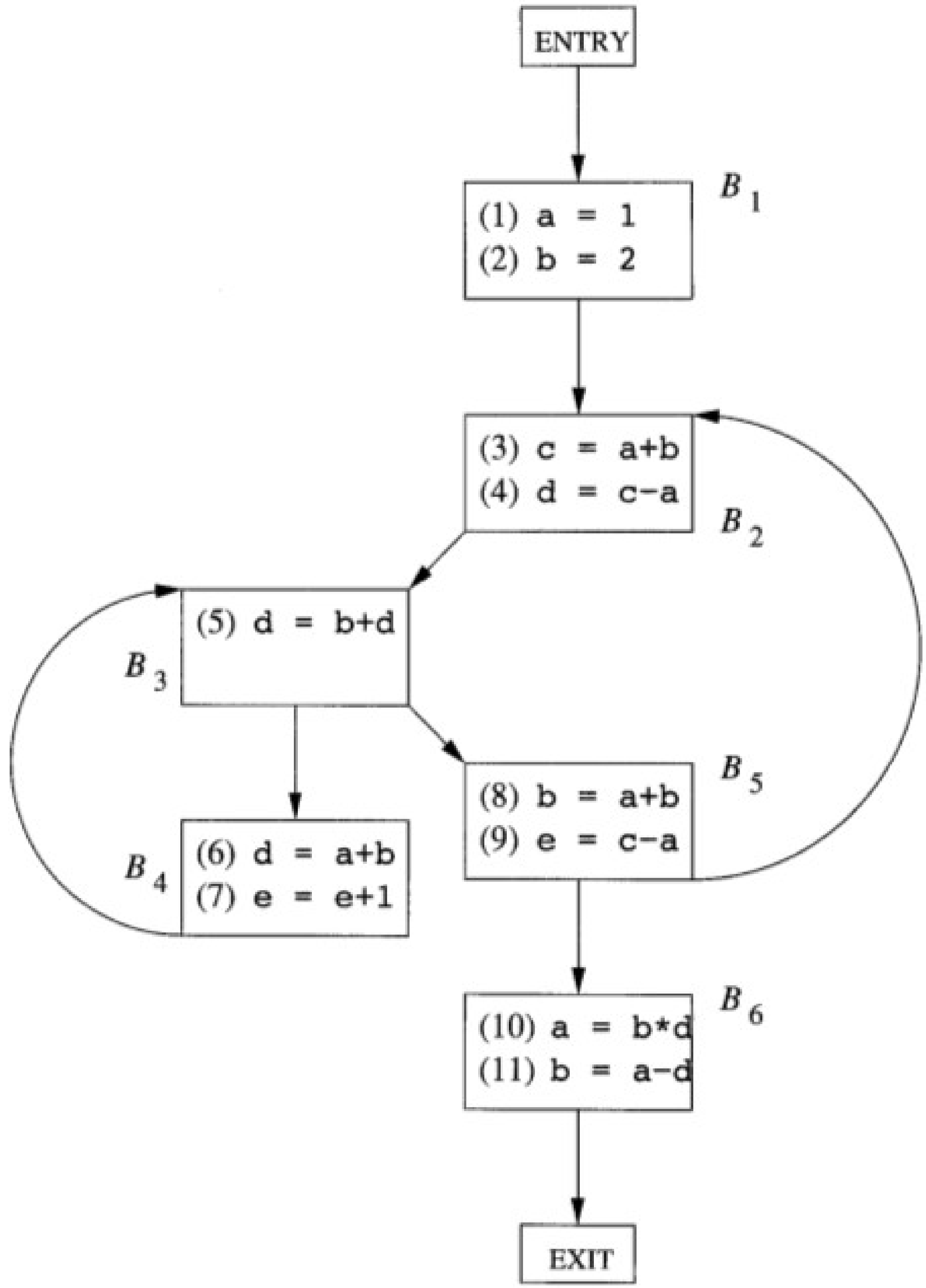
MC910 – Construção de Compiladores

Exercícios – Análise de Longevidade / Alocação de Registradores

Dragon Book

9.2.3 For the flow graph of Fig. 9.10, compute the *def, use,* **IN** and **OUT** sets for live variable analysis.



Appel

10.1 Perform flow analysis on the program of Exercise 8.6:

1. Draw the control-flow graph.
2. Calculate live-in and live-out at each statement.
3. Construct the register interference graph.

1.	$m \leftarrow 0$
2.	$v \leftarrow 0$
3.	if $v \geq n$ goto 15
4.	$r \leftarrow v$
5.	$s \leftarrow 0$
6.	if $r < n$ goto 9
7.	$v \leftarrow v + 1$
8.	goto 3
9.	$x \leftarrow M[r]$
10.	$s \leftarrow s + x$
11.	if $s \leq m$ goto 13
12.	$m \leftarrow s$
13.	$r \leftarrow r + 1$
14.	goto 6
15.	return m

10.5 The DEC Alpha architecture places the following restrictions on floating-point instructions, for programs that wish to recover from arithmetic exceptions:

1. Within a basic block (actually, in any sequence of instructions not separated by a trap-barrier instruction), no two instructions should write to the same destination register.

2. A source register of an instruction cannot be the same as the destination register of that instruction or any later instruction in the basic block.

$r1 + r5 \rightarrow r4$; $r3 \times r2 \rightarrow r4$ violates rule 1.

$r1 + r5 \rightarrow r4$; $r4 \times r2 \rightarrow r1$ violates rule 2.

$r1 + r5 \rightarrow r3$; $r4 \times r2 \rightarrow r4$ violates rule 2.

$r1 + r5 \rightarrow r4$; $r4 \times r2 \rightarrow r6$ is OK.

Show how to express these restrictions in the register interference graph.

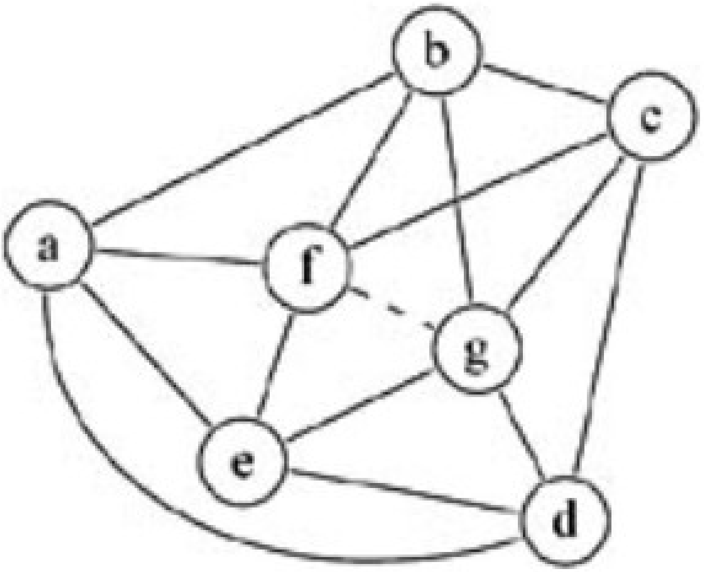
11.2 The table below represents a register-interference graph. Nodes 1−6 are precolored (with colors 1−6), and nodes A−H are ordinary (non-precolored). Every pair of precolored nodes interferes, and each ordinary node interferes with nodes where there is an × in the table.

Assume that register allocation must be done for an 8-register machine.

	1	2	3	4	5	6	A	B	C	D	E	F	G	H
A	x	x	x	x	x	x								
B	x		x	x	x	x								
C			x	x	x	x				x	x	x	x	x
D	x		x	x	x				x		x	x	x	x
E	x		x		x	x			x	x		x	x	x
F	x		x	x		x			x	x	x		x	x
G									x	x	x	x		
H	x			x	x	x			x	x	x	x		

a. Ignoring the MOVE instructions, and without using the *coalesce* heuristic, color this graph using *simplify* and *spill*. Record the sequence (stack) of *simplify* and *potential-spill* decisions, show which potential spills become actual spills, and show the coloring that results.

11.3 *Conservative coalescing* is so called because it will not introduce any (potential) spills. But can it avoid spills? Consider this graph, where the solid edges represent interferences and the dashed edge represents a MOVE:



a. 4-color the graph without coalescing. Show the *select-stack*, indicating the order in which you removed nodes. Is there a potential spill? Is there an actual spill?