```
00000000                                  CPU  "SYMPL64_IL.TBL"
00000000                                  HOF  "bin32"
00000000                                  WDLN 8
                                          ; version 2.02   June 17, 2018
                                          ; Author:  Jerry D. Harthcock
                                          ; SYMPL 64-BIT IEEE 754-2008 Floating-Point ISA assembler test for SYMPL Intermediate Language ("IL") with some straight assembly examples

                               ;private dword storage
00000000         =             bitbucket:  EQU     0x0000                   ;this dword location is reserved.  Don't use it for anything because a lot of garbage can wind up here
00000008         =             work_1:     EQU     0x0008
00000010         =             work_2:     EQU     0x0010
00000018         =             work_3:     EQU     0x0018
00000020         =             capt0_save: EQU     0x0020                  ;alternate delayed exception capture register 0 save location
00000028         =             capt1_save: EQU     0x0028                  ;alternate delayed exception capture register 1 save location
00000030         =             capt2_save: EQU     0x0030                  ;alternate delayed exception capture register 2 save location
00000038         =             capt3_save: EQU     0x0038                  ;alternate delayed exception capture register 3 save location


00000000                                   org    0x0000
00000000 0000000040A00000      _5p0:       dff    0, 5.0
00000001 0000000040000000      _2p0:       dff    0, 2.0


000000FE                                   org     0x00FE

000000FE 0000010B              Constants:  DFL     start                   ;program memory locations 0x000 - 0x0FF reserved for look-up table

000000FE 000001DB              prog_len:   DFL     progend - Constants

                               ;           type    dest = OP:(type:srcA, type:srcB)

00000100                                   org     0x00000100                              ;default interrupt/trap vector locations
00000100                       load_vects:
00000100 12FEF800000202A4                  uh      NMI_VECT = uh:#NMI_                     ;load of interrupt vectors for faster interrupt response
00000101 12FEF000000202D5                  uh      IRQ_VECT = uh:#IRQ_                     ;these registers are presently not visible to app s/w
00000102 12FEE800000202A8                  uh      INV_VECT = uh:#INV_
00000103 12FEE000000202B1                  uh      DIVx0_VECT = uh:#DIVx0_
00000104 12FED800000202BA                  uh      OVFL_VECT = uh:#OVFL_
00000105 12FED000000202C3                  uh      UNFL_VECT = uh:#UNFL_
00000106 12FEC800000202CC                  uh      INEXT_VECT = uh:#INEXT_
00000107 34FF684000060000                  uw      TIMER = uw:#0x60000                     ;load time-out timer with sufficient time to process before timeout
00000108 14FFA04FF887C003                          GOTO start
00000109                       done:
00000109 12FF8C0000020300                          setDone
0000010A 14FFA04FF887C000      spin:               GOTO spin

0000010B                       start:
0000010B 12FF8C0000020200                          clearDone
0000010C 2400084000000000                  uw      work_1  = uw:@_5p0
0000010D 2400104000100000                  uw      work_2  = uw:@_2p0
0000010E 04FF604000800000                  fs      creg = fs:work_1
0000010F 02E7F84000840010                  fh      rem.15 = remainder:(fs:work_1, fs:work_2)           ;14 clocks
00000110 02EB784000840010                  fh      fma.15 = fusedMultiplyAdd:(fs:work_1, fs:work_2, C) ;6 clocks
00000111 02EDF84000840010                  fh      fmul.15 = multiplication:(fs:work_1, fs:work_2)     ;4 clocks
00000112 02EEF84000840010                  fh      fadd.15 = addition:(fs:work_1, fs:work_2)           ;5 clocks
00000113 02EE784000840010                  fh      fsub.15 = subtraction:(fs:work_1, fs:work_2)        ;5 clocks
00000114 02EC784000840010                  fh      fdiv.15 = division:(fs:work_1, fs:work_2)           ;8 clocks
00000115 02EAF84000800000                  fh      log.15 = log:(fs:work_1)                            ;9 clocks
00000116 02EA782EAF800000                  fh      exp.15 = exp:(fh:log.15)                            ;5 clocks
```

```
00000117 02EBF84000800000                  fh      sqrt.15 = squareRoot:(fs:work_1)                    ;6 clocks
00000118 12E5F8000002001E                  fh      sind.3 = sind:(uh:#30)                              ;3 clocks
00000119 12E5D8000002007A                  fh      cosd.3 = cosd:(uh:#122)                             ;3 clocks
0000011A 12E5B800000200DF                  fh      tand.3 = tand:(uh:#223)                             ;3 clocks
0000011B 12E5980000020062                  fh      cotd.3 = cotd:(uh:#98)                              ;3 clocks
0000011C 02E3784000840010                  fh      pow.15 = pow:(fs:work_1, fs:work_2)                 ;13 clocks
0000011D 02E37040008C0010                  fh      pow.14 = pown:(fs:work_1, xfs:work_2)               ;13 clocks
0000011E 02E368C000840010                  fh      pow.13 = powr:(xfs:work_1, fs:work_2)               ;13 clocks
0000011F 1200180000023579                  uh      work_3 = uh:#0x3579

00000120 12E9782EC7800010                  rh.e    rtoi.15 = roundToIntegralTiesToEven:(fh:fdiv.15)     ;3 clocks
00000121 12E9702EC7800018                  rh.a    rtoi.14 = roundToIntegralTiesToAway:(fh:fdiv.15)     ;3 clocks
00000122 12E9682EC7800013                  rh.z    rtoi.13 = roundToIntegralTowardZero:(fh:fdiv.15)     ;3 clocks
00000123 12E9602EC7800011                  rh.p    rtoi.12 = roundToIntegralTowardPositive:(fh:fdiv.15) ;3 clocks
00000124 12E9582EC7800012                  rh.n    rtoi.11 = roundToIntegralTowardNegative:(fh:fdiv.15) ;3 clocks
00000125 02E9502EC7800000                  fh      rtoi.10 = roundToIntegralExact:(fh:fdiv.15)          ;3 clocks
00000126 42E9402EC7800000                  fh.p    rtoi.8  = roundToIntegralExact:(fh:fdiv.15)          ;3 clocks
00000127 82E9382EC7800000                  fh.n    rtoi.7  = roundToIntegralExact:(fh:fdiv.15)          ;3 clocks
00000128 C2E9302EC7800000                  fh.z    rtoi.6  = roundToIntegralExact:(fh:fdiv.15)          ;3 clocks

00000129 12ED780000020022                  fh      itof.15 = convertFromInt:(uh:#0x0022)                ;2 clocks
0000012A 52ED680000020022                  fh.p    itof.13 = convertFromInt:(uh:#0x0022)                ;2 clocks
0000012B 92ED600000020022                  fh.n    itof.12 = convertFromInt:(uh:#0x0022)                ;2 clocks
0000012C D2ED580000020022                  fh.z    itof.11 = convertFromInt:(uh:#0x0022)                ;2 clocks
0000012D 14ECF82ED7800000                  uw.e    ftoi.15 = convertToIntegerTiesToEven:(fh:itof.15)         ;3 clocks
0000012E 14ECF02ED7800003                  uw.z    ftoi.14 = convertToIntegerTowardZero:(fh:itof.15)         ;3 clocks
0000012F 14ECE82ED7800001                  uw.p    ftoi.13 = convertToIntegerTowardPositive:(fh:itof.15)     ;3 clocks
00000130 14ECE02ED7800002                  uw.n    ftoi.12 = convertToIntegerTowardNegative:(fh:itof.15)     ;3 clocks
00000131 14ECD82ED7800008                  uw.a    ftoi.11 = convertToIntegerTiesToAway:(fh:itof.15)         ;3 clocks
00000132 14ECD02ED7800010                  uw.ex   ftoi.10 = convertToIntegerExactTiesToEven:(fh:itof.15)    ;3 clocks
00000133 14ECC82ED7800013                  uw.zx   ftoi.9  = convertToIntegerExactTowardZero:(fh:itof.15)    ;3 clocks
00000134 14ECC02ED7800011                  uw.px   ftoi.8  = convertToIntegerExactTowardPositive:(fh:itof.15);3 clocks
00000135 14ECB82ED7800012                  uw.nx   ftoi.7  = convertToIntegerExactTowardNegative:(fh:itof.15);3 clocks
00000136 14ECB02ED7800018                  uw.ax   ftoi.6  = convertToIntegerExactTiesToAway:(fh:itof.15)    ;3 clocks
00000137 02E8782ED7800000                  fh      logb.15     = logB:(fh:itof.15)                 ;4 clocks
00000138 02E8F82ED7840010                  fh      scaleB.15   = scaleB:(fh:itof.15, fs:work_2)    ;4 clocks
00000139 02E7782ED7800000                  fh      nextUp.7    = nextUp:(fh:itof.15)               ;3 clocks
0000013A 02E7384001000000                  fh      nextDown.7  = nextDown:(fs:work_2)              ;3 clocks
0000013B 02E6984001040008                  fh      minNum.3    = minNum:(fs:work_2, fs:work_1)     ;3 clocks
0000013C 02E6B84001040008                  fh      maxNum.3    = maxNum:(fs:work_2, fs:work_1)     ;3 clocks
0000013D 02E6782EBF800000                  fh      copy.3      = copy:(fh:sqrt.15)                 ;3 clocks
0000013E 02E6584001000000                  fh      negate.3    = negate:(fs:work_2)                ;3 clocks
0000013F 02E6382E65800000                  fh      abs.3       = abs:(fh:negate.3)                 ;3 clocks
00000140 02E618400082E658                  fh      copySign.3  = copySign:(fs:work_1, fh:negate.3) ;3 clocks
00000141 04E9F84000800000                  fs      conv.15     = convertFormat:(fs:work_1)         ;3 clocks

00000142 16E4F82E37800000                  ud      cnvTDCS.15  = convertToDecimalCharacter:(fh:pow.15, ub:#0)     ;8 clocks
00000143 16E4F02EB7800000                  ud      cnvTDCS.14  = convertToDecimalCharacter:(fh:fma.15, ub:#0)     ;8 clocks
00000144 16E4E82EC7800000                  ud      cnvTDCS.13  = convertToDecimalCharacter:(fh:fdiv.15, ub:#0)    ;8 clocks
00000145 16E4E02EAF800000                  ud      cnvTDCS.12  = convertToDecimalCharacter:(fh:log.15, ub:#0)     ;8 clocks
00000146 16E4D82EA7800000                  ud      cnvTDCS.11  = convertToDecimalCharacter:(fh:exp.15, ub:#0)     ;8 clocks
00000147 16E4D02EBF800000                  ud      cnvTDCS.10  = convertToDecimalCharacter:(fh:sqrt.15, ub:#0)    ;8 clocks
00000148 16E4C82E97800000                  ud      cnvTDCS.9   = convertToDecimalCharacter:(fh:rtoi.15, ub:#0)    ;8 clocks
00000149 16E4C02ED7800000                  ud      cnvTDCS.8   = convertToDecimalCharacter:(fh:itof.15, ub:#0)    ;8 clocks
0000014A 16E4B82E87800000                  ud      cnvTDCS.7   = convertToDecimalCharacter:(fh:logb.15, ub:#0)    ;8 clocks
0000014B 16E4B02E8F800000                  ud      cnvTDCS.6   = convertToDecimalCharacter:(fh:scaleb.15, ub:#0)  ;8 clocks
0000014C 16E4A82E77800000                  ud      cnvTDCS.5   = convertToDecimalCharacter:(fh:nextUp.7, ub:#0)   ;8 clocks
0000014D 16E4A02E73800000                  ud      cnvTDCS.4   = convertToDecimalCharacter:(fh:nextDown.7, ub:#0) ;8 clocks
0000014E 16E4982E69800000                  ud      cnvTDCS.3   = convertToDecimalCharacter:(fh:minNum.3, ub:#0)   ;8 clocks
```

```
0000014F 16E4902E6B800000                   ud      cnvTDCS.2   = convertToDecimalCharacter:(fh:maxNum.3, ub:#0)   ;8 clocks
00000150 16E4882E65800000                   ud      cnvTDCS.1   = convertToDecimalCharacter:(fh:negate.3, ub:#0)   ;8 clocks
00000151 16E4802E61800000                   ud      cnvTDCS.0   = convertToDecimalCharacter:(fh:copySign.3, ub:#0) ;8 clocks

00000152 02E5786E4F8EE4F8                   fh      cnvFDCS.15  = convertFromDecimalCharacter:(ud:cnvTDCS.15, sd:cnvTDCS.15)  ;7 clocks
00000153 02E5706E4F0EE4F0                   fh      cnvFDCS.14  = convertFromDecimalCharacter:(ud:cnvTDCS.14, sd:cnvTDCS.14)  ;7 clocks
00000154 02E5686E4E8EE4E8                   fh      cnvFDCS.13  = convertFromDecimalCharacter:(ud:cnvTDCS.13, sd:cnvTDCS.13)  ;7 clocks
00000155 02E5606E4E0EE4E0                   fh      cnvFDCS.12  = convertFromDecimalCharacter:(ud:cnvTDCS.12, sd:cnvTDCS.12)  ;7 clocks
00000156 02E5586E4D8EE4D8                   fh      cnvFDCS.11  = convertFromDecimalCharacter:(ud:cnvTDCS.11, sd:cnvTDCS.11)  ;7 clocks
00000157 02E5506E4D0EE4D0                   fh      cnvFDCS.10  = convertFromDecimalCharacter:(ud:cnvTDCS.10, sd:cnvTDCS.10)  ;7 clocks
00000158 02E5486E4C8EE4C8                   fh      cnvFDCS.9   = convertFromDecimalCharacter:(ud:cnvTDCS.9 , sd:cnvTDCS.9 )  ;7 clocks
00000159 02E5406E4C0EE4C0                   fh      cnvFDCS.8   = convertFromDecimalCharacter:(ud:cnvTDCS.8 , sd:cnvTDCS.8 )  ;7 clocks
0000015A 02E5386E4B8EE4B8                   fh      cnvFDCS.7   = convertFromDecimalCharacter:(ud:cnvTDCS.7 , sd:cnvTDCS.7 )  ;7 clocks
0000015B 02E5306E4B0EE4B0                   fh      cnvFDCS.6   = convertFromDecimalCharacter:(ud:cnvTDCS.6 , sd:cnvTDCS.6 )  ;7 clocks
0000015C 02E5286E4A8EE4A8                   fh      cnvFDCS.5   = convertFromDecimalCharacter:(ud:cnvTDCS.5 , sd:cnvTDCS.5 )  ;7 clocks
0000015D 02E5206E4A0EE4A0                   fh      cnvFDCS.4   = convertFromDecimalCharacter:(ud:cnvTDCS.4 , sd:cnvTDCS.4 )  ;7 clocks
0000015E 02E5186E498EE498                   fh      cnvFDCS.3   = convertFromDecimalCharacter:(ud:cnvTDCS.3 , sd:cnvTDCS.3 )  ;7 clocks
0000015F 02E5106E490EE490                   fh      cnvFDCS.2   = convertFromDecimalCharacter:(ud:cnvTDCS.2 , sd:cnvTDCS.2 )  ;7 clocks
00000160 02E5086E488EE488                   fh      cnvFDCS.1   = convertFromDecimalCharacter:(ud:cnvTDCS.1 , sd:cnvTDCS.1 )  ;7 clocks
00000161 02E5006E480EE480                   fh      cnvFDCS.0   = convertFromDecimalCharacter:(ud:cnvTDCS.0 , sd:cnvTDCS.0 )  ;7 clocks

00000162 16E3F82E7F800000                   ud      cnvTHCS.15  = convertToHexCharacter:(fh:rem.15, ub:#0)     ;5 clocks
00000163 16E3F02EB7800000                   ud      cnvTHCS.14  = convertToHexCharacter:(fh:fma.15, ub:#0)     ;5 clocks
00000164 16E3E82EC7800000                   ud      cnvTHCS.13  = convertToHexCharacter:(fh:fdiv.15, ub:#0)    ;5 clocks
00000165 16E3E02EAF800000                   ud      cnvTHCS.12  = convertToHexCharacter:(fh:log.15, ub:#0)     ;5 clocks
00000166 16E3D82EA7800000                   ud      cnvTHCS.11  = convertToHexCharacter:(fh:exp.15, ub:#0)     ;5 clocks
00000167 16E3D02EBF800000                   ud      cnvTHCS.10  = convertToHexCharacter:(fh:sqrt.15, ub:#0)    ;5 clocks
00000168 16E3C82E97800000                   ud      cnvTHCS.9   = convertToHexCharacter:(fh:rtoi.15, ub:#0)    ;5 clocks
00000169 16E3C02ED7800000                   ud      cnvTHCS.8   = convertToHexCharacter:(fh:itof.15, ub:#0)    ;5 clocks
0000016A 16E3B82E87800000                   ud      cnvTHCS.7   = convertToHexCharacter:(fh:logb.15, ub:#0)    ;5 clocks
0000016B 16E3B02E8F800000                   ud      cnvTHCS.6   = convertToHexCharacter:(fh:scaleb.15, ub:#0)  ;5 clocks
0000016C 16E3A82E77800000                   ud      cnvTHCS.5   = convertToHexCharacter:(fh:nextUp.7, ub:#0)   ;5 clocks
0000016D 16E3A02E73800000                   ud      cnvTHCS.4   = convertToHexCharacter:(fh:nextDown.7, ub:#0) ;5 clocks
0000016E 16E3982E69800000                   ud      cnvTHCS.3   = convertToHexCharacter:(fh:minNum.3, ub:#0)   ;5 clocks
0000016F 16E3902E6B800000                   ud      cnvTHCS.2   = convertToHexCharacter:(fh:maxNum.3, ub:#0)   ;5 clocks
00000170 16E3882E65800000                   ud      cnvTHCS.1   = convertToHexCharacter:(fh:negate.3, ub:#0)   ;5 clocks
00000171 16E3802E61800000                   ud      cnvTHCS.0   = convertToHexCharacter:(fh:copySign.3, ub:#0) ;5 clocks

00000172 02E4786E3F8EE3F8                   fh      cnvFHCS.15  = convertFromHexCharacter:(ud:cnvTHCS.15, sd:cnvTHCS.15) ;7 clocks
00000173 02E4706E3F0EE3F0                   fh      cnvFHCS.14  = convertFromHexCharacter:(ud:cnvTHCS.14, sd:cnvTHCS.14) ;7 clocks
00000174 02E4686E3E8EE3E8                   fh      cnvFHCS.13  = convertFromHexCharacter:(ud:cnvTHCS.13, sd:cnvTHCS.13) ;7 clocks
00000175 02E4606E3E0EE3E0                   fh      cnvFHCS.12  = convertFromHexCharacter:(ud:cnvTHCS.12, sd:cnvTHCS.12) ;7 clocks
00000176 02E4586E3D8EE3D8                   fh      cnvFHCS.11  = convertFromHexCharacter:(ud:cnvTHCS.11, sd:cnvTHCS.11) ;7 clocks
00000177 02E4506E3D0EE3D0                   fh      cnvFHCS.10  = convertFromHexCharacter:(ud:cnvTHCS.10, sd:cnvTHCS.10) ;7 clocks
00000178 02E4486E3C8EE3C8                   fh      cnvFHCS.9   = convertFromHexCharacter:(ud:cnvTHCS.9 , sd:cnvTHCS.9 ) ;7 clocks
00000179 02E4406E3C0EE3C0                   fh      cnvFHCS.8   = convertFromHexCharacter:(ud:cnvTHCS.8 , sd:cnvTHCS.8 ) ;7 clocks
0000017A 02E4386E3B8EE3B8                   fh      cnvFHCS.7   = convertFromHexCharacter:(ud:cnvTHCS.7 , sd:cnvTHCS.7 ) ;7 clocks
0000017B 02E4306E3B0EE3B0                   fh      cnvFHCS.6   = convertFromHexCharacter:(ud:cnvTHCS.6 , sd:cnvTHCS.6 ) ;7 clocks
0000017C 02E4286E3A8EE3A8                   fh      cnvFHCS.5   = convertFromHexCharacter:(ud:cnvTHCS.5 , sd:cnvTHCS.5 ) ;7 clocks
0000017D 02E4206E3A0EE3A0                   fh      cnvFHCS.4   = convertFromHexCharacter:(ud:cnvTHCS.4 , sd:cnvTHCS.4 ) ;7 clocks
0000017E 02E4186E398EE398                   fh      cnvFHCS.3   = convertFromHexCharacter:(ud:cnvTHCS.3 , sd:cnvTHCS.3 ) ;7 clocks
0000017F 02E4106E390EE390                   fh      cnvFHCS.2   = convertFromHexCharacter:(ud:cnvTHCS.2 , sd:cnvTHCS.2 ) ;7 clocks
00000180 02E4086E388EE388                   fh      cnvFHCS.1   = convertFromHexCharacter:(ud:cnvTHCS.1 , sd:cnvTHCS.1 ) ;7 clocks
00000181 02E4006E380EE380                   fh      cnvFHCS.0   = convertFromHexCharacter:(ud:cnvTHCS.0 , sd:cnvTHCS.0 ) ;7 clocks

00000182 00FF084001000000                   ub      clas        = class:(fs:work_2)    ;1 clock

00000183 3400184001000002                   uw.rdx  work_3      = radix:(fs:work_2)    ;1 clock
```

```
00000184 10FF092E65800001                          isSignMinus(fh:negate.3)           ;1 clock
00000185 10FF092EBF800002                          isNormal(fh:sqrt.15)               ;1 clock
00000186 10FF092EBF800004                          isFinite(fh:sqrt.15)               ;1 clock
00000187 10FF092EBF800008                          isZero(fh:sqrt.15)                 ;1 clock
00000188 10FF092EBF800010                          isSubnormal(fh:sqrt.15)            ;1 clock
00000189 10FF092EBF800020                          isInfinite(fh:sqrt.15)             ;1 clock
0000018A 10FF092EBF800040                          isNaN(fh:sqrt.15)                  ;1 clock
0000018B 10FF092EBF800080                          isSignaling(fh:sqrt.15)            ;1 clock
0000018C 10FF092EBF800100                          isCanonical(fh:sqrt.15)            ;1 clock

0000018D 00FF1F4000840010                          compareSignalingEqual(fs:work_1, fs:work_2)            ;1 clock
0000018E 00FF1E4000840010                          compareQuietEqual(fs:work_1, fs:work_2)                ;1 clock
0000018F 00FF1D4000840010                          compareSignalingNotEqual(fs:work_1, fs:work_2)         ;1 clock
00000190 00FF1C4000840010                          compareQuietNotEqual(fs:work_1, fs:work_2)             ;1 clock
00000191 00FF1B4000840010                          compareSignalingGreater(fs:work_1, fs:work_2)          ;1 clock
00000192 00FF1A4000840010                          compareQuietGreater(fs:work_1, fs:work_2)              ;1 clock
00000193 00FF194000840010                          compareSignalingGreaterEqual(fs:work_1, fs:work_2)     ;1 clock
00000194 00FF184000840010                          compareQuietGreaterEqual(fs:work_1, fs:work_2)         ;1 clock
00000195 00FF174000840010                          compareSignalingLess(fs:work_1, fs:work_2)             ;1 clock
00000196 00FF164000840010                          compareQuietLess(fs:work_1, fs:work_2)                 ;1 clock
00000197 00FF154000840010                          compareSignalingLessEqual(fs:work_1, fs:work_2)        ;1 clock
00000198 00FF144000840010                          compareQuietLessEqual(fs:work_1, fs:work_2)            ;1 clock
00000199 00FF134000840010                          compareSignalingNotGreater(fs:work_1, fs:work_2)       ;1 clock
0000019A 00FF124000840010                          compareQuietNotGreater(fs:work_1, fs:work_2)           ;1 clock
0000019B 00FF114000840010                          compareSignalingLessUnordered(fs:work_1, fs:work_2)    ;1 clock
0000019C 00FF104000840010                          compareQuietLessUnordered(fs:work_1, fs:work_2)        ;1 clock
0000019D 00FF0F4000840010                          compareSignalingNotLess(fs:work_1, fs:work_2)          ;1 clock
0000019E 00FF0E4000840010                          compareQuietNotLess(fs:work_1, fs:work_2)              ;1 clock
0000019F 00FF0D4000840010                          compareSignalingGreaterUnordered(fs:work_1, fs:work_2) ;1 clock
000001A0 00FF0C4000840010                          compareQuietGreaterUnordered(fs:work_1, fs:work_2)     ;1 clock
000001A1 00FF0B4000840010                          compareQuietUnordered(fs:work_1, fs:work_2)            ;1 clock
000001A2 00FF0A4000840010                          compareQuietOrdered(fs:work_1, fs:work_2)              ;1 clock

000001A3 12FF8C0000020C00                          enableInt               ;1 clock
000001A4 12FF8C0000020800                          disableInt              ;1 clock
000001A5 12FF8C00000200C0                          setV                    ;1 clock
000001A6 12FF8C0000020080                          clearV                  ;1 clock
000001A7 12FF8C0000020030                          setN                    ;1 clock
000001A8 12FF8C0000020020                          clearN                  ;1 clock
000001A9 12FF8C000002000C                          setC                    ;1 clock
000001AA 12FF8C0000020008                          clearC                  ;1 clock
000001AB 12FF8C0000020003                          setZ                    ;1 clock
000001AC 12FF8C0000020002                          clearZ                  ;1 clock
000001AD 12FF8E0000000300                          setSubsInexact          ;1 clock
000001AE 12FF8E0000000200                          clearSubsInexact        ;1 clock
000001AF 12FF8E00000000C0                          setSubssubsUnderflow    ;1 clock
000001B0 12FF8E0000000080                          clearSubssubsUnderflow  ;1 clock
000001B1 12FF8E0000000030                          setsubsOverflow         ;1 clock
000001B2 12FF8E0000000020                          clearsubsOverflow       ;1 clock
000001B3 12FF8E000000000C                          setsubsDivByZero        ;1 clock
000001B4 12FF8E0000000008                          clearsubsDivByZero      ;1 clock
000001B5 12FF8E0000000003                          setsubsInvalid          ;1 clock
000001B6 12FF8E0000000002                          clearsubsInvalid        ;1 clock

000001B7 00FF074000840010                          totalOrder(fs:work_1, fs:work_2)      ;1 clock
000001B8 00FF064000840010                          totalOrderMag(fs:work_1, fs:work_2)   ;1 clock

000001B9 12FF8F0000000008                          setBinaryRoundingDirection(NEAREST)   ;1 clock
000001BA 12FF8F000000000C                          setBinaryRoundingDirection(AWAY)      ;1 clock
```

```
000001BB 00FE180FF8800000                         getBinaryRoundingDirection()          ;1 clock
000001BC 12FF8F0000000009                         setBinaryRoundingDirection(POSITIVE)  ;1 clock
000001BD 00FE086FF8800000                         saveModes()                           ;1 clock
000001BE 12FF8F000000000A                         setBinaryRoundingDirection(NEGATIVE)  ;1 clock
000001BF 12FF8F000000000B                         setBinaryRoundingDirection(ZERO)      ;1 clock
000001C0 02FF8F0FE0800000                         restoreModes(ub:savedModes)           ;1 clock
000001C1 00FF8F0000000000                         defaultModes()                        ;1 clock

000001C2 10FF040000000015                         raiseFlags(ub:#{invalid | overflow | inexact})  ;1 clock
000001C3 0000010000000000                         is754version1985()                      ;1 clock
000001C4 0000020000000000                         is754version2008()                      ;1 clock
000001C5 00FF006FF8800000                         saveAllFlags()                                                       ;1 clock
000001C6 10FF030000000001                         testFlags(ub:#invalid)                                               ;1 clock
000001C7 10FF050000000015                         lowerFlags(ub:#{invalid | overflow | inexact})                       ;1 clock
000001C8 10FF010FF000000D                         restoreFlags(ub: savedFlags, ub:#{invalid | overflow | underflow})   ;1 clock
000001C9 10FF04000000000A                         raiseFlags(ub:#{divByZero | underflow})                              ;1 clock
000001CA 10FF05000000000A                         lowerFlags(ub:#{divByZero | underflow})                              ;1 clock
000001CB 12FF8A0000000004                         raiseNoFlag(ub:#overflow)                                            ;1 clock
000001CC 14FF8A0000000014                         default(ub:#{overflow | inexact})                                    ;1 clock
000001CD 12FF8A0000000010                         raiseNoFlag(ub:#inexact)                                             ;1 clock

000001CE 10FF020FF000000D                         testSavedFlags(ub: savedFlags, ub:#{invalid | overflow | underflow}) ;1 clock

000001CF 12FF8B0000000015                         raiseSignals(ub:#{invalid | overflow | inexact})                     ;1 clock
000001D0 14FF8B0000000015                         lowerSignals(ub:#{invalid | overflow | inexact})                     ;1 clock
000001D1 12FF8B000000000A                         raiseSignals(ub:#{divByZero | underflow})                            ;1 clock
000001D2 14FF8B000000000A                         lowerSignals(ub:#{divByZero | underflow})                            ;1 clock

000001D3 14FF8D0000000015                         enableAltImmediateHandlers(ub:#{invalid | overflow | inexact})       ;1 clock
000001D4 12FF8D0000000015                         disableAltImmediateHandlers(ub:#{invalid | overflow | inexact})      ;1 clock
000001D5 14FF8D000000000A                         enableAltImmediateHandlers(ub:#{divByZero | underflow})              ;1 clock
000001D6 12FF8D000000000A                         disableAltImmediateHandlers(ub:#{divByZero | underflow})             ;1 clock
000001D7 14FF8D0000000002                         enableAltImmediateHandlers(ub:#divByZero)                            ;1 clock

000001D8 14FFA04FF887C0CB                         IF (754version1985) GOTO: goback              ;1 clock
000001D9 14FFA04FF887C0CA                         IF (754version2008) GOTO: goback              ;1 clock
000001DA 14FF984FF88800C9                         IF (signalingNaN) GOTO: goback                ;1 clock
000001DB 14FF984FF88840C8                         IF (quietNaN) GOTO: goback                    ;1 clock
000001DC 14FF984FF88880C7                         IF (negativeInfinity) GOTO: goback            ;1 clock
000001DD 14FF984FF888C0C6                         IF (negativeNormal) GOTO: goback              ;1 clock
000001DE 14FF984FF88900C5                         IF (negativeSubnormal) GOTO: goback           ;1 clock
000001DF 14FF984FF88940C4                         IF (negativeZero) GOTO: goback                ;1 clock
000001E0 14FF984FF88980C3                         IF (positiveZero) GOTO: goback                ;1 clock
000001E1 14FF984FF889C0C2                         IF (positiveSubnormal) GOTO: goback           ;1 clock
000001E2 14FF984FF88A00C1                         IF (positiveNormal) GOTO: goback              ;1 clock
000001E3 14FF984FF88A40C0                         IF (positiveInfinity) GOTO: goback            ;1 clock
000001E4 14FF984FF88A80BF                         IF (SignMinus) GOTO: goback                   ;1 clock
000001E5 14FF984FF88AC0BE                         IF (Normal) GOTO: goback                      ;1 clock
000001E6 14FF984FF88B00BD                         IF (Finite) GOTO: goback                      ;1 clock
000001E7 14FF984FF88B40BC                         IF (Zero) GOTO: goback                        ;1 clock
000001E8 14FF984FF88B80BB                         IF (Subnormal) GOTO: goback                   ;1 clock
000001E9 14FF984FF88BC0BA                         IF (Infinite) GOTO: goback                    ;1 clock
000001EA 14FF984FF88C00B9                         IF (NaN) GOTO: goback                         ;1 clock
000001EB 14FF984FF88C40B8                         IF (Signaling) GOTO: goback                   ;1 clock
000001EC 14FF984FF88C80B7                         IF (Canonical) GOTO: goback                   ;1 clock
000001ED 14FF984FF88CC0B6                         IF (totalOrder) GOTO: goback                  ;1 clock
000001EE 14FF984FF88D00B5                         IF (totalOrderMag) GOTO: goback               ;1 clock
000001EF 14FF984FF88D40B4                         IF (aFlagRaised) GOTO: goback                 ;1 clock
000001F0 14FF984FF88D80B3                         IF (compareTrue) GOTO: goback                 ;1 clock
```

```
000001F1 14FFA04FF88780B2                          IF NOT(754version1985) GOTO: goback            ;1 clock
000001F2 14FFA04FF88780B1                          IF NOT(754version2008) GOTO: goback            ;1 clock
000001F3 14FFA04FF88800B0                          IF NOT(signalingNaN) GOTO: goback              ;1 clock
000001F4 14FFA04FF88840AF                          IF NOT(quietNaN) GOTO: goback                  ;1 clock
000001F5 14FFA04FF88880AE                          IF NOT(negativeInfinity) GOTO: goback          ;1 clock
000001F6 14FFA04FF888C0AD                          IF NOT(negativeNormal) GOTO: goback            ;1 clock
000001F7 14FFA04FF88900AC                          IF NOT(negativeSubnormal) GOTO: goback         ;1 clock
000001F8 14FFA04FF88940AB                          IF NOT(negativeZero) GOTO: goback              ;1 clock
000001F9 14FFA04FF88980AA                          IF NOT(positiveZero) GOTO: goback              ;1 clock
000001FA 14FFA04FF889C0A9                          IF NOT(positiveSubnormal) GOTO: goback         ;1 clock
000001FB 14FFA04FF88A00A8                          IF NOT(positiveNormal) GOTO: goback            ;1 clock
000001FC 14FFA04FF88A40A7                          IF NOT(positiveInfinity) GOTO: goback          ;1 clock
000001FD 14FFA04FF88A80A6                          IF NOT(SignMinus) GOTO: goback                 ;1 clock
000001FE 14FFA04FF88AC0A5                          IF NOT(Normal) GOTO: goback                    ;1 clock
000001FF 14FFA04FF88B00A4                          IF NOT(Finite) GOTO: goback                    ;1 clock
00000200 14FFA04FF88B40A3                          IF NOT(Zero) GOTO: goback                      ;1 clock
00000201 14FFA04FF88B80A2                          IF NOT(Subnormal) GOTO: goback                 ;1 clock
00000202 14FFA04FF88BC0A1                          IF NOT(Infinite) GOTO: goback                  ;1 clock
00000203 14FFA04FF88C00A0                          IF NOT(NaN)  GOTO: goback                      ;1 clock
00000204 14FFA04FF88C409F                          IF NOT(Signaling) GOTO: goback                 ;1 clock
00000205 14FFA04FF88C809E                          IF NOT(Canonical) GOTO: goback                 ;1 clock
00000206 14FFA04FF88CC09D                          IF NOT(totalOrder) GOTO: goback                ;1 clock
00000207 14FFA04FF88D009C                          IF NOT(totalOrderMag) GOTO: goback             ;1 clock
00000208 14FFA04FF88D409B                          IF NOT(aFlagRaised) GOTO: goback               ;1 clock
00000209 14FFA04FF88D809A                          IF NOT(compareTrue) GOTO: goback               ;1 clock
0000020A 1CFF984FF8880099                          IF (signalingNaN) GOSUB: goback                ;1 clock
0000020B 1CFF984FF8884098                          IF (quietNaN) GOSUB: goback                    ;1 clock
0000020C 1CFF984FF8888097                          IF (negativeInfinity) GOSUB: goback            ;1 clock
0000020D 1CFF984FF888C096                          IF (negativeNormal) GOSUB: goback              ;1 clock
0000020E 1CFF984FF8890095                          IF (negativeSubnormal) GOSUB: goback           ;1 clock
0000020F 1CFF984FF8894094                          IF (negativeZero) GOSUB: goback                ;1 clock
00000210 1CFF984FF8898093                          IF (positiveZero) GOSUB: goback                ;1 clock
00000211 1CFF984FF889C092                          IF (positiveSubnormal) GOSUB: goback           ;1 clock
00000212 1CFF984FF88A0091                          IF (positiveNormal) GOSUB: goback              ;1 clock
00000213 1CFF984FF88A4090                          IF (positiveInfinity) GOSUB: goback            ;1 clock
00000214 1CFF984FF88A808F                          IF (SignMinus) GOSUB: goback                   ;1 clock
00000215 1CFF984FF88AC08E                          IF (Normal) GOSUB: goback                      ;1 clock
00000216 1CFF984FF88B008D                          IF (Finite) GOSUB: goback                      ;1 clock
00000217 1CFF984FF88B408C                          IF (Zero) GOSUB: goback                        ;1 clock
00000218 1CFF984FF88B808B                          IF (Subnormal) GOSUB: goback                   ;1 clock
00000219 1CFF984FF88BC08A                          IF (Infinite) GOSUB: goback                    ;1 clock
0000021A 1CFF984FF88C0089                          IF (NaN) GOSUB: goback                         ;1 clock
0000021B 1CFF984FF88C4088                          IF (Signaling) GOSUB: goback                   ;1 clock
0000021C 1CFF984FF88C8087                          IF (Canonical) GOSUB: goback                   ;1 clock
0000021D 1CFF984FF88CC086                          IF (totalOrder) GOSUB: goback                  ;1 clock
0000021E 1CFF984FF88D0085                          IF (totalOrderMag) GOSUB: goback               ;1 clock
0000021F 1CFF984FF88D4084                          IF (aFlagRaised) GOSUB: goback                 ;1 clock
00000220 1CFF984FF88D8083                          IF (compareTrue) GOSUB: goback                 ;1 clock
00000221 1CFFA04FF8880082                          IF NOT(signalingNaN) GOSUB: goback             ;1 clock
00000222 1CFFA04FF8884081                          IF NOT(quietNaN) GOSUB: goback                 ;1 clock
00000223 1CFFA04FF8888080                          IF NOT(negativeInfinity) GOSUB: goback         ;1 clock
00000224 1CFFA04FF888C07F                          IF NOT(negativeNormal) GOSUB: goback           ;1 clock
00000225 1CFFA04FF889007E                          IF NOT(negativeSubnormal) GOSUB: goback        ;1 clock
00000226 1CFFA04FF889407D                          IF NOT(negativeZero) GOSUB: goback             ;1 clock
00000227 1CFFA04FF889807C                          IF NOT(positiveZero) GOSUB: goback             ;1 clock
00000228 1CFFA04FF889C07B                          IF NOT(positiveSubnormal) GOSUB: goback        ;1 clock
00000229 1CFFA04FF88A007A                          IF NOT(positiveNormal) GOSUB: goback           ;1 clock
0000022A 1CFFA04FF88A4079                          IF NOT(positiveInfinity) GOSUB: goback         ;1 clock
```

```
0000022B 1CFFA04FF88A8078                          IF NOT(SignMinus) GOSUB: goback                ;1 clock
0000022C 1CFFA04FF88AC077                          IF NOT(Normal) GOSUB: goback                   ;1 clock
0000022D 1CFFA04FF88B0076                          IF NOT(Finite) GOSUB: goback                   ;1 clock
0000022E 1CFFA04FF88B4075                          IF NOT(Zero) GOSUB: goback                     ;1 clock
0000022F 1CFFA04FF88B8074                          IF NOT(Subnormal) GOSUB: goback                ;1 clock
00000230 1CFFA04FF88BC073                          IF NOT(Infinite) GOSUB: goback                 ;1 clock
00000231 1CFFA04FF88C0072                          IF NOT(NaN)  GOSUB: goback                     ;1 clock
00000232 1CFFA04FF88C4071                          IF NOT(Signaling) GOSUB: goback                ;1 clock
00000233 1CFFA04FF88C8070                          IF NOT(Canonical) GOSUB: goback                ;1 clock
00000234 1CFFA04FF88CC06F                          IF NOT(totalOrder) GOSUB: goback               ;1 clock
00000235 1CFFA04FF88D006E                          IF NOT(totalOrderMag) GOSUB: goback            ;1 clock
00000236 1CFFA04FF88D406D                          IF NOT(aFlagRaised) GOSUB: goback              ;1 clock
00000237 1CFFA04FF88D806C                          IF NOT(compareTrue) GOSUB: goback              ;1 clock

                                           ;integer operators
00000238 12DF982001825555                  uh      and.3 = and:(uh:work_3, uh:#0x5555)            ;2 clocks
00000239 12DF182001825555                  uh      or.3 = or:(uh:work_3, uh:#0x5555)              ;2 clocks
0000023A 12DE982001825555                  uh      xor.3 = xor:(uh:work_3, uh:#0x5555)            ;2 clocks
0000023B 12DE182001825555                  uh      add.3 = add:(uh:work_3, uh:#0x5555)            ;2 clocks
0000023C 12FF8C000002000C                          setC
0000023D 1ADE2020018A5555                  sh      add.4 = addc:(uh:work_3, sh:#0x5555)           ;2 clocks   signed add with carry
0000023E 12DD182001820055                  uh      sub.3 = sub:(uh:work_3, uh:#0x0055)            ;2 clocks
0000023F 12FF8C000002000C                          setC
00000240 1ADD2020018A0055                  sh      sub.4 = subb:(uh:work_3, sh:#0x0055)           ;2 clocks   signed subtract with borrow
00000241 12DC182001825555                  uh      mul.3 = mul:(uh:work_3, uh:#0x5555)            ;2 clocks
00000242 14DBF84DC1820055                  uw      div.15 = div:(uw:mul.3, uh:#0x0055)            ;11 clocks
00000243 12DA182001825555                  uh      min.3 = min:(uh:work_3, uh:#0x5555)            ;2 clocks
00000244 12DA982001825555                  uh      max.3 = max:(uh:work_3, uh:#0x5555)            ;2 clocks
00000245 12D9982001800001                  uh      bset.3 = bset:(uh:work_3, ub:#1)               ;2 clocks

00000246 12D9182D99800001                  uh      bclr.3 = bclr:(uh:bset.3, ub:#1)               ;2 clocks
00000247 12D9182D99800001                          . uh:bclr.3, uh:bset.3, ub:#1

00000248 10FF894DBF825555                          compare(uw:div.15, uh:#0x5555)                 ;1 clock
00000249 00FF894000840010                          compare(uw:work_1, uw:work_2)                  ;1 clock
0000024A 00FF2F4000840010                          . ub:compare, uw:work_1, uw:work_2

0000024B 14DB002001800000                  uw      shift.0 = shift:(uh:work_3, LEFT, 1)           ;2 clocks
0000024C 14DB082001800804                  uw      shift.1 = shift:(uh:work_3, RIGHT, 3)          ;2 clocks
0000024D 14DB102001802401                  uw      shift.2 = shift:(uh:work_3, LSL, 10)           ;2 clocks
0000024E 14DB182001801002                  uw      shift.3 = shift:(uh:work_3, ASL, 5)            ;2 clocks
0000024F 14DB202001801C03                  uw      shift.4 = shift:(uh:work_3, ROL, 8)            ;2 clocks
00000250 14DB282001801405                  uw      shift.5 = shift:(uh:work_3, LSR, 6)            ;2 clocks
00000251 14DB302001803406                  uw      shift.6 = shift:(uh:work_3, ASR, 14)           ;2 clocks
00000252 14DB382001804C07                  uw      shift.7 = shift:(uh:work_3, ROR, 20)           ;2 clocks

00000253 14DB002001800000                          . uw:shift.0, uh:work_3, LEFT, 1
00000254 14DB082001800804                          . uw:shift.1, uh:work_3, RIGHT, 3
00000255 14DB102001802401                          . uw:shift.2, uh:work_3, LSL, 10
00000256 14DB182001801002                          . uw:shift.3, uh:work_3, ASL, 5
00000257 14DB202001801C03                          . uw:shift.4, uh:work_3, ROL, 8
00000258 14DB282001801405                          . uw:shift.5, uh:work_3, LSR, 6
00000259 14DB302001803406                          . uw:shift.6, uh:work_3, ASR, 14
0000025A 14DB382001804C07                          . uw:shift.7, uh:work_3, ROR, 20

0000025B 06D8806E4F800000                  ud      endi.0 = endi:(ud:cnvTDCS.15)                  ;2 clocks
0000025C 06D8806E4F800000                          . ud:endi.0, ud:cnvTDCS.15

0000025D 36D7F840A5632504                  ud      cnvFBTA.15 = convertFromBinaryToASCII:(uw:#0xA5632504) ;2 clocks
```

```
0000025E 36D7F840A5632504                          . ud:cnvFBTA.15, uw:#0xA5632504
0000025F 04D7786D7F800000                  uw      cnvTBFA.15 = convertToBinaryFromASCII:(ud:cnvFBTA.15) ;2 clocks
00000260 04D7786D7F800000                          . uw:cnvTBFA.15, ud:cnvFBTA.15

00000261 14FFA04FF8800042                          IF (Z==1) GOTO: goback            ;1 clock
00000262 14FF984FF8800041                          IF (Z==0) GOTO: goback            ;1 clock
00000263 14FFA04FF8800040                          IF (A==B) GOTO: goback            ;1 clock
00000264 14FF984FF880003F                          IF (A!=B) GOTO: goback            ;1 clock
00000265 14FFA04FF880403E                          IF (C==1) GOTO: goback            ;1 clock
00000266 14FF984FF880403D                          IF (C==0) GOTO: goback            ;1 clock
00000267 14FFA04FF880803C                          IF (N==1) GOTO: goback            ;1 clock
00000268 14FF984FF880803B                          IF (N==0) GOTO: goback            ;1 clock
00000269 14FFA04FF880C03A                          IF (V==1) GOTO: goback            ;1 clock
0000026A 14FF984FF880C039                          IF (V==0) GOTO: goback            ;1 clock
0000026B 14FFA04FF8874038                          IF (A<B)  GOTO: goback            ;1 clock
0000026C 14FF984FF8874037                          IF (A>=B) GOTO: goback            ;1 clock
0000026D 14FFA04FF8878036                          IF (A<=B) GOTO: goback            ;1 clock
0000026E 14FF984FF8878035                          IF (A>B)  GOTO: goback            ;1 clock

0000026F 1CFFA04FF8800034                          IF (Z==1) GOSUB: goback           ;1 clock
00000270 1CFF984FF8800033                          IF (Z==0) GOSUB: goback           ;1 clock
00000271 1CFFA04FF8800032                          IF (A==B) GOSUB: goback           ;1 clock
00000272 1CFF984FF8800031                          IF (A!=B) GOSUB: goback           ;1 clock
00000273 1CFFA04FF8804030                          IF (C==1) GOSUB: goback           ;1 clock
00000274 1CFF984FF880402F                          IF (C==0) GOSUB: goback           ;1 clock
00000275 1CFFA04FF880802E                          IF (N==1) GOSUB: goback           ;1 clock
00000276 1CFF984FF880802D                          IF (N==0) GOSUB: goback           ;1 clock
00000277 1CFFA04FF880C02C                          IF (V==1) GOSUB: goback           ;1 clock
00000278 1CFF984FF880C02B                          IF (V==0) GOSUB: goback           ;1 clock
00000279 1CFFA04FF887402A                          IF (A<B)  GOSUB: goback           ;1 clock
0000027A 1CFF984FF8874029                          IF (A>=B) GOSUB: goback           ;1 clock
0000027B 1CFFA04FF8878028                          IF (A<=B) GOSUB: goback           ;1 clock
0000027C 1CFF984FF8878027                          IF (A>B)  GOSUB: goback           ;1 clock

0000027D 14FF984001820026                          IF (uw:work_3:[bit8]==0) GOTO: goback    ;1 clock
0000027E 14FFA0400181C025                          IF (uw:work_3:[bit7]==1) GOTO: goback    ;1 clock
0000027F 1CFF984001818024                          IF (uw:work_3:[bit6]==0) GOSUB: goback   ;1 clock
00000280 1CFFA04001814023                          IF (uw:work_3:[bit5]==1) GOSUB: goback   ;1 clock

00000281 14FF982001820022                          btbc uh:work_3, 8, goback
00000282 14FFA0200181C021                          btbs uh:work_3, 7, goback
00000283 14FF982001818020                          btbc uh:work_3, 6, goback
00000284 14FFA0200181401F                          btbs uh:work_3, 5, goback

00000285 12FF98400182001E                          . uh:pcc, uw:work_3, 8, goback
00000286 12FFA0400181C01D                          . uh:pcs, uw:work_3, 7, goback
00000287 12FF98400181801C                          . uh:pcc, uw:work_3, 6, goback
00000288 12FFA0400181401B                          . uh:pcs, uw:work_3, 5, goback

00000289 14FF700000040003                          FOR (LPCNT0 = uw:#3) (       ;1 clock
0000028A 14FFA04FF8878000      loop_0:                 nop                      ;1 clock
0000028B 14FFA04FF8878000                              nop                      ;1 clock
0000028C 14FFA04FF7043FFE                          NEXT LPCNT0 GOTO: loop_0 )   ;1 clock

0000028D 14FF700000040003                          . uw:LPCNT0, uw:#3
0000028E 14FFA06FF8878000      loop_1:             . uw:PCS, ud:STATUS, NEVER, loop_1
0000028F 14FFA06FF887BFFF                          . uw:PCS, ud:STATUS, NEVER, loop_1
00000290 14FFA06FF7043FFE                          . uw:PCS, ud:LPCNT0, 16, loop_1
```

```
00000291 14FF780000040003                        FOR (LPCNT1 = uw:#3) (       ;1 clock
00000292 14FFA04FF8878000      loop_2:               nop                      ;1 clock
00000293 14FFA04FF8878000                            nop                      ;1 clock
00000294 14FFA04FF7843FFE                        NEXT LPCNT1 GOTO: loop_2 )   ;1 clock

00000295 14FFA04FF887C00E                        GOTO goback                  ;1 clock
00000296 1CFFA04FF887C00D                        GOSUB goback                 ;1 clock
                               ;                  RETURN

00000297 14FFB0000004E500                  uw      AR0 = uw:#cnvFDCS.0        ;load AR0 with source address
00000298 14FFB8000004E380                  uw      AR1 = uw:#cnvTHCS.0        ;load AR1 with destination address
00000299 12FF80000002000F                          REPEAT uh:#15
0000029A 1300413004000000                  uh      *AR1++[8] = convertToHexCharacter:(uh:*AR0++[8], ub:#0)

0000029B 14FFB0000004E380                  uw      AR0 = uw:#cnvTHCS.0        ;load AR0 with source address
0000029C 14FFB8000004E400                  uw      AR1 = uw:#cnvFHCS.0        ;load AR1 with destination address
0000029D 14FFC0000004000F                  uw      AR2 = uw:#15
0000029E 02FF801800200000                          REPEAT [AR2]
0000029F 03004170040F8000                  uh      *AR1++[8] = convertFromHexCharacter:(ud:*AR0++[8], sd:*AR0[0])   ;128-bit (16-byte character string) move (* 16 of them)

                                                   ;test divide by zero alternate immediate exception handling and exception capture registers
000002A0 12EC704000840000                  fh      fdiv.14 = division:(fs:work_1, fs:#0x00000000)
000002A1 02E6602EC7000000                  fh      copy.0 = fh:fdiv.14


000002A2 14FFA04FF887FE67                          GOTO done                      ;branch to done

000002A3 02FFA82FF9000000      goback:     uh      PC = uh:PC_COPY


000002A4 0B7FC72FF9000000      NMI_:       sh      *SP--[8] = uh:PC_COPY       ;save return address from non-maskable interrupt (time-out timer in this instance)
000002A5 14FF68000004EA60                  uw      TIMER = uw:#60000           ;put a new value in the timer
000002A6 14FFA04FF8878000                          nop
000002A7 0AFFA83004700000                  sh      PC = uh:*SP++[8]            ;return from interrupt

000002A8 0B7FC72FF9000000      INV_:       sh      *SP--[8] = uh:PC_COPY       ;save return address from floating-point invalid operation exception, which is maskable
000002A9 0600206FF4000000                  ud      capt0_save = ud:CAPTURE0    ;read out CAPTURE0 register and save it
000002AA 0600286FF4800000                  ud      capt1_save = ud:CAPTURE1    ;read out CAPTURE1 register and save it
000002AB 0600306FF5000000                  ud      capt2_save = ud:CAPTURE2    ;read out CAPTURE2 register and save it
000002AC 0600386FF5800000                  ud      capt3_save = ud:CAPTURE3    ;read out CAPTURE3 register and save it
000002AD 14FF8B0000000001                          lowerSignals(ub:#invalid)   ;lower invalid signal
000002AE 10FF040000000001                          raiseFlags(ub:#invalid)     ;raise invalid flag
000002AF 14FF68000004EA60                  uw      TIMER = uw:#60000           ;put a new value in the timer
000002B0 0AFFA83004700000                  sh      PC = uh:*SP++[8]            ;return from interrupt

000002B1 0B7FC72FF9000000      DIVx0_:     sh      *SP--[8] = uh:PC_COPY       ;save return address from floating-point divide by 0 exception, which is maskable
000002B2 0600206FF4000000                  ud      capt0_save = ud:CAPTURE0    ;read out CAPTURE0 register and save it
000002B3 0600286FF4800000                  ud      capt1_save = ud:CAPTURE1    ;read out CAPTURE1 register and save it
000002B4 0600306FF5000000                  ud      capt2_save = ud:CAPTURE2    ;read out CAPTURE2 register and save it
000002B5 0600386FF5800000                  ud      capt3_save = ud:CAPTURE3    ;read out CAPTURE3 register and save it
000002B6 14FF8B0000000002                          lowerSignals(ub:#divByZero) ;lower divByZero signal
000002B7 10FF040000000002                          raiseFlags(ub:#divByZero)   ;raise divByZero flag
000002B8 14FF68000004EA60                  uw      TIMER = uw:#60000           ;put a new value in the timer
000002B9 0AFFA83004700000                  sh      PC = uh:*SP++[8]            ;return from interrupt

000002BA 0B7FC72FF9000000      OVFL_:      sh      *SP--[8] = uh:PC_COPY       ;save return address from floating-point overflow exception, which is maskable
000002BB 0600206FF4000000                  ud      capt0_save = ud:CAPTURE0    ;read out CAPTURE0 register and save it
000002BC 0600286FF4800000                  ud      capt1_save = ud:CAPTURE1    ;read out CAPTURE1 register and save it
000002BD 0600306FF5000000                  ud      capt2_save = ud:CAPTURE2    ;read out CAPTURE2 register and save it
```

```
000002BE 0600386FF5800000                 ud      capt3_save = ud:CAPTURE3    ;read out CAPTURE3 register and save it
000002BF 14FF8B0000000004                         lowerSignals(ub:#overflow)  ;lower overflow signal
000002C0 10FF040000000004                         raiseFlags(ub:#overflow)    ;raise overflow flag
000002C1 14FF68000004EA60                 uw      TIMER = uw:#60000           ;put a new value in the timer
000002C2 0AFFA83004700000                 sh      PC = uh:*SP++[8]            ;return from interrupt

000002C3 0B7FC72FF9000000      UNFL_:     sh      *SP--[8] = uh:PC_COPY       ;save return address from floating-point underflow exception, which is maskable
000002C4 0600206FF4000000                 ud      capt0_save = ud:CAPTURE0    ;read out CAPTURE0 register and save it
000002C5 0600286FF4800000                 ud      capt1_save = ud:CAPTURE1    ;read out CAPTURE1 register and save it
000002C6 0600306FF5000000                 ud      capt2_save = ud:CAPTURE2    ;read out CAPTURE2 register and save it
000002C7 0600386FF5800000                 ud      capt3_save = ud:CAPTURE3    ;read out CAPTURE3 register and save it
000002C8 14FF8B0000000008                         lowerSignals(ub:#underflow)  ;lower underflow signal
000002C9 10FF040000000008                         raiseFlags(ub:#underflow)    ;raise underflow flag
000002CA 14FF68000004EA60                 uw      TIMER = uw:#60000           ;put a new value in the timer
000002CB 0AFFA83004700000                 sh      PC = uh:*SP++[8]            ;return from interrupt

000002CC 0B7FC72FF9000000      INEXT_:    sh      *SP--[8] = uh:PC_COPY       ;save return address from floating-point inexact exception, which is maskable
000002CD 0600206FF4000000                 ud      capt0_save = ud:CAPTURE0    ;read out CAPTURE0 register and save it
000002CE 0600286FF4800000                 ud      capt1_save = ud:CAPTURE1    ;read out CAPTURE1 register and save it
000002CF 0600306FF5000000                 ud      capt2_save = ud:CAPTURE2    ;read out CAPTURE2 register and save it
000002D0 0600386FF5800000                 ud      capt3_save = ud:CAPTURE3    ;read out CAPTURE3 register and save it
000002D1 14FF8B0000000010                         lowerSignals(ub:#inexact)   ;lower inexact signal
000002D2 10FF040000000010                         raiseFlags(ub:#inexact)     ;raise inexact flag
000002D3 14FF68000004EA60                 uw      TIMER = uw:#60000           ;put a new value in the timer
000002D4 0AFFA83004700000                 sh      PC = uh:*SP++[8]            ;return from interrupt

000002D5 0B7FC72FF9000000      IRQ_:      sh      *SP--[8] = uh:PC_COPY       ;save return address (general-purpose, maskable interrupt)
000002D6 14FF68000004EA60                 uw      TIMER = uw:#60000           ;put a new value in the timer
000002D7 14FFA04FF8878000                         nop
000002D8 0AFFA83004700000                 sh      PC = uh:*SP++[8]            ;return from interrupt
000002D9                       progend:
00000000                                  end
```

```
00000000  ABS                0000E620  ABS.0                0000E628  ABS.1
0000E630  ABS.2              0000E638  ABS.3                00000000  ADD
0000DE00  ADD.0              0000DE08  ADD.1                0000DE50  ADD.10
0000DE58  ADD.11             0000DE60  ADD.12               0000DE68  ADD.13
0000DE70  ADD.14             0000DE78  ADD.15               0000DE10  ADD.2
0000DE18  ADD.3              0000DE20  ADD.4                0000DE28  ADD.5
0000DE30  ADD.6              0000DE38  ADD.7                0000DE40  ADD.8
0000DE48  ADD.9              00000000  ADDC                 0000DD80  ADDC.0
0000DD88  ADDC.1             0000DDD0  ADDC.10              0000DDD8  ADDC.11
0000DDE0  ADDC.12            0000DDE8  ADDC.13              0000DDF0  ADDC.14
0000DDF8  ADDC.15            0000DD90  ADDC.2               0000DD98  ADDC.3
0000DDA0  ADDC.4             0000DDA8  ADDC.5               0000DDB0  ADDC.6
0000DDB8  ADDC.7             0000DDC0  ADDC.8               0000DDC8  ADDC.9
00000000  ADDITION           00000035  AFLAGRAISED          0000000C  ALTIMMDIVBYZERO
0000000F  ALTIMMINEXACT      0000000B  ALTIMMINVALID        0000000D  ALTIMMOVERFLOW
0000000E  ALTIMMUNDERFLOW    0000001F  ALWAYS               00000000  AND
0000DF80  AND.0              0000DF88  AND.1                0000DFD0  AND.10
0000DFD8  AND.11             0000DFE0  AND.12               0000DFE8  AND.13
0000DFF0  AND.14             0000DFF8  AND.15               0000DF90  AND.2
0000DF98  AND.3              0000DFA0  AND.4                0000DFA8  AND.5
0000DFB0  AND.6              0000DFB8  AND.7                0000DFC0  AND.8
0000DFC8  AND.9              0000FFB0  AR0                  0000FFB8  AR1
0000FFC0  AR2                0000FFC8  AR3                  0000FFD0  AR4
0000FFD8  AR5                0000FFE0  AR6                  0000003E  AWAY
00000000  BCLR               0000D900  BCLR.0               0000D908  BCLR.1
0000D950  BCLR.10            0000D958  BCLR.11              0000D960  BCLR.12
0000D968  BCLR.13            0000D970  BCLR.14              0000D978  BCLR.15
0000D910  BCLR.2             0000D918  BCLR.3               0000D920  BCLR.4
0000D928  BCLR.5             0000D930  BCLR.6               0000D938  BCLR.7
0000D940  BCLR.8             0000D948  BCLR.9               00000000  BCND
00000000  BIT0               00000001  BIT1                 0000000A  BIT10
0000000B  BIT11              0000000C  BIT12                0000000D  BIT13
0000000E  BIT14              0000000F  BIT15                00000010  BIT16
00000011  BIT17              00000012  BIT18                00000013  BIT19
00000002  BIT2               00000014  BIT20                00000015  BIT21
00000016  BIT22              00000017  BIT23                00000018  BIT24
00000019  BIT25              0000001A  BIT26                0000001B  BIT27
0000001C  BIT28              0000001D  BIT29                00000003  BIT3
0000001E  BIT30              0000001F  BIT31                00000020  BIT32
00000021  BIT33              00000022  BIT34                00000023  BIT35
00000024  BIT36              00000025  BIT37                00000026  BIT38
00000027  BIT39              00000004  BIT4                 00000028  BIT40
00000029  BIT41              0000002A  BIT42                0000002B  BIT43
0000002C  BIT44              0000002D  BIT45                0000002E  BIT46
0000002F  BIT47              00000030  BIT48                00000031  BIT49
00000005  BIT5               00000032  BIT50                00000033  BIT51
00000034  BIT52              00000035  BIT53                00000036  BIT54
00000037  BIT55              00000038  BIT56                00000039  BIT57
0000003A  BIT58              0000003B  BIT59                00000006  BIT6
0000003C  BIT60              0000003D  BIT61                0000003E  BIT62
0000003F  BIT63              00000007  BIT7                 00000008  BIT8
00000009  BIT9               00000000  BITBUCKET            00000000  BSET
0000D980  BSET.0             0000D988  BSET.1               0000D9D0  BSET.10
0000D9D8  BSET.11            0000D9E0  BSET.12              0000D9E8  BSET.13
0000D9F0  BSET.14            0000D9F8  BSET.15              0000D990  BSET.2
0000D998  BSET.3             0000D9A0  BSET.4               0000D9A8  BSET.5
0000D9B0  BSET.6             0000D9B8  BSET.7               0000D9C0  BSET.8
0000D9C8  BSET.9             0000FF98  BTBC                 0000FFA0  BTBS
00000000  BUBL               0000D800  BUBL.0               0000D808  BUBL.1
```

```
0000D850  BUBL.10            0000D858  BUBL.11            0000D860  BUBL.12
0000D868  BUBL.13            0000D870  BUBL.14            0000D878  BUBL.15
0000D810  BUBL.2             0000D818  BUBL.3             0000D820  BUBL.4
0000D828  BUBL.5             0000D830  BUBL.6             0000D838  BUBL.7
0000D840  BUBL.8             0000D848  BUBL.9             00000001  C
00000032  CANONICAL          00000020  CAPT0_SAVE         00000028  CAPT1_SAVE
00000030  CAPT2_SAVE         00000038  CAPT3_SAVE         0000FF40  CAPTURE0
0000FF48  CAPTURE1           0000FF50  CAPTURE2           0000FF58  CAPTURE3
0000FF08  CLAS               00000000  CLASS              0000FF1E  CMPQE
0000FF1A  CMPQG              0000FF18  CMPQGE             0000FF0C  CMPQGU
0000FF16  CMPQL              0000FF14  CMPQLE             0000FF10  CMPQLU
0000FF1C  CMPQNE             0000FF12  CMPQNG             0000FF0E  CMPQNL
0000FF0A  CMPQO              0000FF0B  CMPQU              0000FF1F  CMPSE
0000FF1B  CMPSG              0000FF19  CMPSGE             0000FF0D  CMPSGU
0000FF17  CMPSL              0000FF15  CMPSLE             0000FF11  CMPSLU
0000FF1D  CMPSNE             0000FF13  CMPSNG             0000FF0F  CMPSNL
0000D780  CNVFBTA.0          0000D788  CNVFBTA.1          0000D7D0  CNVFBTA.10
0000D7D8  CNVFBTA.11         0000D7E0  CNVFBTA.12         0000D7E8  CNVFBTA.13
0000D7F0  CNVFBTA.14         0000D7F8  CNVFBTA.15         0000D790  CNVFBTA.2
0000D798  CNVFBTA.3          0000D7A0  CNVFBTA.4          0000D7A8  CNVFBTA.5
0000D7B0  CNVFBTA.6          0000D7B8  CNVFBTA.7          0000D7C0  CNVFBTA.8
0000D7C8  CNVFBTA.9          0000E500  CNVFDCS.0          0000E508  CNVFDCS.1
0000E550  CNVFDCS.10         0000E558  CNVFDCS.11         0000E560  CNVFDCS.12
0000E568  CNVFDCS.13         0000E570  CNVFDCS.14         0000E578  CNVFDCS.15
0000E510  CNVFDCS.2          0000E518  CNVFDCS.3          0000E520  CNVFDCS.4
0000E528  CNVFDCS.5          0000E530  CNVFDCS.6          0000E538  CNVFDCS.7
0000E540  CNVFDCS.8          0000E548  CNVFDCS.9          0000E400  CNVFHCS.0
0000E408  CNVFHCS.1          0000E450  CNVFHCS.10         0000E458  CNVFHCS.11
0000E460  CNVFHCS.12         0000E468  CNVFHCS.13         0000E470  CNVFHCS.14
0000E478  CNVFHCS.15         0000E410  CNVFHCS.2          0000E418  CNVFHCS.3
0000E420  CNVFHCS.4          0000E428  CNVFHCS.5          0000E430  CNVFHCS.6
0000E438  CNVFHCS.7          0000E440  CNVFHCS.8          0000E448  CNVFHCS.9
0000D700  CNVTBFA.0          0000D708  CNVTBFA.1          0000D750  CNVTBFA.10
0000D758  CNVTBFA.11         0000D760  CNVTBFA.12         0000D768  CNVTBFA.13
0000D770  CNVTBFA.14         0000D778  CNVTBFA.15         0000D710  CNVTBFA.2
0000D718  CNVTBFA.3          0000D720  CNVTBFA.4          0000D728  CNVTBFA.5
0000D730  CNVTBFA.6          0000D738  CNVTBFA.7          0000D740  CNVTBFA.8
0000D748  CNVTBFA.9          0000E480  CNVTDCS.0          0000E488  CNVTDCS.1
0000E4D0  CNVTDCS.10         0000E4D8  CNVTDCS.11         0000E4E0  CNVTDCS.12
0000E4E8  CNVTDCS.13         0000E4F0  CNVTDCS.14         0000E4F8  CNVTDCS.15
0000E490  CNVTDCS.2          0000E498  CNVTDCS.3          0000E4A0  CNVTDCS.4
0000E4A8  CNVTDCS.5          0000E4B0  CNVTDCS.6          0000E4B8  CNVTDCS.7
0000E4C0  CNVTDCS.8          0000E4C8  CNVTDCS.9          0000E380  CNVTHCS.0
0000E388  CNVTHCS.1          0000E3D0  CNVTHCS.10         0000E3D8  CNVTHCS.11
0000E3E0  CNVTHCS.12         0000E3E8  CNVTHCS.13         0000E3F0  CNVTHCS.14
0000E3F8  CNVTHCS.15         0000E390  CNVTHCS.2          0000E398  CNVTHCS.3
0000E3A0  CNVTHCS.4          0000E3A8  CNVTHCS.5          0000E3B0  CNVTHCS.6
0000E3B8  CNVTHCS.7          0000E3C0  CNVTHCS.8          0000E3C8  CNVTHCS.9
0000FF2F  COMPARE            00000036  COMPARETRUE        000000FE  CONSTANTS
0000E980  CONV.0             0000E988  CONV.1             0000E9D0  CONV.10
0000E9D8  CONV.11            0000E9E0  CONV.12            0000E9E8  CONV.13
0000E9F0  CONV.14            0000E9F8  CONV.15            0000E990  CONV.2
0000E998  CONV.3             0000E9A0  CONV.4             0000E9A8  CONV.5
0000E9B0  CONV.6             0000E9B8  CONV.7             0000E9C0  CONV.8
0000E9C8  CONV.9             00000000  CONVERTFORMAT      00000000  CONVERTFROMBINARYTOASCII
00000000  CONVERTFROMDECIMALCHARACTER  00000000  CONVERTFROMHEXCHARACTER  00000000  CONVERTFROMINT
00000000  CONVERTTOBINARYFROMASCII  00000000  CONVERTTODECIMALCHARACTER  00000000  CONVERTTOHEXCHARACTER
00000000  CONVERTTOINTEGEREXACTTIESTOAWAY  00000000  CONVERTTOINTEGEREXACTTIESTOEVEN  00000000  CONVERTTOINTEGEREXACTTOWARDNEGATIVE
00000000  CONVERTTOINTEGEREXACTTOWARDPOSITIVE  00000000  CONVERTTOINTEGEREXACTTOWARDZERO  00000000  CONVERTTOINTEGERTIESTOAWAY
```

```
00000000  CONVERTTOINTEGERTIESTOEVEN  00000000  CONVERTTOINTEGERTOWARDNEGATIVE  00000000  CONVERTTOINTEGERTOWARDPOSITIVE
00000000  CONVERTTOINTEGERTOWARDZERO  00000000  COPY              0000E660  COPY.0
0000E668  COPY.1              0000E670  COPY.2              0000E678  COPY.3
00000000  COPYSIGN            0000E600  COPYSIGN.0          0000E608  COPYSIGN.1
0000E610  COPYSIGN.2          0000E618  COPYSIGN.3          00000000  COSD
0000E5C0  COSD.0              0000E5C8  COSD.1              0000E5D0  COSD.2
0000E5D8  COSD.3              00000000  COTD                0000E580  COTD.0
0000E588  COTD.1              0000E590  COTD.2              0000E598  COTD.3
0000FF60  CREG                00000000  DBNZ                00000000  DIV
0000DB80  DIV.0               0000DB88  DIV.1               0000DBD0  DIV.10
0000DBD8  DIV.11              0000DBE0  DIV.12              0000DBE8  DIV.13
0000DBF0  DIV.14              0000DBF8  DIV.15              0000DB90  DIV.2
0000DB98  DIV.3               0000DBA0  DIV.4               0000DBA8  DIV.5
0000DBB0  DIV.6               0000DBB8  DIV.7               0000DBC0  DIV.8
0000DBC8  DIV.9               00000007  DIVBY0FLAG          00000016  DIVBY0SIGNAL
00000002  DIVBYZERO           00000000  DIVISION            000002B1  DIVX0_
0000FEE0  DIVX0_VECT          00000109  DONE                00000004  DONE_BIT
00000000  ENDI                0000D880  ENDI.0              0000D888  ENDI.1
0000D8D0  ENDI.10             0000D8D8  ENDI.11             0000D8E0  ENDI.12
0000D8E8  ENDI.13             0000D8F0  ENDI.14             0000D8F8  ENDI.15
0000D890  ENDI.2              0000D898  ENDI.3              0000D8A0  ENDI.4
0000D8A8  ENDI.5              0000D8B0  ENDI.6              0000D8B8  ENDI.7
0000D8C0  ENDI.8              0000D8C8  ENDI.9              00000005  EXCSOURCE
00000000  EXP                 0000EA00  EXP.0               0000EA08  EXP.1
0000EA50  EXP.10              0000EA58  EXP.11              0000EA60  EXP.12
0000EA68  EXP.13              0000EA70  EXP.14              0000EA78  EXP.15
0000EA10  EXP.2               0000EA18  EXP.3               0000EA20  EXP.4
0000EA28  EXP.5               0000EA30  EXP.6               0000EA38  EXP.7
0000EA40  EXP.8               0000EA48  EXP.9               0000EE80  FADD.0
0000EE88  FADD.1              0000EED0  FADD.10             0000EED8  FADD.11
0000EEE0  FADD.12             0000EEE8  FADD.13             0000EEF0  FADD.14
0000EEF8  FADD.15             0000EE90  FADD.2              0000EE98  FADD.3
0000EEA0  FADD.4              0000EEA8  FADD.5              0000EEB0  FADD.6
0000EEB8  FADD.7              0000EEC0  FADD.8              0000EEC8  FADD.9
00000003  FD                  0000EC00  FDIV.0              0000EC08  FDIV.1
0000EC50  FDIV.10             0000EC58  FDIV.11             0000EC60  FDIV.12
0000EC68  FDIV.13             0000EC70  FDIV.14             0000EC78  FDIV.15
0000EC10  FDIV.2              0000EC18  FDIV.3              0000EC20  FDIV.4
0000EC28  FDIV.5              0000EC30  FDIV.6              0000EC38  FDIV.7
0000EC40  FDIV.8              0000EC48  FDIV.9              00000001  FH
0000002C  FINITE              0000EB00  FMA.0               0000EB08  FMA.1
0000EB50  FMA.10              0000EB58  FMA.11              0000EB60  FMA.12
0000EB68  FMA.13              0000EB70  FMA.14              0000EB78  FMA.15
0000EB10  FMA.2               0000EB18  FMA.3               0000EB20  FMA.4
0000EB28  FMA.5               0000EB30  FMA.6               0000EB38  FMA.7
0000EB40  FMA.8               0000EB48  FMA.9               0000ED80  FMUL.0
0000ED88  FMUL.1              0000EDD0  FMUL.10             0000EDD8  FMUL.11
0000EDE0  FMUL.12             0000EDE8  FMUL.13             0000EDF0  FMUL.14
0000EDF8  FMUL.15             0000ED90  FMUL.2              0000ED98  FMUL.3
0000EDA0  FMUL.4              0000EDA8  FMUL.5              0000EDB0  FMUL.6
0000EDB8  FMUL.7              0000EDC0  FMUL.8              0000EDC8  FMUL.9
00000002  FS                  0000EE00  FSUB.0              0000EE08  FSUB.1
0000EE50  FSUB.10             0000EE58  FSUB.11             0000EE60  FSUB.12
0000EE68  FSUB.13             0000EE70  FSUB.14             0000EE78  FSUB.15
0000EE10  FSUB.2              0000EE18  FSUB.3              0000EE20  FSUB.4
0000EE28  FSUB.5              0000EE30  FSUB.6              0000EE38  FSUB.7
0000EE40  FSUB.8              0000EE48  FSUB.9              0000EC80  FTOI.0
0000EC88  FTOI.1              0000ECD0  FTOI.10             0000ECD8  FTOI.11
0000ECE0  FTOI.12             0000ECE8  FTOI.13             0000ECF0  FTOI.14
```

```
0000ECF8  FTOI.15            0000EC90  FTOI.2             0000EC98  FTOI.3
0000ECA0  FTOI.4             0000ECA8  FTOI.5             0000ECB0  FTOI.6
0000ECB8  FTOI.7             0000ECC0  FTOI.8             0000ECC8  FTOI.9
00000000  FUSEDMULTIPLYADD   000002A3  GOBACK             00000010  INEXACT
000002CC  INEXT_             0000FEC8  INEXT_VECT         0000002F  INFINITE
00000001  INVALID            00000006  INVFLAG            00000015  INVSIGNAL
000002A8  INV_               0000FEE8  INV_VECT           0000001B  IRQ
0000001A  IRQEN              000002D5  IRQ_               0000FEF0  IRQ_VECT
0000FF09  IS                 00000000  ISCANONICAL        00000000  ISFINITE
00000000  ISINFINITE         00000000  ISNAN              00000000  ISNORMAL
00000000  ISSIGNALING        00000000  ISSIGNMINUS        00000000  ISSUBNORMAL
00000000  ISZERO             0000ED00  ITOF.0             0000ED08  ITOF.1
0000ED50  ITOF.10            0000ED58  ITOF.11            0000ED60  ITOF.12
0000ED68  ITOF.13            0000ED70  ITOF.14            0000ED78  ITOF.15
0000ED10  ITOF.2             0000ED18  ITOF.3             0000ED20  ITOF.4
0000ED28  ITOF.5             0000ED30  ITOF.6             0000ED38  ITOF.7
0000ED40  ITOF.8             0000ED48  ITOF.9             00000100  LOAD_VECTS
00000000  LOG                0000EA80  LOG.0              0000EA88  LOG.1
0000EAD0  LOG.10             0000EAD8  LOG.11             0000EAE0  LOG.12
0000EAE8  LOG.13             0000EAF0  LOG.14             0000EAF8  LOG.15
0000EA90  LOG.2              0000EA98  LOG.3              0000EAA0  LOG.4
0000EAA8  LOG.5              0000EAB0  LOG.6              0000EAB8  LOG.7
0000EAC0  LOG.8              0000EAC8  LOG.9              00000000  LOGB
0000E800  LOGB.0             0000E808  LOGB.1             0000E850  LOGB.10
0000E858  LOGB.11            0000E860  LOGB.12            0000E868  LOGB.13
0000E870  LOGB.14            0000E878  LOGB.15            0000E810  LOGB.2
0000E818  LOGB.3             0000E820  LOGB.4             0000E828  LOGB.5
0000E830  LOGB.6             0000E838  LOGB.7             0000E840  LOGB.8
0000E848  LOGB.9             0000028A  LOOP_0             0000028E  LOOP_1
00000292  LOOP_2             00000000  LOWERFLAGS         0000FF05  LOWFLG
0000FF70  LPCNT0             0000FF78  LPCNT1             00000000  MAX
0000DA80  MAX.0              0000DA88  MAX.1              0000DAD0  MAX.10
0000DAD8  MAX.11             0000DAE0  MAX.12             0000DAE8  MAX.13
0000DAF0  MAX.14             0000DAF8  MAX.15             0000DA90  MAX.2
0000DA98  MAX.3              0000DAA0  MAX.4              0000DAA8  MAX.5
0000DAB0  MAX.6              0000DAB8  MAX.7              0000DAC0  MAX.8
0000DAC8  MAX.9              00000000  MAXNUM             0000E6A0  MAXNUM.0
0000E6A8  MAXNUM.1           0000E6B0  MAXNUM.2           0000E6B8  MAXNUM.3
0000E6E0  MAXNUMMAG.0        0000E6E8  MAXNUMMAG.1        0000E6F0  MAXNUMMAG.2
0000E6F8  MAXNUMMAG.3        00000000  MIN                0000DA00  MIN.0
0000DA08  MIN.1              0000DA50  MIN.10             0000DA58  MIN.11
0000DA60  MIN.12             0000DA68  MIN.13             0000DA70  MIN.14
0000DA78  MIN.15             0000DA10  MIN.2              0000DA18  MIN.3
0000DA20  MIN.4              0000DA28  MIN.5              0000DA30  MIN.6
0000DA38  MIN.7              0000DA40  MIN.8              0000DA48  MIN.9
00000000  MINNUM             0000E680  MINNUM.0           0000E688  MINNUM.1
0000E690  MINNUM.2           0000E698  MINNUM.3           00000000  MINNUMMAG
0000E6C0  MINNUMMAG.0        0000E6C8  MINNUMMAG.1        0000E6D0  MINNUMMAG.2
0000E6D8  MINNUMMAG.3        0000FE00  MONITR_REG         00000000  MOV
00000000  MUL                0000DC00  MUL.0              0000DC08  MUL.1
0000DC50  MUL.10             0000DC58  MUL.11             0000DC60  MUL.12
0000DC68  MUL.13             0000DC70  MUL.14             0000DC78  MUL.15
0000DC10  MUL.2              0000DC18  MUL.3              0000DC20  MUL.4
0000DC28  MUL.5              0000DC30  MUL.6              0000DC38  MUL.7
0000DC40  MUL.8              0000DC48  MUL.9              00000000  MULTIPLICATION
00000002  N                  00000030  NAN                00000000  NEGATE
0000E640  NEGATE.0           0000E648  NEGATE.1           0000E650  NEGATE.2
0000E658  NEGATE.3           00000022  NEGATIVEINFINITY   00000023  NEGATIVENORMAL
00000024  NEGATIVESUBNORMAL  00000025  NEGATIVEZERO       0000001E  NEVER
```

```
00000000  NEXTDOWN           0000E700  NEXTDOWN.0         0000E708  NEXTDOWN.1
0000E710  NEXTDOWN.2         0000E718  NEXTDOWN.3         0000E720  NEXTDOWN.4
0000E728  NEXTDOWN.5         0000E730  NEXTDOWN.6         0000E738  NEXTDOWN.7
00000000  NEXTUP             0000E740  NEXTUP.0           0000E748  NEXTUP.1
0000E750  NEXTUP.2           0000E758  NEXTUP.3           0000E760  NEXTUP.4
0000E768  NEXTUP.5           0000E770  NEXTUP.6           0000E778  NEXTUP.7
000002A4  NMI_               0000FEF8  NMI_VECT           0000002B  NORMAL
0000001C  NOTZANDV           0000000A  NXACTFLAG          00000019  NXACTSIGNAL
00000000  OR                 0000DF00  OR.0               0000DF08  OR.1
0000DF50  OR.10              0000DF58  OR.11              0000DF60  OR.12
0000DF68  OR.13              0000DF70  OR.14              0000DF78  OR.15
0000DF10  OR.2               0000DF18  OR.3               0000DF20  OR.4
0000DF28  OR.5               0000DF30  OR.6               0000DF38  OR.7
0000DF40  OR.8               0000DF48  OR.9               00000004  OVERFLOW
00000008  OVFLFLAG           00000017  OVFLSIGNAL         000002BA  OVFL_
0000FED8  OVFL_VECT          0000FFA8  PC                 0000FF98  PCC
0000FFA0  PCS                0000FF90  PC_COPY            0000FFF8  PC_REL
00000029  POSITIVEINFINITY   00000028  POSITIVENORMAL     00000027  POSITIVESUBNORMAL
00000026  POSITIVEZERO       00000000  POW                0000E300  POW.0
0000E308  POW.1              0000E350  POW.10             0000E358  POW.11
0000E360  POW.12             0000E368  POW.13             0000E370  POW.14
0000E378  POW.15             0000E310  POW.2              0000E318  POW.3
0000E320  POW.4              0000E328  POW.5              0000E330  POW.6
0000E338  POW.7              0000E340  POW.8              0000E348  POW.9
00000000  POWN               00000000  POWR               000002D9  PROGEND
000000FE  PROG_LEN           00000021  QUIETNAN           00000000  RADIX
0000FE10  RADIX_ADDRS        00000000  RAISEFLAGS         0000FF04  RASFLG
0000E780  REM.0              0000E788  REM.1              0000E7D0  REM.10
0000E7D8  REM.11             0000E7E0  REM.12             0000E7E8  REM.13
0000E7F0  REM.14             0000E7F8  REM.15             0000E790  REM.2
0000E798  REM.3              0000E7A0  REM.4              0000E7A8  REM.5
0000E7B0  REM.6              0000E7B8  REM.7              0000E7C0  REM.8
0000E7C8  REM.9              00000000  REMAINDER          00000000  RESTOREFLAGS
0000003C  RM0                0000003D  RM1                0000003F  RM_ATTRIB
0000FE18  RNDDIR_REG         00000011  RNF_DIVBY0         00000010  RNF_INV
00000014  RNF_NXACT          00000012  RNF_OVFL           00000013  RNF_UNFL
00000000  ROUNDTOINTEGRALEXACT  00000000  ROUNDTOINTEGRALTIESTOAWAY  00000000  ROUNDTOINTEGRALTIESTOEVEN
00000000  ROUNDTOINTEGRALTOWARDNEGATIVE  00000000  ROUNDTOINTEGRALTOWARDPOSITIVE  00000000  ROUNDTOINTEGRALTOWARDZERO
0000FF80  RPT                0000FF01  RSTFLG             0000E900  RTOI.0
0000E908  RTOI.1             0000E950  RTOI.10            0000E958  RTOI.11
0000E960  RTOI.12            0000E968  RTOI.13            0000E970  RTOI.14
0000E978  RTOI.15            0000E910  RTOI.2             0000E918  RTOI.3
0000E920  RTOI.4             0000E928  RTOI.5             0000E930  RTOI.6
0000E938  RTOI.7             0000E940  RTOI.8             0000E948  RTOI.9
00000000  SAVEALLFLAGS       0000FF00  SAVEDFLAGS         0000FE08  SAVEDMODES
00000000  SAVEMODES          00000004  SB                 00000000  SCALEB
0000E880  SCALEB.0           0000E888  SCALEB.1           0000E8D0  SCALEB.10
0000E8D8  SCALEB.11          0000E8E0  SCALEB.12          0000E8E8  SCALEB.13
0000E8F0  SCALEB.14          0000E8F8  SCALEB.15          0000E890  SCALEB.2
0000E898  SCALEB.3           0000E8A0  SCALEB.4           0000E8A8  SCALEB.5
0000E8B0  SCALEB.6           0000E8B8  SCALEB.7           0000E8C0  SCALEB.8
0000E8C8  SCALEB.9           0000FF30  SCHEDCMP           0000FF38  SCHEDULER
00000007  SD                 00000005  SH                 00000000  SHFT
00000000  SHIFT              0000DB00  SHIFT.0            0000DB08  SHIFT.1
0000DB50  SHIFT.10           0000DB58  SHIFT.11           0000DB60  SHIFT.12
0000DB68  SHIFT.13           0000DB70  SHIFT.14           0000DB78  SHIFT.15
0000DB10  SHIFT.2            0000DB18  SHIFT.3            0000DB20  SHIFT.4
0000DB28  SHIFT.5            0000DB30  SHIFT.6            0000DB38  SHIFT.7
0000DB40  SHIFT.8            0000DB48  SHIFT.9            00000031  SIGNALING
```

```
00000020  SIGNALINGNAN       0000002A  SIGNMINUS          00000000  SIND
0000E5E0  SIND.0             0000E5E8  SIND.1             0000E5F0  SIND.2
0000E5F8  SIND.3             0000FFE8  SP                 0000010A  SPIN
0000FFF0  SP_TOS             0000EB80  SQRT.0             0000EB88  SQRT.1
0000EBD0  SQRT.10            0000EBD8  SQRT.11            0000EBE0  SQRT.12
0000EBE8  SQRT.13            0000EBF0  SQRT.14            0000EBF8  SQRT.15
0000EB90  SQRT.2             0000EB98  SQRT.3             0000EBA0  SQRT.4
0000EBA8  SQRT.5             0000EBB0  SQRT.6             0000EBB8  SQRT.7
0000EBC0  SQRT.8             0000EBC8  SQRT.9             00000000  SQUAREROOT
0000010B  START              0000FF88  STATUS             00000000  SUB
0000DD00  SUB.0              0000DD08  SUB.1              0000DD50  SUB.10
0000DD58  SUB.11             0000DD60  SUB.12             0000DD68  SUB.13
0000DD70  SUB.14             0000DD78  SUB.15             0000DD10  SUB.2
0000DD18  SUB.3              0000DD20  SUB.4              0000DD28  SUB.5
0000DD30  SUB.6              0000DD38  SUB.7              0000DD40  SUB.8
0000DD48  SUB.9              00000000  SUBB               0000DC80  SUBB.0
0000DC88  SUBB.1             0000DCD0  SUBB.10            0000DCD8  SUBB.11
0000DCE0  SUBB.12            0000DCE8  SUBB.13            0000DCF0  SUBB.14
0000DCF8  SUBB.15            0000DC90  SUBB.2             0000DC98  SUBB.3
0000DCA0  SUBB.4             0000DCA8  SUBB.5             0000DCB0  SUBB.6
0000DCB8  SUBB.7             0000DCC0  SUBB.8             0000DCC8  SUBB.9
0000002E  SUBNORMAL          00000038  SUBS_DIVBY0        00000037  SUBS_INV
0000003B  SUBS_NXACT         00000039  SUBS_OVFL          0000003A  SUBS_UNFL
00000000  SUBTRACTION        00000006  SW                 00000000  TAND
0000E5A0  TAND.0             0000E5A8  TAND.1             0000E5B0  TAND.2
0000E5B8  TAND.3             00000000  TESTFLAGS          00000000  TESTSAVEDFLAGS
0000FF68  TIMER              0000FF07  TORD               0000FF06  TORDM
00000033  TOTLORDER          00000034  TOTLORDERMAG       0000FF03  TSTFLG
0000FF02  TSTSFLG            00000000  UB                 00000003  UD
00000001  UH                 00000008  UNDERFLOW          00000009  UNFLFLAG
00000018  UNFLSIGNAL         000002C3  UNFL_              0000FED0  UNFL_VECT
00000002  UW                 00000003  V                  00000008  WORK_1
00000010  WORK_2             00000018  WORK_3             00000000  XCU.0
00000001  XCU.1              0000000A  XCU.10             0000000B  XCU.11
0000000C  XCU.12             0000000D  XCU.13             0000000E  XCU.14
0000000F  XCU.15             00000002  XCU.2              00000003  XCU.3
00000004  XCU.4              00000005  XCU.5              00000006  XCU.6
00000007  XCU.7              00000008  XCU.8              00000009  XCU.9
00000001  XCU0               00000002  XCU1               00000400  XCU10
00000800  XCU11              00001000  XCU12              00002000  XCU13
00004000  XCU14              00008000  XCU15              00000004  XCU2
00000008  XCU3               00000010  XCU4               00000020  XCU5
00000040  XCU6               00000080  XCU7               00000100  XCU8
00000200  XCU9               0000FDF8  XCU_CNTRL_REG      0000FDE0  XCU_MON_REQUEST
0000FDD8  XCU_PUSH_ALL       0000FDF0  XCU_STATUS_REG     00000007  XFD
00000005  XFH                00000006  XFS                00000000  XOR
0000DE80  XOR.0              0000DE88  XOR.1              0000DED0  XOR.10
0000DED8  XOR.11             0000DEE0  XOR.12             0000DEE8  XOR.13
0000DEF0  XOR.14             0000DEF8  XOR.15             0000DE90  XOR.2
0000DE98  XOR.3              0000DEA0  XOR.4              0000DEA8  XOR.5
0000DEB0  XOR.6              0000DEB8  XOR.7              0000DEC0  XOR.8
0000DEC8  XOR.9              00000000  Z                  0000002D  ZERO
0000001D  ZORV               00000001  _2P0               00000000  _5P0
```