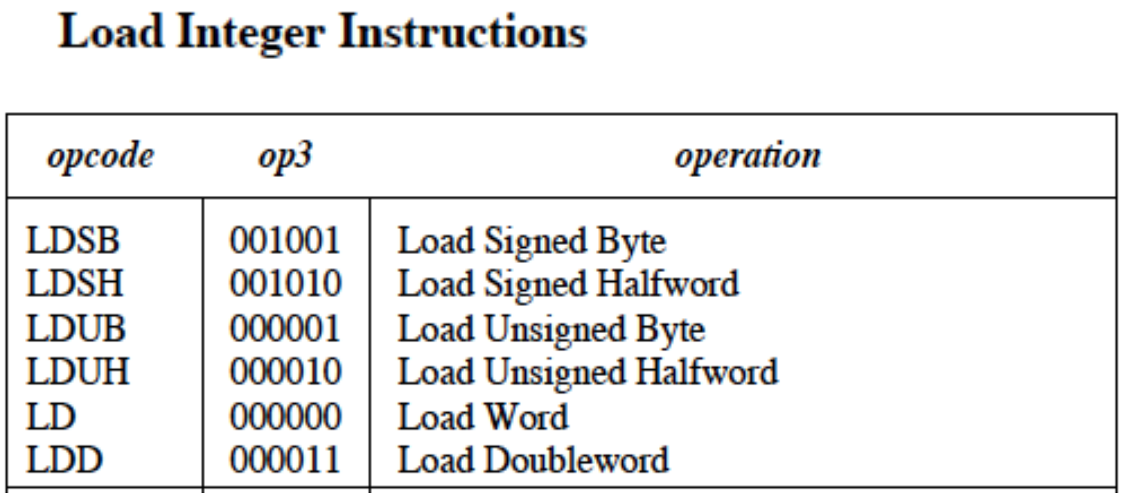
Sparc V8 Instructions

Instructions fall into following categories :

- Load/store
- Arithmetic/logical/shift
- Control transfer
- Read/write control register
- Floating-point/Coprocessor operate

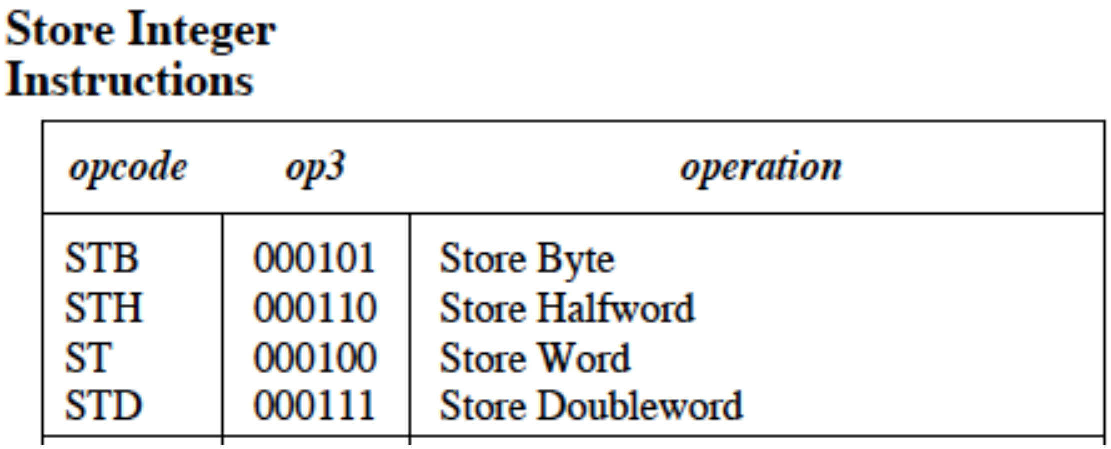
Sparc V8 Instructions



Load Integer Instructions

opcode	*op3*	*operation*
LDSB	001001	Load Signed Byte
LDSH	001010	Load Signed Halfword
LDUB	000001	Load Unsigned Byte
LDUH	000010	Load Unsigned Halfword
LD	000000	Load Word
LDD	000011	Load Doubleword

ld [%sp – 4], %o2



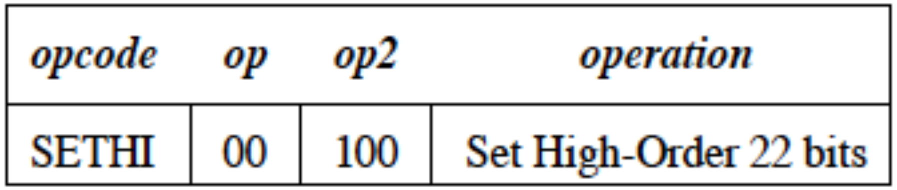
Store Integer Instructions

opcode	*op3*	*operation*
STB	000101	Store Byte
STH	000110	Store Halfword
ST	000100	Store Word
STD	000111	Store Doubleword

sh %o4, [%fp + 8]

Sparc V8 Instructions

SETHI Instruction



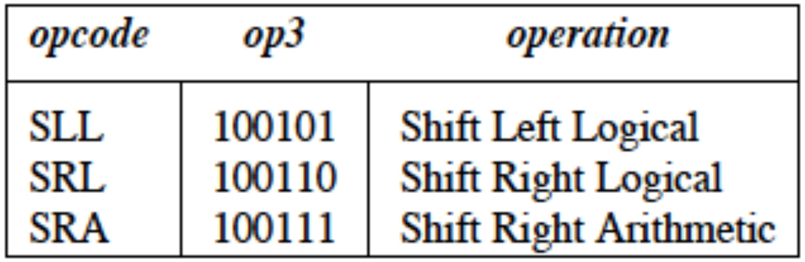
opcode	*op*	*op2*	*operation*
SETHI	00	100	Set High-Order 22 bits

```
sethi   0x4A3, %g2
sethi   %hi(Z), %o3
```

SETHI zeroes the least significant 10 bits of "r[*rd*]", and replaces its high-order 22 bits with the value from its *imm22* field.

A SETHI instruction with *rd* = 0 and *imm22* = 0 is defined to be a NOP

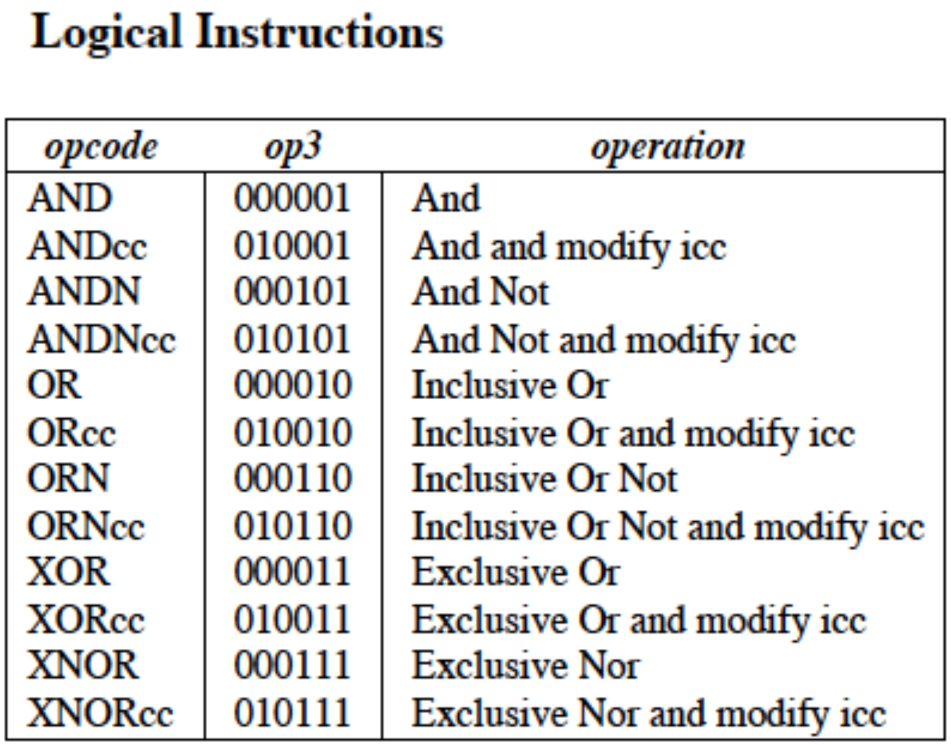
Shift Instructions



opcode	*op3*	*operation*
SLL	100101	Shift Left Logical
SRL	100110	Shift Right Logical
SRA	100111	Shift Right Arithmetic

```
sll     %g2, 4, %g2
sra    %o4,%g4,%g2
```

Sparc V8 Instructions



Logical Instructions

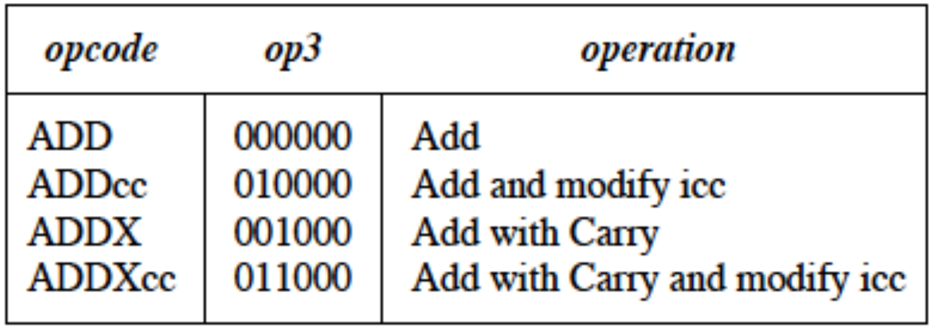
opcode	*op3*	*operation*
AND	000001	And
ANDcc	010001	And and modify icc
ANDN	000101	And Not
ANDNcc	010101	And Not and modify icc
OR	000010	Inclusive Or
ORcc	010010	Inclusive Or and modify icc
ORN	000110	Inclusive Or Not
ORNcc	010110	Inclusive Or Not and modify icc
XOR	000011	Exclusive Or
XORcc	010011	Exclusive Or and modify icc
XNOR	000111	Exclusive Nor
XNORcc	010111	Exclusive Nor and modify icc

```
and      %g2, 4, %g2
xorcc    %o4,%g4,%g2
```

13-bit immediate operands are sign extended to 32 bits

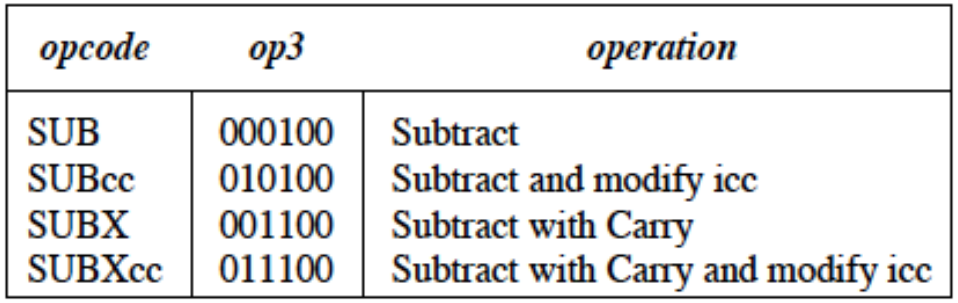
Sparc V8 Instructions

Add and subtract instructions



opcode	*op3*	*operation*
ADD	000000	Add
ADDcc	010000	Add and modify icc
ADDX	001000	Add with Carry
ADDXcc	011000	Add with Carry and modify icc

```
addcc    %g2, 4, %g2
addxcc   %o4,%g4,%g2
```

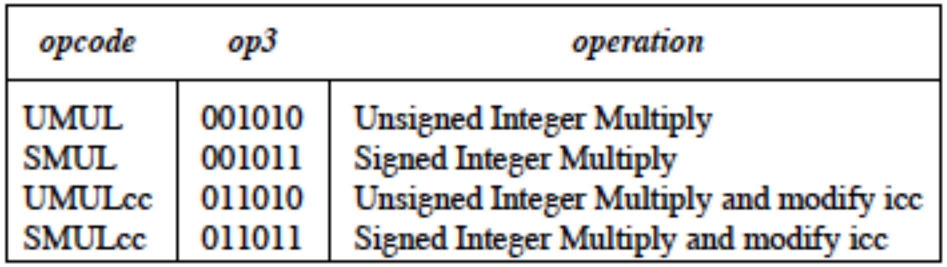


opcode	*op3*	*operation*
SUB	000100	Subtract
SUBcc	010100	Subtract and modify icc
SUBX	001100	Subtract with Carry
SUBXcc	011100	Subtract with Carry and modify icc

```
sub     %g2, 4, %g2
subx    %o4,%g4,%g2
```

Sparc V8 Instructions

Multiply Instructions

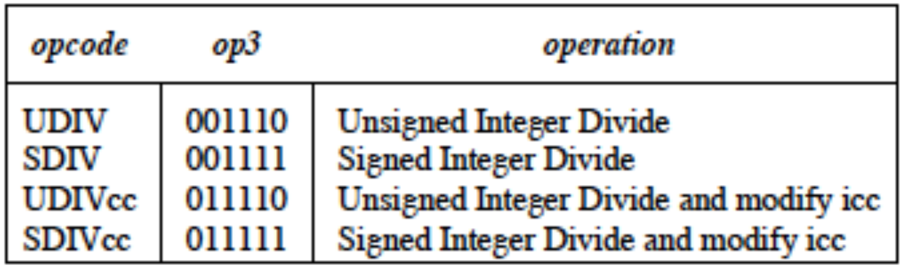


opcode	*op3*	*operation*
UMUL	001010	Unsigned Integer Multiply
SMUL	001011	Signed Integer Multiply
UMULcc	011010	Unsigned Integer Multiply and modify icc
SMULcc	011011	Signed Integer Multiply and modify icc

```
smul     %g2, 4, %g2
umul    %o4,%g4,%g2
```

Result=Reg * sign-extended 13-bit immediate
Or = reg1 * reg2
Low part of product goes to rd
High part of product goes to Y register

Divide Instructions



opcode	*op3*	*operation*
UDIV	001110	Unsigned Integer Divide
SDIV	001111	Signed Integer Divide
UDIVcc	011110	Unsigned Integer Divide and modify icc
SDIVcc	011111	Signed Integer Divide and modify icc

```
udiv          %g2, 4, %g2
sdiv          %o4,%g4,%g2
```

Y:reg1 ÷ sign-extended 13-bit immediate
Or Y:reg1 ÷ reg2
32-bit quotient goes into rd
Remainder is discarded

wry instruction writes Y register
rdy instruction reads Y register

Sparc V8 Instructions

FPU instructions employ separate float registers

Floating point arithmetic	(fadds, fsubs, fmuls, fmuld, fdivs, fdivd, fsqt)
Conversion between float and integer	(fitos, fitod, fstoi, fdtoi)
Floating point comparison	(fcmps, fcmpd)
Load & store floating point operands in memory	(ldf, lddf, stf, stdf)
Copy between float registers	(fmovs, fnegs, fabss)