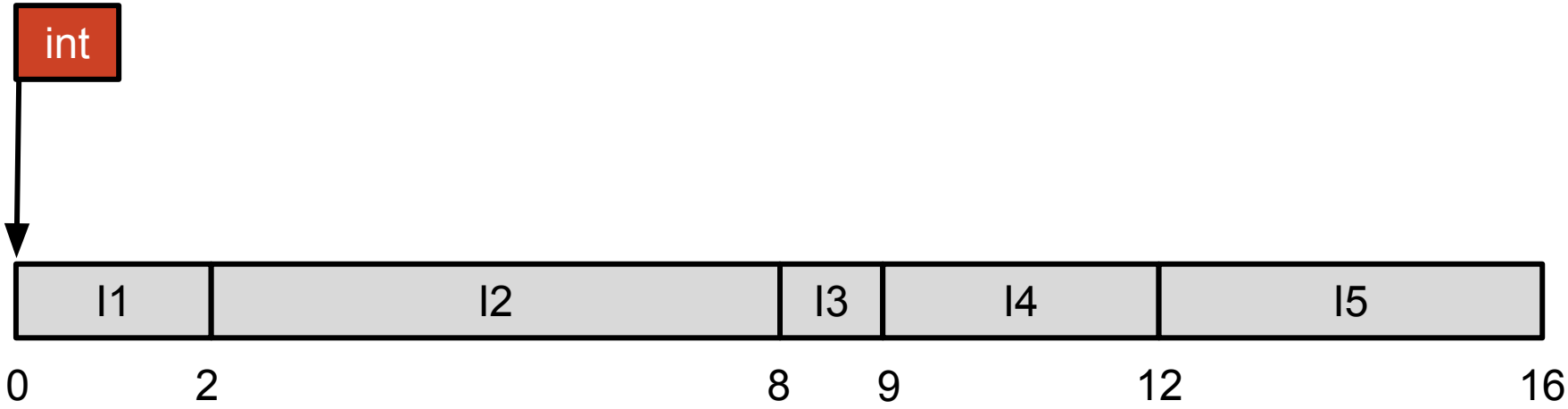
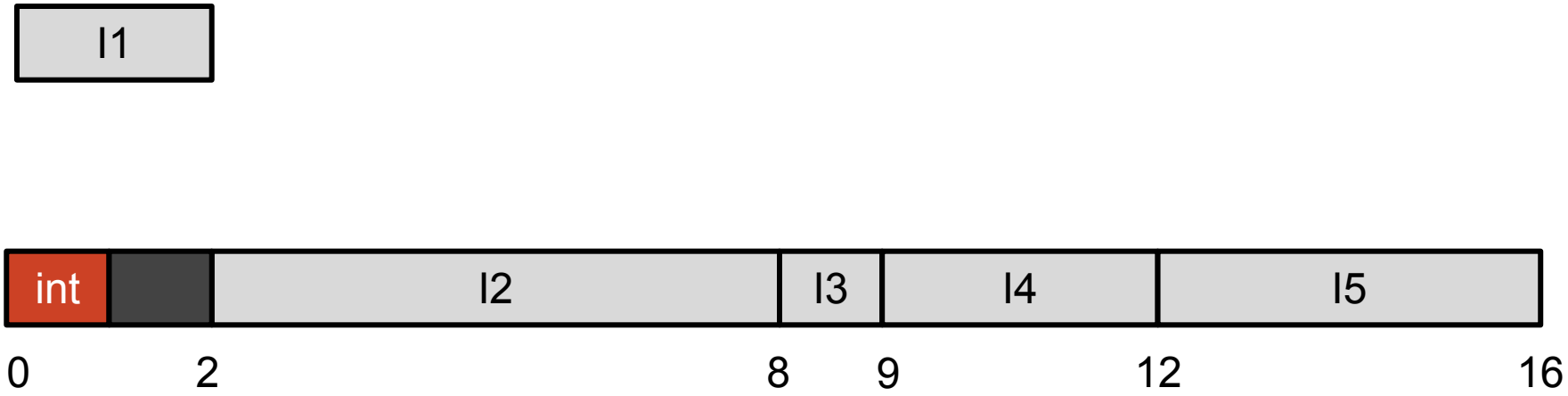
Fast & Flexible Dynamic Tracepoints in x86

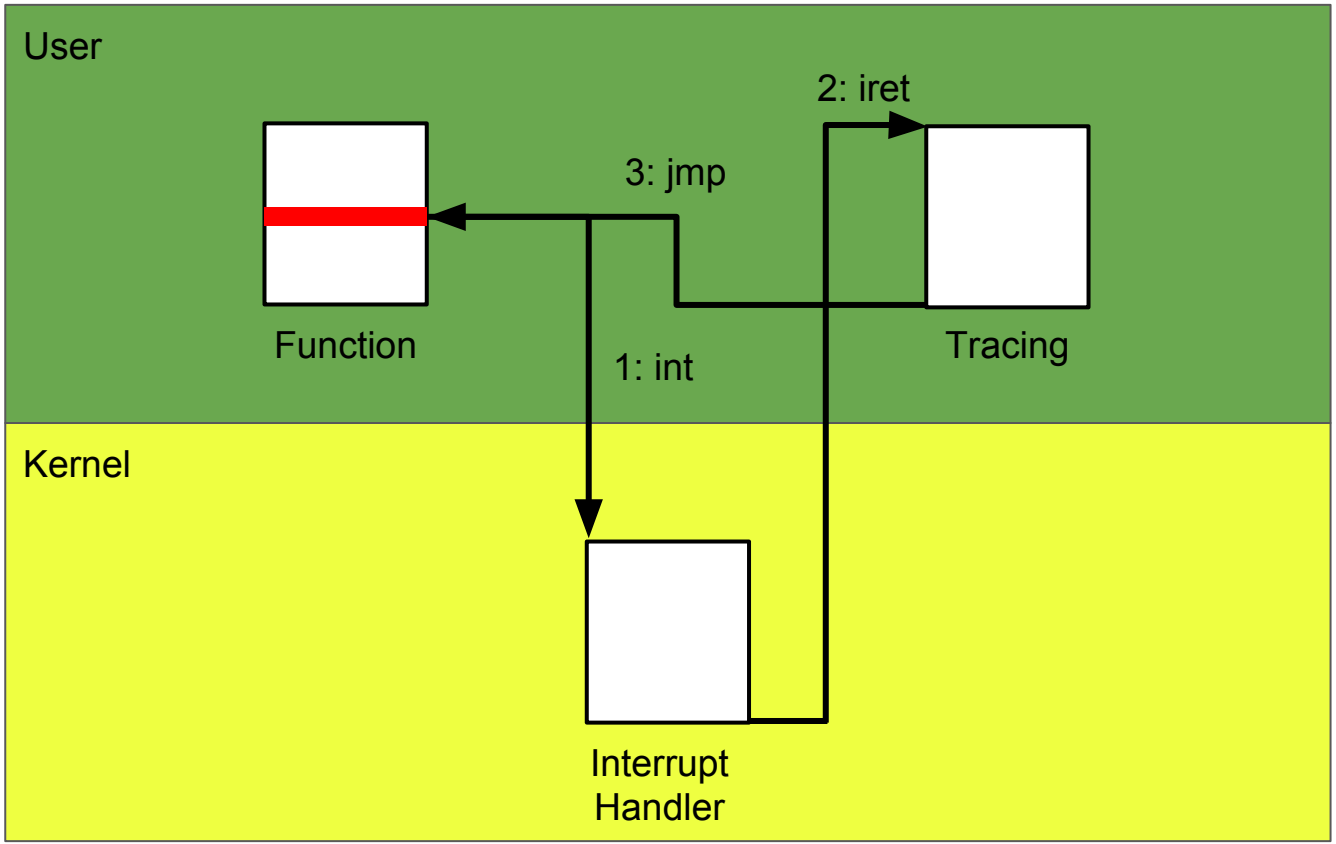
Christian Harper-Cyr
École Polytechnique de Montréal
Laboratoire DORSAL

Plan

Dynamic Tracing
Fast tracepoints
Flexible tracepoints


int
I1
I2
I3
I4
I5
0
2
8
9
12
16

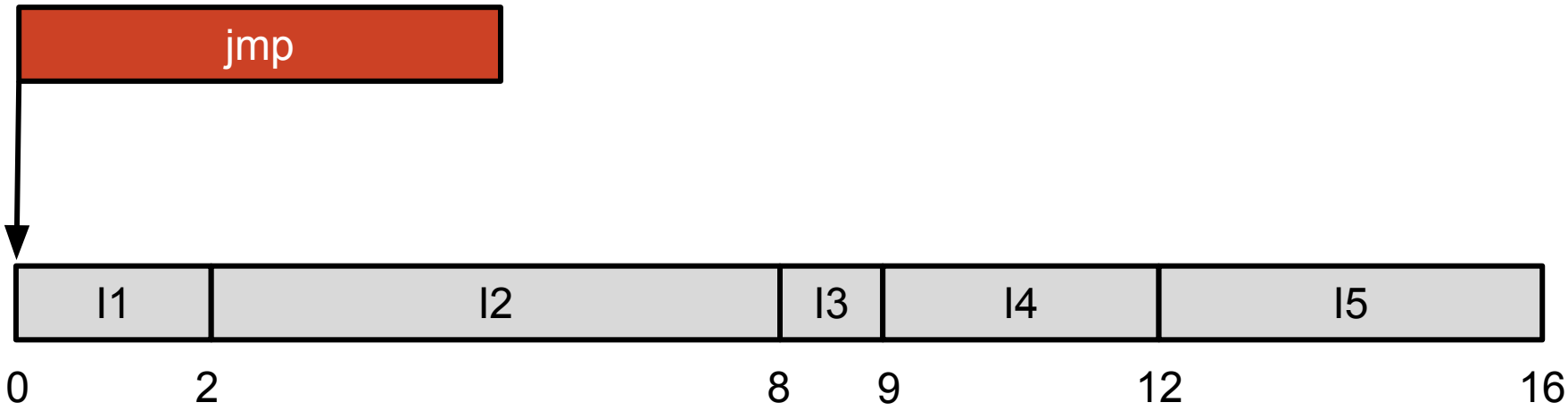

I1
int
I2
I3
I4
I5
0
2
8
9
12
16

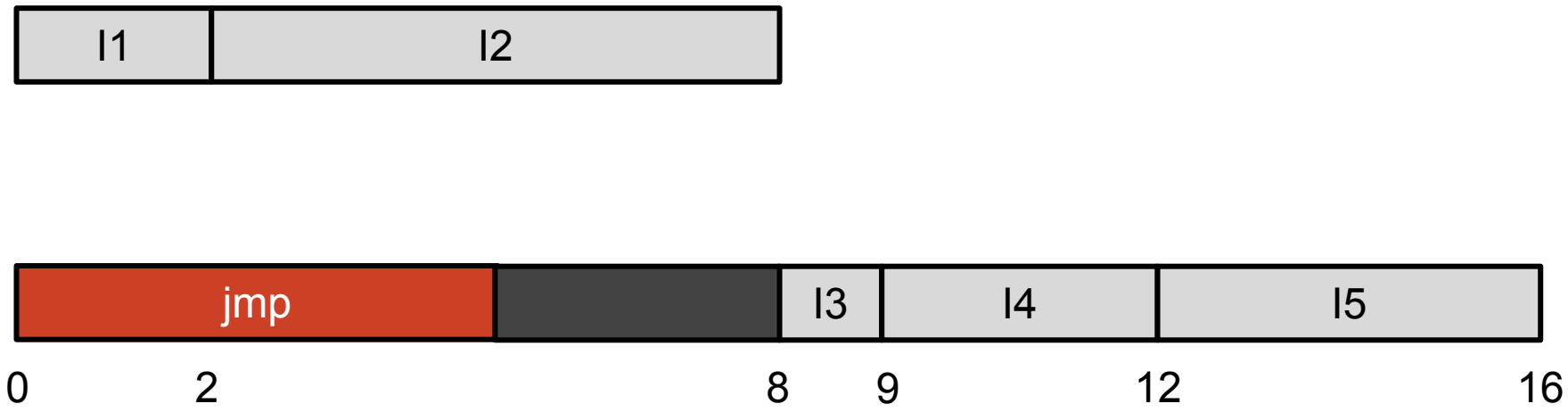


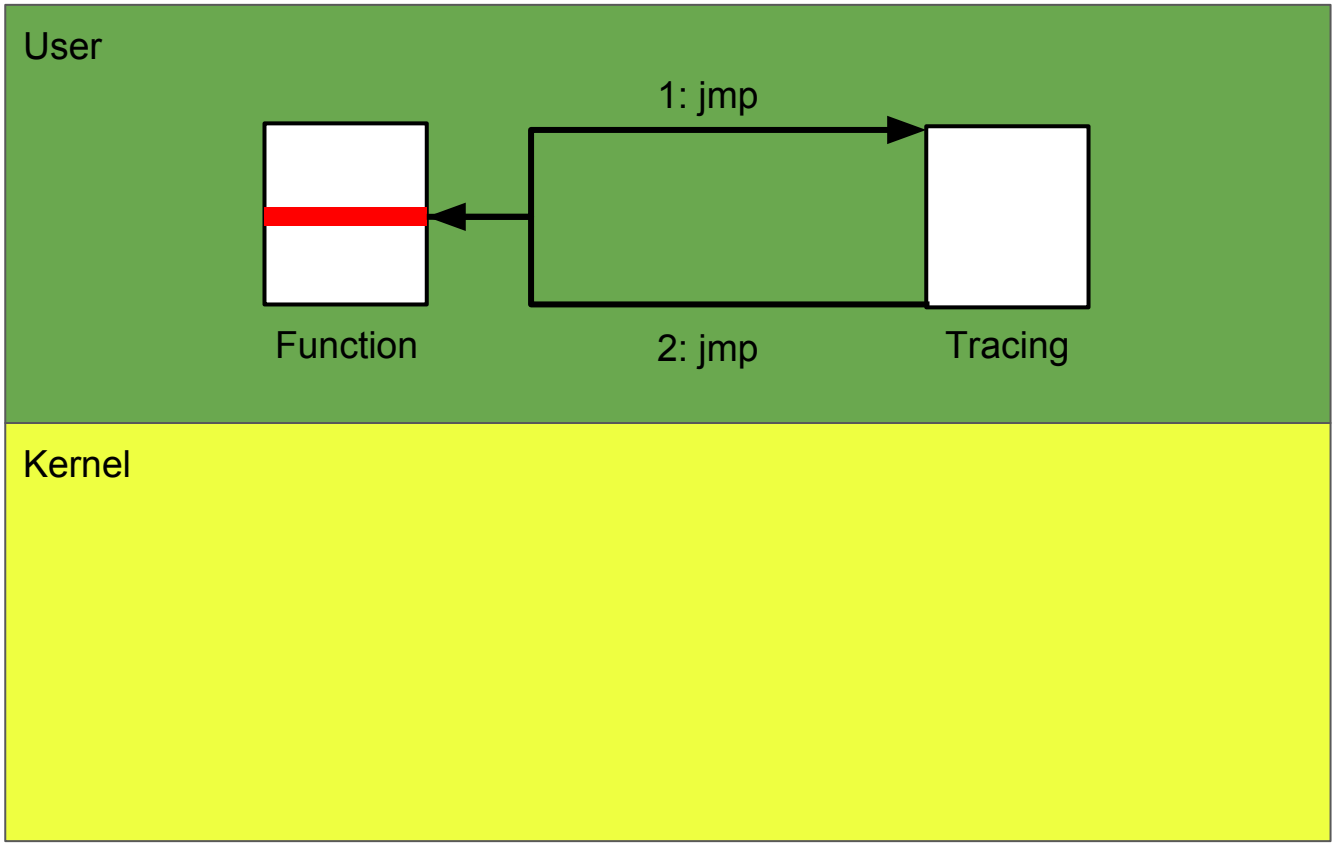
500µs

GDB, x86, ~2GHz [1]

Fast Tracepoints

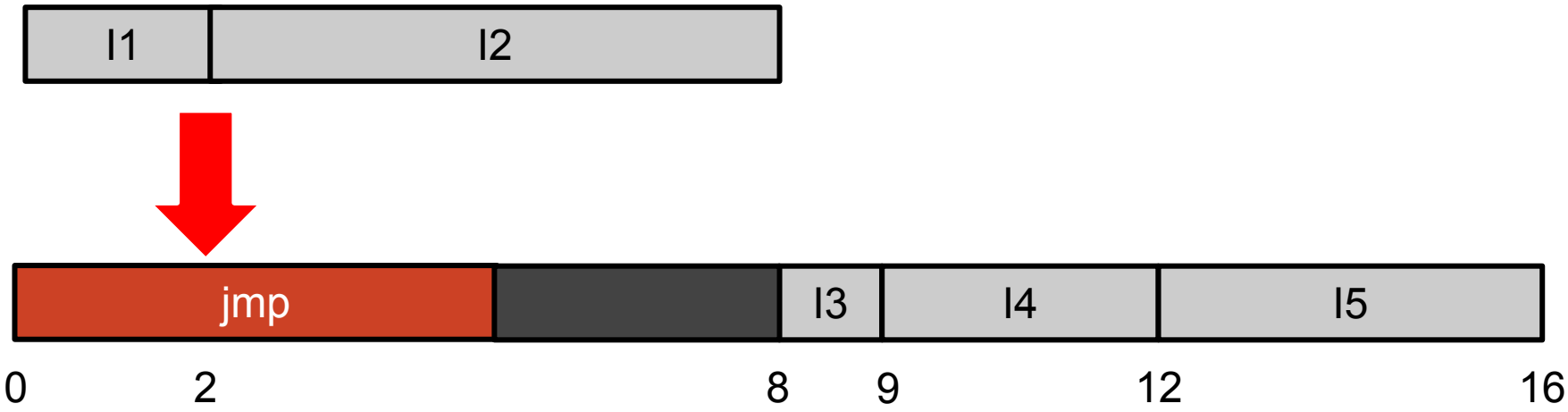

jmp
I1
I2
I3
I4
I5
0
2
8
9
12
16


I1
I2
jmp
I3
I4
I5
0
2
8
9
12
16



300ns

GDB, x86, ~2GHz [1]


I1
I2
jmp
I3
I4
I5
0
2
8
9
12
16

Flexible Tracepoints

```
$ clang -c -emit-llvm -g foo.c -o foo.c.bc
$ llc -filetype=obj -relocation-model=pic -dwarf-add-basic-block-info foo.c.bc -o foo.c.o
$ clang -g foo.c.o -o foo
```

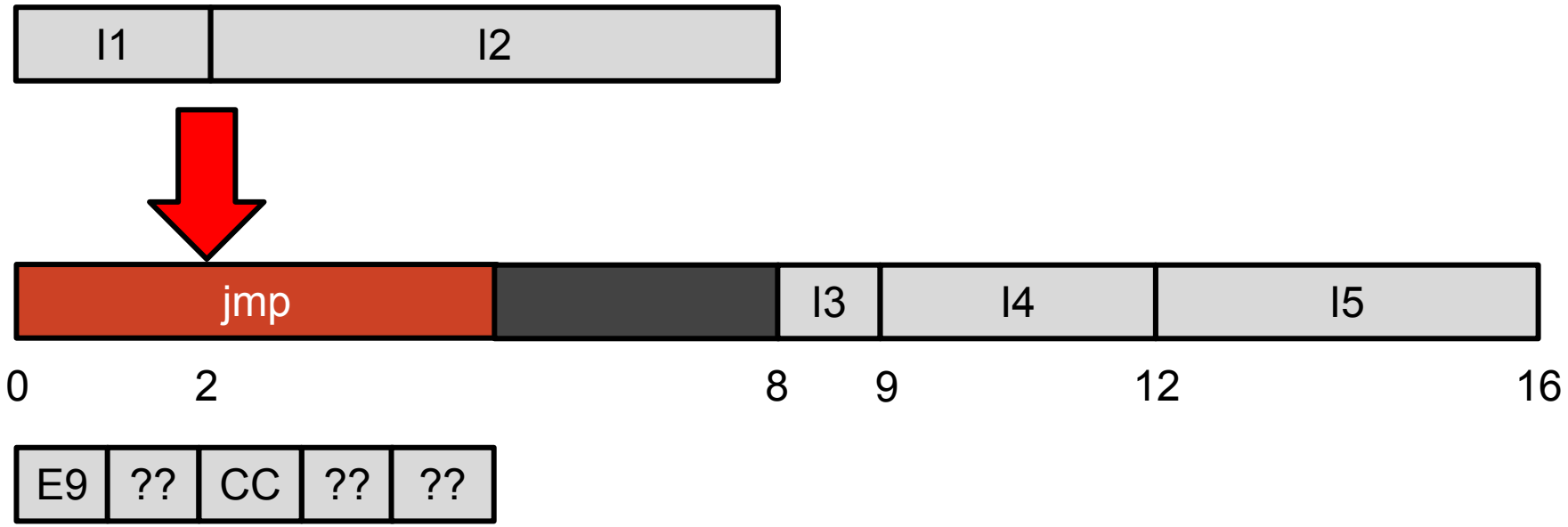
```
0000000000000640 <foo>:
 640:   55                      push   %rbp
 641:   48 89 e5                mov    %rsp,%rbp
 644:   89 7d f8                mov    %edi,-0x8(%rbp)
 647:   89 75 fc                mov    %esi,-0x4(%rbp)
 64a:   8b 45 f8                mov    -0x8(%rbp),%eax
 64d:   03 45 fc                add    -0x4(%rbp),%eax
 650:   5d                      pop    %rbp
 651:   c3                      retq

0000000000000660 <main>:
 660:   55                      push   %rbp
 661:   48 89 e5                mov    %rsp,%rbp
 664:   48 83 ec 10             sub    $0x10,%rsp
 668:   c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
 66f:   c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
 676:   83 7d fc 0a             cmpl   $0xa,-0x4(%rbp)
 67a:   7d 2a                   jge    6a6 <main+0x46>
 67c:   8b 7d fc                mov    -0x4(%rbp),%edi
 67f:   8b 75 fc                mov    -0x4(%rbp),%esi
 682:   0f af 75 fc             imul   -0x4(%rbp),%esi
 686:   e8 b5 ff ff ff          callq  640 <foo>
 68b:   48 8d 3d a2 00 00 00    lea    0xa2(%rip),%rdi        # 734 <_IO_stdin_used+0x4>
 692:   89 c6                   mov    %eax,%esi
 694:   b0 00                   mov    $0x0,%al
 696:   e8 85 fe ff ff          callq  520 <printf@
 69b:   8b 45 fc                mov    -0x4(%rbp),%eax
 69e:   83 c0 01                add    $0x1,%eax
 6a1:   89 45 fc                mov    %eax,-0x4(%rbp)
 6a4:   eb d0                   jmp    676 <main+0x16>
 6a6:   31 c0                   xor    %eax,%eax
 6a8:   48 83 c4 10             add    $0x10,%rsp
 6ac:   5d                      pop    %rbp
 6ad:   c3                      retq
```

```c
#include <stdio.h>

int foo(int a, int b)
{
  return a + b;
}

int main()
{
  for(int i = 0; i < 10; ++i)
  {
      printf("%d\n", foo(i, i*i));
  }
  return 0;
}
```


I1
I2
jmp
I3
I4
I5
0
2
8
9
12
16
E9
??
CC
??
??

Live demo time !

Future Work

32-bit x86
Actual tracing (LTTng)
Full ecosystem for dynamic tracing
Profiling & Benchmarking

Code available at

dyntrace https://github.com/charpercyr/dyntrace
LLVM https://github.com/charpercyr/llvm

Thanks for listening !

1. Stan Shebs (2009). *GDB Tracepoints, Redux*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.739.1074&rep=rep1&type=pdf#page=101