Chapter 13 RISC

Major Advances

- separate I/O processor
- family concept
- microprogrammed control unit
- memory hierarchy – cache, virtual memory (demand page)
- pipelining – instruction pipeline, vector processor (data pipeline)
- parallel processor
- RISC architecture

RISC Architecture

- a limited and simple instruction set
- a large number of general-purpose registers – fast operand referencing
- careful design of instruction pipeline – conditional branch and procedure call

Table 13.1 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	Berkeley RISC I
Year developed	1973	1978	1989	1987	1991	1981
Number of instructions	208	303	235	69	94	31
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	138
Control memory size (Kbits)	420	480	246			
Cache size (KBytes)	64	64	8	32	128	

Reduced Instruction Set Computer (RISC)

Motivation

Example. 5 x 10

```
[CISC]      mov   ax, 10
            mov   bx, 5
            mul   bx        // result on dx:ax //
```

```
[RISC]      mov   ax, 0
            mov   bx, 10
            mov   cx, 5
        L1: add   ax, bx
            loop  L1
```

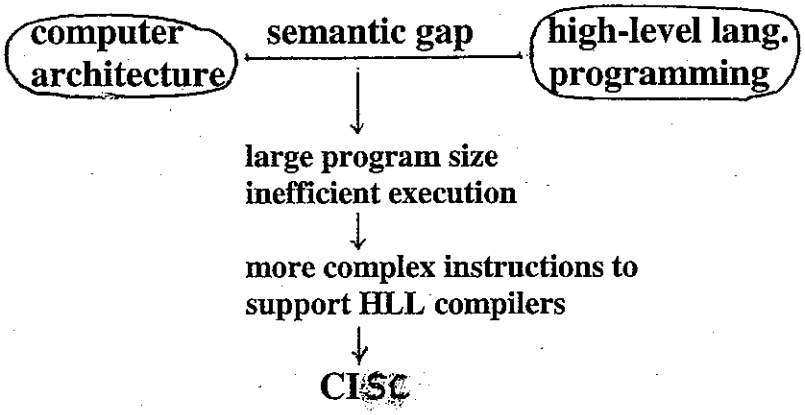
Assumption. 1 cycle per instruction
30 cycles per multiplication

CISC: (2 moves x 1 cycle) + (1 multiplication x 30 cycles) = 32 cycles

RISC: (3 moves x 1 cycle) + (5 adds x 1 cycle) + (5 loops x 1 cycle) = 13 cycles

Also, RISC clock cycle is shorter than the CISC cycle.

Instruction Execution Characteristics



RISC ➔ make the architecture that supports the HLL simpler

Background and Motivation

Instruction execution characteristics
- **operations performed**
- **operands used**
- **execution sequencing**

- **Operation**

TABLE 13.2 Weighted Relative Dynamic Frequency of HLL Operations [PATT82a]

	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

dynamic frequency: Assignment 45%
Sequence control 34% } **optimize**
timing: procedure call/return – most time-consuming

→ careful design of instruction pipelining (∵ high propotion of conditional branch and procedure call)

- **Operands**

TABLE 13.3 Dynamic Percentage of Operands

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

80% are local variables → localized scalar - subject to optimize
→ large number of registers (to reduce memory access)

o Procedure Call

arguments: 98% fewer than 6 arguments
local variables: 92% fewer than 6 local variables
depth is narrow

→ simple instruction set

Large Number of Registers

- register is the fastest among all (cache, main memory, ...)
strategy: most frequently accessed operands should be kept in registers.
1. compiler – smart compiler for optimization (software approach)
2. more registers (hardware approach)

Overlapping Register Windows

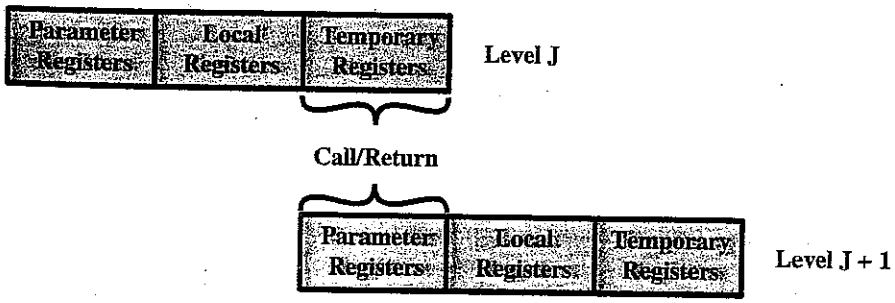


Figure 13.1 Overlapping Register Windows.

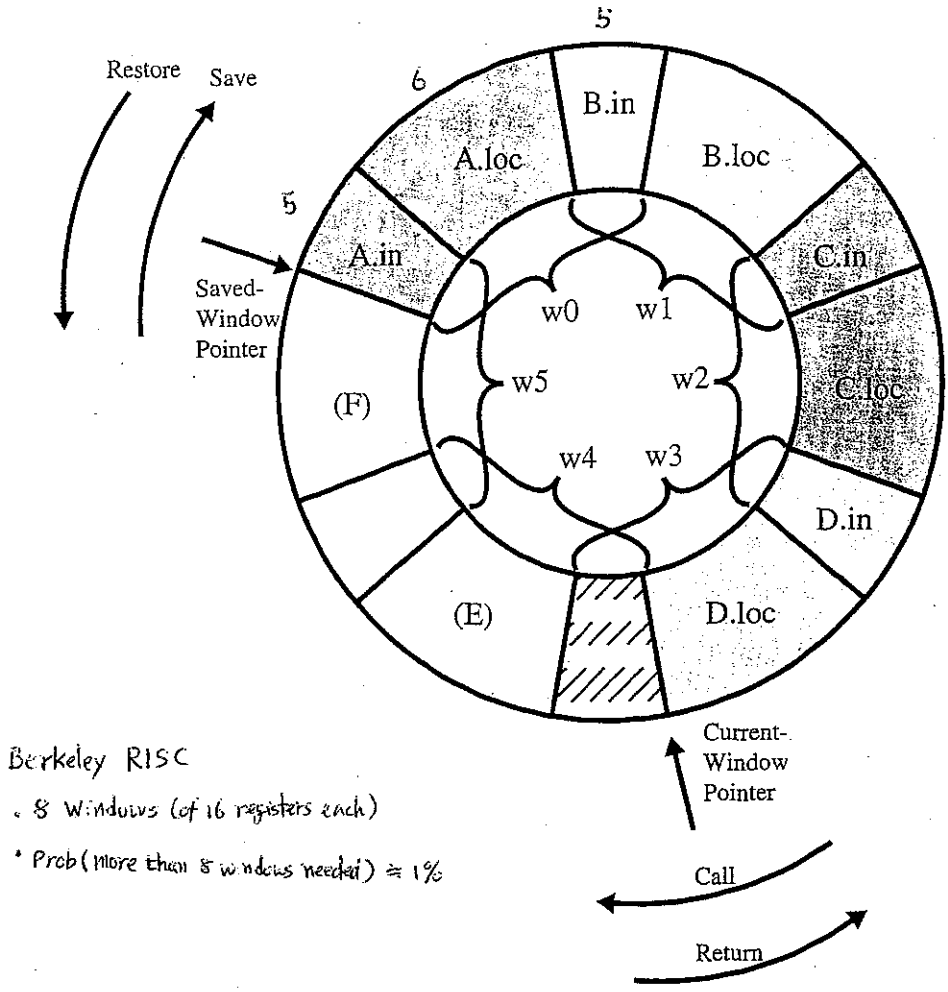


Figure 13.2 Circular-Buffer Organization of Overlapped Windows.

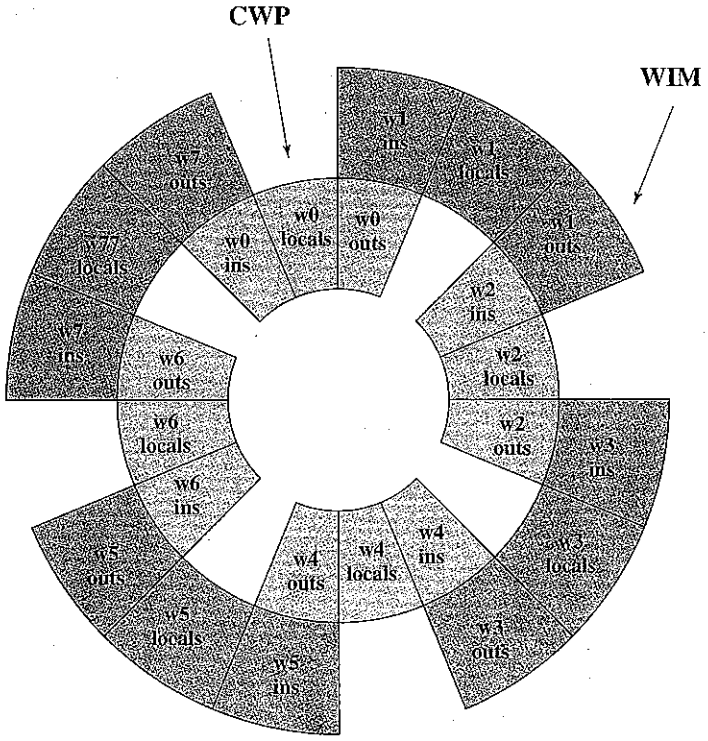
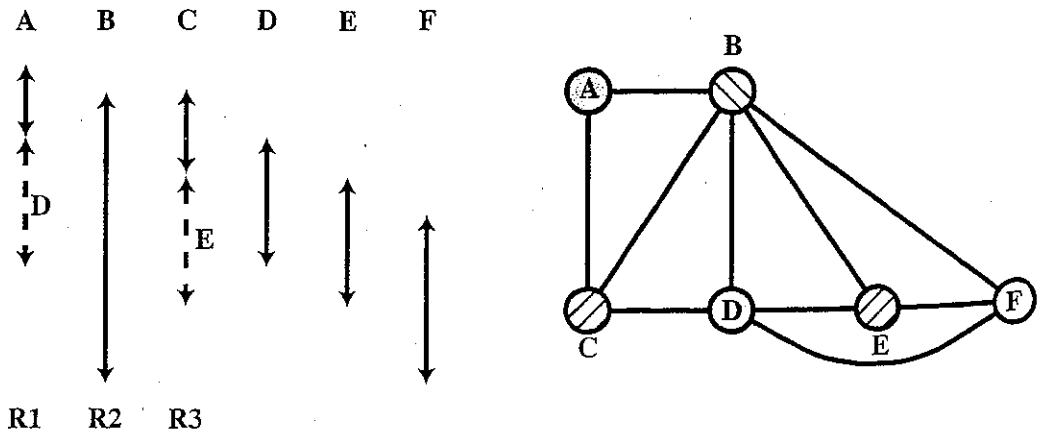


Figure 13.12 Eight Register Windows Forming a Circular Stack in SPARC

13.3 Compiler-Based Register Optimization

symbolic registers
physical registers
mapping (Find the way to use minimum # of registers)



(a) Time Sequence of Active Use of Registers (b) Register Interference Graph

Figure 13.4 Graph Coloring Approach.

Note. 32 ~ 64 registers are enough with register optimization technique

13.4 Reduced Instruction Set Architecture

Why CISC?

(1) Simple compiler – machine instruction ≒ HLL instruction
(2) Smaller program – expect less memory, fewer instruction fetch, less page faults, etc
(3) Faster program

Reality

- CISC program not much shorter

Table 13.6 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	

- Speed
  CICS – memory
  RISC – registers See Fig 13.5

Note: Because of large number of registers in use, RISC instruction needs less number of memory access → fast execution

Characteristics of RISC

- large number of registers
- one instruction per cycle – hard-wired machine instruction (instead of microcoding)
- register-to-register operations – simple instruction set and simple control unit – one size
- simple addressing modes – no indirect addressing (< 5)
- simple instruction formats (4 bytes)

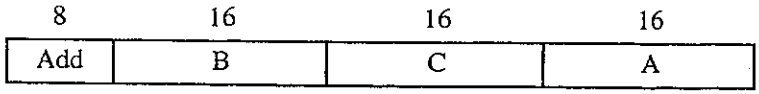
→ Microchip design process becomes easier.

Table 13.7 Design and Layout Effort for Some Microprocessors

CPU	Transistors (thousands)	Design (person-months)	Layout (person-months)
RISC I	44	15	12
RISC II	41	18	12
M68000	68	100	70
Z8000	18	60	70
Intel iAPx-432	110	170	90

Trend

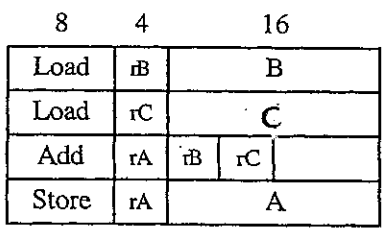
Hybrid



8	16	16	16
Add	B	C	A

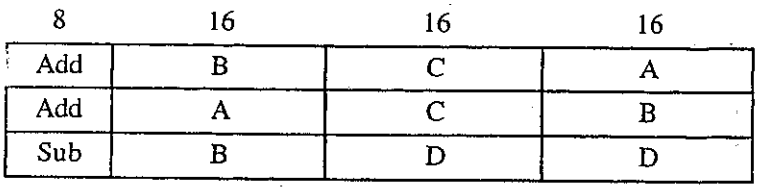
Memory-to-Memory
I = 56, D = 96, M = 152

(a) $A \leftarrow B + C$



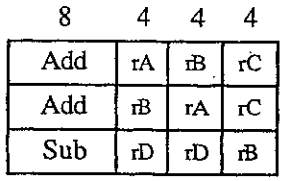
8	4	16		
Load	rB	B		
Load	rC	C		
Add	rA	rB	rC	
Store	rA	A		

Register-to-Memory
I = 104, D = 96, M = 200



8	16	16	16
Add	B	C	A
Add	A	C	B
Sub	B	D	D

Memory-to-Memory
I = 168, D = 288, M = 456



8	4	4	4
Add	rA	rB	rC
Add	rB	rA	rC
Sub	rD	rD	rB

Register-to-Memory
I = 60, D = 0, M = 60

(b) $A \leftarrow B + C$; $B \leftarrow A + C$; $D \leftarrow D - B$

I = Size of executed instructions
D = Size of executed data
M = I + D = Total memory traffic

Figure 13.5 Two Comparisons of Register-to-Register and Memory-to-Memory Approaches.

1W = 32 bits

13-9

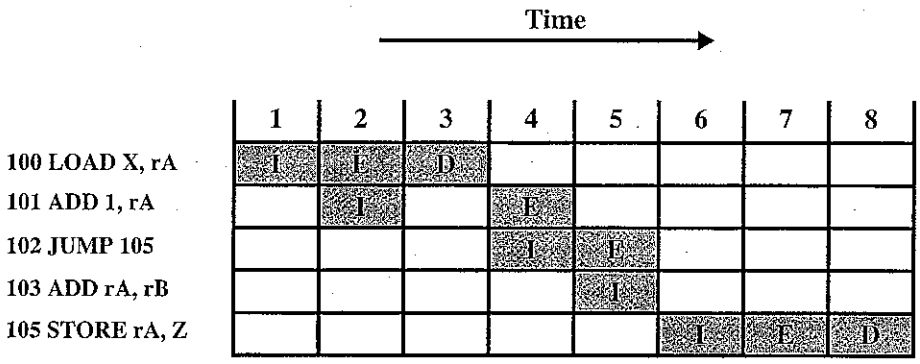
Table 13.7 Characteristics of Some Processors

Processor	Number of instruction sizes	Max instruction 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]

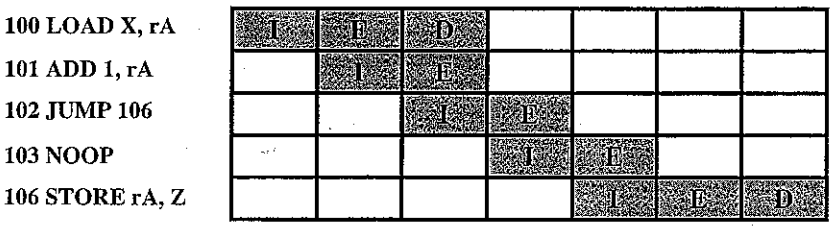
[a]RISC that does not conform to this characteristic.
[b]CISC that does not conform to this characteristic.

RISC Pipelining - "Delayed Branch"

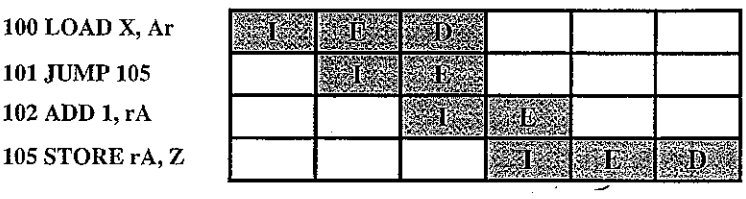
- o Load/Store
  1. I (instruction fetch)
  2. E (execute – calculate memory address)
  3. D (memory – data movement)
- o Other instructions
  1. I
  2. E



(a) Traditional Pipeline



(b) RISC Pipeline with Inserted NOOP



(c) Reversed Instructions

Figure 13.7 Use of the Delayed Branch