LLVM Greedy Register Allocator – Improving Region Split Decisions

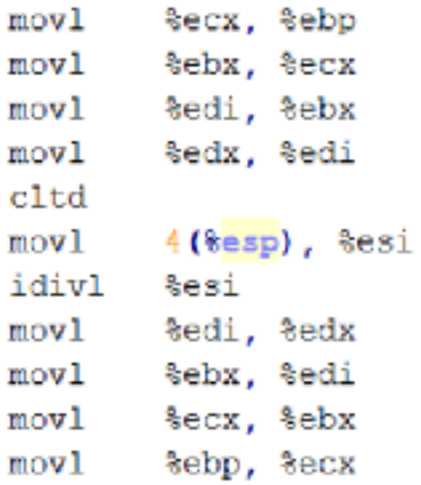
Marina Yatsina

Intel Corporation, Israel

April 16-17, 2018 European Developers Meeting

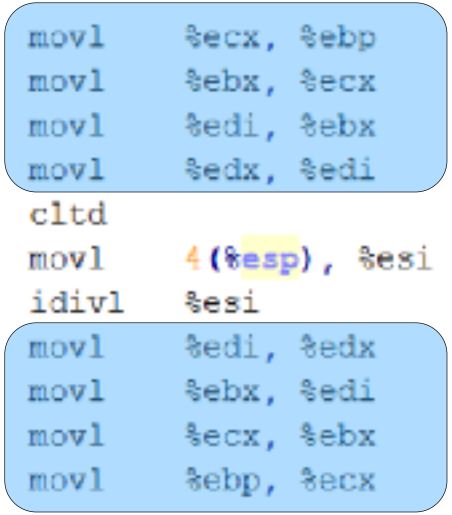
Bristol, United Kingdom

Motivation



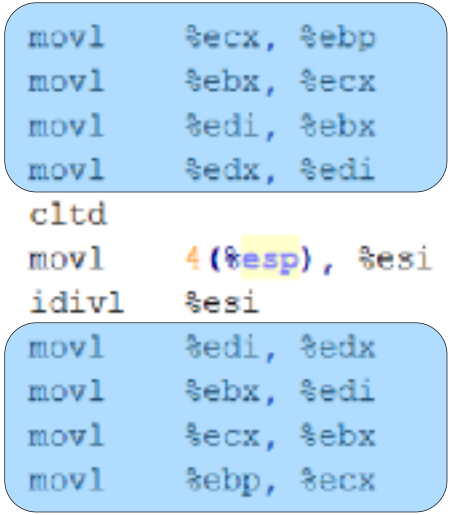
```
movl     %ecx, %ebp
movl     %ebx, %ecx
movl     %edi, %ebx
movl     %edx, %edi
cltd
movl     4(%esp), %esi
idivl    %esi
movl     %edi, %edx
movl     %ebx, %edi
movl     %ecx, %ebx
movl     %ebp, %ecx
```

Motivation



```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Motivation

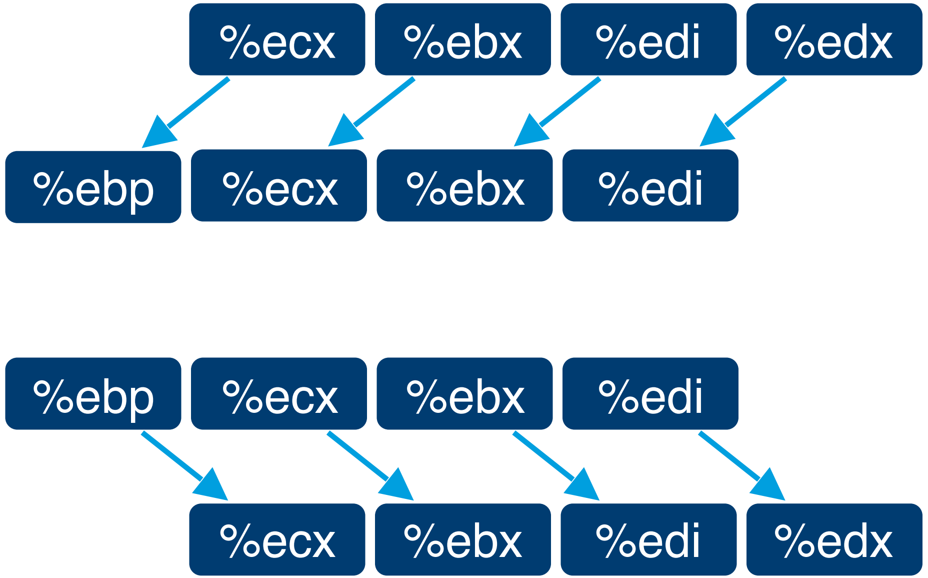


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

* idiv implicitly clobbers %edx

Motivation

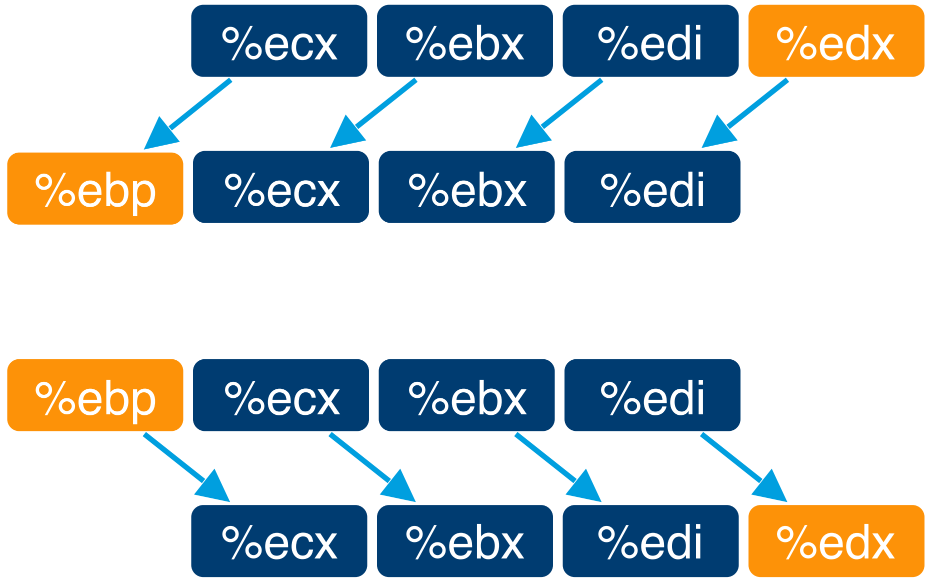
```
movl     %ecx, %ebp
movl     %ebx, %ecx
movl     %edi, %ebx
movl     %edx, %edi
cltd
movl     4(%esp), %esi
idivl    %esi
movl     %edi, %edx
movl     %ebx, %edi
movl     %ecx, %ebx
movl     %ebp, %ecx
```



* idiv implicitly clobbers %edx

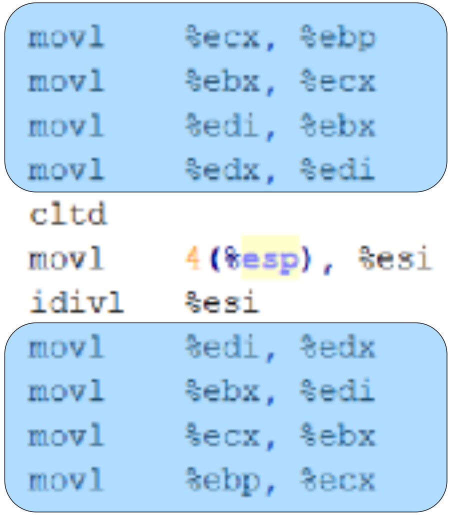
Motivation

```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

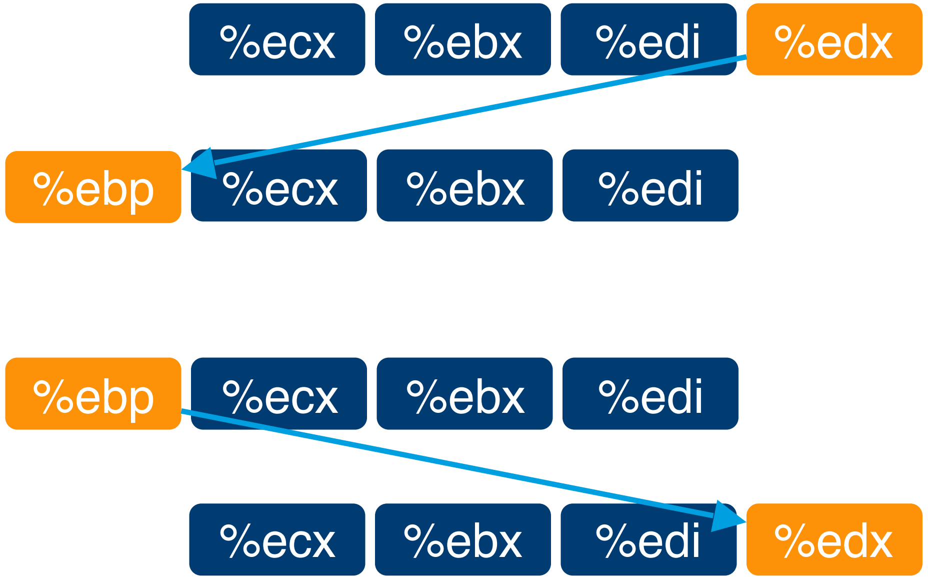


* idiv implicitly clobbers %edx

Motivation



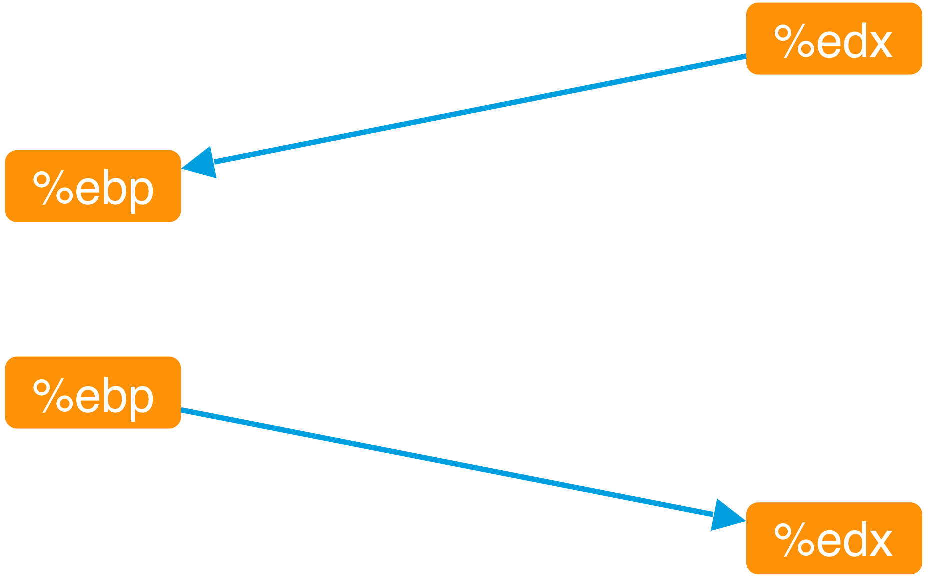
```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```



* idiv implicitly clobbers %edx

Motivation

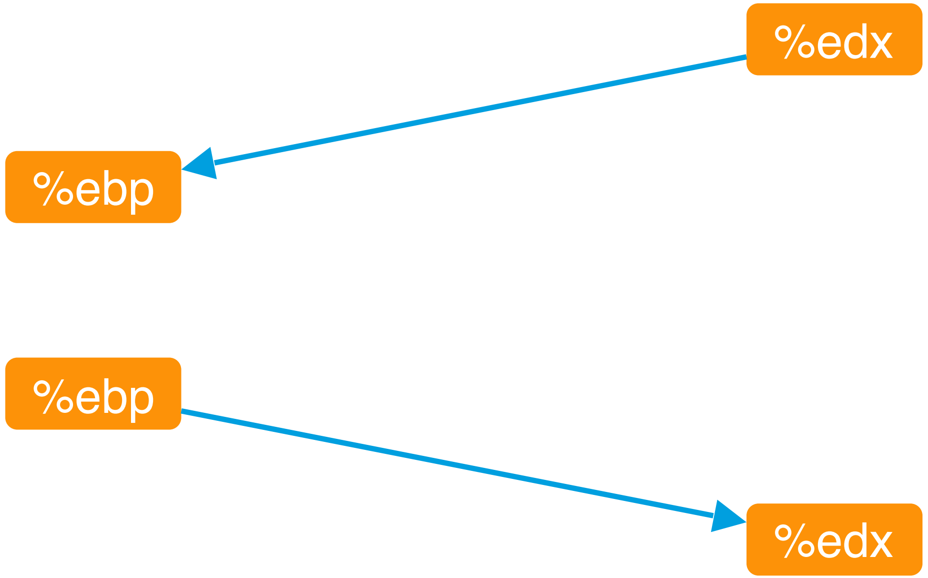
```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```



* idiv implicitly clobbers %edx

Motivation

```
movl    %edx, %ebp
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %ebp, %edx
```



* idiv implicitly clobbers %edx

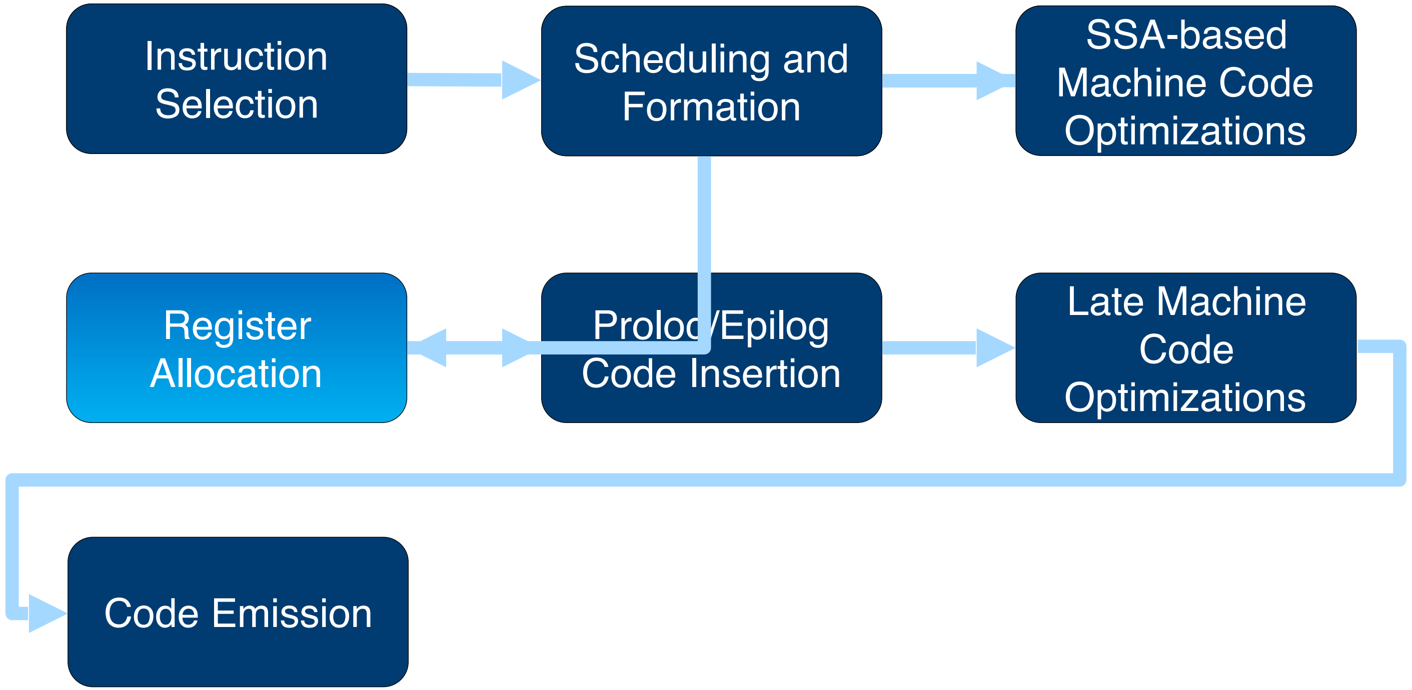
Greedy Register Allocator

- Greedy Register Allocator Overview
- Region Split
- Encountered Issues
- Performance Impact

Greedy Register Allocator

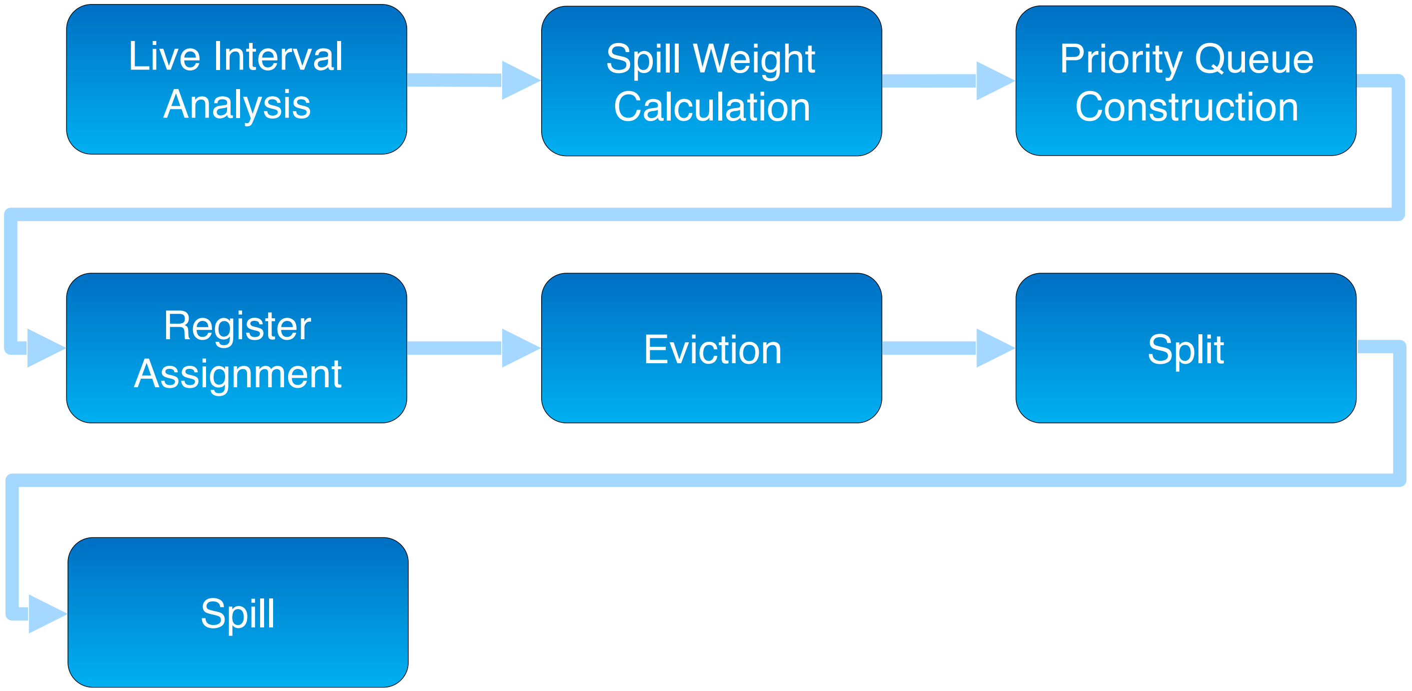
- **Greedy Register Allocator Overview**
- Region Split
- Encountered Issues
- Performance Impact

High Level Design of Code Generator


Instruction Selection
Scheduling and Formation
SSA-based Machine Code Optimizations
Register Allocation
Prolog/Epilog Code Insertion
Late Machine Code Optimizations
Code Emission

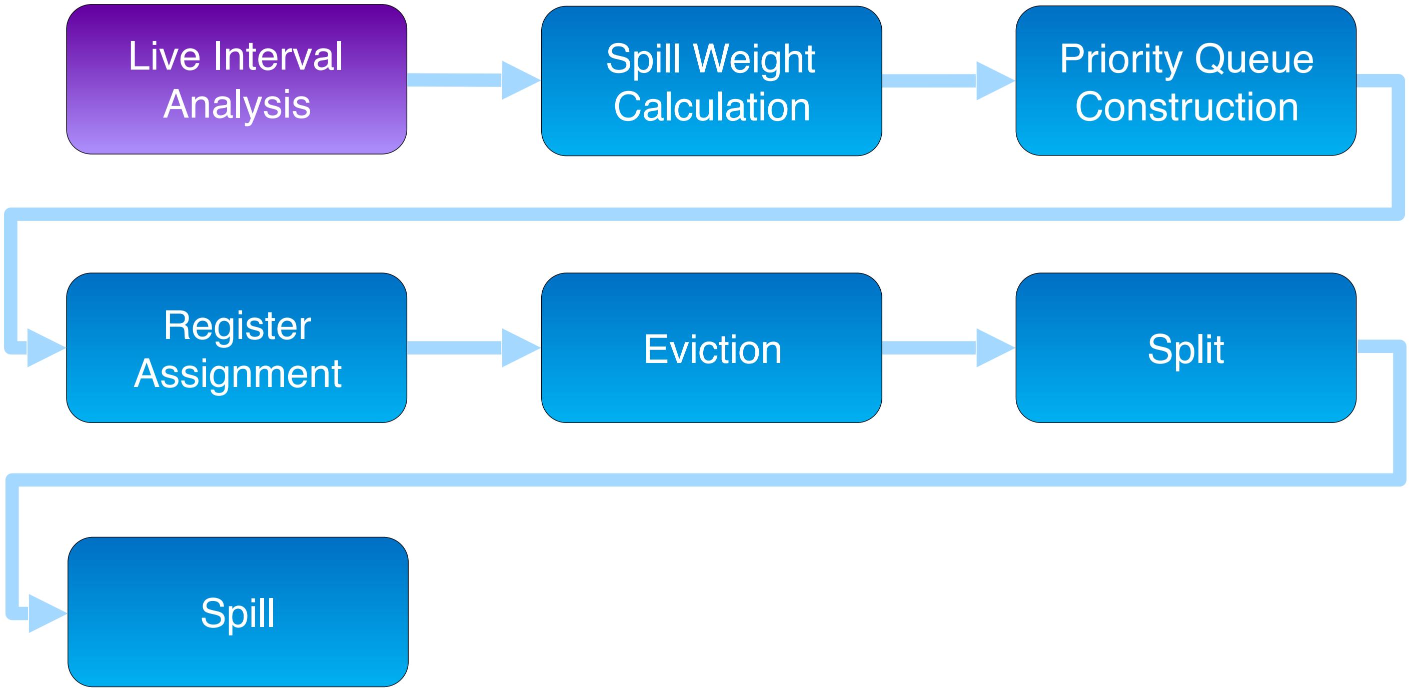
Greedy Register Allocator Overview

- General flow



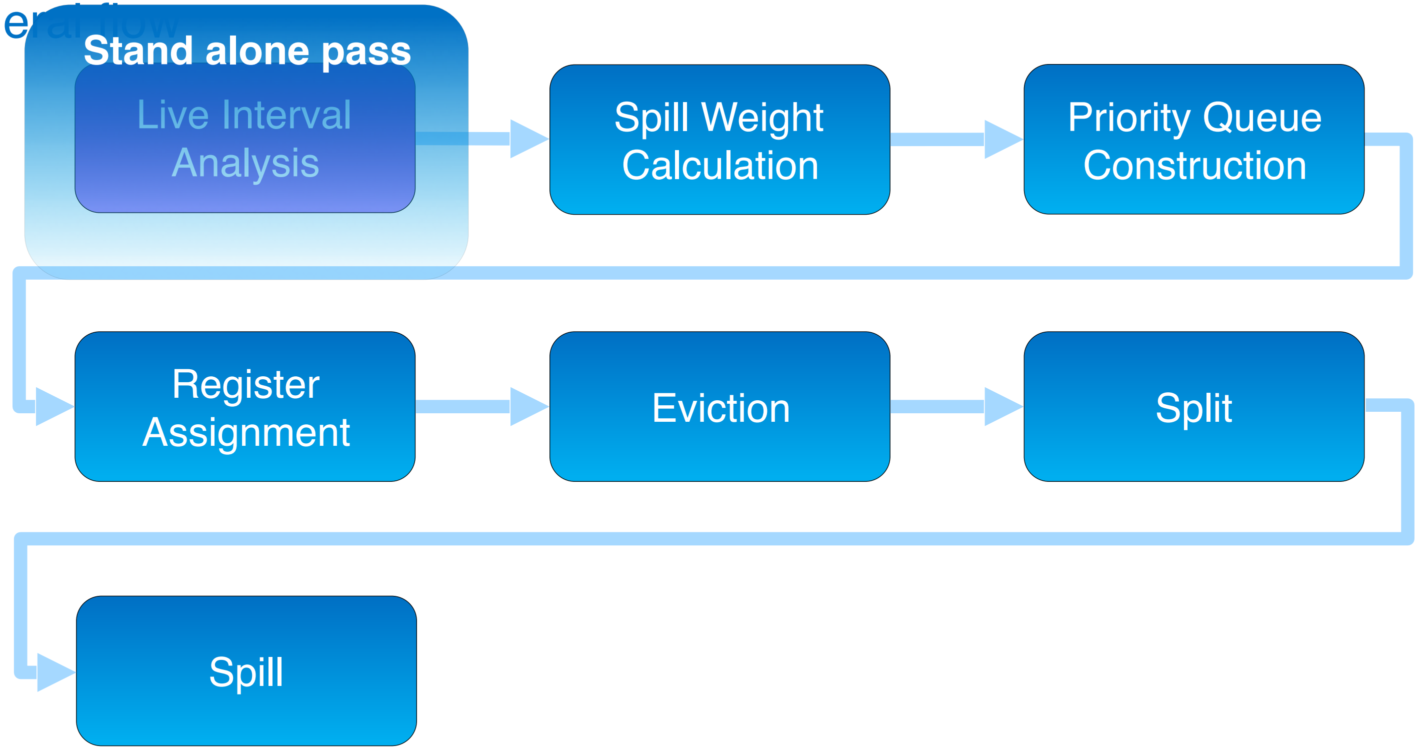
Greedy Register Allocator Overview

- General flow

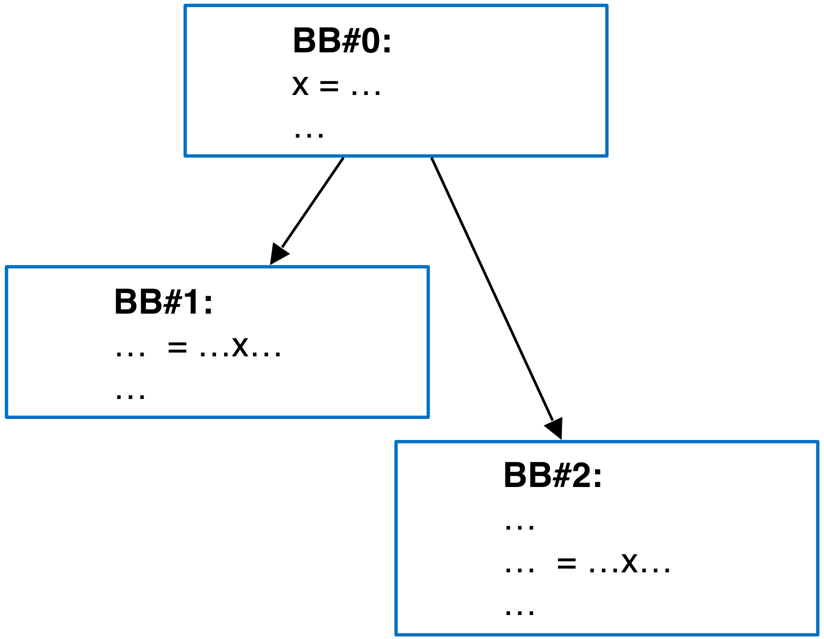


Greedy Register Allocator Overview

- General flow

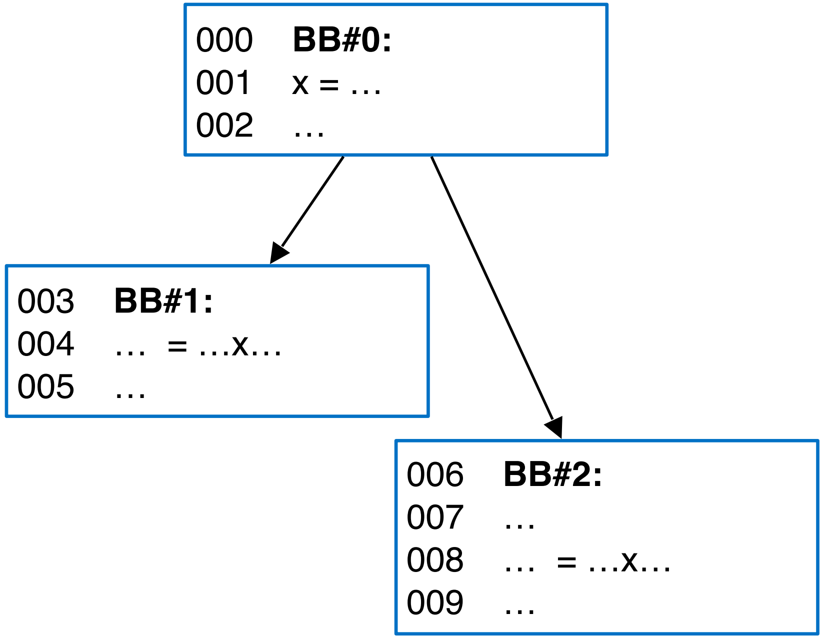


Live Interval Analysis



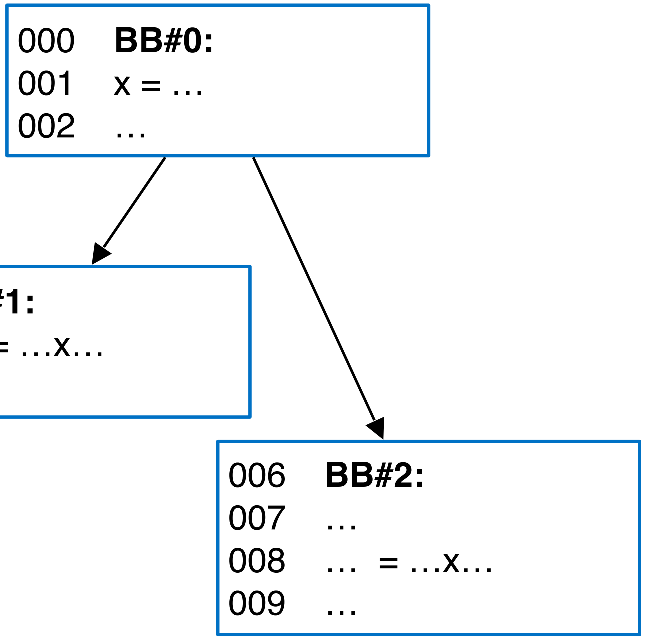
Live Interval Analysis

- Earlier pass named SlotIndexes added numbering to the instructions



Live Interval Analysis

- Analyze x's live interval



```
000   BB#0:
001   x = …
002   …
```

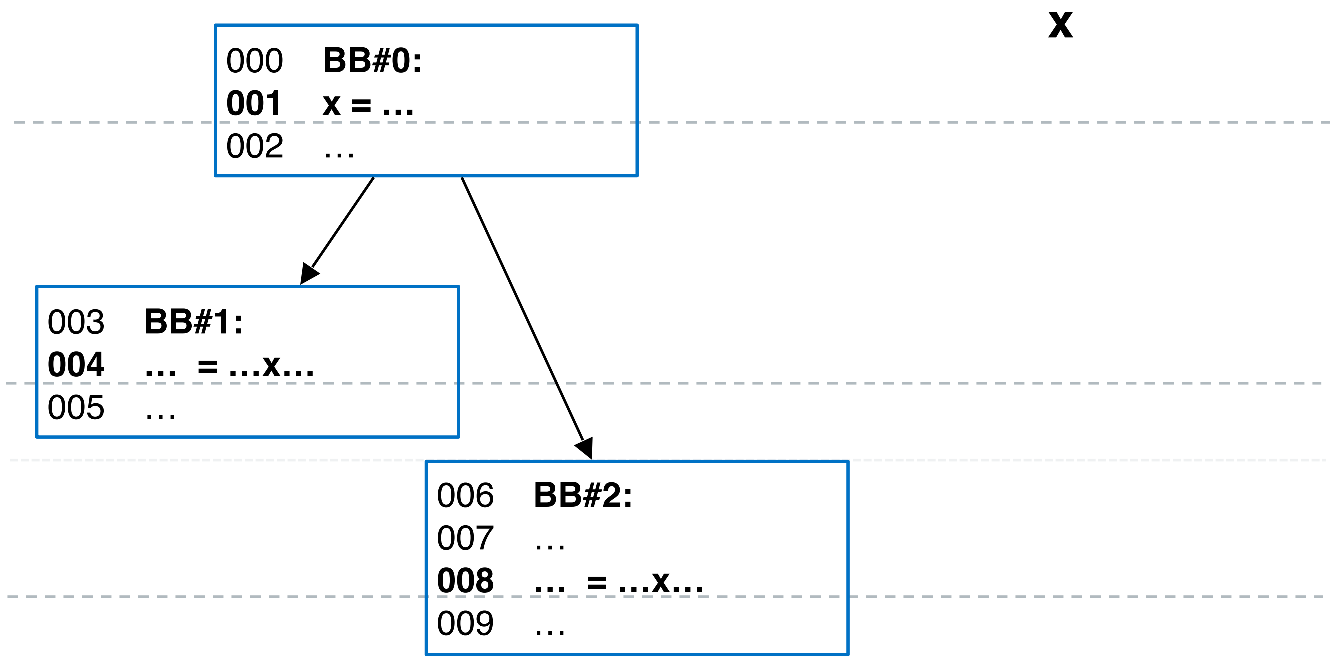
```
003   BB#1:
004   …  = …x…
005   …
```

```
006   BB#2:
007   …
008   …  = …x…
009   …
```

x

Live Interval Analysis

- Look for uses and defs



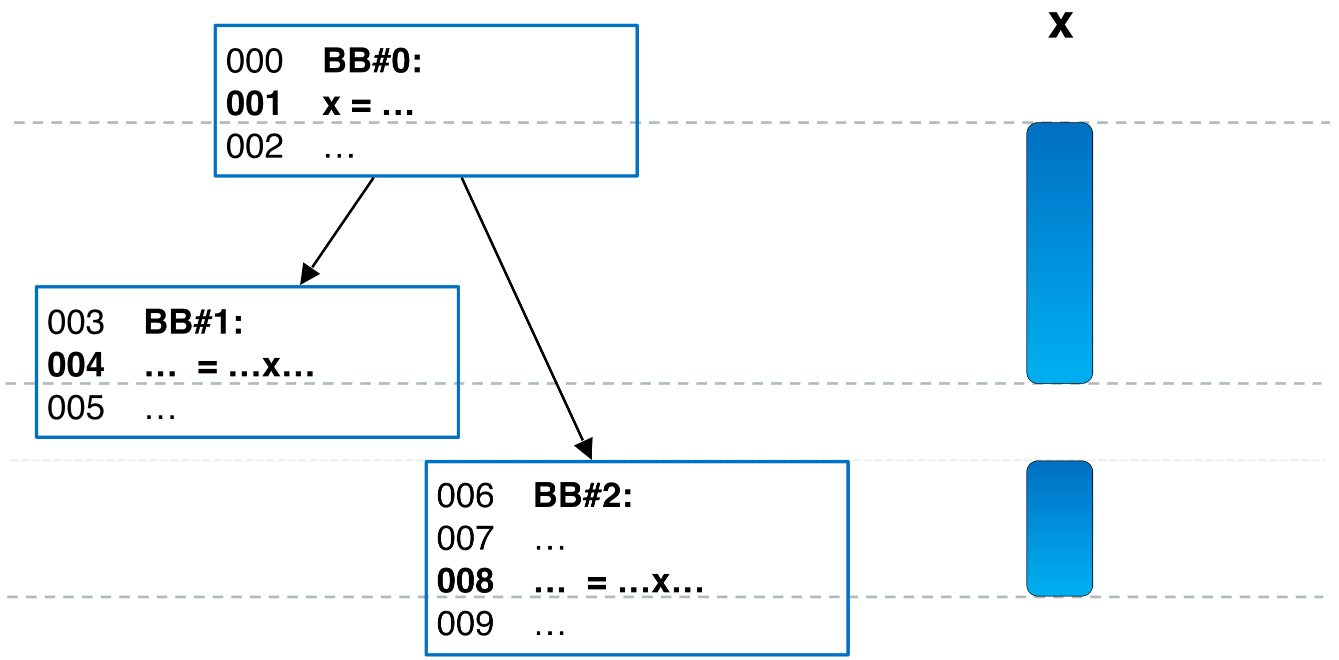
```
000   BB#0:
001   x = …
002   …

003   BB#1:
004   …  = …x…
005   …

006   BB#2:
007   …
008   …  = …x…
009   …
```

Live Interval Analysis

- Connect them into intervals



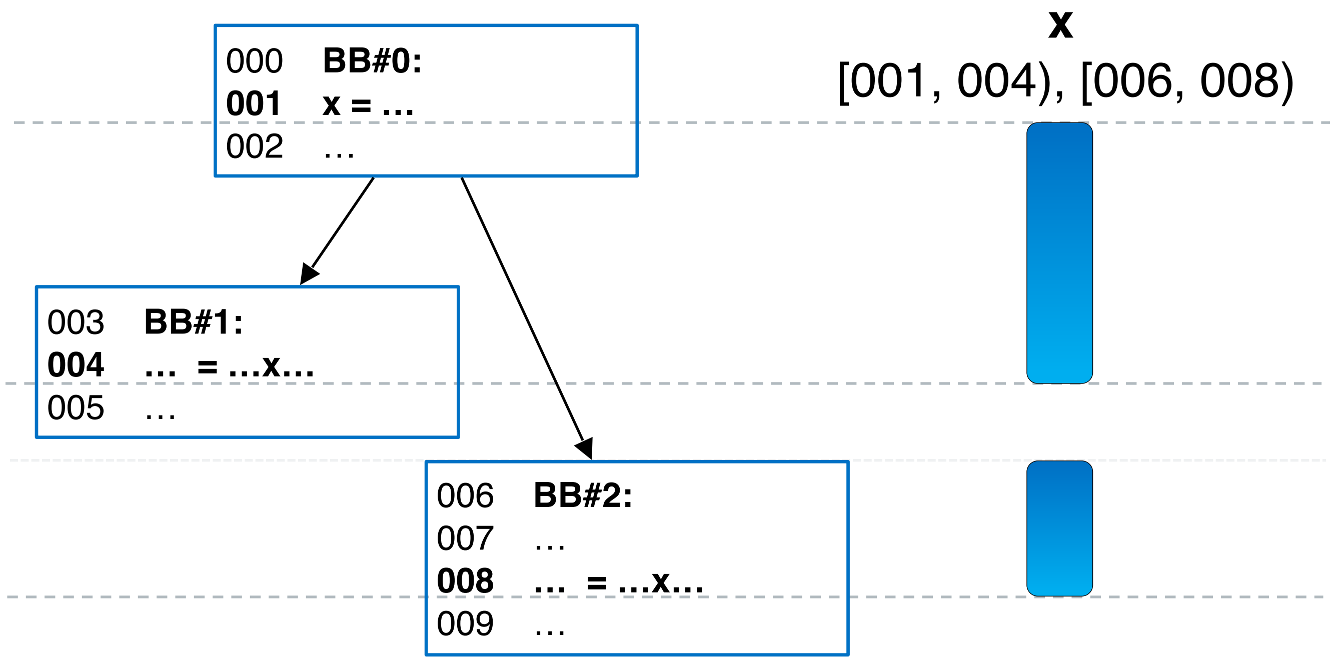
```
000   BB#0:
001   x = …
002   …
```

```
003   BB#1:
004   …  = …x…
005   …
```

```
006   BB#2:
007   …
008   …  = …x…
009   …
```

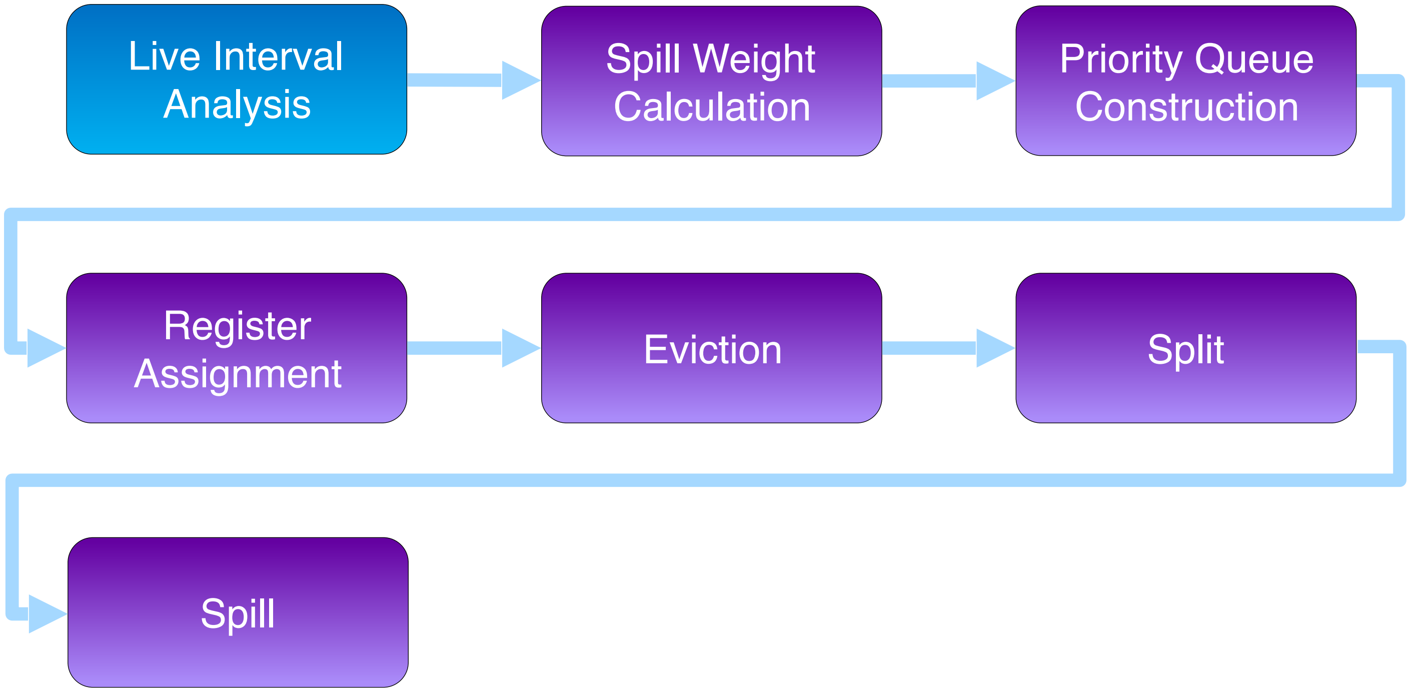
Live Interval Analysis

- Represented x's liveness as a collection of segments



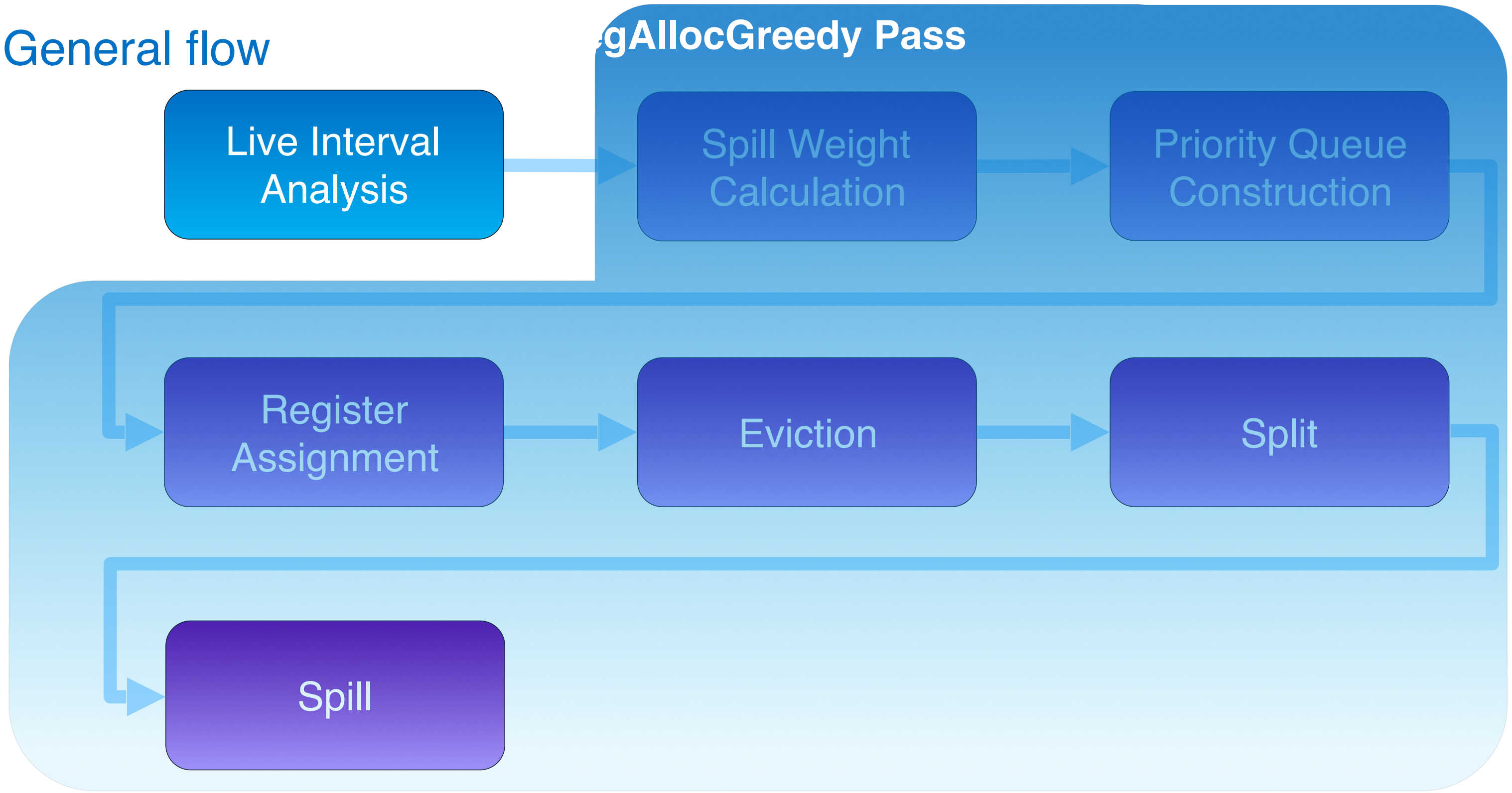
Greedy Register Allocator Overview

- General flow



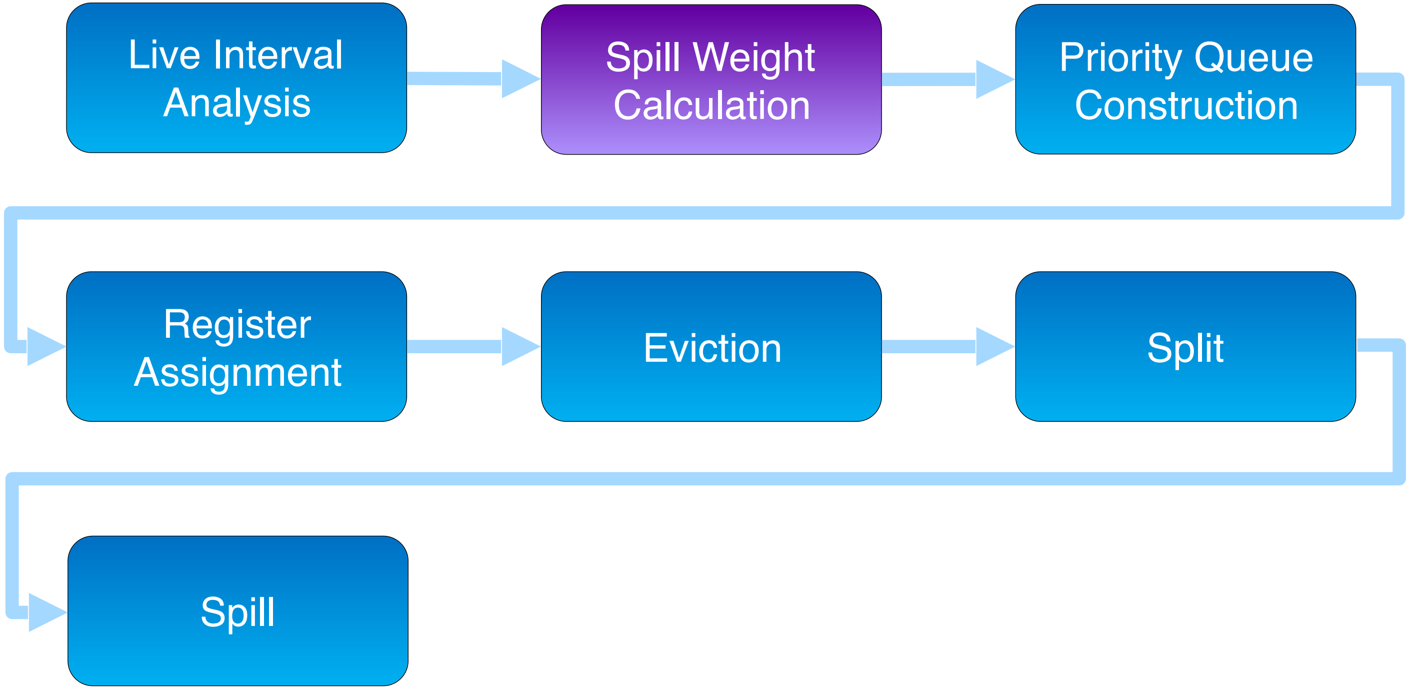
Greedy Register Allocator Overview

- General flow



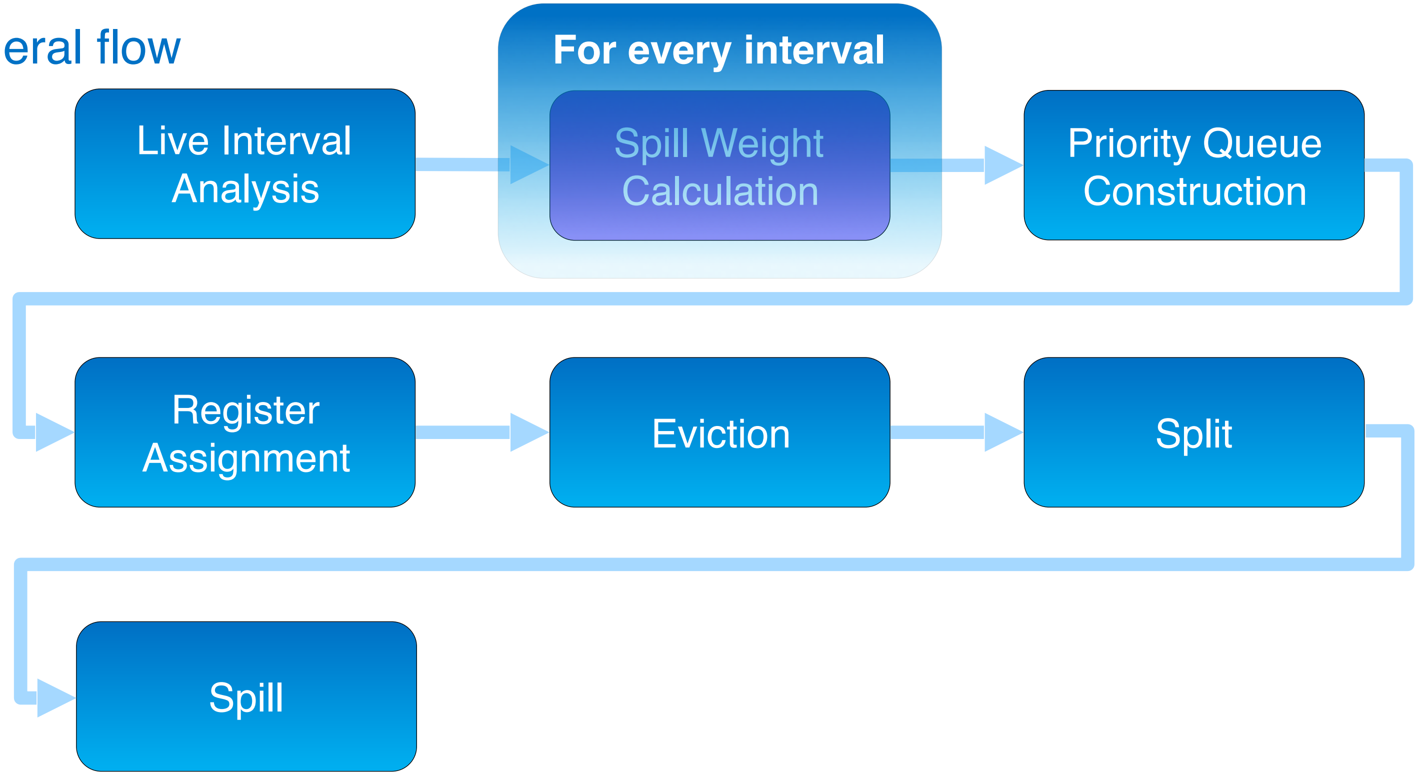
Greedy Register Allocator Overview

- General flow



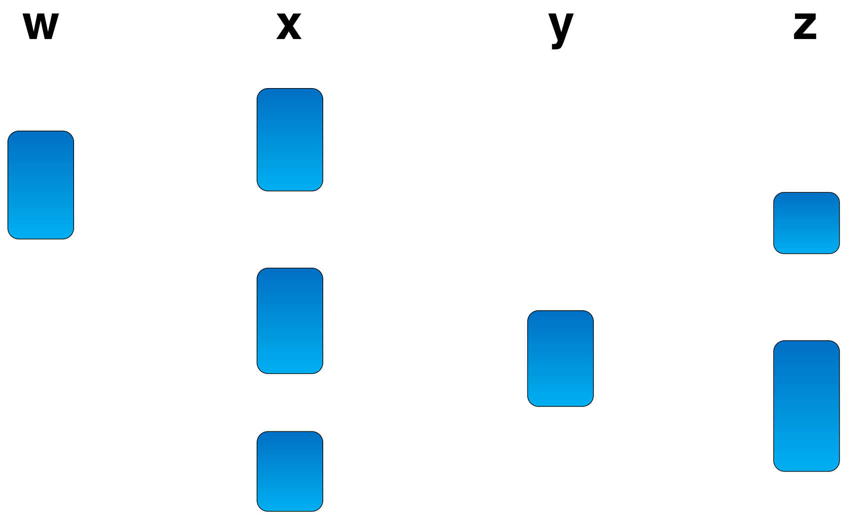
Greedy Register Allocator Overview

- General flow



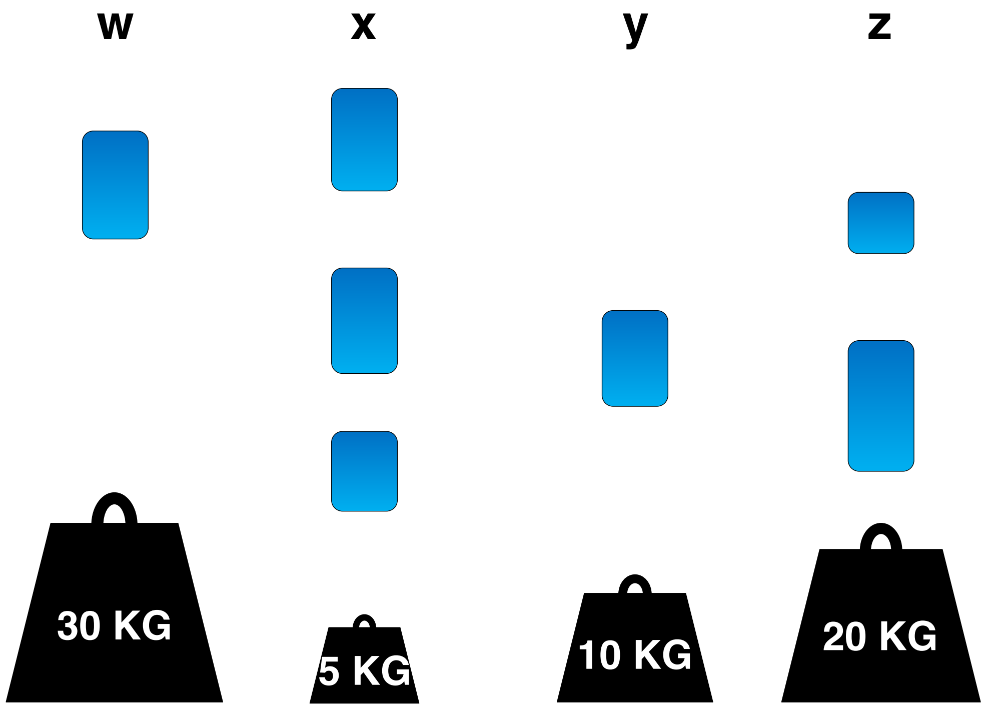
Spill Weight Calculation

- Intervals calculated by Live Interval Analysis



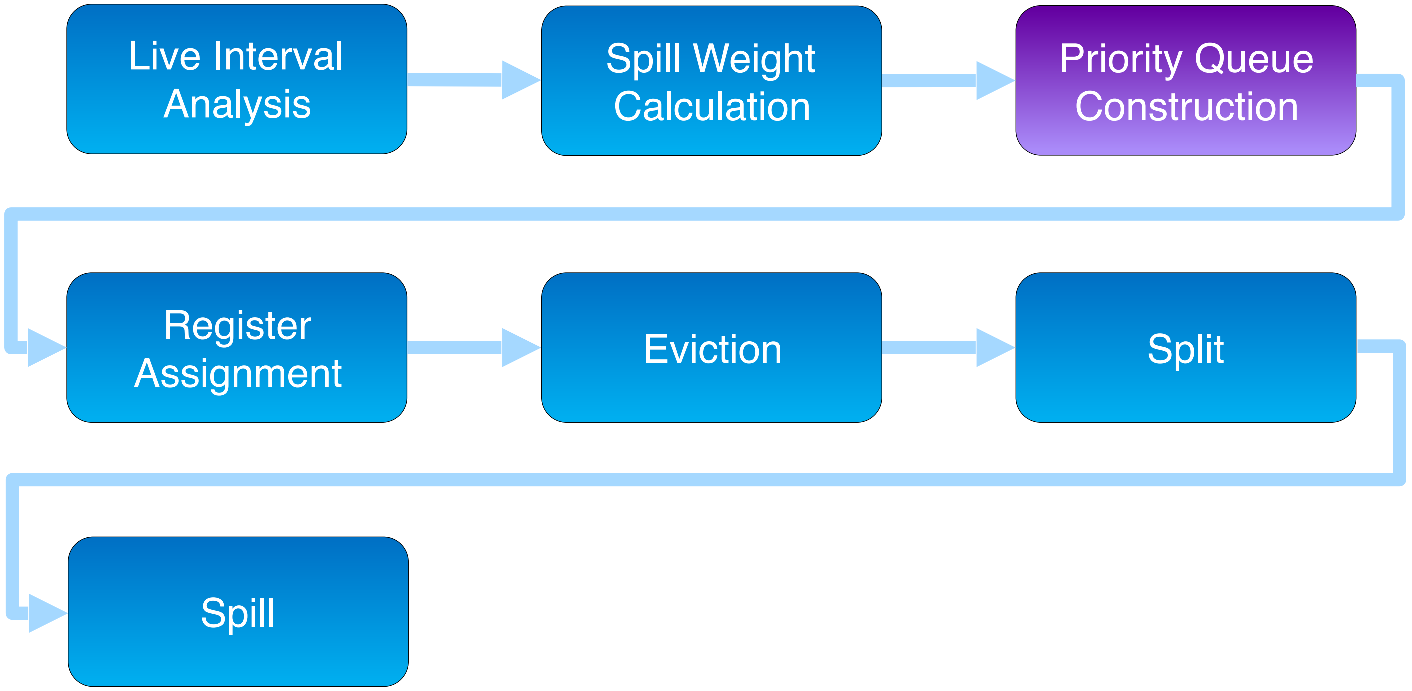
Spill Weight Calculation

- Estimate spill weight of each interval based on interval characteristics
  - w has uses in a hot loop
    - Higher spill weight
  - x is cheaply rematerializable
    - Lower spill weight



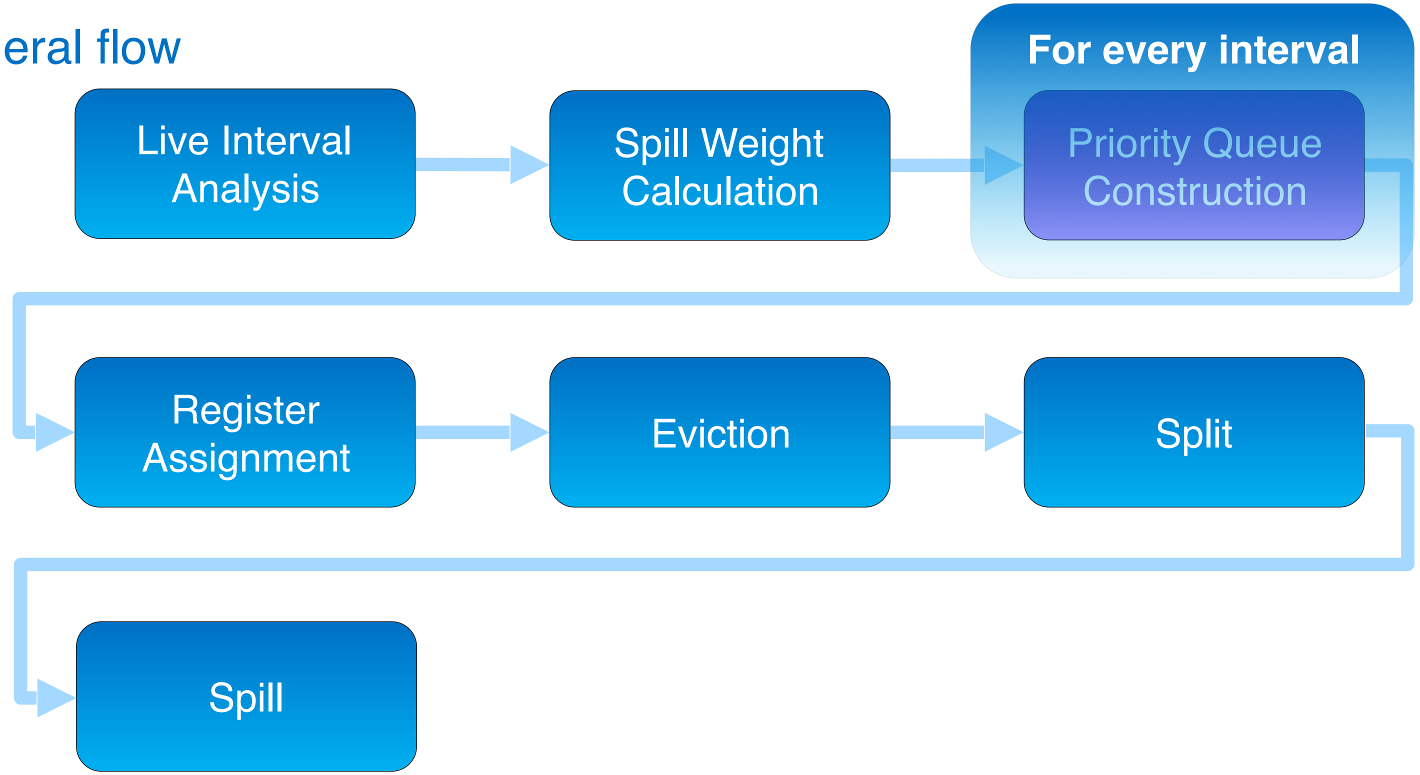
Greedy Register Allocator Overview

- General flow

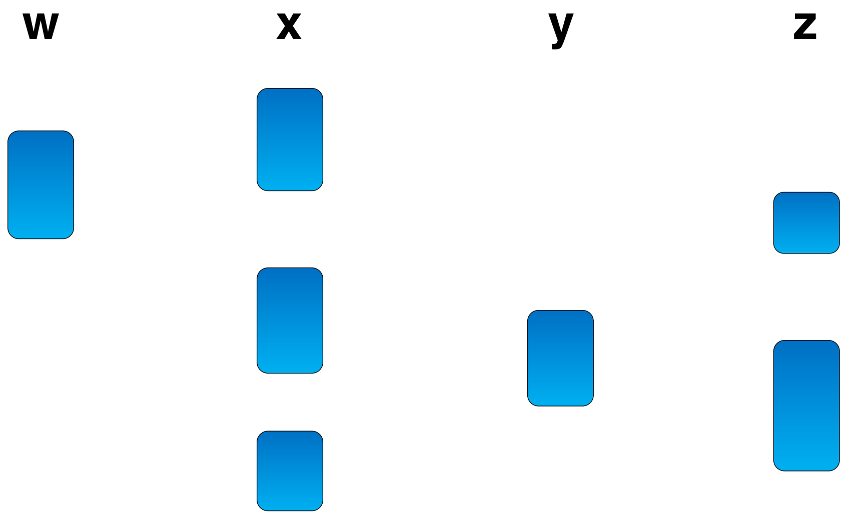


Greedy Register Allocator Overview

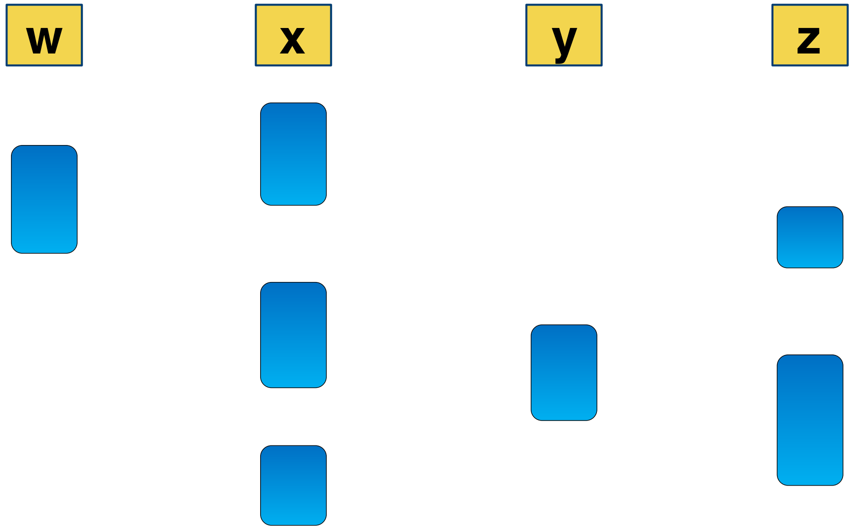
- General flow



Priority Queue Construction



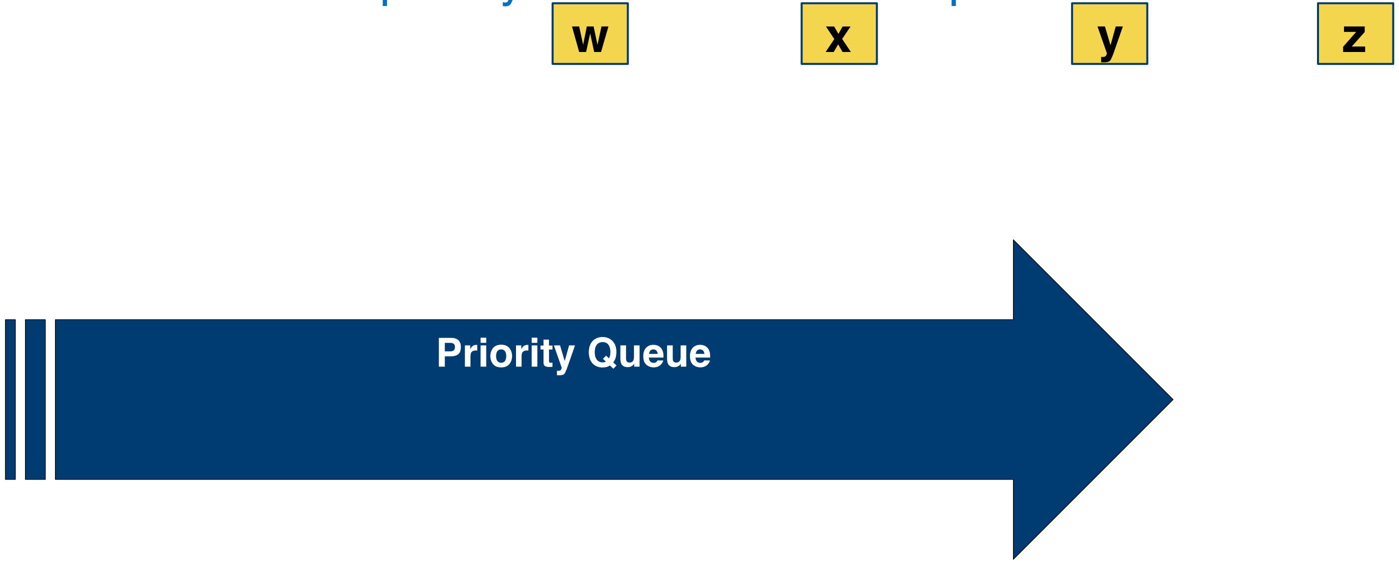
Priority Queue Construction



Priority Queue Construction

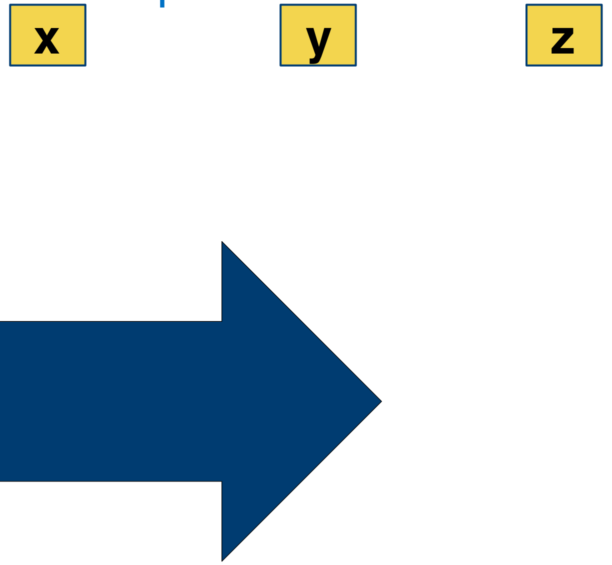
Priority Queue Construction

- Calculate interval allocation priority and insert into the queue



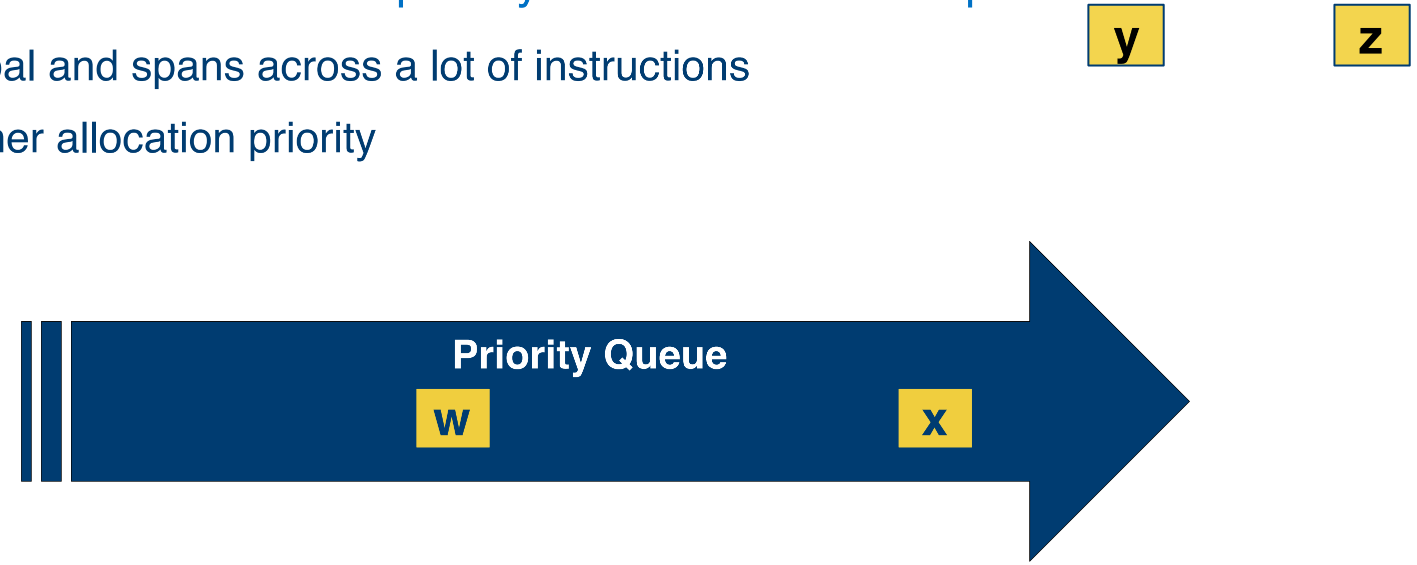
Priority Queue Construction

- Calculate interval allocation priority and insert into the queue
  - w is local in one basic block
    - Lower allocation priority



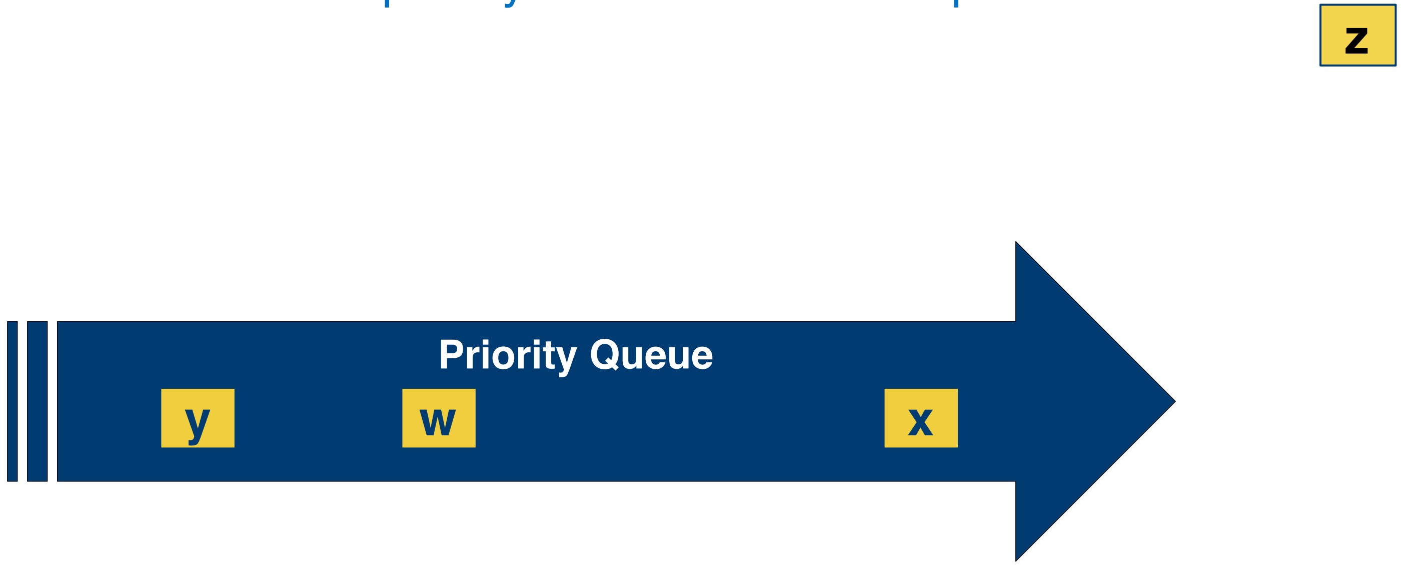
Priority Queue Construction

- Calculate interval allocation priority and insert into the queue
  - x is global and spans across a lot of instructions
    - Higher allocation priority



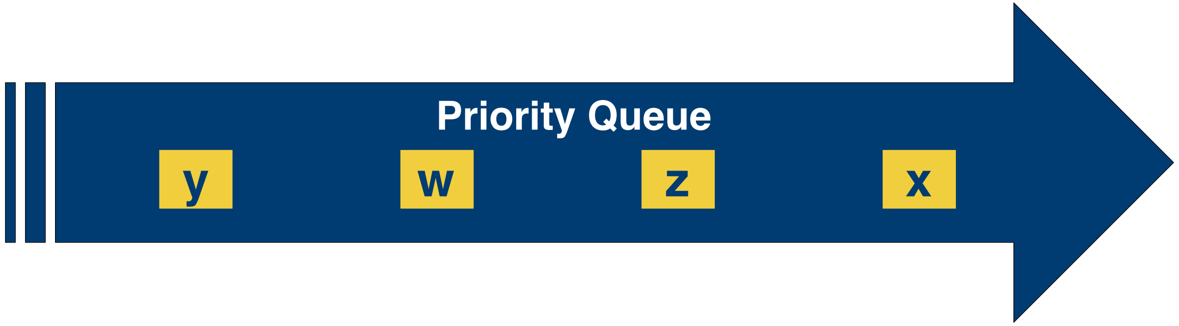
Priority Queue Construction

- Calculate interval allocation priority and insert into the queue



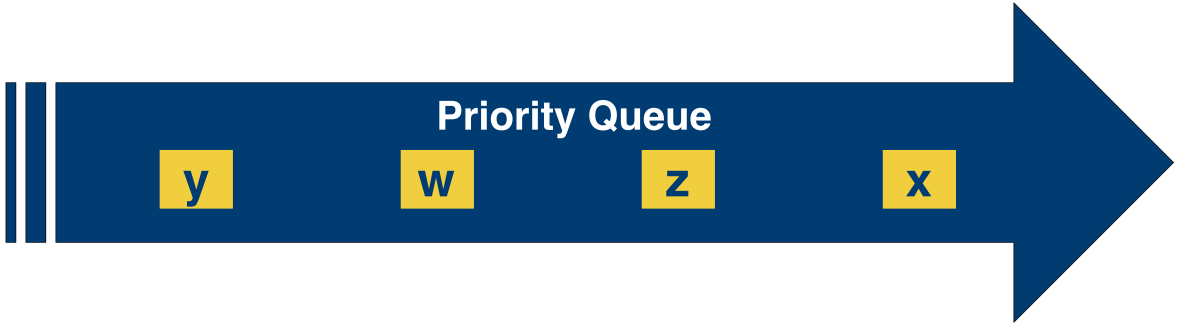
Priority Queue Construction

- Calculate interval allocation priority and insert into the queue



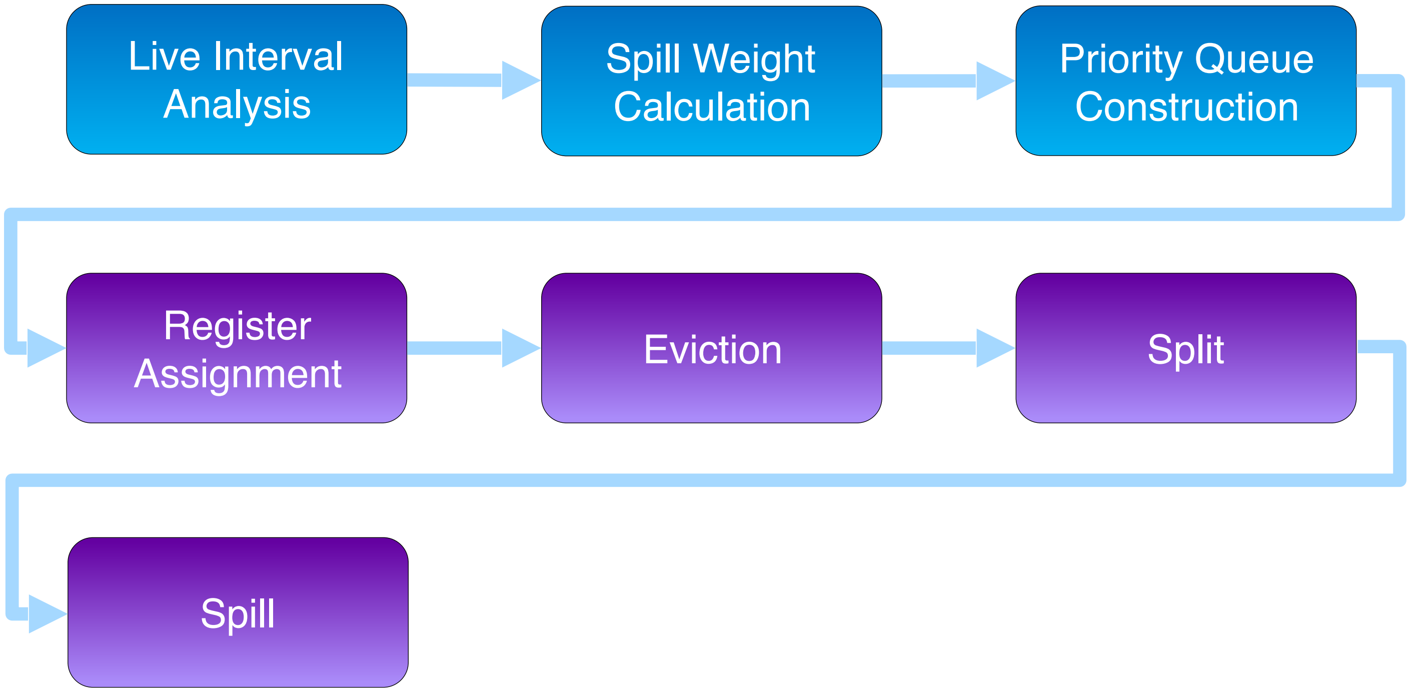
Priority Queue Construction

- The Priority Queue will always dequeue the interval with the highest priority



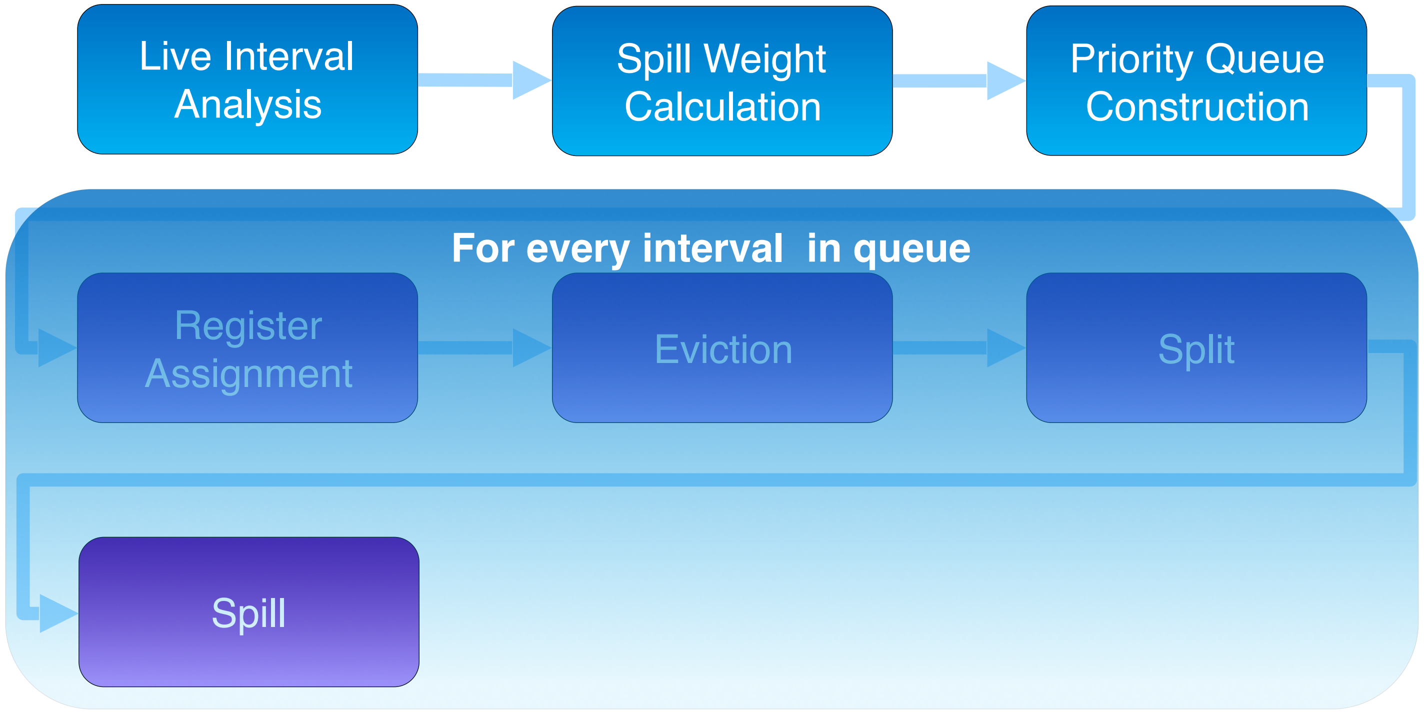
Greedy Register Allocator Overview

- General flow



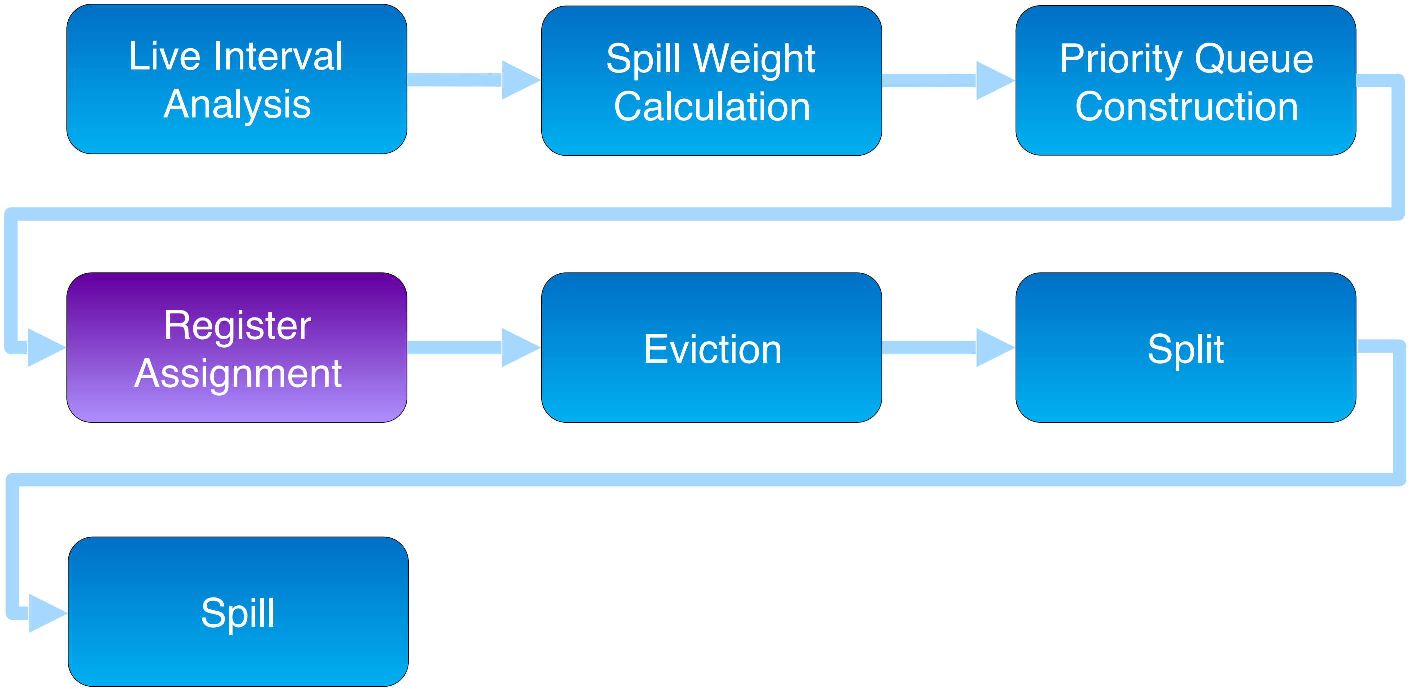
Greedy Register Allocator Overview

- General flow



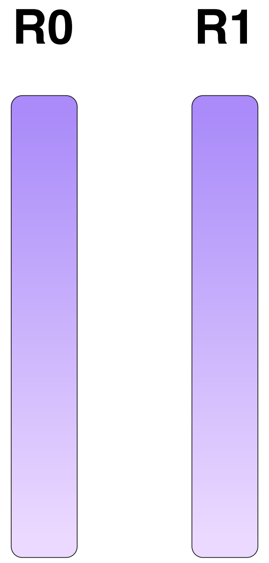
Greedy Register Allocator Overview

- General flow

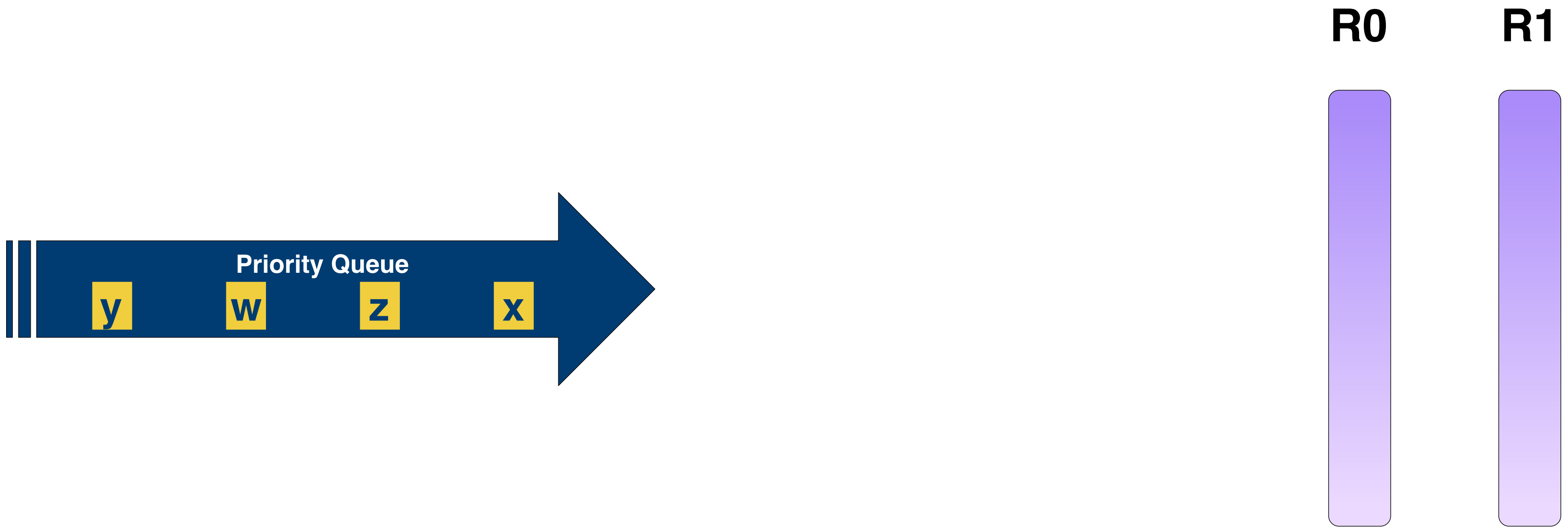


Register Assignment

- R0, R1 are the physical registers in the system

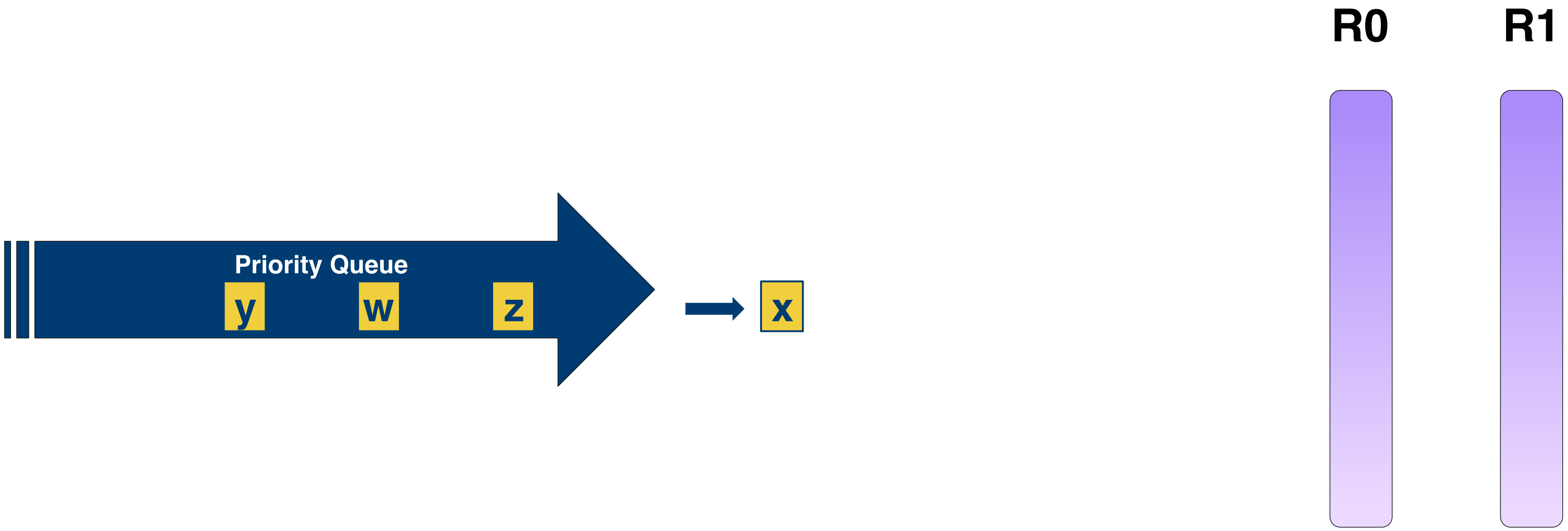


Register Assignment

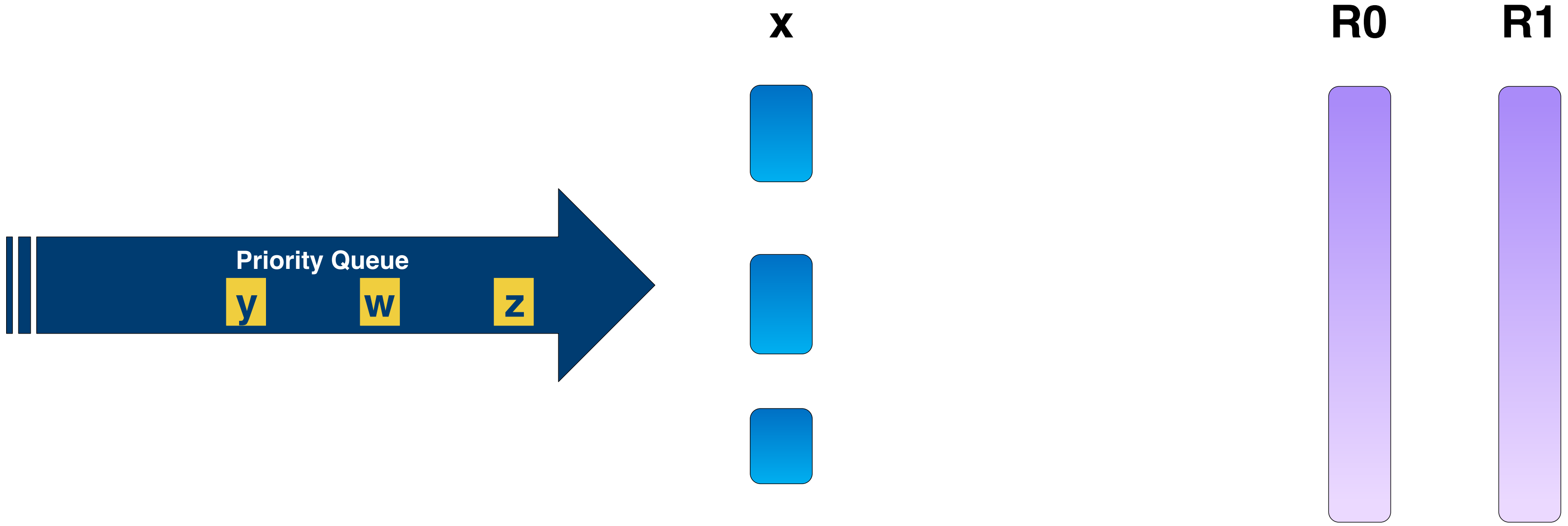


Register Assignment

- Dequeue interval with highest priority

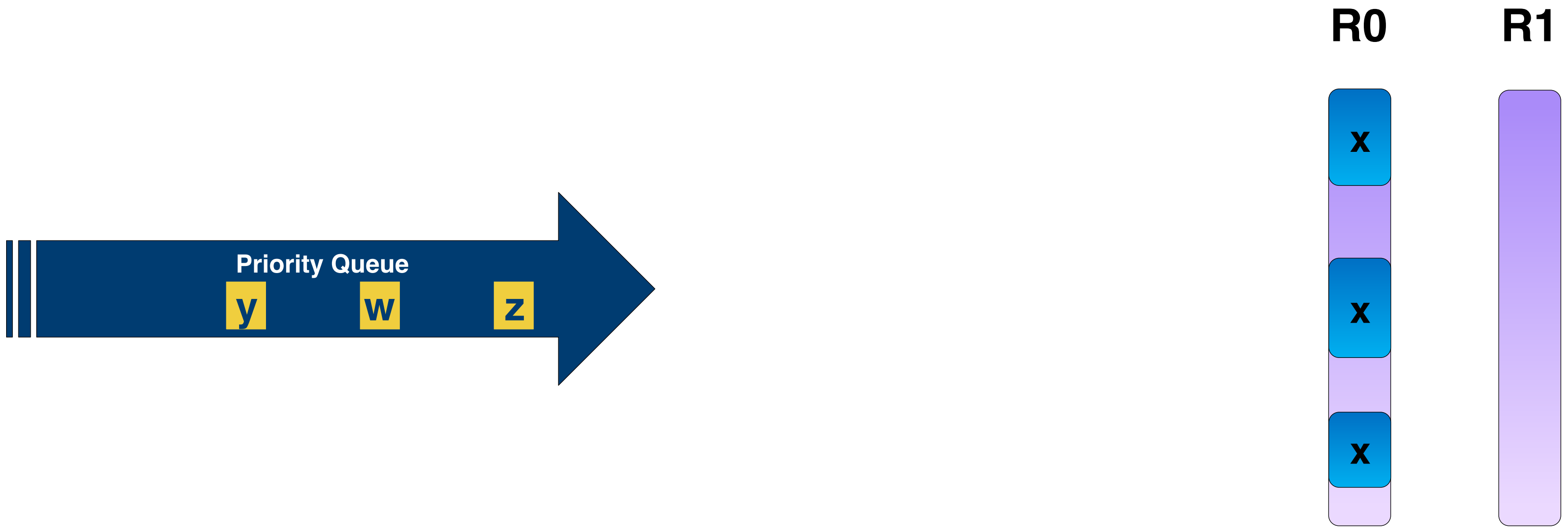


Register Assignment



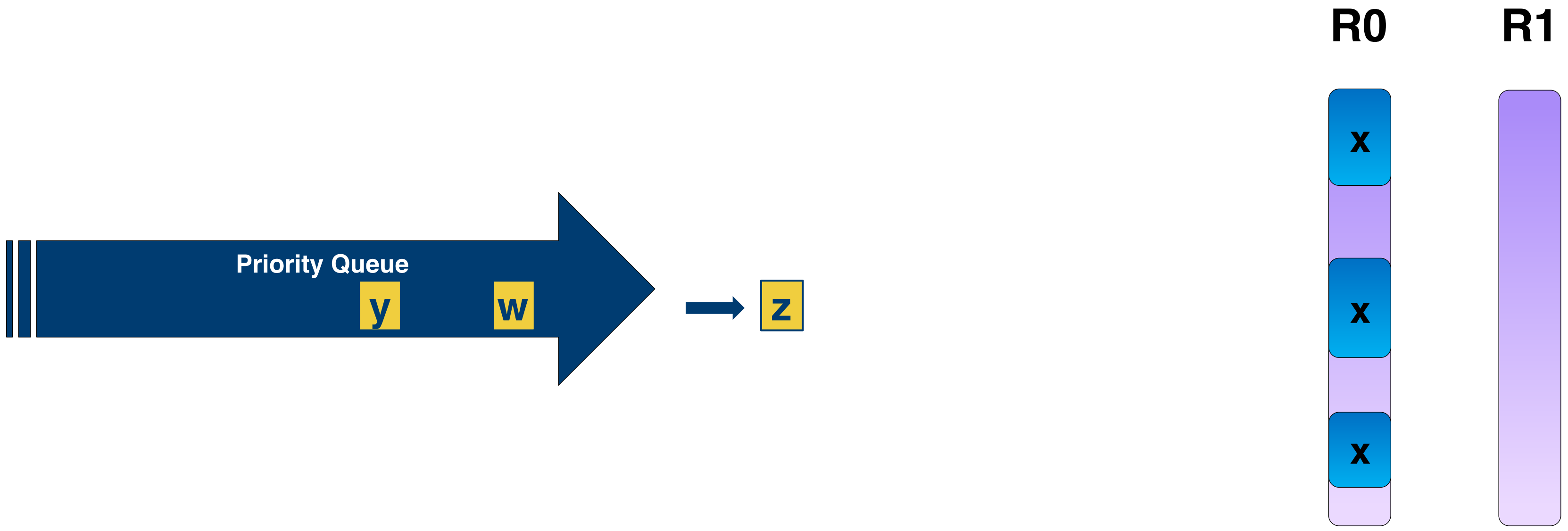
Register Assignment

- Assign to available register if possible

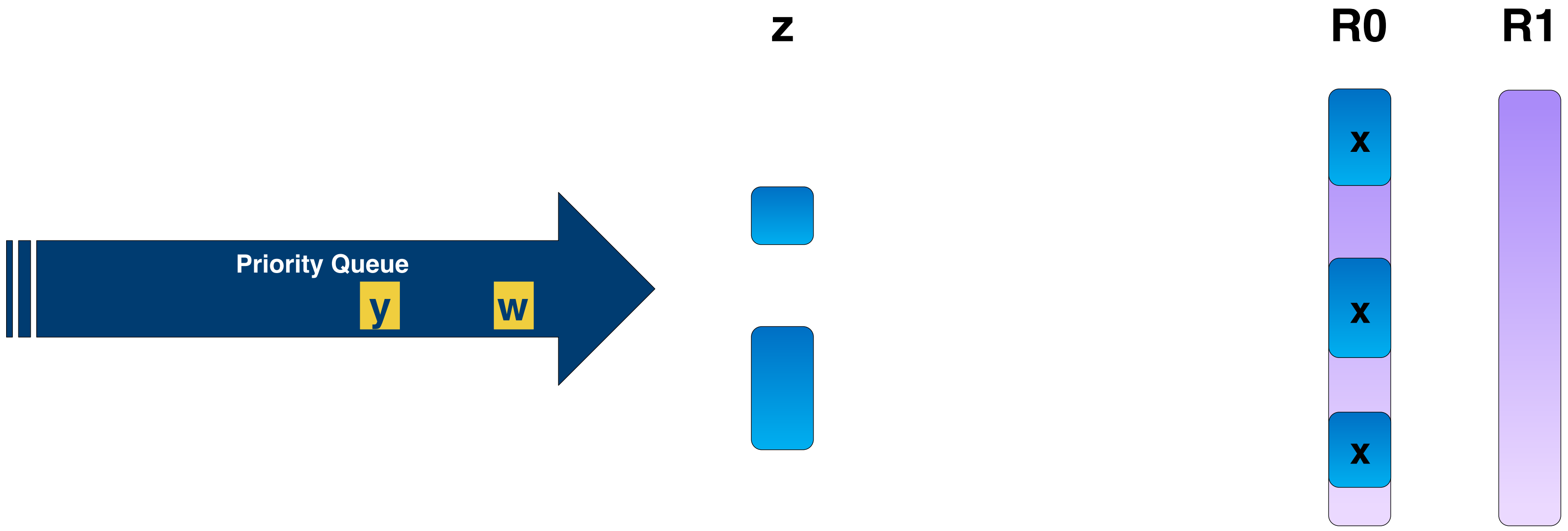


Register Assignment

- Dequeue interval with highest priority

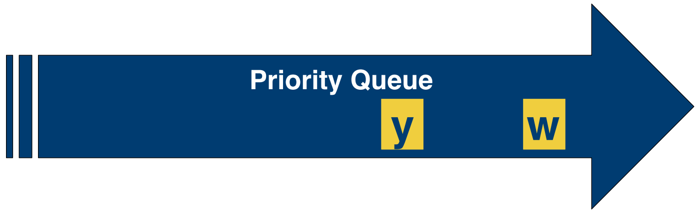


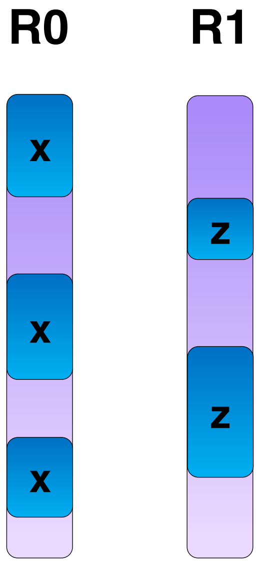
Register Assignment



Register Assignment

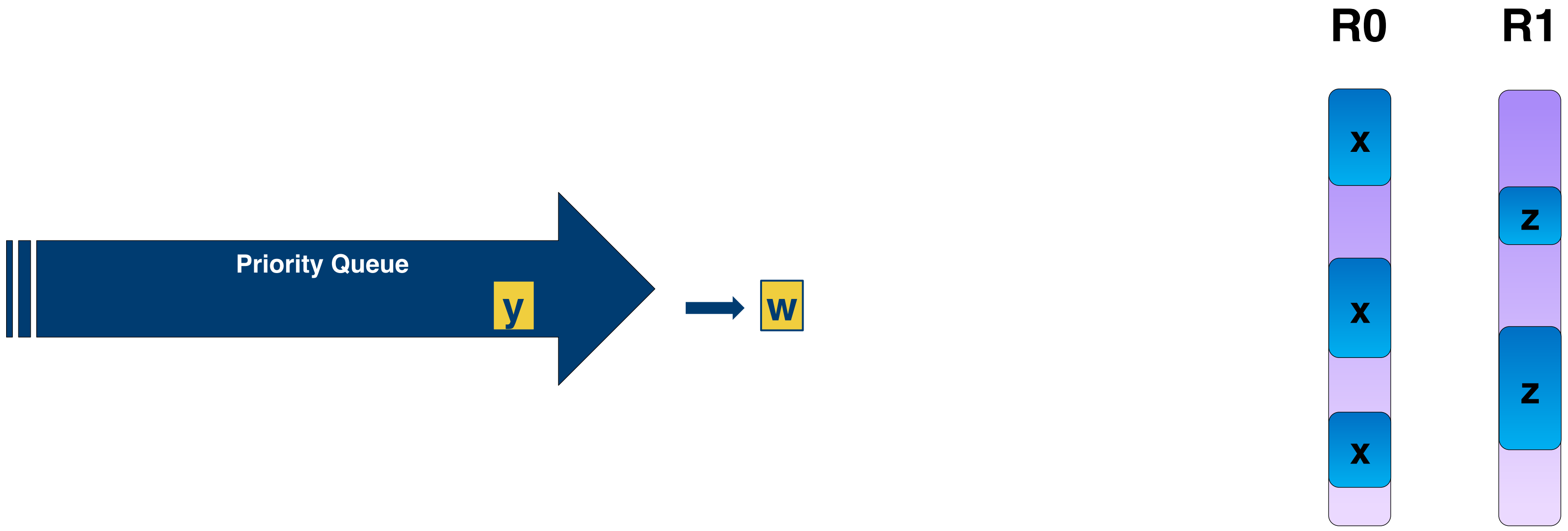
- Assign to available register if possible



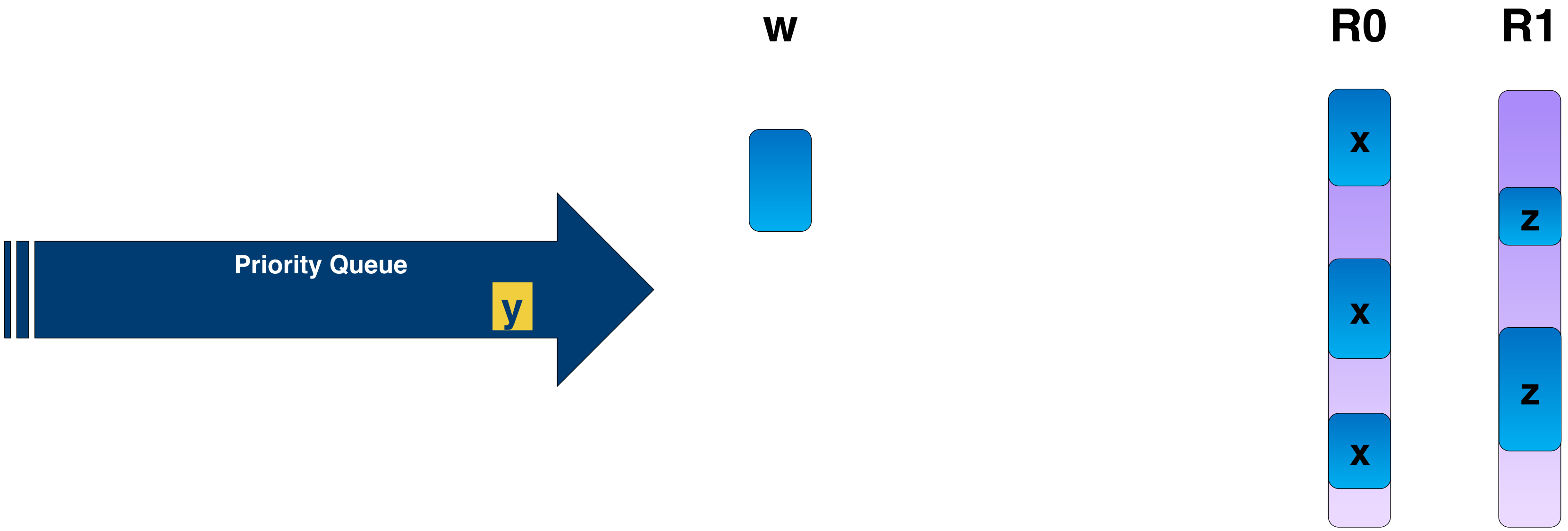


Register Assignment

- Dequeue interval with highest priority

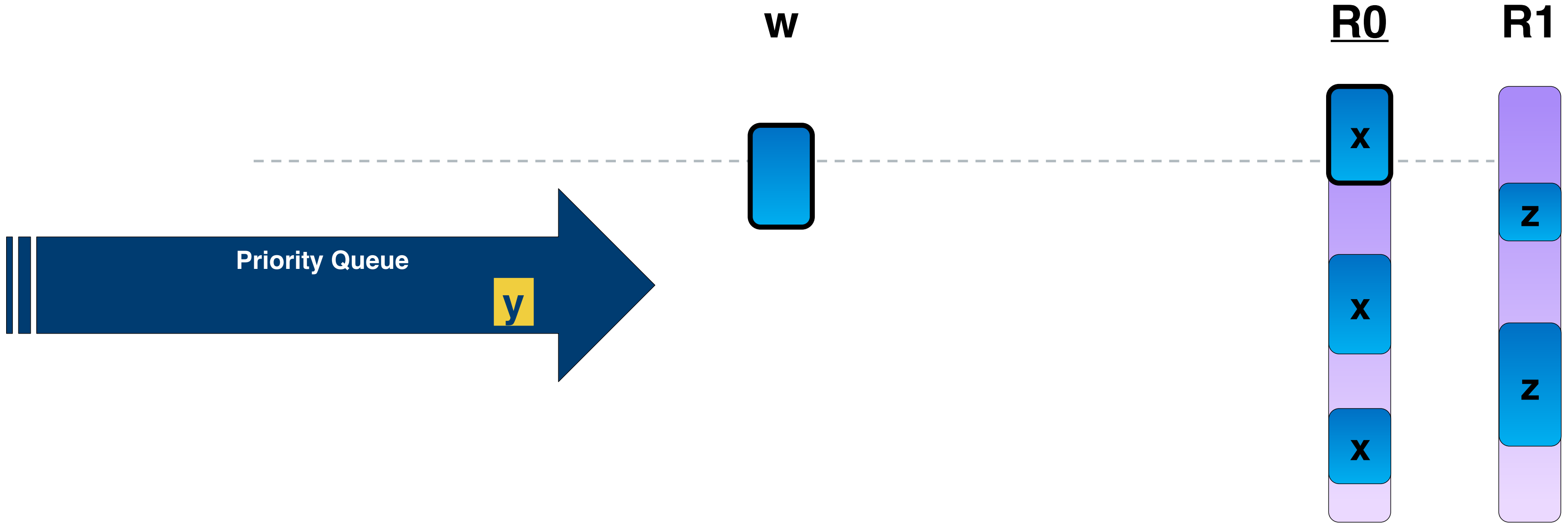


Register Assignment



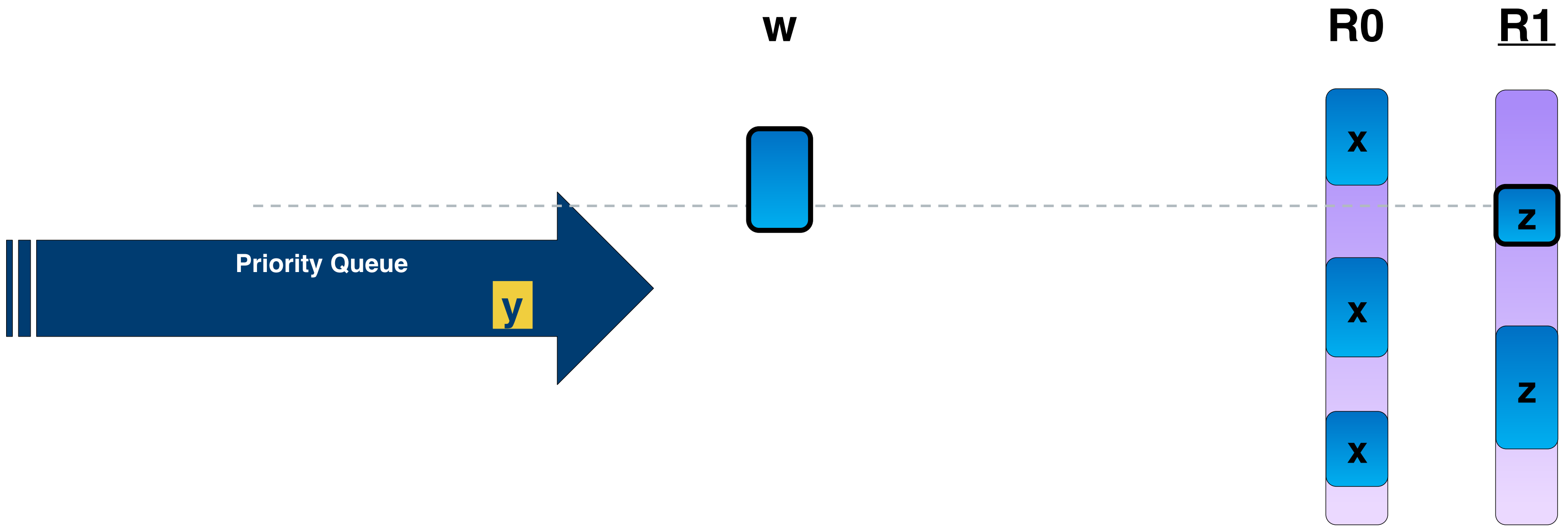
Register Assignment

- Interference with x in R0



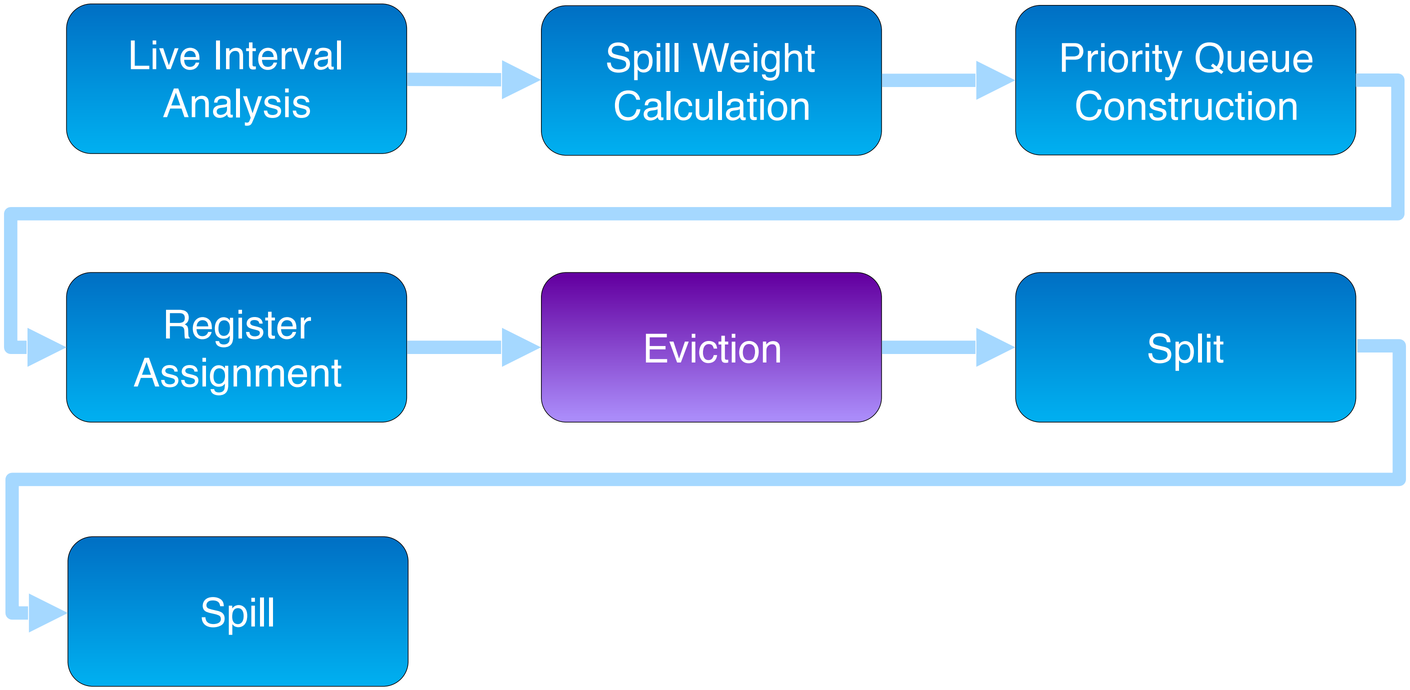
Register Assignment

- Interference with z in R1

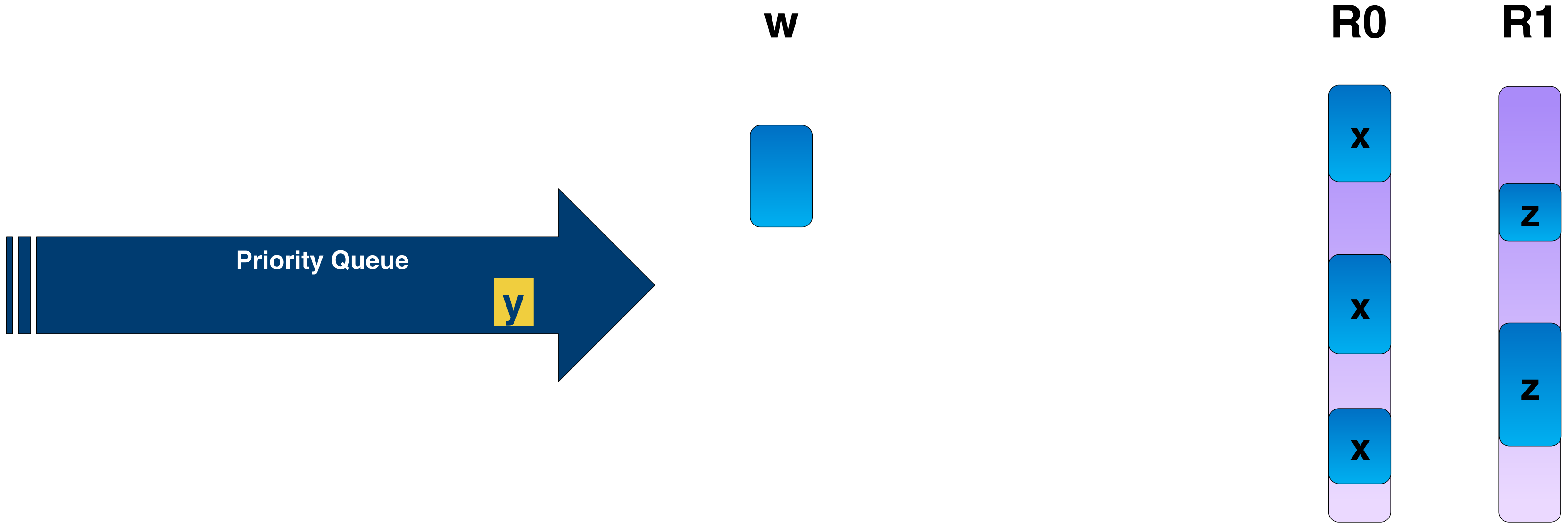


Greedy Register Allocator Overview

- General flow

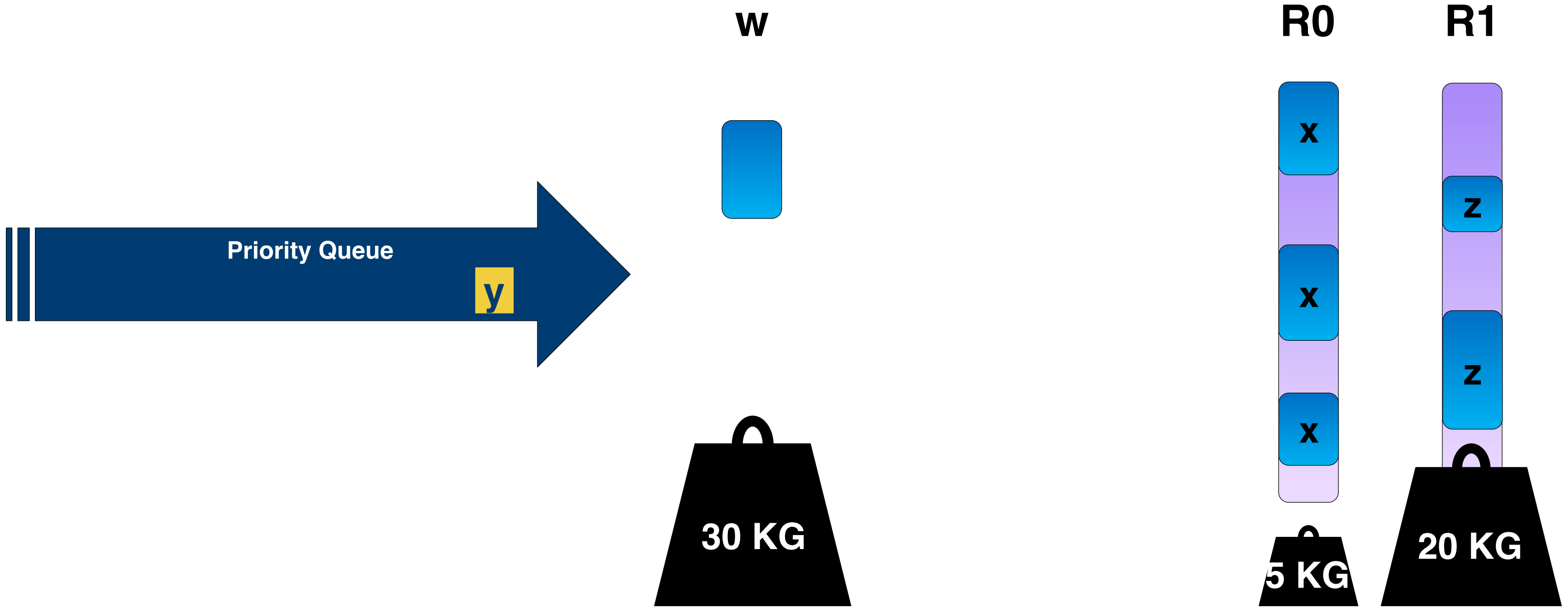


Eviction



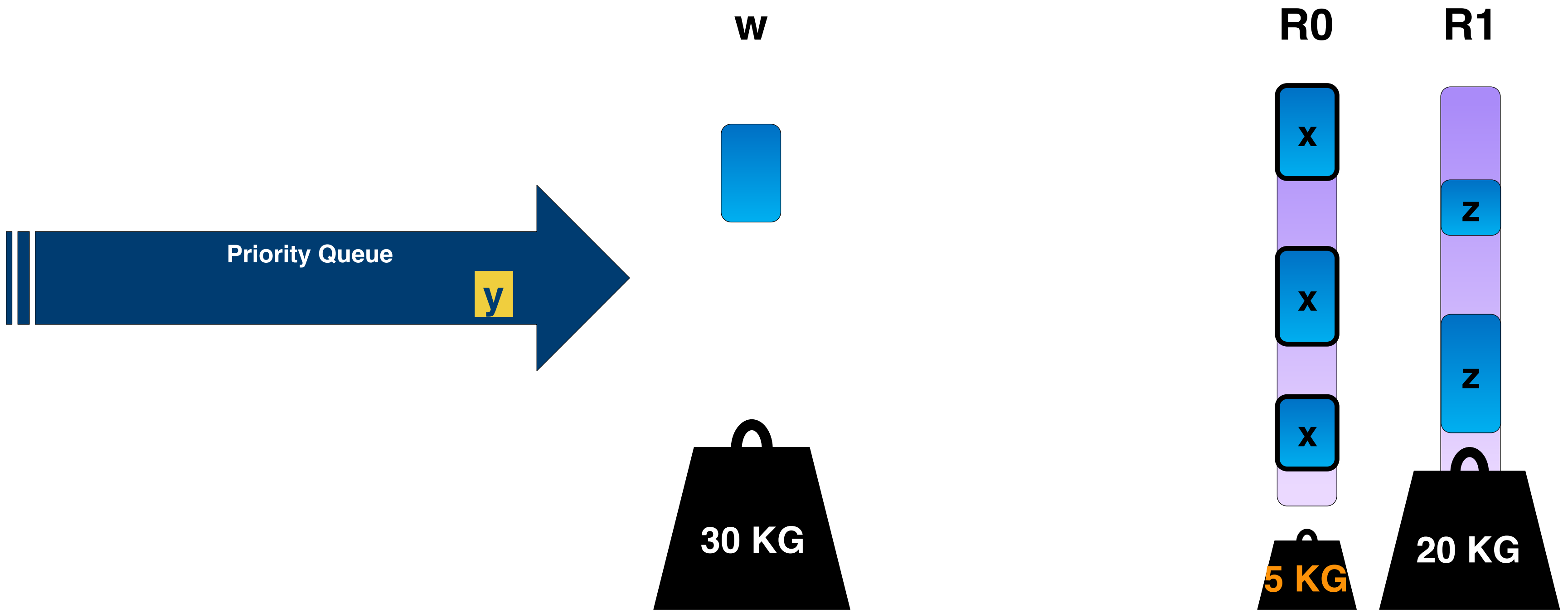
Eviction

- Compare spill weights of interfering intervals



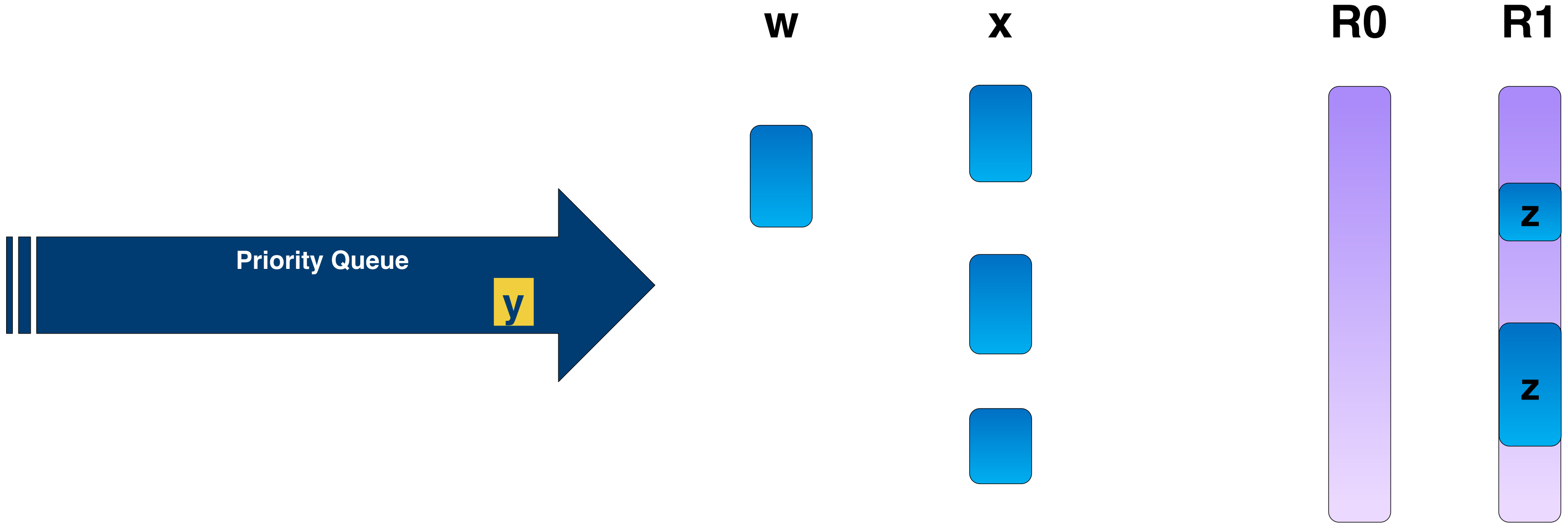
Eviction

- x cheaper than w



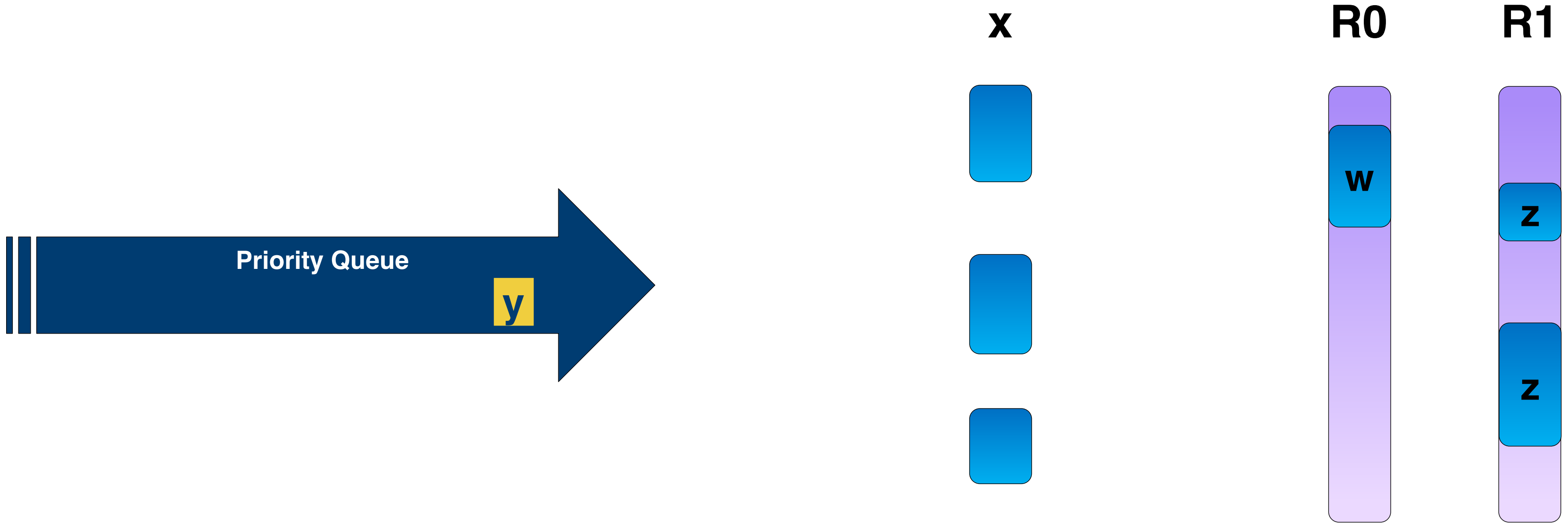
Eviction

- Evict x

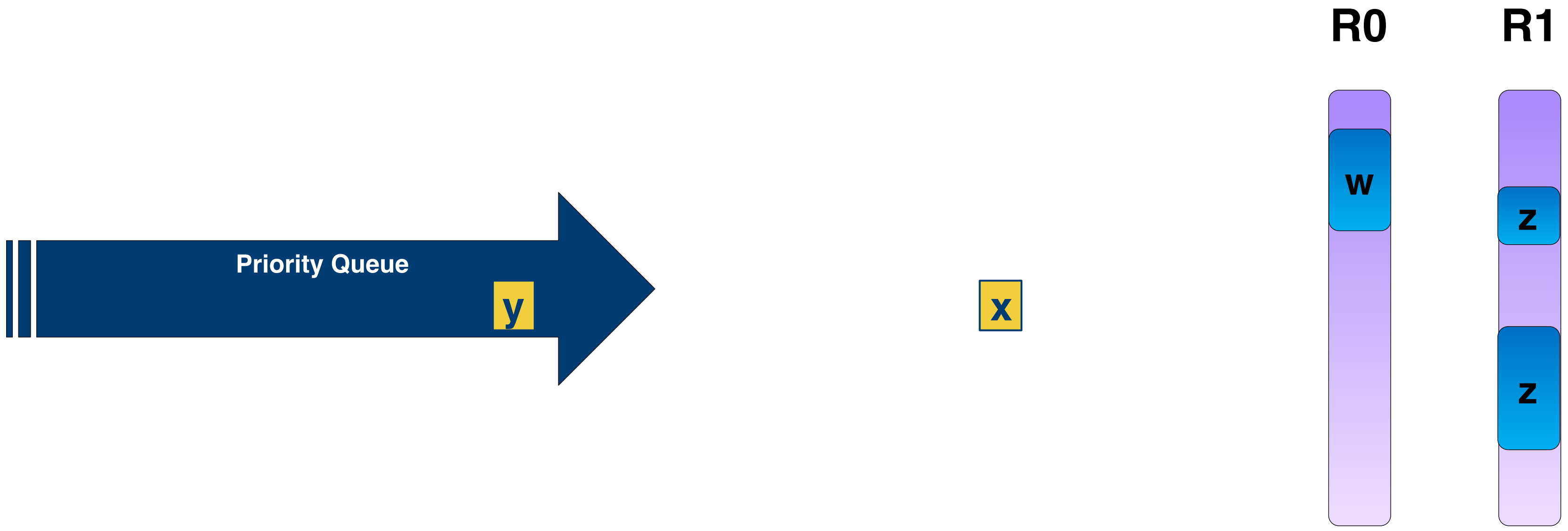


Eviction

- Assign w

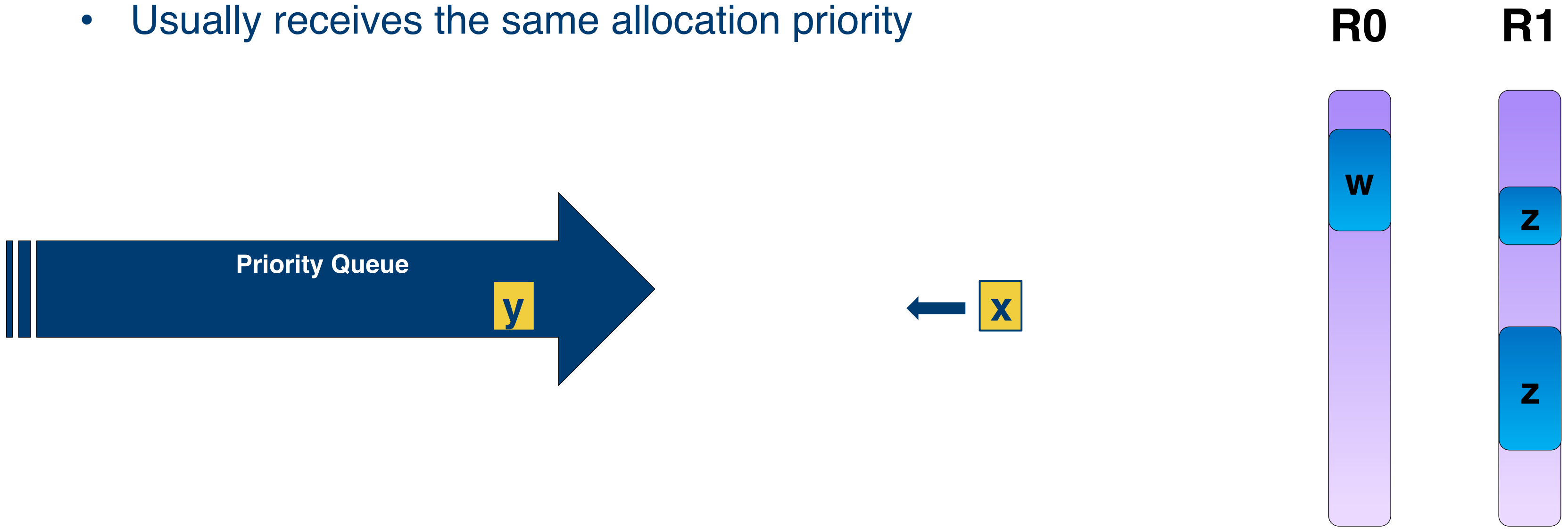


Eviction

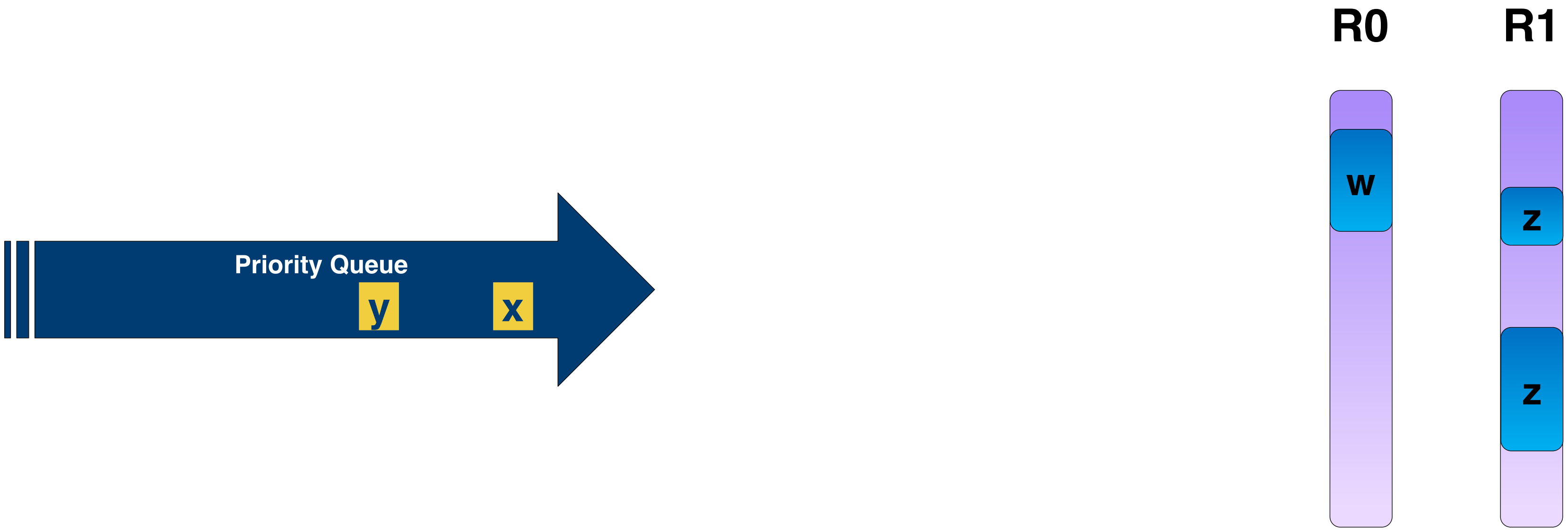


Eviction

- Enqueue x back to the queue
    - Usually receives the same allocation priority

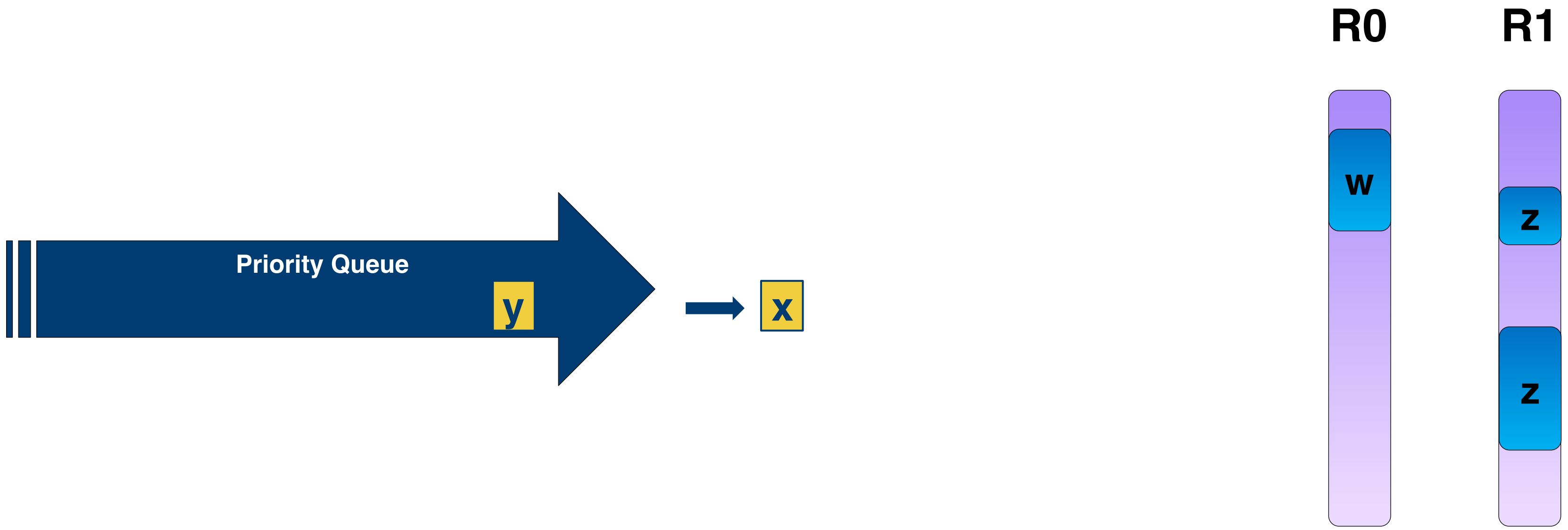


Register Assignment

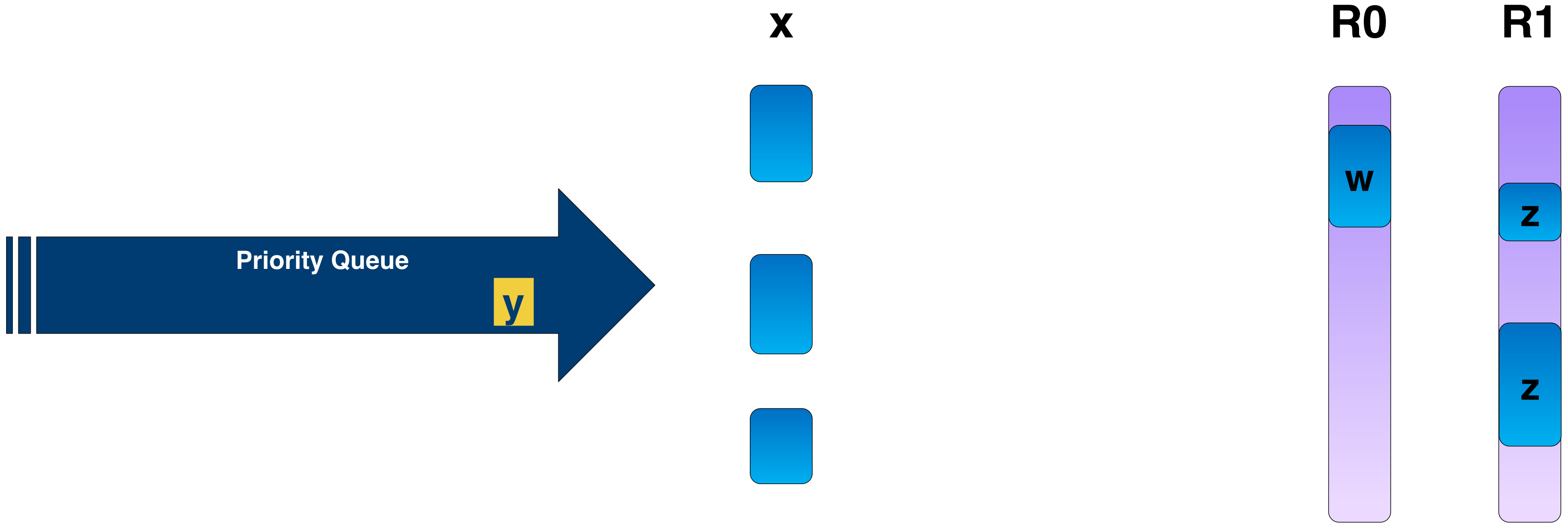


Register Assignment

- Dequeue interval with highest priority

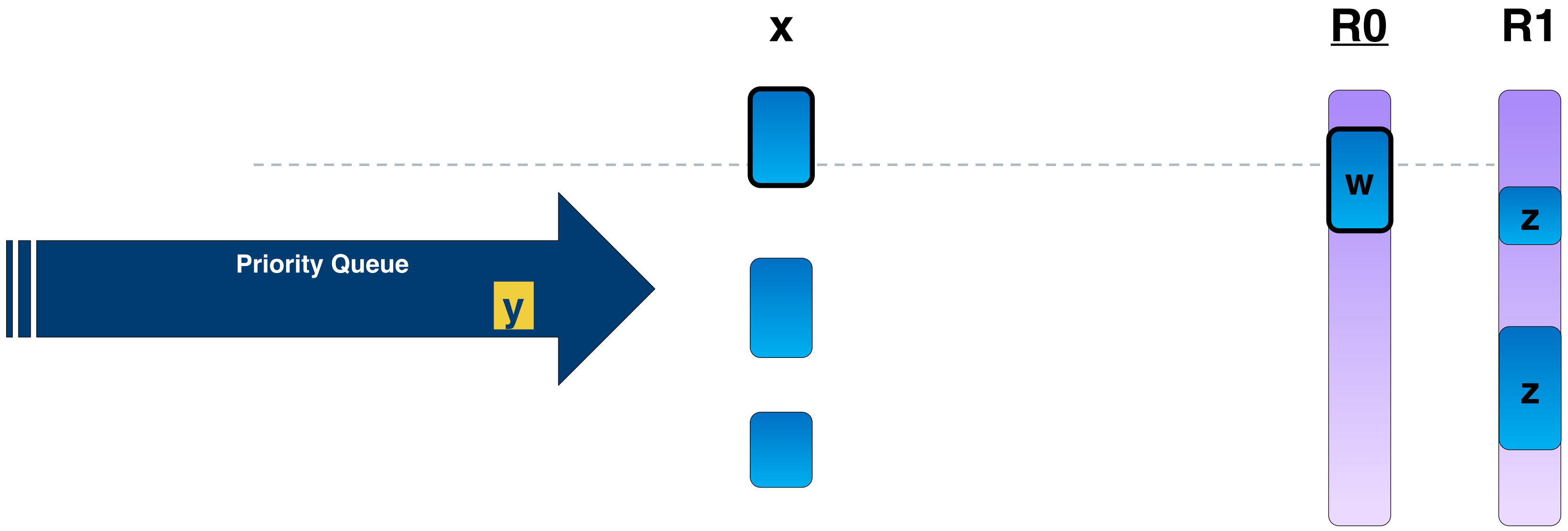


Register Assignment



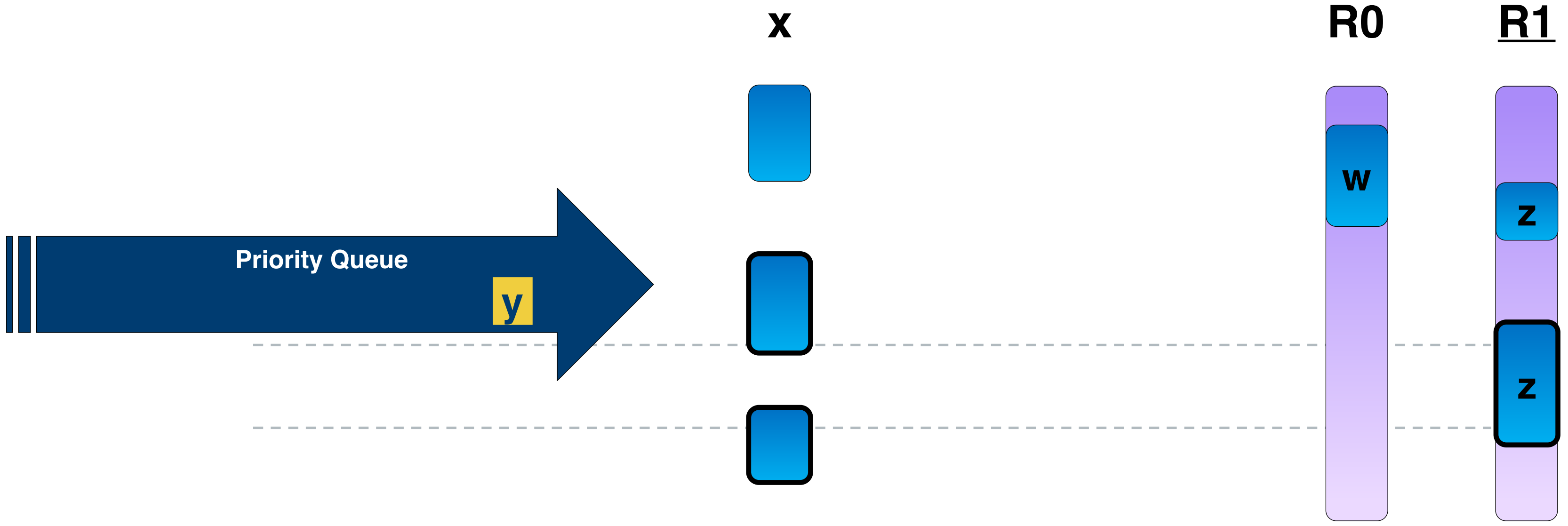
Register Assignment

- Interference with w in R0



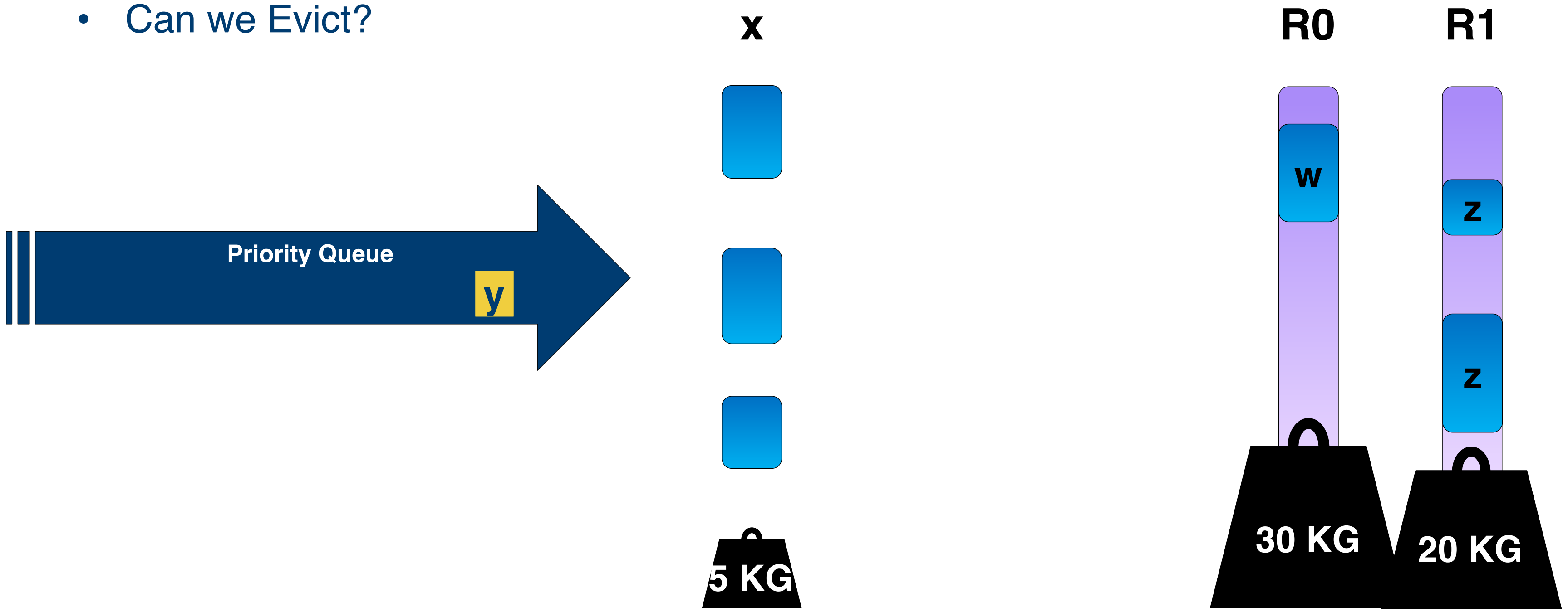
Register Assignment

- Interference with z in R1



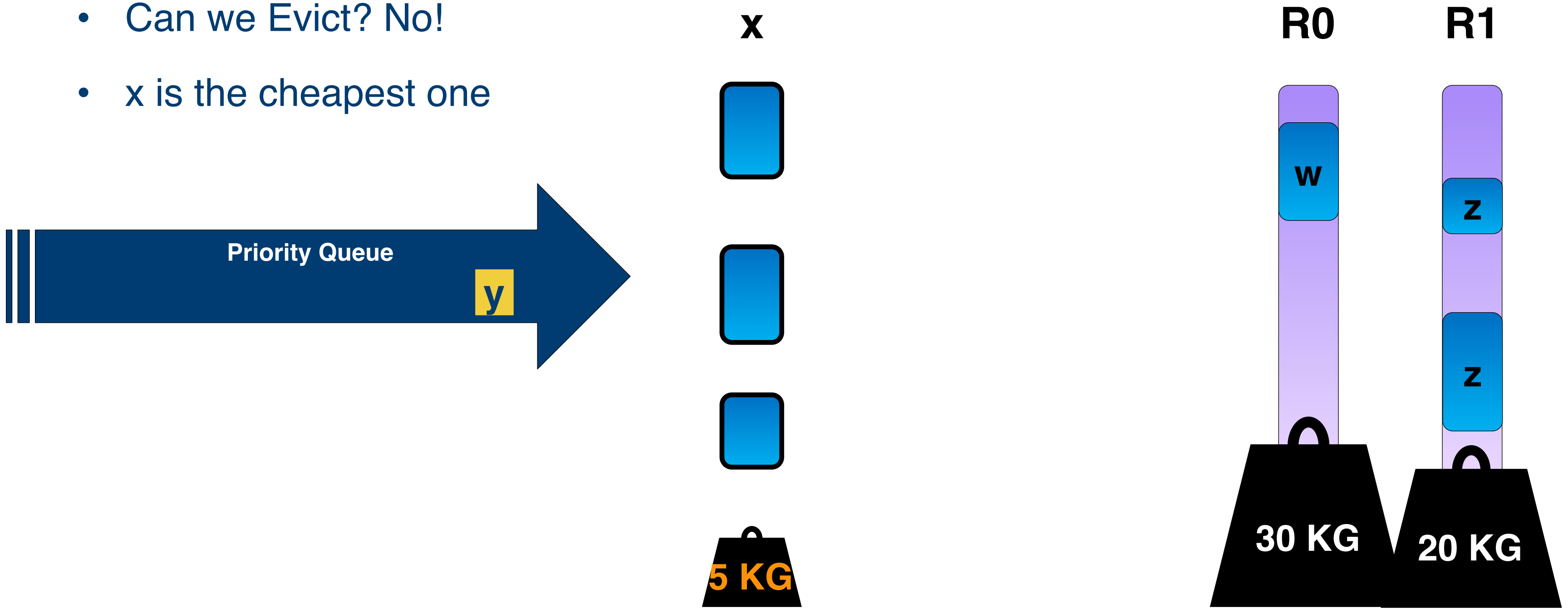
Eviction

- Compare spill weights of interfering intervals
  - Can we Evict?

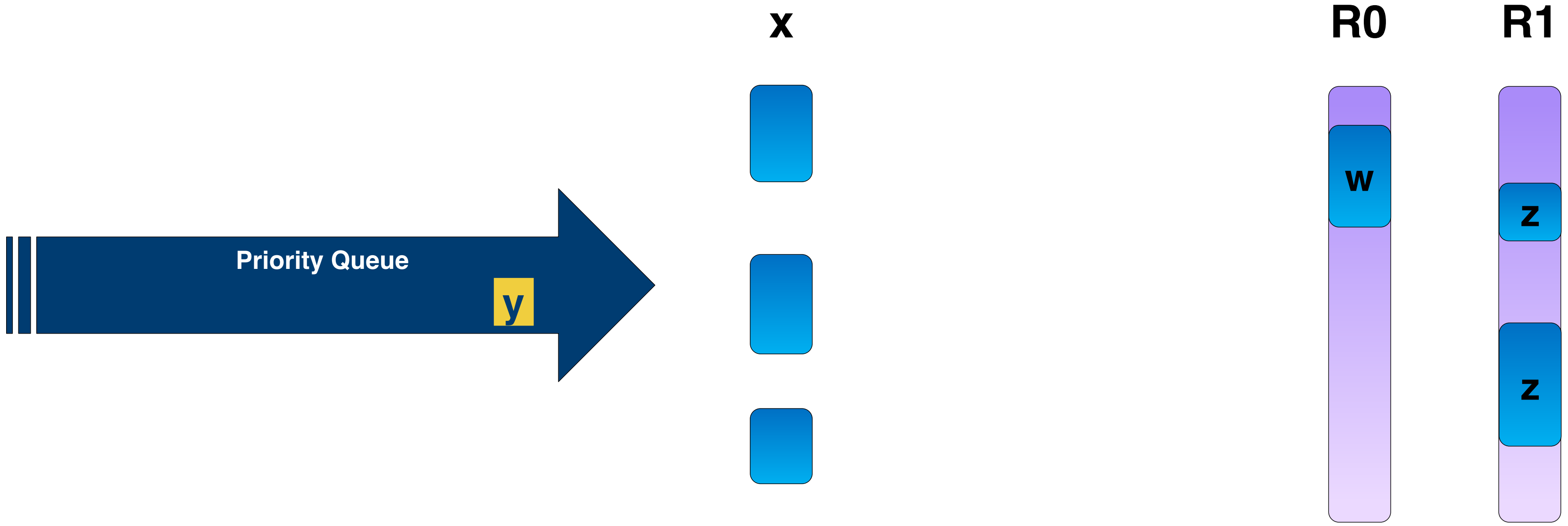


Eviction

- Compare spill weights of interfering intervals
  - Can we Evict? No!
  - x is the cheapest one

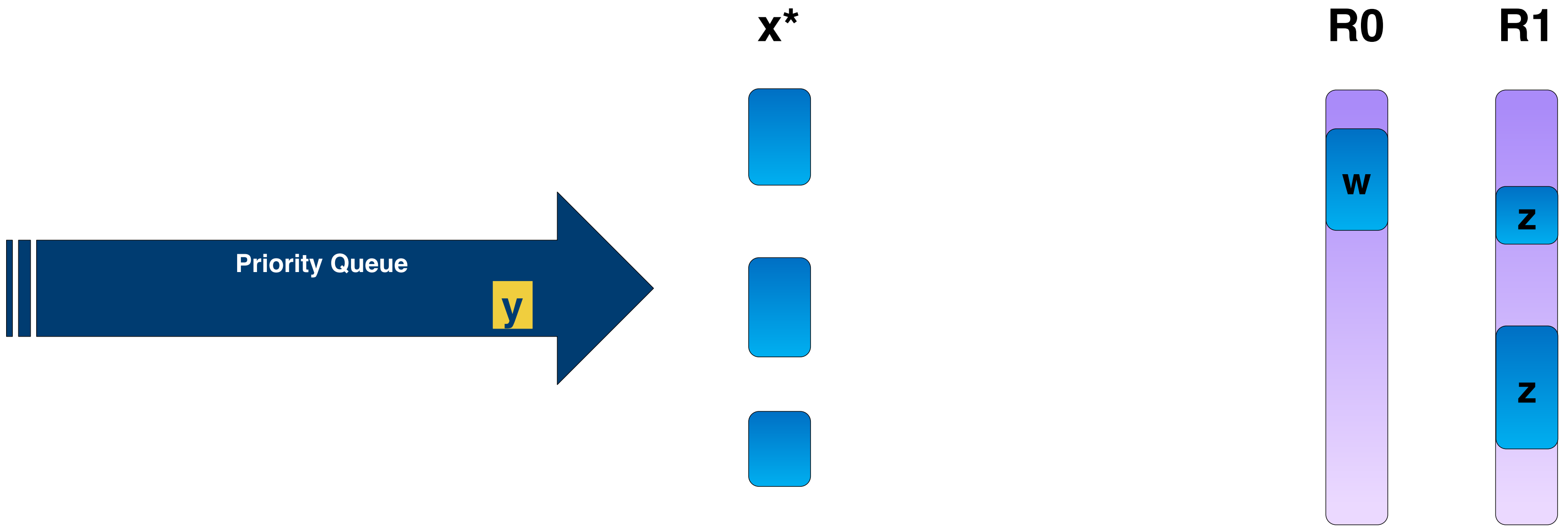


Register Assignment

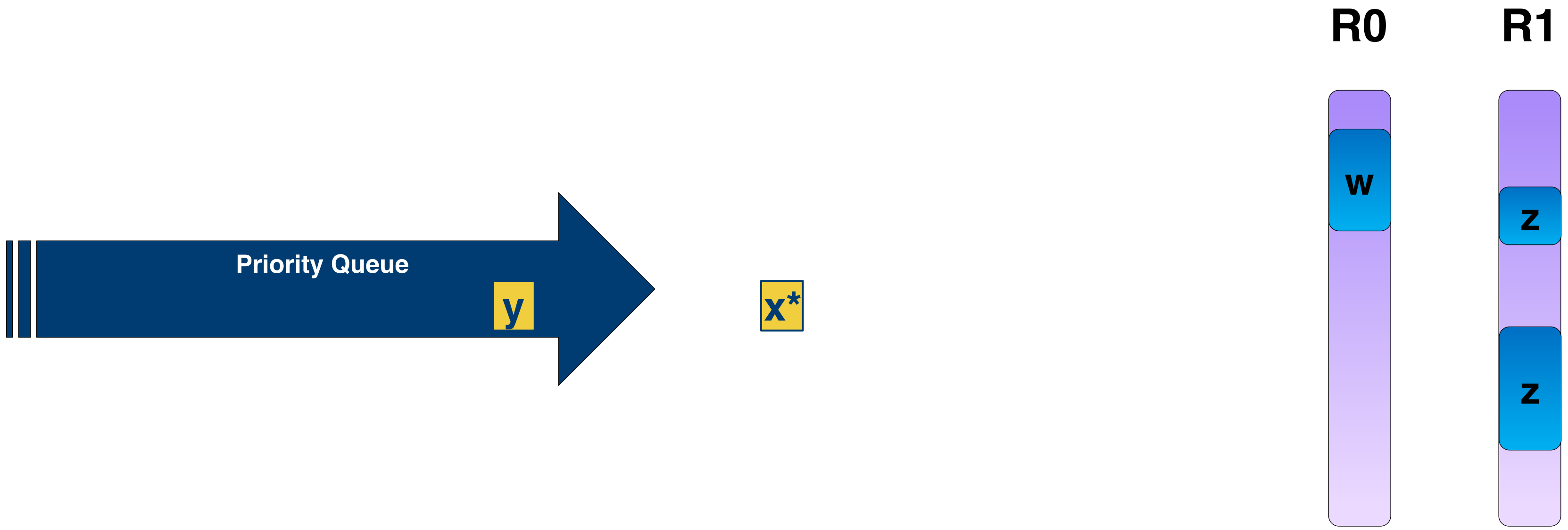


Register Assignment

- Mark x to be split

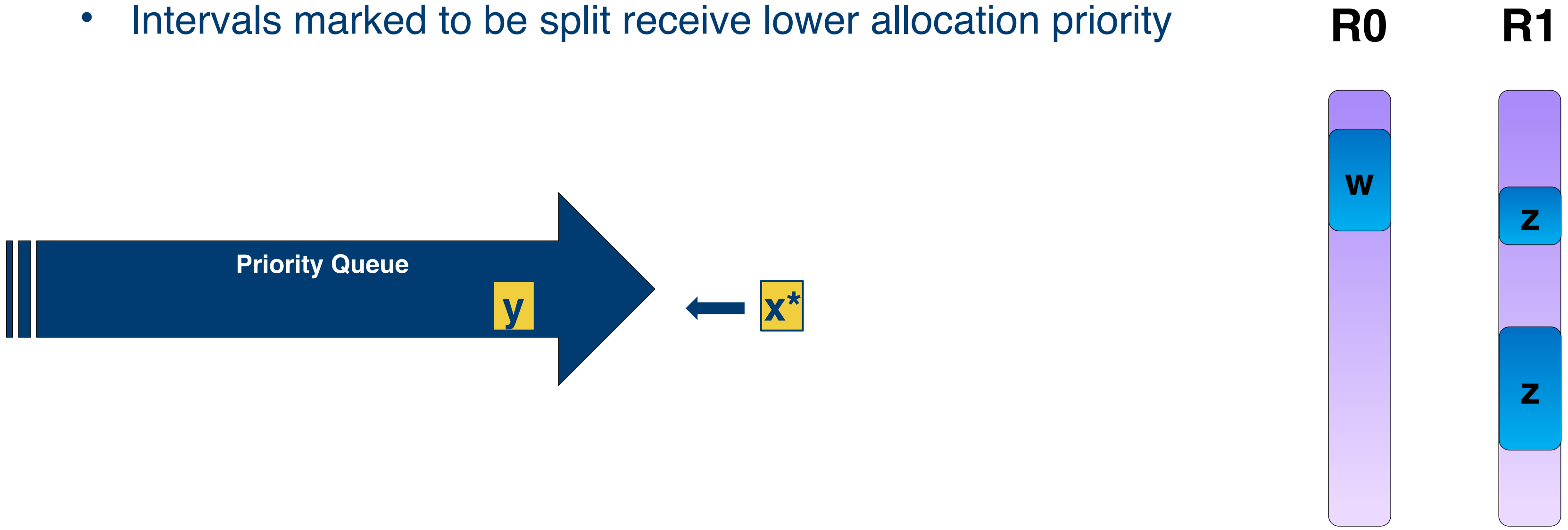


Register Assignment

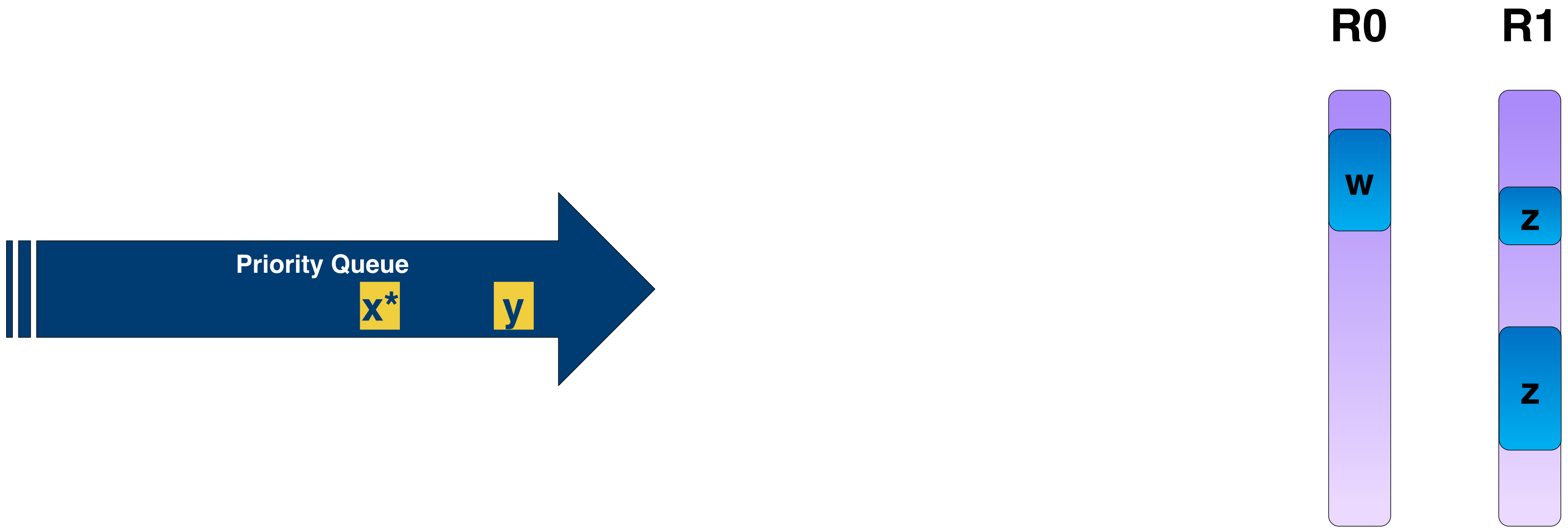


Register Assignment

- Enqueue x* back to the queue
  - Intervals marked to be split receive lower allocation priority

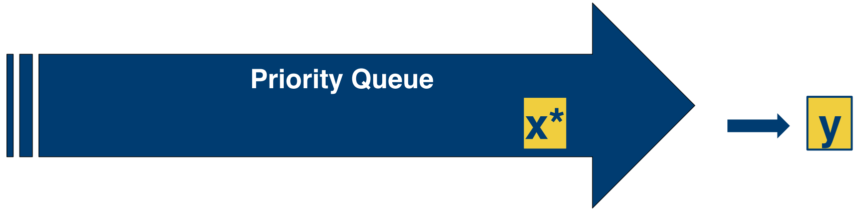


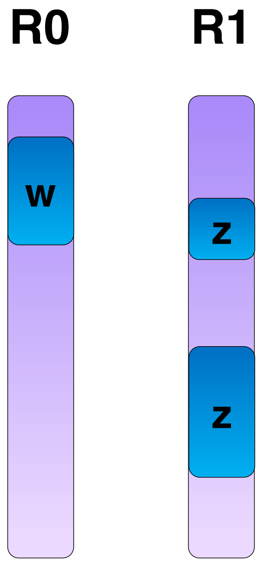
Register Assignment



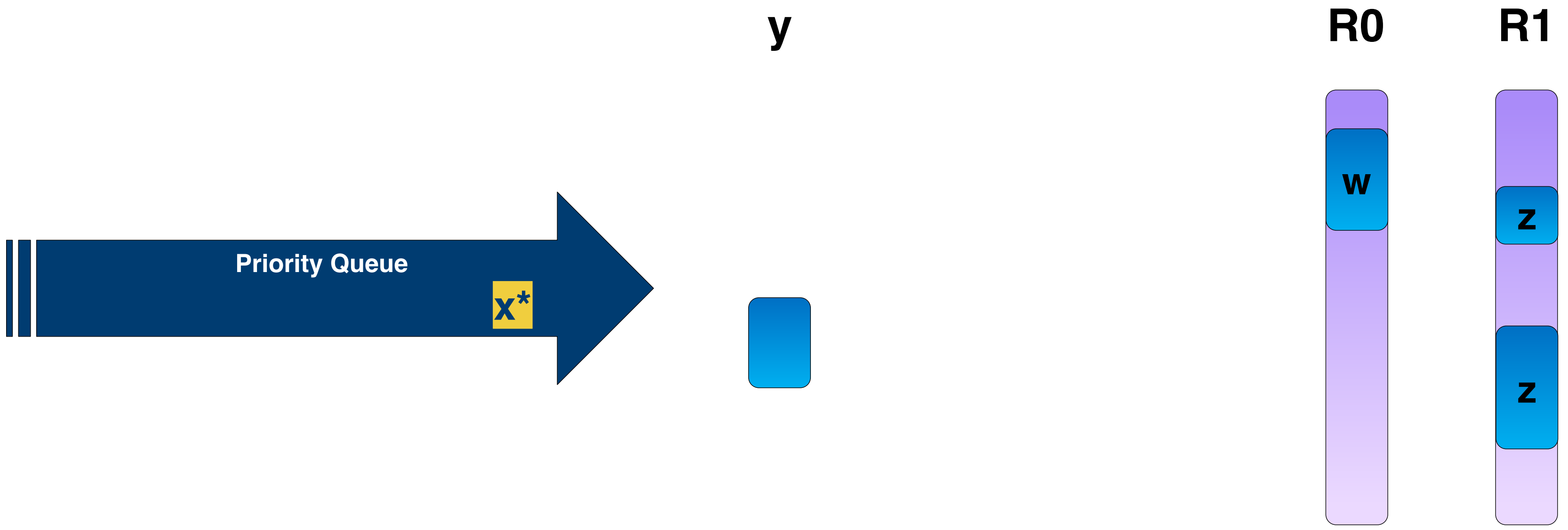
Register Assignment

- Dequeue interval with highest priority



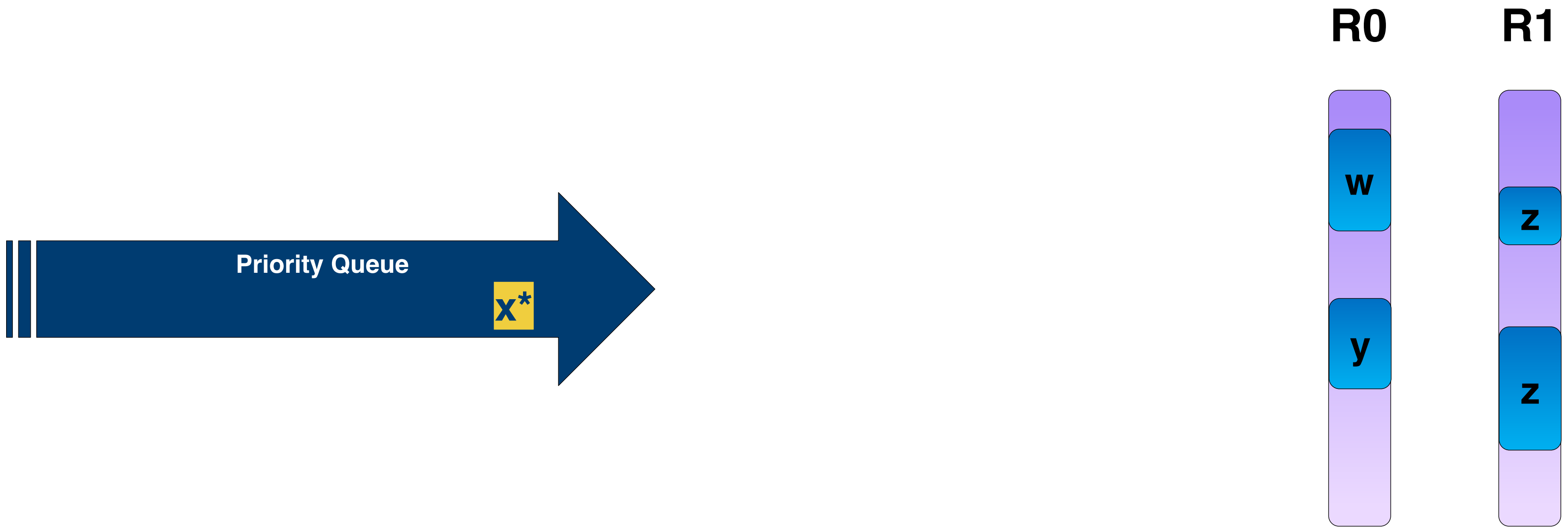


Register Assignment



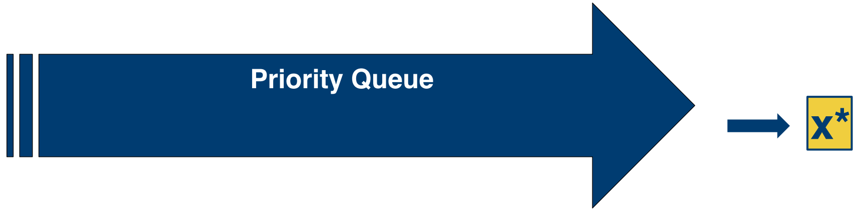
Register Assignment

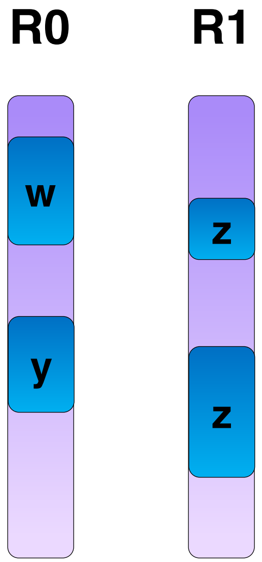
- Assign to available register if possible



Register Assignment

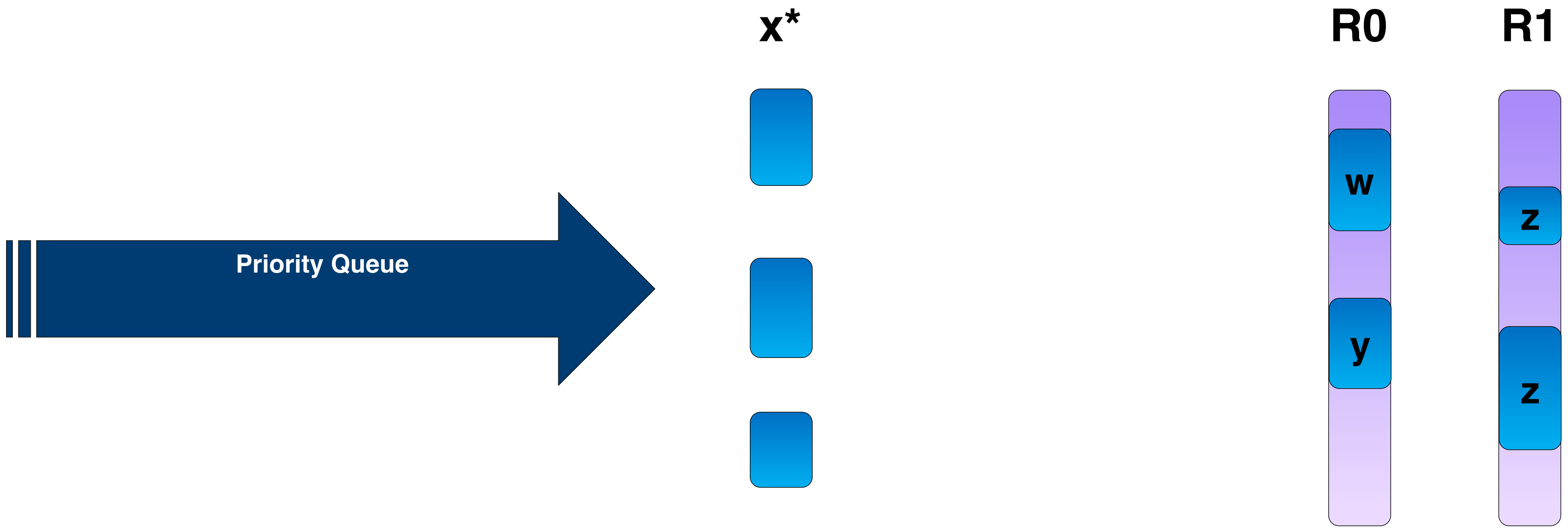
- Dequeue interval with highest priority





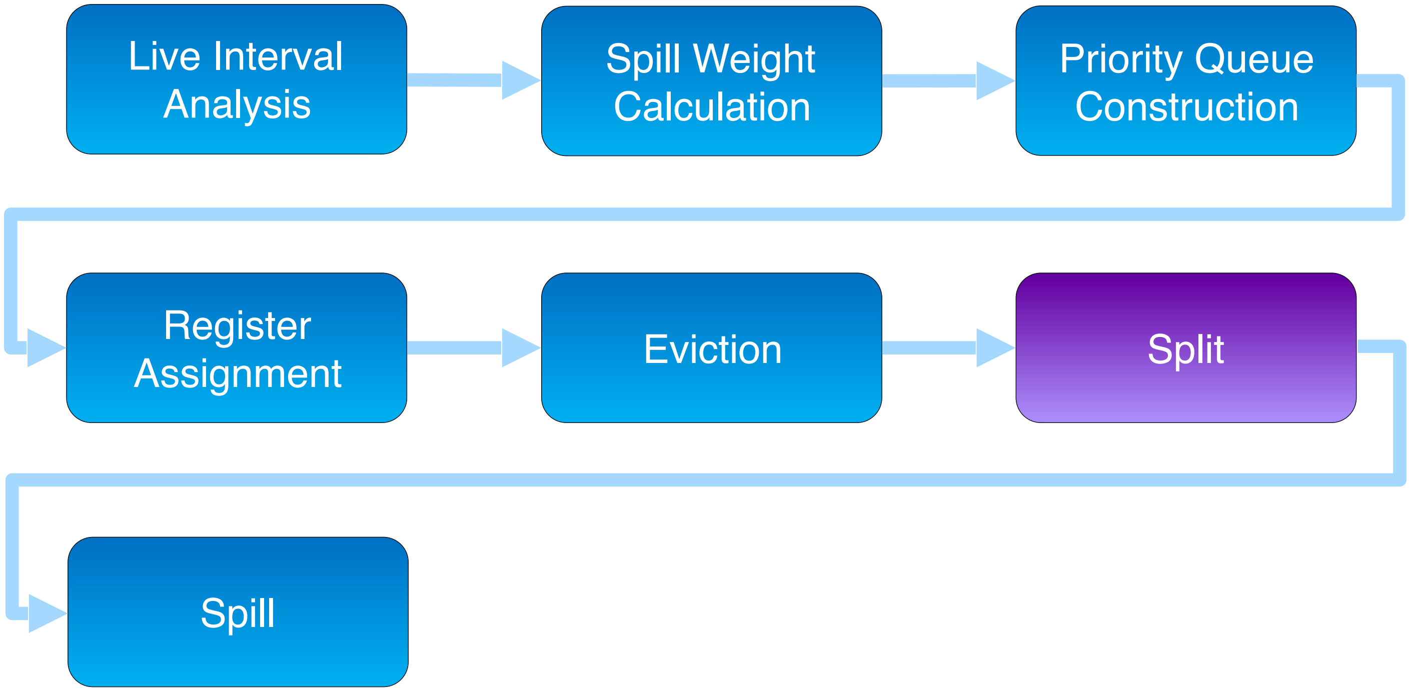
Register Assignment

- x is marked to be split



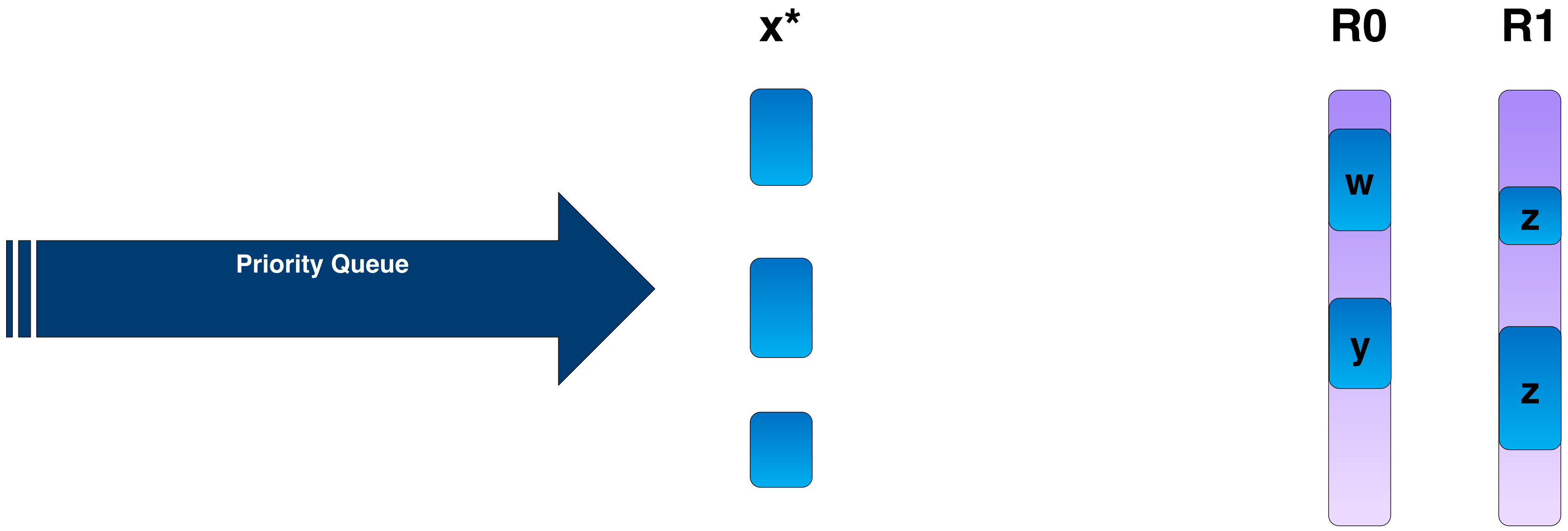
Greedy Register Allocator Overview

- General flow



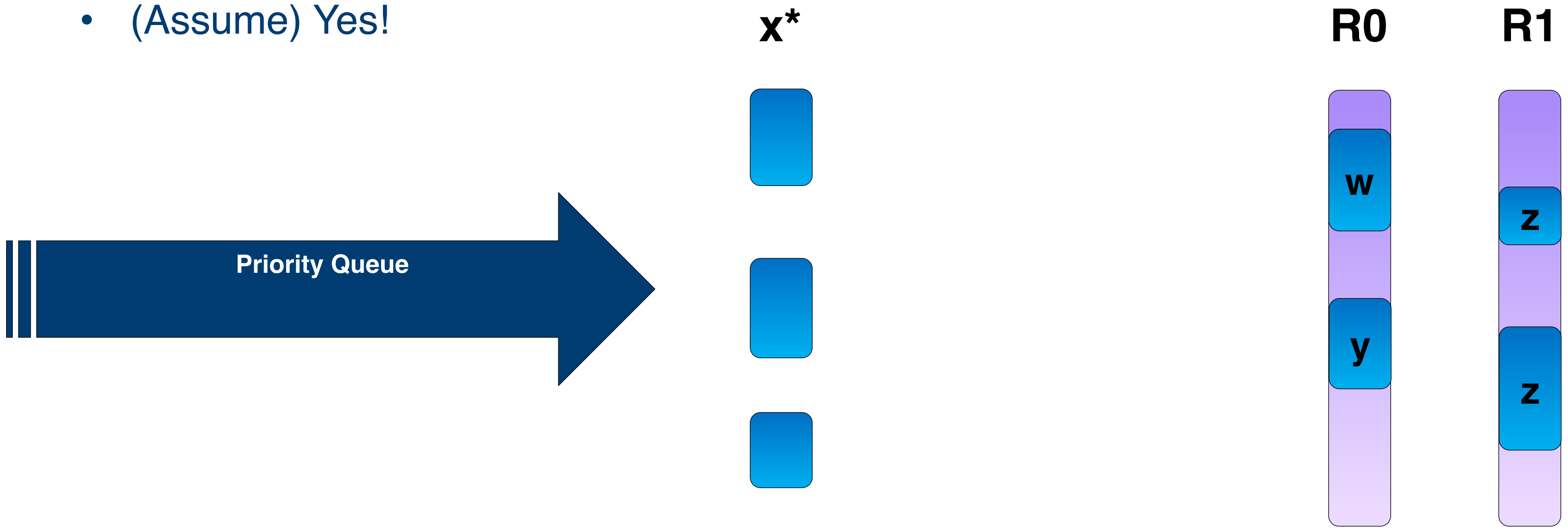
Split

- Is split beneficial?



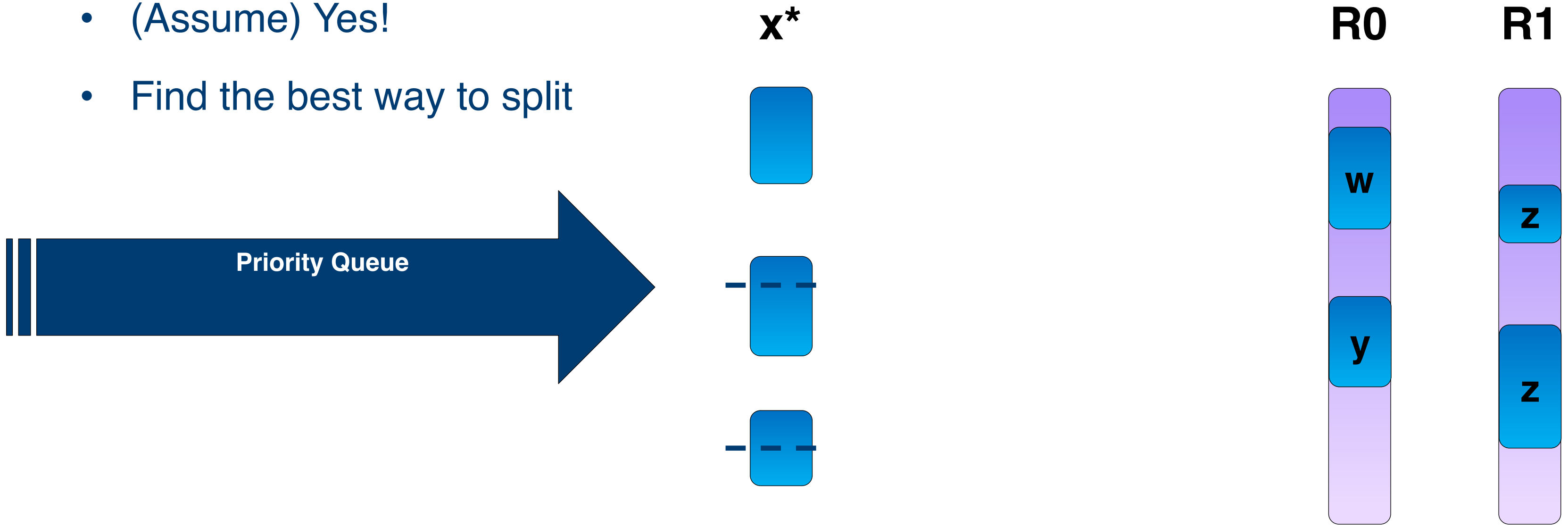
Split

- Is split beneficial?
  - (Assume) Yes!



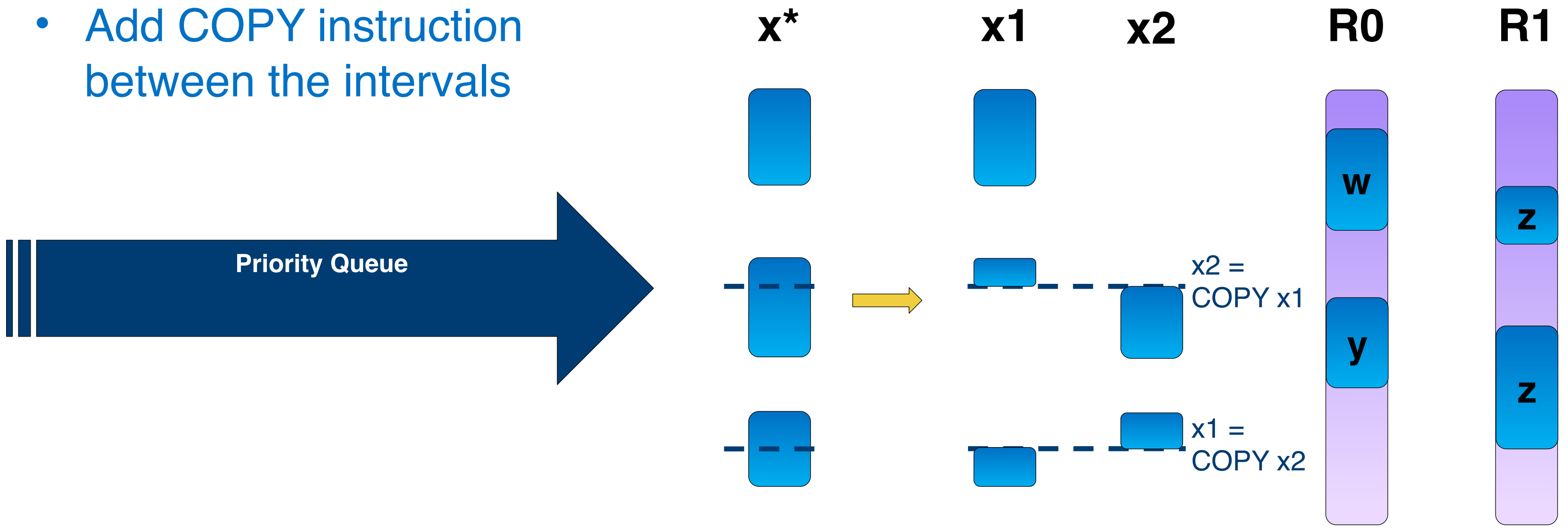
Split

- Is split beneficial?
  - (Assume) Yes!
  - Find the best way to split

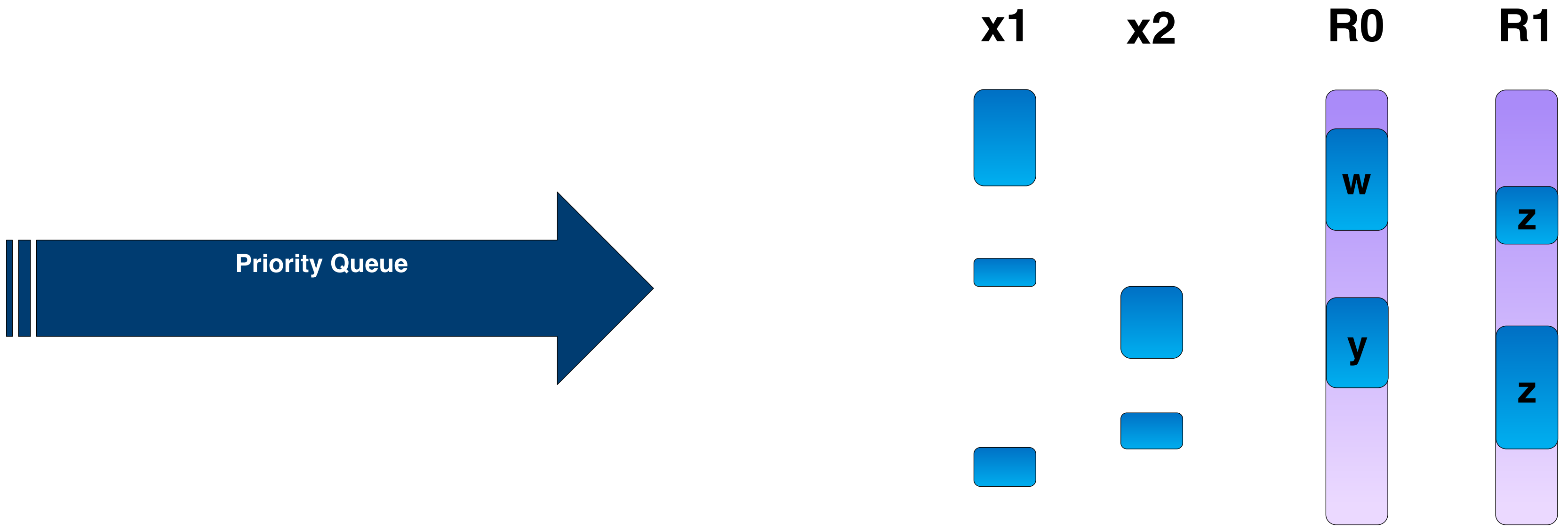


Split

- Do the split
- Add COPY instruction between the intervals

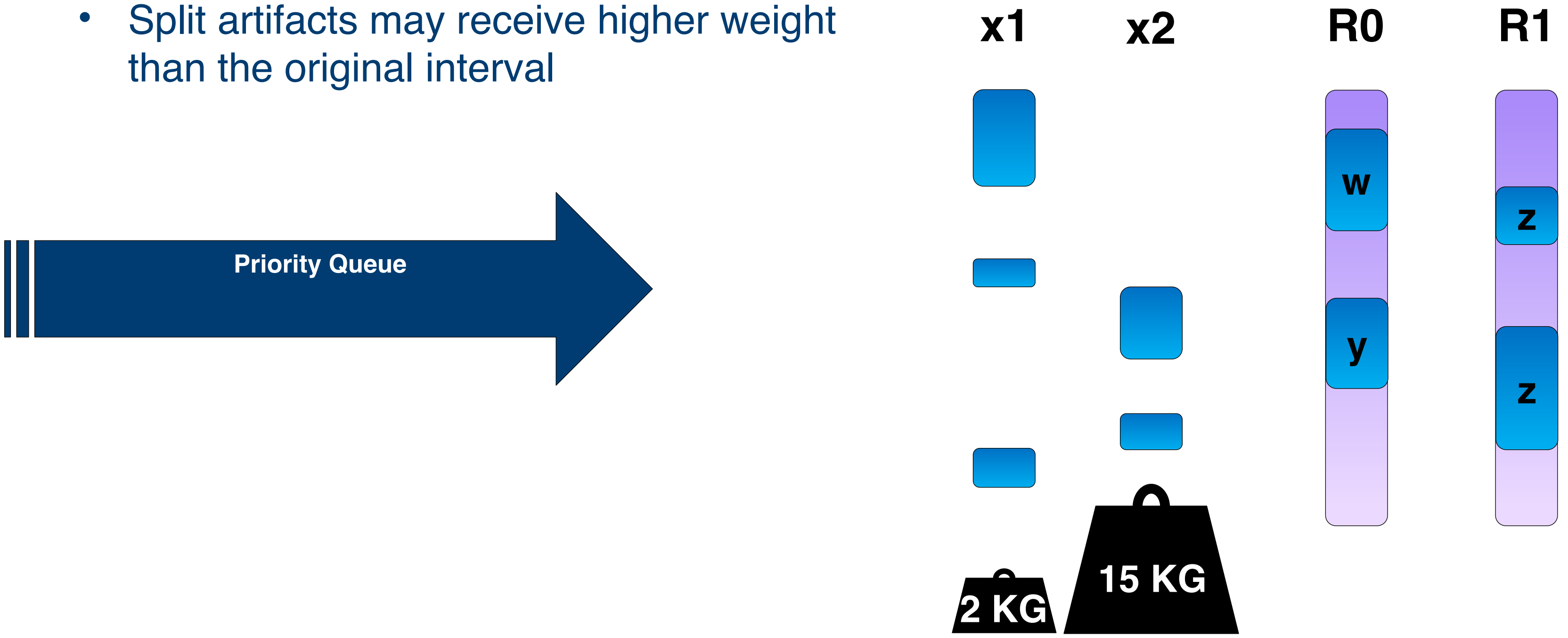


Split

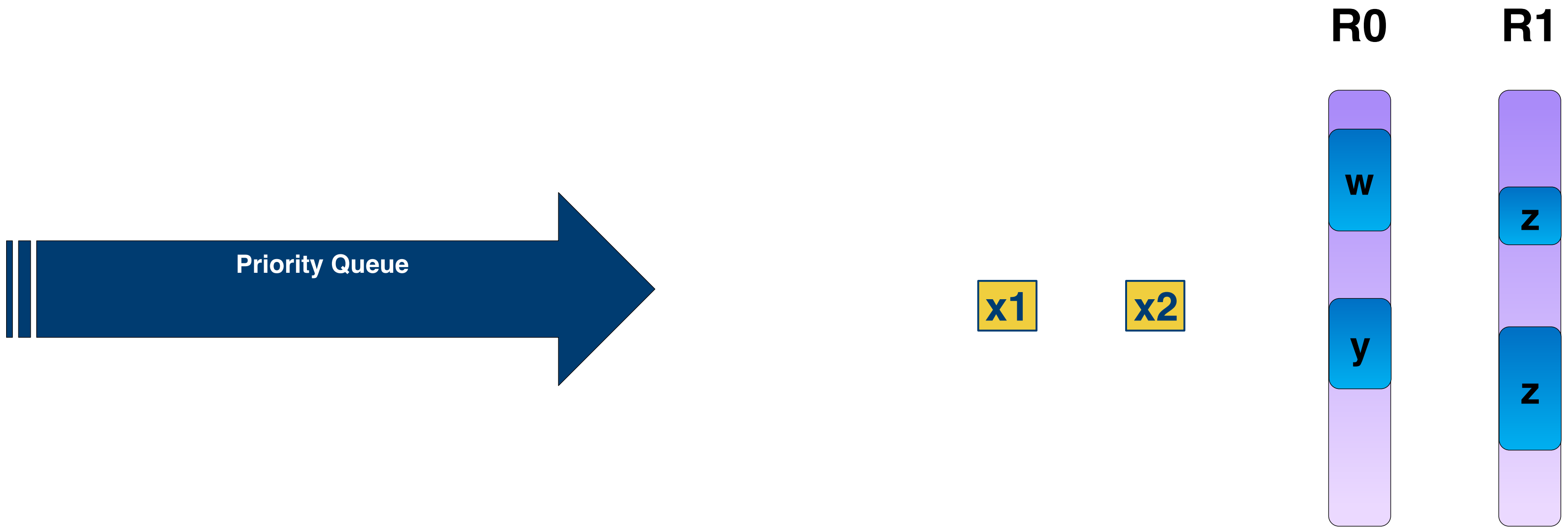


Split

- Calculate spill weights
  - Split artifacts may receive higher weight than the original interval

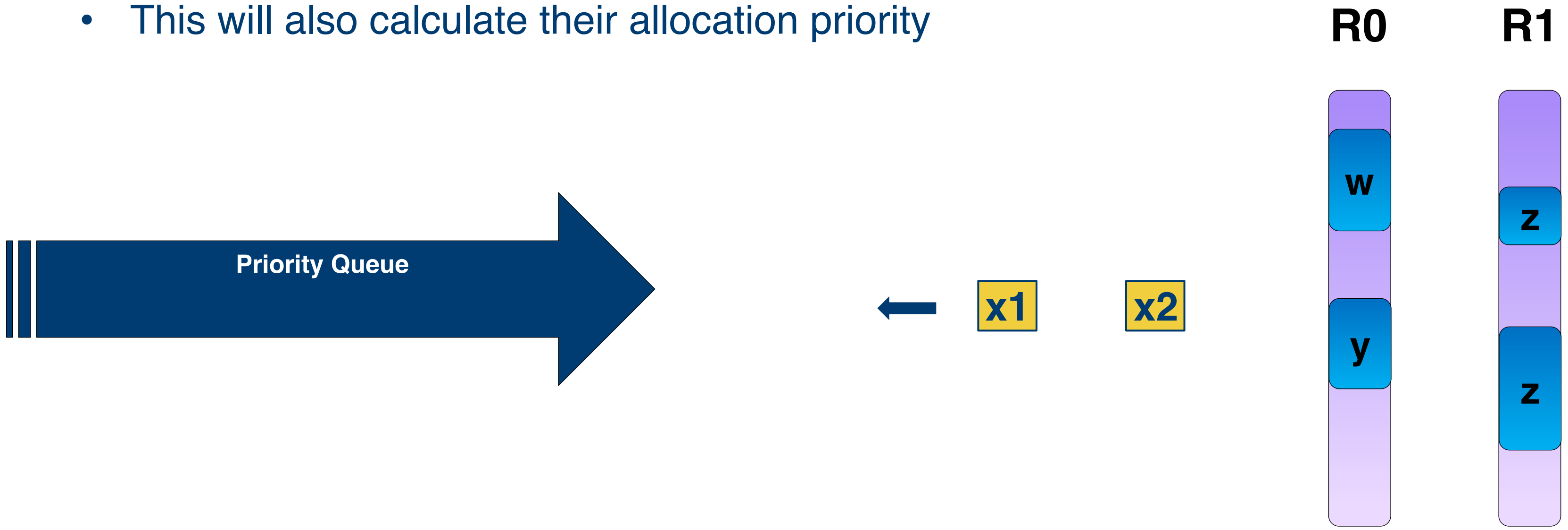


Split

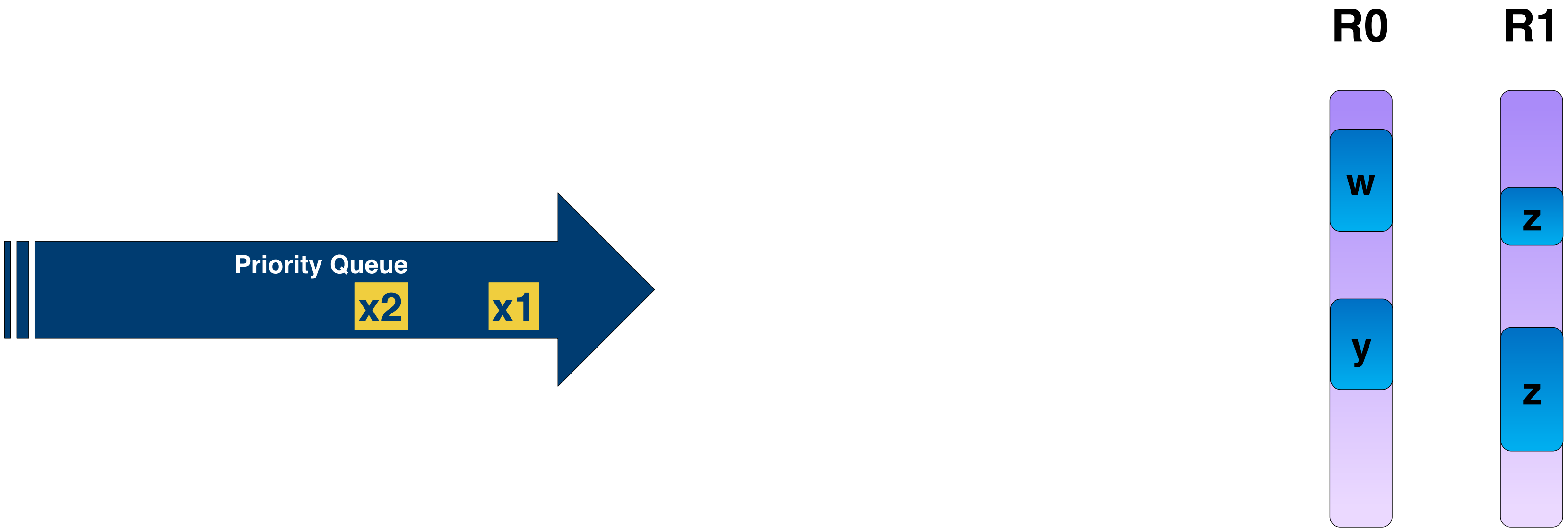


Split

- Enqueue x1, x2 into the queue
    - This will also calculate their allocation priority

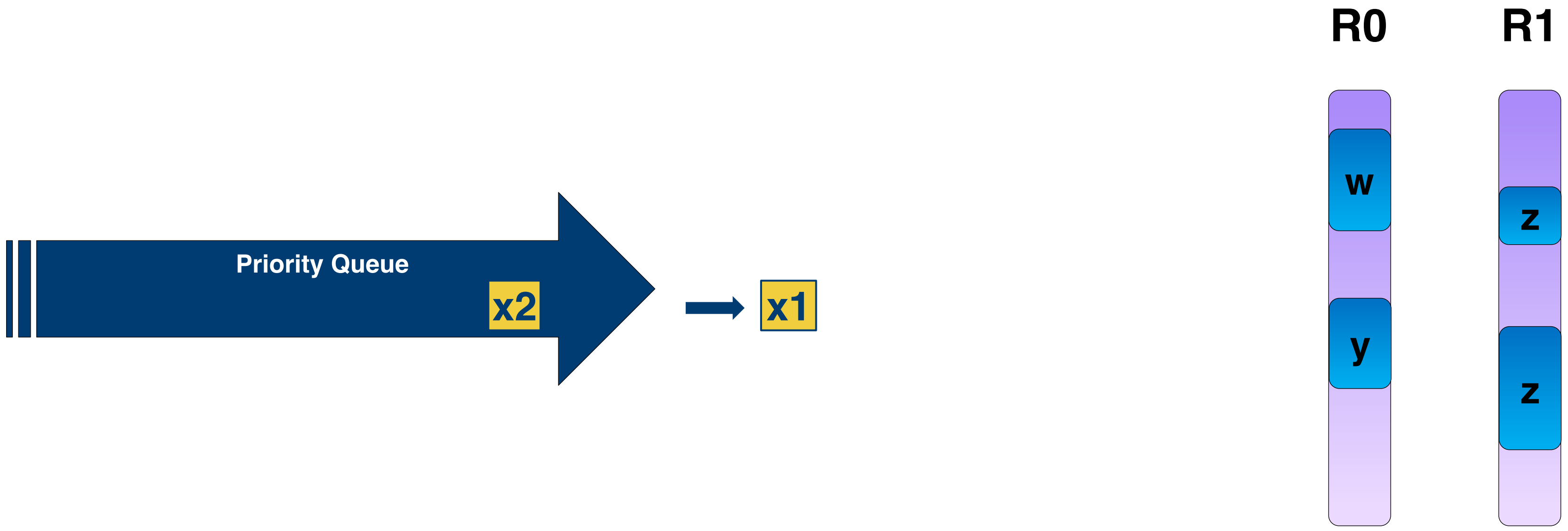


Register Assignment

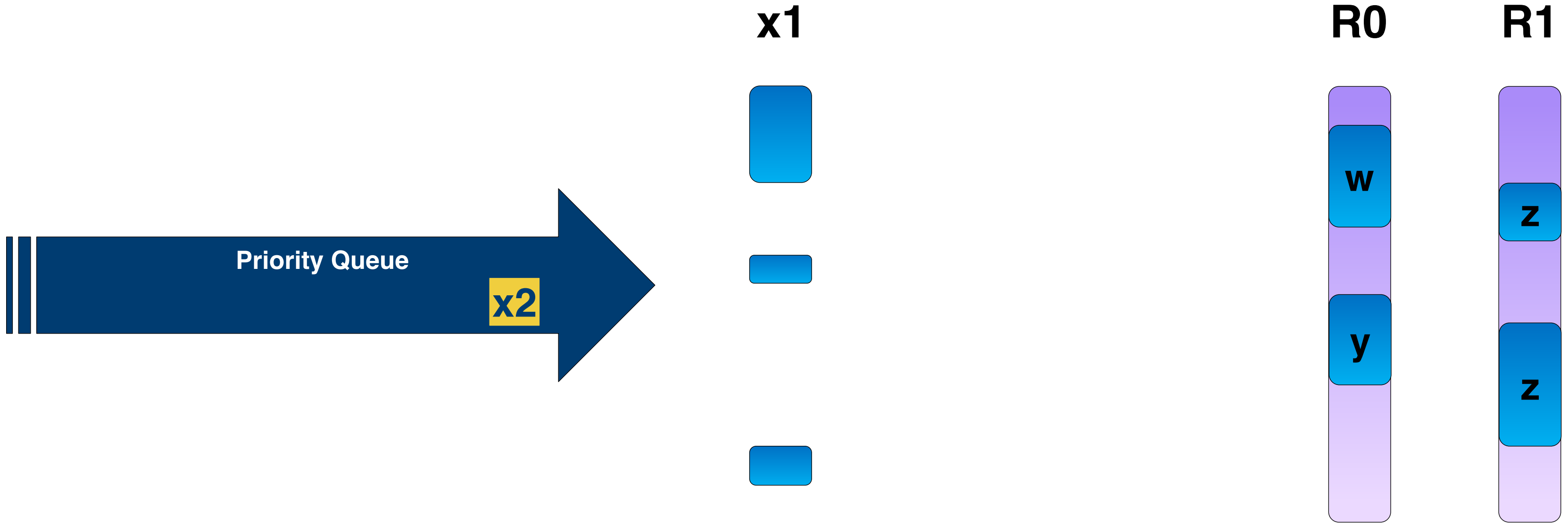


Register Assignment

- Dequeue interval with highest priority

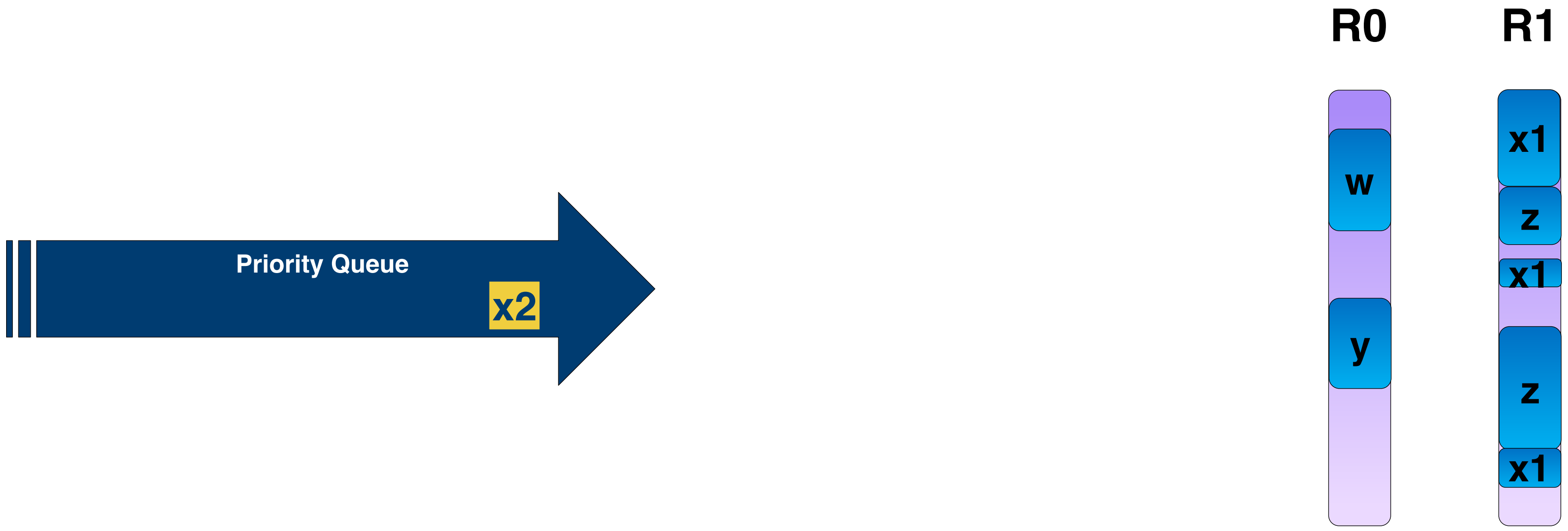


Register Assignment



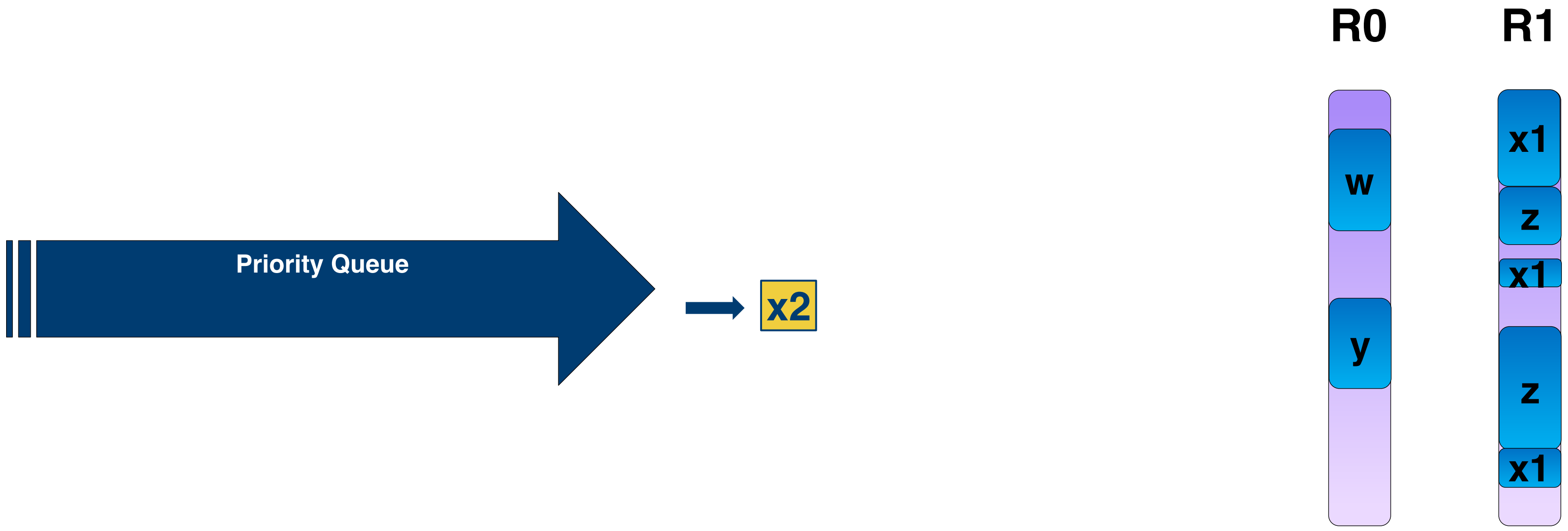
Register Assignment

- Assign to available register if possible

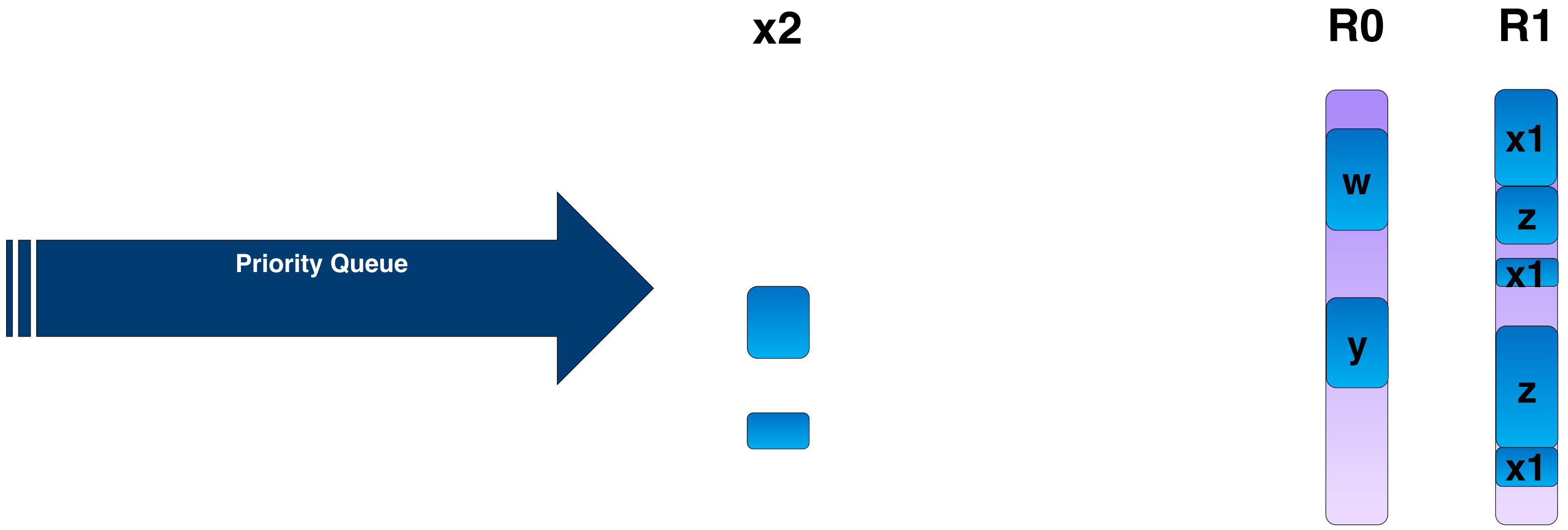


Register Assignment

- Dequeue interval with highest priority

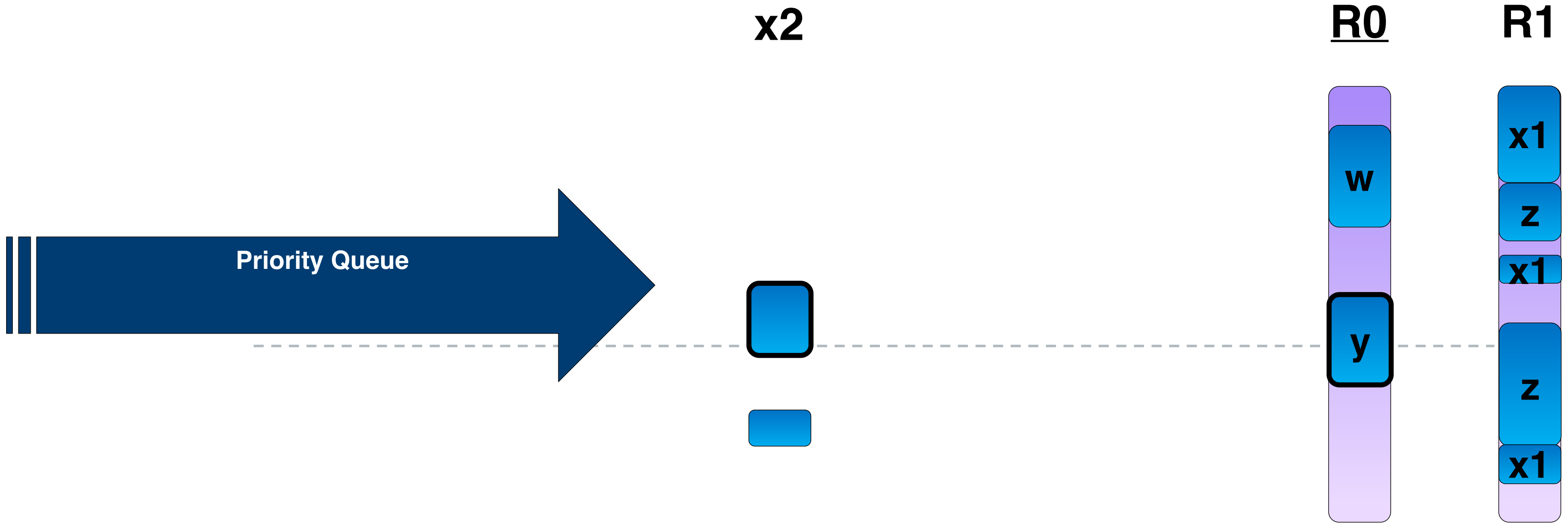


Register Assignment



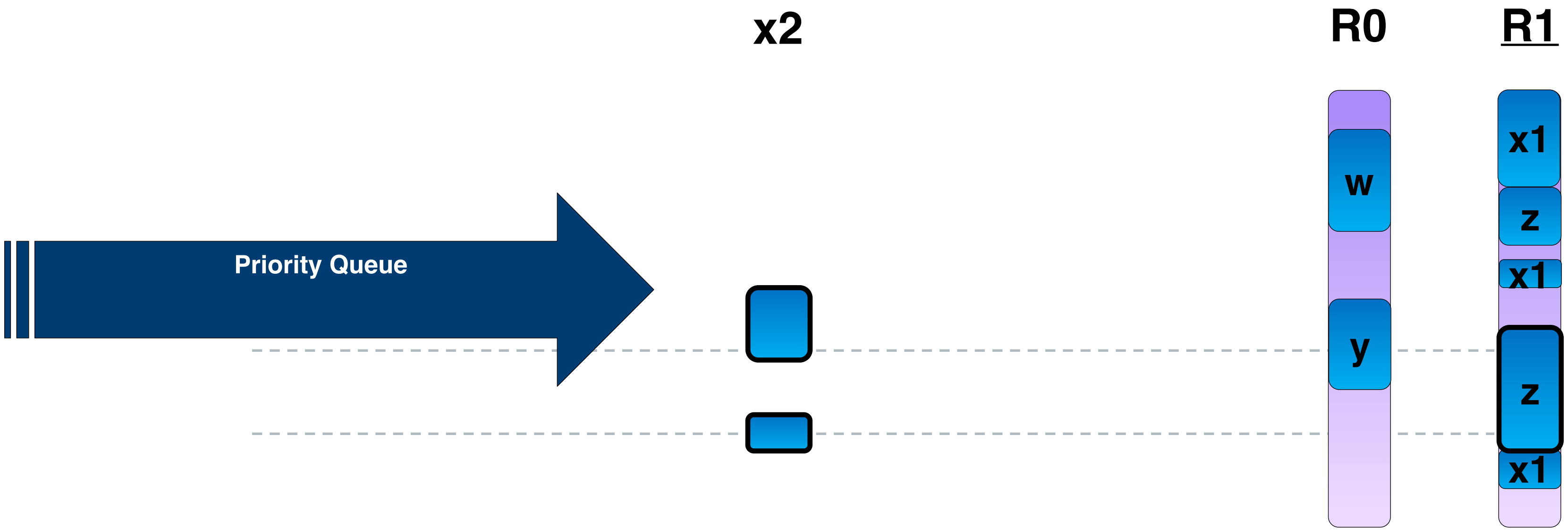
Register Assignment

- Interference with y in R0

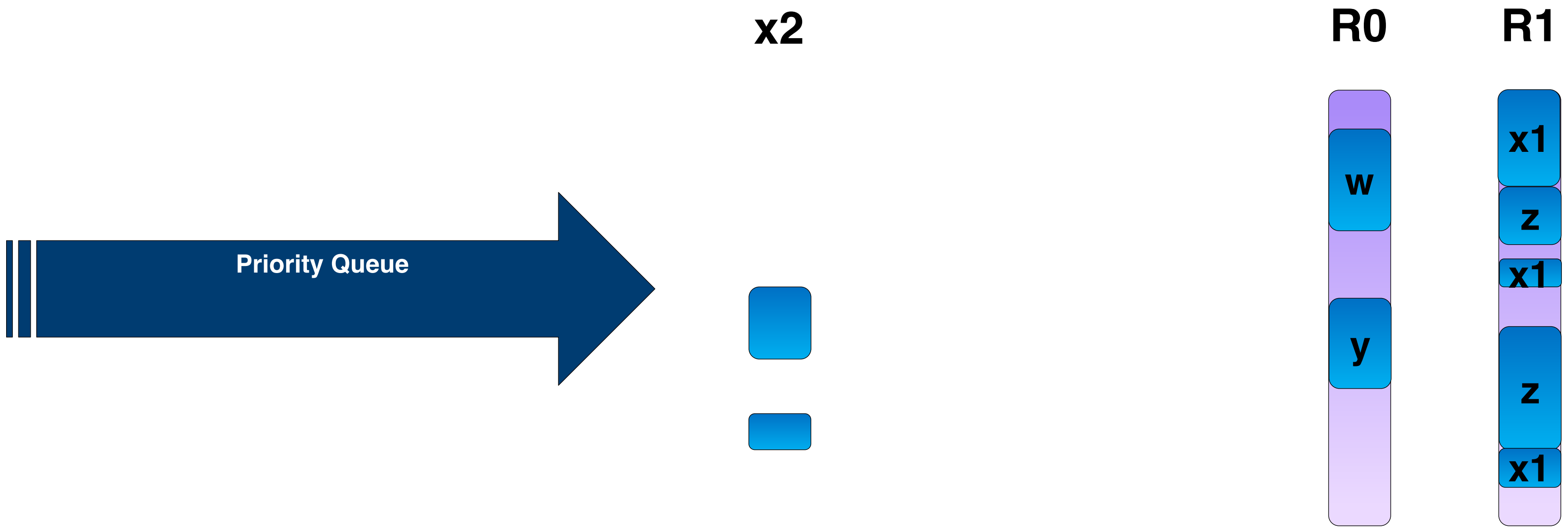


Register Assignment

- Interference with z in R1

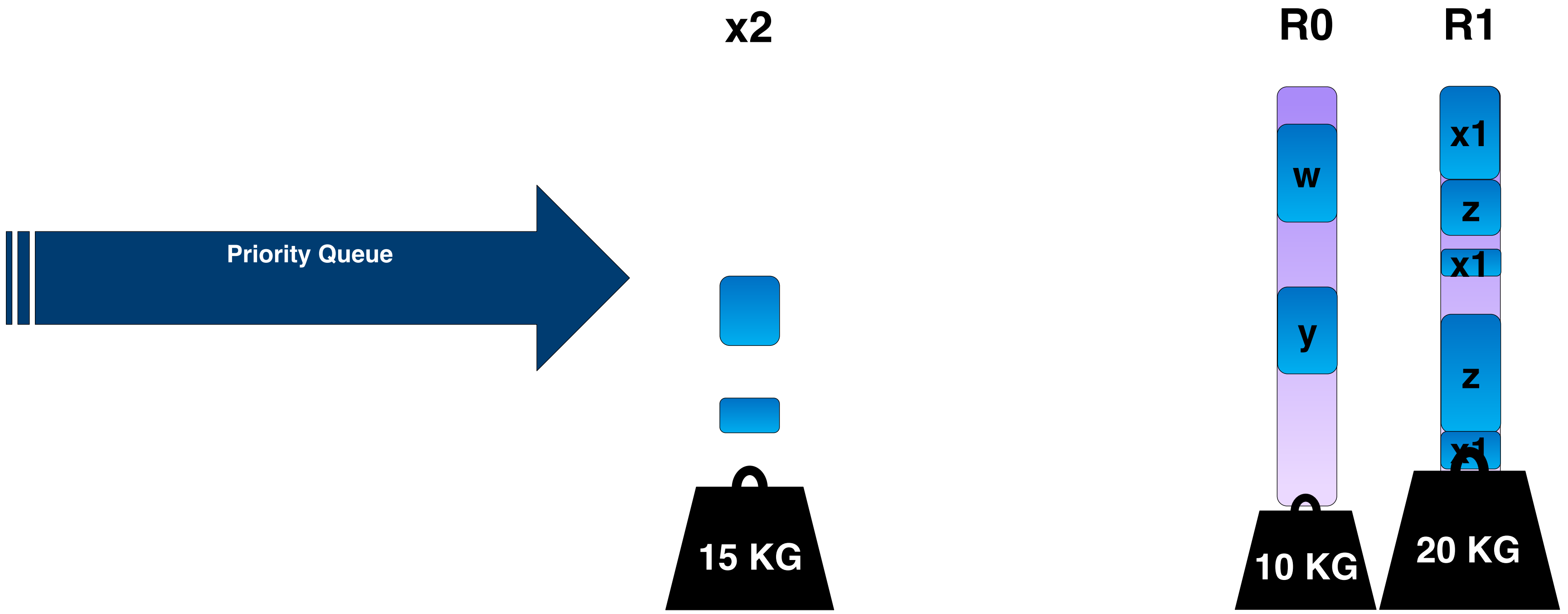


Register Assignment



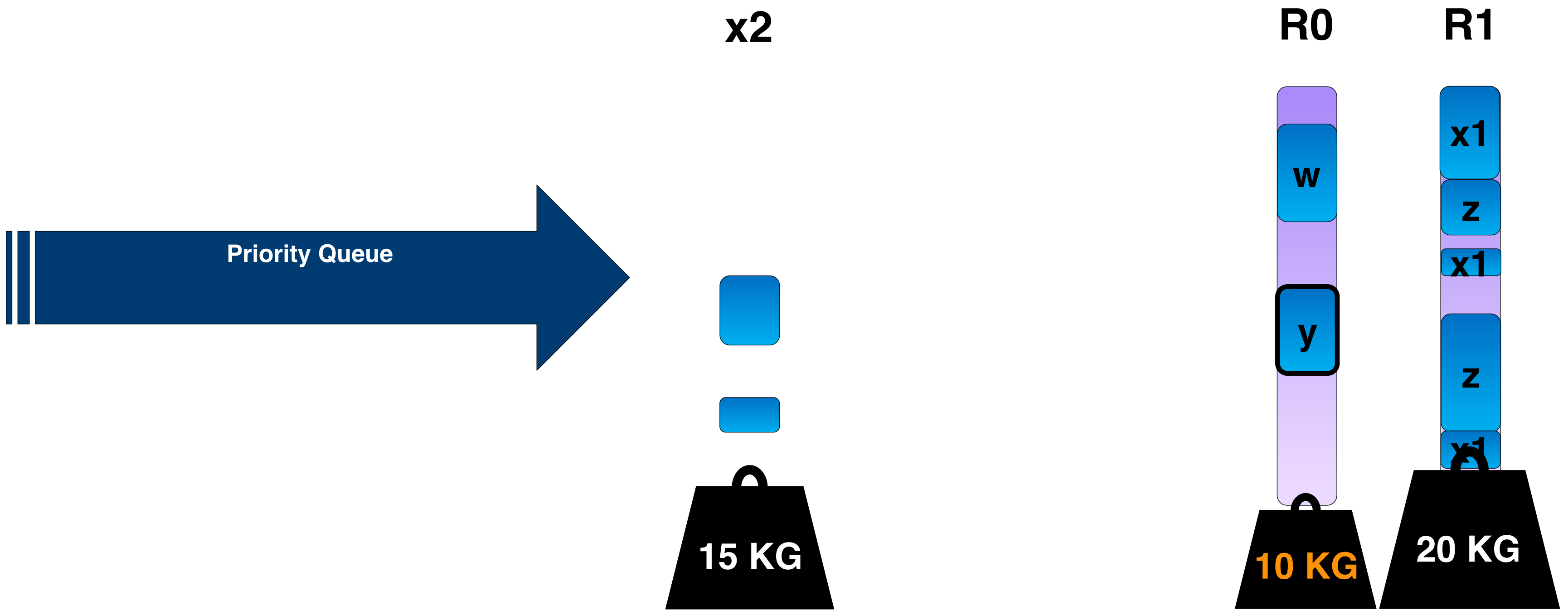
Eviction

- Compare spill weights of interfering intervals



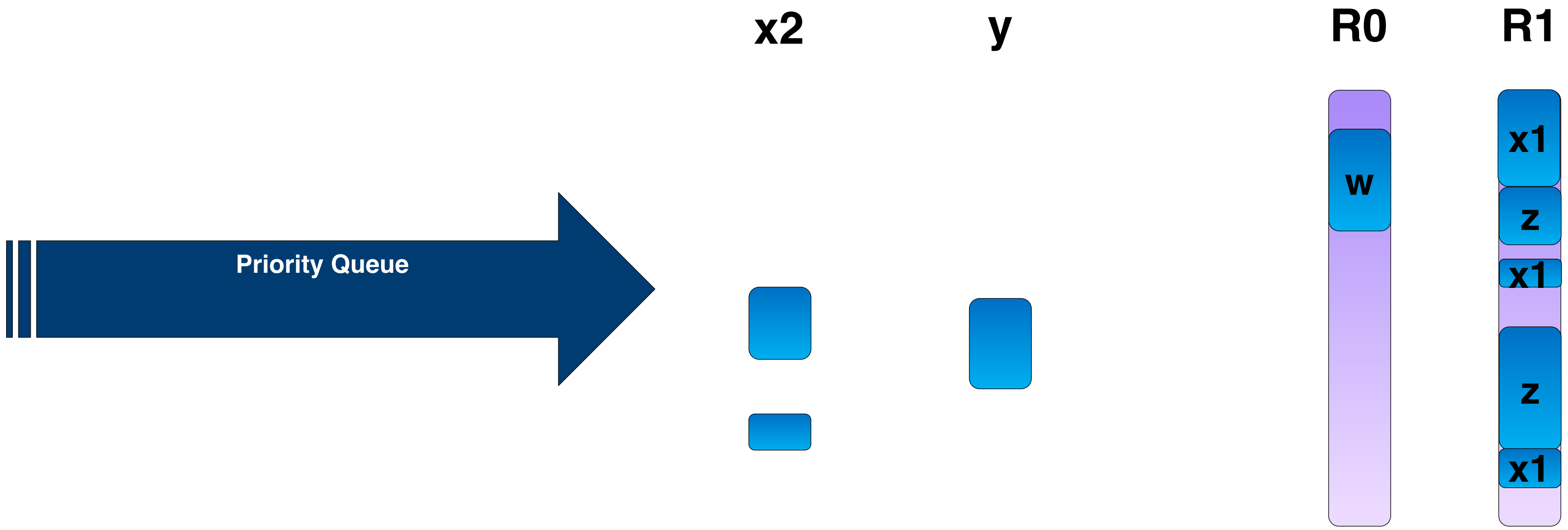
Eviction

- y cheaper than x2



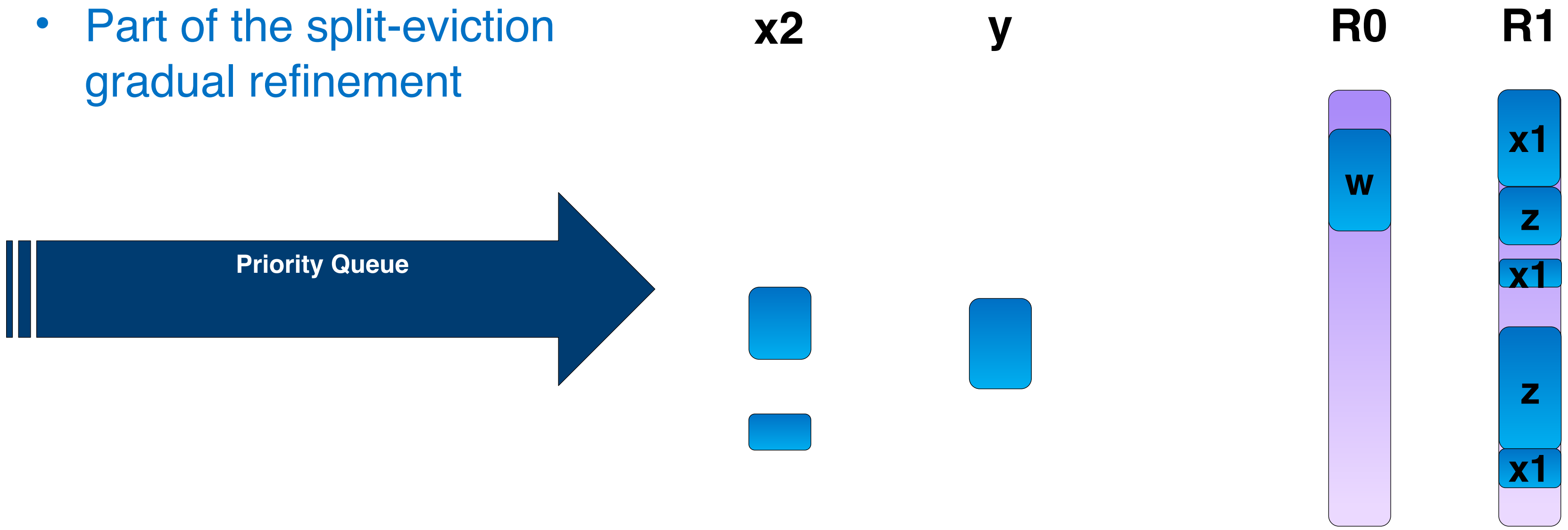
Eviction

- Evict y



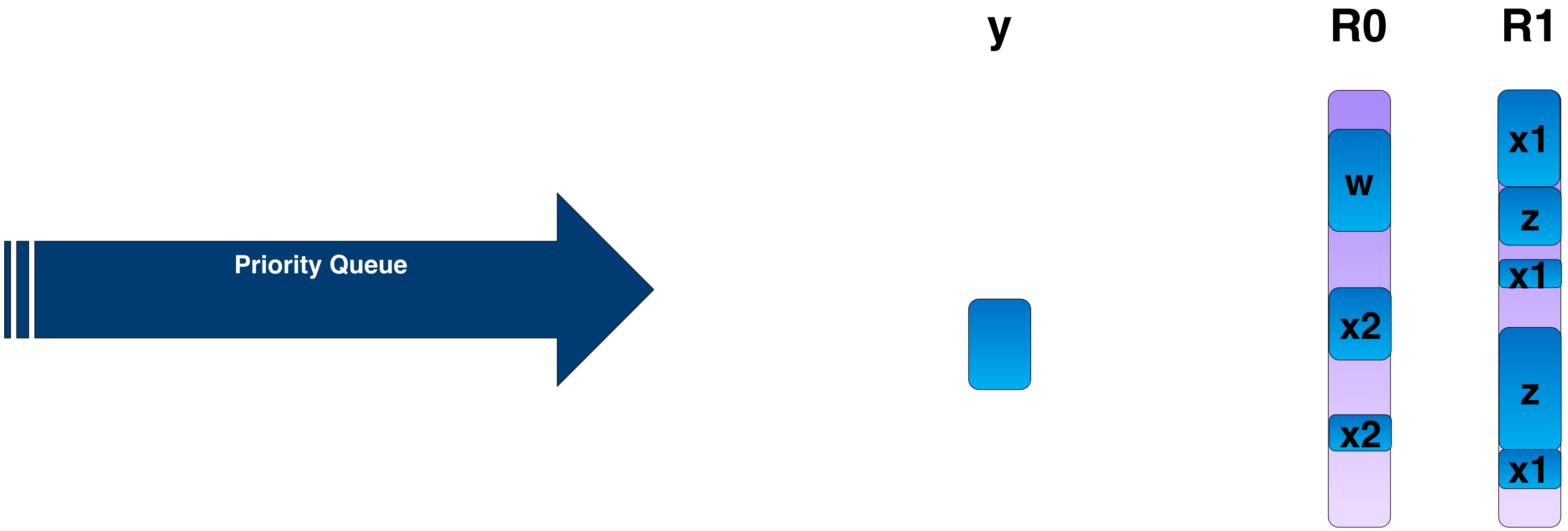
Eviction

- A split artifact can evict original interval
- Part of the split-eviction gradual refinement

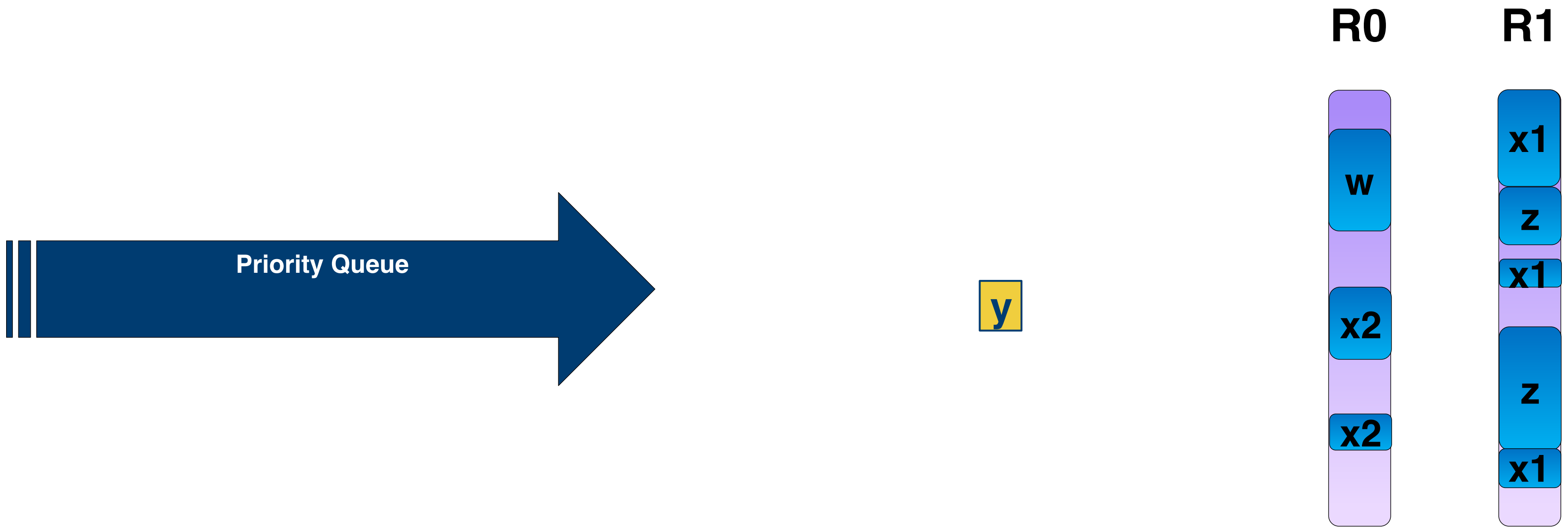


Eviction

- Assign x2

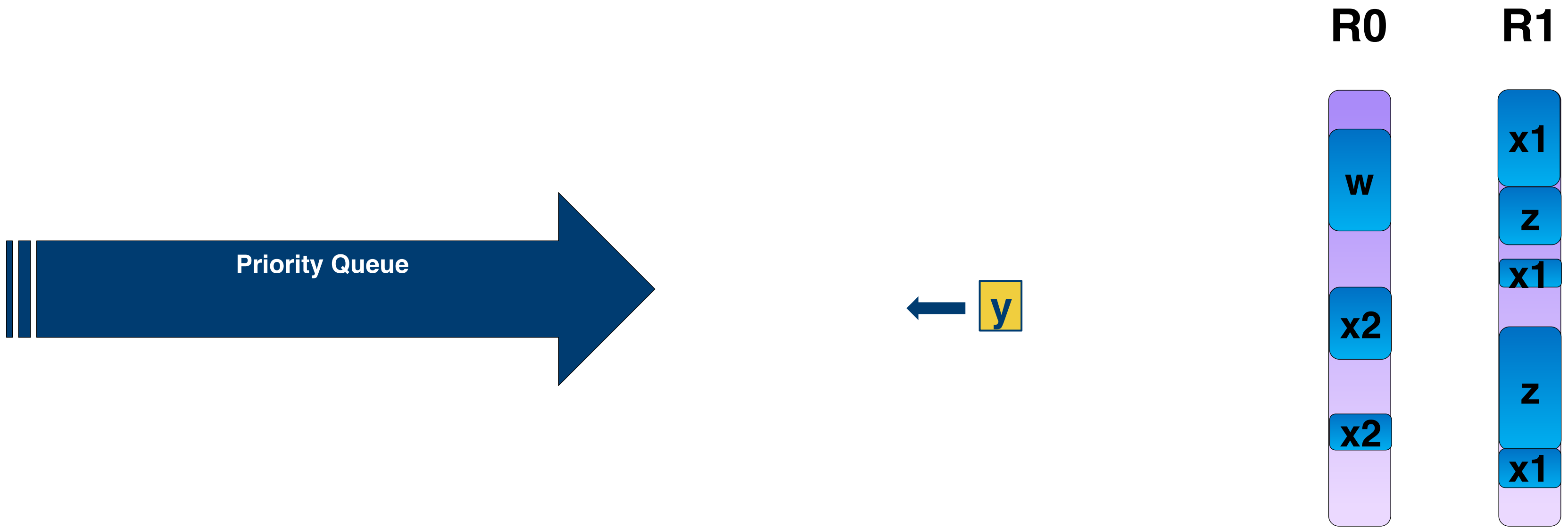


Eviction

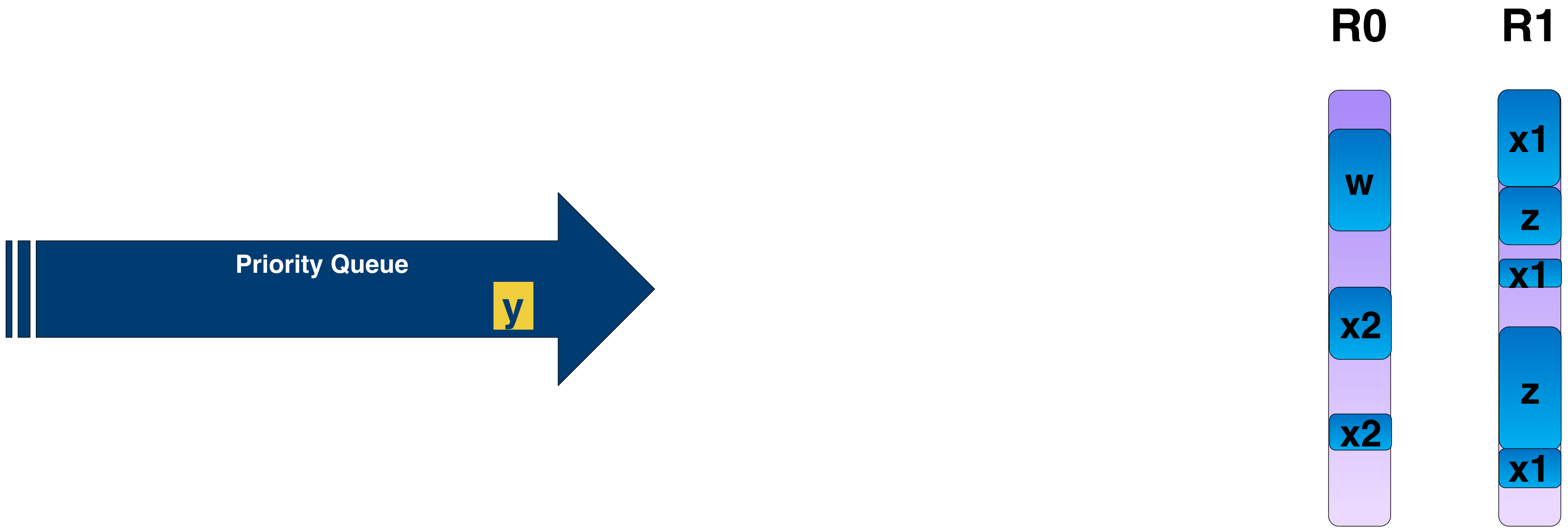


Eviction

- Enqueue y back to the queue

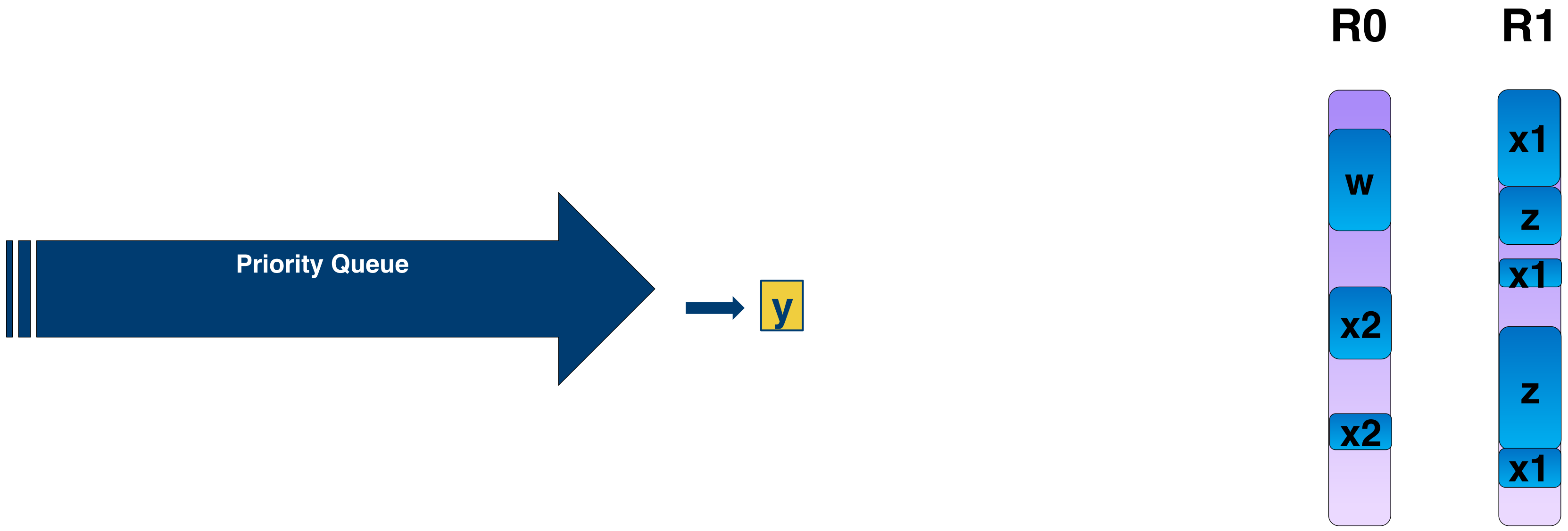


Register Assignment

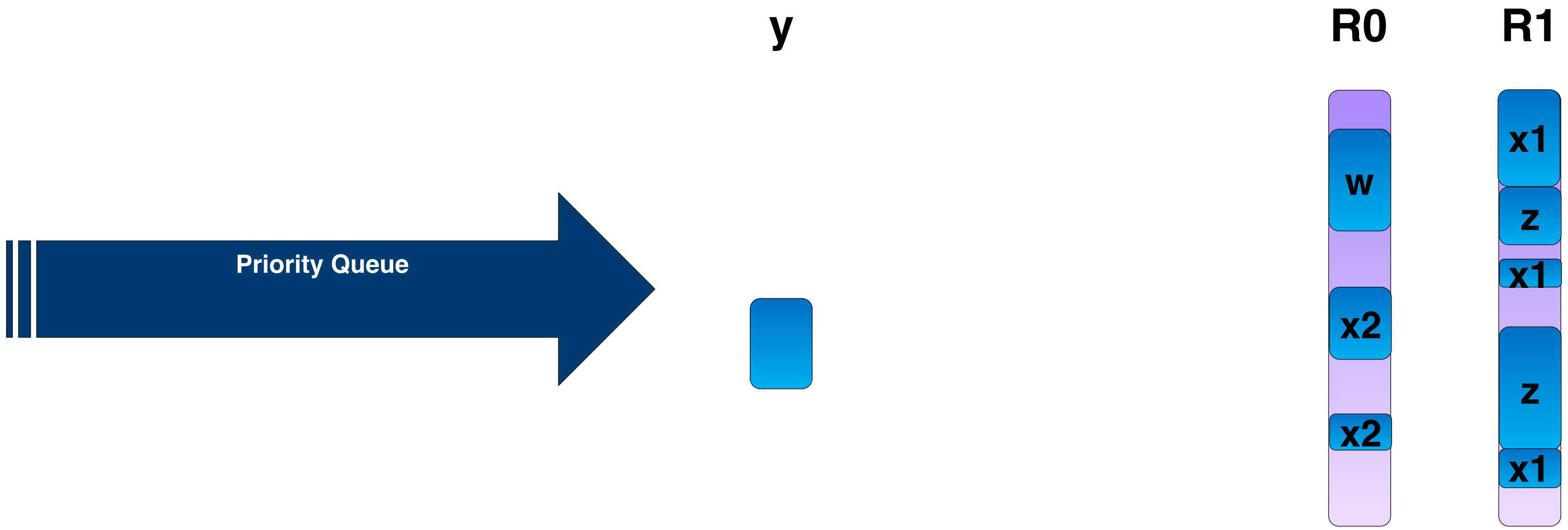


Register Assignment

- Dequeue interval with highest priority

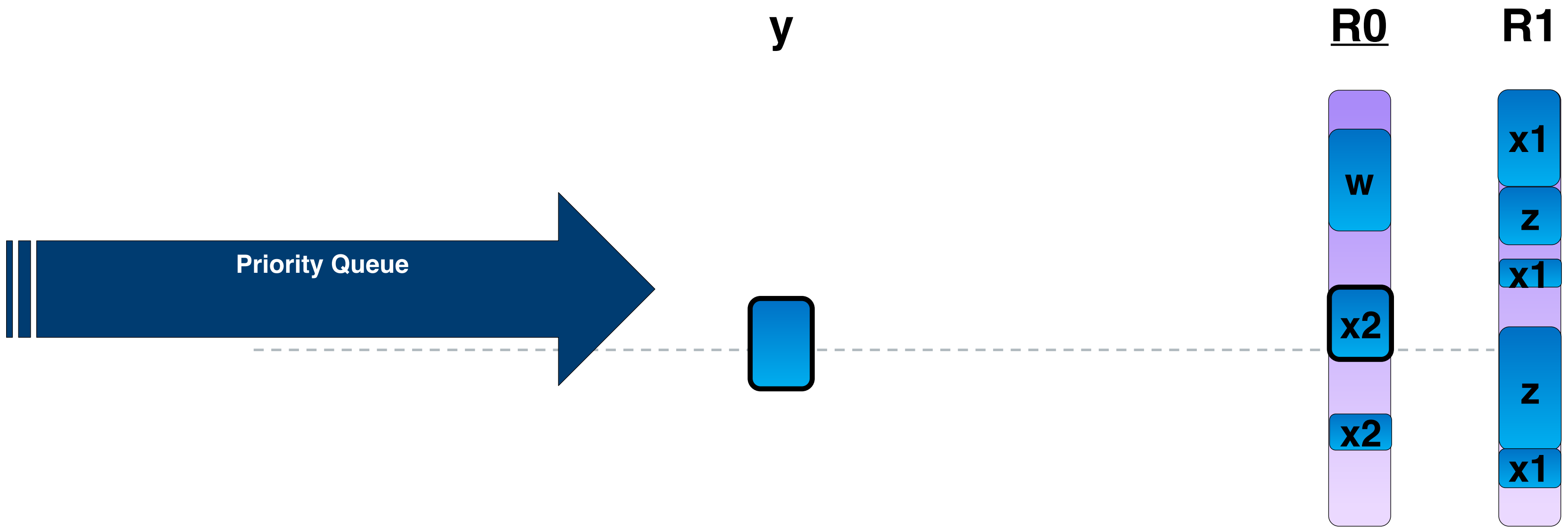


Register Assignment



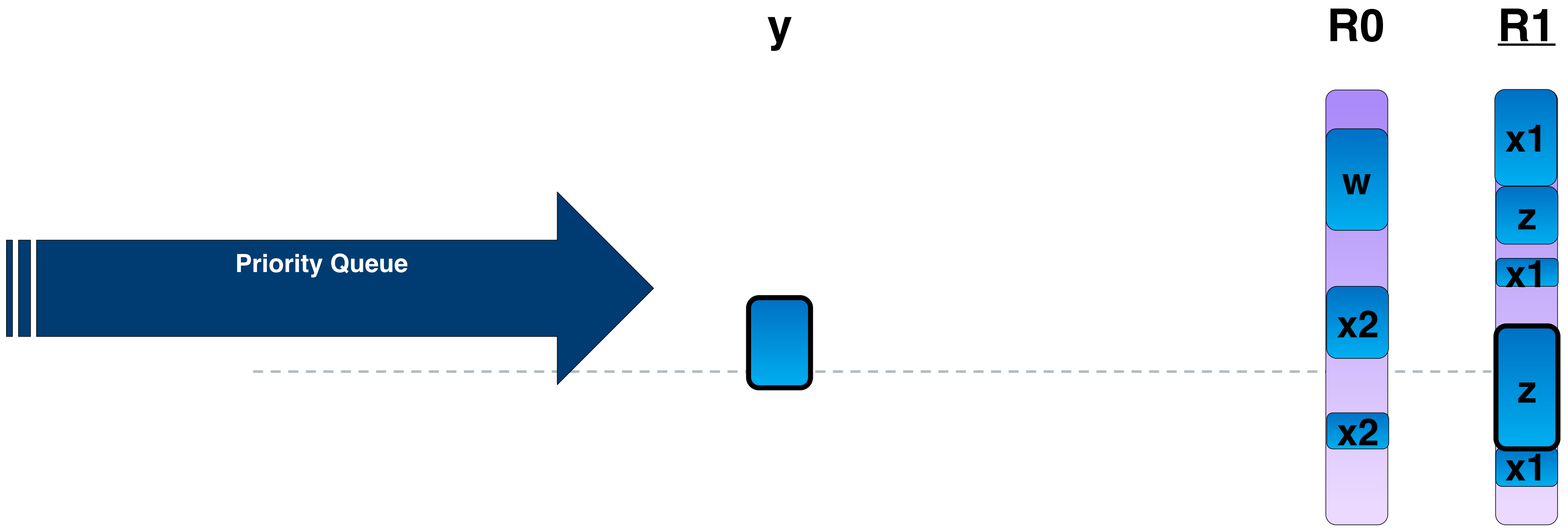
Register Assignment

- Interference with x2 in R0



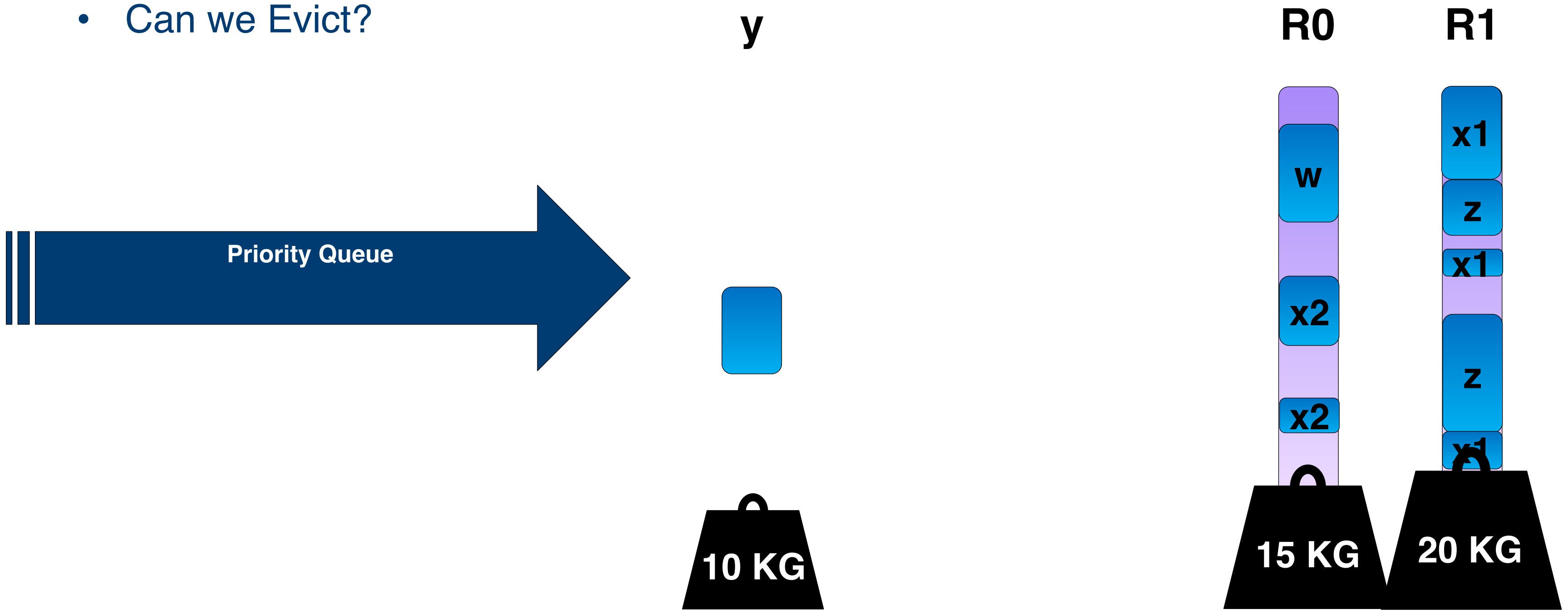
Register Assignment

- Interference with z in R1



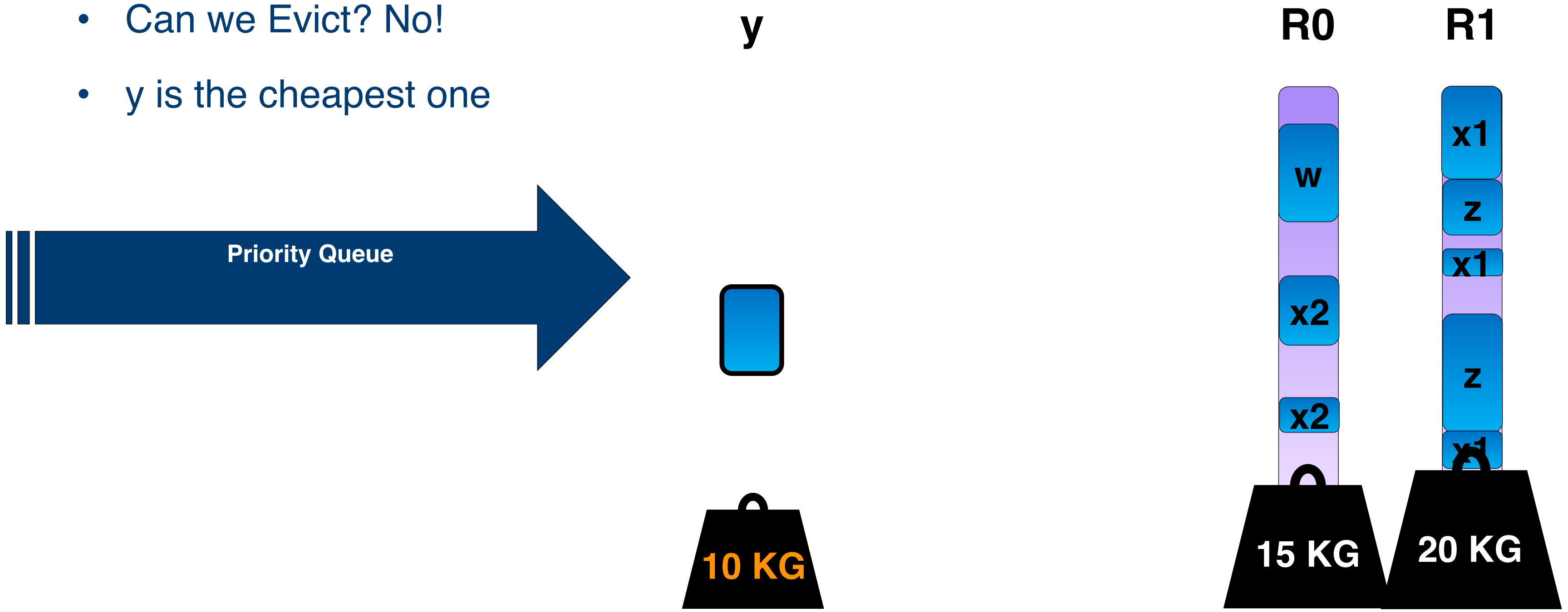
Eviction

- Compare spill weights of interfering intervals
  - Can we Evict?

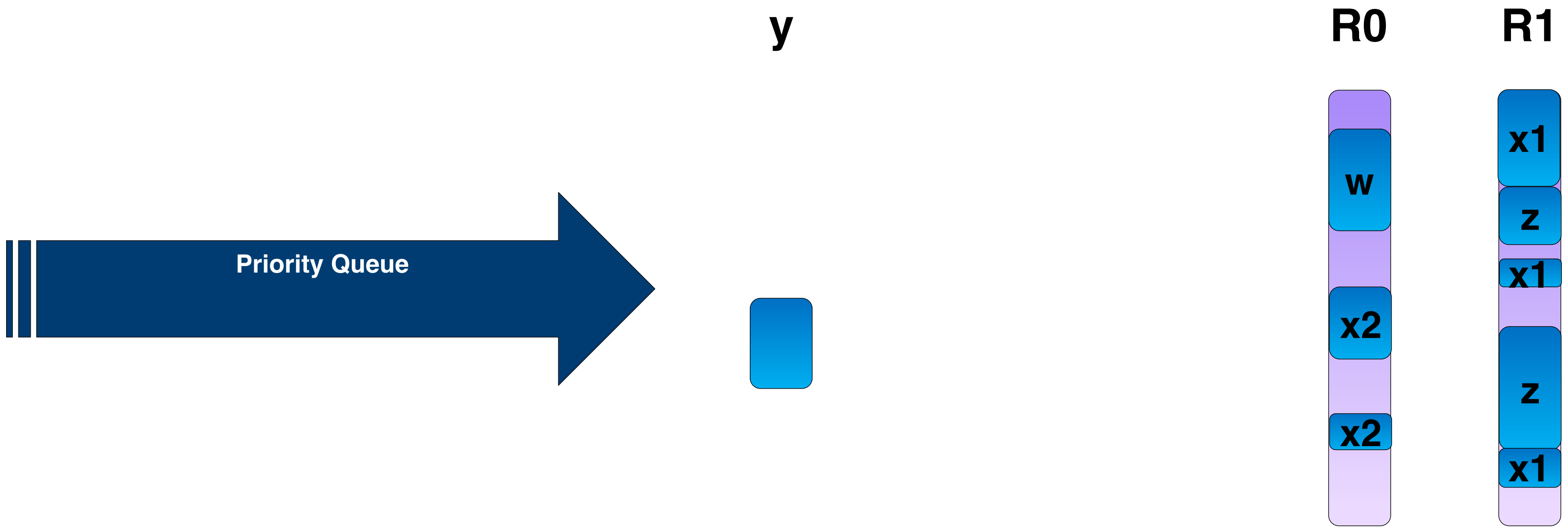


Eviction

- Compare spill weights of interfering intervals
  - Can we Evict? No!
  - y is the cheapest one

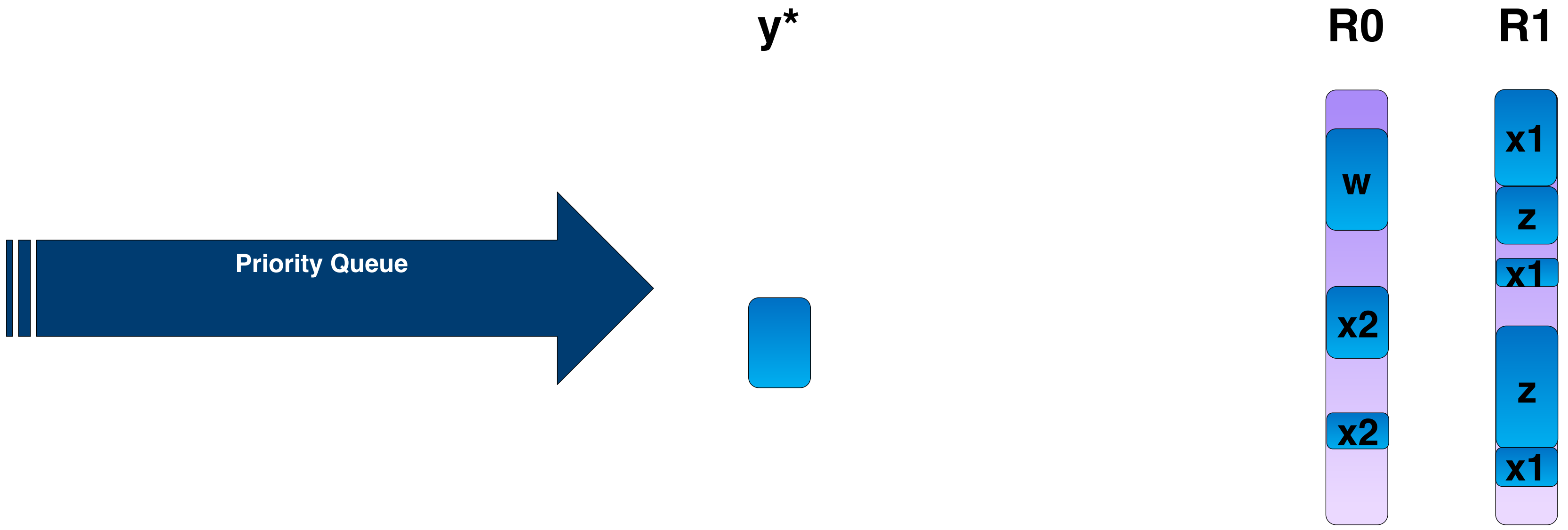


Eviction

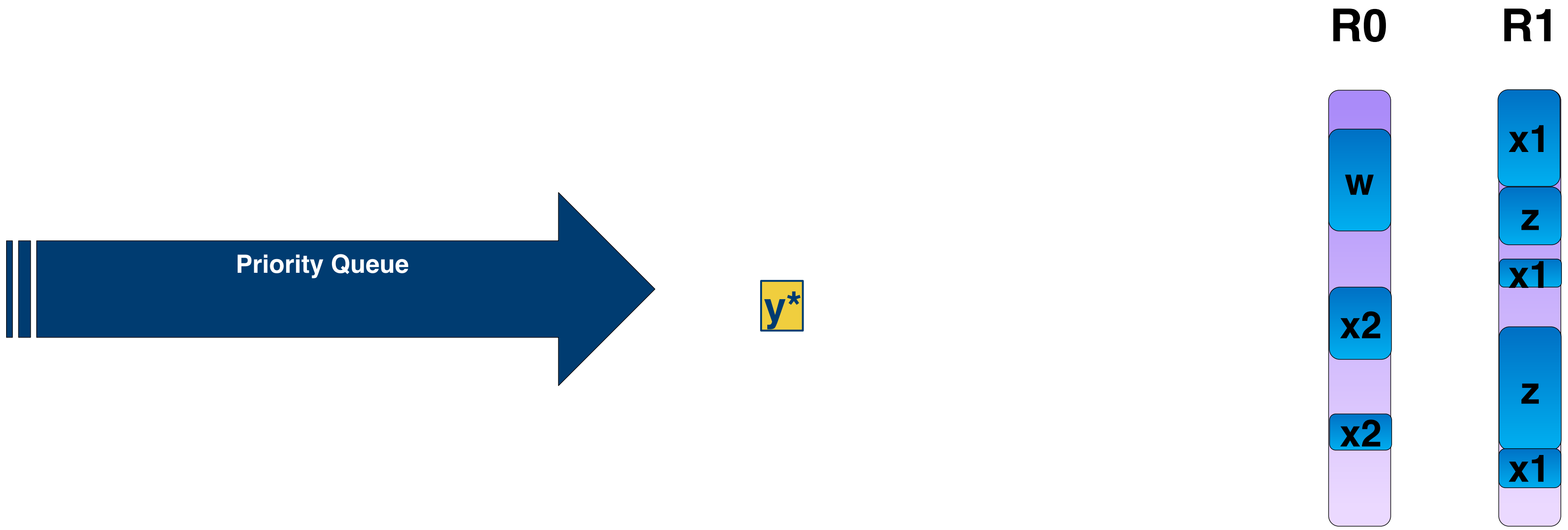


Register Assignment

- Mark y to be split

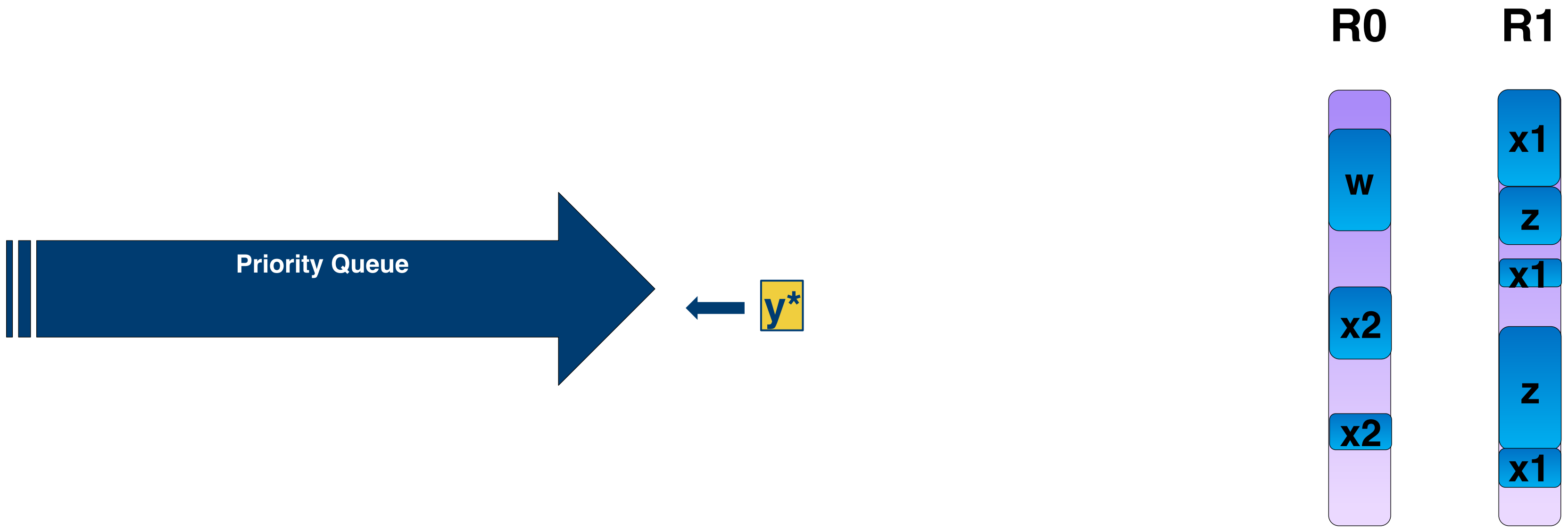


Register Assignment

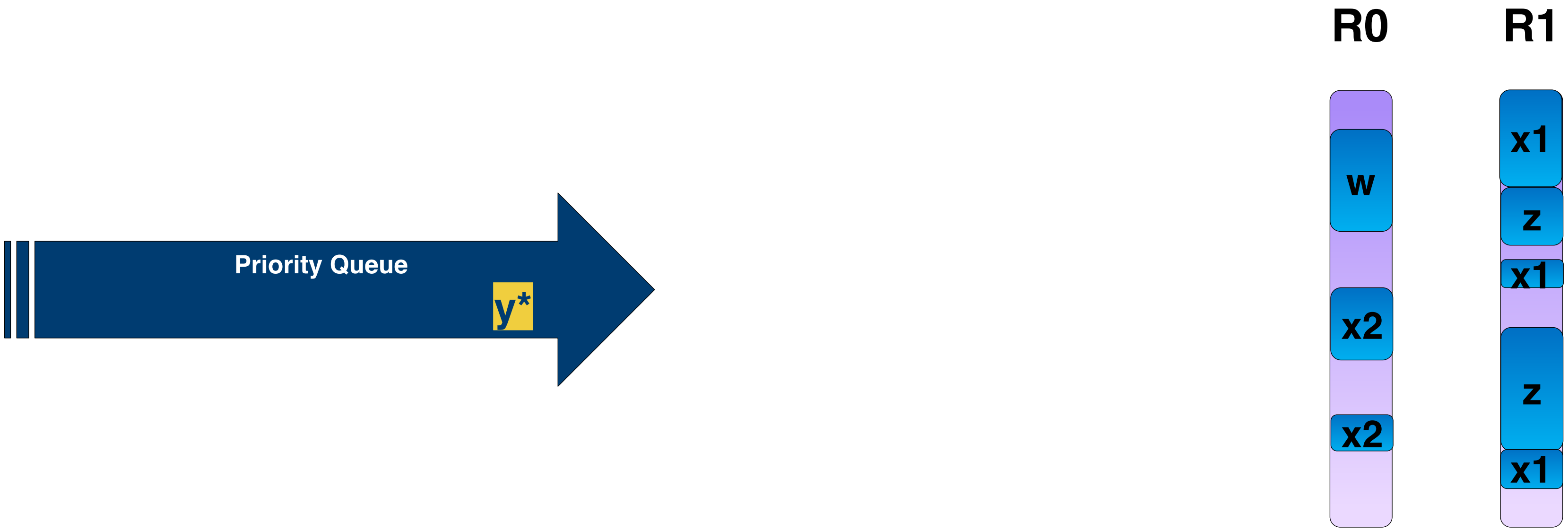


Register Assignment

- Enqueue y* back to the queue

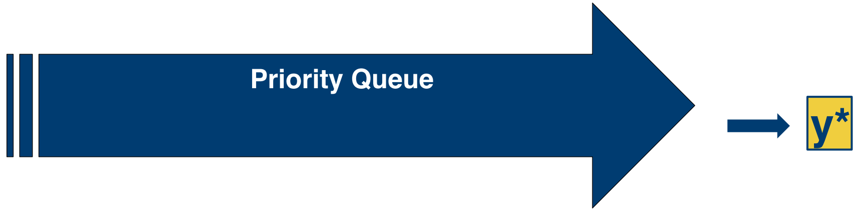


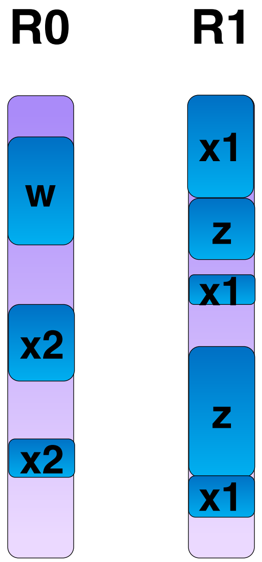
Register Assignment



Register Assignment

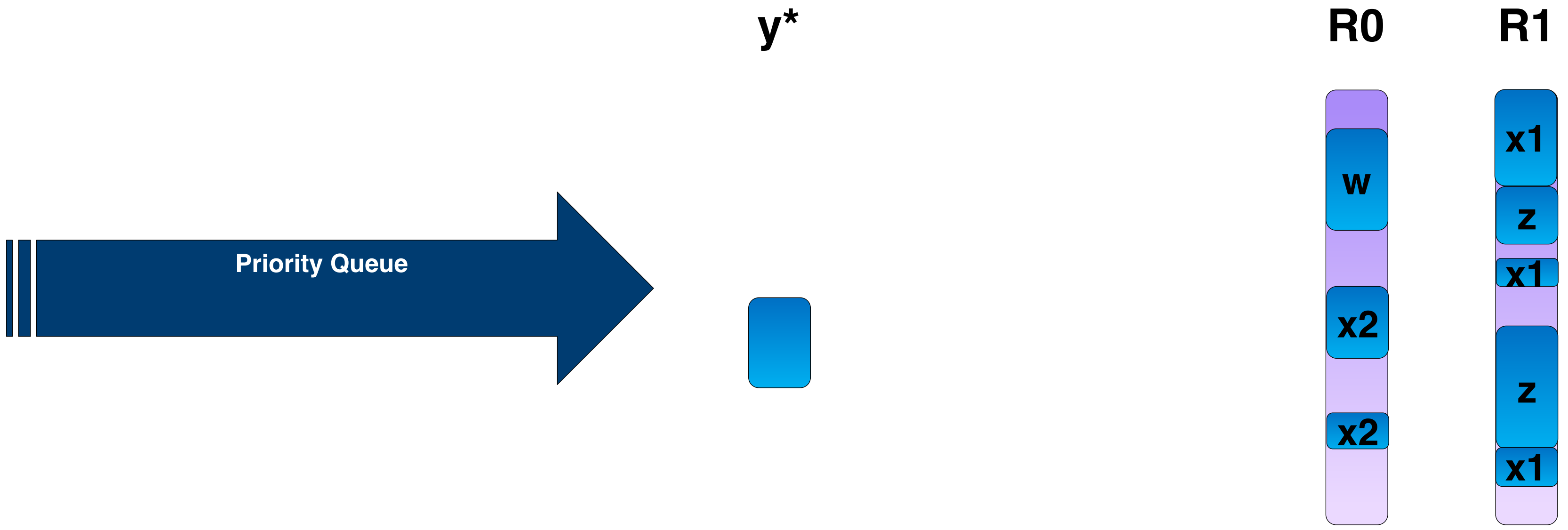
- Dequeue interval with highest priority





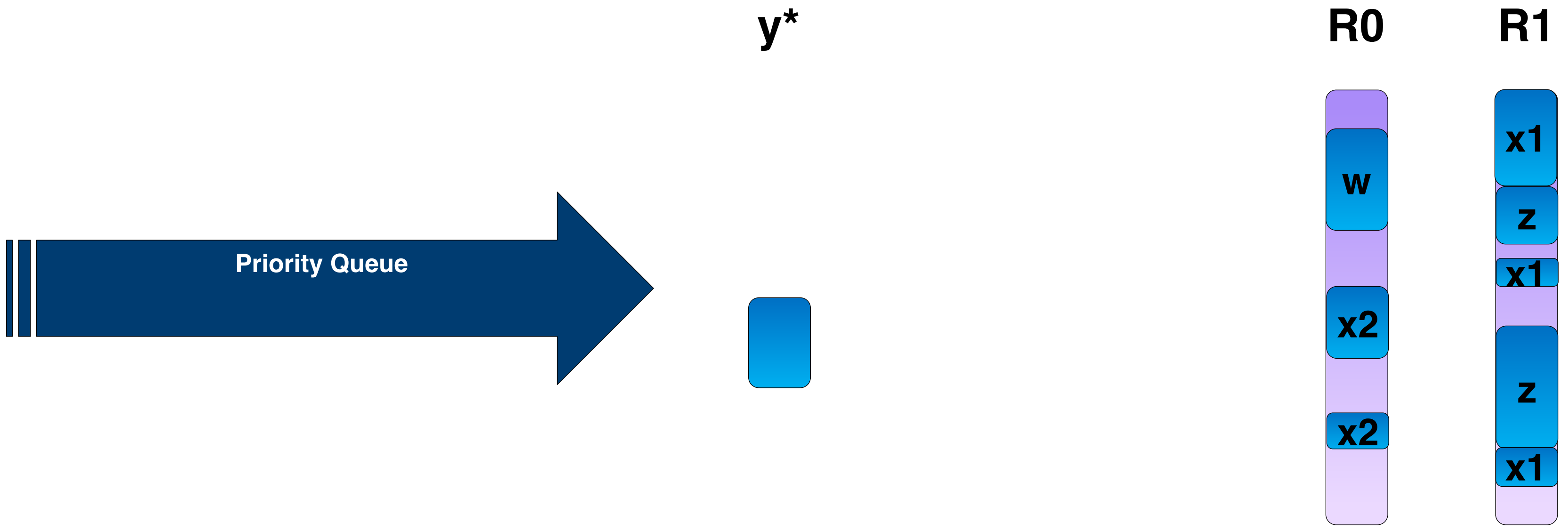
Register Assignment

- Register Assignment



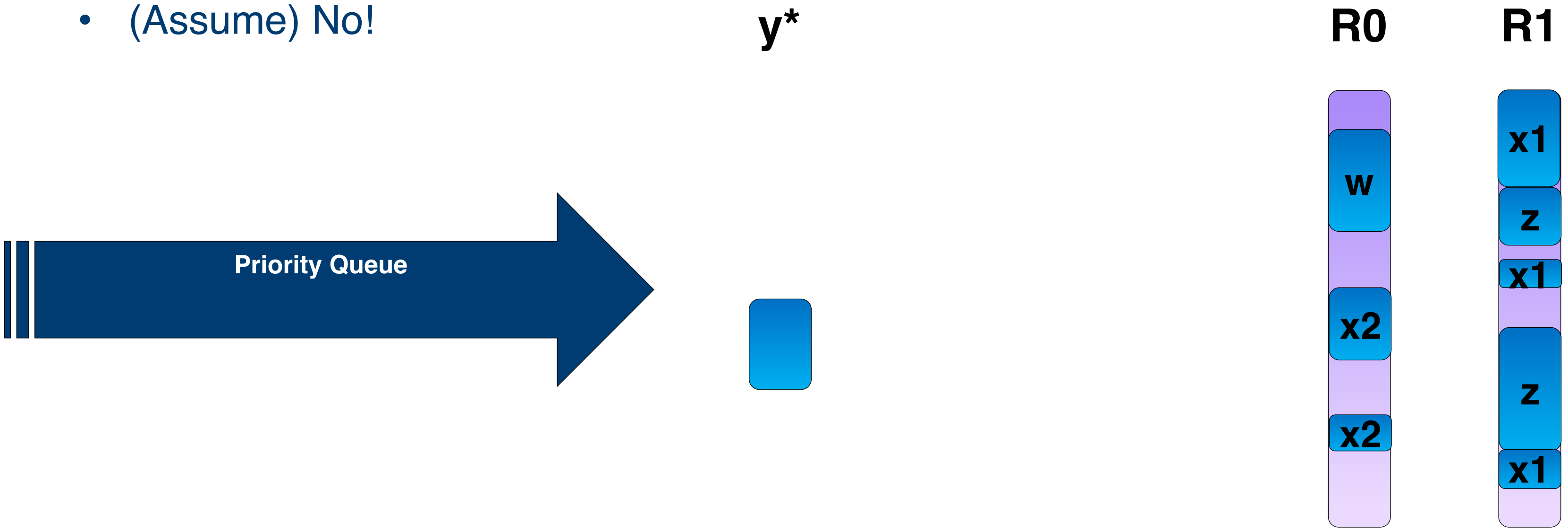
Split

- Is split beneficial?



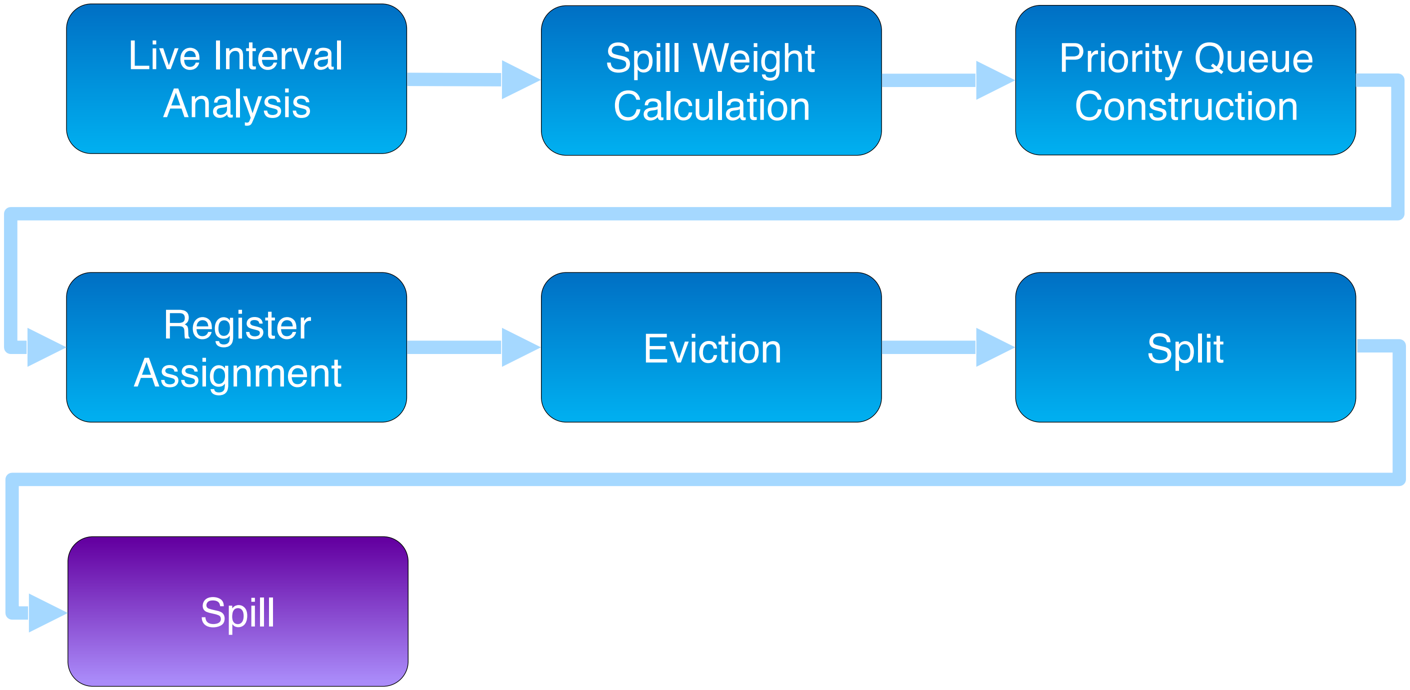
Split

- Is split beneficial?
  - (Assume) No!

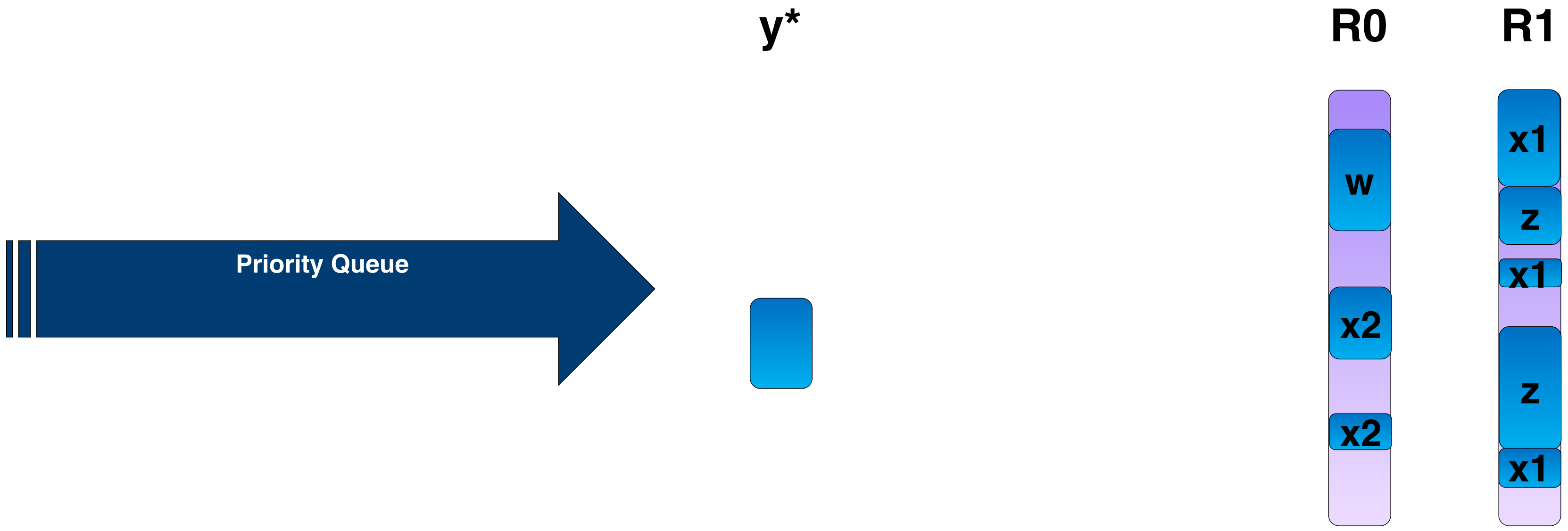


Greedy Register Allocator Overview

- General flow

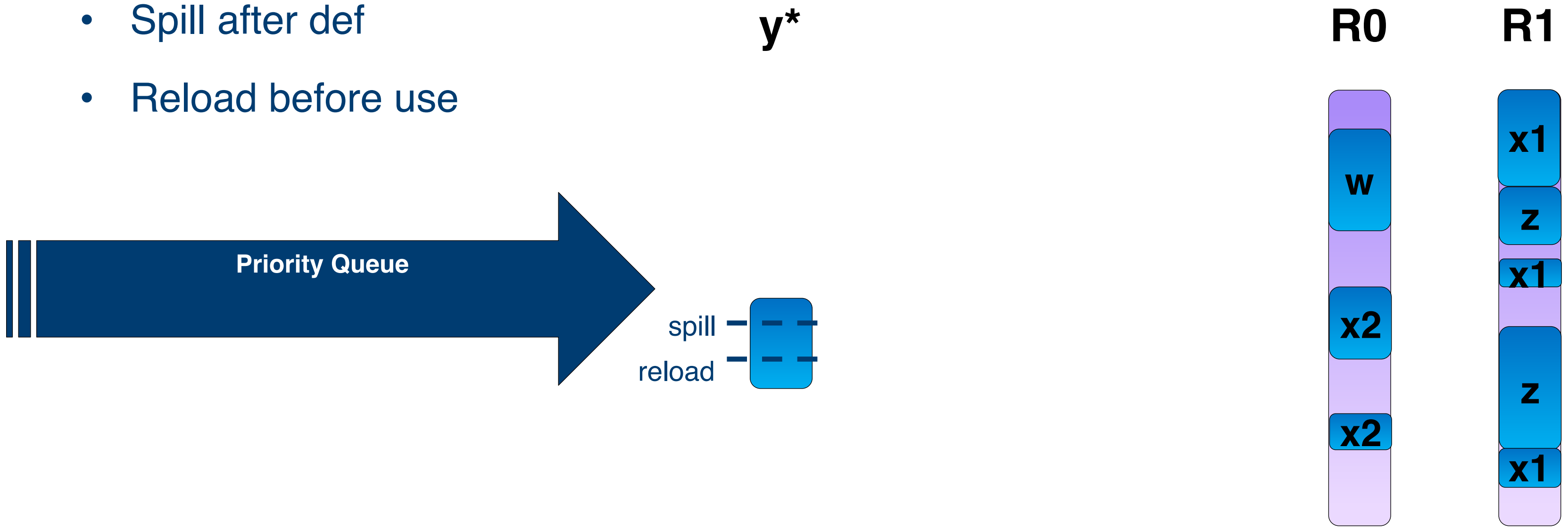


Spill



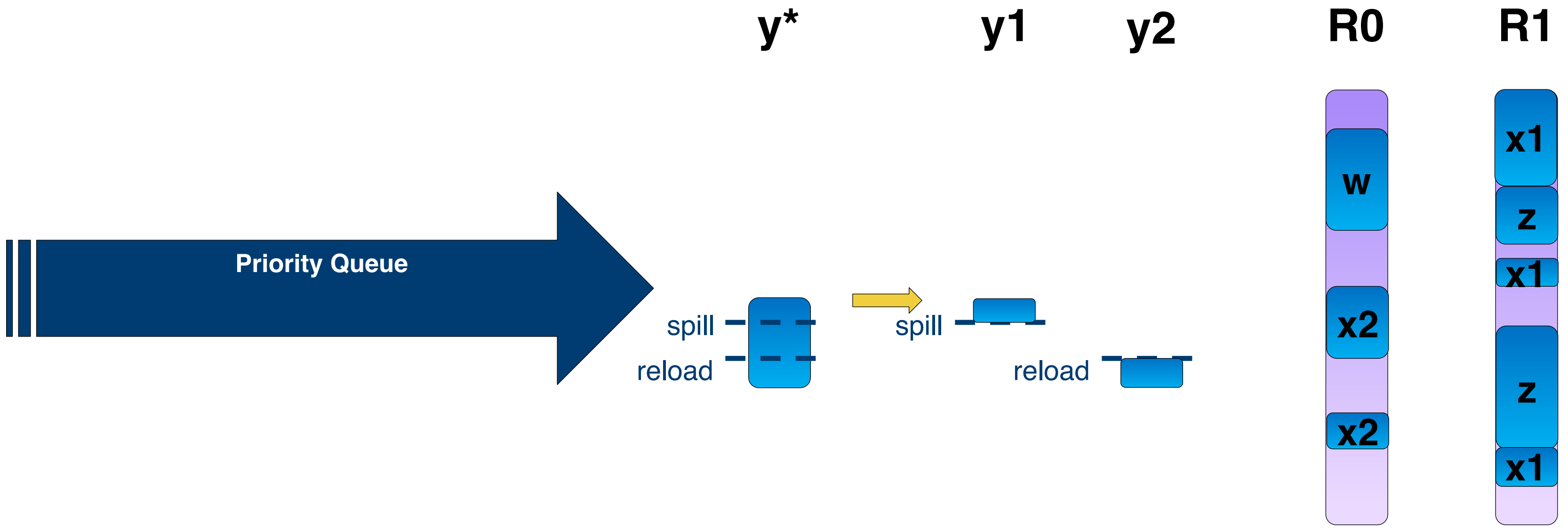
Spill

- Spill around uses
  - Spill after def
  - Reload before use

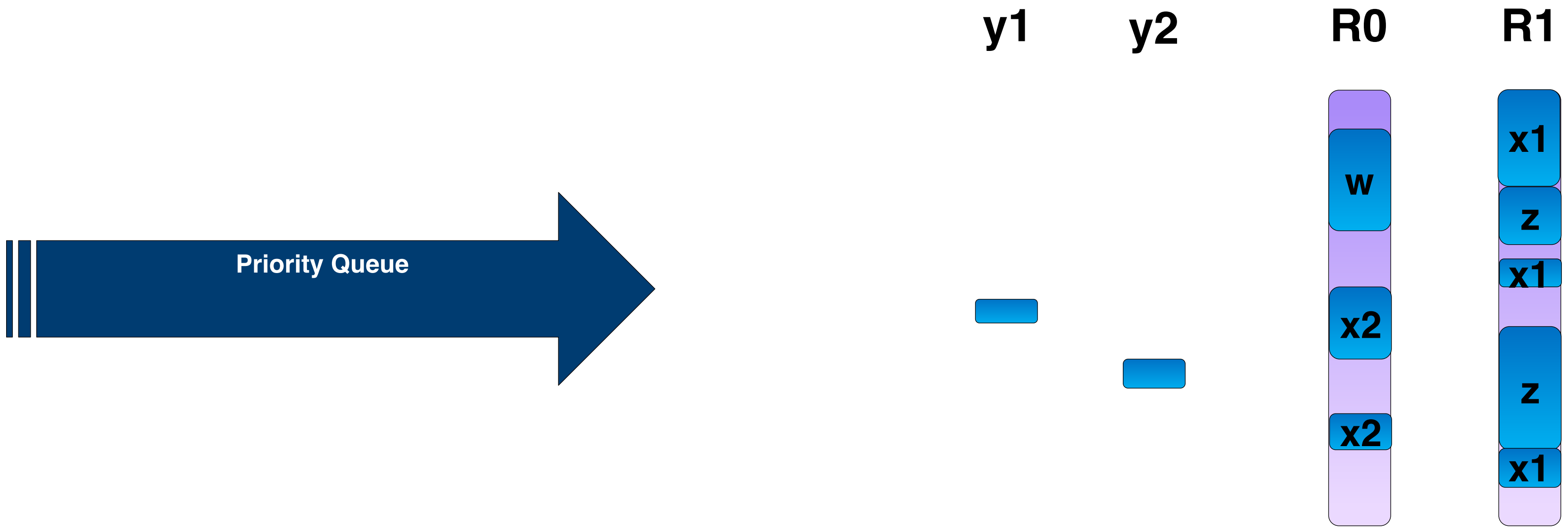


Spill

- Create new intervals for spills and reloads

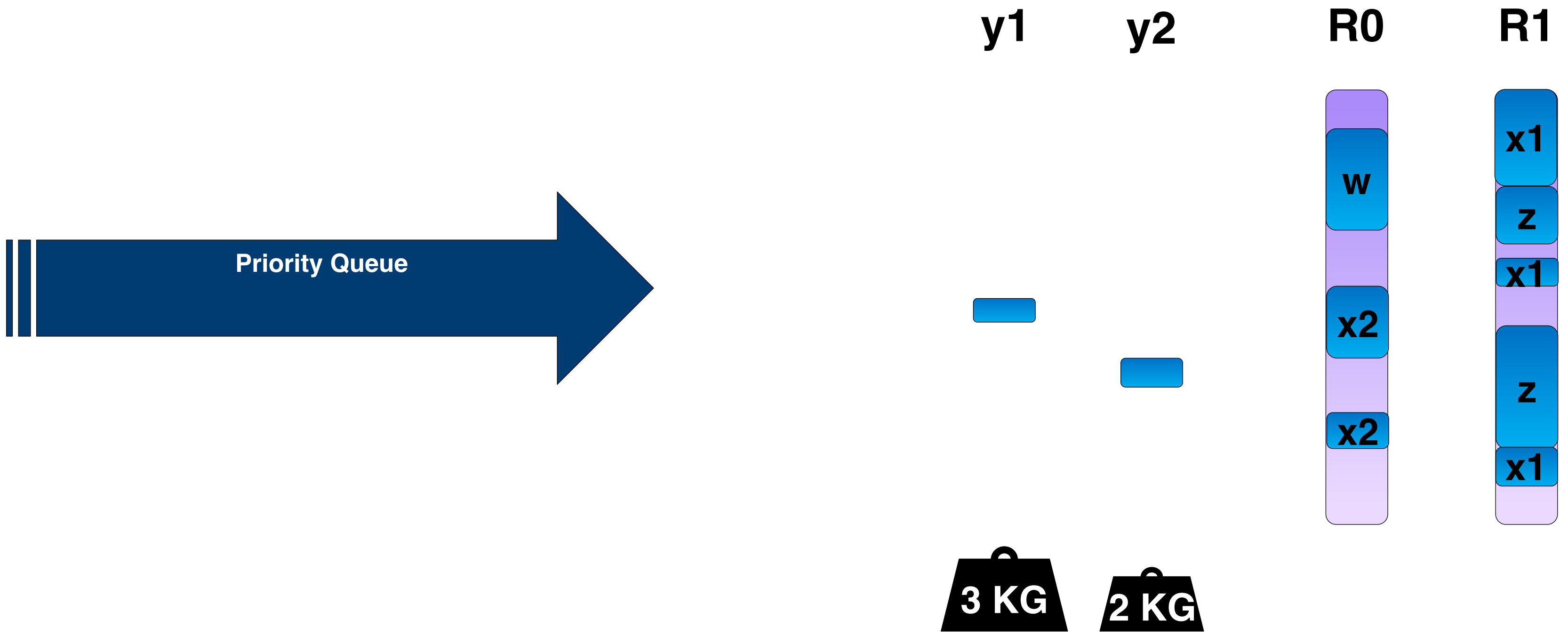


Spill



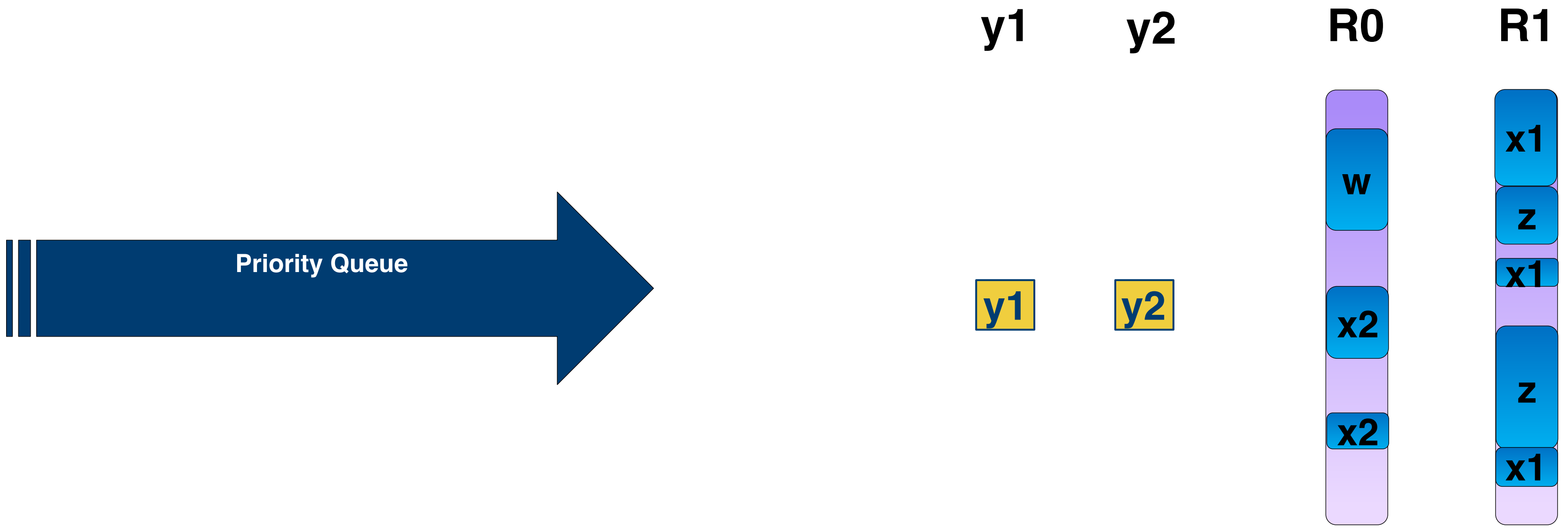
Spill

- Calculate spill weights



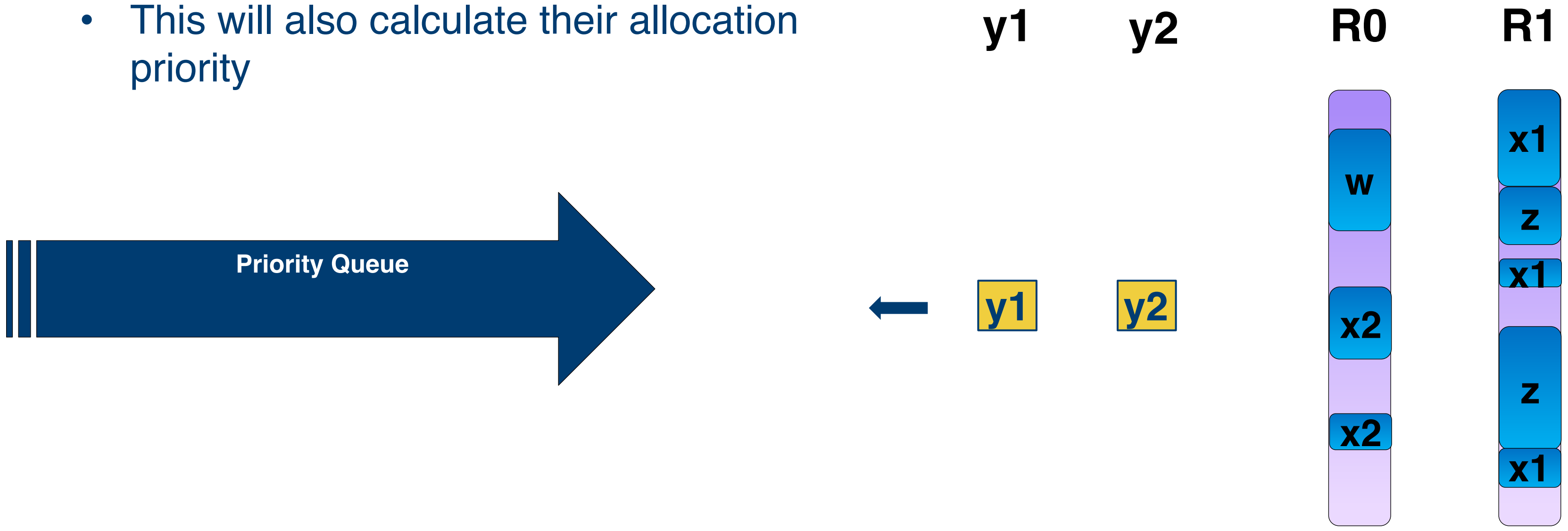
Spill

- Spill

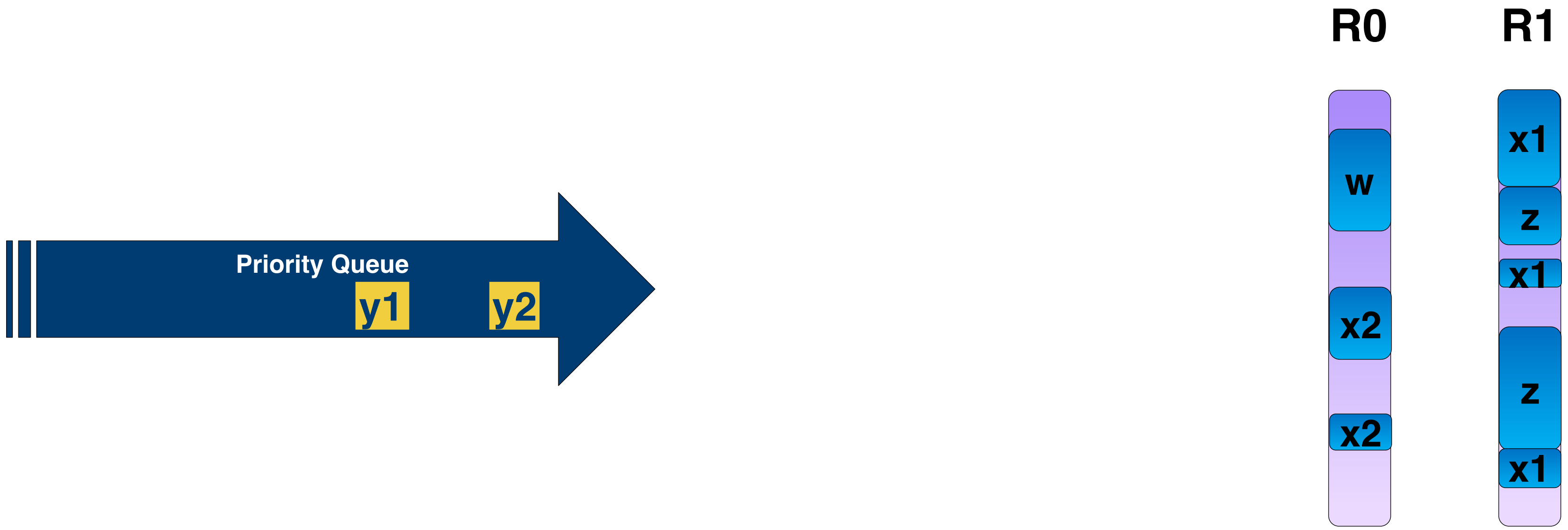


Spill

- Enqueue y1, y2 into the queue
  - This will also calculate their allocation priority

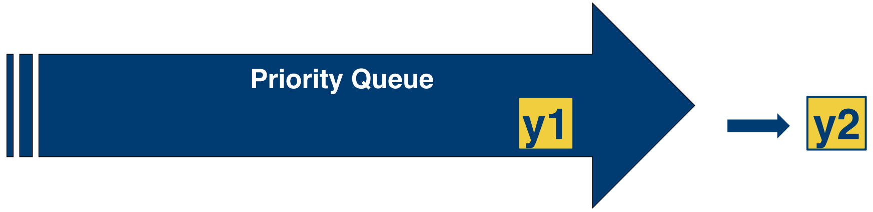


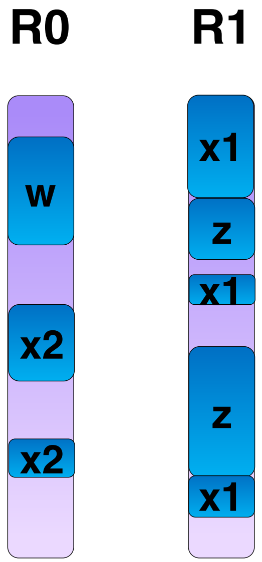
Register Assignment



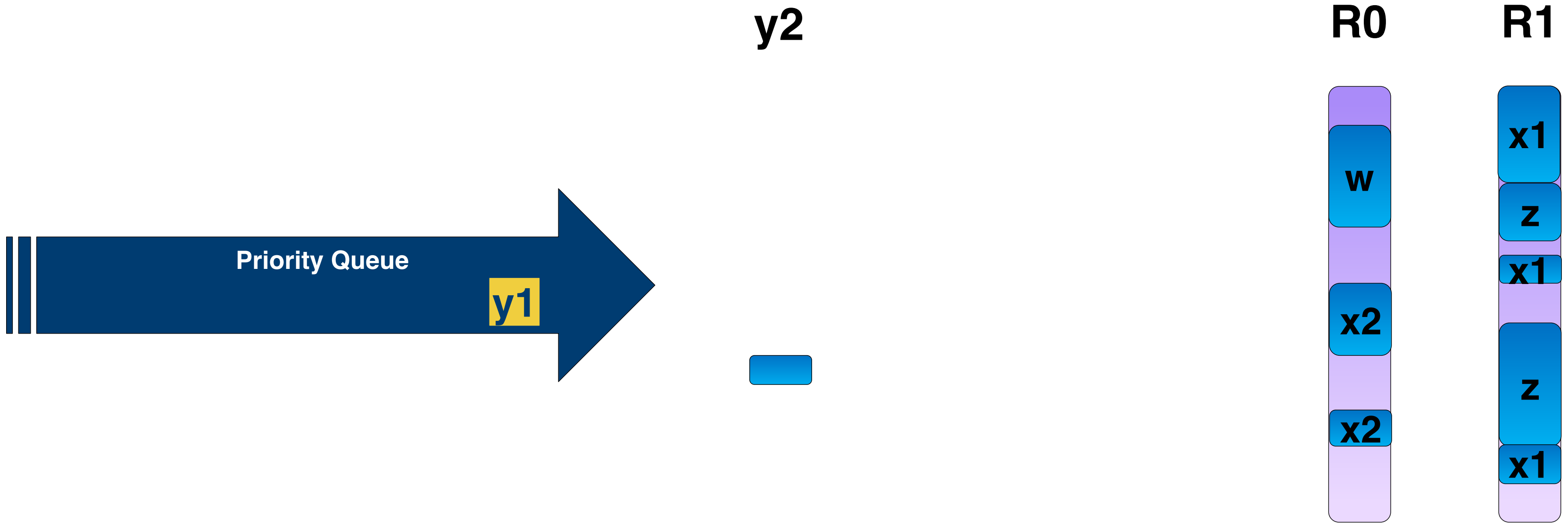
Register Assignment

- Dequeue interval with highest priority



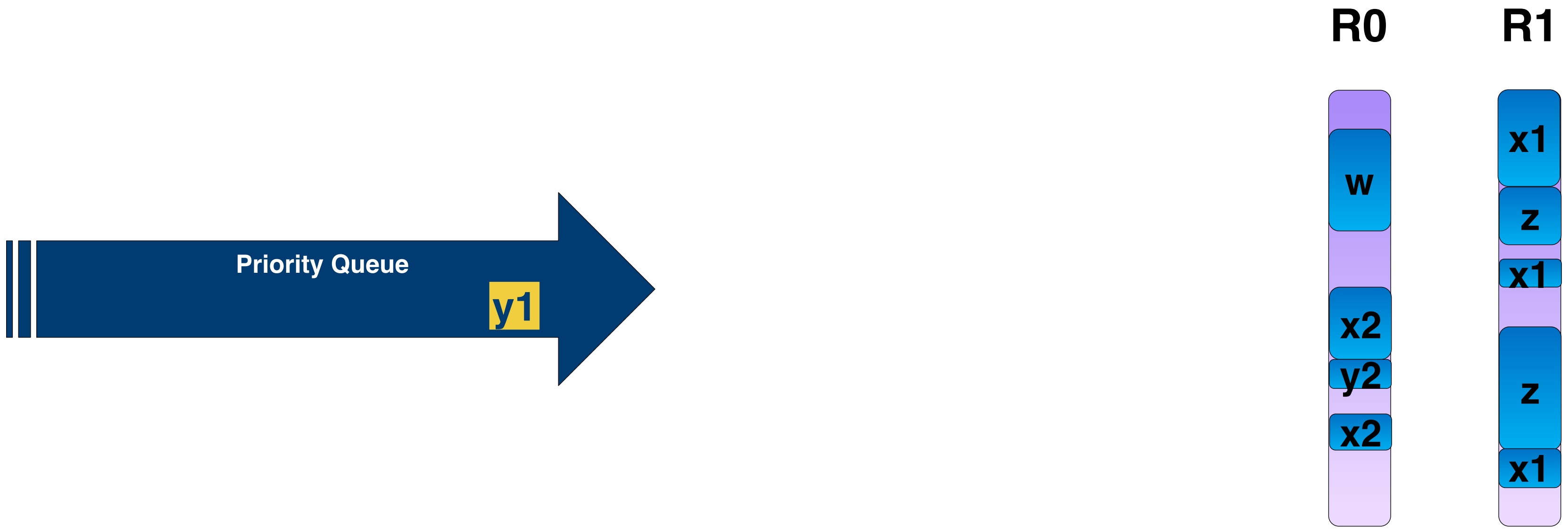


Register Assignment



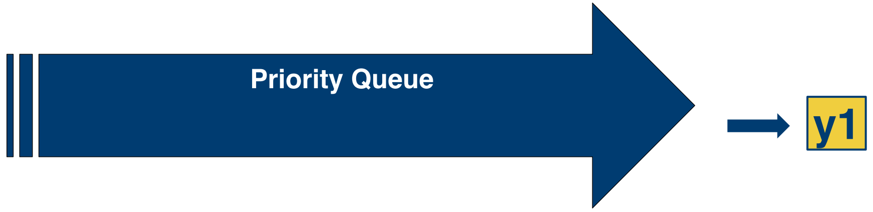
Register Assignment

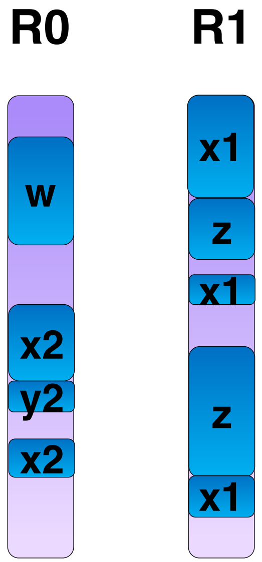
- Assign to available register if possible



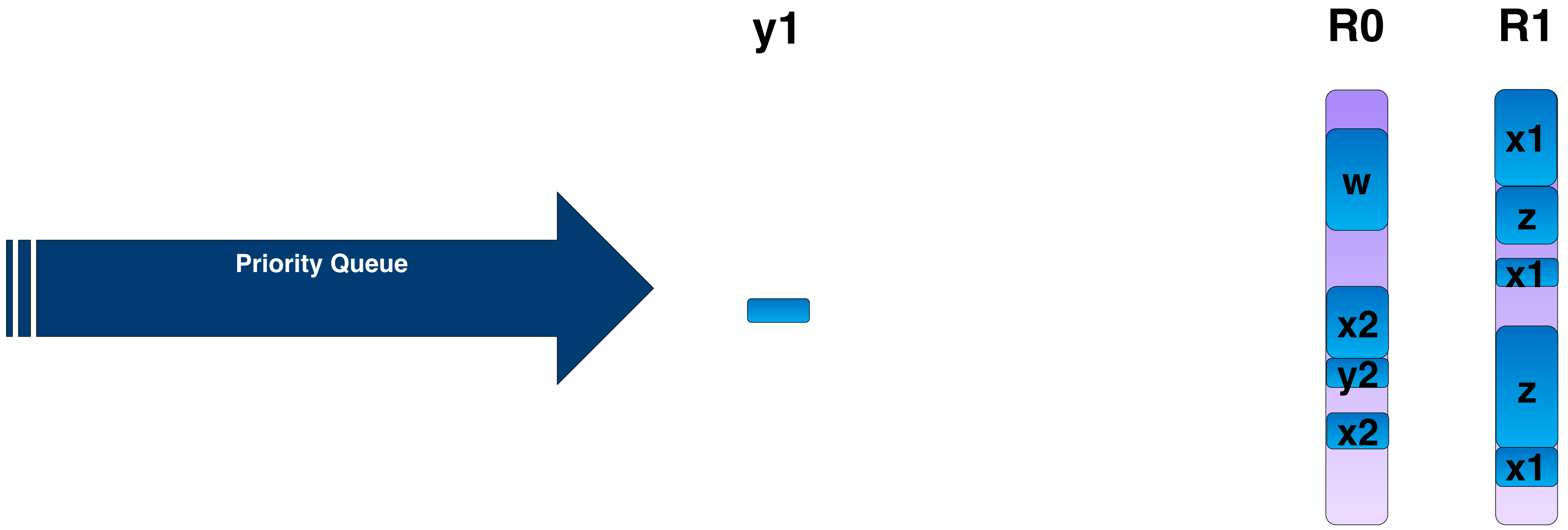
Register Assignment

- Dequeue interval with highest priority



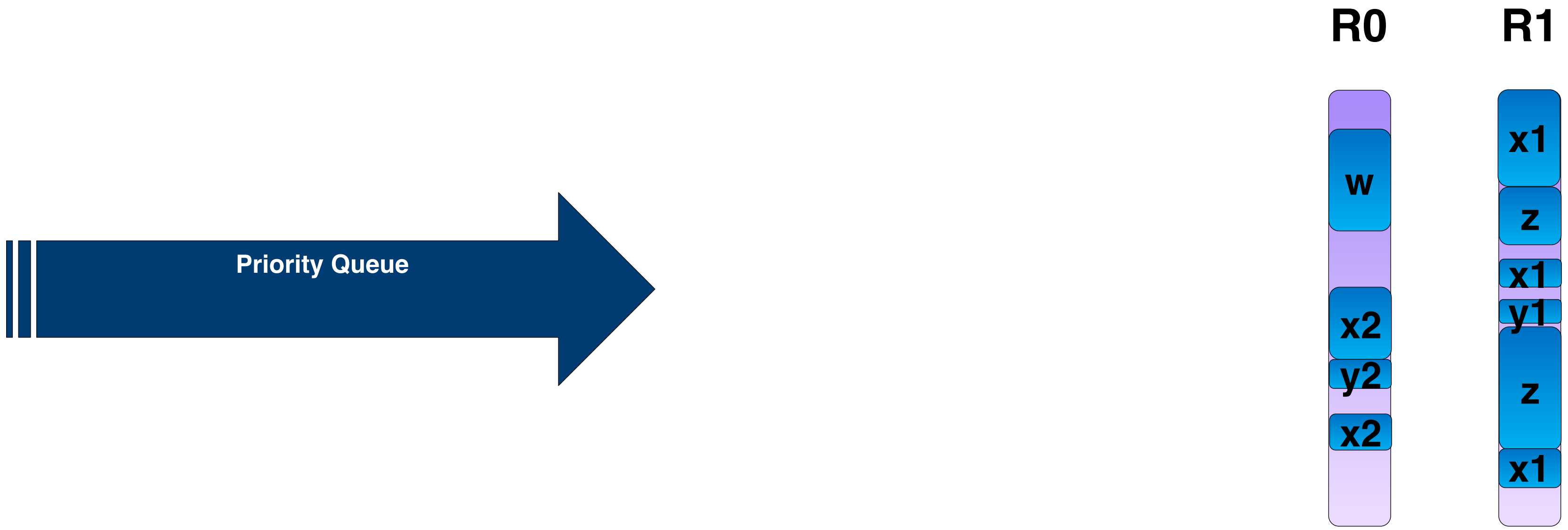


Register Assignment



Register Assignment

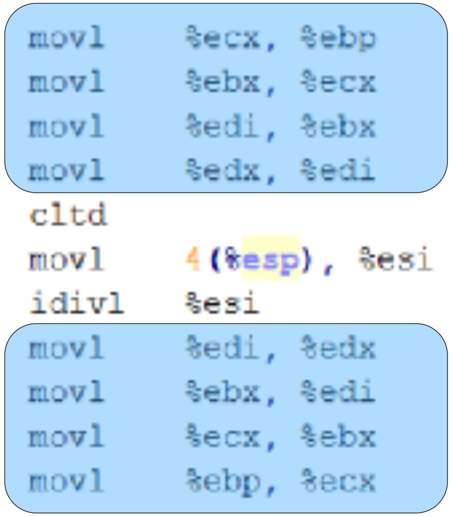
- Assign to available register if possible



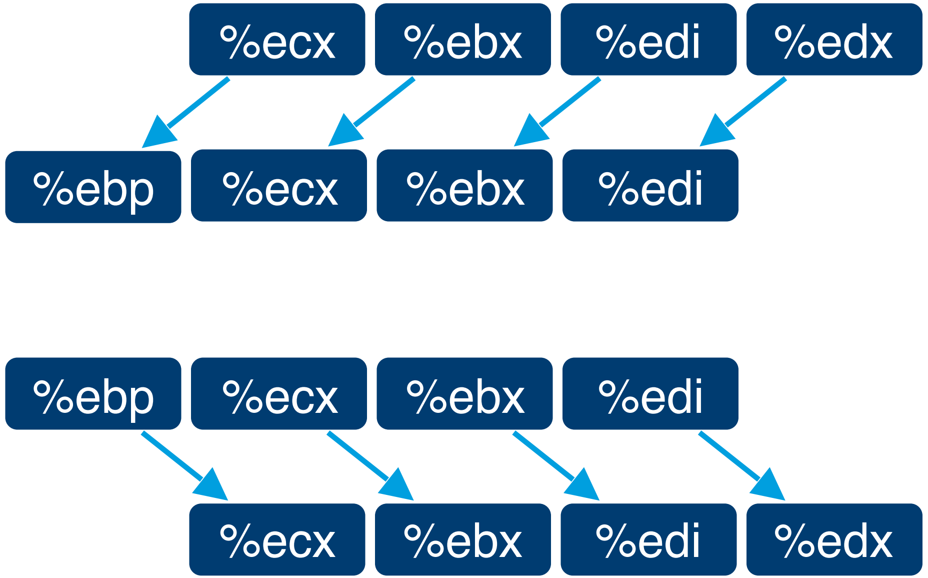
Greedy Register Allocator

- Greedy Register Allocator Overview
- **Region Split**
- Encountered Issues
- Performance Impact

Motivation

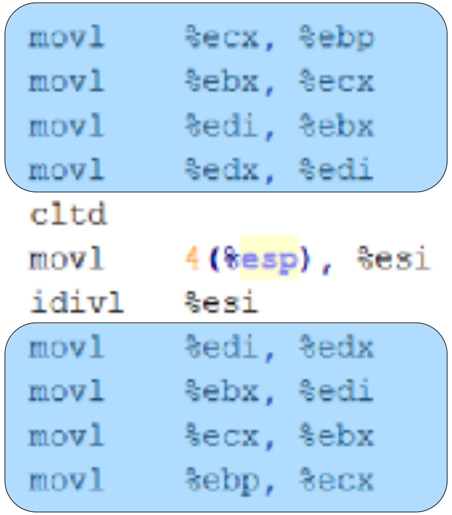


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```



Exploration

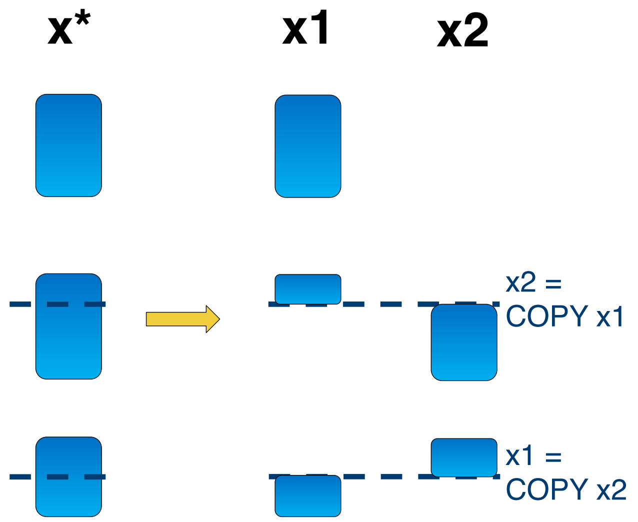
- Why did the register allocator create these redundant mov instructions?

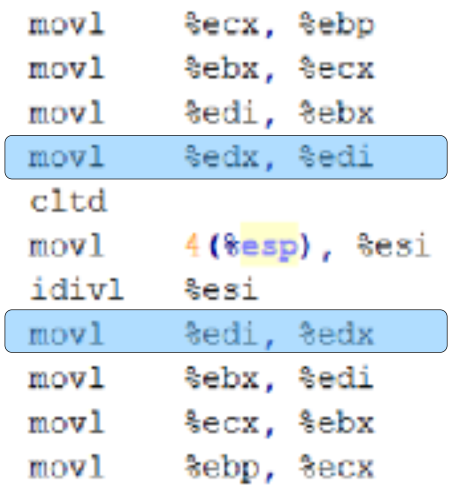


```
movl     %ecx, %ebp
movl     %ebx, %ecx
movl     %edi, %ebx
movl     %edx, %edi
cltd
movl     4(%esp), %esi
idivl    %esi
movl     %edi, %edx
movl     %ebx, %edi
movl     %ecx, %ebx
movl     %ebp, %ecx
```

Exploration

- Why did the register allocator create these redundant mov instructions?
  - Artifacts of split

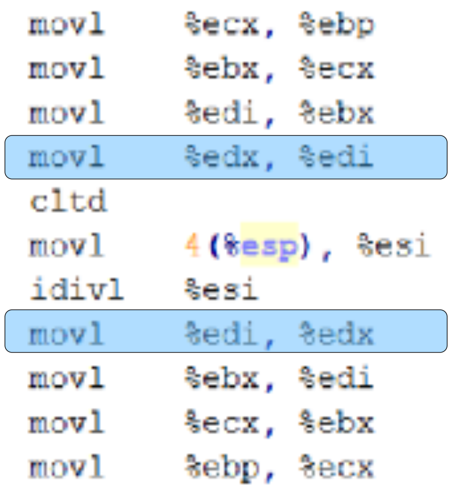




```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Exploration

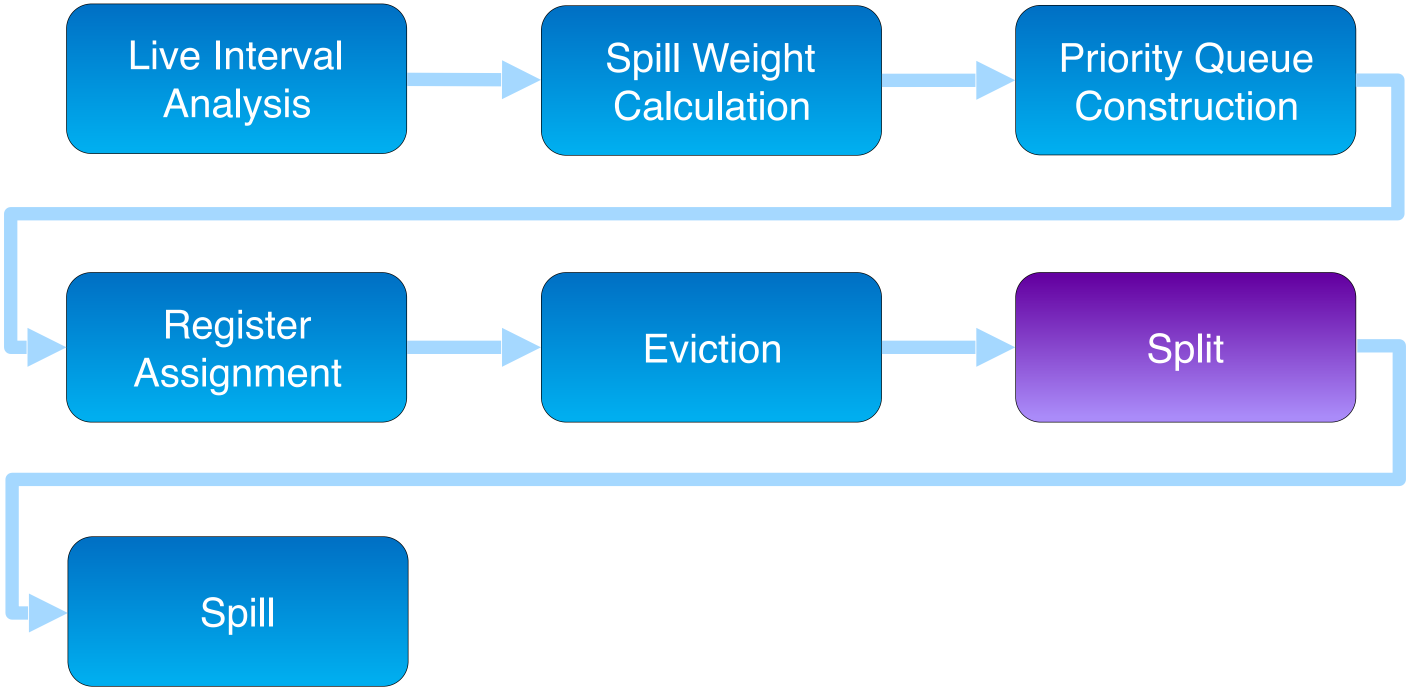
- Why did the register allocator create these redundant mov instructions?
  - Artifacts of split
  - If we would have chosen to do the split differently we could have avoided the redundant mov instructions
  - Why was this way to split was chosen?



```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

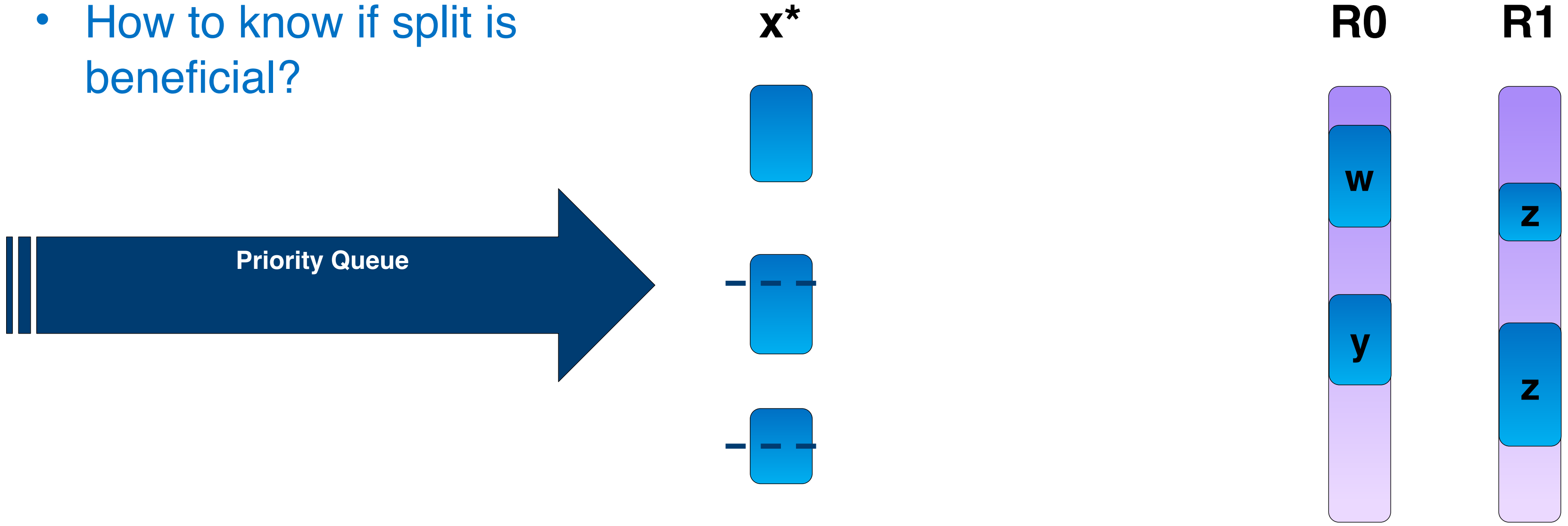
Greedy Register Allocator Overview

- General flow

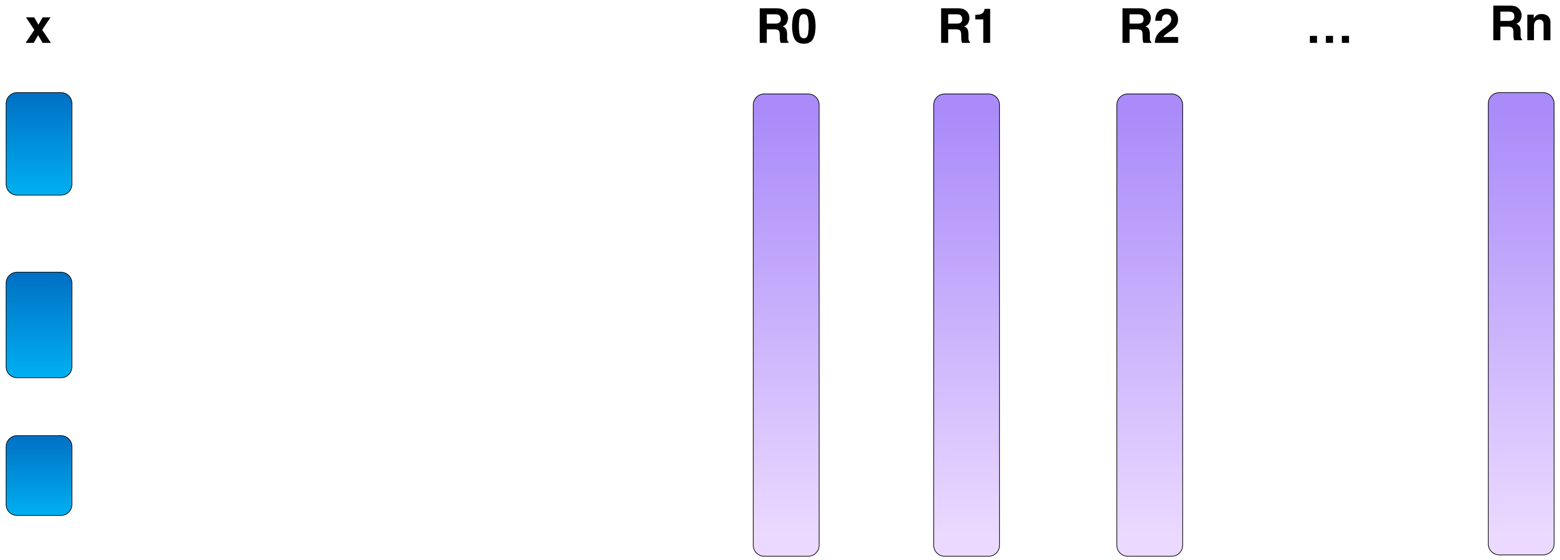


Region Split

- How to find the best way to split?
- How to know if split is beneficial?

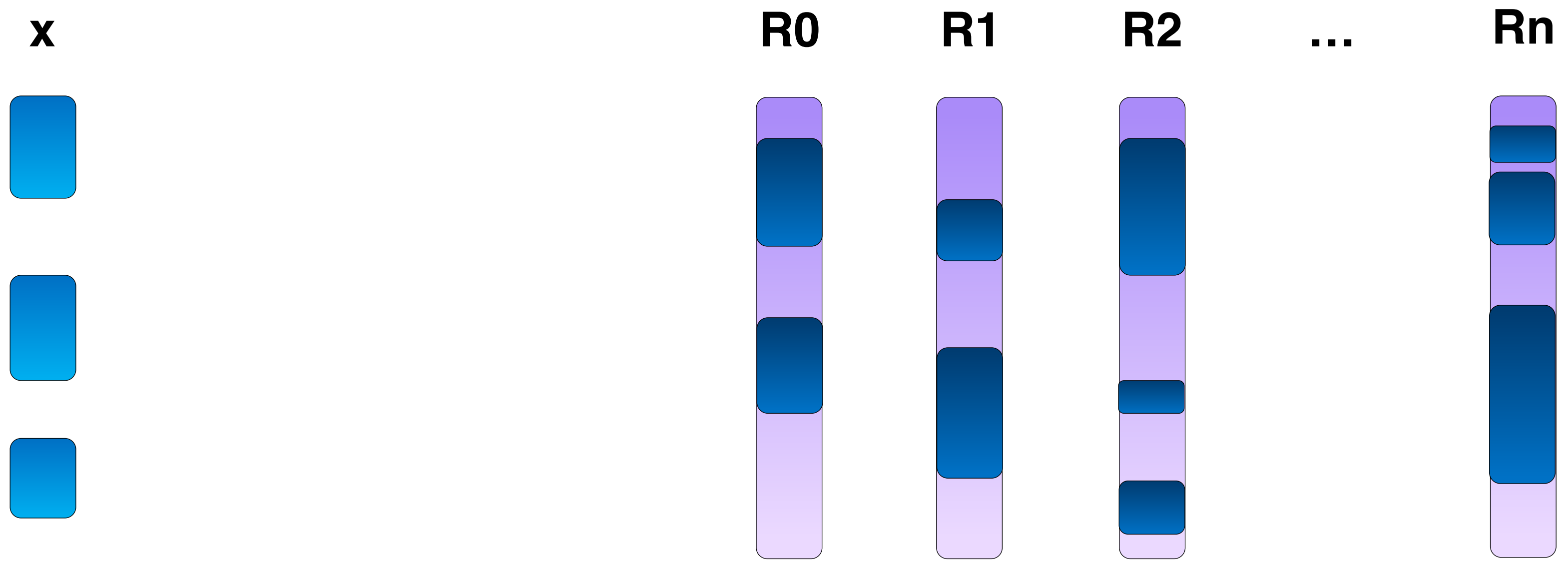


Find Best Split



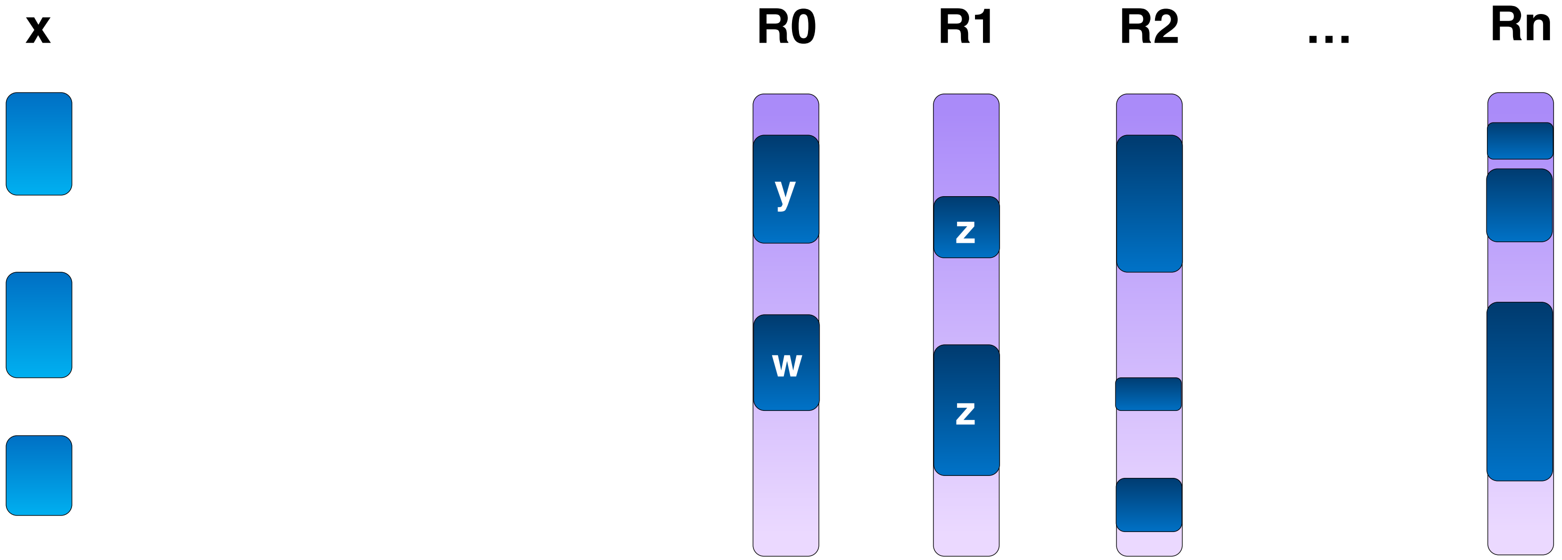
Find Best Split

- The registers already have assigned intervals



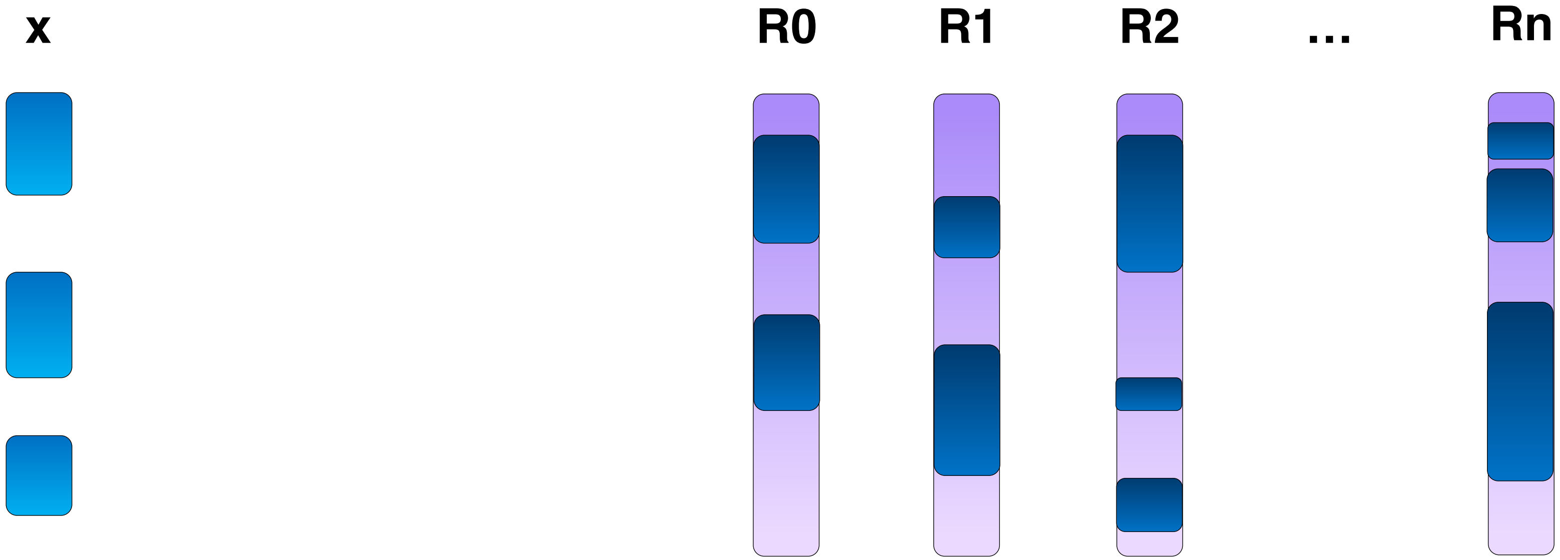
Find Best Split

- The registers already have assigned intervals



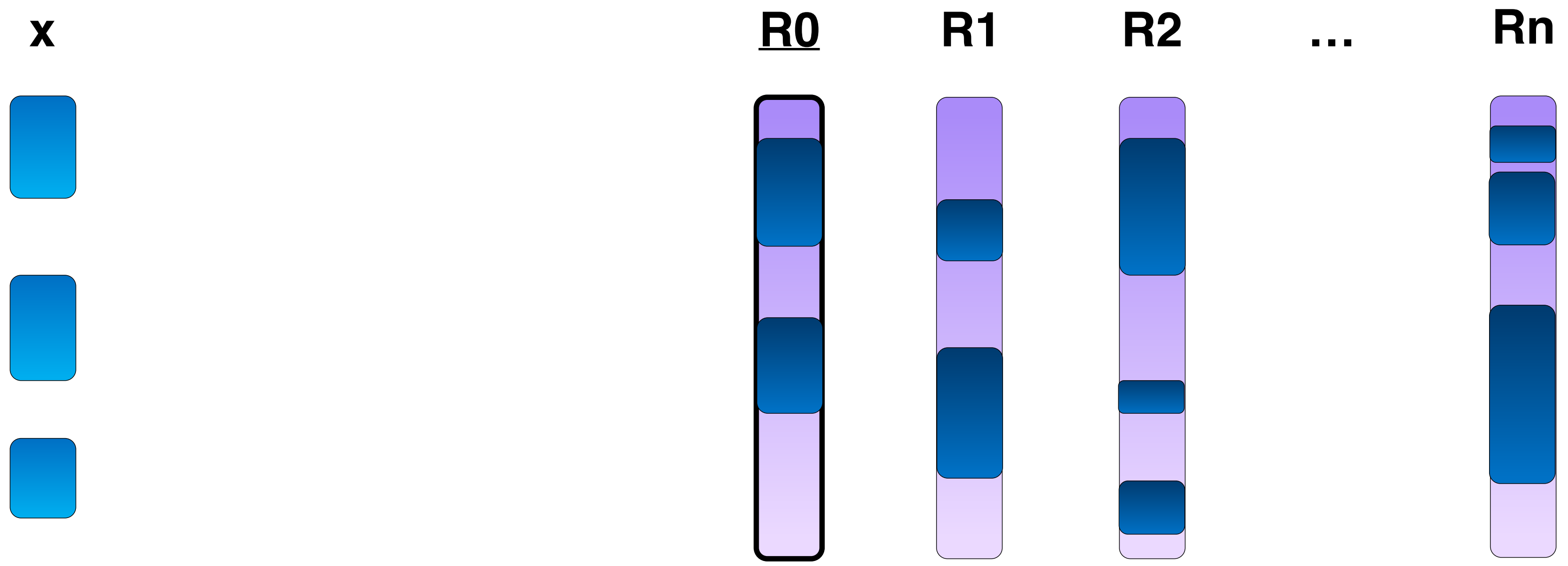
Find Best Split

- The registers already have assigned intervals
  - These intervals impose allocation constraints



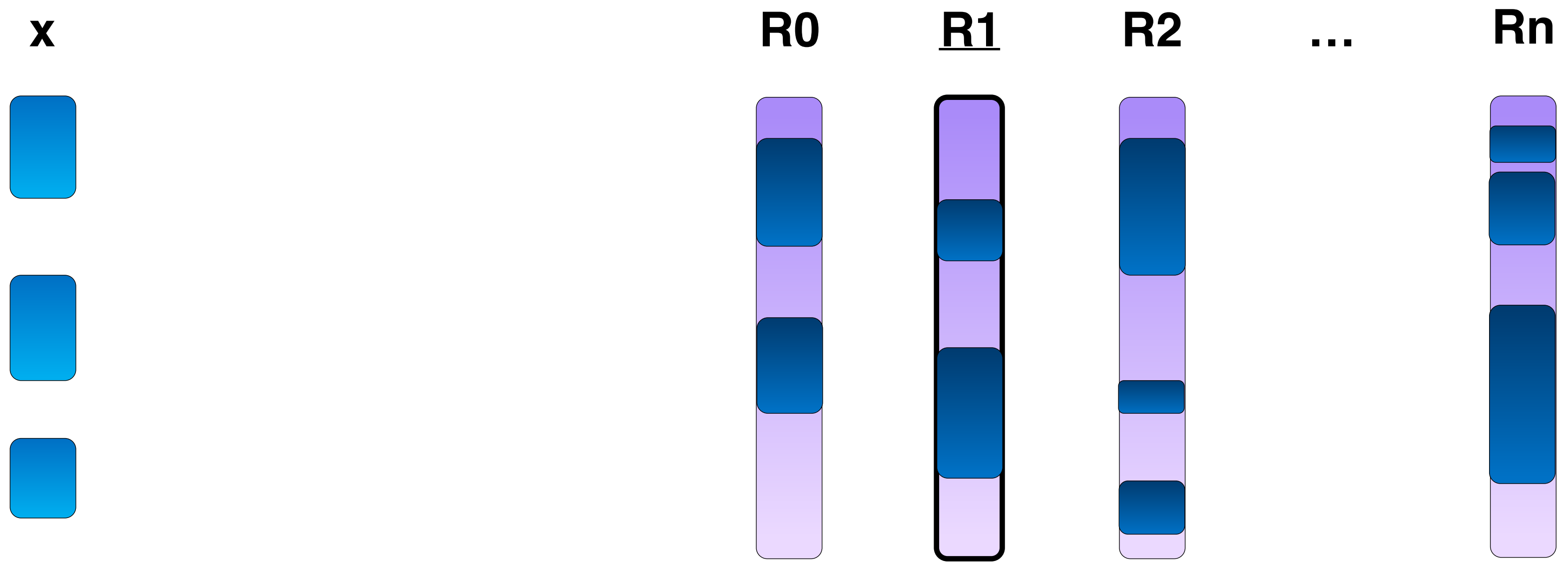
Find Best Split

- Do the split of x for each one of the registers



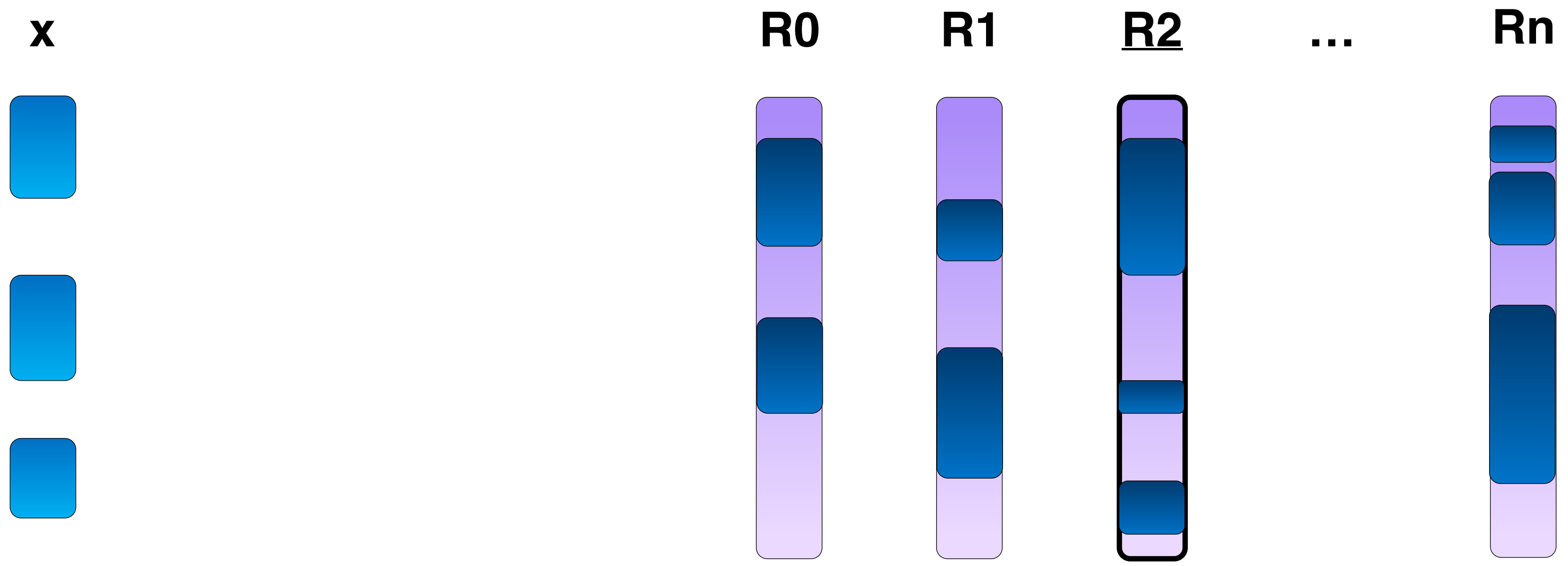
Find Best Split

- Do the split of x for each one of the registers



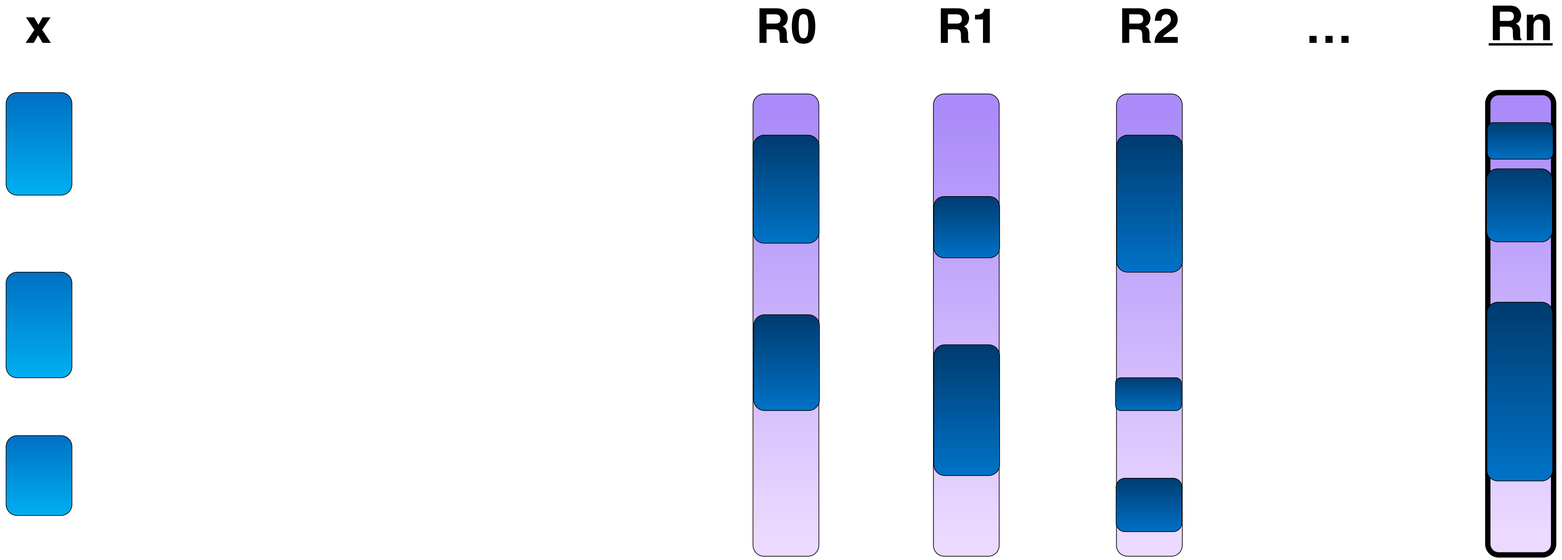
Find Best Split

- Do the split of x for each one of the registers



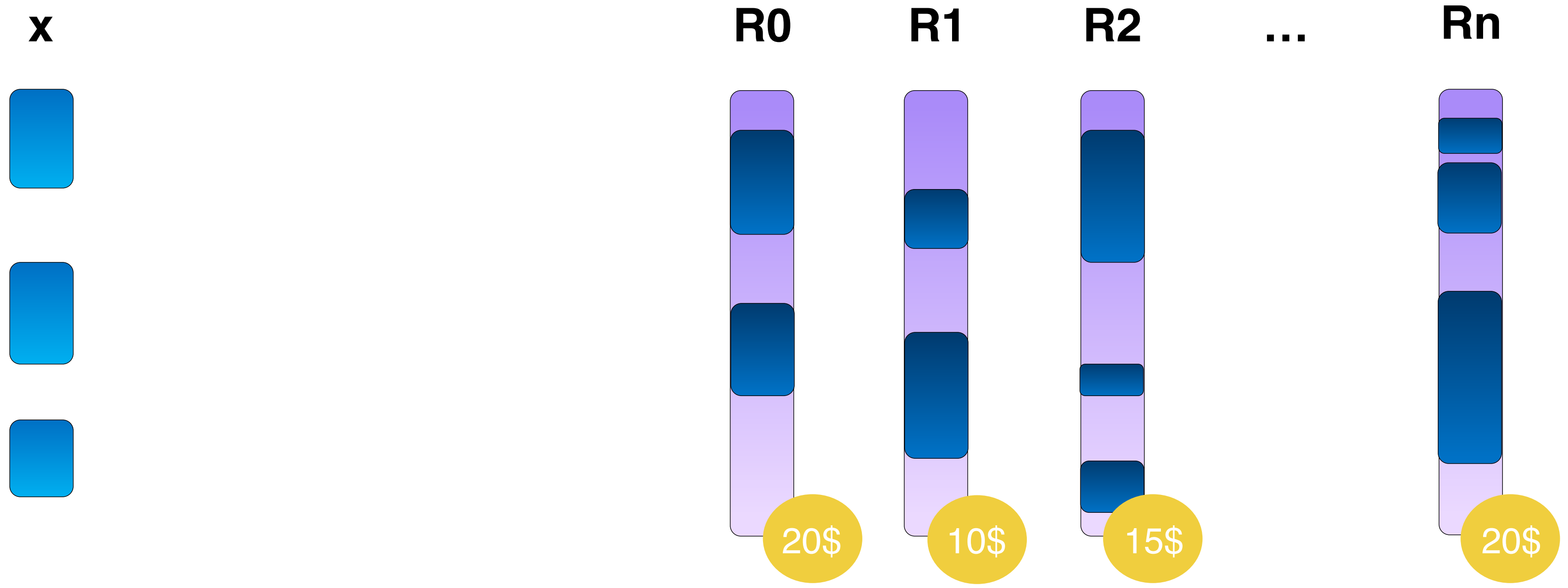
Find Best Split

- Do the split of x for each one of the registers



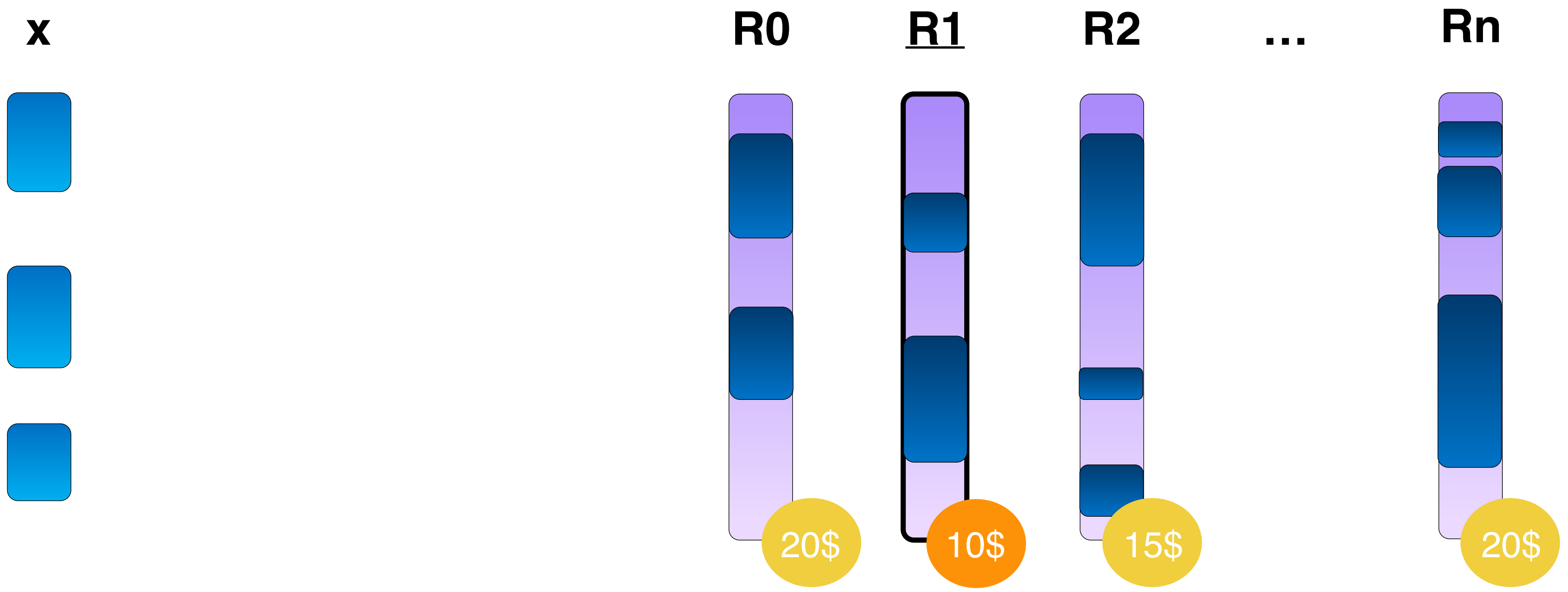
Find Best Split

- Do the split of x for each one of the registers
  - Estimate split cost, e.g. the amount of spill code this split may cause



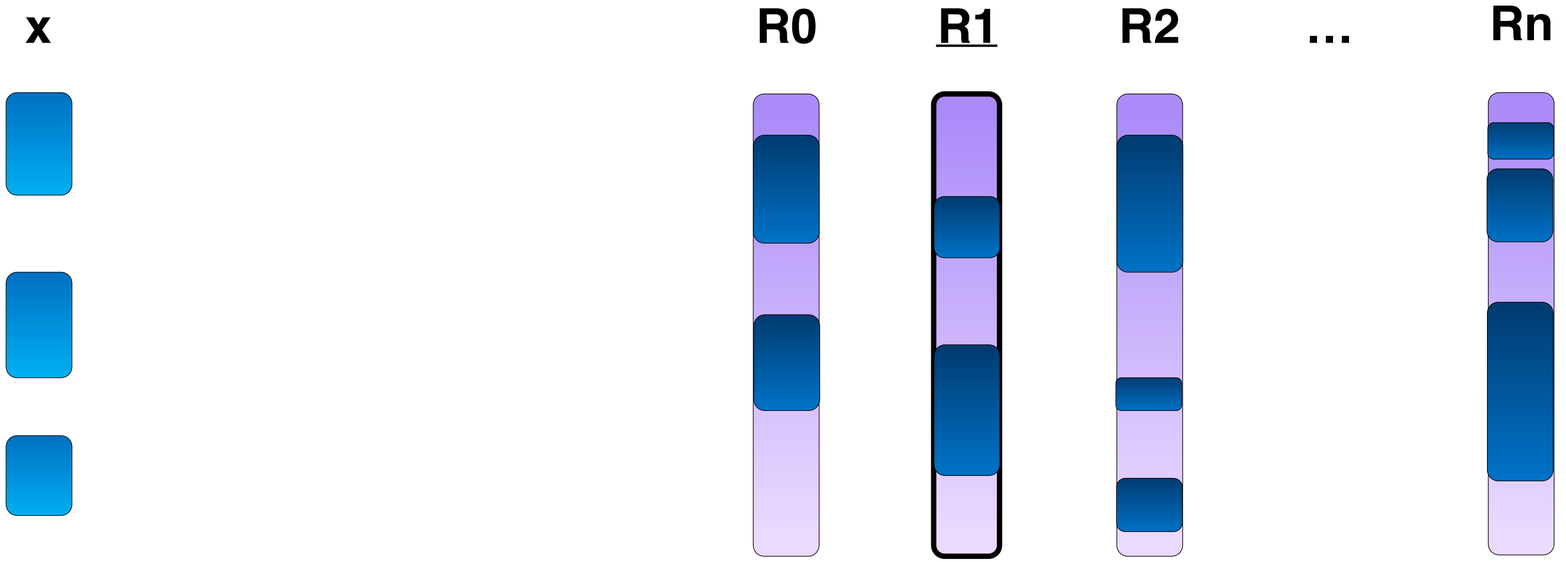
Find Best Split

- Do the split of x for each one of the registers
  - Choose the cheapest one



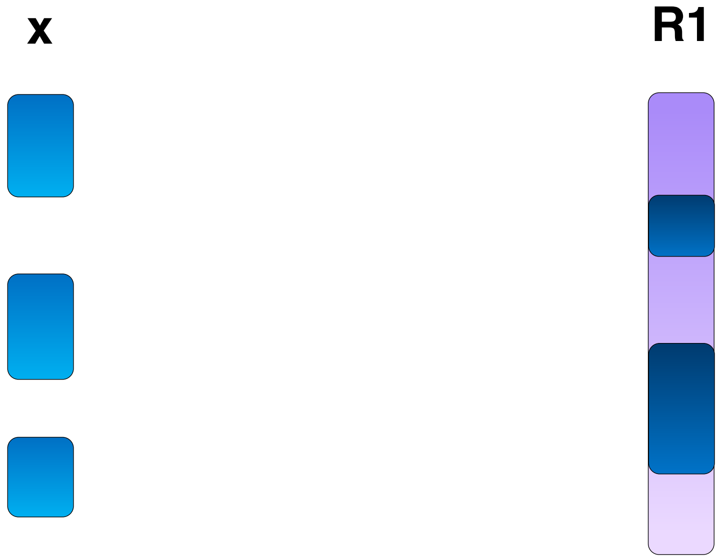
Find Best Split

- Do the split of x for each one of the registers



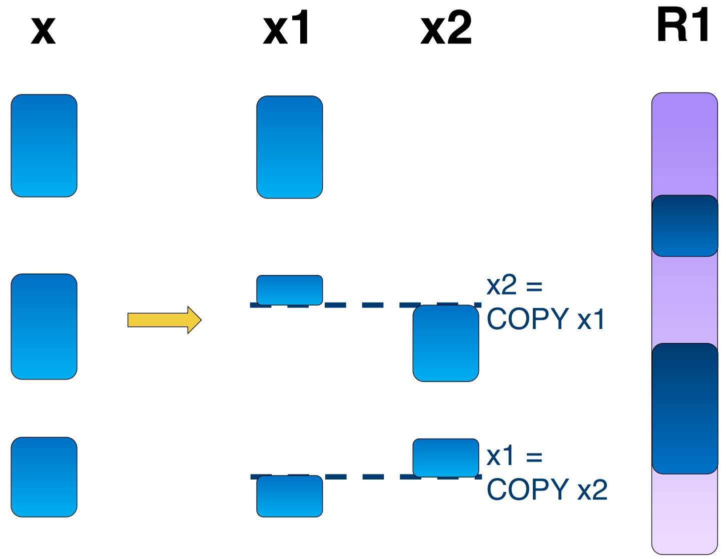
Find Best Split for Given Register

- How do we do the best split for a given register R1?



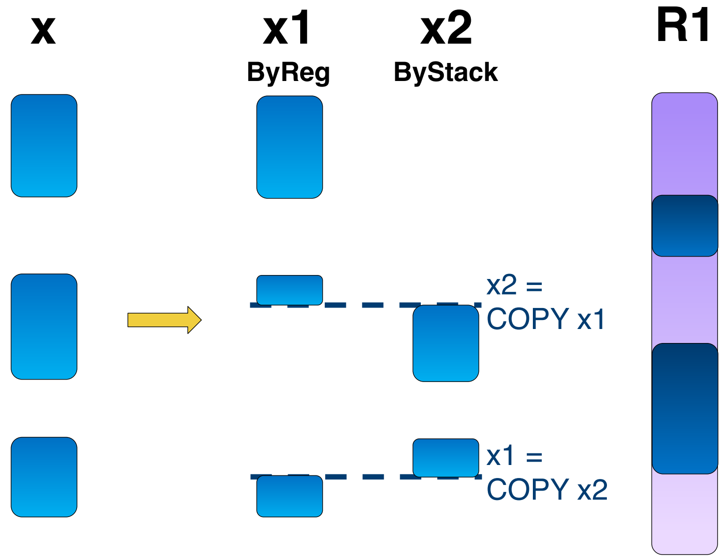
Find Best Split for Given Register

- The region split is usually divided into 2 intervals



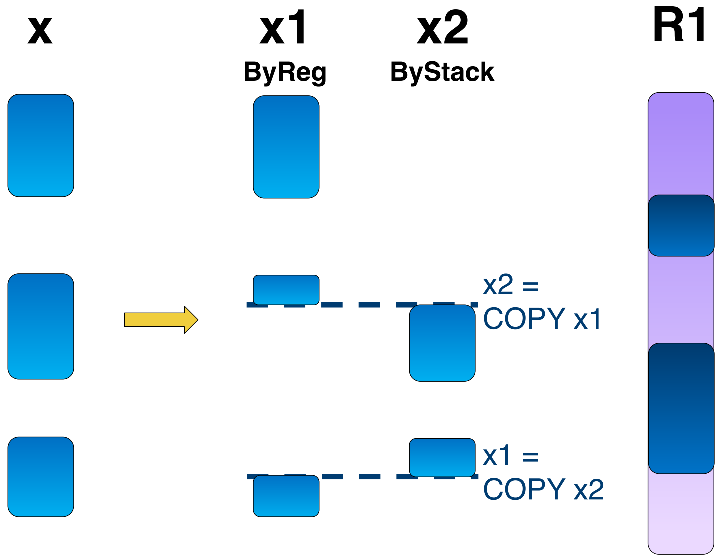
Find Best Split for Given Register

- The region split is usually divided into 2 intervals
  - ByReg
    - Parts of x that pass on R1 register
    - Should comply with current allocation constraints provided by intervals already assigned to R1
  - ByStack
    - Parts of x that pass on or on the stack
    - Usually where the already allocated R1 intervals interfere with x



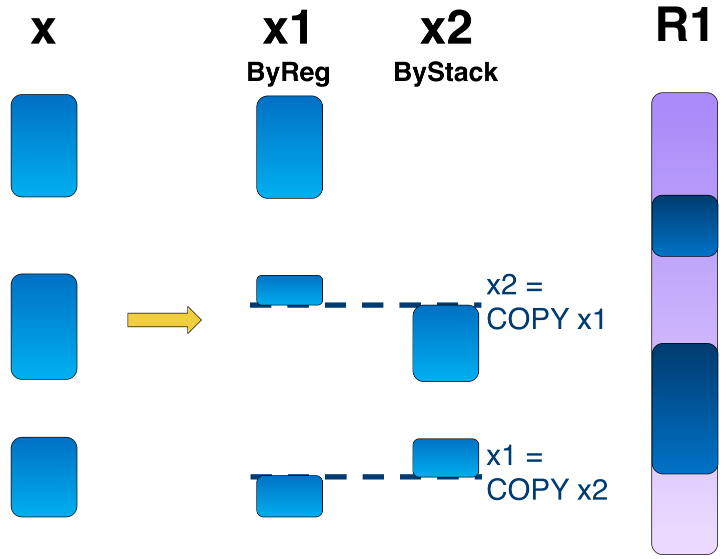
Find Best Split for Given Register

- A good split



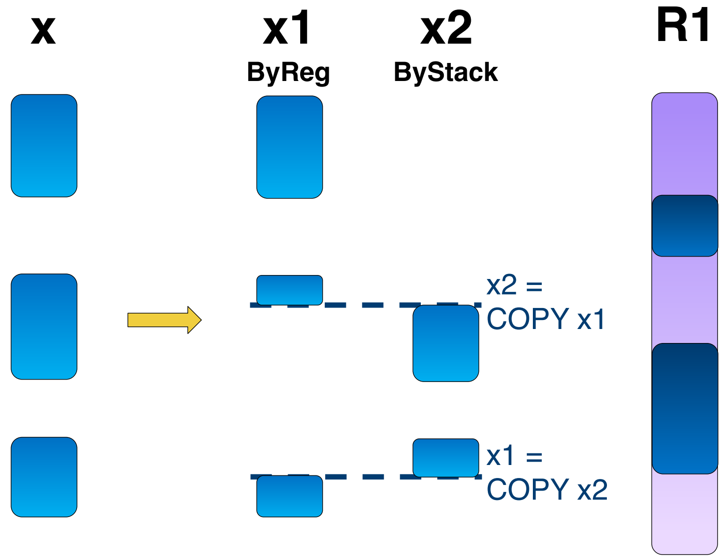
Find Best Split for Given Register

- A good split
  - Reduces the transitions between ByReg and ByStack
    - Each such transitions is potentially a spill/reload
      - In case ByStack is not allocated to another register
  - Places the transitions in blocks less frequently executed



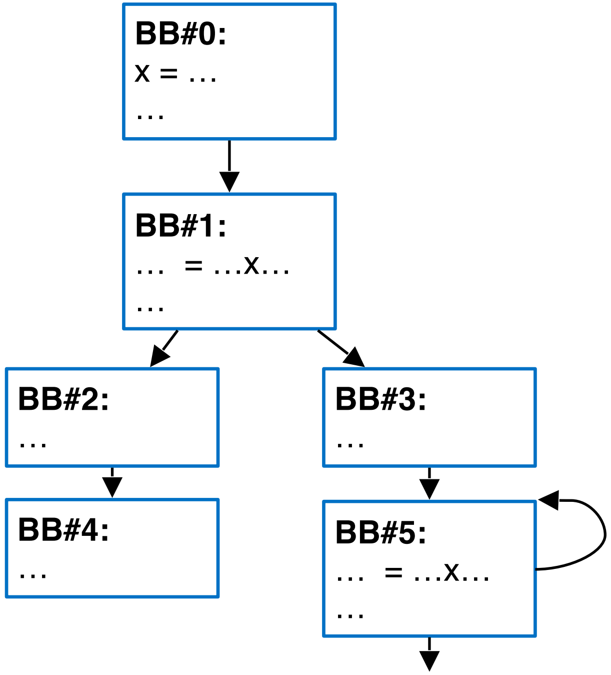
Find Best Split for Given Register

- A good split
  - Reduces the transitions between ByReg and ByStack
    - Each such transitions is potentially a spill/reload
      - In case ByStack is not allocated to another register
  - Places the transitions in blocks less frequently executed
  - Use Hopfield neural network
    - Converges to a result that satisfies the above characteristics



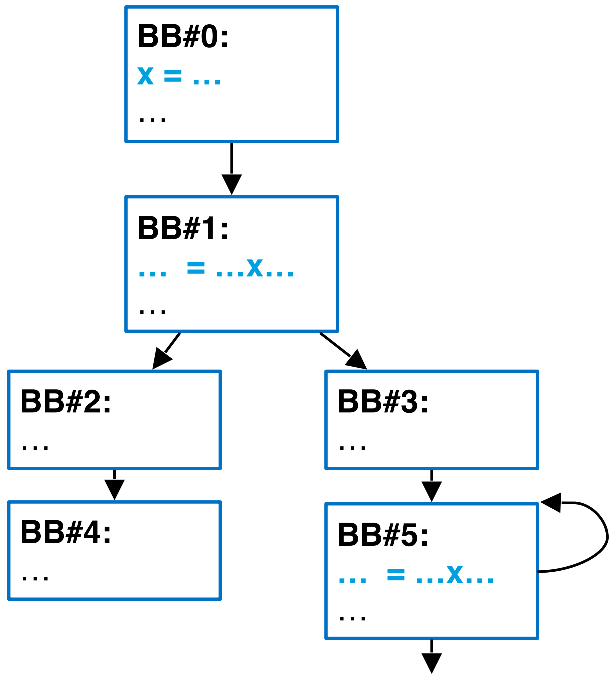
Determine if Split is Beneficial

- Split reduces the amount of spills compared to spilling around uses



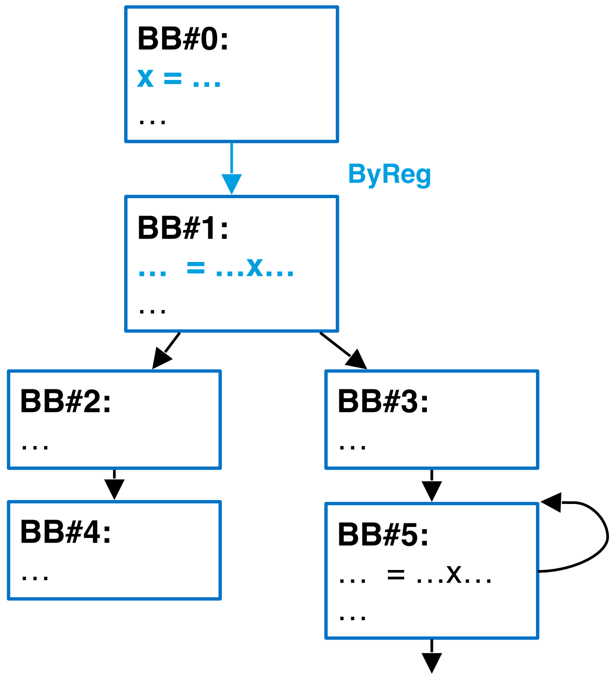
Determine if Split is Beneficial

- Split reduces the amount of spills compared to spilling around uses
  - Use/def blocks must have x in a register at some point



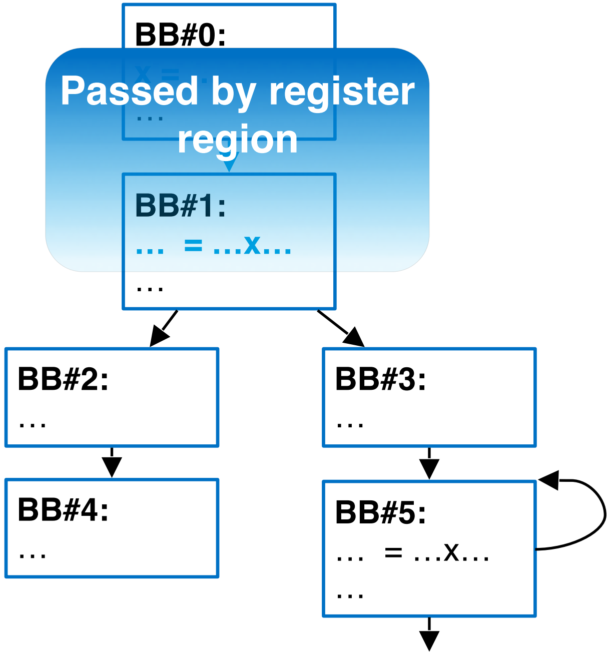
Determine if Split is Beneficial

- Split reduces the amount of spills compared to spilling around uses
  - Use/def blocks must have x in a register at some point
  - If the split can create “regions” of several basic blocks where x is passed by register this will reduce the amount of spills
    - Only if constraints allow it



Determine if Split is Beneficial

- Split reduces the amount of spills compared to spilling around uses
  - Use/def blocks must have x in a register at some point
  - If the split can create “regions” of several basic blocks where x is passed by register this will reduce the amount of spills
    - Only if constraints allow it



Greedy Register Allocator

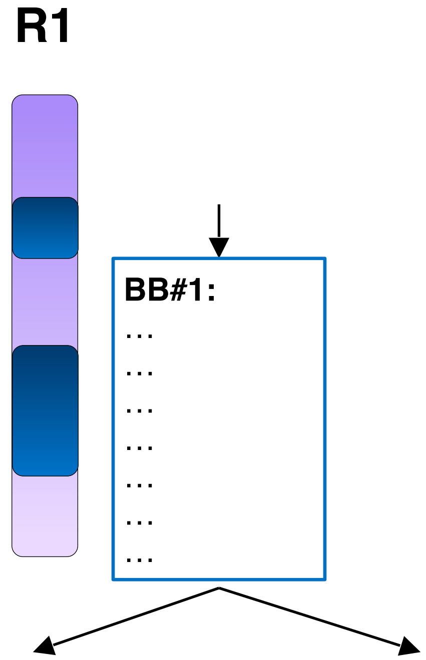
- Greedy Register Allocator Overview
- Region Split
- **Encountered Issues**
- Performance Impact

Region Split Cost Issues

- Inaccurate split cost calculation
  - Root cause of the following encountered issues
- Does not model the affect of local interference caused by the split
  - Makes the split cost inaccurate
  - The “cheapest” split may actually be more expensive than other splits
  - Can choose suboptimal split

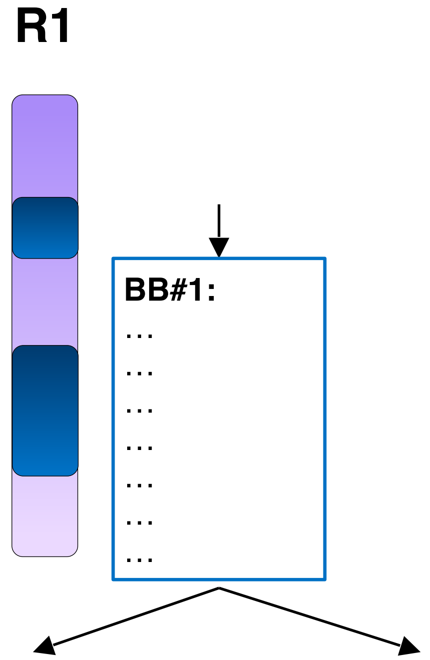
Local Interference

- Find best split of interval x for R1
  - Using Hopfield neural network



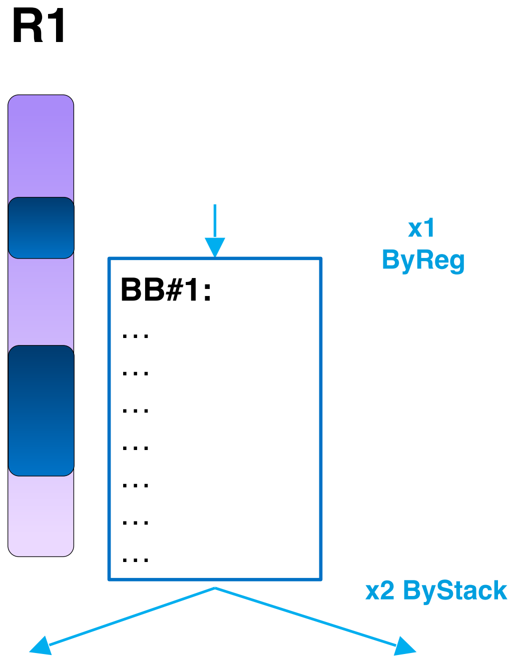
Local Interference

- Find best split of interval x for R1
    - Using Hopfield neural network
    - The network determines how x will be passed on the CFG edges



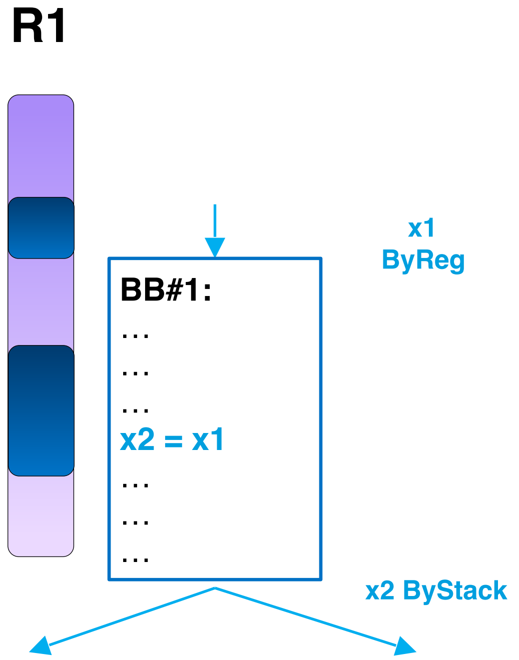
Local Interference

- Find best split of interval x for R1
  - Using Hopfield neural network
  - The network determines how x will be passed on the CFG edges
    - “ByReg” interval or “By stack” interval



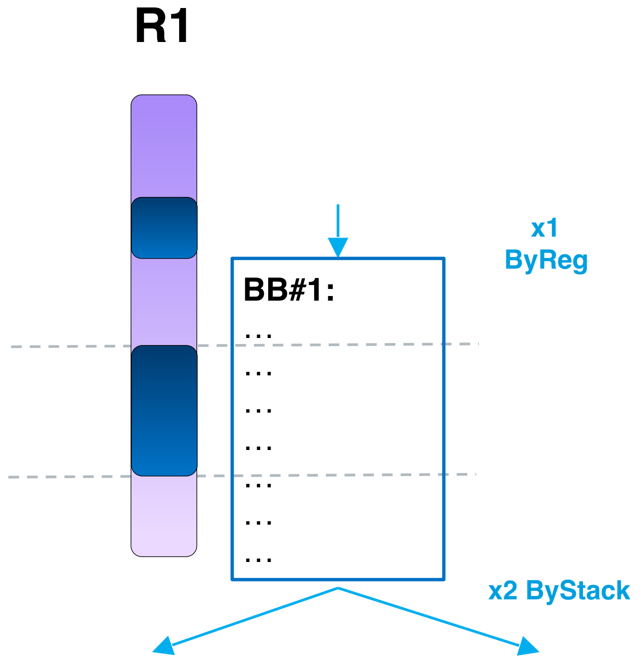
Local Interference

- Find best split of interval x for R1
  - Using Hopfield neural network
  - The network determines how x will be passed on the CFG edges
    - “ByReg” interval or “By stack” interval
    - Determined which basic block will have a copy between these two intervals



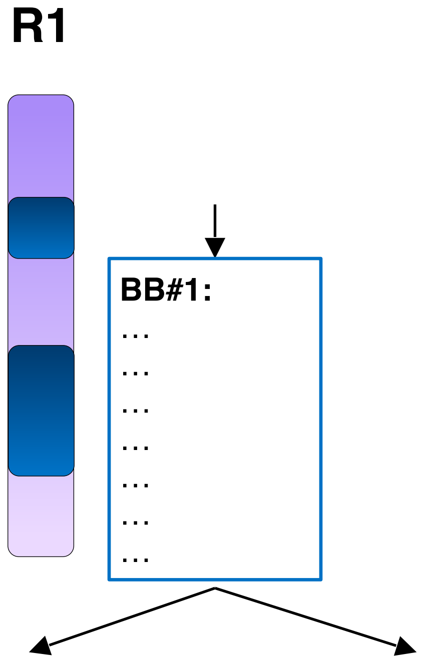
Local Interference

- Find best split of interval x for R1
  - Using Hopfield neural network
  - The network determines how x will be passed on the CFG edges
    - “ByReg” interval or “By stack” interval
    - Determined which basic block will have a copy between these two intervals
  - The Hopfield neural network does not model what happens to x inside the basic block



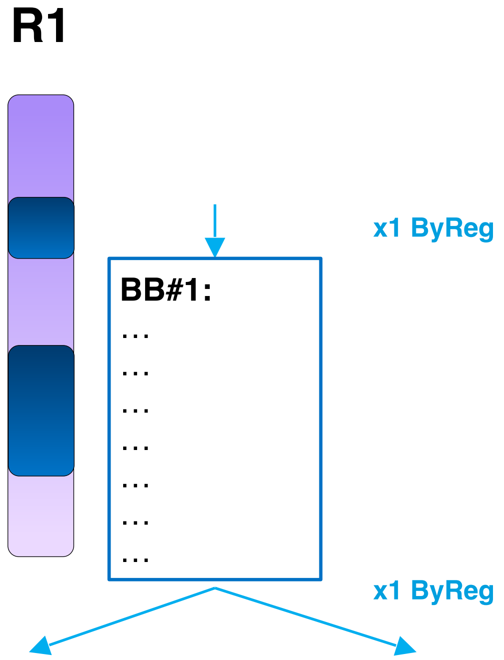
Local Interference

- The Hopfield neural network does not model what happens to x inside the basic block



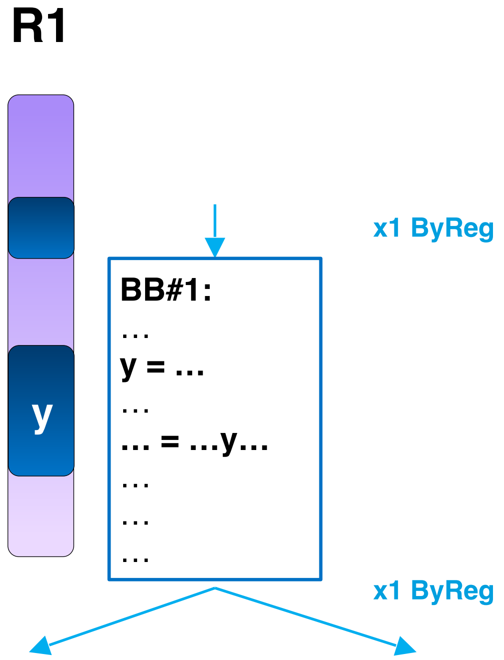
Local Interference

- The Hopfield neural network does not model what happens to x inside the basic block
  - x split for R1 determined x's ByReg interval should enter and leave BB#1



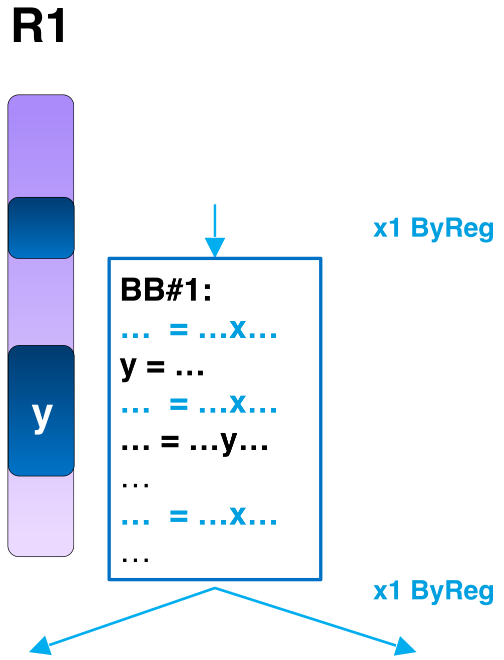
Local Interference

- The Hopfield neural network does not model what happens to x inside the basic block
  - x split for R1 determined x's ByReg interval should enter and leave BB#1
  - y in BB#1 is already assigned to R1



Local Interference

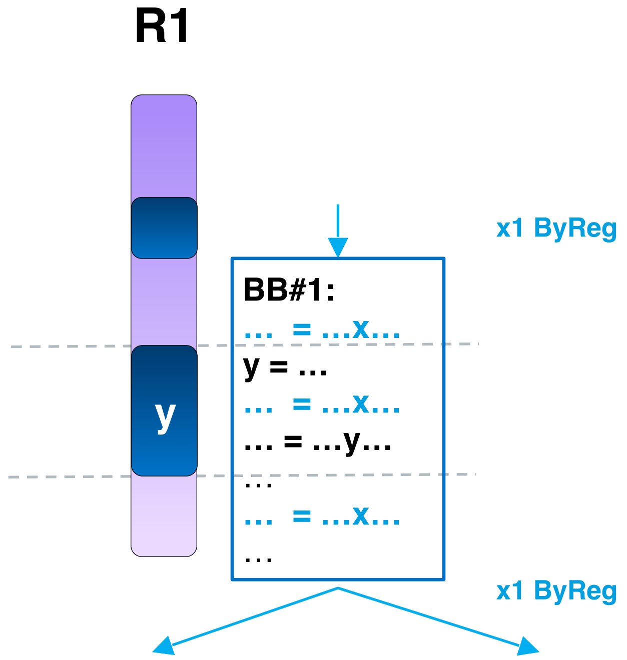
- The Hopfield neural network does not model what happens to x inside the basic block
  - x split for R1 determined x’s ByReg interval should enter and leave BB#1
  - y in BB#1 is already assigned to R1
  - x is used in BB#1



```
BB#1:
… = …x…
y = …
… = …x…
… = …y…
…
… = …x…
…
```

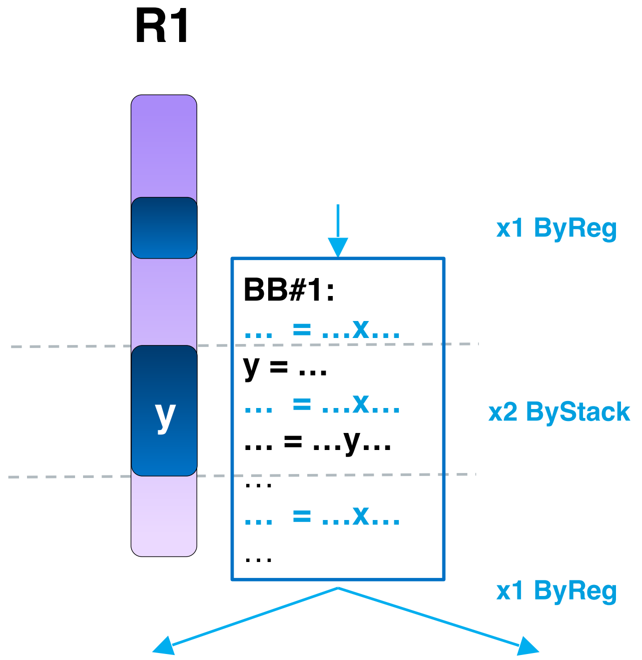
Local Interference

- The Hopfield neural network does not model what happens to x inside the basic block
  - x split for R1 determined x's ByReg interval should enter and leave BB#1
  - y in BB#1 is already assigned to R1
  - x is used in BB#1
  - y interferes with assigning x to R1 locally in BB#1



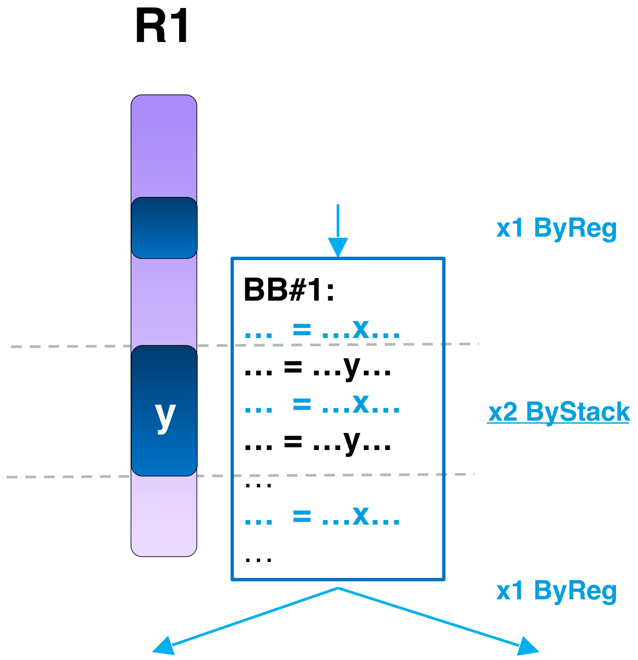
Local Interference

- The Hopfield neural network does not model what happens to x inside the basic block
  - x split for R1 determined x's ByReg interval should enter and leave BB#1
  - y in BB#1 is already assigned to R1
  - x is used in BB#1
  - y interferes with assigning x to R1 locally in BB#1
    - The part of x that contains this local interference will be added to x's "ByStack" split artifact



Local Interference

- Local interferences may have very negative affects on assignment of the "ByStack" split artifact
  - Can cause bad eviction chains
    - Encountered issues #1, #2
  - Can cause a lot of reloads
    - Encountered issue #3
- This affect is not considered during split cost calculation

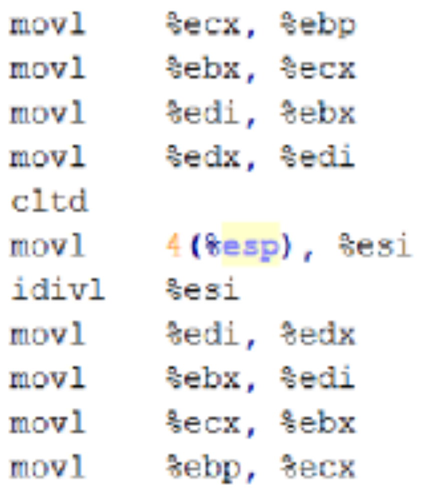


Encountered Issue

- Bad eviction chain
  - Cyclic eviction/split chain – Issue #1
  - Domino effect eviction – Issue #2
- Multiple reloads from the same location
  - Issue #3

Encountered Issue #1

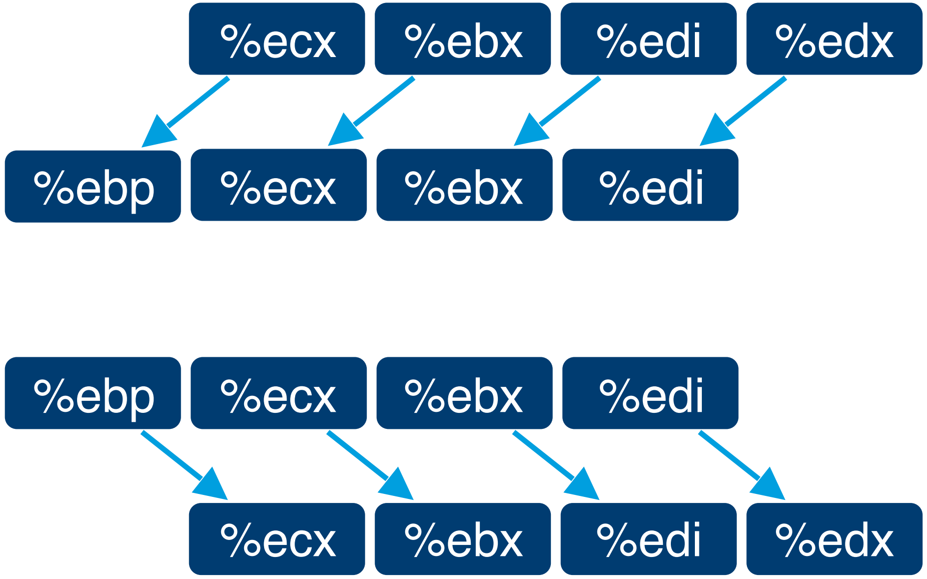
- Bad eviction chain – scenario 1
  - llvm/test/CodeGen/X86/greedy_regalloc_bad_eviction_sequence.ll



```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

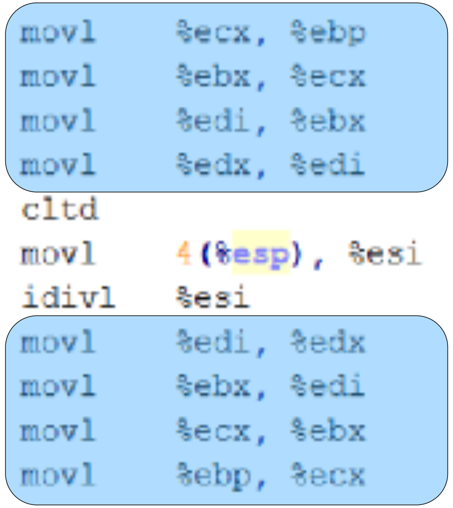
- Bad eviction chain – scenario 1
  - llvm/test/CodeGen/X86/greedy_regalloc_bad_eviction_sequence.ll



```
movl     %ecx, %ebp
movl     %ebx, %ecx
movl     %edi, %ebx
movl     %edx, %edi
cltd
movl     4(%esp), %esi
idivl    %esi
movl     %edi, %edx
movl     %ebx, %edi
movl     %ecx, %ebx
movl     %ebp, %ecx
```

Encountered Issue #1

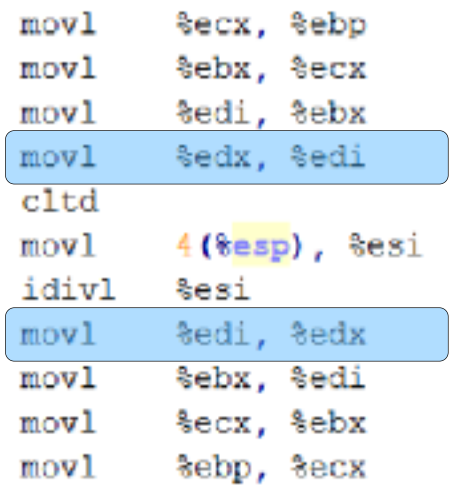
- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain



```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

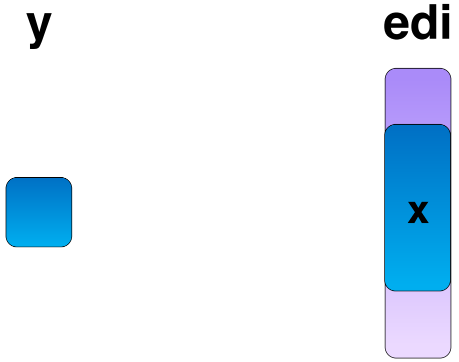
- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain

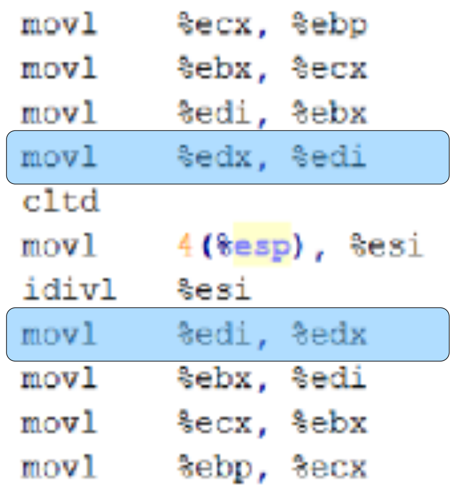


```
movl     %ecx, %ebp
movl     %ebx, %ecx
movl     %edi, %ebx
movl     %edx, %edi
cltd
movl     4(%esp), %esi
idivl    %esi
movl     %edi, %edx
movl     %ebx, %edi
movl     %ecx, %ebx
movl     %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi

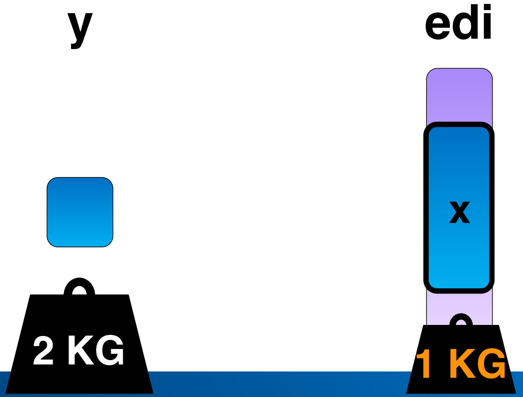


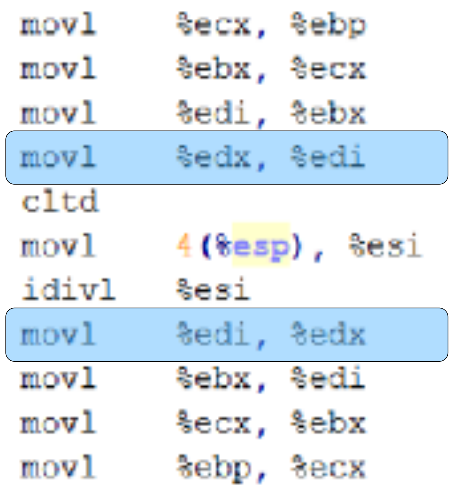


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi

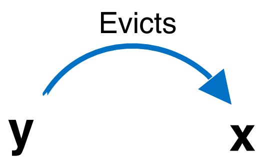


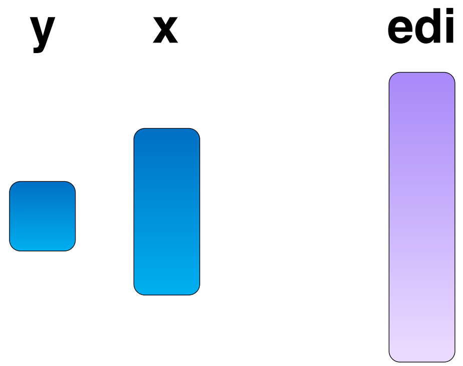


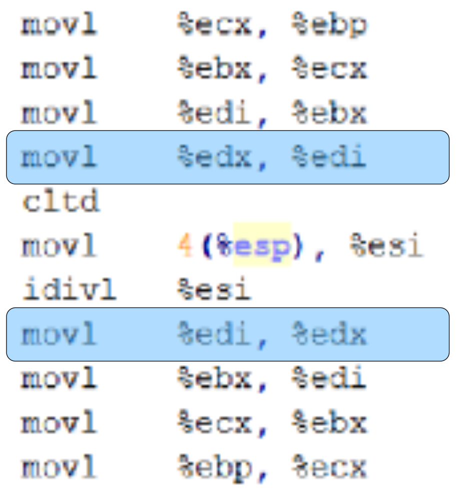
```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi



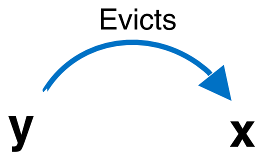


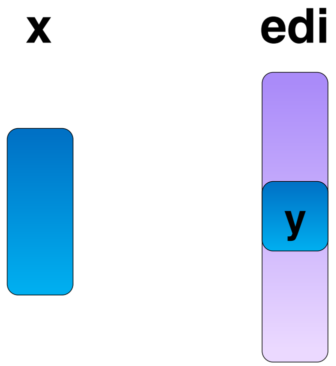


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi

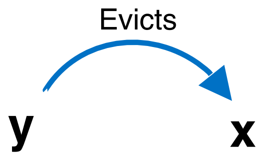


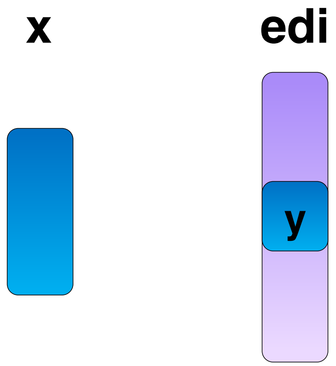


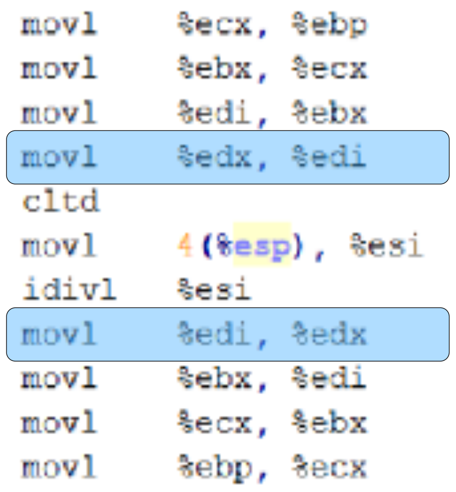
```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2



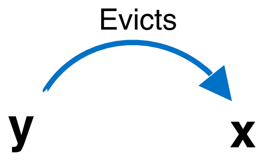


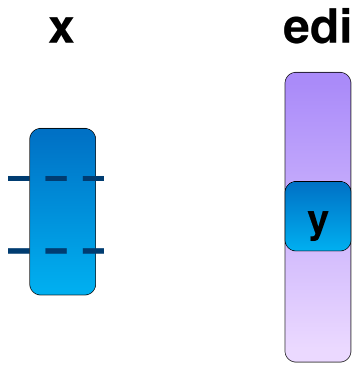


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2

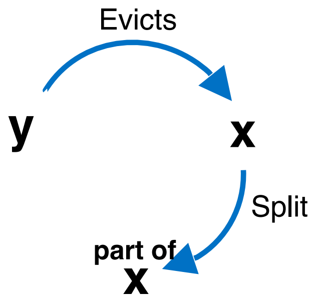


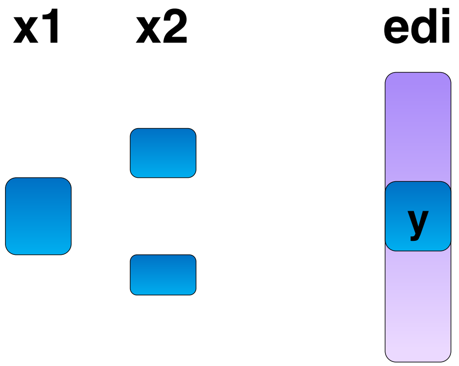


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2

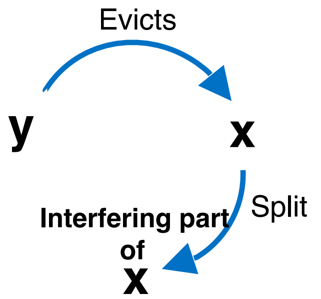


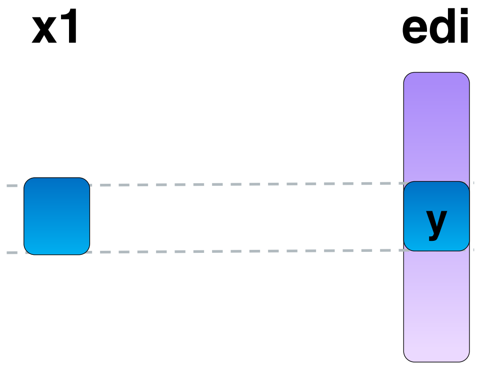


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2
      - x1 represent the part of the split that has local interference with y

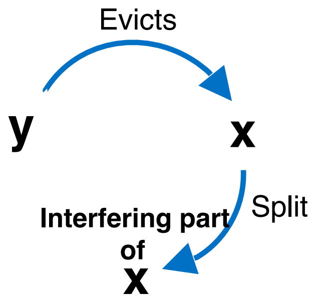


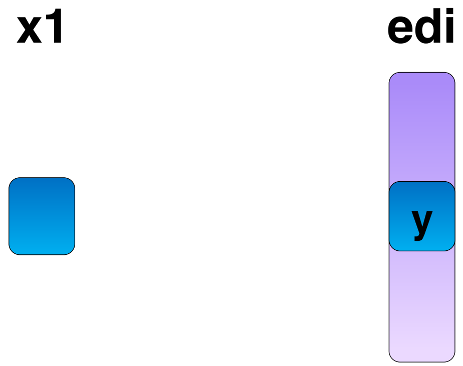


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2
      - x1 represent the part of the split that has local interference with y

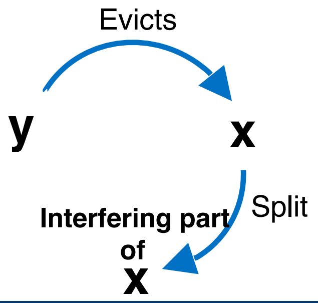


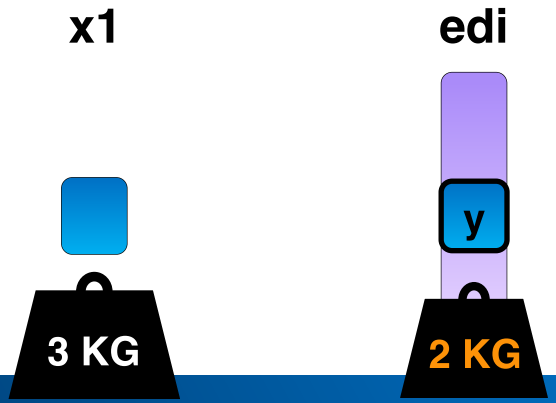


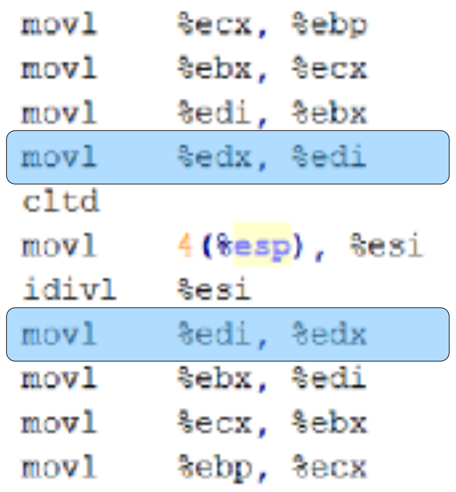
```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2
      - x1 represent the part of the split that has local interference with y
    - **x1 evicts y** from edi



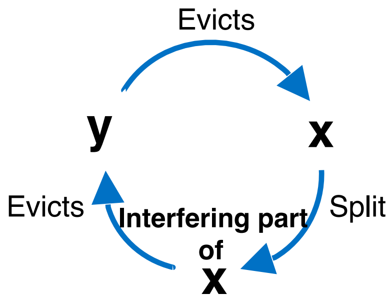


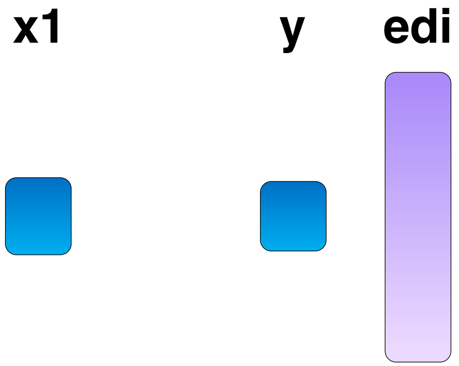


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2
      - x1 represent the part of the split that has local interference with y
    - **x1 evicts y** from edi

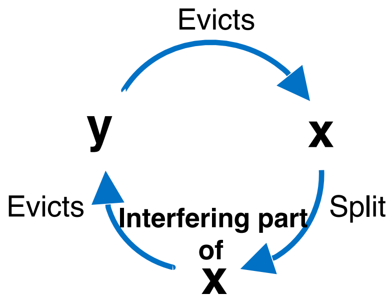


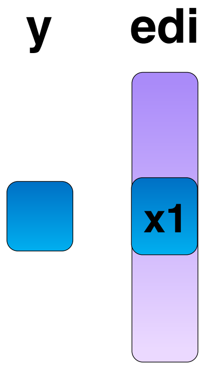


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - **y evicts x** from edi
    - x is split into x1 and x2
      - x1 represent the part of the split that has local interference with y
    - **x1 evicts y** from edi

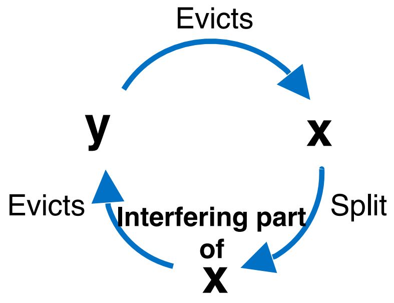


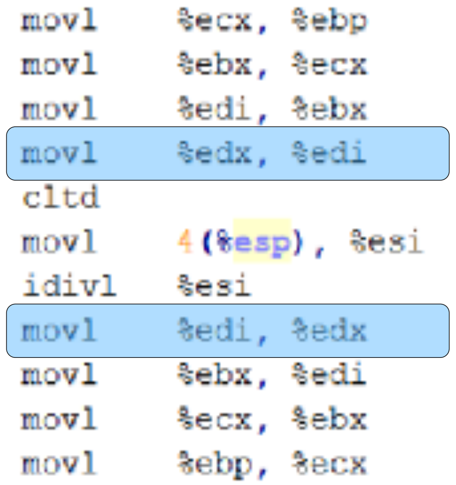


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

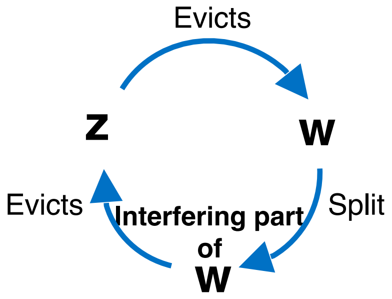
- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - Every such “movl” duo was created by cyclic eviction/split chain

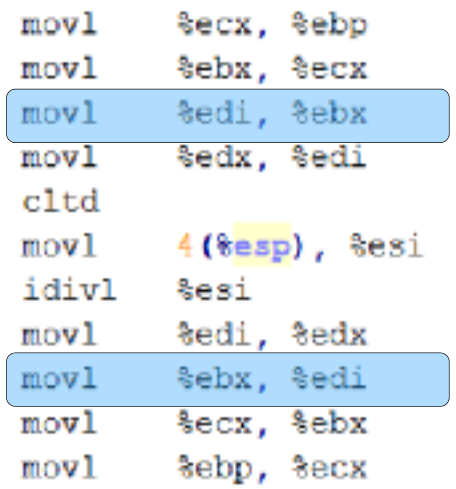



```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - Every such “movl” duo was created by cyclic eviction/split chain

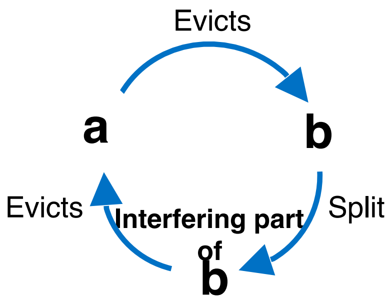


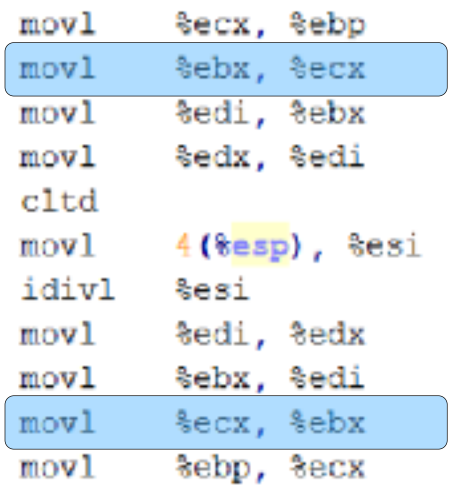


```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - Cyclic eviction/split chain
    - Every such “movl” duo was created by cyclic eviction/split chain

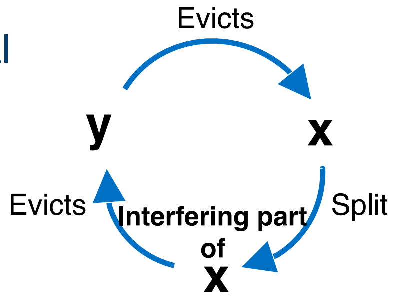


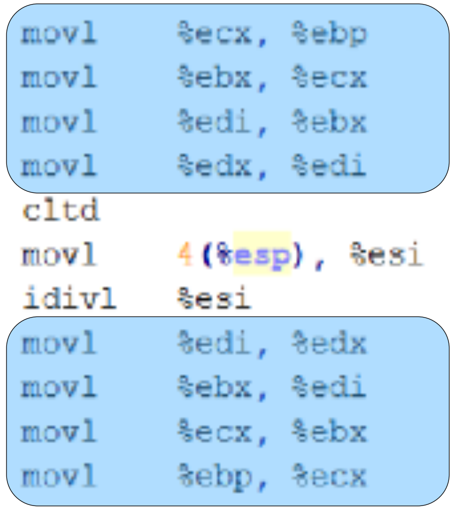


```
movl      %ecx, %ebp
movl      %ebx, %ecx
movl      %edi, %ebx
movl      %edx, %edi
cltd
movl      4(%esp), %esi
idivl     %esi
movl      %edi, %edx
movl      %ebx, %edi
movl      %ecx, %ebx
movl      %ebp, %ecx
```

Encountered Issue #1

- Bad eviction chain – scenario 1
  - The problem
    - x is split in such a way that creates local interference split artifact
    - That artifact causes cyclic eviction
  - The solution
    - Tailored for this case
      - Identify if a split will create a local interference artifact
      - Identify if that split artifact will cause a cyclic eviction
      - Increase split cost
        - Make this split le 20$ ttractive compared to other splits
    - Commit: https://reviews.llvm.org/rL316295

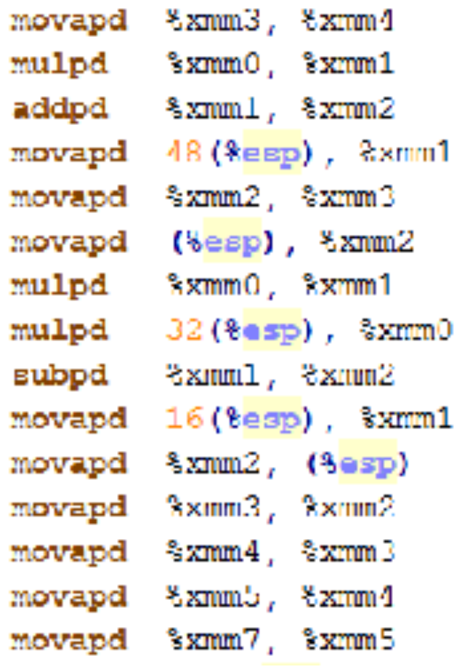




```
movl    %ecx, %ebp
movl    %ebx, %ecx
movl    %edi, %ebx
movl    %edx, %edi
cltd
movl    4(%esp), %esi
idivl   %esi
movl    %edi, %edx
movl    %ebx, %edi
movl    %ecx, %ebx
movl    %ebp, %ecx
```

Encountered Issue #2

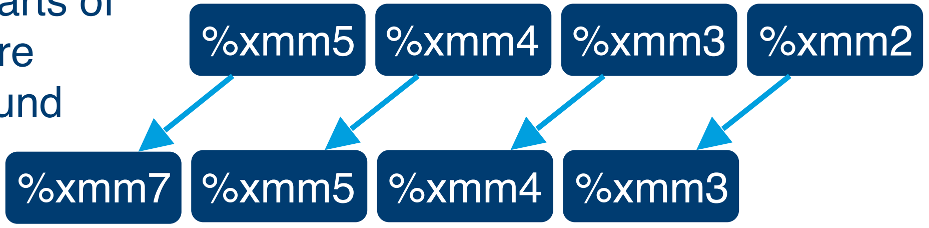
- Bad eviction chain – scenario 2
  - https://bugs.llvm.org/show_bug.cgi?id=26810

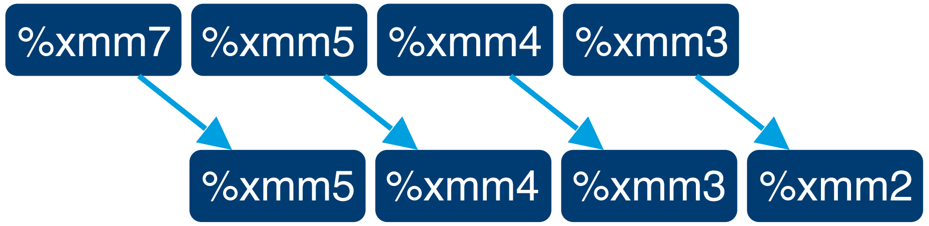


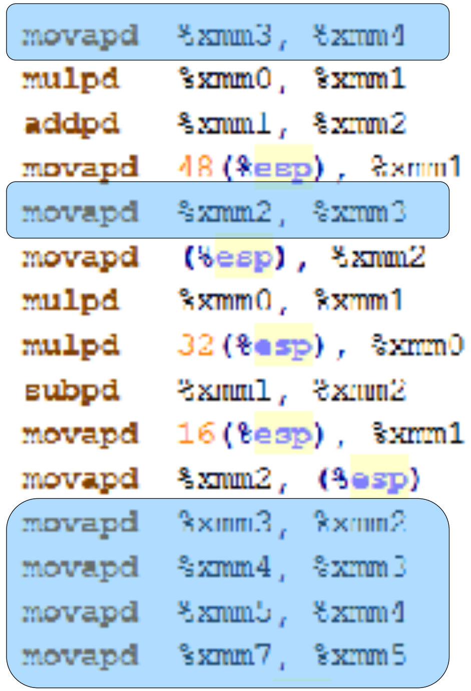
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - https://bugs.llvm.org/show_bug.cgi?id=26810
  - This time parts of the chain are spread around



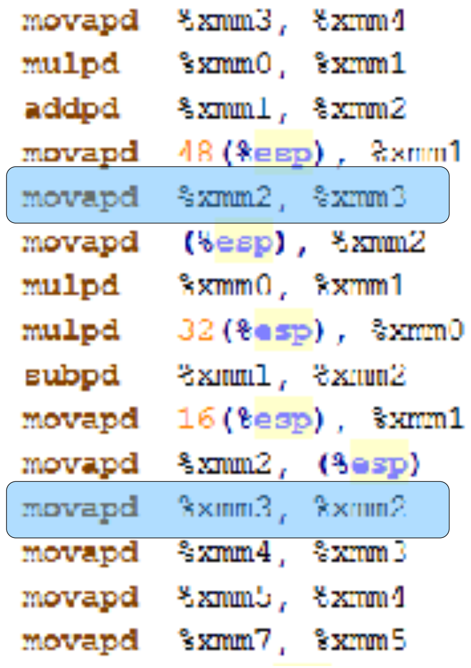




```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

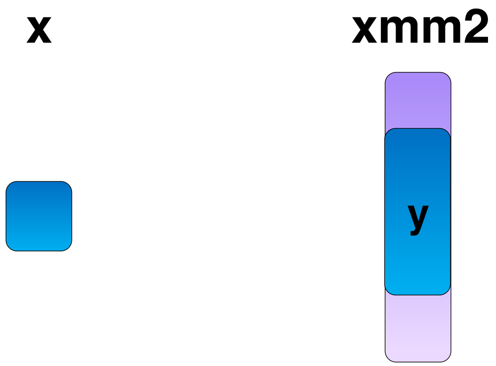
- Bad eviction chain – scenario 2
  - Domino effect eviction

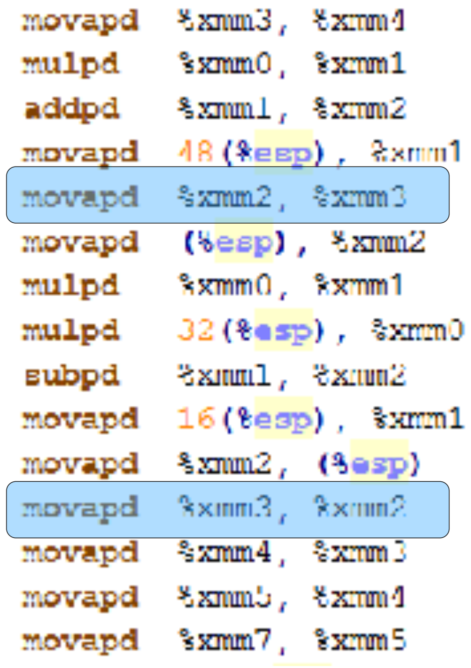


```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2

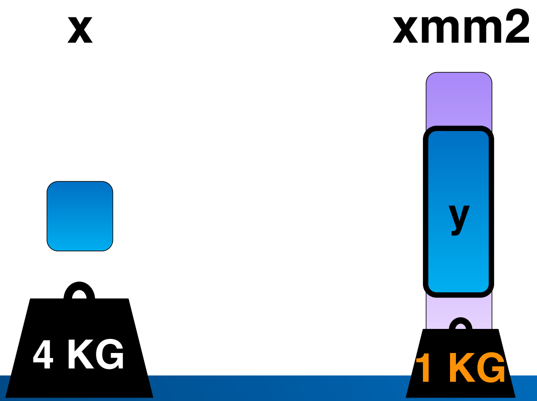


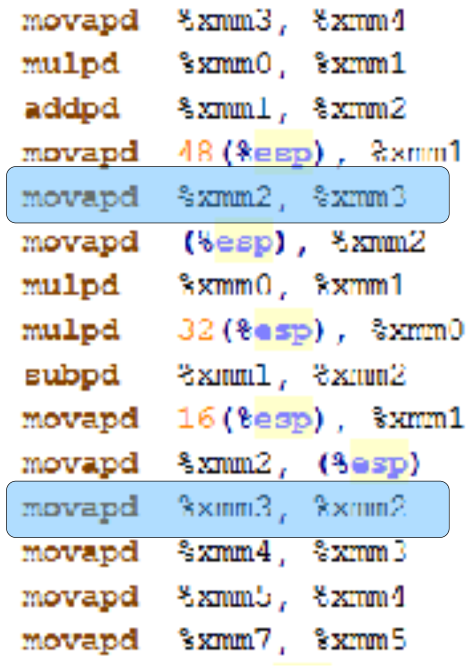


```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2

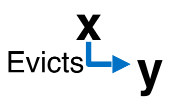


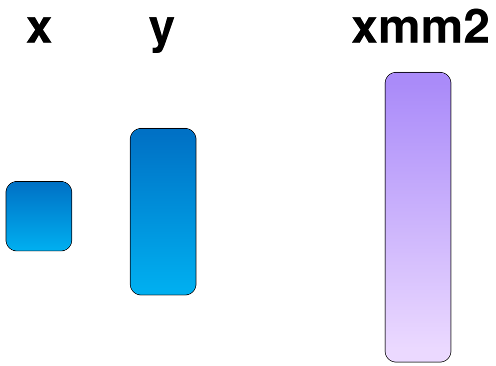


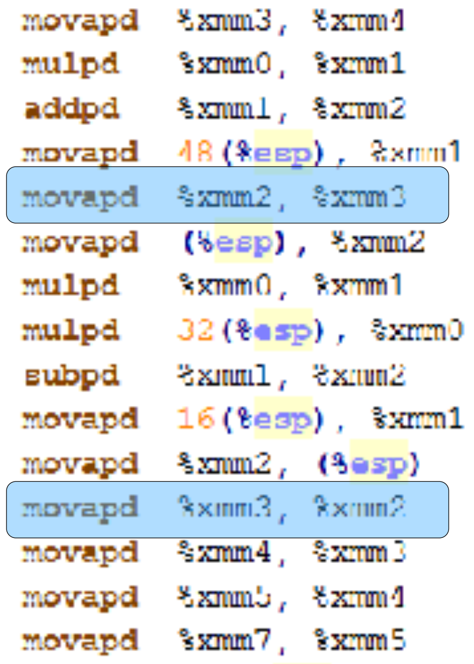
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2



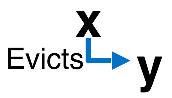


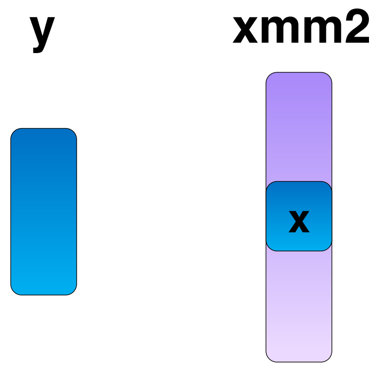


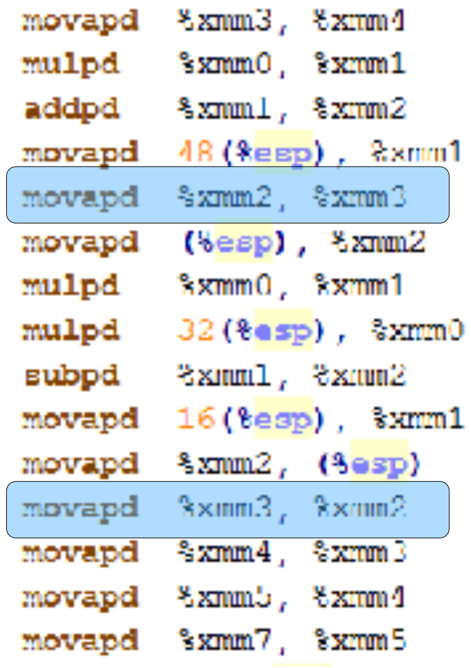
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2



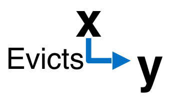


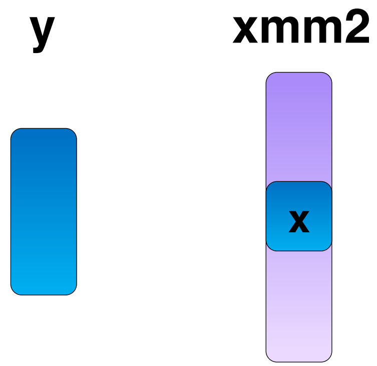


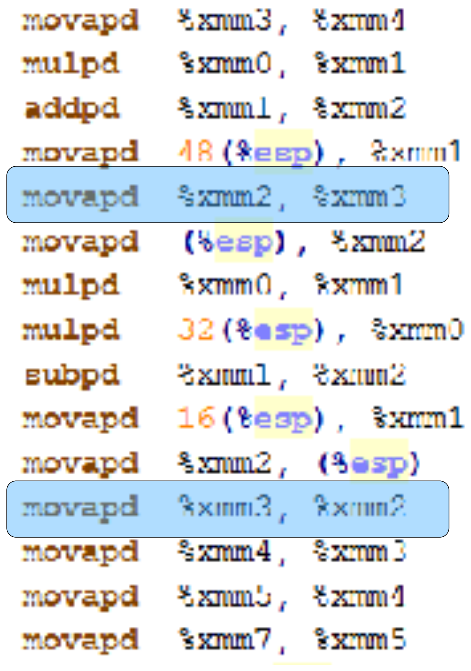
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2

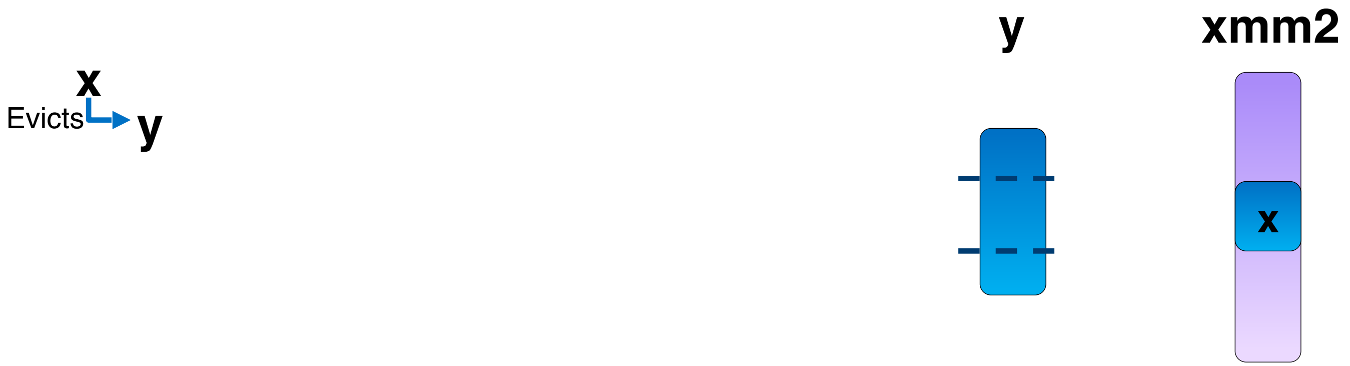


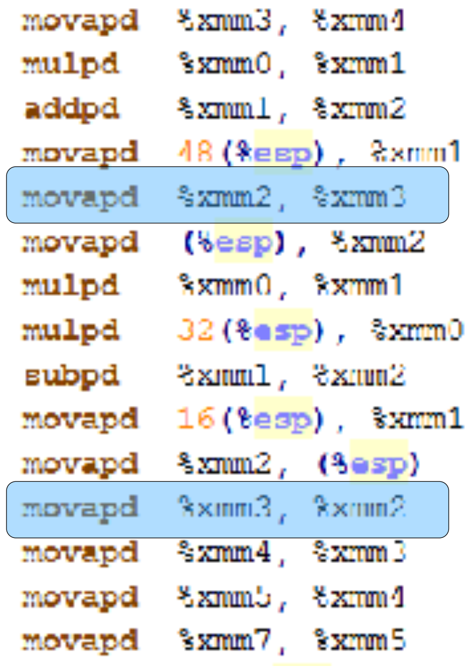



```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2

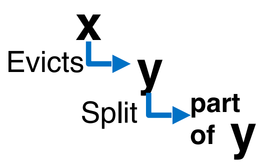


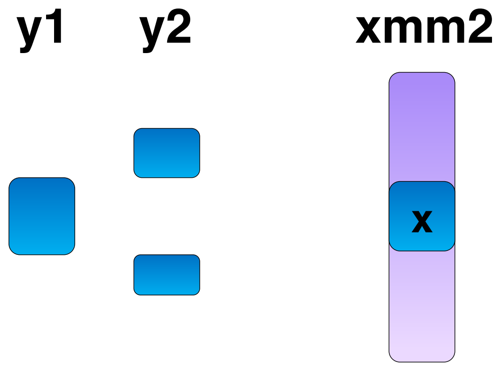


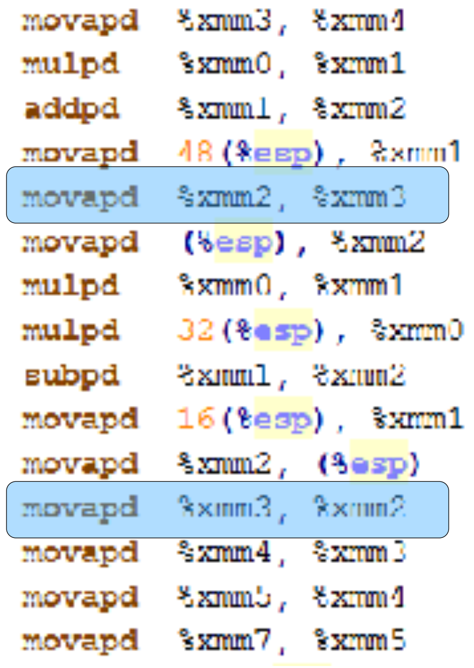
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2



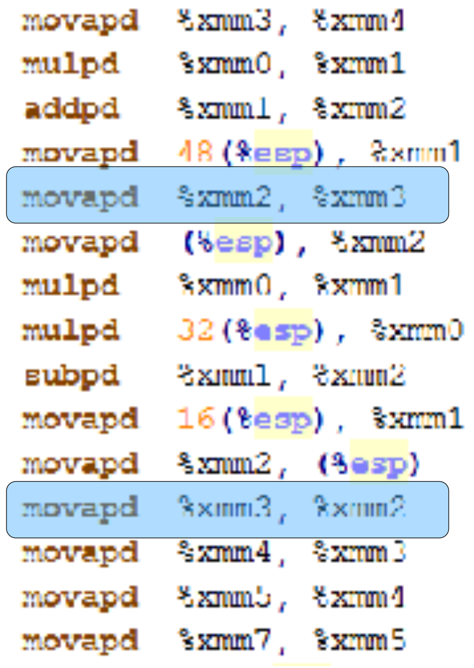




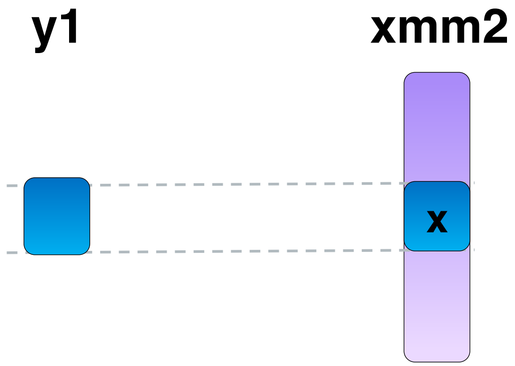
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

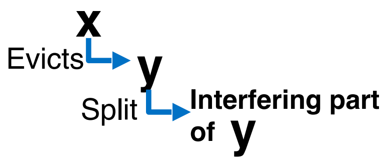
Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x



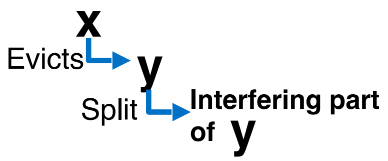
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

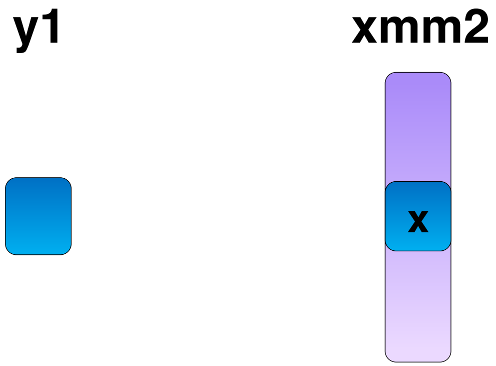


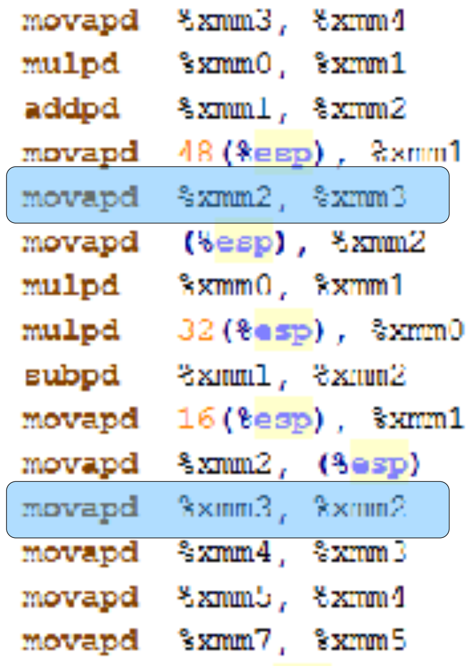


Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x

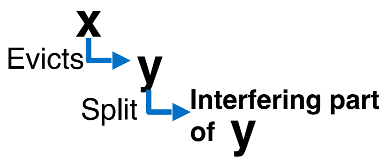


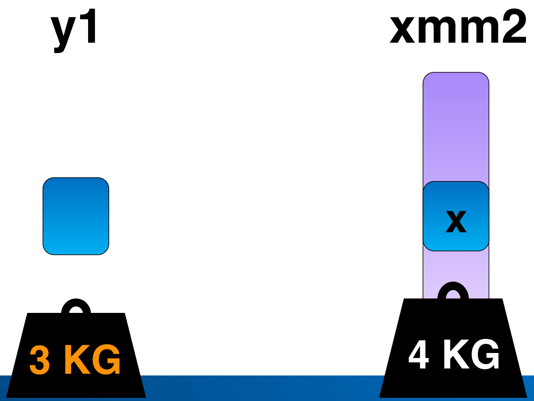


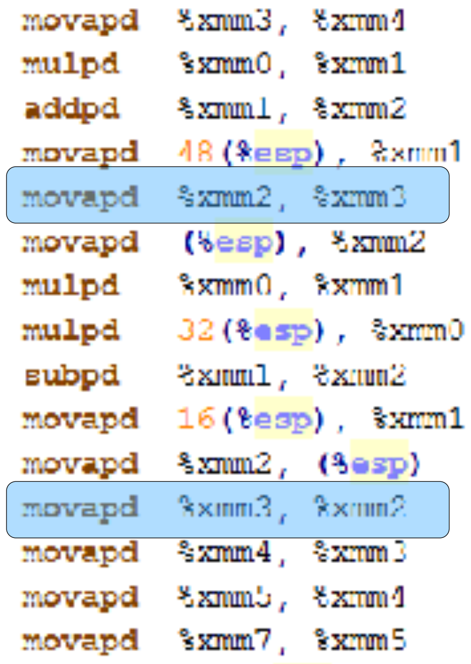

```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2



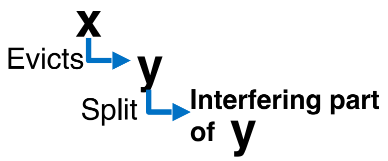


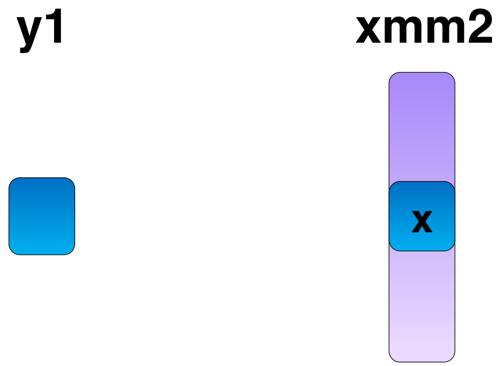


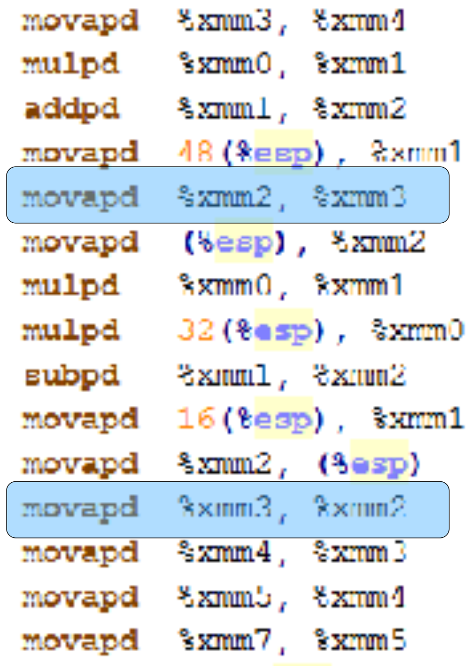
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2



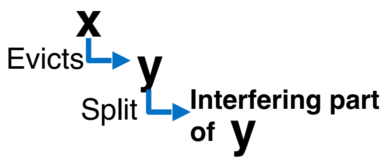


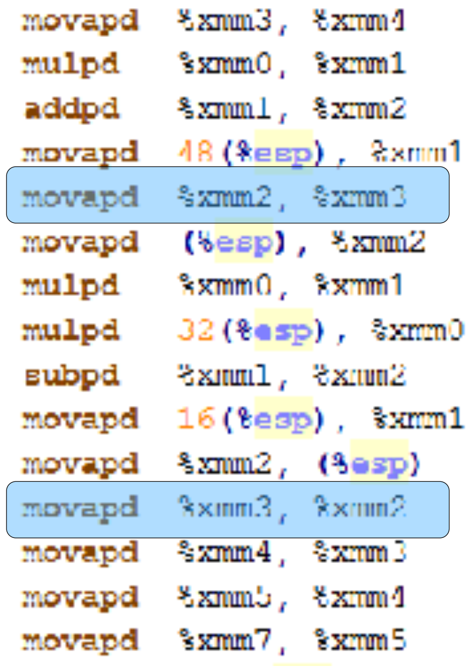


```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2

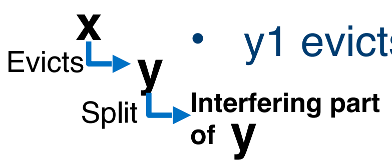


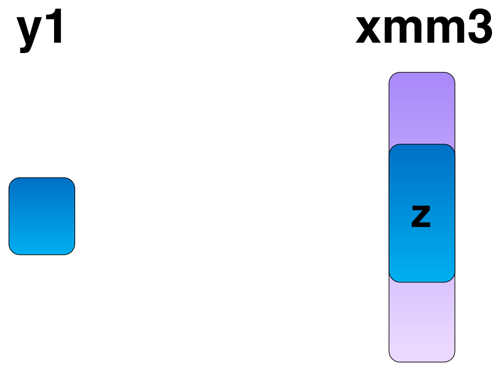


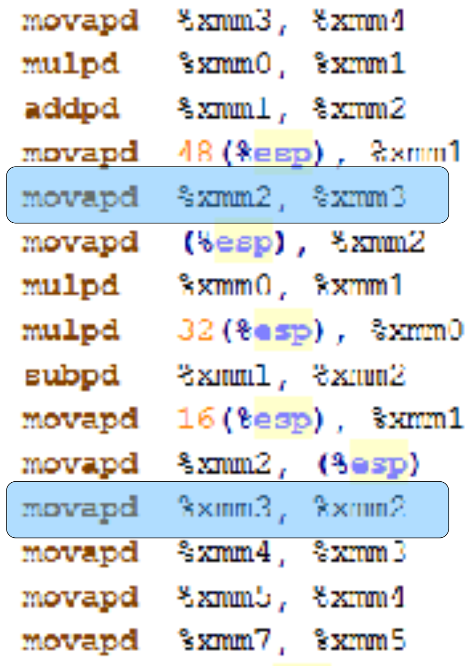
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2
    - y1 evicts z from xmm3



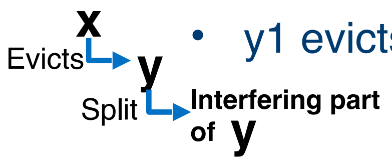


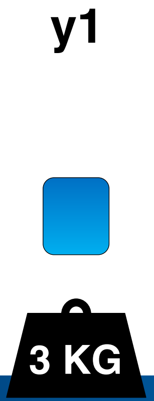


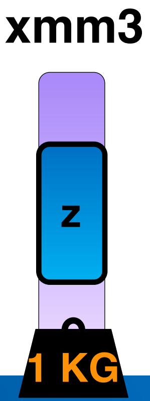
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

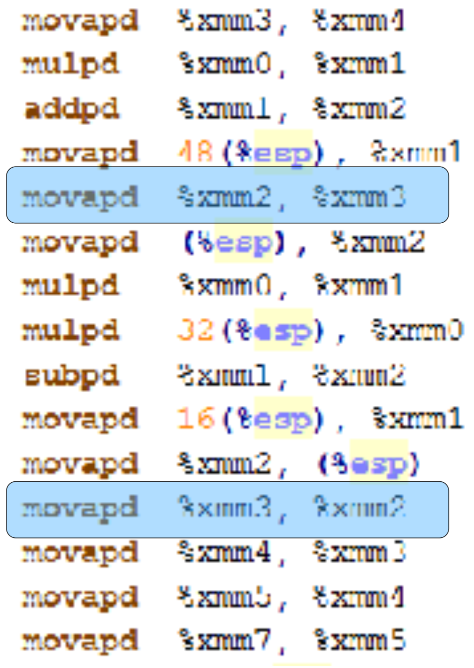
Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2
    - y1 evicts z from xmm3





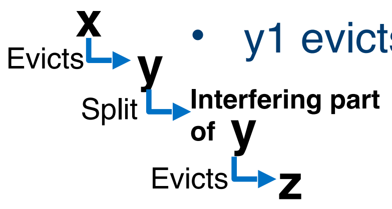


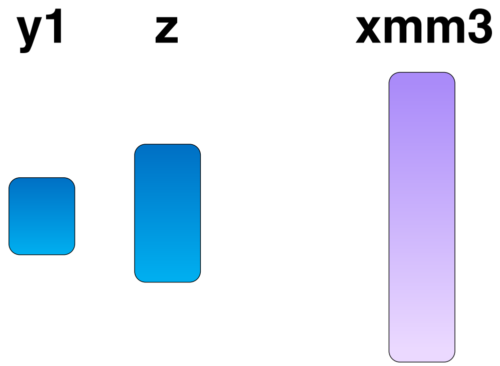


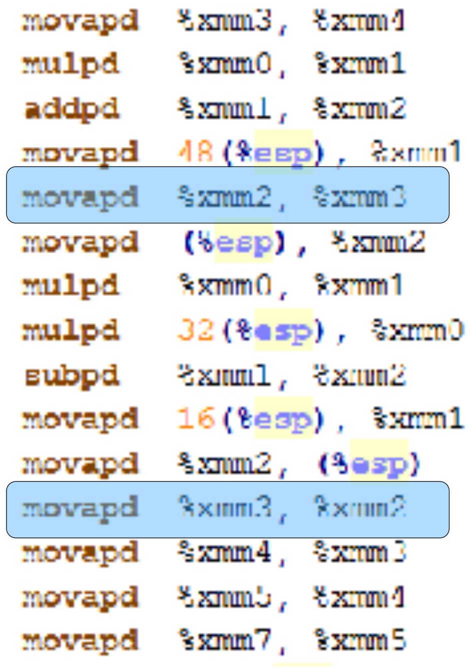
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2
    - y1 evicts z from xmm3



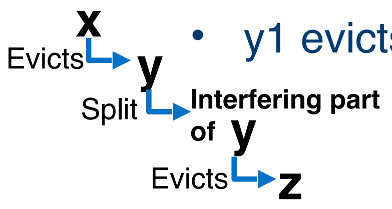


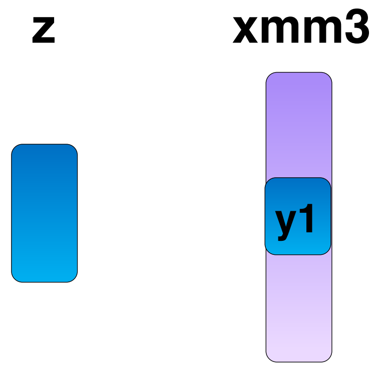


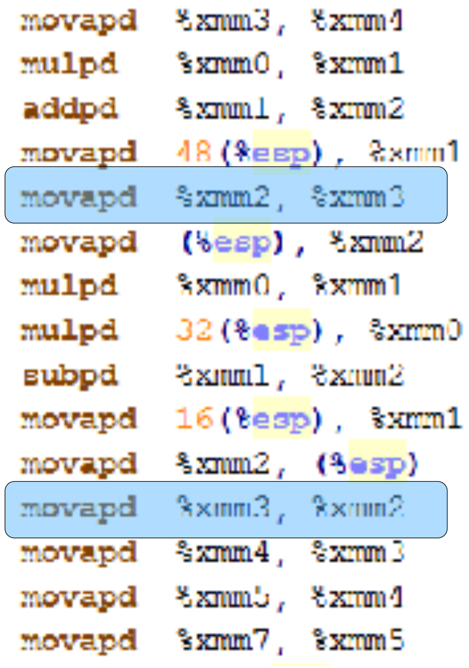
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - x evicts y from xmm2
    - y is split into y1 and y2 for xmm2
      - y1 represent the part of the split that has local interference with x
    - y1 cannot evict x from xmm2
    - y1 evicts z from xmm3

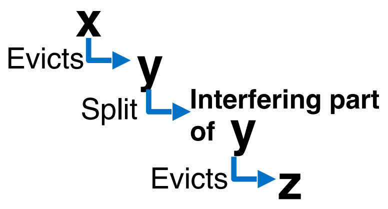


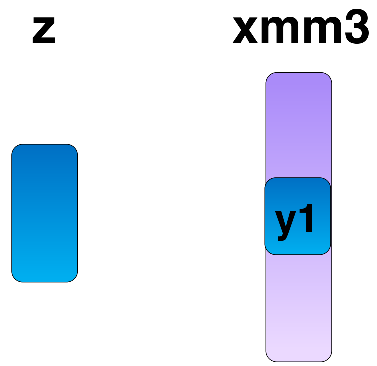


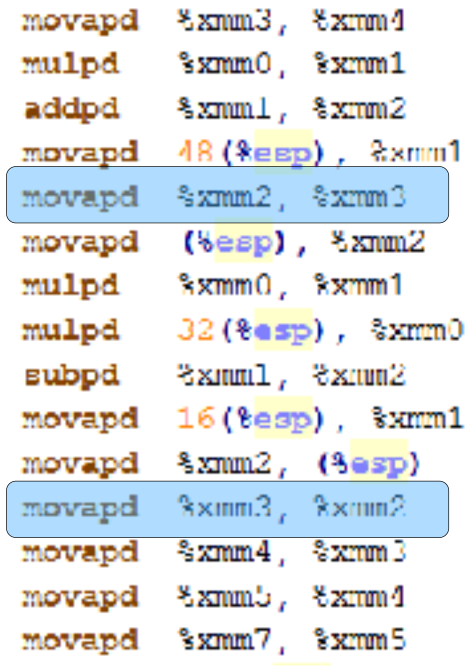

```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3



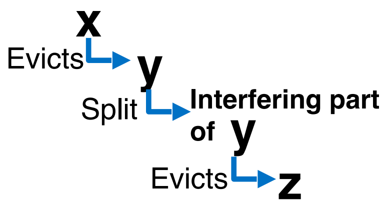


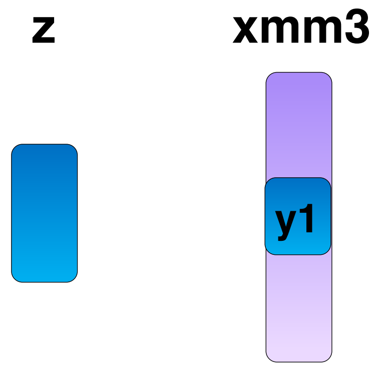


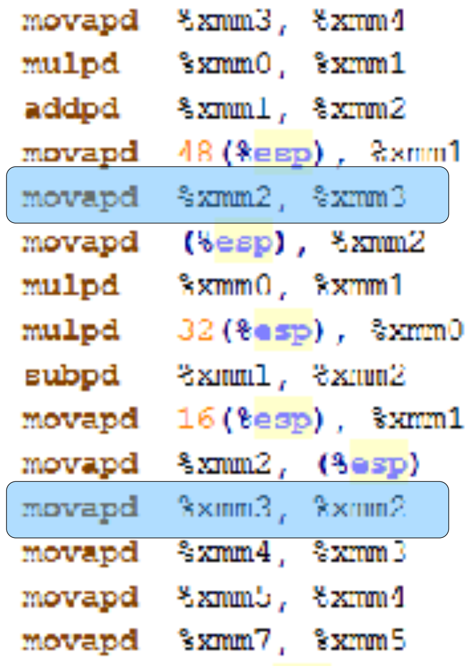
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3



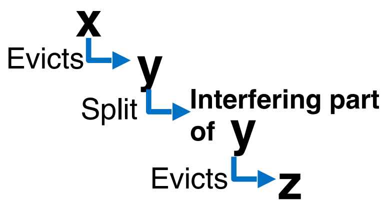


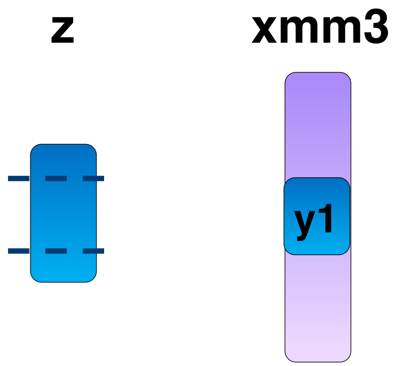


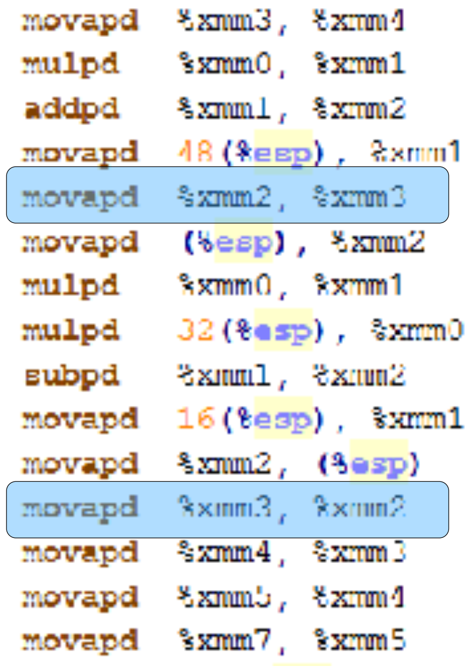
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3



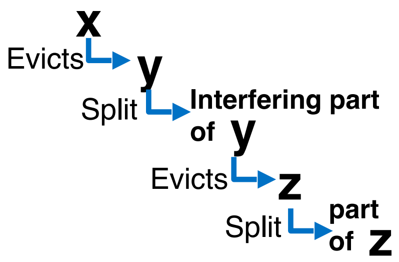


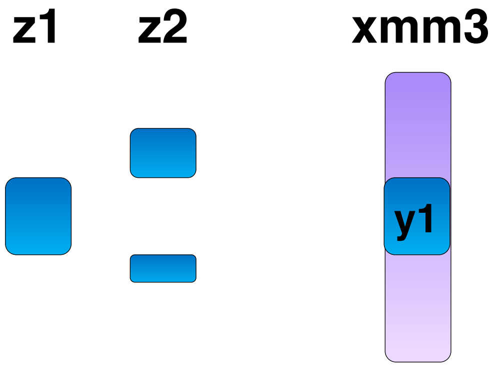


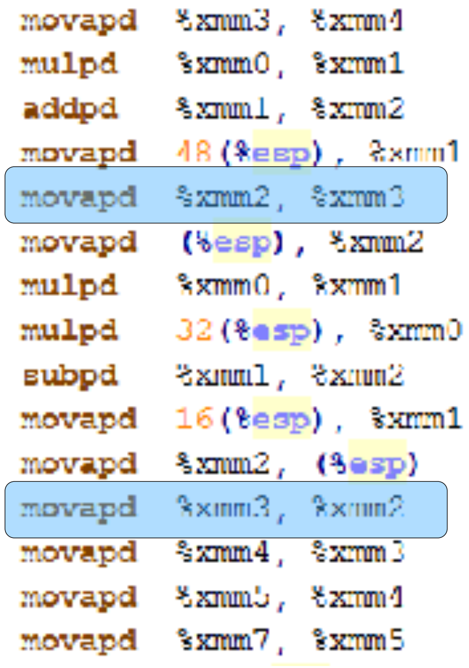
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3



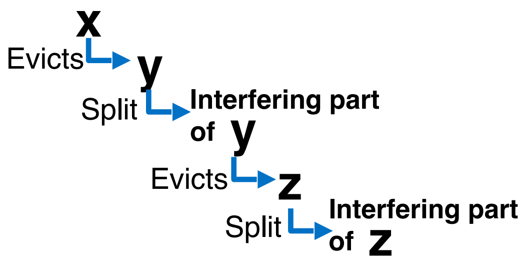


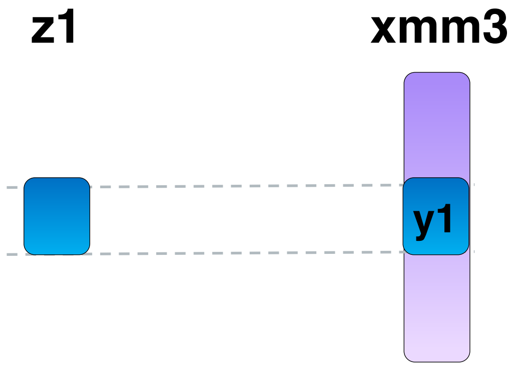


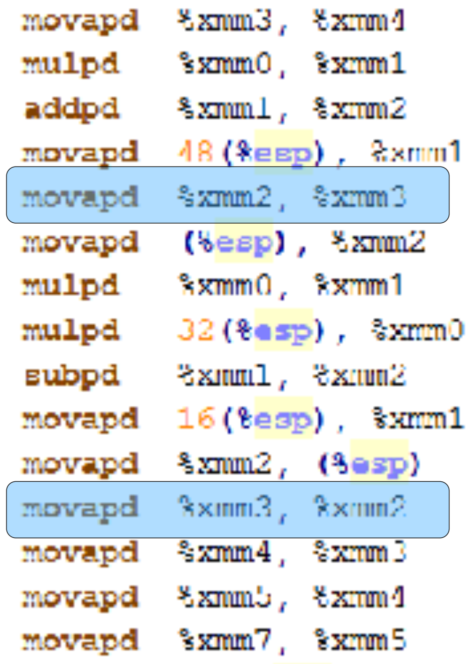
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1



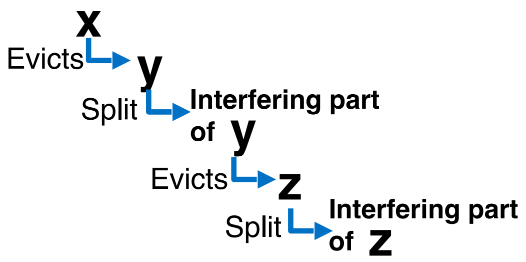


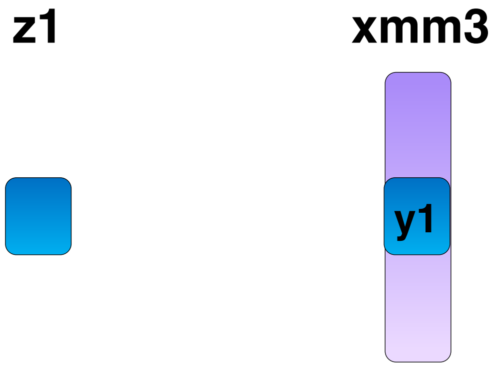


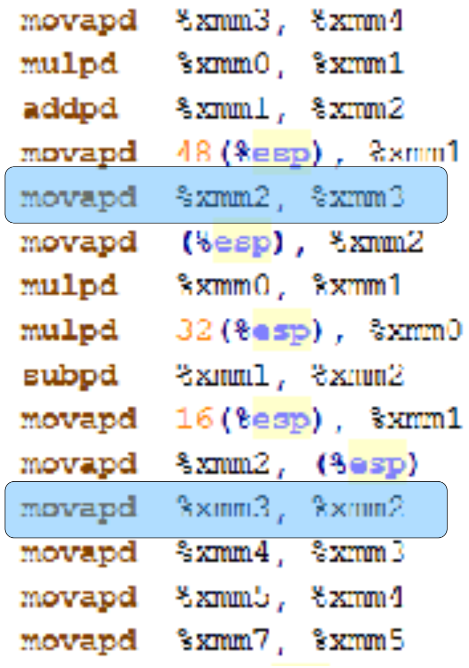
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1



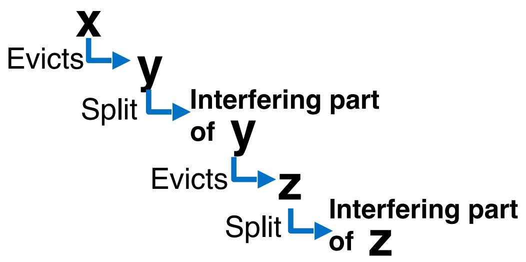


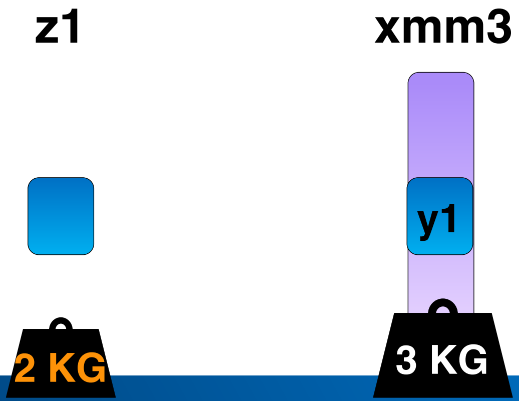


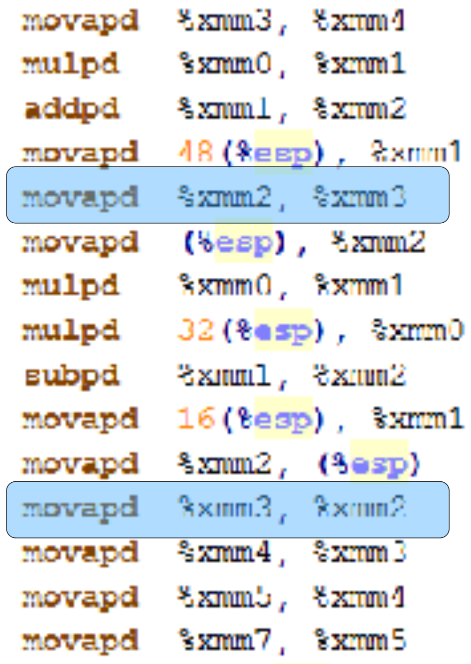
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3



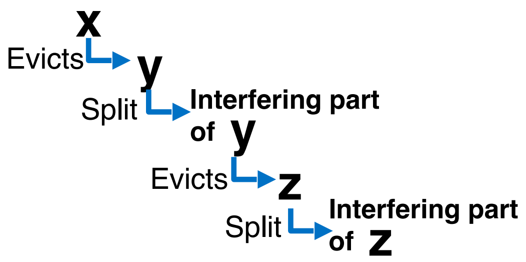


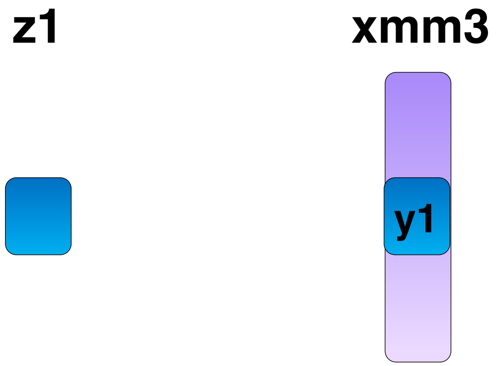


```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3



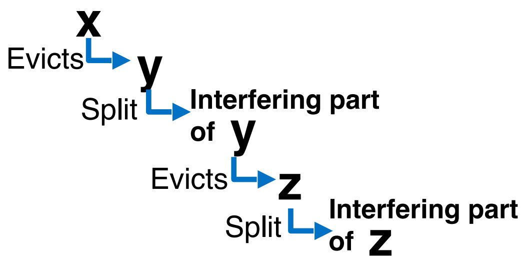


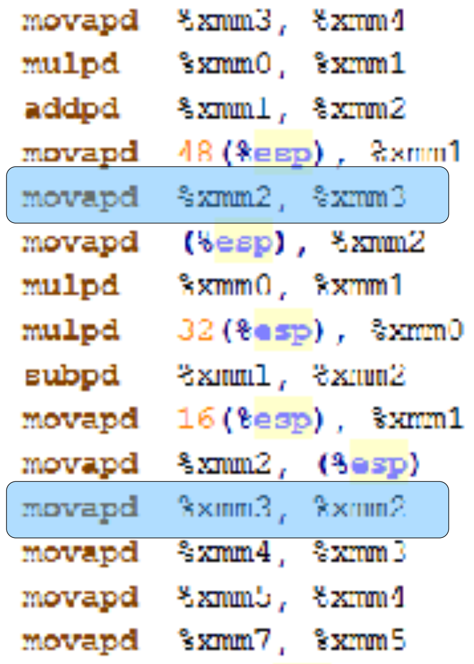
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3

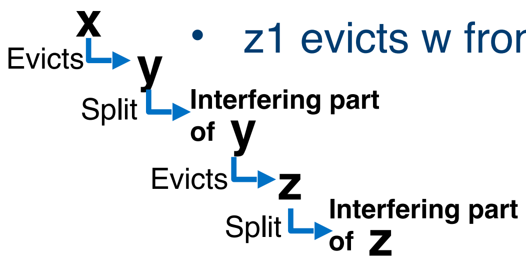
z1

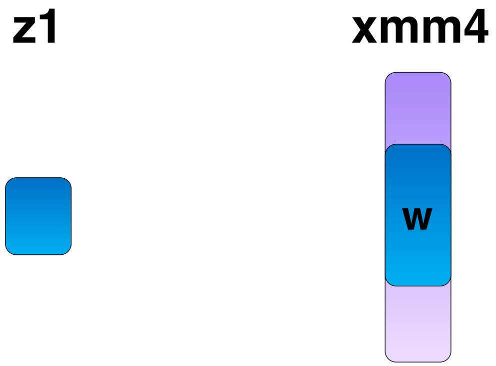



```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3
    - z1 evicts w from xmm4

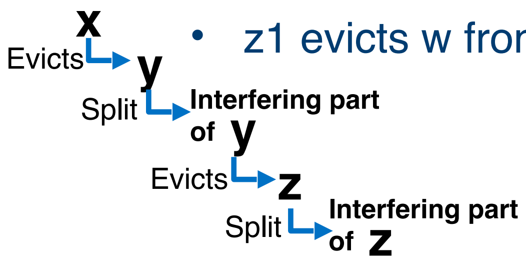


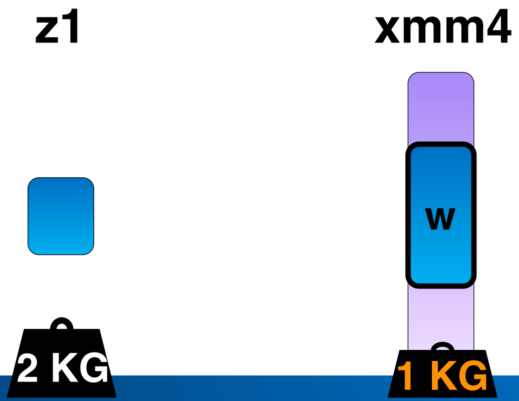


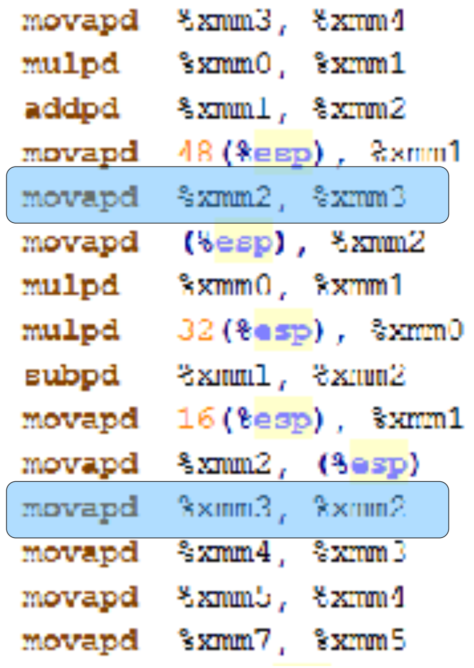
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3
    - z1 evicts w from xmm4



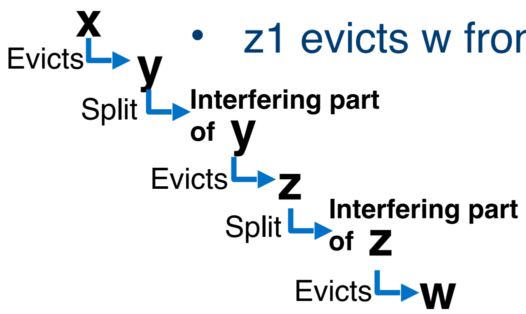


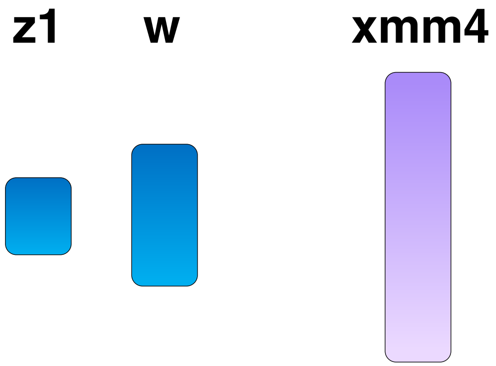


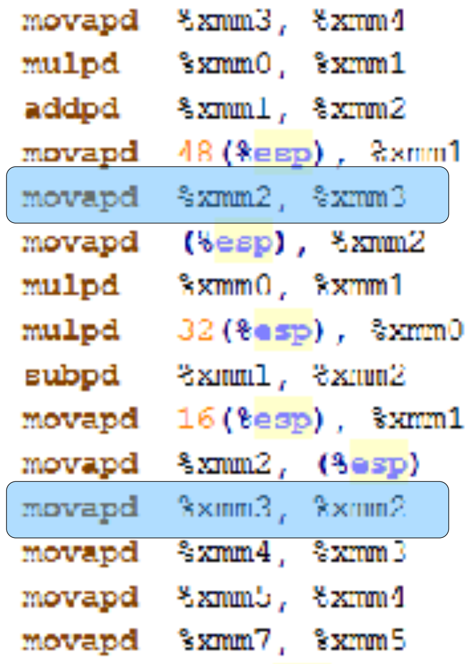
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3
    - z1 evicts w from xmm4





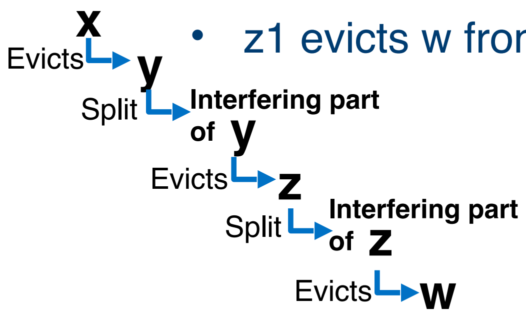


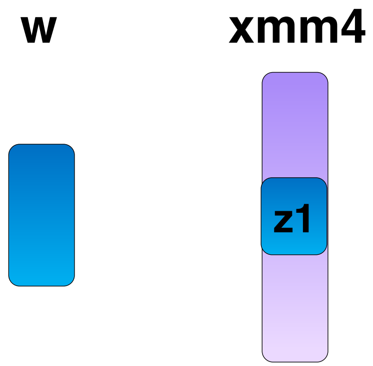
```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - y1 evicts z from xmm3
    - z is split into z1 and z2 for xmm3
      - z1 represent the part of the split that has local interference with y1
    - z1 cannot evict y1 from xmm3
    - z1 evicts w from xmm4

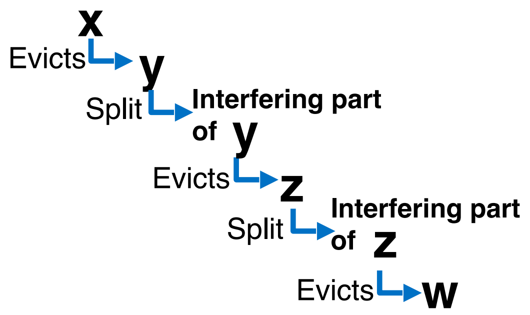
```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

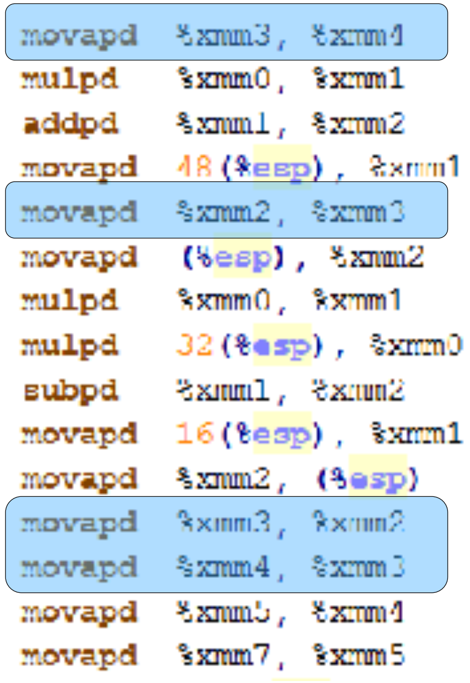




Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - Every such “movl” duo was created by the domino effect

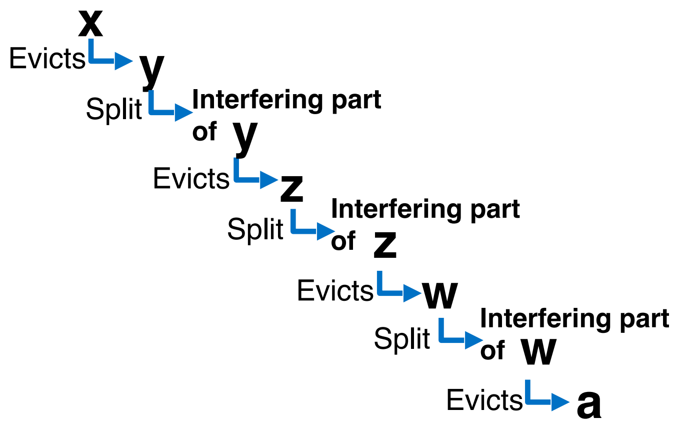


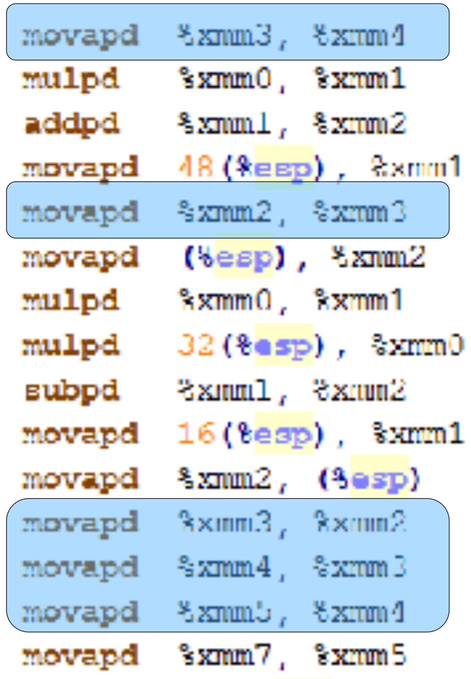


```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - Every such “movl” duo was created by the domino effect

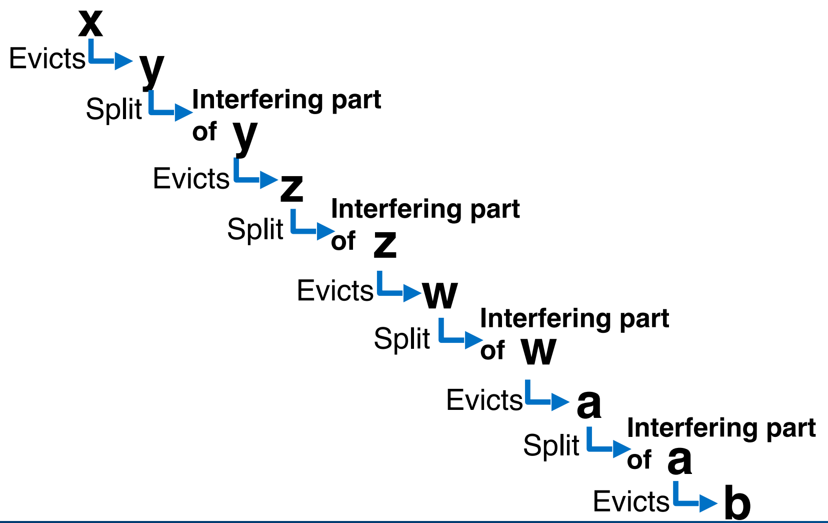


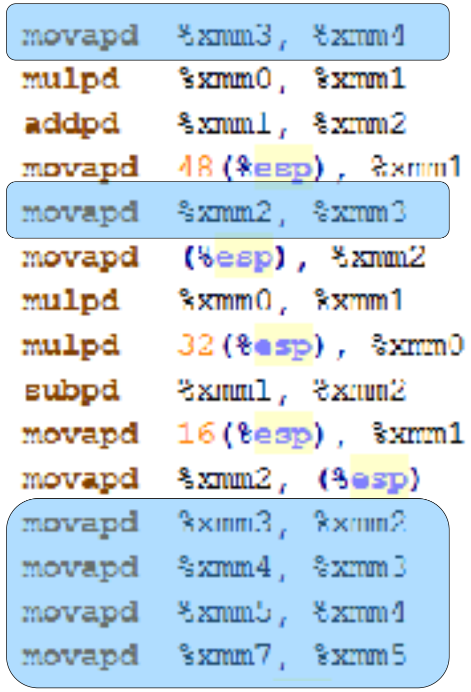


```
movapd   %xmm3, %xmm4
mulpd    %xmm0, %xmm1
addpd    %xmm1, %xmm2
movapd   48(%esp), %xmm1
movapd   %xmm2, %xmm3
movapd   (%esp), %xmm2
mulpd    %xmm0, %xmm1
mulpd    32(%esp), %xmm0
subpd    %xmm1, %xmm2
movapd   16(%esp), %xmm1
movapd   %xmm2, (%esp)
movapd   %xmm3, %xmm2
movapd   %xmm4, %xmm3
movapd   %xmm5, %xmm4
movapd   %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - Domino effect eviction
    - Every such “movl” duo was created by the domino effect

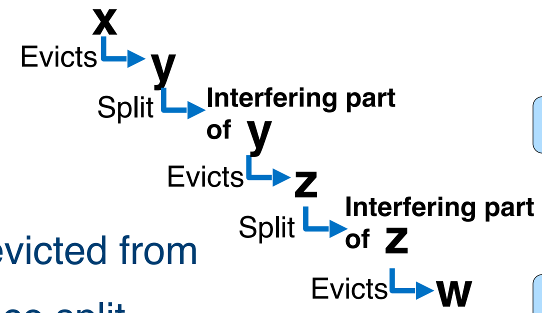


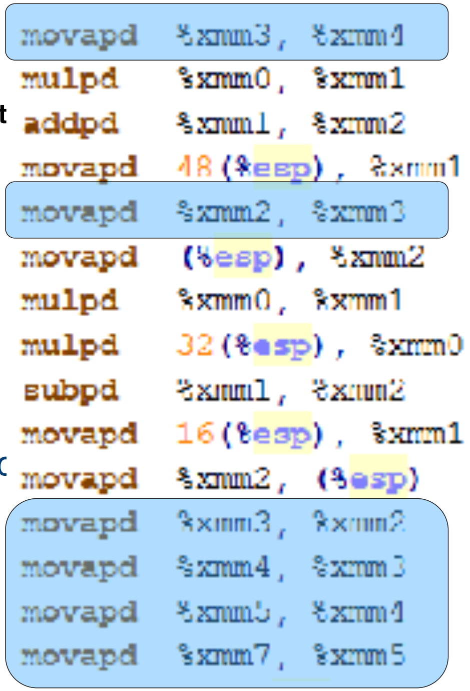


```
movapd  %xmm3, %xmm4
mulpd   %xmm0, %xmm1
addpd   %xmm1, %xmm2
movapd  48(%esp), %xmm1
movapd  %xmm2, %xmm3
movapd  (%esp), %xmm2
mulpd   %xmm0, %xmm1
mulpd   32(%esp), %xmm0
subpd   %xmm1, %xmm2
movapd  16(%esp), %xmm1
movapd  %xmm2, (%esp)
movapd  %xmm3, %xmm2
movapd  %xmm4, %xmm3
movapd  %xmm5, %xmm4
movapd  %xmm7, %xmm5
```

Encountered Issue #2

- Bad eviction chain – scenario 2
  - The problem
    - y is split to fit the register it was evicted from
    - This split creates local interference split artifact that causes domino effect eviction
  - The solution
    - Tailored for this case
      - Identify if a split for evicted register creates local interference artifac
      - Identify if that split artifact will cause domino effect eviction
      - Increase split cost
        - Make this split le 20$ ttractive compared to other splits
    - Commit: https://reviews.llvm.org/rL316295

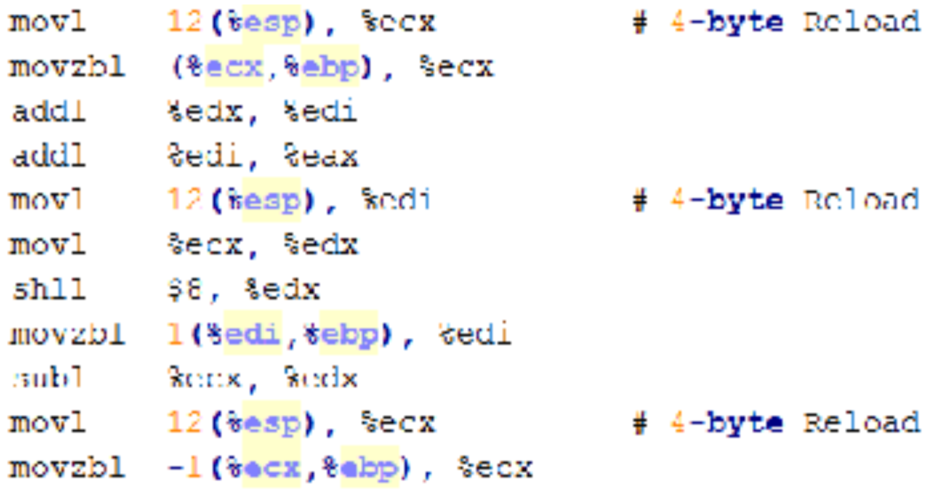




```
movapd   %xmm3,  %xmm4
mulpd    %xmm0,  %xmm1
addpd    %xmm1,  %xmm2
movapd   48(%esp),  %xmm1
movapd   %xmm2,  %xmm3
movapd   (%esp),  %xmm2
mulpd    %xmm0,  %xmm1
mulpd    32(%esp),  %xmm0
subpd    %xmm1,  %xmm2
movapd   16(%esp),  %xmm1
movapd   %xmm2,  (%esp)
movapd   %xmm3,  %xmm2
movapd   %xmm4,  %xmm3
movapd   %xmm5,  %xmm4
movapd   %xmm7,  %xmm5
```

Encountered Issue #3

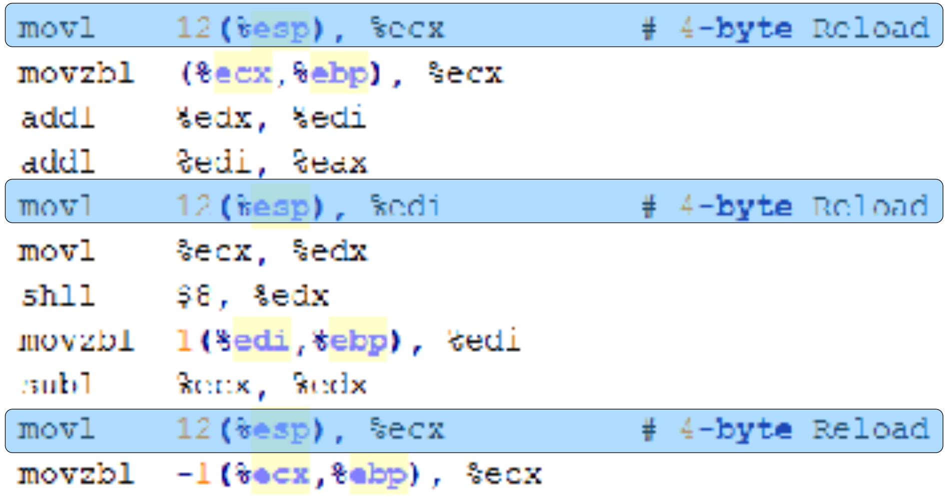
- Multiple reloads from the same location



```
movl     12(%esp), %ecx              # 4-byte Reload
movzbl   (%ecx,%ebp), %ecx
addl     %edx, %edi
addl     %edi, %eax
movl     12(%esp), %edi              # 4-byte Reload
movl     %ecx, %edx
shll     $8, %edx
movzbl   1(%edi,%ebp), %edi
subl     %ecx, %edx
movl     12(%esp), %ecx              # 4-byte Reload
movzbl   -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

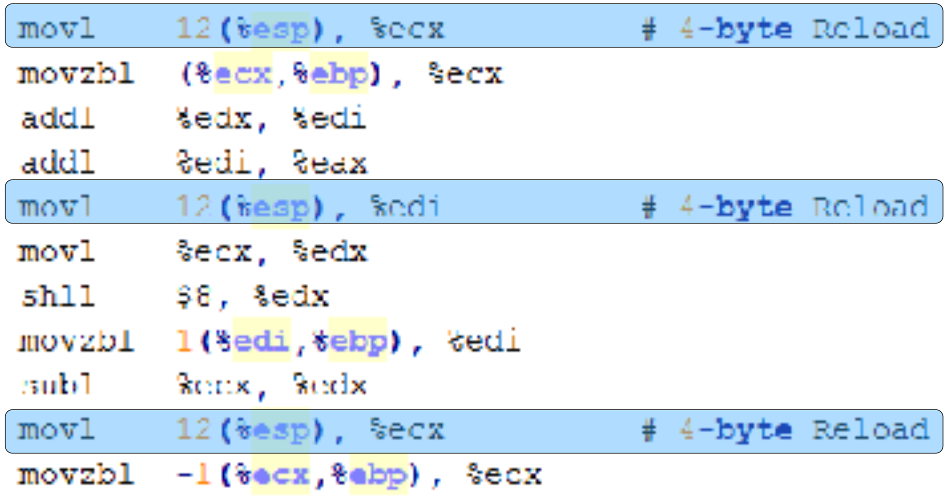
- Multiple reloads from the same location
  - All the reloads are from the same location



```
movl     12(%esp), %ecx                # 4-byte Reload
movzbl   (%ecx,%ebp), %ecx
addl     %edx, %edi
addl     %edi, %eax
movl     12(%esp), %edi                # 4-byte Reload
movl     %ecx, %edx
shll     $8, %edx
movzbl   1(%edi,%ebp), %edi
subl     %ecx, %edx
movl     12(%esp), %ecx                # 4-byte Reload
movzbl   -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

- Multiple reloads from the same location
  - All the reloads are from the same location
  - Appeared in a hot loop after a higher level change

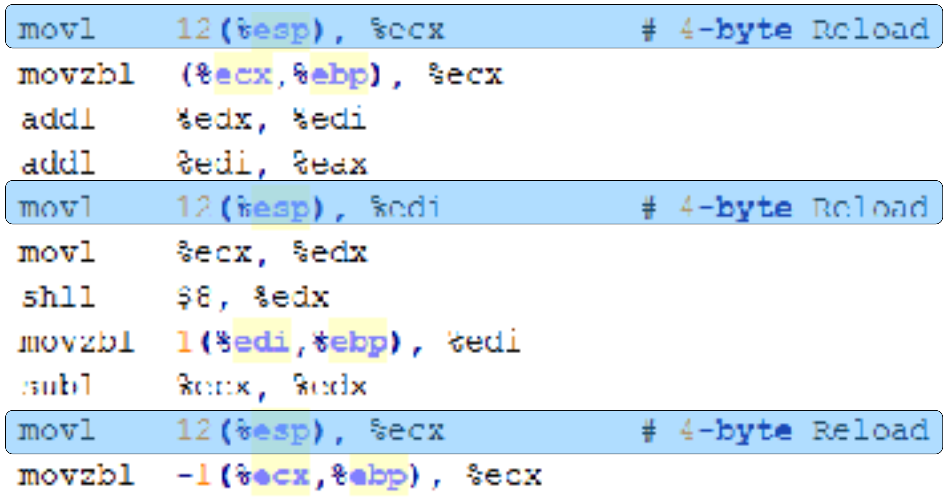


```
movl    12(%esp), %ecx             # 4-byte Reload
movzbl  (%ecx,%ebp), %ecx
addl    %edx, %edi
addl    %edi, %eax
movl    12(%esp), %edi             # 4-byte Reload
movl    %ecx, %edx
shll    $8, %edx
movzbl  1(%edi,%ebp), %edi
subl    %ecx, %edx
movl    12(%esp), %ecx             # 4-byte Reload
movzbl  -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

- Multiple reloads from the same location

Before Change	After Change
Loop MBB is the same until Greedy	Loop MBB is the same until Greedy

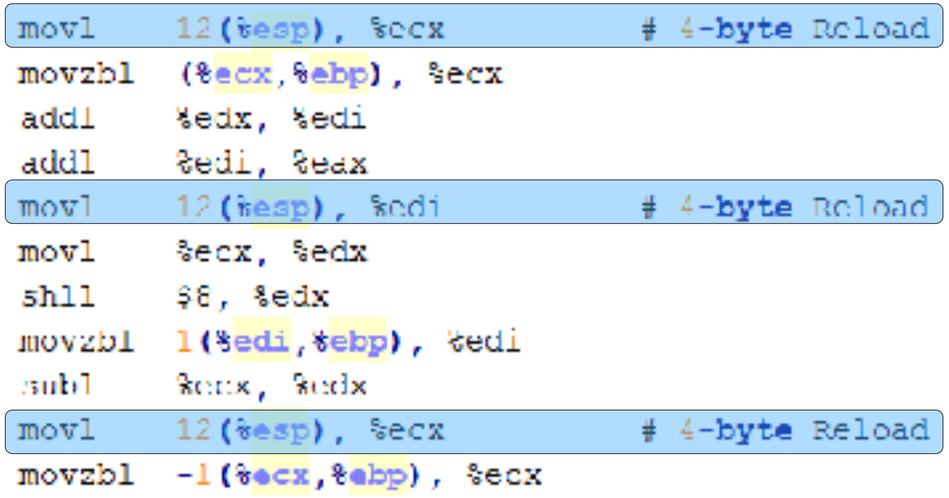


```
movl      12(%esp), %ecx                # 4-byte Reload
movzbl    (%ecx,%ebp), %ecx
addl      %edx, %edi
addl      %edi, %eax
movl      12(%esp), %edi                # 4-byte Reload
movl      %ecx, %edx
shll      $8, %edx
movzbl    1(%edi,%ebp), %edi
subl      %ecx, %edx
movl      12(%esp), %ecx                # 4-byte Reload
movzbl    -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

- Multiple reloads from the same location

Before Change	After Change
Loop MBB is the same until Greedy x is split for R0	Loop MBB is the same until Greedy x is split for R1

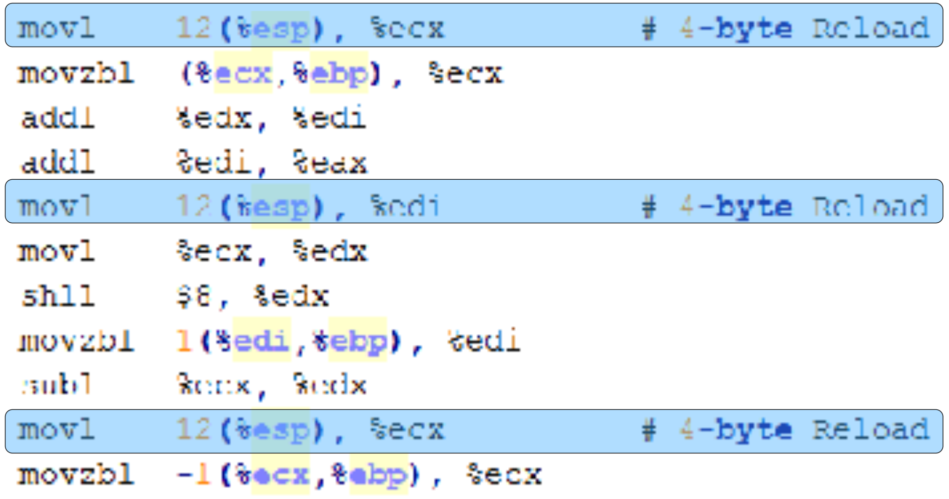


```
movl     12(%esp), %ecx                # 4-byte Reload
movzbl   (%ecx,%ebp), %ecx
addl     %edx, %edi
addl     %edi, %eax
movl     12(%esp), %edi                # 4-byte Reload
movl     %ecx, %edx
shll     $8, %edx
movzbl   1(%edi,%ebp), %edi
subl     %ecx, %edx
movl     12(%esp), %ecx                # 4-byte Reload
movzbl   -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

- Multiple reloads from the same location

Before Change	After Change
Loop MBB is the same until Greedy	Loop MBB is the same until Greedy
x is split for R0	x is split for R1
Split doesn’t have local interferences	Split has local interference in Loop’s MBB

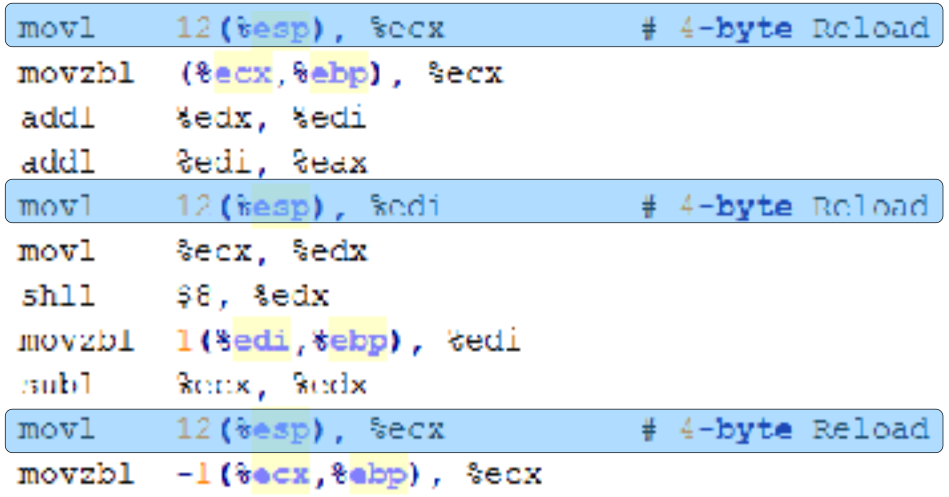


```
movl     12(%esp), %ecx              # 4-byte Reload
movzbl   (%ecx,%ebp), %ecx
addl     %edx, %edi
addl     %edi, %eax
movl     12(%esp), %edi              # 4-byte Reload
movl     %ecx, %edx
shll     $8, %edx
movzbl   1(%edi,%ebp), %edi
subl     %ecx, %edx
movl     12(%esp), %ecx              # 4-byte Reload
movzbl   -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

- Multiple reloads from the same location

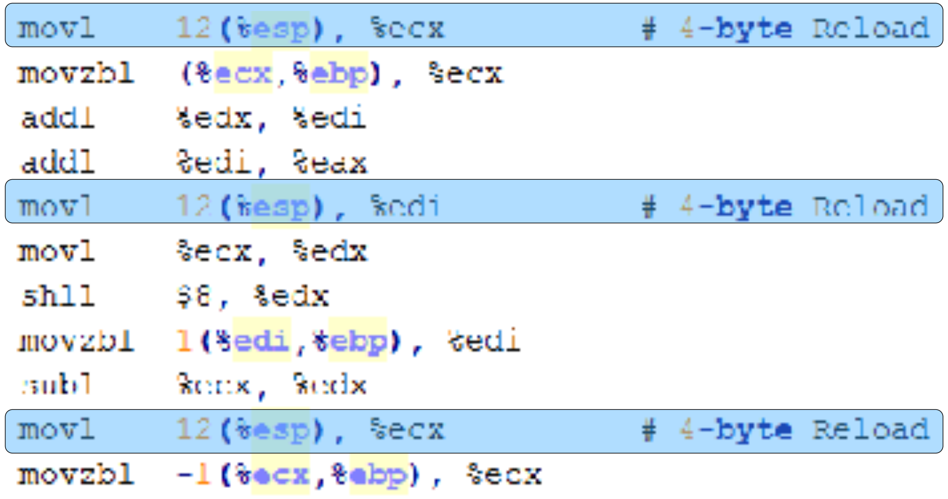
Before Change	After Change
Loop MBB is the same until Greedy x is split for R0 Split doesn't have local interferences	Loop MBB is the same until Greedy x is split for R1 Split has local interference in Loop's MBB Local interference spilled & reloaded around uses



```
movl      12(%esp), %ecx                # 4-byte Reload
movzbl    (%ecx,%ebp), %ecx
addl      %edx, %edi
addl      %edi, %eax
movl      12(%esp), %edi                # 4-byte Reload
movl      %ecx, %edx
shll      $8, %edx
movzbl    1(%edi,%ebp), %edi
subl      %ecx, %edx
movl      12(%esp), %ecx                # 4-byte Reload
movzbl    -1(%ecx,%ebp), %ecx
```

Encountered Issue #3

- Multiple reloads from the same location
  - The problem
    - Local interference interval has a lot of uses
    - This interval is spilled and reloaded
  - Solution
    - Identify if the created local interference interval will spill
    - Increase split cost
      - Make this split less a 20$ ive compared to other splits
    - Commit: https://reviews.llvm.org/rL323870



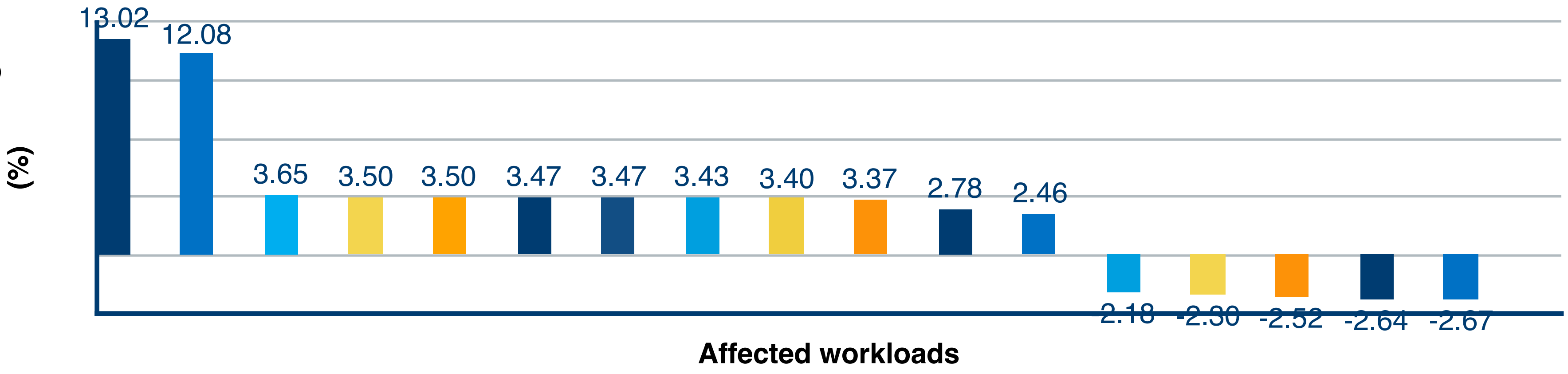
```
movl    12(%esp), %ecx          # 4-byte Reload
movzbl  (%ecx,%ebp), %ecx
addl    %edx, %edi
addl    %edi, %eax
movl    12(%esp), %edi          # 4-byte Reload
movl    %ecx, %edx
shll    $8, %edx
movzbl  1(%edi,%ebp), %edi
subl    %ecx, %edx
movl    12(%esp), %ecx          # 4-byte Reload
movzbl  -1(%ecx,%ebp), %ecx
```

Greedy Register Allocator

- Greedy Register Allocator Overview
- Region Split
- Encountered Issues
- **Performance Impact**

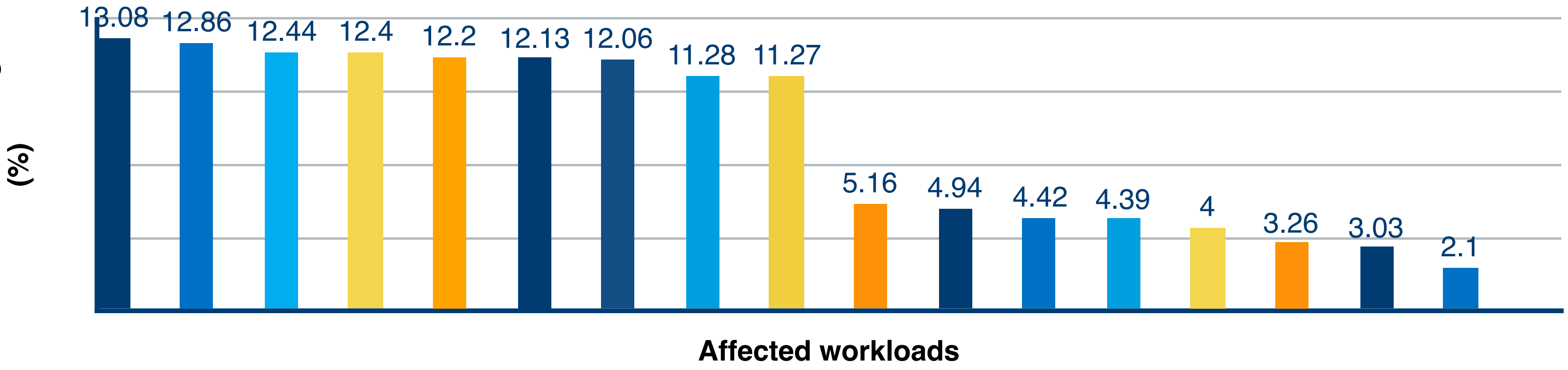
Fix for Bad Eviction Chains - Issues #1, #2

- Fix affected mostly EEMBC workloads
- Regressions unrelated to this change
- No actual compile time impact on CTMark



Fix for Multiple Reloads - Issue #3

- Fix affected mostly EEMBC workloads
- No actual compile time impact on CTMark



Conclusions

- Local interference caused by split may have a negative affect
- Current split cost does not take this affect into account
- Committed solutions tailored to catch 3 specific scenarios
- Need a more holistic approach for quantifying the cost of local interferences caused by split

marina.yatsina@intel.com