FSCA (Floating Point Sine And Cosine Approximate): Floating-Point Instruction

PR	Format	Operation	Instruction Code	T Bit
0	FSCA FPUL, DRn	sin(FPUL) → FRn cos(FPUL) → FR[n+1]	`1111 nnn0 1111 1101`	—
1	—	reserved	`1111 nnnn 1111 1101`	—

Description

Finds the approximate value (absolute error less than ±2^–21) of the sine and cosine of the angle indicated by FPUL as a single-precision floating-point number, and writes the sine value to FRn and the cosine value to FR[n+1]. As this is an approximate operation instruction, it always generates an inexact exception request (even when the input is zero).

If the I bit in the FPSCR.enable field is enabled, an FPU exception trap will be raised regardeless of the exception occurrence, FPSCR.cause and FPSCR.flag fields reflect the actual FPU exception status, and DRn does not change. Appropriate actions must be taken by software.

Operation

```
FSCA(int n)
{
float angle;
long offset = 0x00010000;
long fraction = 0x0000ffff;
   case((FPSCR_PR){
      0: clear_cause ();
         set_I();
      /* extract sub-rotation (fraction) part */
         fraction &= FPUL;
      /* convert to float */
         angle = fraction;
      /* convert to radian */
         angle = 2*M_PI*angle / offset;
         FR[n]   sin(angle);
         FR[n+1]   cos(angle);
         pc += 2; break;
      1: undefined_operation();    /* reserved */
   }
}
```

Source Operand Data Format: Angle Specification Method

The angle is specified by a rotational value expressed as a signed fixed-point number.

Higher 16 bits and lower 16 bits are integer and fraction parts respectively.

16 bits	16 bits
MSB	LSB

0x7FFF FFFF to 0x0000 0001:	$360*2^{15} – 360/2^{16}$ to $360/2^{16}$ degrees
0x0000 0000:	0
0xFFFF FFFF to 0x8000 0000:	$–360*2^{15}$ to $–360/2^{16}$

Exceptions

Inexact

FSRRA (Floating Point Square Root Reciprocal Approximate): Floating-Point Instruction

PR	Format	Operation	Instruction Code	T Bit
0	FSRRA FRn	$1/\sqrt{FRn} \rightarrow FRn$	`1111 nnnn 0111 1101`	—
1	—	reserved	`1111 nnnn 0111 1101`	—

Description

Finds the approximate value of the reciprocal of the arithmetic square root of the single-precision floating-point number in FRn, and writes this value to FRn. As this is an approximate operation instruction, it generates an inexact exception request when the input is a normalized number (but only in this case).

If the I bit in the FPSCR.enable field is enabled, an FPU exception trap will be raised regardeless of the exception occurrence, FPSCR.cause and FPSCR.flag fields reflect the actual FPU exception status, and FRn does not change. Appropriate actions must be taken by software.

Operation

```
FSRRA(int n){
    case(FPSCR_PR){
          0:  fsrra_single(n);       break;
          1:  undefined_operation();  break;
    }
    pc += 2;
}
fsrra_single(int n)
{   clear_cause();
    case(data_type_of(n)){
      NORM :    if(sign_of(n) == 0)
                   set_I();
                   FR[n]   1/sqrt(FR[n]);
                else   invalid(n);  break;
      DENORM:      if(sign_of(n) == 0)
                   fpu_error();             break;
                else   invalid(n);  break;
      PZERO :
      NZERO :      dz(n, sign_of(n));       break;
      PINF  :   FR[n] = 0;          break;
      NINF  :   invalid(n);         break;
      qNaN  :   qnan(n);                    break;
      sNaN  :   invalid(n);         break;
    }
}
```

FSRRA Special Cases

FRn	+Norm	–Norm	+Denorm	-Denorm	+0	–0	+INF	–INF	qNaN	sNaN
FSPRA(FRn)	1/SQRT	Invalid	Error	Invalid	DZ	DZ	+0	Invalid	qNaN	INvalid

Note: When DN = 1, a denormalized number is treated as zero.

Exceptions

FPU error
Invalid operation
Divide by zero
Inexact