bcc	title	prefix/class-number.revision
	THE REMOTE PROCESSOR UNIT	RPU/S-33

checked	authors	approval date / revision date
	Paul Heckel	5/1/70 /
checked: Judith Behrinc		classification: Specification
approved: Mel		distribution: Company private / pages: 14

ABSTRACT and CONTENTS

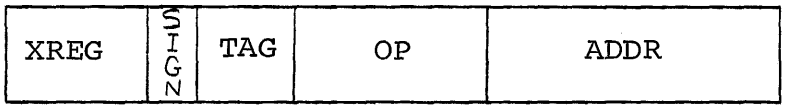
This document describes the Remote Processing Unit of the Data Concentrator (also known as the farout processing unit due to its location relative to M1). This is part of the Data Concentrator Microcode.

This document obsoletes Appendix I of RC/S-23.

THE REMOTE PROCESSOR UNIT

Instruction Format

Each instruction consists of 16 bits divided into four fields: the opcode field, the tag field, the index register field and the address field:



The first stage of the address computation forms a 16 bit Preliminary Address by adding (if sign = Ø) or subtracting (if sign = 1) the 6 bit address in the instruction (ADDR) from the 16 bit contents of the register specified by XREG. The tag field specifies one of the 4 types of addressing: indirect, direct, immediate, and scratchpad.

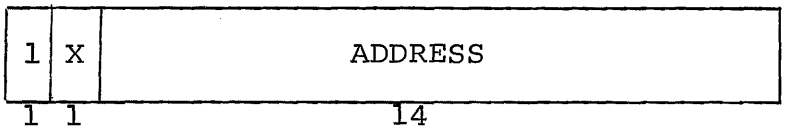
Direct Addressing: The Preliminary Address is the Effective Address of the instruction.

Immediate Addressing: The Preliminary Address is treated as the operand of the instruction. This makes no sense for stores or byte instructions.

Scratchpad Addressing: The Preliminary Address is the number of the scratchpad register being addressed.

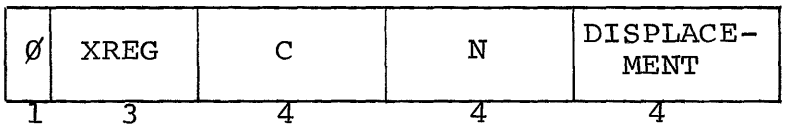
Indirect Addressing: The Preliminary Address is the address of an indirect word.

An indirect word may be of the following format:



The first bit must be a 1. The second bit may specify indexing by register 1. The address field specifies a 14 bit address. The Effective address is ADDRESS if X = Ø, ADDRESS + XR1 if X = 1.

Alternatively the format may be as follows to specify a field:



The displacement (4 bits, unsigned) is added to the contents of the register specified by XREG to specify the word to be addressed. A field of length L, and rightmost bit R (in a 16 bit word - fields may not cross word boundaries) is specified by setting the field N to 17-L, and the field C to 14-R mod 16. All of the previous numbers are decimal.

The effective Address is a word address for all instructions

except load and store byte. The contents of this word is called the operand. If addressing is immediate, the Preliminary Address is the operand. If addressing is indirect through a field descriptor, the bits selected by the descriptor are right justified in a word of zeros to form the operand. In the case of a store the L rightmost bits of the register being stored replace the bits selected by the field descriptor, and nothing else is affected.

Machine Registers

The machine has 7 addressable registers, which constitute the state of a RPU process.

0) Z register always contains zero
1) Index register
2) Program counter
3) Accumulator
4) Link
5) XR5
6) XR6
7) XR7

Machine Instructions

LXR: Load index register

BRU: Branch (load program counter)

LAC: Load accumulator

LRL: Load return link

LR5: Load LR5

LR6: Load LR6

LR7: Load LR7

The specified register is loaded with the operand of the instruction.

STA: Store

The contents of the accumulator is put into the location specified by the Effective Address. Immediate addres-

sing is illegal.

LDB: Load byte.

The Effective Address is treated as a byte address. I.e. the 15 high order bits are treated as a word address. The low order bit specifies the left (if zero) or right (if one) byte of the word. This instruction (and STB) are the only instructions that use byte addressing. Direct addressing and full word indirect addressing are the only types of addressing that can be used with LDB and STB.

Note that LDB* FOO treats FOO as a word address and the contents of FOO as a byte address, whereas LDB FOO treats FOO as a byte address.

The addressed byte is loaded into the right half of the accumulator, and the left half of the accumulator is cleared.

STB: Store byte.

The right half of the accumulator is stored in the byte addressed by the Effective Address. (see LDB)

ADM: Add to Memory.

The contents of the accumulator is added to the location addressed by the Effective Address. Immediate addressing is illegal.

ISZ: Increment and skip if zero.

The contents of the location addressed by the Effective Address is increased by 1 and if the result is zero the next instruction is skipped. Immediate addressing is illegal.

BSL: Branch and Save Link.

The location counter plus 1 is put into the Return Link (XR4) and the Effective Address is put into the location counter.

STL: Store Link.

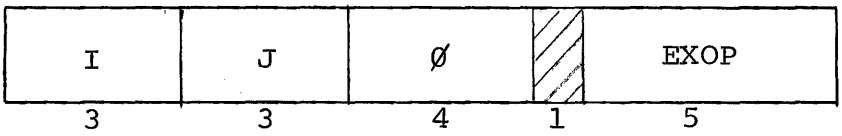
The Return Link (XR4) is stored in the location specified by the Effective Address.

Spare Not yet defined.

EXT: Extended Opcode.

The Extended Opcodes

The opcodes in the extended set are of the form:



The effective address of extended opcodes is ignored. Thus there are 32 possible extended opcodes, each of which specifies 2 registers: an I register and a J register.

Note: the eleventh bit is not used.

All instructions that modify a register will modify the J register.

ADD:

The contents of the I register is added to the contents of the J register.

SUB:

The contents of the I register is subtracted from the contents of the J register.

OR: Logical Or

The contents of the I register is Logically Ored into the J register.

AND: Logical And

The contents of the I register is Logically Anded into the J register.

FOR: Exclusive or

The contents of the I register is Exclusively Ored into the J register.

SKE, SKNE, SKG, SKL:

These four instructions will compare the I register with the J register and skip if the I register is equal to, not equal to, greater than, or less than the J register. Note that SKNG (I,J) = SKG (J,I) and SKNL (I,J) = SKL (J,I) thus allowing all six skips.

SKA, SKNA: Skip on And Zero, Skip on And Not Zero

SKA will logically AND the I register with the J register and skip if the result is zero. SKNA is identical except that it will skip if the result is not zero. Neither register is changed.

ISG: Increment and Skip if Greater

The J register is incremented by 1 and then compared with the I register. If the new J register is greater than the I register the instruction skips.

DSL: Decrement and Skip if Less

The J register is decremented by 1 and the instruction skips if the new J register is less than the I register.

LLB: Load Left Byte

The left byte of the I register is put into the right byte of the J register, the high order bits being cleared.

LRB: Load Right Byte

The right byte of the I register is loaded into the right byte of the J register, the high order bits being cleared.

ACTT: Activate Task

The task specified by the I register is activated (if J >= Ø) or deactivated (if J < Ø). If the current task is deactivated by this process it will stop computing.

LBRB: Load Byte from Ring Buffer

The I register contains a pointer to a ring buffer descriptor (diagram in Appendix II). A ring buffer descriptor is a 2 word entry. The second word (EB) points to the end of the ring buffer. The first word is split into two halves: the right half (RP) specifies the read pointer, and the left half (WP) specifies the write pointer.

EB is the address of the word of the ring buffer with the largest memory address. RP and WP are byte addresses relative to EB. Thus the byte last read

is at EB-RP and the byte last written at EB-WP. Byte Ø is not part of the buffer (i.e. RP or WP = Ø is not allowed), and it contains the address of the first byte of the buffer.

All this may be clarified by the following example. If memory contains

Memory Address	Left byte	Right byte
100	1 (WP)	5 (RP)
101	104	
102	'A'	'B'
103	'C'	'D'
104	'E'	5

then the buffer descriptor at 100 addresses the string 'BCDE'. 'A' is the byte last read, and 'E' is the byte last written. If we had

100	5 (WP)	2 (RP)

the descriptor would address the string 'EA'.

If RP = WP the buffer is empty. If writing another byte would cause RP = WP, the buffer is full, and the write will not be allowed.

LBRB is equivalent to a NOP (execute the next instruction) if the buffer is empty. Otherwise it changes the read pointer to point to the next byte, loads it into the J register, and skips the next instruction.

SBRB: Store Byte in Ring Buffer

If the buffer is full (incrementing the write pointer would cause it to equal the read pointer) the instruction is a NOP. Otherwise the write pointer is incremented, the right byte of the J register is stored, and the next instruction is skipped.

GOML: Goto Microprocessor Location

Control will go to the microprocessor location specified by the I register. A microcoded return will cause control to continue with the next RPU instruction.

LCY: Left Cycle

The J register is cycled left by the number specified by the contents of the I register.

POT: Output

The device specified by the I register is selected. The word in the J register is output to this device.

PIN: Input

The input from the device selected by the I register is put in the J register.

MBLK: Move Block

The block of length specified by the (I register Mod 256) is moved from the place specified by the J register to the place specified by X1.

APPENDIX I

Addressing:

- Direct Addressing 3
- Immediate Addressing 2
- Scratchpad Addressing 1
- Indirect Addressing 0

Opcodes:

- ADM: 5
- BRH: 16
- BSL: 3
- EXT: Ø
- ISZ: 4
- LAC: 15
- LDB: 7
- LRL: 14
- LR5: 13
- LR6: 12
- LR7: 11
- LXR: 17
- Spare: 1
- STA: 1Ø
- STB: 6
- STL: 2

Extended Opcodes:

- ACTT: 23
- ADD: Ø
- AND: 2Ø
- COPY: 3
- DSL: 14
- EOR: 16
- GOML: 27
- ISG: 7
- LBRB: 6
- LCY: 26
- LLB: 22
- LRB: 17
- MBLK: 25
- OR: 11
- PIN: 21
- POT: 24
- SBRB: 2
- SKA: 13
- SKE: 4
- SKG: 15
- SKL: 1Ø
- SKNA: 12
- SKNE: 5
- SUB: 1

APPENDIX II

Addressing:

Direct Addressing 3

Immediate Addressing 2

Scratchpad Addressing 1

Indirect Addressing Ø

Opcodes:

Ø EXT
1 Spare
2 STL
3 BSL
4 ISZ
5 ADM
6 STB
7 LDB
1Ø STA
11 LR7
12 LR6
13 LR5
14 LRL
15 LAC
16 BRH
17 LXR

Extended Opcodes:

Ø ADD
1 SUB
2 SBRB
3 COPY
4 SKE
5 SKNE
6 LBRB
7 ISG
1Ø SKL
11 OR
12 SKNA
13 SKA
14 DSL
15 SKG
16 EOR
17 LRB
2Ø AND
21 PIN
22 LLB
23 ACTT
24 POT
25 MBLK
26 LCY
27 GOML

APPENDIX III

DATA FORMATS

Ring Buffer Pointer