Multiple Precision Programs for Stretch

by Dura Sweeney

The objective of the design of the registers and operation was directed towards a sophisticated command set for single-precision floating point operations, but additional operational specifications and instructions were inserted to facilitate the programming of multiple precision operations.

The following three programs illustrate the ease with which double precision operations are effected by Stretch. They are nominal accuracy programs, i.e. there is no rounding and low-order bit effects are ignored.

Add: (DE) + (JK)

1. Load AB (K)
2. nAdd AB (E)
3. nAdd AB (J)
4. nAdd AB (D)

Multiply: $(DE) \cdot (JK)$

1. Load A (K)
2. MPY and Set T_1 (D)
3. Store A (T_2)
4. Load A (E)
5. MPY and Set T_3 (J)
6. Add AB and Store T_2 (T_2)
7. Load A (T_1)
8. MPY (T_3)
9. Add AB (T_2)

Divide: $(DE) \div (JK)$

1. Load A (D)
2. Load B (E)
3. DIV and Set T_1 (J)
4. Store A (T_2)
5. Store B (T_3)
6. -MPY (K)
7. Add AB T_3
8. DIV T_1
9. Add A T_2

The following program illustrates the case of programming a triple precision addition in floating point with automatic sign control. Note that the only tests necessary are those to find the relative difference between exponents. The problem introduced in most multiple precision programme of interaction (e.g. borrow or carry) between data words which cannot all fit in the accumulator at once is handled by the operation, 'Borrow', which subtracts one from the magnitude of the Mantissa and stores the result in the specified register, then replaces the mantissa ½ and subtracts 47 from the Exponent.

Addition: (DEF) + (JKL)

Only two of the four possible paths are shown. The other two paths are the same if the factors (DEF) and (JKL) are interchanged. Further programming effort can considerably reduce the number of steps necessary in these other paths, but would be of tedious detail to this example.

(DEF) + (JKL)

1. Load A and set T_1 (D)
2. Compare exponents (J)

$D \geq J$

3. Store S (T_2)
4. Load A and set T_3 (E)
5. Compare exponents (T_2)

$J > E$

6. Load A (T_1)
7. n Add A and Store T_1 (T_2)
8. Load A (B)
9. n Add A (T_3)
10. n Add AB (K)
11. Store my B (T_3)
12. n Add A and Store T_1 (T_1)
13. Store my B (T_2)
14. Load A (F)
15. n Add A (L)
16. n Add AB (T_3)
17. Store my B (T_5)
18. n Add A (T_2)
19. Store my B (T_4)
20. n Add A and Store T_1 (T_1)
21. Load A (B)
22. $\overline{n}$ Add A (T_4)
23. $\overline{n}$ Add AB (T_5)

$J > D$ and $K \geq D$ loops not shown.

$E \geq J$

6. Load A (T_1)
7. Borrow (T_1)
8. Borrow and Store T_5 (T_4)
9. Load A (K)
10. Load B (L)
11. n Add AB (F)
12. n Add AB (T_5)
13. n Add AB (T_2)
14. Store my B (T_5)
15. n Add A (T_3)
16. n Add AB (T_4)
17. Transfer (19.)

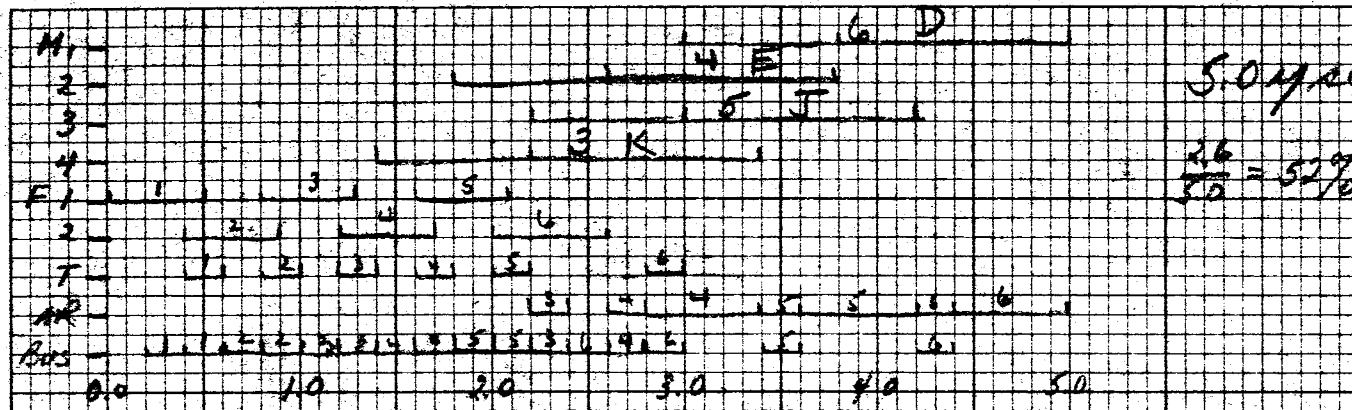
Multiplication (DEF)•(JKL)

Of the nine possible products only six are considered. These are

$D \cdot J$
$+ D \cdot K$
$+ E \cdot J$
$+ D \cdot L$
$+ E \cdot K$
$+ F \cdot J$

1.	Load A, set T_1	(D)
2.	nMPY A, set T_2	(K)
3.	Store A	(T_5)
4.	Store B.	(T_6)
5.	Load A, set T_4	(J)
6.	nMPY A, set T_9	(E)
7.	Store A	(T_7)
8.	Store B	(T_8)
9.	Load A	(T_1)
10.	nMPY A	(T_4)
11.	$\overline{n}$ Add AB	(T_5)
12.	$\overline{n}$ Add AB, Store T_5	(T_7)
13.	Store B	(T_7)
14.	$\overline{n}$ Sub S from AB	(S)
15.	Load A	(B)
16.	$\overline{n}$ Add A	(T_6)
17.	$\overline{n}$ Add AB, Store T_6	(T_8)
18.	Store B	(T_8)
19.	Load A	(T_2)
20.	nMPY A, Store T_2	(T_3)
21.	Store B	(T_3)
22.	Load A	(T_4)
23.	nMPY A	(F)
24.	$\overline{n}$ Add AB	(T_3)
25.	$\overline{n}$ Add AB, Store T_2	(T_2)
26.	Store B	(T_3)
27.	Load A	(T_1)
28.	nMPY A	(L)
29.	$\overline{n}$ Add AB	(T_2)
30.	$\overline{n}$ Add AB	(T_3)
31.	$\overline{n}$ Add AB	(T_6)
32.	$\overline{n}$ Add AB	(T_8)
33.	Store B	(T_8)
34.	$\overline{n}$ Add A	(T_7)
35.	$\overline{n}$ Add AB	(T_5)

ADD: (DE)+(JK)

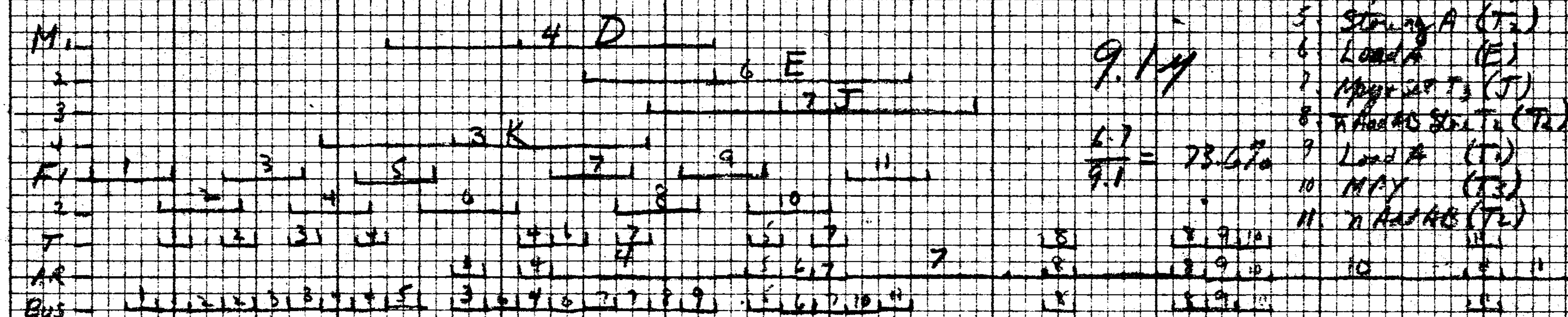


5.0 μsec

$\frac{26}{50} = 52\%$

1. Set index 1
2. Set index 2
3. Load A (K)
4. n Add A (E)
5. n Add AB (J)
6. n Add AB (D)

MPY: (DE)·(JK)

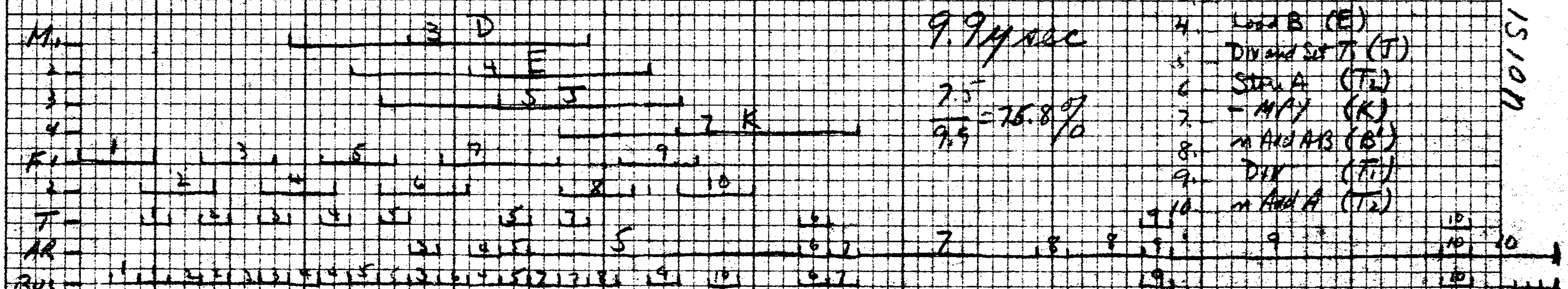


9.1 μ

$\frac{6.7}{9.1} = 73.6\%$

1. Set index 1
2. Set index 2
3. Load A (K)
4. Mpy Set T_1 (D)
5. Store A (T_2)
6. Load A (E)
7. Mpy Set T_3 (J)
8. n Add AB Set T_2 (T_2)
9. Load B (T_1)
10. MPY (T_3)
11. n Add AB (T_2)

DIV: (DE)÷(JK)



9.9 μsec

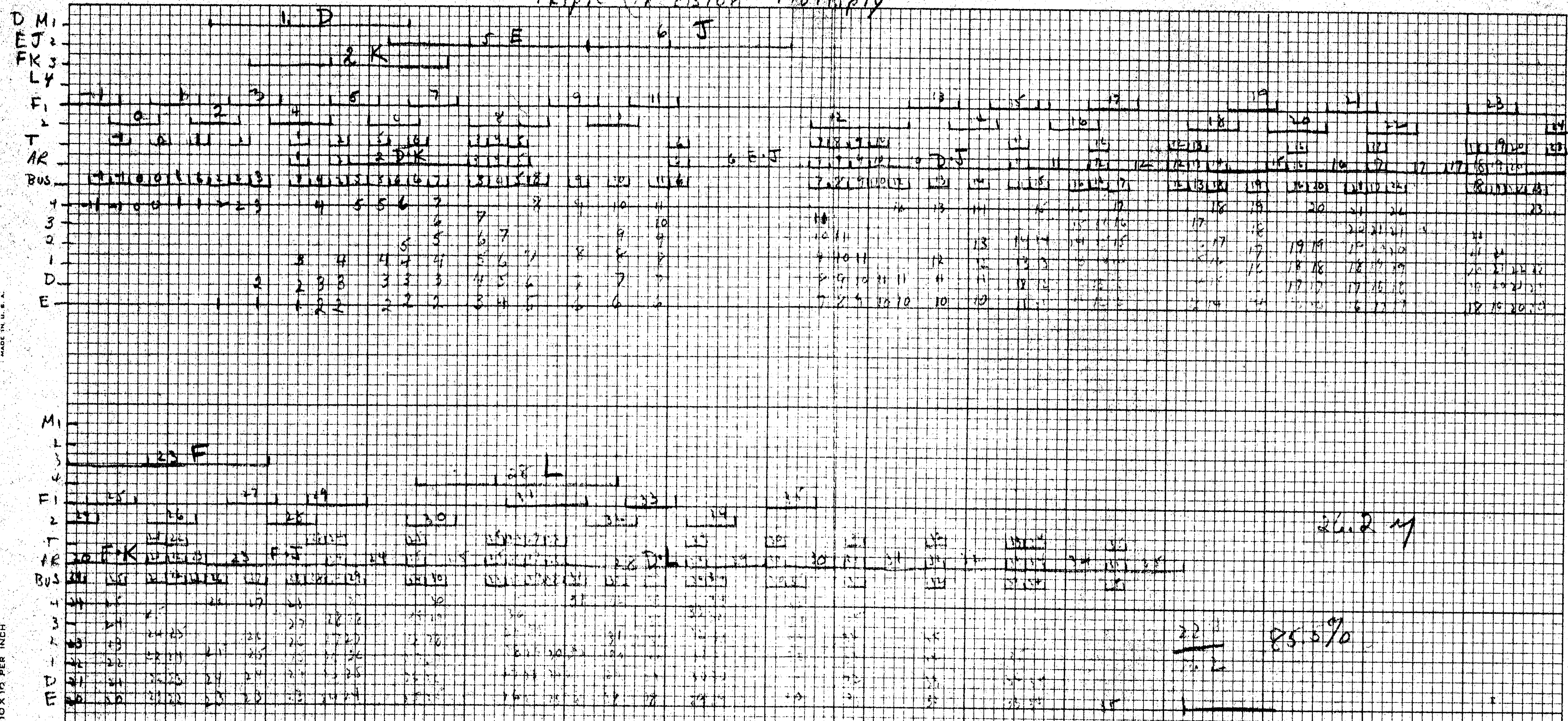
$\frac{7.5}{9.9} = 75.8\%$

1. Set index 1
2. Set index 2
3. Load A (D)
4. Load B (E)
5. Div and Set T_1 (J)
6. Store A (T_2)
7. – MPY (K)
8. n Add AB (B')
9. DIV (T_1)
10. n Add A (T_2)

Double Precision

[DEF]·[JKL]

Triple Precision Multiply



Inst. -1 & 0 are [illegible]

Inst. 1—35 are as shown [illegible] program

11/1/56

Triple Precision Add (DEF) + (JKL)

Inst -1 # 0 are relative 1 and 2 index

16.8 μsec

$\frac{11}{16} = 68.7\%$

17.4 μ

$\frac{11.2}{17.4} = 65\%$