MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper 80 November 1974

CONS

Tom Knight

DRAFT: Comments and corrections, technical or typographical, are solicited.

This work was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract number N00014-70-A-0362-0003

DRAFT

Acknowlegement

Ideas for this design were taken from the Xerox Parc ALTO microprocessor, the DEC PDP-11/40, and the 11/40 extensions from CMU. Sam Fuller from CMU suggested the instruction modification technique. Locally, Jack Holloway contributed ideas on the timing of scratchpad references, and Richard Greenblatt has helped make sure the machine will do enough to perform its intended work. Pitts Jarvis wrote the microcode assembler and assisted in documenting the instruction formats.

0.0 Overview

The CONS microprocessor is a general purpose processor designed for easy programming of list manipulation and emulation of complex order codes. It is the central processor in the lisp machine project, where it emulates the 16 bit order code produced by the lisp compiler. It is organized around three 32 bit data paths: an A source, an M source, and an output bus. Source specifications and functions applied to the data are entirely under control of a 42 bit microcode word. The machine has the capability of addressing up to 4K of writable microprogram, although the first version has only 2K provided.

There are four micro-instructions defined. Each specifies sources for the A and M busses, and optionally a destination. The four operations are:

1) an ALU operation which performs adds, subtracts, and boolean operations

2) a byte operation which performs byte extraction and deposit, as well as selective field deposit.

3) a conditional transfer instruction, conditional on the value of any bit accessible to the M bus, or the carry and equal flags of previous ALU operations

4) a dispatch, which allows field extraction, masking, and dispatching to assigned locations depending on the resulting field.

There are several special sources and destinations whose loading and use invokes special action by the microprocessor. These include the memory address and memory data registers, whose use initiates main memory cycles.

Several of the ALU operations are conditionally of two forms, depending upon the low order bit in the Q register. These operations are used for MUS and DIS (multiply step and divide step).

The main features of this machine which make it suitable for interpreting the Lisp Machine order code are its writable microcode, its very flexible dispatching and subroutining, and its excellent byte manipulation abilities. A conscious attempt has been made to avoid features that are special purpose. The goal is a machine that happens to be good at emulating this particular order code. Hopefully, it can emulate others almost as well.

Stacks...

virtual memory...

0.1 Notation

All numbers in this manual are decimal, unless followed by a single quote (') implying octal. Arithmetic is done in two's complement. Bits in the registers and on busses are referenced by integers in brockets <> following the register name, counting from the right as if the register or bus were 32 bits wide. Thus O<31> is the sign bit of the output bus, and PC<11> is the high order bit of the microprogram counter.

Since the use of the term "micro" in refering to registers and instructions becomes redundant, its usage will be dropped from here on in the manual. All instructions discussed are microinstructions.

1.0 Control

The control section of the processor consists of a 12 bit program counter (the PC), a 32 location PC stack (SPC), and a 1K dispatch memory (DPC), used during the dispatch instruction. Unlike some micro-processors, and like most traditional machines, the normal mode of operation is to execute the next sequential instruction.

Two op codes affect flow of control in the machine. The conditional jump specifies a new PC and transfer type in the jump instruction, while the dispatch instruction looks up the new PC and transfer type in the dispatch table (DPC). In either case, the new PC is loaded into the PC register, and the operation specified by the three bit transfer type is performed. These operations are:

N bit - if on, inhibits execution of the (physically) next instruction.

The cycle that would have executed that instruction is wasted.

P and R bits are decoded as follows:

P	R	Effect
0	0	JUMP (no return saved)
1	0	CALL (save PC+2 on the SPC stack)
0	1	RETURN (ignore new PC, pop a PC off the SPC and load it into the PC register
1	1	WRITE (write xxx into the microcode memory at the address of the PC specified in the instruction

The JUMP transfer type is the normal program transfer, without saving a return address.

The CALL transfer type pushes the current PC, plus two, onto the SPC stack. This stack is 32 locations long. It is the responsibility of the programmer to avoid overflows.

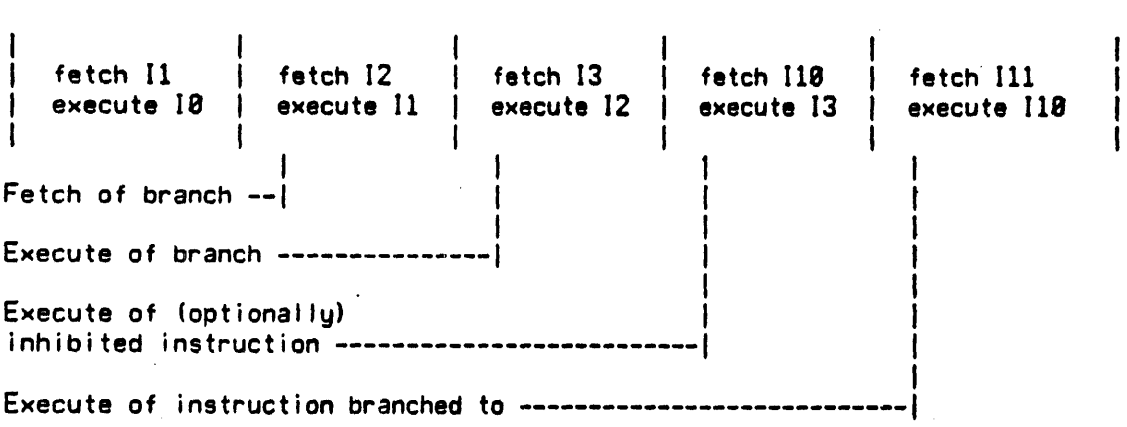
The RETURN transfer type pops a return PC from the SPC stack and uses it in place of the PC specified in the instrucion or dispatch table.

The WRITE transfer type is the mechanism for writing instructions into the microprogram instruction store. The reason for its odd location in the instruction set is due to the way in which it operates. It causes the same operations as the CALL transfer type, resulting in the PC register being loaded with the address to be modified. Then, when the instruction RAM would normally be fetching the instruction to be executed from that location, a write pulse is generated, causing the data present in xxx to be written into the memory. Meanwhile, the IR latches are loaded

with a RETURN transfer instruction, causing instruction execution to proceed from where it left off. Note that this instruction requires use of a word on the SPC stack.

The processor uses single instruction look ahead, i.e. the lookup of the next instruction is overlapped with execution of the current one. This implies that transfer and dispatch instructions normally execute the following instruction, even if the branch was successful. Provision is made in these instructions to inhibit this execution (with the N bit), but the cycle it would have used will then be wasted.

I2 is a branch instruction to location of I10

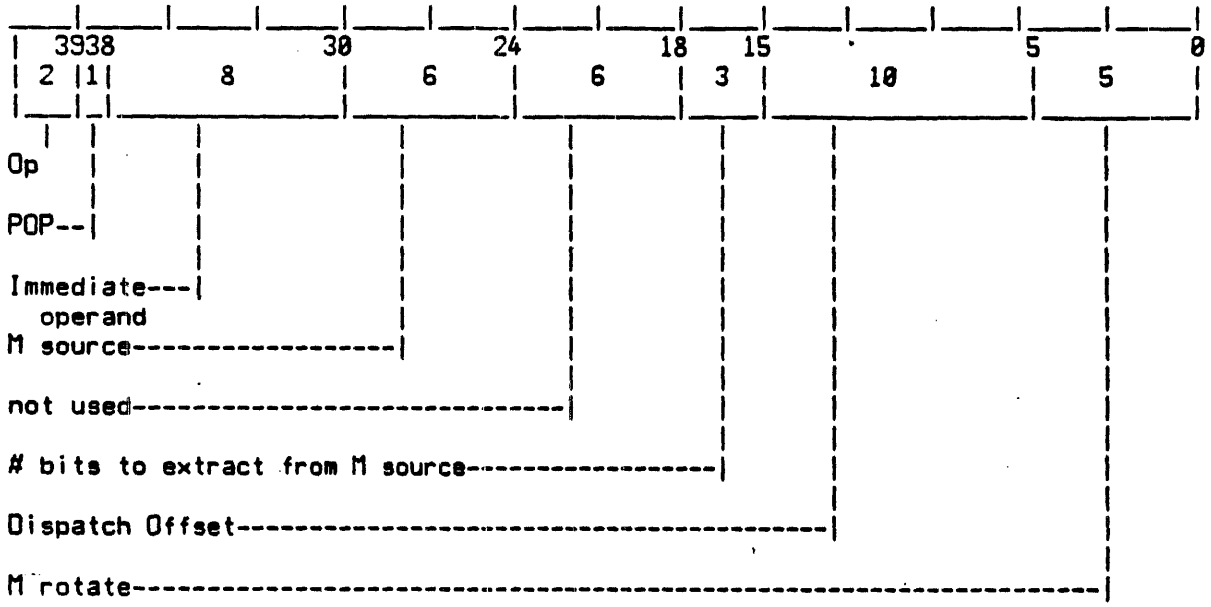


An additional bit in each instruction (the POP bit) allows specification of simultaneous execution of a POP transfer type along with execution of any instruction. (i.e. it does the same thing as if this instruction, in addition to whatever else it does, executes a POP transfer type jump without the N bit on) It is the responsibility of the programmer to avoid obvious conflicts in the use of this bit simultaneously with other types of transfers.

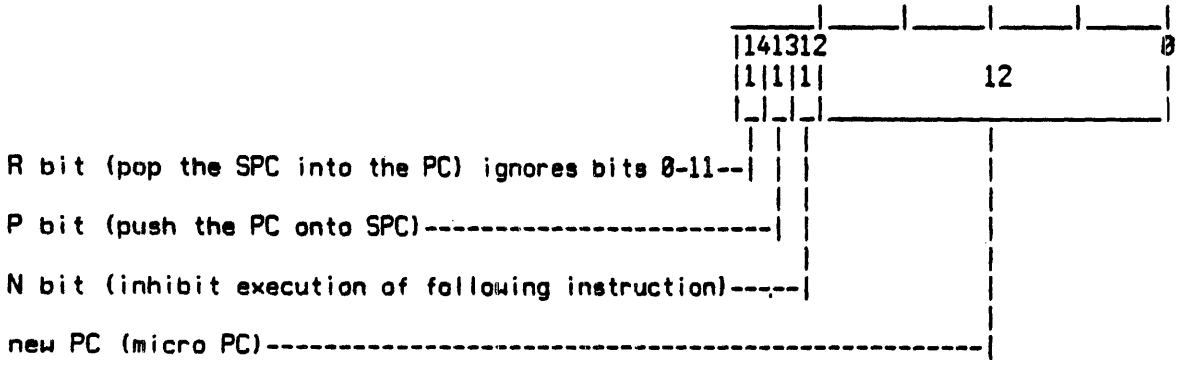
1.1 Dispatching

The dispatch instruction allows selection of any source available on the M multiplexor [see description of M bus sources in the Data Path section], and the dispatch on any sub-field of up to 8 bits from the selected word. The selected subfield is ORed with the "dispatch offset" field of the instruction to produce a 10 bit address. This address is used to look up a 12 bit PC and 3 bit transfer type in the dispatch ram.

Opcode 0 DISPATCH (DISPATCH (M-source (size bit-pos)) offset)



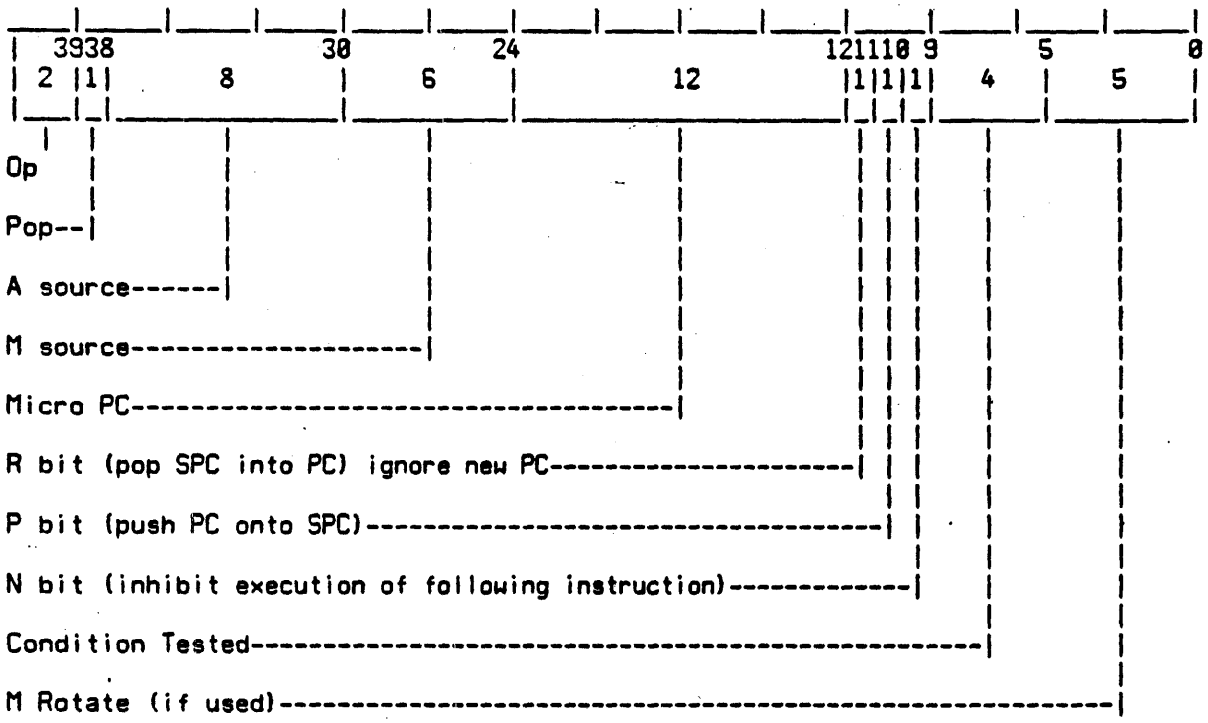
Dispatch RAM



1.2 Conditionals

The branch conditions are of two major types. First, it is possible to test the state of any bit accessible to the M multiplexor by specifying the source and a shift which will leave the tested bit in the low order bit position of the output bus. This allows testing of all the flag bits, since they are accessible from the M multiplexor. The second type of conditional is the arithmetic conditional, where two operands are specified, and an ALU op is performed, resulting in tests of the carry, zero, and overflow bits in the ALU. This is useful for comparing two numbers.

Opcode 1 JUMP (IF (condition A-source M-source) (opcode destination))



2.0 Data paths

The data paths of the machine consist of two source busses, which provide data to ALU and Byte extracter, and an output bus which is selected from the ALU (optionally shifted left or right) or the output of the Byte extracter. We first describe the specification of the source busses, which are identically loaded for all instruction, then the destination specifiers which control where the data is stored, and finally, the two operations for controlling the ALU and the Byte extracter.

2.1 Sources

All instructions specify sources in the same way. There are two source busses in the machine, the A bus and the M bus. The A bus is driven only from the A scratchpad memory of 256 locations. The M bus is driven from the M scratchpad of 32 locations, or from up to six other sources. Among these sources are the main memory data, the PC stack (for restoring the state of the processor after traps), a word of processor flag bits, and the Q register. Addresses for the A and M scratchpads are taken directly from the instruction. The alternate sources of data for the M source are specified with an additional bit in the M source field.

IR<37-30> = A source address

IR<29-24> = M source address
If IR<29> = 0
IR<28-24> = M scratchpad address
If IR<29> = 1
IR<28-24> = M multiplexor source
0 - M scratchpad (illegal)
1 - M scratchpad pass around path (illegal)
2 - Main memory data
3 - Q register
4 - Flag bits
5 - SPC data (12 bits),SPC pointer (5 bits)
Stack adr register (10 bits)
6 -
7 - Stack (Pop)
17 - Stack (indexed by stack adr register)

2.2 Destinations

The 10 bit destination field in the Byte and ALU instructions specifies where the result of the instruction is deposited. It is in one of two forms, depending upon the high order bit. The high order bit on indicates that the low order 8 bits are an address of an A memory scratchpad. If the high order bit is a zero, the remaining 9 bit field is divided into two fields, a 4 bit register select field, and a 5 bit M scratchpad address. Both of the registers specified by these fields get written.

```
IR<23-14> = destination
        If IR<23> = 1
                IR<21-14> = A scratchpad write address
        If IR<23> = 0
                IR<22-19> = Register write address
                        0 - none
                        1 - MA (read)
                        2 - MA (write now)
                        3 - MA (write wait for data)
                        4 - Memory output data
                        5 - stack (push)
                        6 - stack (index of pointer)
                        7 - Stack pointer register
                       10 - SPC data write
                       11 - Instruction modification (first half)
                       12 - Instruction modification (second half)
```

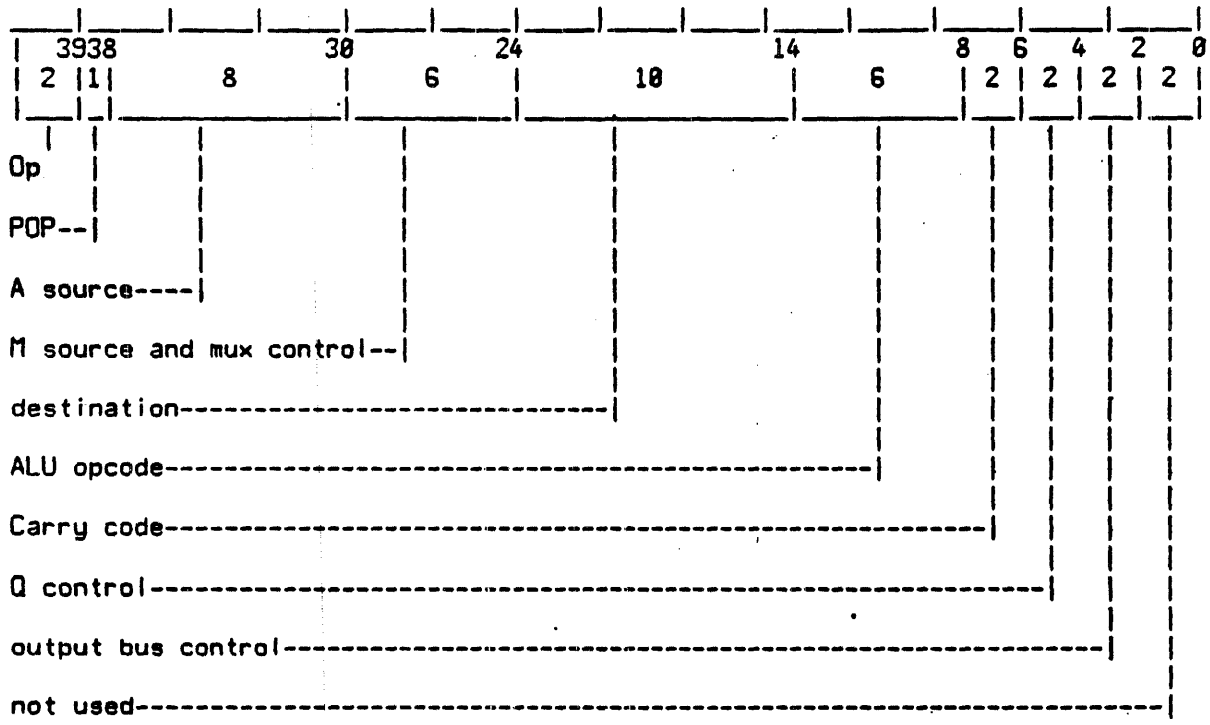
The conditional branch and dispatch instructions have no destination field.

2.3 ALU operations

The ALU operation performs most of the arithmetic in the machine. It specifies two sources of 32 bit numbers, and an operation to be performed by the ALU. The operation can be any of the 16 booleans, two's complement add, subtract (in one direction only), left shift, and several less useful operations. The carry into the ALU can be from one of four sources, a one, a zero, the high order bit of the Q register, or the last carry out in an ALU instruction. Additionally, the ALU op specifies one of four operations upon the Q register. These are do nothing, shift left, shift right, and load from the output bus. An additional bit in the ALU operation field is decoded to indicate variable operations, and the operation performed with this bit set is determined partially by the low order bit in the Q register. This is how the MUS and DIS instructions are specified for bitwise multiplication and division.

IR<40-39> = 2
IR<38> = POP transfer
IR<37-30> = A memory source
IR<29-24> = M memory source and M mux control
IR<23-14> = Destination
IR<13-8> = ALU op
        If IR<13> = 0
                IR<12-8> = ALU op code
        If IR<13> = 1
                IR<12-8> = Conditional ALU op code
IR<7-6> = Carry code
        0 - carry zero
        1 - carry one
        2 - carry from carry out of last ALU op
        3 - carry from low order bit of Q
IR<5-4> = Q control
        0 - do nothing
        1 - shift Q left
        2 - shift Q right
        3 - load Q from output bus
IR<3-2> = output bus control
        0 - ALU output
        1 - ALU output shifted right one
        2 - ALU shifted left one
        3 - Masker output (not particularly useful)

Opcode 2 ALU (opcode (A-source B-source) destination)



ALU bit operation functions (from Table 1 of 74181 specifications)
(number in parentheses after arithmetic opcodes is the low order carry in)
(all arithmetic operations are two's complement)

	boolean	arithmetic
0	setca	inca(0)
1	andcb	
2	andca	
3	setz	
4	orcb	
5	setcm	
6	xor	sub(1)
7	andcm	
10	orca	
11	eqv	add(0)
12	setm	
13	and	
14	seto	lsha(0)
15	orcm	
16	or	
17	seta	deca(0)

2.4 Byte operations

The byte operation specifies two sources and a destination in the same way as the ALU operation, but the operation performed is one of selective insertion of a byte field of the M source into an equal length field in the A source. The rotation of the M source is specified by the SR bit as either zero or equal to the contents of the ROTATE field. The rotation of the mask used to select the bits replaced is specified by the MR bit as either zero or equal to the contents of the ROTATE field. The length of the mask field used for replacement is specified in the LENGTH field. The four states of the SR and MR bits yield the following states:

MR=0 SR=0 : not useful (subset of other modes)

MR=0 SR=1 : PDP-10 LDB instruction (except the unmasked bits are from the A source)

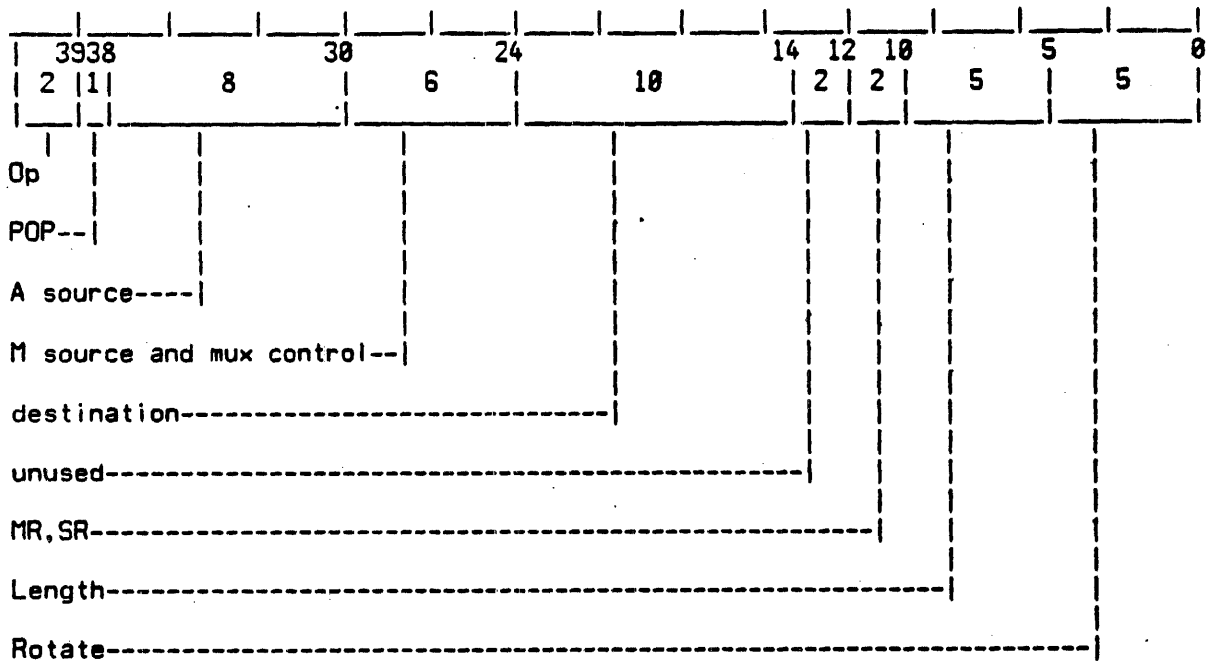
MR=1 SR=0 : PDP-10 DPB instruction

MR=1 SR=1 : Selective deposit of the masked field from one word into the same length and position byte in the second word.

Byte operations automatically assert output bus mux source from the masker output.

IR<40-39> = 3
IR<38> = POP transfer
IR<37-30> = A memory source
IR<29-24> = M memory source and M mux control
IR<23-14> = Destination
IR<13-11> unused
IR<11> = SR
IR<10> = MR
IR<9-5> = Length of mask byte
IR<4-0> = Rotation of mask or M source

Opcode 3 BYTE (BYTE (size (A-source bit-pos) (M-source bit-pos)) destination)



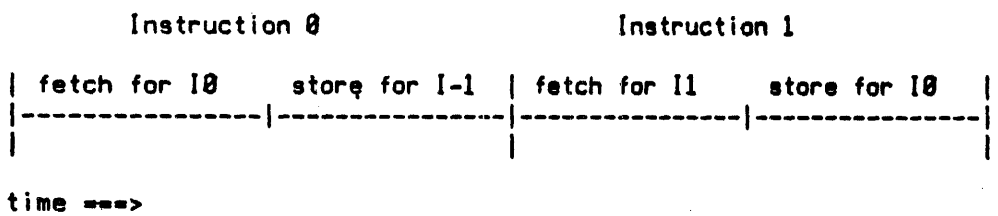
3.0 Program Modification

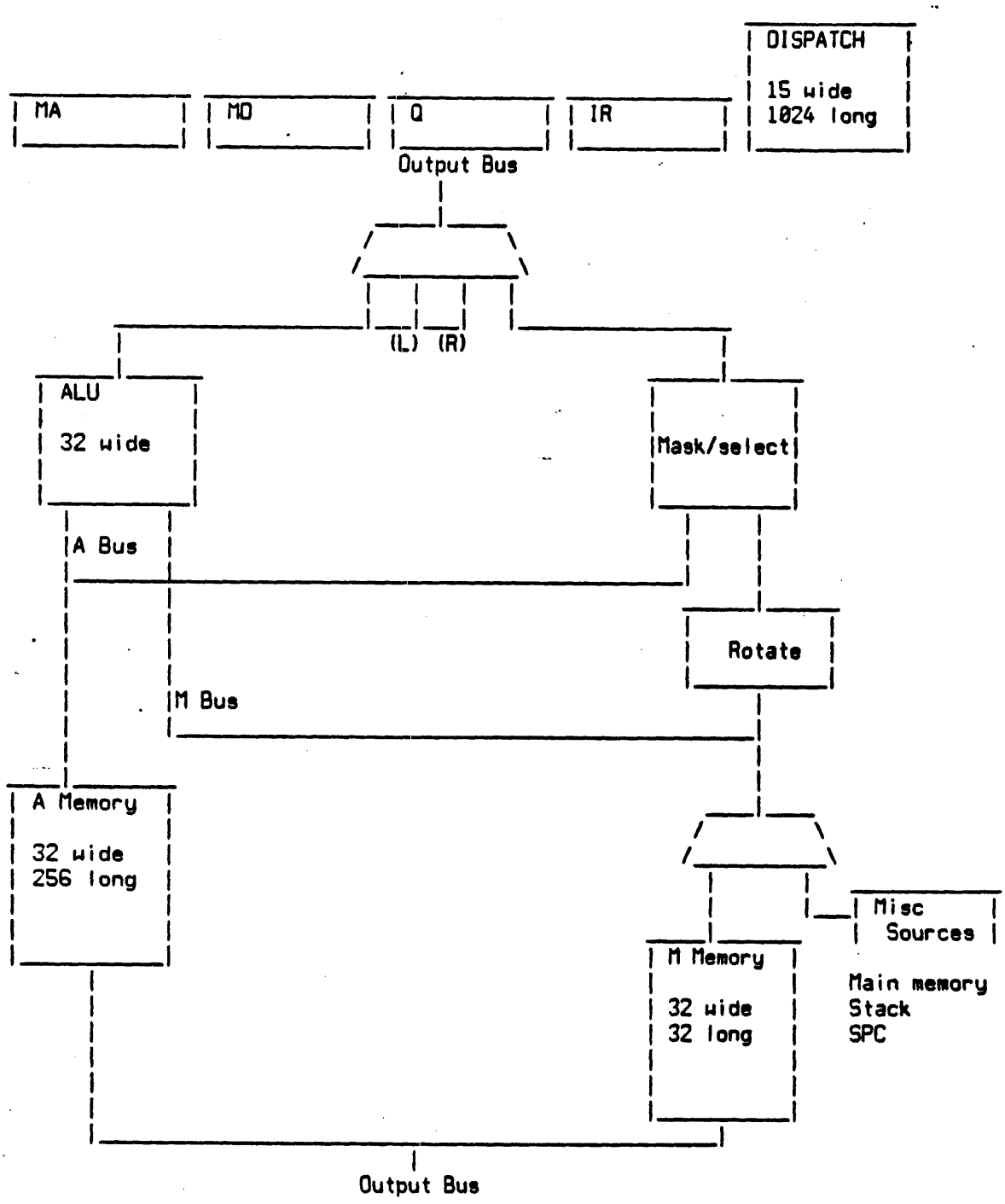
A novel technique is used for variabilizing fields in the program instruction. One of the "destinations" of the output bus is a (conceptual) register, whose contents get ORed with the next instruction executed. Combined with the shifter/masker ability to move any contiguous set of bits into an arbitrary field, this feature provides, for example, variable rotates and the ability to use program determined addresses of registers.

4.0 Clocks

This processor uses only one clock, occuring at the end of every cycle. This clock loads output data into the designated registers, and a new PC and instruction is latched. The only events which do not take place synchronous with the clock are the control signals for the A and M scratchpads in the processor and the PC stack. For these devices, a two stage cycle is performed. During the first phase, the source addresses of the respective devices are gated into the address registers. After the output data has settled, the outputs of these devices are latched. Then, the address is changed to that specified as the write location from the →previous← instruction. After the address has settled, a write pulse is generated for the scratchpad memory to perform the write. A pass-around path is provided (invisibly to the programmer) which notices and corrects read references to a location which was written into on the previous cycle, but not yet actually written into the scratchpad.

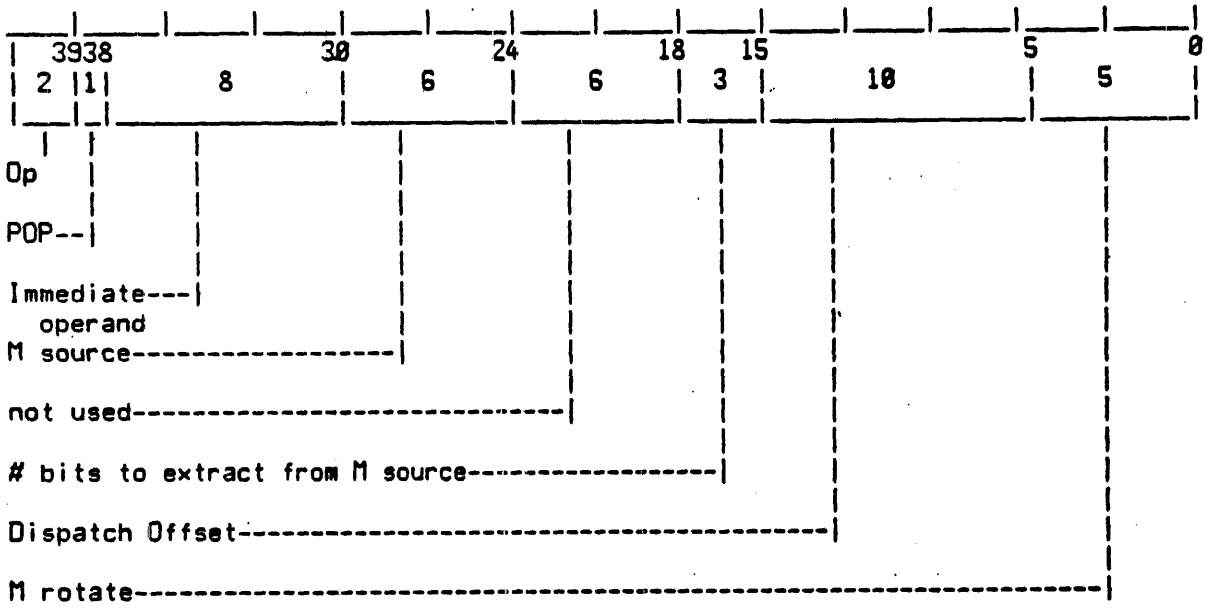
Timing of scratchpad references



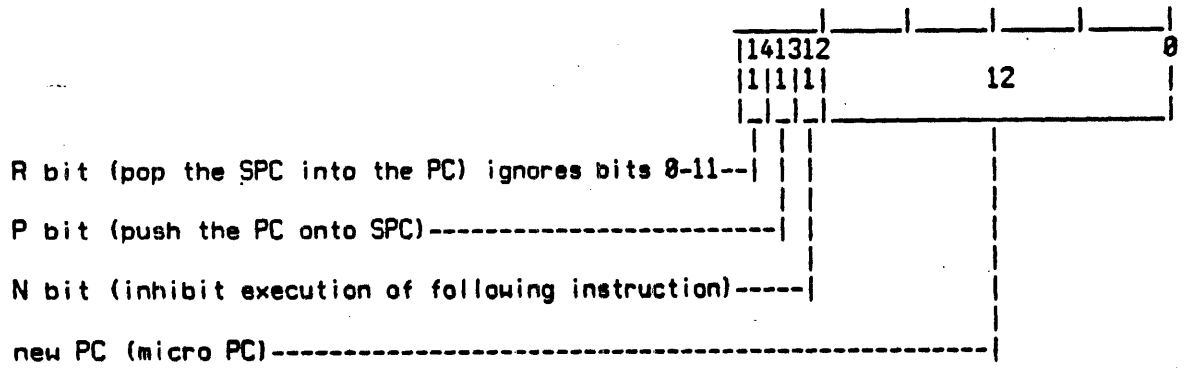

DISPATCH
15 wide
1024 long
MA
MD
Q
IR
Output Bus
(L)
(R)
ALU
32 wide
Mask/select
A Bus
Rotate
M Bus
A Memory
32 wide
256 long
Misc
Sources
M Memory
32 wide
32 long
Main memory
Stack
SPC
Output Bus

5.0 Word Formats

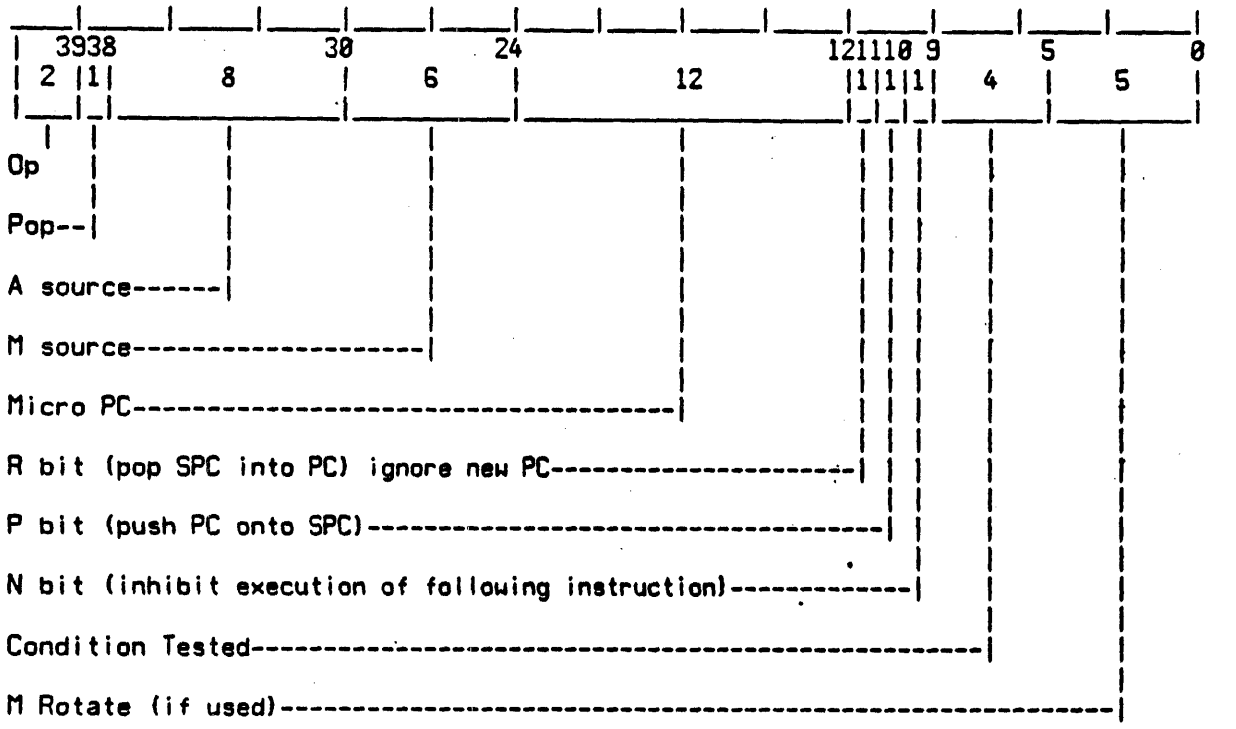
Opcode 0 DISPATCH (DISPATCH (M-source (size bit-pos)) offset)



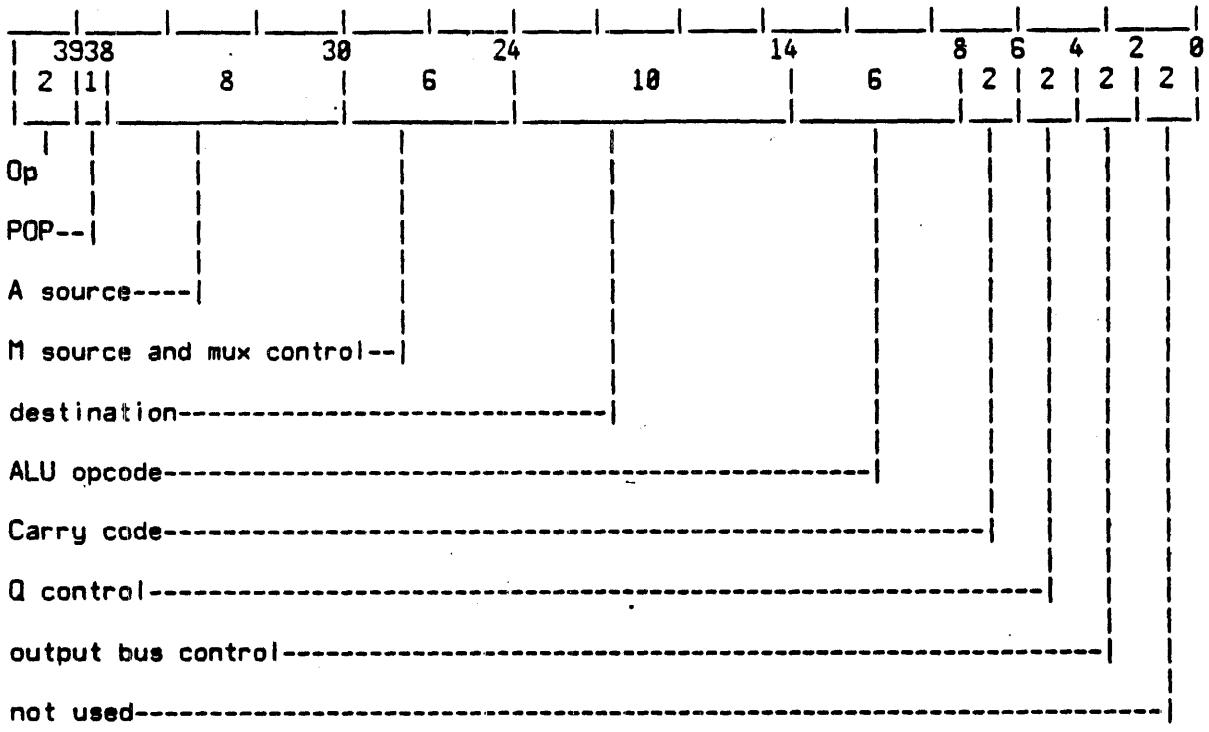
Dispatch RAM



Opcode 1 JUMP (IF (condition A-source M-source) (opcode destination))



Opcode 2 ALU (opcode (A-source B-source) destination)



ALU bit operation functions (from Table 1 of 74181 specifications)
(number in parentheses after arithmetic opcodes is the low order carry in)
(all arithmetic operations are two's complement)

	boolean	arithmetic
0	setca	inca(0)
1	andcb	
2	andca	
3	setz	
4	orcb	
5	setcm	
6	xor	sub(1)
7	andcm	
10	orca	
11	eqv	add(0)
12	setm	
13	and	
14	seto	lsha(0)
15	orcm	
16	or	
17	seta	deca(0)

Opcode 3 BYTE (BYTE (size (A-source bit-pos) (M-source bit-pos)) destination)