SUBROUTINE AND ENABLE INTERRUPT, CONDITIONAL

Addressing Mode:	Implicit
Size:	1 byte
Signetics Mnemonic:	RETE,v
CALM Mnemonic:	RETION,EQ or RETION,AO RETION,GT RETION,LT or RETION,NO RETION
IEEE-694 Mnemonic:	RETIEQ RETIGT RETILT RETI
Binary Code:	$34-$37

```
001101vv
76543210
```

v: condition (2 bits)

Execution Time:	3 cycles (9 clock periods)
Processor Registers Affected:	SP, II
Condition Code Setting:	N/A

Description:

This one-byte instruction is used by a subroutine to conditionally effect a return of control to the program which last issued a subroutine branch instruction. Additionally, if the return is effected, the Interrupt Inhibit (II) bit in the Program Status Word is cleared to zero, thus enabling interrupts. This instruction is mainly intended to be used by an interrupt handling routine because receipt of an interrupt causes a

subroutine branch to be effected and the Interrupt Inhibit (II) bit to be set to 1. The interrupt handling routine must be able to return and enable simultaneously so that the interrupt routine cannot be interrupted unless that is specifically desired.

If the two-bit v field in the instruction matched the Condition Code (CC) field in the Program Status Word (PSW), the following action is taken: The address contained in the top of the Return Address Stack (RAS) replaces the previous contents of the Instruction Address Register (IAR), the Stack Pointer (SP) is decremented by one and the II bit is cleared to zero.

If the v field does not match CC, the return is not effected and the next instruction to be executed is taken from the location following this instruction.

If v is specified as $3, the return is executed unconditionally.

Pseudocode:

```
RETE,eq             ;if == then { PSU &= ~PSU_II; return; }  ;3,1
RETE,gt             ;if > then { PSU &= ~PSU_II; return; }   ;3,1
RETE,lt             ;if < then { PSU &= ~PSU_II; return; }   ;3,1
RETE,un             ;PSU &= ~PSU_II; return;                 ;3,1
```

RRL

Contents | < Browse | Browse >

ROTATE REGISTER LEFT

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: RRL,r
CALM Mnemonic: RL|RLC|SL|ASL reg
IEEE-694 Mnemonic: ROL|ROLC|SHL|SHLA reg
Binary Code: $D0-$D3

```
110100rr
76543210
```

r: register (2 bits)

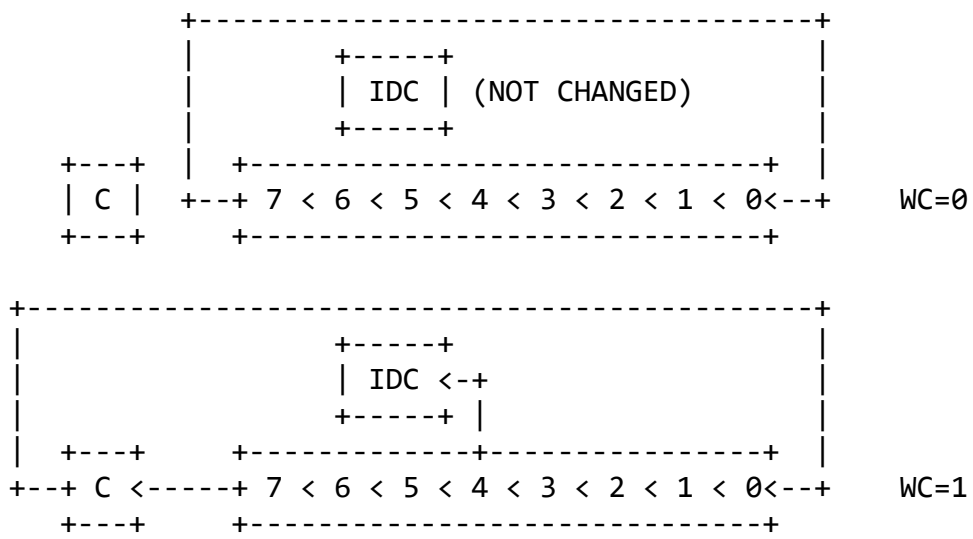
Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This one-byte instruction causes the contents of the specified register, r, to be shifted left one bit. If the WC bit in the Program Status Word (PSW) is set to zero, bit #7 of register r flows into bit #0; if WC=1, then bit #7 flows into the Carry (C) bit and the Carry (C) bit flows into bit #0.

Register bit #4 flows into the IDC if WC=1.



Note:
Whenever a rotate causes bit #7 of the specified register to change polarity, the OVF bit is set in the PSL.

Pseudocode:

```
RRL,rn              ;rn <<= 1;                          ;2,1
```

RRR

Contents | < Browse | Browse >

ROTATE REGISTER RIGHT

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: RRR,r
CALM Mnemonic: RR|RRC|SR reg
IEEE-694 Mnemonic: ROR|RORC|SHR reg
Binary Code: $50-$53

010100rr
76543210

r: register (2 bits)

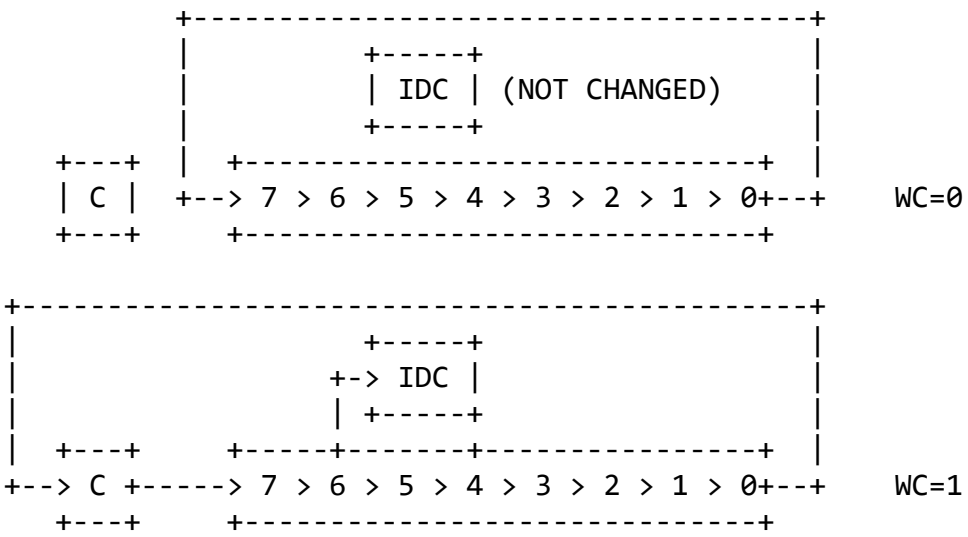
Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the contents of the specified register,

r, to be shifted right one bit. If the WC bit in the Program Status Word (PSW) is set to zero, bit #0 of register r flows into bit #7; if WC=1, then bit #0 flows into the Carry (C) bit and the Carry (C) bit flows into bit #7.

Register bit #6 flows into the IDC if WC=1.



Note:

Whenever a rotate causes bit #7 of the specified register to change polarity, the OVF bit is set in the PSL.

Pseudocode:

```
RRR,rn              ;rn >>= 1;                              ;2,1
```

SPSL

Contents | < Browse | Browse >

STORE PROGRAM STATUS, LOWER to register zero

Addressing Mode: Immediate
Size: 1 byte
Signetics Mnemonic: SPSL
CALM Mnemonic: LOAD A,L
IEEE-694 Mnemonic: MOV .0,.L
Binary Code: $13

00010011
76543210

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the current contents of the Lower Program Status Byte to be transferred into register zero.
See Program Status Word description for bit assignments.

Pseudocode:

```
SPSL                ;r0 = PSL;                              ;2,1
```

SPSU

Contents | < Browse | Browse >

STORE PROGRAM STATUS, UPPER to register zero

Addressing Mode: Immediate
Size: 1 byte
Signetics Mnemonic: SPSU
CALM Mnemonic: LOAD A,U
IEEE-694 Mnemonic: MOV .0,.U
Binary Code: $12

```
00010010
76543210
```

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the current contents of the Upper Program Status Byte to be transferred into register zero.
See Program Status Word description for bit assignments. Bits #4 and #3 which are unassigned will always be stored as zeroes.

Pseudocode:

```
SPSU                ;r0 = PSU;                              ;2,1
```

STPL

Contents | < Browse | Browse >

STORE PROGRAM STATUS, LOWER to memory (2650-B only)

Addressing Mode: Absolute
Size: 3 bytes

Signetics Mnemonic: STPL (*)a
CALM Mnemonic: LOAD abs,L
LOAD @abs,L
IEEE-694 Mnemonic: ST .L,/abs
ST .L,/@abs
Binary Code: $11

00010001	ippaaaaa	aaaaaaaa
76543210	76543210	76543210
Byte 0	Byte 1	Byte 2

i: indirect addressing flag
p: page
a: address

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This three-byte instruction causes the current contents of the Lower Program Status Byte to be transferred into the byte of memory pointed to by the effective address.
See Program Status Word description for bit assignments.

Pseudocode:

```
STPL    abs         ;*(abs) = PSL;                          ;4,3
STPL    *abs        ;*(*(abs)) = PSL;                       ;4,3
```

STRA

Contents | < Browse | Browse >

STORE ABSOLUTE

Addressing Mode: Absolute
Size: 3 bytes
Signetics Mnemonic: STRA,r (*)a(,X)
CALM Mnemonic: LOAD abs,reg
LOAD @abs,reg
LOAD (reg)+abs,A
LOAD (reg)+@abs,A
LOAD (+reg)+abs,A
LOAD (+reg)+@abs,A
LOAD (-reg)+abs,A
LOAD (-reg)+@abs,A
IEEE-694 Mnemonic: ST reg,/abs
ST reg,/@abs
ST reg,/abs(reg)
ST reg,/@abs(reg)
ST reg,/abs(+reg)
ST reg,/@abs(+reg)
ST reg,/abs(-reg)
ST reg,/@abs(-reg)

Binary Code: $CC-$CF

```
    110011rr    iccaaaaa    aaaaaaaa
    76543210    76543210    76543210
```

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time: 4 cycles (12 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This three-byte instruction transfers a byte of data from the specified register, r, into the byte of memory pointed to by the effective address. The contents of register r remain unchanged and the contents of the memory byte are replaced.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

Pseudocode:

```
    STRA,rn abs         ;*(abs) = rn;                          ;4,3
    STRA,r0 abs,rn      ;*(abs + rn) = r0;                     ;4,3
    STRA,r0 abs,rn+     ;*(abs + ++rn) = r0;                   ;4,3
                        ;or, rn++; *(abs + rn) = r0;
    STRA,r0 abs,rn-     ;*(abs + --rn) = r0;                   ;4,3
                        ;or, rn--; *(abs + rn) = r0;
    STRA,rn *abs        ;*(*(abs)) = rn;                       ;6,3
    STRA,r0 *abs,rn     ;*(*(abs) + rn) = r0;                  ;6,3
    STRA,r0 *abs,rn+    ;*(*(abs) + ++rn) = r0;                ;6,3
                        ;or, rn++; *(*(abs) + rn) = r0;
    STRA,r0 *abs,rn-    ;*(*(abs) + --rn) = r0;                ;6,3
                        ;or, rn--; *(*(abs) + rn) = r0;
```

STRR

Contents | < Browse | Browse >

STORE RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: STRR,r (*)a
CALM Mnemonic:
```
LOAD   .+rel,reg
LOAD   @.+rel,reg
```
IEEE-694 Mnemonic:
```
ST     reg,$rel
ST     reg,$@rel
```
Binary Code: $C8-$CB

```
    110010rr    iaaaaaaa
    76543210    76543210
```

```
r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This two-byte instruction transfers a byte of data from the specified register, r, into the byte of memory pointed to by the effective address. The contents of register r remain unchanged and the contents of the memory byte are replaced.
Indirect addressing may be specified.

Pseudocode:

```
STRR,rn rel          ;*(rel) = rn;                          ;3,2
STRR,rn *rel         ;*(*(rel)) = rn;                       ;5,2
```

STRZ

Contents | < Browse | Browse >

STORE REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: STRZ r
CALM Mnemonic: LOAD reg,A
IEEE-694 Mnemonic: MOV reg,.0
Binary Code: $C1-$C3

```
110000rr
76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction transfers the contents of register zero into the specified register, r. The previous contents of register r are lost. The contents of register zero remain unchanged.

Description:
Register r may not be specified as zero. This operation code, %11000000,

is reserved for NOP.

Pseudocode:

```
    STRZ    rn          ;rn = r0;                               ;2,1
```

SUBA

Contents | < Browse | Browse >

SUBTRACT ABSOLUTE

Addressing Mode:	Absolute
Size:	3 bytes
Signetics Mnemonic:	SUBA,r (*)a(,X)
CALM Mnemonic:	SUB|SUBB reg,abs
	SUB|SUBB reg,@abs
	SUB|SUBB A,(reg)+abs
	SUB|SUBB A,(reg)+@abs
	SUB|SUBB A,(+reg)+abs
	SUB|SUBB A,(+reg)+@abs
	SUB|SUBB A,(-reg)+abs
	SUB|SUBB A,(-reg)+@abs
IEEE-964 Mnemonic:	SUB|SUBC reg,/abs
	SUB|SUBC reg,/@abs
	SUB|SUBC .0,/abs(reg)
	SUB|SUBC .0,/@abs(reg)
	SUB|SUBC .0,/abs(+reg)
	SUB|SUBC .0,/@abs(+reg)
	SUB|SUBC .0,/abs(-reg)
	SUB|SUBC .0,/@abs(-reg)
Binary Code:	$AC-$AF

101011rr	iccaaaaa	aaaaaaaa
76543210	76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
c: index control bits (2 bits)
a: absolute address (13 bits)

Execution Time:	4 cycles (12 clock periods)
Processor Registers Affected:	C, CC, IDC, OVF

Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This three-byte instruction causes the contents of the byte of memory pointed to by the effective address to be subtracted from the contents of register r. The result of the subtraction replaces the contents of register r.

The subtraction is performed by taking the binary two's complement of the contents of the memory byte and adding that result to the contents of register r.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits #1 and #0, byte #0, indicate the index register and the destination of the operation implicitly becomes register zero.

Note:

Subtract with Borrow may be effected. See Carry (C) bit.

Pseudocode:

```
SUBA,rn abs          ;rn -= *(abs);                          ;4,3
SUBA,r0 abs,rn       ;r0 -= *(abs + rn);                     ;4,3
SUBA,r0 abs,rn+      ;r0 -= *(abs + ++rn);                   ;4,3
                     ;or, rn++; r0 -= *(abs + rn);
SUBA,r0 abs,rn-      ;r0 -= *(abs + --rn);                   ;4,3
                     ;or, rn--; r0 -= *(abs + rn);
SUBA,rn *abs         ;rn -= *(*(abs));                       ;6,3
SUBA,r0 *abs,rn      ;r0 -= *(*(abs) + rn);                  ;6,3
SUBA,r0 *abs,rn+     ;r0 -= *(*(abs) + ++rn);                ;6,3
                     ;or, rn++; r0 -= *(*(abs) + rn);
SUBA,r0 *abs,rn-     ;r0 -= *(*(abs) + --rn);                ;6,3
                     ;or, rn--; r0 -= *(*(abs) + rn);
```

SUBI

Contents | < Browse | Browse >

SUBTRACT IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: SUBI,r v
CALM Mnemonic: SUB|SUBB reg,#imm
IEEE-694 Mnemonic: SUB|SUBC reg,#imm
Binary Code: $A4-$A7

101001rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This two-byte instruction causes the contents of the second byte of this

instruction to be subtracted from the contents of register r. The result of the subtraction replaces the contents of register r.

The subtraction is performed by taking the binary two's complement of the contents of the second instruction byte and adding that result to the contents of register r.

Note:

Subtract with Borrow may be effected. See Carry (C) bit.

Pseudocode:

```
SUBI,rn imm          ;rn -= imm;                              ;2,2
```

SUBR

Contents | < Browse | Browse >

SUBTRACT RELATIVE

Addressing Mode:	Relative
Size:	2 bytes
Signetics Mnemonic:	SUBR,r (*)a
CALM Mnemonic:	SUB|SUBB reg,.+rel
	SUB|SUBB reg,@.+rel
IEEE-694 Mnemonic:	SUB|SUBC reg,$rel
	SUB|SUBC reg,$@rel
Binary Code:	$A8-$AB

101010rr	iaaaaaaa
76543210	76543210

r: register (2 bits)
i: indirect addressing flag (1 bit)
a: relative address (7 bits)

Execution Time:	3 cycles (9 clock periods)
Processor Registers Affected:	C, CC, IDC, OVF

Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:

This two-byte instruction causes the contents of the byte of memory pointed to by the effective address to be subtracted from the contents of register r. The result of the subtraction replaces the contents of register r.

The subtraction is performed by taking the binary two's complement of the contents of the byte of memory and adding that result to the contents of register r.

Indirect addressing may be specified.

Note:

Subtract with Borrow may be effected. See Carry (C) bit.

Pseudocode:

```
SUBR,rn rel          ;rn -= *(rel);                          ;3,2
SUBR,rn *rel         ;rn -= *(*(rel));                       ;5,2
```

SUBZ

Contents | < Browse | Browse >

SUBTRACT FROM REGISTER ZERO

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: SUBZ r
CALM Mnemonic: SUB|SUBB A,reg
IEEE-694 Mnemonic: SUB|SUBC .0,reg
Binary Code: $A0-$A3

```
101000rr
76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: C, CC, IDC, OVF
Condition Code Setting:

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

Description:
This one-byte instruction causes the contents of the specified register, r, to be subtracted from the contents of register zero. The result of the subtraction replaces the contents of register zero.

The subtraction is performed by taking the binary two's complement of the contents of register r and adding that result to the contents of register zero. The contents of register r remain unchanged.

Note:
Subtract with Borrow may be effected. See Carry (C) bit.

Pseudocode:

```
SUBZ    rn           ;r0 -= rn;                              ;2,1
```

TMI

Contents | < Browse | Browse >

TEST UNDER MASK IMMEDIATE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: TMI,r v
CALM Mnemonic: TEST reg,#imm
IEEE-694 Mnemonic: TEST reg,#imm
Binary Code: $F4-$F7

111101rr	vvvvvvvv
76543210	76543210

r: register (2 bits)
v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

Register r	CC1	CC0
All of the selected bits are 1s	0	0
Not all of the selected bits are 1s	1	0

Description:
This two-byte instruction tests individual bits in the specified register r to determine if they are set to binary one. During execution, each bit in the v field of the instruction is tested for a one, and if a particular bit in the v field contains a one, the corresponding bit in register r is tested for a one or zero. The condition code is set to reflect the result of the operation.
If a bit in the v field is zero, the corresponding bit in register r is not tested.

Pseudocode:

```
TMI,rn  imm          ;CC = (rn & imm == imm) ? EQ : LT;       ;3,2
```

TPSL

Contents | < Browse | Browse >

TEST PROGRAM STATUS LOWER, SELECTIVE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: TPSL v
CALM Mnemonic: TEST L,#imm
TEST CARRY
TEST LOGICOMP
TEST OVERFLOW
TEST WITHCARRY
TEST BANK or TEST BANK1
IEEE-694 Mnemonic: TEST .L,#imm
Binary Code: $B5

```
    10110101    vvvvvvvv
    76543210    76543210
```

v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

	CC1	CC0
All of the selected bits in PSL are 1s	0	0
Not all of the selected bits in PSL are 1s	1	0

Description:
This two-byte instruction tests individual bits in the Lower Program Status Byte to determine if they are set to binary one. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is tested for a one or zero. The Condition Code is set to reflect the result of this operation.

If a bit in the v field is zero, the corresponding bit in the status byte is not tested.

Pseudocode:

```
    TPSL    imm          ;CC = (PSL & imm == imm) ? EQ : LT;      ;3,2
```

TPSU

Contents | < Browse | Browse >

TEST PROGRAM STATUS UPPER, SELECTIVE

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: TPSU v
CALM Mnemonic: TEST U,#imm
TEST IOF
TEST OUTPUT
TEST INPUT
IEEE-694 Mnemonic: TEST .U,#imm
Binary Code: $B4

```
    10110100    vvvvvvvv
    76543210    76543210
```

v: value (8 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: CC
Condition Code Setting:

	CC1	CC0
All of the selected bits in PSU are 1s	0	0
Not all of the selected bits in PSU are 1s	1	0

Description:
  This two-byte instruction tests individual bits in the Upper Program Status Byte to determine if they are set to binary one. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is tested for a one or zero. The Condition Code is set to reflect the result of this operation.
  If a bit in the v field is zero, the corresponding bit in the status byte is not tested.

Pseudocode:

```
    TPSU    imm          ;CC = (PSU & imm == imm) ? EQ : LT;      ;3,2
```

WRTC

Contents | < Browse | Browse >

WRITE CONTROL

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: WRTC,r
CALM Mnemonic: LOAD $CTRL,reg
IEEE-694 Mnemonic: OUT reg,CTRL
Binary Code: $B0-$B3

```
    110100rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
  This one-byte output instruction causes a byte of data to be made available to an external device.
  The byte to be output is taken from register r and is made available on the data bus. Signals on the busses are true levels, ie. high levels are ones.
  When executing this instruction, the processor raises the Operation Request (OPREQ) line and simultaneously places the data on the data bus. Along with OPREQ, the M/IO line is switched to IO, the R/W line is switched to W (Write), the D/C line is switched to C (Control), the E/NE line is switched to NE (Non-extended), and a Write Pulse (WRP) is generated.

Pseudocode:

```
    WRTC,rn              ;*(CONTROLBUS) = rn;                    ;2,1
```

WRTD

Contents | < Browse | Browse >

WRITE DATA

Addressing Mode: Register
Size: 1 byte
Signetics Mnemonic: WRTD,r
CALM Mnemonic: LOAD $DATA,reg
IEEE-694 Mnemonic: OUT reg,DATA
Binary Code: $F0-$F3

```
    111100rr
    76543210
```

r: register (2 bits)

Execution Time: 2 cycles (6 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This one-byte output instruction causes a byte of data to be made available to an external device. The byte to be output is taken from register r and is made available on the data bus. Signals on the busses are true levels, ie. high levels are ones.

When executing this instruction, the processor raises the Operation Request (OPREQ) line and simultaneously places the data on the data bus. Along with OPREQ, the M/IO line is switched to IO, the R/W line is switched to W (Write), and a Write Pulse (WRP) is generated. Also, during the valid OPREQ signals, the D/C line is switched to D (Data) and the E/NE line is switched to NE (Non-extended).

Pseudocode:

```
    WRTD,rn             ;*(DATABUS) = rn;                    ;2,1
```

WRTE

Contents | < Browse | Browse >

WRITE EXTENDED

Addressing Mode: Immediate
Size: 2 bytes
Signetics Mnemonic: WRTE,r v
CALM Mnemonic: LOAD $port,reg
IEEE-694 Mnemonic: OUT reg,port
Binary Code: $D4-$D7

```
    110101rr    vvvvvvvv
```

```
    76543210    76543210

r: register (2 bits)
v: value (8 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This two-byte output instruction causes a byte of data to be made available to an external device. The byte to be output is taken from register r and is made available on the data bus. Simultaneously, the data in the second byte of this instruction is made available on the address bus. The second byte, v, may be interpreted as a device address.

Signals on the busses are true levels, ie. high levels are ones.

When executing this instruction, the processor raises the Operation Request (OPREQ) line and simultaneously places the data from register r on the data bus and the data from the second byte of this instruction on the address bus. Along with OPREQ, the M/IO line is switched to IO, the R/W line is switched to W (Write), the E/NE line is switched to E (Extended), and a Write Pulse (WRP) is generated.

Pseudocode:

```
    WRTE,rn $vv          ;*(ADDRESSBUS) = $vv; *(DATABUS) = rn;   ;3,2
```

ZBRR

Contents | < Browse | Browse >

ZERO BRANCH, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: ZBRR (*)a
CALM Mnemonic: JUMP 0+zero
JUMP @0+zero
IEEE-694 Mnemonic: BR !zero
BR !@zero
Binary Code: $9B

```
    10011011    iaaaaaaa
    76543210    76543210

i: indirect addressing flag (1 bit)
r: relative address (7 bits)
```

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: None
Condition Code Setting: N/A

Description:
This two-byte unconditional relative branch instruction directs the processor to calculate the effective address differently than the usual

calculation for the Relative Addressing mode.
  The specified value, a, is interpreted as a relative displacement from page zero, byte zero. Therefore, displacement may be specified from -64 to +63 bytes. The address calculation is modulo 8192, so the negative displacement actually will develop addresses at the end of page zero. For example, ZBRR -8 will develop an effective address of 8184, and ZBRR +52 will develop an effective address of 52.
  This instruction causes the processor to clear address bits #13 and #14, the page address bits, and to replace the contents of the Instruction Address Register (IAR) with the effective address of the instruction. This instruction may be executed anywhere within addressable memory.
  Indirect addressing may be specified.

Pseudocode:

```
    ZBRR    zero        ;goto zero;                         ;3,2
    ZBRR    *zero       ;goto *(zero);                      ;5,2
```

ZBSR

Contents | < Browse | Browse >

ZERO BRANCH TO SUBROUTINE, RELATIVE

Addressing Mode: Relative
Size: 2 bytes
Signetics Mnemonic: ZBSR (*)a
CALM Mnemonic: CALL 0+zero
CALL @0+zero
IEEE-694 Mnemonic: CALL !zero
CALL !@zero
Binary Code: $BB

```
    10111011    iaaaaaaa
    76543210    76543210
```

i: indirect addressing flag (1 bit)
r: relative address (7 bits)

Execution Time: 3 cycles (9 clock periods)
Processor Registers Affected: SP
Condition Code Setting: N/A

Description:
  This two-byte unconditional relative subroutine branch instruction directs the processor to calculate the effective address differently than the usual calculation for the Relative Addressing mode.
  The specified value, a, is interpreted as a relative displacement from page zero, byte zero. Therefore, displacement may be specified from -64 to +63 bytes. The address calculation is modulo 8192, so the negative displacement actually will develop addresses at the end of page zero. For example, ZBRR -8 will develop an effective address of 8184, and ZBRR +52 will develop an effective address of 52.
  This instruction causes the processor to clear address bits #13 and #14, the page address bits, and may be executed anywhere within addressable

memory.
  Indirect addressing may be specified.
  When executed, this instruction causes the Stack Pointer (SP) to be incremented by one, the address of the byte following this instruction is pushed into the Return Address Stack (RAS), and control is transferred to the effective address.

Pseudocode:

```
    ZBSR    zero        ;gosub zero;                        ;3,2
    ZBSR    *zero       ;gosub *(zero);                     ;5,2
```

C: INSTRUCTIONS, ADDITIONAL INFORMATION

Contents | < Browse | Browse >

The 2650 uses variable length instructions that are one, two or three bytes long. The instruction length is determined by the nature of the operation being performed and the addressing mode being used. Thus, the instruction can be expressed in one byte where no memory operand addressing is necessary, as with register-to-register or rotate instructions. On the other hand, for direct addressing instructions, three bytes are allocated. The relative and immediate addressing modes allow two-byte instructions to be implemented.
  The 2650 uses explicit operand addressing; that is, each instruction species the operand address. The first byte of each 2650 instruction is divided into three fields and specified the operation to be performed, the addressing mode to be used and, where appropriate, the register or condition code mask to be used.

```
    fffcccrr
    76543210
```

f: function field (3 bits)
c: class field (3 bits)
r: register field (2 bits)

The class field species the instruction group, the major address mode and the number of processor cycles required for each instruction. The class field also specifies, with one exception, the number of bytes in the instruction. The following table shows the specifications for each class.

Class Field	Instruction Group	Address Register	Byte Length	Direct Cycles
0	Arithmetic	Register	1	2
1	Arithmetic	Immediate	2	2
2	Arithmetic	Relative	2	3
3	Arithmetic	Absolute	3	4
4	Control (incl. rotate)	Various	1	2
5	Control	Various	1-2	3
6	Branch	Relative	2	3
7	Branch	Absolute	3	3

Within the arithmetic group (classes 0, 1, 2 and 3), the function field specifies one of the eight operations as follows:

Function Field	Arithmetic Operation
0	Load
1	Exclusive OR
2	AND
3	Inclusive OR
4	Add
5	Subtract
6	Store
7	Compare

Within the branch group (classes 6 and 7), the function field specifies one of the eight operations as follows:

Function Field	Branch Operation
0	Branch on Condition True
1	Branch to Subroutine on Condition True
2	Branch on Register Non-Zero
3	Branch to Subroutine on Register Non-Zero
4	Branch on Condition False
5	Branch to Subroutine on Condition False
6	Branch on Incrementing Register
7	Branch on Decrementing Register

There is very little pattern to the use of the function field within the control group (classes 4 and 5).

The register field is used to specify the index register, to specify the operand source register, to specify the destination register, or a condition code mask. For the register-to-register and the indexed instructions, register zero is implicitly assumed to be the source or the destination for the instruction. For all other instructions that involve a register, the register field allows any of four registers to be specified, except for indexed branch instructions which require that register 3 be specified.

Conditional branch instructions utilize the 2-bit register field as a condition code mask field. A few instructions use the register field as part of the operation code and consequently allow no variation in register usage.

E: SUMMARY OF 2650 INSTRUCTION MNEMONICS

Contents | < Browse | Browse >

In these tables parentheses are used to indicate options. In no case are they coded in any instruction. The following abbreviations are used:

- r - register expression, must evaluate to 0 <= r <= 3.
- v - value expression
- * - indirect indicator
- a - address expression
- x - index register expression
- X - index register expression with optional auto-increment or auto-decrement

NOTES:
- the use of the indirect indicator is always optional.
- when an index register expression is specified, it can be followed by ',+' or ',-' which indicates use of auto-increment or auto-decrement of the index register. For example:

```
      LODA,r0 DPR,r3,+
```

  BXA and BSXA are exceptions and do not permit auto-increment or auto-decrement.
- even though an address expression is specified in a hardware relative addressing instruction, the assembler develops it into a value of (-64 <= V <= +63).
- a memory reference instruction which requires indexing may use only register zero as the destination of the operation.
- if an index register expression is used with either the BXA or BSXA instructions it must specify index register #3 (either register bank) for indexing. Any other value in the index field will produce an error during assembly. However, it is not necessary to use an index register expression with these instructions; a blank in this field will default to register #3.

LOAD/STORE

1	LODZ	r	Load Register Zero
2	LODI,r	v	Load Immediate
2	LODR,r	(*)a	Load Relative
3	LODA,r	(*)a(,X)	Load Absolute
1	STRZ	r	Store Register Zero
2	STRR,r	(*)a	Store Relative
3	STRA,r	(*)a(,X)	Store Absolute

ARITHMETIC

1	ADDZ	r	Add to Register Zero
2	ADDI,r	v	Add Immediate
2	ADDR,r	(*)a	Add Relative
3	ADDA,r	(*)a(,X)	Add Absolute
1	SUBZ	r	Subtract from Register Zero
2	SUBI,r	v	Subtract Immediate
2	SUBR,r	(*)a	Subtract Relative
3	SUBA,r	(*)a(,X)	Subtract Absolute

LOGICAL

1	IORZ	r	Inclusive OR to Register Zero
2	IORI,r	v	Inclusive OR Immediate
2	IORR,r	(*)a	Inclusive OR Relative
3	IORA,r	(*)a(,X)	Inclusive OR Absolute
1	EORZ	r	Exclusive OR to Register Zero
2	EORI,r	v	Exclusive OR Immediate
2	EORR,r	(*)a	Exclusive OR Relative
3	EORA,r	(*)a(,X)	Exclusive OR Absolute
1	ANDZ	r	AND to Register Zero
2	ANDI,r	v	AND Immediate
2	ANDR,r	(*)a	AND Relative
3	ANDA,r	(*)a(,X)	AND Absolute

BRANCH

2	BCTR,v	(*)a	Branch on Condition True Relative

```
   2 BCTA,v (*)a     Branch on Condition False Relative
   3 BCFR,v (*)a     Branch on Condition True Absolute
   3 BCFA,v (*)a     Branch on Condition False Absolute
   2 BRNR,r (*)a     Branch on Register Non-Zero Relative
   3 BRNA,r (*)a     Branch on Register Non-Zero Absolute
   2 BIRR,r (*)a     Branch on Incrementing Register
                                 Relative
   3 BIRA,r (*)a     Branch on Incrementing Register
                                 Absolute
   2 BDRR,r (*)a     Branch on Decrementing Register
                                 Relative
   3 BDRA,r (*)a     Branch on Decrementing Register
                                 Absolute
   3 BXA    (*)a(,x) Branch Indexed, Absolute
   2 ZBRR   (*)a     Zero Branch, Relative

 SUBROUTINE BRANCH
   2 BSTR,v (*)a     Branch to Subroutine on Condition True,
                                 Relative
   3 BSTA,v (*)a     Branch to Subroutine on Condition True,
                                 Absolute
   2 BSFR,v (*)a     Branch to Subroutine on Condition
                                 False, Relative
   3 BSFA,v (*)a     Branch to Subroutine on Condition
                                 False, Absolute
   2 BSNR,r (*)a     Branch to Subroutine on Non-Zero
                                 Register, Relative
   3 BSNA,r (*)a     Branch to Subroutine on Non-Zero
                                 Register, Absolute
   3 BSXA   (*)a(,x) Branch to Subroutine Indexed, Absolute
   2 ZBSR   (*)a     Zero Branch to Subroutine, Relative

 SUBROUTINE RETURN
   1 RETC,v          Return from Subroutine, Conditional
   1 RETE,v          Return from Subroutine and Enable
                                 Interrupt, Conditional

 COMPARISON
   1 COMZ   r        Compare to Register Zero
   2 COMI,r v        Compare Immediate
   2 COMR,r (*)a     Compare Relative
   3 COMA,r (*)a(,X) Compare Absolute

 INPUT/OUTPUT
   1 REDC,r          Read Control
   1 REDD,r          Read Data
   1 WRTC,r          Write Control
   1 WRTD,r          Write Data
   2 REDE,r v        Read Extended
   2 WRTE,r v        Write Extended

 PROGRAM STATUS MANIPULATION
   1 LPSU            Load Program Status, Upper from r0
   1 LPSL            Load Program Status, Lower from r0
   3 LDPL   (*)a     Load Program Status, Lower from memory
   1 SPSU            Store Program Status, Upper to r0
   1 SPSL            Store Program Status, Lower to r0
   3 STPL   (*)a     Store Program Status, Lower to memory
```

```
  2 CPSU   v        Clear Program Status, Upper
  2 CPSL   v        Clear Program Status, Lower
  2 PPSU   v        Preset Program Status, Upper
  2 PPSL   v        Preset Program Status, Lower
  2 TPSU   v        Test Program Status, Upper
  2 TPSL   v        Test Program Status, Lower

ROTATE
  1 RRR ,r          Rotate Register Right
  1 RRL ,r          Rotate Register Left

MISCELLANEOUS
  1 NOP             No Operation
  1 HALT            Halt, Enter Wait State
  2 TMI ,r v        Test Under Mask Immediate
  1 DAR ,r          Decimal Adjust Register
```

F: NOTES ABOUT THE 2650 PROCESSOR

Contents | < Browse

---

1. Auto-increment, decrement of index register: This feature is optional on any instruction which uses indexing with the exception of BXA and BSXA. The increment or decrement occurs before the index register is added to the displacement in the instruction.

2. The contents of registers when used for indexing are considered to be unsigned absolute numbers. Consequently, index registers can contain values from 0 to 255. They "wrap-around" so that the number following 255 is 0.

3. Only absolute addressing instructions can be indexed.

4. The BDRA, BDRR, BIRA and BIRR instructions perform the increment or decrement before testing for zero. The only time the branch is not taken is when the register contains zero.

5. All hardware relative addressing is implemented as modulo 8K and therefore relative addressing across the top of a page boundary will result in a physical address near the bottom of the page being accessed. For example:

   ```
   $1FFC   LODR,r2 $+16
   ```

   This instruction results, during execution, in accessing the byte at location $000C in the same page as the instruction. Similarly, negative relative addresses from near the bottom of a page may result in an effective address near the top of the page.

6. Page boundaries cannot be indexed across.

7. Data can always be accessed across a page boundary through use of relative indirect or absolute indirect addressing modes.

8. The only way to transfer control to a program in some other page is to

branch absolute or branch indirectly to the new page. Program execution cannot flow across a page boundary.

9. Unconditional branch or branch to subroutine instructions are coded by specifying a value of 3 in the register/value field of BSTA, BSTR, BCTA or BCTR. For example:

```
UN      EQU     3
        ...
        ...
        ...
        BSTA,un PAL
        BCTR,r3 LOOP
```

Unconditional branches on conditions false (BCFA, BCFR) are not allowed.

Converted using GuideML 3.10