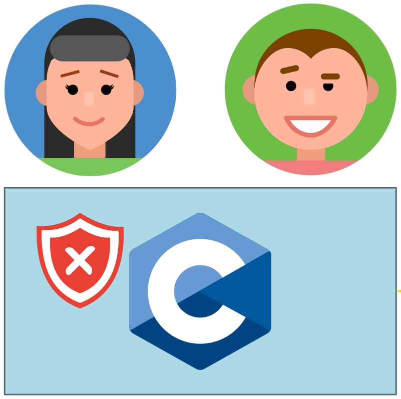
MELTDOWN and SPECTRE are timing-channel attacks

Interesting because they're

- recently discovered
- relatively practical (before mitigations)
- attacks about fundamental strategies for system performance

Like other timing channels, these attacks are probably not something you'll need to worry about in day-to-day programming but they're worth understanding as an example of security as a system-wide, cross-cutting concern

Safety and Isolation

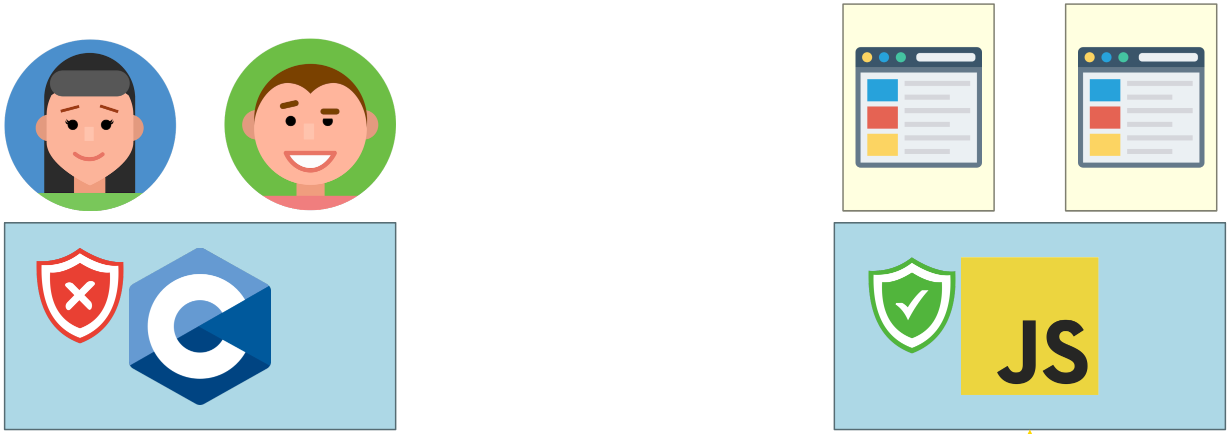


Unsafety of C is a security concern when multiple clients use the same C application

Safety and Isolation

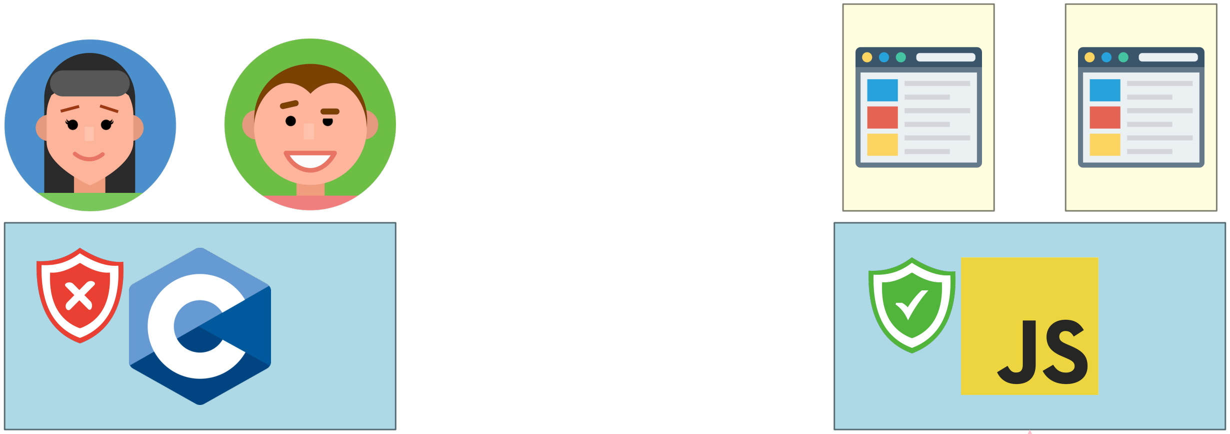
✓ Safety simplifies reasoning about isolation of data

Safety and Isolation



Safety also enables isolation between web pages within a browser

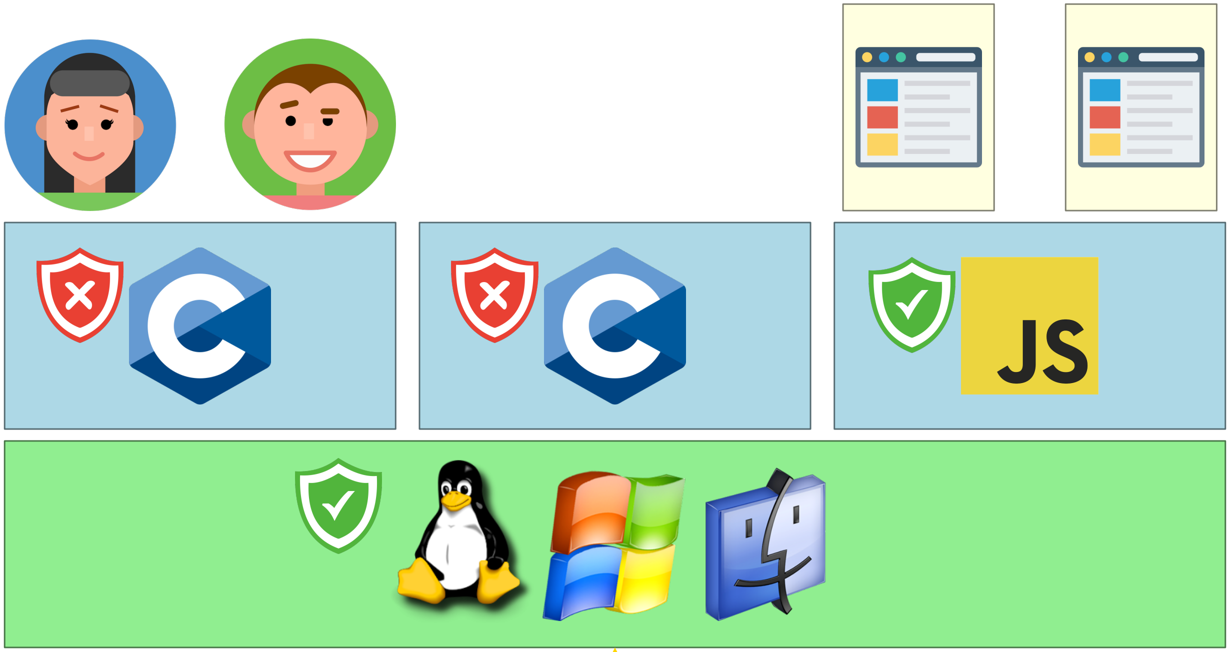
Safety and Isolation



Safety depends on run-time checks:

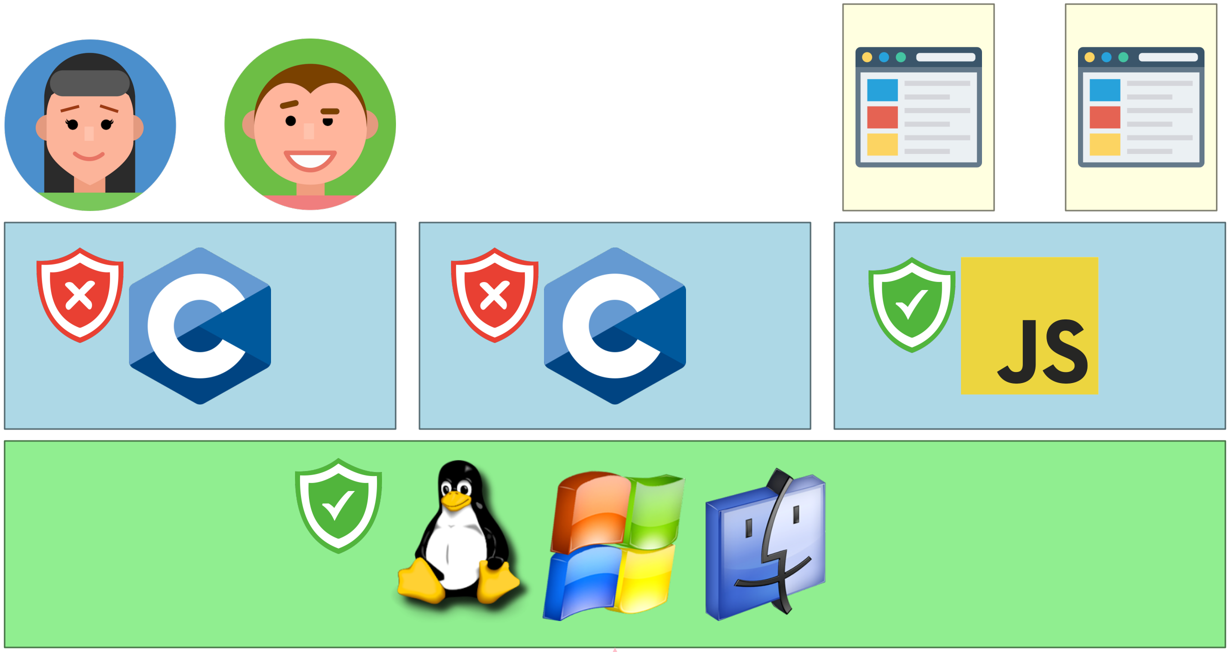
- bounds checks
- tag checks

Safety and Isolation



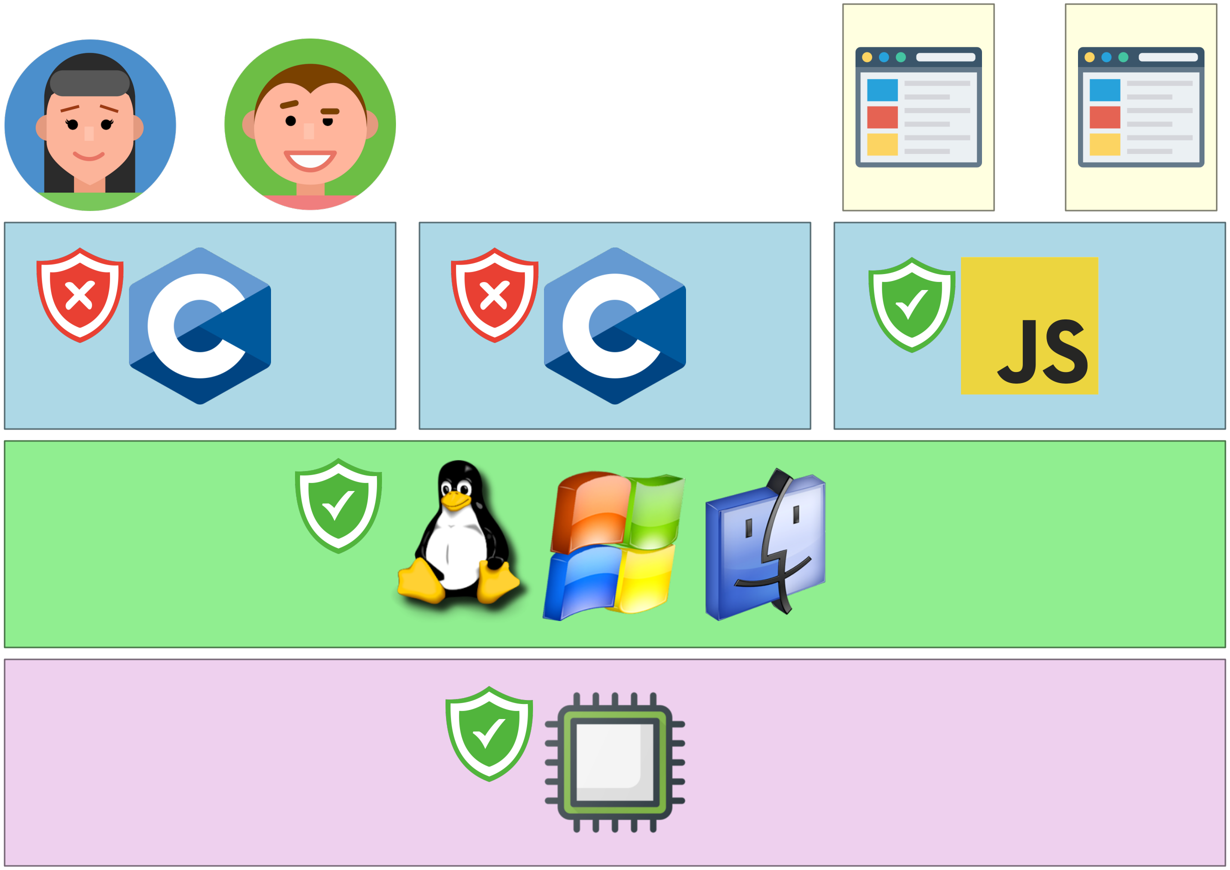
An operating system needs to be safe to isolate applications from each other

Safety and Isolation

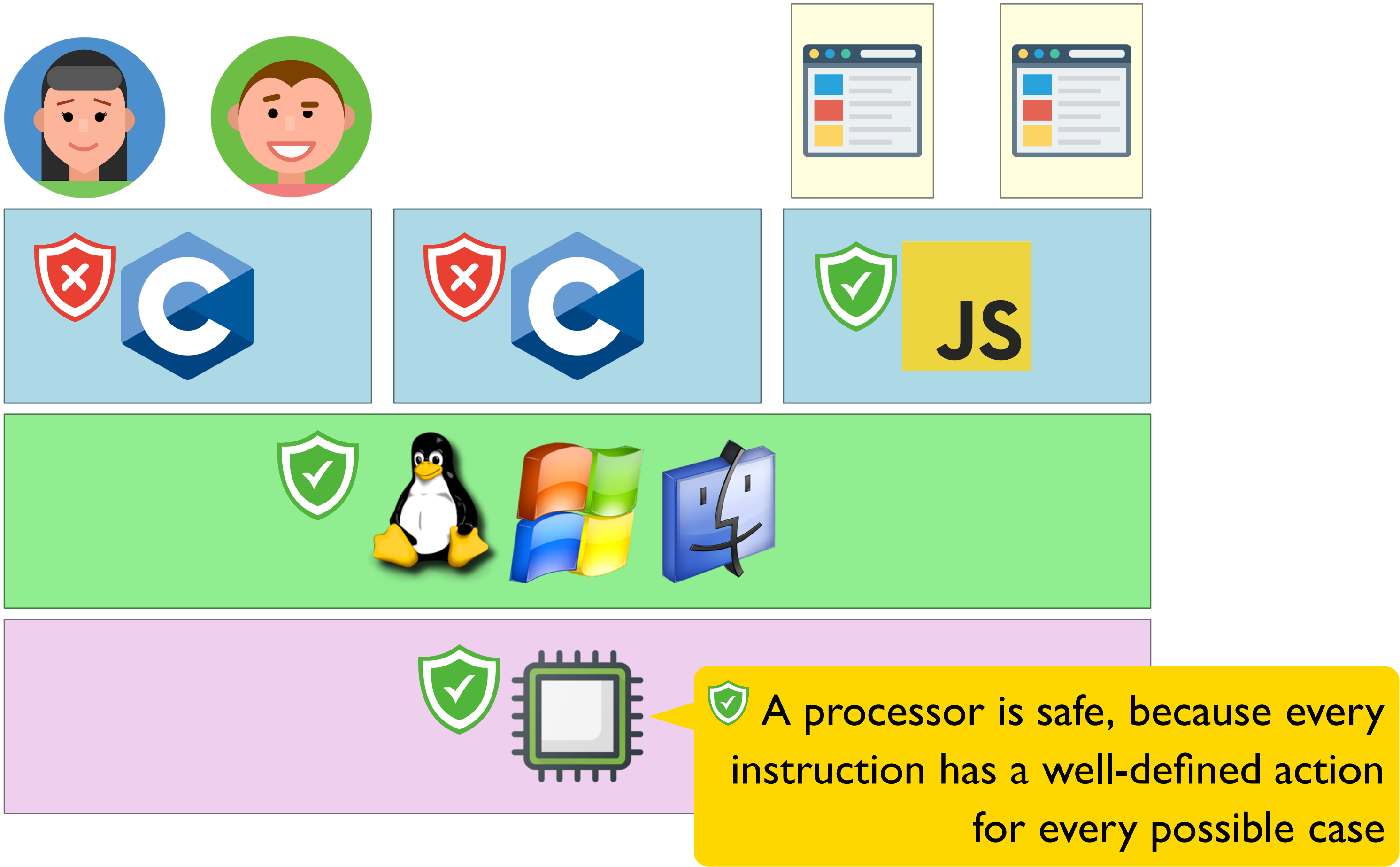


Safety depends on page protection

Safety and Isolation

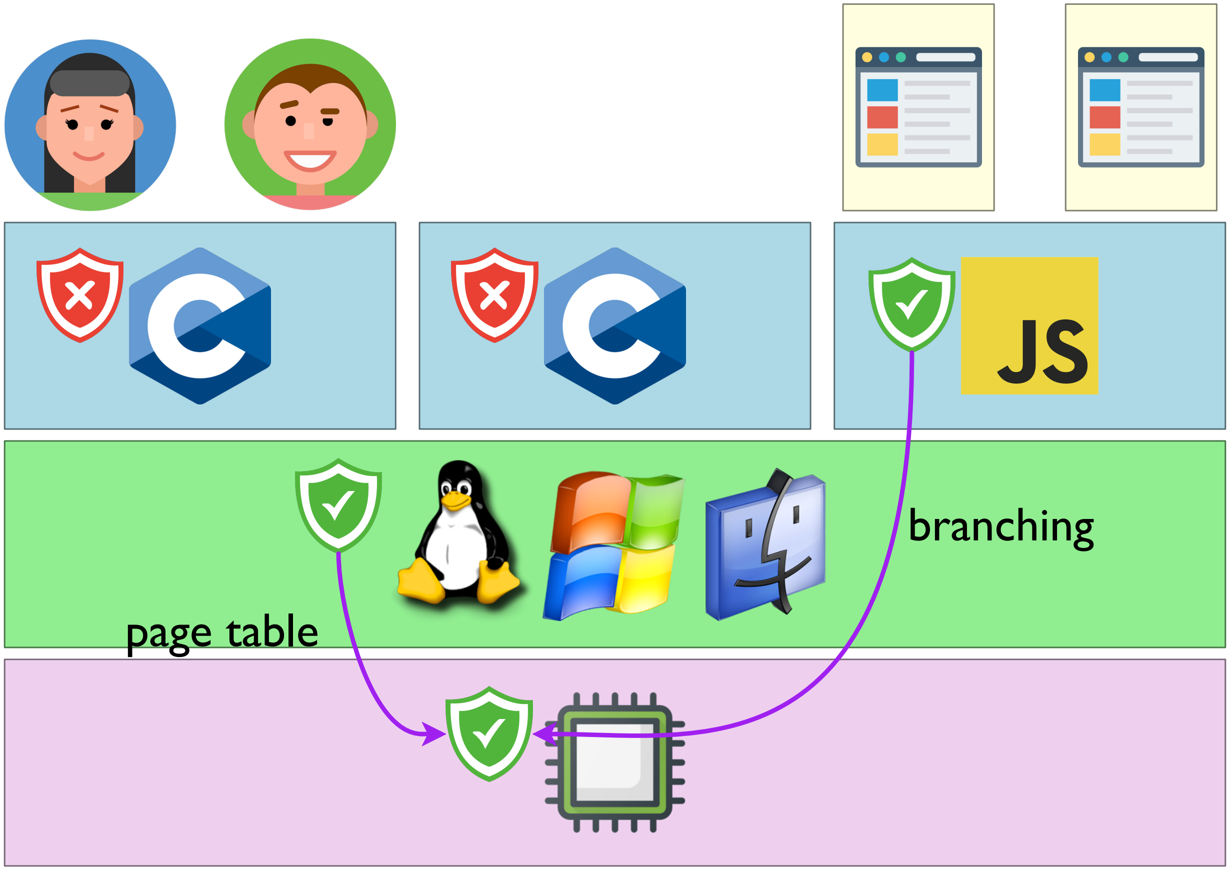


Safety and Isolation

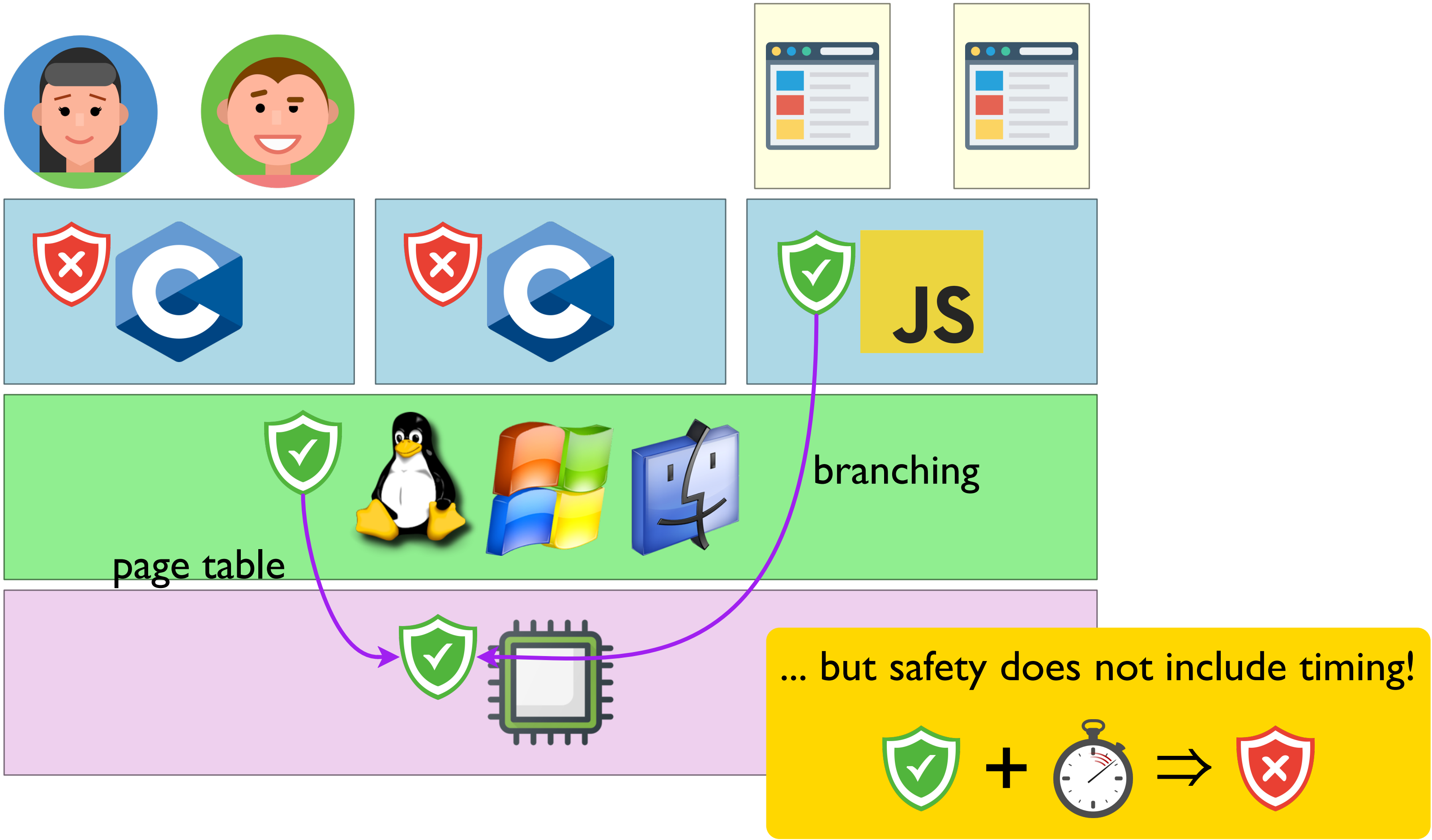


A processor is safe, because every instruction has a well-defined action for every possible case

Safety and Isolation



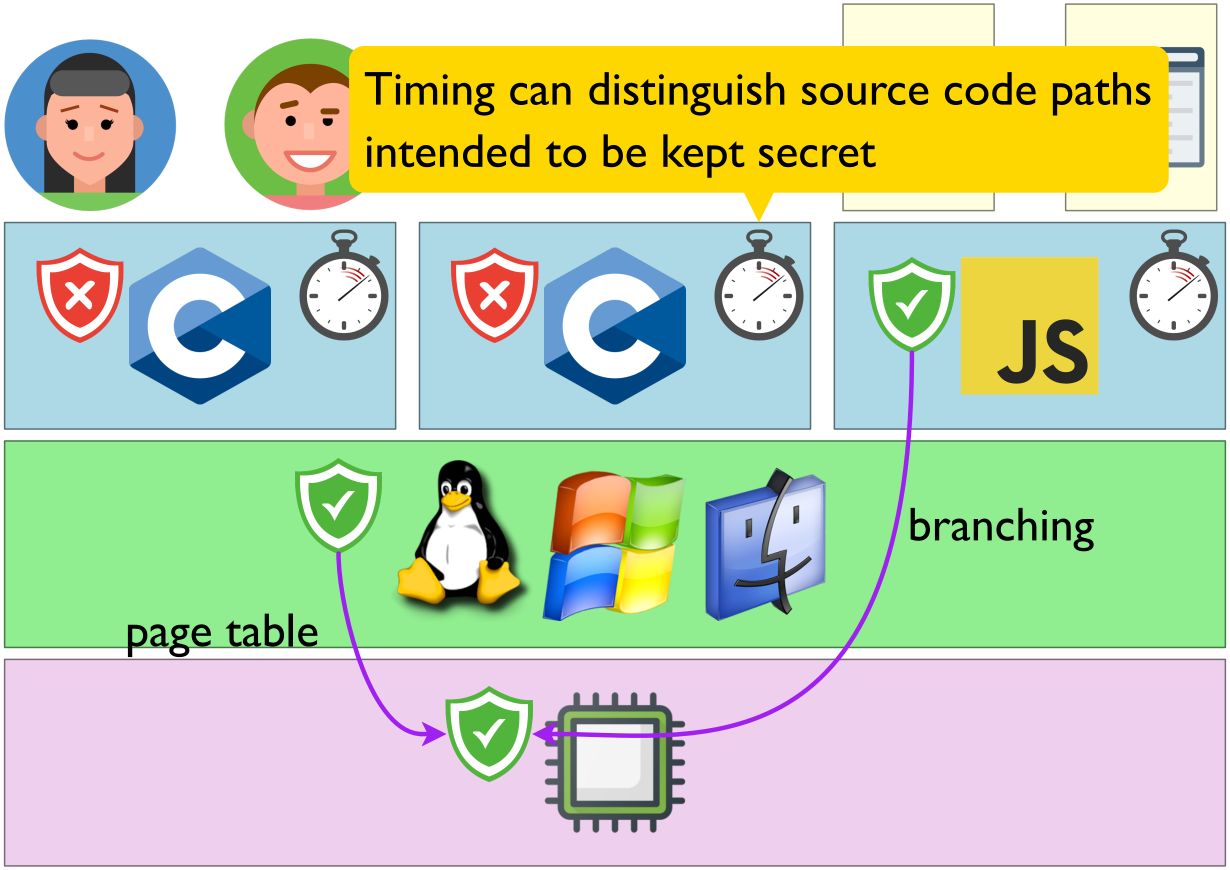
Safety and Isolation



... but safety does not include timing!

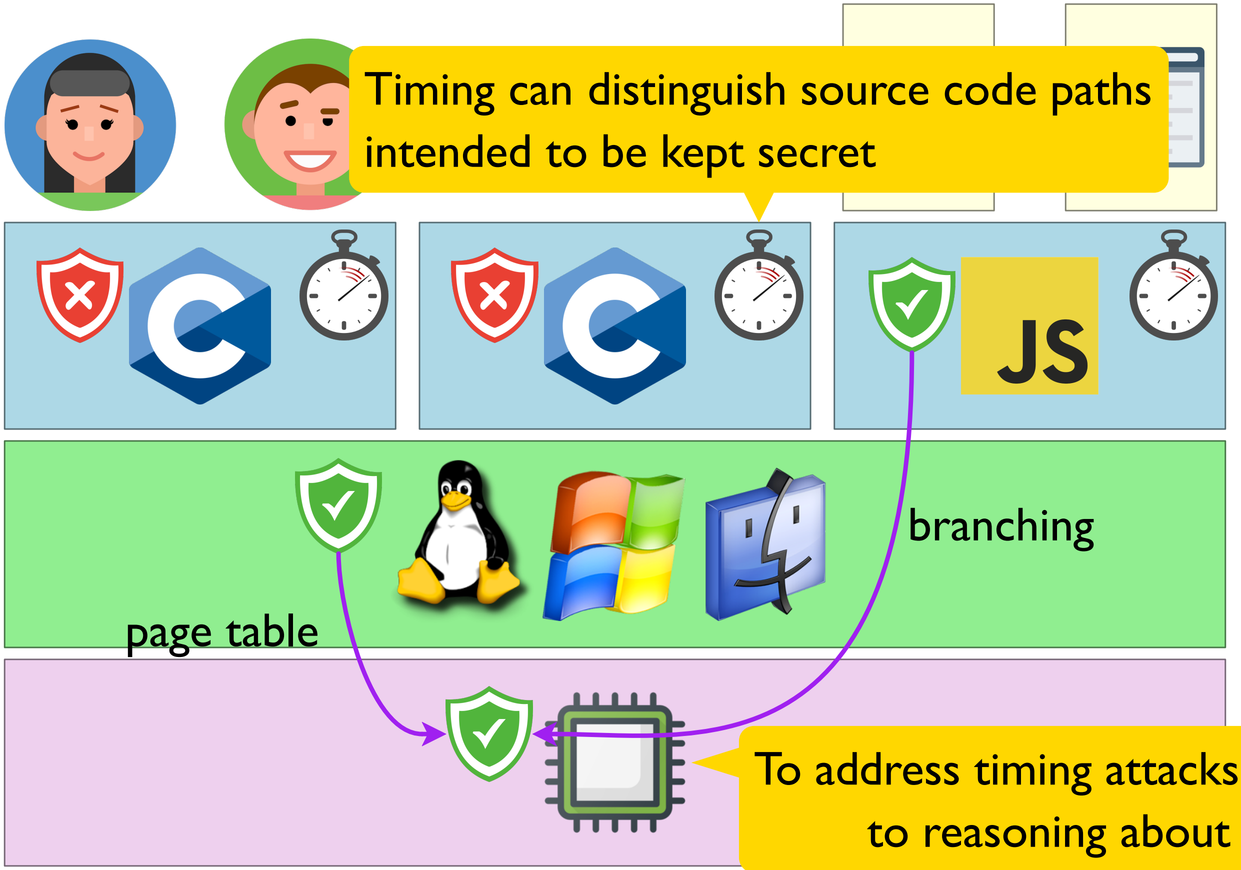
✓ + ⏱ ⇒ ✗

Safety and Isolation



Timing can distinguish source code paths intended to be kept secret

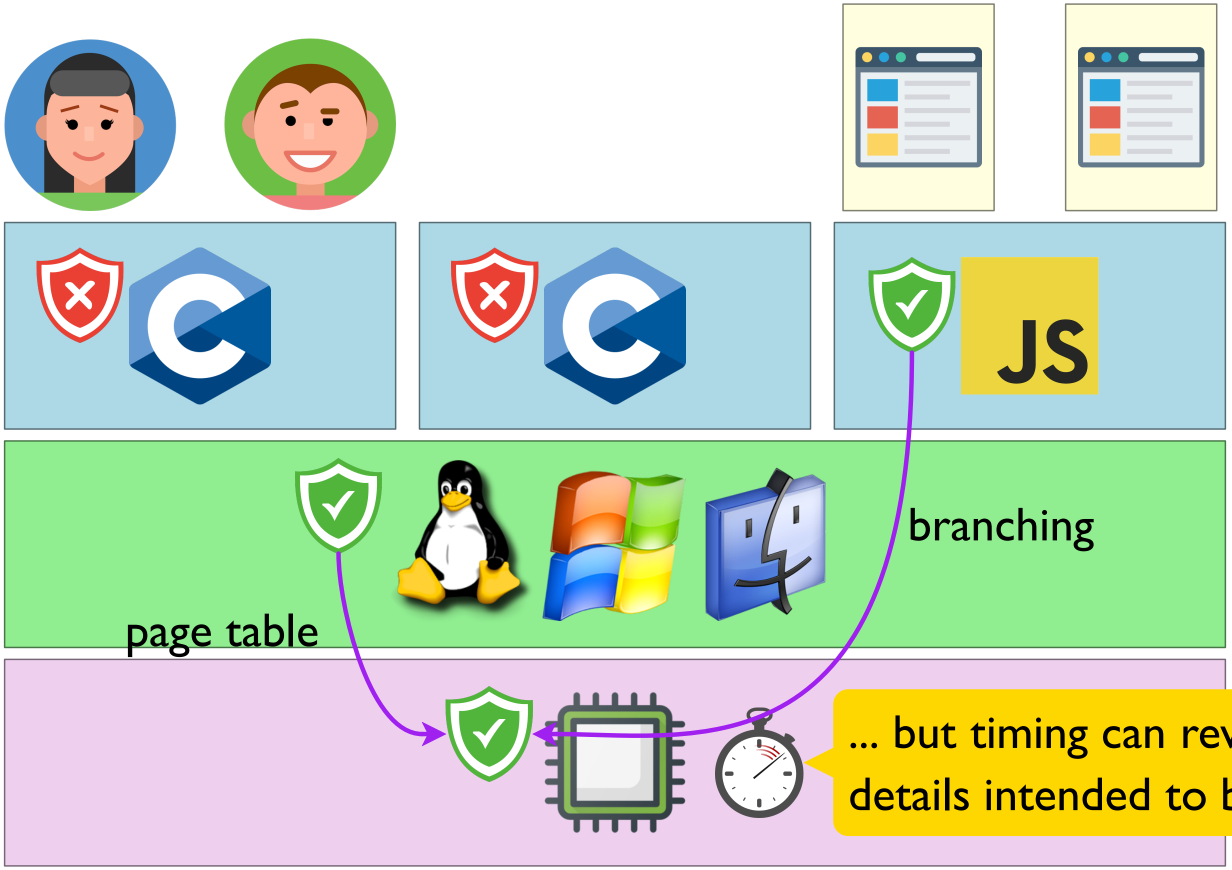
Safety and Isolation



Timing can distinguish source code paths intended to be kept secret

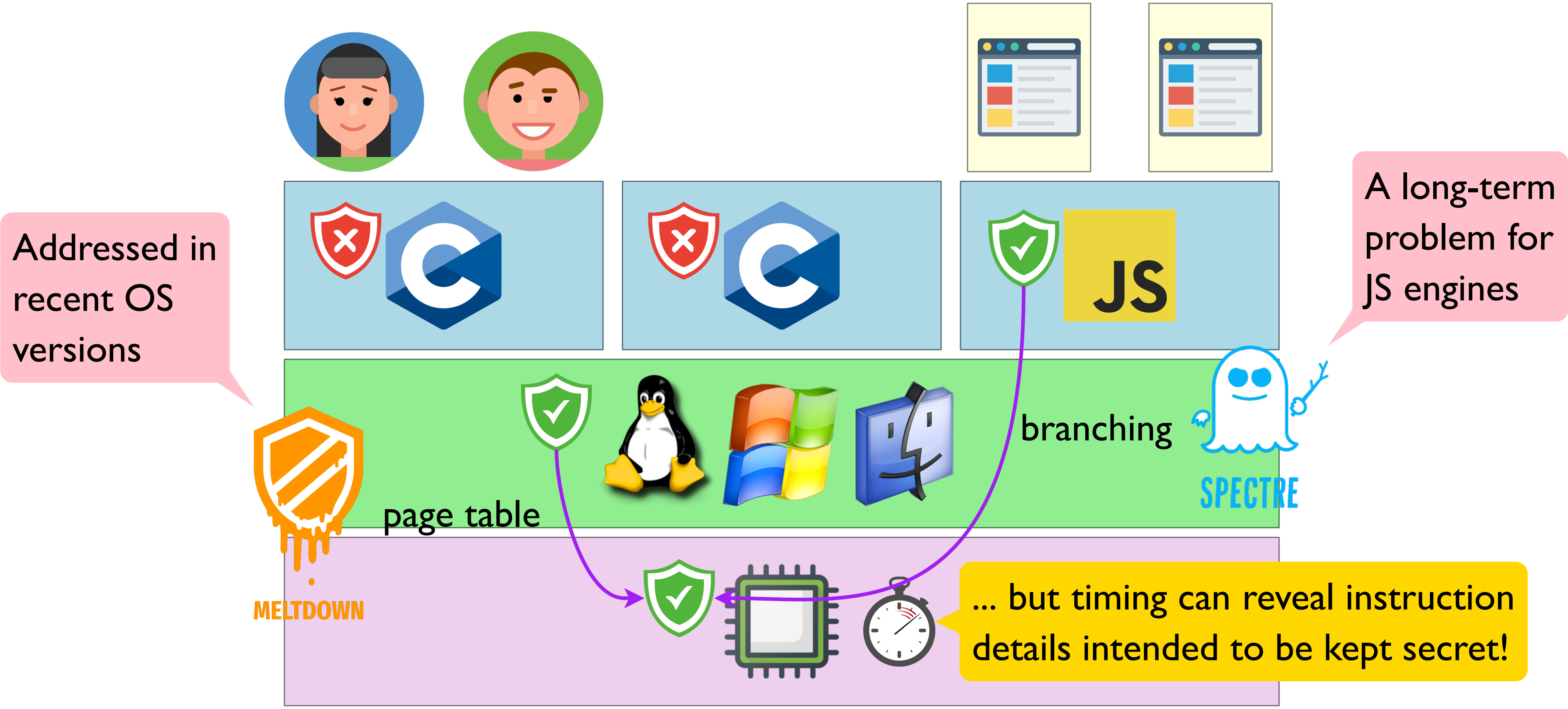
To address timing attacks, we resorted to reasoning about instructions...

Safety and Isolation



... but timing can reveal instruction details intended to be kept secret!

Safety and Isolation



Addressed in recent OS versions

A long-term problem for JS engines

... but timing can reveal instruction details intended to be kept secret!

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

Simple model of a processor: a minion that steps through instructions one-by-one

Each instruction might take a different amount of time from other instructions, but always the same itself?

Example Program

```
long sum_path(long *array, long len, long pos, long count) {
  long i = 0, sum = 0;

  while (i < count) {
    if (pos >= 0 && pos < len)
      pos = array[pos];
    else
      return -1;
    sum = sum + pos;
    i = i + 1;
  }

  return sum;
}



#define N 10
int aray[N] = { 9, 0, 1, 2, 3, 4, 5, 6, 7, 8 };
```

Example Program

```
long sum_path(long *array, long len, long pos, long count) {
  long i = 0, sum = 0;

  while (i < count) {
    if (pos >= 0 && pos < len)
      pos = array[pos];
    else
      return -1;
    sum = sum + pos;
    i = i + 1;
  }

  return sum;
}
```

Same instructions, very different times!

small `array` with small jumps:	150ms
big `array` with big jumps:	750ms

```
for (long i = 0; i < 100; i++)
  v = sum_path(array, N, i % N, 1000000);
```

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

0000

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
```

More realistic: memory reads are cached to speed up uses of the same address

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

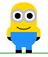
0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache

			
	0x3023 0000		

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
100003e1c  mov  x0, #0x0
100003e20  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
```

memory

•••

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

•••

cache

	0x3023 0000		

Typical **L1 cache** holds
most recently accessed 128k bytes

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache

	0x3023 0000		

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
```

Time needed to fetch address contents tells us whether it was used recently

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003de4  cmp  x3, #0x1
100003de8  b.lt 0x100003e1c
100003dec  mov  x9, #0x0
100003df0  mov  x8, #-0x1
100003df4  tbnz x2, #0x3f, 0x100003e14
100003df8  cmp  x2, x1
100003dfc  b.ge 0x100003e14
100003e00  ldr  x2, [x0, x2, lsl  #3]
100003e04  add  x9, x2, x9
100003e08  mov  x8, x9
100003e0c  subs x3, x3, #0x1
100003e10  b.ne 0x100003df0
100003e14  mov  x0, x8
100003e18  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

cache

	0x3023 0000		

Time needed to fetch address contents tells us whether it was used recently

which is no problem in itself...

Example Program

```
long sum_path(long *array, long len, long pos, long count) {
  long i = 0, sum = 0, product = 1;

  while (i < count) {
    if (pos >= 0 && pos < len)
      pos = array[pos];
    else
      return -1;
    sum = sum + pos;
    product = product * pos;
    i = i + 1;
  }

  return sum + product;
}
```

small `array` with small jumps:	150ms
big `array` with big jumps:	750ms

Example Program

```
long sum_path(long *array, long len, long pos, long count) {
  long i = 0, sum = 0, product = 1;

  while (i < count) {
    if (pos >= 0 && pos < len)
      pos = array[pos];
    else
      return -1;
    sum = sum + pos;
    product = product * pos;
    i = i + 1;
  }

  return sum + product;
}
```

No difference with more instructions?

small `array` with small jumps:	150ms
big `array` with big jumps:	750ms

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

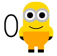
...

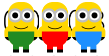
More realistic: multiple independent instructions execute at once

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache





```
100003db4  cmp   x3, #0x1
100003db8  b.lt  0x100003df4
100003dbc  mov   x9, #0x0
100003dc0  mov   x1, #0x1
100003dc4  mov   x8, #-0x1
100003dc8  tbnz  x2, #0x3f, 0x100003dec
100003dcc  cmp   x2, x1
100003dd0  b.ge  0x100003dec
100003dd4  ldr   x2, [x0, x2, lsl  #3]
100003dd8  add   x9, x2, x9
100003ddc  mul   x1, x2, x1
100003de0  subs  x3, x3, #0x1
100003de4  b.ne  0x100003dc8
100003de8  add   x8, x1, x9
100003dec  mov   x0, x8
100003df0  ret
100003df4  mov   w8, #0x1
100003df8  mov   x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dccc cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
                x1, x9
                x8

100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

cache

Data-dependent instructions must wait for earlier result

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
1000 de8   add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000

Addition and multiplication can proceed in parallel

cache


```
100003db4   cmp  x3, #0x1
100003db8   b.lt 0x100003df4
100003dbc   mov  x9, #0x0
100003dc0   mov  x1, #0x1
100003dc4   mov  x8, #-0x1
100003dc8   tbnz x2, #0x3f, 0x100003dec
        cc  cmp  x2, x1
        d0  b.ge 0x100003dec
        d4  ldr  x2, [x0, x2, lsl  #3]
100003dd8   add  x9, x2, x9
100003ddc   mul  x1, x2, x1
100003de0   subs x3, x3, #0x1
100003de4   b.ne 0x100003dc8
1000  de8   add  x8, x1, x9
100003dec   mov  x0, x8
100003df0   ret
100003df4   mov  w8, #0x1
100003df8   mov  x0, x8
100003dfc   ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003d
100003d
100003d
100003dfc  ret
```

branch prediction: guess at branch result based on previous times through

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db
100003db
100003db
100003dc
100003dc4  mov  x8, #-0x1
1000 dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

Unlike a data dependency, we could start early and then undo if the predicted branch was wrong — and that's **speculative execution**

ory

				x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
				x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
				x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
				x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

…

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc
100003de0
100003de4
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

Speculative read can start fetching memory, which might affect the cache

cache

memory

…

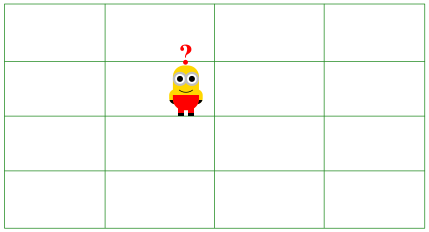
0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

…

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache



```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000	0x3023 0000	0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache


```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

?

0x3023
0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

cache

	?		
	0x3023 0000		

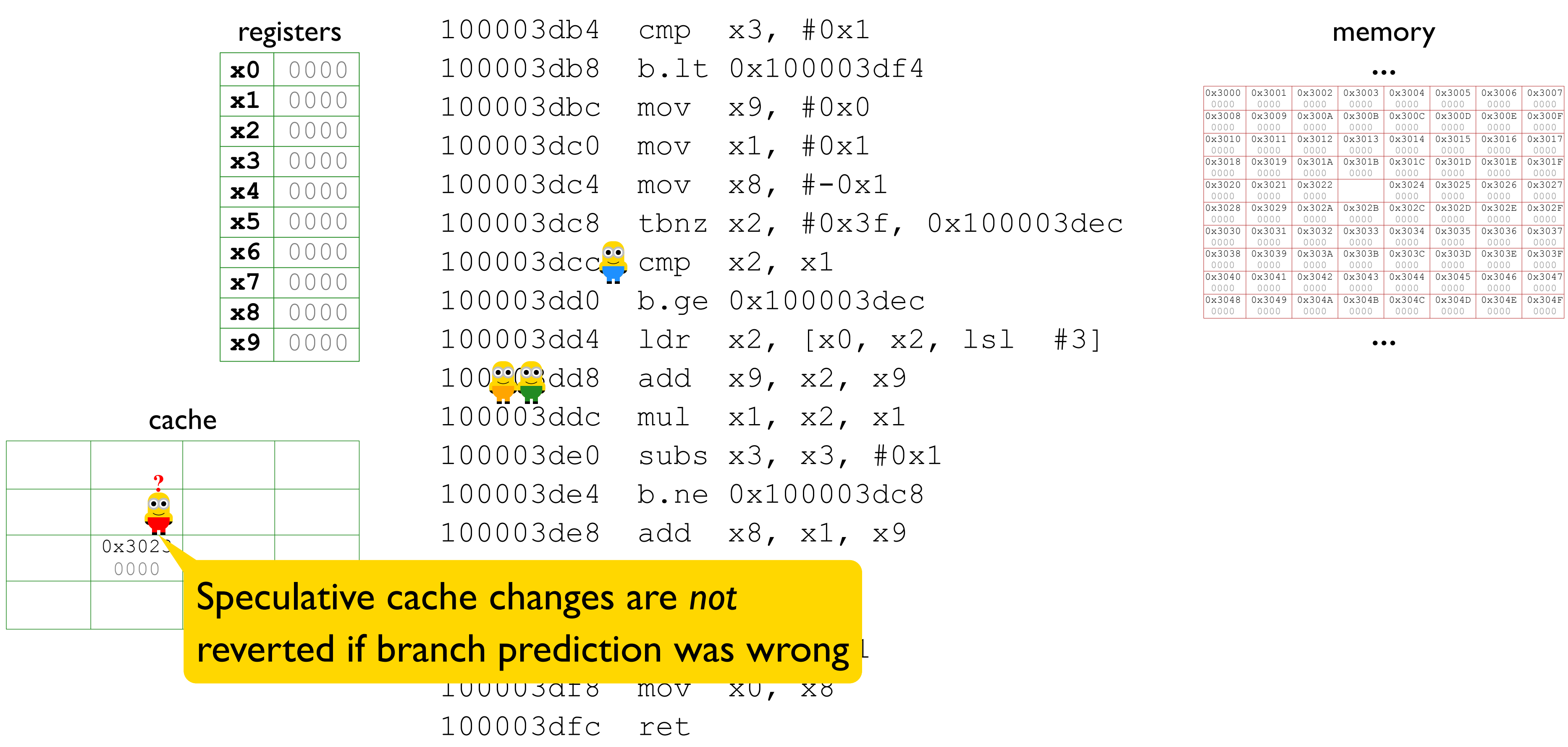
```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003dec  mov  x0, x8
100003df0  ret
100003df4  mov  w8, #0x1
100003df8  mov  x0, x8
100003dfc  ret
```

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...



registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003df8  mov  x0, x8
100003dfc  ret
```

cache

	?		
	0x3023 0000		

Speculative cache changes are *not* reverted if branch prediction was wrong

memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

registers

x0	0000
x1	0000
x2	0000
x3	0000
x4	0000
x5	0000
x6	0000
x7	0000
x8	0000
x9	0000

```
100003db4  cmp  x3, #0x1
100003db8  b.lt 0x100003df4
100003dbc  mov  x9, #0x0
100003dc0  mov  x1, #0x1
100003dc4  mov  x8, #-0x1
100003dc8  tbnz x2, #0x3f, 0x100003dec
100003dcc  cmp  x2, x1
100003dd0  b.ge 0x100003dec
100003dd4  ldr  x2, [x0, x2, lsl  #3]
100003dd8  add  x9, x2, x9
100003ddc  mul  x1, x2, x1
100003de0  subs x3, x3, #0x1
100003de4  b.ne 0x100003dc8
100003de8  add  x8, x1, x9
100003df8  mov  x0, x8
100003dfc  ret
```

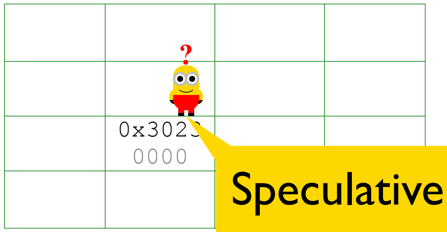
memory

...

0x3000 0000	0x3001 0000	0x3002 0000	0x3003 0000	0x3004 0000	0x3005 0000	0x3006 0000	0x3007 0000
0x3008 0000	0x3009 0000	0x300A 0000	0x300B 0000	0x300C 0000	0x300D 0000	0x300E 0000	0x300F 0000
0x3010 0000	0x3011 0000	0x3012 0000	0x3013 0000	0x3014 0000	0x3015 0000	0x3016 0000	0x3017 0000
0x3018 0000	0x3019 0000	0x301A 0000	0x301B 0000	0x301C 0000	0x301D 0000	0x301E 0000	0x301F 0000
0x3020 0000	0x3021 0000	0x3022 0000		0x3024 0000	0x3025 0000	0x3026 0000	0x3027 0000
0x3028 0000	0x3029 0000	0x302A 0000	0x302B 0000	0x302C 0000	0x302D 0000	0x302E 0000	0x302F 0000
0x3030 0000	0x3031 0000	0x3032 0000	0x3033 0000	0x3034 0000	0x3035 0000	0x3036 0000	0x3037 0000
0x3038 0000	0x3039 0000	0x303A 0000	0x303B 0000	0x303C 0000	0x303D 0000	0x303E 0000	0x303F 0000
0x3040 0000	0x3041 0000	0x3042 0000	0x3043 0000	0x3044 0000	0x3045 0000	0x3046 0000	0x3047 0000
0x3048 0000	0x3049 0000	0x304A 0000	0x304B 0000	0x304C 0000	0x304D 0000	0x304E 0000	0x304F 0000

...

cache



Speculative cache changes are *not* reverted if branch prediction was wrong

which is no problem in itself...

Speculation and Caching Consequences

Caching and speculative execution are **crucial** to performance, and speculation may run many instructions — not just one!

Speculative execution can try things that shouldn't happen

- due to a bounds check or tag check that guards a read
- due to page protection, which is a similarly pipelined check

These things-that-should-never-happen are rolled back, so the program never sees the effects

... execpt via timing

⏱

The Attack

```
int *my_pages[256]; // array of pointers to different pages

int probe(int check, void *addr) {
  if (check) {
    int8 index = *(int8 *)addr;
    return *(my_pages[index]);
  } else
    return 0;
}
```

Step 1: call `probe` many times with `true` and a good address

⇒ convince the branch predictor that `check` is probably true

The Attack

```
int *my_pages[256]; // array of pointers to different pages

int probe(int check, void *addr) {
  if (check) {
    int8 index = *(int8 *)addr;
    return *(my_pages[index]);
  } else
    return 0;
}
```

Step 2: read a lot of memory not in `my_pages`

⇒ no pages in `my_pages` are cached

The Attack

```
int *my_pages[256]; // array of pointers to different pages

int probe(int check, void *addr) {
  if (check) {
    int8 index = *(int8 *)addr;
    return *(my_pages[index]);
  } else
    return 0;
}
```

JavaScript variant: use an array and index instead of a raw address

Step 3: call `probe` with `false` and an `addr` to attack

⇒ byte *that you should never see* at `addr` is read speculatively

⇒ byte is used to index `my_pages`, bringing one page into the cache

The Attack

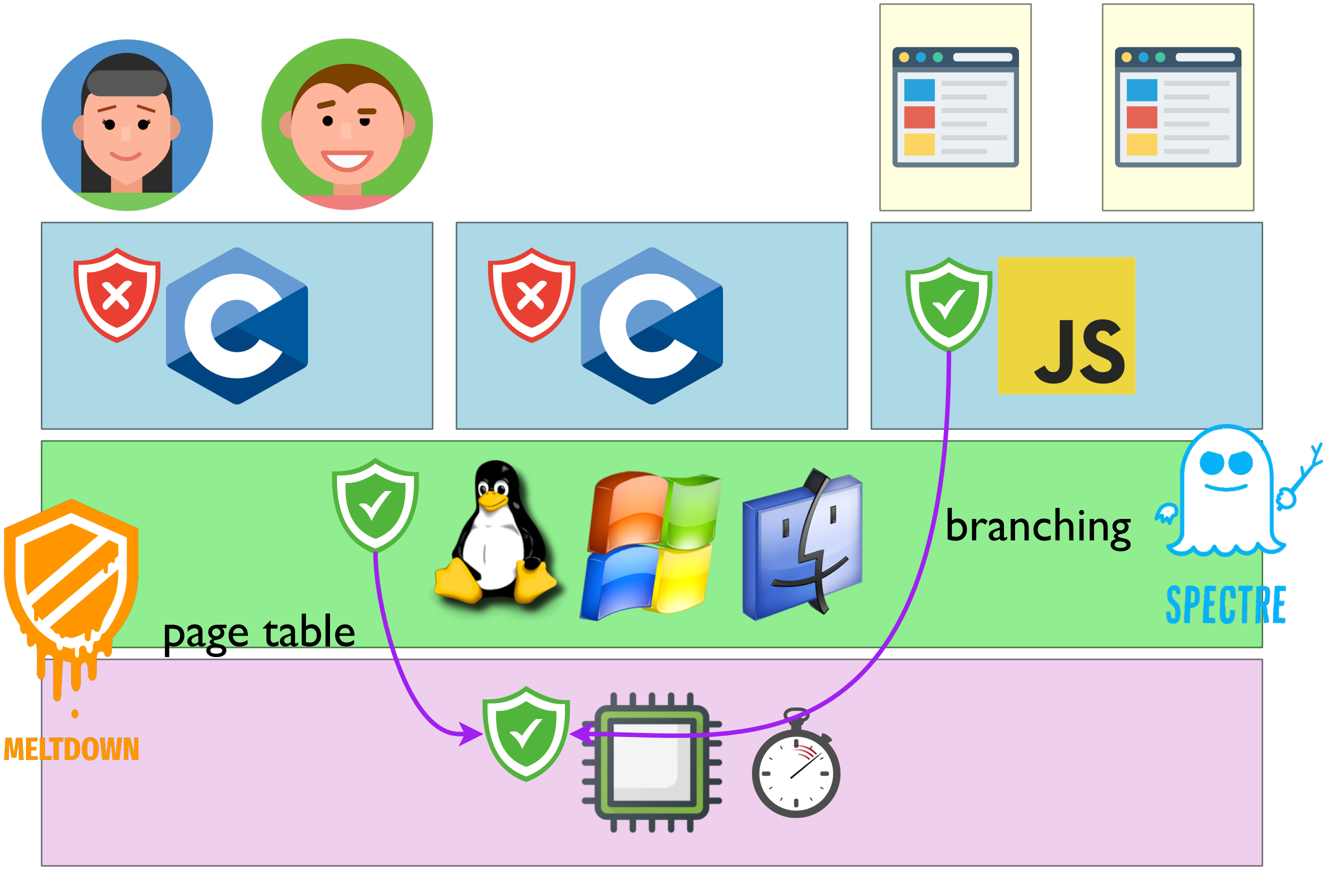
```
int *my_pages[256]; // array of pointers to different pages

int probe(int check, void *addr) {
  if (check) {
    int8 index = *(int8 *)addr;
    return *(my_pages[index]);
  } else
    return 0;
}
```

Step 4: ⏱ time a read of each page in `my_pages`

⇒ the fast one tells you what byte was read speculatively!

Safety and Isolation



JavaScript implication: use array+index to read any byte of memory

Browser implication: pages are not isolated

Solutions:

- turn off speculation, which makes things slow
- generate code to cooperate less, which makes things slow
- reduce timer resolution, which makes attacks harder
- 🤷‍♂️

MELTDOWN

Different processes have different memory pages

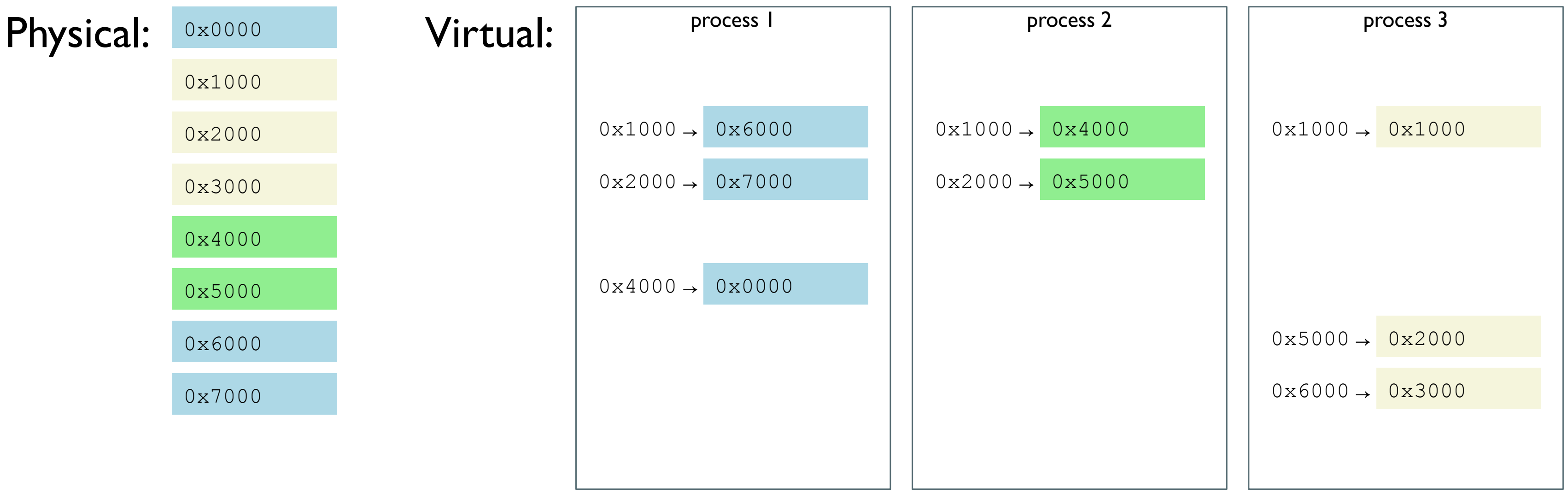
Within one process, C can read any address, anyway

so what’s the big deal?

As it turns out, OS kernels map all physical memory in every process, but guards it with page protection

- enables access across process boundaries
- avoids expensive page-table resets

Page Tables



Physical:

0x0000
0x1000
0x2000
0x3000
0x4000
0x5000
0x6000
0x7000

Virtual:

process 1

0x1000 → 0x6000
0x2000 → 0x7000
0x4000 → 0x0000

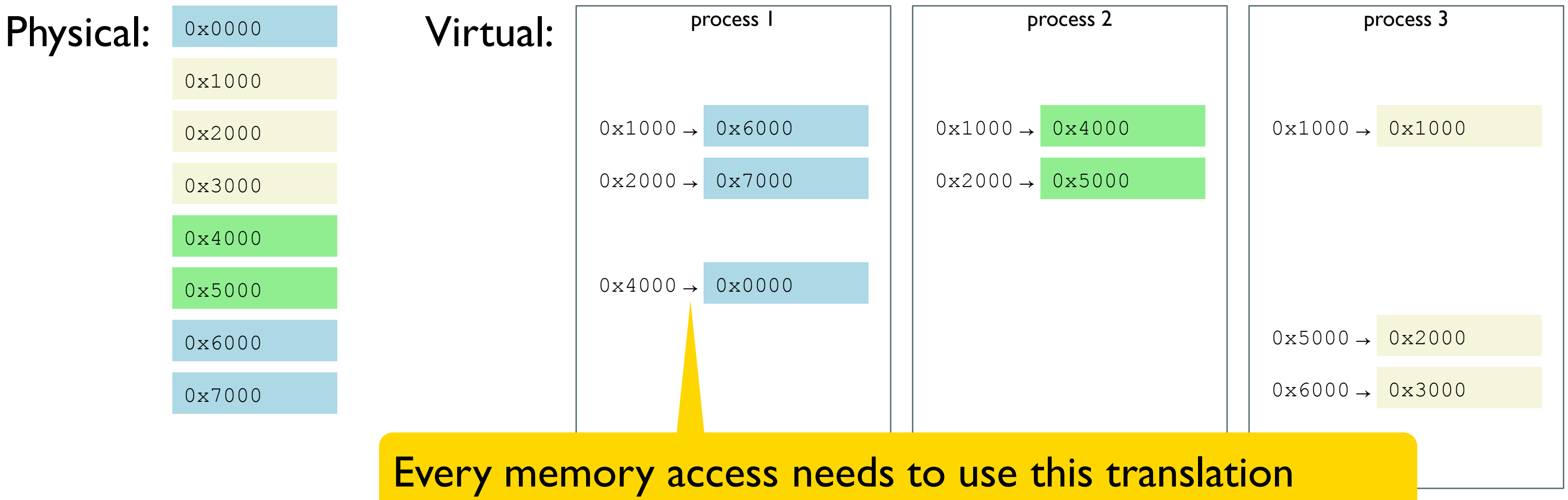
process 2

0x1000 → 0x4000
0x2000 → 0x5000

process 3

0x1000 → 0x1000
0x5000 → 0x2000
0x6000 → 0x3000

Page Tables



Physical:

0x0000
0x1000
0x2000
0x3000
0x4000
0x5000
0x6000
0x7000

Virtual:

process 1

0x1000 → 0x6000
0x2000 → 0x7000
0x4000 → 0x0000

process 2

0x1000 → 0x4000
0x2000 → 0x5000

process 3

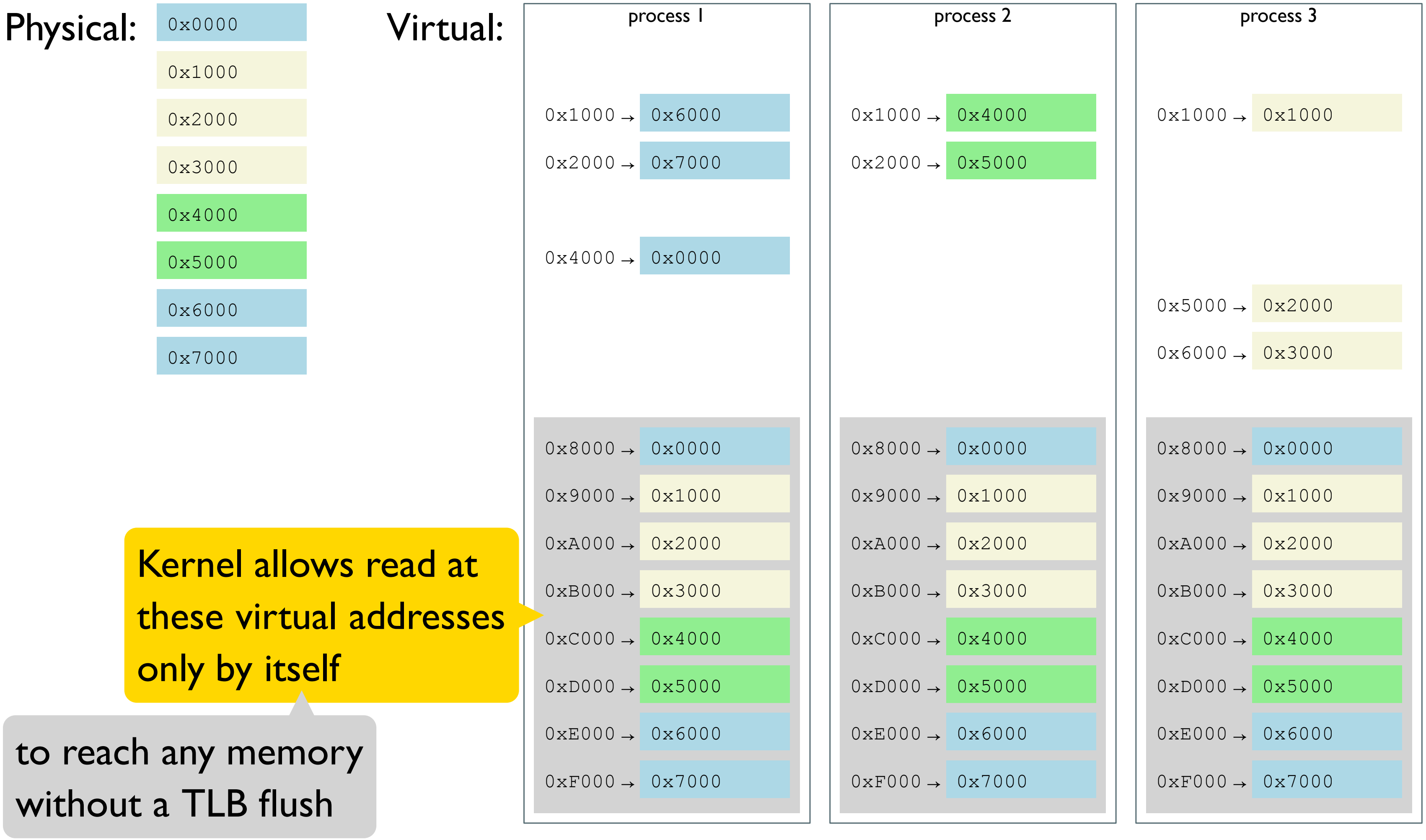
0x1000 → 0x1000
0x5000 → 0x2000
0x6000 → 0x3000

Every memory access needs to use this translation

The **translation lookaside buffer (TLB)** caches it

must be flushed when changing processes

Page Tables



Physical:

0x0000
0x1000
0x2000
0x3000
0x4000
0x5000
0x6000
0x7000

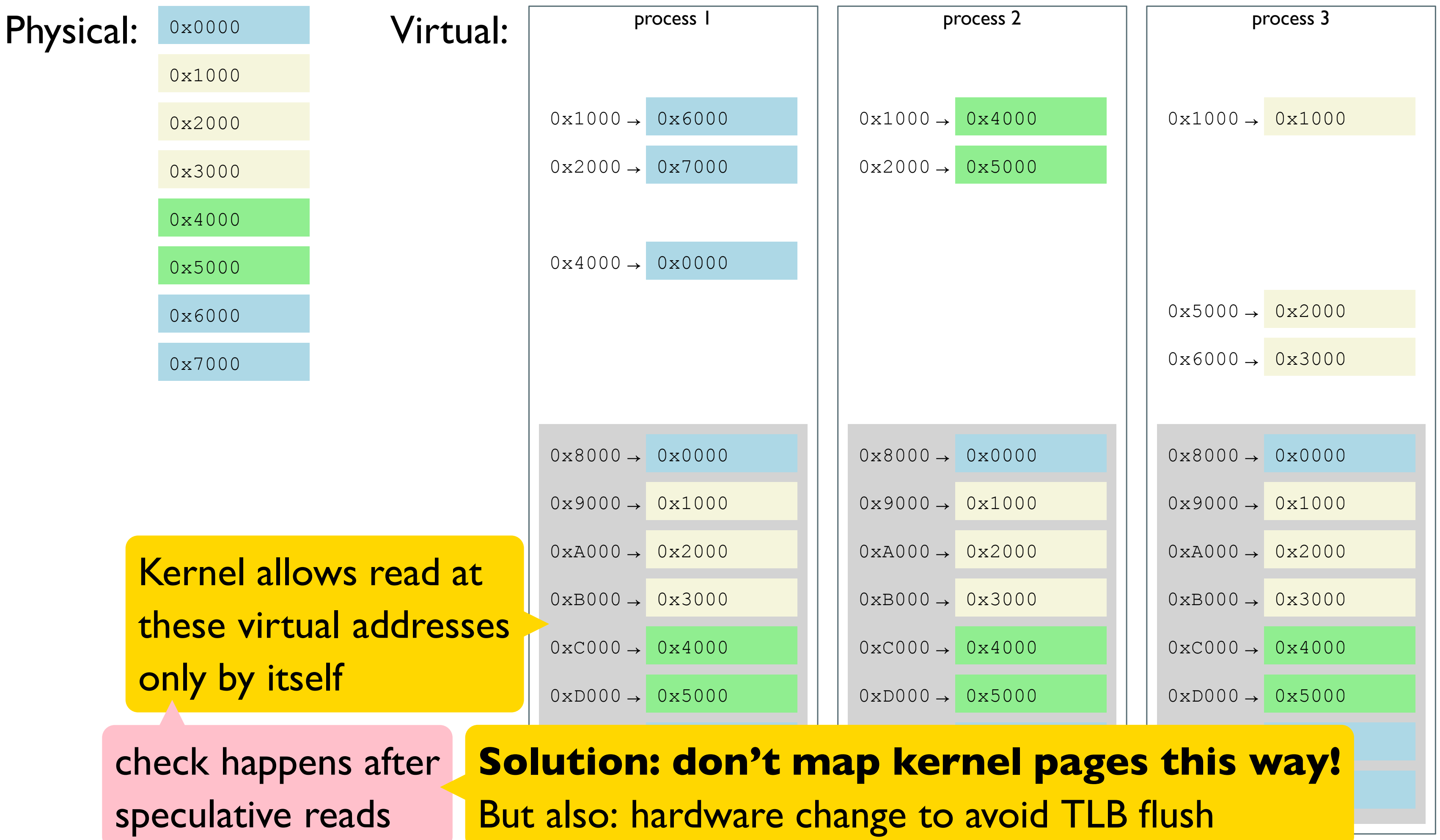
Virtual:

process 1	process 2	process 3
0x1000 → 0x6000	0x1000 → 0x4000	0x1000 → 0x1000
0x2000 → 0x7000	0x2000 → 0x5000	
0x4000 → 0x0000		
		0x5000 → 0x2000
		0x6000 → 0x3000
0x8000 → 0x0000	0x8000 → 0x0000	0x8000 → 0x0000
0x9000 → 0x1000	0x9000 → 0x1000	0x9000 → 0x1000
0xA000 → 0x2000	0xA000 → 0x2000	0xA000 → 0x2000
0xB000 → 0x3000	0xB000 → 0x3000	0xB000 → 0x3000
0xC000 → 0x4000	0xC000 → 0x4000	0xC000 → 0x4000
0xD000 → 0x5000	0xD000 → 0x5000	0xD000 → 0x5000
0xE000 → 0x6000	0xE000 → 0x6000	0xE000 → 0x6000
0xF000 → 0x7000	0xF000 → 0x7000	0xF000 → 0x7000

Kernel allows read at these virtual addresses only by itself

to reach any memory without a TLB flush

Page Tables



Kernel allows read at these virtual addresses only by itself

check happens after speculative reads

Solution: don't map kernel pages this way!

But also: hardware change to avoid TLB flush

Summary

Meltdown and **Spectre** are recently discovered side-channel attacks

These exploits are not easy to block, because they take advantage of key implementation techniques for making processors run fast:

- **memory caches**
- **speculative execution**