Lecture 7: Hyperblocks

COS 598C – Advanced Compilers

Prof. David August
Department of Computer Science
Princeton University

FROM LAST TIME: Control Dependences

- Recall
  - Post dominator – BBX is post dominated by BBY if every path from BBX to EXIT contains BBY
  - Immediate post dominator – First breadth first successor of a block that is a post dominator
- Control dependence – BBY is control dependent on BBX iff
  1. There exists a directed path P from BBX to BBY with all BBZ in P (excluding BBX and BBY) post dominated by BBY
  2. BBX is not post dominated by BBY
- In English,
  - A BB is control dependent on the closest BB(s) that determine(s) its execution
  - Its actually not a BB, it's a control flow edge coming out of a BB

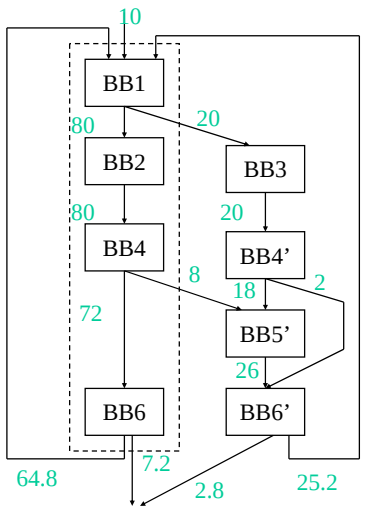
Class Problem from Last Time

```
if (a > 0) {
   r = t + s
   if (b > 0 || c > 0)
      u = v + 1
   else if (d > 0)
      x = y + 1
   else
      z = z + 1
}
```

a. Draw the CFG
b. Compute CD
c. If-convert the code

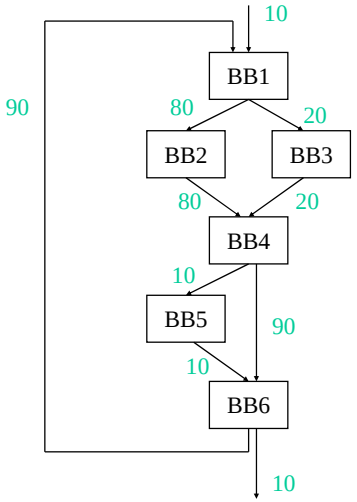
Region Formation + If-conversion

- Control flow representation
  - branches
  - predicated operations
- If-conversion not all all or nothing deal
  - Often bad to apply in blanket mode
  - Selectively apply
- Regions
  - Extend a superblock to contain if-converted code
  - Convert off-trace transitions to on-trace
  - A hyperblock is born
  - Superblock is a special case HB where all guarding predicates are True



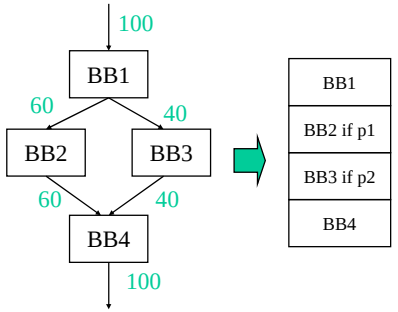
When to Apply If-conversion

- Positives
  - Remove branch
    - No disruption to sequential fetch
    - No prediction or mispredict
    - No use of branch resource
  - Increase potential for operation overlap
  - Enable more aggressive compiler xforms
    - Software pipelining
    - Height reduction
- Negatives
  - Max or Sum function applied when overlap
    - Resource usage
    - Dependence height
    - Hazard presence
  - Executing useless operations



Negative 1: Resource Usage

Resource usage is additive
for all BBs that are if-converted



Case 1: Each BB requires 3 resources
Assume processor has 2 resources

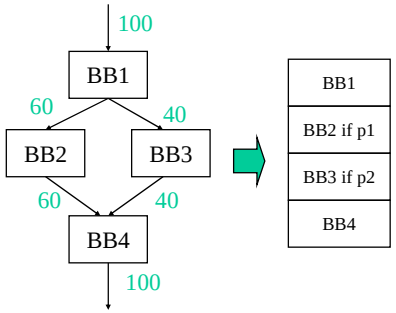
No IC: 1*3 + .6*3 + .4*3 + 1*3 = 9
9 / 2 = 4.5 = 5 cycles
IC: 1(3 + 3 + 3+ 3) = 12
12 / 2 = 6 cycles

Case 2: Each BB requires 3 resources
Assume processor has 6 resources

No IC: 1*3 + .6*3 + .4*3 + 1*3 = 9
9 / 6 = 1.5 = 2 cycles
IC: 1(3+3+3+3) = 12
12 / 6 = 2 cycles

Negative 2: Dependence Height

Dependence height is max of
for all BBs that are if-converted
(dep height = schedule length
with infinite resources)



Case 1: height(bb1) = 1, height(bb2) = 3
Height(bb3) = **9**, height(bb4) = 2

No IC: 1*1 + .6*3 + .4*9 + 1*2 = 8.4

IC: 1*1 + 1*MAX(3,9) + 1*3 = 13

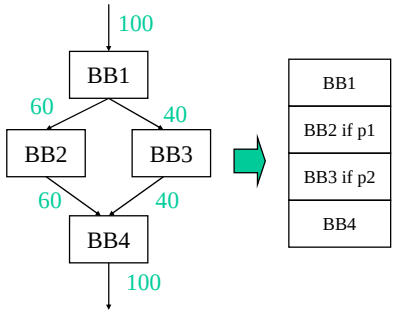
Case 2: height(bb1) = 1, height(bb2) = 3
Height(bb3) = **3**, height(bb4) = 2

No IC: 1*1 + .6*3 + .4*3 + 1*2 = 6

IC: 1*1 + 1*MAX(3,3) + 1*2 = 6

Negative 3: Hazard Presence

Hazard = operation that forces
the compiler to be conservative,
so limited reordering or optimization,
e.g., subroutine call, pointer store, …



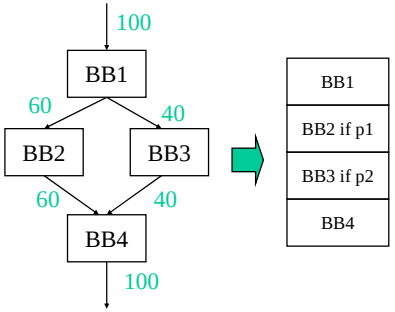
Case 1: Hazard in BB3

No IC : SB out of BB1, 2, 4, operations
In BB4 free to overlap with those in
BB1 and BB2

IC: operations in BB4 cannot overlap
With those in BB1 (BB2 ok)

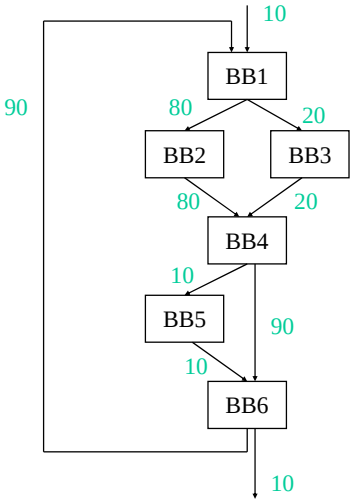
When To If-convert

- Resources
  - Small resource usage ideal for less important paths
- Dependence height
  - Matched heights are ideal
  - Close to same heights is ok
- Remember everything is relative for resources and dependence height !
- Hazards
  - Avoid hazards unless on most important path
- Estimate of benefit
  - Branches/Mispredicts removed
  - Fudge factor

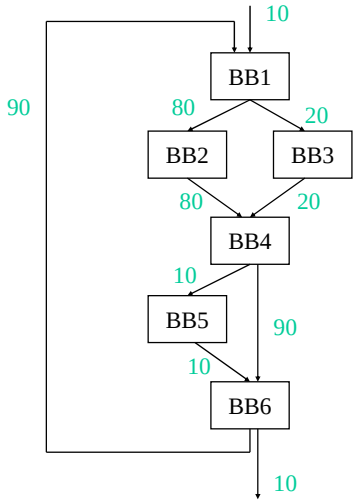


The Hyperblock

- Hyperblock - Collection of basic blocks in which control flow may only enter at the first BB. *All internal control flow is eliminated via if-conversion*
  - "Likely control flow paths"
  - Acyclic (outer backedge ok)
  - Multiple intersecting traces with no side entrances
  - Side exits still exist
- Hyperblock formation
  - 1. Block selection
  - 2. Tail duplication
  - 3. If-conversion



Block Selection

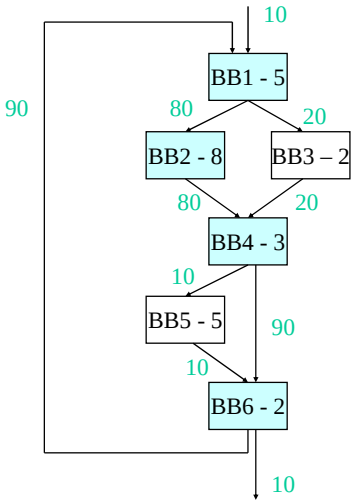


- Block selection
  - Select subset of BBs for inclusion in HB
  - Difficult problem
  - Weighted cost/benefit function
    - Height overhead
    - Resource overhead
    - Hazard overhead
    - Branch elimination benefit
    - Weighted by frequency

Block Selection

- Create a trace →"main path"
  - Use a heuristic function to select other blocks that are "compatible" with the main path
  - Consider each BB by itself for simplicity
    - Compute priority for other BB's
    - Normalize against main path.
- BSV_i = (K x (weight_bb_i / size_bb_i) x (size_main_path / weight_main_path) x bb_char_i)
  - weight = execution frequency
  - size = number of operations
  - bb_char = characteristic value of each BB
    - Max value = 1, Hazardous instructions reduce this to 0.5, 0.25, ...
  - K = constant to represent processor issue rate
- Include BB when BSV_i > Threshold

Example - Step 1 - Block Selection



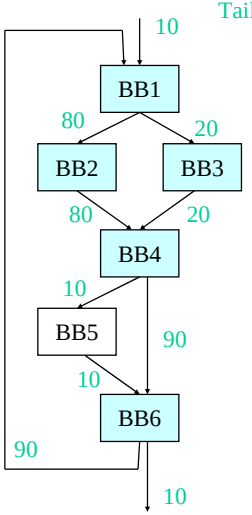
main path = 1,2,4,6
num_ops = 5 + 8 + 3 + 2 = 18
weight = 80

Calculate the BSVs for BB3, BB5 assuming no hazards, K = 4

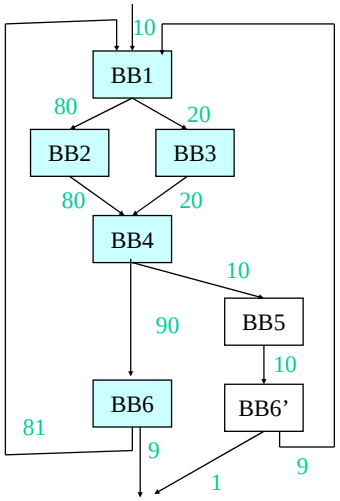
BSV3 = 4 x (20 / 2) x (18 / 80) = 9
BSV5 = 4 x (10 / 5) x (18 / 80) = 1.8

If Threshold = 2.0, select BB3 along with main path

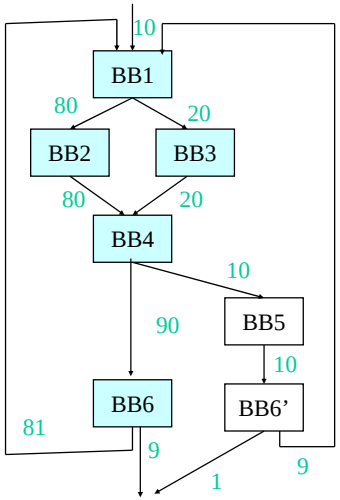
Example - Step 2 - Tail Duplication



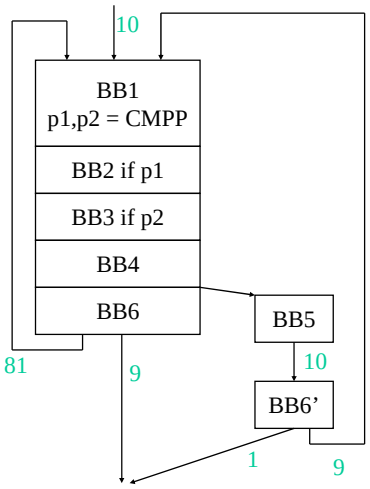
Tail duplication same as with Superblock formation



Example - Step 3 – If-conversion

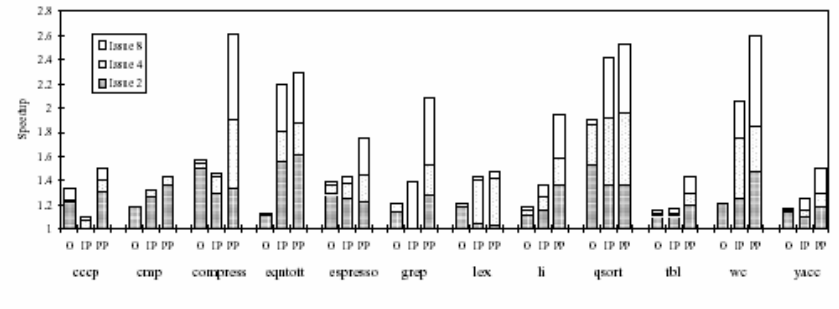


If-convert intra-HB branches only!!

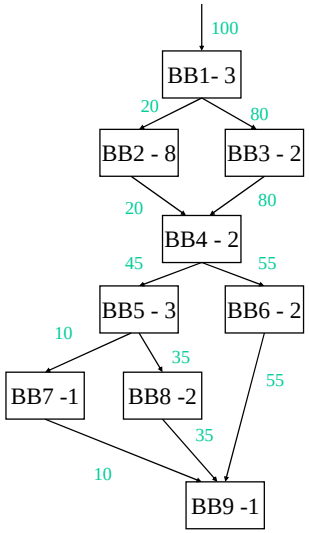


Hyperblock Performance Evaluation (1)

- O = BB code
- IP = Structural if-conversion
  - All innermost loops, acyclic SEME regions
- PP = Selective if-conversion



Class Problem



Form the HB for this subgraph
Assume K = 4, BSV Threshold = 2

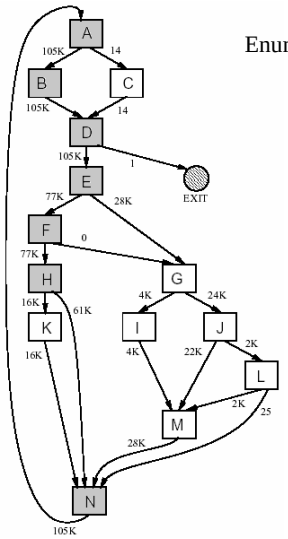
Block Selection – Try 2

- Problems with BSV formula
  - Ignore dependence height
  - Blocks considered independently (control flow ignored)
- Enumerate all paths of execution through region of interest
  - Consider a path – execution from entry to some exit
  - Give priority to path as a whole
- Path priority
  - dep_ratioi = 1.0 – (dep_heighti / max dep_height)
  - op_ratioi = 1.0 – (num_opsi / max num_ops)
  - priorityi = (probabilityi x hazardi) x (dep_ratioi + op_ratioi + K)
    - Hazard multiplier was 0.25 for paths containing subroutine call or unresolvable memory store
    - K = base contribution for a path (0.1 used)

Block Selection – Try 2 (continued)

- Path selection
  - Rank paths from highest to lowest priority
  - Include paths until either:
    - Estimated available resources full
    - Priority drops too low
  - Exclude any paths with excessive resource util or dep height
- Use union of selected paths to form Hyperblock
  - Causes some lower priority paths to be included

Block Selection - Try 2 - Example

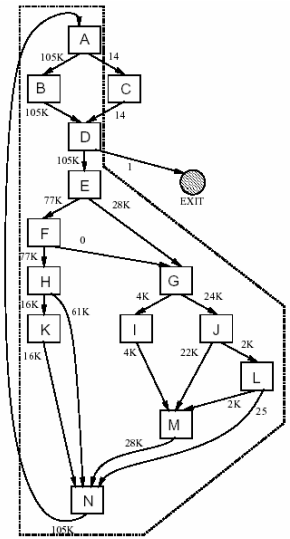


Enumerate all paths, rank by priority

1. A-B-D-E-F-H-N
2. A-B-D-E-F-H-K-N
3. A-B-D-E-G-J-M-N
4. A-B-D-E-G-J-L-M-N
5. A-B-D-E-G-I-M-N
6. A-B-D-E-G-J-L-N
7. A-B-D
8. A-C-D-E-F-H-N
9. A-C-D-E-F-H-K-N
10. A-C-D-E-G-J-M-N
11. A-C-D-E-G-J-L-M-N
12. A-C-D-E-G-I-M-N
13. A-C-D-E-G-J-L-N
14. A-C-D
15. A-B-D-E-F-G-I-M-N
16. A-B-D-E-F-G-J-M-N
17. A-B-D-E-F-G-J-L-M-N
18. A-B-D-E-F-G-J-L-N
19. A-B-C-E-F-G-I-M-N
20. A-B-C-E-F-G-J-M-N
21. A-B-C-E-F-G-J-L-M-N
22. A-B-C-E-F-G-J-L-N

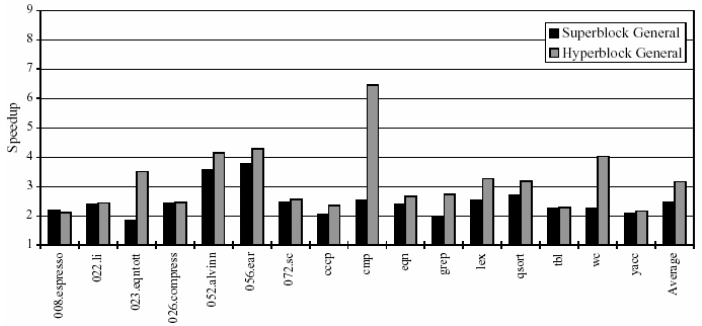
Block Selection – Try 2 – Example continued

1. A–B–D–E–F–H–N
2. A–B–D–E–F–H–K–N
3. A–B–D–E–G–J–M–N
4. A–B–D–E–G–J–L–M–N
5. A–B–D–E–G–I–M–N
6. A–B–D–E–G–J–L–N
7. A–B–D
8. A–C–D–E–F–H–N
9. A–C–D–E–F–H–K–N
10. A–C–D–E–G–J–M–N
11. A–C–D–E–G–J–L–M–N
12. A–C–D–E–G–I–M–N
13. A–C–D–E–G–J–L–N
14. A–C–D
15. A–B–D–E–F–G–I–M–N
16. A–B–D–E–F–G–J–M–N
17. A–B–D–E–F–G–J–L–M–N
18. A–B–D–E–F–G–J–L–N
19. A–C–D–E–F–G–I–M–N
20. A–C–D–E–F–G–J–M–N
21. A–C–D–E–F–G–J–L–M–N
22. A–C–D–E–F–G–J–L–N



Hyperblock Performance Using Paths

4 - issue



8 - issue