# Predicated Execution

Wen-mei Hwu
ECE
University of Illinois,
Urbana-Champaign

# Predicated Execution

- Allows conditional execution of instructions based on a Boolean source operand
- Provides compiler with an alternative to guarding instruction execution with branches

# Execution Model

- Additional Boolean operand
  - e.g., Load r1, r2, r3 <p1>
  - If p1 is TRUE, instruction executes normally
  - If p1 is FALSE, instruction treated as NOP (with some exceptions)

# Full Predication Support

- Predicate defining instructions
- Full set of predicated instructions
- Separate predicate register file
- Best performance
- Cydra-5, IA-64, TI-C60, StarCore

# Partial Predication Support

- Adds limited set of predicated instructions to existing ISA
  - no extension to operand format
  - CMOV, SELECT
- Brings some performance increase to existing ISA's
- SPARC, Alpha, MIPS, P6

# HP-PD Predicate Defines

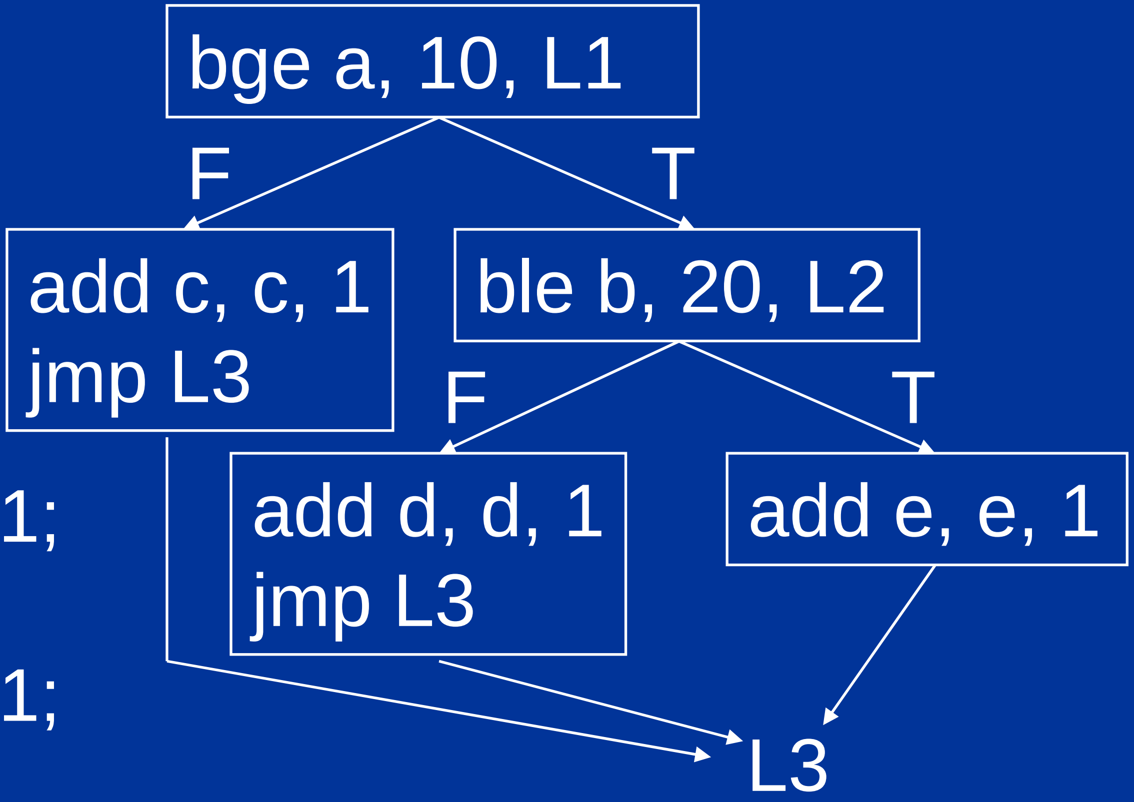
**pred_< *cmp* > P1 < *type* >, P2 < *type* >, src1, src2 (Pin)**

- < *cmp* > - condition: =, >, <, etc.
- < *type* >
  - Unconditional (U, U)
  - OR-type (O, O)
  - AND-type (A, A)

# Uncond. Predicate Defines

- For blocks reached on one condition

```
If (a < 10)
        c= c+1;
else
        if (b > 20)
                d = d+1;
        else
                e = e+1;
```






IMPACT

# Uncond. Predicate Defines

- See figure.

# Or-type Predicate Defines

- For blocks reached on multiple conditions
- see figure.

# AND-type Predicate Defines

- Handle blocks executed under conjunction of condition
- see figure.

# Instruction Issue Models

- Conversion to NOP model
  - Instruction converted to NOP at issue if predicate is 0
  - Most efficient model, useful for in-order issue processors

# Instruction Issue Semantics

- Data selection model
  - Facilitates dynamic register renaming and out-of-order issue
    - If (p==1) dest = op(src1, src2, ...)
    - Else dest = old_dest
  - Predicated instructions always produce a result

# Data Selection Model

A: add r1, r4, r5 (p)

B: load r6, r1, 0

- A reserves its dest register (r1)
- B awaits A for r1
- Necessary when instructions reserve destination registers before their predicate is known

# Data Selection Model

- Overhead
  - Destination operand read as additional source operand
  - Output dependent instructions converted to flow dependent instructions

# Predicated Compilation

- Region formation
- if-conversion and RIC
- Optimizing predicated code
- Speculation in predicated code
- Scheduling predicated code
- Code generation

# Hyperblock

- Set of connected basic blocks
  - One entry block dominates all others
  - Control flow may only enter hyperblock at entry block
  - Control flow may exit hyperblock at any number of locations

# If-conversion

- Control flow between blocks in hyperblock is eliminated
- Each hyperblock contains an implicit acyclic CFG.

# Hyperblock Example

- See figure

# A Real Hyperblock

- See figure

# Hyperblock Formation

- Relatively well understood for loops
- Relies on profile information
- Uses code exclusion and tail duplication to avoid imbalanced dependences and over use of resources

# Hyperblock Formation Steps

- Identify region
  - loop, hammock, dominating block
- Backedge/exit coalescing
- Block selection
- Tail duplication
- If-conversion

# Formation Example

- See figure (wc).

# Block Selection

- Enumerate all paths in region
- Calculate path priority
  - dep_ratio(i) = 1.0 - (dep_height / MAX(dep_height(j))
  - op_ratio(I) = 1.0 - (num_ops / MAX(num_ops(j))
  - priority(i) = probability(i) * (dep_ratio(i) + op_ratio(i) + hazard(I)+K)

# Block Selection (cont.)

- Include paths from highest to lowest priority until:
  - Processor resources saturated
  - Minimum priority threshold reached

# Block Selection Example

- See figure

# Tail Duplication

- Eliminate side entrances into hyperblocks
  - control flow from non-selected blocks to selected blocks
  - mark all blocks with side entrance
  - mark all successor blocks
  - duplicate all marked blocks
  - adjust side entrance to duplicates

# Tail Duplication Example

- See figure.

# If-conversion

- Convert control flow into data flow
- RK algorithm [Park & Schlansker]
  - minimal in predicates and predicate defines used
  - compute control dependences
  - assign predicate to each unique value
  - insert predicate defines
  - remove control flow

# If-Conversion Example

- See figure.



IMPACT