INSTRUCCIONES Y FORMATOS ARQUITECTURA SPARC V8

Direcciones de registros

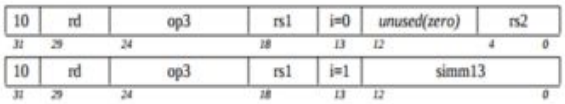
Window Addressing

Windowed Register Address	*r* Register Address
in[0] – in[7]	r[24] – r[31]
local[0] – local[7]	r[16] – r[23]
out[0] – out[7]	r[8] – r[15]
global[0] – global[7]	r[0] – r[7]

Instrucciones suma (Add):

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

Format (3):

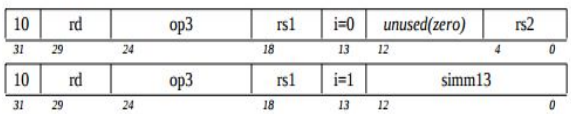


Suggested Assembly Language Syntax	
`add`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`addcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`addx`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`addxcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}

Instrucciones resta (Sub):

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

Format (3):



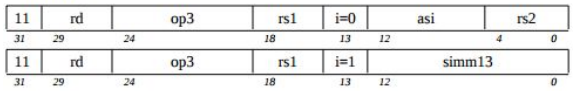
Suggested Assembly Language Syntax	
`sub`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`subcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`subx`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`subxcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}

Instrucciones de traer de memoria (Load).

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
LDSBA†	011001	Load Signed Byte from Alternate space
LDSHA†	011010	Load Signed Halfword from Alternate space
LDUBA†	010001	Load Unsigned Byte from Alternate space
LDUHA†	010010	Load Unsigned Halfword from Alternate space
LDA†	010000	Load Word from Alternate space
LDDA†	010011	Load Doubleword from Alternate space

† privileged instruction

Format (3):



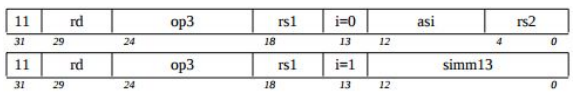
Suggested Assembly Language Syntax	
`ldsb`	[*address*], reg_{rd}
`ldsh`	[*address*], reg_{rd}
`ldub`	[*address*], reg_{rd}
`lduh`	[*address*], reg_{rd}
`ld`	[*address*], reg_{rd}
`ldd`	[*address*], reg_{rd}
`ldsba`	[*regaddr*] *asi* , reg_{rd}
`ldsha`	[*regaddr*] *asi* , reg_{rd}
`lduba`	[*regaddr*] *asi* , reg_{rd}
`lduha`	[*regaddr*] *asi* , reg_{rd}
`lda`	[*regaddr*] *asi* , reg_{rd}
`ldda`	[*regaddr*] *asi* , reg_{rd}

Instrucciones de guardar en memoria (Store):

opcode	*op3*	*operation*
STB	000101	Store Byte
STH	000110	Store Halfword
ST	000100	Store Word
STD	000111	Store Doubleword
STBA†	010101	Store Byte into Alternate space
STHA†	010110	Store Halfword into Alternate space
STA†	010100	Store Word into Alternate space
STDA†	010111	Store Doubleword into Alternate space

† privileged instruction

Format (3):

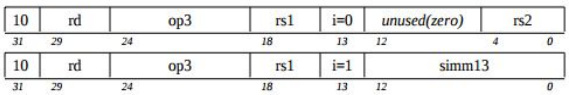


Suggested Assembly Language Syntax		
`stb`	reg_{rd} , [*address*]	(*synonyms:* `stub`, `stsb`)
`sth`	reg_{rd} , [*address*]	(*synonyms:* `stuh`, `stsh`)
`st`	reg_{rd} , [*address*]	
`std`	reg_{rd} , [*address*]	
`stba`	reg_{rd} , [*regaddr*] *asi*	(*synonyms:* `stuba`, `stsba`)
`stha`	reg_{rd} , [*regaddr*] *asi*	(*synonyms:* `stuha`, `stsha`)
`sta`	reg_{rd} , [*regaddr*] *asi*	
`stda`	reg_{rd} , [*regaddr*] *asi*	

Instrucciones Lógicas:

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

Format (3):

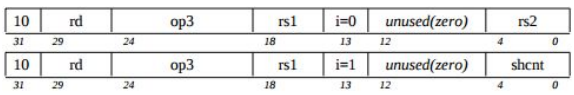


Suggested Assembly Language Syntax	
`and`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`andcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`andn`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`andncc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`or`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`orcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`orn`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`orncc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`xor`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`xorcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`xnor`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`xnorcc`	reg_{rs1} , *reg_or_imm* , reg_{rd}

Instrucciones shift:

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

Format (3):

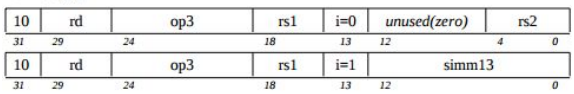


Suggested Assembly Language Syntax	
`sll`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`srl`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`sra`	reg_{rs1} , *reg_or_imm* , reg_{rd}

Instrucciones save and restore:

opcode	*op3*	*operation*
SAVE	111100	Save caller's window
RESTORE	111101	Restore caller's window

Format (3):



Suggested Assembly Language Syntax	
`save`	reg_{rs1} , *reg_or_imm* , reg_{rd}
`restore`	reg_{rs1} , *reg_or_imm* , reg_{rd}

OP:

Format	*op*	Instructions
1	1	CALL
2	0	Bicc, FBfcc, CBccc, SETHI
3	3	memory instructions
3	2	arithmetic, logical, shift, and remaining

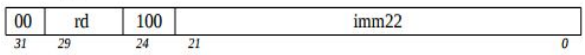
OP2:

op2	Instructions
0	UNIMP
1	unimplemented
2	Bicc
3	unimplemented
4	SETHI
5	unimplemented
6	FBfcc
7	CBccc

Instrucción SETHI:

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

Format (2):



Suggested Assembly Language Syntax	
`sethi`	*const22* , reg_{rd}
`sethi`	`%hi(`*value*`)` , reg_{rd}

Instrucción NOP:

opcode	*op*	*op2*	*operation*
NOP	00	100	No Operation

Format (2):

Suggested Assembly Language Syntax
`nop`

Instrucciones de salto Branch Integer condition codes :

opcode	*cond*	*operation*	*icc test*
BA	1000	Branch Always	1
BN	0000	Branch Never	0
BNE	1001	Branch on Not Equal	**not** Z
BE	0001	Branch on Equal	Z
BG	1010	Branch on Greater	**not** (Z or (N **xor** V))
BLE	0010	Branch on Less or Equal	Z **or** (N **xor** V)
BGE	1011	Branch on Greater or Equal	**not** (N **xor** V)
BL	0011	Branch on Less	N **xor** V
BGU	1100	Branch on Greater Unsigned	**not** (C **or** Z)
BLEU	0100	Branch on Less or Equal Unsigned	(C **or** Z)
BCC	1101	Branch on Carry Clear (Greater than or Equal, Unsigned)	**not** C
BCS	0101	Branch on Carry Set (Less than, Unsigned)	C
BPOS	1110	Branch on Positive	**not** N
BNEG	0110	Branch on Negative	N
BVC	1111	Branch on Overflow Clear	**not** V
BVS	0111	Branch on Overflow Set	V

Format (2):

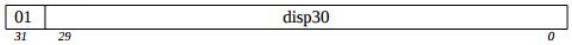
Suggested Assembly Language Syntax		
`ba{,a}`	*label*	
`bn{,a}`	*label*	
`bne{,a}`	*label*	(*synonym:* `bnz`)
`be{,a}`	*label*	(*synonym:* `bz`)
`bg{,a}`	*label*	
`ble{,a}`	*label*	
`bge{,a}`	*label*	
`bl{,a}`	*label*	
`bgu{,a}`	*label*	
`bleu{,a}`	*label*	
`bcc{,a}`	*label*	(*synonym:* `bgeu`)
`bcs{,a}`	*label*	(*synonym:* `blu`)
`bpos{,a}`	*label*	
`bneg{,a}`	*label*	
`bvc{,a}`	*label*	
`bvs{,a}`	*label*	

Instrucciones CALL and JMPL:

Call:

opcode	*op*	*operation*
CALL	01	Call and Link

Format (1):

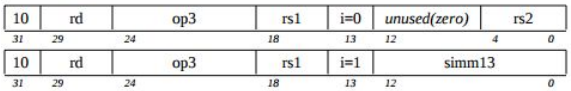


Suggested Assembly Language Syntax
`call` *label*

JMPL:

opcode	*op3*	*operation*
JMPL	111000	Jump and Link

Format (3):



Suggested Assembly Language Syntax
`jmpl` *address* , reg_{rd}