Split Register Allocation

Boubacar Diouf [1], Albert Cohen [1],
Fabrice Rastello [2], John Cavazos [3]

[1] INRIA Saclay

[2] Ecole Normale Supérieure

[3] University of Delaware

Outline

Introduction

- Just-in-time (JIT) compilation has been introduced to ally:

Introduction

- Just-in-time (JIT) compilation has been introduced to ally:
  1. portability (interpretation)

INTRODUCTION

- Just-in-time (JIT) compilation has been introduced to ally:
  1. portability (interpretation)
  2. better performance (static compilation)

Split compilation: motivation

Hard Optimisation Problem:

- Exponential complexity algorithm
- Execution context dependent (data-set, target)

Difficulty:

1. Exploit information from expensive analysis
2. When execution context is known

How to achieve these goals simultaneously?

Beyond annotation-enhanced JIT Compilation

1. Reducing dynamic compilation time [Krintz'01]
2. Improving performance of generated code [Jones'00]

Beyond annotation-enhanced JIT compilation

1. Reducing dynamic compilation time [Krintz'01]
2. Improving performance of generated code [Jones'00]
3. The goal is to split complex and target-dependent optimisations into two coordinated stages: offline (static compiler) and online (JIT compiler)

A case for Split Compilation

Register allocation:

1. Have frequent accessed variables in processor's registers
2. One of the most profitable optimisation
3. It is not trivial to split and we will see why and how

Register Allocation

- Graph Coloring framework [Chaitin'81], [Briggs'94]

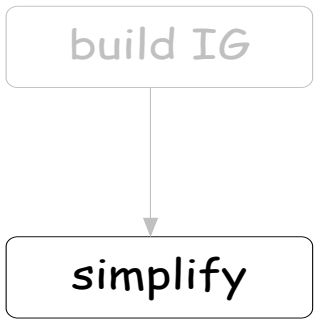
REGISTER ALLOCATION

- Graph Coloring framework [Chaitin'81], [Briggs'94]

build IG

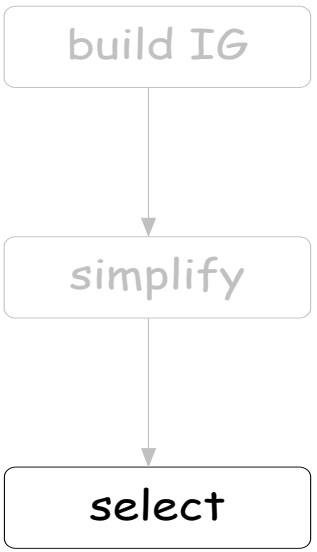
REGISTER ALLOCATION

- Graph Coloring framework [Chaitin'81], [Briggs'94]



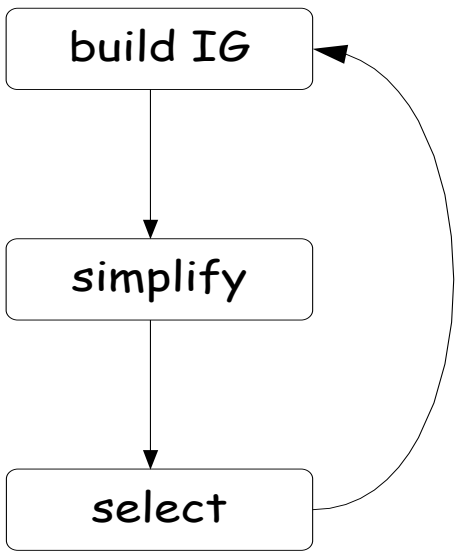
Register Allocation

- Graph Coloring framework [Chaitin'81], [Briggs'94]



REGISTER ALLOCATION

- Graph Coloring framework [Chaitin'81], [Briggs'94]



Decoupled Register Allocation

- [Appel'01], [Hack'06], [Bouchez'06]

Procedure

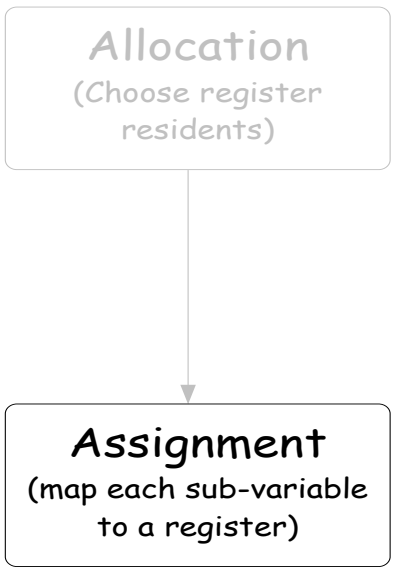
Allocation
(Choose register residents)

Example

Decoupled Register Allocation

- [Appel'01], [Hack'06], [Bouchez'06]

Procedure

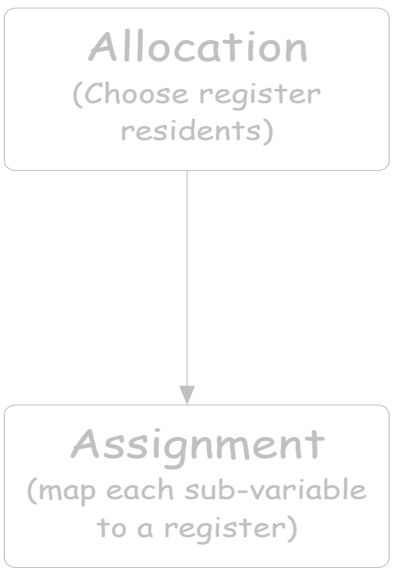


Example

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE



EXAMPLE

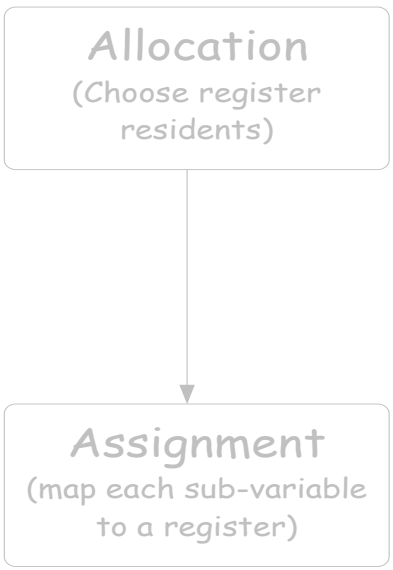
code

```
1: d = ...
2: b = load …
3: b = b * d
4: a = load …
5: a = d / a
6: c = a / b
7: a = b + c
8: store c
9: store a
```

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE



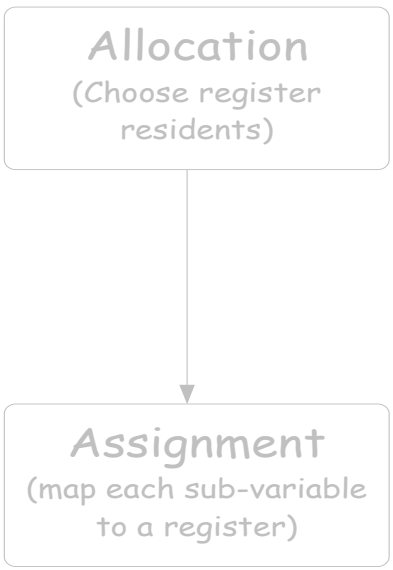
EXAMPLE

code	live
1: d = ...	d
2: b = load …	b,d
3: b = b * d	b,d
4: a = load …	a,b,d
5: a = d / a	a,b
6: c = a / b	b,c
7: a = b + c	a,c
8: store c	a
9: store a	

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE



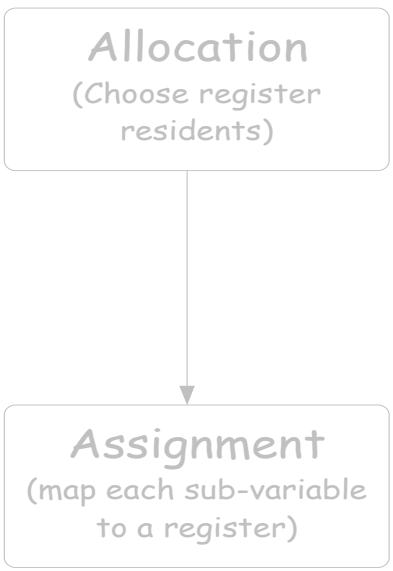
EXAMPLE

code	live	
1: d = ...	d	1
2: b = load …	b,d	2
3: b = b * d	b,d	2
4: a = load …	a,b,d	3
5: a = d / a	a,b	2
6: c = a / b	b,c	2
7: a = b + c	a,c	2
8: store c	a	1
9: store a		

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE



EXAMPLE

code	live	
1: d = ...	d	1
2: b = load …	b,d	2
3: b = b * d	b,d	2
4: a = load …	a,b,d	3
5: a = d / a	a,b	2
6: c = a / b	b,c	2
7: a = b + c	a,c	2
8: store c	a	1
9: store a		

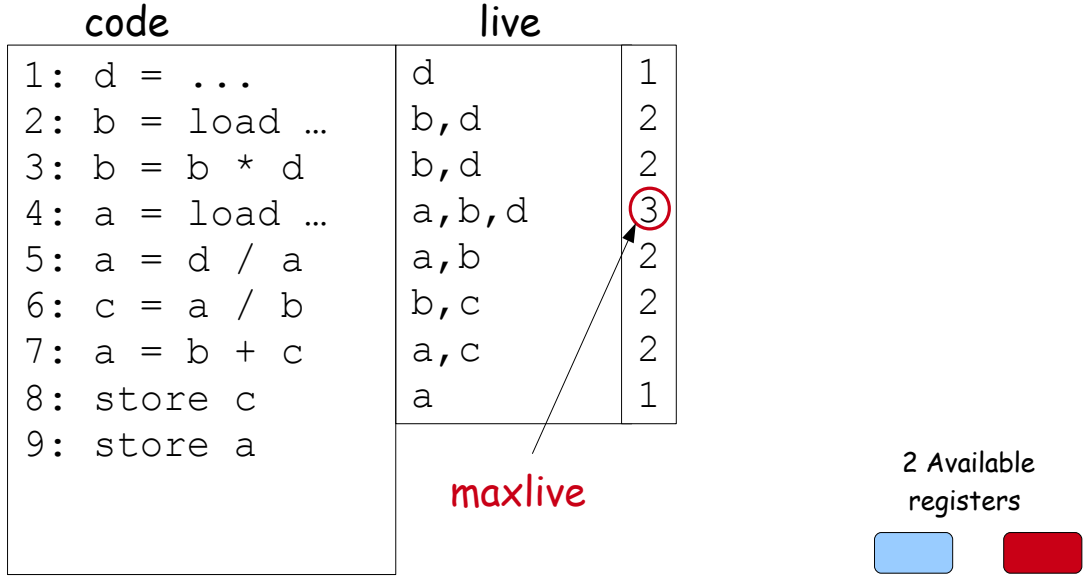
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



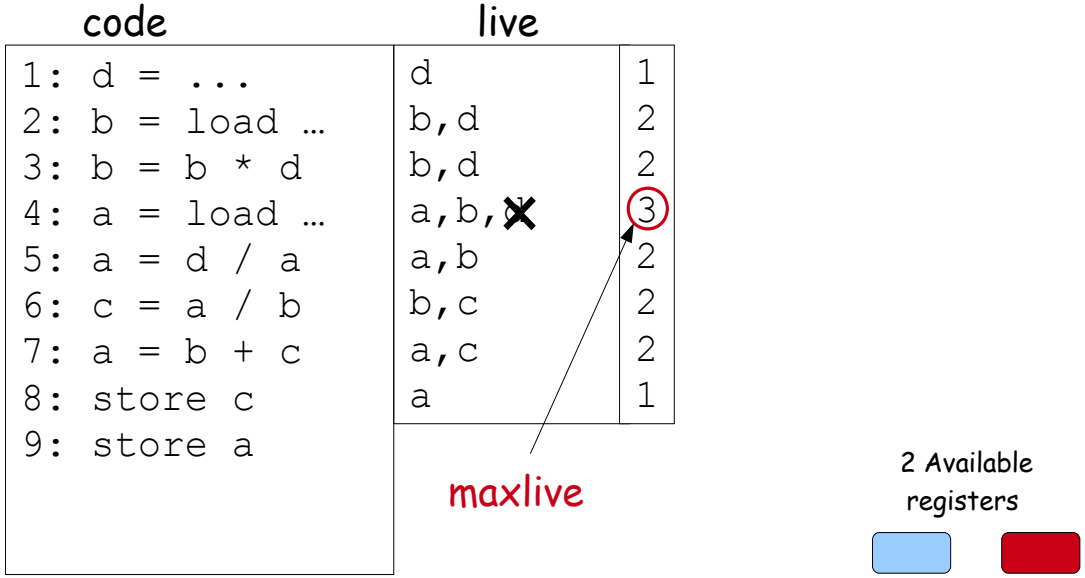
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



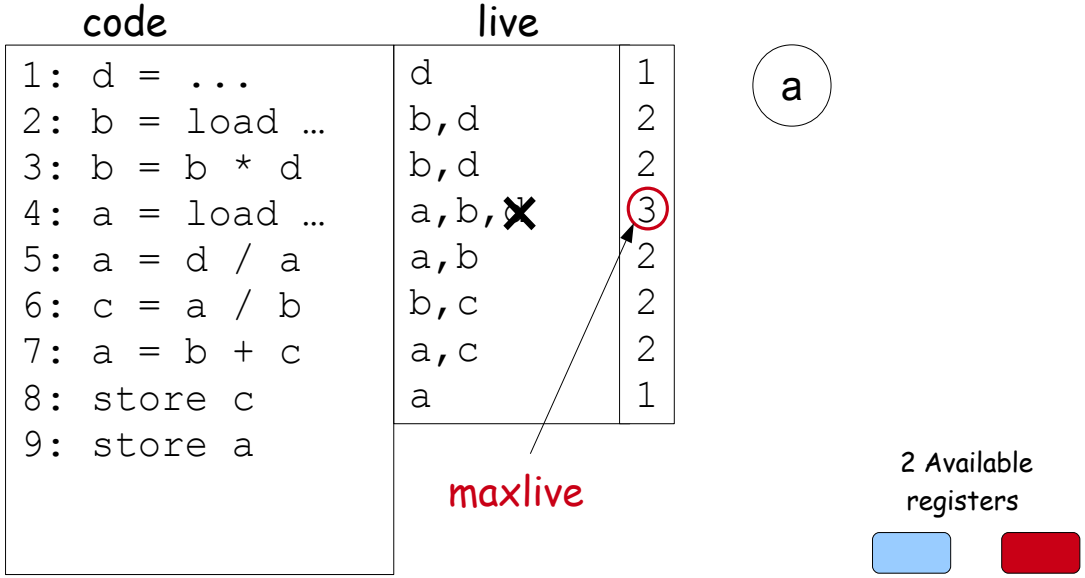
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



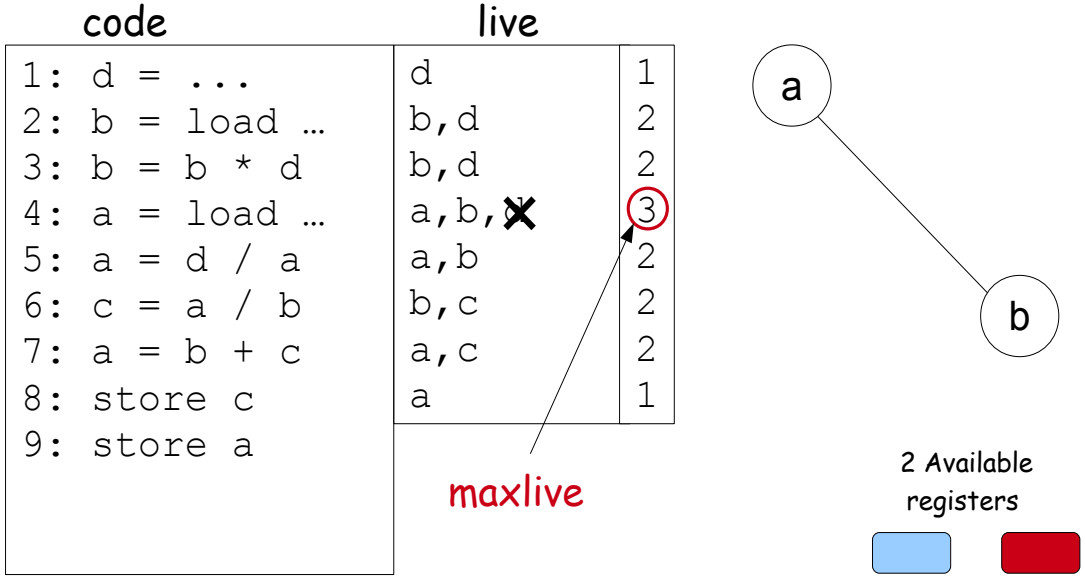
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



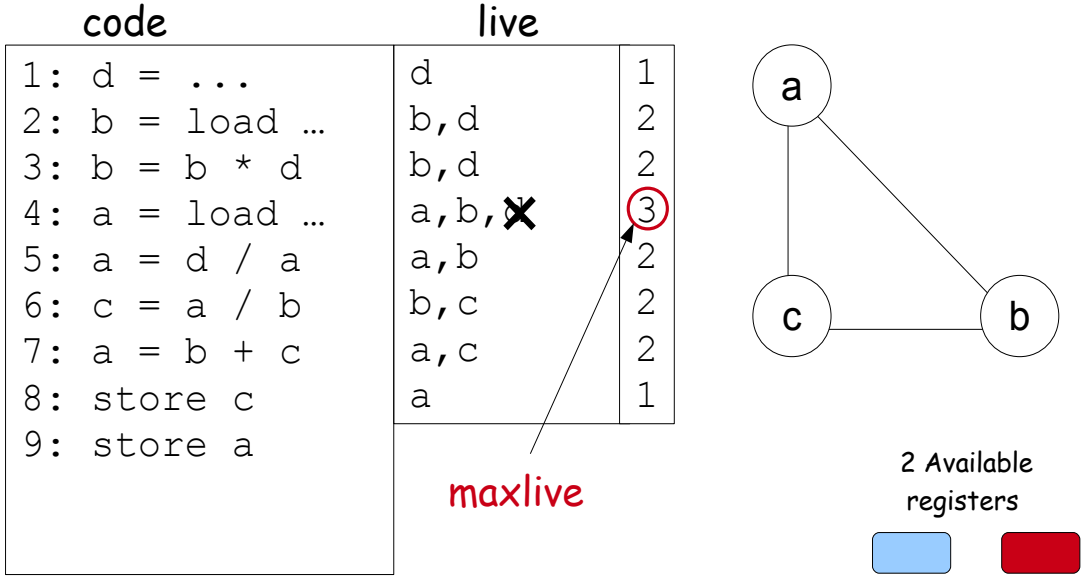
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



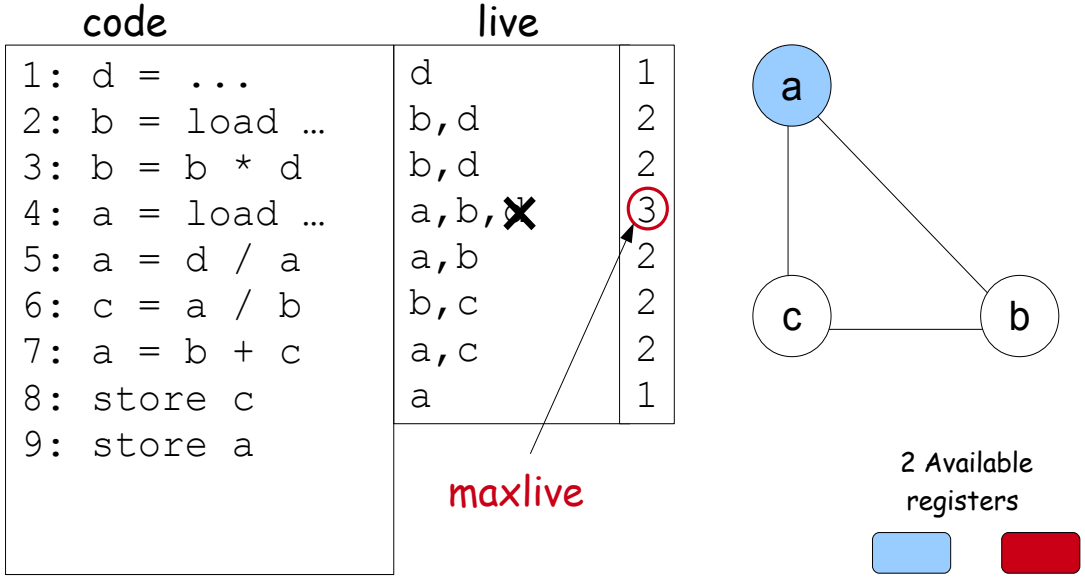
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



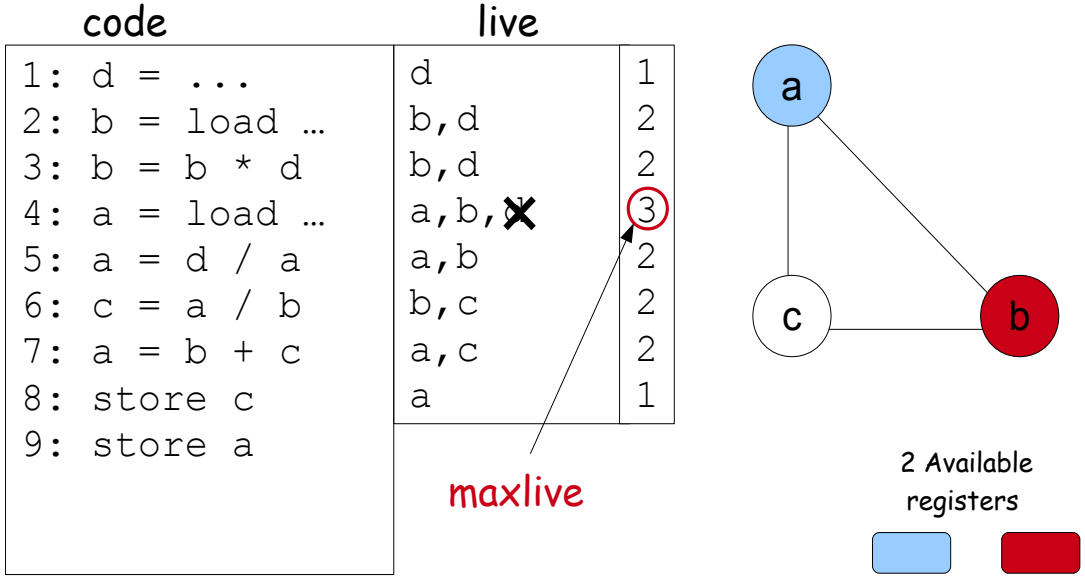
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)

EXAMPLE



DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE

code	live	
1: d = ...	d	1
2: b = load …	b,d	2
3: b = b * d	b,d	2
4: **a** = load …	a,b,d	3
5: a = d / **a**	a,b	2
6: c = a / b	b,c	2
7: a = b + c	a,c	2
8: store c	a	1
9: store a		

2 Available registers

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE

code	live	
1: d = ...	d	1
2: b = load …	b,d	2
3: b = b * d	b,d	2
4: **a** = load …	a,b,d	3
5: **a** = d / **a**	a,b	2
6: c = **a** / b	b,c	2
7: a = b + c	a,c	2
8: store c	a	1
9: store a		

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE

code	live	
1: d = ...	d	1
2: b = load …	b,d	2
3: b = b * d	b,d	2
4: **a** = load …	a,b,d	3
5: **a** = d / **a**	a,b	2
6: c = **a** / b	b,c	2
7: **a** = b + c	a,c	2
8: store c	a	1
9: store **a**		

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE

code	live	
1: d = ...	d	1
2: b = load …	b,d	2
3: b = b * d	b,d	2
4: **a1** = load …	a,b,d	3
5: **a2** = d / **a1**	a,b	2
6: c = **a2** / b	b,c	2
7: **a3** = b + c	a,c	2
8: store c	a	1
9: store **a3**		

DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE

code	live	
1: d = ...	d	1
2: b1= load …	b1,d	2
3: b2= b1 * d	b2,d	2
4: a1= load …	a1,b2,d	3
5: a2= d / a1	a2,b2	2
6: c1= a2 / b2	b2,c1	2
7: a3= b2 + c1	a3,c1	2
8: store c1	a3	1
9: store a3		

2 Available registers

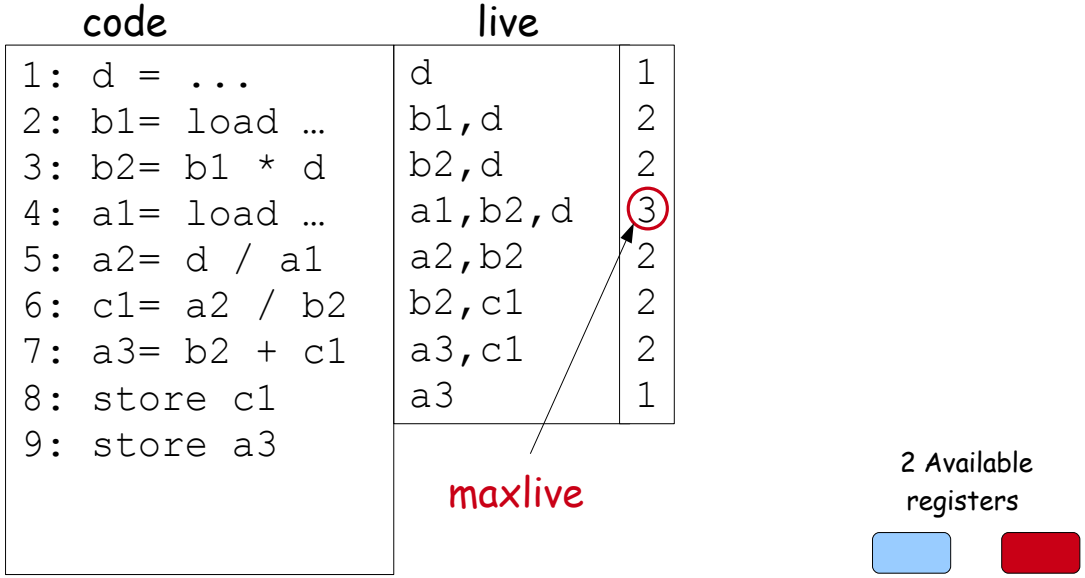
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE



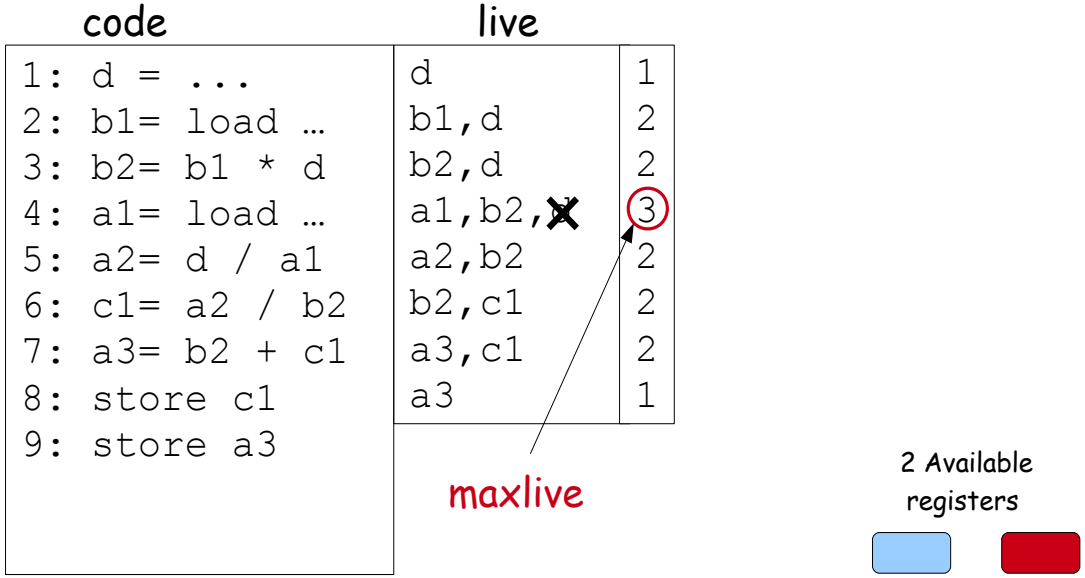
DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE



DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

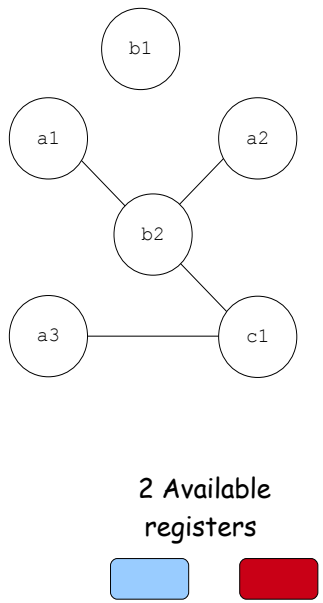
PROCEDURE

Allocation
(Choose register residents)
splitting

EXAMPLE

code	live	
1: d = ...	d	1
2: b1= load …	b1,d	2
3: b2= b1 * d	b2,d	2
4: a1= load …	a1,b2,	3
5: a2= d / a1	a2,b2	2
6: c1= a2 / b2	b2,c1	2
7: a3= b2 + c1	a3,c1	2
8: store c1	a3	1
9: store a3		

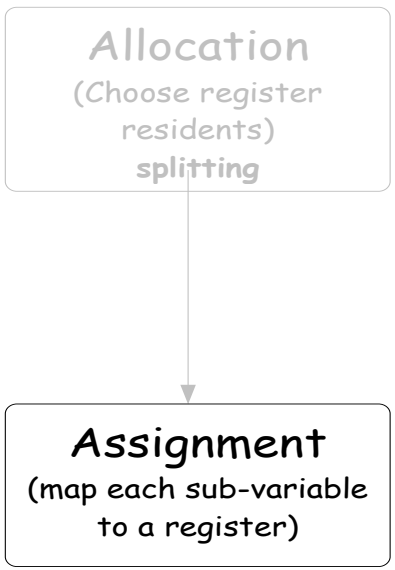
maxlive



DECOUPLED REGISTER ALLOCATION

- [Appel'01], [Hack'06], [Bouchez'06]

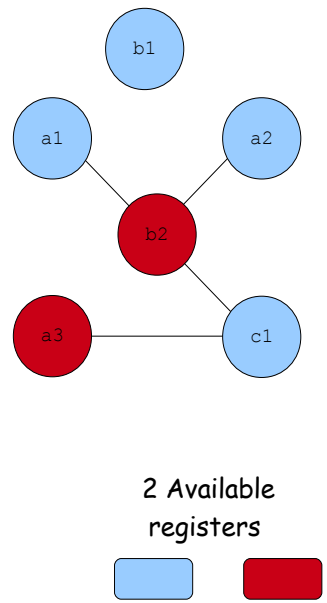
PROCEDURE



EXAMPLE

code	live	
1: d = ...	d	1
2: b1= load …	b1,d	2
3: b2= b1 * d	b2,d	2
4: a1= load …	a1,b2,~~d~~	3
5: a2= d / a1	a2,b2	2
6: c1= a2 / b2	b2,c1	2
7: a3= b2 + c1	a3,c1	2
8: store c1	a3	1
9: store a3		

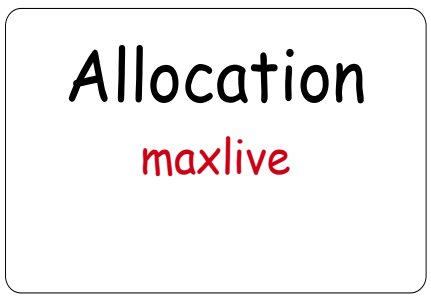
maxlive



Global view of Split Register Allocation

Global view of Split Register Allocation

GLOBAL VIEW OF SPLIT REGISTER ALLOCATION



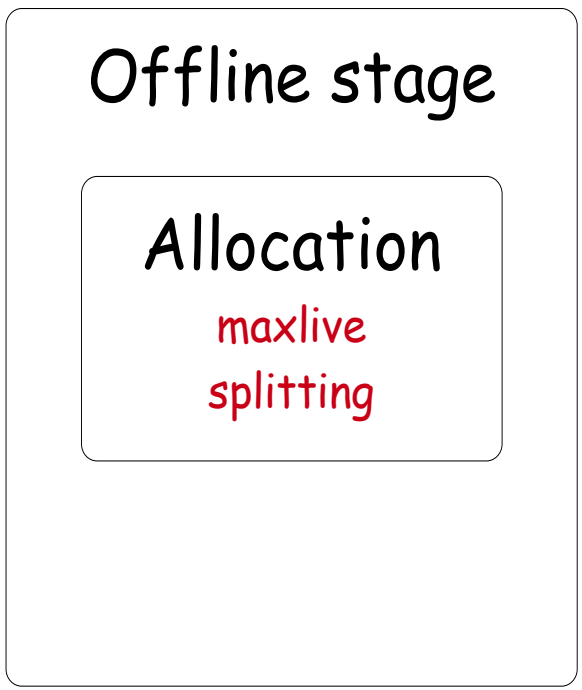
GLOBAL VIEW OF SPLIT REGISTER ALLOCATION

Allocation

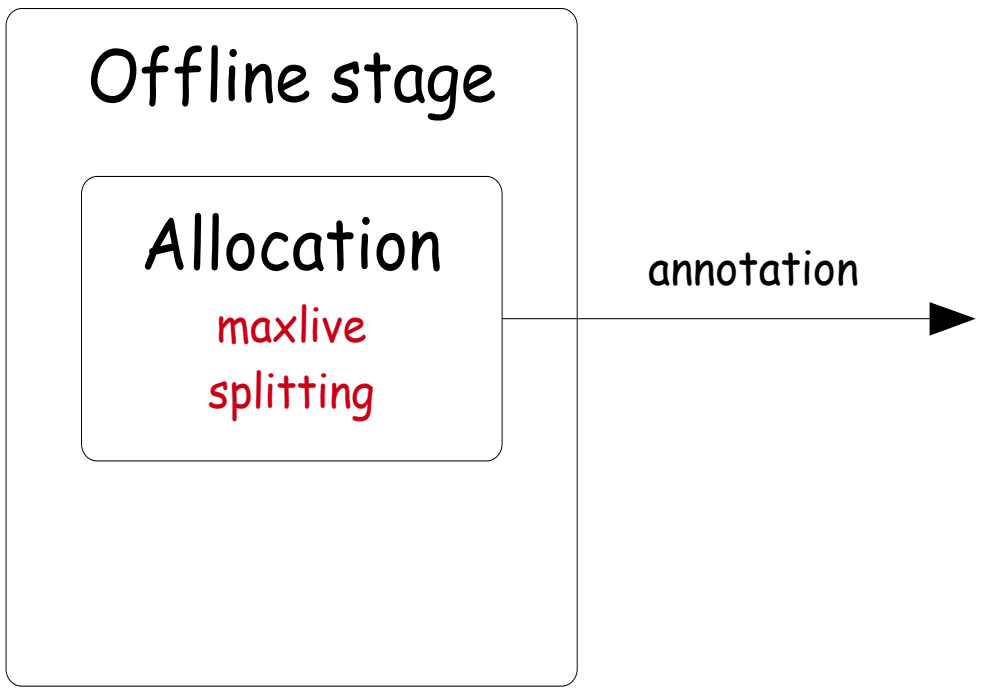
maxlive

splitting

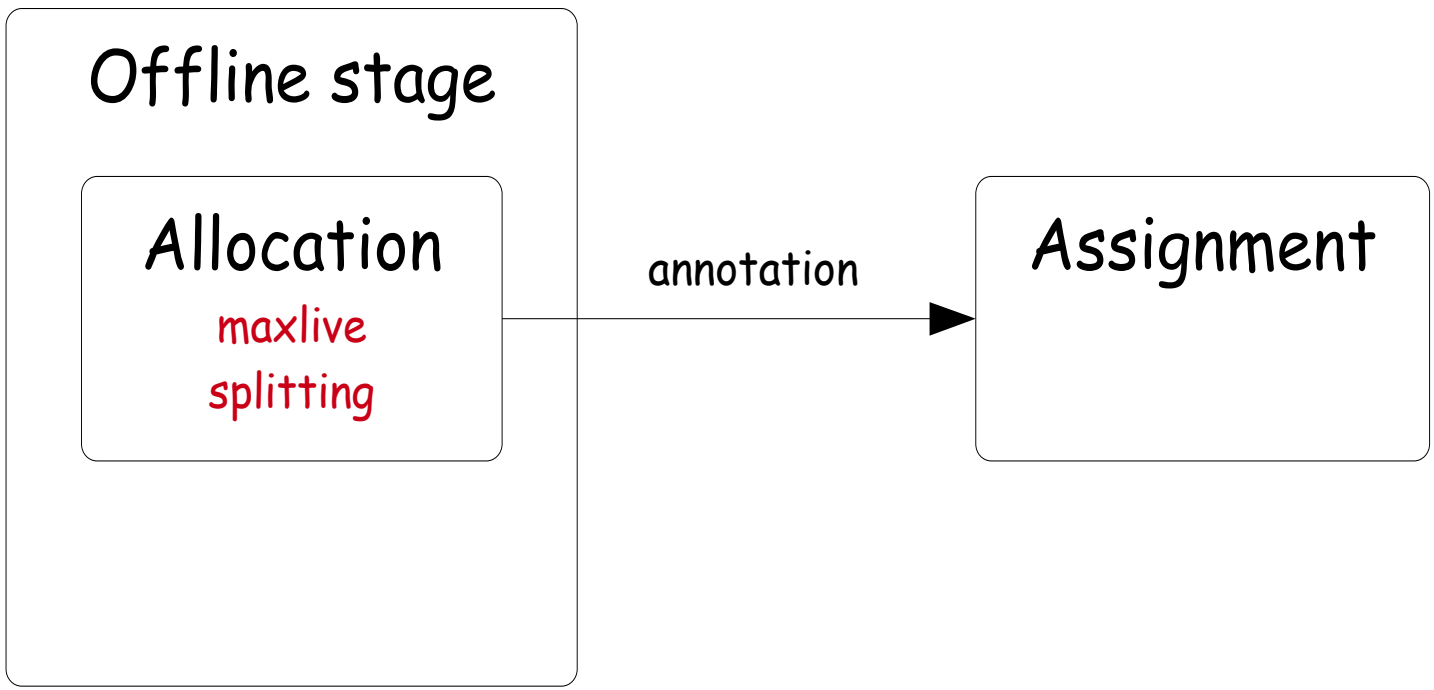
Global view of Split Register Allocation



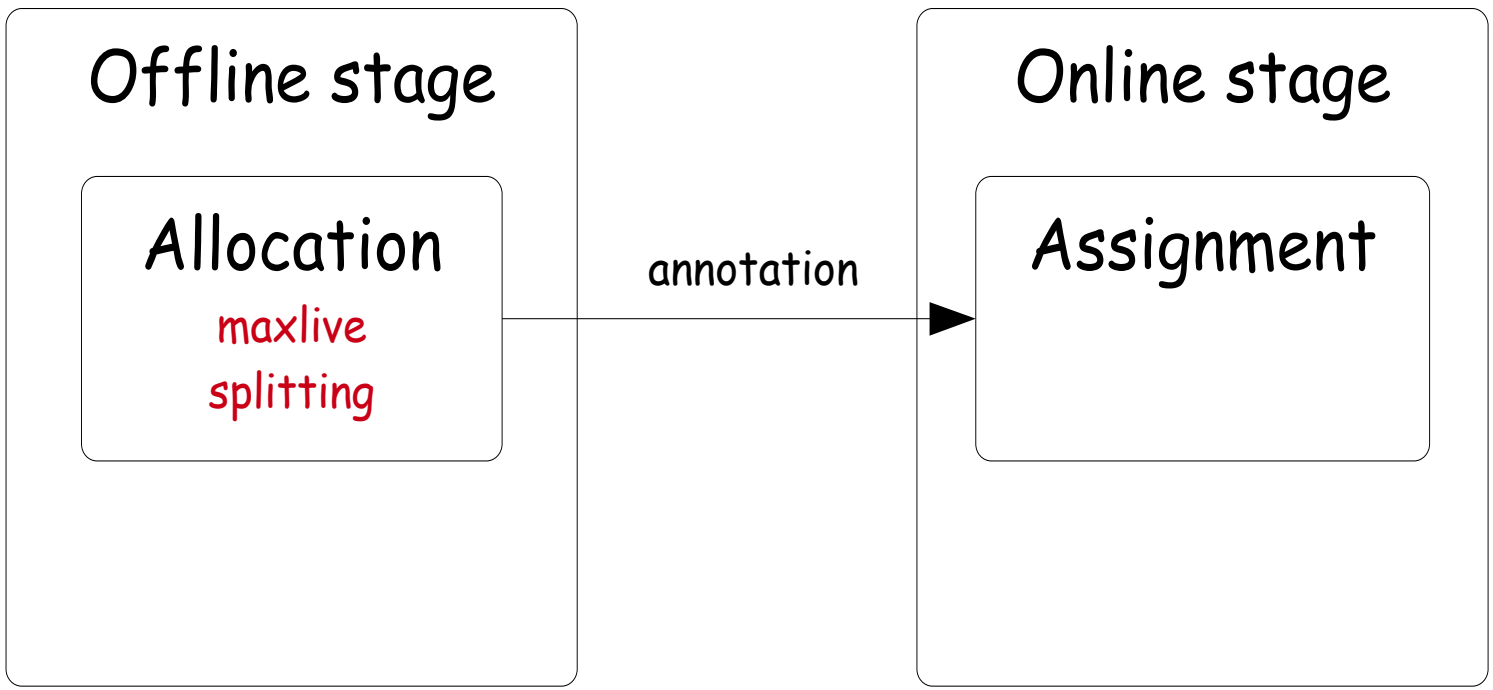
GLOBAL VIEW OF SPLIT REGISTER ALLOCATION



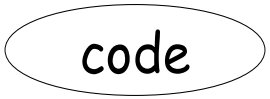
Global view of Split Register Allocation



GLOBAL VIEW OF SPLIT REGISTER ALLOCATION

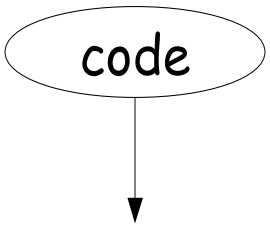


STAGES OF SPLIT REGISTER ALLOCATION



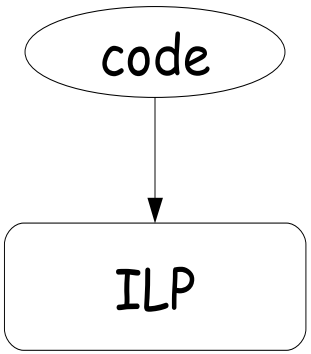
offline

Stages of Split Register Allocation



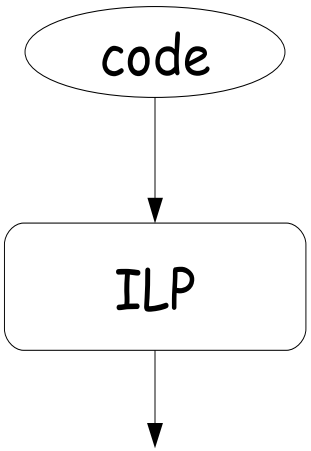
offline

STAGES OF SPLIT REGISTER ALLOCATION



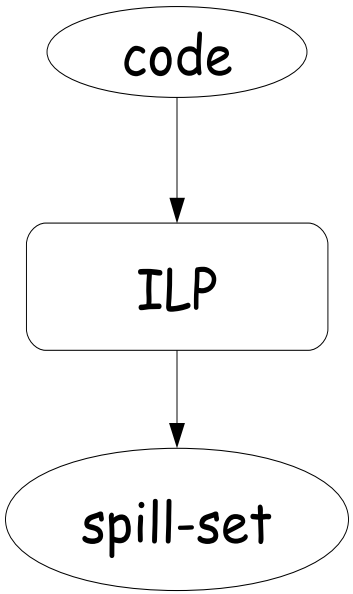
offline

Stages of Split Register Allocation



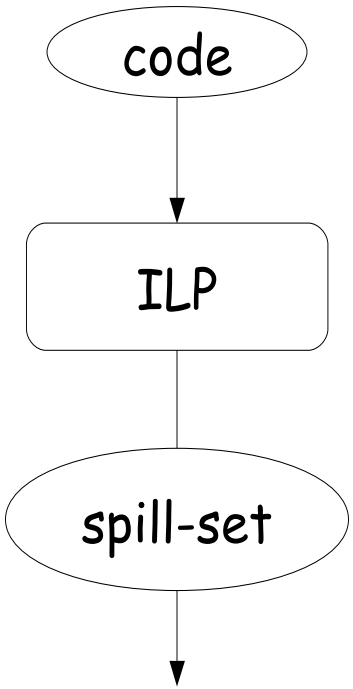
offline

STAGES OF SPLIT REGISTER ALLOCATION



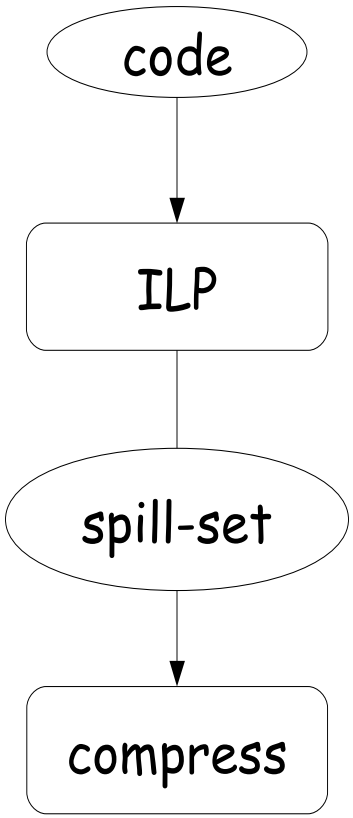
offline

STAGES OF SPLIT REGISTER ALLOCATION



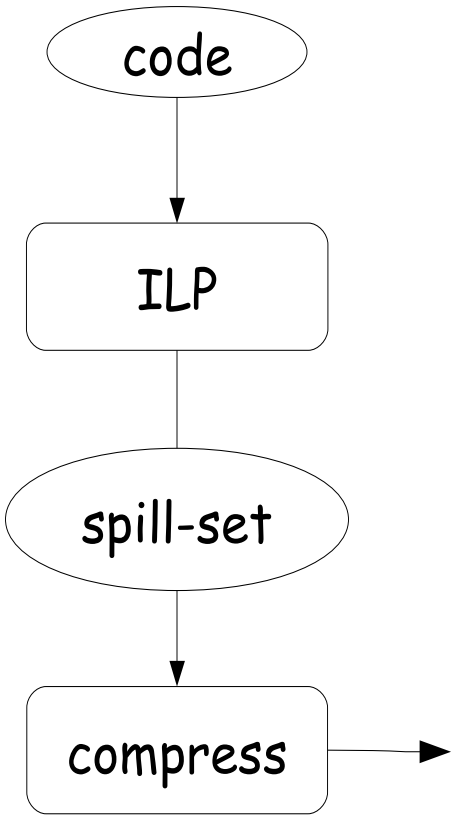
offline

STAGES OF SPLIT REGISTER ALLOCATION

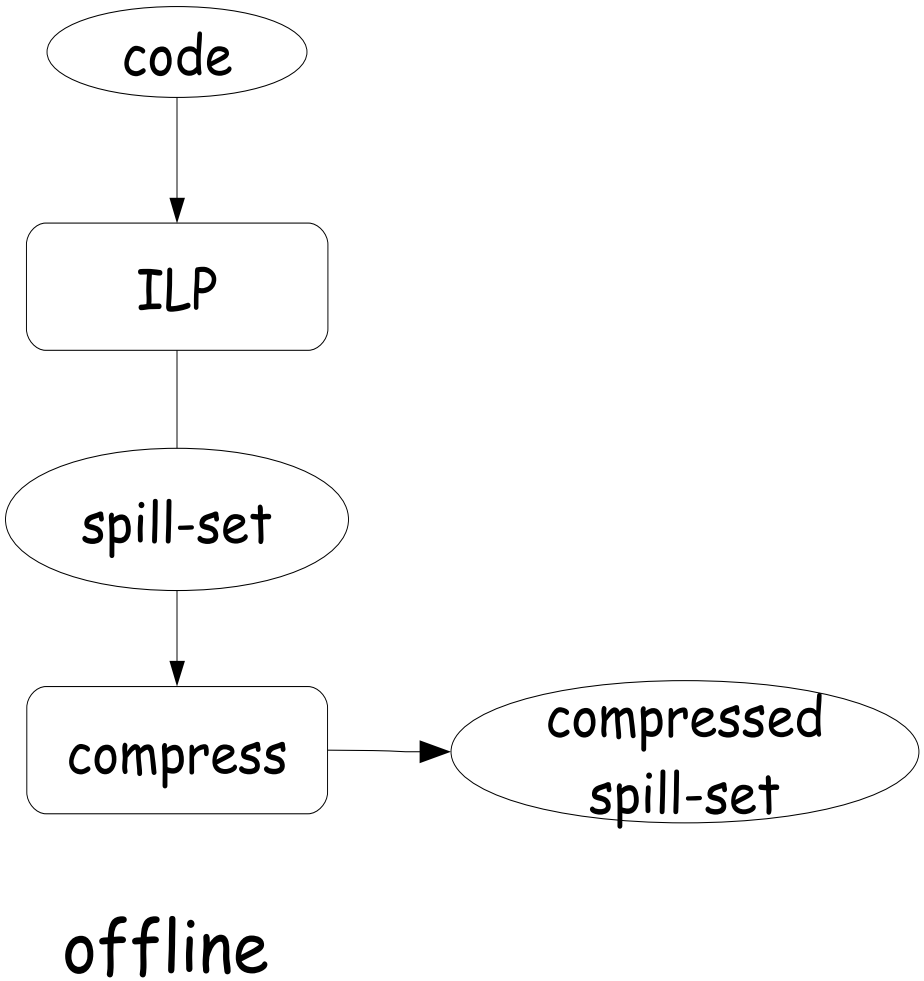


offline

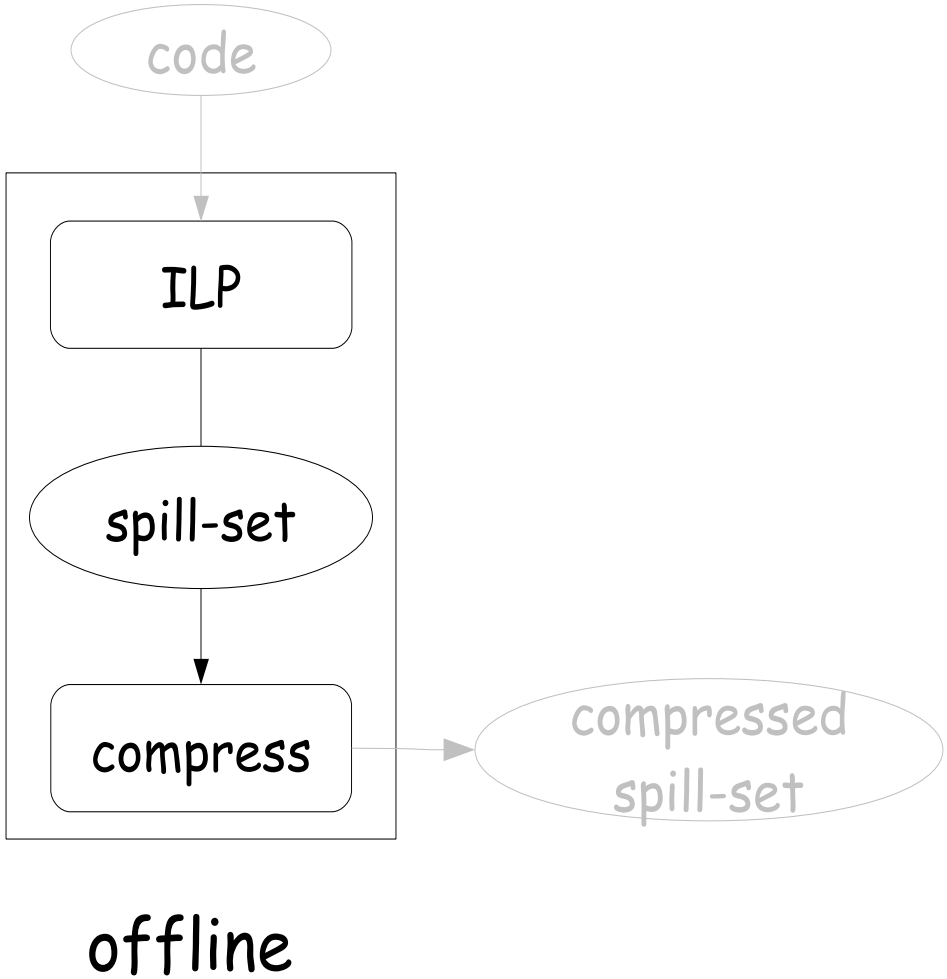
STAGES OF SPLIT REGISTER ALLOCATION



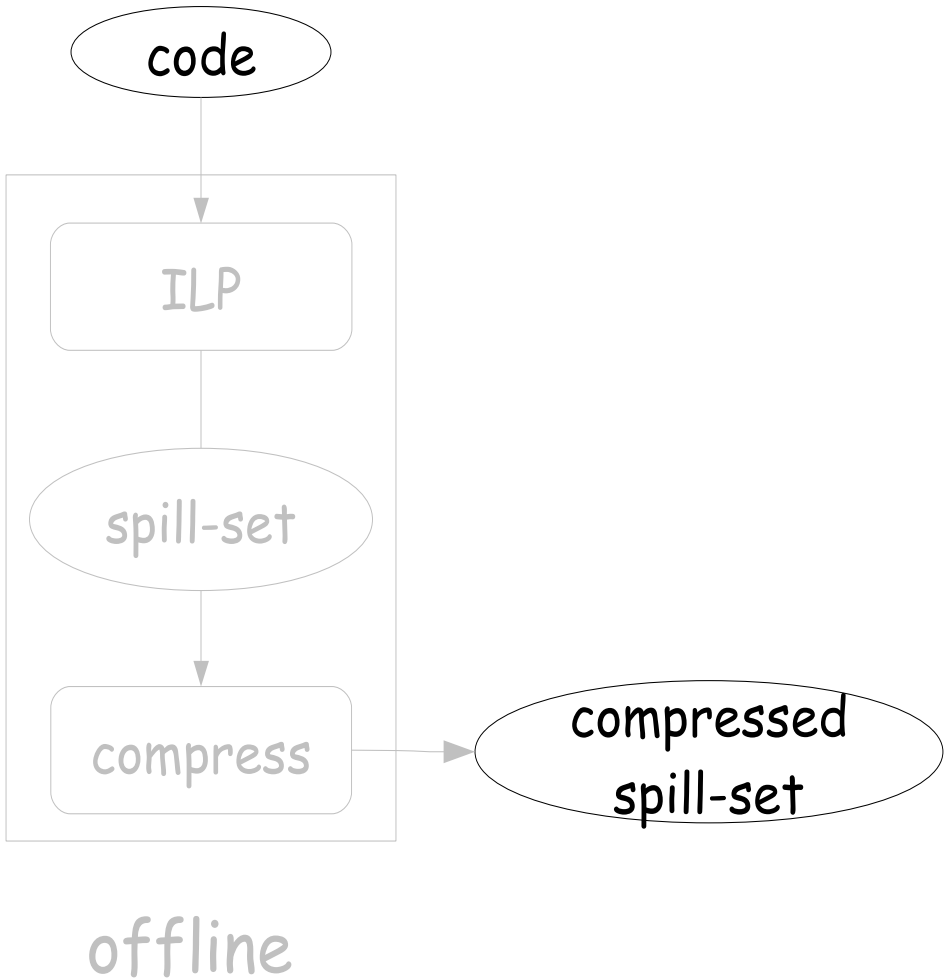
STAGES OF SPLIT REGISTER ALLOCATION



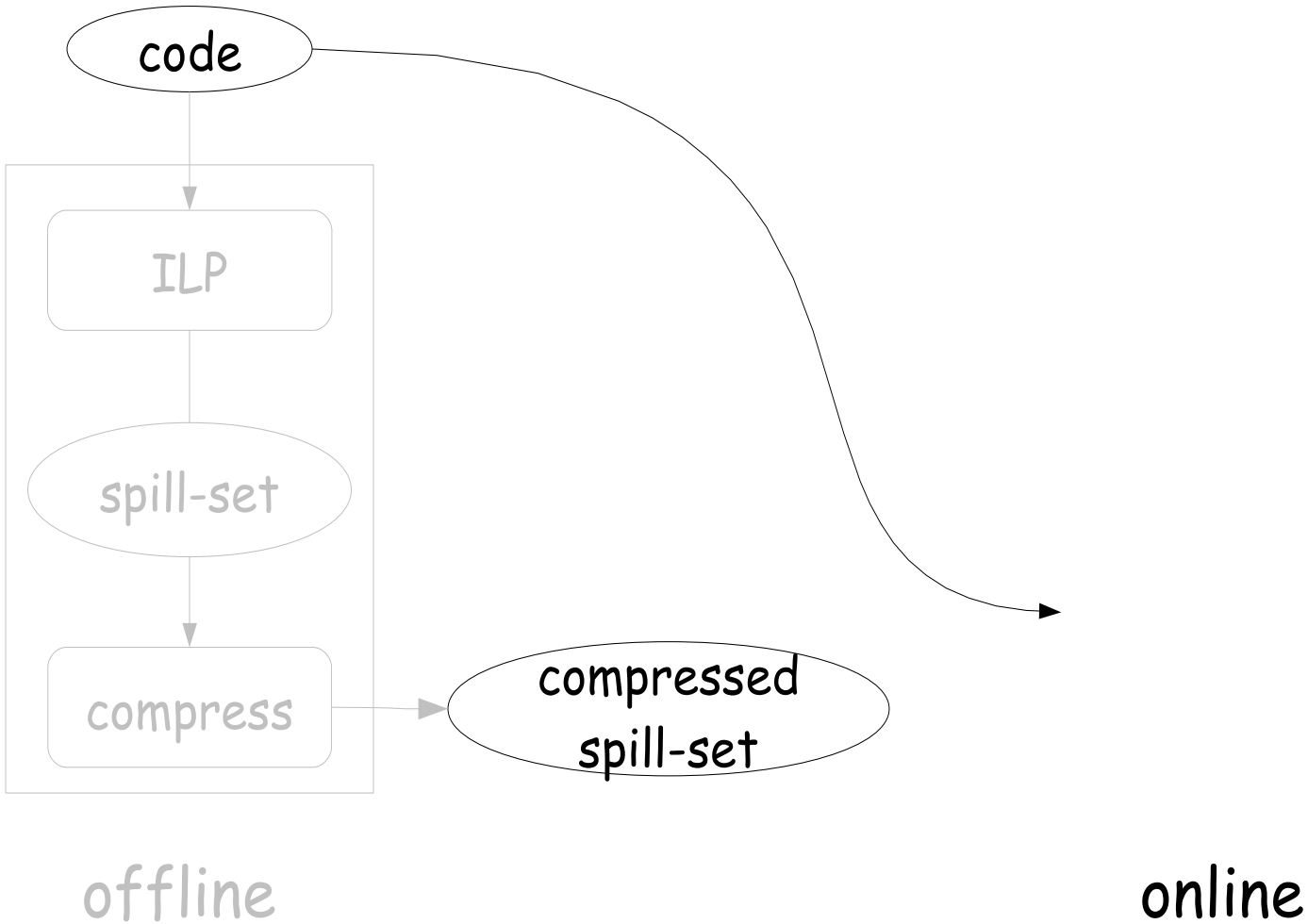
STAGES OF SPLIT REGISTER ALLOCATION



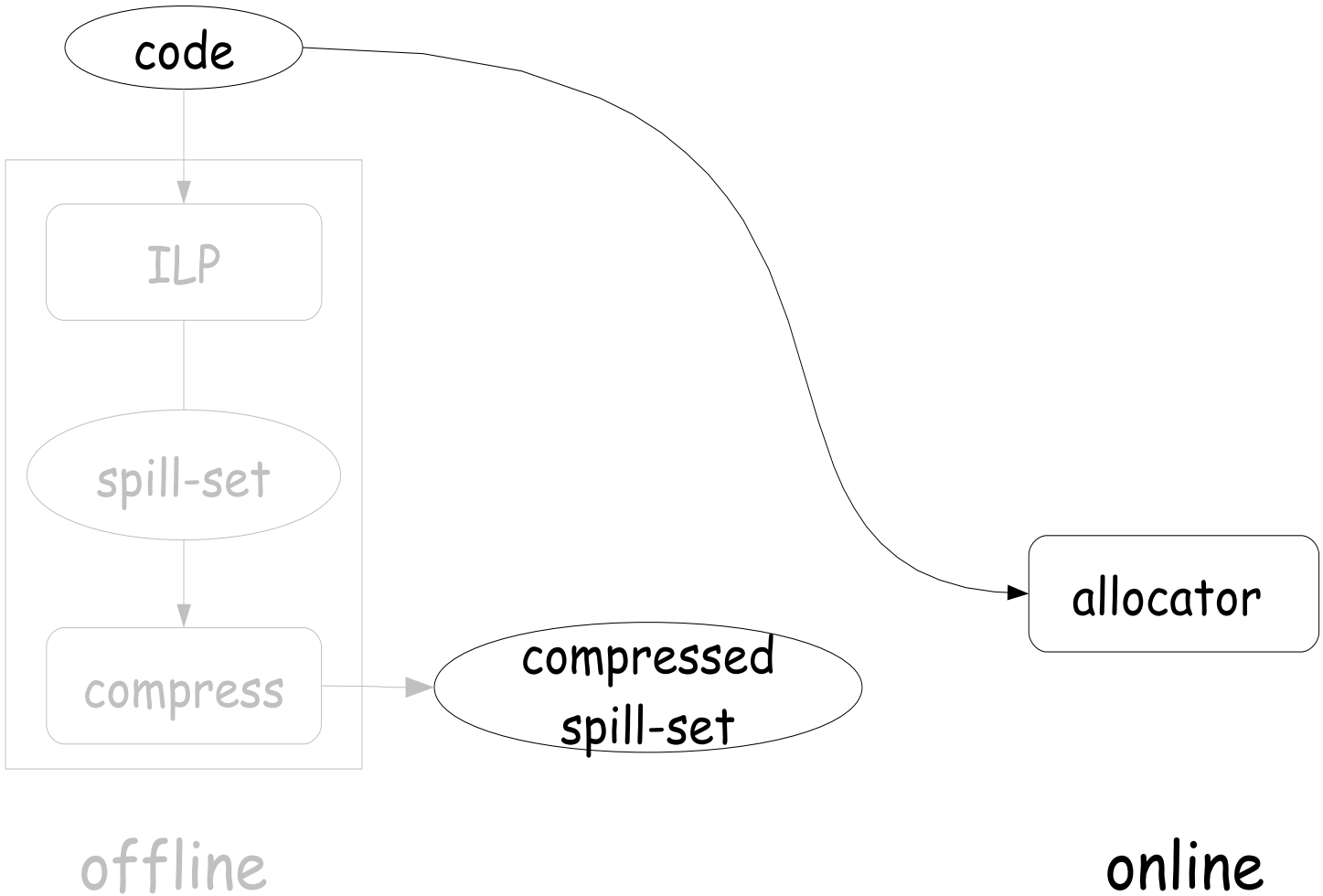
STAGES OF SPLIT REGISTER ALLOCATION



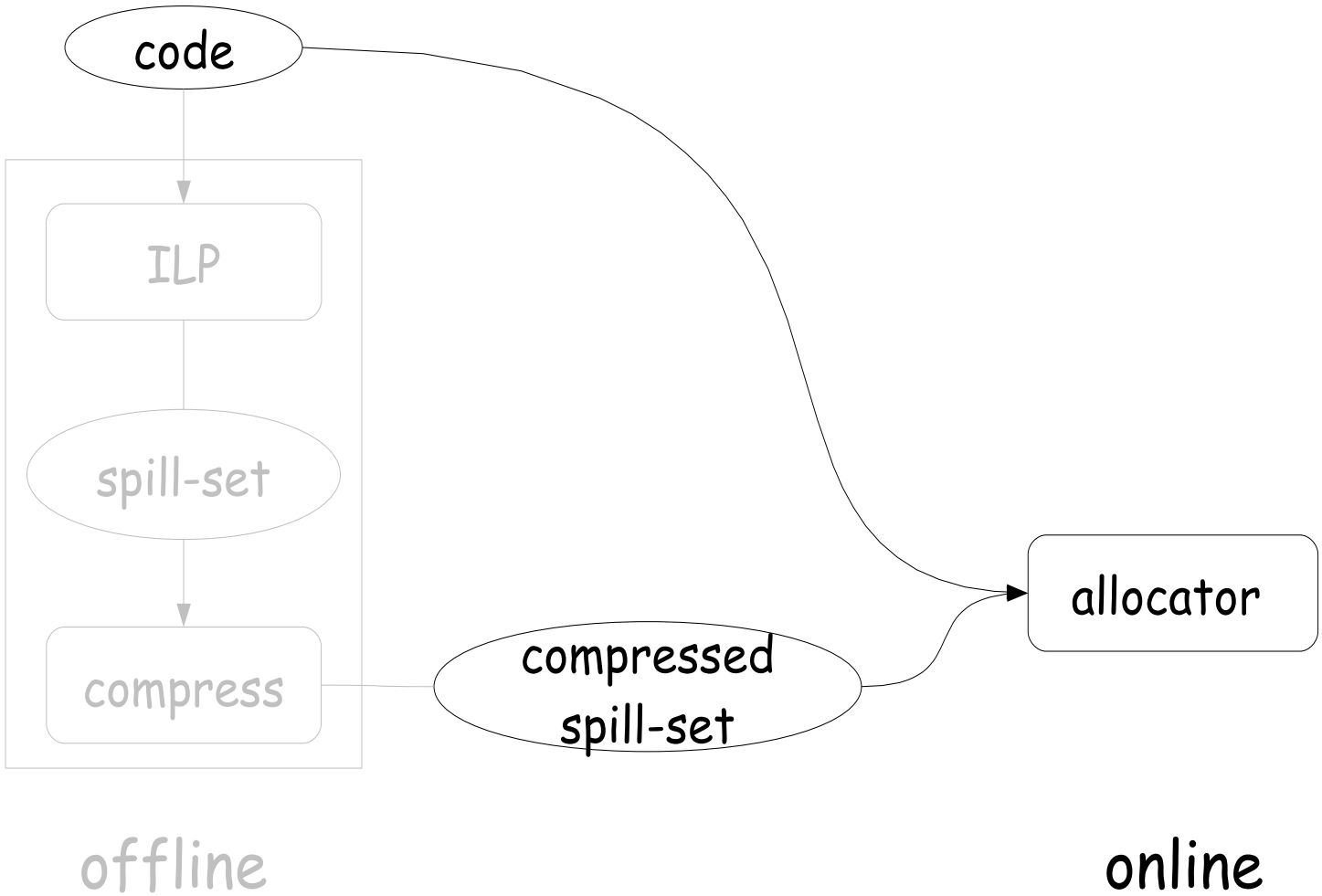
STAGES OF SPLIT REGISTER ALLOCATION



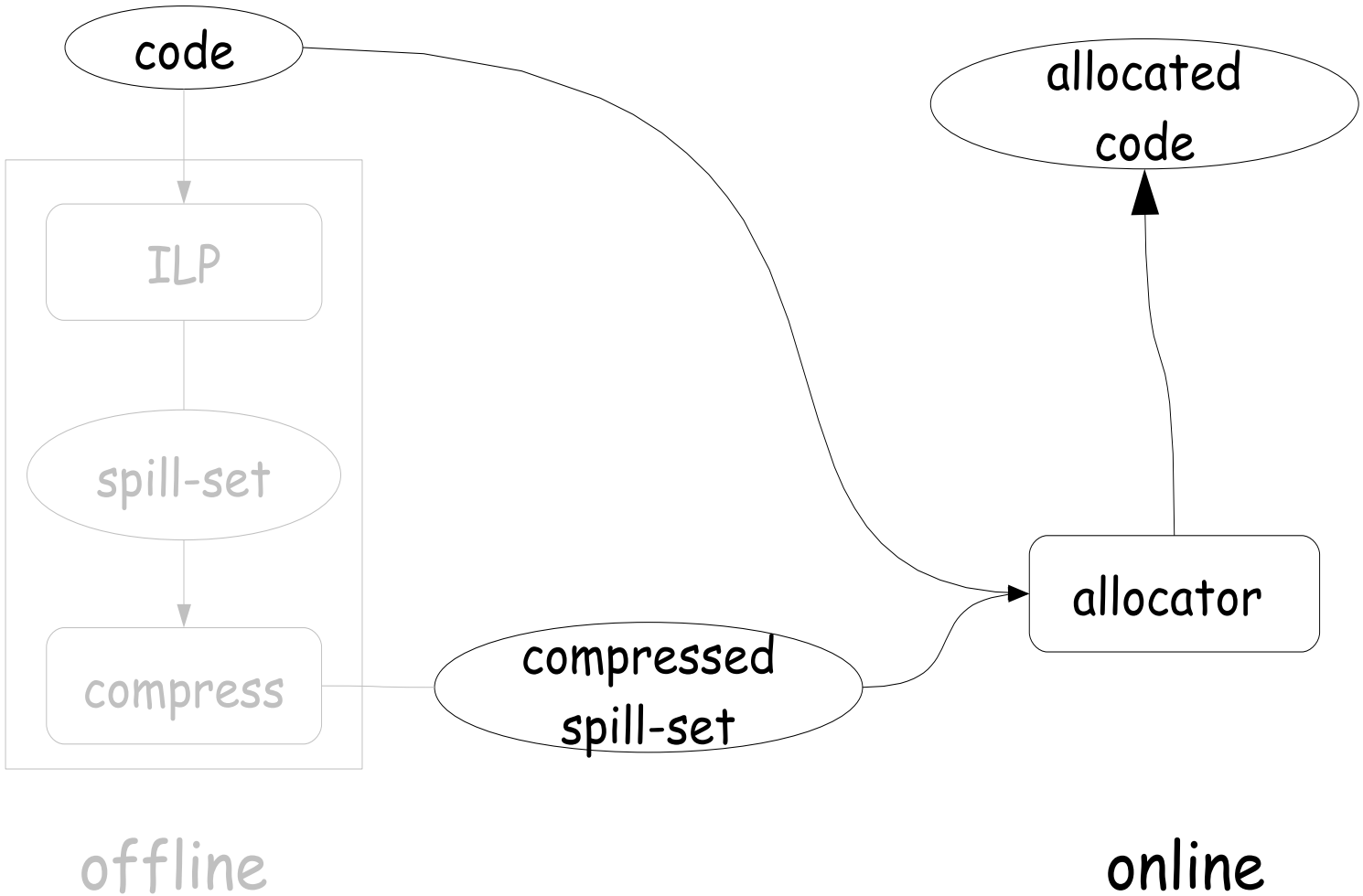
STAGES OF SPLIT REGISTER ALLOCATION



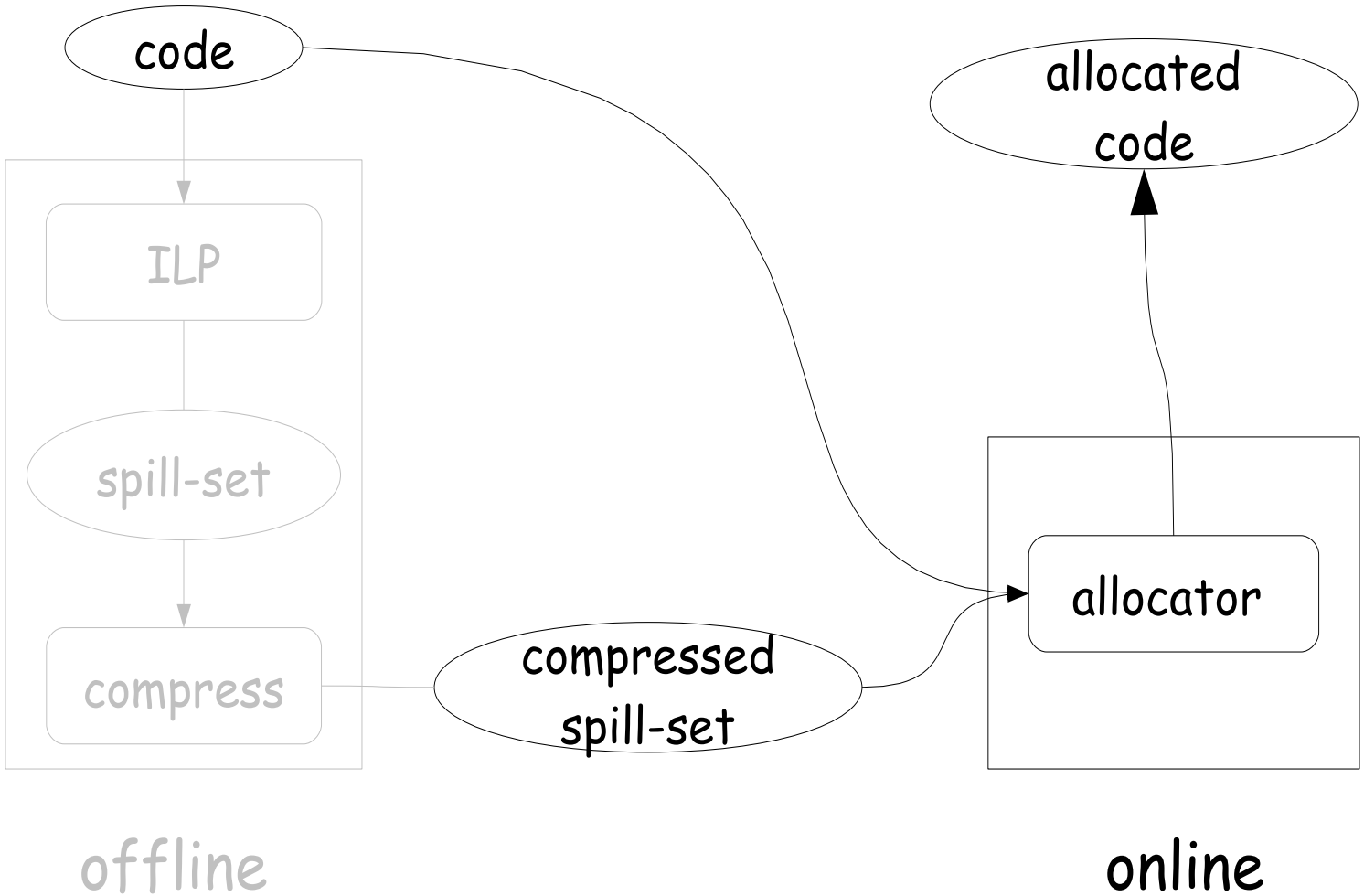
STAGES OF SPLIT REGISTER ALLOCATION



STAGES OF SPLIT REGISTER ALLOCATION



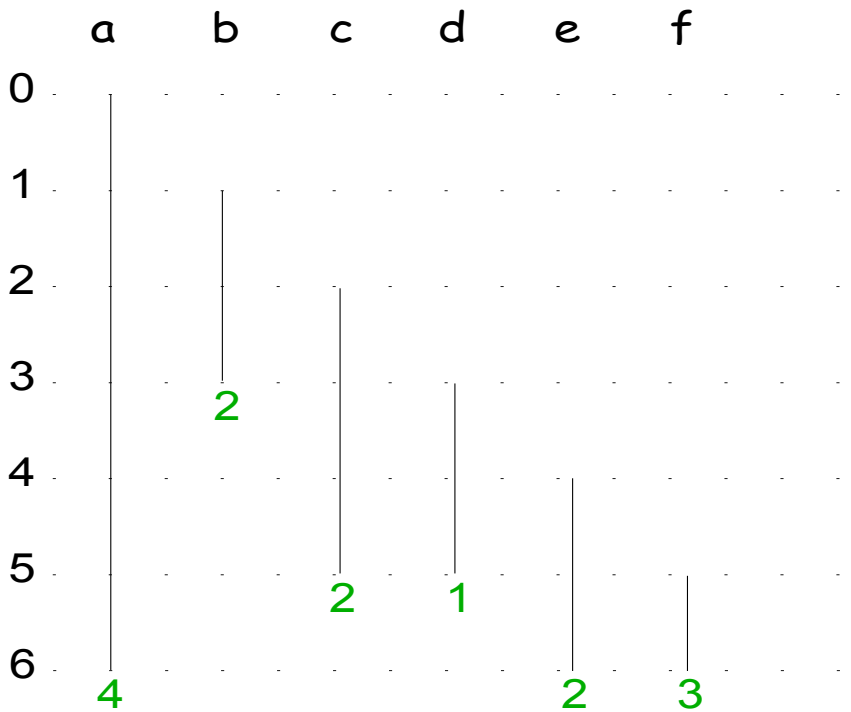
STAGES OF SPLIT REGISTER ALLOCATION



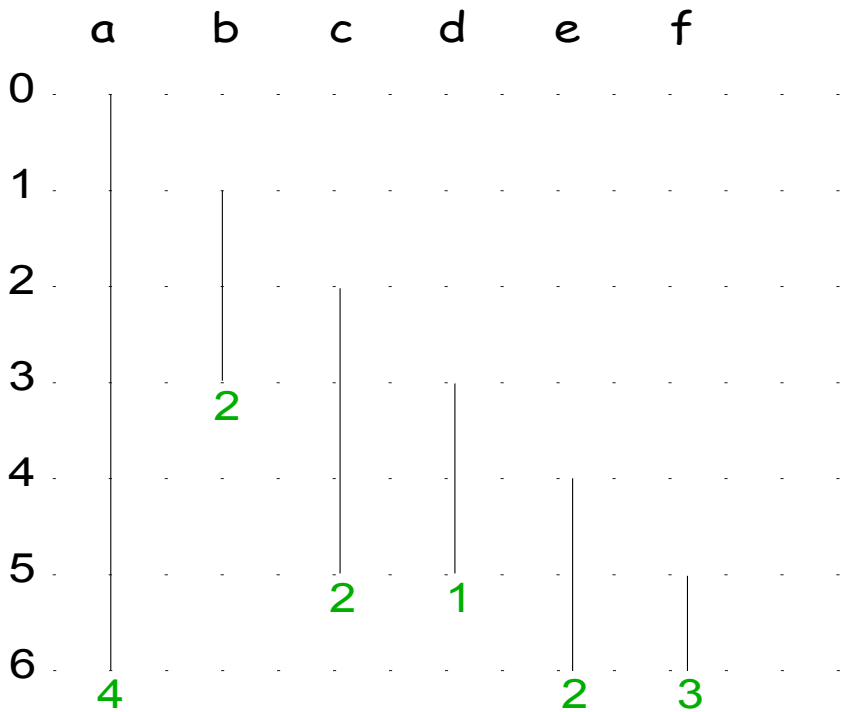
Compression

- Spill-Set produced by ILP can be used as annotation
- But we want to reduce the annotation size
  - Run the online allocator
  - Drop from the annotation any spilled variable that would be found by the online allocator

Compression Algorithm

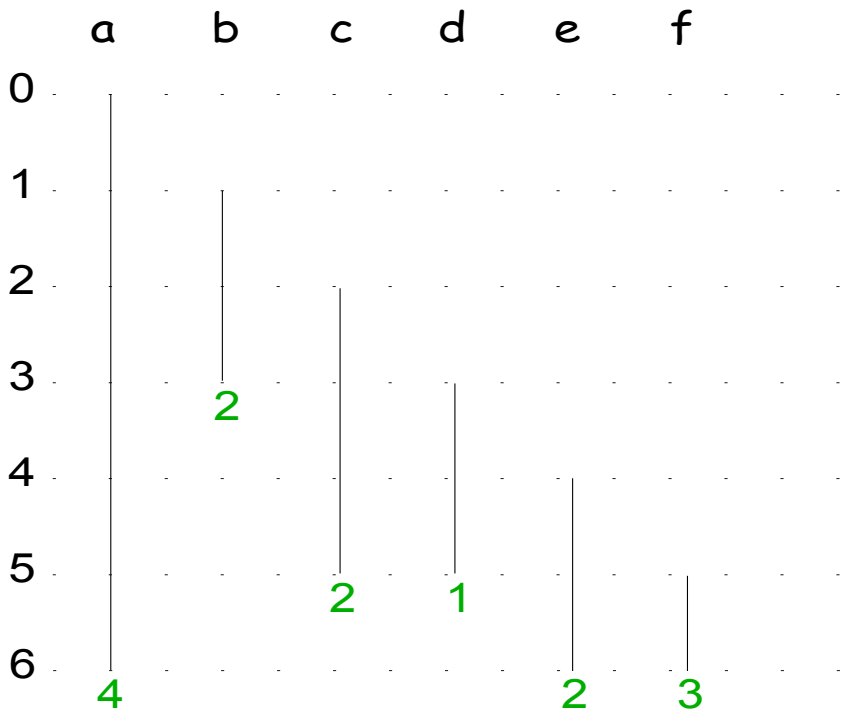


COMPRESSION ALGORITHM



2 available registers

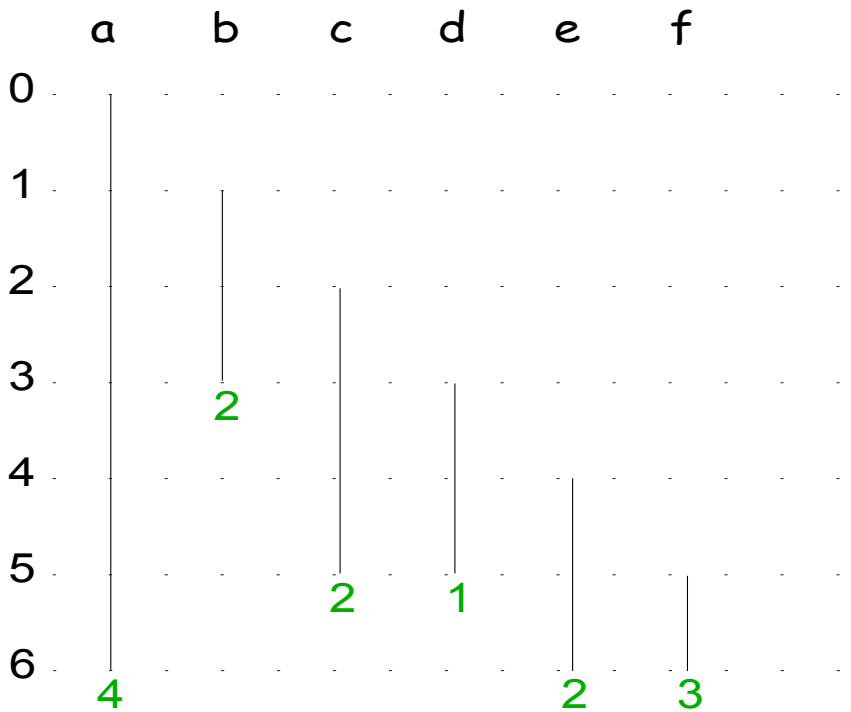
COMPRESSION ALGORITHM



2 available registers

Optimal spill-set: {c,e}

COMPRESSION ALGORITHM

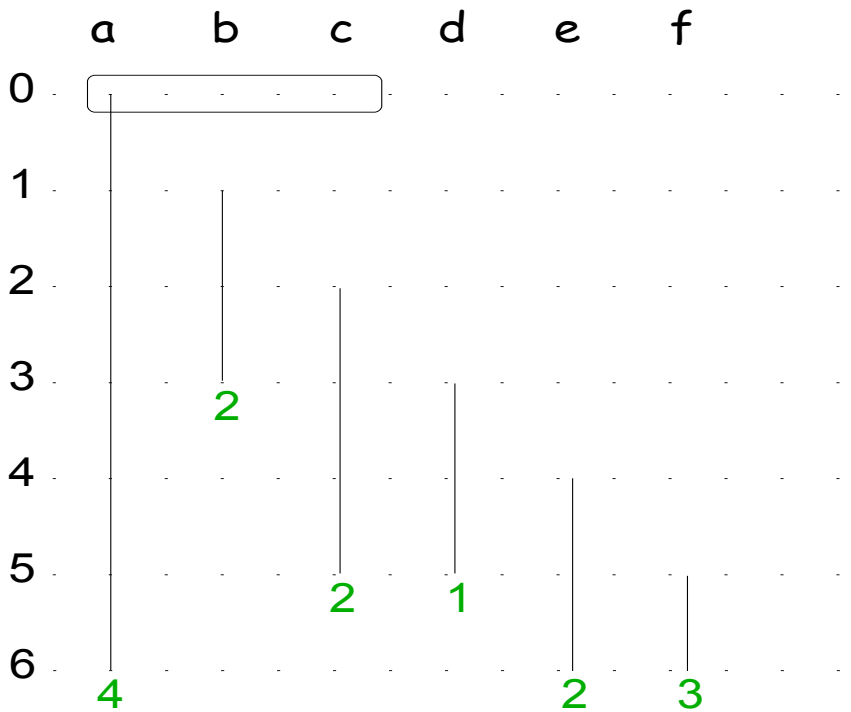


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

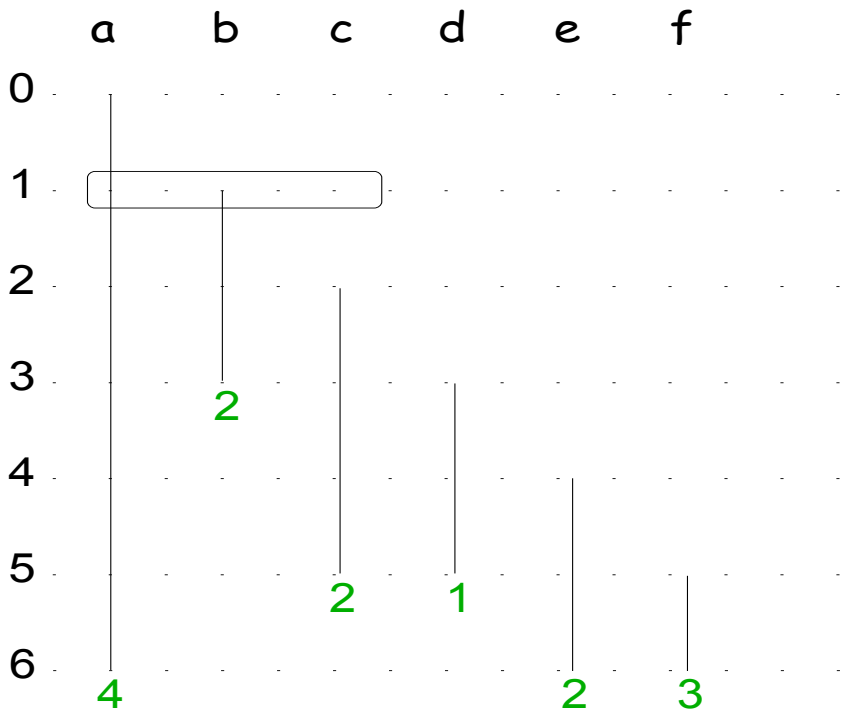


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

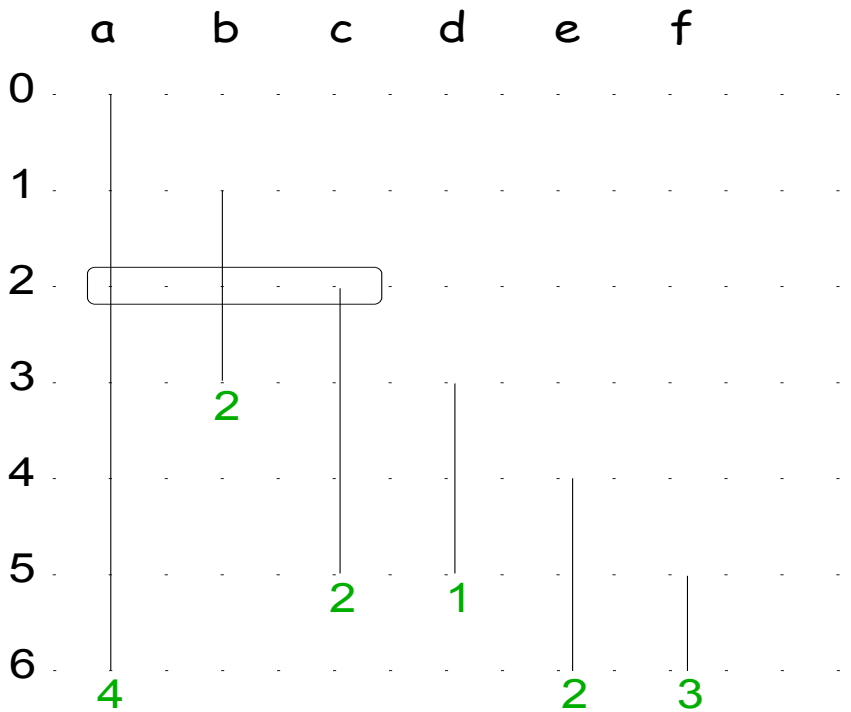


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

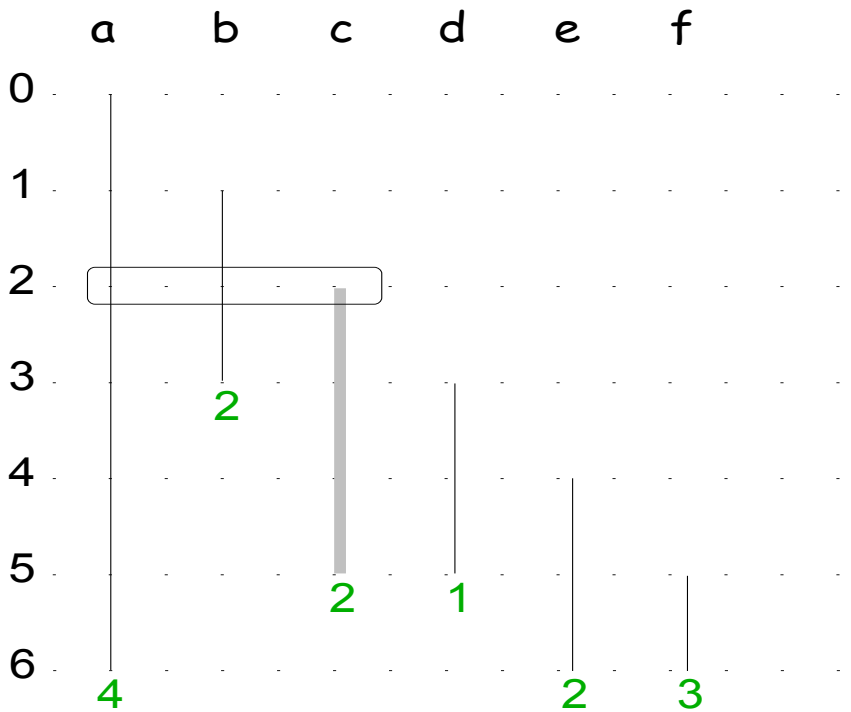


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

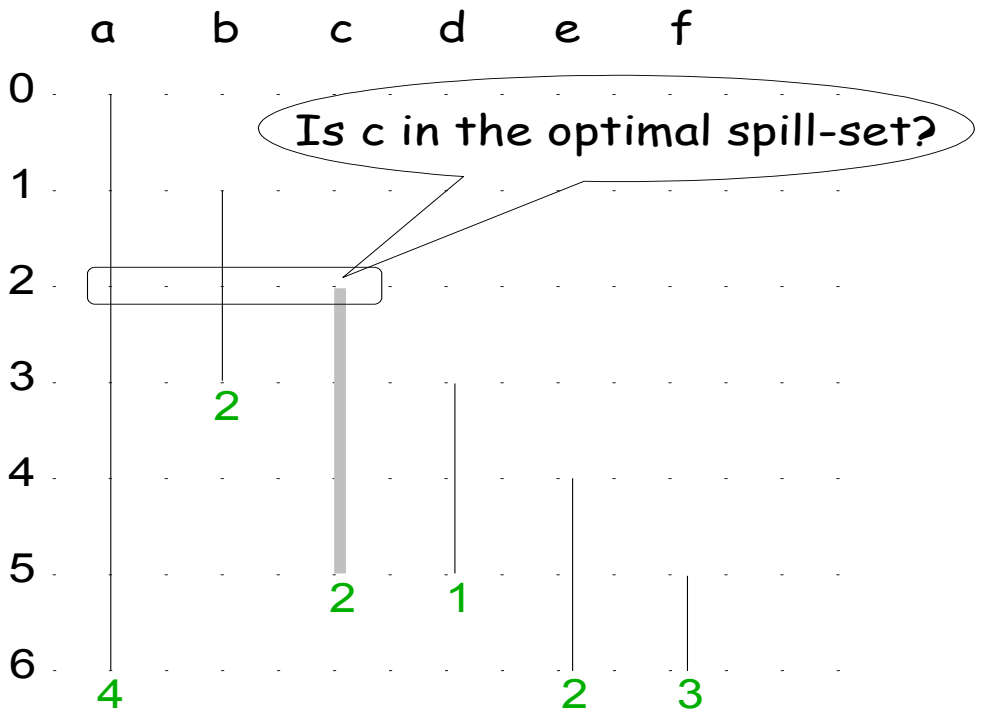


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

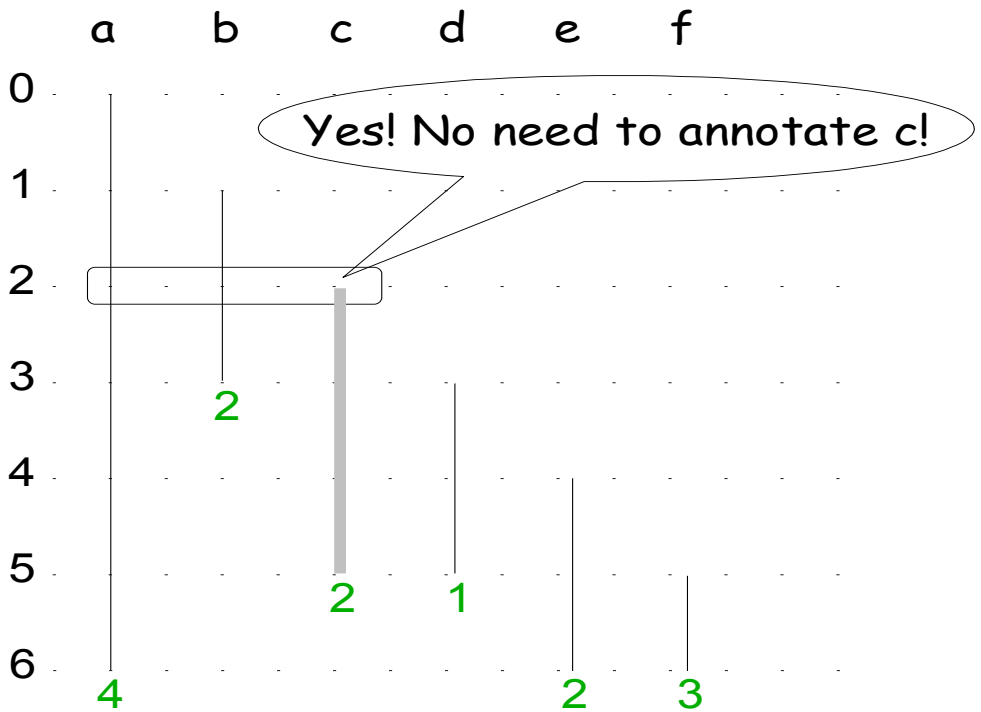


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

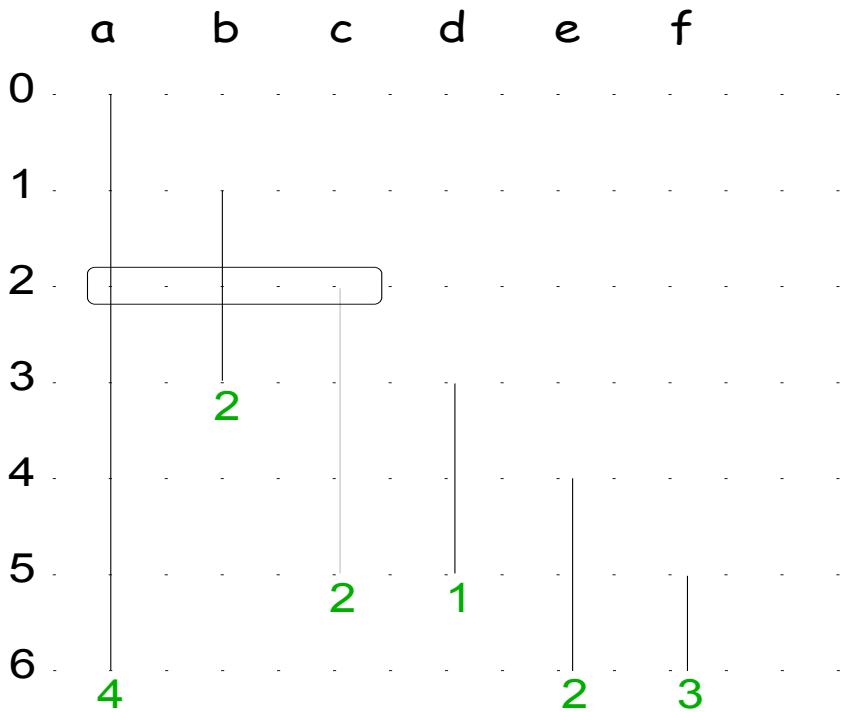


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

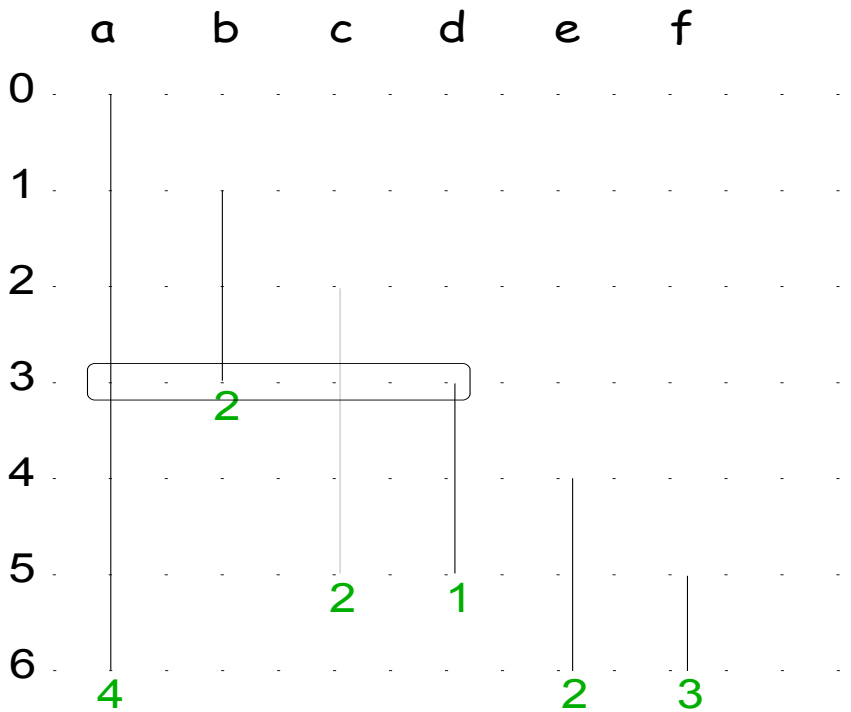


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

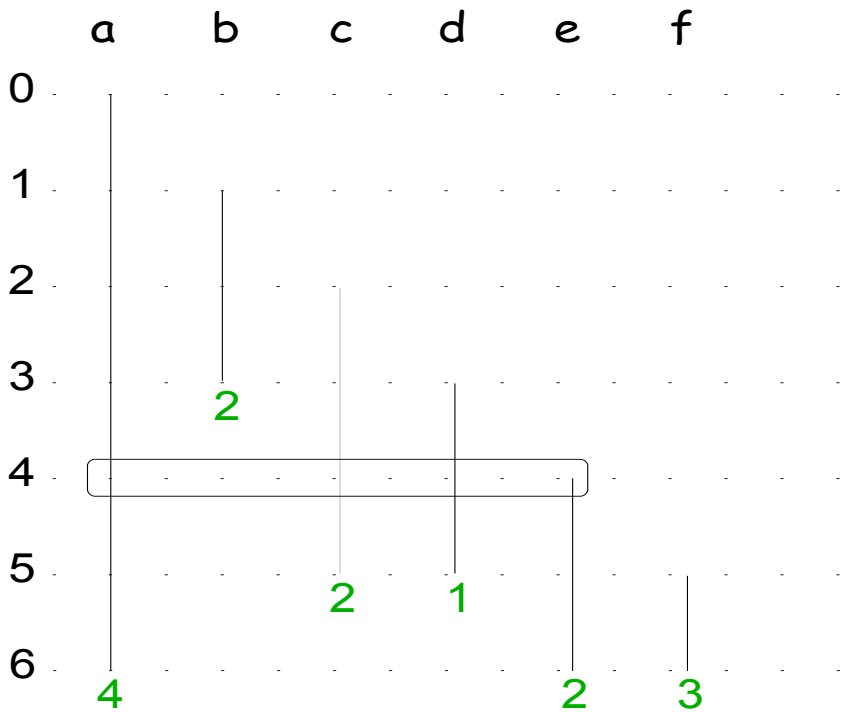


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

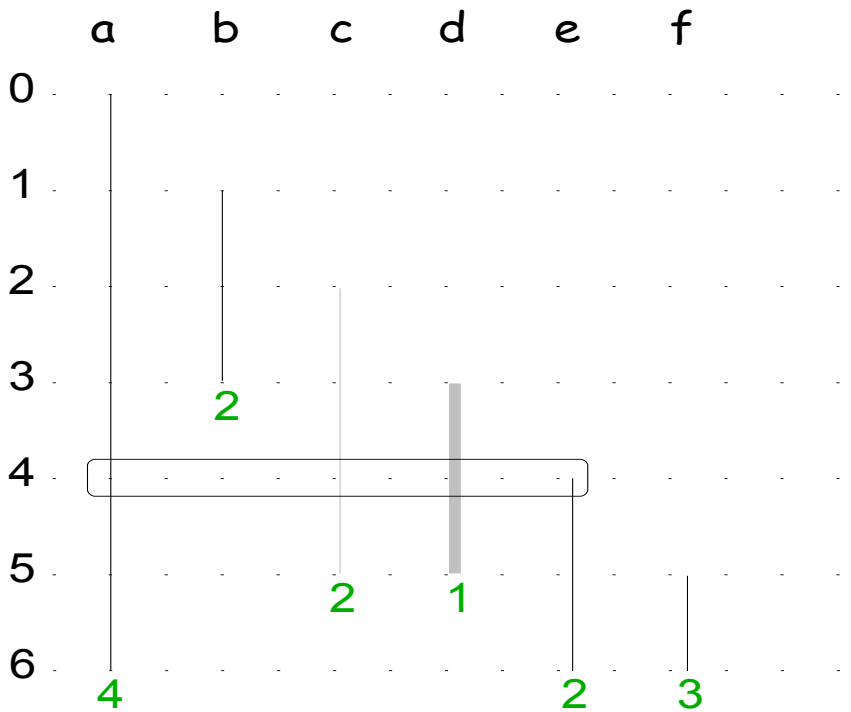


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

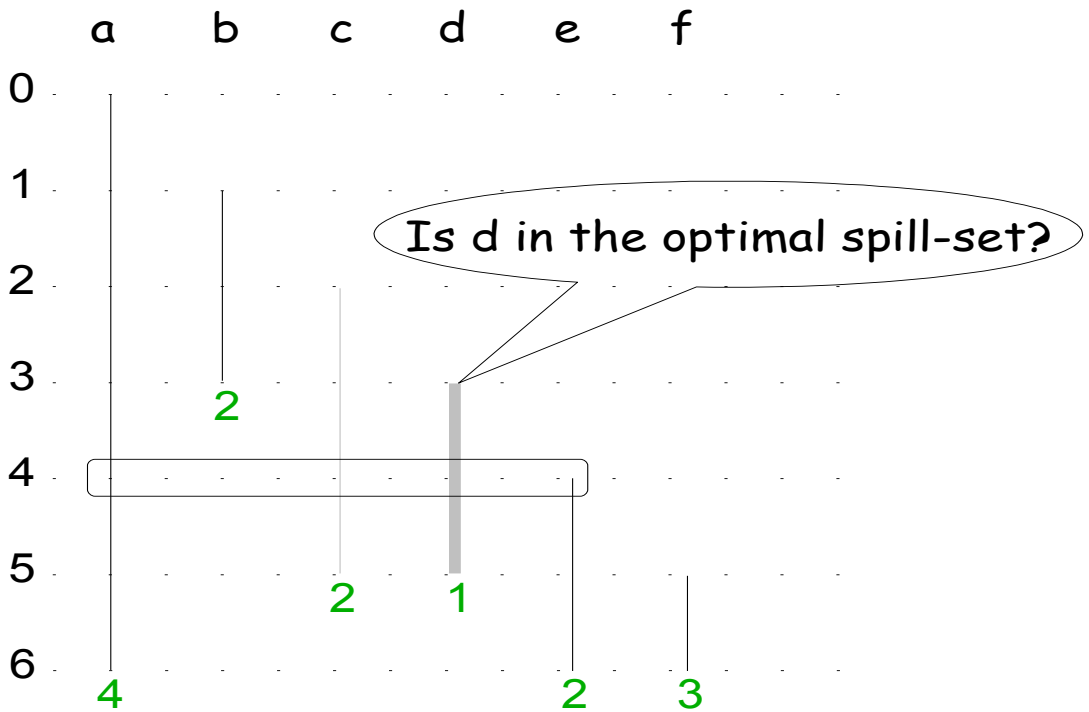


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

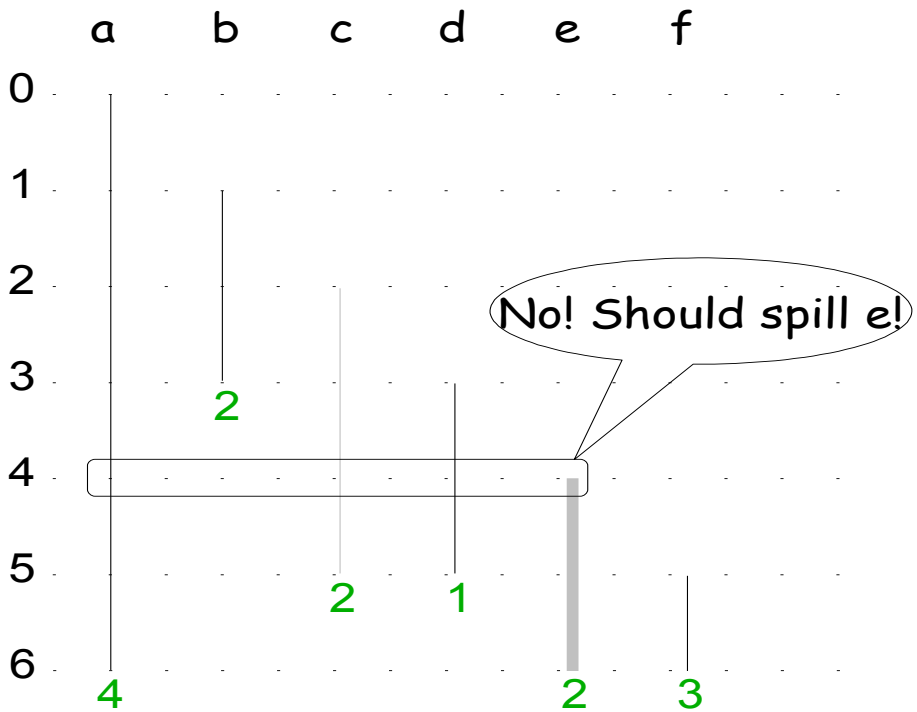


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

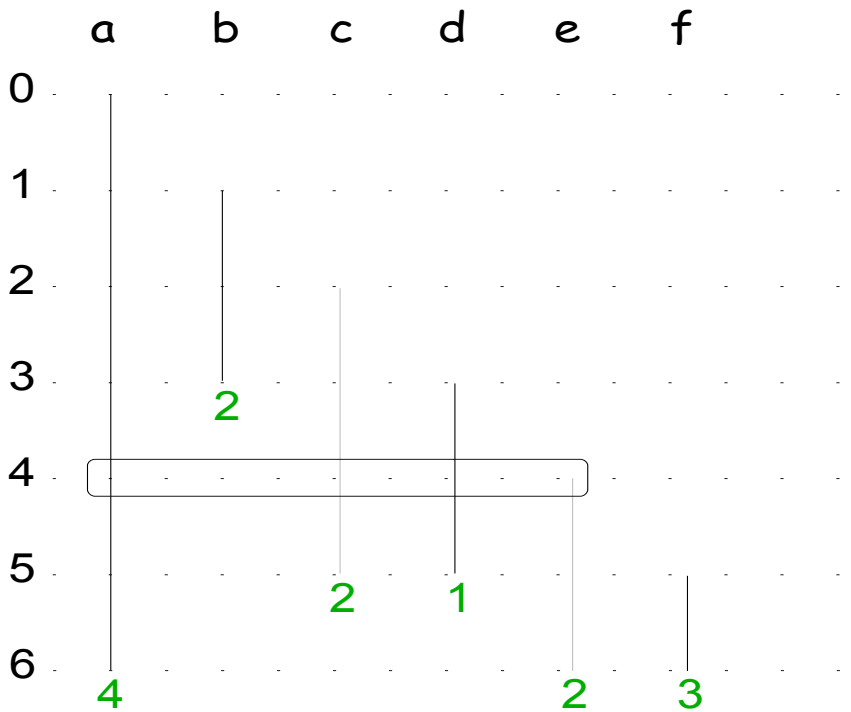


2 available registers

Optimal spill-set: {c,e}

Annotated spills:

COMPRESSION ALGORITHM

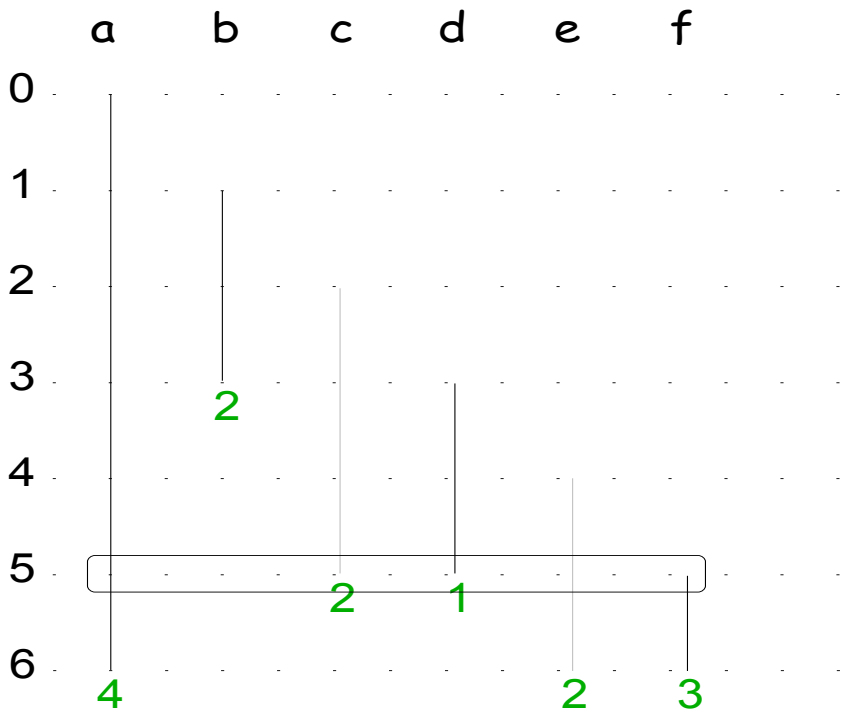


2 available registers

Optimal spill-set: {c,e}

Annotated spills: {e}

COMPRESSION ALGORITHM

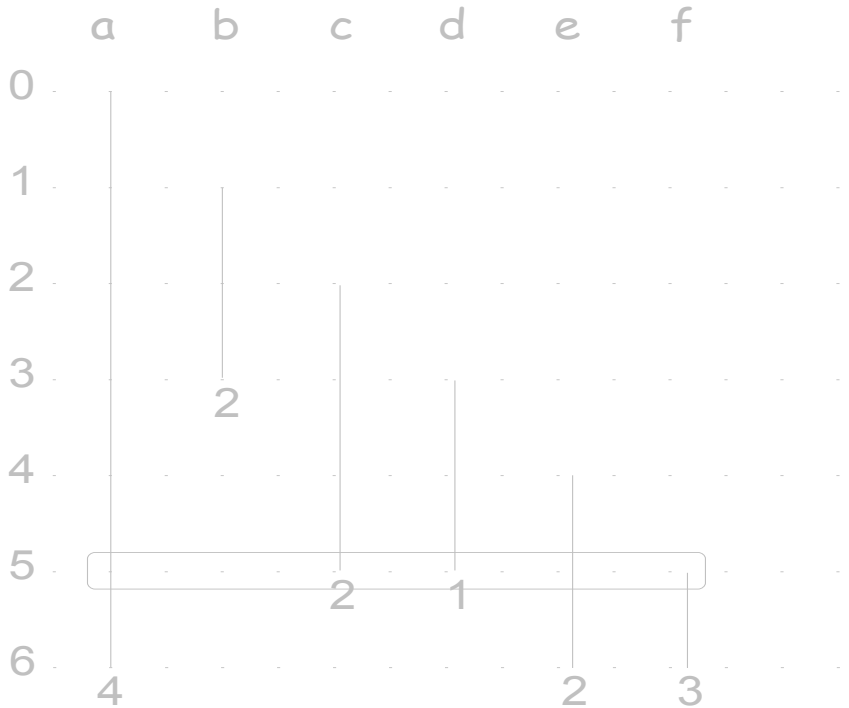


2 available registers

Optimal spill-set: {c,e}

Annotated spills: {e}

COMPRESSION ALGORITHM



2 available registers

Optimal spill-set: {c,e}

Annotated spills: {e}

Experiments: Framework

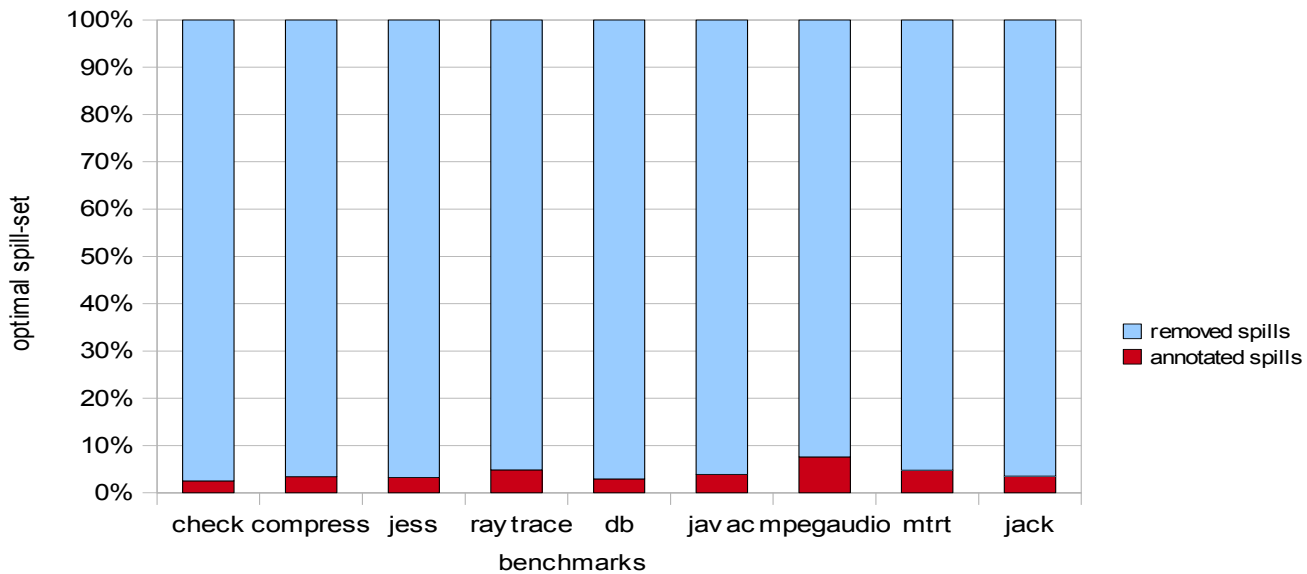
Framework:

- JikesRvm 3.0.1
- CPLEX
- SPEC JVM98 benchmarks

Experiments: Annotation

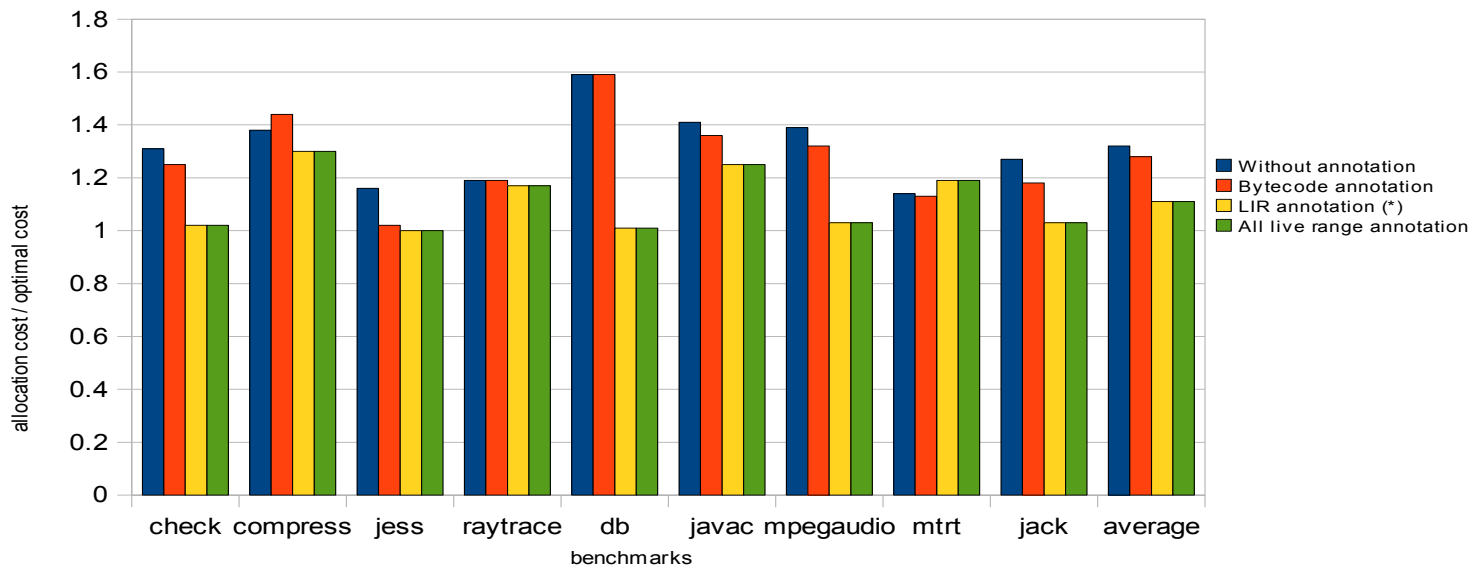
- Preserving the information collected in the offline stage requires at most 0.26% of the live ranges to be annotated

EXPERIMENTS: COMPRESSION



compression rate

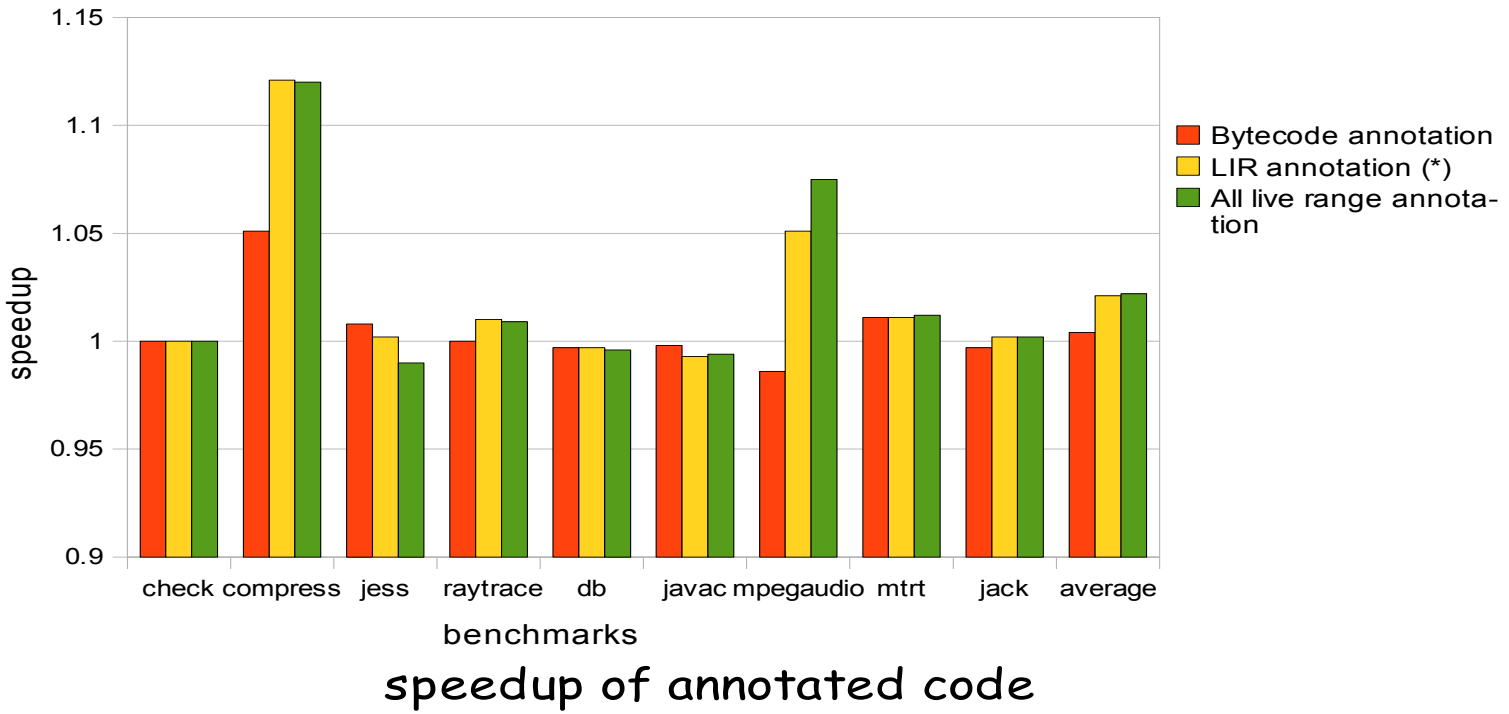
EXPERIMENTS: ALLOCATION COST



split compilation cost w.r.t. optimal cost

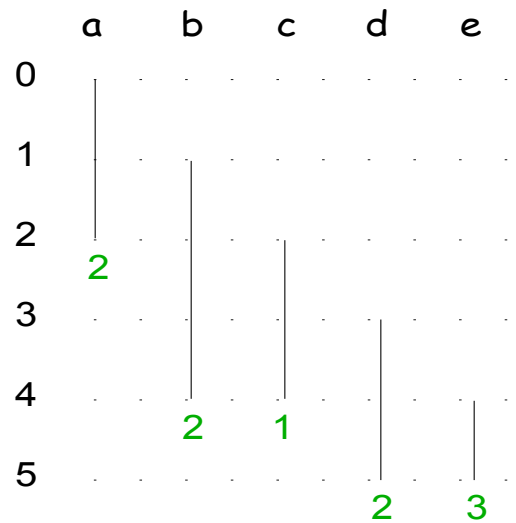
lower is better

EXPERIMENTS: SPEEDUPS

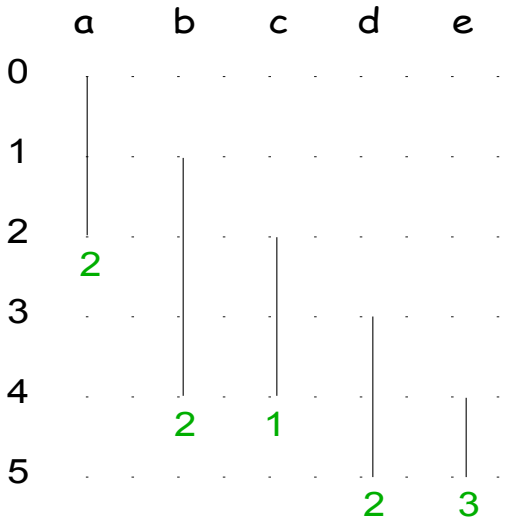


speedup of annotated code

PORTABILITY IN REGISTER COUNTS 1/2

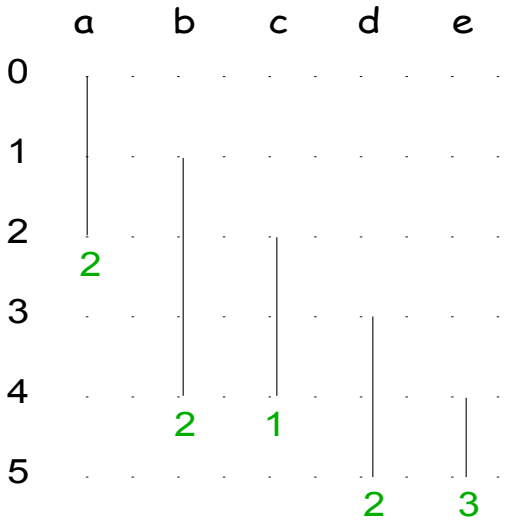


PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

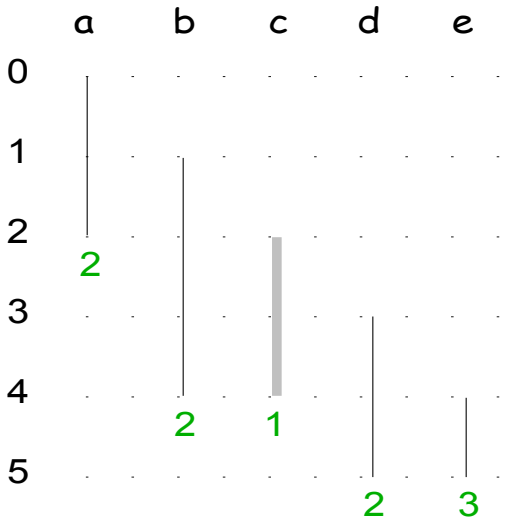
PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

Optimal spill-set =

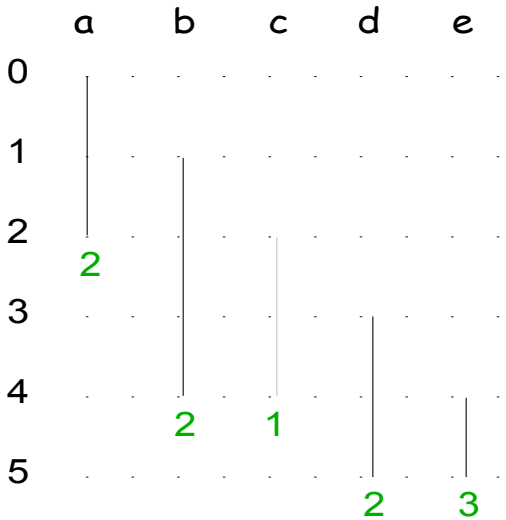
PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

Optimal spill-set =

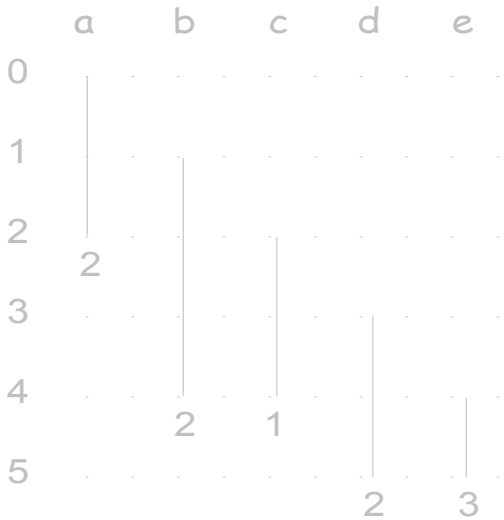
PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

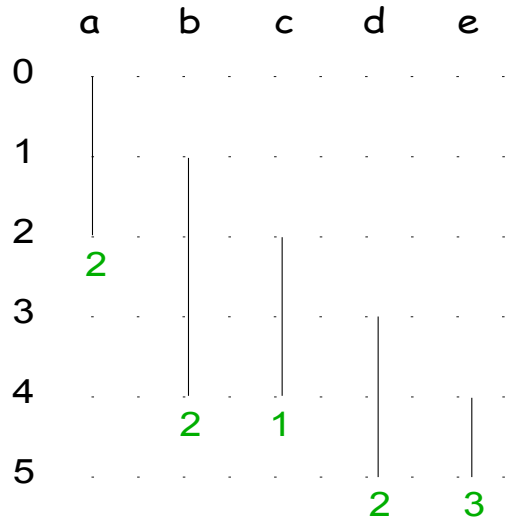
Optimal spill-set = {c}

PORTABILITY IN REGISTER COUNTS 1/2

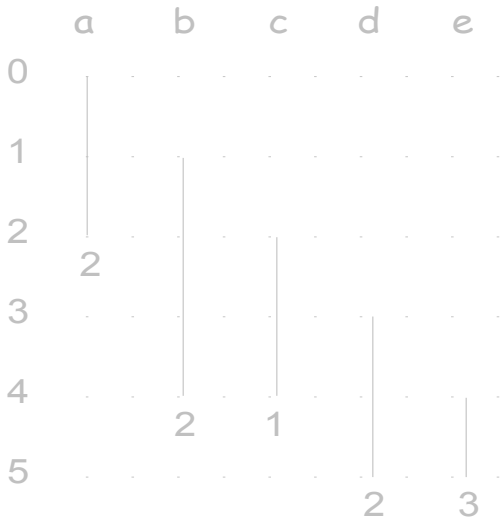


2 available registers

Optimal spill-set = {c}

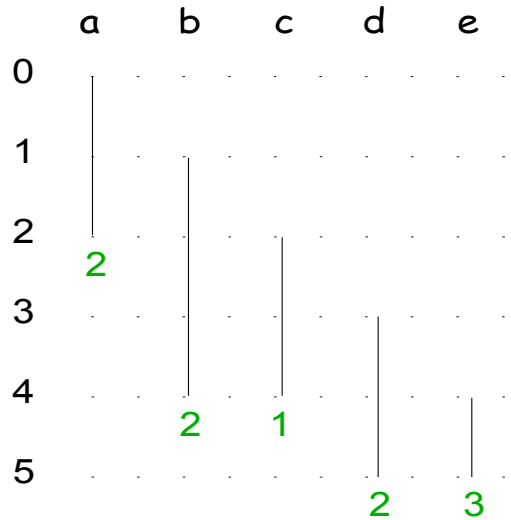


PORTABILITY IN REGISTER COUNTS 1/2



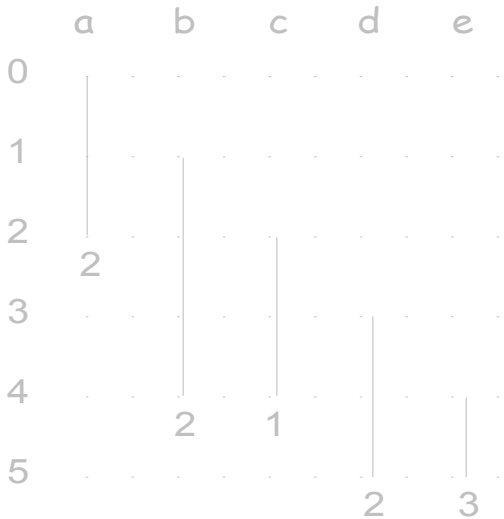
2 available registers

Optimal spill-set = {c}



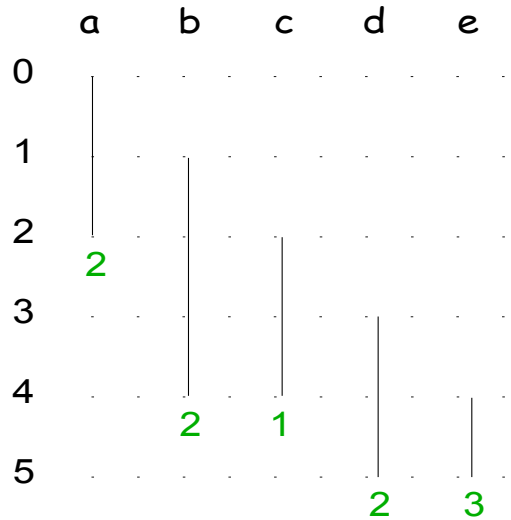
1 available register

PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

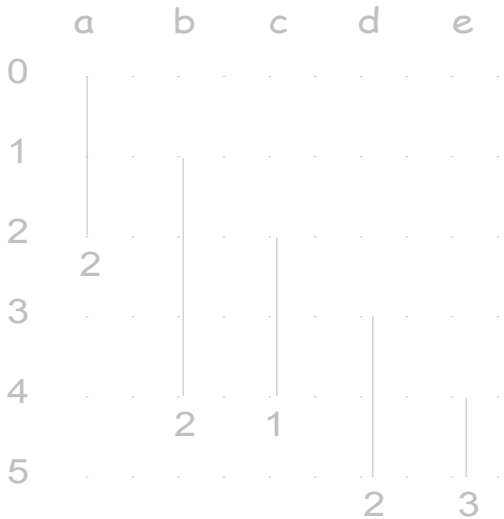
Optimal spill-set = {c}



1 available register

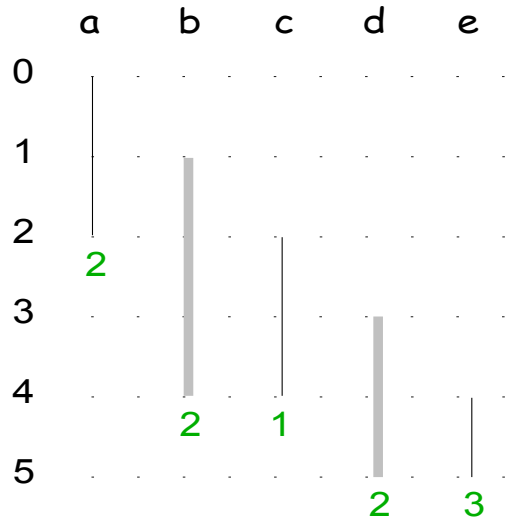
Optimal spill-set =

PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

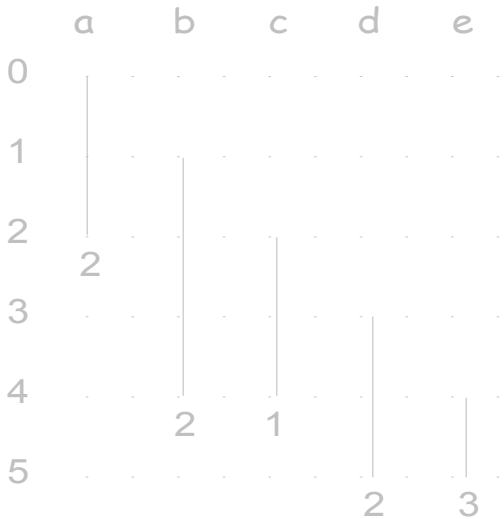
Optimal spill-set = {c}



1 available register

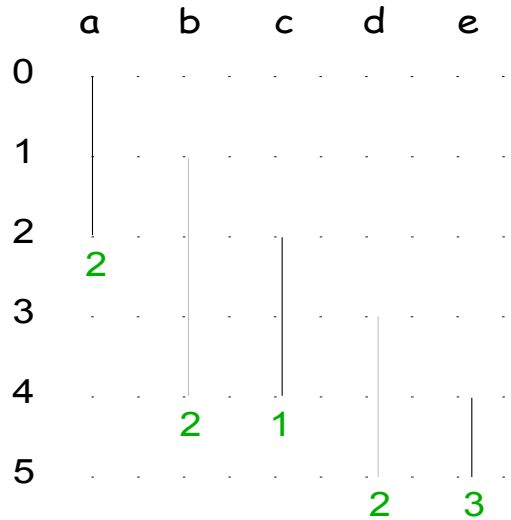
Optimal spill-set =

PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

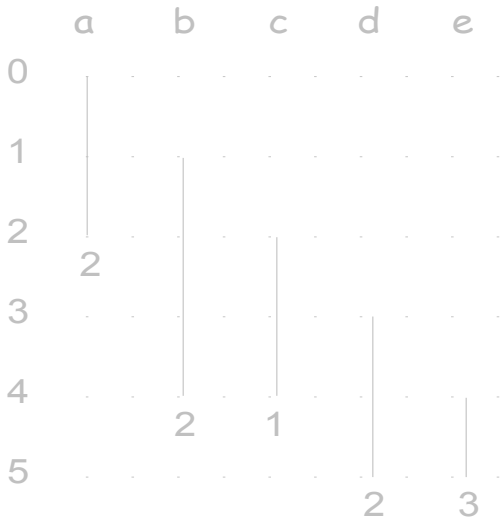
Optimal spill-set = {c}



1 available register

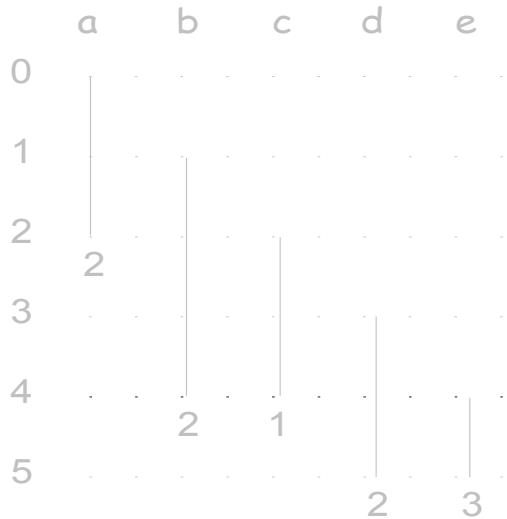
Optimal spill-set = {b,d}

PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

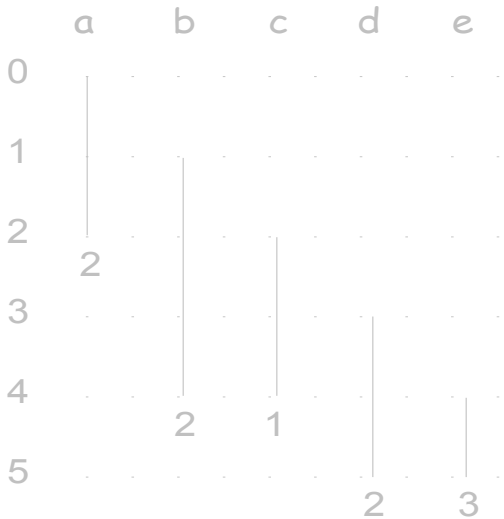
Optimal spill-set = {c}



1 available register

Optimal spill-set = {b,d}

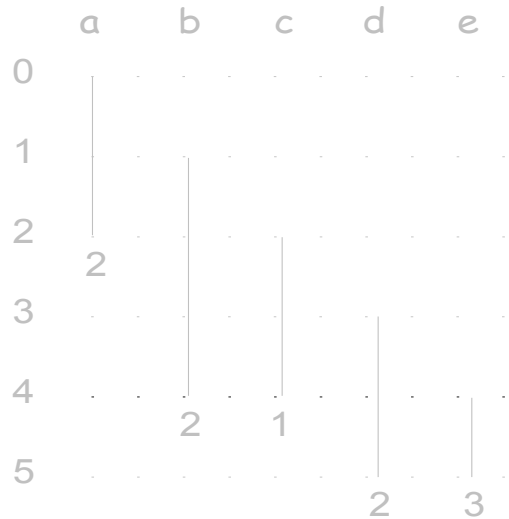
PORTABILITY IN REGISTER COUNTS 1/2



2 available registers

Optimal spill-set = {c}

$\{c\} \not\subset \{b,d\}$



1 available register

Optimal spill-set = {b,d}

Portability in register counts 2/2

Experimental study

- Varying the number of registers from 2
- . . . to the maximal number where spilling is needed

Result

- Inclusion property holds for 99.83% of the SPEC JVM98's methods

Separate compilation

What is the impact of calling an annotated code that will be :

Separate compilation

What is the impact of calling an annotated code that will be :

- later inlined

Separate compilation

What is the impact of calling an annotated code that will be :

- later inlined
- later called from a different context

Separate compilation

What is the impact of calling an annotated code that will be :

- later inlined
- later called from a different context

1. Most frequent calling context

Separate compilation

What is the impact of calling an annotated code that will be :

- later inlined
- later called from a different context

1. Most frequent calling context
2. Multi-versionning depending on the calling context

Conclusion

Recipe for split compilation success:

1. What can be done offline?
2. What should be done online?
3. Leverage fundamental algorithmic results to coordinate the two stages
   - live range splitting
   - maxlive colorability criterion

References


[Krintz'01]: C. Krintz, B. Calder.
Using annotations to reduce dynamic optimization time.


[Jones'00]: J. Jones, S. N. Kamin.
Annotating java class files with virtual registers for performance.


[Chaitin'81]: G. J. Chaitin, et al.
Register Allocation via coloring.


[Briggs'94]: P. Briggs, K. D. Cooper, L. Torczon.
Improvements to graph coloring register allocation.


[Appel'01]: A. W. Appel, L. George.
Optimal spilling for CISC machines with few registers.


[Hack'06]: S. Hack, D. Grund, G. Goos.
Register allocation for program s in SSA-form.


[Bouchez'06]: F. Bouchez, A. Darte, C. Guillon, F Rastello.
Register allocation: What does the NP-completeness proof of Chaitin et al. really prove?