

Class 8: RISC Example: SPARC

Announcements

- There WILL be this week
 - Partners
 - You may work with one partner for in-lab, post-lab (same partner)
 - Credit any other help received from others
- Homework
 - MUST be legible, but you don't need to type it.
 - Anything illegible **will** be given a grade of 0

CPI and MIPS

- Instruction mix
 - 20% branches, 4.2 ns
 - 60% ALU, 1.9 ns
 - 20% loads, stores
- Clock period - 500 ps
- Average CPI?
- Average MIPS?

SPARC

Scalable Processor
Architecture

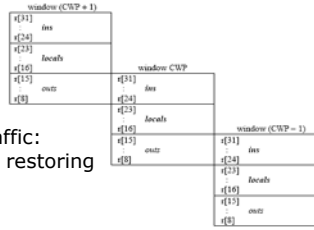
SPARC Architecture

- 1987, Sun Microsystems
- Open architecture
 - Implementation is made available (VHDL (v8) or Verilog (v9))
- Original memory model
 - Linear, 32-bit virtual address space
 - Virtual memory
 - Technique used to allow simulating more main memory than is physically available on the machine (uses disks)

SPARC Basics

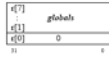
- First implementation
 - 16.6 MHz clock, 10 MIPS (Ave. CPI 1.66)
- Load-Store
- General register
- Processing Units
 - Integer unit
 - FP unit
 - Coprocessor

Register Windows

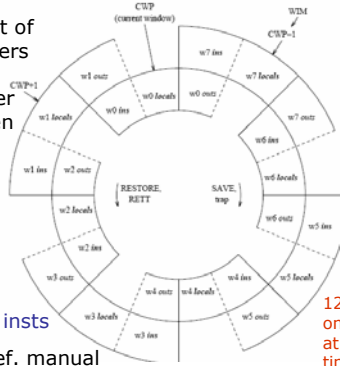


Memory traffic:
Saving and restoring registers

Sparc v8 ref. manual



Only subset of total registers visible to programmer at any given time



Adjust CWP with save and restore insts

Sparc v8 ref. manual

120 registers, only 24 visible at any given time

Register Windows

- Windows wrap around
 - When full, first window is spilled to memory

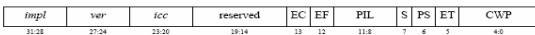
Why Have Windows?

- Parameter passing
 - Saves memory accesses, most data can be held in registers
 - When no room, spill to memory
- Why not make all 120 visible?

Processor and Memory

- 2 PCs
 - PC
 - nPC - next value of PC
- 32 programmer visible general registers (32-bit),
 - r0=0, calls write to r[15] (out[7])
- Processor Status Register (PSR)

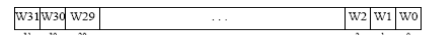
Figure 4-3 PSR Fields



Processor and Memory

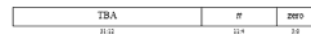
- Window invalid mask register (WIM)

Figure 4-5 WIM Fields



- Trap base register (TBR)

Figure 4-6 TBR Fields



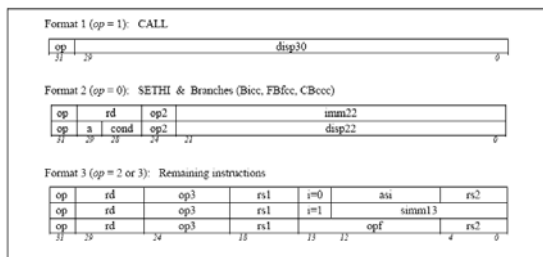
Processor and Memory

- Multiply step register (Y)
- Memory addresses
 - 32 bits
 - Big endian
 - word == 32 bits
- Memory mapping unit (MMU)
 - Allows multiple address spaces
 - load/store alternate instructions
- 32 FP registers (no windows)
 - May be accessed randomly or as a stack

Available Data Formats

- Signed, unsigned integer
 - 8, 16, 32, 64
 - Floating point
 - 32, 64, 128 bit
- Tagged data (ls 2 bit reserved for user-defined type)
- word doubleword quadword
-

Instruction Formats and Addressing Modes



Loads/Stores

- 2 modes
 - Register+register
 - Register + sign-extended, immediate 13-bit constant
- Other modes can be synthesized using the above

Instruction Set

- Data movement
 - Load
 - Store
 - Swap
- Arithmetic
 - Original: no multiply or divide
 - logic, area
 - changed with integration densities

Instructions (2)

- Branch and Control Transfer
 - Branch delay slot
 - Branch insts have annul bit
 - Allows programmer to control whether the instruction in the slot is executed or not
 - non-delayed, delayed, conditional delayed

Delay Instruction

- nPC
 - Often next sequential instruction
 - If after a delayed control transfer instruction, target of the branch

PC before execution	nPC before execution	Instruction
8	12	not-a-CTI
12	16	delayed CTI to 40
16	40	not-a-CTI
⋮	⋮	
40	44	...

Conditional Delay

- Depends on annul bit value

a bit	Type of branch	Delay instruction executed?
a = 0	conditional, taken	Yes
	conditional, not taken	Yes
	unconditional, taken (BA, etc)	Yes
	unconditional, not taken (BN, etc)	Yes
a = 1	conditional, taken	Yes
	conditional, not taken	No (annulled)
	unconditional, taken (BA, etc)	No (annulled)
	unconditional, not taken (BN, etc)	No (annulled)

SPARC Assembly

- label: instruction !comments
- destination is always rightmost
- Register names:
 - output: %r8-%r15 aka %o0-%o7
 - local: %r16 - %r23 aka %l0-%l7
 - input: %r24-%r31 aka %i0-%i7
- Also, can load 32-bit address to register with:


```
sethi %hi(w), %r1
ld [%r1+%lo(w)], %r2
```

Interrupts and Traps

- Interrupts (outside)
 - Traps (internal)
1. Save the register window
 2. Store PC, nPC, PSR
 3. Disable traps
 4. Transfer control to trap handler

Pipelining

- MB86900 – 4 pipeline stages
 - Instruction fetch
 - Instruction decode
 - Instruction execute
 - Write results

SPARC Summary

- RISC-like
 - Load/Store
 - Arithmetic only operates on registers
 - Regular instruction format
 - Limited addressing modes
 - Faster decoding and operand resolution
 - Branch delays (pipelining)