PDP-11

DESIGN NOTE: Operational Details in the MUL and DIV Instructions
DATE: November 17, 1969

from: Bruce Delagi

Reviewed by:

K - project

11 Software

11 Engineering Manager

15 Engineering Manager

Abstract

The operation of condition codes, the effect of the contents of the source and destination addresses, and the word order of hi and lo-parts of operands are described. The order: low-to-high precision in memory is re-affirmed.

Ø.Ø Scope - This design note considers details of the operation of the MUL and DIV instructions. Implementation techniques, however, are not discussed.

1.Ø Condition Codes

In MUL and DIV, the Z and N-bits are set in accordance with the result (product or quotient). Both words of the product must be zero to set the Z bit. The V bit is set on arithmetic overflow. Since no overflow is possible in multiply, MUL always clears the V bit. Overflow is set in DIV if the quotient is too large to fit in one word - $/Q/ \geq 2^{15}$. The C bit holds the 17th bit of the quotient so that if $2^{16} > /Q/ \geq 2^{15}$, half the true quotient can be determined by a rotate right. The C bit is cleared by a MUL instruction.

2.Ø Word order of operands and results.

In MUL the source is taken as the multiplier and the destination is taken as the multiplicand. The destination address will hold the low order word of the product. The high order word of the product is in the following word. (In the next register if the destination is a register.)

In DIV the source is taken as the divisor, the destination as the lo-order word of the dividend, and the hi word of the dividend is stored in the following word (or register). The quotient replaces the hi-order word, and the remainder replaces the lo-order word.

Thus the stack operations to multiply two words and replace them with their product are:

```
MOV MCAND,-(SP)
MOV (SP),-(SP)          ;replicate word
MOV MPLIER,-(SP)
MUL (SP)+,(SP)          ;double word result available here
[MOV (SP)+,(SP)         ;just save lo-order on stack]
```

Stack division is done as follows - dividend presumed on stack

```
MOV DVSOR,-(SP)
DIV (SP)+,(SP)          ;or DIV (SP)+,(SP)+ if remainder
                         not required.
```

2.1 Order of bytes in words and words in double words.

The byte/word order chosen for the 11 permits an operand to be addressed at the same location whether taken as byte or word data, presuming the binary point on the right (integer representation).

The alternative order - such that byte Ø is the high order part of a word and word Ø is the high order part of the double word - has the advantages of:

1. Being the more customary representation.
2. Reading ASCII from left to right in a word string.
3. Allowing the same data to be accessed at the same address independent of precision in a fractional representation (binary point on the left).

Programming incompatibilities at the source level would be introduced by changing to this representation in future machines:

1. Wherever the mantissa of a floating point number is referenced explicitly implying modification of the floating point package.
2. Whenever a byte is intended to represent the low-order part of a word. Most programs will not mix word and byte data because of odd address restriction on word accesses. However, some user programs will be incompatible and we will create confusion by labling the bytes one way in one machine then another.
3. ICX will require modification to work on other processors in the line.
4. Bytes would presumably be loaded into the high part of registers so that memory and registers would have the same characteristics.

```
MOV #n,R
TSTB R
```

should give the same results whether R is a byte register. The automatic high extension now done would be lost so that it would be up to the

to get word results from arithmetic operations on bytes.

Since most of the work done with 11's will be real time interaction with the real world, the current format is probably superior:

1. Few laboratory instruments send or receive ASCII byte streams.

2. Most data is handled as integers.

3. 8 bits is often a convenient size to hold raw data but intermedrate results call for additional accuracy (16 bits) to avoid introduction of truncation errors.