Sparc V8 Instructions

PC and NPC are used in executing and fetching instructions

1. The instruction pointed to by the PC executes while the next is fetched using NPC

2. Contents of NPC get copied into PC and NPC is updated
   - NPC is incremented by 4
   - For taken branches, NPC is overwritten with branch target address

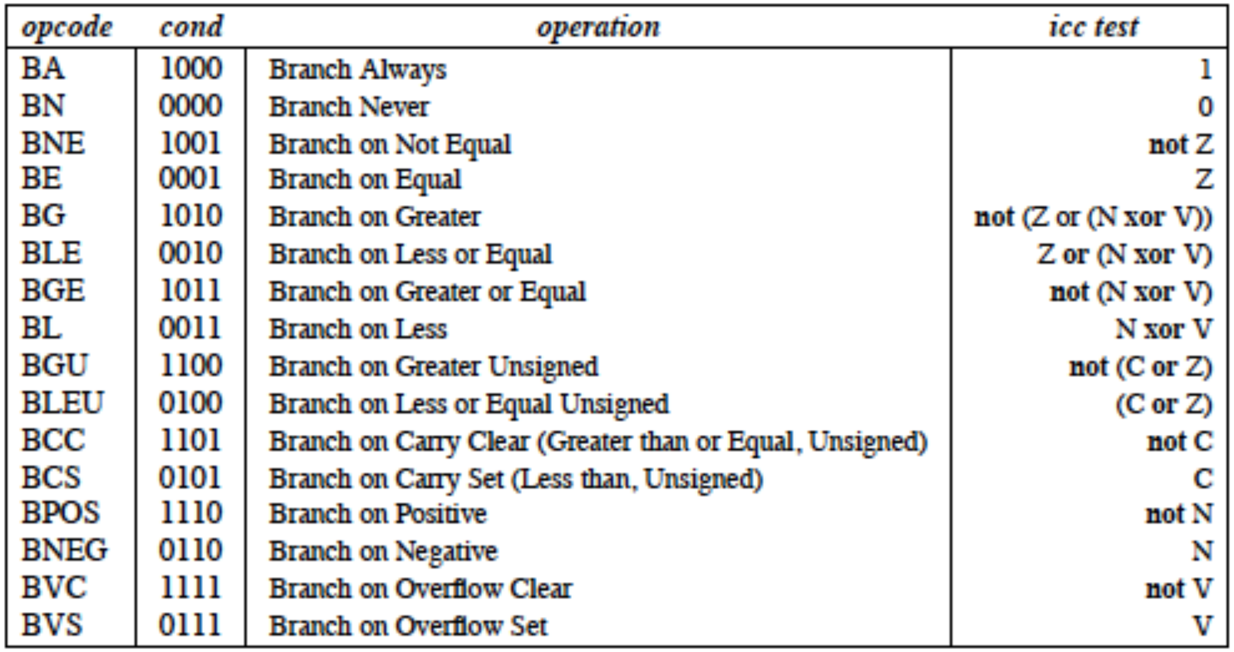
Branches use PC-relative addressing

Target address = PC + 4*sign-extended(disp22)
NPC = target address
Control is transferred by copying NPC into PC

JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

Sparc V8 Instructions

Branch Instructions



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

Conditional branching is based on condition codes

Delayed branching is used

instruction in delay slot always executes unless it is annulled

Sparc V8 Instructions

Annulment (controlled by bit29, the annul bit, in machine instruction)

Branches have a single delay slot (contains the delay instruction)

For conditional branches that are taken, the delay instruction is always executed (independent of the annul bit).

For conditional branches that are NOT taken, a=1 annuls delay instruction
(i.e., the instruction in delay slot is not executed)

E.g.: `bne,a   %g2,done`

Unconditional branches are always taken, a=1 annuls the delay instruction
(if the a bit = 0, the delay instruction is executed for unconditional branches)

E.g.: `ba,a    exit`

Sparc V8 Instructions

Two instructions support calling subroutines: call   and jmpl

call   func1   overwrites pc with the address corresponding to func1
        address of call instruction is written into %o7 (link register)
        target address = pc + 30-bit signed-displacement*4

        return address = %o7 + 8   to get past the instruction = the delay slot

call   func1   has same effect as   jmpl %o2, %o7   if %o2 contains address of func1
        using register as function pointer allows for dynamic addresses (e.g., jump table)

jmpl   %o7+8, %g0   same as ret (return) synthetic instruction (%g0 is read-only)

Caution: if function executes a save instruction, a new register window appears
        (%o7 becomes %i7)

Sparc V8 Instructions

Window Management

Subroutines and functions can use the save instruction to slide the register window
    new registers become visible (no need to save registers on stack)
    may also allocate space on stack (stack frame)

Example: save %sp, -512, %sp   gets new register window and allocates 512 bytes

return address = %i7 + 8   to get past the instruction = the delay slot

The restore instruction slides window back to previous registers (restores %sp)

Example:	ret	same as jmpl %i7+8, %g0
	restore	executes in delay slot of the ret instruction