



# Optimizing Software With Intel Compilers

By Eric W Moore



# Objectives

- After completing this class, you will be able to:
  - Optimize software for Intel® architecture
  - Use key compiler optimization switches

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Agenda

- **Introduction**
- Optimization with switches
  - General and processor-specific optimization
  - Interprocedural optimization (IPO)
  - Profile-guided optimization (PGO)
- Compiler Reports



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# EPIC Architecture Principles

- “The compiler should play the key role in designing the plan of execution, and the architecture should provide the requisite support for it to do so successfully.
- The architecture should provide features that assist the compiler in exploiting statistical ILP.
- The architecture should provide mechanisms to communicate the compiler's plan of execution to the hardware.”
  - Schlansker, Michael S. and Rau, B. Ramakrishna (HP Labs),
  - “EPIC: Explicitly Parallel Instruction Computing”
  - Computer, Vol 33 Issue: 2 , Feb. 2000 pp 37-45

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Agenda

- Introduction
- Optimization with switches
  - General and processor-specific optimization
  - Interprocedural optimization (IPO)
  - Profile-guided optimization (PGO)
- Compiler Reports



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# General Optimization Switches

|  | Linux* |
|--|--------|
| Disable optimization   | -O0    |
| Optimize for speed without increasing code size                  | -O1    |
| Optimize for speed (default)                                     | -O2    |
| High-Level optimizer   | -O3    |
| Create symbols for debugging                                     | -g     |
| Generate assembly files with .s and stop the compilation process | -S     |



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# High-level Optimizer

- Overview
  - Loop level optimizations (including loop unrolling, cache blocking, etc.)
  - Converts source code algorithm to use more optimal memory access pattern
- How
  - (Linux\*) -O3
  - (Windows\*) /O3
- Loops must meet certain criteria ...

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# High Level Optimizer

- Single precision floating point code fragment

```
for (j=1; j<1000; j++) {  
    y[j] = y[j] + a*x[j]; }  
}
```

- turns into:

```
for (j=1; j<1000; j+=2) {  
    y[j] = y[j] + a*x[j];  
    y[j+1] = y[j+1] + a*x[j+1];  
}
```

- Now it can use `ldfp` instead of `ldf`



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# OpenMP\* Support

- OpenMP 2.0 for Fortran and C++
  - The OpenMP\* 2.0 WORKSHARE directive is now supported in Fortran9.0.
- Also supports the OpenMP extension “workqueuing model” from Intel to exploit task level parallelism.
- Usage Model:

Linux\*

-openmp

-openmp\_report[n]

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Auto-parallelization

- Auto-parallelization: the automatic threading of loops without having to manually insert the OpenMP\* directive
  - Compiler can identify “easy” candidates for parallelization
  - (Linux\*) -parallel
  - OpenMP directives give greater flexibility

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Agenda

- Introduction
- Optimization with switches
  - General and processor-specific optimization
  - Profile-guided optimization (PGO) Interprocedural optimization (IPO)
- Compiler Reports



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Profile-Guided Optimizations (PGO)

- Uses execution-time feedback to guide optimization
- Helps I-cache, paging, branch prediction
- Enabled optimizations:
  - Basic block ordering
  - Better register allocation
  - Better decision of functions to inline
  - Function ordering
  - Switch-statement optimization

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Usage: Three-Step Process

## Step 1

**Instrumented Compilation**  
(Linux\*) `icc -prof_gen prog.c`

Instrumented  
executable

## Step 2

**Instrumented Execution**  
Run program on a typical dataset

DYN file containing  
dynamic info: `.dyn`

## Step 3

**Feedback Compilation**  
(Linux) `icc -prof_use prog.c`

Merged DYN  
summary file: `.dpi`  
Delete old dyn files if  
you do not want the info  
included

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# When to Use

- Applications with lots of functions, calls, or branches
  - Examples: databases, decision-support (enterprise), MCAD
  - Applications with computation spread throughout; not confined to kernels
  - Trip counts for loops can enable more aggressive optimization
- Considerations:
  - Different paradigm for builds - three steps
  - Schedule time in final stages of development when code is more stable
  - Use representative data sets (not for corner cases)

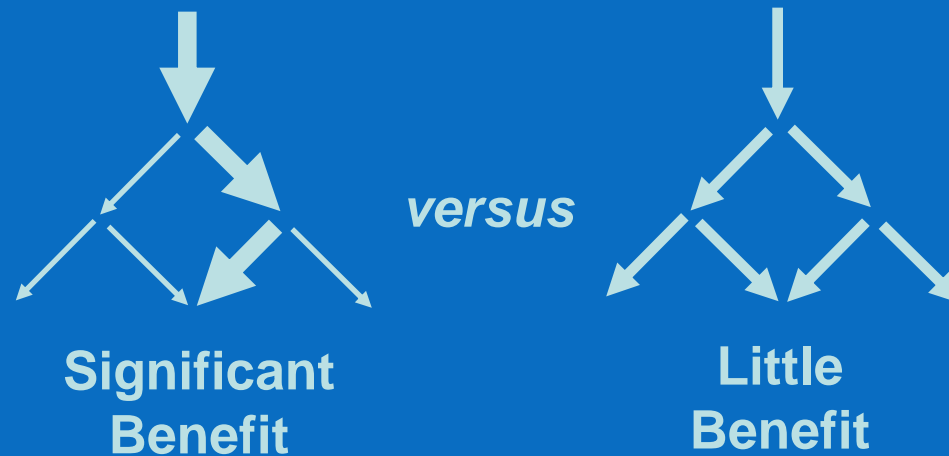


Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Programs That Benefit

- Consistent hot paths
- Many if statements or switches
- Nested if statements or switches



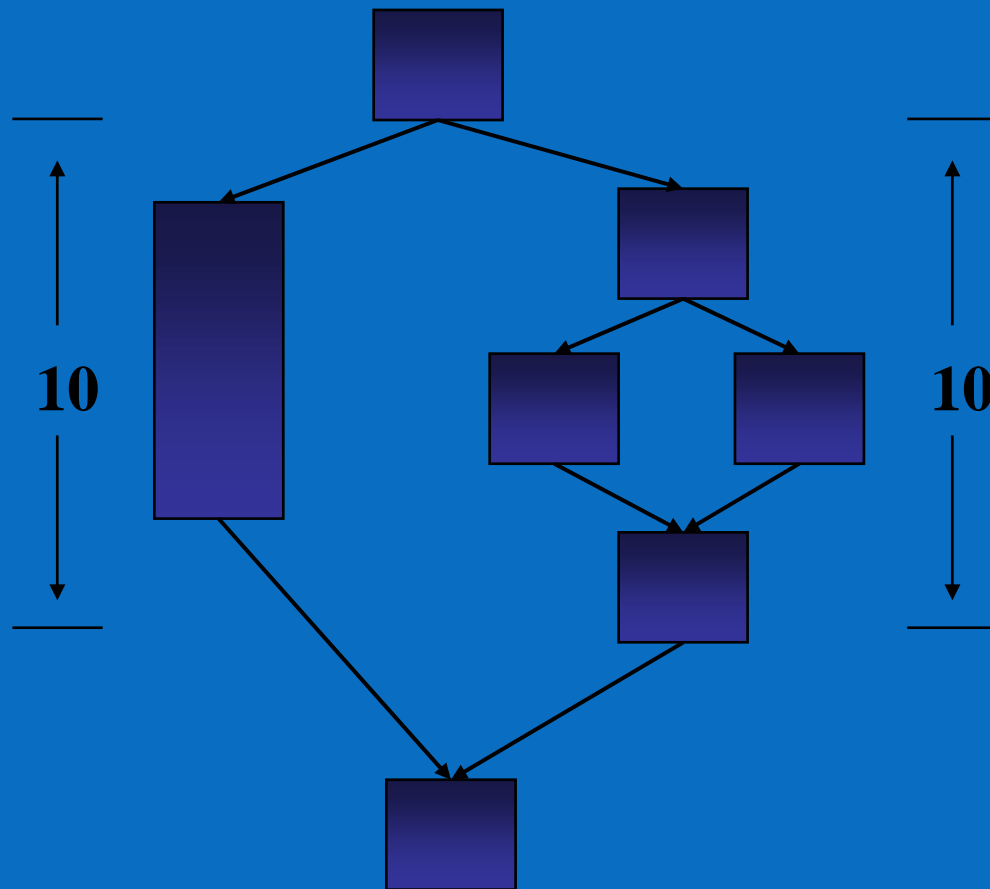
# Profile-Guided Optimizer

- Affects many compiler optimizations
  - Dynamic branch prediction
  - Loop (pipeline versus unroll versus none)
  - Cache utilization
  - Predication
  - Speculation
  - Classical (block order, inline, reg alloc)
  - Function splitting

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



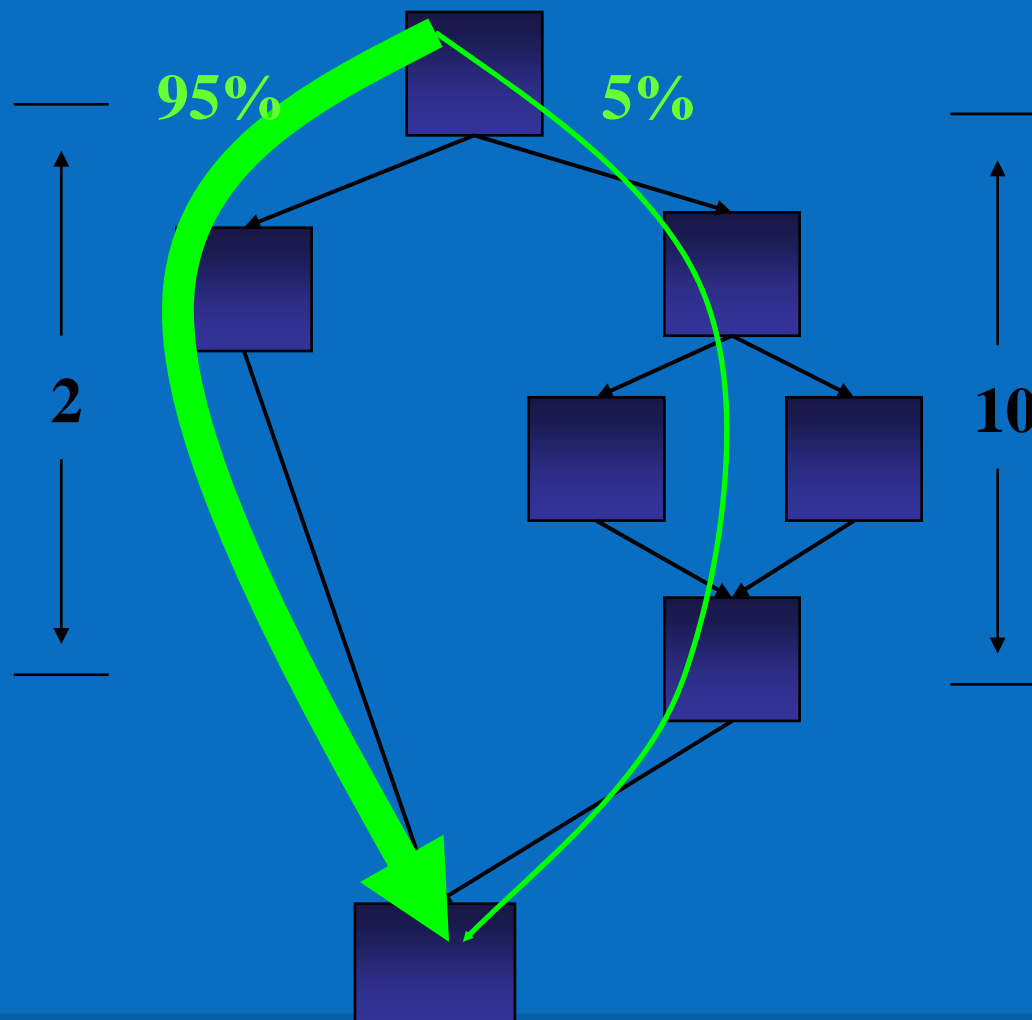
# Helps Improve Predication



Do we predicate?

Balanced Paths:  
Good opportunity  
to predicate  
independent  
of profile

# Predication: PGO Benefits



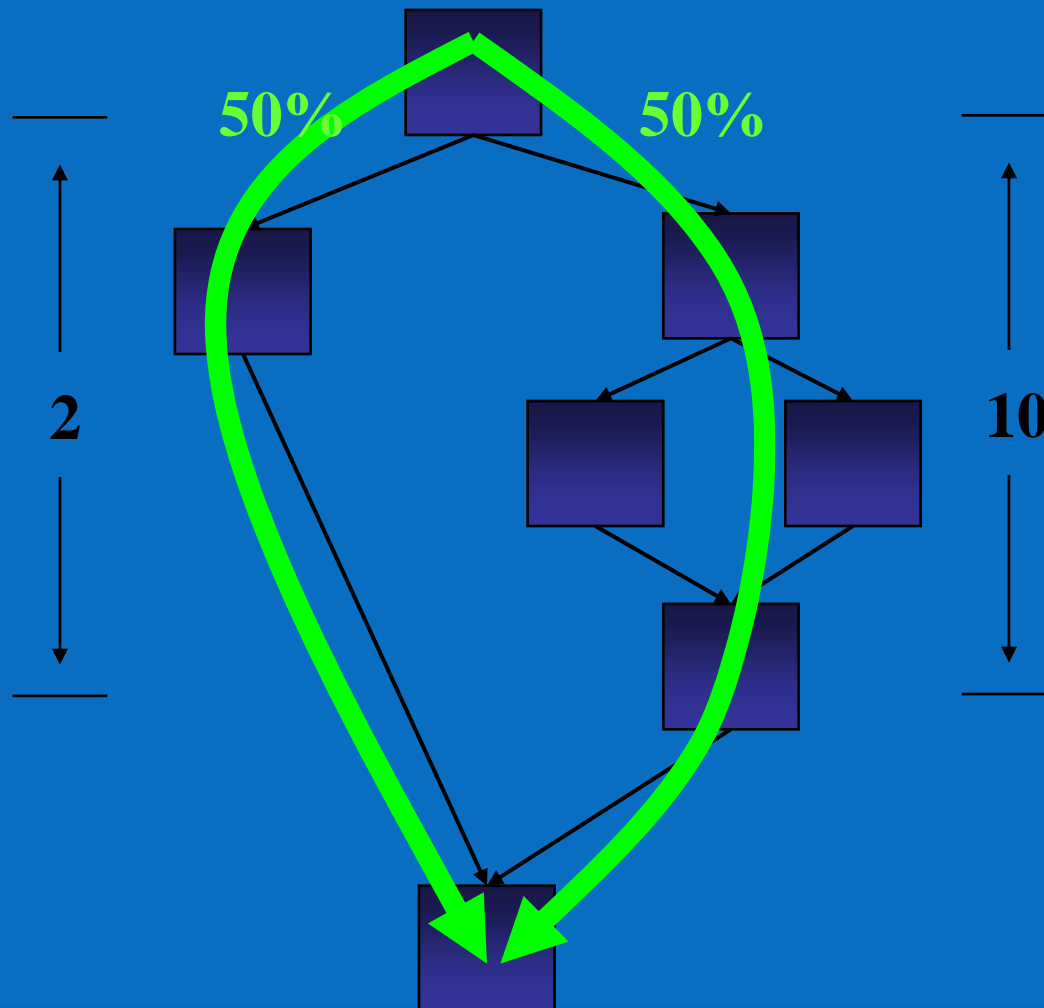
Do we predicate?

Bad Move!  
Main path  
length increases  
from 2 to 10.

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Predication: PGO Benefits



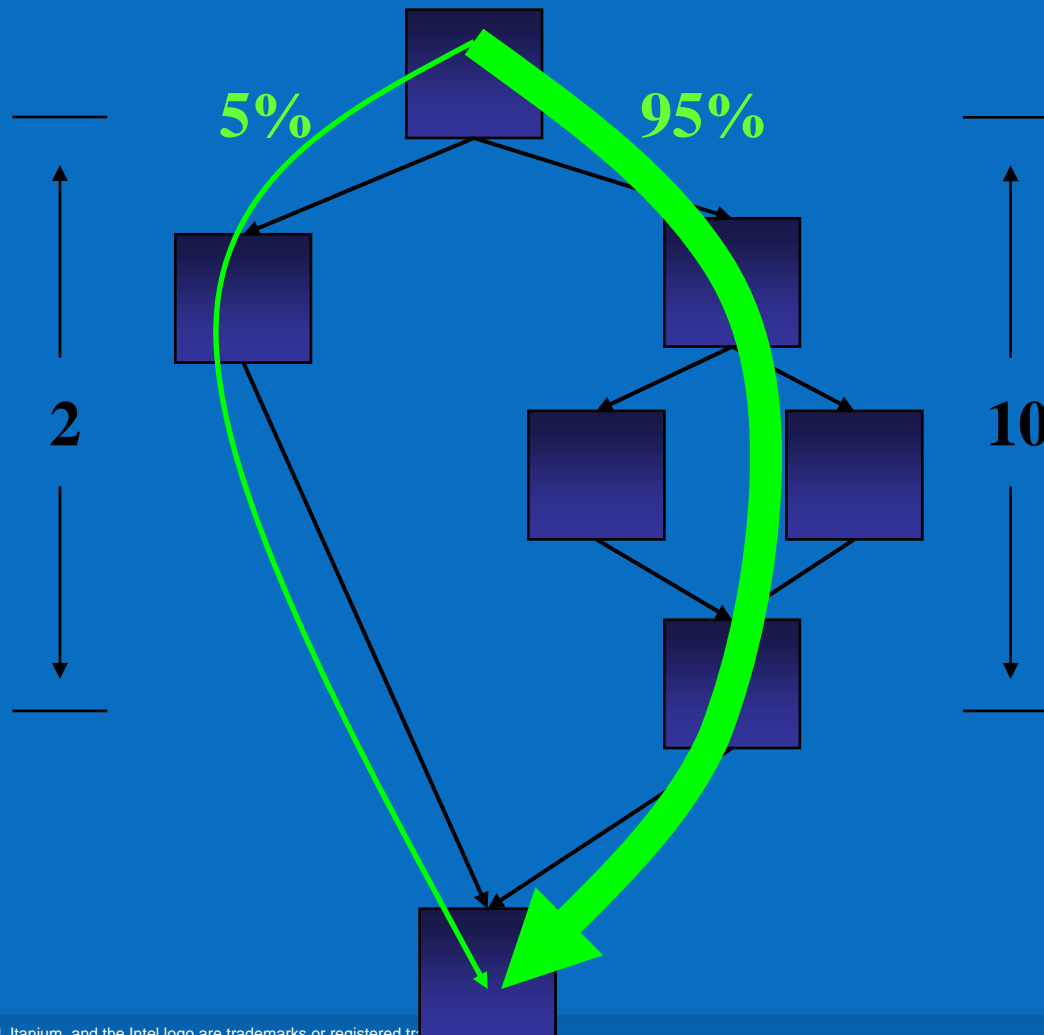
Do we predicate?

Not as clear.  
Main path  
length increased  
but mispredicts  
reduced.

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Predication: PGO Benefits



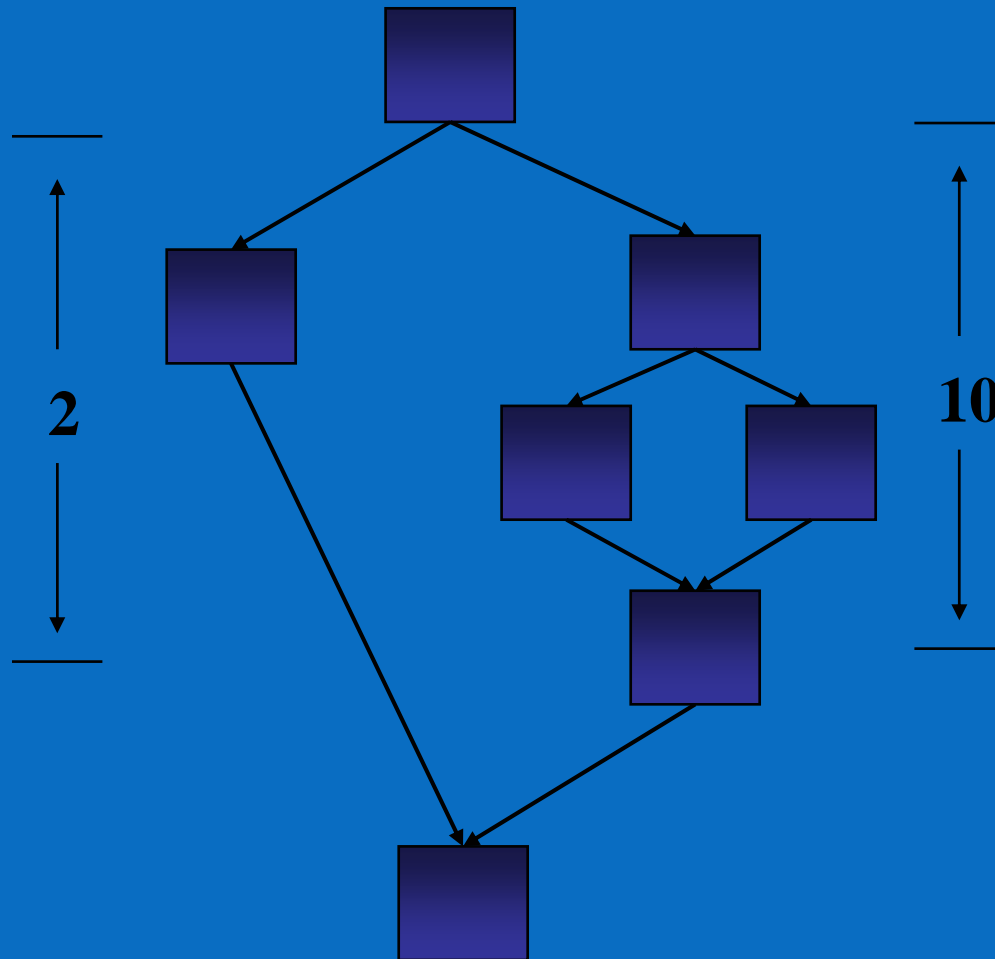
Do we predicate?

Good move.  
The left side will  
slide in for free

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Predication: PGO Benefits



Without profile we will not predicate

Not any worse than traditional architectures.

Forfeit chance to improve performance.



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Agenda

- Introduction
- Optimization with switches
  - General and processor-specific optimization
  - Profile-guided optimization (PGO)
  - **Interprocedural optimization (IPO)**
- Compiler Reports



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Interprocedural Optimizations (IPO)

(Linux\*)

-ip

- Enables interprocedural optimizations for single file compilation.

(Linux\*)

-ipo

- Enables interprocedural optimizations across files

- Enhances optimization when used in combination with other compiler features

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



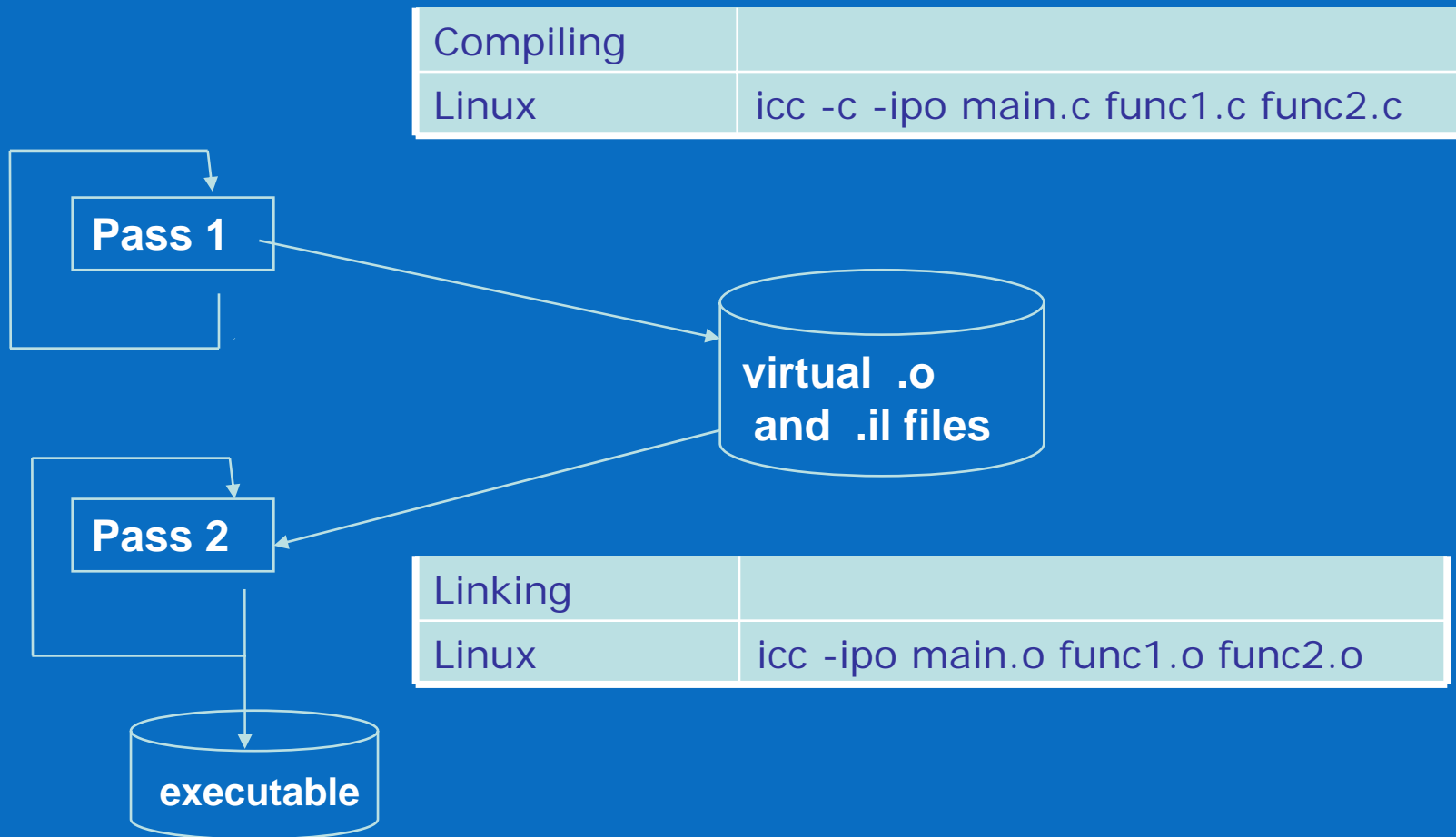
# Interprocedural Optimizations (IPO)

- Benefits
  - Inlining
  - Partial inlining
  - Help in software pipelining through memory disambiguation
  - Interprocedural constant propagation
  - Passing of arguments in registers
  - Loop-invariant code motion
  - Dead code elimination

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Usage: Two-Step Process



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Agenda

- Introduction
- Optimization with switches
  - General and processor-specific optimization
  - Interprocedural optimization (IPO)
  - Profile-guided optimization (PGO)
- **Compiler Reports**



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Generating Compiler reports

- Generating the full report
  - Compilers 9.1 and earlier
    - `icl [optimizations] /Qopt_report /Qopt_report_level max /Qpar_report3 &> compreport.txt`
- Note: The compiler will only generate a report for phases you turned on with optimization switches (i.e. no vector report generated without a vectorization switch: `-xT`)
- BKM – use `/Qopt_report_level max` - The Default level – does not identify compiler problems or identify potential solutions



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Filter The Data in the Reports

- Choose only specific phases relevant to what you are looking for
    - `-opt_report_phase [phase]`
      - Enables the report for only the selected phases
- Most Useful phases
- `hlo`
  - `ipo_inl`
  - `ecg_swp`
- `-par_report[0..3]`
    - Explains which loops Parallelized and why not
      - (including alias information)
- `-opt_report_routine functionname`
    - Views only the data for a particular function

# Example of Inlining Report (9.1 Compiler)

```
int main(int argc, int argv[]) {
    init(a, 1000);
    mysum = sum(a, 1000, 3);
    printf("%d\n", mysum);
}
```

```
void init(int a[], int n) {
    for (i = 0; i < 1000; i++)
        a[i] = 0;
}
```

```
int sum(int a[], int n, int b) {
    int mysum = 0;
    for (i = 0 ; i < n; i++)
        mysum = add(mysum,a[i]);
    return mysum;
}
```

- INLINING REPORT: (main)
- -> printf(EXTERN)
- -> INLINE: sum(5) (isz = 18) (sz = 30 (16+14))
- -> INLINE: add(6) (isz = 0) (sz = 8 (3+5))
- -> init(2) (isz = 14) (sz = 22 (14+8))
- [[ Callee not marked with inlining directive or pragma ]]

# HLO Report – daxpy.c

Prefetch distance



High Level Optimizer Report for: daxpy

Total #of lines prefetched in daxpy for loop in line

# of dynamic\_mapped\_array prefetches in daxpy for loop in line 5=2, dist=53

.....

Versioned for runtime  
dependence analysis

x, y independent  
x, y not independent

Loop dual-path'd for swp:

Loop at 5 -- selected for multiversion - DD

Block, Unroll, Jam Report:

(loop line numbers, unroll factors and type of transformation)

Loop at line 5(rt-dd-ver 1) unrolled with remainder by 8

Loop at line 5(rt-dd-ver 2) unrolled with remainder by 8

Loadpair Report:

Load-pair formed at line 5 , 5 [Method = Multiversion(2-way)]

Versioned for  
alignment

...

```
void daxpy(int n, double a, double *x, double *y) {  
    int i;  for (i=0;i<n;i++) *(x+i) += *(y+i)*a ; }  
}
```

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# New to Intel® Compiler 9.1 for HLO

## Suggestions and Rationale

```
for (j=0; j<N; j++) {  
    A[j] = j + B[j][1];  
    for (i=0; i<N; i++)  
        B[i][j] += A[j];  
}
```

**Loop Interchange Not Done due to: Imperfect Loop Nest**

**Advice: Loop Interchange, if possible, might help Loopnest at lines: 6 8**

**: Suggested Permutation: (1 2) --> (2 1)**

# Example of Par\_report3

```
for(i=0;i < VAL;i++) {  
    x = h * ((double)i - 0.5);  
    p += 4.0 / (1.0 + x*x);  
}  
p*=h
```

pi.c(14) : (col. 9) remark: LOOP WAS AUTO-PARALLELIZED.

procedure: func

parallel loop: line 14

shared : { }

private : { "i" "x" }

**first priv.: { "h" }**

reductions : { "p" }



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Software Pipeline Report Example 1

Resource II = 34  
Recurrence II = 9  
Minimum II = 34  
Scheduled II = 57

Estimated GCS II = 160

Percent of Resource II needed by arithmetic ops = 15%  
Percent of Resource II needed by memory ops = 12%  
Percent of Resource II needed by floating point ops = 100%

Number of stages in the software pipeline = 3



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Software Pipeline Report Example 2

Resource II = 27  
Recurrence II  $\geq$  28\*  
Minimum II = 27  
Last attempted II = 30

Estimated GCS II = 29

\*Too many cycles, swp gave up computing recurrence II

Software pipeliner estimated that it is more profitable to schedule the loop using the global acyclic scheduler than to pipeline the loop

# Other Considerations

- See Compiler User's Guide and Reference
- `icc -help`



Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.



# Summary

- Intel® compiler major optimization switches
  - High level optimizations
  - Interprocedural optimizations
  - Profile-guided optimizations
- Intel® compiler takes advantage of current Itanium® architecture

Intel, Itanium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

