Clive's KNI Rough Guide

Intel has now released official SSE documentation

Intel Architecture Optimization Reference Manual

Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference Manual

---

Written by Clive Turvey clive@tbcnet.com
Back to Clive's Katmai Page

ADDPS, ADDSS, ANDNPS, ANDPS, CMPEQPS, CMPEQSS, CMPLEPS, CMPLESS, CMPLTPS, CMPLTSS, CMPNEQPS, CMPNEQSS, CMPNLEPS, CMPNLESS, CMPNLTPS, CMPNLTSS, CMPORDPS, CMPORDSS, CMPUNORDPS, CMPUNORDSS, COMISS, CVTPI2PS, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2PI, CVTTSS2SI, DIVPS, DIVSS, FXRSTOR, FXSAVE, LDMXCSR, MASKMOVQ, MAXPS, MAXSS, MINPS, MINSS, MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVNTPS, MOVNTQ, MOVSS, MOVUPS, MULPS, MULSS, ORPS, PAVGB, PAVGW, PEXTRW, PINSRW, PMAXSW, PMAXUB, PMINSW, PMINUB, PMOVMSKB, PMULHUW, PREFETCHNTA, PREFETCHT0, PREFETCHT1, PREFETCHT2, PSADBW, PSHUFW, RCPPS, RCPSS, RSQRTPS, RSQRTSS, SFENCE, SHUFPS, SQRTPS, SQRTSS, STMXCSR, SUBPS, SUBSS, UCOMISS, UNPCKHPS, UNPCKLPS & XORPS.

```
Please note, this is a work-in-progress (ie BETA).

Timings are of approximate throughput cycles using average from TSC, the
latency and ranges are indicated where known.
```

```
ADDPS           Add Parallel Scalars

Opcode          Cycles  Instruction
0F 58           2 (3)   ADDPS xmm reg,xmm reg/mem128

ADDPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] + op2[0]
        op1[1] = op1[1] + op2[1]
        op1[2] = op1[2] + op2[2]
        op1[3] = op1[3] + op2[3]
```

```
ADDSS           Add Single Scalar

Opcode          Cycles  Instruction
F3 0F 58        1 (3)   ADDSS xmm reg,xmm reg/mem32

ADDPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value
```

```
        op1[0] = op1[0] + op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
ANDNPS          And Not Parallel Scalars (bitwise)

Opcode          Cycles  Instruction
0F 55           2       ANDNPS xmm reg,xmm reg/mem128

ANDNPS op1, op2

op1 contains 1 128-bit value
op2 contains 1 128-bit value

        op1 = !op1 & op2
```

```
ANDPS           And Parallel Scalars (bitwise)

Opcode          Cycles  Instruction
0F 54           2       ANDPS xmm reg,xmm reg/mem128

ANDPS op1, op2

op1 contains 1 128-bit value
op2 contains 1 128-bit value

        op1 = op1 & op2
```

```
CMPEQPS         Compare Equal Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 00     2 (3)   CMPEQPS xmm reg,xmm reg/mem128

CMPEQPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] == op2[0]
        op1[1] = op1[1] == op2[1]
        op1[2] = op1[2] == op2[2]
        op1[3] = op1[3] == op2[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPEQSS         Compare Equal Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 00  1 (3)   CMPEQSS xmm reg,xmm reg/mem32

CMPEQSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] == op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPLEPS         Compare Less than or Equal Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 02     2 (3)   CMPLEPS xmm reg,xmm reg/mem128

CMPLEPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] <= op2[0]
        op1[1] = op1[1] <= op2[1]
        op1[2] = op1[2] <= op2[2]
        op1[3] = op1[3] <= op2[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPLESS         Compare Less than or Equal Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 02  1 (3)   CMPLESS xmm reg,xmm reg/mem32

CMPLESS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] <= op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPLTPS         Compare Less Than Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 01     2 (3)   CMPLTPS xmm reg,xmm reg/mem128

CMPLTPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] < op2[0]
        op1[1] = op1[1] < op2[1]
        op1[2] = op1[2] < op2[2]
        op1[3] = op1[3] < op2[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPLTSS         Compare Less Than Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 01  1 (3)   CMPLTSS xmm reg,xmm reg/mem32

CMPLTSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value
```

```
        op1[0] = op1[0] < op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPNEQPS        Compare Not Equal Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 04     2 (3)   CMPNEQPS xmm reg,xmm reg/mem128

CMPNEQPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] != op2[0]
        op1[1] = op1[1] != op2[1]
        op1[2] = op1[2] != op2[2]
        op1[3] = op1[3] != op2[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPNEQSS        Compare Not Equal Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 04  1 (3)   CMPNEQSS xmm reg,xmm reg/mem32

CMPNEQSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] != op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPNLEPS        Compare Not Less than or Equal Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 06     2 (3)   CMPNLEPS xmm reg,xmm reg/mem128

CMPNLEPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] > op2[0]
        op1[1] = op1[1] > op2[1]
        op1[2] = op1[2] > op2[2]
        op1[3] = op1[3] > op2[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPNLESS        Compare Not Less than or Equal Single Scalar
```

```
Opcode          Cycles  Instruction
F3 0F C2 .. 06  1 (3)   CMPNLESS xmm reg,xmm reg/mem32

CMPNLESS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] > op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPNLTPS        Compare Not Less Than Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 05     2 (3)   CMPNLTPS xmm reg,xmm reg/mem128

CMPNLTPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] >= op2[0]
        op1[1] = op1[1] >= op2[1]
        op1[2] = op1[2] >= op2[2]
        op1[3] = op1[3] >= op2[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPNLTSS                Compare Not Less Than Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 01  1 (3)   CMPNLTSS xmm reg,xmm reg/mem32

CMPNLTSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] >= op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPORDPS        Compare Ordered Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 07     2 (3)   CMPORDPS xmm reg,xmm reg/mem128

CMPORDPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = (op1[0] != NaN) && (op2[0] != NaN)
```

```
        op1[1] = (op1[1] != NaN) && (op2[1] != NaN)
        op1[2] = (op1[2] != NaN) && (op2[2] != NaN)
        op1[3] = (op1[3] != NaN) && (op2[3] != NaN)

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPORDSS        Compare Ordered Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 07  1 (3)   CMPORDSS xmm reg,xmm reg/mem32

CMPORDSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = (op1[0] != NaN) && (op2 != NaN)
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPUNORDPS      Compare Unordered Parallel Scalars

Opcode          Cycles  Instruction
0F C2 .. 03     2 (3)   CMPUNORDPS xmm reg,xmm reg/mem128

CMPUNORDPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = (op1[0] == NaN) || (op2[0] == NaN)
        op1[1] = (op1[1] == NaN) || (op2[1] == NaN)
        op1[2] = (op1[2] == NaN) || (op2[2] == NaN)
        op1[3] = (op1[3] == NaN) || (op2[3] == NaN)

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
CMPUNORDSS      Compare Unordered Single Scalar

Opcode          Cycles  Instruction
F3 0F C2 .. 03  1 (3)   CMPUNORDSS xmm reg,xmm reg/mem32

CMPUNORDSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = (op1[0] == NaN) || (op2 == NaN)
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        TRUE  = 0xFFFFFFFF
        FALSE = 0x00000000
```

```
COMISS          Compare Integer Single Scalar

Opcode          Cycles  Instruction
```

```
0F 2F                   COMISS xmm reg,xmm reg/mem32

COMISS op1, op2
```

```
CVTPI2PS        Convert Parallel Integer to Parallel Scalars

Opcode          Cycles  Instruction
0F 2A                   CVTPI2PS xmm reg,mm reg/mem64

CVTPI2PS op1, op2

op1 contains 2 single precision 32-bit floating point values
op2 contains 2 32-bit integer values

        op1[0] = (float)op2[0]
        op1[1] = (float)op2[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
CVTPS2PI        Convert Parallel Scalars to Parallel Integers

Opcode          Cycles  Instruction
0F 2D                   CVTPS2PI mm reg,xmm reg/mem128

CVTPS2PI op1, op2

op1 contains 2 32-bit integer values
op2 contains 2 single precision 32-bit floating point values

        op1[0] = (long)op2[0]
        op1[1] = (long)op2[1]
```

```
CVTSI2SS        Convert Parallel Integers to Parallel Scalars

Opcode          Cycles  Instruction
F3 0F 2A                CVTSI2SS xmm reg,reg32/mem32

CVTSI2SS op1, op2

op1 contains 1 single precision 32-bit floating point value
op2 contains 1 32-bit integer value

        op1[0] = (float)op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
CVTSS2SI        Convert Single Scalar to Single Integer

Opcode          Cycles  Instruction
F3 0F 2D                CVTSS2SI reg32,xmm reg/mem128

CVTPS2PI op1, op2

op1 contains 1 32-bit integer value
op2 contains 1 single precision 32-bit floating point value

        op1 = (long)op2[0]
```

```
CVTTPS2PI       Convert Parallel Scalars to Parallel Integers

Opcode          Cycles  Instruction
0F 2C                   CVTTPS2PI mm reg,xmm reg/mem128
```

```
CVTTPS2PI op1, op2

op1 contains 2 32-bit integer values
op2 contains 2 single precision 32-bit floating point values

        op1[0] = (long)op2[0]
        op1[1] = (long)op2[1]
```

```
CVTTSS2SI       Convert Single Scalar to Single Integer

Opcode          Cycles  Instruction
F3 0F 2C                CVTTSS2SI reg32,xmm reg/mem128

CVTTSS2SI op1, op2

op1 contains 1 32-bit integer value
op2 contains 1 single precision 32-bit floating point value

        op1 = (long)op2[0]
```

```
DIVPS           Divide Parallel Scalars

Opcode          Cycles  Instruction
0F 5E           15-115  DIVPS xmm reg,xmm reg/mem128

DIVPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] / op2[0]
        op1[1] = op1[1] / op2[1]
        op1[2] = op1[2] / op2[2]
        op1[3] = op1[3] / op2[3]
```

```
DIVSS           Divide Single Scalar

Opcode          Cycles  Instruction
F3 0F 5E        7-98    DIVSS xmm reg,xmm reg/mem32

DIVSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] / op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
FXRSTOR         Floating Point Extended Restore

Opcode          Cycles  Instruction
0F AE xx001xxx          FXRSTOR mem

FXRSTOR op1

op1 contains a 512 byte register context, paragraph aligned
```

```
FXSAVE          Floating Point Extended Save

Opcode          Cycles  Instruction
0F AE xx000xxx          FXSAVE mem
```

```
FXSAVE op1

op1 contains a 512 byte register context, paragraph aligned
```

```
LDMXCSR         Load Multimedia Extended Control Status Register

Opcode          Cycles  Instruction
0F AE xx010xxx          LDMXCSR mem32

LDMXCSR op1

op1 contains 1 32-bit register

        MXCSR = op1
```

```
MASKMOVQ

Opcode          Cycles  Instruction
0F F7                   MASKMOVQ mm reg,mm reg

MASKMOVQ op1, op2
```

```
MAXPS           Maximum Parallel Scalars

Opcode          Cycles  Instruction
0F 5F           2 (3)   MAXPS xmm reg,xmm reg/mem128

MAXPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = max(op1[0], op2[0])
        op1[1] = max(op1[1], op2[1])
        op1[2] = max(op1[2], op2[2])
        op1[3] = max(op1[3], op2[3])
```

```
MAXSS           Maximum Single Scalar

Opcode          Cycles  Instruction
F3 0F 5F        1 (3)   MAXSS xmm reg,xmm reg/mem32

MAXSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = max(op1[0], op2)
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
MINPS           Minimum Parallel Scalars

Opcode          Cycles  Instruction
0F 5D           2 (3)   MINPS xmm reg,xmm reg/mem128

MINPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = min(op1[0], op2[0])
        op1[1] = min(op1[1], op2[1])
```

```
        op1[2] = min(op1[2], op2[2])
        op1[3] = min(op1[3], op2[3])
```

```
MINSS           Minimum Single Scalar

Opcode          Cycles  Instruction
F3 0F 5D        1 (3)   MINSS xmm reg,xmm reg/mem32

MINSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = min(op1[0], op2)
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
MOVAPS          Aligned Move Parallel Scalars

Opcode          Cycles  Instruction
0F 28                   MOVAPS xmm reg,xmm reg/mem128
0F 29                   MOVAPS mem128,xmm reg

MOVAPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op2[0]
        op1[1] = op2[1]
        op1[2] = op2[2]
        op1[3] = op2[3]

        * Addresses must be paragraph aligned
```

```
MOVHPS          Move High Pair Parallel Scalars

Opcode          Cycles  Instruction
0F 16                   MOVHPS xmm reg,mem64
0F 17                   MOVHPS mem64,xmm reg

MOVHPS op1, op2

op1 contains 2 single precision 32-bit floating point values
op2 contains 2 single precision 32-bit floating point values

        op1[2] = op2[0] (xmm reg,mem64)
        op1[3] = op2[1]

        op1[0] = op2[2] (mem64,xmm reg)
        op1[1] = op2[3]
```

```
MOVHLPS         Move High to Low Pair Parallel Scalars

Opcode          Cycles  Instruction
0F 12           1       MOVHLPS xmm reg,xmm reg

MOVHLPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op2[2]
```

```
        op1[1] = op2[3]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
MOVLPS          Move Low Pair Parallel Scalars

Opcode          Cycles  Instruction
0F 12                   MOVLPS xmm reg,mem64
0F 13                   MOVLPS mem64,xmm reg

MOVLPS op1, op2

op1 contains 2 single precision 32-bit floating point values
op2 contains 2 single precision 32-bit floating point values

        op1[0] = op2[0]
        op1[1] = op2[1]
```

```
MOVLHPS         Move Low to High Pair Parallel Scalars

Opcode          Cycles  Instruction
0F 16           1       MOVLHPS xmm reg,xmm reg

MOVLHPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0]
        op1[1] = op1[1]
        op1[2] = op2[0]
        op1[3] = op2[1]
```

```
MOVMSKPS

Opcode          Cycles  Instruction
0F 50                   MOVMSKPS reg32,xmm reg

MOVMSKPS op1, op2
```

```
MOVNTPS         Uncached Move Parallel Scalars

Opcode          Cycles  Instruction
0F 2B                   MOVNTPS mem128,xmm reg

MOVNTPS op1, op2

op1 contains 1 128-bit value
op2 contains 1 128-bit value

        op1 = op2
```

```
MOVNTQ          Uncached Move Quad Word

Opcode          Cycles  Instruction
0F E7                   MOVNTQ mem64,mm reg

MOVNTQ op1, op2

op1 contains 1 64-bit value
op2 contains 1 64-bit value

        op1 = op2
```

```
MOVSS           Move Single Scalar

Opcode          Cycles  Instruction
F3 0F 10                MOVSS xmm reg,xmm reg/mem32
F3 0F 11                MOVSS mem32,xmm reg

MOVSS op1, op2

op1 contains 1 single precision 32-bit floating point value
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op2[0]
```

```
MOVUPS          Unaligned Move Parallel Scalars

Opcode          Cycles  Instruction
0F 10                   MOVUPS xmm reg,xmm reg/mem128
0F 11                   MOVUPS mem128,xmm reg

MOVUPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op2[0]
        op1[1] = op2[1]
        op1[2] = op2[2]
        op1[3] = op2[3]
```

```
MULPS           Multiply Parallel Scalars

Opcode          Cycles  Instruction
0F 59           2 (4)   MULPS xmm reg,xmm reg/mem128

MULPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] * op2[0]
        op1[1] = op1[1] * op2[1]
        op1[2] = op1[2] * op2[2]
        op1[3] = op1[3] * op2[3]
```

```
MULSS           Multiply Single Scalar

Opcode          Cycles  Instruction
F3 0F 59        1 (4)   MULSS xmm reg,xmm reg/mem32

MULSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] * op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
ORPS            Or Parallel Scalars (bitwise)

Opcode          Cycles  Instruction
0F 56           2       ORPS xmm reg,xmm reg/mem128
```

```
ORPS op1, op2

op1 contains 1 128-bit value
op2 contains 1 128-bit value

        op1 = op1 | op2
```

```
PAVGB           Parallel Integer Average Byte

Opcode          Cycles  Instruction
0F E0                   PAVGB mm reg,mm reg/mem64

PAVGB op1, op2

op1 contains 8 8-bit integer values
op2 contains 8 8-bit integer values

        op1[0] = (op1[0] + op2[0]) / 2
        op1[1] = (op1[1] + op2[1]) / 2
        op1[2] = (op1[2] + op2[2]) / 2
        op1[3] = (op1[3] + op2[3]) / 2
        op1[4] = (op1[4] + op2[4]) / 2
        op1[5] = (op1[5] + op2[5]) / 2
        op1[6] = (op1[6] + op2[6]) / 2
        op1[7] = (op1[7] + op2[7]) / 2
```

```
PAVGW           Parallel Integer Average Word

Opcode          Cycles  Instruction
0F E3                   PAVGW mm reg,mm reg/mem64

PAVGW op1, op2

op1 contains 4 16-bit integer values
op2 contains 4 16-bit integer values

        op1[0] = (op1[0] + op2[0]) / 2
        op1[1] = (op1[1] + op2[1]) / 2
        op1[2] = (op1[2] + op2[2]) / 2
        op1[3] = (op1[3] + op2[3]) / 2
```

```
PEXTRW

Opcode          Cycles  Instruction
0F C5                   PEXTRW reg32,mm reg,imm8

PEXTRW op1, op2, op3
```

```
PINSRW

Opcode          Cycles  Instruction
0F C4                   PINSRW mm reg,reg32/mem32,imm8

PINSRW op1, op2, op3
```

```
PMAXSW          Parallel Integer Maximum Signed Word

Opcode          Cycles  Instruction
0F EE                   PMAXSW mm reg,mm reg/mem64

PMAXSW op1, op2

op1 contains 4 16-bit signed integer values
op2 contains 4 16-bit signed integer values
```

```
        op1[0] = max(op1[0], op2[0])
        op1[1] = max(op1[1], op2[1])
        op1[2] = max(op1[2], op2[2])
        op1[3] = max(op1[3], op2[3])
```

```
PMAXUB          Parallel Integer Maximum Unsigned Byte

Opcode          Cycles  Instruction
0F DE                   PMAXUB mm reg,mm reg/mem64

PMAXUB op1, op2

op1 contains 8 8-bit unsigned integer values
op2 contains 8 8-bit unsigned integer values

        op1[0] = max(op1[0], op2[0])
        op1[1] = max(op1[1], op2[1])
        op1[2] = max(op1[2], op2[2])
        op1[3] = max(op1[3], op2[3])
        op1[4] = max(op1[4], op2[4])
        op1[5] = max(op1[5], op2[5])
        op1[6] = max(op1[6], op2[6])
        op1[7] = max(op1[7], op2[7])
```

```
PMINSW          Parallel Integer Minimum Signed Word

Opcode          Cycles  Instruction
0F EA                   PMINSW mm reg,mm reg/mem64

PMINSW op1, op2

op1 contains 4 16-bit signed integer values
op2 contains 4 16-bit signed integer values

        op1[0] = min(op1[0], op2[0])
        op1[1] = min(op1[1], op2[1])
        op1[2] = min(op1[2], op2[2])
        op1[3] = min(op1[3], op2[3])
```

```
PMINUB          Parallel Integer Minimum Unsigned Byte

Opcode          Cycles  Instruction
0F DA                   PMINUB mm reg,mm reg/mem64

PMINUB op1, op2

op1 contains 8 8-bit unsigned integer values
op2 contains 8 8-bit unsigned integer values

        op1[0] = min(op1[0], op2[0])
        op1[1] = min(op1[1], op2[1])
        op1[2] = min(op1[2], op2[2])
        op1[3] = min(op1[3], op2[3])
        op1[4] = min(op1[4], op2[4])
        op1[5] = min(op1[5], op2[5])
        op1[6] = min(op1[6], op2[6])
        op1[7] = min(op1[7], op2[7])
```

```
PMOVMSKB

Opcode          Cycles  Instruction
0F D7                   PMOVMSKB reg32,mm reg
```

```
PMOVMSKB op1, op2
```

```
PMULHUW         Multiply unsigned word store high

Opcode          Cycles  Instruction
0F E4                   PMULHUW mm reg,mm reg/mem64

PMULHUW op1, op2

op1 contains 4 16-bit unsigned integer values
op2 contains 4 16-bit unsigned integer values

        op1[0] = (op1[0] * op2[0]) >> 16
        op1[1] = (op1[1] * op2[1]) >> 16
        op1[2] = (op1[2] * op2[2]) >> 16
        op1[3] = (op1[3] * op2[3]) >> 16
```

```
PREFETCHNTA     Prefetch Non-caching Aligned ?

Opcode          Cycles  Instruction
0F 18 xx000xxx          PREFETCHNTA mem8

PREFETCHNTA op1
```

```
PREFETCHT0      Prefetch Task 0 ?

Opcode          Cycles  Instruction
0F 18 xx001xxx          PREFETCHT0 mem8

PREFETCHT0 op1
```

```
PREFETCHT1      Prefetch Task 1 ?

Opcode          Cycles  Instruction
0F 18 xx010xxx          PREFETCHT1 mem8

PREFETCHT1 op1
```

```
PREFETCHT2      Prefetch Task 2 ?

Opcode          Cycles  Instruction
0F 18 xx011xxx          PREFETCHT2 mem8

PREFETCHT2 op1
```

```
PSADBW

Opcode          Cycles  Instruction
0F F6                   PSADBW mm reg,mm reg/mem64

PSADBW op1, op2
```

```
PSHUFW          Shuffle Parallel Words

Opcode          Cycles  Instruction
0F 70           1 (1)   PSHUFW mm reg,mm reg/mem64,imm8

PSHUFW op1, op3, op3

op1 contains 4 16-bit integer values
op2 contains 4 16-bit integer values
op3 contains a bit map dd:cc:bb:aa (MSB to LSB)

        op1[0] = op2[aa]
```

```
        op1[1] = op2[bb]
        op1[2] = op2[cc]
        op1[3] = op2[dd]
```

```
RCPPS           Reciprocal Parallel Scalars

Opcode          Cycles  Instruction
0F 53           2 (2)   RCPPS xmm reg,xmm reg/mem128

RCPPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = 1 / op2[0]
        op1[1] = 1 / op2[1]
        op1[2] = 1 / op2[2]
        op1[3] = 1 / op2[3]

        * The results have 12-bit accuracy
```

```
RCPSS           Reciprocal Single Scalar

Opcode          Cycles  Instruction
F3 0F 53        1 (1)   RCPSS xmm reg,xmm reg/mem32

RCPSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = 1 / op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        * The results have 12-bit accuracy
```

```
RSQRTPS         Reciprocal Square Root Parallel Scalars

Opcode          Cycles  Instruction
0F 52           2 (2)   RSQRTPS xmm reg,xmm reg/mem128

RSQRTPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = 1 / sqrt(op2[0])
        op1[1] = 1 / sqrt(op2[1])
        op1[2] = 1 / sqrt(op2[2])
        op1[3] = 1 / sqrt(op2[3])

        * The results have 12-bit accuracy
```

```
RSQRTSS         Reciprocal Square Root Single Scalar

Opcode          Cycles  Instruction
F3 0F 52        1 (1)   RSQRTSS xmm reg,xmm reg/mem32

RSQRTSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value
```

```
        op1[0] = 1 / sqrt(op2)
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]

        * The results have 12-bit accuracy
```

```
SFENCE          Stream Fence

Opcode          Cycles  Instruction
0F AE FF                SFENCE

SFENCE

Provides a demarkation in write combining buffers to force
current states to be committed. In other words writes to the
same location cannot combine to one if there is a fence placed
between them.

        * Presumed function from AGP definition of fencing
```

```
SHUFPS          Shuffle Parallel Scalars

Opcode          Cycles  Instruction
0F C6           3       SHUFPS xmm reg, xmm reg/mem128, imm8

SHUFPS op1, op3, op3

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values
op3 contains a bit map dd:cc:bb:aa (MSB to LSB)

        op1[0] = op1[aa]
        op1[1] = op1[bb]
        op1[2] = op2[cc]
        op1[3] = op2[dd]
```

```
SQRTPS          Square Root Parallel Scalars

Opcode          Cycles  Instruction
0F 51           16-134  SQRTPS xmm reg,xmm reg/mem128

SQRTPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = sqrt(op2[0])
        op1[1] = sqrt(op2[1])
        op1[2] = sqrt(op2[2])
        op1[3] = sqrt(op2[3])
```

```
SQRTSS          Square Root Single Scalar

Opcode          Cycles  Instruction
F3 0F 51        8-105   SQRTSS xmm reg,xmm reg/mem32

SQRTSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = sqrt(op2)
```

```
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
STMXCSR         Store Multimedia Extended Control Status Register

Opcode          Cycles  Instruction
0F AE xx011xxx          STMXCSR mem32

STMXCSR op1

op1 contains 1 32-bit register

        op1 = MXCSR
```

```
SUBPS           Subtract Parallel Scalars

Opcode          Cycles  Instruction
0F 5C           2 (3)   SUBPS xmm reg,xmm reg/mem128

SUBPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0] - op2[0]
        op1[1] = op1[1] - op2[1]
        op1[2] = op1[2] - op2[2]
        op1[3] = op1[3] - op2[3]
```

```
SUBSS           Subtract Single Scalar

Opcode          Cycles  Instruction
F3 0F 5C        1 (3)   SUBSS xmm reg,xmm reg/mem32

SUBSS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 1 single precision 32-bit floating point value

        op1[0] = op1[0] - op2
        op1[1] = op1[1]
        op1[2] = op1[2]
        op1[3] = op1[3]
```

```
UCOMISS

Opcode          Cycles  Instruction
0F 2E                   UCOMISS xmm reg,xmm reg/mem32

UCOMISS op1, op2
```

```
UNPCKHPS        Unpack High Parallel Scalars

Opcode          Cycles  Instruction
0F 15           2       UNPCKHPS xmm reg,xmm reg/mem128

UNPCKHPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[2]
        op1[1] = op2[2]
```

```
        op1[2] = op1[3]
        op1[3] = op2[3]
```

```
UNPCKLPS        Unpack Low Parallel Scalars

Opcode          Cycles  Instruction
0F 14           2       UNPCKLPS xmm reg,xmm reg/mem128

UNPCKLPS op1, op2

op1 contains 4 single precision 32-bit floating point values
op2 contains 4 single precision 32-bit floating point values

        op1[0] = op1[0]
        op1[1] = op2[0]
        op1[2] = op1[1]
        op1[3] = op2[1]
```

```
XORPS           Exclusive-Or Parallel Scalars (bitwise)

Opcode          Cycles  Instruction
0F 57           2       XORPS xmm reg,xmm reg/mem128

XORPS op1, op2

op1 contains 1 128-bit value
op2 contains 1 128-bit value

        op1 = op1 ^ op2
```
