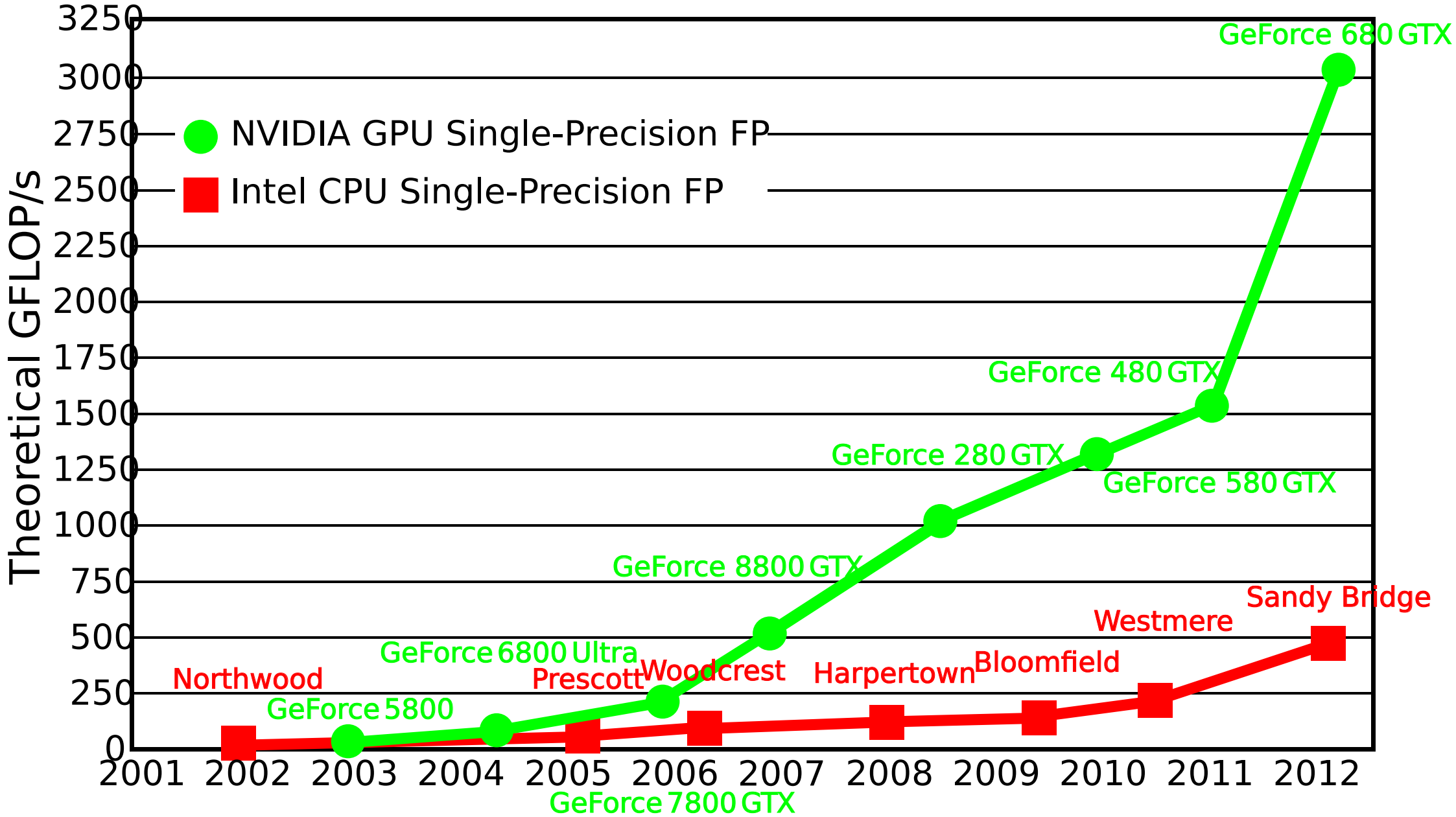
Warp-Aware Trace Scheduling for GPUS

James Jablin (Brown)
Thomas Jablin (UIUC)
Onur Mutlu (CMU)
Maurice Herlihy (Brown)

Historical Trends in GFLOPS: CPUs vs. GPUs



reproduced from NVIDIA CUDA C Programming Guide (Version 5.0)

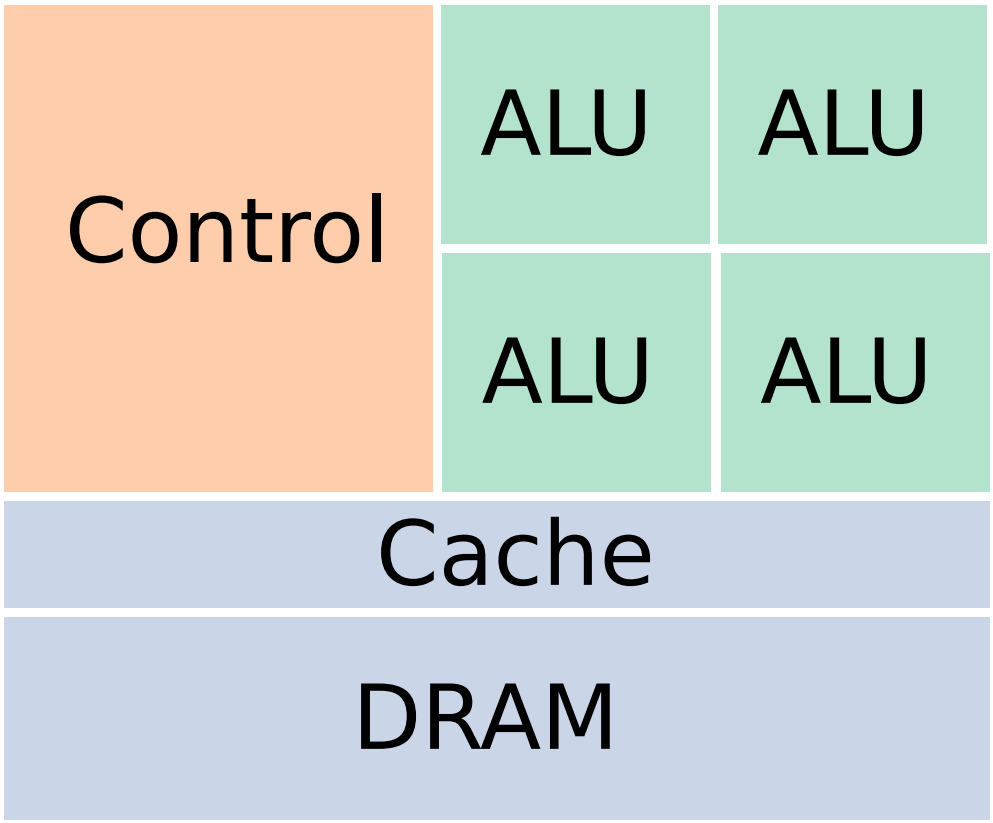
Performance Pitfalls

Control flow can
negatively affect performance.

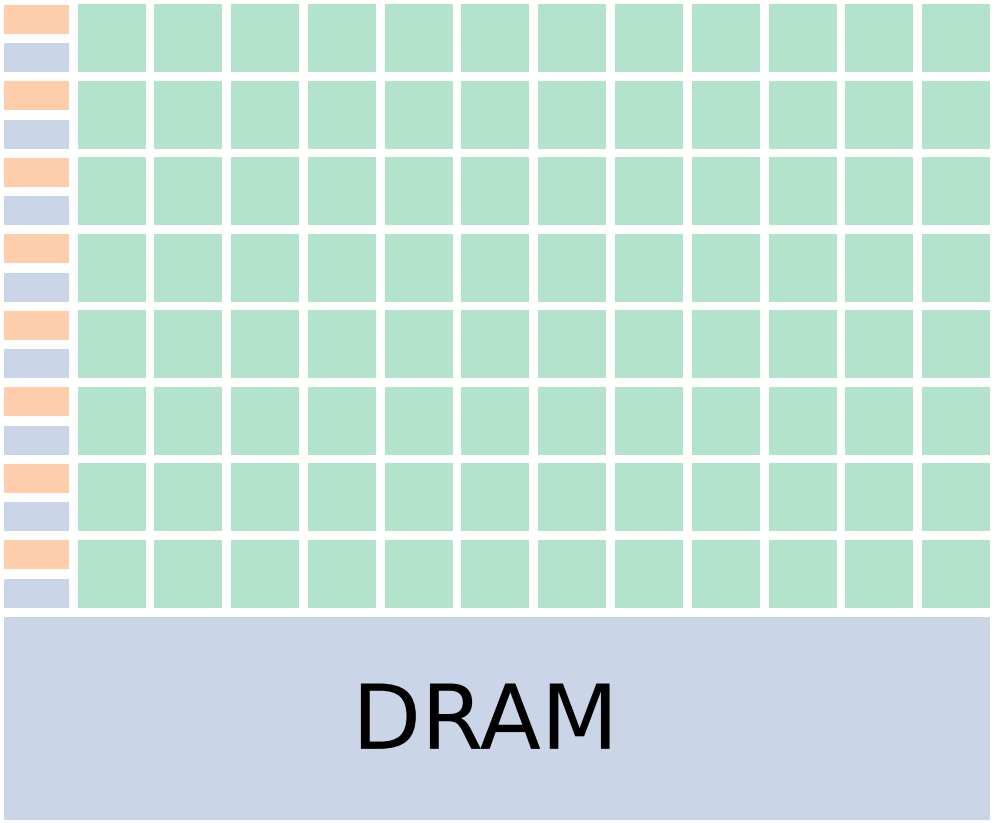
Performance Pitfalls

Pipeline Stall - execution delay in an instruction pipeline to resolve a dependency

Hardware: CPU versus GPU

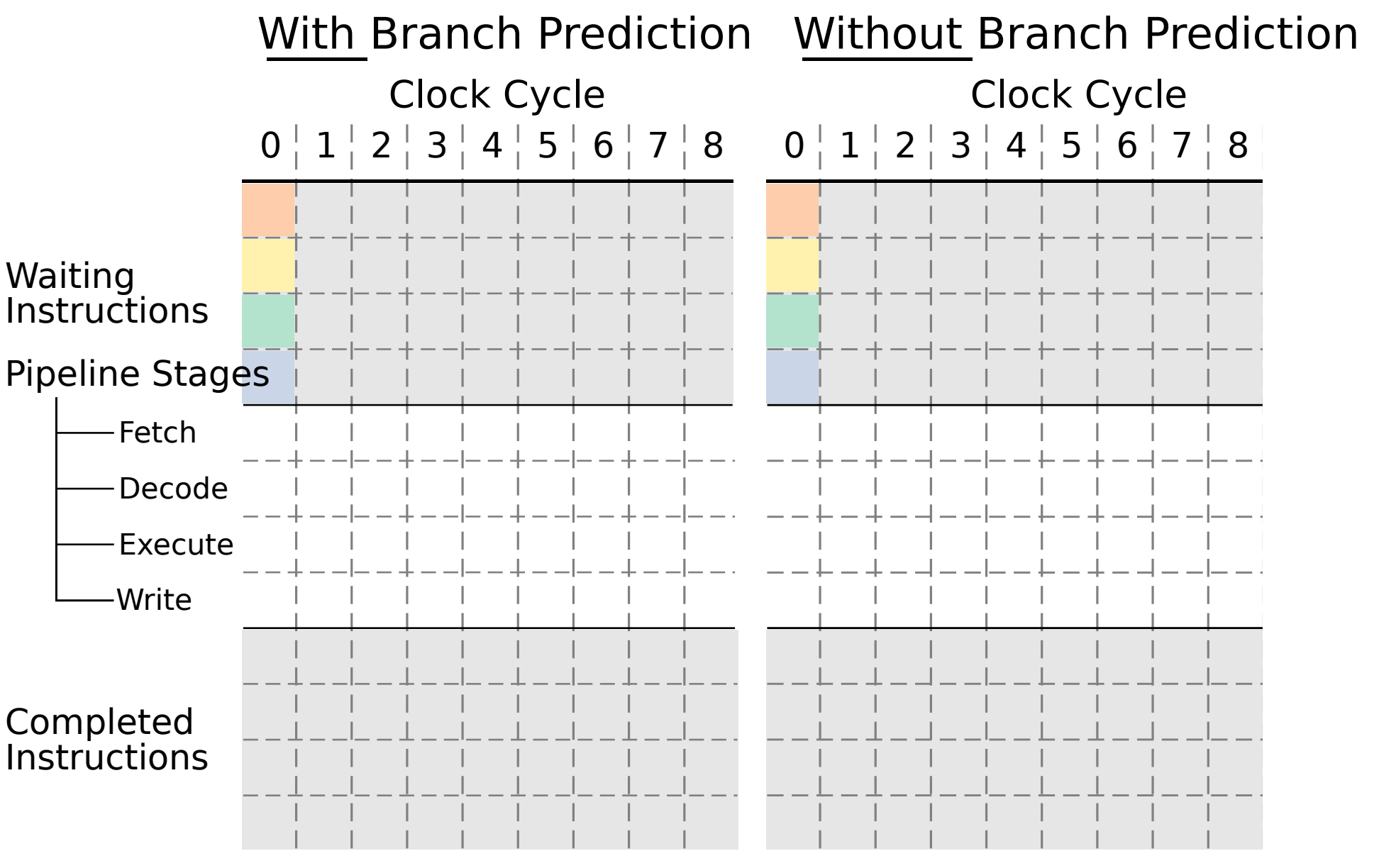


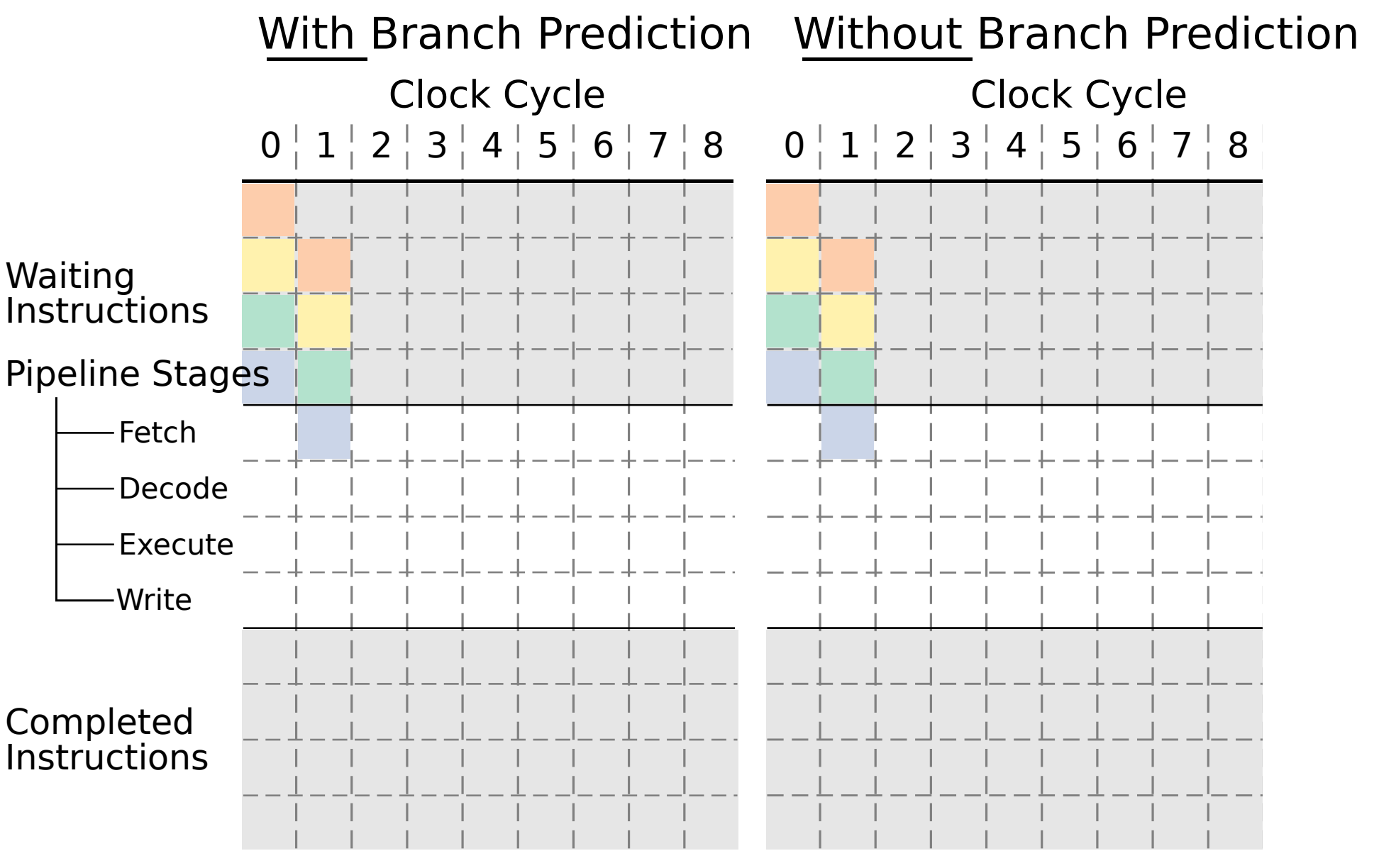
CPU

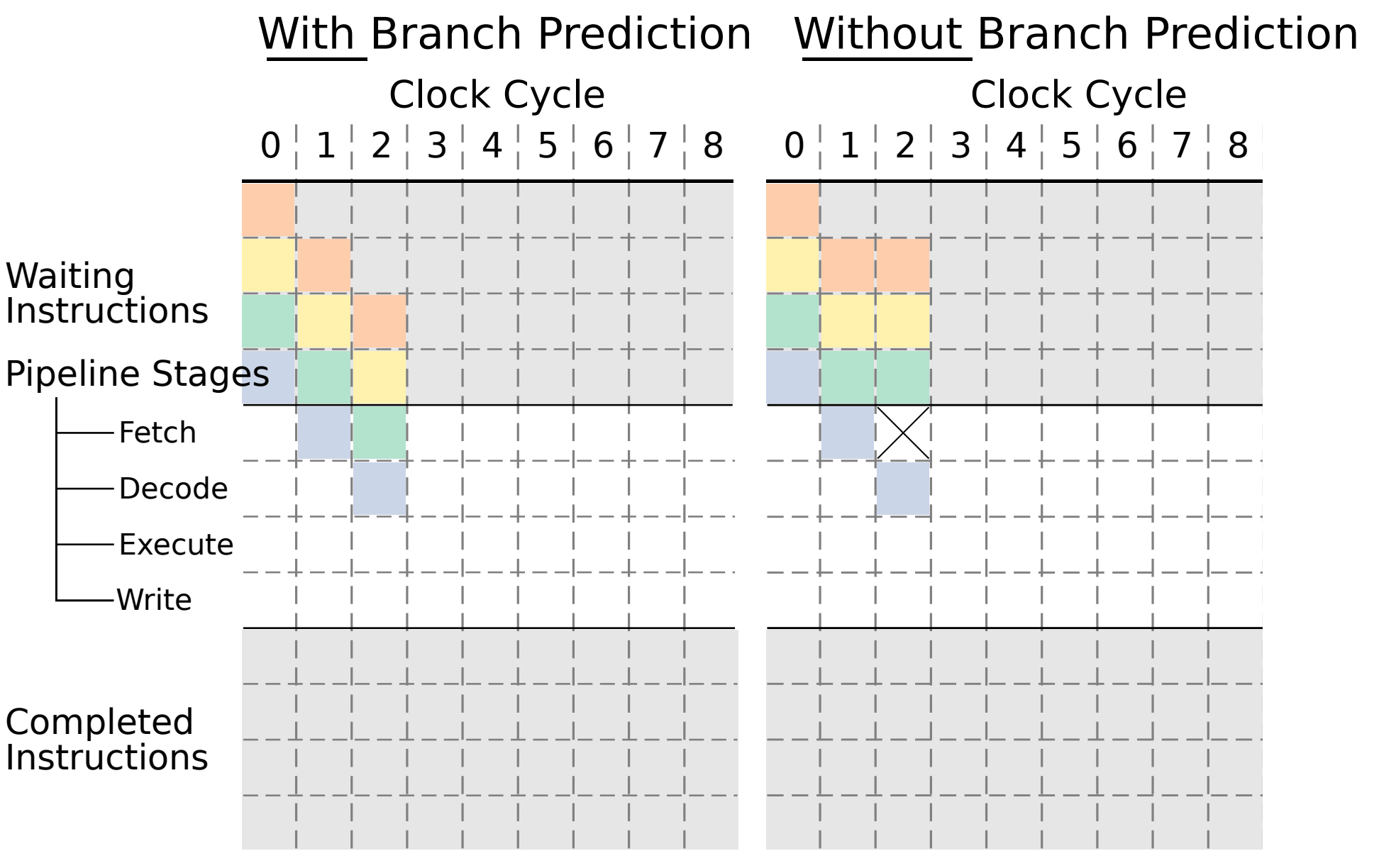


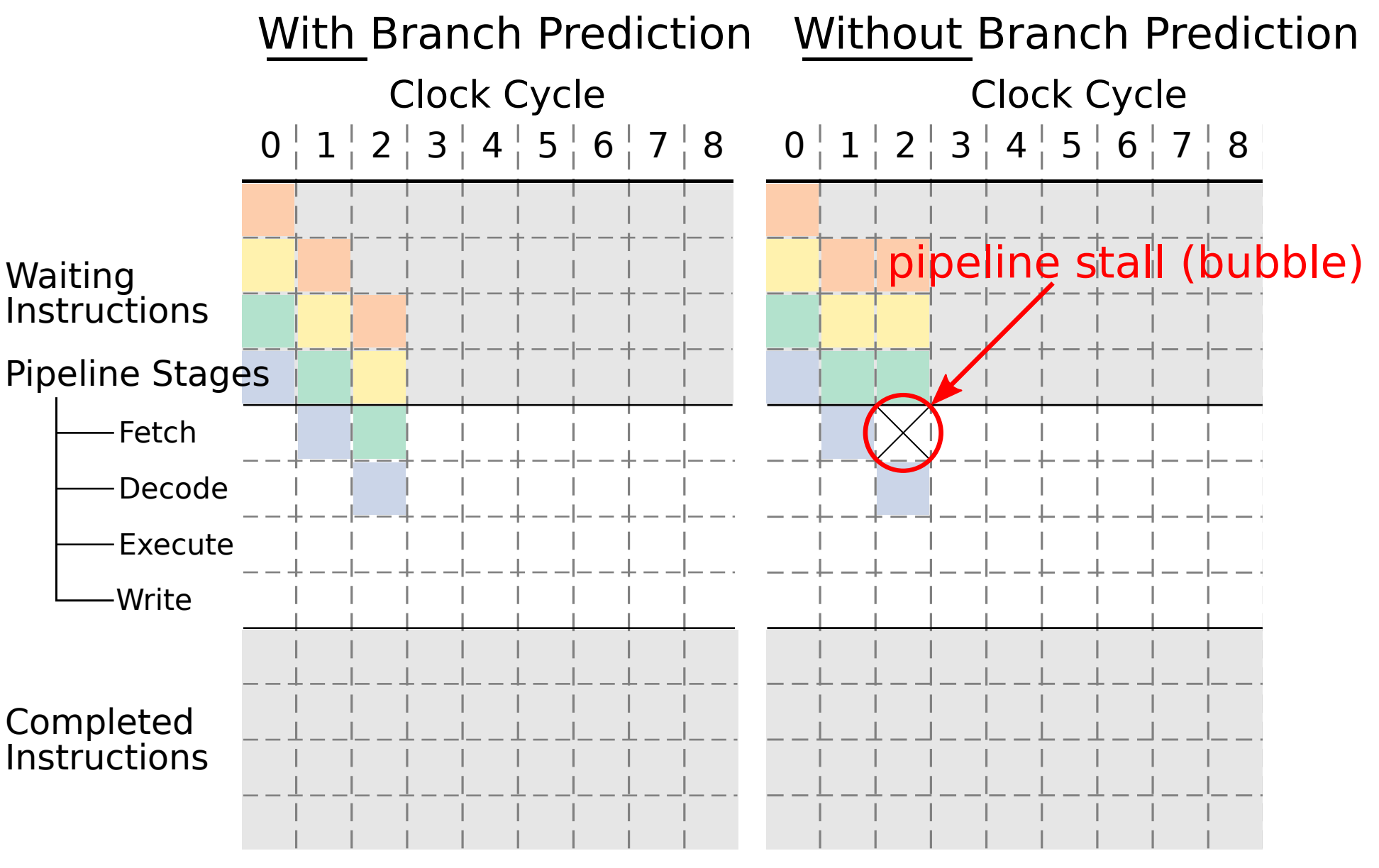
GPU

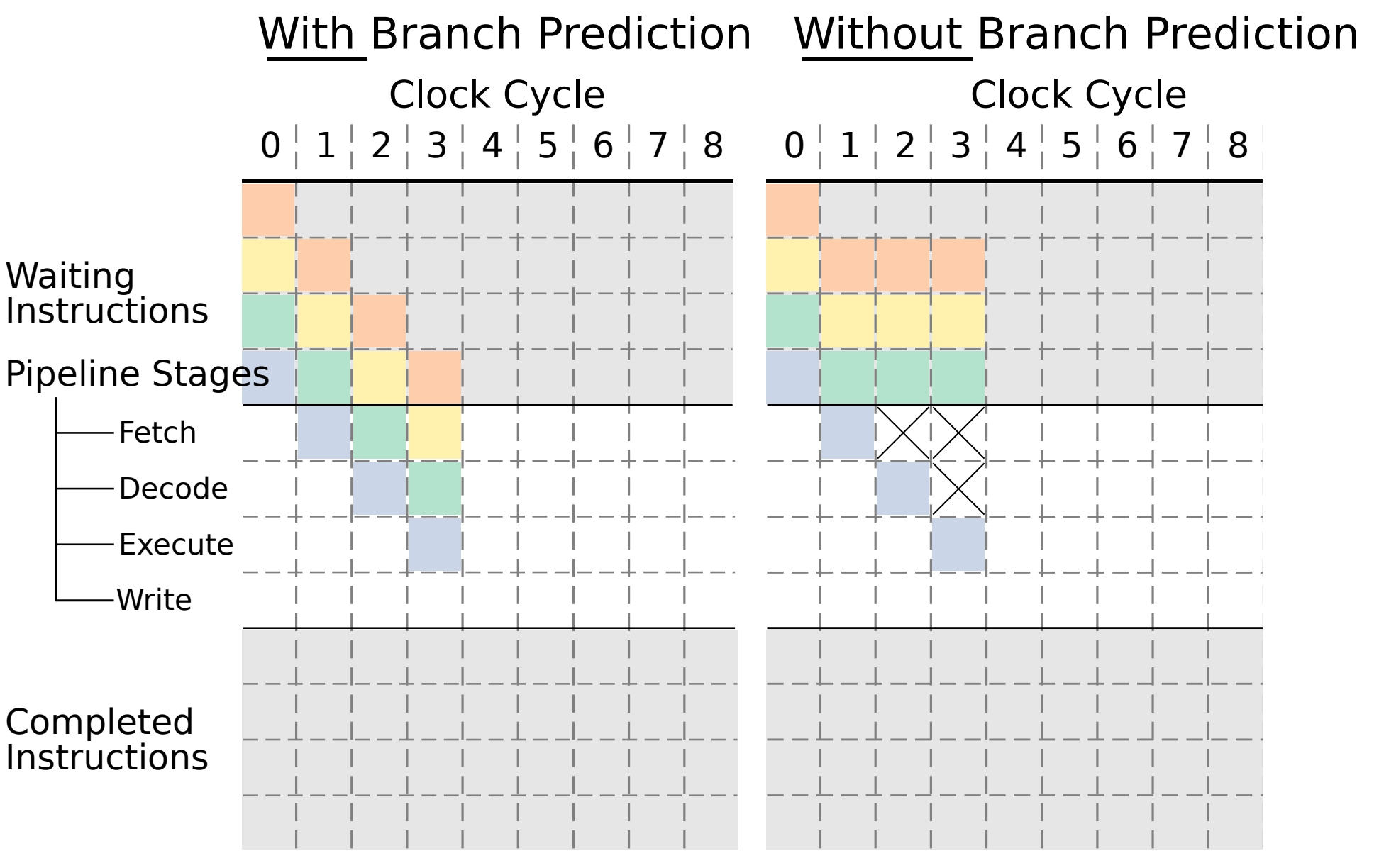
reproduced from NVIDIA CUDA C Programming Guide (Version 5.0)

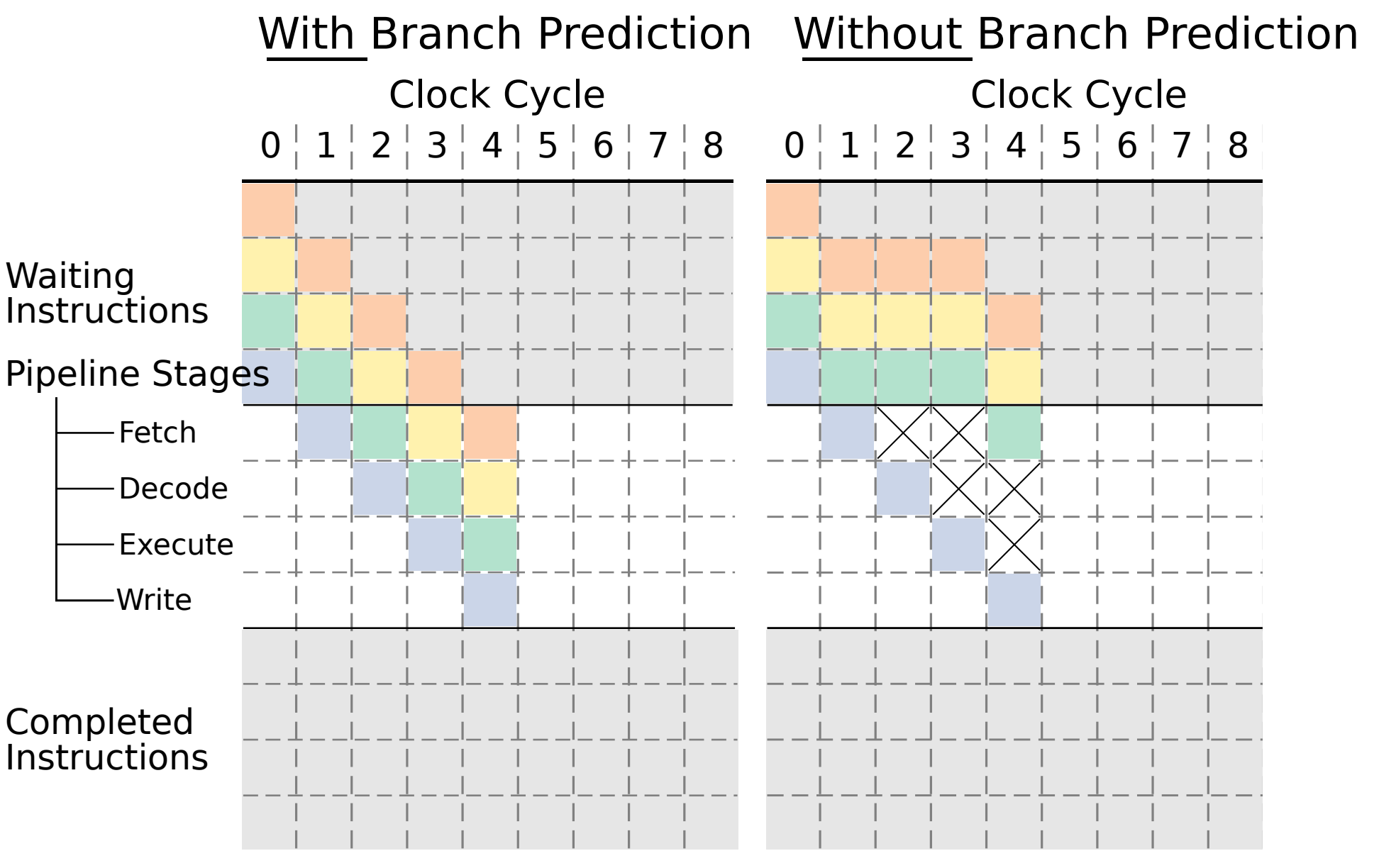

With Branch Prediction
Without Branch Prediction
Clock Cycle
0 1 2 3 4 5 6 7 8
Clock Cycle
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

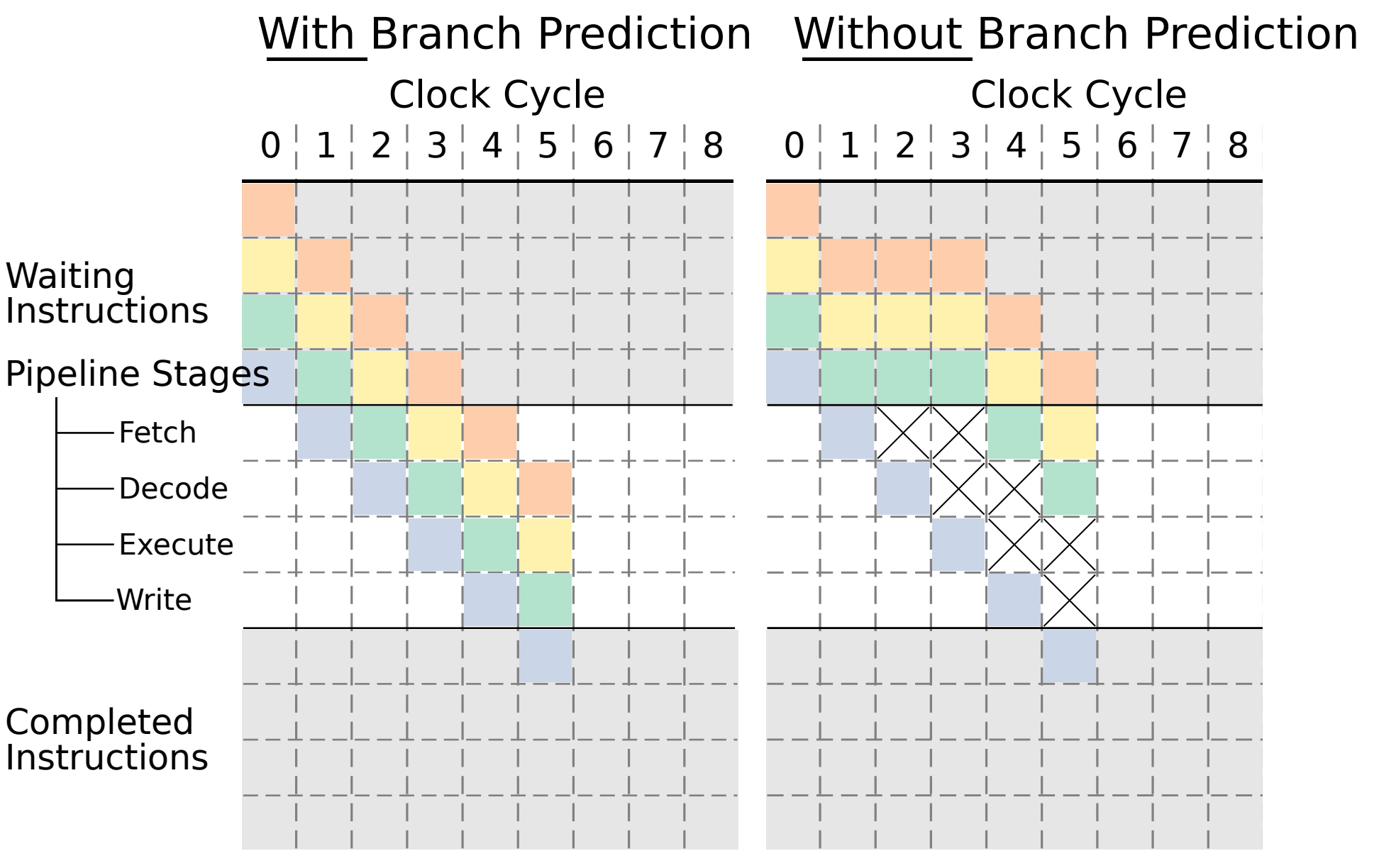

With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

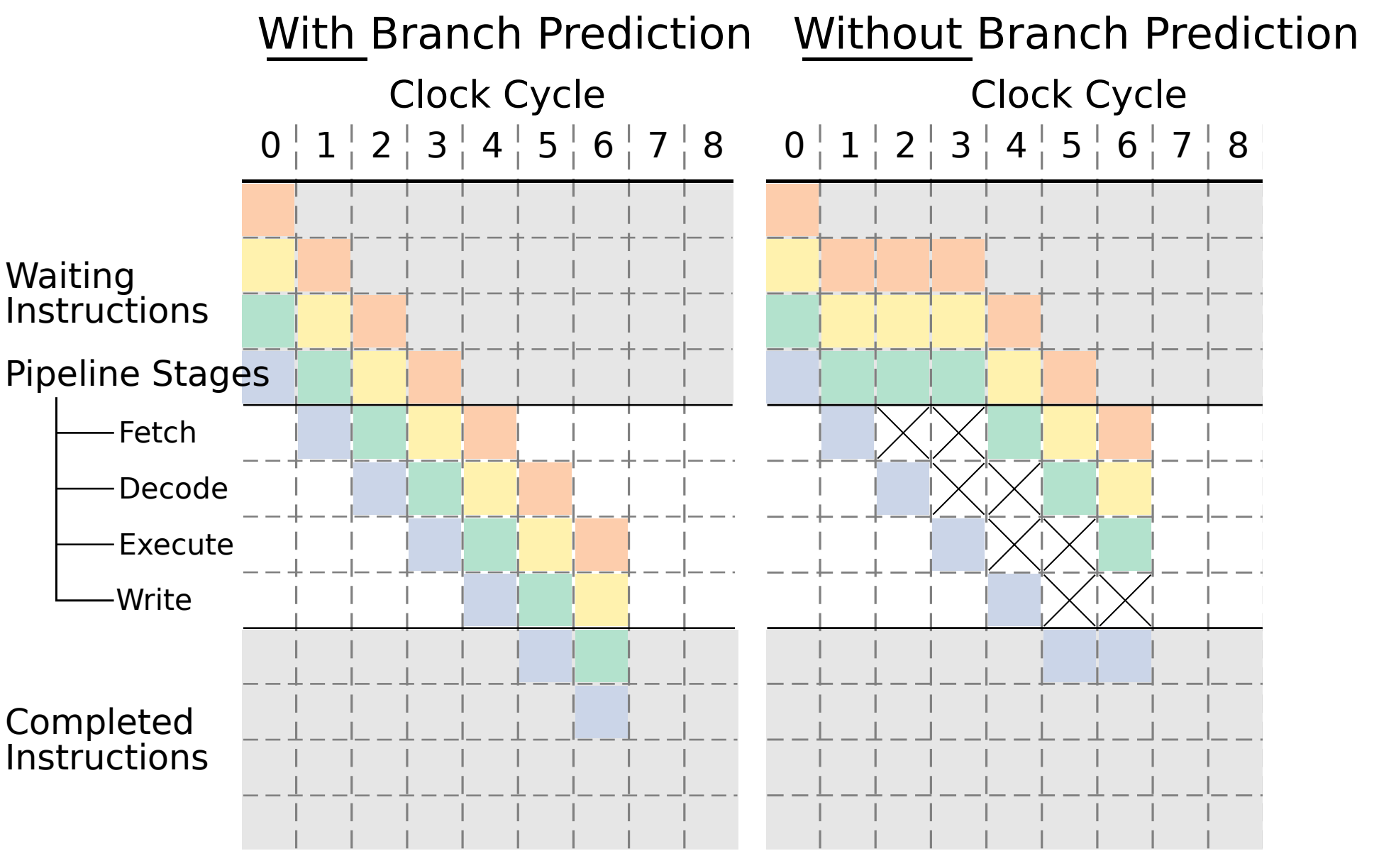

With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

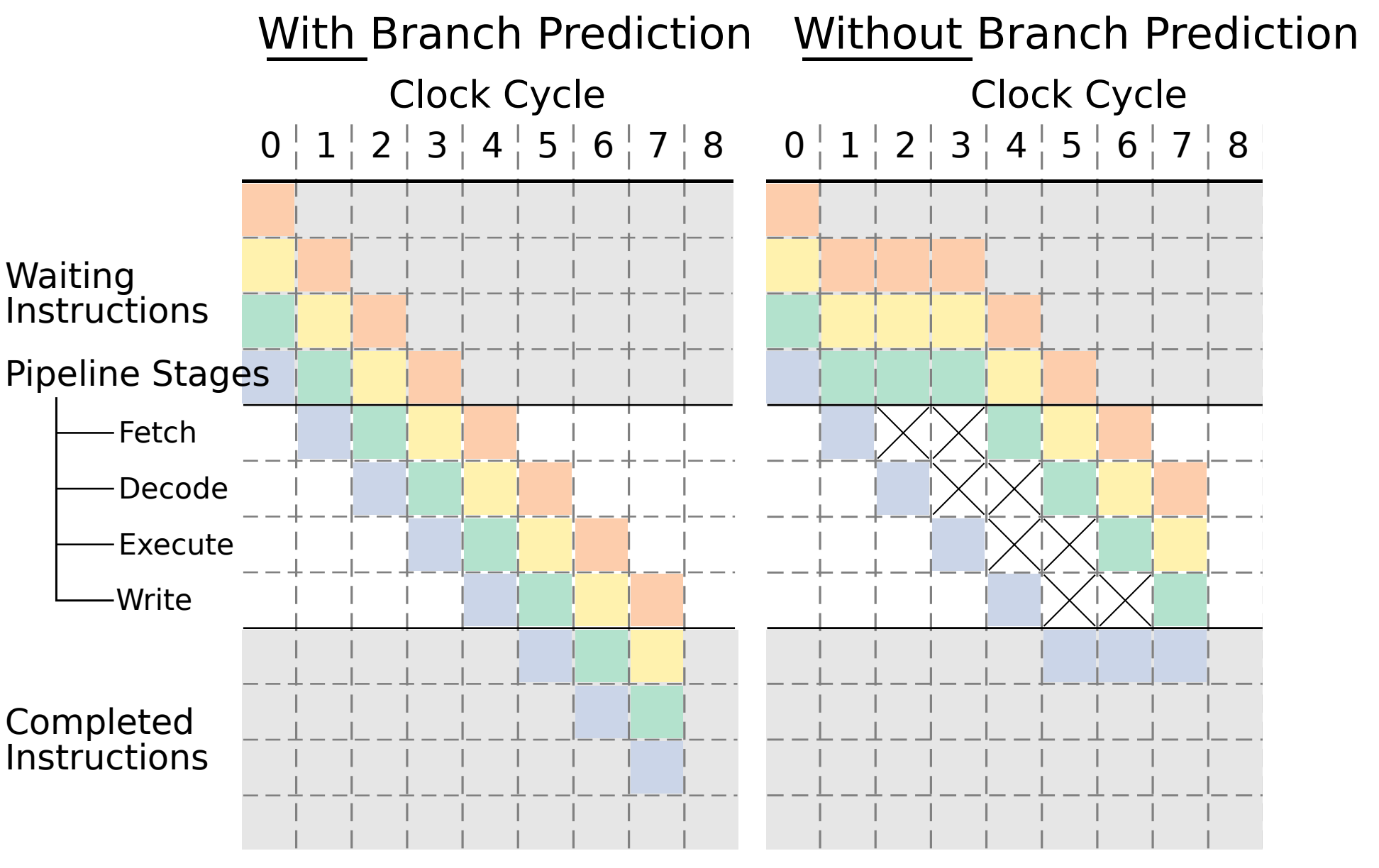

With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
pipeline stall (bubble)
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

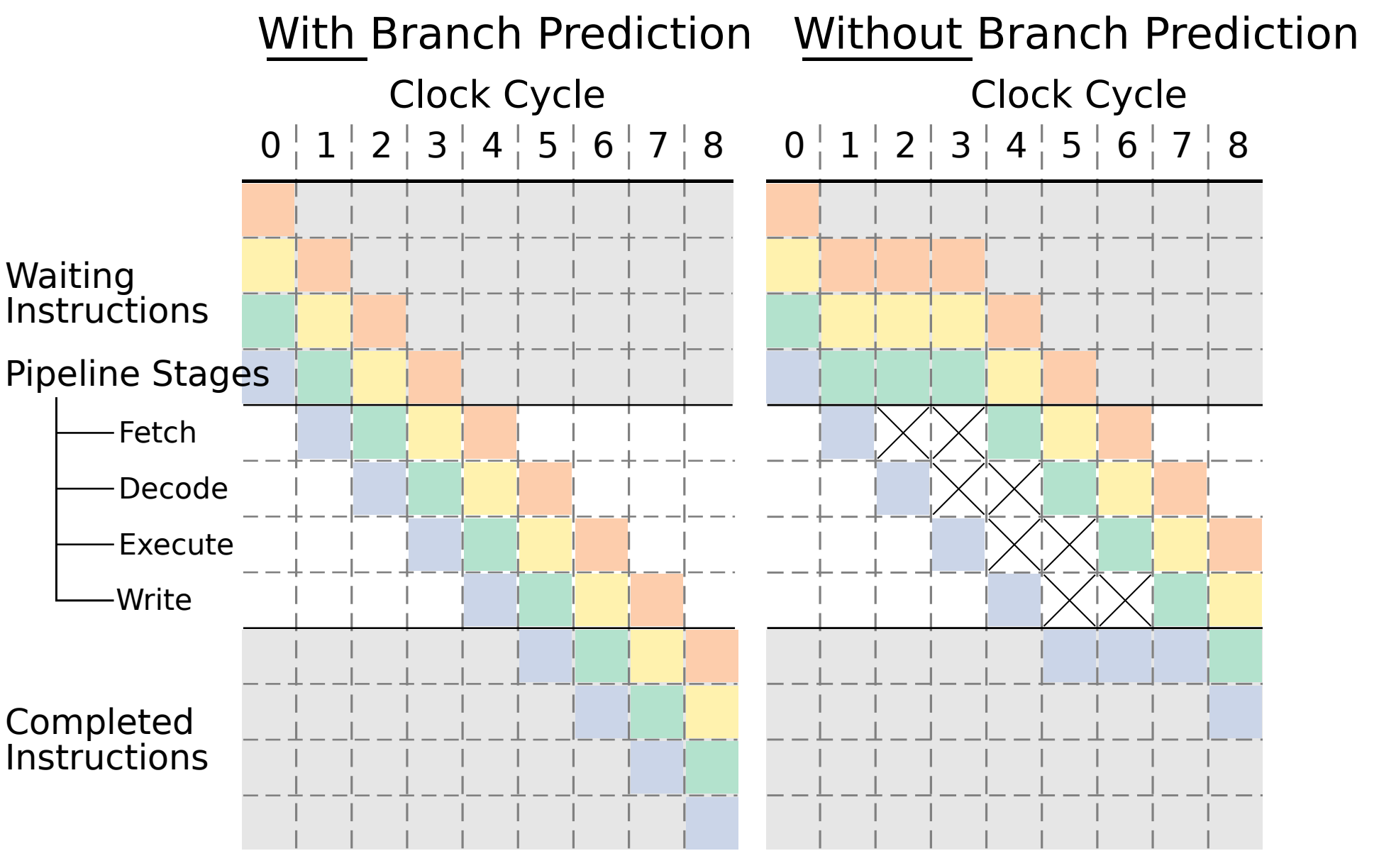

With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

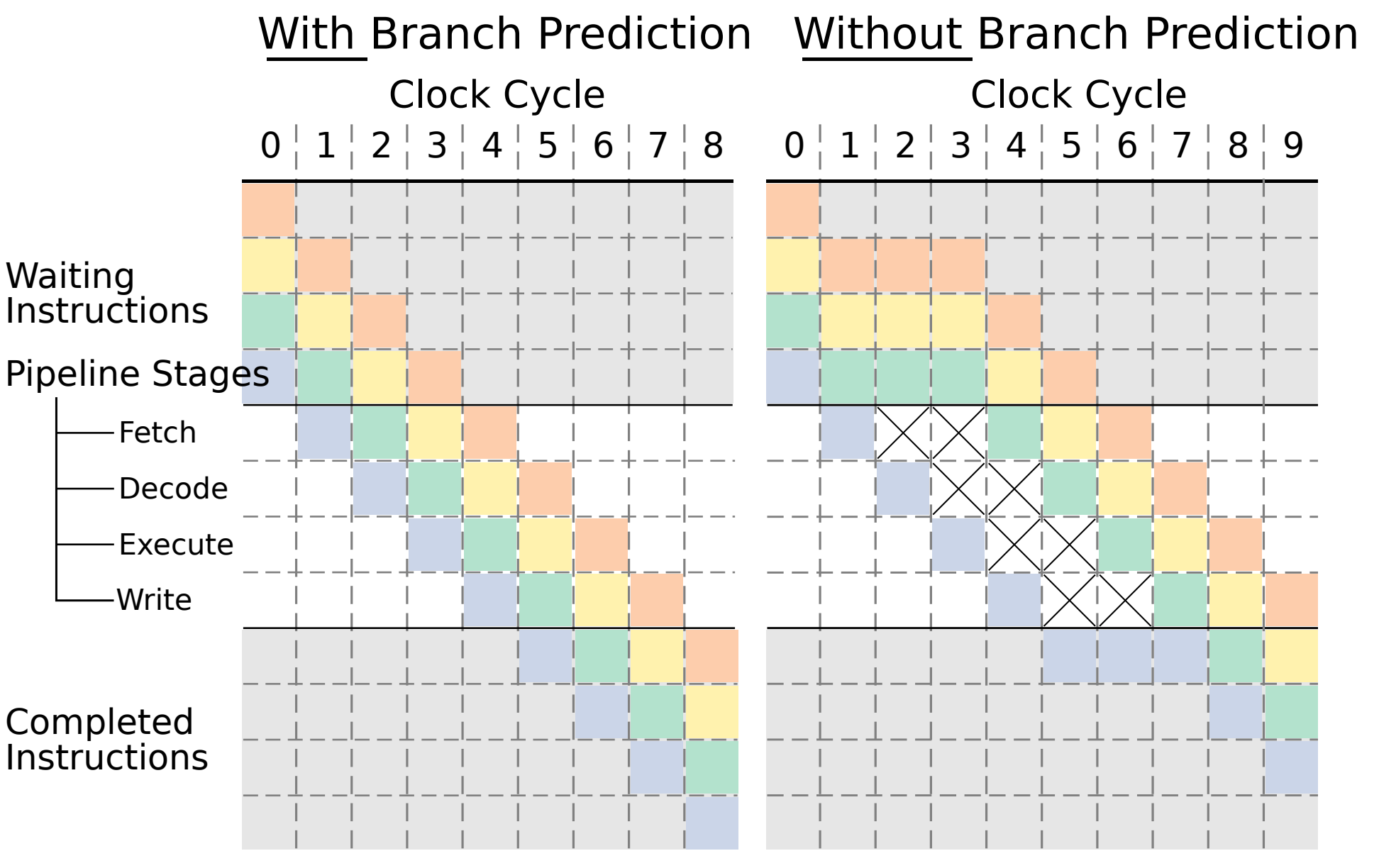

With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

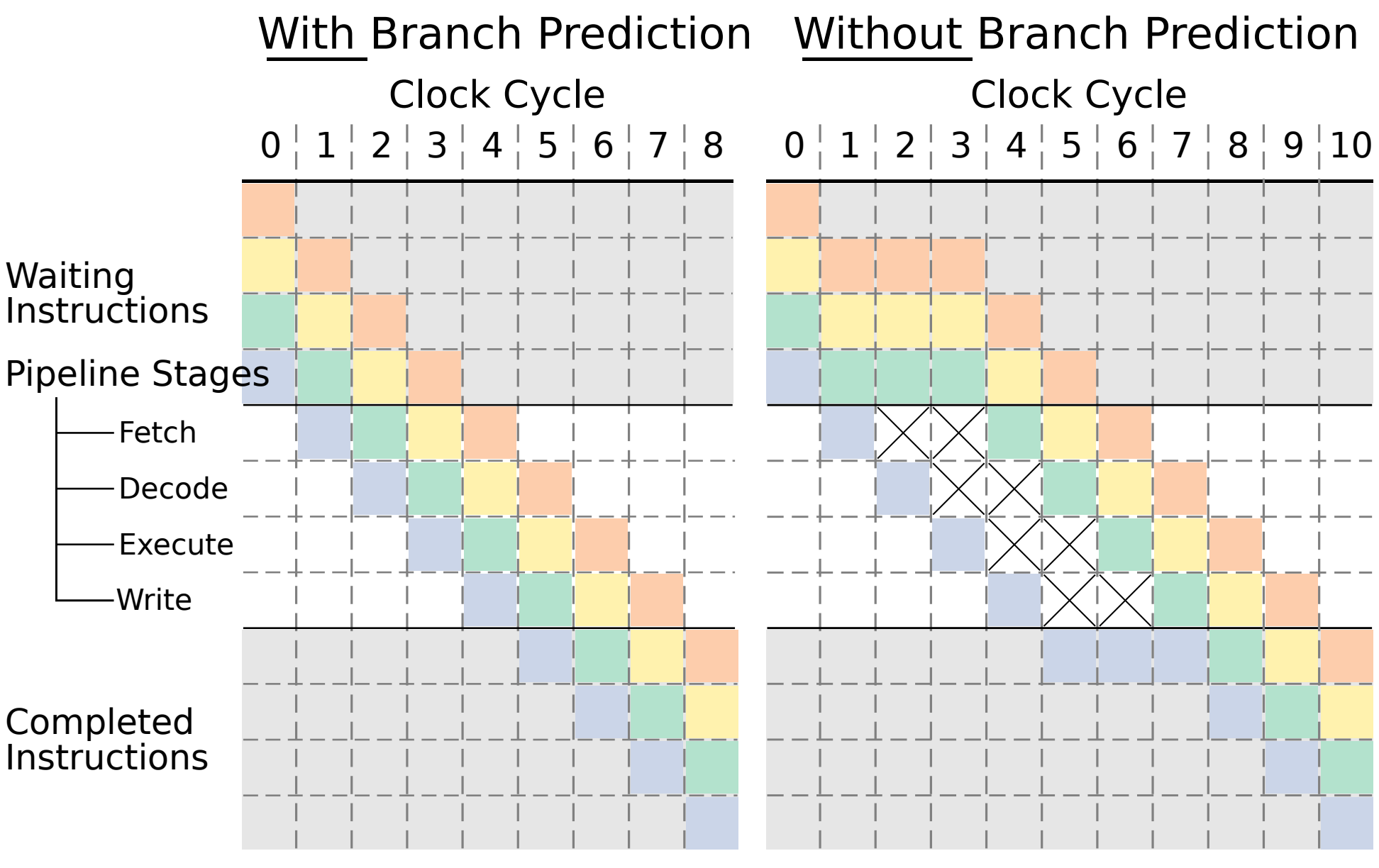

With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions


With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions


With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions


With Branch Prediction
Without Branch Prediction
Clock Cycle
0 1 2 3 4 5 6 7 8
Clock Cycle
0 1 2 3 4 5 6 7 8
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions


With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions


With Branch Prediction
Without Branch Prediction
Clock Cycle
Clock Cycle
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9 10
Waiting Instructions
Pipeline Stages
Fetch
Decode
Execute
Write
Completed Instructions

Performance Pitfalls

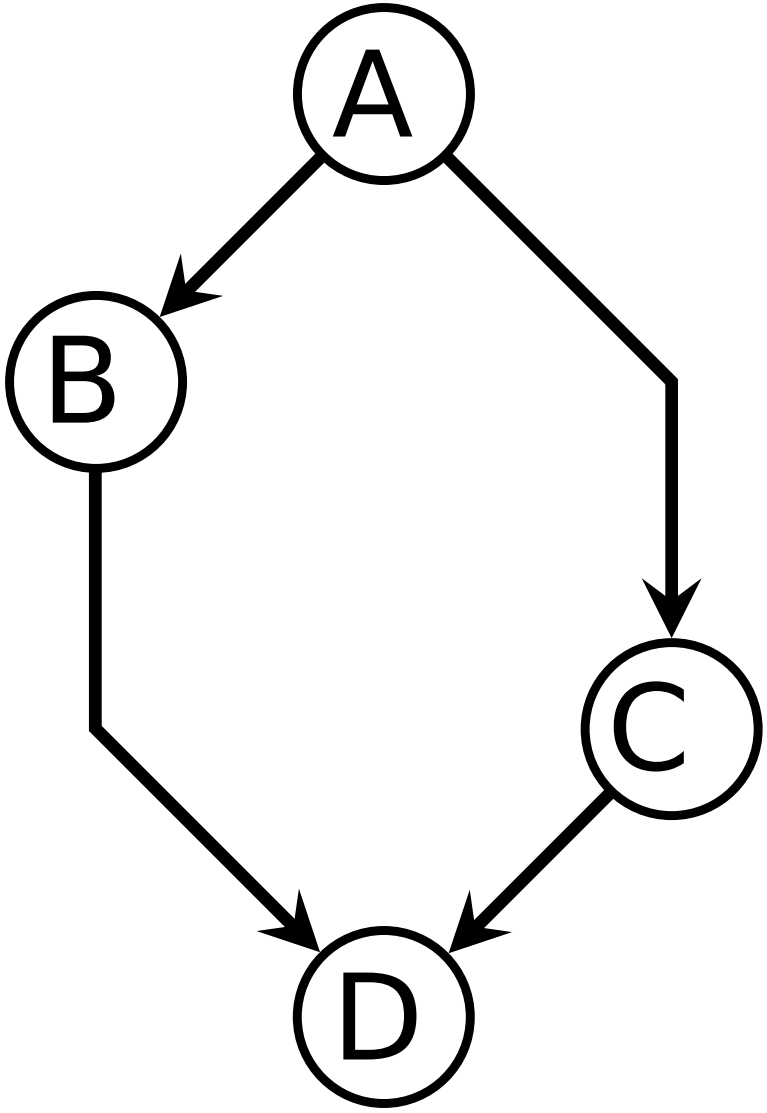
Pipeline Stall - execution delay in an instruction pipeline to resolve a dependency

Performance Pitfalls

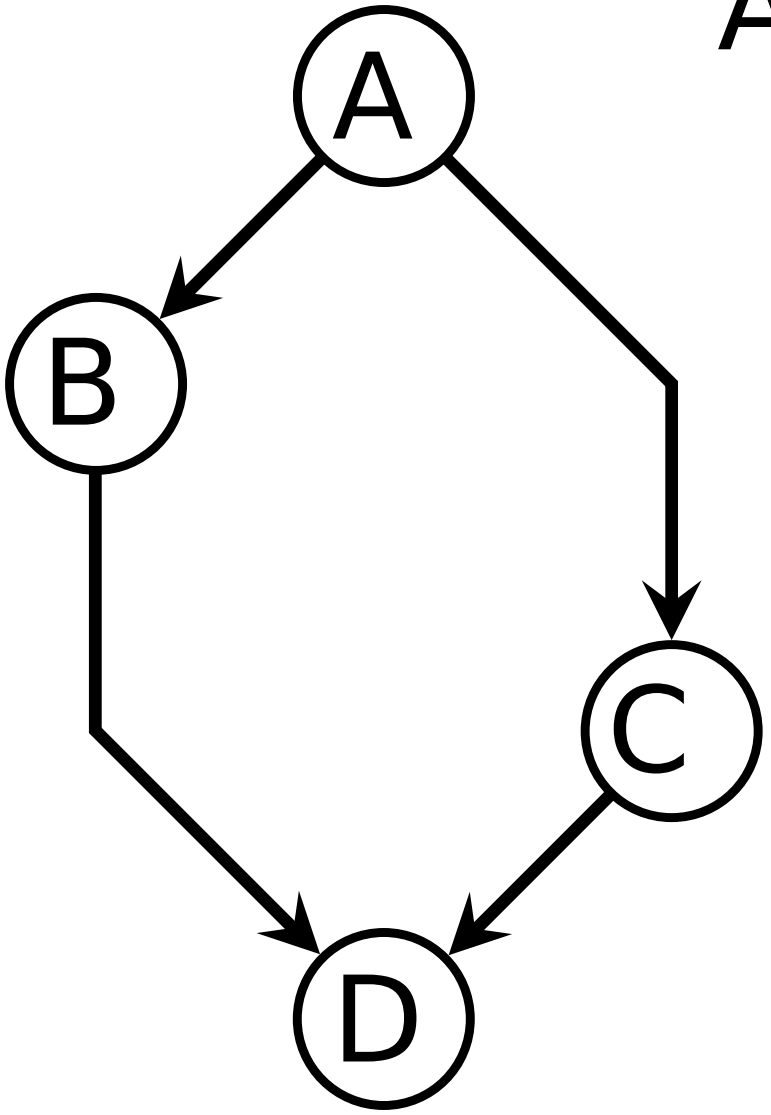
Pipeline Stall - execution delay in an instruction pipeline to resolve a dependency

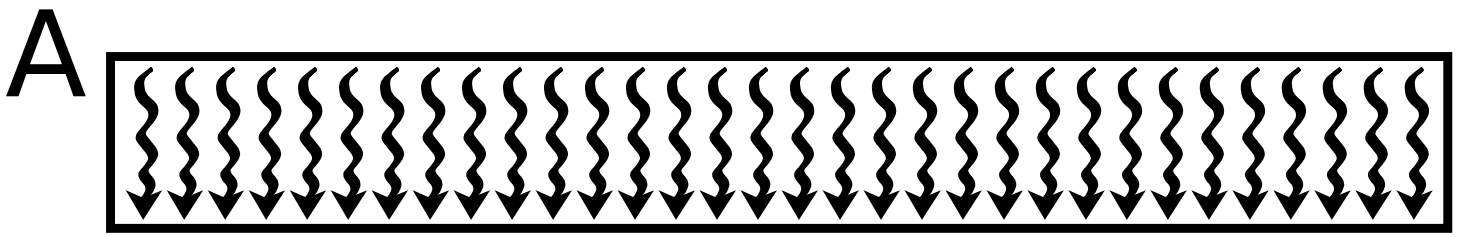
Warp Divergence - threads within a warp take different paths and the different execution paths are serialized

Warp Divergence Example

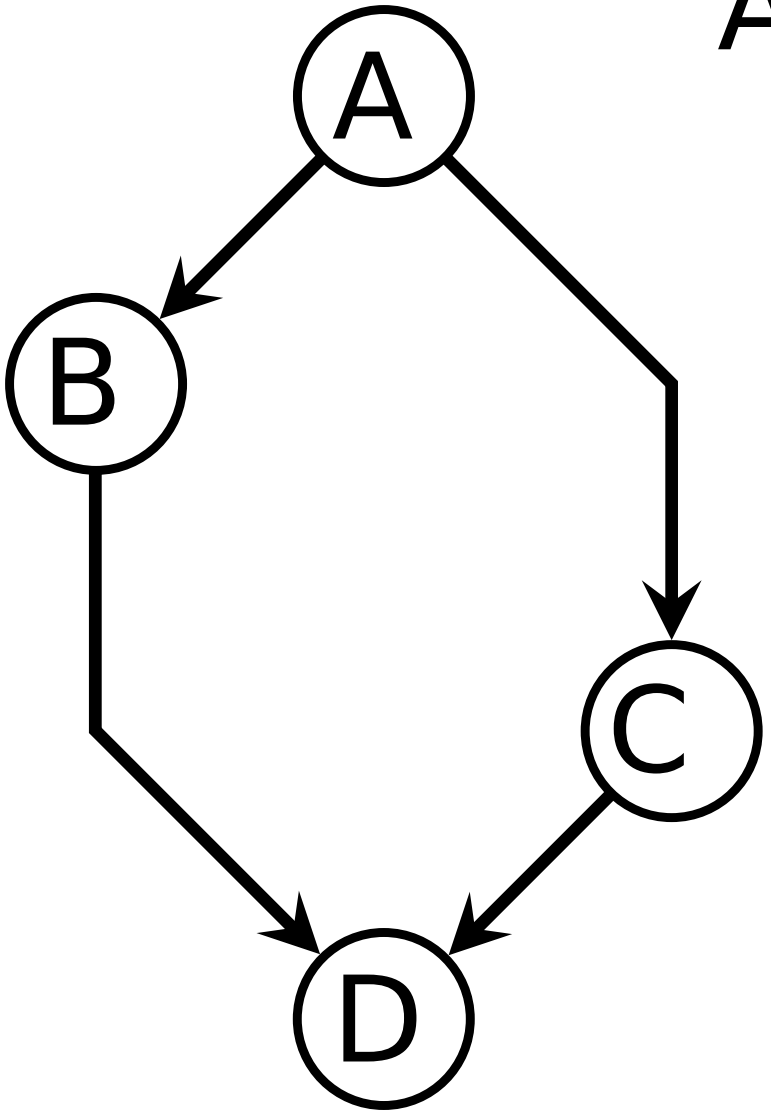


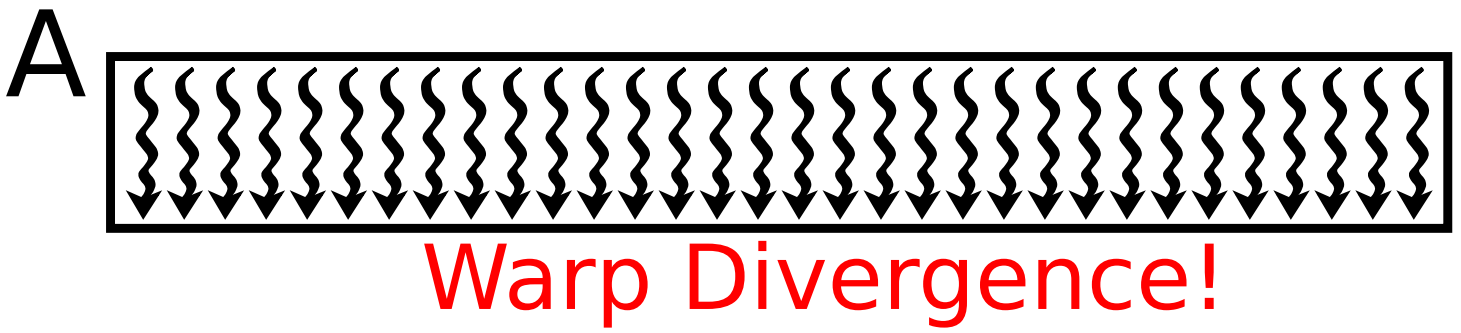
Warp Divergence Example



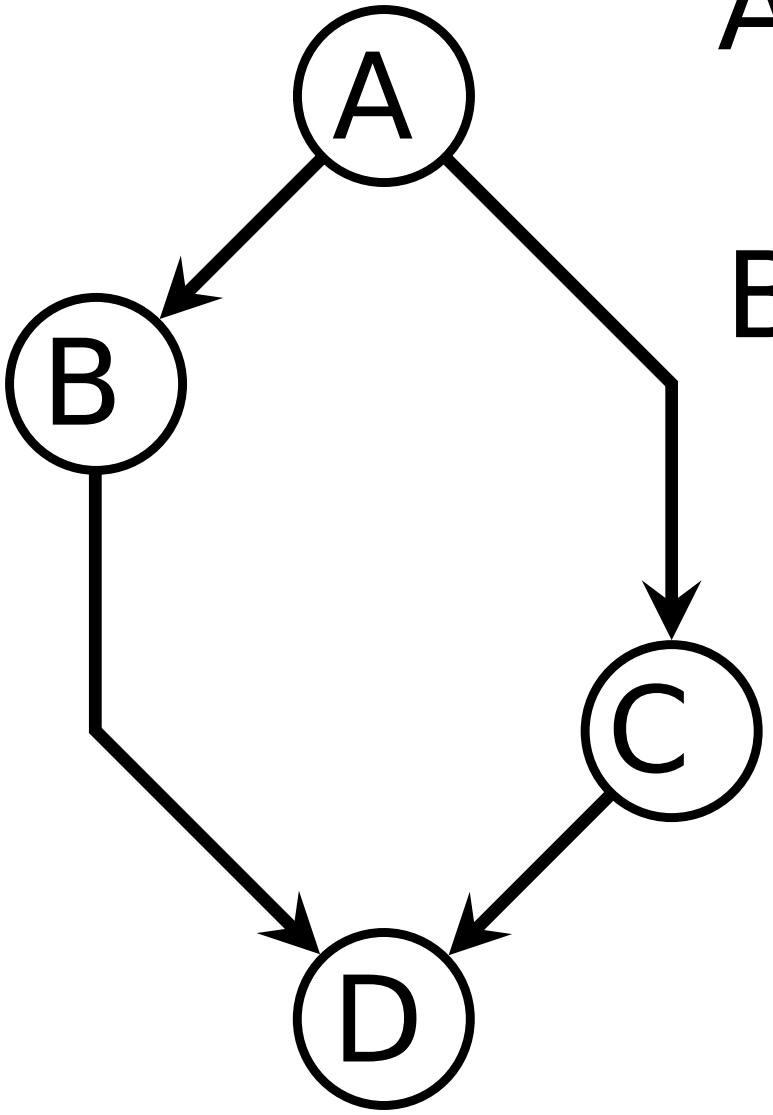


Warp Divergence Example

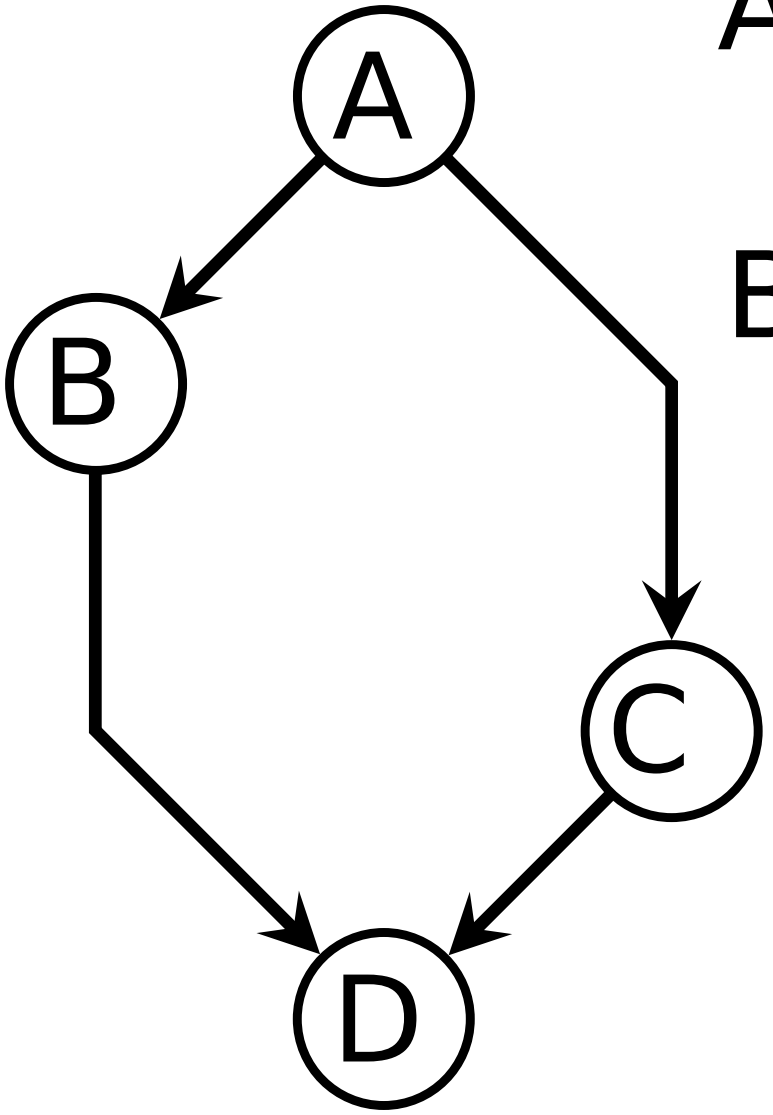




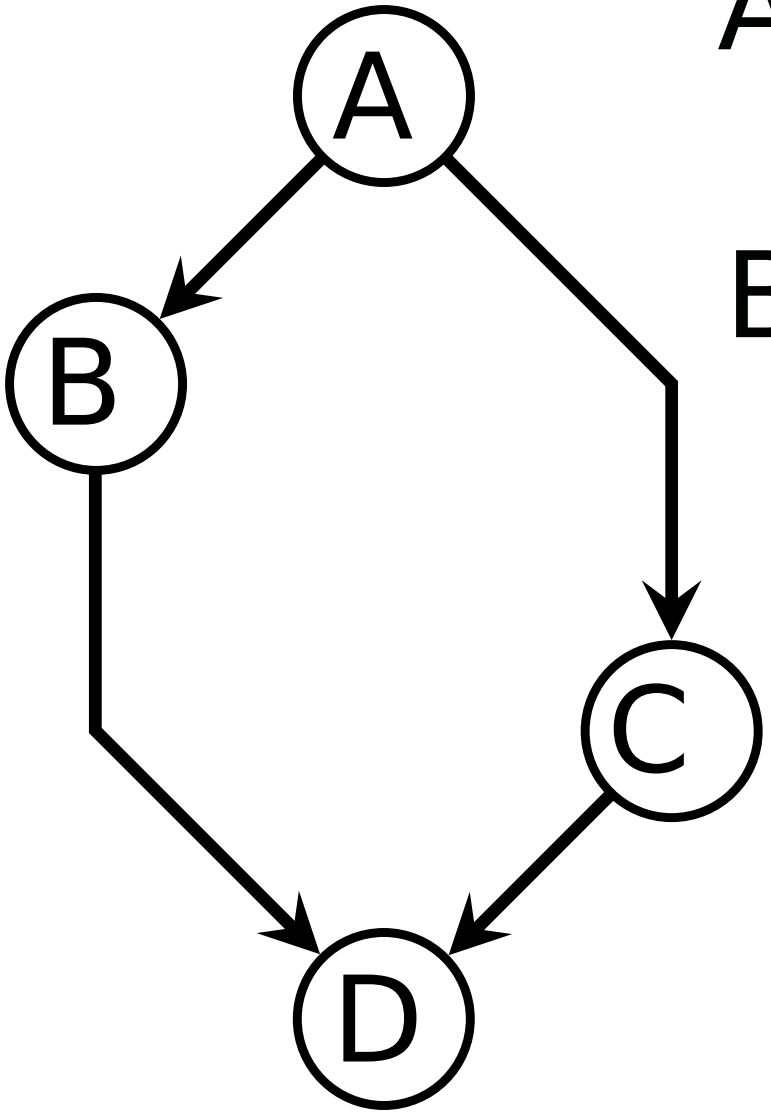
Warp Divergence Example

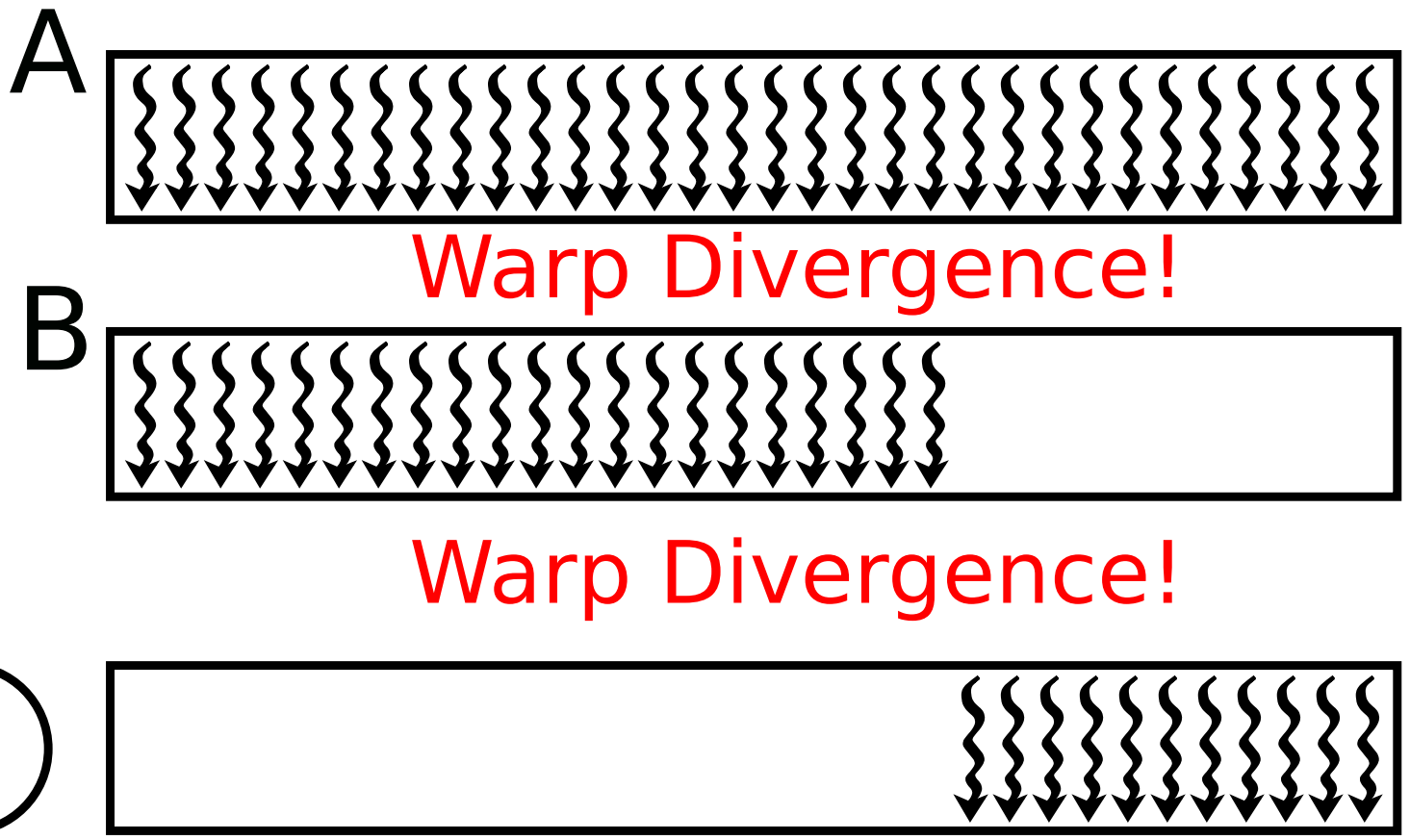


Warp Divergence Example

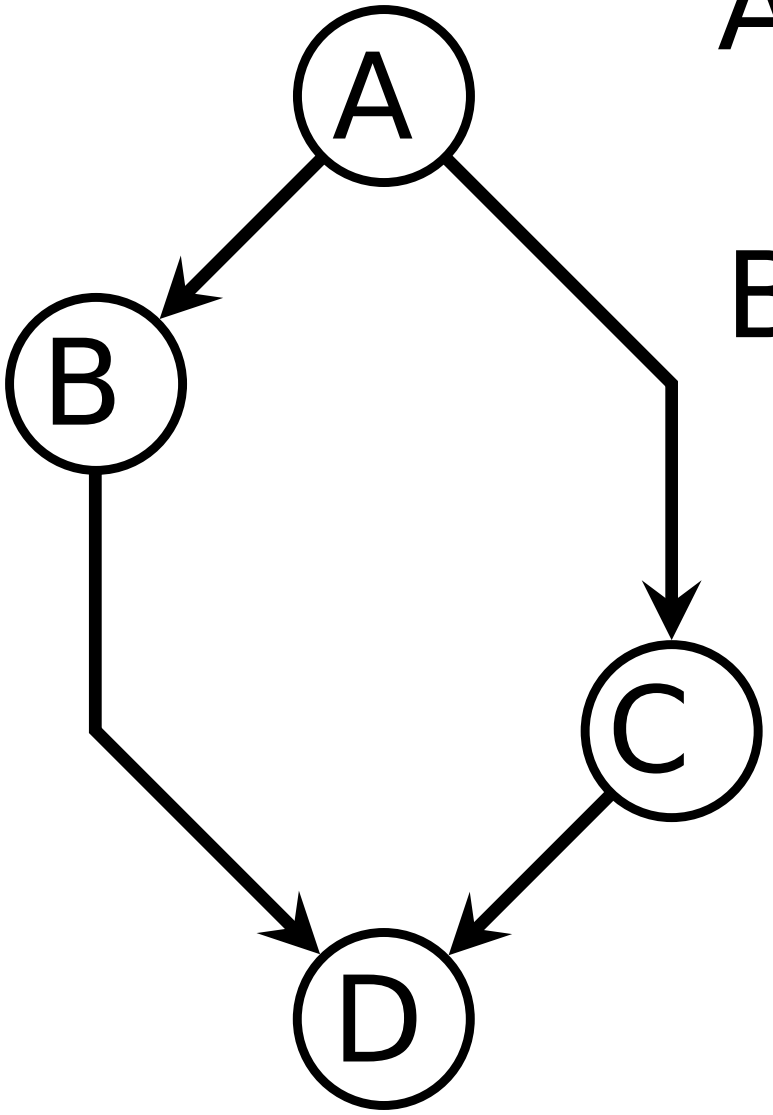


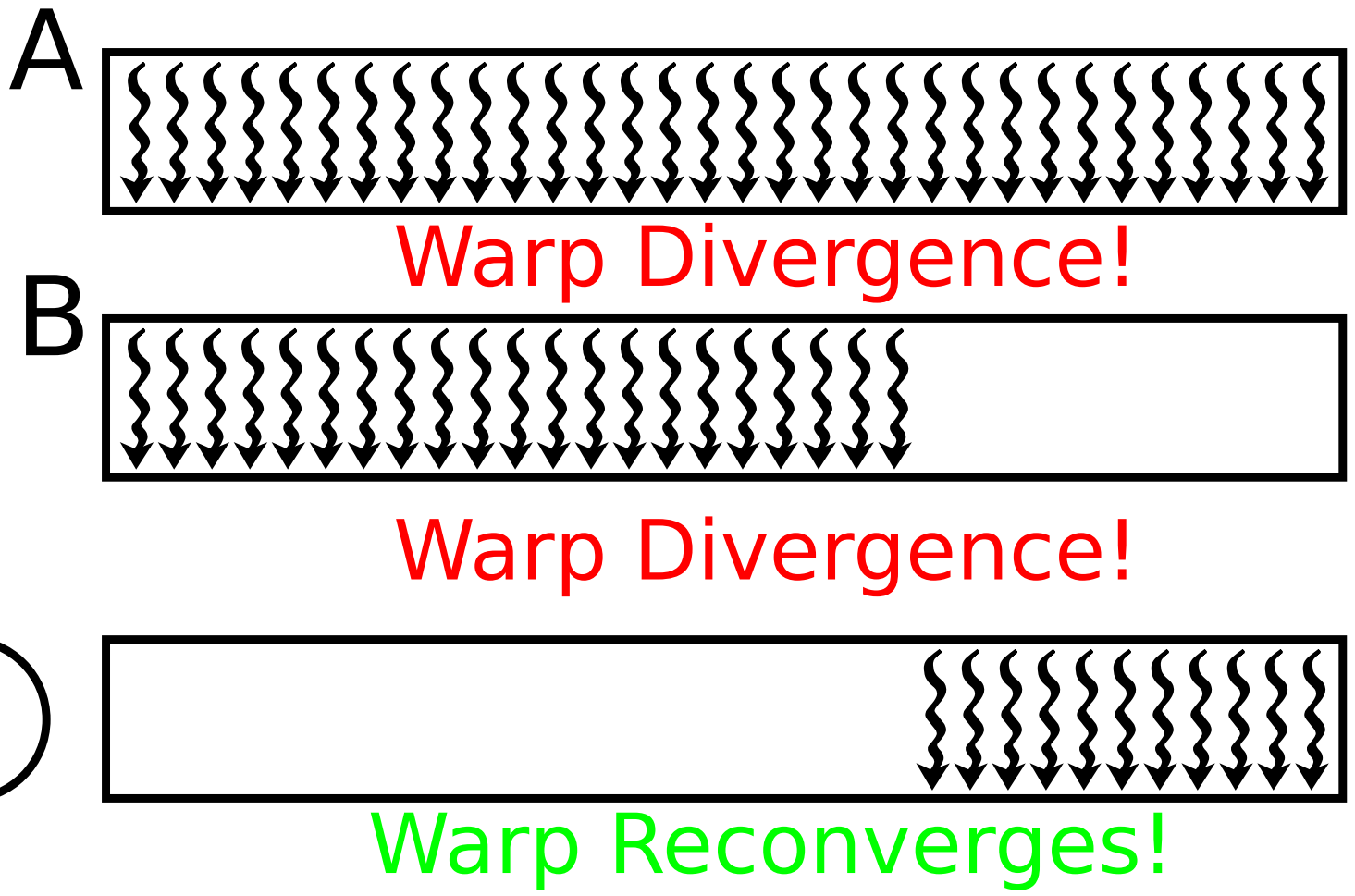
Warp Divergence Example



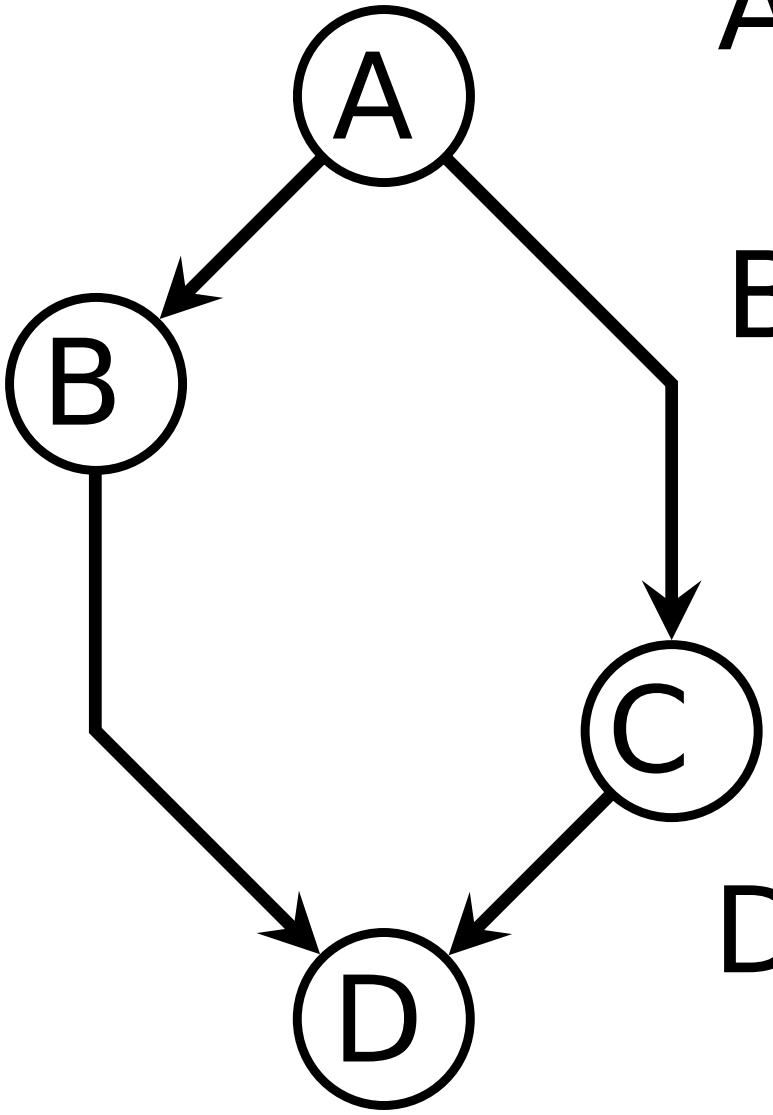


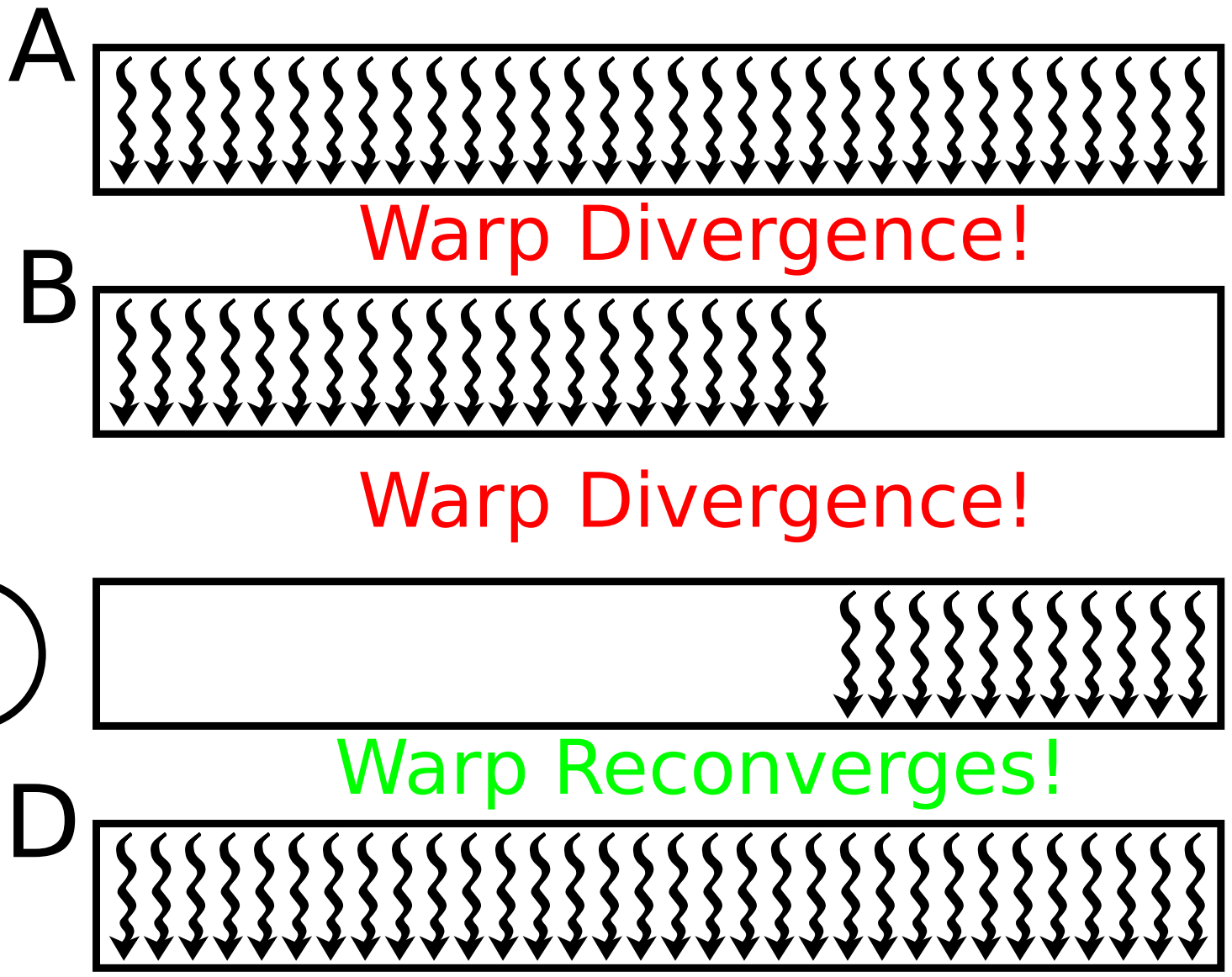
Warp Divergence Example





Warp Divergence Example





Warp-Aware Trace Scheduling

Schedule instructions across basic block boundaries to expose additional ILP...

Warp-Aware Trace Scheduling

Schedule instructions across basic block boundaries to expose additional ILP...

while managing and optimizing warp divergence.

Origins: Microcode Trace Scheduling

...generalizing local and disparate vertical-to-horizontal microcode compaction

Step	Description

Origins: Microcode Trace Scheduling

...generalizing local and disparate vertical-to-horizontal microcode compaction

Step	Description
1. Trace Selection	

Origins: Microcode Trace Scheduling

...generalizing local and disparate vertical-to-horizontal microcode compaction

Step	Description
1. Trace Selection	
2. Trace Formation	

Origins: Microcode Trace Scheduling

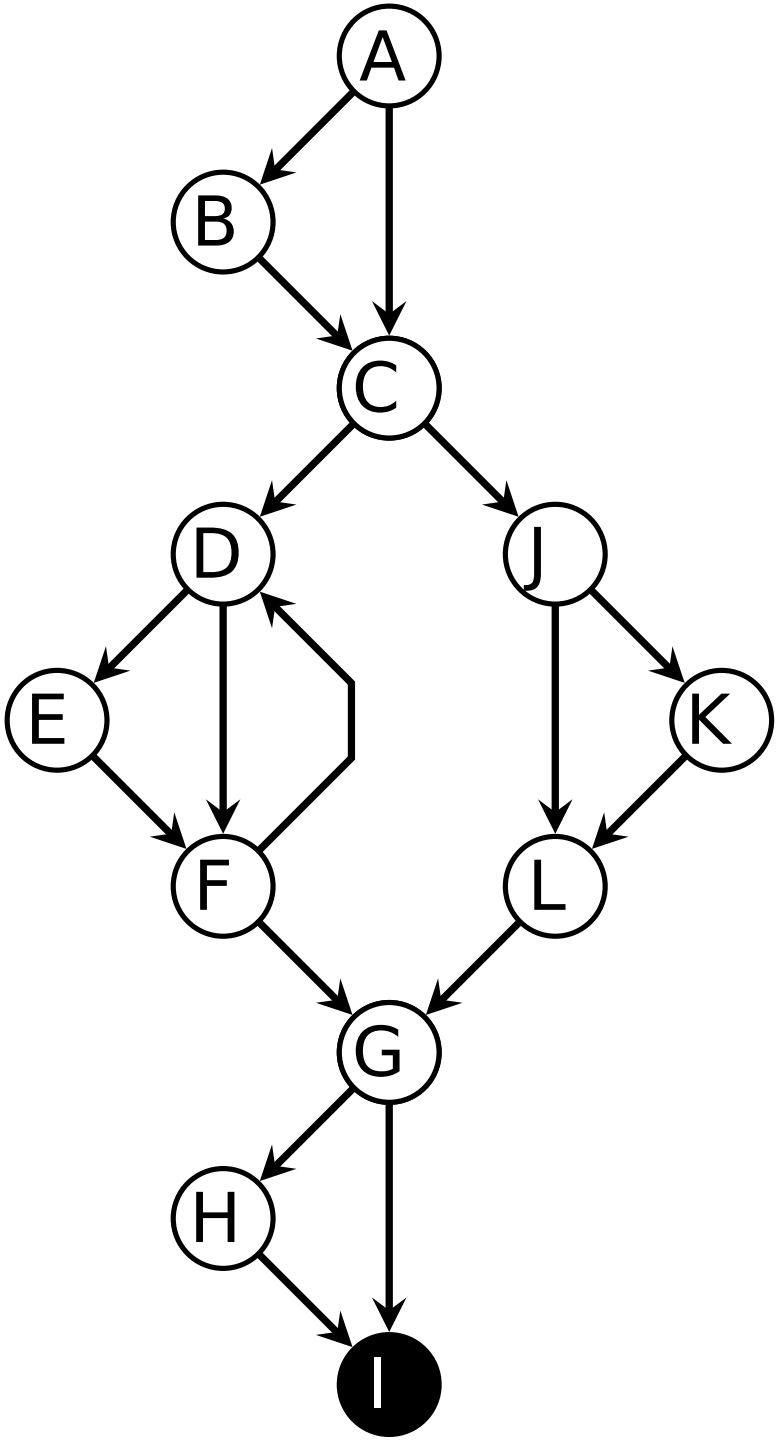
...generalizing local and disparate vertical-to-horizontal microcode compaction

Step	Description
1. Trace Selection	
2. Trace Formation	
3. Local Scheduling	

Origins: Microcode Trace Scheduling

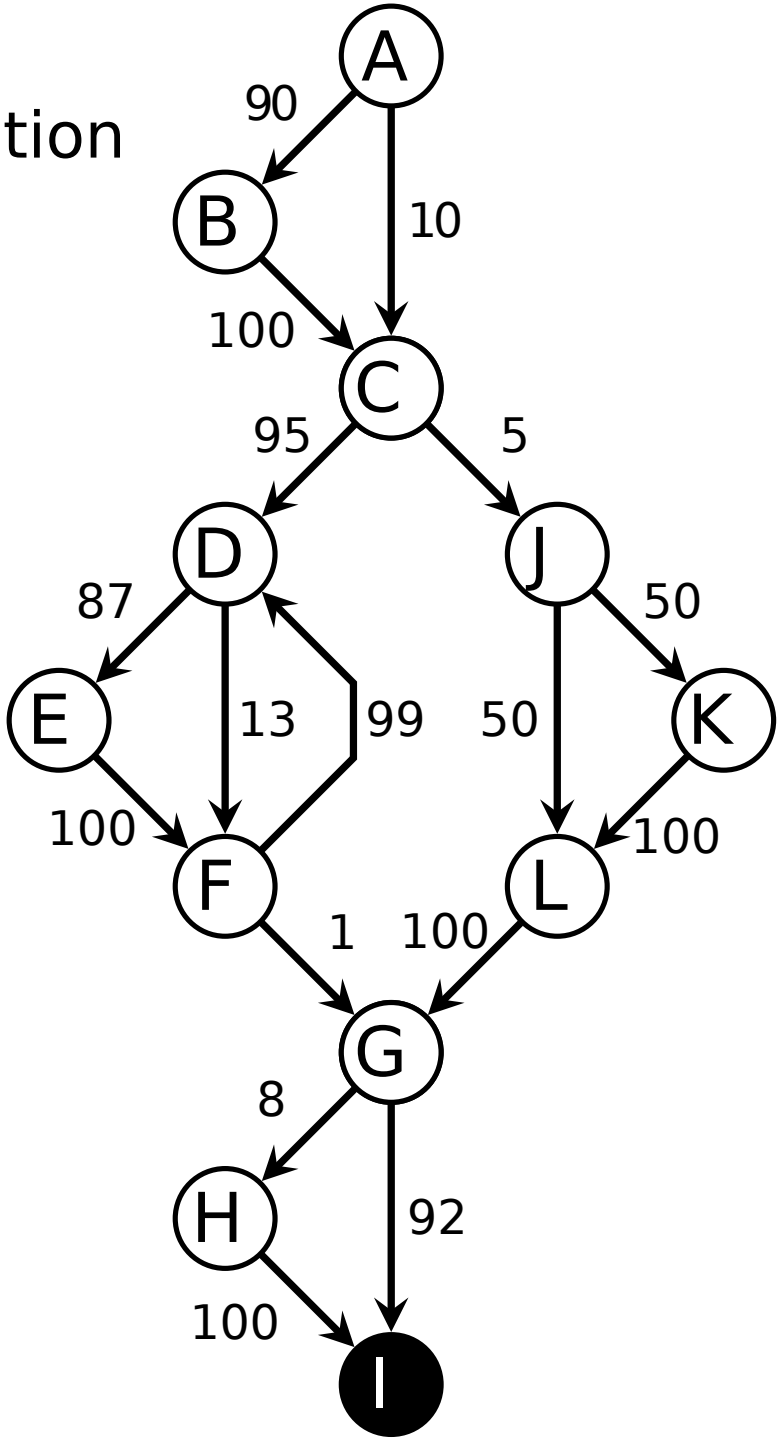
...generalizing local and disparate vertical-to-horizontal microcode compaction

Step	Description
1. Trace Selection	partition basic blocks into regions
2. Trace Formation	facilitate local scheduling, potentially adding nodes and edges
3. Local Scheduling	schedule instructions within each region


A
B
C
D
J
E
K
F
L
G
H
I

Annotate CFG

- dynamic profiling
- static branch prediction



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

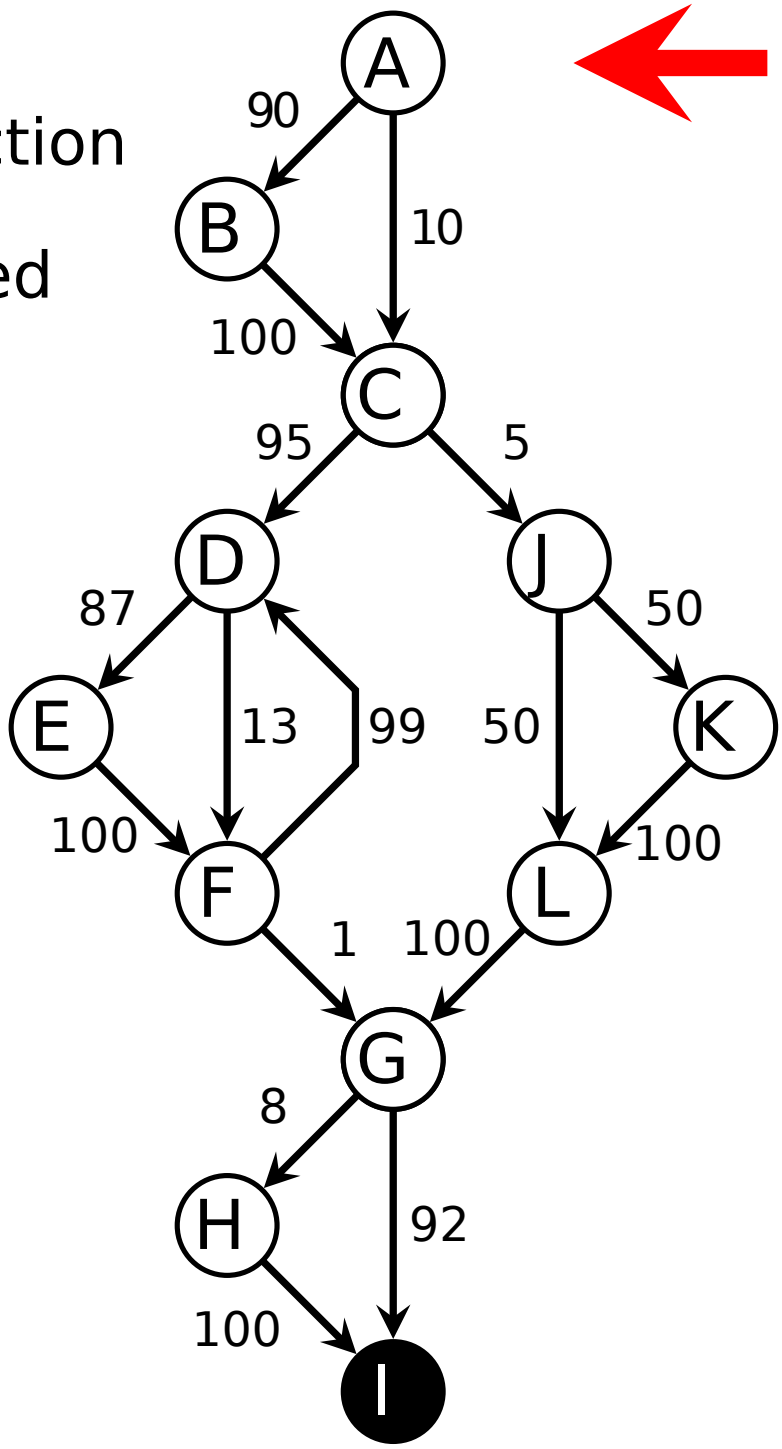
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

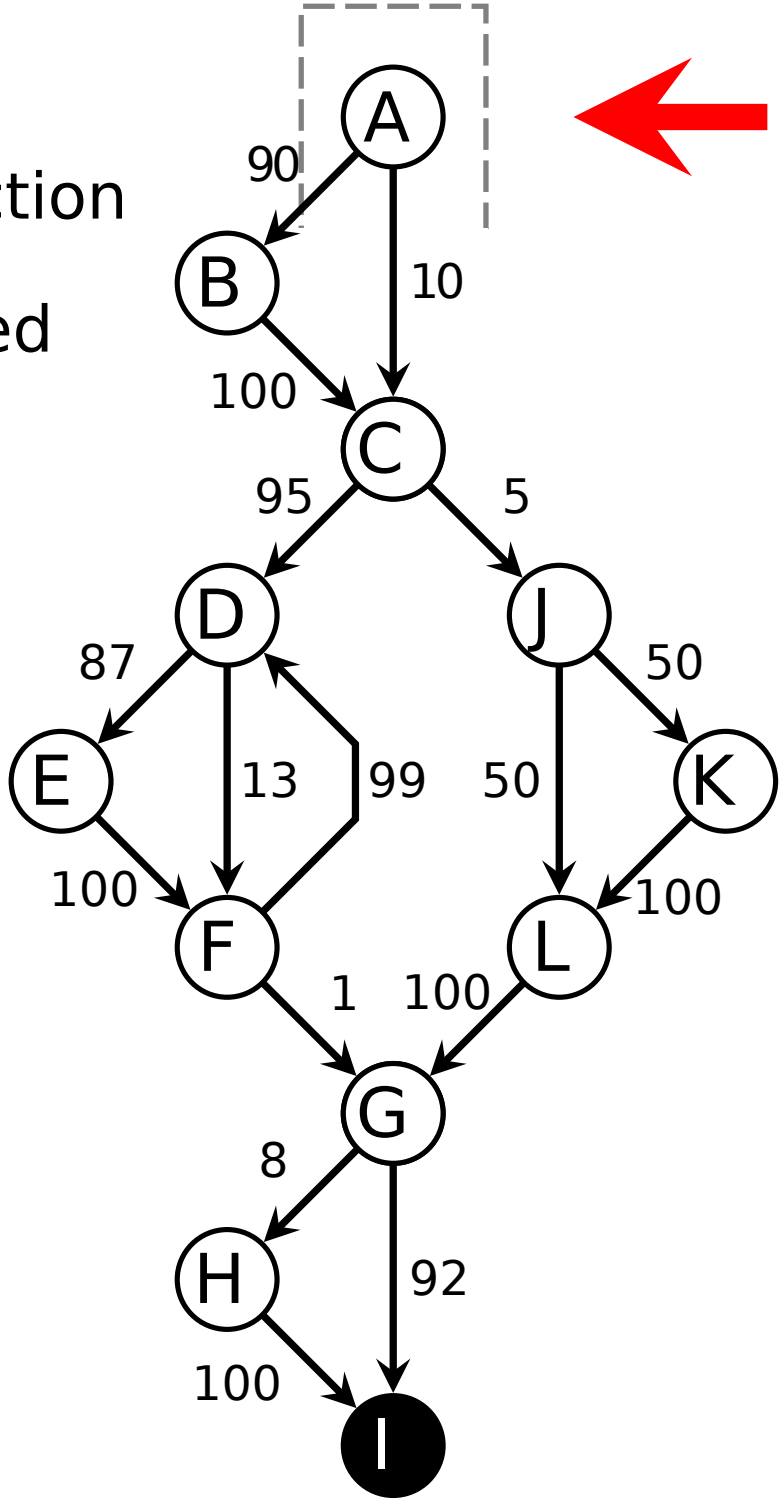
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

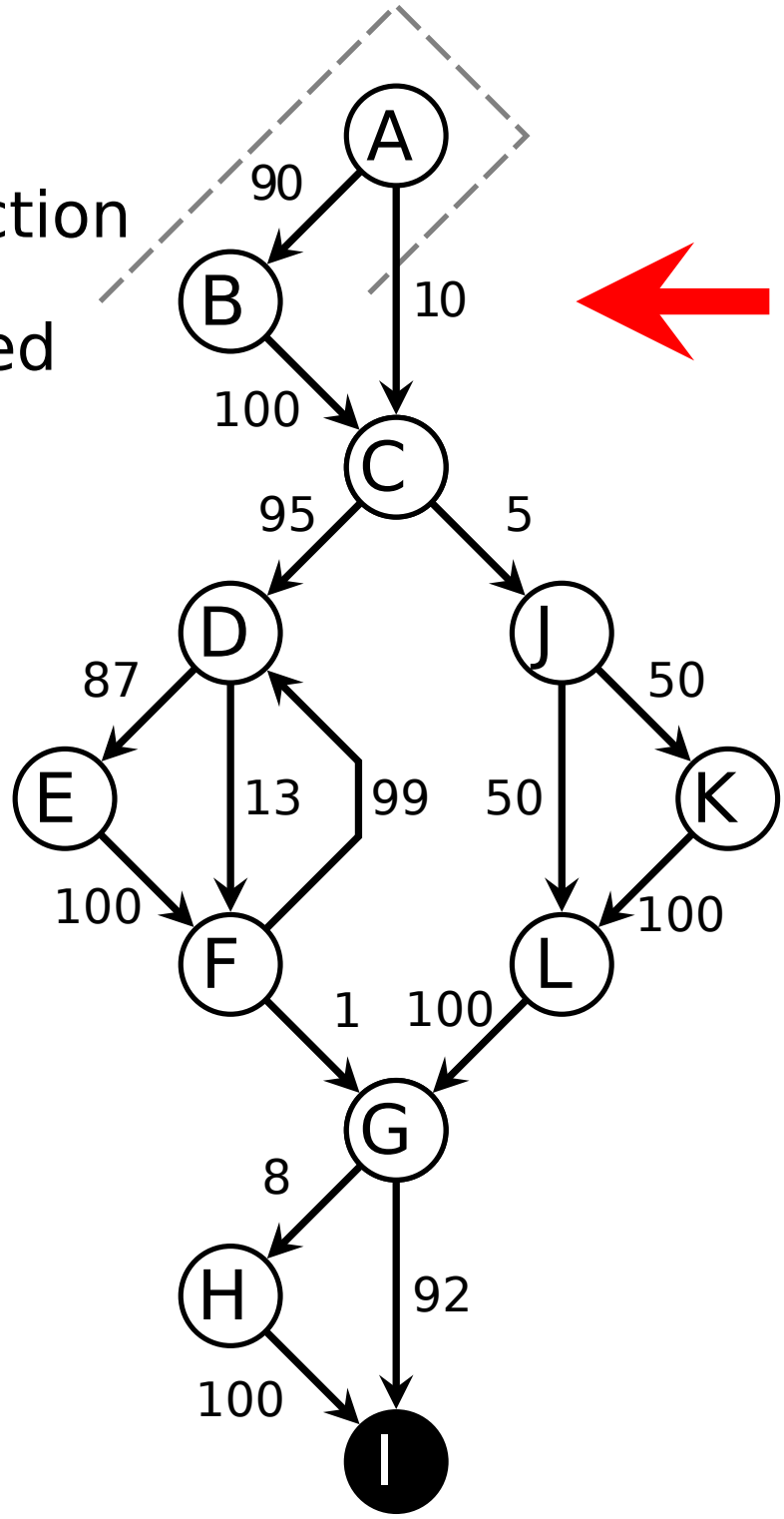
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

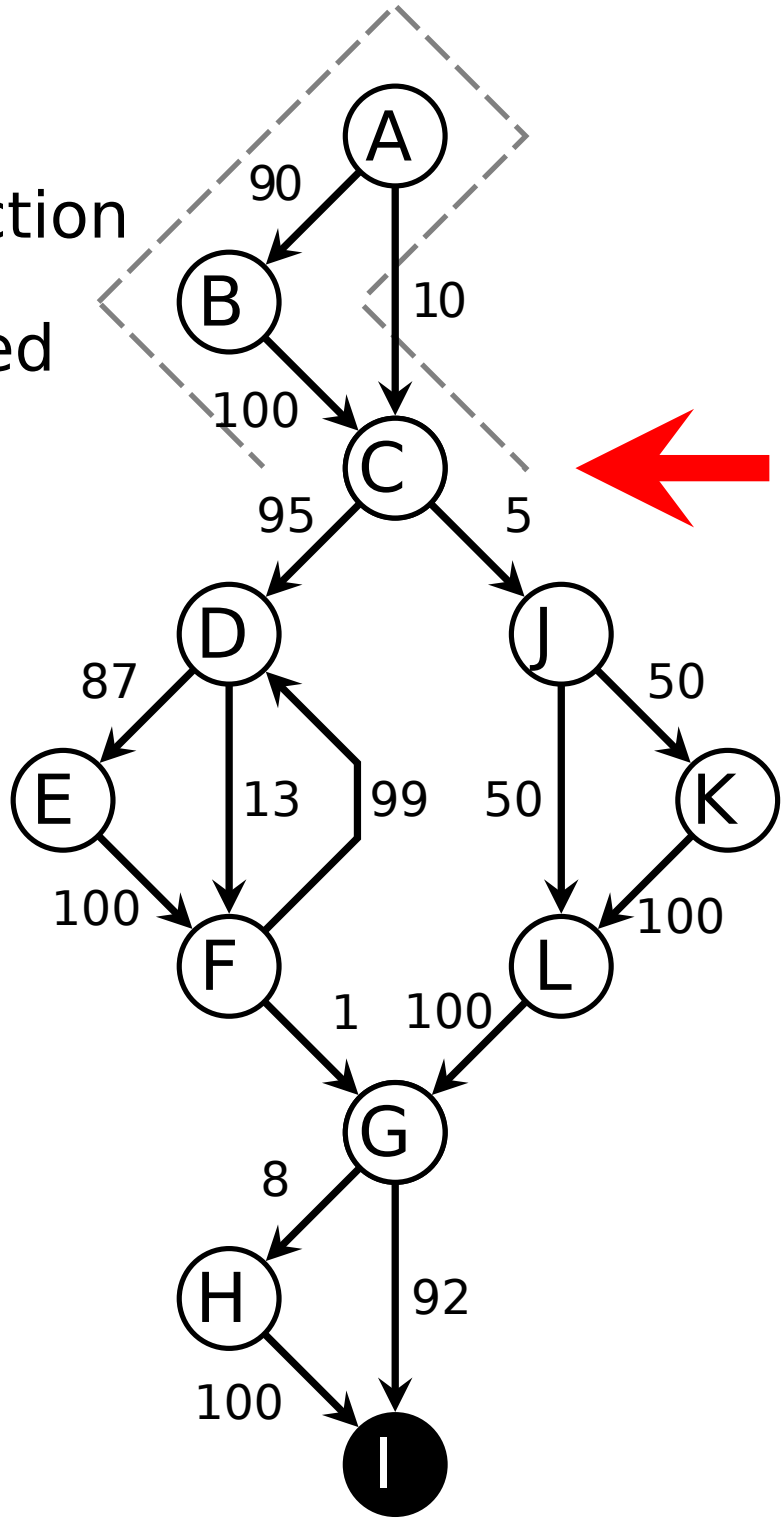
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

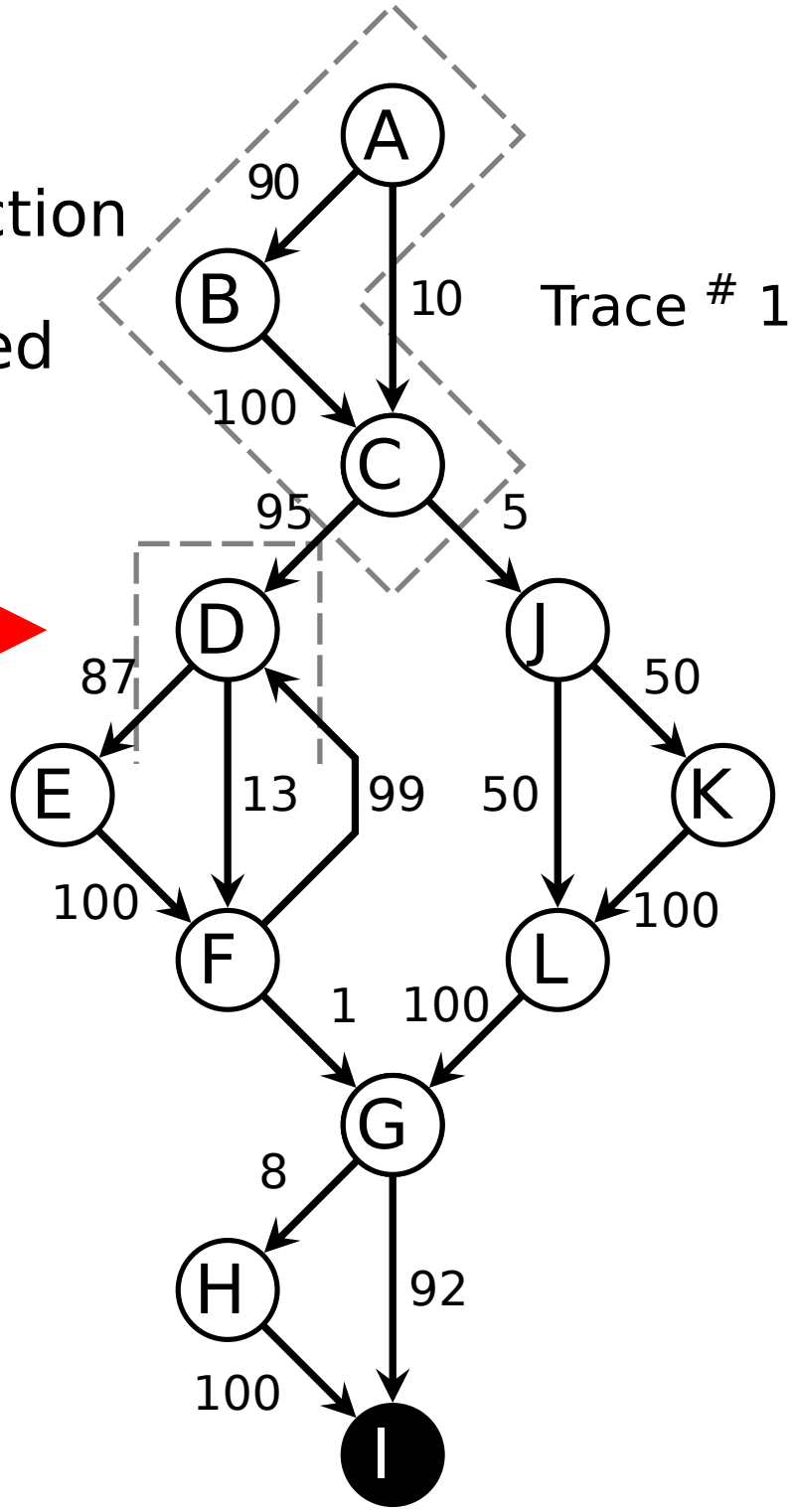
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

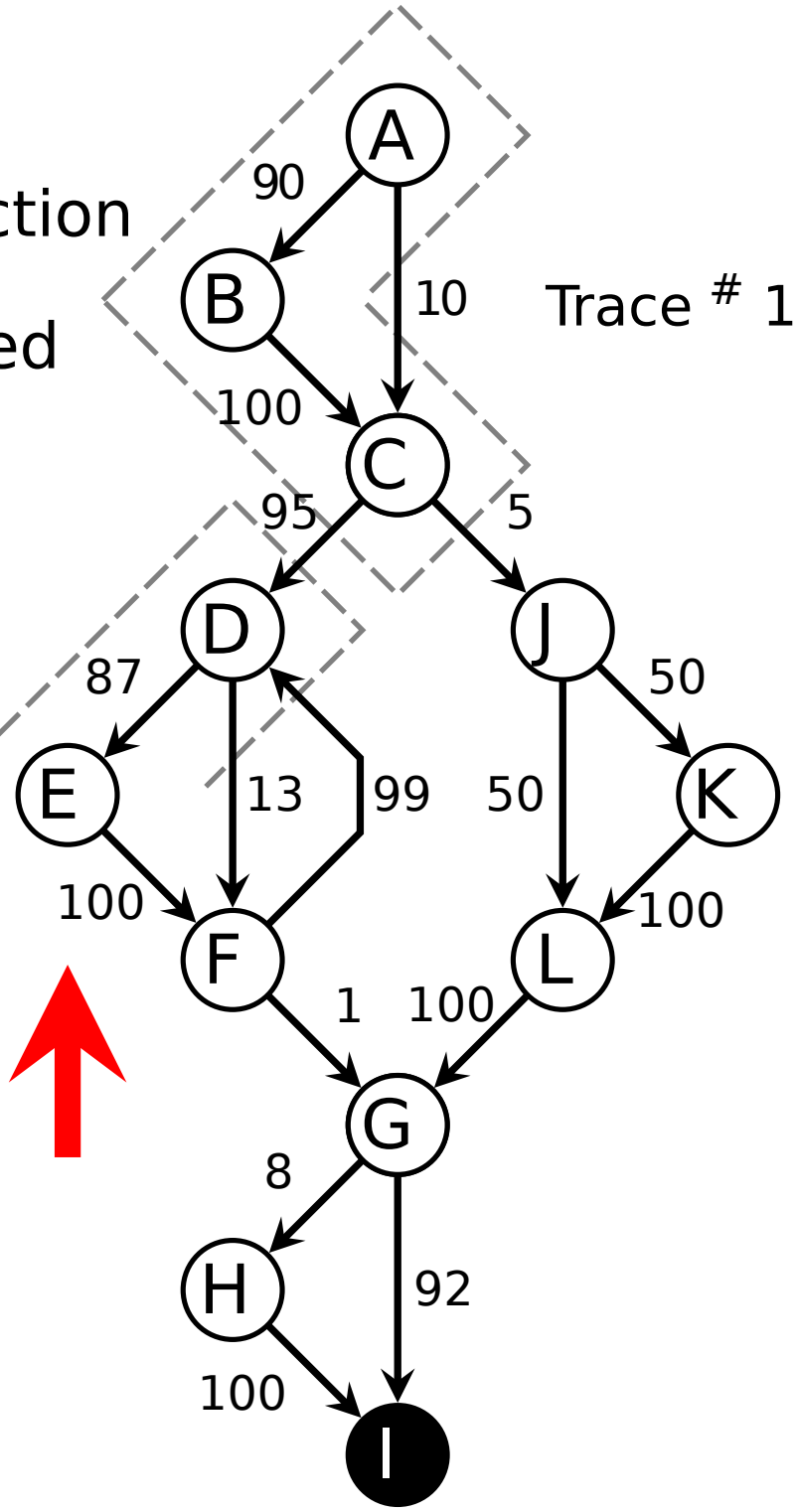
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

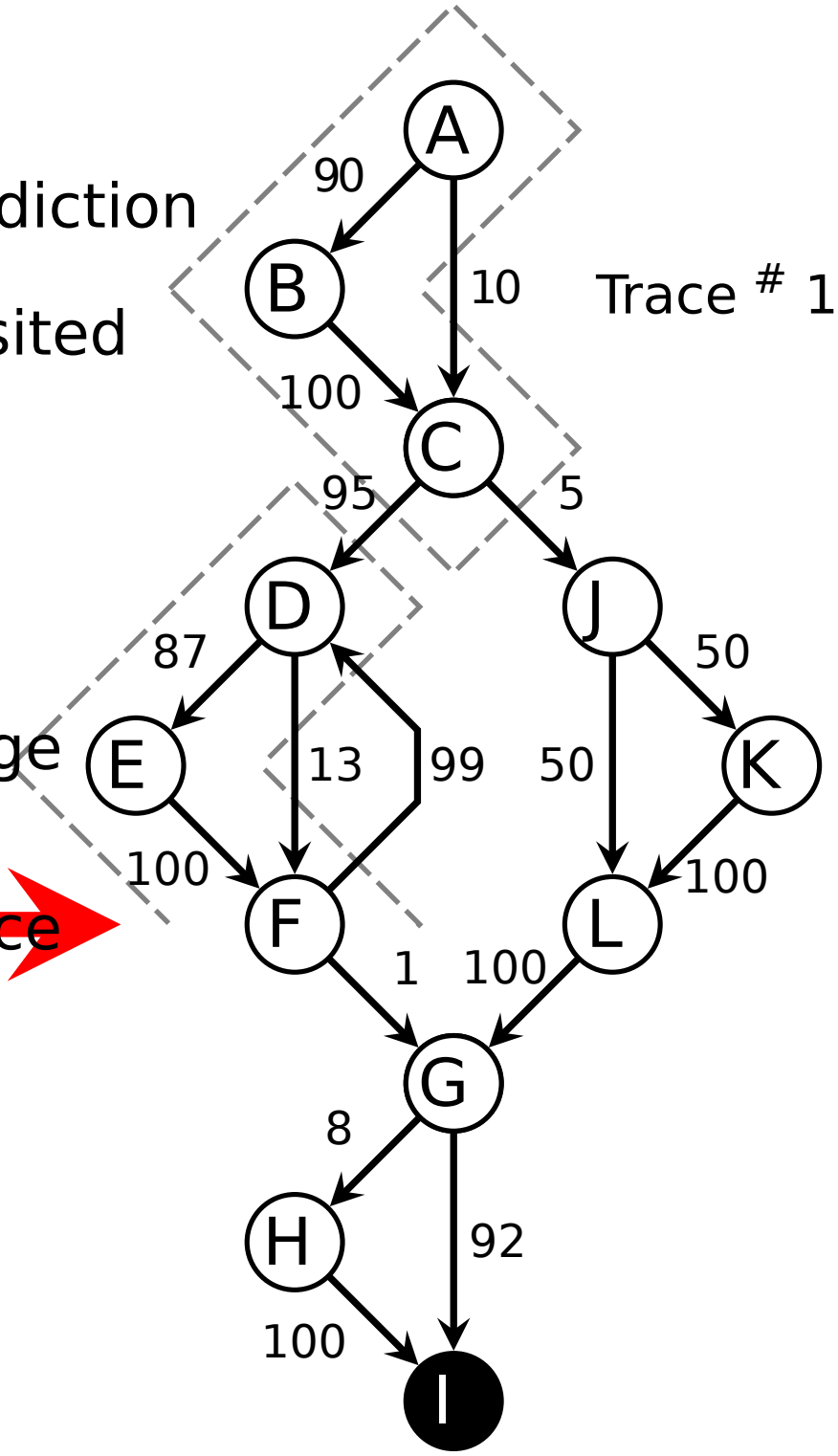
loop

Find the next unvisited node, with highest edge weight

Add node to trace

end loop

end while



Annotate CFG
- dynamic profiling
- static branch prediction

while there are unvisited nodes

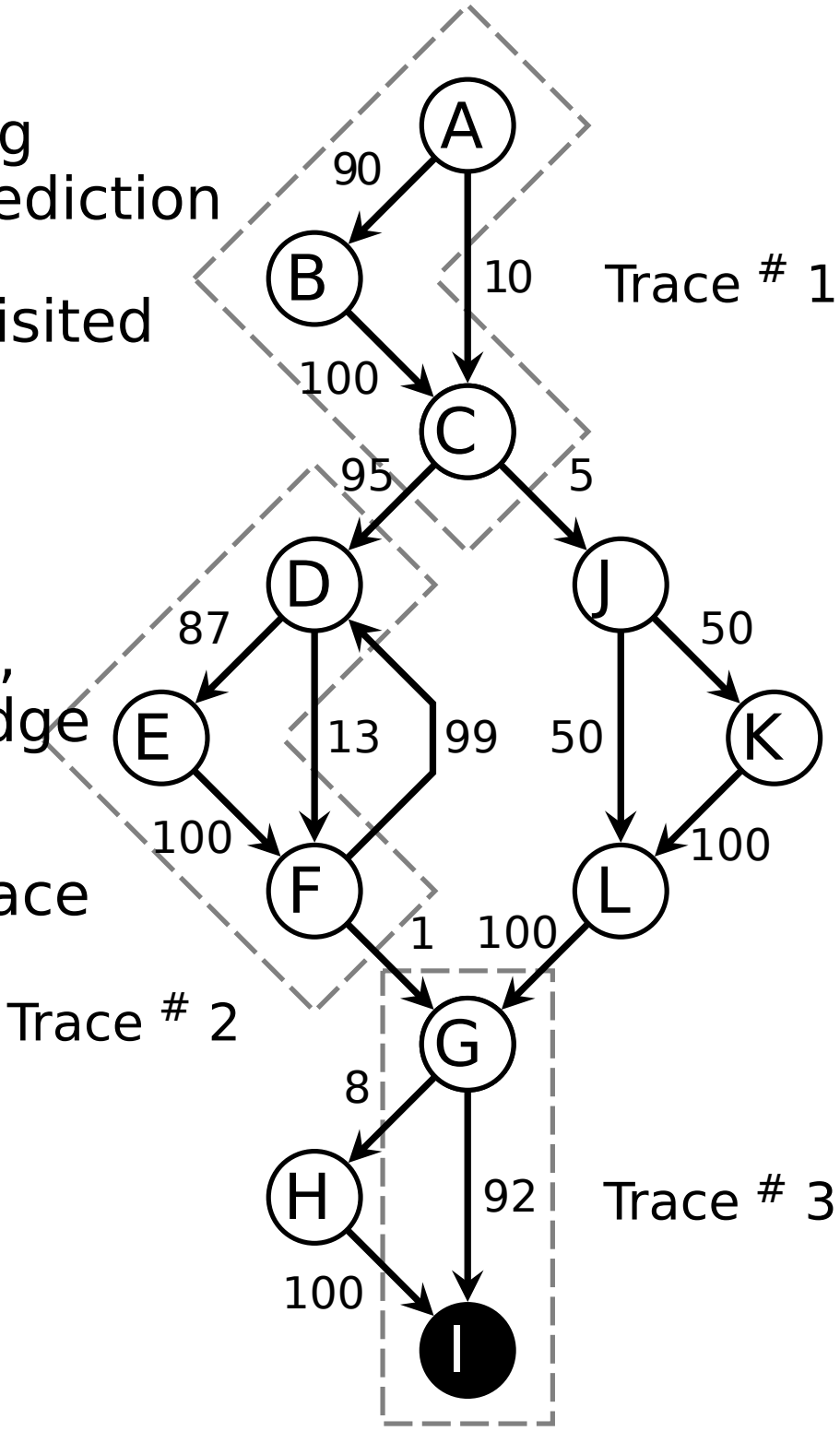
loop

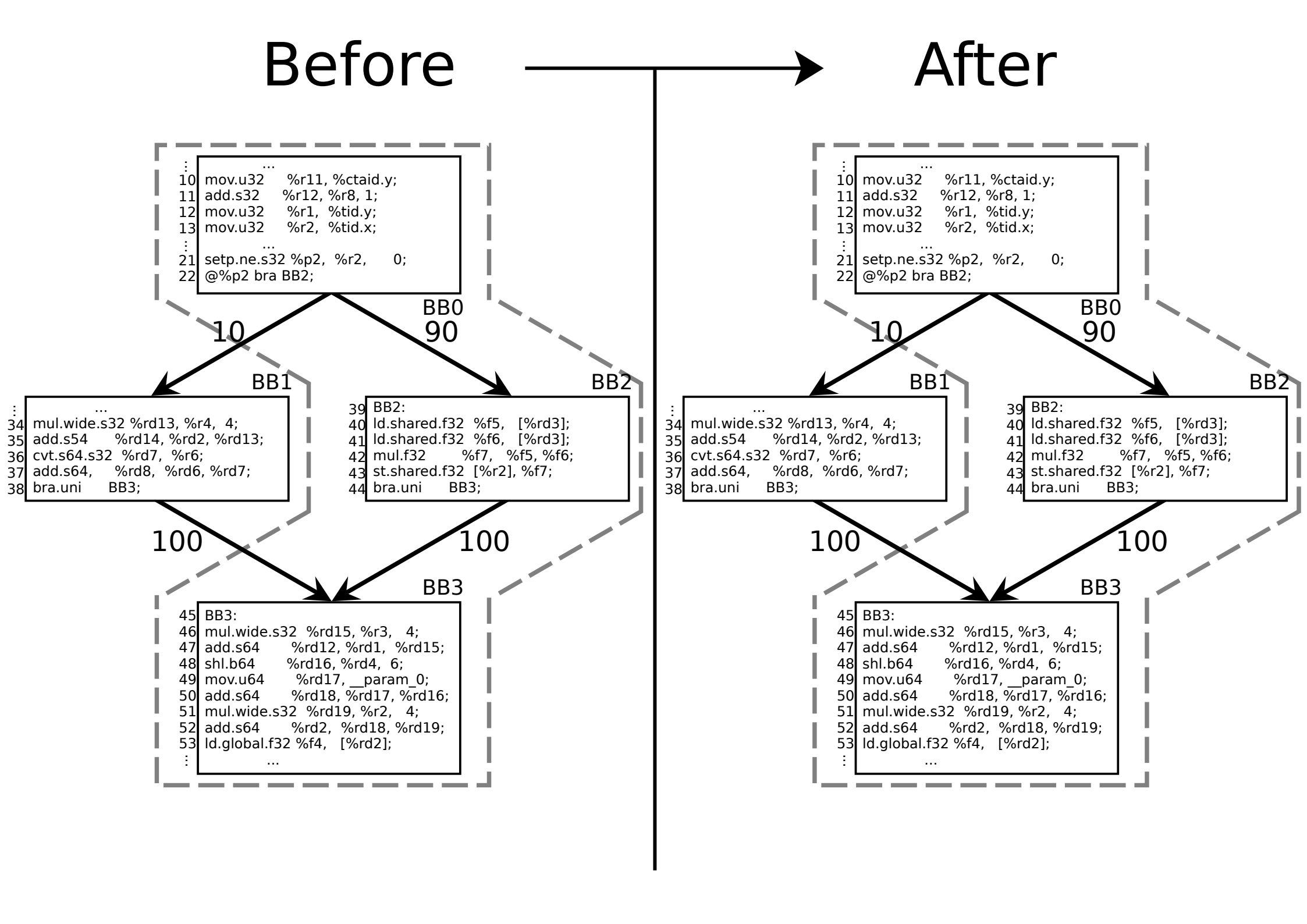
Find the next unvisited node, with highest edge weight

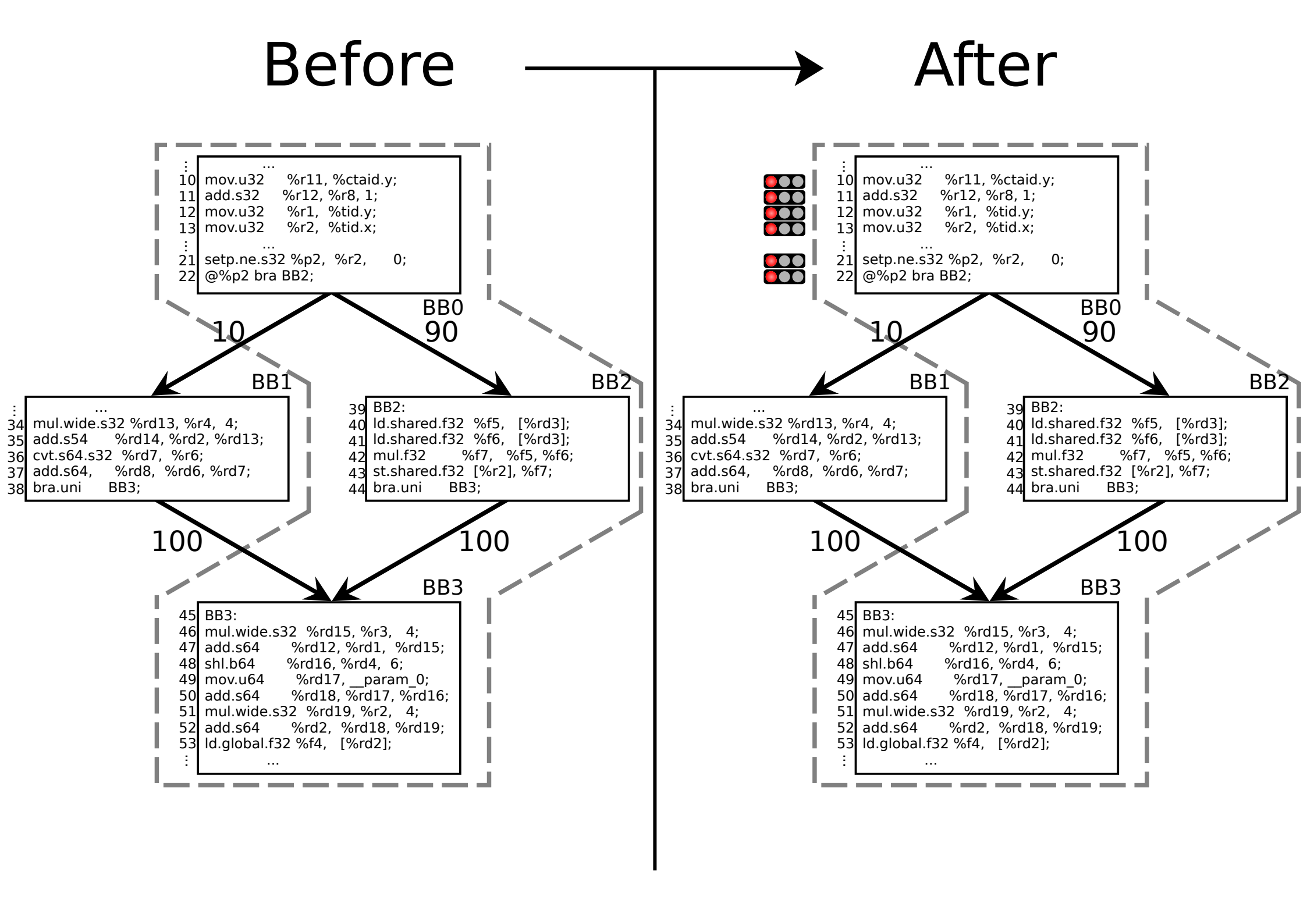
Add node to trace

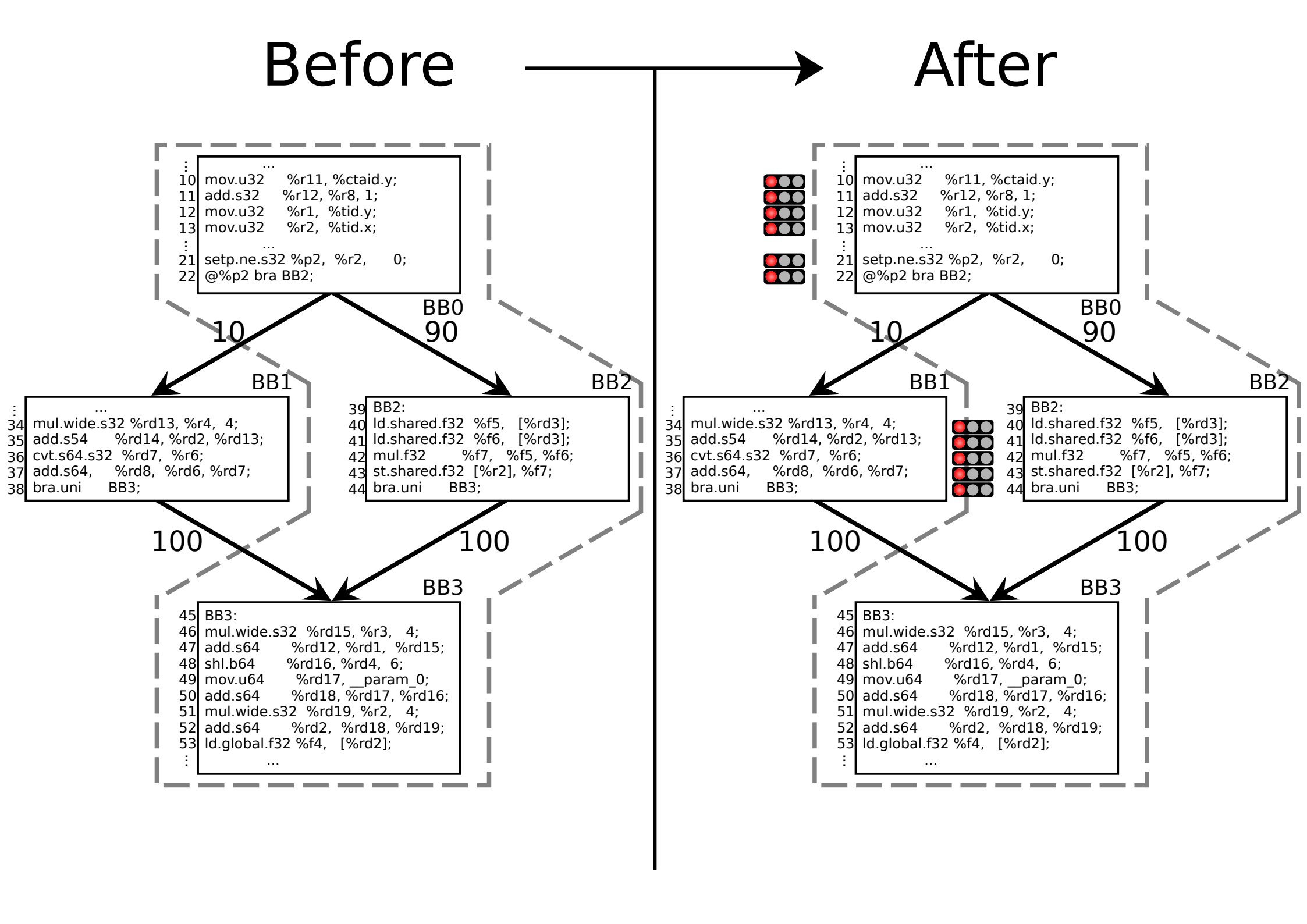
end loop

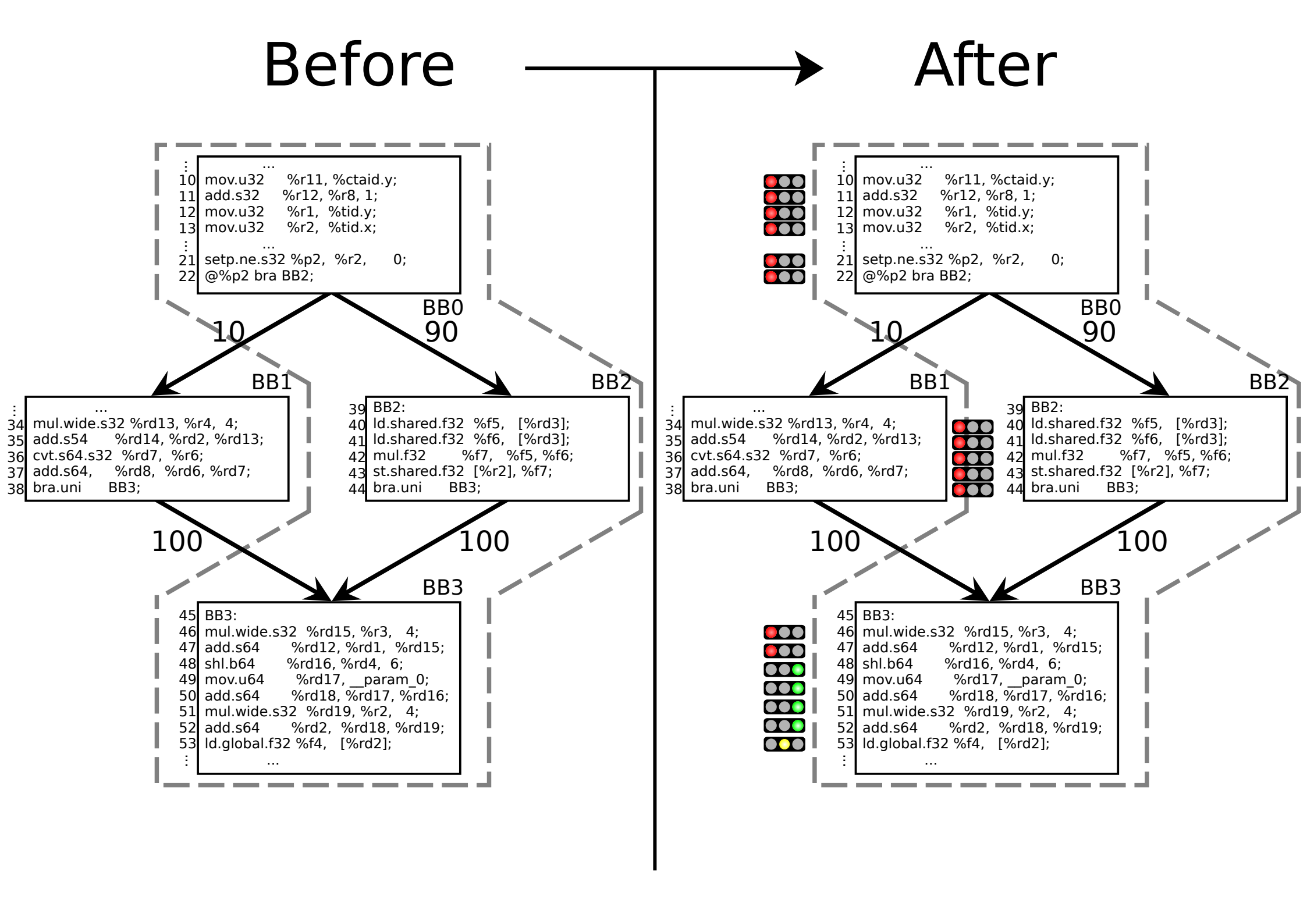
end while

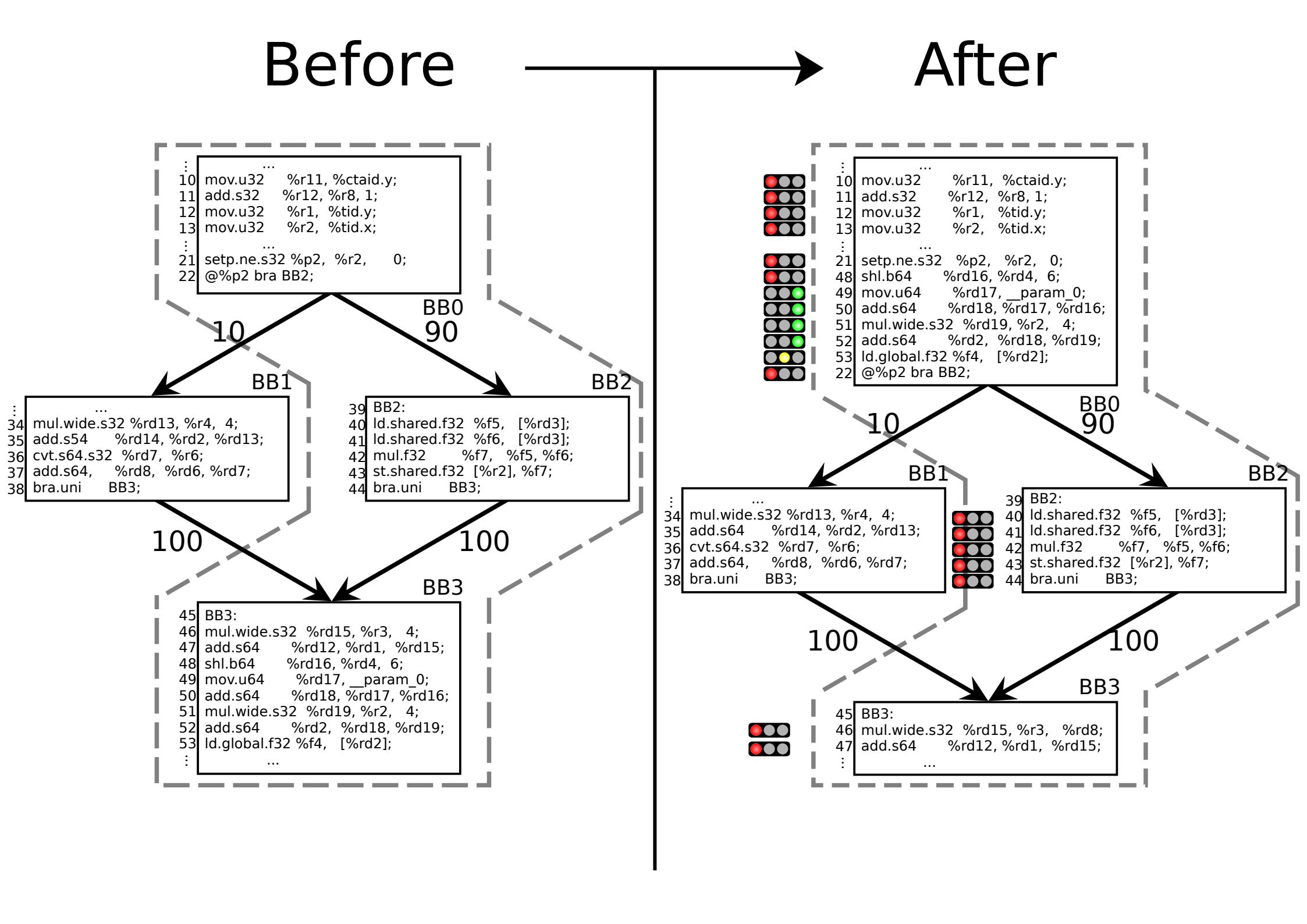


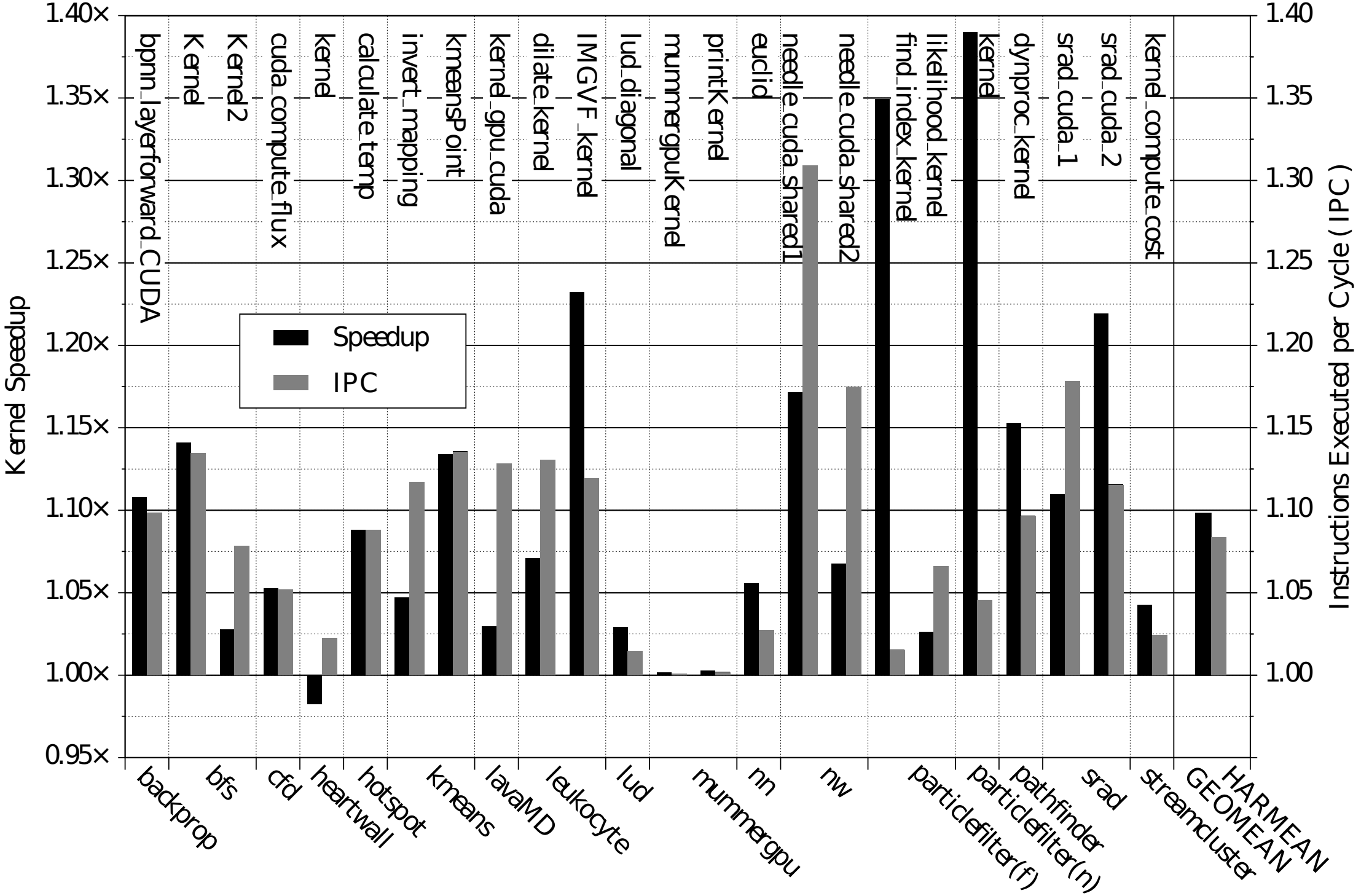

Before
After
...
10 mov.u32 %r11, %ctaid.y;
11 add.s32 %r12, %r8, 1;
12 mov.u32 %r1, %tid.y;
13 mov.u32 %r2, %tid.x;
...
21 setp.ne.s32 %p2, %r2, 0;
22 @%p2 bra BB2;
BB0
10
90
BB1
...
34 mul.wide.s32 %rd13, %r4, 4;
35 add.s54 %rd14, %rd2, %rd13;
36 cvt.s64.s32 %rd7, %r6;
37 add.s64, %rd8, %rd6, %rd7;
38 bra.uni BB3;
BB2
39 BB2:
40 ld.shared.f32 %f5, [%rd3];
41 ld.shared.f32 %f6, [%rd3];
42 mul.f32 %f7, %f5, %f6;
43 st.shared.f32 [%r2], %f7;
44 bra.uni BB3;
100
100
BB3
45 BB3:
46 mul.wide.s32 %rd15, %r3, 4;
47 add.s64 %rd12, %rd1, %rd15;
48 shl.b64 %rd16, %rd4, 6;
49 mov.u64 %rd17, __param_0;
50 add.s64 %rd18, %rd17, %rd16;
51 mul.wide.s32 %rd19, %r2, 4;
52 add.s64 %rd2, %rd18, %rd19;
53 ld.global.f32 %f4, [%rd2];
...


Before
After
BB0
10
90
BB1
BB2
BB3
100
100
...
10 mov.u32 %r11, %ctaid.y;
11 add.s32 %r12, %r8, 1;
12 mov.u32 %r1, %tid.y;
13 mov.u32 %r2, %tid.x;
...
21 setp.ne.s32 %p2, %r2, 0;
22 @%p2 bra BB2;
34 mul.wide.s32 %rd13, %r4, 4;
35 add.s54 %rd14, %rd2, %rd13;
36 cvt.s64.s32 %rd7, %r6;
37 add.s64, %rd8, %rd6, %rd7;
38 bra.uni BB3;
39 BB2:
40 ld.shared.f32 %f5, [%rd3];
41 ld.shared.f32 %f6, [%rd3];
42 mul.f32 %f7, %f5, %f6;
43 st.shared.f32 [%r2], %f7;
44 bra.uni BB3;
45 BB3:
46 mul.wide.s32 %rd15, %r3, 4;
47 add.s64 %rd12, %rd1, %rd15;
48 shl.b64 %rd16, %rd4, 6;
49 mov.u64 %rd17, __param_0;
50 add.s64 %rd18, %rd17, %rd16;
51 mul.wide.s32 %rd19, %r2, 4;
52 add.s64 %rd2, %rd18, %rd19;
53 ld.global.f32 %f4, [%rd2];
...


Before
After
BB0
BB1
BB2
BB3
10
90
100
100
10
90
100
100
...
10 mov.u32 %r11, %ctaid.y;
11 add.s32 %r12, %r8, 1;
12 mov.u32 %r1, %tid.y;
13 mov.u32 %r2, %tid.x;
...
21 setp.ne.s32 %p2, %r2, 0;
22 @%p2 bra BB2;
...
34 mul.wide.s32 %rd13, %r4, 4;
35 add.s54 %rd14, %rd2, %rd13;
36 cvt.s64.s32 %rd7, %r6;
37 add.s64, %rd8, %rd6, %rd7;
38 bra.uni BB3;
39 BB2:
40 ld.shared.f32 %f5, [%rd3];
41 ld.shared.f32 %f6, [%rd3];
42 mul.f32 %f7, %f5, %f6;
43 st.shared.f32 [%r2], %f7;
44 bra.uni BB3;
45 BB3:
46 mul.wide.s32 %rd15, %r3, 4;
47 add.s64 %rd12, %rd1, %rd15;
48 shl.b64 %rd16, %rd4, 6;
49 mov.u64 %rd17, __param_0;
50 add.s64 %rd18, %rd17, %rd16;
51 mul.wide.s32 %rd19, %r2, 4;
52 add.s64 %rd2, %rd18, %rd19;
53 ld.global.f32 %f4, [%rd2];
...


Before
After
BB0
BB1
BB2
BB3
10
90
100
100
...
10 mov.u32 %r11, %ctaid.y;
11 add.s32 %r12, %r8, 1;
12 mov.u32 %r1, %tid.y;
13 mov.u32 %r2, %tid.x;
...
21 setp.ne.s32 %p2, %r2, 0;
22 @%p2 bra BB2;
...
34 mul.wide.s32 %rd13, %r4, 4;
35 add.s54 %rd14, %rd2, %rd13;
36 cvt.s64.s32 %rd7, %r6;
37 add.s64, %rd8, %rd6, %rd7;
38 bra.uni BB3;
39 BB2:
40 ld.shared.f32 %f5, [%rd3];
41 ld.shared.f32 %f6, [%rd3];
42 mul.f32 %f7, %f5, %f6;
43 st.shared.f32 [%r2], %f7;
44 bra.uni BB3;
45 BB3:
46 mul.wide.s32 %rd15, %r3, 4;
47 add.s64 %rd12, %rd1, %rd15;
48 shl.b64 %rd16, %rd4, 6;
49 mov.u64 %rd17, __param_0;
50 add.s64 %rd18, %rd17, %rd16;
51 mul.wide.s32 %rd19, %r2, 4;
52 add.s64 %rd2, %rd18, %rd19;
53 ld.global.f32 %f4, [%rd2];
...

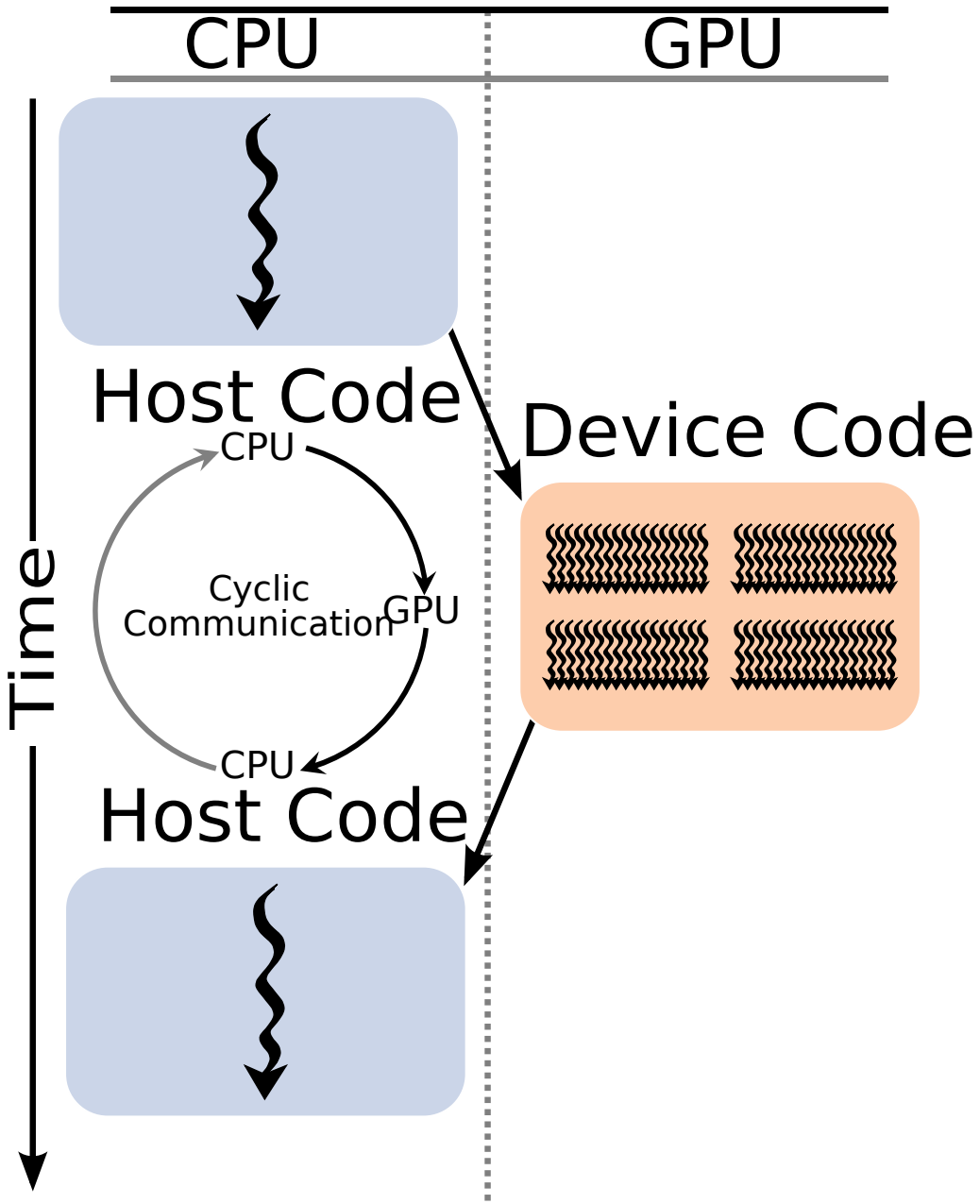

Before
After
BB0
BB1
BB2
BB3
10
90
100
100
...
10 mov.u32 %r11, %ctaid.y;
11 add.s32 %r12, %r8, 1;
12 mov.u32 %r1, %tid.y;
13 mov.u32 %r2, %tid.x;
...
21 setp.ne.s32 %p2, %r2, 0;
22 @%p2 bra BB2;
34 mul.wide.s32 %rd13, %r4, 4;
35 add.s54 %rd14, %rd2, %rd13;
36 cvt.s64.s32 %rd7, %r6;
37 add.s64, %rd8, %rd6, %rd7;
38 bra.uni BB3;
39 BB2:
40 ld.shared.f32 %f5, [%rd3];
41 ld.shared.f32 %f6, [%rd3];
42 mul.f32 %f7, %f5, %f6;
43 st.shared.f32 [%r2], %f7;
44 bra.uni BB3;
45 BB3:
46 mul.wide.s32 %rd15, %r3, 4;
47 add.s64 %rd12, %rd1, %rd15;
48 shl.b64 %rd16, %rd4, 6;
49 mov.u64 %rd17, __param_0;
50 add.s64 %rd18, %rd17, %rd16;
51 mul.wide.s32 %rd19, %r2, 4;
52 add.s64 %rd2, %rd18, %rd19;
53 ld.global.f32 %f4, [%rd2];
...
10 mov.u32 %r11, %ctaid.y;
11 add.s32 %r12, %r8, 1;
12 mov.u32 %r1, %tid.y;
13 mov.u32 %r2, %tid.x;
...
21 setp.ne.s32 %p2, %r2, 0;
48 shl.b64 %rd16, %rd4, 6;
49 mov.u64 %rd17, __param_0;
50 add.s64 %rd18, %rd17, %rd16;
51 mul.wide.s32 %rd19, %r2, 4;
52 add.s64 %rd2, %rd18, %rd19;
53 ld.global.f32 %f4, [%rd2];
22 @%p2 bra BB2;
34 mul.wide.s32 %rd13, %r4, 4;
35 add.s64 %rd14, %rd2, %rd13;
36 cvt.s64.s32 %rd7, %r6;
37 add.s64, %rd8, %rd6, %rd7;
38 bra.uni BB3;
39 BB2:
40 ld.shared.f32 %f5, [%rd3];
41 ld.shared.f32 %f6, [%rd3];
42 mul.f32 %f7, %f5, %f6;
43 st.shared.f32 [%r2], %f7;
44 bra.uni BB3;
45 BB3:
46 mul.wide.s32 %rd15, %r3, %rd8;
47 add.s64 %rd12, %rd1, %rd15;
...


Comparing Speedup and IPC Using Dynamic Profiling
Kernel Speedup
Instructions Executed per Cycle (IPC)
1.40×
1.35×
1.30×
1.25×
1.20×
1.15×
1.10×
1.05×
1.00×
0.95×
1.40
1.35
1.30
1.25
1.20
1.15
1.10
1.05
1.00
Speedup
IPC
bpnn_layerforward_CUDA
Kernel
Kernel2
cuda_compute_flux
kernel
calculate_temp
invert_mapping
kmeansPoint
kernel_gpu_cuda
dilate_kernel
IMGVF_kernel
lud_diagonal
mummergpuKernel
printKernel
euclid
needle_cuda_shared1
needle_cuda_shared2
likelihood_kernel
find_index_kernel
kernel
dynproc_kernel
srad_cuda_1
srad_cuda_2
kernel_compute_cost
backprop
bfs
cfd
heartwall
hotspot
kmeans
lavaMD
leukocyte
lud
mummergpu
nn
nw
particlefilter(f)
particlefilter(n)
pathfinder
srad
streamcluster
HARMEAN
GEOMEAN

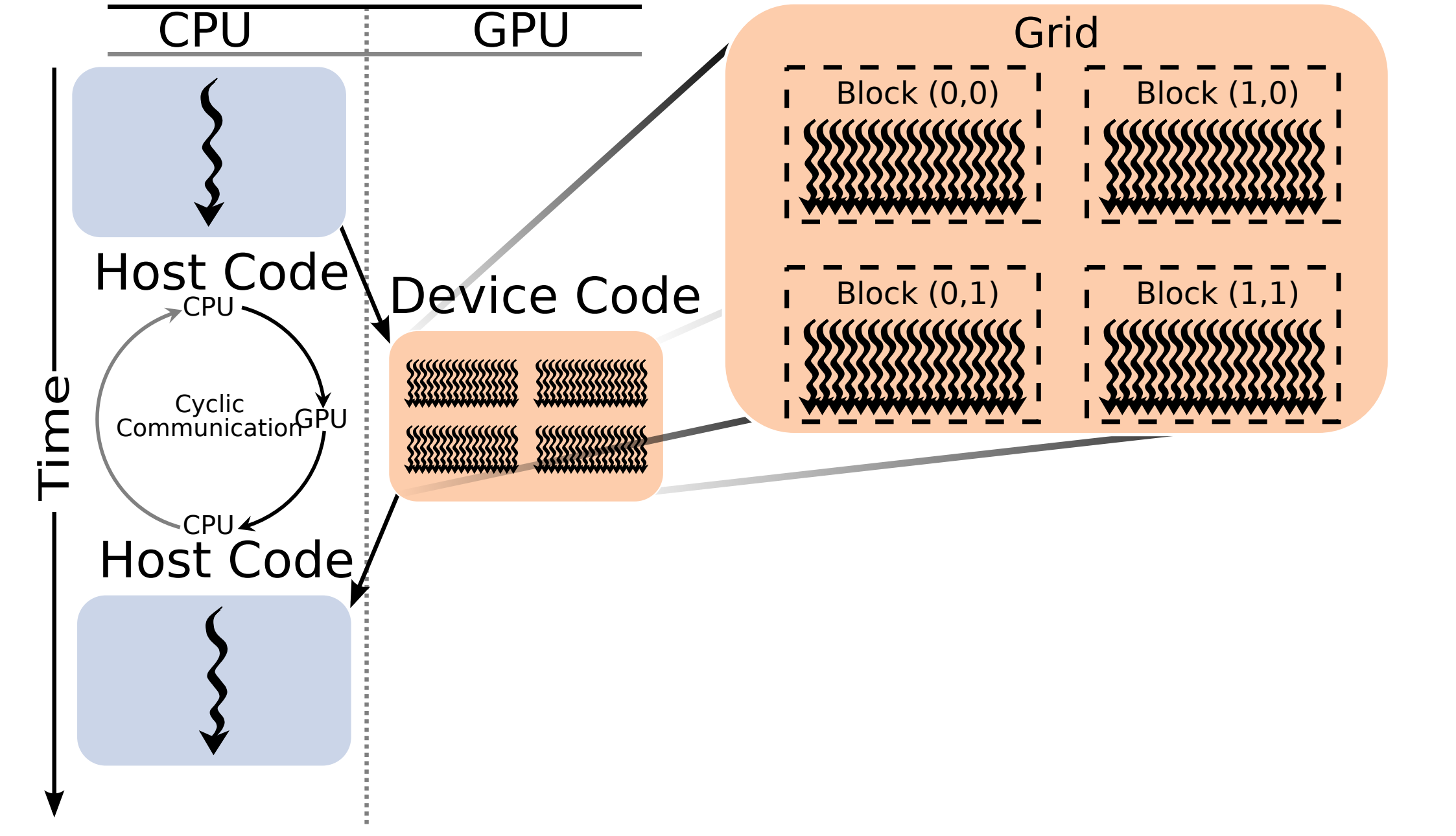
Backup Slides

	Restricted [6]	General [6]	Boosting [36]	Deviant (GPU)
Scheduling Restrictions	Legal and Safe	Legal	none	excludes texture, shared and constant memory operations and all store instructions
Hardware Support	none	non-trapping instructions	shadow register file, shadow store buffer, and support for re-executing instructions	none
Exception Handling for Speculative Instructions	prohibited	ignored	supported	absent

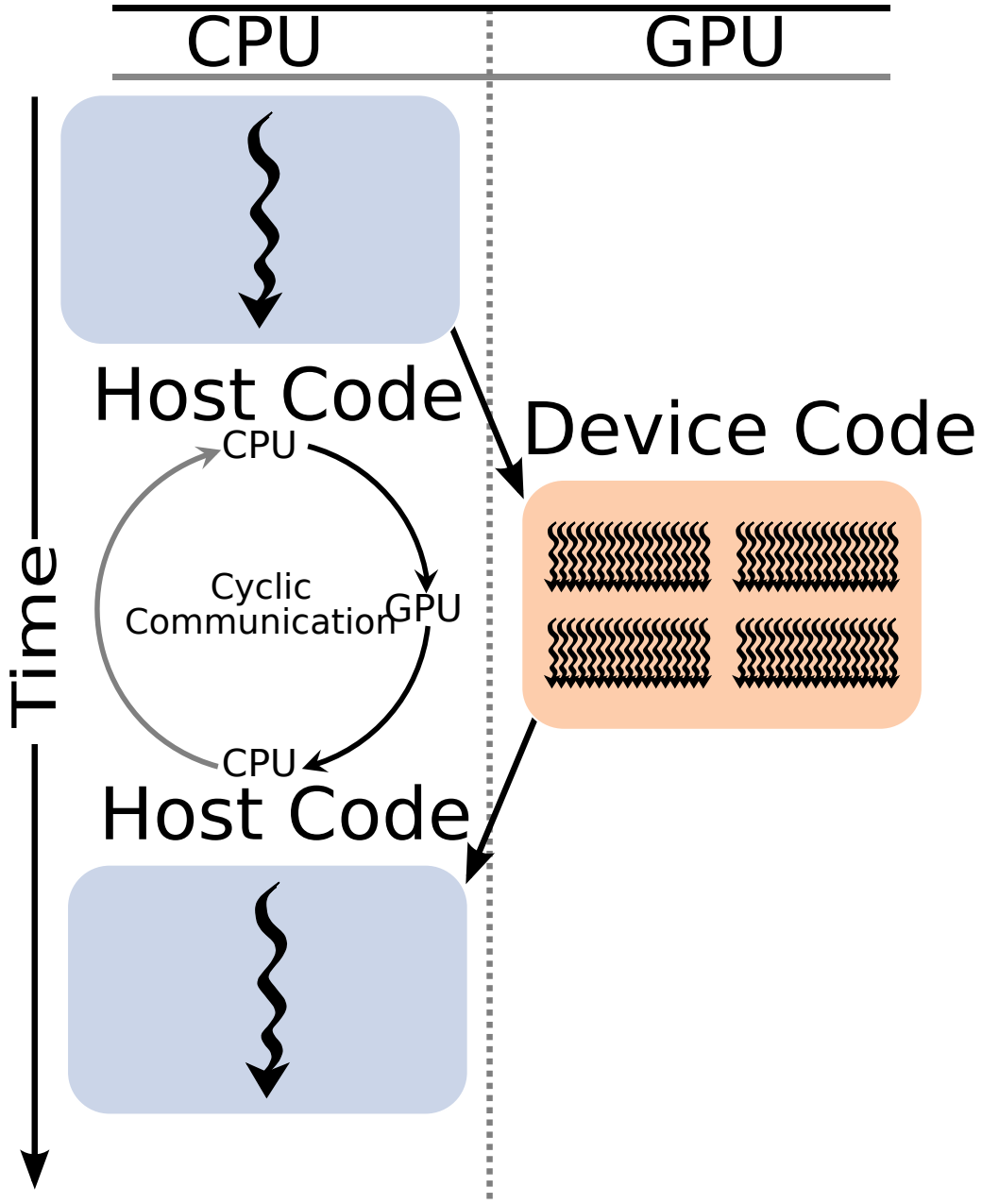
GPU Programming Model

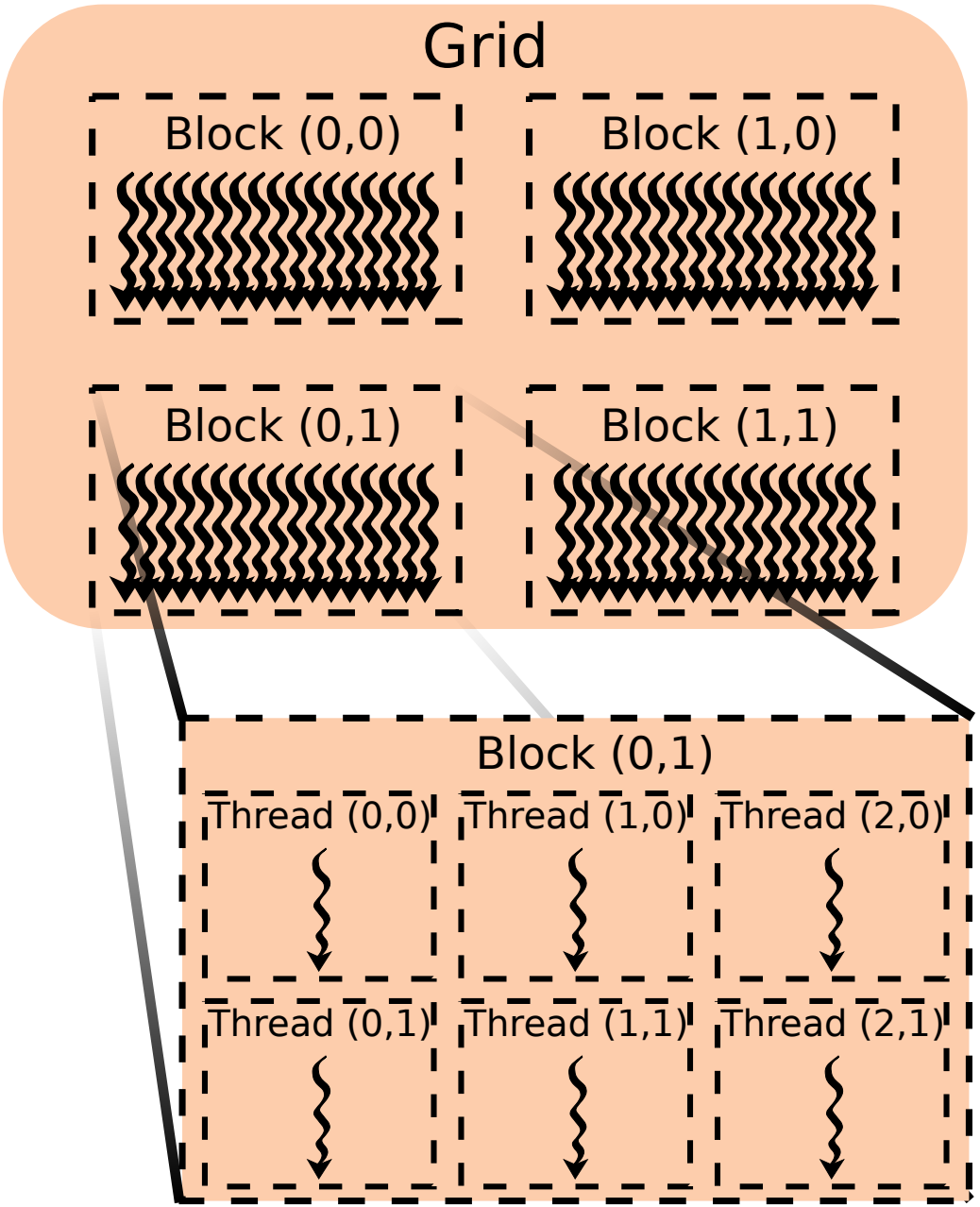


GPU Programming Model

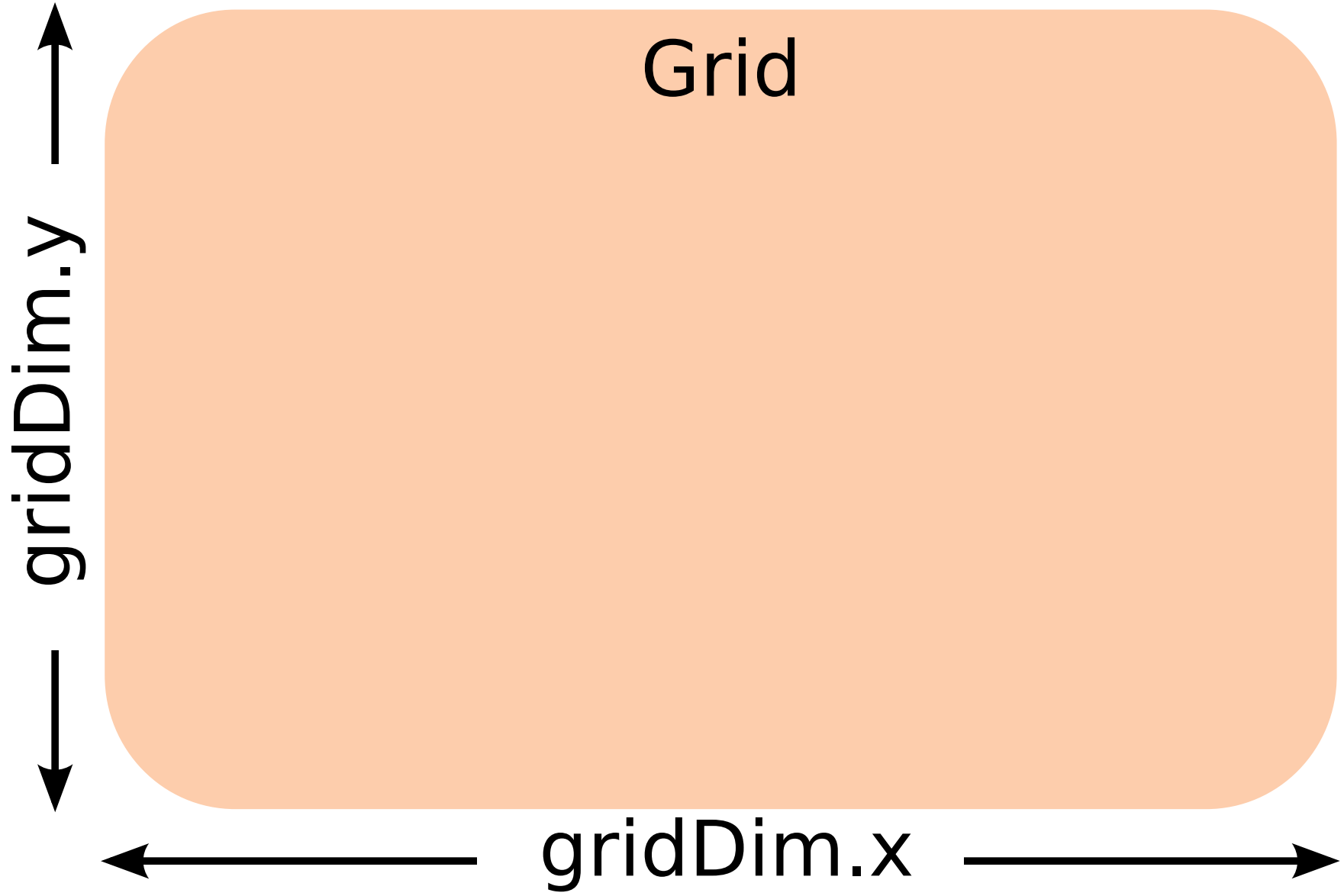


GPU Programming Model

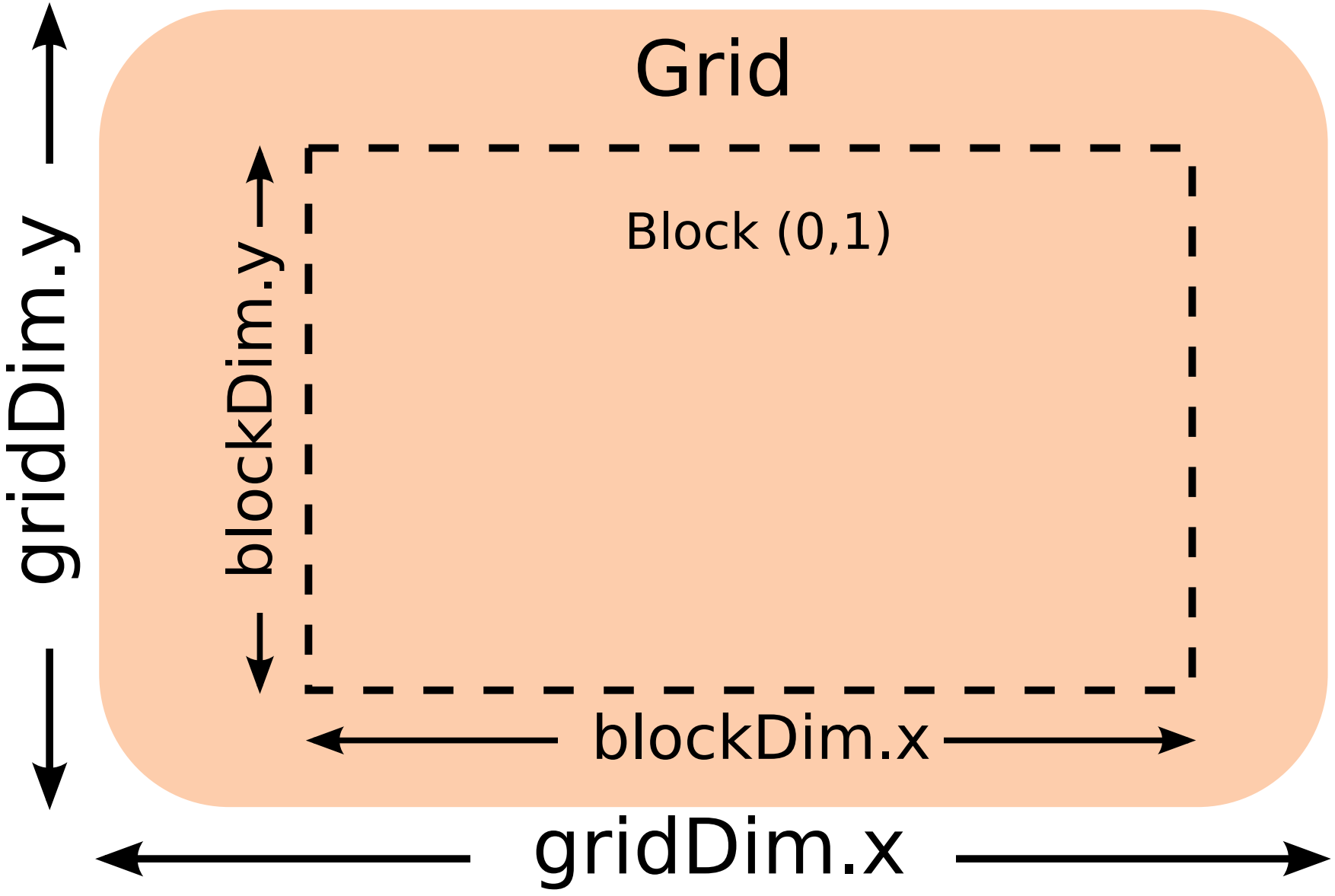




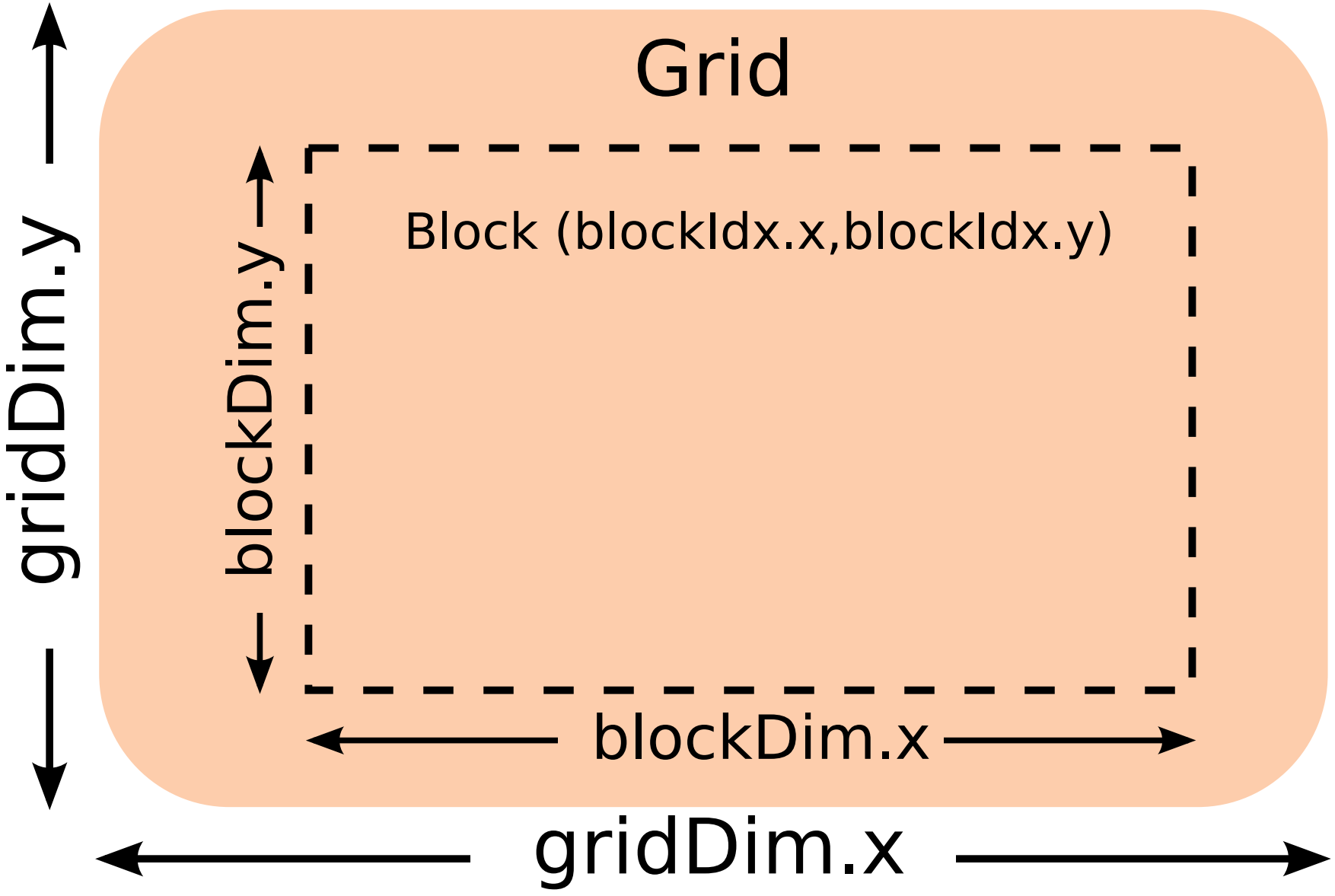
Characterizing the Grid...



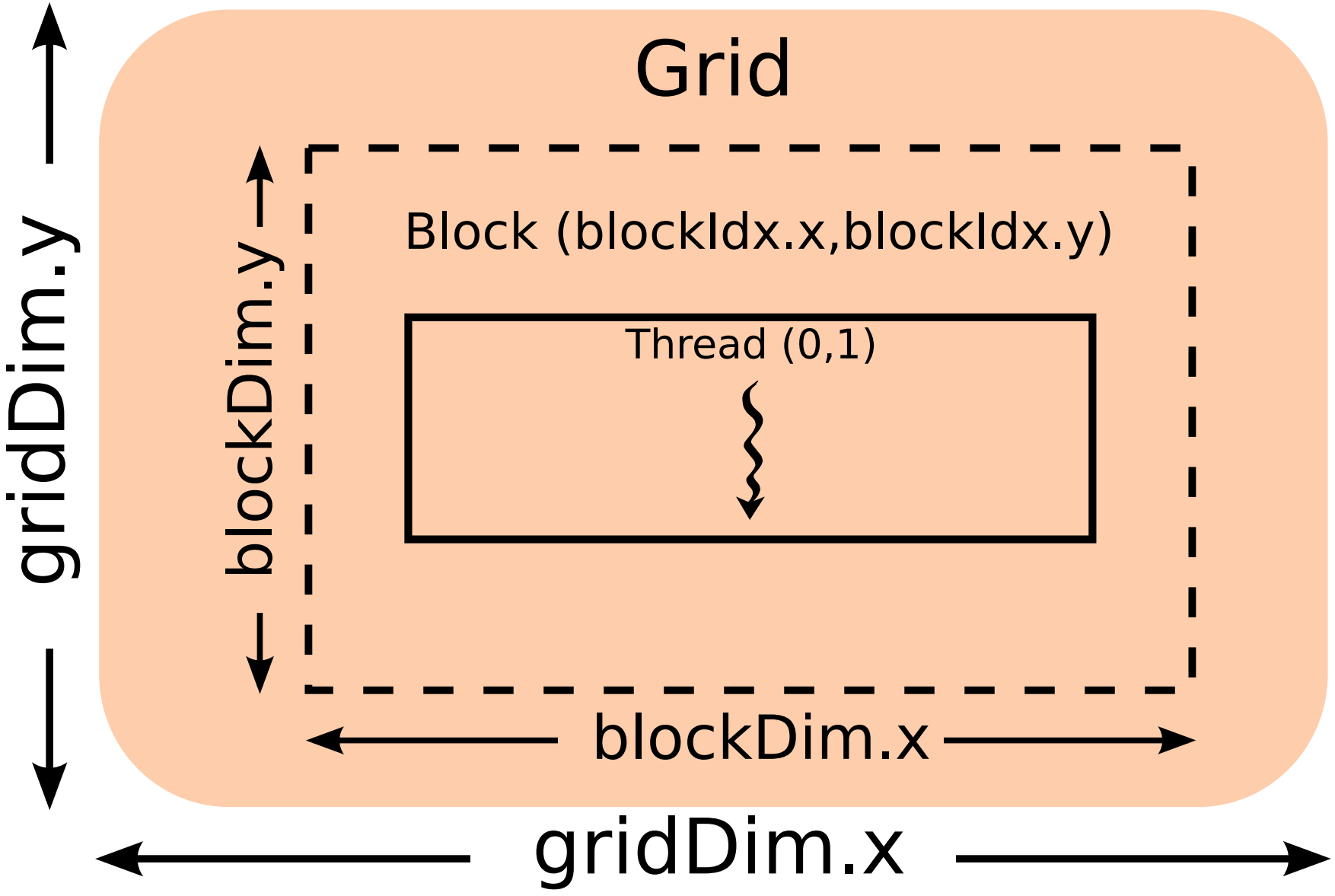
Characterizing the Grid, Blocks...



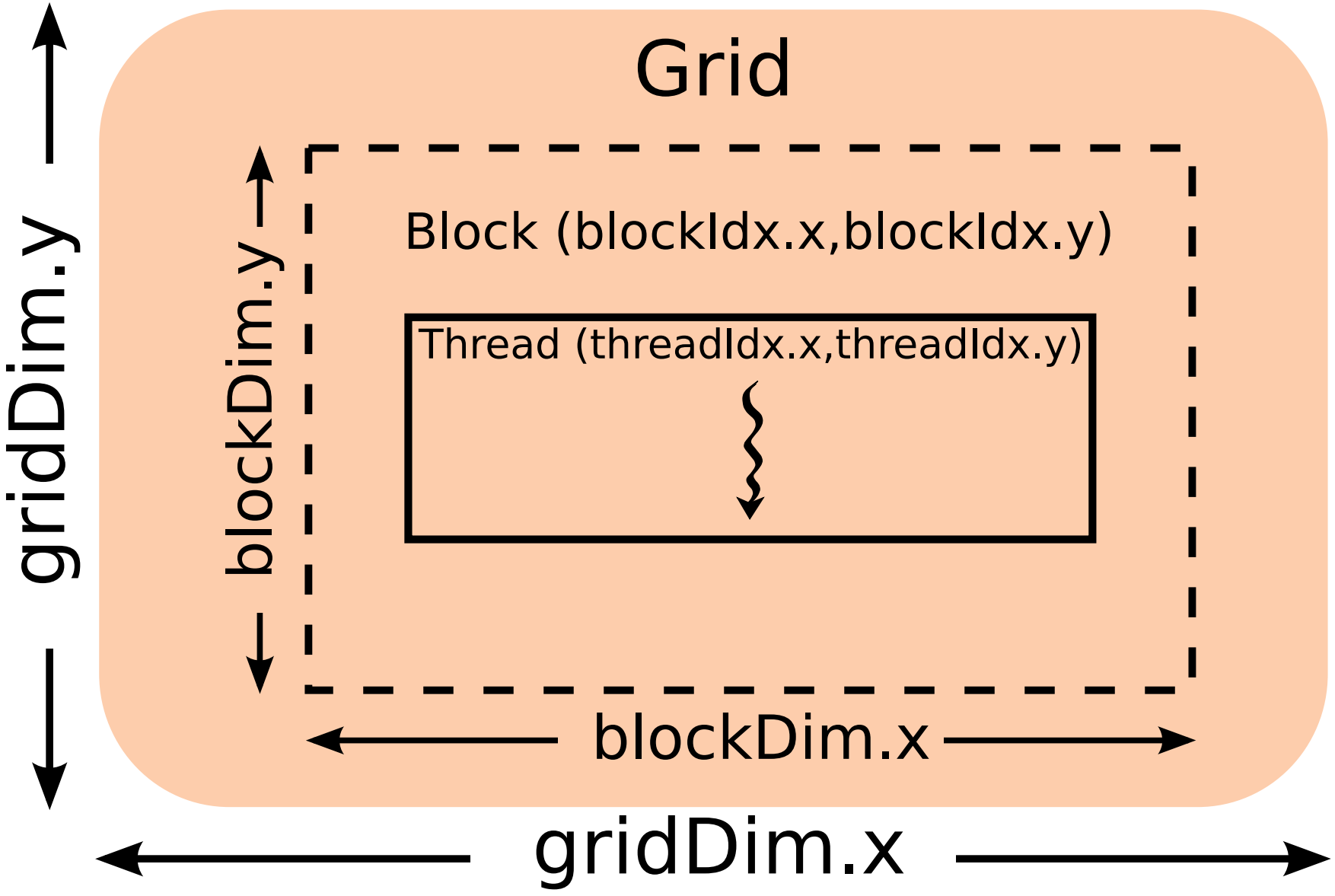
Characterizing the Grid, Blocks...



Characterizing the Grid, Blocks, and Threads



Characterizing the Grid, Blocks, and Threads



Warp Divergence Examples

Assuming one block of 128 threads...

Example	Divergence?
if (threadIdx.x < 32) { }	

Warp Divergence Examples

Assuming one block of 128 threads...

Example	Divergence?
if (threadIdx.x < 32) { }	NO
if (threadIdx.x > 15) { }	

Warp Divergence Examples

Assuming one block of 128 threads...

Example	Divergence?
if (threadIdx.x < 32) { }	NO
if (threadIdx.x > 15) { }	YES
if (threadIdx.x > 65) { }	

Warp Divergence Examples

Assuming one block of 128 threads...

Example	Divergence?
if (threadIdx.x < 32) { }	NO
if (threadIdx.x > 15) { }	YES
if (threadIdx.x > 65) { }	YES
if (BlockIdx.x > 1) { }	

Warp Divergence Examples

Assuming one block of 128 threads...

Example	Divergence?
if (threadIdx.x < 32) { }	NO
if (threadIdx.x > 15) { }	YES
if (threadIdx.x > 65) { }	YES
if (blockIdx.x > 1) { }	NO