Assignment 1

Computer Architecture

Nidhi Hegde (180472)

Part A:

Implementation:

The analysis routine *MyPredicatedAnalysis()* takes in the pointer to the counter of the respective type of instruction and increments it along with the total number of instructions. It is called by the instrumentation routine for all the instructions with true predicates. For loads and stores, the access size (in terms of 32-bit chunks) is calculated and passed as an additional argument to the analysis routine.

The following 8 tables contain the dynamic counts and percentages of the following seventeen types of instructions for each benchmark application. The percentages have been reported with a precision of two decimal places.

400.perlbench diffmail.pl

Operations	Dynamic Counts	Percentages
Loads	356547222	22.82
Stores	205816021	13.17
NOPs	960783	0.06
Direct calls	12746655	0.82
Indirect calls	2834598	0.18
Returns	15581254	1
Unconditional branches	30534343	1.95
Conditional branches	130017908	8.32
Logical operations	100154687	6.41
Rotate and shift	4294543	0.27
Flag operations	862308	0.06
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	934932	0.06
The rest	700910991	44.87

401.bzip2 input.source

Operations	Dynamic Counts	Percentages
Loads	452705779	26.88
Stores	231174375	13.73
NOPs	36514	0
Direct calls	791570	0.05
Indirect calls	13	0
Returns	791579	0.05
Unconditional branches	21299292	1.26
Conditional branches	129922983	7.72
Logical operations	71000978	4.22
Rotate and shift	61832255	3.67
Flag operations	6130	0
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	0	0
The rest	714314602	42.42

403.gcc cp-decl.i

Operations	Dynamic Counts	Percentages
Loads	137772726	9.2
Stores	359693980	24.02
NOPs	190367	0.01
Direct calls	4574309	0.31
Indirect calls	502312	0.03
Returns	5076621	0.34
Unconditional branches	4852742	0.32
Conditional branches	133369840	8.91
Logical operations	131967344	8.81
Rotate and shift	2355383	0.16
Flag operations	184964	0.01
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	5	0
The rest	716857708	47.87

429.mcf

Operations	Dynamic Counts	Percentages
Loads	415213334	27.22
Stores	110039596	7.21
NOPs	1477640	0.1
Direct calls	12556562	0.82
Indirect calls	0	0
Returns	12556562	0.82
Unconditional branches	8314430	0.55
Conditional branches	178242843	11.69
Logical operations	75119472	4.93
Rotate and shift	3516420	0.23
Flag operations	0	0
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	0	0
The rest	708216072	46.43

450.soplex ref.mps

Operations	Dynamic Counts	Percentages
Loads	546414462	33.17
Stores	100980923	6.13
NOPs	11471	0
Direct calls	3199443	0.19
Indirect calls	135	0
Returns	3199578	0.19
Unconditional branches	13095077	0.79
Conditional branches	103143033	6.26
Logical operations	13892531	0.84
Rotate and shift	10353390	0.63
Flag operations	23165267	1.41
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	306669175	18.62
The rest	523270901	31.76

456.hmmer nph3.hmm

Operations	Dynamic Counts	Percentages
Loads	547670251	33.74
Stores	75698809	4.66
NOPs	34317	0
Direct calls	144578	0.01
Indirect calls	937	0
Returns	145515	0.01
Unconditional branches	205820	0.01
Conditional branches	144361446	8.89
Logical operations	1158482	0.07
Rotate and shift	294094	0.02
Flag operations	5648	0
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	40212	0
The rest	853607989	52.58

471.omnetpp

Operations	Dynamic Counts	Percentages
Loads	371211745	23.19
Stores	229780193	14.35
NOPs	804063	0.05
Direct calls	21323365	1.33
Indirect calls	3687030	0.23
Returns	25010401	1.56
Unconditional branches	22176705	1.39
Conditional branches	117347436	7.33
Logical operations	60023082	3.75
Rotate and shift	7164442	0.45
Flag operations	20147034	1.26
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	96893910	6.05
The rest	625422533	39.06

483.xalancbmk

Operations	Dynamic Counts	Percentages
Loads	364139297	23.82
Stores	165186080	10.8
NOPs	23397370	1.53
Direct calls	12660394	0.83
Indirect calls	8911388	0.58
Returns	21571182	1.41
Unconditional branches	8263153	0.54
Conditional branches	182174866	11.92
Logical operations	36399581	2.38
Rotate and shift	5387629	0.35
Flag operations	1595830	0.1
Vector instructions	0	0
Conditional moves	0	0
MMX and SSE instructions	0	0
System calls	0	0
Floating-point	6965917	0.46
The rest	692218806	45.28

Part B:

The CPI of all the benchmarks have been reported in the table below:

Benchmarks	CPI
perl	18.64
bzip	20.9
gcc	17.28
mcf	17.87
soplex	20.26
omnetpp	19.39
hmmer	19.82
xalanc	17.96

Part C:

Implementation:

A map is maintained whose keys represent the indices of the respective memory chunks accessed by the application. Each chunk is assumed to be 32 bytes in size.

We can maintain 2 maps, one corresponding to the instruction footprint and another corresponding to the data footprint. After the application completes executing, we count the number of keys present in both the maps and get the number of unique 32-byte instruction/data chunks accessed. The instruction and data memory footprints of all the application benchmarks are represented in the table below:

Benchmarks	Instruction Footprint Size	Data Footprint Size
perl	90656 bytes	1001600 bytes
bzip	24288 bytes	81190208 bytes
gcc	95200 bytes	36429472 bytes
mcf	2080 bytes	373539424 bytes
soplex	37760 bytes	182238912 bytes
omnetpp	28832 bytes	17798560 bytes
hmmer	14784 bytes	2730976 bytes
xalanc	73696 bytes	30706720 bytes

Part D:

Implementation:

For each of the parts, a hash map is maintained whose keys comprise the metric over which the distribution needs to be inferred. For e.g. the keys for D1 correspond to the instruction length. In this case, each key stores the number of instructions with a given length.

D1: Distribution of Instruction Lengths

Instruction Length	Instruction Count							
Benchmarks	perl	bzip	gcc	mcf	soplex	omnetpp	hmmer	xalanc
1	117334791	38611338	129926495	80626126	77139726	154549640	25078176	144696992
2	256704594	219201564	592160413	485394906	441479435	308497243	302861046	325051661
3	274935238	436966482	124478533	315526918	399075154	382141092	296150039	445729702
4	52880872	75326806	116198060	50531705	16506340	34343198	270389216	30713531
5	78522538	22047104	11032981	22076082	3277723	45111088	24861122	23198666
6	185632072	141356784	15426698	5249475	40937900	48834339	68360587	22848623
7	33989856	51341681	10718416	40594789	17645061	26522980	416475	7386124
8	28	15085632	58405	0	3938662	0	11883340	250579
9	0	0	0	0	0	0	0	59138
10	12	62610	0	0	0	421	0	64985

D2: Distribution of number of operands

Number of Operands	Instruction Count							
Benchmarks	perl	bzip	gcc	mcf	soplex	omnetpp	hmmer	xalanc
0	960783	36514	190367	1477640	11471	804063	34887	23397370
1	1075368	6147	4598900	0	23	225590	3491	713009
2	520714010	597647733	349126281	484783902	415191691	518309549	566127313	418619986
3	355203874	382870265	403895143	457346943	395064633	281807989	432938351	418864375
4	103189558	2693765	33820217	43834954	186496387	172690299	712697	98462137
5	15582918	14191369	204113810	12556562	3235796	25023998	159485	23439831
6	3273490	2554208	4255283	0	0	1138513	23777	16503293

D3: Distribution of number of Register Read operands

Operand Count	Instruction Count							
Benchmarks	perl	bzip	gcc	mcf	soplex	omnetpp	hmmer	xalanc
0	10100056	5189069	2139333	3759790	23375660	29000002	610918	30642400
1	260259400	183351949	168878213	148509225	216025036	197502870	75619958	239212109
2	537519906	533608432	472115050	677533255	577510211	540435761	562398832	454066325
3	179158597	215105747	62393306	170197731	139955577	219778544	281425833	249615607
4	7094923	46790797	90680713	0	43097185	8459924	79905776	1163131
5	2593629	13399799	199538103	0	36332	3684387	14907	8797136
6	3273490	2554208	4255283	0	0	1138513	23777	16503293

D4: Distribution of number of Register Write operands

Operand Count	Instruction Count							
Benchmarks	perl	bzip	gcc	mcf	soplex	omnetpp	hmmer	xalanc
0	135836989	132331600	125850144	70905978	69474624	165353154	75149545	111975309
1	685101042	712782003	418002638	770283388	764701845	665529695	755279878	688396968
2	175784920	152332173	451748619	158772123	165823532	167602896	169546801	183122751
3	2391540	2554225	4398600	38512	0	375743	23777	16504973
4	885510	0	0	0	0	1138513	0	0

D5: Distribution of number of Memory operands

Operand Count	Instruction Count							
Benchmarks	**perl**	**bzip**	**gcc**	**mcf**	**soplex**	**omnetpp**	**hmmer**	**xalanc**
0	453562267	399852743	518860232	487181827	530050637	449250269	376808511	501438618
1	530722845	516406192	464676023	500383418	439771271	542983206	623024513	470206932
2	15547890	83736981	16395340	12434756	30178093	7766526	166014	27900566

D6: Distribution of number of Memory Read operands

Operand Count	Instruction Count							
Benchmarks	**perl**	**bzip**	**gcc**	**mcf**	**soplex**	**omnetpp**	**hmmer**	**xalanc**
0	644248434	547290137	862158872	584786667	582944643	661698775	452336670	637636411
1	354699081	452705779	137772723	415213334	417055358	337162713	547662368	361909705
2	885487	0	0	0	0	1138513	0	0

D7: Distribution of number of Memory Write operands

Operand Count	Instruction Count							
Benchmarks	**perl**	**bzip**	**gcc**	**mcf**	**soplex**	**omnetpp**	**hmmer**	**xalanc**
0	794484432	768821541	640237615	889960405	916927902	780923482	924304865	835447757
1	205348570	231174375	359693980	110039596	83072099	219076519	75694173	164098359

D8: Number of memory bytes touched

Benchmarks	Maximum	Average
perl	8	3.81
bzip	8	4.03
gcc	8	4.05
mcf	8	4.1
soplex	8	5.51
omnetpp	8	4.24
hmmer	8	4
xalanc	8	4.19

D9: Maximum and Minimum Value of Immediate Field

Benchmarks	Maximum	Minimum
perl	2147483647	-2147483648
bzip	1431655766	-858993459
gcc	1073741823	-2147483587
mcf	1374389535	-100000000
soplex	2147483647	-1074790400
omnetpp	2147483647	-2092037281
hmmer	2147483647	-987654321
xalanc	2147483647	-1431655765

D10: Maximum and minimum Value of Displacement Field

Benchmarks	Maximum	Minimum
perl	135918104	-1408
bzip	135000192	-4848
gcc	138634432	-1744
mcf	134957120	-76
soplex	135856732	-344
omnetpp	136090116	-104
hmmer	135294312	-580
xalanc	139655605	-1392

Overview of Main analysis Routines:

MyAnalysisRoutine():

Called for every instruction(predicated+non-predicated). Maintains and updates the values associated with part D1, D2, D3, D4 and D9, and also updates the instruction memory footprint map associated with part C.

MyPredicatedAnalysisRoutine():

Called for every instruction with predicated bit true. Maintains and updates all data structures for Part A, D5, D6 and D7.

RecordMemRead/Write():

Predicated analysis routine for loads and stores (Part A). Takes an extra argument as the number of 32-bit data chunks accessed, and increments the counter for load, store and total instructions appropriately. Called for every memory operand using a predicated insert call.

MemOperations():

Deals with maintaining all information related to part D8 and D10. Called for every memory operand in an instruction using a predicated insert call.

DataFootprint():

Takes in the memory operand address and size, and updates the keys of the data footprint map accordingly. Called for each memory operand using a Predicated call.

InsCount():

Keeps track of number of instructions. Called for all instructions.

MemInsCount():

Keeps track of number of instructions with memory operands, whose predicated bit is true. Useful while calculating the average number of memory bytes touched in part D8.