Arithmetic and Bit Instructions

```
add rs1, op2, rd          rd = rs1 + op2
sub rs1, op2, rd          rd = rs1 - op2
sll rs1, op2, rd          rd = rs1 << op2
sra rs1, op2, rd          rd = rs1 >> op2 (arith.)
srl rs1, op2, rd          rd = rs1 >> op2 (logical)
and rs1, op2, rd          rd = rs1 & op2
or  rs1, op2, rd          rd = rs1 | op2
xor rs1, op2, rd          rd = rs1 ^ op2
```

where op2 is either a register or a constant in the range -4096..4095

Comparison

```
subcc rs1, op2, rd        rd = rs1 - op2
                          and sets condition codes
                          based on comparison of
                          result with zero
```

where op2 is either a register or a constant in the range -4096..4095

Conditional Branches

```
ba  label                 Branch always
bn  label                 Branch never

bl  label                 Branch on < 0
ble label                 Branch on <= 0
be  label                 Branch on == 0
bne label                 Branch on != 0
bge label                 Branch on >= 0
bg  label                 Branch on > 0
```

Load and Store

```
ldub [address], rd        load unsigned byte
ldsb [address], rd        load signed byte
ld   [address], rd        load 32-bit word
stb  rs, [address]        store byte
st   rs, [address]        store 32-bit word
```

where address is either a register, register plus a constant (in the range -4096..4095), or sum of two registers

Subroutine Instructions

```
jmpl address, rd          Jump to address, put
                          address of jmpl into rd
call address              Same as jmpl address, %o7
ret                       Same as jmpl %i7+8, %g0

save %sp, const, %sp      Push register window
                          and add const to stack pointer
restore                   Pop register window
```

Warnings:

- jmpl, call, and ret all have delay slots
- Also, if you don't put the restore in the ret delay slot then you can't use the ret instruction! (After the restore, return address is found in %o7, not %i7.)
- Also, constant in the save should be a negative multiple of 8.

Multiplication and Division

```
call .mul                 %o0 = %o0 * %o1
call .umul                %o0 = %o0 * %o1
call .div                 %o0 = %o0 / %o1
```

Warning: these calls have a delay slot, and (like any call) apparently may overwrite %o1..%o5

Miscellaneous

```
nop                           does nothing
mov op1, rd                   rd = op1
   where op1 is either a register or a constant in the
   range  -4096..4095
set value, rd                 rd = value
   where value is either a label (address) or a constant.
       [Warning:  set is actually a macro, which may
        expand to the two instructions
              sethi %hi(value), rd
              or    %rd, %lo(value), rd
       so don't put this in a delay slot!]

label:                        creates a label
```

If All Else Fails…

- Write a short C routine that does what you want, and look at what assembly code the compiler generates.
- The command `cc -S foo.c` generates the assembly file `foo.s`