Lecture 5: Superblocks

COS 598C – Advanced Compilers

Prof. David August
Department of Computer Science
Princeton University

Regions

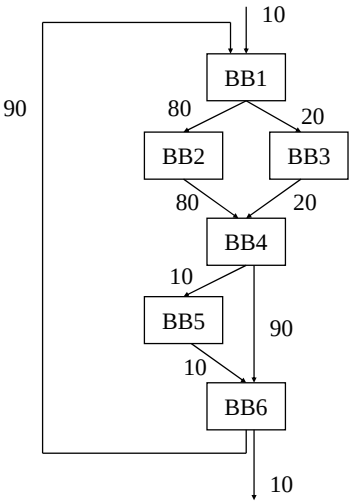
- Region: A collection of operations that are treated as a single unit by the compiler
  - Examples
    - Basic block
    - Procedure
    - Body of a loop
  - Properties
    - Connected subgraph of operations
    - Control flow is the key parameter that defines regions
    - Hierarchically organized
- Problem
  - Basic blocks are too small (3-5 operations)
    - Hard to extract sufficient parallelism
  - Procedure control flow too complex for many compiler xforms
    - Plus only parts of a procedure are important (90/10 rule)

Regions (2)

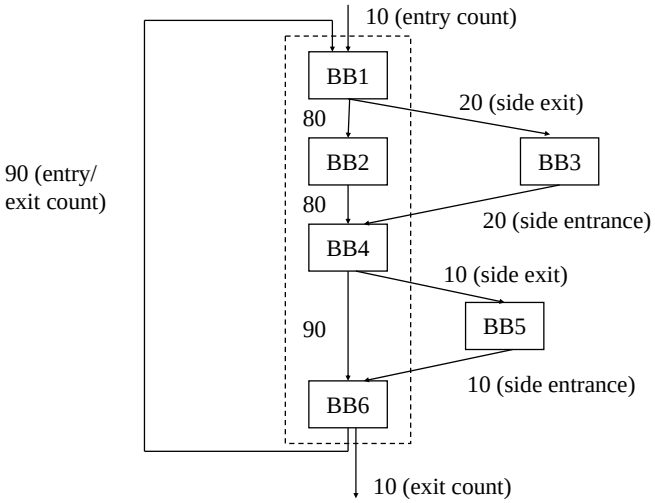
- Want
  - Intermediate sized regions with simple control flow
  - Bigger basic blocks would be ideal !!
  - Separate important code from less important
  - Optimize frequently executed code at the expense of the rest
- Solution
  - Define new region types that consist of multiple BBs
  - Profile information used in the identification
  - Sequential control flow (sorta)
  - Pretend the regions are basic blocks

Region Type 1 – Trace

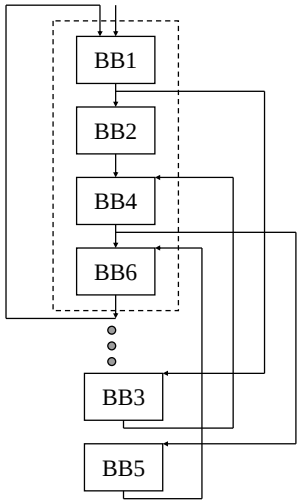
- Trace - Linear collection of basic blocks that tend to execute in sequence
  - "Likely control flow path"
  - Acyclic (outer backedge ok)
- Side entrance – branch into the middle of a trace
- Side exit – branch out of the middle of a trace
- Compilation strategy
  - Compile assuming path occurs 100% of the time
  - Patch up side entrances and exits afterwards
- Motivated by scheduling (i.e., trace scheduling)



Linearizing a Trace

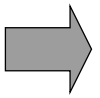


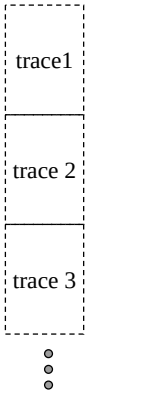
Intelligent Trace Layout for I-Cache Performance



Trace view

Intraprocedural code placement
Procedure positioning
Procedure splitting



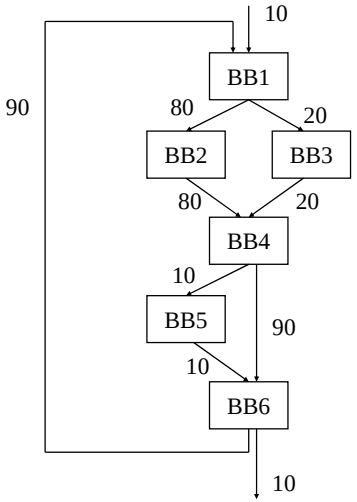


The rest

Procedure view

Issues With Selecting Traces

- Acyclic
  - Cannot go past a backedge
- Trace length
  - Longer = better ?
  - Not always !
- On-trace / off-trace transitions
  - Maximize on-trace
  - Minimize off-trace
  - Compile assuming on-trace is 100% (ie single BB)
  - Penalty for off-trace
- Tradeoff (heuristic)
  - Length
  - Likelihood remain within the trace



Trace Selection Algorithm

```
i = 0;
mark all BBs unvisited
while (there are unvisited nodes) do
      seed = unvisited BB with largest execution freq
      trace[i] += seed
      mark seed visited
      current = seed
      /* Grow trace forward */
      while (1) do
         next = best_successor_of(current)
         if (next == 0) then break
         trace[i] += next
         mark next visited
         current = next
      endwhile
      /* Grow trace backward analogously */
      i++
endwhile
```

Best Successor/Predecessor

Node weight vs. edge weight
- edge more accurate

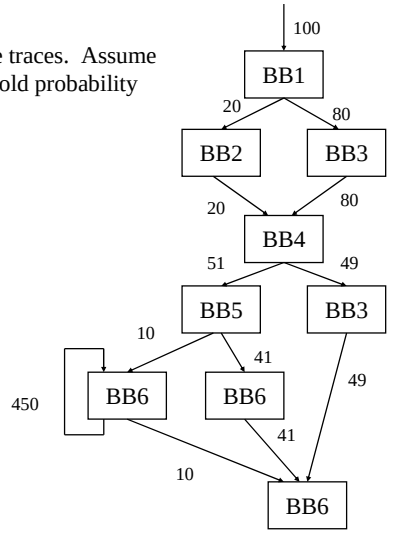
THRESHOLD
- controls off-trace probability
- 60-70% found best

Notes on this algorithm
- BB only allowed in 1 trace
- Cumulative probability ignored
- Min weight for seed to be chose (ie executed 100 times)

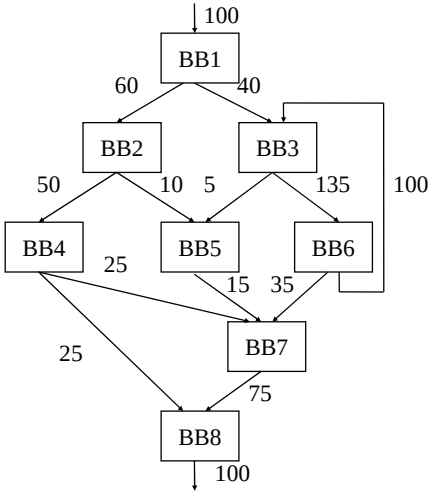
```
best_successor_of(BB)
    e = control flow edge with highest
          probability leaving BB
    if (e is a backedge) then
        return 0
    endif
    if (probability(e) <= THRESHOLD) then
        return 0
    endif
    d = destination of e
    if (d is visited) then
        return 0
    endif
    return d
endprocedure
```

Class Problem 1

Find the traces. Assume a threshold probability of 60%.



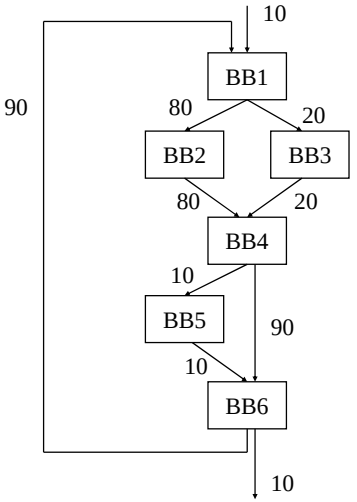
Class Problem 2



Find the traces. Assume a threshold probability of 60%.

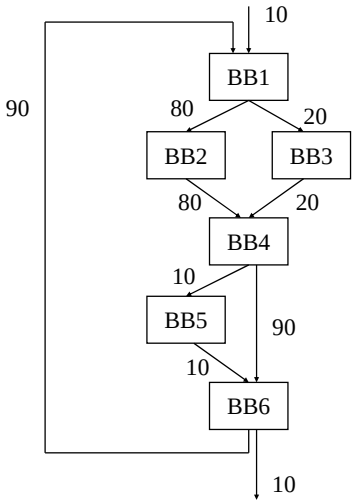
Traces are Nice, But …

- Treat trace as a big BB
  - Transform trace ignoring side entrance/exits
  - Insert fixup code
    - AKA bookkeeping
  - Side entrance fixup is more painful
  - Sometimes not possible so transform not allowed
- Solution
  - Eliminate side entrances
  - The superblock is born



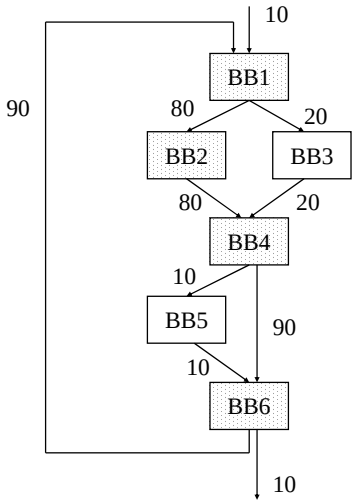
Region Type 2 – Superblock

- Superblock - Linear collection of basic blocks that tend to execute in sequence *in which control flow may only enter at the first BB*
  - "Likely control flow path"
  - Acyclic (outer backedge ok)
  - Trace with no side entrances
  - Side exits still exist
- Superblock formation
  - 1. Trace selection
  - 2. Eliminate side entrances

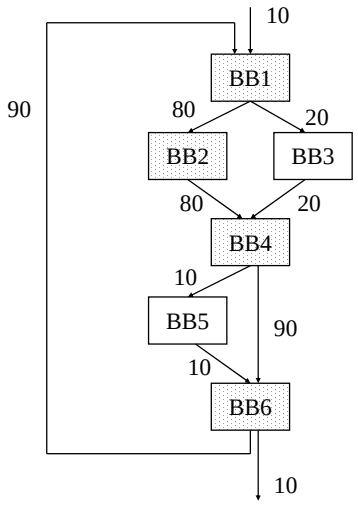


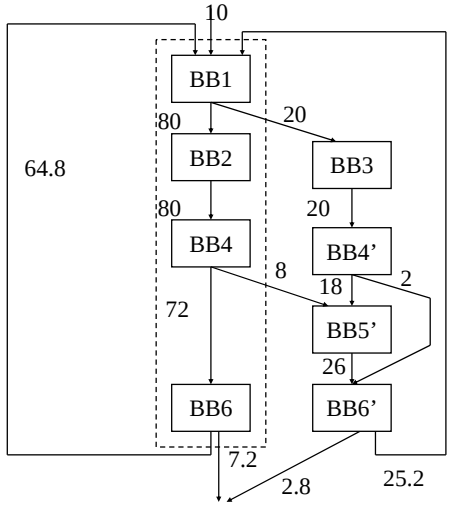
Tail Duplication

- To eliminate all side entrances replicate the "tail" portion of the trace
  - Identify first side entrance
  - Replicate all BB from the target to the bottom
  - Redirect all side entrances to the duplicated BBs
  - Copy each BB only once
  - Max code expansion = 2x-1 where x is the number of BB in the trace
  - Adjust profile information



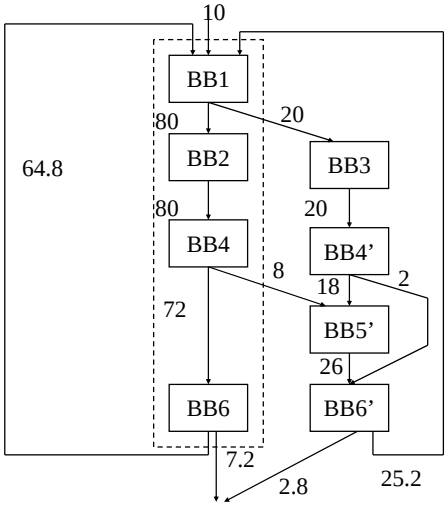
Superblock Formation



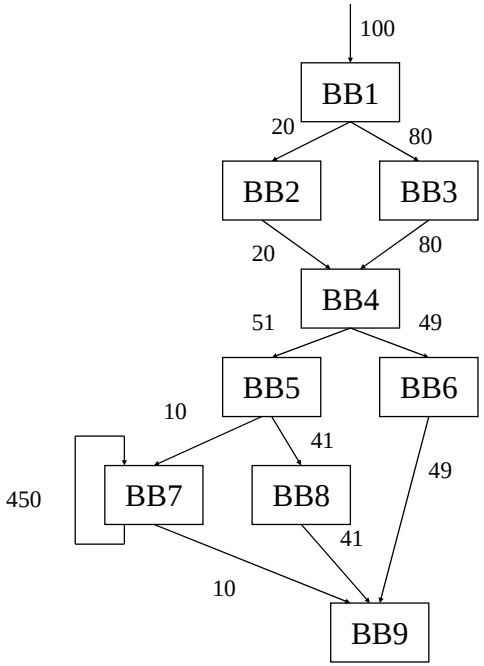


Issues with Superblocks

- Central tradeoff
  - Side entrance elimination
    - Compiler complexity
    - Compiler effectiveness
  - Code size increase
- Apply intelligently
  - Most frequently executed BBs are converted to SBs
  - Set upper limit on code expansion
  - 1.0 – 1.10x are typical code expansion ratios from SB formation



Class Problem 3



Create the superblocks, trace threshold is 60%