Chapter 15

Reduced Instruction Set Computers (RISC)

Table 15.1

Characteristics of Some CISCs, RISCs, and Superscalar Processors

	Complex Instruction Set (CISC) Computer			Reduced Instruction Set (RISC) Computer	
Characteristic	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000
Year developed	1973	1978	1989	1987	1991
Number of instructions	208	303	235	69	94
Instruction size (bytes)	2–6	2–57	1–11	4	4
Addressing modes	4	22	11	1	1
Number of general-purpose registers	16	16	8	40 - 520	32
Control memory size (kbits)	420	480	246	—	—
Cache size (kB)	64	64	8	32	128

Table 15.1

Characteristics of Some CISCs, RISCs, and Superscalar Processors

	Superscalar		
Characteristic	PowerPC	Ultra SPARC	MIPS R10000
Year developed	1993	1996	1996
Number of instructions	225		
Instruction size (bytes)	4	4	4
Addressing modes	2	1	1
Number of general-purpose registers	32	40 - 520	32
Control memory size (kbits)	—	—	—
Cache size (kB)	16-32	32	64

(Table can be found on page 538 in the textbook.)

High Level Languages (HLL) and Semantic Gap

HLL:

1. Allows Programmer to express algorithms more concisely
2. Allows the Compiler to take care of details that are not

Important in the programmer's expression of algorithms

3. Often support naturally the use of structured programming and/or object oriented design.

These give rise to SEMANTIC GAP:

Which is the difference between the operations provided in HLL and those Provided in Computer Architecture.

The symptoms of this Semantic Gap are:

1. Execution in-efficiency.
2. Excessive Machine Program size.
3. Compiler Complexity.

Initiative to Reduce the gap:

1. Large Instruction Set
2. Dozens of addressing mode.
3. Various HLL statements implemented Hardware.

Table 15.2

Weighted Relative Dynamic Frequency of HLL Operations [PATT82a]
(Research on Program Execution Characteristics on CISC Architecture)

	Dynamic Occurrence		Machine-Instruction Weighted		Memory-Reference Weighted	
	Pascal	C	Pascal	C	Pascal	C
ASSIGN	45%	38%	13%	13%	14%	15%
LOOP	5%	3%	42%	32%	33%	26%
CALL	15%	12%	31%	33%	44%	45%
IF	29%	43%	11%	21%	7%	13%
GOTO	—	3%	—	—	—	—
OTHER	6%	1%	3%	1%	2%	1%

Conclusion:
Procedure Call/Return is the most time-consuming operation in typical HLL program.

(Table can be found on page 564(10e) in the textbook.)

Table 15.3

Dynamic Percentage of Operands

	Pascal	C	Average
Integer Constant	16%	23%	20%
Scalar Variable	58%	53%	55%
Array/Structure	26%	24%	25%

Conclusion:
Most Memory references are to the simple scalar local (to the procedure) Variables.

Procedure Calls

1. Procedure calls and returns are an important aspect of HLL programs. Because these are most time consuming operations in compiled HLL program.
2. Two Significant Aspects:

a) Number of parameters and variables that a procedure deal with.
b) Depth of Nesting

3. 98 % procedures pass fewer than 6 six arguments.
4. 92% procedures use fewer than 6 local variables.
5. It is rare to have a long uninterrupted sequence of procedure calls followed by the corresponding sequence of returns.
6. Rather program remains confined to a narrow window of procedure-invocation depth.

Implications

- HLLs can best be supported by optimizing performance of the most time-consuming features of typical HLL programs

- Three elements characterize RISC architectures:
  - Use a large number of registers (required for local variables)or use a compiler to optimize register usage. This will reduce memory reference.
  - Careful attention needs to be paid to the design of instruction pipelines (because of high proportion of conditional branch and procedure calls)
  - Instructions should have predictable costs (measured in execution time, code size and in energy dissipation) and be consistent with a high-performance implementation

The Use of a Large Register File

Software Solution

- Requires compiler to allocate registers
- Allocates based on most used variables in a given time
- Requires sophisticated program analysis

Hardware Solution

- More registers
- Thus more variables will be in registers

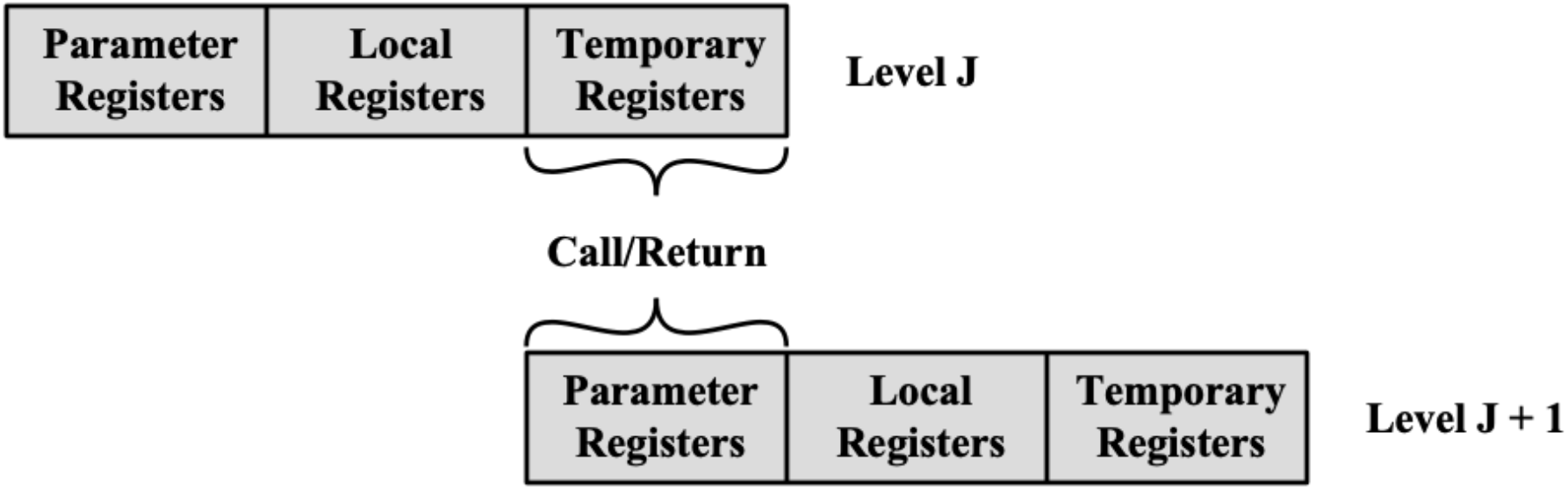


Figure 15.1 Overlapping Register Windows

1. An N-window register
File can only hold N-1 Procedure Activation.

2. It has been seen that with 8 active Windows, a save or restore needed on only 1% of the calls or returns.

3. Berkley RISC Computers use 8 Windows of 16 registers each.

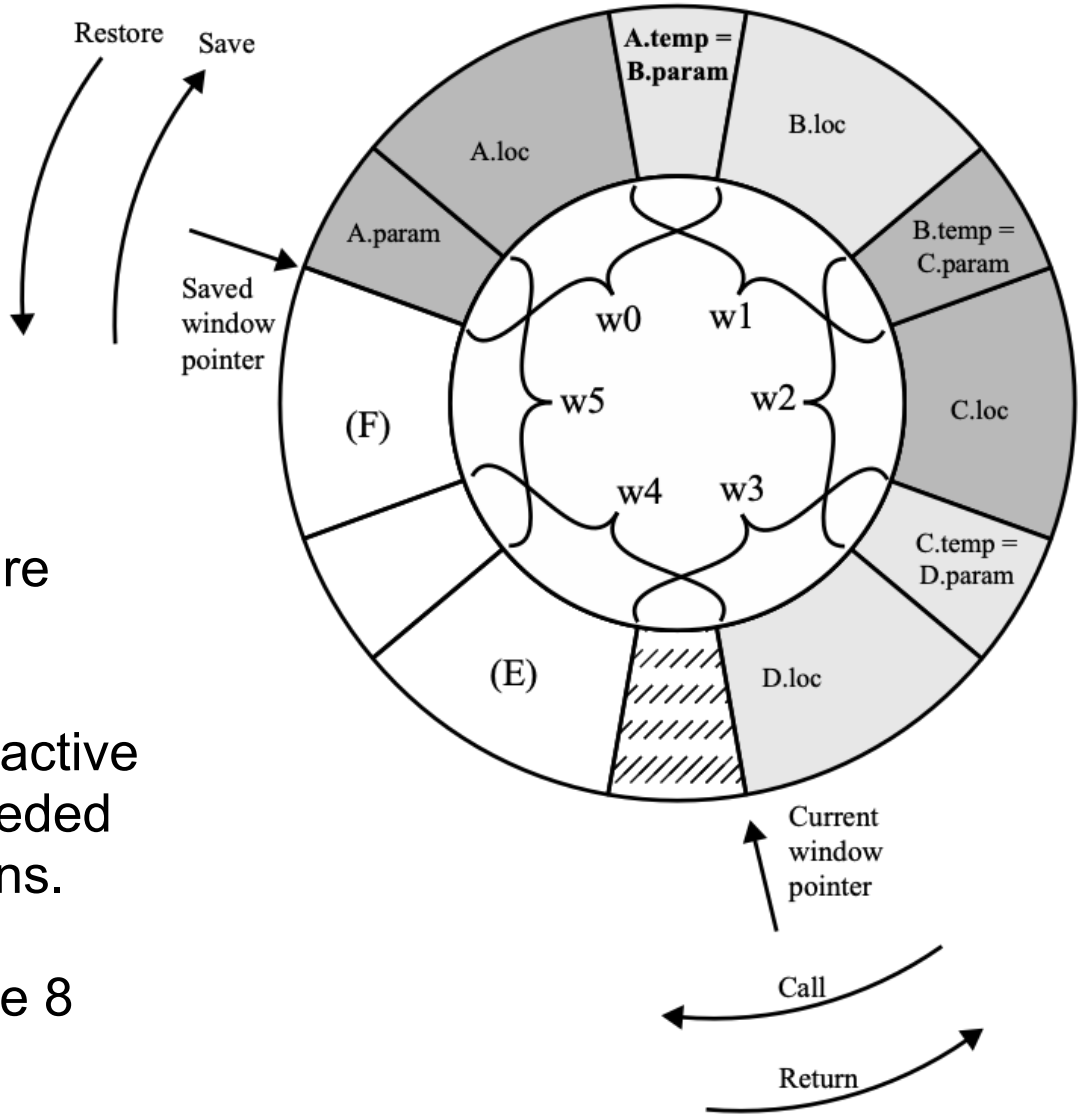


Figure 15.2 Circular-Buffer Organization of Overlapped Windows

Global Variables

- Variables declared as global in an HLL can be assigned memory locations by the compiler and all machine instructions that reference these variables will use memory reference operands
  - However, for frequently accessed global variables this scheme is inefficient

- Alternative is to incorporate a set of global registers in the processor
  - These registers would be fixed in number and available to all procedures
  - A unified numbering scheme can be used to simplify the instruction format, like register 0 to 7 could refer to unique global registers, and references to register 8 to 31 could be offset to refer to physical registers in the current window.

- There is an increased hardware burden to accommodate the split in register addressing

- In addition, the linker must decide which global variables should be assigned to registers

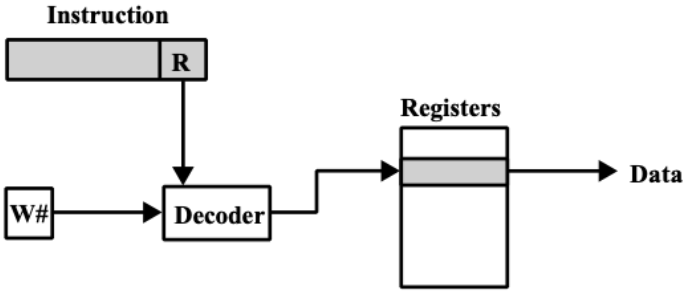
Table 15.5

Characteristics of Large-Register-File and Cache Organizations

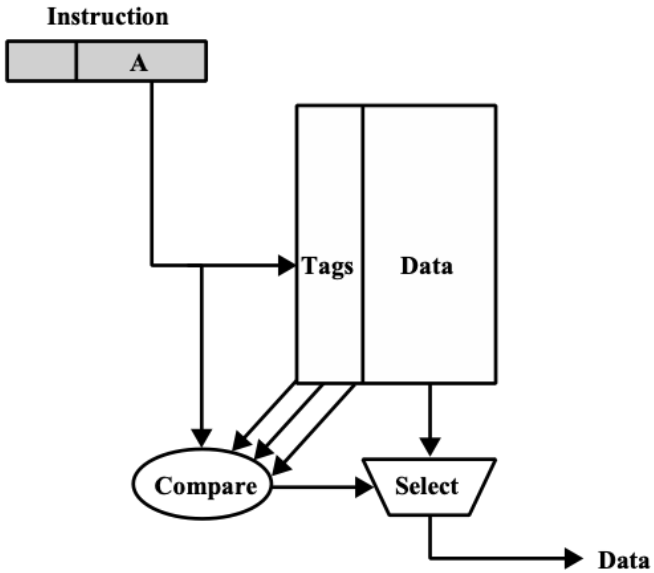
Large Register File	Cache
All local scalars	Recently-used local scalars
Individual variables	Blocks of memory
Compiler-assigned global variables	Recently-used global variables
Save/Restore based on procedure nesting depth	Save/Restore based on cache replacement algorithm
Register addressing	Memory addressing
Multiple operands addressed and accessed in one cycle	One operand addressed and accessed per cycle

(Table can be found on page 546 in the textbook.)

From the performance point of view, for local scalars, the window-based register file is superior compared to Cache.



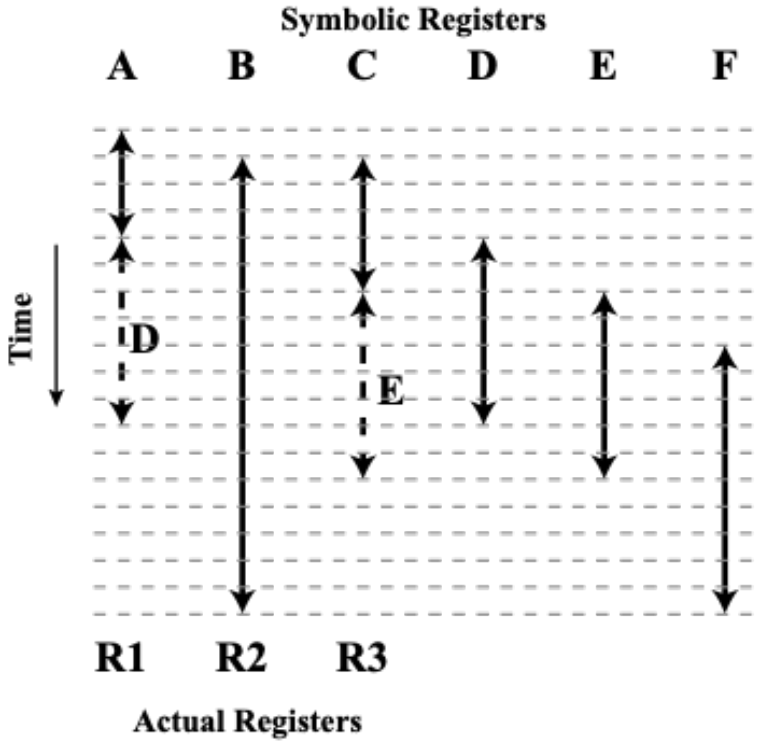
(a) Windows-based register file



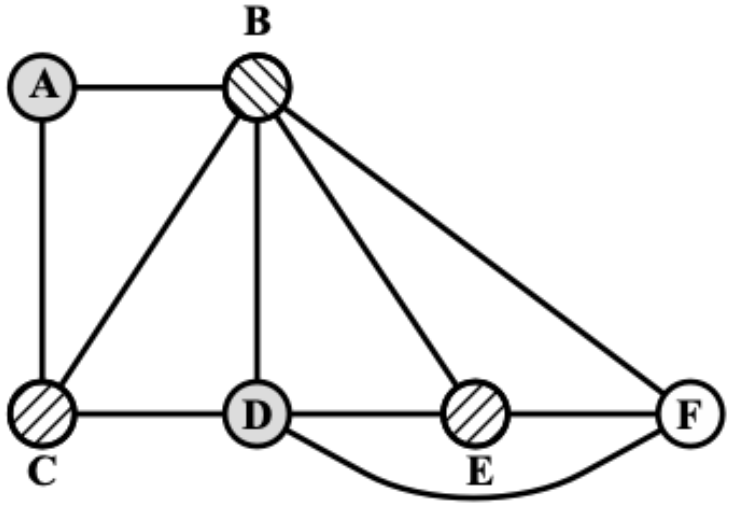
(b) Cache

Figure 15.3 Referencing a Scalar

Compiler-Based Register Optimization



(a) Time sequence of active use of registers



(b) Register interference graph

Figure 15.4 Graph Coloring Approach

Why CISC ?

(Complex Instruction Set Computer)

- There is a trend to richer instruction sets which include a larger and more complex number of instructions

- Two principal reasons for this trend:
  - A desire to simplify compilers
  - A desire to improve performance

- There are two advantages to smaller programs:
  - The program takes up less memory
  - Should improve performance
    - Fewer instructions means fewer instruction bytes to be fetched
    - In a paging environment smaller programs occupy fewer pages, reducing page faults
    - More instructions fit in cache(s)

Table 15.6

Code Size Relative to RISC I

	[PATT82a] 11 C Programs	[KATE83] 12 C Programs	[HEAT84] 5 C Programs
RISC I	1.0	1.0	1.0
VAX-11/780	0.8	0.67	
M68000	0.9		0.9
Z8002	1.2		1.12
PDP-11/70	0.9	0.71	

[Vax-11 has much more complex instruction set then PDP-11]

(Table can be found on page 550 in the textbook.)

Characteristics of Reduced Instruction Set Architectures

One machine instruction per machine cycle	• *Machine cycle* --- the time it takes to fetch two operands from registers, perform an ALU operation, and store the result in a register
Register-to-register operations	• Only simple LOAD and STORE operations accessing memory • This simplifies the instruction set and therefore the control unit
Simple addressing modes	• Simplifies the instruction set and the control unit
Simple instruction formats	• Generally only one or a few formats are used • Instruction length is fixed and aligned on word boundaries • Opcode decoding and register operand accessing can occur simultaneously

Typical Characteristics of Classic RISC Processors

1. A Single Instruction Size
2. That size is typically 4 bytes.
3. A small number of Data Addressing Modes, typically <5.
4. NO Indirect Addressing
5. No Load/Store with Arithmetic operations.
6. One Memory-Addressed operand per Instruction.
7. Does not support Arbitrary Alignment of data for load/store operations.
8. At least 32 or more integer registers (5bit in the instruction)
9. At least 16 or more floating-point registers (4bit in the instruction)

- Item 1 to 3 reduces Instruction decode complexity.
- Item 4 to 7 helps better pipelining performance.
- Item 8 and 9 helps writing better compiler.

Table 15.7

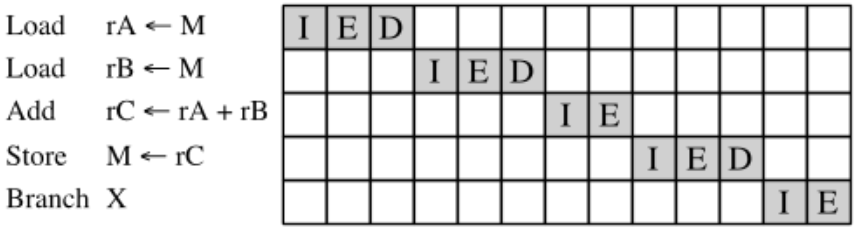
Characteristics of Some Processors
(first 8 are RISC, next 5 are CISC and last two are mixed)

Processor	Number of instruc-tion sizes	Max instruc-tion size in bytes	Number of addressing modes	Indirect addressing	Load/store combined with arithmetic	Max number of memory operands	Unaligned addressing allowed	Max Number of MMU uses	Number of bits for integer register specifier	Number of bits for FP register specifier
AMD29000	1	4	1	no	no	1	no	1	8	3[a]
MIPS R2000	1	4	1	no	no	1	no	1	5	4
SPARC	1	4	2	no	no	1	no	1	5	4
MC88000	1	4	3	no	no	1	no	1	5	4
HP PA	1	4	10[a]	no	no	1	no	1	5	4
IBM RT/PC	2[a]	4	1	no	no	1	no	1	4[a]	3[a]
IBM RS/6000	1	4	4	no	no	1	yes	1	5	5
Intel i860	1	4	4	no	no	1	no	1	5	4
IBM 3090	4	8	2[b]	no[b]	yes	2	yes	4	4	2
Intel 80486	12	12	15	no[b]	yes	2	yes	4	3	3
NSC 32016	21	21	23	yes	yes	2	yes	4	3	3
MC68040	11	22	44	yes	yes	2	yes	8	4	3
VAX	56	56	22	yes	yes	6	yes	24	4	0
Clipper	4[a]	8[a]	9[a]	no	no	1	0	2	4[a]	3[a]
Intel 80960	2[a]	8[a]	9[a]	no	no	1	yes[a]	—	5	3[a]

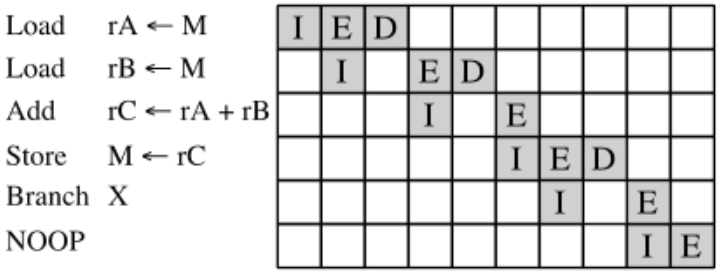
a RISC that does not conform to this characteristic.
b CISC that does not conform to this characteristic.

(Table can be found on page 578 in the 10e textbook.)

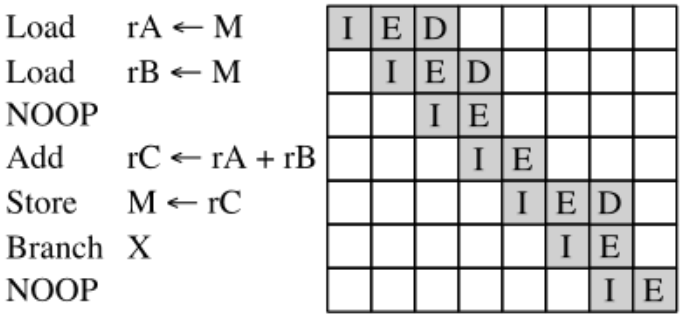
RISC Pipelining



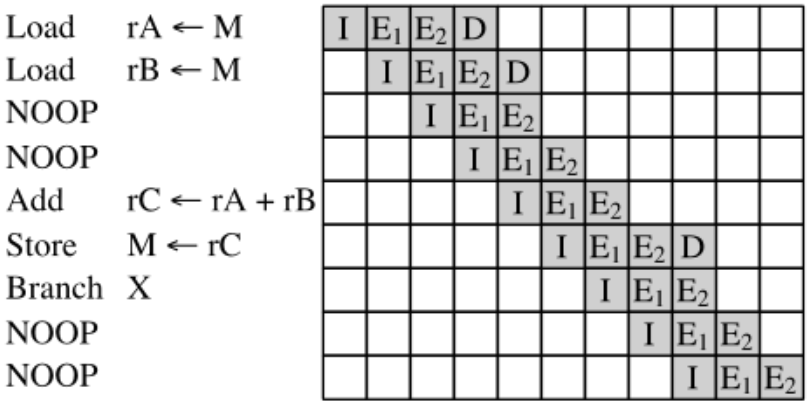
(a) Sequential execution



(b) Two-stage pipelined timing



(c) Three-stage pipelined timing



(d) Four-stage pipelined timing

I: Instruction Fetch
E: Execute
D: Data Access

E phase is longer, so divide
E_1: Register File Read
E_2: ALU Ops and Register Write

Figure 15.6 The Effects of Pipelining

MIPS R4000

One of the first commercially available RISC chip sets was developed by MIPS Technology Inc.

Inspired by an experimental system developed at Stanford

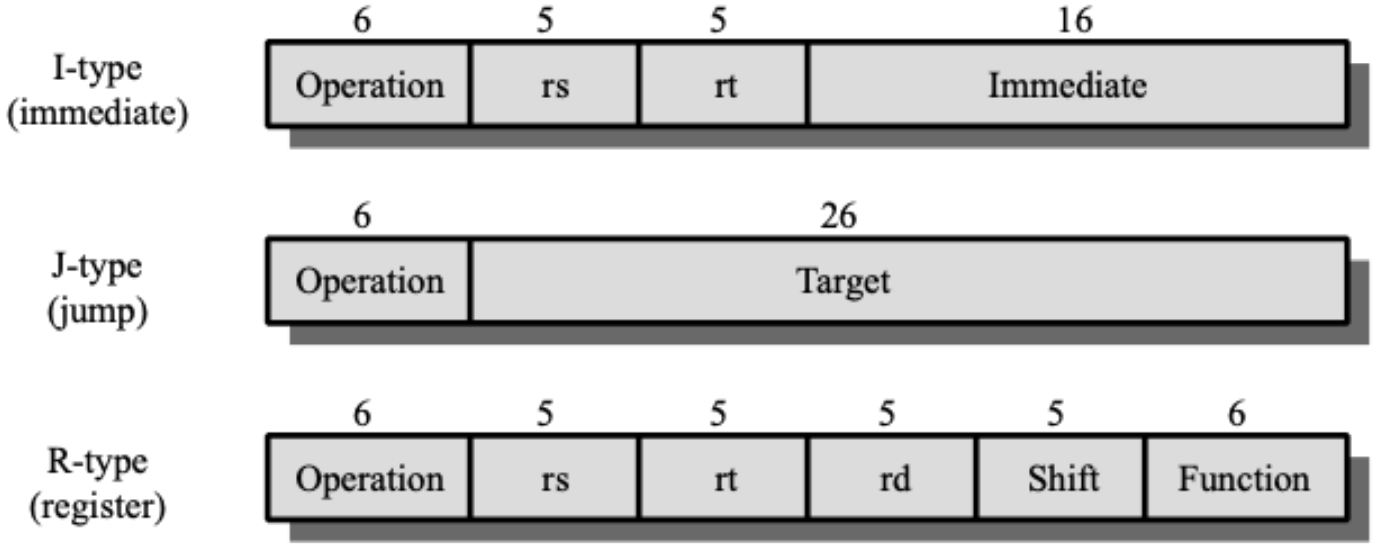
Has substantially the same architecture and instruction set of the earlier MIPS designs (R2000 and R3000)

Uses 64 bits for all internal and external data paths and for addresses, registers, and the ALU

Is partitioned into two sections, one containing the CPU and the other containing a coprocessor for memory management

Supports thirty-two 64-bit registers

Provides for up to 128 Kbytes of high-speed cache, half each for instructions and data

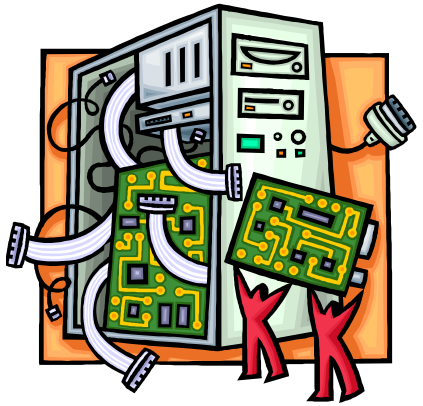


Operation	Operation code
rs	Source register specifier
rt	Source/destination register specifier
Immediate	Immediate, branch, or address displacement
Target	Jump target address
rd	Destination register specifier
Shift	Shift amount
Function	ALU/shift function specifier

Figure 15.9 MIPS Instruction Formats

SPARC

Scalable Processor Architecture



- Architecture defined by Sun Microsystems
- Sun licenses the architecture to other vendors to produce SPARC-compatible machines
- Inspired by the Berkeley RISC 1 machine, and its instruction set and register organization is based closely on the Berkeley RISC model

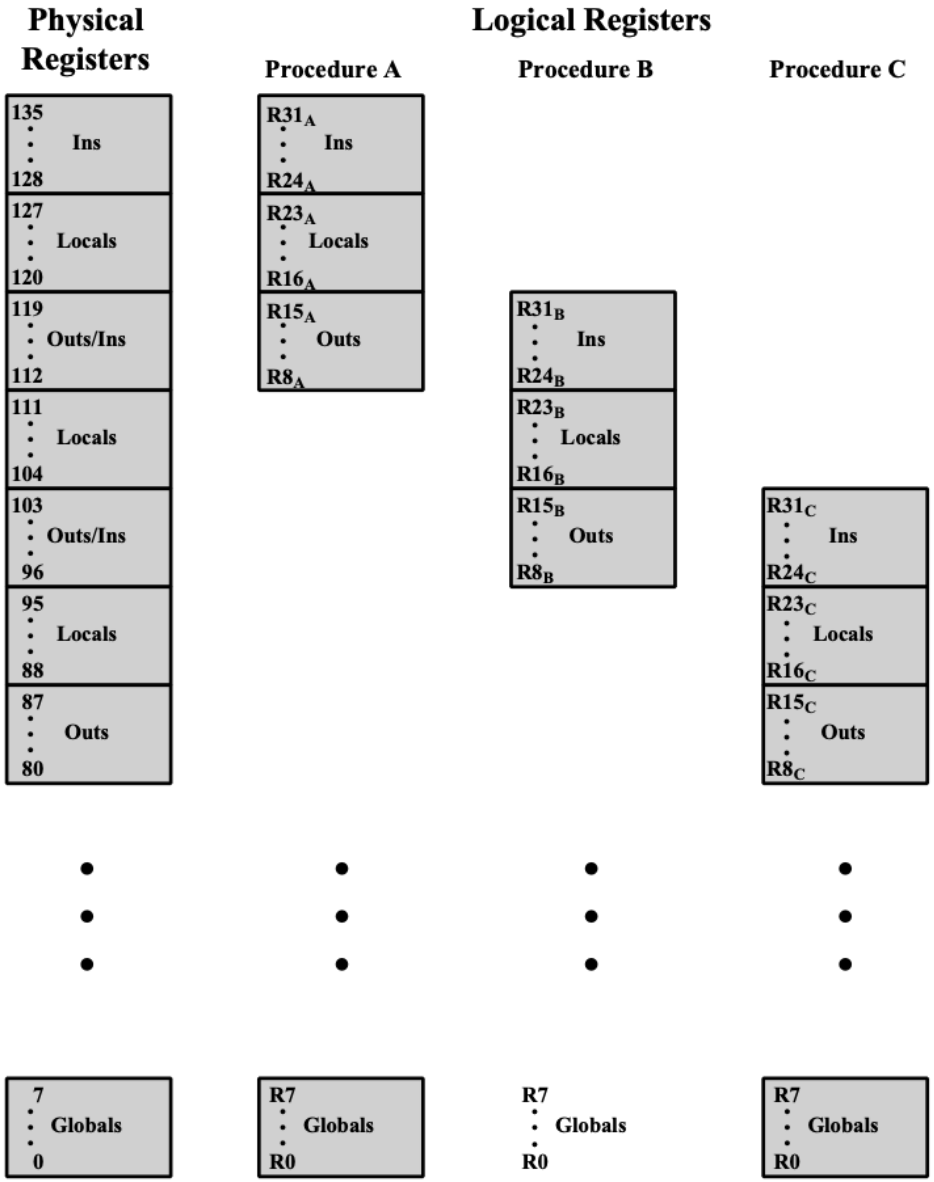


Figure 15.12 SPARC Register Window Layout with Three Procedures

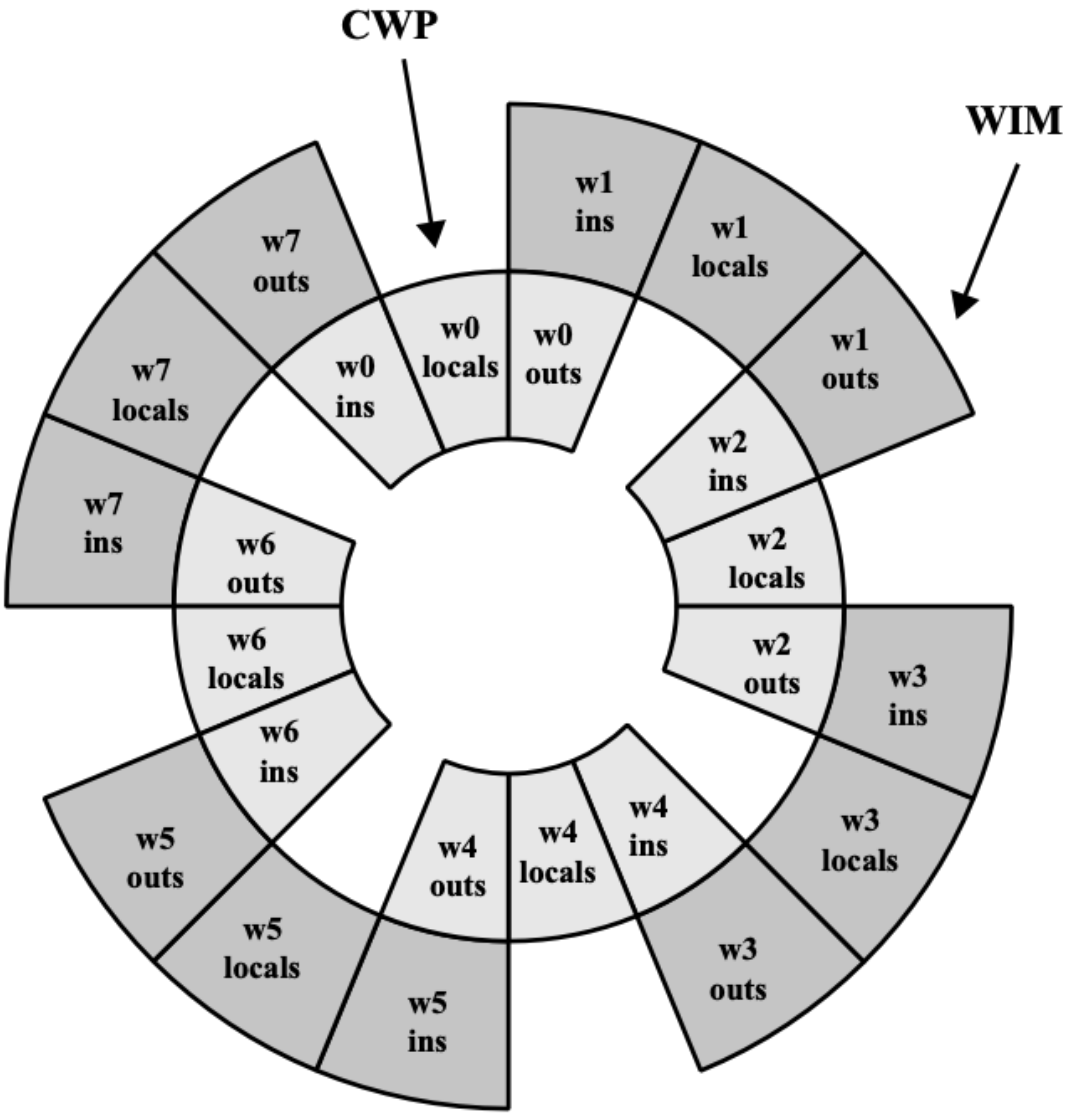


Figure 15.13 Eight Register Windows Forming a Circular Stack in SPARC

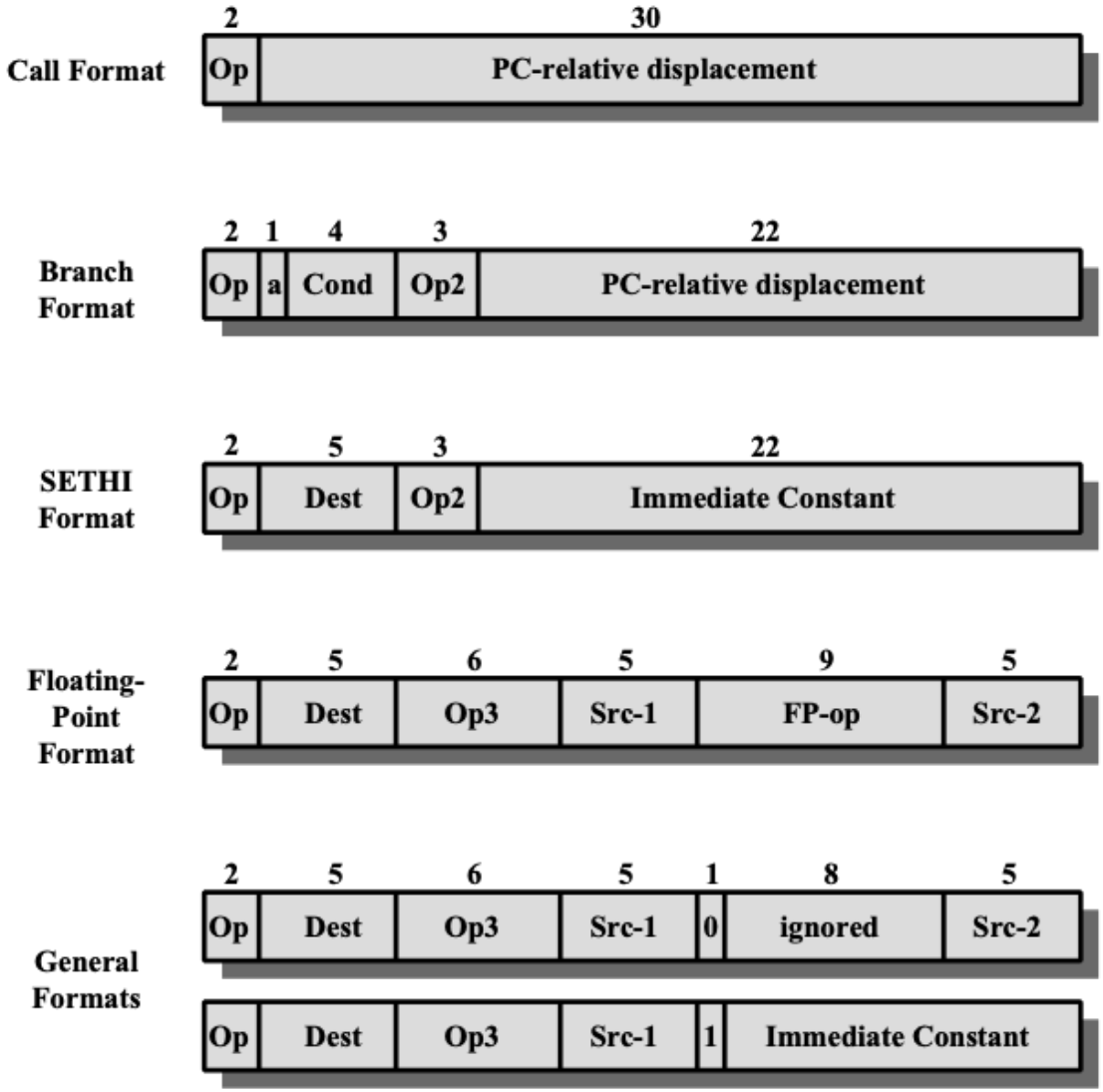


Figure 15.14 SPARC Instruction Formats

RISC versus CISC Controversy

- Quantitative
  - Compare program sizes and execution speeds of programs on RISC and CISC machines that use comparable technology
- Qualitative
  - Examine issues of high level language support and use of VLSI real estate
- Problems with comparisons:
  - No pair of RISC and CISC machines that are comparable in life-cycle cost, level of technology, gate complexity, sophistication of compiler, operating system support, etc.
  - No definitive set of test programs exists
  - Difficult to separate hardware effects from complier effects
  - Most comparisons done on "toy" rather than commercial products
  - Most commercial devices advertised as RISC possess a mixture of RISC and CISC characteristics