SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

DISTRIBUTED INTERACTIVE COMPUTING NOTE 741

PERQ UNIX Implementation Note # 40
Benchmark Results for PERQ and PDP 11/70

Issued by
James Collis

1 December 1982

DISTRIBUTION:

R W Witty
K Robinson
C Prosser
A S Williams
L O Ford
T Watson
E V C Fielding
I D Benest
J C Malone
P J Smith
A J Kinroy
J M Loveluck
C P Wadsworth
J R Collis
P Palmer (ICL Dalkeith)
RL Support/PERQ/Unix Implementation Notes File
CBP PERQ Evaluation and Benchmarking

1. Introduction

This note sumarises the results of running three sets of C programs on :

PDP 11/70
PERQ running Accent Unix
half MB PERQ running Microcode Unix
one MB PERQ running Microcode Unix

Because the PDP 11/70 is a 16 bit machine and the PERQ 32 bit, the C instructions were executed with single and double precision. Results for the half and one MB PERQs were the same and are not duplicated. Full details are available in PERQ Evaluation and Benchmarking file.

2. Summary of results.

Obviously individual results vary, but generally, Microcode Unix executes 32 bit C instructions in comparable time with the PDP , while Accent Unix is slower. With 16 bit instrucions the PDP is of the order of 3 to 4 times faster than both Accent and Microcode Unix. The Kashtan and Zilog benchmarks were again comparable between the PDP and the C Machine, and when they ran, slower on Accent Unix.

3. C Instructions

Times for various instructions in C using single precision on the three machines. All times are in microseconds unless otherwise specified.

Single Precision (16 bits)			
Instruction	AccUnix PERQ	Mi-Unix PERQ	PDP 11/70
FOR loop (1)	195	239	48
REPEAT loop (1)	157	220	48
WHILE loop (1)	197	239	55
x = 0;	2.3	6	2.4
x = a;	3.7	11	2.4
x = a/2*3+4-5;	41	35	14
x = a/a*a+a_a;	52	60	16
ar[j] = ar[j-1];	22	22	6
IF (x<50) ;ELSE ;	13	11	1.6
IF (x<1) ;ELSE ;	13	9	1.4
5 nested proc calls	484	78	110
with value par's	590	120	117
with ref par's	522	99	114
x = a + b;	14	15	4
x = a - b;	14	15	4
x = a * b;	24	24	7
x = a / b;	33	32	10
x = a % b;	33	32	10
x = *p;	5	8	3
*p++;	11	6	4

(1) 10 iterations of each loop.

Times for various instructions in C using double precision on the three machines. All times are in microseconds unless otherwise specified.

Double Precision (32 bits)			
Instruction	AccUnix PERQ	Mi-Unix PERQ	PDP 11/70
FOR loop (1)	232	155	102
REPEAT loop (1)	194	147	122
WHILE loop (1)	222	156	89
x = 0;	1.9	1.4	5.3
x = a;	4.2	2.1	5.4
x = a/2*3+4-5;	36	28	97
x = a/a*a+a_a;	44	36	102
ar[j] = ar[j-1];	40	22	11
IF (x<50) ;ELSE ;	14	7.6	2.3
IF (x<1) ;ELSE ;	12	5	2.1
5 nested proc calls	484	78	110
with value par's	537	103	130
with ref par's	504	102	132
Open & Close a file	>1sec	26ms	7ms
1 empty proc call	72	15	22
x = getpid();	70ms	223us	370us
x = a + b;	9	4	8
x = a - b;	9	4	8
x = a * b;	19	13	46
x = a / b;	28	21	55
x = a % b;	28	21	56
x = *p;	10	4	6
*p++;	11	6	5
x = chararray[k];	24	20	4
x = a[k].ifield;	23	14	9
stat(buf,&statbuf);	208ms	24ms	8ms

(1) 10 iterations of each loop.

4. Kashtan Programs

Of the following programs only kt8 ran on Accent Unix and it took 8 seconds. Results for the PDP and C Machine are presented below the description of what the programs do.

kt5 Writes 2 MBytes to disk.
kt6 Randomly positions within a disk file, writes half KByte on to disk and repeats 4000 times.
kt7 Two processes signal each other 20,000 times.
kt8 One process signalling itself 1,000 times.
kt9 Writes 5 MBytes to itself through a pipe.
kt10 Transmits 5 MBytes between two processes through a pipe.
kt11 Two processes each transmitting 5 MBytes to each other.

All times in minutes and seconds.

	PDP 11/70			Microcode Unix		
Name	Real	User	Sys	Real	User	Sys
kt5	1:42	0.1	22.1	45	0.1	27.8
kt6	30	0.7	10.4	7:53	0.2	3:58
kt7	1:24	1.3	25.9	1:10	2.2	32.4
kt8	5	0.0	2.5	5	0.0	4.1
kt9	1:00	0.4	47.7	32	0.3	30.7
kt10	1:06	0.3	26.2	42	0.7	19.2
kt11	2:15	0.7	53.9	1:24	0.6	50.8

5. Zilog Programs

What each program does:

Sieve	20 iterations of sieve on 0 - 8191.
Disk_test	a) 1000 sequential writes. b) 1000 sequential reads. c) 1000 random reads.
Proc_call	1,000,000 function calls to dummy(a,b,c) where dummy returns (a + b + c).
System_call	10,000 calls to "getpid".
Piper	2,000 pipe writes of 512 bytes. Total of 1,024,000 bytes.

Result times in seconds.

	sieve	disk test			proc	sys	piper
		a	b	c			
PDP 11/70	3.7	32.2	17.5	37.4	36.1	3.1	16.2
Acc Unix	27.0				148.4	712	398
Microcode	11.5	42.8	36.3	59.7	41.9	2.3	7.7