Women in Compilers and Tools Virtual Meetup Series

From Packing Decimals With Cobol to Optimizing Tweets With Scala: A Journey Through Space, Time, & Culture with Compilers

Uma Srinivasan

June 24, 2021 - 6:00pm PDT

@umatweep

#TwitterVMTeam

What is a Compiler?

X is a compiler

Ryan Cavanaugh
@SeaRyanC

"X is a compiler" alignment chart

"X is a compiler" alignment chart

	Output Purist Output must be binary	**Output Neutral** Output must be instructions	**Output Rebel** Output can be anything
Input Purist Input must be a program	gcc is a compiler	prettier is a compiler	An orchestra is a compiler
Input Neutral Input must be text	Microsoft Word is a compiler	Javadoc is a compiler	AI Dungeon is a compiler
Input Rebel Input can be anything	A coin flip is a compiler	Bop It! is a compiler	The sun is a compiler

12:30 PM · Dec 2, 2020 · Twitter Web App

1,144 Retweets **143** Quote Tweets **3,994** Likes

uma
@umatweep

I'm a compiler, you're a compiler, it's compilers all the way down.

Ryan Cavanaugh @SeaRyanC · Dec 2, 2020
"X is a compiler" alignment chart

"X is a compiler" alignment chart

	Output Purist Output must be binary	**Output Neutral** Output must be instructions
Input Purist Input must be a program	gcc is a compiler	prettier is a compiler

It’s compilers all the way down…..

…...

Compilers in the 1980s….

Compilers in the 1990s….

Compilers in the 2000s….

Compilers in the 2010s….

Compilers in the 2020s….

…….

Compilers in the 1980s ….

- Many programming languages
  - Cobol, RPG, Transact, Pascal, Fortran, C and …… Ada
- 1 target hardware architecture & implementation - PA-RISC
  - A common backend code generator - written in Pascal
  - IRs - Ucode, SLLIC
  - Low level optimizer - both machine dependent & independent opts - in C
- 1 Operating System - HPUX
  - Released once every couple of years
- Cobol Packed Decimal code generation & ANDF
- Questions
  - Why was C following a different codegen path?
  - What should Ada follow?
  - How about a machine independent optimizer?
  - Which backend? Apollo or HP?
  - Automatic code gen?

Hewlett-Packard Journal January 1986 Volume 37 Number 1

June 29, 1987

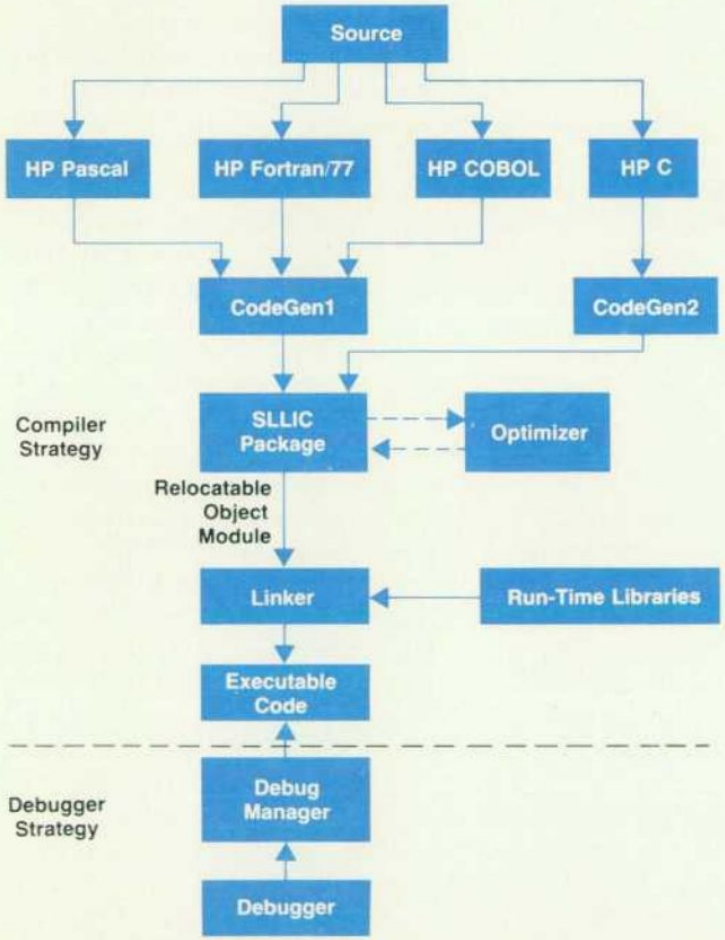


Fig. 2. *The compiler system for HP's new generation of high-precision-architecture computers.*

Patent: [Method for improved code generation in reduced instruction set computers]

```
$subprogram$
$include  'globopts.h'$
$include 'condit.h'$

{   evasc3 contains  procedures to evaluate external decimal operators

  date    who  routine        modification
02-01-86  wbb                 created module
}
{search}
{HCRAES}
program ucode_to_sllic;

const
$include 'opcodes.h'$
$include 'selt.h'$
$include 'compl.h'$
$include 'asrt.h'$
$include 'condcomp.h'$
$include 'uops.h'$
$include 'err.h'$
$include 'opndconsts.h'$
$include 'globals.h'$
$include 'stkopnds.h'$
$include 'sllic.h'$
$include 'util.h'$
$include 'tagutil.h'$
$include 'emit.h'$
$include 'ldutil.h'$
$include 'regutil.h'$
$include 'macroutil.h'$
$include 'cobutil.h'$
$include 'symutil.h'$
$include 'emitbr.h'$
$include 'ascutil1.h'$
$include 'ascutil2.h'$
$include 'ascutil3.h'$
$include 'cmputil.h'$
$include 'evutil.h'$
$include 'ldstr.h'$
$include 'ldstrutil.h'$
$include 'moveutil.h'$
```

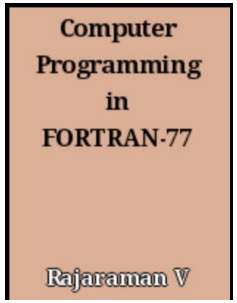
```
{1111111111111111111111111111111111111111111111111111111111111111111111
1
1**** do_gadd_reg
1
1   This level3 evaluator calls millicode to perform external
1   decimal addition.
1
1111111111111111111111111111111111111111111111111111111111111111111111}
function do_gadd_reg( res_loc , l_loc , r_loc : location
                    ; digit_count : integer ):register;
const
   routine = 'do_gadd_reg';
   parm_flags = has_arg0 + has_arg1 + has_arg2 + has_arg3 + has_arg4;
   ret_flags = has_arg5+has_arg6+has_arg7+has_ret0+has_ret1;
var
   tmp_reg,
   len_reg   : register;
   milli_name    : label_type;

begin  { do_gadd_reg }
     { COBOL doesn't use long pointers.. (yet) }
assert( res_loc.base.rct <> 2 , asrt_unimp_long_ptr , assert_failure );
assert( l_loc.base.rct <> 2 , asrt_unimp_long_ptr , assert_failure );
assert( r_loc.base.rct <> 2 , asrt_unimp_long_ptr , assert_failure );
     { allocate alias sets }
open_set;
alias_set(modified_by,res_loc.loc_ta.tags);
alias_set(used_by,l_loc.loc_ta.tags);
alias_set(used_by,r_loc.loc_ta.tags);
     { create pointers }
adjust_location( res_loc );


adjust_location( l_loc );
adjust_location( r_loc );
     { load parms }
emit_gr_copy( res_loc.base.reg , ARG0 );
emit_gr_copy( l_loc.base.reg , ARG1 );
emit_gr_copy( r_loc.base.reg , ARG2 );
len_reg := load_immediate( (digit_count div 4) );
emit_gr_copy( len_reg , ARG3 );
emit_gr_copy( get_ascii_table_ptr , ARG4 );
     { do millicall }
milli_name := '$$g_add_reg';
call_macro(milli_name  , parm_flags , ret_flags , used_by );
     { copy overflow flag out }
tmp_reg := get_scratch_reg();
emit_gr_copy( RET1 , tmp_reg );
do_gadd_reg := tmp_reg;
tmp_reg := get_scratch_reg();
emit_gr_copy(arg5,tmp_reg);
tmp_reg := get_scratch_reg();
emit_gr_copy(arg6,tmp_reg);
tmp_reg := get_scratch_reg();
emit_gr_copy(arg7,tmp_reg);
tmp_reg := get_scratch_reg();
emit_gr_copy(ret0,tmp_reg);
end;   { do_gadd_reg }
```

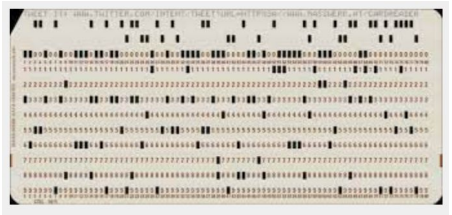
call millicode routine

```
{1111111111111111111111111111111111111111111111111111111111111111111111
1
1**** n_do_gequ_1
1
1       This routine performs an inplace comparison for equality and
1       inequality by testing the last digits of two ascii operands whose
1       lengths are one digit.
1       The last digits are first translated
1       through table-lookup, so as to adjust the values compared for
1       lbrace,rbrace,' ','0', etc.  It handles jump-on-condition as well as
1       the materialization of the boolean result.  Constants are handled
1       specially.
1
1111111111111111111111111111111111111111111111111111111111111111111111}
function n_do_gequ_1(q : quad_ptr;
                     opnd1 : location;  {operand 1's location}
                     opnd2 : location  {operand 2's location} ):
                                              register_descriptor;

const
   routine = 'n_do_gequ_1';
   l_op = 2;
   r_op = 1;

var
   table_ptr,
   tmp1_reg,
   tmp2_reg,


   digit1,
   digit2,
   xdigit1,
   xdigit2                  : register;
   result                   : register_descriptor;
   target_label             : sllic_expr_ptr;
   const_compare            : boolean;
   unsigned                 : boolean;
   tmp_opnd                 : location;
   const_opnd,
   const_aux,
   const_digit,
   const_xdigit             : integer;
   completer1               : completer;

begin {n_do_gequ_1}

$if 'inhouse'$
dump_aux(q^.opnd[leftaux].opnd_aux,'left operand',routine);
dump_aux(q^.opnd[rightaux].opnd_aux,'right operand',routine);
$endif$

result := null_register_descriptor;
result.rct := 1;
result.reg := get_scratch_reg();

{ find out if one operand is constant, and put into canonical form }
const_compare := false;
unsigned := false;
if q^.opnd[l_op].opnd_p^.symtab_class = g_const then
   begin
   const_compare := true;
   const_opnd := l_op;
   const_aux := leftaux;
   unsigned := (q^.opnd[leftaux].opnd_aux^.a_sign = asc_unsign) and
           (q^.opnd[leftaux].opnd_aux^.a_sign_op = asc_no_sign_op);
   end
else if q^.opnd[r_op].opnd_p^.symtab_class = g_const then
   begin
   const_compare := true;
   const_opnd := r_op;
   const_aux := rightaux;
   tmp_opnd := opnd1;
   opnd1 := opnd2;
   opnd2 := tmp_opnd;
   unsigned := (q^.opnd[rightaux].opnd_aux^.a_sign = asc_unsign) and
           (q^.opnd[rightaux].opnd_aux^.a_sign_op = asc_no_sign_op);
   end;
```

debug dumps

From India to the USA

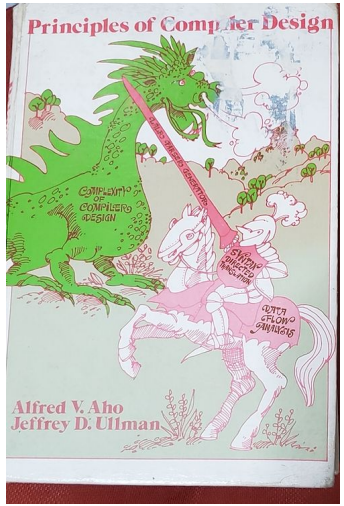




- IIT Madras
  - ***Fortran*** 77, **wrote** programs, **typed** on punch cards, IBM 360
  - [8080 *assembly*]
  - VAX workstation, prototyped x86 code generator in ***C***
  - Computers were only available in air conditioned labs, quite a distance from the dorm

U. of Wisconsin, Madison

- First compiler course, project - developing an ***Ada*** code generator
- Computer labs in the basement of the department building
- TA in the Mac lab - ***Pascal*** compiler UI reported syntax errors even as you typed - WoW!



HP Work Culture *The HP Way*

- Team work
  - Entirely local team - HP Cupertino Bldg 47 Lower
  - Favorite restaurants: Cicero's, La Fiesta, Florentine's,...
    - hardly any vegetarian and no vegan options then
  - Apollo acquisition and friction
  - Open cubicles, code printouts, dumb terminals, solid wooden desk to hide under - Loma Prieta earthquake
  - Engineers did not travel except for conferences
- Women were respected and in influential positions in CLL (Compiler Language Lab)
  - My interviewers, first and hiring manager, second level manager, lots of women managers, no dual ladder in the engineering labs, family friendly
- Benefits and Incentives
  - Profit sharing, ESPP, Stock awards and options
  - Leave accumulation across years

Role Models, Mentors, Colleagues, Friends and Family

GANAPATHI, M., AND FISCHER, C. N. "Description-driven code generation using attribute grammars." In *Proc. 9th Ann. ACM Symp. Principles of Programming Languages* (Albuquerque, New Mex., Jan. 25–27), ACM, New York, 1982.

GANAPATHI, M., FISCHER, C. N., AND HENNESSY, J. L. Retargetable compiler code generation. *ACM Comput. Surv. 14*, 4 (Dec. 1982), 573–592.

[Fran Allen](#)

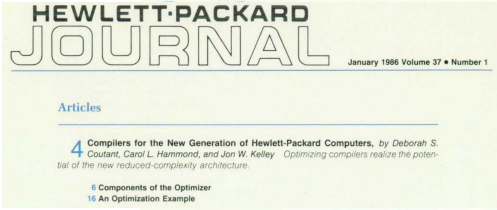
[Ada](#)
[#AdaLoveLaceDay](#)

Aho, A. V., Ganapathi, M., and Tjiang, S. W. K., "Code Generation Using Tree Matching and Dynamic Programming," ACM Trans. Program Lang. Syst., Vol. 2, No. 4, Oct. 1989, pp. 491-561.

Role Models, Mentors, Colleagues, Friends and Family

William B. Buzbee

With HP since 1984, Bill Buzbee has worked on code generation and optimization for HP Precision Architecture. Before joining HP he was a journalist and held positions ranging from sports writer to managing editor of a small daily newspaper. He is named inventor for a patent application related to a new method for generating code to perform complex operations. Bill was born in Chanute, Kansas and educated at the University of Kansas (BS journalism 1980 and MS computer science 1984). He's married and lives in Milpitas, California.



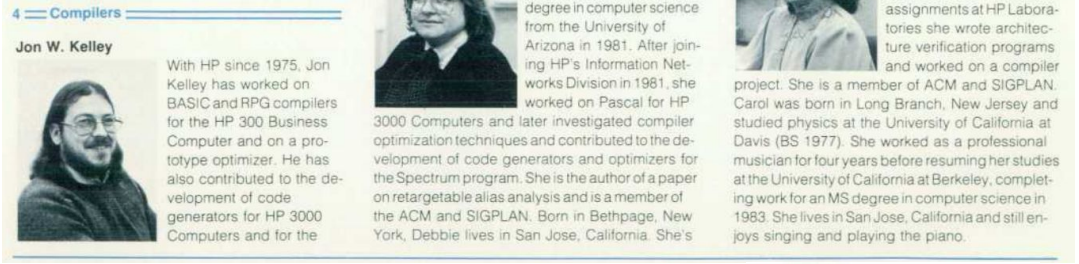
HEWLETT-PACKARD JOURNAL

January 1986 Volume 37 • Number 1

Articles

4 Compilers for the New Generation of Hewlett-Packard Computers, *by Deborah S. Coutant, Carol L. Hammond, and Jon W. Kelley* Optimizing compilers realize the potential of the new reduced-complexity architecture.

6 Components of the Optimizer
16 An Optimization Example



4 Compilers

Jon W. Kelley

With HP since 1975, Jon Kelley has worked on BASIC and RPG compilers for the HP 300 Business Computer and on a prototype optimizer. He has also contributed to the development of code generators for HP 3000 Computers and for the Spectrum program. Jon graduated in 1974 from the University of California at Berkeley with a BA degree in computer science. He lives in Sunnyvale, California and lists fly-fishing, hunting, and flying as outside interests.

Deborah S. Coutant

Debbie Coutant earned a BA degree in psychology from the University of Arizona in 1977 and an MS degree in computer science from the University of Arizona in 1981. After joining HP's Information Networks Division in 1981, she worked on Pascal for HP 3000 Computers and later investigated compiler optimization techniques and contributed to the development of code generators and optimizers for the Spectrum program. She is the author of a paper on retargetable alias analysis and is a member of the ACM and SIGPLAN. Born in Bethpage, New York, Debbie lives in San Jose, California. She's married and enjoys playing the French horn in community orchestras. Her other outside interests include racquetball and camping.

Carol L. Hammond

With HP since 1982, Carol Hammond manages an optimizer project in the computer language laboratory of HP's Information Technology Group. In earlier assignments at HP Laboratories she wrote architecture verification programs and worked on a compiler project. She is a member of ACM and SIGPLAN. Carol was born in Long Branch, New Jersey and studied physics at the University of California at Davis (BS 1977). She worked as a professional musician for four years before resuming her studies at the University of California at Berkeley, completing work for an MS degree in computer science in 1983. She lives in San Jose, California and still enjoys singing and playing the piano.

[Hewlett-Packard Precision Architecture Compiler Performance](#) *by Karl W. Pettis and William B. Buzbee*

[27] Jain, Suneel and Thompson, Carol, "An Efficient Approach to Data Flow Analysis in a Multiple Pass Global Optimizer", *PLDI*, June 1988.

[37] Pettis, Karl and Hansen, Robert, "Profile Guided Code Positioning", *PLDI*, June 1990.

[Register Reassociation in PA-RISC Compilers](#) *by Vatsa Santhanam*

Karl W. Pettis

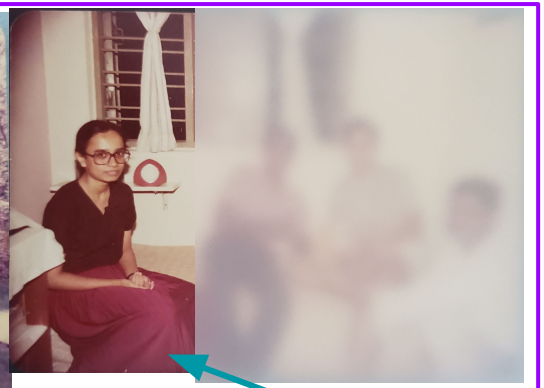
Born in Gainesville, Florida, Karl Pettis attended Michigan State University and completed work for a BS degree in mathematics in 1975 and an MS degree in computer science in 1977. He continued his studies at Yale University and was awarded another MS computer science degree in 1978. He also did PhD-level work at the University of Arizona before joining HP in 1981. In addition to his work on the optimizer for HP Precision Architecture, he has contributed to the development of HP Business BASIC and HP MemoMaker. He's coauthor of a technical paper on pattern recognition. A resident of San Jose, California, Karl likes games, comic books, and music by Gilbert and Sullivan.

Vatsa Santhanam

A computer language engineer/scientist at HP's California language laboratory, Vatsa Santhanam works on the design and implementation of compiler optimization techniques. He joined HP in 1984 at HP's Santa Clara Division. While there he worked as a test system software design engineer on a VLSI tester project. He has since worked on different optimization projects including an investigation of interprocedural optimizations. He also worked on a project that produced HP's response to an Open Software Foundation's request for technology for an architecture neutral software distribution format (ANDF). He received a Bachelors of Technology degree in electrical engineering (1982) from the Indian Institute of Technology in Madras, India and an MS in computer science (1984) from the University of Wisconsin at Madison. He also worked as a teaching assistant at the University of Wisconsin. He has coauthored three papers on compiler technology and is named as a coinventor on patent applications for an interprocedural register allocation technique and an architecture neutral distribution format. Vatsa was born in Madras, India and grew up in Japan and Hong Kong. He is married, and when he is not pursuing his professional interests in compilers and computer architecture, he likes to play chess, dabble in Hindu astrology, and listen to Indian classical

IIT-M

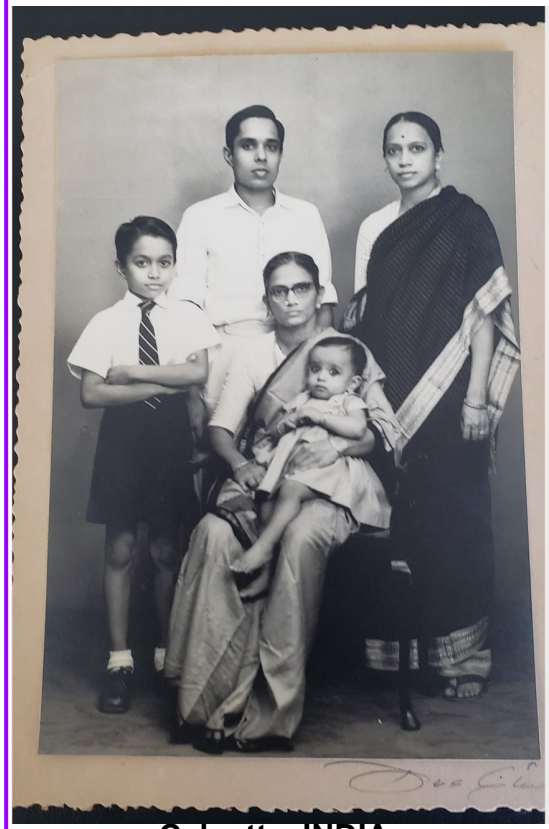
campus



dorm

one of 6 girls in a batch of 240 students!!

USA
wearing
contact lenses



Calcutta, INDIA

Univ. of Wisconsin Madison - *Commons*

look, a CRAY-2 poster

my personal phone :-)

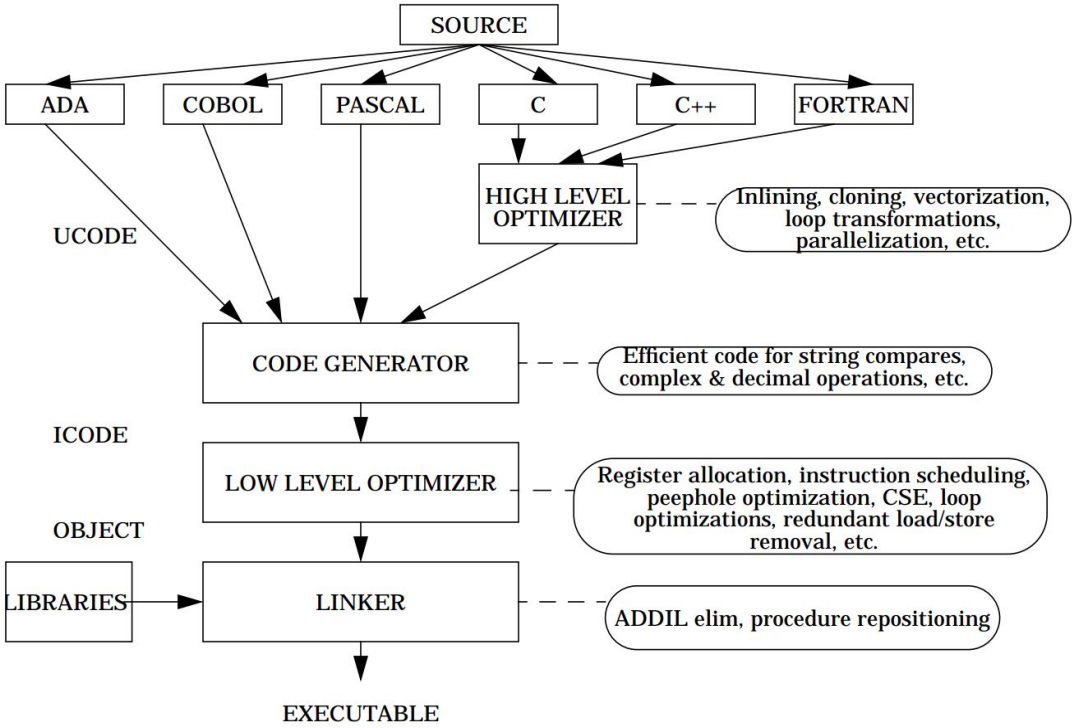
dancing \|/

Santa Clara, CA

ME

Compilers in the 1990s

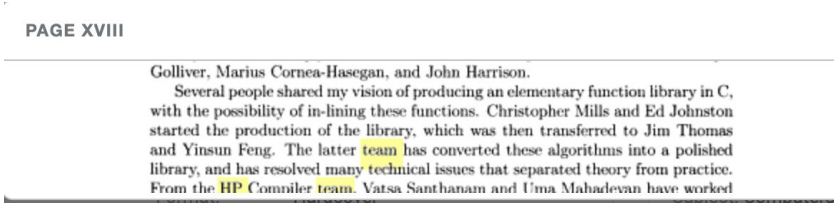
- High Level Optimizer
- Code Generator - in C
- 1 LLO per target machine architecture
  - Loop unrolling, Instruction Scheduling, Code layout
- My first publication & patent CC' 94, Intelligent Loop Unrolling



Compiler Optimizations for the PA-8000 - IEEE COMPCON 97

Compilers in the 1990s

- The Itanium compiler - EPIC code generation and optimization
- Compiling for IA-64
  - Modulo Scheduling
  - Optimizing IA-64 Math Functions →
- PACT 2000 papers & patents
  - Control & Data Speculation framework
  - Modulo Scheduling
    - Uncounted loops
    - Data Speculation
    - Rotating register assignment



PAGE XVIII

Golliver, Marius Cornea-Hasegan, and John Harrison.

Several people shared my vision of producing an elementary function library in C, with the possibility of in-lining these functions. Christopher Mills and Ed Johnston started the production of the library, which was then transferred to Jim Thomas and Yinsun Feng. The latter team has converted these algorithms into a polished library, and has resolved many technical issues that separated theory from practice. From the HP Compiler team, Vatsa Santhanam and Uma Mahadevan have worked

Work Culture

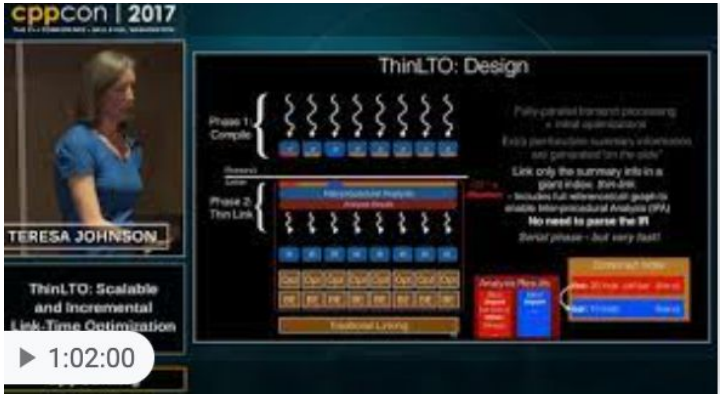
- Dual ladder for career advancement
- Teamwork and collaboration across the country, orgs and companies
  - CLL (California), MLL (Massachusetts) and then Texas (Compaq) compiler teams
  - HP Research Labs and Production R&D teams
  - HP & Intel Itanium compiler teams
- Benefits
  - Family leave, Telecommuting (ISDN), beginnings of WFH
  - Significant other

Role Models, Mentors, Colleagues, Friends and Family

[Bob Rau](), [Peter Markstein](), HP Labs

Anne is a member of technical staff at Uber. Anne has over 20 years of experience as a software engineer at companies including VMware, Omnishift Technologies, Transmeta and HP. She holds a master's degree in computer science from Duke University and a Doctorate in computer science from University of Virginia.

Anne Holler
Software Engineer
Uber

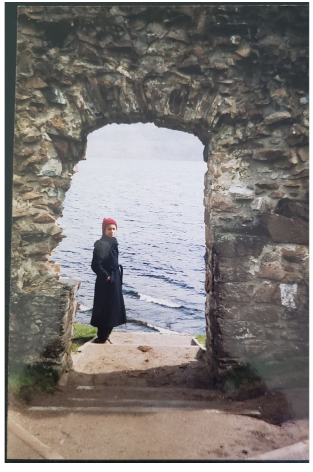


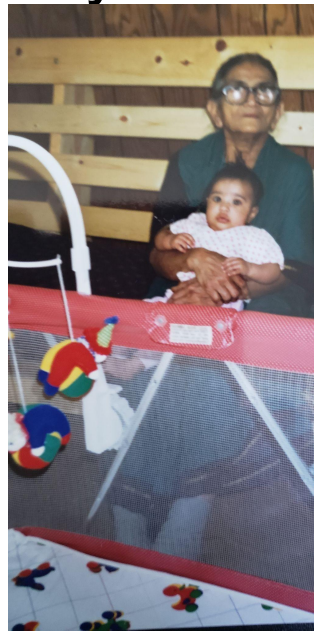
Role Models, Mentors, Colleagues, Friends and Family

Madras, India

Oregon?, USA

Scotland - CC'94





California, USA

Compilers in the 2000s

- Middle-ends in production - first paper in CGO

- Java JITs
  - First JavaOne talk "Improving the performance of Java technology on IA-64 processors" - JavaOne 2001
  - Oracle/Sun Hotspot JVM on Itanium Linux/Windows
    - Intel/Sun joint session "Maximizing Enterprise Java™ Performance on Multi-core Platforms" - JavaOne 2008
    - PMU based optimizations

- Hardware software co-design in earnest
  - Performance study of 2 bioinformatics applications on x86
  - In-order Atom (x86) processor compilation - superblock scheduling
  - Hardware atomicity for reliable software speculation - ISCA 2007

Intel Work Culture

- Shifting from being entirely Hardware centric to one of growing Software awareness
  - In production TIMING is everything
    - roadmaps, TPMs & PMs galore, Disagree & Commit
  - More women being hired into the workforce
  - Research labs under pressure to connect with production teams
- Globalization of engineering workforce
  - had managers in Oregon, Folsom, east coast
  - team members across the US, China, Russia, Australia, Argentina, Poland, Israel
  - Intel Jet available to all employees - frequent day trips to Hillsboro (OR), Folsom
  - Travels to Novosibirsk, Gdansk
- Results oriented, Meritocracy
- Women @ Intel
  - Anita Borg Institute - GHC - a panel on HW/SW co-design, met with Fran Allen
  - First Women Fellows
- Benefits
  - RSUs, Sabbatical leave

Explaining *"Shangri-La – Domain Specific Programming System for Multi-core Architectures"* to Paul Ottelini on **Research @ Intel Day 2005**

Role Models, Mentors, Colleagues, Friends and Family

Women Who Code @WomenWhoCode · Mar 2

👩‍🦰Grace Hopper was one of the first programmers of the Harvard Mark I computer and invented the first compiler for a computer programming language!

#InternationalWomensHistoryMonth #IWD #IWD2021 #WomeninSTEM #WomeninScience

Association for Computing Machinery @TheOffici... · Dec 9, 2020

Grace Hopper was born on this day in 1906. Hopper helped devise the theory of machine-independent programming languages. She was instrumental in the development of the COBOL, which went on to become the most ubiquitous language for business ever.

[Carole DuLong](), [Tatiana Shpeisman](),

Intel Research Labs

with Sandhya Viswanathan - *Intel JVM Compiler Engineer*

Role Models, Mentors, Colleagues, Friends and Family

The lineage of Gurus and my Chinmaya family

2 nieces graduated: MS & PhD, Computer Science majors, UCB, Stanford

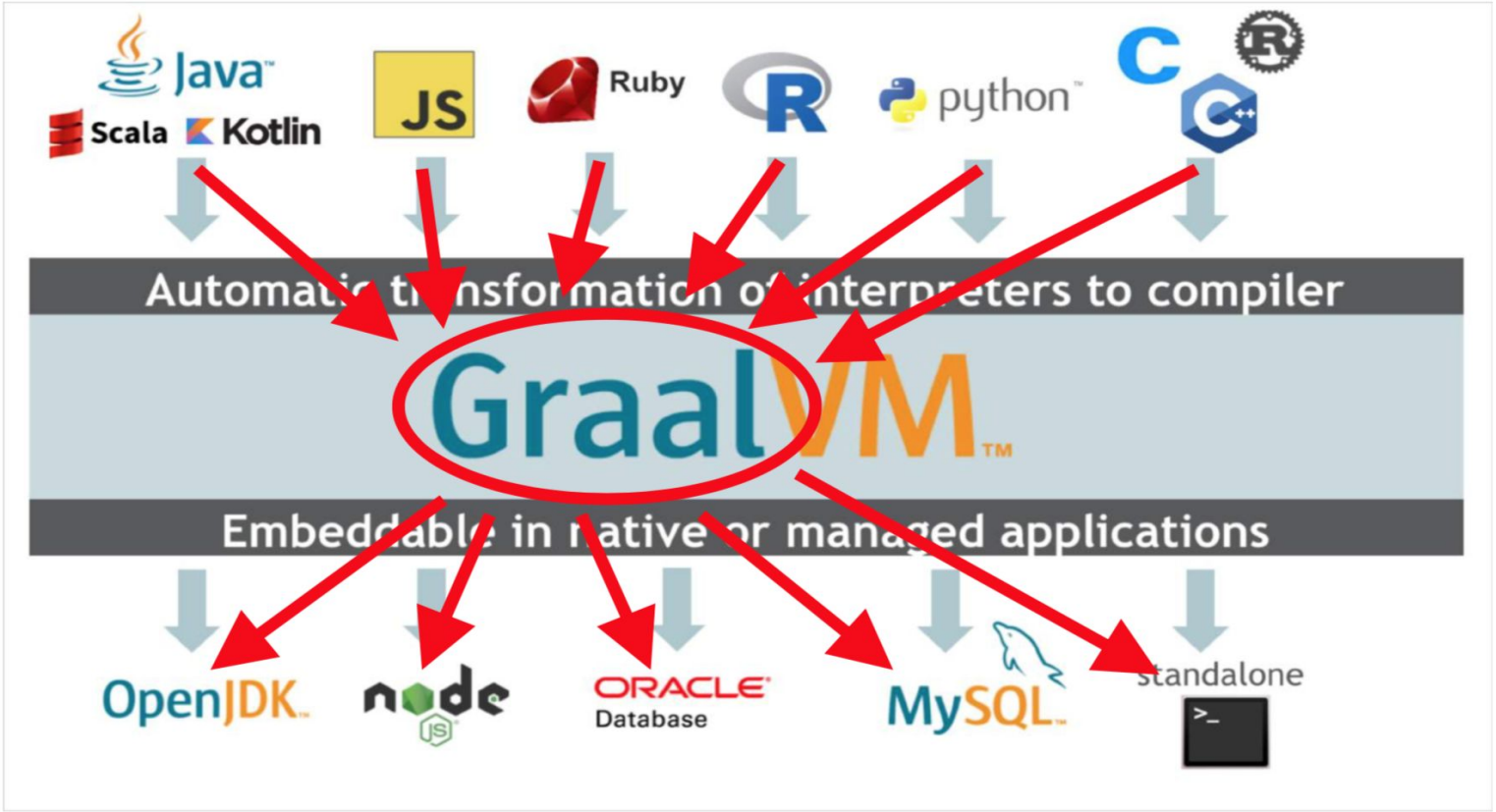
Compilers in the 2010s

- HW/SW co-design continues
  - With GPUs
    - CGO 2012 - Keynote - Hardware Software Co-design for Visual Computing
    - Gen Graphics Architecture Development
      - Performance impact of Gen architectural features evaluated to provide recommendations
      - FP16 performance evaluation, feature enhancements on Gen8+ architectures with 3D (OpenGL) and Compute (OpenCL) workloads
    - New LLVM based shader compiler - IGC
  - Dataflow machines - patent
    - LLVM based code generation prototype for spatial accelerators
    - Co-design with an ex-VAX/Alpha HW architect and an ex-Multiflow/Itanium compiler architect

- Scala compilation and tools
  - Front end written in Scala
    - Heavy dependence on IDE and tools - Scalafix @ Twitter scale
  - VM Graal compiler - written in Java - the Graal workshop @ CGO
    - Twitter & Graal, Scala ❤ Graal, Autovectorization, Neuroevolution based inlining

First Graal workshop @ CGO 2019 (Keynote)

What is Graal?



Graal workshop @ CGO 2019 (Keynote)

Applications of Graal

- JIT compiler for apps written in JVM languages (openjdk/hotspotVM/JVMCI)
- Specialized compiler for JVM apps (jruby-graal)
- AOT compiler for apps in JVM languages (subtrateVM/native-image)
- JIT for dynamic languages
- JIT for native languages
- Tool for embedding languages (Oracle DB, MySQL)

Twitter Culture

#OneTeam

- A new meaning for Inclusion & Diversity
  - BRGs - @womeng, #twitter-women, #twitter-asians, #twitter-faith, #twitter-open, ….
  - from Boomers to Gen Z
  - #TechWomen, #maleallies
  - #WomenWhoCode, #GirlsWhoCode
- Lines fudged
  - Work and Family
  - Onsite open offices and Home offices
  - Customers and employees
  - Humans, pets and plants belong in the family - each have their own identity on Twitter ;-)
- Benefits
  - Meditation rooms
  - Unlimited leave
  - Free lunches

#TechWomen @twitter 2018

Role Models, Mentors, Colleagues, Friends and Family

@intel

- Sylvia Downing - GPU Architect and the GPU architecture team
- the Graphics Compiler (IGC) team - several women engineers on this team
- the Dataflow architecture team

Role Models, Mentors, Colleagues, Friends and Family

@twitter

- @jack and staff … Leslie, Vijaya
- @jenniferfraser (#TechWomen), @kevino (#maleallies)
- @womeng friends - @dordogh, @catia3045, @elizdeng
- @igb & the #TwitterVMTeam - Nora, Yunjie, Maura

Role Models, Mentors, Colleagues, Friends and Family

- daughter graduated: BS & MEd,
  Cognitive Science & Education majors, UCLA

YAY

- 2 grand nephews &
  1 grand niece

Compilers in the 2020s…..

- Compilers for ML
  - C4ML workshop @ CGO
- ML for Compilers
  - Autotune - Performance tuning Twitter services with Graal and Machine Learning
  - Improving Compiler optimizations by employing machine learning
    - Graal workshop @ CGO 2021
  - Compiler 2.0: Using ML to modernize compiler technology - C4ML workshop @ CGO 2020
- Compilers for Accelerators, TPUs, …
  - The Golden Age of Compiler Design… - ASPLOS 2021 keynote
- Twitter Women@ML

My Learnings

Life is one heck of a Deep Learning Neural Network

….experiences/inferences depend on the input/training received...

…..heavily weighted by role models, mentors, colleagues, friends & family

What are yours?

Thoughts may be fleeting but Tweets live forever…..

@umatweep would love to get your tweets #WiCTUma21