Subroutines That Might Be Useful in a SPARC Assembler Program

- .mul (multiply)
  - Load arguments into %o0 and %o1
    - Values in %o0 and %o1 may be destroyed
  - Execute “call .mul”
  - Result is in %o0
- .div (divide)
  - Similar, but load dividend into %o0, and divisor into %o1
- _printf (Unix printf) (single value only…)
  - Load address of format into %o0
  - Load argument into %o1
- _scanf (Unix scanf) (single value only…)
  - Load address of format into %o0
  - Execute “add %fp,–12,%o1” before scanf, and “ld [%fp–12],%o1” after to get result

Calling Unix printf and scanf from a SPARC Assembler Program

```
    .data
prompt: .ascii    "Please enter an integer: \0"
infmt:  .ascii    "%d\0"
res:    .ascii    "Value is %d\n\0"

    .text
    .global   _main          ! main must be global
    .global   _printf        ! linker will find printf
    .global   _scanf         ! linker will find printf
_main:
    save %sp, -64, %sp       ! space to save register

    set   prompt, %o0        ! load address of "prompt"
    call  _printf            ! call printf to print out
    nop                      !   the prompt message

    set   infmt, %o0         ! load addr of input format
    add   %fp,-12,%o1        ! store result on stack
    call  _scanf             ! call scanf to read an
    nop                      !   integer
    ld    [%fp-12],%o1       ! move result into %o1

    set   res, %o0           ! load address of "prompt"
    call  _printf            ! call printf to print out
    nop                      ! value returned by scanf

    mov   1, %g1             ! exit request
    ta    0                  ! trap (return) to Unix
```


SPARC Branch Instructions (Partial List)

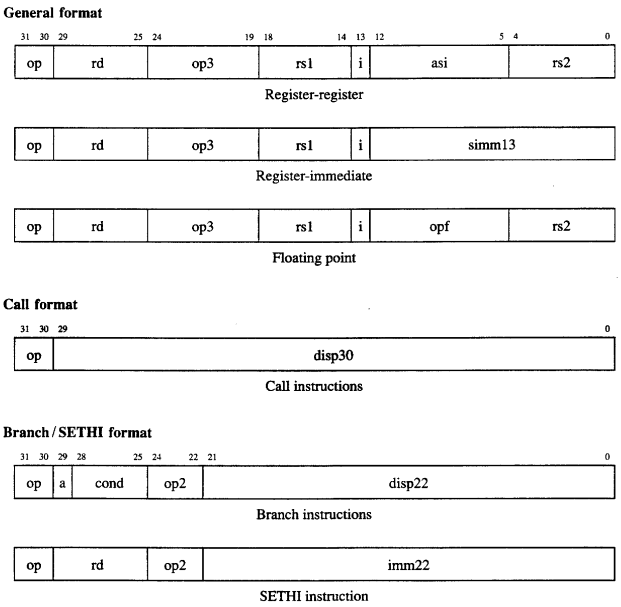
- Branch instructions (use condition codes)

ba	branch always
be	branch if equal (***to zero***)
bne	branch if not equal (***to zero***)
bl	branch if less than ***zero***
ble,bg,bge	branch if <=, >, >= ***zero***
bpos	branch if positive
bneg	branch if negative

  - These names make the most sense if the previous instruction is “subcc”
  - For now, always include “nop” (no operation) after a branch instruction
- cmp (compare) (synthetic instruction)

cmp	%l2,%l3	subcc	%l2,%l3,%g0
cmp	%l2,300	subcc	%l2,300,%g0

SPARC Instruction Formats



Finding Largest Integer in an Array

```
        .data
arr:    .word   1,45,-16,23,38,17       ! int arr[6] = {...}
msg:    .ascii  "Value is %d\n\0"

        .text
        .global _main                   ! main must be global
        .global _printf                 ! linker will find printf
_main:
        save %sp, -64, %sp              ! space to save registers

        mov  0, %l0                     ! %l0 (counter) = 0
        set  arr, %l1                   ! %l1 is base of arr
        mov  0, %l2                     ! %l2 (index) = 0
        ld   [%l2+%l1],%l3              ! %l3 (maxnum) = arr[0]

for:    cmp  %l0,6                      ! if (counter < 6) enter loop
        bge  end                        ! otherwise print answer
        nop
        ld   [%l1+%l2],%l4              ! %l4 (temp) = arr[index]
        cmp  %l4,%l3                    ! if (arr[index} > maxnum)
        ble  ok
        nop
        mov  %l4,%l3                    ! max num = arr[index]
ok:     inc  %l0                        ! counter++
        add  %l2,4,%l2                  ! index = index + 4
        ba   for
        nop
```


Finding Largest Integer in an Array (cont.)

```
end:    mov  %l3, %o1                   ! copy maxnum into %o1
        set  msg, %o0                   ! load address of "msg"
        call _printf                    ! call printf to print out
        nop                             !   the number in %o1

        mov  1, %g1                     ! exit request
        ta   0                          ! trap (return) to Unix
```