APPENDIX C
FLOATING-POINT MODELS AND CONVERSIONS

This appendix gives examples of how the floating-point conversion instructions can be used to perform various conversions.

C.1 Conversion from Floating-Point Number to Signed Fixed-Point Integer Word

The full convert to signed fixed-point integer word function can be implemented with the sequence shown below, assuming that the floating-point value to be converted is in FPR1, the result is returned in GPR3, and a double word at displacement “disp” from the address in GPR1 can be used as scratch space.

```
fctiw[z]    f2,f1               #convert to fx int
stfd        f2,disp(r1)         #store float
lwz         r3,disp+4(r1)       #load word and zero
```

C.2 Conversion from Floating-Point Number to Unsigned Fixed-Point Integer Word

The full convert to unsigned fixed-point integer word function can be implemented with the sequence shown below, assuming that the floating-point value to be converted is in FPR1, the value 0 is in FPR0, the value 2^{32} is in FPR3, the value 0x0000 0000 7FFF FFFF is in FPR4, the value 2^{31} is in FPR5 and GPR5, the result is returned in GPR3, and a double word at displacement "disp" from the address in GPR1 can be used as scratch space.

```
fmr         f2,f0               #use 0 if < 0
fcmpu       cr2,f1,f0
bl          cr2,store
fmr         f2,f4               #use max if > max
fcmpu       cr2,f1,f3
bgt         cr2,store
fsub        f2,f1,f5            #subtract 2**31
fcmpu       cr2,f1,f5           #use diff if Š 2**31
bnl         cr2,$+8
fmr         f2,f1
fctiw[z]    f2,f2               #convert to fx int store-
stfd        f2,disp(r1)         #store float
lwz         r3,disp+4(r1)       #load word
bl          cr2,$+8             #add 2**31 if input
add         r3,r3,r5            #was Š 2**31
```

C.3 Floating-Point Models

This section describes models for floating-point instructions.

C.3.1 Floating-Point Round to Single-Precision Model

The following algorithm describes the operation of the floating-point round to single-precision (**frsp**) instruction.

```
If FRB[1:11]<897 and FRB[1:63]>0 then
   Do
      If FPSCR[UE]=0 then goto Disabled Exponent Underflow
      If FPSCR[UE]=1 then goto Enabled Exponent Underflow
   End

If FRB[1:11]>1150 and FRB[1:11]<2047 then
   Do
      If FPSCR[OE]=0 then goto Disabled Exponent Overflow
      If FPSCR[OE]=1 then goto Enabled Exponent Overflow
   End

If FRB[1:11]>896 and FRB[1:11]<1151 then goto Normal Operand

If FRB[1:63]=0 then goto Zero Operand

If FRB[1:11]=2047 then
   Do
      If FRB[12:63]=0 then goto Infinity Operand
      If FRB[12]=1 then goto QNaN Operand
      If FRB[12]=0 and FRB[13:63]>0 then goto SNaN Operand
   End
```

Disabled Exponent Underflow:

```
sign ← FRB0
If FRB[1:11]=0 then
   Do
      exp ← -1022
      frac ← 0b0 || FRB[12:63]
   End
If FRB[1:11]>0 then
   Do
      exp ← FRB[1:11] - 1023
      frac ← 0b1 || FRB[12:63]
   End
Denormalize operand:
   G || R || X ← 0b000
   Do while exp<-126
      exp ← exp + 1
      frac || G || R || X ← 0b0 || frac || G || (R | X)
   End
FPSCR[UX] < frac[24:52] || G || R || X>0
If frac[24:52] || G || R || X>0 then FPSCR[XX] ← 1
Round single(sign,exp,frac,G,R,X)
If frac=0 then
   Do
      FRT00 ← sign
      FRT0[1:63] ← 0
      If sign=0 then FPSCR[FPRF] ← "+zero"
      If sign=1 then FPSCR[FPRF] ← "-zero"
   End
If frac>0 then
   Do
      If frac[0]=1 then
         Do
            If sign=0 then FPSCR[FPRF] ← "+normal number"
            If sign=1 then FPSCR[FPRF] ← "-normal number"
         End
      If frac[0]=0 then
         Do
            If sign=0 then FPSCR[FPRF] ← "+denormalized number"
            If sign=1 then FPSCR[FPRF] ← "-denormalized number"
         End
      Normalize operand-
         Do while frac[0]=0
            exp ← exp-1
            frac || G || R ← frac[1:52] || G || R || 0b0
```

```
         End
      FRT[0] ← sign
      FRT[1:11] ← exp + 1023
      FRT[12:63] ← frac[1:23] || 0b 0 0000 0000 0000 0000 0000 0000 0000
   End
Done
```

Enabled Exponent Underflow

```
FPSCR[UX] ← 1
sign ← FRB[0]
If FRB[1:11]=0 then
   Do
      exp ← -1022
      frac ←0b0 || FRB[12:63]
   End
If FRB[1:11]>0 then
   Do
      exp ← FRB[1:11] - 1023
      frac ← 0b1 || FRB[12:63]
   End
Normalize operand-
   Do while frac[0]=0
      exp ← exp - 1
      frac ← frac[1:52] || 0b0
   End
If frac[24:52]>0 then FPSCR[XX] ← 1
Round single(sign,exp,frac,0,0,0)
exp ← exp + 192
FRT[0] ← sign
FRT[1:11] ← exp + 1023
FRT[12:63] ← frac1-23 || 0b0 0000 0000 0000 0000 0000 0000 0000
If sign=0 then FPSCR[FPRF] ← "+normal number"
If sign=1 then FPSCR[FPRF] ← "-normal number"
Done
```

Disabled Exponent Overflow

```
inc ←0
FPSCR[OX] ← 1
FPSCR[XX] ← 1
If FPSCR[RN]= 0b00 then    /* Round to Nearest */
   Do
     inc ←0
     If FRB[0]=0 then FRT ← 0x7FF0 0000 0000 0000
     If FRB[0]=1 then FRT ←0xFFF0 0000 0000 0000
     If FRB[0]=0 then FPSCR[FPRF] ← "+infinity"
     If FRB[0]=1 then FPSCR[FPRF] ← "-infinity"
   End
If FPSCR[RN]= 0b01 then/* Round Truncate */
   Do
     If (0b0 || FRB[1:63]) < 0x047EF FFFF E000 0000 then inc ←0
     If FRB[0]=0 then FRT ←0x47EF FFFF E000 0000
     If FRB[0]=1 then FRT ←0xC7EF FFFF E000 0000
     If FRB[0]=0 then FPSCR[FPRF] ← "+normal number"
     If FRB[0]=1 then FPSCR[FPRF] ← "-normal number"
   End
If FPSCR[RN]= 0b10 then /* Round to +Infinity */
   Do
     If FRB[0]=0 then inc ←0
     If (FRB[0]=1 & (FRB > 0xC7EF FFFF E000 0000 then inc ← 1)
     If FRB[0]=0 then FRT ← 0x7FF0 0000 0000 0000
     If FRB[0]=1 then FRT ← 0xC7EF FFFF E000 0000
     If FRB[0]=0 then FPSCR[FPRF] ← "+infinity"
     If FRB[0]=1 then FPSCR[FPRF] ← "-normal number"
   End
If FPSCR[RN]= 0b11 then/* Round to -Infinity */
```

```
   Do
     (If FRB[0]=0 & FRB < 0x47EF FFFF E000 0000) then inc ← 1
     If FRB[0]= 1 then inc ← 1
     If FRB[0]=0 then FRT ← 0x47EF FFFF E000 0000
     If FRB[0]=1 then FRT ← 0xFFF0 0000 0000 0000
     If FRB[0]=0 then FPSCR[FPRF] ← "+normal number"
     If FRB[0]=1 then FPSCR[FPRF] ← "-infinity"
   End
FPSCR[FR] ← inc
FPSCR[FI] ← 1
Done
```

Enabled Exponent Overflow

```
sign ← FRB[0]
exp ← FRB[1:11] - 1023
   frac ← 0b1 || [12:63]
   If frac[24:52]>0 then FPSCR[XX] ← 1
   Round single(sign,exp,frac,0,0,0)
Enabled Overflow
   FPSCR[OX] ← 1
   exp ← exp - 192
   FRT[0] ← sign
   FRT[1:11] ← exp + 1023
   FRT[12:63] ← frac[1:23] || 0b0 0000 0000 0000 0000 0000 0000 0000
   If sign=0 then FPSCR[FPRF] ← "+normal number"
   If sign=1 then FPSCR[FPRF] ← "-normal number"
Done
```

Zero Operand

```
FRT ← FRB
If FRB[0]=0 then FPSCR[FPRF] ← "+zero"
If FRB[0]=1 then FPSCR[FPRF] ← "-zero"
FPSCR[FR FI] ← 0b00
Done
```

Infinity Operand

```
FRT ← FRB
If FRB[0]=0 then FPSCR[FPRF] ← "+infinity"
If FRB[0]=1 then FPSCR[FPRF] ← "-infinity" Done
QNaN Operand-
FRT ← FRB[0:34] || 0b0 0000 0000 0000 0000 0000 0000 0000
FPSCR[FPRF] ← "QNaN"
FPSCR[FR FI] ← 0b00
Done
```

QNaN Operand

```
FRT ← FRB[0:34] || 0b0 0000 0000 0000 0000 0000 0000 0000
FPSCR[FPRF] ← "QNaN"
FPSCR[FR FI] ← 0b00
Done
```

SNaN Operand

```
FPSCR[VXSNAN] ← 1
If FPSCR[VE]=0 then
   Do
      FRT[0:11] ← FRB[0:11]
      FRT[12] ← 1
      FRT[13:63] ← FRB[13:34] || 0b0 0000 0000 0000 0000 0000 0000 0000
      FPSCR[FPRF] ← "QNaN"
   End
FPSCR[FR FI] ← 0b00
Done
```

Normal Operand

```
sign ← FRB[0]
exp ← FRB[1:11] - 1023
frac ← 0b1 || FRB[12:63]
If frac[24:52]>0 then FPSCR[XX] ← 1
Round single(sign,exp,frac,0,0,0)
If exp>+127 and FPSCR[OE]=0 then go to Disabled Exponent Overflow
If exp>+127 and FPSCR[OE]=1 then go to Enabled Overflow
FRT[0] ← sign
FRT[1:11] ← exp + 1023
FRT[12:63] ← frac[1:23] || 0b0 0000 0000 0000 0000 0000 0000 0000
If sign=0 then FPSCR[FPRF] ← "+normal number"
If sign=1 then FPSCR[FPRF] ← "-normal number"
Done
```

Round Single (sign,exp,frac,G,R,X)

```
inc ← 0
lsb ← frac[23]
gbit ← frac[24]
rbit ← frac[25]
xbit ← (frac[26:52]||G||R||X)¦0
If FPSCR[RN]=0b00 then
  Do
    If sign || lsb || gbit || rbit || xbit = 0bu11uu then inc ← 1
    If sign || lsb || gbit || rbit || xbit = 0bu011u then inc ← 1
    If sign || lsb || gbit || rbit || xbit = 0bu01u1 then inc ← 1
  End
If FPSCR[RN]= 0b10 then
  Do
    If sign || lsb || gbit || rbit || xbit = 0b 0u1uu then inc ← 1
    If sign || lsb || gbit || rbit || xbit = 0b0uu1u then inc ← 1
    If sign || lsb || gbit || rbit || xbit = 0b0uuu1 then inc ← 1
  End
If FPSCR[RN]= 0b11then
  Do
    If sign || lsb || gbit || rbit || xbit = 0b1u1uu then inc ← 1
    If sign || lsb || gbit || rbit || xbit = 0b1uu1u then inc ← 1
    If sign || lsb || gbit || rbit || xbit = 0b1uuu1 then inc ← 1
  End
frac[0:23] ← frac[0:23] + inc
If carry_out=1 then
  Do
    frac[0-23] ← 0b1 || frac[0:22]
    exp ← exp + 1
  End
FPSCR[FR] ← inc
FPSCR[FI] ← gbit | rbit | xbit
Return
```

C.3.2 Floating-Point Convert to Integer Model

The following algorithm describes the operation of the floating-point convert to integer instructions. In this example, u represents an undefined hexadecimal digit.

```
If Floating Convert to Integer Word
   Then Do
      Then round_mode ← FPSCR[RN]
      tgt_precision ← "32-bit integer"
   End
If Floating Convert to Integer Word with round toward Zero
   Then Do
      round_mode ← 0b01
      tgt_precision ← "32-bit integer"
   End
If Floating Convert to Integer Doubleword
```

```
   Then Do
      round_mode ← FPSCR[RN]
      tgt_precision ← "64-bit integer"
   End
If Floating Convert to Integer Doubleword with round toward Zero
   Then Do
      round_mode ← 0b01
      tgt_precision ← "64-bit integer"
   End
If FRB[1:11]=2047 and FRB[12:63]=0 then goto Infinity Operand
If FRB[1:11]=2047 and FRB12=0 then goto SNaN Operand
If FRB[1:11]=2047 and FRB12=1 then goto QNaN Operand
If FRB[1:11]>1086 then goto Large Operand


sign ← FRB0
If FRB[1:11]>0 then exp ← FRB[1:11] - 1023 /* exp - bias */
If FRB[1:11]=0 then exp ← -1022
If FRB[1:11]>0 then frac[0:64]←0b01 ||FRB[12:63]||0b00000000000 /
*normal*/
If FRB[1:11]=0 then frac[0:64]←b'00'||FRB[12:63]||0b00000000000 /
*denormal*/


gbit || rbit || xbit ← 0b000
Do i=1,63-exp
   frac[0:64] || gbit || rbit || xbit ← 0b0 || frac[0:64] || gbit ||
(rbit|xbit)
End


If gbit | rbit | xbit then FPSCR[XX] ← 1
```

Round Integer (frac,gbit,rbit,xbit,round_mode)

In this example, u represents an undefined hexadecimal digit. Comparisons ignore the u bits.

```
If sign=1 then frac[0:64] ← ¬frac[0:64] + 1

If tgt_precision="32-bit integer" and frac[0:64]>+2(31)-1
   then goto Large Operand
If tgt_precision="64-bit integer" and frac[0:64]>+2(63)-1
   then goto Large Operand
If tgt_precision="32-bit integer" and frac[0:64]<-2(31) then goto Large
Operand
If tgt_precision="64-bit integer" and frac[0:64]<-2(63) then goto Large
Operand
If tgt_precision="32-bit integer"
   then FRT ← 0x xuuuuuuu || frac[33:64]
If tgt_precision="64-bit integer" then FRT ← frac[1:64]
FPSCR[FPRF] ← undefined
Done
```

Round Integer (frac,gbit,rbit,xbit,round_mode)

In this example, u represents an undefined hexadecimal digit. Comparisons ignore the u bits.

```
inc ← 0
If round_mode= 0b00 then
   Do
      If sign || frac[64] || gbit || rbit || xbit = 0bu11ux then inc ← 1
      If sign || frac[64] || gbit || rbit || xbit = 0bu011x then inc ← 1
      If sign || frac64 || gbit || rbit || xbit = 0bu01u1 then inc ← 1
   End
If round_mode= 0b10 then
      Do
         If sign || frac64 || gbit || rbit || xbit = 0b0u1ux then inc ← 1
         If sign || frac64 || gbit || rbit || xbit = 0b0uu1x then inc ← 1
         If sign || frac64 || gbit || rbit || xbit = 0b0uuu1 then inc ← 1
      End
If round_mode = 0b11 then
   Do
      If sign || frac64 || gbit || rbit || xbit = 0b1u1ux then inc ← 1
   If sign || frac64 || gbit || rbit || xbit = 0b1uu1x then inc ← 1
      If sign || frac64 || gbit || rbit || xbit = 0b1uuu1 then inc ← 1
   End
frac[0:64] ← frac[0:64] + inc
FPSCR[FR] ← inc
FPSCR[FI] ← gbit | rbit | xbit
Return
```

Infinity Operand

```
FPSCR[FR FI VXCVI] ← 0b001
If FPSCR[VE]=0 then Do
   If tgt_precision="32-bit integer" then
      Do
         If sign=0 then FRT ← 0xuuuu uuuu 7FFF FFFF
         If sign=1 then FRT ← 0xuuuu uuuu 8000 0000
      End
   Else
      Do
         If sign=0 then FRT ← 0x7FFF FFFF FFFF FFFF
         If sign=1 then FRT ← 0x8000 0000 0000 0000
      End
   FPSCR[FPRF] < undefined
   End
Done
```

SNaN Operand

```
FPSCR[FR FI VXCVI VXSNAN] ← 0b0011
If FPSCR[VE]=0 then
   Do
      If tgt_precision="32-bit integer"
         then FRT ← 0xuuuu uuuu 8000 0000
      If tgt_precision="64-bit integer"
         then FRT ← 0x8000 0000 0000 0000
      FPSCR[FPRF] ← undefined
   End
Done
```

QNaN Operand

```
FPSCR[FR FI VXCVI] ← 0b001
If FPSCR[VE]=0 then
   Do
     If tgt_precision="32-bit integer" then FRT ← 0xuuuu uuuu 8000 0000
     If tgt_precision="64-bit integer" then FRT ← 0x8000 0000 0000 0000
     FPSCR[FPRF] < undefined
   End
Done
```

Large Operand

```
FPSCR[FR FI VXCVI] ← 0b001
If FPSCR[VE]=0 then Do
   If tgt_precision="32-bit integer" then
      Do
         If sign=0 then FRT ← 0xuuuu uuuu 7FFF FFFF
         If sign=1 then FRT ← 0xuuuu uuuu 8000 0000
      End
   Else
      Do
         If sign=0 then FRT ← 0x7FFF FFFF FFFF FFFF
         If sign=1 then FRT ← 0x8000 0000 0000 0000
      End
   FPSCR[FPRF] ← undefined
   End
Done
```

C.4 Floating-Point Convert from Integer Model

The following algorithm describes the operation of the floating-point convert from integer instructions.

```
sign ← FRB[0]
exp ← 63
frac ← FRB


If frac=0 then go to Zero Operand
If sign=1 then frac ← ¬frac + 1


Do until frac[0]=1
   frac ← frac[1:63] || 0b0
   exp ← exp - 1
End
```

Round Float (sign,exp,frac,FPSCR[RN])

```
If sign=1 then FPSCR[FPRF] ← "-normal number"
If sign=0 then FPSCR[FPRF] ← "+normal number"
FRT[0] ← sign
FRT[1:11] ← exp + 1023 /* exp + bias */
FRT[12:63] ← frac[1:52]
Done
```

Zero Operand

```
FPSCR[FR FI] ← 0b00
FPSCR[FPRF] ← "+zero"
FRT ← 0x0000 0000 0000 0000
Done
```

Round Float (sign,exp,frac,round_mode)

In this example, the bits designated as u are ignored in comparisons.

```
inc ← 0
lsb ← frac[52]
gbit ← frac[53]
rbit ← frac[54]
xbit ← frac[55-63]>0
If round_mode=0b00 then
   Do
      If sign || lsb || gbit || rbit || xbit = 0bu11uu then inc ← 1
      If sign || lsb || gbit || rbit || xbit = 0bu011u then inc ← 1
      If sign || lsb || gbit || rbit || xbit = 0bu01u1 then inc ← 1
   End
If round_mode= 0b10 then
   Do
      If sign || lsb || gbit || rbit || xbit = 0b0u1uu then inc ← 1
      If sign || lsb || gbit || rbit || xbit = 0b0uu1u then inc ← 1
      If sign || lsb || gbit || rbit || xbit = 0b0uuu1 then inc ← 1
   End
If round_mode= 0b11 then
   Do
      If sign || lsb || gbit || rbit || xbit = 0b1u1uu then inc ← 1
      If sign || lsb || gbit || rbit || xbit = 0b1uu1u then inc ← 1
      If sign || lsb || gbit || rbit || xbit = 0b1uuu1 then inc ← 1
   End
frac[0:52] ← frac[0:52] + inc
If carry_out=1 then exp ← exp + 1
FPSCR[FR] ← inc
FPSCR[FI] ← gbit | rbit | xbit
If (gbit | rbit | xbit) then FPSCR[XX] ← 1
Return
```