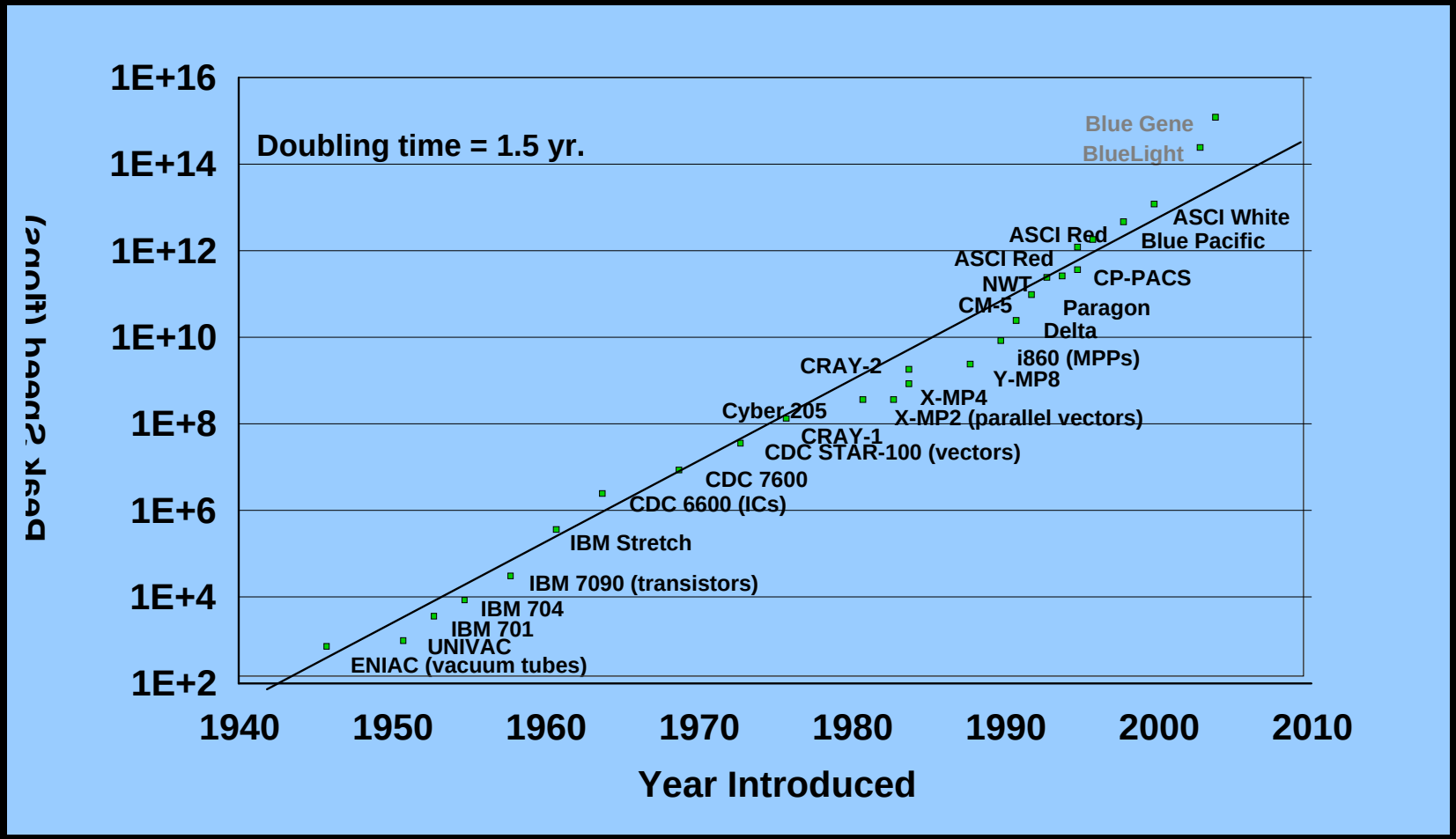
High Performance Computers and Compilers: A Personal Perspective

Fran Allen
allen@watson.ibm.com

Virginia Tech
September 11, 2009

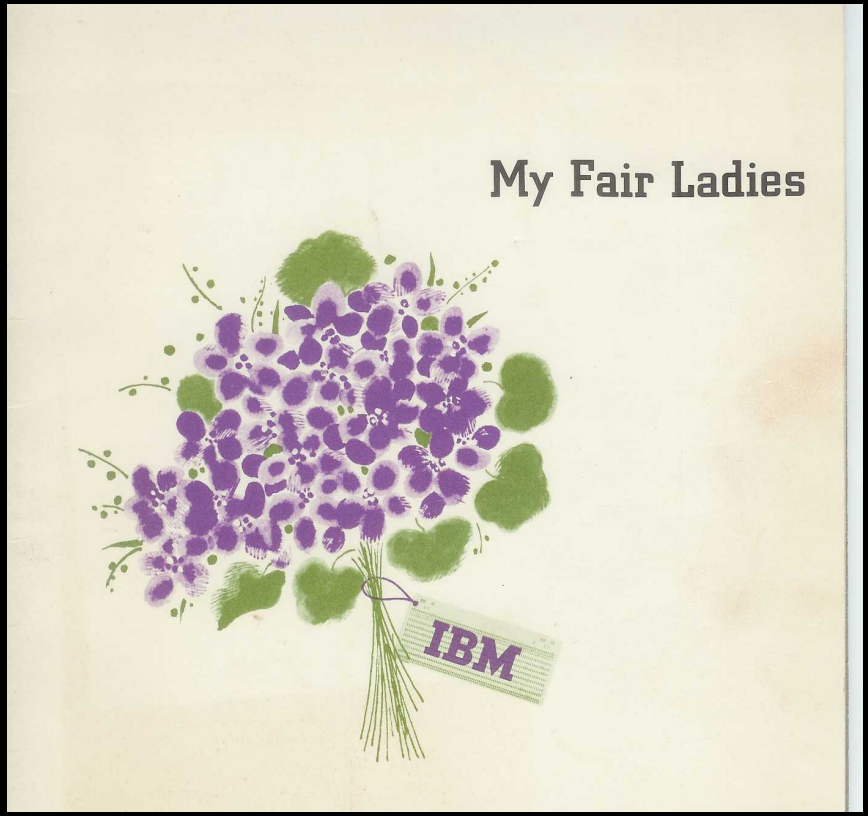
Peak Performance Computers by Year



Talk outline

- **A personal tour of compilers and computers for high performance systems**
- **The new performance challenge**
- **Addressing the performance challenge**
- **Discussion**

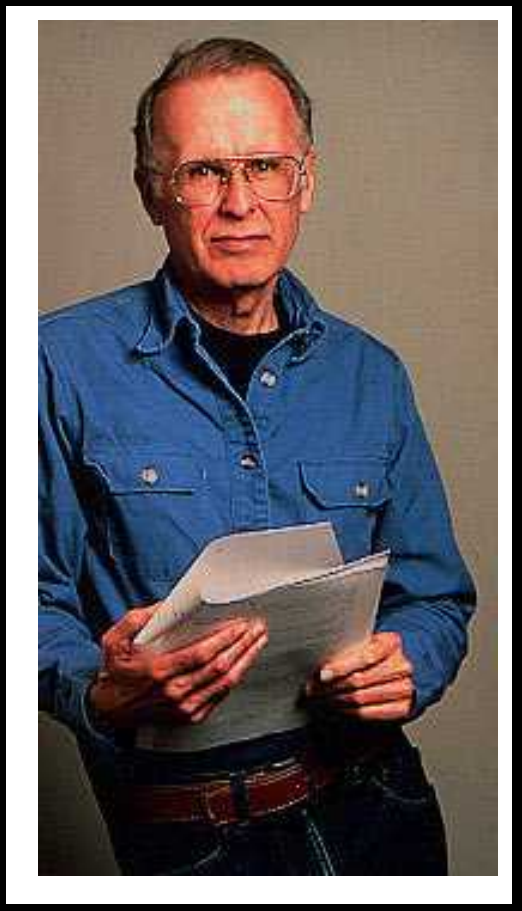
In 1957 I joined IBM Research as a Programmer



1957 IBM Recruiting Brochure

Fortran Project (1954-1957) Goals

- User Productivity
- Program Performance

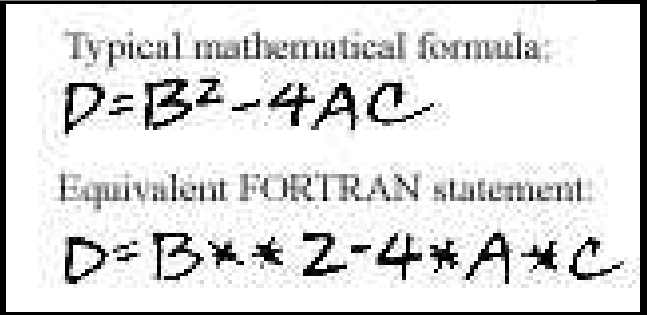


John Backus

THE FORTRAN GOALS BECAME MY GOALS

The Fortran Language and Compiler

- Available April 15, 1957



- Some features:
  - Beginnings of formal parsing techniques
  - Intermediate language form for optimization
  - Control flow graphs
  - Common sub-expression elimination
  - Generalized register allocation - for only 3 registers!
- Spectacular object code!!

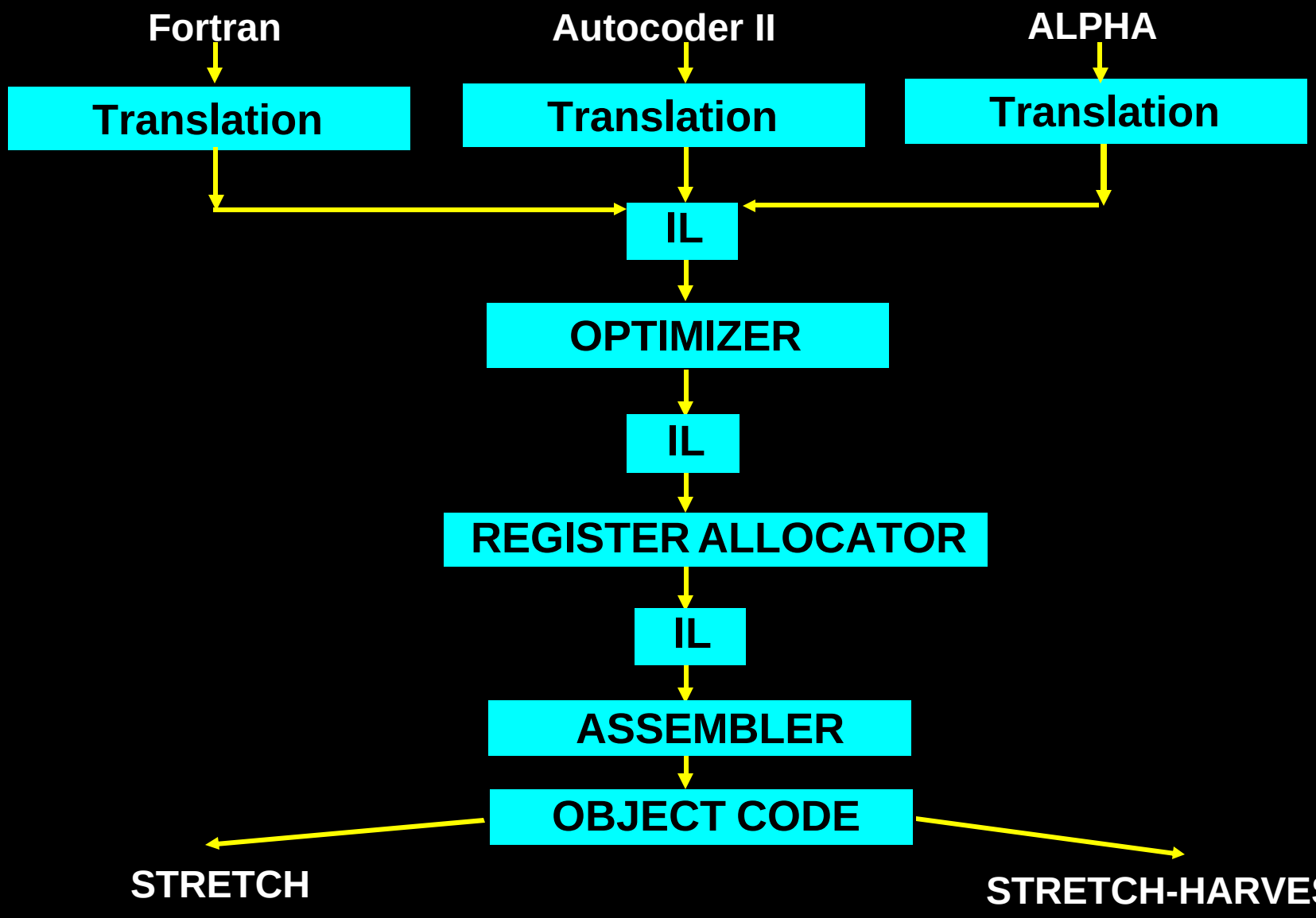
Stretch (1956-1961)

- Goal: 100 times faster than any existing machine
- Main Performance Limitation: Memory Access Time
- Extraordinarily ambitious hardware
- Equally ambitious compiler

HARVEST (1958 - 1962)

- Built for NSA for code breaking
- Hosted by Stretch
- Streaming data computation model
- Eight instructions and unbounded execution times
- Only system with balanced I/O, memory and computational speeds (per conversation with Jim Pomerene 11/2000)
- ALPHA: a language designed to fit the problem and the machine

Stretch – Harvest Compiler Organization

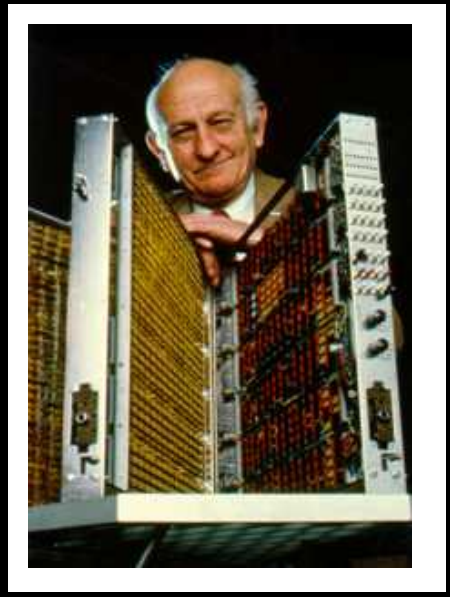


Stretch - Harvest Outcomes

- Stretch machine missed 100 x goal by 50%!
- A new Fortran compiler replaced original
- But "Stretch defined the limits of the possible for later generations of computer designers and users."
  (Dag Spicer - Curator Computer History Museum)
- National Security Agency used Harvest for 14 years

Advanced Computing System (ACS) 1962-1968

- **Goal: Fastest Machine in the World**
  - Pipelined and superscalar
  - Branch prediction
  - Out of order instruction execution
  - Instruction and data caches
- **Experimental Compiler:**
  - Built early to drive hardware design
  - Compiler code often faster than the best hand code



John Cocke

ACS Compiler Optimization Results

- Language-independent machine-independent optimization
- A theoretical basis for program analysis and optimization
- A Catalogue of Optimizations which included:
  - Procedure integration
  - Loop transformations: unrolling, jamming, unswitching
  - Redundant subexpression elimination, code motion, constant folding, dead code elimination, strength reduction, linear function test replacement, carry optimization, anchor pointing
- Instruction scheduling
- Register allocation

IBM CANCELLED ACS PROJECT IN 1968!

PTRAN: Automatic Parallelization (1980s to 1995)

- Research
  - Static Single Assignment (SSA)
  - Constructing Useful Parallelism
  - Whole Program Analysis Framework
- Compiler development
  - RP3/NYU Ultra Computer
  - IBM's XL Family of Compilers
  - Fortran 90
- Run-time technologies
  - Dynamic Process Scheduling
  - Debugging
  - Visualization

1994 was a bad year for compilers and parallelism

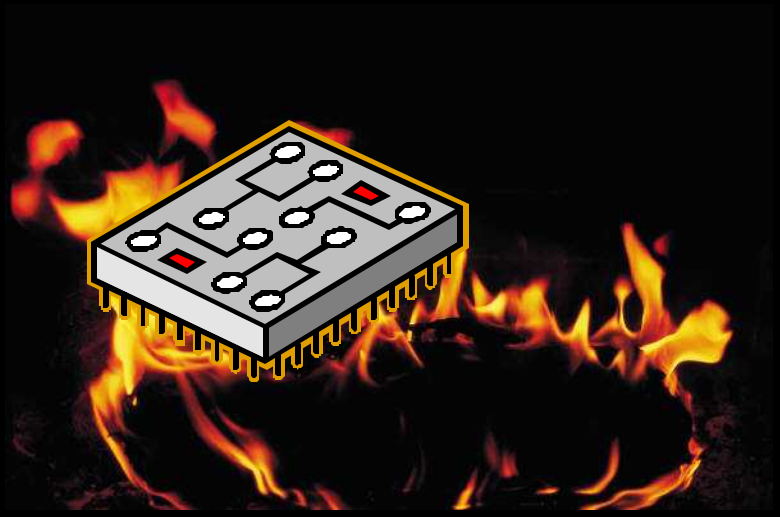
- PTRAN project at IBM cancelled
  - "IBM will never build another compiler."
  - "Parallelism is dead."
- HPF project at Rice cancelled

Talk outline

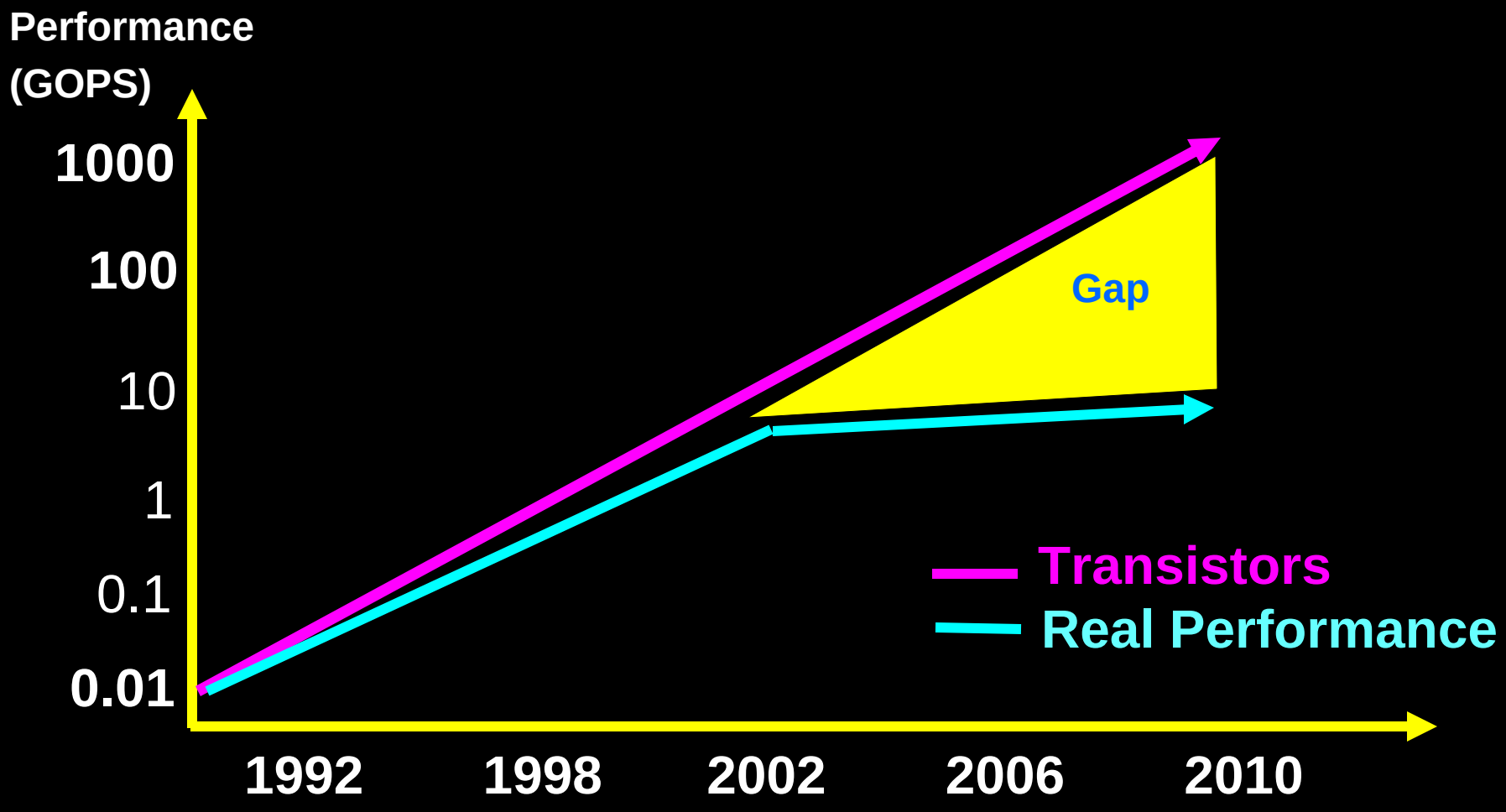
- A personal tour of some languages, compilers, and computers for high performance systems
- **The new performance challenge**
- Addressing the performance challenge
- Discussion

Technology is Hitting a Performance Limit

- Transistors continue to shrink
- More and more transistors fit on a chip
- The chips are faster and faster
- Result: HOT CHIPS!



Real Performance Stops Growing as Fast



Hardware Performance Solution: Multicores

- Simpler, slower, cooler processors (multicores)
- More processors on a chip
- Software (and users) organize tasks to execute in parallel on the processors
- Parallelism will provide the performance!!!

Parallelism

- High performance computing applications and computers have long used parallelism for performance.

  - Current software cannot provide the parallelism needed

  - Users can't either

Two Perspectives on the Performance Challenge

- "The biggest problem Computer Science has ever faced." John Hennessy
- "The best opportunity Computer Science has to improve user productivity, application performance, and system integrity." Fran Allen

Talk outline

- The new performance challenge
- A personal tour of some languages, compilers, and computers for high performance systems
- **Addressing the performance challenge**
- Discussion

Urgent To-Dos

- New, very high level languages
- New compilers
- New compiler techniques to manage data: locality, integrity, ownership, … in the presence of parallelism.
- Eliminate caches
- Remember the John Backus and Grace Hopper goals:
  - User Productivity
  - Program Performance

END OF TALK

BEGINNING OF A NEW ERA IN LANGUAGES and COMPILERS (I HOPE)

End Note

“The fastest way to succeed is to double your failure rate.” – T. J. Watson, Sr.