Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, Massachusetts 02139
617-253-7807
617-253-6765

CONS

Tom Knight

August 28, 1975 23:29

Draft Draft Draft

Comments and corrections, technical or typographical, are solicited.

DRAFT

Acknowledgement

Ideas for this design were taken from 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 maintained the Stanford drawing system and assisted in documenting the instruction formats. Thanks also are due to Dick Helliwell and the Stanford Foonly design crew who developed the drawing system without which this machine would be impossibly difficult to design.

0.0 Overview

The CONS microprocessor is a general purpose processor designed for convenient emulation of complex order codes, particularly those involving stacks and pointer manipulation. It is the central processor in the lisp machine project, where it interprets the bit-efficient 16 bit order code produced by the lisp compiler. The data paths of the machine are 32 bits wide. Each 42 bit wide micro-code instruction specifies two 32 bit data sources from a variety of internal scratchpad registers, and the two data-mainipulation instructions specify a destination address.

The internal scratchpads include a 1K pointer addressable RAM intended for storing the top of the emulated stack, in a manner similar to a cache. Since a large percentage of main memory references are to the stack, this should materially speed up the machine.

The machine has a 12 bit program counter, which behaves much like that of a traditional processor, allowing up to 4K of writeable microprogram memory. The control portion also includes a 32 location micro subroutine return stack.

Memory is accessed through a two level virtual paging system, which maps 23 bit virtual addresses into a 20 bit physical address.

There are four micro-instruction classes 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 substitution

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 upon the value in the resulting field

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

Two of the ALU operations are conditionally of two forms, depending upon the low order bit in the Q register and the sign of the previous ALU result. 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, its excellent byte manipulation abilities, and its internal stack storage. A conscious attempt has been made to avoid features that are special purpose. The goal is a machine that happens to be good at interpreting this particular order code. Hopefully, it can interpret others almost as well.

Since the use of the term "micro" in refering to registers and instructions becomes redundant, its use 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, 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.

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

TIME ===>

|              |              |              |              |              |
|  fetch I1    |  fetch I2    |  fetch I3    |  fetch I10   |  fetch I11   |
|  execute I0  |  execute I1  |  execute I2  |  execute I3  |  execute I10 |
|              |              |              |              |              |
                 |              |              |              |
Fetch of branch --|              |              |              |
                                |              |              |
Execute of branch ---------------|              |              |
                                               |              |
Execute of (optionally)                        |              |
inhibited instruction --------------------------|              |
                                                              |
Execute of instruction branched to ----------------------------|
```

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) [PC+1 if the N bit is also on]
0	1	RETURN (ignore new PC, pop a PC off the SPC and load it into the PC register)
1	1	in JUMPS: WRITE (write contents of specified A and M scratchpads into to the microcode memory at the address of the PC specified in the instruction) in DISPATCHES: FALL through to next instruction (don't dispatch)

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

The CALL transfer type pushes the current PC, plus two, [plus 1 if the N bit is also on] 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 instruction 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 fetched from the A and M scratchpads to be written into the memory. It is required that the data be fetched from the scratchpads, and not some other source, including any of the pass around paths that might be in use. Meanwhile, the machine simulates 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 and requires an extra cycle. It is highly recommended that the N bit also be on during this instruction, since the processor will be executing a POP transfer type unconditionally during what should be the execution of the intruction following the write. If this does not conflict with other things that this following instruction specifies, then it may be executed. Care is required.

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 7 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 2 DISPATCH (DISPATCH (M-source (size bit-pos)) offset)

```
____|_____|_____|____|____|_____|____|____|____|____|_____|____|_____|____|
|  3938            30        24               14  12  10   8   5         0
| 2 |1|       8      |    6     |       10       | 2 | 2 | 2 | 3 |   5    |
|___|_|______________|__________|________________|___|___|___|___|________|
  |  |      |            |              |          |   |   |   |     |
Op   |      |            |              |          |   |   |   |     |
     |      |            |              |          |   |   |   |     |
POP--|      |            |              |          |   |   |   |     |
            |            |              |          |   |   |   |     |
Dispatch ---|            |              |          |   |   |   |     |
  constant               |              |          |   |   |   |     |
M source-----------------|              |          |   |   |   |     |
                                        |          |   |   |   |     |
Dispatch Offset-------------------------|          |   |   |   |     |
                                                   |   |   |   |     |
not used-------------------------------------------|   |   |   |     |
                                                       |   |   |     |
Misc Function------------------------------------------|   |   |     |
                                                           |   |     |
not used---------------------------------------------------|   |     |
                                                               |     |
# of bits to extract from M source-----------------------------|     |
                                                                     |
M rotate-------------------------------------------------------------|
```

Dispatch RAM

```
                                                  _____|_____|____|____|____|
                                                  |141312                   0
                                                  |1|1|1|        12         |
                                                  |_|_|_|___________________|
                                                   | | |         |
R bit (pop the SPC into the PC) ignores bits 0-11--| | |         |
                                                     | |         |
P bit (push the PC onto SPC)-------------------------| |         |
                                                       |         |
N bit (inhibit execution of following instruction)-----|         |
                                                                 |
new PC ----------------------------------------------------------|
```

The dispatch constant field is loaded into the dispatch constant register (acessible from the M multiplexer) on every dispatch instruction.

The M rotate field contains the rotate necessary to move the field being dispatched on into the low order bits of the word. The rotate is to the left.

The miscellaneous functions are:

Function 2: Load the dispatch ram. Inhibits the normal action of the instruction and instead loads the dispatch ram with the low order contents of the M memory scratchpad location specified in the source. The parity (bit 15) is also loaded, and it is the responsibility of the programmer to load correct parity into the ram. Normal addressing of the RAM is in effect, so you probably want the # of bits field equal to zero to uniquely address a particular location.

Function 3: Force the high order bit of the rotate field equal to one if the low order bit of the emulated PC is one.

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 subtract is performed, resulting in tests of the carry, zero, and overflow bits in the ALU. This is useful for comparing two numbers. Other miscellaneous conditionals test the validity of the currently loaded main memory address with the memory map, and test for external interrupt conditions.

<u>Opcode 1 JUMP</u>   (IF (condition A-source M-source) (opcode destination))

```
____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
|  3938           30        24                  12 10 9 8 7 6 5        0
| 2 |1|     8      |    6    |        12         | 2 |1|1|1|1|1|   5   |
|___|_|____________|_________|___________________|___|_|_|_|_|_|_______|
  |  |      |           |              |           |  | | | | |    |
Op   |      |           |              |           |  | | | | |    |
     |      |           |              |           |  | | | | |    |
Pop--|      |           |              |           |  | | | | |    |
            |           |              |           |  | | | | |    |
A source----|           |              |           |  | | | | |    |
                        |              |           |  | | | | |    |
M source----------------|              |           |  | | | | |    |
                                       |           |  | | | | |    |
new PC---------------------------------|           |  | | | | |    |
                                                   |  | | | | |    |
Misc Function--------------------------------------|  | | | | |    |
                                                      | | | | |    |
R bit (pop SPC into PC) ignore new PC-----------------| | | | |    |
                                                        | | | |    |
P bit (push PC onto SPC)--------------------------------| | | |    |
                                                          | | |    |
N bit (inhibit execution of following instruction)--------| | |    |
                                                            | |    |
Invert Sense of test if 1-----------------------------------| |    |
                                                              |    |
Test Bit (0) or Test Condition (1)----------------------------|    |
                                                                   |
M Rotate (if Test Bit) or Condition number-------------------------|
```

The tested conditions are:

0 illegal 1 M<A M source less than A source 2 M>A M source less than or equal to A source 3 M=A A and M sources are equal 4 interrupt 5 pager fault 6 pager fault or interrupt 7 always true

The shift field contains the number of bits necessary to rotate the M source left

such that the tested bit is in bit position zero. That is, to test the sign bit, it should contain a 1.

The miscellaneous functions are:

Function 3: Force the high order bit of the shift field on if the low order bit of the emulated PC is a one.

2.0 Data paths

The data paths of the machine consist of two source busses, which provide data to the 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), the internal stack buffer, 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
                       12 - Main memory write data
                       22 - VMA
                       32 - Memory map data
                        3 - Q register
                        4 - PDL INDEX <9-0> PDL POINTER <19-10>
                        5 - SPC <17-0>, SPC pointer <31-27>
                       15 - as above, but also causes a pop of the SPC pointer
                        6 - Dispatch constant <7-0>
                        7 - PDL (pointer) POP
                       17 - PDL (pointer)  no POP
                       27 - PDL (index)
```

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 location. 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 - Memory data
                         2 - Memory data (write now)
                         3 -
                         4 - VMA
                         5 - VMA (read)
                         6 - VMA (write now)
                         7 - VMA (write map)
                        10 -
                        11 - PDL (pointer), push
                        12 - PDL (index)
                        13 - PDL index
                        14 - PDL pointer
                        15 - SPC (causes a push)
                        16 - next instruction modifier, bits <44-24>
                        17 - next instruction modifier, bits <23-0>
                IR<18-14> = M scratch pad write address
```

Note: The Q-register is loaded using the ALU instruction.

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 forced to a one or zero. 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> = 0
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<13-12> = output bus control
        0 - masker output (illegal)
        1 - ALU output
        2 - ALU output shifted right one (sign shifted in)
        3 - ALU shifted left one (high order bit of Q shifted in)

IR<11-10> = Misc Function
        0 -
        1 -
        2 -
        3 - Load low order bit of emulated PC from ALU low order bit output

IR<9-4> = ALU op
        If IR<9> = 0
                IR<8-4> = ALU op code
        If IR<9> = 1
                IR<8-4> = Conditional ALU op code
                        0 => multiply step
                        1 => divide step
IR<3-2> = Carry code
        0 - carry zero
        1 - carry one

IR<1-0> = Q control
        0 - do nothing
        1 - shift Q left (ALU sign output shifted in)
        2 - shift Q right (ALU low order bit output shifted in)
        3 - load Q from output bus
        3 - Masker output (not particularly useful)
```

Opcode 0 ALU (opcode (A-source B-source) destination)

```
____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
|  3938          30        24                    14  12  10          4   2   0
| 2 |1|      8        |    6     |        10        | 2 | 2 |     6      | 2 | 2 |
|___|_|_______________|__________|__________________|___|___|____________|___|___|
  |  |       |             |            |             |   |       |        |   |
Op   |       |             |            |             |   |       |        |   |
     |       |             |            |             |   |       |        |   |
POP--|       |             |            |             |   |       |        |   |
             |             |            |             |   |       |        |   |
A source----|              |            |             |   |       |        |   |
                           |            |             |   |       |        |   |
M source and mux control--|             |             |   |       |        |   |
                                        |             |   |       |        |   |
destination-----------------------------|             |   |       |        |   |
                                                      |   |       |        |   |
output bus control------------------------------------|   |       |        |   |
                                                          |       |        |   |
Misc Function---------------------------------------------|       |        |   |
                                                                  |        |   |
ALU opcode--------------------------------------------------------|        |   |
                                                                           |   |
Carry code-----------------------------------------------------------------|   |
                                                                               |
Q control----------------------------------------------------------------------|
```

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	M		M + 1 (carry set)
1	(A M)		
2	(M) A		
3	0		
4	(M A)		
5	A		
6	M xor A	M - A - 1 (carry clear)	M - A (carry set)
7	M (A)		
10	(M) A		
11	M eqv A	M + A	M + A + 1 (carry set)
12	A		
13	M A	(M A) - 1 (carry clear)	
14	1	M + M (carry clear)	M + M + 1 (carry set)
15	M (A)		
16	M A		
17	M	M - 1 (carry clear)	

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 ZR 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 ZM 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 ZR and ZM bits yield the following operations:

ZR=0 ZM=0 :	not useful (subset of other modes)
ZR=0 ZM=1 :	PDP-10 LDB instruction (except the unmasked bits are from the A source)
ZR=1 ZM=0 :	Selective deposit of the masked field from one word into the same length and position byte in the second word.
ZR=1 ZM=1 :	PDP-10 DPB instruction

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> = ZM
IR<12> = ZR
IR<11-10> = Misc Function
        3 - force high order bit of the ROTATE field to a one if the low order bit
            bit of the emulated PC is a one.
IR<9-5> = Length of mask byte - 1  (i.e. zero => one bit)
IR<4-0> = Rotation of mask or M source
```

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

```
____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
| 3938            30          24                  14  12  10        5          0
| 2 |1|       8       |     6     |        10        | 2 | 2 |    5    |    5    |
|___|_|_______________|___________|__________________|___|___|_________|_________|
  |  |      |             |             |            |    |    |        |
Op   |      |             |             |            |    |    |        |
     |      |             |             |            |    |    |        |
POP--|      |             |             |            |    |    |        |
            |             |             |            |    |    |        |
A source----|             |             |            |    |    |        |
                          |             |            |    |    |        |
M source and mux control--|             |            |    |    |        |
                                        |            |    |    |        |
destination-----------------------------|            |    |    |        |
                                                     |    |    |        |
ZR,ZM------------------------------------------------|    |    |        |
                                                          |    |        |
Misc Function---------------------------------------------|    |        |
                                                               |        |
Length---------------------------------------------------------|        |
                                                                        |
Rotate------------------------------------------------------------------|
```

3.0 Program Modification

A novel technique is used for variabilizing fields in the program instruction. Two of the "destinations" of the output bus are (conceptual) registers, 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. The micro-instruction is divided between bits 23 and 24, which is a natural dividing line for all microinstructions.

4.0 Clocks

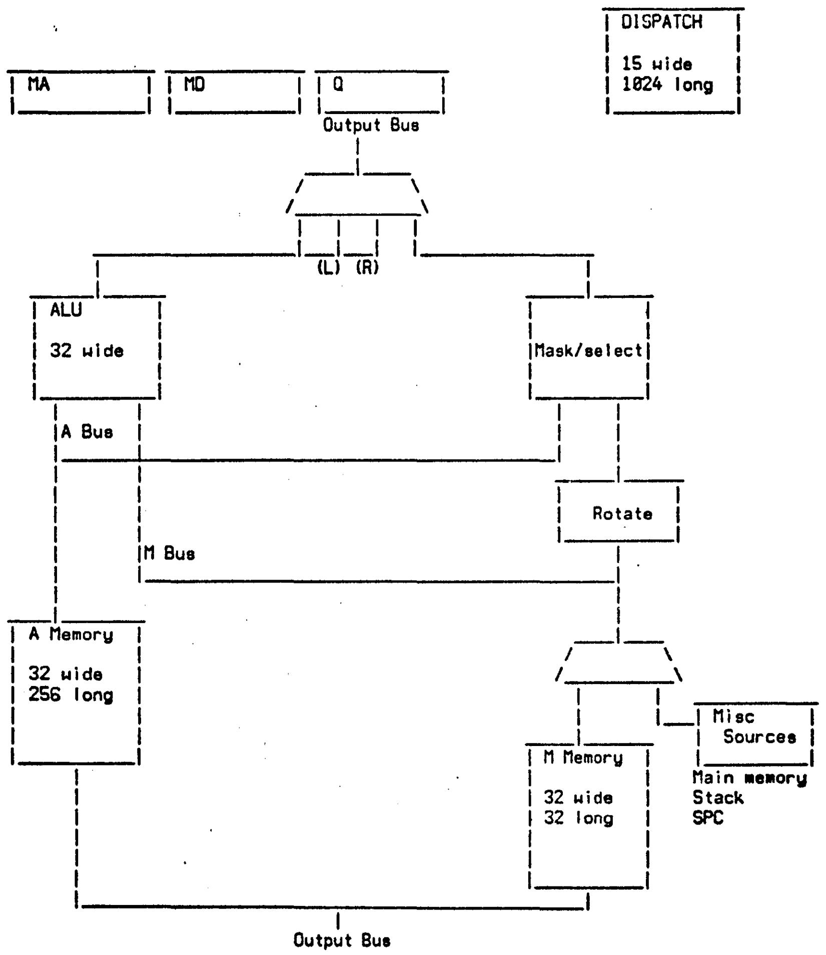
This processor uses only one clock, occuring at the end of every cycle. This clock loads output data into the designated registers, and a new PC and instruction is latched. The only events which do not take place synchronous with the clock are the control signals for the A and M scratchpads in the processor 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

```
       Instruction 0                    Instruction 1

| fetch for I0     store for I-1 | fetch for I1     store for I0    |
|----------------|----------------|----------------|----------------|
|                                 |                                 |

time ===>
```


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