SPARC Architecture

• 8-bit cell (byte) is smallest addressable unit

• 32-bit addresses, i.e., 32-bit virtual address space

SPARC is a big-endian (big end, or most-significant end, first) machine

SPARC Registers

• 32, 32-bit wide general-purpose registers

%r0 … %r31
%g0 … %g7
%r0 … %r7
%g0 … %g7
%o0 … %o7
%r8 … %r15
%l0 … %l7
%i0 … %i7
%f0 … %f31

• The groups relate to procedure calling conventions

• Some registers have dedicated uses

• Other registers

SPARC Register Map

• Other special registers (manipulated by special instructions):

- Register %g0 always has the value 0 when read; writing it has no effect

- Floating point registers (%f0 … %f31)

- Program counter (%pc)

- Next program counter (%npc)

SPARC Register Map, cont'd

• Larger size: 4 address

• 32-bit addresses, i.e., 32-bit virtual address space

• 8-bit cell (byte) is smallest addressable unit

See page 193 in the SPARC Architecture Manual.
Assembly vs. Machine Language

- **Machine language** is the *bit patterns* that represent instructions
- **Assembly language** is a *symbolic representation* of machine language
- **Assemblers** translate from assembly language to machine language
  
  ```
  add %i1,360,%o2 is a format 3 instruction: 022401460550
  ```

Assemblers: mapping an assembly instruction to a machine instruction (1-to-1)

Compilers: mapping a statement to 1 or many assembly instructions

- **Disassemblers** translate from machine language to an assembly language

---

SPARC Instruction Set

- Instruction groups
  - load/store instructions
  - integer arithmetic and bitwise logical instructions
  - control instructions (branches, calls)
  - special instructions (operating system)
  - floating point arithmetic and conversion

Instruction formats (see page 44, Ch. 8 in Paul)

<table>
<thead>
<tr>
<th>op</th>
<th>rd</th>
<th>op2</th>
<th>imm22</th>
<th>disp2</th>
<th>31 29 24 18 13 12 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>25</td>
<td>360</td>
<td>0 25 1</td>
<td>360</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>op</th>
<th>rd</th>
<th>op2</th>
<th>imm13</th>
<th>disp2</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>12</td>
<td>0</td>
<td>31 1</td>
<td>550 octal</td>
</tr>
</tbody>
</table>

Addresses must be aligned: for address `A`
- halfword
- word
- double word

---

Store Instructions

- Move data from a register to memory
  - Storing bytes and halfwords
    - the rightmost bits are stored
    - the leftmost bits are ignored
  - Storing double words
    - must be even

---

Load Instructions

- Load: move data from memory to a register
  - Fetched byte or halfword appears right-justified in the 32-bit register
  - Leftmost bits are zero-filled or sign-extended
  - A double word is loaded into a register pair and must be even
  - The most-significant word lands in the least-significant word in the register
  - Addresses must be aligned: for address `A`
    - halfword
    - word
    - double word
SPARC has two addressing modes to yield an effective address:

1. add the contents of two registers
2. add the contents of a register and a signed, 13-bit number

Common names:
1. register indirect or deferred
   `ld [%o1],%o2`
2. register indexed (above is a special case that uses `%g0`)
   `st %o1,[%o2+%o3]`

Assembly-language syntax:

- `%reg reg + %g0 reg + N N + reg %g0 + N`
- `%reg reg + 10 %g0 + 10 %reg`

Address synonym:

- `N` is a 13-bit integer constant