Profile of instruction sizes | formats

target/environment	2-byte	4-byte	$\geq$ 6-byte	comp. average
RISC	0%	100%	0%	4 bytes
IBM/390 scient./system	40%	60%	0%	3.2 bytes
IBM/390 commercial	30%	50%	20%	3.8 bytes
VAX scient./system	25%	50%	25%	4 bytes
VAX commercial	30%	50%	20%	3.8 bytes

Instructions executed per 100 HLL statements (average)

architecture	machine instructions
RISC	200
IBM/390	180
VAX	120

Profile of machine instructions executed per 100 HLL statements in

(a) scientific applications (FORTRAN programs)

instruction class	RISC	RM	VAX
MOVEs	107 | 54%	87 | 48%	27 | 23%
BRANCHes	26 | 13%	26 | 15%	26 | 22%
FL_ARITH	24 | 12%	24 | 13%	24 | 20%
INT_ARITH	16 | 8%	16 | 9%	16 | 13%
LOGICs	27 | 13%	27 | 15%	27 | 22%
total	200	180	120

(b) commercial applications (COBOL programs)

instruction class	RISC	RM | VAX
MOVE.Ws (int)	75 | 11%	28 | 15%
MOVE.Cs (char)	364 | 53%	16 | 9%
BRANCHes	138 | 20%	72 | 40%
DEC_ARITH	46 | 7%	13 | 7%
INT_ARITH	15 | 2%	15 | 8%
LOGIC (on ints)	15 | 2%	15 | 8%
LOGIC (on chars)	35 | 5%	22 | 12%
total	688	181

(c) system runs and transaction processing

instruction class	RISC	RM	VAX
MOVEs	107 | 54%	87 | 48%	27 | 23%
BRANCHes	47 | 23%	47 | 26%	47 | 39%
INT_ARITH	19 | 10%	19 | 11%	19 | 16%
LOGICs	27 | 13%	27 | 15%	27 | 22%
total	200	180	120

Some data on process management

Procedure calls

Number of (RISC-) instructions between calls and returns

programs	maximum	average
C (10 programs)	333	100
PASCAL (13 programs)	72	53
ALGOL (2 programs)	40	27

Number of (RISC-) instructions involved in procedure calls

operation	average no_of_instr
call | return	2
local stack adjustment	2
parameter passing	1.5
register save | restore	5
returning a value	0.6
total	11

Call frequency per HLL statements

low	average	high
<< 1%	1..3%	10..12%

Fraction of program execution time spent in

user mode	system mode	library functions
60	30	10

System calls

Instructions executed per operating system function

operation	VAX	R2000	SPARC	i860
system call | return	12	84	128	86
traps | interrupts	14	103	145	155
page table swaps	11	36	15	559
context switches	9	135	326	618

Process (register) state to be changed by context switches

	VAX	S/390	R2000	SPARC	i860
GP registers	16	16	32	136	132
FP registers	0	8	32	32	32
others (PSW etc.)	1	2	5	6	9

The curse of RISC architectures

Modern operating systems

$\rightarrow$ are decomposed into independent servers communicating with users and among themselves using RPC (remote procedure calls) $\rightsquigarrow$ require frequent system calls | effect frequent interrupts
$\rightarrow$ support more fine-grained multithreaded applications $\rightsquigarrow$ adding more process state $\rightsquigarrow$ system calls
$\rightarrow$ generally require more dynamic memory management

with the consequence that

$\rightarrow$ very large register sets limit the performance of modern operating systems due to expensive context switches
$\rightarrow$ frequent context switches also have a negative effect on the performance of RISC processors due to complete flushes of cache memories
$\rightarrow$ out-of-order instruction execution in superscalar processors significantly complicates task management and context switches $\rightsquigarrow$ imprecise interrupts

$\rightarrow$ the performance gained at the user level may be lost at the system level

Data relating to pipeline processing

Various profiles of branch instructions

Distribution of branch classes across all application areas

(un-)condit branches	loop control	proc calls | returns
72%	10%	18%

Target address

in register	per address computation
18%	82%

Percentage of unconditional vs. all branches

scientific	commercial	system
20	40	35

Percentage of conditional branches that go to target

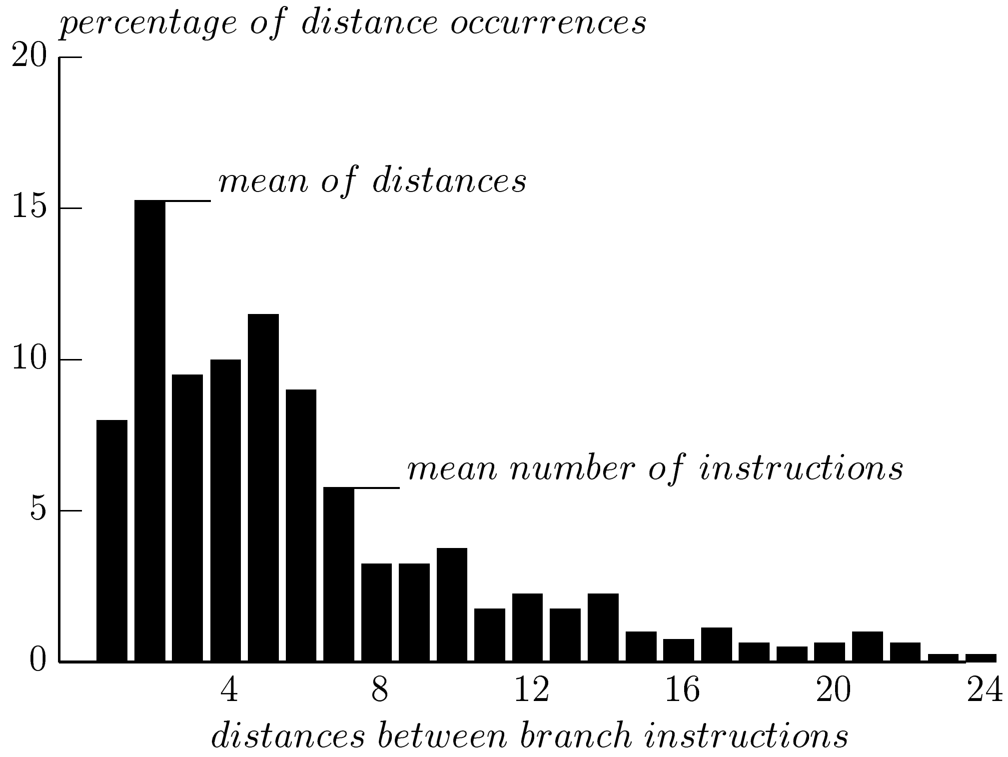
scientific	commercial	system
54	41	50

Loop control (all applications)

going to target	going in line
91%	9%

Unconditional branches and procedure calls | returns of course go to target 100%

Distribution of distances between occurrences of branch instructions in typical RM machine programs



Note that though the mean distance between branch instructions is 7.5, roughly 66% of all distances are shorter than that (3 to 4 instructions on average).

Distribution of arithmetic instruction over 100 arithmetic instructions total

<table>
<tr><th></th><th>add|sub</th><th>mult</th><th>div</th><th>move (pack |unpack)</th></tr>
<tr><td>FL_instr (scient)</td><td>54%</td><td>43%</td><td>3%</td><td>−−</td></tr>
<tr><td>FP_instr (all apps)</td><td>80%</td><td>15%</td><td>5%</td><td>−−</td></tr>
<tr><td>DEC_instr (comm)</td><td>39%</td><td>4%</td><td>3%</td><td>54%</td></tr>
</table>

Profile of MOVE instructions per 100 HLL statements

(a) scientific applications

<table>
<tr><th></th><th>RISC</th><th>RM</th><th>VAX</th></tr>
<tr><td>int load LD.W</td><td>36 | 34%</td><td>24 | 28%</td><td>11 | 41%</td></tr>
<tr><td>int store ST.W</td><td>16 | 15%</td><td>13 | 15%</td><td>6 | 22%</td></tr>
<tr><td>float load LD.F</td><td>23 | 21%</td><td>17 | 21%</td><td>2 | 7%</td></tr>
<tr><td>float store ST.F</td><td>18 | 17%</td><td>16 | 18%</td><td>1 | 4%</td></tr>
<tr><td>int reg move MOVE.W</td><td>10 | 9%</td><td>12 | 14%</td><td>3 | 11%</td></tr>
<tr><td>float reg move MOVE.F</td><td>2 | 2%</td><td>3 | 3%</td><td>2 | 7%</td></tr>
<tr><td>load | store mult LDM | STM</td><td>2 | 2%</td><td>2 | 2%</td><td>2 | 8%</td></tr>
<tr><td>total</td><td>107</td><td>87</td><td>27</td></tr>
</table>

(b) commercial applications

<table>
<tr><th></th><th>RISC</th><th>RM| VAX</th></tr>
<tr><td>int load LD.W</td><td>54 | 12%</td><td>18 | 41%</td></tr>
<tr><td>int store ST.W</td><td>16 | 4%</td><td>5 | 11%</td></tr>
<tr><td>mem_to_mem move MOVE.C</td><td>364 | 83%</td><td>16 | 36%</td></tr>
<tr><td>load mult LDM</td><td>4 | 1%</td><td>4 | 9%</td></tr>
<tr><td>store mult STM</td><td>1 | 0%</td><td>1 | 3%</td></tr>
<tr><td>total</td><td>439</td><td>44</td></tr>
</table>

(c) system runs and transaction processing

	RISC	RM	VAX
int load `LD.W`	65 | 61%	50 | 57%	14 | 52%
int store `ST.W`	23 | 21%	18 | 21%	5 | 19%
int reg move `MOVE.W`	13 | 12%	13 | 15%	2 | 7%
load mult `LDM`	4 | 4%	4 | 5%	4 | 15%
store mult `STM`	2 | 2%	2 | 2%	2 | 7%
total	107	87	27

Address generation and data dependencies of instructions

distance	address interlocks	data interlocks
1 instr back	9.9%	40.3%
2 instr back	8.4%	14.7%
3 instr back	2.8%	3.6%
4 instr back	2.2%	4.9%
5 instr back	0.3%	6.6%
6 instr back	0.6%	1.7%
7 instr back	0.1%	3.2%
8 instr back	0.4%	2.5%
9 instr back	0.6%	2.8%
10 instr back	0.5%	2.2%
11 instr back	0.7%	3.5%
12 instr back	0.2%	1.5%
13 instr back	0.5%	1.2%
14 instr back	0.3%	1.6%
15 and more instr back	72.6%	9.6%

Variable operand length instructions | commercial applications

average lengths of character / decimal strings

instruction	rand_length
Move character strings	32.6 bytes
Move decimal numbers	3.8 bytes
Test | compare | logic	3.5 bytes | 2.0 bytes exec.
Decimal arithmetic	2.5 . . . 4.2 bytes

Frequency of instruction occurrences / 100 HLL instr

instruction	no_of_occurrences	percentage
Move character	16	9%
Move decimal	7	4%
Test | compare | logic	22	12%
Decimal arithmetic	6	3%