



OS|ACA

Open Source | Architecture Code Analyzer

Jan Laukemann

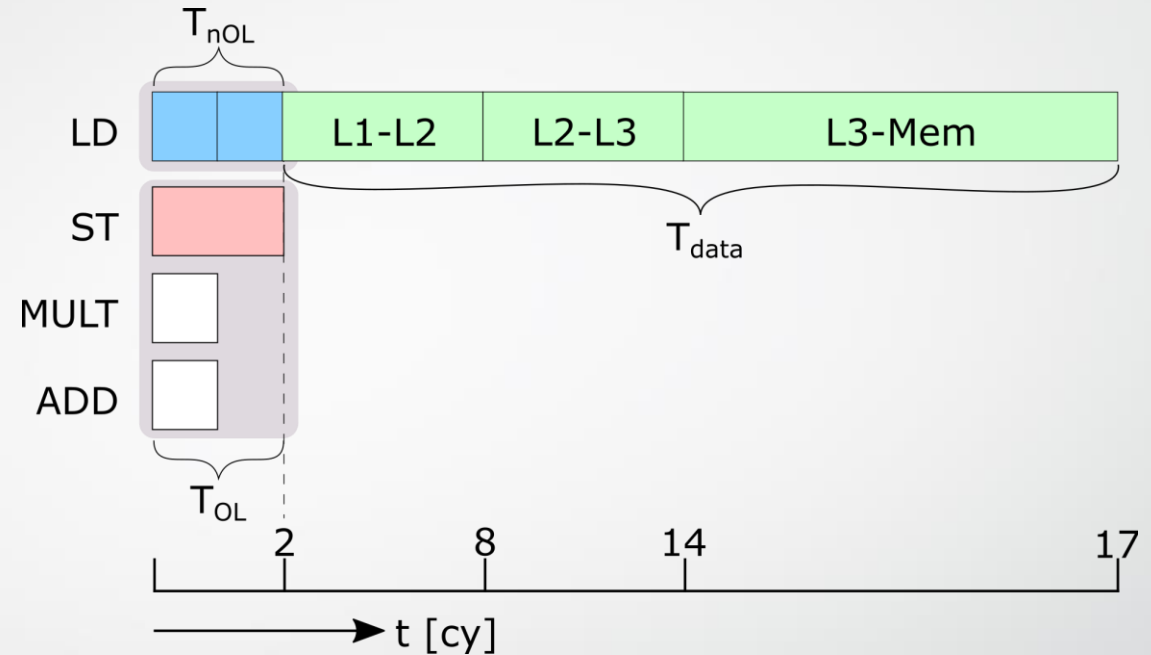
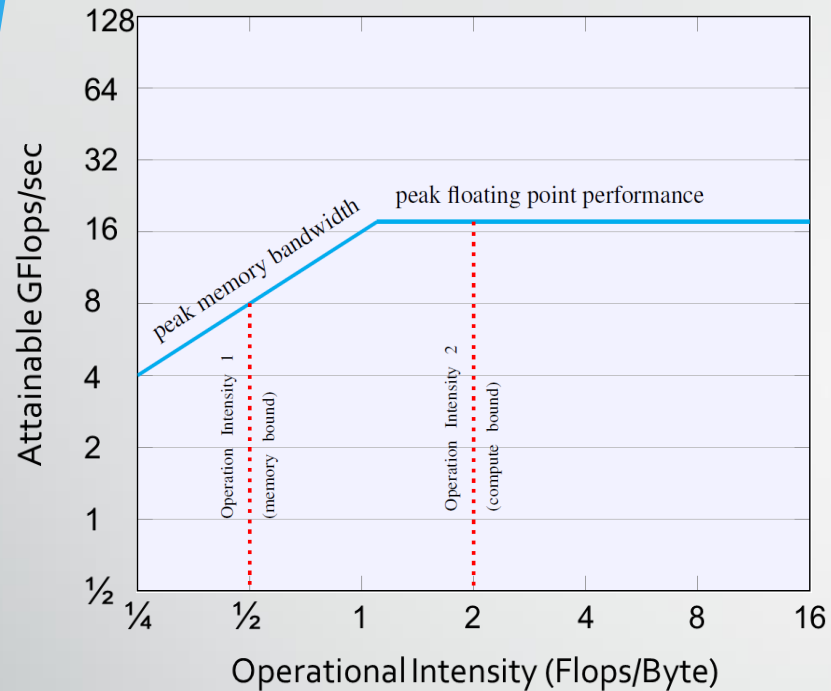


Design and Implementation of a
Framework for Predicting Instruction Throughput

Agenda

- Motivation
- IACA as role model
- OSACA: design and functionality
- Examples
- Conclusion / Future Challenges

Motivation



- Performance model for optimization
→ concentrate on loop body
- Analysis of source code depending on microarchitecture
- **Incore prediction** [cycles] for more complex performance models necessary

IACA

Intel Architecture Code Analyzer

- Analyzes maximum throughput assuming optimal execution conditions (All memory accesses hit L1 cache, no page faults)
- Supports Intel 64 code for architectures from Sandybridge to Skylake X

// C or C++ usage of IACA

#include "iacaMarks.h"

```
while(cond){  
    IACA_START  
    // Loop body  
    // ...  
}  
IACA_END
```

; ASM usage of IACA

movl \$111, %ebx

.byte 100, 103, 144 ; Start marker

.innermostlooplabel:

; Loop body

; ...

jb .innermostlooplabel ; conditional branch

movl \$222, %ebx

.byte 100, 103, 144 ; End marker

Need to compile
into object code
for IACA

IACA

Intel(R) Architecture Code Analyzer Version - 2.3 build:246dfea (Thu, 6 Jul 2017 13:38:05 +0300)

Analyzed File - helloIaca

Binary Format - 64Bit

Architecture - IVB

Analysis Type - Throughput

Throughput Analysis Report

Block Throughput: 1.90 Cycles

Throughput Bottleneck: FrontEnd

Port Binding In Cycles Per Iteration:

Port	0 - DV	1	2 - D	3 - D	4	5
Cycles	1.0 0.0	1.0	1.0 0.5	1.0 0.5	1.0	1.9

N - port number or number of cycles resource conflict caused delay, DV - Divider pipe (on port 0)

D - Data fetch pipe (on ports 2 and 3), CP - on a critical path

F - Macro Fusion with the previous instruction occurred

* - instruction micro-ops not bound to a port

^ - Micro Fusion happened

- ESP Tracking sync uop was issued

@ - SSE instruction followed an AVX256/AVX512 instruction, dozens of cycles penalty is expected

X - instruction not supported, was not accounted in Analysis

Num Of Uops	0 - DV	1	2 - D	3 - D	4	5	
2		1.0	0.5 0.5	0.5 0.5			vaddss xmm3, xmm2, dword ptr [rsp+rax*4-0x4]
2			0.5	0.5	1.0		vmovss dword ptr [rsp+rax*4], xmm3
1	0.9					0.1 CP	inc rax
1						1.0 CP	cmp rax, 0x3e8
0F							j1 0xfffffffffffffd8
1	0.1	0.1				0.9 CP	mov edi, 0x401d58

Total Num Of Uops: 7

Why OSACA?

- Open Source
- Future development path of IACA unclear
- Based on benchmarks of individual instructions
- Easy setting of markers in high level source code
- Provides whole toolchain of Instruction fetching, benchmarking and throughput analysis
- Perspective of
 - Analyses on non Intel architectures
 - Latency analyses (Critical Path, inter loop dependencies)

OSACA

- Analyzes average throughput assuming optimal execution conditions
 - all memory accesses hit L1 cache, no page faults, steady-state, all instructions in cache
- Currently supports Intel architectures from Sandybridge to Skylake X

```
//usage of OSACA marker
```

```
//STARTLOOP  
while(cond){  
    // Loop body  
    // ...  
}
```

```
; ASM usage of IACA byte markers for OSACA
```

```
movl    $111, %ebx
```

```
.byte   100, 103, 144 ; Start marker
```

```
.innermostlooplabel:
```

```
; Loop body
```

```
; ...
```

```
jb .innermostlooplabel ; conditional branch
```

```
movl    $222, %ebx
```

```
.byte   100, 103, 144 ; End marker
```

No need for
compiling
assembly!

OSACA functionality

- Benchmarking of application codes and extraction of relevant instructions
- Producing benchmarks for not yet known instructions, adding data to data file
- Throughput analysis of ASM code snippet
- Optional: Insert IACA markers for better loop identification


```
graph LR; UI[User input] --> OSACA[OSACA marker]; UI --> IACA[IACA byte markers];
```

The diagram illustrates the flow of data from 'User input' to two processing components. A red box labeled 'User input' contains a C code snippet. Two arrows originate from the bottom of this box: one points to a grey box labeled 'OSACA marker', and the other points to a grey box labeled 'IACA byte markers'.

```
// Multiply i with immediate
// and add to array
int t = 0.19;
int main(void){
    int i = 0;
    //STARTLOOP
    while(i < 1000){
        arr[i] = arr[i-1] + i * t;
        i += 1;
    }
}
```

The diagram illustrates the structure of a packet. It consists of a top grey bar labeled "OSACA marker" and a bottom grey bar labeled "IACA byte markers". The space between these two bars is white and contains a large, faint watermark that reads "OSACA".

	Memory	GPR	XMM
xor		%eax, %eax	
lea	0x1(%rax, %rax, 1),	%edx	
vcvtsi2ss	%edx,	%xmm2, %xmm2	
vmulss	%xmm2,	%xmm0, %xmm3	
lea	0x2(%rax, %rax, 1),	%ecx	
vaddss	%xmm3,	%xmm1, %xmm4	
vxorps	%xmm1,	%xmm1, %xmm1	
vcvtsi2ss	%ecx,	%xmm1, %xmm1	
vmulss	%xmm1,	%xmm0, %xmm5	
vmovss	%xmm4,	0x4(%rsp, %rax, 8)	

OSACA

CSV
data files

Throughput analysis

Port Binding in Cycles Per Iteration:

Port	0	1	2	3	4	5
Cycles	4.0	5.0	3.0	3.0	2.0	2.0

Ports Pressure in cycles						
0	1	2	3	4	5	
0.50	1.00	1.00	1.00		0.50	lea 0x1(%rax,%rax,1),%edx
1.00						vcvttsi2ss %edx,%xmm2,%xmm2
						vmulss %xmm2,%xmm0,%xmm3
	1.00	1.00	1.00			lea 0x2(%rax,%rax,1),%ecx
0.33	0.33					vaddss %xmm3,%xmm1,%xmm4
0.50	1.00				0.33	vxorps %xmm1,%xmm1,%xmm1
1.00					0.50	vcvttsi2ss %ecx,%xmm1,%xmm1
						vmulss %xmm1,%xmm0,%xmm5
	1.00	0.50	0.50	1.00		vmovss %xmm4,0x4(%rsp,%rax,8)
						vaddss %xmm5,%xmm4,%xmm1
		0.50	0.50	1.00		vmovss %xmm1,0x8(%rsp,%rax,8)
0.33	0.33				0.33	inc %rax
0.33	0.33				0.33	cmp \$0xf3,%rax
						jb 400bc2 <main+0x62>

Total number of estimated throughput: 5.0

Port	0	1	2	3	4	5
Cycles	4.0	5.0	3.0	3.0	2.0	2.0

Ports Pressure in cycles						
0	1	2	3	4	5	
0.50 1.00	1.00	1.00	1.00		0.50	
	1.00	1.00	1.00			
0.33 0.50 1.00	0.33 1.00				0.33 0.50	
	1.00	0.50	0.50	1.00		
		0.50	0.50	1.00		
0.33 0.33	0.33 0.33				0.33 0.33	

```
Total number of estimated throughput: 5.0
```

```
lea     0x1(%rax,%rax,1),%edx
vcvtsi2ss %edx,%xmm2,%xmm2
vmulss  %xmm2,%xmm0,%xmm3
lea     0x2(%rax,%rax,1),%ecx
vaddss  %xmm3,%xmm1,%xmm4
vxorps  %xmm1,%xmm1,%xmm1
vcvtsi2ss %ecx,%xmm1,%xmm1
vmulss  %xmm1,%xmm0,%xmm5
vmovss  %xmm4,0x4(%rsp,%rax,8)
vaddss  %xmm5,%xmm4,%xmm1
vmovss  %xmm1,0x8(%rsp,%rax,8)
inc     %rax
cmp     $0x1f3,%rax
jb      400bc2<main+0x62>
```

```

1  # Benchmark file
2
3  #define INSTRS vccat2as
4
5  #define NINSTRS 32
6
7  #define N 1024
8
9  #define INSTRS rdb
10
11  #define INSTRS rdb
12
13  #define INSTRS rdb
14
15  #define INSTRS rdb
16
17  #define INSTRS rdb
18
19  #define INSTRS rdb
20
21  #define INSTRS rdb
22
23  #define INSTRS rdb
24
25  #define INSTRS rdb
26
27  #define INSTRS rdb
28
29  #define INSTRS rdb
30
31  #define INSTRS rdb
32
33  #define INSTRS rdb
34
35  #define INSTRS rdb
36
37  #define INSTRS rdb
38
39  #define INSTRS rdb
40
41  #define INSTRS rdb
42
43  #define INSTRS rdb
44
45  #define INSTRS rdb
46
47  #define INSTRS rdb
48
49  #define INSTRS rdb
50
51  #define INSTRS rdb
52
53  #define INSTRS rdb
54
55  #define INSTRS rdb
56
57  #define INSTRS rdb
58
59  #define INSTRS rdb
60
61  #define INSTRS rdb
62
63  #define INSTRS rdb
64
65  #define INSTRS rdb
66
67  #define INSTRS rdb
68
69  #define INSTRS rdb
70
71  #define INSTRS rdb
72
73  #define INSTRS rdb
74
75  #define INSTRS rdb
76
77  #define INSTRS rdb
78
79  #define INSTRS rdb
80
81  #define INSTRS rdb
82
83  #define INSTRS rdb
84
85  #define INSTRS rdb
86
87  #define INSTRS rdb
88
89  #define INSTRS rdb
90
91  #define INSTRS rdb
92
93  #define INSTRS rdb
94
95  #define INSTRS rdb
96
97  #define INSTRS rdb
98
99  #define INSTRS rdb
100
101  #define INSTRS rdb
102
103  #define INSTRS rdb
104
105  #define INSTRS rdb
106
107  #define INSTRS rdb
108
109  #define INSTRS rdb
110
111  #define INSTRS rdb
112
113  #define INSTRS rdb
114
115  #define INSTRS rdb
116
117  #define INSTRS rdb
118
119  #define INSTRS rdb
120
121  #define INSTRS rdb
122
123  #define INSTRS rdb
124
125  #define INSTRS rdb
126
127  #define INSTRS rdb
128
129  #define INSTRS rdb
130
131  #define INSTRS rdb
132
133  #define INSTRS rdb
134
135  #define INSTRS rdb
136
137  #define INSTRS rdb
138
139  #define INSTRS rdb
140
141  #define INSTRS rdb
142
143  #define INSTRS rdb
144
145  #define INSTRS rdb
146
147  #define INSTRS rdb
148
149  #define INSTRS rdb
150
151  #define INSTRS rdb
152
153  #define INSTRS rdb
154
155  #define INSTRS rdb
156
157  #define INSTRS rdb
158
159  #define INSTRS rdb
160
161  #define INSTRS rdb
162
163  #define INSTRS rdb
164
165  #define INSTRS rdb
166
167  #define INSTRS rdb
168
169  #define INSTRS rdb
170
171  #define INSTRS rdb
172
173  #define INSTRS rdb
174
175  #define INSTRS rdb
176
177  #define INSTRS rdb
178
179  #define INSTRS rdb
180
181  #define INSTRS rdb
182
183  #define INSTRS rdb
184
185  #define INSTRS rdb
186
187  #define INSTRS rdb
188
189  #define INSTRS rdb
190
191  #define INSTRS rdb
192
193  #define INSTRS rdb
194
195  #define INSTRS rdb
196
197  #define INSTRS rdb
198
199  #define INSTRS rdb
200
201  #define INSTRS rdb
202
203  #define INSTRS rdb
204
205  #define INSTRS rdb
206
207  #define INSTRS rdb
208
209  #define INSTRS rdb
210
211  #define INSTRS rdb
212
213  #define INSTRS rdb
214
215  #define INSTRS rdb
216
217  #define INSTRS rdb
218
219  #define INSTRS rdb
220
221  #define INSTRS rdb
222
223  #define INSTRS rdb
224
225  #define INSTRS rdb
226
227  #define INSTRS rdb
228
229  #define INSTRS rdb
230
231  #define INSTRS rdb
232
233  #define INSTRS rdb
234
235  #define INSTRS rdb
236
237  #define INSTRS rdb
238
239  #define INSTRS rdb
240
241  #define INSTRS rdb
242
243  #define INSTRS rdb
244
245  #define INSTRS rdb
246
247  #define INSTRS rdb
248
249  #define INSTRS rdb
250
251  #define INSTRS rdb
252
253  #define INSTRS rdb
254
255  #define INSTRS rdb
256
257  #define INSTRS rdb
258
259  #define INSTRS rdb
260
261  #define INSTRS rdb
262
263  #define INSTRS rdb
264
265  #define INSTRS rdb
266
267  #define INSTRS rdb
268
269  #define INSTRS rdb
270
271  #define INSTRS rdb
272
273  #define INSTRS rdb
274
275  #define INSTRS rdb
276
277  #define INSTRS rdb
278
279  #define INSTRS rdb
280
281  #define INSTRS rdb
282
283  #define INSTRS rdb
284
285  #define INSTRS rdb
286
287  #define INSTRS rdb
288
289  #define INSTRS rdb
290
291  #define INSTRS rdb
292
293  #define INSTRS rdb
294
295  #define INSTRS rdb
296
297  #define INSTRS rdb
298
299  #define INSTRS rdb
300
301  #define INSTRS rdb
302
303  #define INSTRS rdb
304
305  #define INSTRS rdb
306
307  #define INSTRS rdb
308
309  #define INSTRS rdb
310
311  #define INSTRS rdb
312
313  #define INSTRS rdb
314
315  #define INSTRS rdb
316
317  #define INSTRS rdb
318
319  #define INSTRS rdb
320
321  #define INSTRS rdb
322
323  #define INSTRS rdb
324
325  #define INSTRS rdb
326
327  #define INSTRS rdb
328
329  #define INSTRS rdb
330
331  #define INSTRS rdb
332
333  #define INSTRS rdb
334
335  #define INSTRS rdb
336
337  #define INSTRS rdb
338
339  #define INSTRS rdb
340
341  #define INSTRS rdb
342
343  #define INSTRS rdb
344
345  #define INSTRS rdb
346
347  #define INSTRS rdb
348
349  #define INSTRS rdb
350
351  #define INSTRS rdb
352
353  #define INSTRS rdb
354
355  #define INSTRS rdb
356
357  #define INSTRS rdb
358
359  #define INSTRS rdb
360
361  #define INSTRS rdb
362
363  #define INSTRS rdb
364
365  #define INSTRS rdb
366
367  #define INSTRS rdb
368
369  #define INSTRS rdb
370
371  #define INSTRS rdb
372
373  #define INSTRS rdb
374
375  #define INSTRS rdb
376
377  #define INSTRS rdb
378
379  #define INSTRS rdb
380
381  #define INSTRS rdb
382
383  #define INSTRS rdb
384
385  #define INSTRS rdb
386
387  #define INSTRS rdb
388
389  #define INSTRS rdb
390
391  #define INSTRS rdb
392
393  #define INSTRS rdb
394
395  #define INSTRS rdb
396
397  #define INSTRS rdb
398
399  #define INSTRS rdb
400
401  #define INSTRS rdb
402
403  #define INSTRS rdb
404
405  #define INSTRS rdb
406
407  #define INSTRS rdb
408
409  #define INSTRS rdb
410
411  #define INSTRS rdb
412
413  #define INSTRS rdb
414
415  #define INSTRS rdb
416
417  #define INSTRS rdb
418
419  #define INSTRS rdb
420
421  #define INSTRS rdb
422
423  #define INSTRS rdb
424
425  #define INSTRS rdb
426
427  #define INSTRS rdb
428
429  #define INSTRS rdb
430
431  #define INSTRS rdb
432
433  #define INSTRS rdb
434
435  #define INSTRS rdb
436
437  #define INSTRS rdb
438
439  #define INSTRS rdb
440
441  #define INSTRS rdb
442
443  #define INSTRS rdb
444
445  #define INSTRS rdb
446
447  #define INSTRS rdb
448
449  #define INSTRS rdb
450
451  #define INSTRS rdb
452
453  #define INSTRS rdb
454
455  #define INSTRS rdb
456
457  #define INSTRS rdb
458
459  #define INSTRS rdb
460
461  #define INSTRS rdb
462
463  #define INSTRS rdb
464
465  #define INSTRS rdb
466
467  #define INSTRS rdb
468
469  #define INSTRS rdb
470
471  #define INSTRS rdb
472
473  #define INSTRS rdb
474
475  #define INSTRS rdb
476
477  #define INSTRS rdb
478
479  #define INSTRS rdb
480
481  #define INSTRS rdb
482
483  #define INSTRS rdb
484
485  #define INSTRS rdb
486
487  #define INSTRS rdb
488
489  #define INSTRS rdb
490
491  #define INSTRS rdb
492
493  #define INSTRS rdb
494
495  #define INSTRS rdb
496
497  #define INSTRS rdb
498
499  #define INSTRS rdb
500
501  #define INSTRS rdb
502
503  #define INSTRS rdb
504
505  #define INSTRS rdb
506
507  #define INSTRS rdb
508
509  #define INSTRS rdb
510
511  #define INSTRS rdb
512
513  #define INSTRS rdb
514
515  #define INSTRS rdb
516
517  #define INSTRS rdb
518
519  #define INSTRS rdb
520
521  #define INSTRS rdb
522
523  #define INSTRS rdb
524
525  #define INSTRS rdb
526
527  #define INSTRS rdb
528
529  #define INSTRS rdb
530
531  #define INSTRS rdb
532
533  #define INSTRS rdb
534
535  #define INSTRS rdb
536
537  #define INSTRS rdb
538
539  #define INSTRS rdb
540
541  #define INSTRS rdb
542
543  #define INSTRS rdb
544
545  #define INSTRS rdb
546
547  #define INSTRS rdb
548
549  #define INSTRS rdb
550
551  #define INSTRS rdb
552
553  #define INSTRS rdb
554
555  #define INSTRS rdb
556
557  #define INSTRS rdb
558
559  #define INSTRS rdb

```

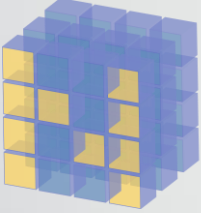
ibench



ibench output

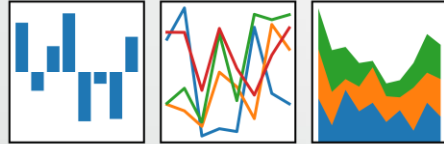
vcvtsi2ss-xmm_xmm_r32	1.0
vcvtsi2ss-xmm_xmm_r32-TP	3.0
vmulss-xmm xmm xmm	1.0

OSACA dependencies

-  NumPy

- pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Kerncraft (<https://github.com/RRZE-HPC/kerncraft>)
- ibench (<https://github.com/hofm/ibench>)

Example 1: STREAM Scale

Initial situation

```
double a[n], b[N];  
double s;  
  
//STARTLOOP  
for(int i = 0; i < N; ++i)  
    a[i] = s * b[i];
```

\$ osaca --arch IVB PATH/TO/FILE

Throughput Analysis Report

X - No information for this instruction in database
* - Instruction micro-ops not bound to a port

Port Binding in Cycles Per Iteration:

Port	0	1	2	3	4	5
Cycles	2.00	1.00	5.0	5.0	2.0	1.00

Ports Pressure in cycles						
0	1	2	3	4	5	
		0.50	0.50	1.00		movl \$0x0,-0x24(%rbp)
						jmp 10b <scale+0x10b>
		0.50	0.50			mov -0x48(%rbp),%rax
		0.50	0.50			mov -0x24(%rbp),%edx
0.33	0.33				0.33	movslq %edx,%rdx
		0.50	0.50			vmovsd (%rax,%rdx,8),...
1.00		0.50	0.50			vmulsd -0x50(%rbp),...
		0.50	0.50			mov -0x38(%rbp),%rax
		0.50	0.50			mov -0x24(%rbp),%edx
0.33	0.33				0.33	movslq %edx,%rdx
		0.50	0.50	1.00		vmovsd %xmm0,...
						X addl \$0x1,-0x24(%rbp)
		0.50	0.50			mov -0x24(%rbp),%eax
0.33	0.33	0.50	0.50		0.33	cmp -0x54(%rbp),%eax
						jle e4 <scale+0xe4>

Total number of estimated throughput: 5.0

addl-mem_imd
(TP & LT)

#define INSTR vcvt2si2ss
#define INSTR 32
#define N edi
#define i r8d
.intel_syntax noprefix
.globl minst
.data
minst:
.long INSTR
.align 32
PI:
.long 0xf01b866e, 0x400921f9
.text
.globl latency
.type latency, @function
.align 32
loop:
inc i
INSTR xmm3, xmm0, eax
INSTR xmm4, xmm1, ebx
INSTR xmm5, xmm2, ecx
INSTR xmm6, xmm0, eax
INSTR xmm7, xmm1, ebx
INSTR xmm8, xmm2, ecx

```
$ ./ibench ./AVX 2.2
```

```
Using frequency 2.20GHz.
```

```
add-mem_imd-TP: 1.023 (clock cycles) [DEBUG - result: 1.000000]
```

```
add-mem_imd:    6.050 (clock cycles) [DEBUG - result: 1.000000]
```



```
$ osaca --include-ibench -arch IVB PATH/TO/IBENCH_OUTPUT
```

```
Ibench output FILE successfully in database included.  
2 values were added.
```

\$ osaca --arch IVB PATH/TO/FILE

```
Throughput Analysis Report
-----
X - No information for this instruction in database
* - Instruction micro-ops not bound to a port

Port Binding in Cycles Per Iteration:
-----
| Port | 0 | 1 | 2 | 3 | 4 | 5 |
-----
| Cycles | 2.33 | 1.33 | 6.0 | 6.0 | 3.0 | 1.33 |
-----

Ports Pressure in cycles
| 0 | 1 | 2 | 3 | 4 | 5 |
-----
| | | 0.50 | 0.50 | 1.00 | | movl $0x0,-0x24(%rbp)
| | | | | | | jmp 10b <scale+0x10b>
| | | 0.50 | 0.50 | | | mov -0x48(%rbp),%rax
| | | 0.50 | 0.50 | | | mov -0x24(%rbp),%edx
| 0.33 | 0.33 | | | | 0.33 | movslq %edx,%rdx
| | | 0.50 | 0.50 | | | vmovsd (%rax,%rdx,8),...
| 1.00 | | 0.50 | 0.50 | | | vmulsd -0x50(%rbp),...
| | | 0.50 | 0.50 | | | mov -0x38(%rbp),%rax
| | | 0.50 | 0.50 | | | mov -0x24(%rbp),%edx
| 0.33 | 0.33 | | | | 0.33 | movslq %edx,%rdx
| | | 0.50 | 0.50 | 1.00 | | vmovsd %xmm0,...
| 0.33 | 0.33 | 1.00 | 1.00 | 1.00 | 0.33 | addl $0x1,-0x24(%rbp)
| | | 0.50 | 0.50 | | | mov -0x24(%rbp),%eax
| 0.33 | 0.33 | 0.50 | 0.50 | | 0.33 | cmp -0x54(%rbp),%eax
| | | | | | | jl e4 <scale+0xe4>

Total number of estimated throughput: 6.0
```

Example 2: 2D-5pt stencil

Initial situation

```
for(j=1; j<M-1; ++j){  
    #pragma vector aligned  
    //STARTLOOP  
    for(int i=1; i<N-1; ++i){  
        b[j][i] = (a[j][i-1] + a[j][i+1]  
                   + a[j-1][i] + a[j+1][i]) * s;  
    }  
}
```



```
$ osaca --insert-marker PATH/TO/ASM_FILE
```

Blocks found in assembly file:

block	OPs	pck.	AVX	Registers	YMM	XMM	GP	ptr.inc
0 ..B1.8	8	0	0	15 (4)	4 (1)	0 (0)	11 (3)	128
1 ..B1.13	5	0	0	6 (4)	1 (1)	0 (0)	5 (3)	32
2 ..B1.17	12	0	0	28 (13)	1 (1)	11 (6)	16 (6)	None

Choose block to be marked [2]:

Comparison OSACA vs. IACA

Throughput Analysis Report

X - No information for this instruction in database
* - Instruction micro-ops not bound to a port

Port Binding in Cycles Per Iteration:

Port	0	1	2	3	4	5
Cycles	1.67	3.67	2.5	2.5	1.0	0.67

Ports Pressure in cycles

0	1	2	3	4	5
---	---	---	---	---	---

		0.50	0.50			vmovsd	(%r14,%r15,8), %xmm2
	1.00	0.50	0.50			vaddsd	16(%r14,%r15,8),%xmm2,%xmm3
	1.00	0.50	0.50			vaddsd	8(%rax,%r15,8),%xmm3,%xmm4
	1.00	0.50	0.50			vaddsd	8(%rdx,%r15,8),%xmm4,%xmm5
1.00						vmulsd	%xmm5,%xmm1,%xmm6
		0.50	0.50	1.00		vmovsd	%xmm6,8(%r12,%r15,8)
0.33	0.33				0.33	incq	%r15
0.33	0.33				0.33	cmpq	%r13,%r15
						jb	..B1.17

Total number of estimated throughput: 3.67

Throughput Analysis Report

Block Throughput: 3.00 Cycles Throughput Bottleneck: FrontEnd

Port Binding In Cycles Per Iteration:

Port	0	- DV	1	2	- D	3	- D	4	5
Cycles	1.0	0.0	3.0	2.5	2.0	2.5	2.0	1.0	2.0

N - port number or number of cycles resource conflict caused delay, DV - Divider pipe (on port 0)
D - Data fetch pipe (on ports 2 and 3), CP - on a critical path
F - Macro Fusion with the previous instruction occurred
* - instruction micro-ops not bound to a port
^ - Micro Fusion happened
- ESP Tracking sync uop was issued
@ - SSE instruction followed an AVX256/AVX512 instruction, dozens of cycles penalty is expected
X - instruction not supported, was not accounted in Analysis

Num Of	Ports pressure in cycles					
Uops	0 - DV	1	2 - D	3 - D	4	5

1			1.0	1.0			vmovsd	xmm2, qword ptr [r14+r15*8]
2		1.0		1.0	1.0		CP	vaddsd xmm3, xmm2, qword ptr ...
2		1.0	1.0	1.0			CP	vaddsd xmm4, xmm3, qword ptr ...
2		1.0		1.0	1.0		CP	vaddsd xmm5, xmm4, qword ptr ...
1	1.0							vmulsd xmm6, xmm1, xmm5
2			0.5	0.5		1.0		vmovsd qword ptr [r12+r15*8+0x8]...
1						1.0		inc r15
1						1.0		cmp r15, r13
0F								jb 0xffffffffffffd4

Total Num Of Uops: 12

Current level of development

- Fetch instructions out of compiled high-level code with OSACA marker
- Fetch instructions out of assembly code with IACA byte markers (no matter if compiled)
- Automated creation of μ -benchmarks for ibench¹
- Template for easy manual creation of μ -benchmarks
- Continuous integration of benchmark results in database
- Throughput analysis with average port pressure for code snippets
- Supports Intel architectures from Sandy Bridge to Skylake

¹ <https://github.com/hofm/ibench>

Future Challenges

- Automatic identification of **processor port binding**
- Identification of **critical path** (for latency analysis)
- Identification of **loop-carried dependencies**
- Different x86 and non-x86 **architectures to support** (ARM, Power, AMD, ...)
- **User defined test values** for benchmarking in ibench (rand, NaN, 0, all same regs, ...)
- Publishing on Python Package Index (pypi)
- Enhance instruction fetching (with or without objdump)

<https://github.com/RRZE-HPC/osaca>

OSACA

