RIPTIDE – II PROGRAMMER MANUAL

Table of Contents

Introduction

RIPTIDE-II Changes

The **RIPTIDE-II** CPU differs from the **8X-RIPTIDE** in a number of ways, including:

- Pipeline was reduced from 8 stages to 7 stages (this affects I/O timing, resulting in lower latency).
- IV Bus was replaced with separate data and address buses.
  - Data and address can be read from the CPU concurrently.
  - Both banks of each bus are also readable concurrently.
- Cache Controller support: P_Miss and D_Miss inputs have been added as a cache controller interface. While a Miss input is high, the CPU will not read data from the corresponding bus.
  - P_Miss – Program Memory Miss
    - When high, the program instruction data is assumed to be invalid.
  - D_Miss – Data Miss
    - When high, the CPU I/O input data is assumed to be invalid.
    - *Additionally,* when high, no write commands are issued by the CPU.

Original Introduction

The following is included from the **8X-RIPTIDE** Programming Manual, and serves as an overview and background for the architecture.

"8X-RIPTIDE is an 8-bit Harvard architecture with a 16-bit program memory bus. It is based on the 8X300 architecture but was re-engineered for significantly higher performance at the cost of higher complexity and size. 8X-RIPTIDE CPUs are implemented with an 8-stage pipeline, and have separate IO buses for read and write. The shift and merge capabilities of the 8X-RIPTIDE are similar to those of the 8X300, except the 8X-RIPTIDE has internal IO registers that hold the last value written to the IO bus. This allows the CPU to perform merge operations without performing IO reads, but since a copy of the last written data is kept internally merge instructions can only operate on the data that was last written. If the data changes on external devices the merge operation will yield incorrect results, as it will use the outdated internal buffer as the starting point.

In addition to differences in shift and merge capabilities, the 8X-RIPTIDE has two new instructions, which are special cases of the NZT instruction. These two new instructions are call and return instructions, which add the ability to have subroutines in a program. Because of the addition of these new instructions the behavior of the NZT instruction had to be modified. Illegal instructions to use an IV bus address as the source register for NZT will result in the execution of a call or return operation. The naming of the bits in the 8X-RIPTIDE architecture is the same as for the 8X300, which is different from the naming scheme used in most modern designs. Bits are named 0 to n, with 0 being the MSB and n being the LSB.

Example:
Bits 1 0 1 0 0 1 0 1
Bit # 0 1 2 3 4 5 6 7 where bit 7 is the LSB and bit 0 is the MSB"

Instruction Set Summary

The instruction set of the RIPTIDE-II is the same as the 8X-RIPTIDE instruction set. A brief description of each instruction is given below.

MOVE: Data from the source register or IV bus is copied to the destination register or IV bus. The data may be rotated any number of places and/or masked to any length during the MOVE operation. The source data field remains unchanged after the operation.

ADD: Data from the source register or IV bus is added to the contents of the AUX register and the result is placed in the destination register or IV bus. The data may be rotated and/or masked during the operation. The source data field and the AUX register remain unchanged after the operation unless one is also the destination. AUX can be explicitly used as the source, in which case it is added with itself and the result can also be stored in AUX if it is specified as the destination.

AND: Data from the source register or IV bus is ANDed with the contents of the AUX register and the result is placed in the destination register or IV bus. The data may be rotated and/or masked during the operation. The source data field and the AUX register remain unchanged after the operation unless one is also the destination. AUX can also be used as the source field.

XOR: Data from the source register or IV bus is XORed with the contents of the AUX register and the result is stored in the destination register or IV bus. The data may be rotated and/or masked during the operation. The source data field and the AUX register remain unchanged after the operation unless one is also the destination. AUX can also be used as the source field.

XEC: Causes the execution of the instruction at the address formed by replacing the least significant bits of the last address with the sum of the I field and the data in the source register or IV bus. After the execution of the instruction of the specified address, instruction execution continues at the address following the XEC instruction, unless the executed instruction caused a jump.

NZT: The least significant bits of the instruction address are replaced by the I field data if the register or IV bus specified by the source field has non-zero contents. The tested data field remains unchanged. The IV bus address registers cannot be used as source registers for any instruction, however NZT instructions that attempt to use IV bus addresses as the source field will be decoded differently and result in either a CALL or a RET instruction being executed. The encoding details for CALL and RET will be shown separately as if they were entirely unrelated instructions.

XMIT: The data in the I field is placed in the register or IV bus specified as the destination.

JMP: The address of the next instruction to be executed is changed to that specified by the 13-bit A field of the instruction.

CALL: The address of the next instruction to be executed is changed to that specified by combining the page register with the 8-bit I field of the instruction. The AUX register is always used as the page register. The address of the instruction following the CALL instruction is pushed onto an 8-level hardware call stack. This instruction is encoded as an NZT instruction with register 07 as the source. Using register 07 as the source of an NZT instruction is not allowed in 8X300 processors, but in the RIPTIDE-II this condition results in the instruction being interpreted in an alternative way. The address 07 is not used as a source or destination register, but instead serves as an extension of the op code.

RET: The address of the next instruction to be executed is changed to that specified by the top of the call stack. The top entry is popped off the stack. No page register is needed, as the stack contains the full address. This instruction is encoded as an NZT instruction with register 17 as the source. Using register 17 as the source of an NZT instruction is not allowed in 8X300 processors, but in the RIPTIDE-II this condition results in the instruction being interpreted in an alternative way. The address 17 is not used as a source or destination, but instead serves as an extension of the op code.

MOVE instructions – Op Code 0

MOVE, register, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: (S) → D

Description:
The contents of the register specified by S are right rotated as specified by R and placed in the destination register specified by D. The contents of the source register remain unchanged. The original contents of the destination register are lost.

S specifies the source register

R specifies the number of places the data is to be rotated

D specifies the destination register.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11
- R: 0, 1, 2, 3, 4, 5, 6, 7
- D: 00, 01, 02, 03, 04, 05, 06, 11

Example:
Move the contents of R1, right rotated 2 places, to the AUX register.

Instruction word

000 | 00 001 | 010 | 00 000
OP S R D

Assembler notation: MOVE R1(2), AUX

Instruction operation

10010110 Copy source register R1 (S = 01)
10100101 Rotate 2 places (R = 2)
10100101 Move result into AUX (D = 00)

Result

The original contents of the AUX register are replaced by the rotated data of R1. The contents of R1 are not changed.

Data flow

Register file → right rotate → Mask (disabled) → ALU → register file

MOVE, register, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: (S) → IV bus address

Description:
The data stored in S is rotated by R places and stored in the IV bus address register specified by D. The source register is left unchanged.

S: Specifies source register.

R: Specifies the number of places that the source data is to be rotated.

D: Specifies the destination IV bus address register.
D = 07: Left bank address
D = 17: Right bank address

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11
- R: 0, 1, 2, 3, 4, 5, 6, 7
- D: 07, 17

Example:
Set the address of the right IO bank to the address given by the contents of R3. Assembler notation: MOVE R3(0), R17

Instruction word

000 | 00 011 | 000 | 01 111
OP S R D

Result

The address on the right IO bank is changed to the value stored in R3.

Data flow
Register file → rotate unit → mask unit (disabled) → ALU → shift and merge unit (disabled) → IO module.

MOVE, register, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) → D

Description:
Move the least significant L bits of the source register to the IV bus.

S: Specifies source register.

L: Specifies the number of bits to be copied. L = 0 selects an 8 bit field.

D1: D1 = 2 selects left bank, D1 = 3 selects right bank.

D0: Specifies the bit position in the IV bus with which the LSB of the source data will be aligned.

The order of the operation is:

Read data from source register.

Propagate source data through the right rotate unit.

Propagate source data through the mask unit.

Propagate source data through ALU.

Shift source data and read data from IV latch.

Merge the IV latch and source data.

Write result to IV bus.

Only L bits (aligned to D0) of the IV bus are changed, the remaining bits and the source register are not modified.

Valid operands:

- S: 0, 1, 2, 3, 4, 5, 6, 10, 11.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:
Move the contents of the least significant 3 bits of register 11 to the IV bus at the left bank, with bit 5 as the least significant position in the IV bus.

Assembler notation: MOVE R11, 3, LIV5
Instruction word: 000 01001 011 10101

Effect of instruction:
Original data in IV latch: 11101111
Contents of R11: 11011000
Masked data: *****000
Shifted data: ***000**
New data in IV latch: 11100011
* represents masked bits.

MOVE, IV bus, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (IV byte) → D

Description:
Move the L-bit field of the IV bus data to the least significant L bits of the register specified by D.

S1: Specifies the bank of the IV bus which is the data source.
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0: Specifies the bit which will be the least significant bit of the input data field after rotation.

L: Specifies the number of bits to be copied. L = 0 selects an 8 bit field.

D: Specifies the address of the destination register.

The order of the operation is:

Read data on the IV bus specified by S1..

Right rotate the source data as specified by S0.

Mask the rotated source data as specified by L.

Propagate source data through ALU.

Write masked field to the least significant L bits of the destination register, with zeros in the unmasked positions.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D: 00, 01, 02, 03, 04, 05, 06, 11.

Example:
Move bits 1, 2, and 3 of the enabled IV byte at the right bank to register 6.

Assembler notation: MOVE RIV3, 3, R6
Instruction word: 000 11011 011 00110

MOVE, IV bus, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) → D

Description:
Move the variable length field, specified by S0 and L, from the bank specified by S1 to the field and bank specified by D.

S1: Specifies the bank of the IV bus which is the data source.
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0: Specifies the bit which will be the least significant bit of the input data field after rotation.

L: Specifies the length of the masked field that is to be processed and merged with the last data that was written to the target IV bus.

D1: Specifies the bank of the IV bus which is the destination.
D1 = 2 selects the left bank;
D1 = 3 selects the right bank.

D0: Specifies the bit position in the target IV bus with which the least significant bit of the processed data field should be aligned. This means that the processed data field is left-shifted so that bit 7 is aligned with bit D0 in the target IV bus.

The order of the operation is:

Read data on the IV bus specified by S1.

Right rotate the source data as specified by S0.

Mask the rotated source data as specified by L.

Propagate source data through ALU.

Left-shift source data and read data from IV latch as specified by D0.

Merge the IV latch and source data.

Write result to the target IV bus.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:

Move bits 0, 1, and 2, of the IV byte at the left bank to bits 3, 4,and 5 of the same IV byte.

Assembler notation: MOVE LIV2, 3, LIV5
Instruction word: 000 10010 011 10101

MOVE, IV bus, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: Enable the IV byte at the bank specified by D, whose address is given by the bus data specified by S.

(S) → IV Bus Address

Description:
Copy the data from the IV bus specified by S1, right rotate the data field until bit S0 is in the least significant position, keep the least significant L bits masking off the rest, and write the result to the specified IV bus address register.

S1: Specifies the bank of the IV bus which is the data source.
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0: Specifies the bit which will be the least significant bit of the input data field after rotation.

L: Specifies the length of the masked field that is to be processed and written to the IV bus address.

D: Specifies the bank of the IV bus address which is the destination.
D = 07 selects the left bank (IVL);
D = 17 selects the right bank (IVR).

The order of the operation is:

Read data on the IV bus specified by S1.

Right rotate the source data as specified by S0.

Mask the rotated source data as specified by L.

Propagate source data through ALU.

Propagate data through shift unit.

Propagate data through merge unit.

Write result to the target IV bus address.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D: 07, 17.

Note that L = 0 specifies an 8-bit field.

Example:

Set the left bank IV bus address to the value of bits 2:4 of the left IV bus.

Assembler notation: MOVE LIV4, 3, IVL
Instruction word: 000 10100 011 00111

NOP Instruction – Op Code 0

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=0			(S=D)					R=0			(D=S)				

Operation: No operation, will always execute in 1 clock cycle.

Description:
Special case of the MOVE instruction that does not modify the source or destination register. A NOP is performed when MOVE-ing with no rotation (R=0) where the destination register is the same register as the source register.

Bits 3-15 should be set to zero when performing an explicit NOP.

Note: For the MOVE instruction to be considered a NOP, it must use the same general purpose register for source and destination. MOVE operations using the IV Bus will never be considered as NOPs.

Valid operands:
When performing an explicit NOP, there are no operands.

Example:
Perform a NOP

Assembler notation: NOP

Instruction word: 000 00000 000 00000

ADD Instructions – Op Code 1

ADD, register, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=1			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: (S) + (AUX) → D

Description:
Add the right rotated contents of the source register S with the contents of the AUX register and place the result in register D. If overflow occurs during the addition, bit 7 of the OVF register is set to 1, otherwise it is set to 0.

S specifies the source register.
R specifies the number of places the data is to be rotated.
D specifies the destination register.

The order of operation is:
Fetch data from the source register.
Right rotate the data from the source register.
Propagate the rotated data through the mask unit and fetch data from the AUX register.
Add the rotated data and the AUX data. The carry flag is updated.
Store data in the destination register.

The source register and the AUX register are not modified unless one if them is also the destination register.

Valid Operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- R: 0, 1, 2, 3, 4, 5, 6, 7.
- D: 00, 01, 02, 03, 04, 05, 06, 11.

Example:
Add the contents of R1, right rotated 4 places, to the contents of the AUX register and store the result in R3.

Assembler notation: ADD R1 (4), R3
Instruction word: 001 00001 100 00011

ADD, register, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=1			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: (S) + (AUX) → IV Bus Address

Description:
Write to the IV bus the address formed by adding the right rotated contents of the source register to the contents of the AUX register at the bank specified by D.

S specifies the source register.
R specifies the number of places the source data is to be rotated.
D specifies the destination bank of the IV bus for the address data.
D = 07 specifies the left bank (IVL);
D = 17 specifies the right bank (IVR).

The order of operation is:
Read the data from the source register.

Right rotate the source data.

Propagate the source data through the mask unit and read the AUX register.

Add the rotated data and the AUX register data, updating the carry flag.

Propagate data through the shift and merge unit.

Write the data to the IV bus.
The content of S and AUX remain unchanged after the operation.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- R: 0, 1, 2, 3, 4, 5, 6, 7.
- D: 07, 17.

Example:
Enable the IV byte at the right bank whose address is the sum of the contents of R3 and AUX.

Assembler notation: ADD R3 (0), IVR OR ADD R3, IVR
Instruction word: 001 00011 000 01111

ADD, register, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=1			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) + (AUX) → D

Description:
Add the contents of the AUX register to the contents of the source register. Store the least significant L bits of the result in the IV bus as specified by the D field.

S specifies the source register.

L: Specifies the number of bits to be copied. L = 0 selects an 8 bit field.

D1: Specifies the bank of the IV bus that is the destination.
D1 = 2 selects the left bank
D1 = 3 selects the right bank

D0: Specifies the bit position in the destination with which the least significant bit of the result should be aligned.

The order of the operation is:

Read data from source register.

Propagate source data through the right rotate unit.

Propagate source data through the mask unit and read the AUX register.

Add the source data with the contents of the AUX register. Update OVF flag.

Shift source data and read data from IV latch.

Merge the IV latch data with the ALU data.

Write result to IV bus.

The content of the source and AUX register remains unchanged after the operation.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:
Add the contents of R11 to the contents of the AUX register and output the least significant 4 bits of the sum to bits 0:3 of the IV bus at the left bank.

Assembler notation: ADD R11, 4, LIV3
Instruction word: 001 01001 100 10011

ADD, IV bus, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=1			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) + (AUX) → D

Description:
Add the L-bit field of the IV bus source data to the contents of the AUX register and move the least significant L bits of the result to the IV bus specified by D0.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed and merged with the existing IV bus data.

D1 specifies the bank of the IV bus which is the destination:
D1 = 2 selects the left bank;
D1 = 3 selects the right bank.

D0 specifies the bit position in the IV byte with which the least significant bit of the processed data field should be aligned. This means that the processed data field is left-shifted so that bit 7 is aligned with D0 of the IV bus.

The order of the operation is:

Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

Add the source data with the contents of the AUX register. Update OVF flag.

Shift the ALU result data data and read data from IV latch.

Merge the IV latch data with the ALU data.

Write result to IV bus.

Note that during the merge phase, the original values of the source field outside the masked field are preserved. The original contents of the destination field are lost.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:

Add the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and move the least significant 4 bits of the sum to the most significant 4 bits of the IV byte at the left bank.

Assembler notation: ADD LIV7, 4, LIV3
Instruction word: 001 10111 100 10011

ADD, IV bus, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=1			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) + (AUX) → D

Description:
Add the L-bit field of the IV bus source data to the contents of the AUX register and move the least significant L bits of the result to the register specified by D. Set the overflow flag as appropriate.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed.

D specifies the address of the destination register.

The order of the operation is:
Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

Add the source data with the contents of the AUX register. Update OVF flag.

Write result to target register.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D: 00, 01, 02, 03, 04, 05, 06, 11.
- Note that L = 0 selects an 8-bit field.

Example:
Add the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and move the result of the sum to register 11.

Assembler notation: ADD LIV7, 4, R11
Instruction word: 001 10111 100 01001

ADD, IV bus, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=1			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) + (Aux) → IV Bus Address

Description:
Add the L-bit field of the IV bus source data to the contents of the AUX register and write the result to the IV bus address register specified by D. Set the overflow flag as appropriate.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed.

D specifies the destination IV bus register.
D = 07 specifies the left bank address (IVL);
D = 17 specifies the right bank address (IVR).

The order of the operation is:
Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

Add the source data with the contents of the AUX register. Update OVF flag.

Propagate data through shift unit.

Propagate data through merge unit.

Write result to target IV bus address register.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D: 07, 17.
- Note that L = 0 selects an 8-bit field.

Example:
Add the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and write the result of the sum to the right bank IV bus address.

Assembler notation: ADD LIV7, 4, IVR
Instruction word: 001 10111 100 01111

AND Instructions – Op Code 2

AND, register, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=2			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ∧ (AUX) → D

Description:
AND the right rotated contents of the source register S with the contents of the AUX register and place the result in register D.

S specifies the source register.
R specifies the number of places the data is to be rotated.
D specifies the destination register.

The order of operation is:
Fetch data from the source register.
Right rotate the data from the source register.
Propagate the rotated data through the mask unit and fetch data from the AUX register.
And the rotated data and the AUX data.
Store data in the destination register.

The source register and the AUX register are not modified unless one if them is also the destination register.

Valid Operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- R: 0, 1, 2, 3, 4, 5, 6, 7.
- D: 00, 01, 02, 03, 04, 05, 06, 11.

Example:
And the contents of R1, right rotated 4 places, with the contents of the AUX register and store the result in R3.

Assembler notation: AND R1 (4), R3
Instruction word: 010 00001 100 00011

AND, register, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=2			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: Enable IV byte with address (S) ∧ (AUX)

Description:
Write to the IV bus the address formed by ANDing the right rotated contents of the source register to the contents of the AUX register at the bank specified by D.

S specifies the source register.
R specifies the number of places the source data is to be rotated.
D specifies the destination bank of the IV bus for the address data.
D = 07 specifies the left bank (IVL);
D = 17 specifies the right bank (IVR).
The order of operation is:
Read the data from the source register.
Right rotate the source data.
Propagate the source data through the mask unit and read the AUX register.
AND the rotated data and the AUX register data.
Propagate data through the shift and merge unit.
Write the data to the IV bus.

The content of the source and AUX register remains unchanged after the operation.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- R: 0, 1, 2, 3, 4, 5, 6, 7.
- D: 07, 17.

Example:
Enable the IV byte at the right bank whose address is the result of ANDing of the contents of R3 and AUX.

Assembler notation: AND R3 (0), IVR OR ADD R3, IVR
Instruction word: 010 00011 000 01111

AND, register, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=2			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ∧ (AUX) → D

Description:
AND the contents of the AUX register to the contents of the source register. Store the least significant L bits of the result in the IV bus as specified by the D field.

S specifies the source register.

L: Specifies the number of bits to be copied. L = 0 selects an 8 bit field.

D1: Specifies the bank of the IV bus that is the destination.
D1 = 2 selects the left bank
D1 = 3 selects the right bank

D0: Specifies the bit position in the destination with which the least significant bit of the result should be aligned.

The order of the operation is:
Read data from source register.

Propagate source data through the right rotate unit.

Propagate source data through the mask unit and read the AUX register.

AND the source data with the contents of the AUX register.

Shift source data and read data from IV latch.

Merge the IV latch data with the ALU data.

Write result to IV bus.

The content of the source and AUX register remains unchanged after the operation.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:
AND the contents of R11 to the contents of the AUX register and output the least significant 4 bits of the sum to bits 0:3 of the IV bus at the left bank.

Assembler notation: AND R11, 4, LIV3
Instruction word: 010 01001 100 10011

AND, IV bus, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=2			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ∧ (AUX) → D

Description:
AND the L-bit field of the IV bus source data with the contents of the AUX register and move the least significant L bits of the result to the register specified by D.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed.

D specifies the address of the destination register.

The order of the operation is:
Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

AND the masked data to the contents of the AUX register.

Write result to target register.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7
- S1: 2, 3
- L: 1, 2, 3, 4, 5, 6, 7, 0
- D: 00, 01, 02, 03, 04, 05, 06, 11

Note that L = 0 selects an 8-bit field.

Example:
AND the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and move the result of the sum to register 11.

Assembler notation: AND LIV7, 4, R11
Instruction word: 010 10111 100 01001

AND, IV bus, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=2			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ∧ (AUX) → D

Description:
AND the L-bit field of the IV bus source data to the contents of the AUX register and move the least significant L bits of the result to the IV bus specified by D0.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed and merged with the existing IV bus data.

D1 specifies the bank of the IV bus which is the destination:
D1 = 2 selects the left bank;
D1 = 3 selects the right bank.

D0 specifies the bit position in the IV byte with which the least significant bit of the processed data field should be aligned. This means that the processed data field is left-shifted so that bit 7 is aligned with D0 of the IV bus.

The order of the operation is:

Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

AND the source data with the contents of the AUX register.

Shift the ALU result data data and read data from IV latch.

Merge the IV latch data with the ALU data.

Write result to IV bus.

Note that during the merge phase, the original values of the source field outside the masked field are preserved. The original contents of the destination field are lost.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:

AND the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and move the least significant 4 bits of the sum to the most significant 4 bits of the IV byte at the left bank.

Assembler notation: AND LIV7, 4, LIV3
Instruction word: 010 10111 100 10011

AND, IV bus, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=2			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ∧ (AUX) → D

Description:
AND the L-bit field of the IV bus source data to the contents of the AUX register and write the result to the IV bus address register specified by D.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed.

D specifies the destination IV bus register.
D = 07 specifies the left bank address (IVL);
D = 17 specifies the right bank address (IVR).

The order of the operation is:
Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

AND the source data with the contents of the AUX register.

Propagate data through shift unit.

Propagate data through merge unit.

Write result to target IV bus address register.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7
- S1: 2, 3
- L: 1, 2, 3, 4, 5, 6, 7, 0
- D: 07, 17
- Note that L = 0 selects an 8-bit field.

Example:
AND the contents of bits 4 to 7 of the IV byte at the left bank with the contents of the AUX register and write the result of the sum to the right bank IV bus address.

Assembler notation: AND LIV7, 4, IVR
Instruction word: 010 10111 100 01111

XOR Instructions – Op Code 3

XOR, register, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=3			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ⊕ (AUX) → D

Description:
XOR the right rotated contents of the source register S with the contents of the AUX register and place the result in register D.

S specifies the source register.
R specifies the number of places the data is to be rotated.
D specifies the destination register.

The order of operation is:

Fetch data from the source register.

Right rotate the data from the source register.

Propagate the rotated data through the mask unit and fetch data from the AUX register.

And the rotated data and the AUX data.

Store data in the destination register.

The source register and the AUX register are not modified unless one if them is also the destination register.

Valid Operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11
- R: 0, 1, 2, 3, 4, 5, 6, 7
- D: 00, 01, 02, 03, 04, 05, 06, 11

Example:
XOR the contents of R1, right rotated 4 places, with the contents of the AUX register and store the result in R3.

Assembler notation: XOR R1 (4), R3
Instruction word: 011 00001 100 00011

XOR, register, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=3			S					R			D				
			S_1		S_0						D_1		D_0		

Operation: Enable the IV byte with address (S) ⊕ (AUX)

Description:
Write to the IV bus the address formed by XORing the right rotated contents of the source register to the contents of the AUX register at the bank specified by D.

S specifies the source register.
R specifies the number of places the source data is to be rotated.
D specifies the destination bank of the IV bus for the address data.
D = 07 specifies the left bank (IVL);
D = 17 specifies the right bank (IVR).

The order of operation is:

Read the data from the source register.

Right rotate the source data.

Propagate the source data through the mask unit and read the AUX register.

XOR the rotated data and the AUX register data.

Propagate data through the shift and merge unit.

Write the data to the IV bus.

The content of the source and AUX register remains unchanged after the operation.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- R: 0, 1, 2, 3, 4, 5, 6, 7.
- D: 07, 17.

Example:
Enable the IV byte at the right bank whose address is the result of XORing the contents of R3 and AUX.

Assembler notation: XOR R3 (0), IVR OR ADD R3, IVR
Instruction word: 011 00011 000 01111

XOR, register, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=3			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ⊕ (AUX) → D

Description:
XOR the contents of the AUX register to the contents of the source register. Store the least significant L bits of the result in the IV bus as specified by the D field.

S specifies the source register.

L: Specifies the number of bits to be copied. L = 0 selects an 8 bit field.

D1: Specifies the bank of the IV bus that is the destination.
D1 = 2 selects the left bank
D1 = 3 selects the right bank

D0: Specifies the bit position in the destination with which the least significant bit of the result should be aligned.

The order of the operation is:
Read data from source register.

Propagate source data through the right rotate unit.

Propagate source data through the mask unit and read the AUX register.

XOR the source data with the contents of the AUX register.

Shift source data and read data from IV latch.

Merge the IV latch data with the ALU data.

Write result to IV bus.

The content of the source and AUX register remains unchanged after the operation.

Valid operands:

- S: 00, 01, 02, 03, 04, 05, 06, 10, 11.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:
XOR the contents of R11 to the contents of the AUX register and output the least significant 4 bits of the sum to bits 0:3 of the IV bus at the left bank.

Assembler notation: XOR R11, 4, LIV3
Instruction word: 011 01001 100 10011

XOR, IV bus, register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=3			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ⊕ (AUX) → D

Description:
XOR the L-bit field of the IV bus source data with the contents of the AUX register and move the least significant L bits of the result to the register specified by D.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed.

D specifies the address of the destination register.

The order of the operation is:
Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

XOR the masked data to the contents of the AUX register.

Write result to target register.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7
- S1: 2, 3
- L: 1, 2, 3, 4, 5, 6, 7, 0
- D: 00, 01, 02, 03, 04, 05, 06, 11

Note that L = 0 selects an 8-bit field.

Example:
XOR the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and move the result of the sum to register 11.

Assembler notation: XOR LIV7, 4, R11
Instruction word: 011 10111 100 01001

XOR, IV bus, IV, bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=3			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ⊕ (AUX) → D

Description:
XOR the L-bit field of the IV bus source data to the contents of the AUX register and move the least significant L bits of the result to the IV bus specified by D0.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed and merged with the existing IV bus data.

D1 specifies the bank of the IV bus which is the destination:
D1 = 2 selects the left bank;
D1 = 3 selects the right bank.

D0 specifies the bit position in the IV byte with which the least significant bit of the processed data field should be aligned. This means that the processed data field is left-shifted so that bit 7 is aligned with D0 of the IV bus.

The order of the operation is:

Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

XOR the source data with the contents of the AUX register.

Shift the ALU result data data and read data from IV latch.

Merge the IV latch data with the ALU data.

Write result to IV bus.

Note that during the merge phase, the original values of the source field outside the masked field are preserved. The original contents of the destination field are lost.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7.
- S1: 2, 3.
- L: 1, 2, 3, 4, 5, 6, 7, 0.
- D0: 0, 1, 2, 3, 4, 5, 6, 7.
- D1: 2, 3.

Example:

XOR the contents of bits 4 to 7 of the IV byte at the left bank to the contents of the AUX register and move the least significant 4 bits of the sum to the most significant 4 bits of the IV byte at the left bank.

Assembler notation: XOR LIV7, 4, LIV3
Instruction word: 011 10111 100 10011

XOR, IV bus, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=3			S					L			D				
			S_1		S_0						D_1		D_0		

Operation: (S) ⊕ (AUX) → D

Description:
AND the L-bit field of the IV bus source data to the contents of the AUX register and write the result to the IV bus address register specified by D.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 selects the left bank;
S1 = 3 selects the right bank.

S0 specifies the bit which will be the least significant bit of the rotated input data field.

L specifies the length of the masked field that is to be processed.

D specifies the destination IV bus register.
D = 07 specifies the left bank address (IVL);
D = 17 specifies the right bank address (IVR).

The order of the operation is:
Read data from the IV bus.

Right rotate the input data as specified by S0.

Mask rotated data as specified by L and read the AUX register.

XOR the source data with the contents of the AUX register.

Propagate data through shift unit.

Propagate data through merge unit.

Write result to target IV bus address register.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7
- S1: 2, 3
- L: 1, 2, 3, 4, 5, 6, 7, 0
- D: 07, 17

Note that L = 0 selects an 8-bit field.

Example:
XOR the contents of bits 4 to 7 of the IV byte at the left bank with the contents of the AUX register and write the result of the sum to the right bank IV bus address.

Assembler notation: XOR LIV7, 4, IVR
Instruction word: 011 10111 100 01111

XEC Instructions – Op Code 4

XEC, Register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=4			S					I							
			S_1		S_0										

Operation: Execute the instruction at the current page address offset by I + (S). Return to the instruction following the XEC instruction unless an unconditional jump or a satisfied conditional jump is encountered.

Description:
Execute the instruction at the address formed by replacing the 8 least significant bits of the contents of the address register with the 8-bit sum of I and the contents of the register specified by S.

S specifies the source register.

I is the 8-bit integer value for address modification.

Only the least significant 8-bits of the address register can be changed by this instruction, so that a range of 256 addresses is available. This range of 256 addresses is termed the address page, determined by the five most significant bits of the address register. When the sum of (S) + I is greater than 255 (377 octal) only the least significant 8 bits are used; the overflow register is not changed.

The program counter is not altered by the XEC instruction, so that the original address within the page is retained. During the instruction to be executed, the program counter is incremented by one in the normal way to point to the instruction following the XEC instruction. However, if the executed instruction is a JMP or NZT, the program counter can be changed to the jump address and instruction execution does not return to the address following the XEC instruction.

The order of the operation is:

Copy the data from the source register

Propagate data through the right rotate unit

Propagate data through the mask unit

Form the 8-bit sum of the I field value and the source register contents

Modify the address register with the 8-bit sum

Valid operands:

- S: 0, 1, 2, 3, 4, 5, 6, 10, 11
- I: $0 \leq I \leq 255$

Example:
Execute the instruction whose address is given by replacing the least significant 8 bits of the contents of R3 and the octal integer 315.

Assembler notation: XEC 315 (R3)
Instruction word: 100 00011 11001101

XEC, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=4			S					L			I				
			S_1		S_0										

Operation:
Execute the instruction at the current page address offset by I + (S). Return to the instruction following the XEC instruction unless an unconditional jump or a satisfied conditional is encountered.

Description:
Execute the instruction at the current page address offset formed by replacing the 5 least significant bits of contents of the address register with the 5-bit sum of I and the contents of the IV bus field specified by S.

S1 specifies the bank of the IV bus which is the data source:
S1 = 2 specifies the left bank
S1 = 3 specifies the right bank

S0 specifies the bit which will be the least significant bit of the input data field after rotation.

I is the 5-bit integer value for address modification.

L specifies the length (number of bits) of the masked field. The maximum value of L that may be specified is L = 5.

Only the least significant 5 bits of the address register can be changed by this instruction, so that a range of 32 addresses is available. This range of 32 addresses is termed the address page, determined by the eight most significant bits of the address register. When the sum (S) + I is greater than 31 (37 octal) only the least significant 5 bits are used; the overflow register is not changed.

The program counter is not altered by the XEC instruction, so that the original address within the page is retained. During the instruction to be executed, the program counter is incremented by one in the normal way to point to the instruction following the XEC instruction. However, if the executed instruction is a JMP or NZT, the program counter can be changed to the jump address and instruction execution does not return to the address following the XEC instruction.

The order of operation is:

Read the IV bus data into the input latches

Rotate the input data field as given by S0

Mask off the least significant L bits

Add the masked field to 5-bit integer

Replace the least significant 5-bits of the contents of the address register with the 5-bit result of the add operation.

Valid operands:

- S0: 0, 1, 2, 3, 4, 5, 6, 7
- S1: 2, 3
- L: 1, 2, 3, 4, 5
- I: $0 \leq I \leq 37_8$

Example:

Execute the instruction whose address is given by replacing the least significant 5 bits of the contents of the address register with the sum of the octal integer 26 and the contents of bits 2,3 and 4 of the IV byte at the left bank.

Assembler notation: XEC @26 (LIV4,3)
Instruction word: 100 10100 011 10110

NZT Instructions – Op Code 5

NZT, Register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=5			S					I							
			S_1		S_0										

Operation: Jump if (S) ≠ 0

Description:
If (S) ≠ 0, jump to the address formed by replacing the 8 least significant bits of the contents of the address register and program counter with the value of the I field.

If (S) = 0, increment the program counter by one.

S specified the register which is the subject of the test.

I is the 8-bit integer for address modification.

The order of operation is:

Copy the contents of the source register

Test the register contents for all zeros

If the contents are not zeros, replace the least significant 8 bits of the contents of the address register and program counter with the value of the I-field.

If the contents are zeros, increment the program counter by one

Valid operands:

- S: 0, 1, 2, 3, 4, 5, 6, 10, 11
- I: $0 \leq I \leq 255$

Example:
Jump to address 5432_{16} if the content of R6 is not zero, and the current address is $54XX_{16}$

Assembler notation: NZT R6, $32
Instruction word: 101 00110 00110010

NZT, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=5			S					L			I				
			S_1		S_0										

Operation: If (S) ≠ 0, jump to the address formed by replacing the 5 least significant bits of the contents of the address register and program counter with the value in the I-field.

Description:
If the contents of the L-bit field of the IV bus source data is non-zero, insert the value of the 5-bit I-field into the 5 least significant bits of the address register and program counter. If the contents are zeros, the program counter is incremented by one.

S1 specifies the bank of the IV bus which is the data source
S1 = 2 specifies the left bank
S1 = 3 specifies the right bank

S0 specifies the bit which will be the least significant bit of the input data field after rotation.

I is the 5-bit integer value for address modification.

L specifies the length (number of bits) of the masked field.
Note that L=0 specifies an 8-bit field

The order of operation is:

Read the IV bus data into the input latches

Rotate the copied input data until bit S0 becomes the LSB

Mask off the least significant L bits

Test the contents of the masked field

If the contents of the masked field are non-zero, replace the 5 least significant bits of the program counter and address register with the value of the I-field

If the contents of the masked field are zero, increment the program counter by one

Valid operands:

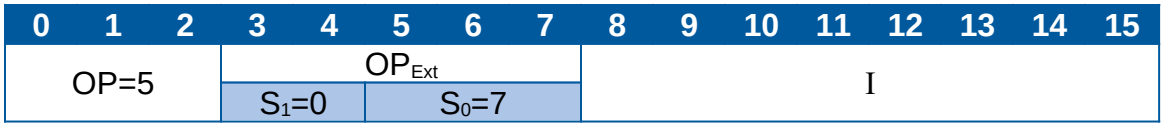
- S0: 0, 1, 2, 3, 4, 5, 6, 7
- S1: 2/3
- L: 1, 2, 3, 4, 5, 6, 7, 0
- I: $0 \leq I \leq 37_8$

Example:
Jump to address 115_8 if the content of bit 5 of the IV byte at the left bank is not zero.

Assembler notation: NZT LIV5, 1, 15H
Instruction word: 101 10101 001 01101

CALL Instruction – Opcode 5

Format:



Operation:
{AUX, I} → PC (And push PC + 1 to the call stack)

Description:
Sets the high 8 bits of the Program Counter to the value stored in the AUX register, and the low 8 bits of the Program Counter to the supplied immediate value, I.

OP_{Ext} is a 5-bit field that extends the Opcode and must be set to @07

I is the 8-bit field specifying the low 8 bits of the address to call.

The address of the instruction following the call (PC + 1) is saved by pushing it on to the 8 level hardware call stack.

Order of operation:

Read the contents of the AUX register

Propagate the contents of the AUX register through the Right Rotate unit

Propagate the contents of the AUX register through the Mask unit

Propagate the contents of the AUX register through the ALU

Set the PC register using the contents of the AUX register and the supplied I field.

Valid operands:

- OP Ext: 07_8
- I: $0 \leq I \leq 255$

Example:
Call the subroutine at address $1234

Assembler notation: CALL $34 *(Assumes the value stored in AUX is $12)*

Instruction word: 101 00111 00110100

RETURN Instruction – Op Code 5

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=5			OP_{Ext}					(Reserved) = 0							
			S_1=1												
					S_0=7										

Operation:
Pops a return address from the call stack and sets the Program Counter to its value.

Description:
Returns to the address stored by the last CALL instruction.
Note: A RETURN instruction that occurs when the call stack is empty has undefined behavior.

OP_{Ext} is a 5-bit field that extends the Opcode and must be set to @17

Bits 8 through 15 are reserved for future use, and should be set to 0.

Order of operation:

Pop a return address from the hardware call stack

Write the address to the Program Counter

Valid operands:

- OP Ext: 17_8

Example:
Return from a subroutine

Assembler notation: RET
Instruction word: 101 01111 00000000

XMIT Instructions – Op Code 6

XMIT, Register

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=6			D					I							
			D_1		D_0										

Operation: I → D

Description:
Store the value of the 8-bit integer in the register specified by D.

D specifies the register to be loaded.

I is the 8-bit field containing the value to be loaded into the register.

Valid Operands:

- D: 0, 1, 2, 3, 4, 5, 6, 11
- I: $0 \leq I \leq 377_8$

Example:
Set the value 377_8 in the AUX register.

Assembler notation: XMIT @377, AUX
Instruction word: 110 00000 11111111

XMIT, IV bus address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=6			D					I							
			D_1		D_0										

Operation: I → D

Description:
Enable the IV byte, at the bank specified by D, whose address is the 8-bit integer I.

D specifies the destination bank of the IV bus for the address data:
D = 07 specifies the left bank address
D = 17 specifies the right bank address

I is the 8-bit field specifying the address of the byte to be enabled.

The order of operation is:
Copy the 8 least significant bits of the instruction word

Output the 8-bit field to the IV bus as an address on the bank specified by D

Valid operands:
- D: 7, 17
- I: $0 \leq I \leq 377_8$

Example:
Enable the IV byte at the left bank whose address is 53_8

Assembler notation: XMIT @027, IVL
Instruction operation: 110 00111 00101011

XMIT, IV bus

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=6			D					L			I				
			D_1		D_0										

Operation: I → D

Description:
Transmit the least significant L bits of the I field to the L-bit field of the IV bus specified by D. If L is greater than 5 bits, the most significant bits of the destination field are filled with zeros.

D1 specifies the bank of the IV bus which is the destination:

D1 = 2 selects the left bank

D1 = 3 selects the right bank

L specifies the length of the destination field (number of bits).
Note that L = 0 selects an 8-bit field

D0 specifies the bit position in the IV bus with which the least significant bit of the I field data should be aligned. This means that the I field data is left-shifted so that bit 7 is aligned with bit D0 of the IV bus.

The order of operation is:

Read the contents of the destination IV byte into the input latches

Copy the least significant 5 bits of the instruction word

Left shift the copied 5-bit field as specified by D0

Merge the shifted field, as specified by L, with the contents of the IV latches and output the result to the IV bus.

Valid operands:

- D0: 0, 1, 2, 3, 4, 5, 6, 7
- D1: 2, 3
- L: 1, 2, 3, 4, 5, 6, 7, 0
- I: $0 \leq I \leq 37_8$

Example:
Transmit the value 6 to bits 2, 3, 4, and 5 of the IV byte at the left bank.

Assembler notation: XMIT $06, LIV5, 4
Instruction word: 110 10101 100 00110

JMP Instruction – Op Code 7

JMP, address

Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP=7			A												

Operation:
Set the value in the A field into the program counter and address register.

Description:
Jump to the instruction address specified by the A field, and continue normal program execution from the address. The contents of the 13-bit A field are loaded into the program counter and address register. Then next instruction to be executed is then the instruction at then new address.

A is the 13-bit field specifying the address to which the jump is made.

The order of operation is:

Load the address register and program counter with the contents of the A field

New address value is used for next instruction

Valid operands:

- A: $0 \leq A \leq 17777_8$ (8191_{10})

Example:
Jump to the address 512_8

Assembler notation: JMP @512
Instruction word: 111 0000101001010