Computer Architecture of the 1960's

presentation at StrangeLoop 2012

Revisions to slides after talk

- •Changed slide to say Curry, not Cleary {a senior moment}

•Added some Burroughs ads

•Added endnotes

- URLs to Burroughs history
- URLs to classic Haskell talks/info

-Presenter Carlton Mills

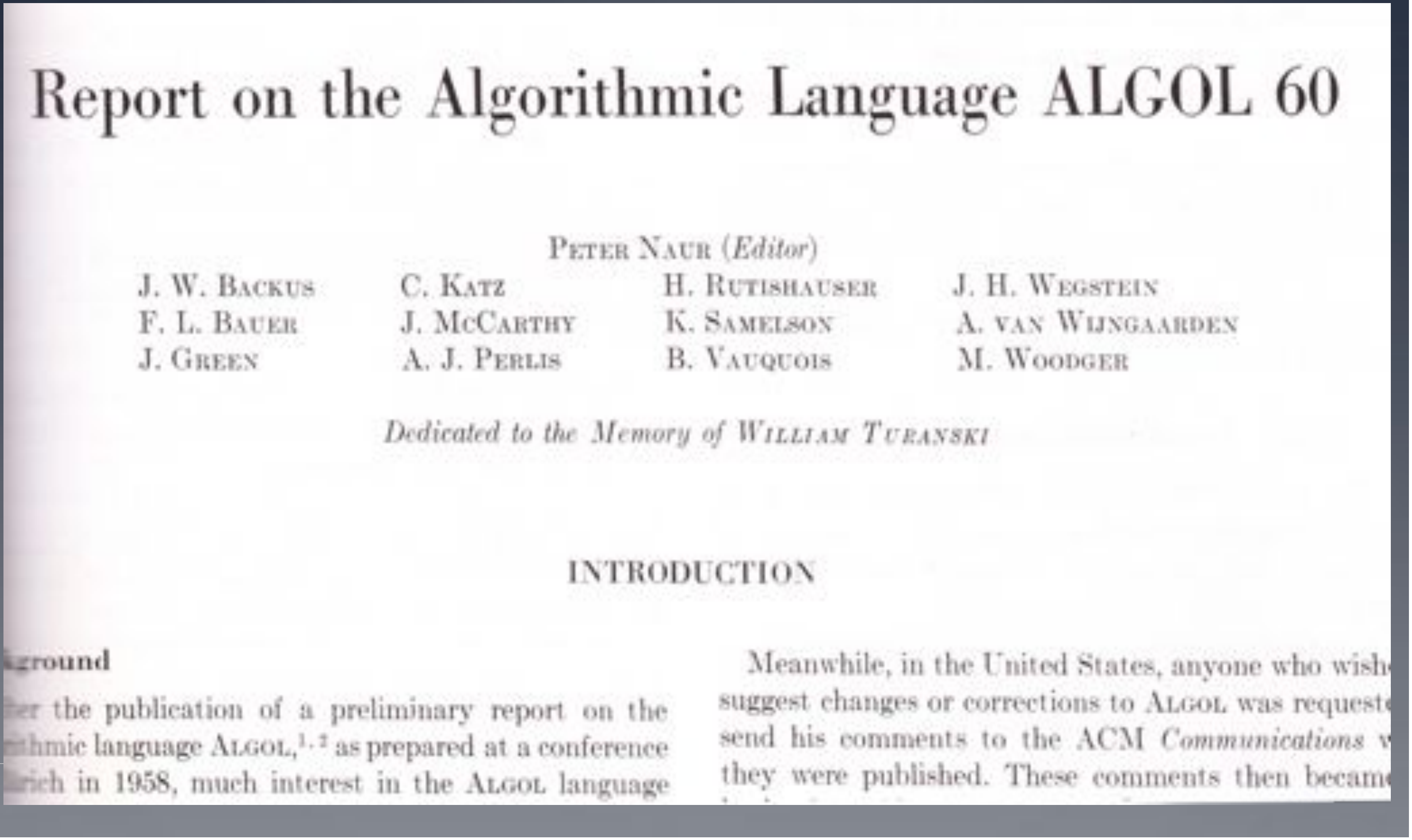
Resume

- Carlton Mills

1963 Programmed vacuum tube IBM 650
1966 Programmer for Civil Engineering - Wrote payroll system in Fortran on an IBM 1620 (punched cards only – no disk or tape)
1968 System Programming B-5500
1969 System programmer for Illiac IV project
1974 Data Processing Manager National Bahá'í Center, Wilmette, IL
1978 Blue Cross/Blue Shield computer network
1983 Gcom (X.25,SNA,SDLC support, wrote Bisync – all in C)
2005 Retired

Algol 60 Report

CACM May 1960



Report on the Algorithmic Language ALGOL 60

PETER NAUR (*Editor*)

J. W. BACKUS C. KATZ H. RUTISHAUSER J. H. WEGSTEIN
F. L. BAUER J. MCCARTHY K. SAMELSON A. VAN WIJNGAARDEN
J. GREEN A. J. PERLIS B. VAUQUOIS M. WOODGER

Dedicated to the Memory of WILLIAM TURANSKI

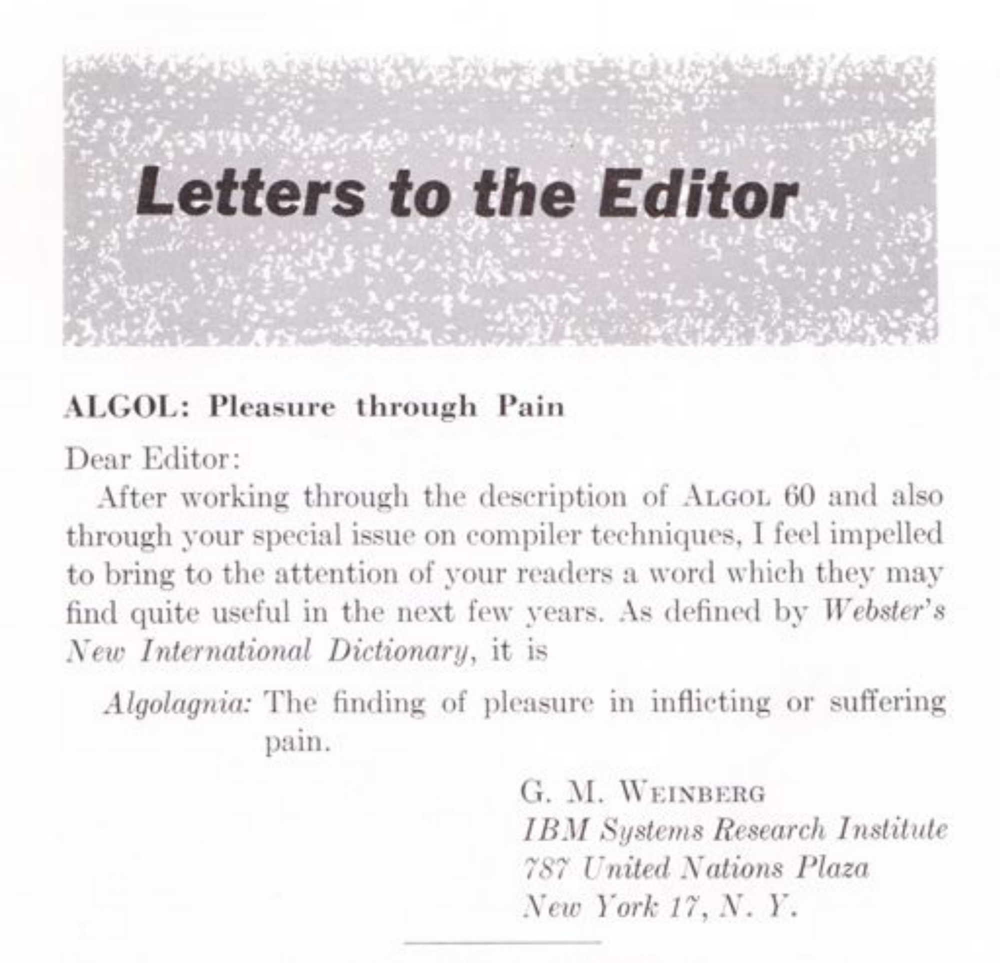
INTRODUCTION

ground

ter the publication of a preliminary report on the ithmic language ALGOL,[1,2] as prepared at a conference rich in 1958, much interest in the ALGOL language

Meanwhile, in the United States, anyone who wish suggest changes or corrections to ALGOL was request send his comments to the ACM *Communications* w they were published. These comments then becam

Far seeing letter



Letters to the Editor

ALGOL: Pleasure through Pain

Dear Editor:

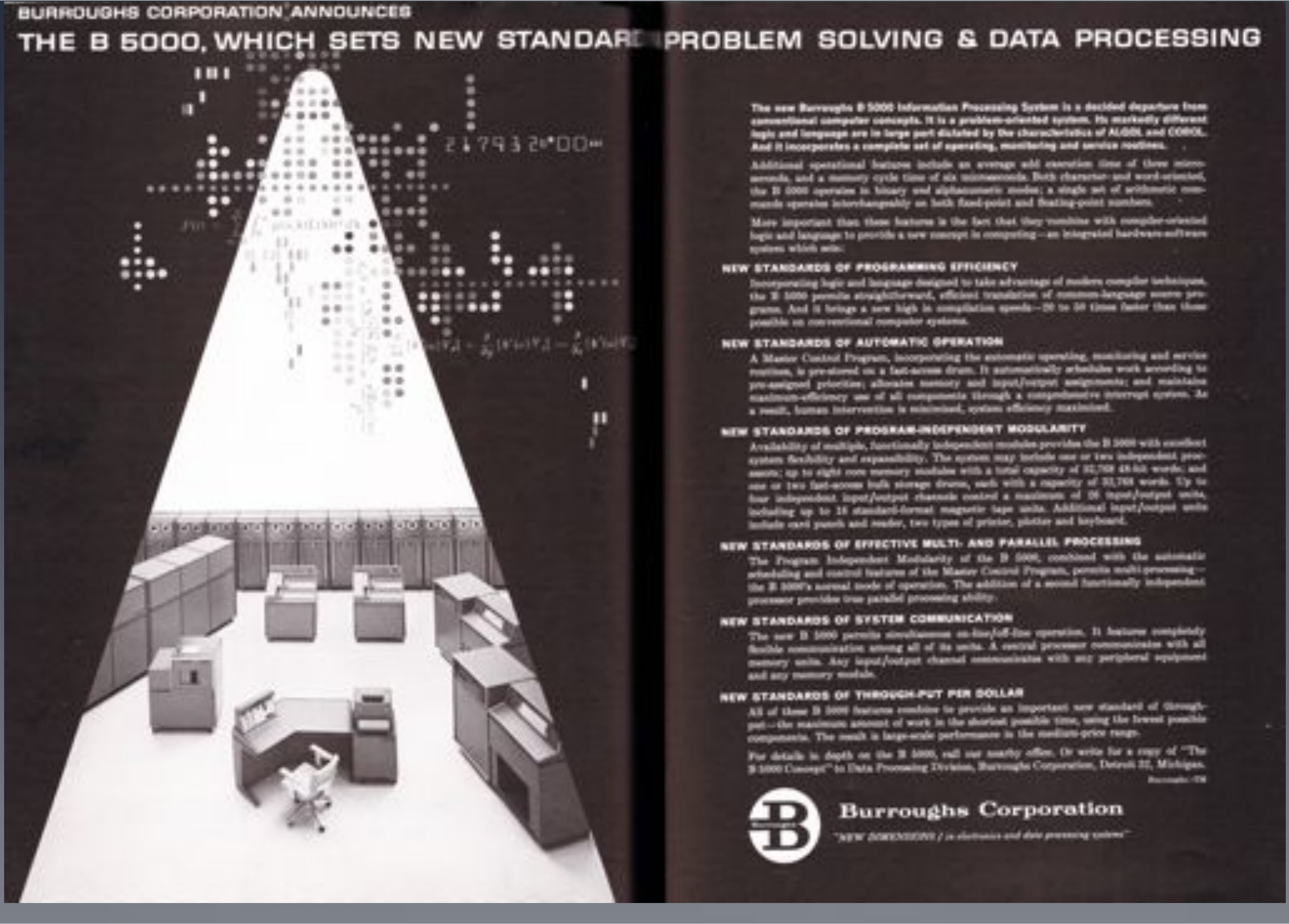
After working through the description of ALGOL 60 and also through your special issue on compiler techniques, I feel impelled to bring to the attention of your readers a word which they may find quite useful in the next few years. As defined by *Webster's New International Dictionary*, it is

Algolagnia: The finding of pleasure in inflicting or suffering pain.

G. M. WEINBERG
IBM Systems Research Institute
787 United Nations Plaza
New York 17, N. Y.

First B-5000 ad

CACM March 1961



BURROUGHS CORPORATION ANNOUNCES

THE B 5000, WHICH SETS NEW STANDARD PROBLEM SOLVING & DATA PROCESSING

The new Burroughs B 5000 Information Processing System is a decided departure from conventional computer concepts. It is a problem-oriented system. Its markedly different logic and language are in large part dictated by the characteristics of ALGOL and COBOL. And it incorporates a complete set of operating, monitoring and service routines.

Additional operational features include an average add execution time of three microseconds, and a memory cycle time of six microseconds. Both character- and word-oriented, the B 5000 operates in binary and alphanumeric modes; a single set of arithmetic commands operates interchangeably on both fixed-point and floating-point numbers.

More important than these features is the fact that they combine with compiler-oriented logic and language to provide a new concept in computing—an integrated hardware-software system which sets:

NEW STANDARDS OF PROGRAMMING EFFICIENCY

Incorporating logic and language designed to take advantage of modern compiler techniques, the B 5000 permits straightforward, efficient translation of common-language source programs. And it brings a new high in compilation speeds—20 to 50 times faster than those possible on conventional computer systems.

NEW STANDARDS OF AUTOMATIC OPERATION

A Master Control Program, incorporating the automatic operating, monitoring and service routines, is pre-stored on a fast-access drum. It automatically schedules work according to pre-assigned priorities; allocates memory and input/output assignments; and maintains maximum-efficiency use of all components through a comprehensive interrupt system. As a result, human intervention is minimized, system efficiency maximized.

NEW STANDARDS OF PROGRAM-INDEPENDENT MODULARITY

Availability of multiple, functionally independent modules provides the B 5000 with excellent system flexibility and expansibility. The system may include one or two independent processors; up to eight core memory modules with a total capacity of 32,768 48-bit words; and one or two fast-access bulk storage drums, each with a capacity of 32,768 words. Up to four independent input/output channels control a maximum of 32 input/output units, including up to 16 standard-format magnetic tape units. Additional input/output units include card punch and reader, two types of printer, plotter and keyboard.

NEW STANDARDS OF EFFECTIVE MULTI- AND PARALLEL PROCESSING

The Program Independent Modularity of the B 5000, combined with the automatic scheduling and control features of the Master Control Program, permits multi-processing—the B 5000's normal mode of operation. The addition of a second functionally independent processor provides true parallel processing ability.

NEW STANDARDS OF SYSTEM COMMUNICATION

The new B 5000 permits simultaneous on-line/off-line operation. It features completely flexible communication among all of its units. A central processor communicates with all memory units. Any input/output channel communicates with any peripheral equipment and any memory module.

NEW STANDARDS OF THROUGH-PUT PER DOLLAR

All of these B 5000 features combine to provide an important new standard of through-put—the maximum amount of work in the shortest possible time, using the fewest possible components. The result is large-scale performance in the medium-price range.

For details in depth on the B 5000, call our nearby office. Or write for a copy of "The B 5000 Concept" to Data Processing Division, Burroughs Corporation, Detroit 32, Michigan.

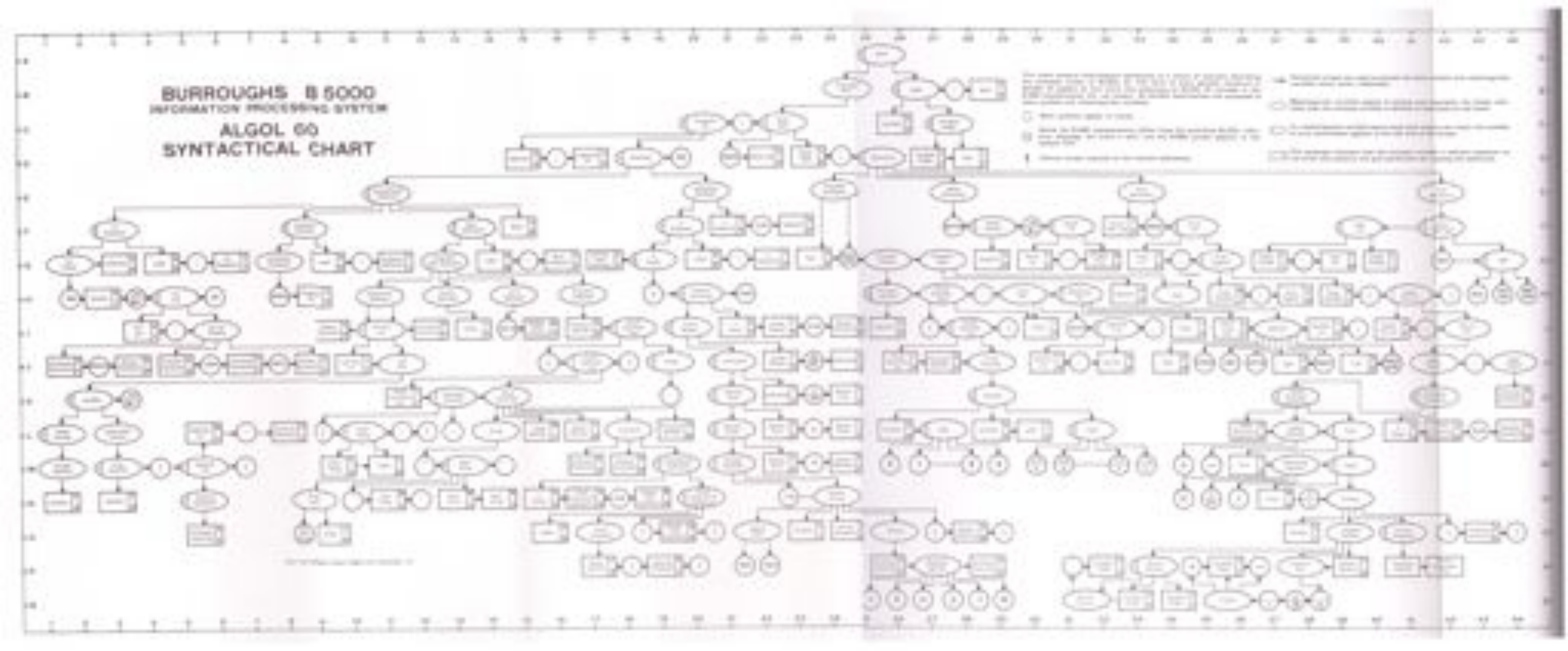
Burroughs—TM

Burroughs Corporation

"NEW DIMENSIONS / in electronics and data processing systems"

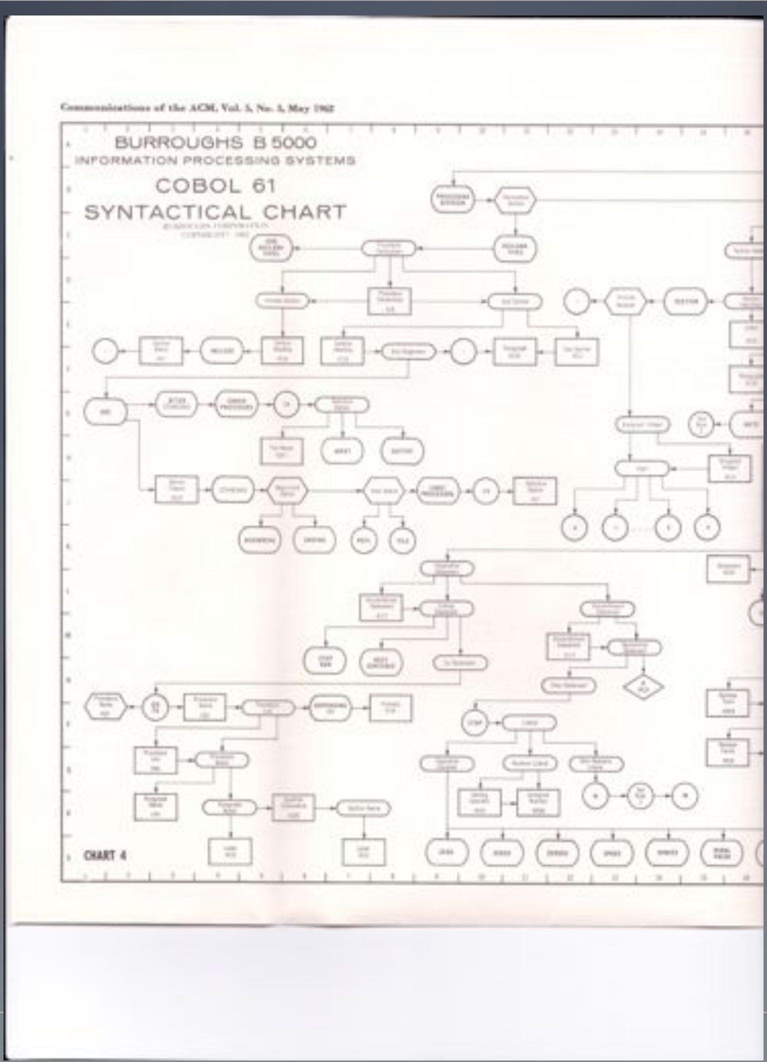
Algol syntax chart

CACM Sep 1961



<?> syntax chart

CACM May 62



Missing ad

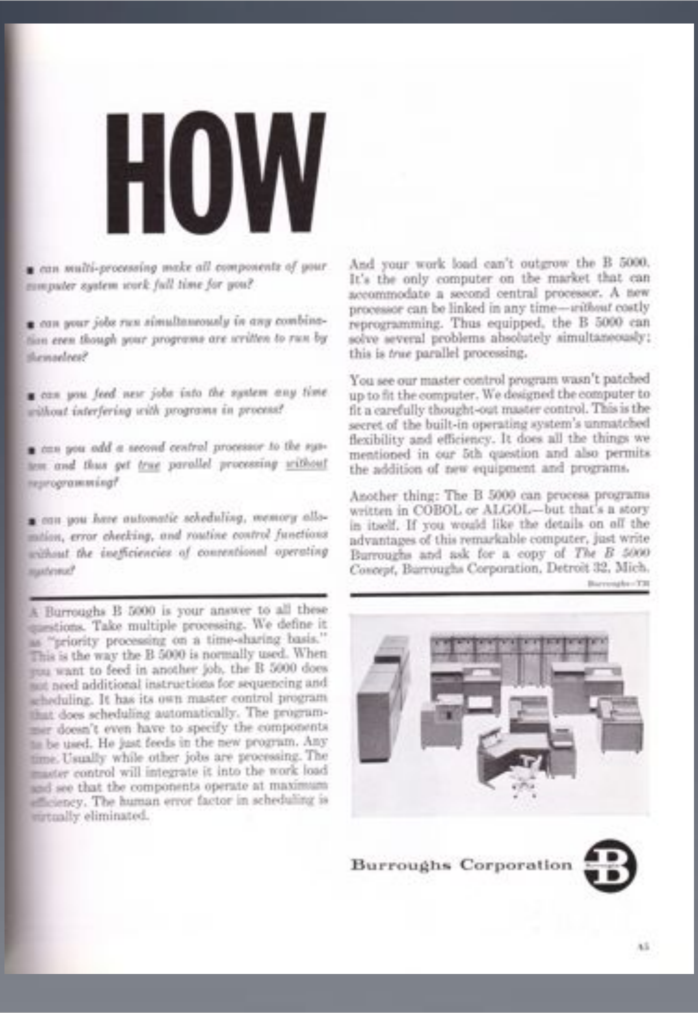
You are now reading the only language you need to now to program the Burroughs B-5000.

```
MULTIPLY PAY-RATE TIMES HOURS-WORKED GIVING TOTAL-PAY.
```

- I saw a Burroughs ad where the above was the main text. But I could not find it to scan for today's presentation.

HOW

CACM Feb 62



HOW

- *can multi-processing make all components of your computer system work full time for you?*

- *can your jobs run simultaneously in any combination even though your programs are written to run by themselves?*

- *can you feed new jobs into the system any time without interfering with programs in process?*

- *can you add a second central processor to the system and thus get true parallel processing without reprogramming?*

- *can you have automatic scheduling, memory allocation, error checking, and routine control functions without the inefficiencies of conventional operating systems?*

A Burroughs B 5000 is your answer to all these questions. Take multiple processing. We define it as "priority processing on a time-sharing basis." This is the way the B 5000 is normally used. When you want to feed in another job, the B 5000 does not need additional instructions for sequencing and scheduling. It has its own master control program that does scheduling automatically. The programmer doesn't even have to specify the components to be used. He just feeds in the new program. Any time. Usually while other jobs are processing. The master control will integrate it into the work load and see that the components operate at maximum efficiency. The human error factor in scheduling is virtually eliminated.

And your work load can't outgrow the B 5000. It's the only computer on the market that can accommodate a second central processor. A new processor can be linked in any time—*without* costly reprogramming. Thus equipped, the B 5000 can solve several problems absolutely simultaneously; this is *true* parallel processing.

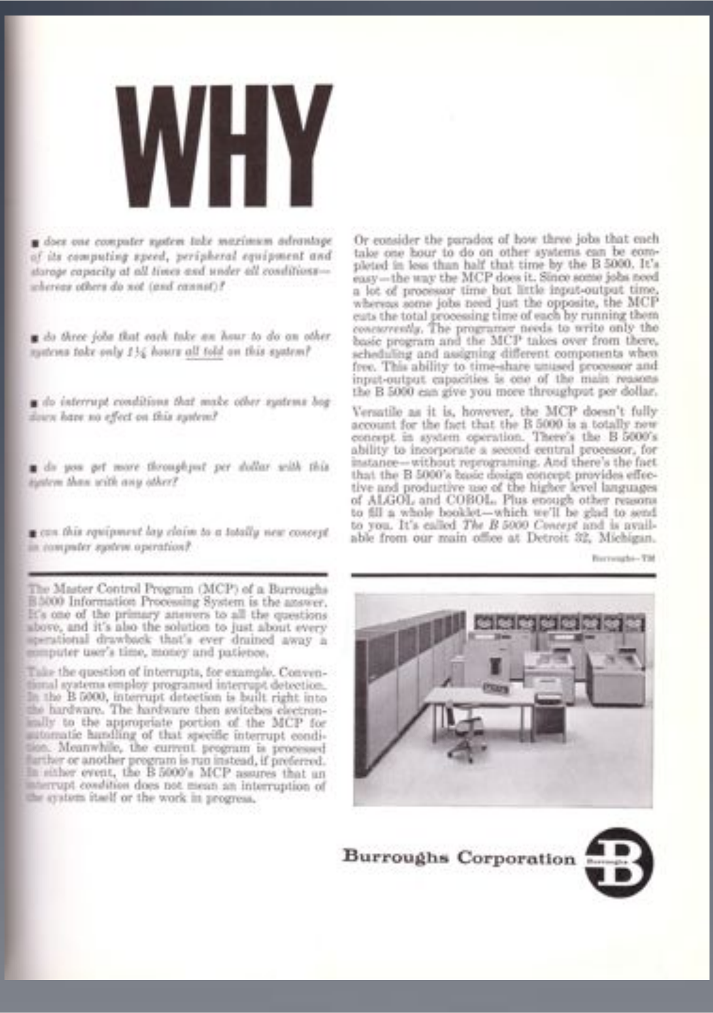
You see our master control program wasn't patched up to fit the computer. We designed the computer to fit a carefully thought-out master control. This is the secret of the built-in operating system's unmatched flexibility and efficiency. It does all the things we mentioned in our 5th question and also permits the addition of *new* equipment and programs.

Another thing: The B 5000 can process programs written in COBOL or ALGOL—but that's a story in itself. If you would like the details on *all* the advantages of this remarkable computer, just write Burroughs and ask for a copy of *The B 5000 Concept*, Burroughs Corporation, Detroit 32, Mich.

Burroughs—TM

Burroughs Corporation

Why
CACM May 1962



WHY

■ *does one computer system take maximum advantage of its computing speed, peripheral equipment and storage capacity at all times and under all conditions—whereas others do not (and cannot)?*

■ *do three jobs that each take an hour to do on other systems take only 1¼ hours all told on this system?*

■ *do interrupt conditions that make other systems bog down have no effect on this system?*

■ *do you get more throughput per dollar with this system than with any other?*

■ *can this equipment lay claim to a totally new concept in computer system operation?*

The Master Control Program (MCP) of a Burroughs B 5000 Information Processing System is the answer. It's one of the primary answers to all the questions above, and it's also the solution to just about every operational drawback that's ever drained away a computer user's time, money and patience.

Take the question of interrupts, for example. Conventional systems employ programed interrupt detection. In the B 5000, interrupt detection is built right into the hardware. The hardware then switches electronically to the appropriate portion of the MCP for automatic handling of that specific interrupt condition. Meanwhile, the current program is processed further or another program is run instead, if preferred. In either event, the B 5000's MCP assures that an interrupt *condition* does not mean an interruption of the system itself or the work in progress.

Or consider the paradox of how three jobs that each take one hour to do on other systems can be completed in less than half that time by the B 5000. It's easy—the way the MCP does it. Since some jobs need a lot of processor time but little input-output time, whereas some jobs need just the opposite, the MCP cuts the total processing time of each by running them *concurrently*. The programer needs to write only the basic program and the MCP takes over from there, scheduling and assigning different components when free. This ability to time-share unused processor and input-output capacities is one of the main reasons the B 5000 can give you more throughput per dollar.

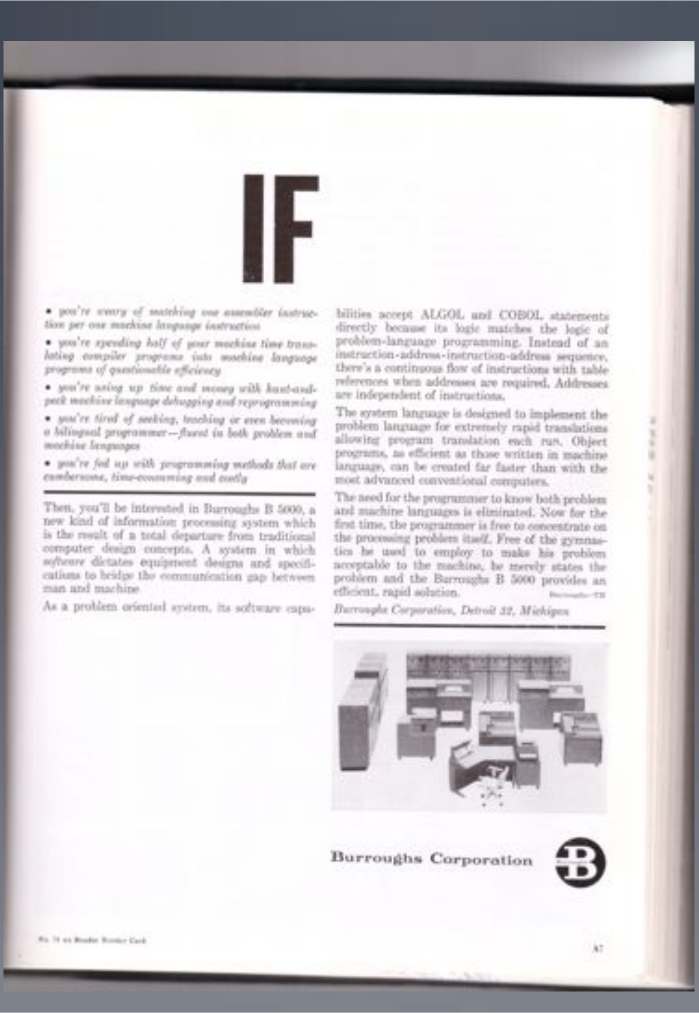
Versatile as it is, however, the MCP doesn't fully account for the fact that the B 5000 is a totally new concept in system operation. There's the B 5000's ability to incorporate a second central processor, for instance—without reprograming. And there's the fact that the B 5000's basic design concept provides effective and productive use of the higher level languages of ALGOL and COBOL. Plus enough other reasons to fill a whole booklet—which we'll be glad to send to you. It's called *The B 5000 Concept* and is available from our main office at Detroit 32, Michigan.

Burroughs—TM

Burroughs Corporation

IF

CACM July 1962



IF

- *you're weary of matching one assembler instruction per one machine language instruction*
- *you're spending half of your machine time translating compiler programs into machine language programs of questionable efficiency*
- *you're using up time and money with hunt-and-peck machine language debugging and reprogramming*
- *you're tired of seeking, teaching or even becoming a bilingual programmer—fluent in both problem and machine languages*
- *you're fed up with programming methods that are cumbersome, time-consuming and costly*

Then, you'll be interested in Burroughs B 5000, a new kind of information processing system which is the result of a total departure from traditional computer design concepts. A system in which *software* dictates equipment designs and specifications to bridge the communication gap between man and machine.

As a problem oriented system, its software capabilities accept ALGOL and COBOL statements directly because its logic matches the logic of problem-language programming. Instead of an instruction-address-instruction-address sequence, there's a continuous flow of instructions with table references when addresses are required. Addresses are independent of instructions.

The system language is designed to implement the problem language for extremely rapid translations allowing program translation each run. Object programs, as efficient as those written in machine language, can be created far faster than with the most advanced conventional computers.

The need for the programmer to know both problem and machine languages is eliminated. Now for the first time, the programmer is free to concentrate on the processing problem itself. Free of the gymnastics he used to employ to make his problem acceptable to the machine, he merely states the problem and the Burroughs B 5000 provides an efficient, rapid solution. Burroughs—TM

Burroughs Corporation, Detroit 32, Michigan

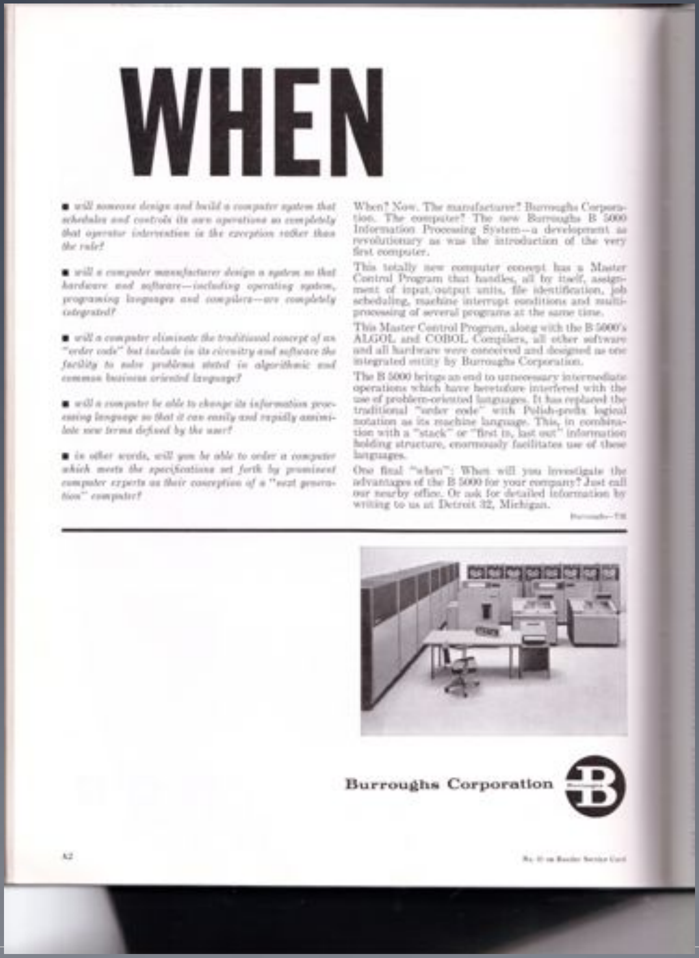
Burroughs Corporation

No. 74 on Reader Service Card

A7

When

CACM Aug 1962



WHEN

■ *will someone design and build a computer system that schedules and controls its own operations so completely that operator intervention is the exception rather than the rule?*

■ *will a computer manufacturer design a system so that hardware and software—including operating system, programing languages and compilers—are completely integrated?*

■ *will a computer eliminate the traditional concept of an "order code" but include in its circuitry and software the facility to solve problems stated in algorithmic and common business oriented language?*

■ *will a computer be able to change its information processing language so that it can easily and rapidly assimilate new terms defined by the user?*

■ *in other words, will you be able to order a computer which meets the specifications set forth by prominent computer experts as their conception of a "next generation" computer?*

When? Now. The manufacturer? Burroughs Corporation. The computer? The new Burroughs B 5000 Information Processing System—a development as revolutionary as was the introduction of the very first computer.

This totally new computer concept has a Master Control Program that handles, all by itself, assignment of input/output units, file identification, job scheduling, machine interrupt conditions and multiprocessing of several programs at the same time.

This Master Control Program, along with the B 5000's ALGOL and COBOL Compilers, all other software and all hardware were conceived and designed as one integrated entity by Burroughs Corporation.

The B 5000 brings an end to unnecessary intermediate operations which have heretofore interfered with the use of problem-oriented languages. It has replaced the traditional "order code" with Polish-prefix logical notation as its machine language. This, in combination with a "stack" or "first in, last out" information holding structure, enormously facilitates use of these languages.

One final "when": When will you investigate the advantages of the B 5000 for your company? Just call our nearby office. Or ask for detailed information by writing to us at Detroit 32, Michigan.

Burroughs—TM

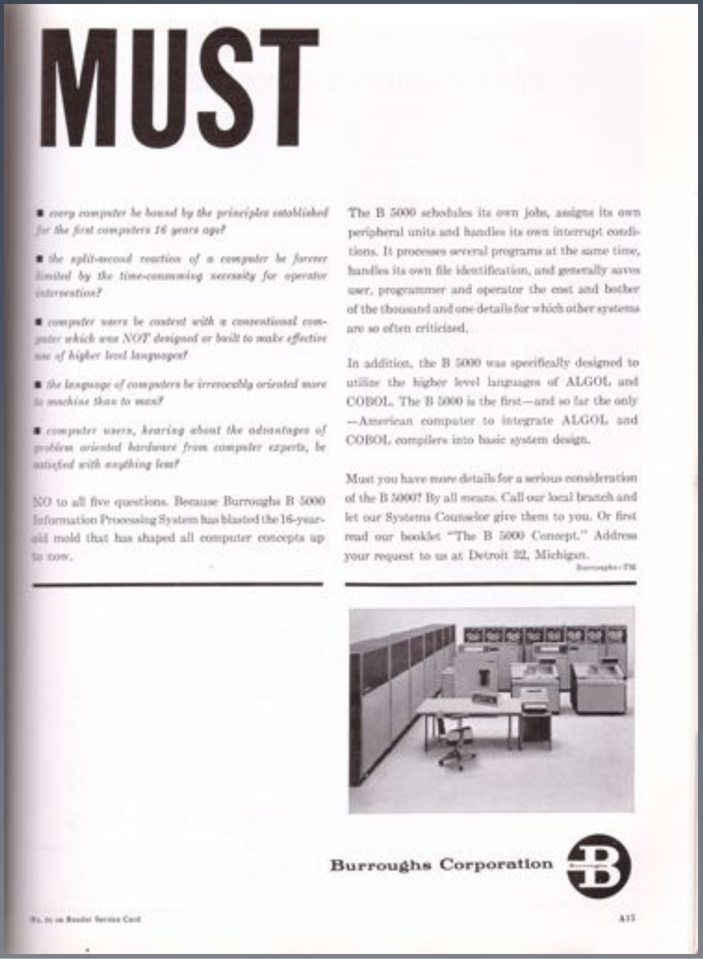
Burroughs Corporation

A2

No. 11 on Reader Service Card

MUST

CACM Oct 62



MUST

- *every computer be bound by the principles established for the first computers 16 years ago?*
- *the split-second reaction of a computer be forever limited by the time-consuming necessity for operator intervention?*
- *computer users be content with a conventional computer which was NOT designed or built to make effective use of higher level languages?*
- *the language of computers be irrevocably oriented more to machine than to man?*
- *computer users, hearing about the advantages of problem oriented hardware from computer experts, be satisfied with anything less?*

NO to all five questions. Because Burroughs B 5000 Information Processing System has blasted the 16-year-old mold that has shaped all computer concepts up to now.

The B 5000 schedules its own jobs, assigns its own peripheral units and handles its own interrupt conditions. It processes several programs at the same time, handles its own file identification, and generally saves user, programmer and operator the cost and bother of the thousand and one details for which other systems are so often criticized.

In addition, the B 5000 was specifically designed to utilize the higher level languages of ALGOL and COBOL. The B 5000 is the first—and so far the only—American computer to integrate ALGOL and COBOL compilers into basic system design.

Must you have more details for a serious consideration of the B 5000? By all means. Call our local branch and let our Systems Counselor give them to you. Or first read our booklet "The B 5000 Concept." Address your request to us at Detroit 32, Michigan.

Burroughs—TM

Burroughs Corporation

No. [illegible] on Reader Service Card

A15

University of Illinois Subroutine library (circa 1963)

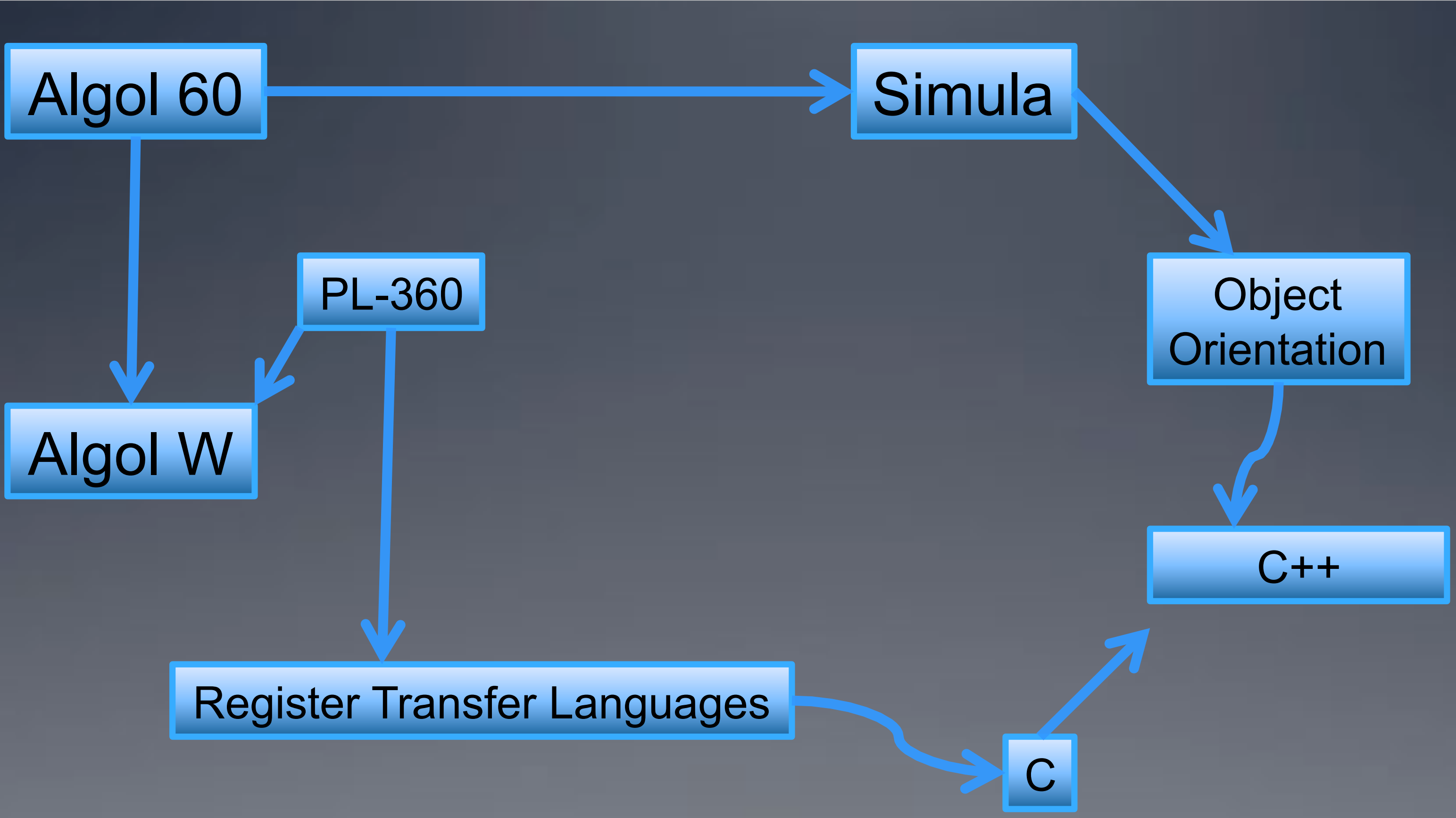
Isn't this a simply a beautiful way to store cards

The name is Gold Star . . . the world's most beautiful and most complete card storage and handling system. Notice the gleaming chrome legs, the attractive and practical stainless steel handles and accent panels, the bonded plastic work surface, and the three-wide drawers that open all the way to expose 12,000 cards at a time. This fifteen drawer file is one of a family of files ranging from one drawer to 30 drawers and including trucks and transfer filing units. Write for complete information.

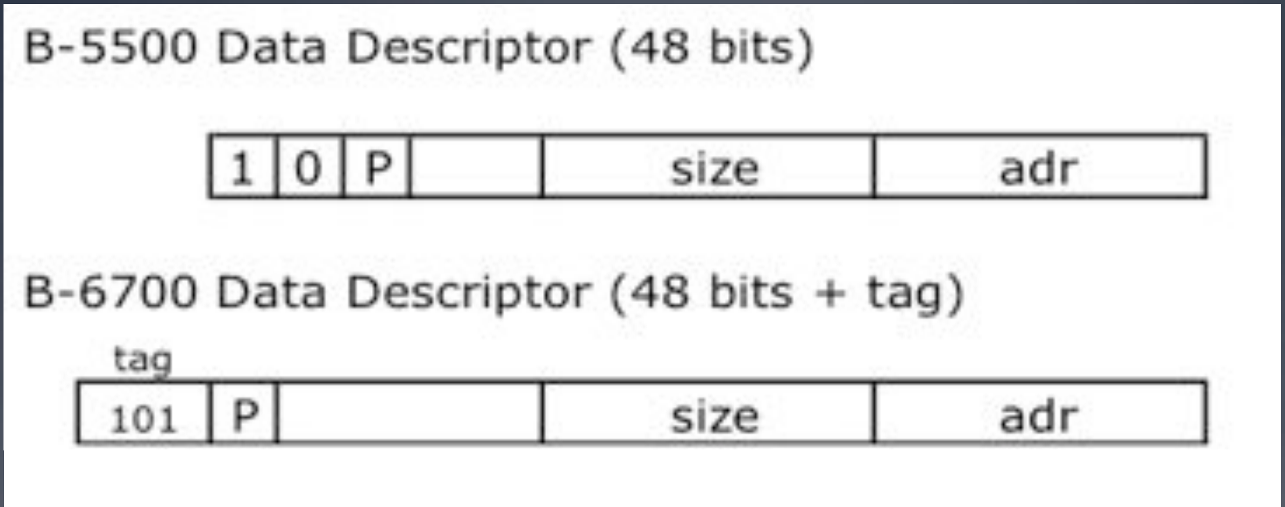
Wright LINE

DATA PROCESSING ACCESS

Algol 60 begat



Dynamically sized arrays



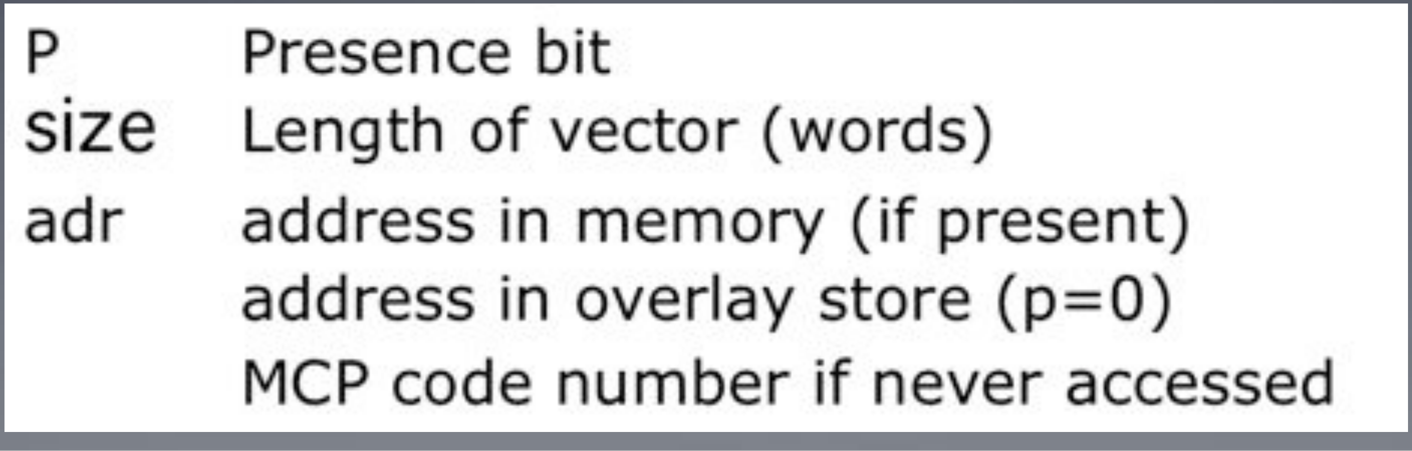
B-5500 Data Descriptor (48 bits)

1	0	P		size	adr

B-6700 Data Descriptor (48 bits + tag)

tag

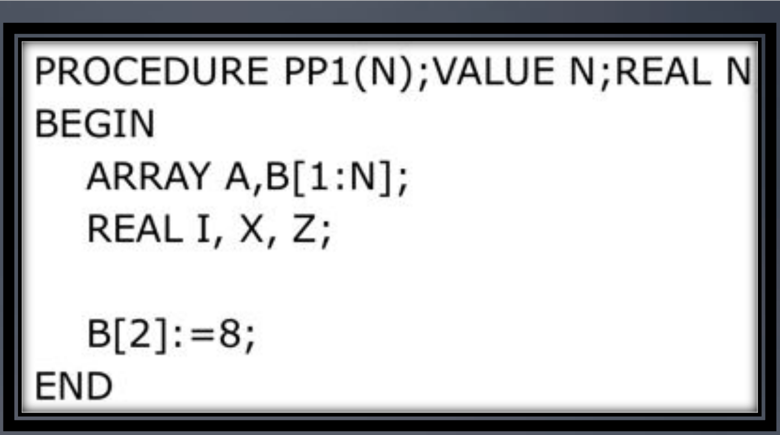
101	P		size	adr



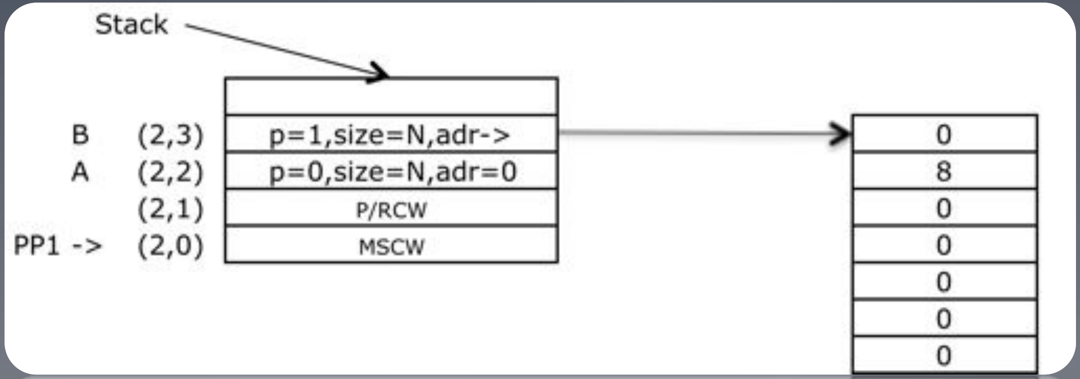
P	Presence bit
size	Length of vector (words)
adr	address in memory (if present) address in overlay store (p=0) MCP code number if never accessed

Lazy Array Allocation

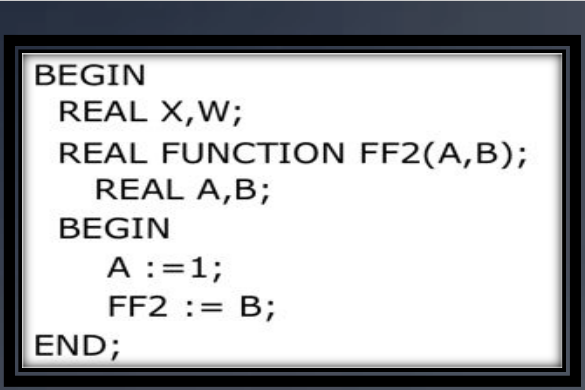
- Algol dynamic array



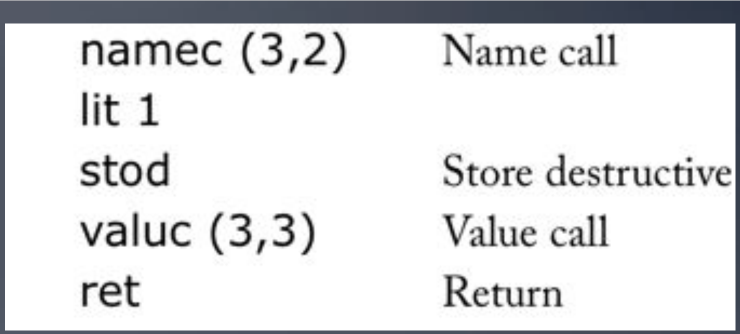
```
PROCEDURE PP1(N);VALUE N;REAL N
BEGIN
   ARRAY A,B[1:N];
   REAL I, X, Z;

   B[2]:=8;
END
```



“Call by Name”

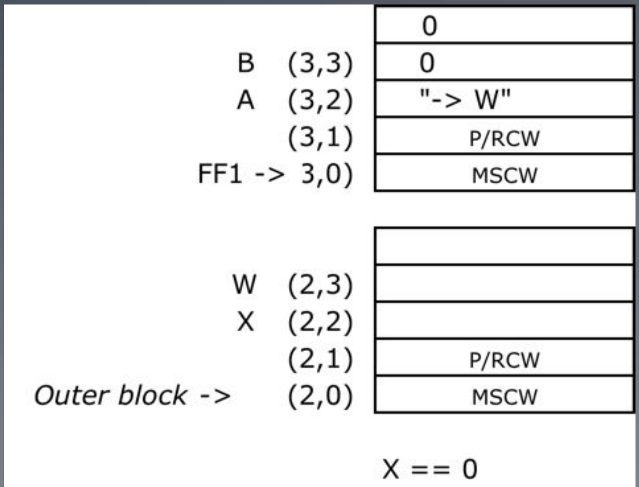


```
BEGIN
  REAL X,W;
  REAL FUNCTION FF2(A,B);
     REAL A,B;
  BEGIN
      A :=1;
      FF2 := B;
END;
```



namec (3,2)	Name call
lit 1	
stod	Store destructive
valuc (3,3)	Value call
ret	Return

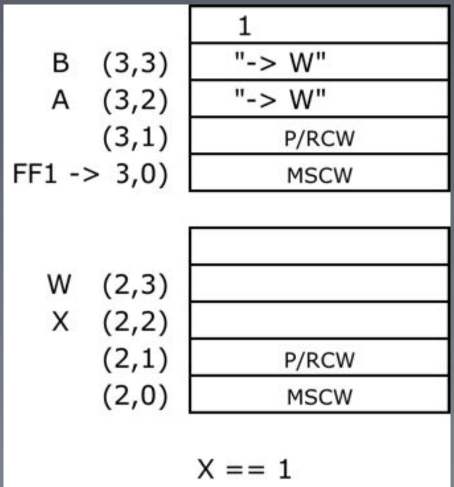
X := FF2(W,0);



		0
B	(3,3)	0
A	(3,2)	"-> W"
	(3,1)	P/RCW
FF1 ->	3,0)	MSCW
W	(2,3)	
X	(2,2)	
	(2,1)	P/RCW
Outer block ->	(2,0)	MSCW

X == 0

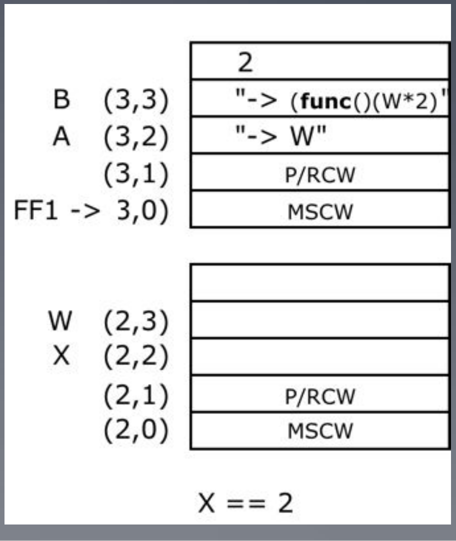
X := FF2(W,W);



		1
B	(3,3)	"-> W"
A	(3,2)	"-> W"
	(3,1)	P/RCW
FF1 ->	3,0)	MSCW
W	(2,3)	
X	(2,2)	
	(2,1)	P/RCW
	(2,0)	MSCW

X == 1

X := FF2(W,W*2);

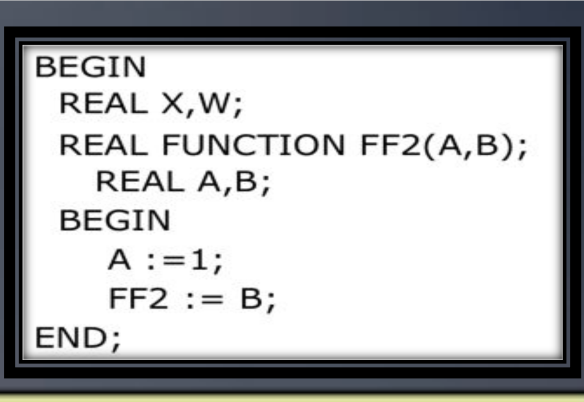


		2
B	(3,3)	"-> (**func**()(W*2)'
A	(3,2)	"-> W"
	(3,1)	P/RCW
FF1 ->	3,0)	MSCW
W	(2,3)	
X	(2,2)	
	(2,1)	P/RCW
	(2,0)	MSCW

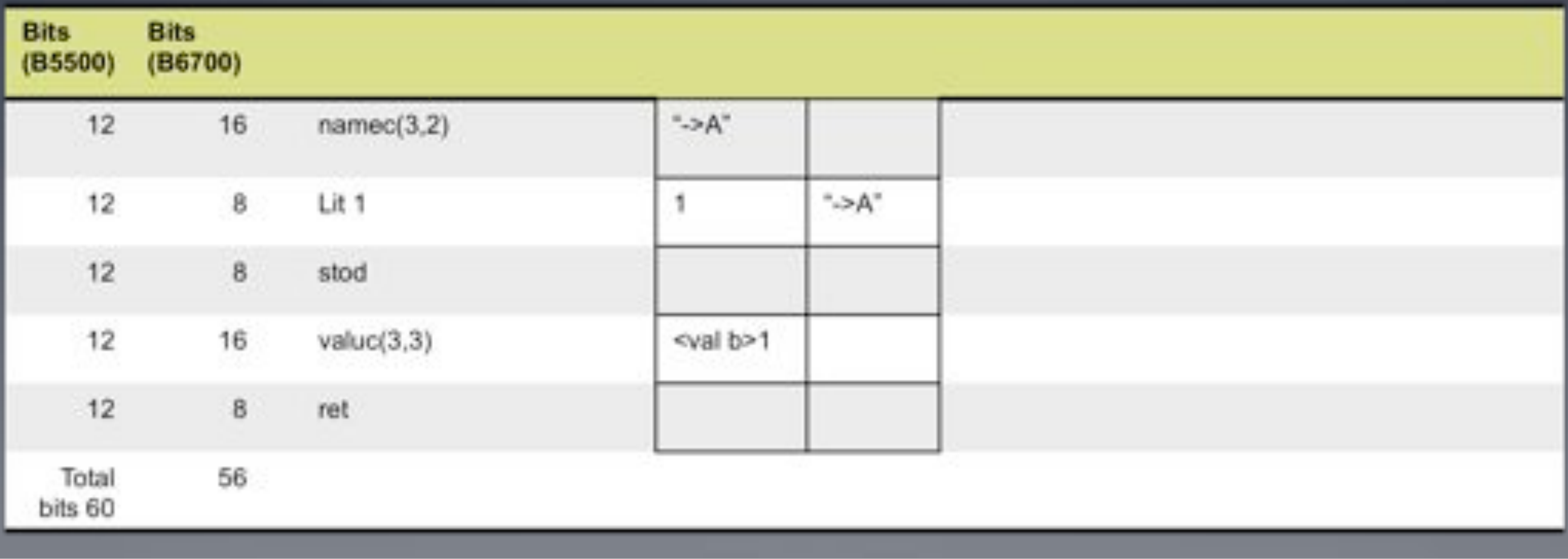
X == 2

Hardware instructions

- High density code
- Address size independent



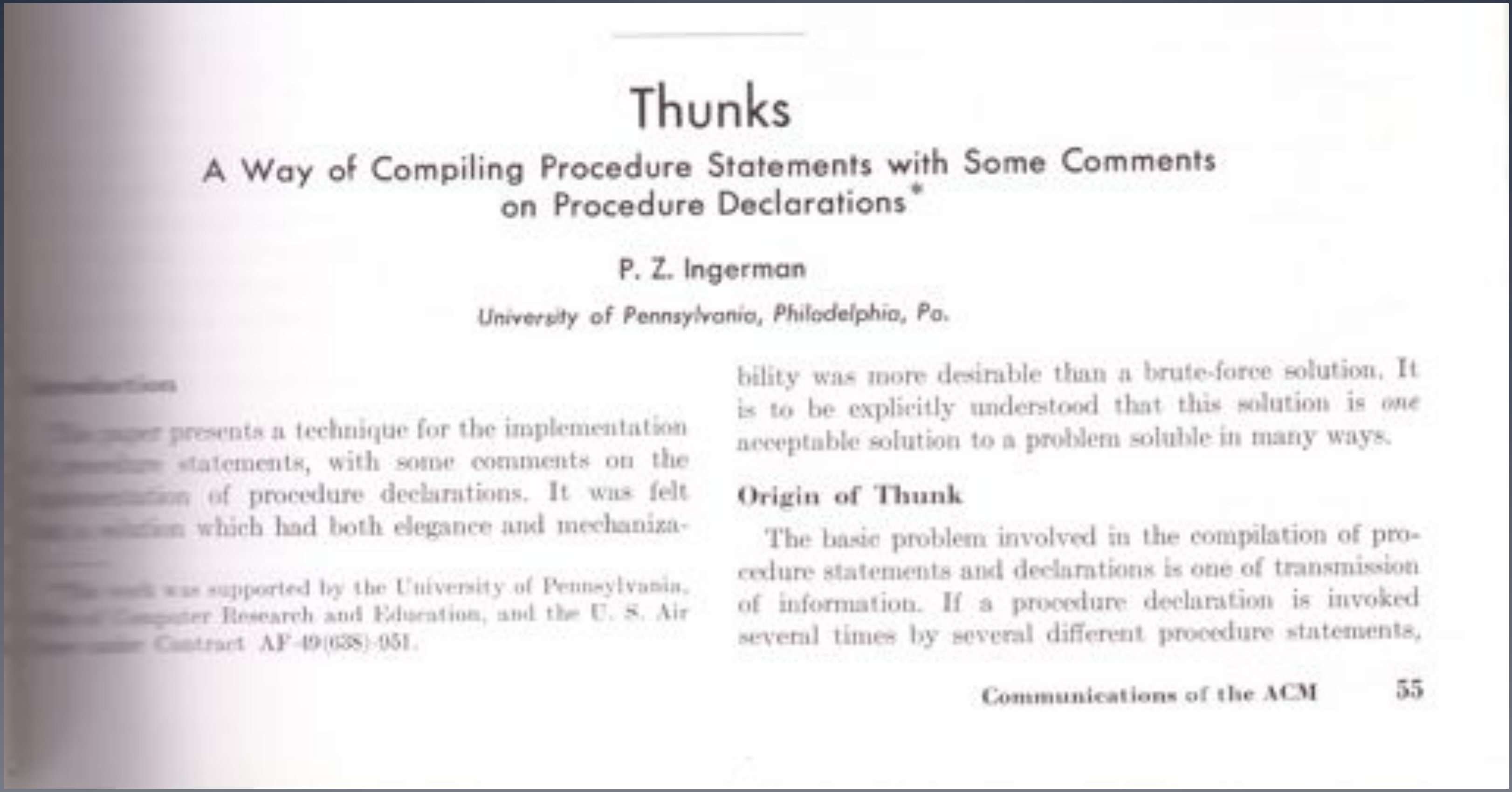
```
BEGIN
  REAL X,W;
  REAL FUNCTION FF2(A,B);
     REAL A,B;
  BEGIN
      A :=1;
      FF2 := B;
END;
```



Bits (B5500)	Bits (B6700)			
12	16	namec(3,2)	"->A"	
12	8	Lit 1	1	"->A"
12	8	stod		
12	16	valuc(3,3)	<val b>1	
12	8	ret		
Total bits 60	56			

Thunks

CACM Jan 61



Thunks

A Way of Compiling Procedure Statements with Some Comments on Procedure Declarations*

P. Z. Ingerman

University of Pennsylvania, Philadelphia, Pa.

Introduction

This paper presents a technique for the implementation of procedure statements, with some comments on the implementation of procedure declarations. It was felt that a solution which had both elegance and mechanizability was more desirable than a brute-force solution. It is to be explicitly understood that this solution is *one* acceptable solution to a problem soluble in many ways.

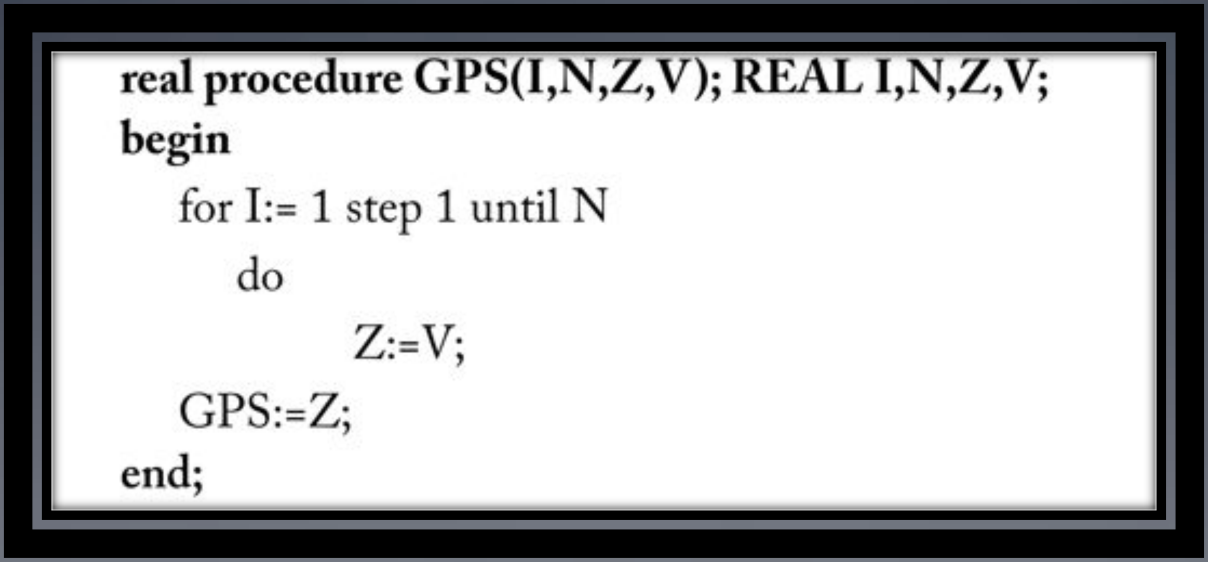
Origin of Thunk

The basic problem involved in the compilation of procedure statements and declarations is one of transmission of information. If a procedure declaration is invoked several times by several different procedure statements,

*This work was supported by the University of Pennsylvania, Office of Computer Research and Education, and the U. S. Air Force under Contract AF-49(638)-951.

GPS
General Problem Solver

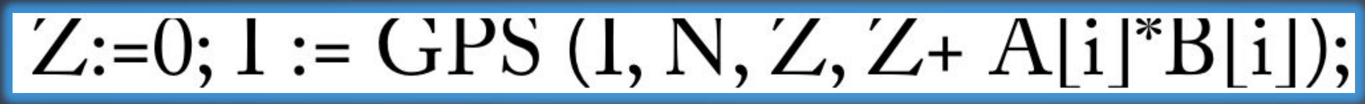
In *Algol 60 Confidential*, Knuth and Merner, CACM June 61



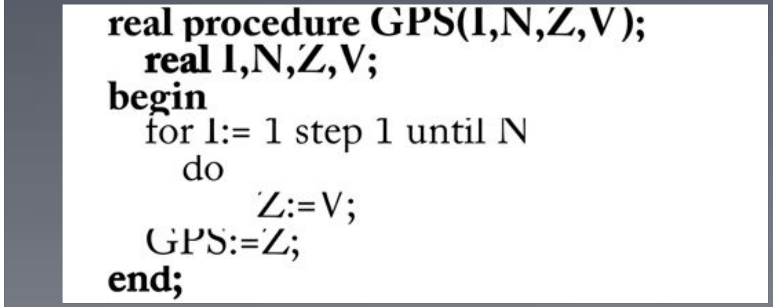
```
real procedure GPS(I,N,Z,V); REAL I,N,Z,V;
begin
    for I:= 1 step 1 until N
        do
                Z:=V;
    GPS:=Z;
end;
```

GPS does inner product

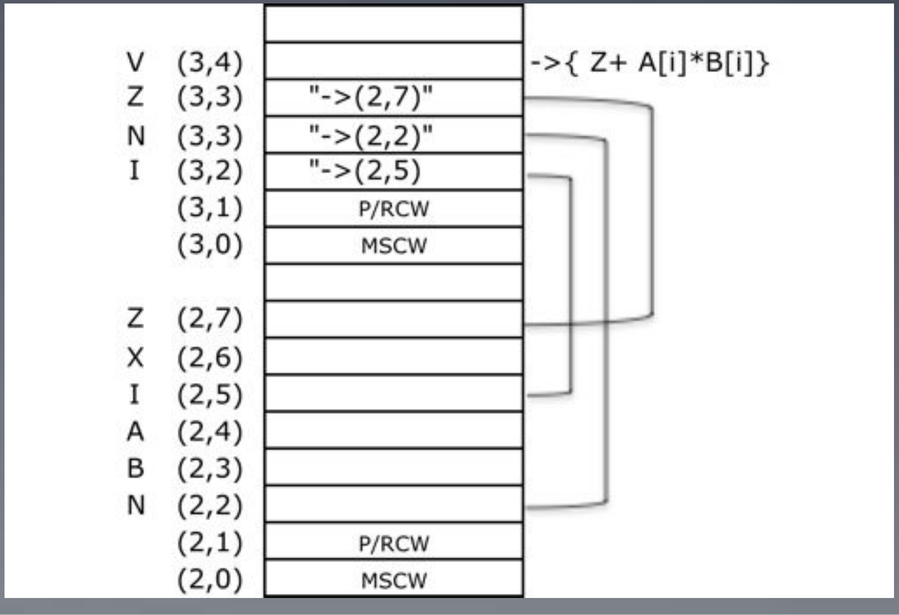
Functional style: **sum(a[i]*b[i] | i =1..n)**



```
Z:=0; I := GPS (I, N, Z, Z+ A[i]*B[i]);
```

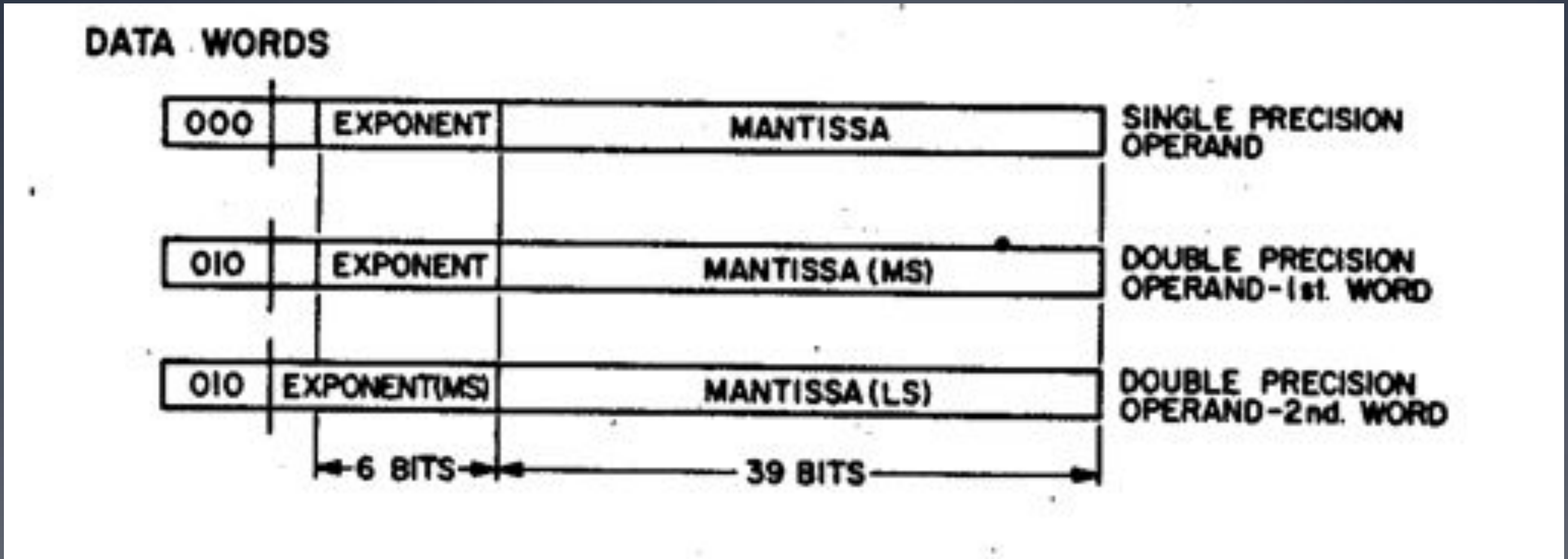


```
real procedure GPS(I,N,Z,V);
  real I,N,Z,V;
begin
  for I:= 1 step 1 until N
    do
        Z:=V;
  GPS:=Z;
end;
```



V	(3,4)		->{ Z+ A[i]*B[i]}
Z	(3,3)	"->(2,7)"	
N	(3,3)	"->(2,2)"	
I	(3,2)	"->(2,5)	
	(3,1)	P/RCW	
	(3,0)	MSCW	
Z	(2,7)		
X	(2,6)		
I	(2,5)		
A	(2,4)		
B	(2,3)		
N	(2,2)		
	(2,1)	P/RCW	
	(2,0)	MSCW	

B 6500 Arithmetic values



B 6500 Descriptors

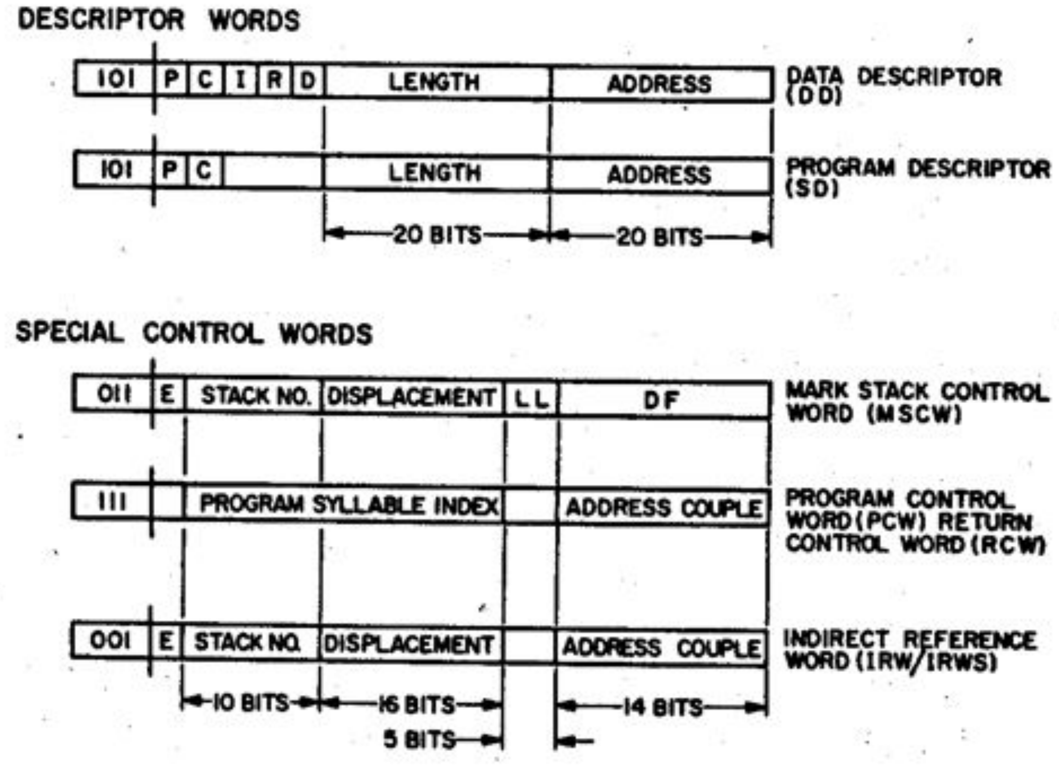
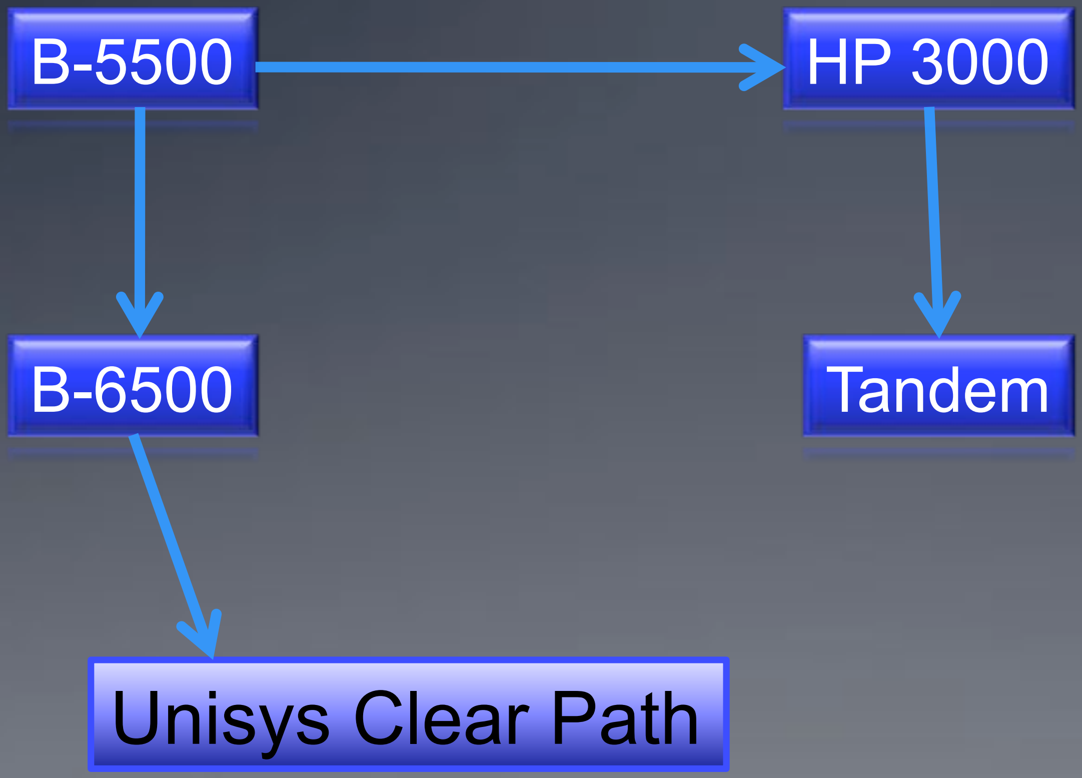
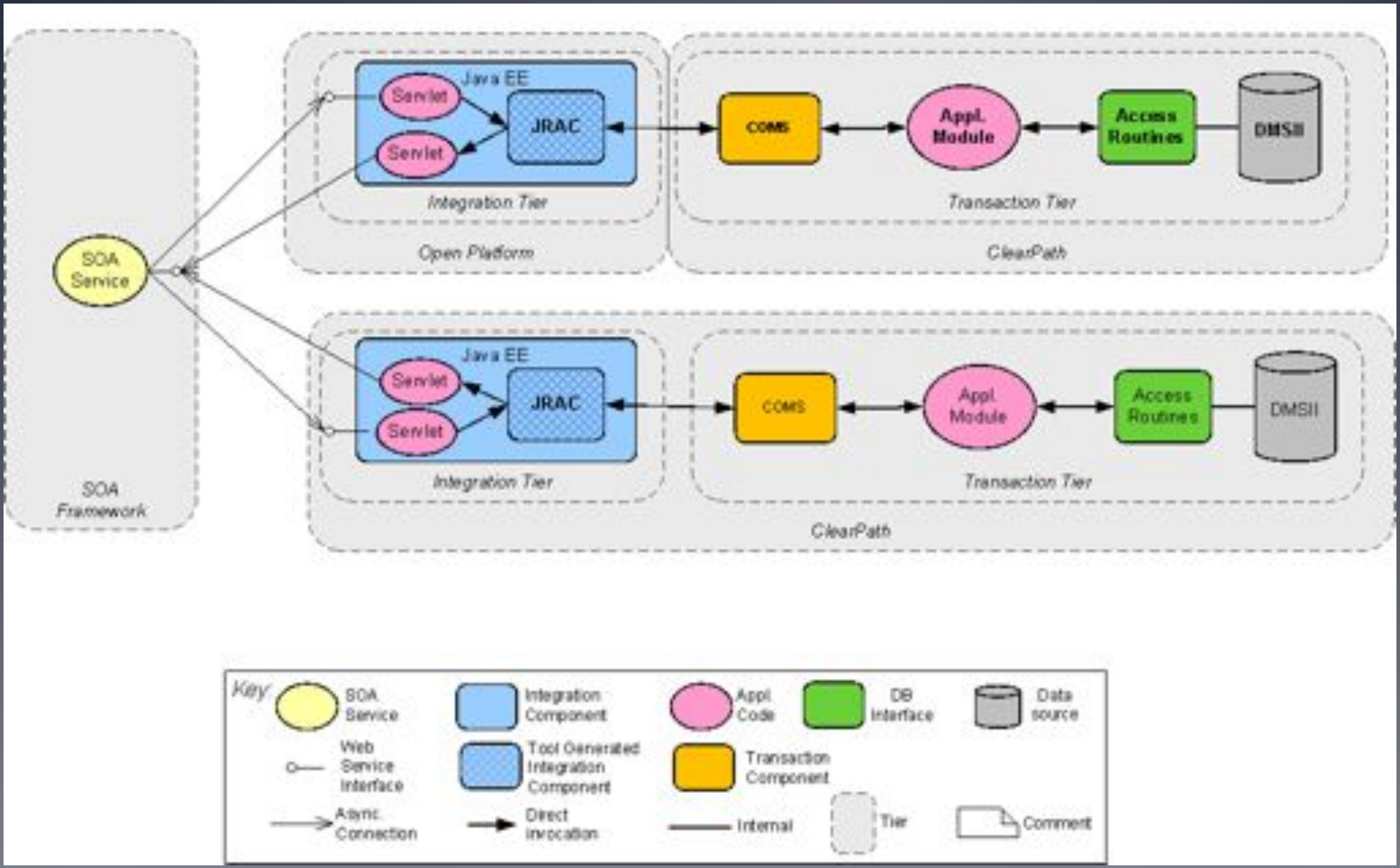


Figure 1 – B6500/B7500 word formats

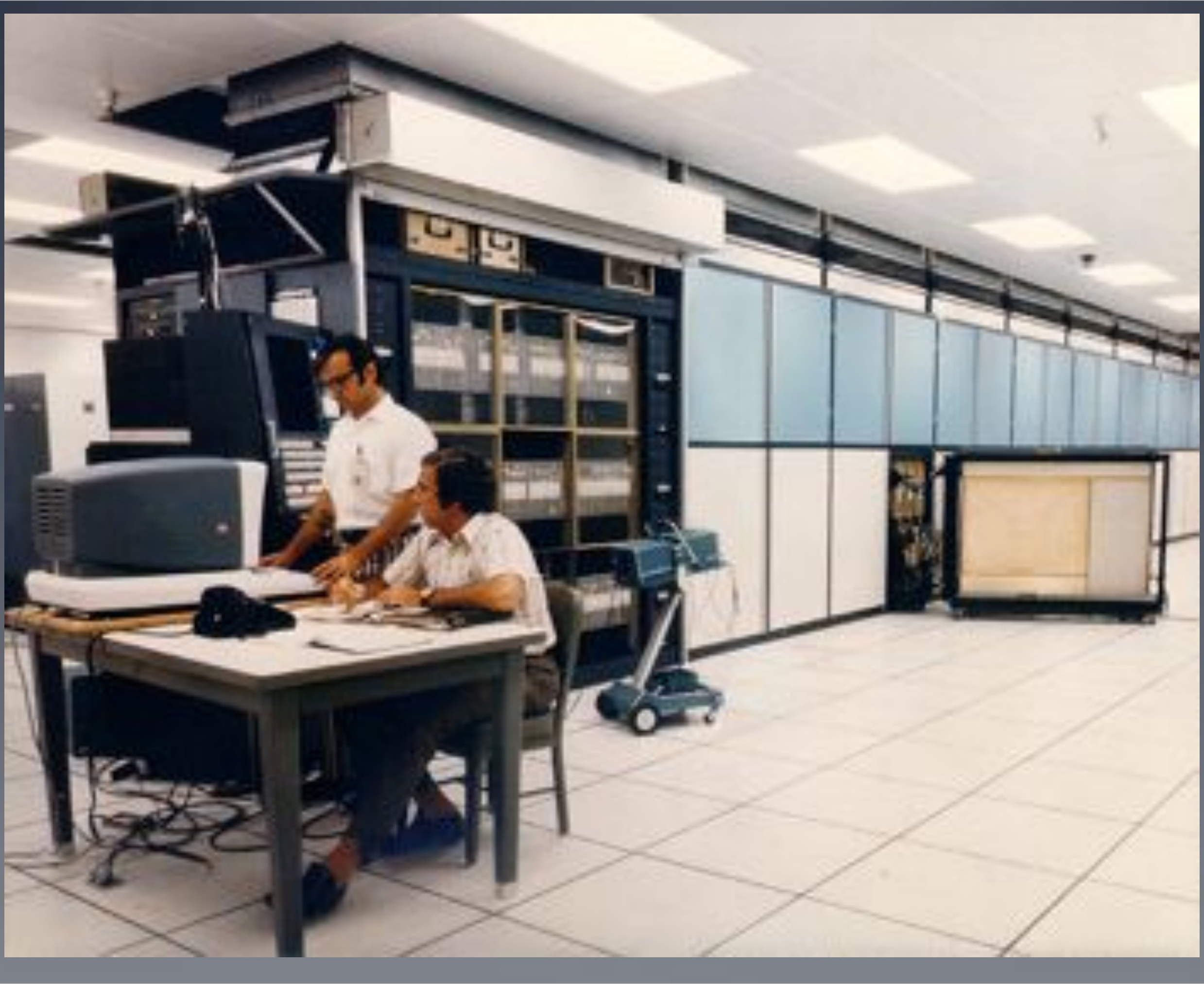
B5000 begat



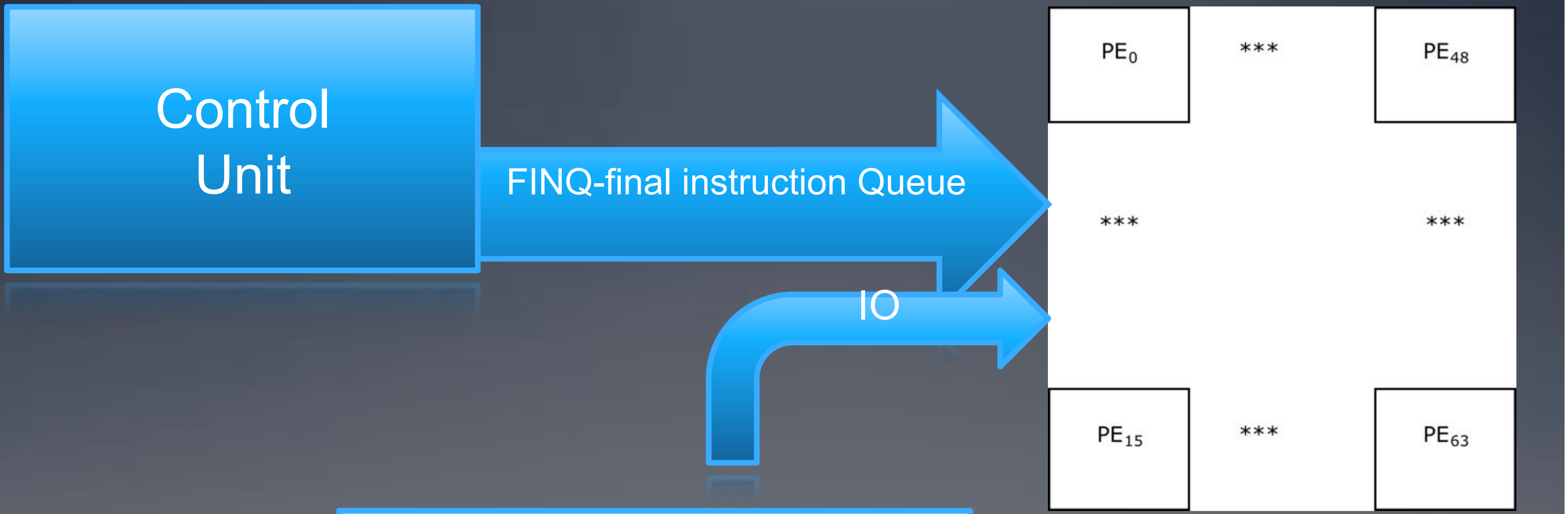
ClearPath Reference Architecture
Web Services for MCP (COMS - Java)



Illiac IV

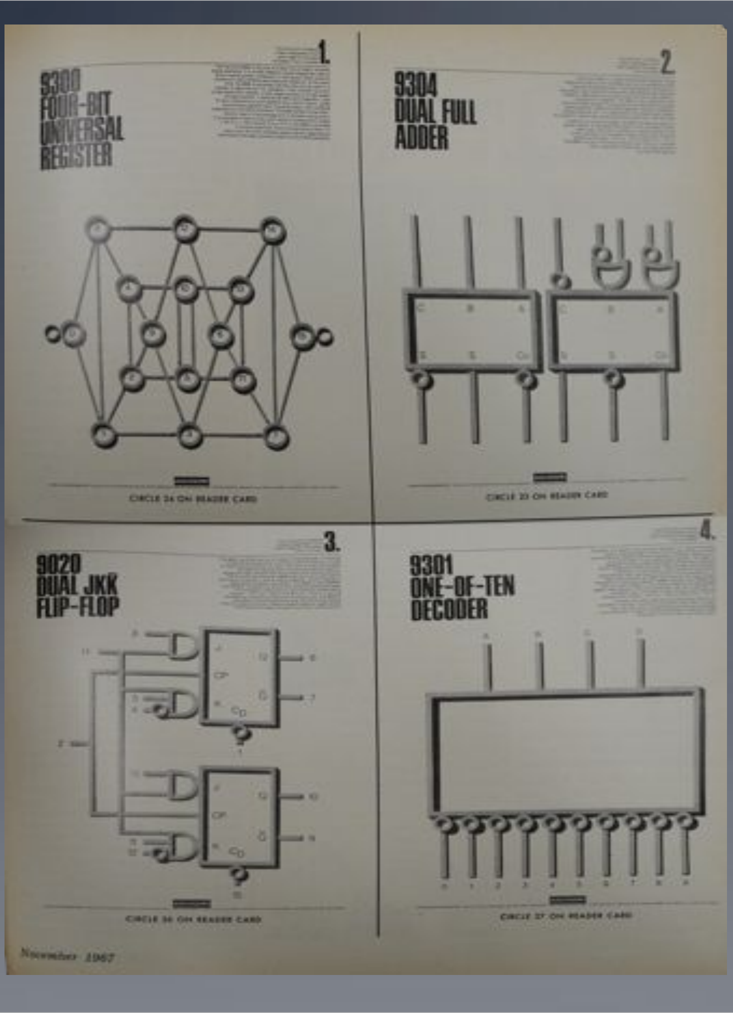


Illiac IV



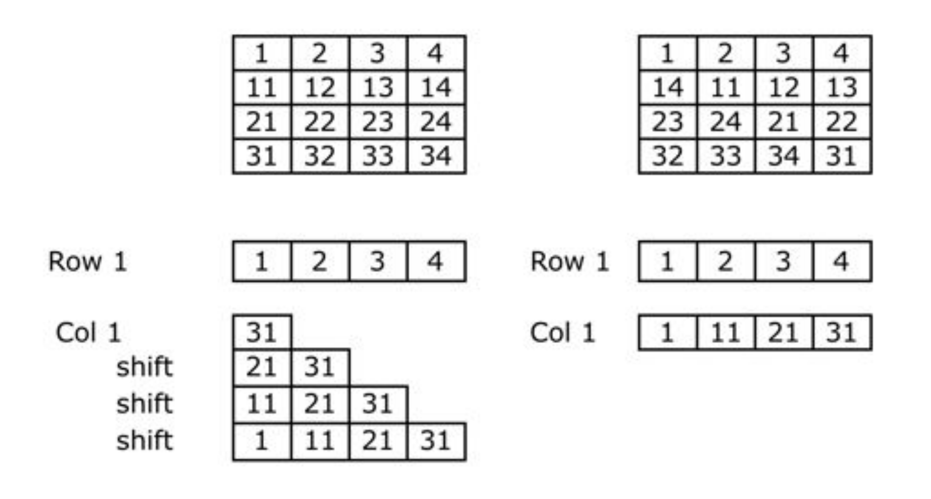
Disk delivered 1024 bits per micro-second (128 mega-bytes per second)

Hardware ad


1.
9300 FOUR-BIT UNIVERSAL REGISTER
CIRCLE 24 ON READER CARD
2.
9304 DUAL FULL ADDER
CIRCLE 25 ON READER CARD
3.
9020 DUAL JKK FLIP-FLOP
CIRCLE 26 ON READER CARD
4.
9301 ONE-OF-TEN DECODER
CIRCLE 27 ON READER CARD
November 1967

Illiac IV

Improve performance by rotate/skew



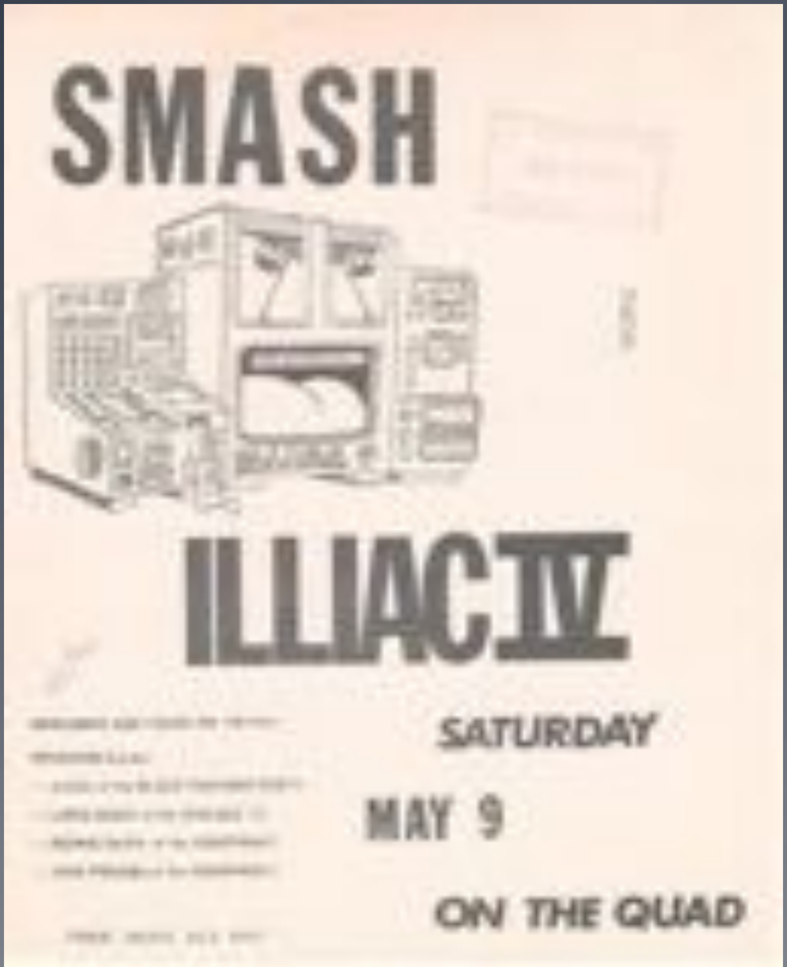
1	2	3	4
11	12	13	14
21	22	23	24
31	32	33	34

1	2	3	4
14	11	12	13
23	24	21	22
32	33	34	31

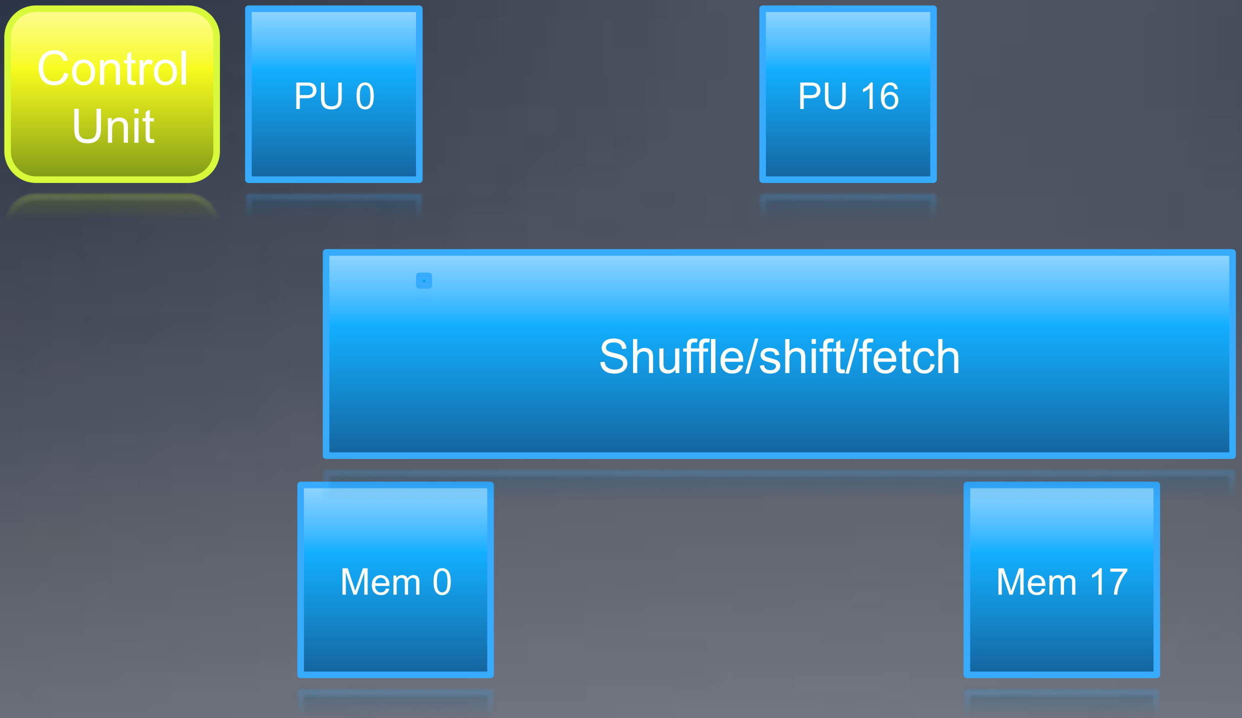
Row 1	1	2	3	4
Col 1	31			
shift	21	31		
shift	11	21	31	
shift	1	11	21	31

Row 1	1	2	3	4
Col 1	1	11	21	31

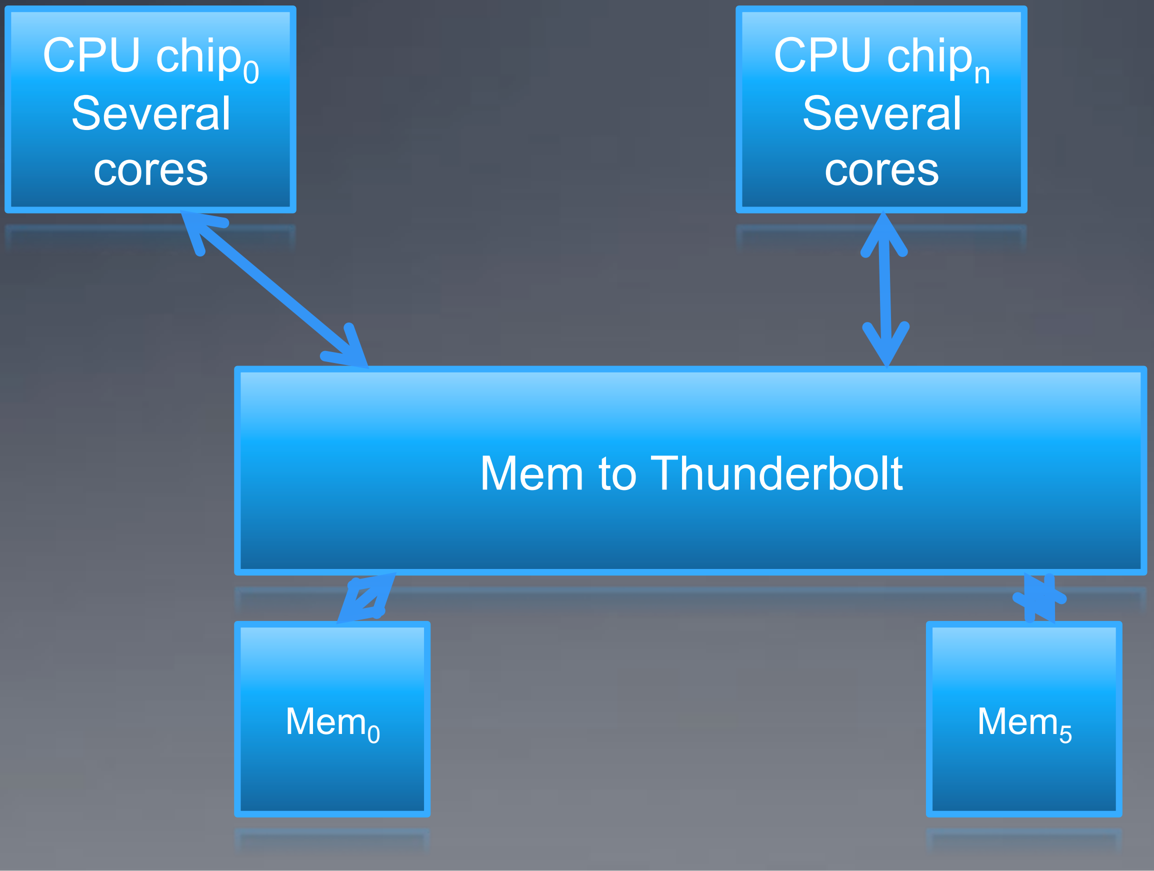
Illiac IV



BSP design

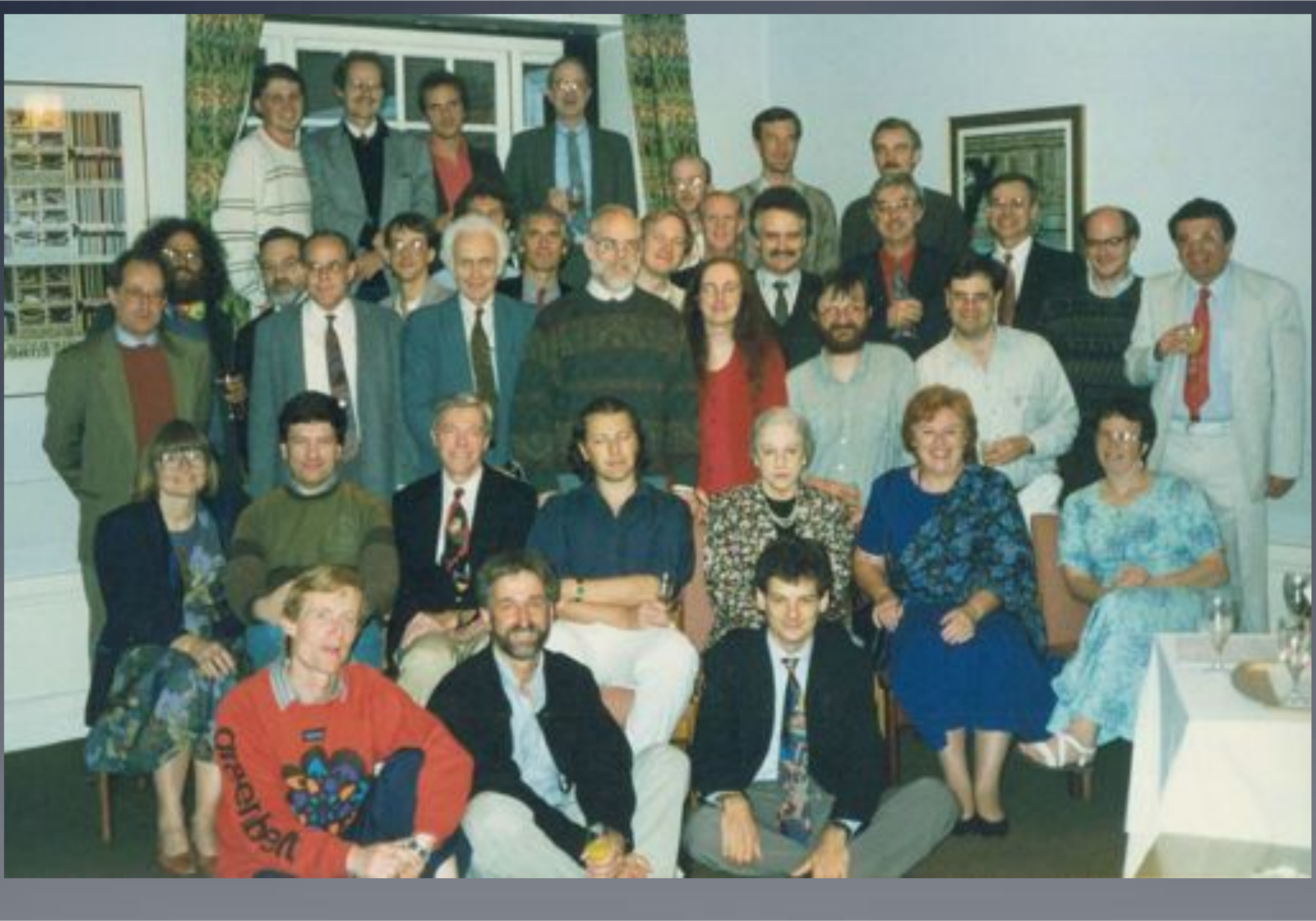


EE Senior project



Déjà vu all over again

Members and guests of IFIP Working Group 2.8, Oxford, 1992



Simon Peyton Jones and Simon Marlow receive **the SIGPLAN Software** Award as the authors of the Glasgow Haskell Compiler (GHC)

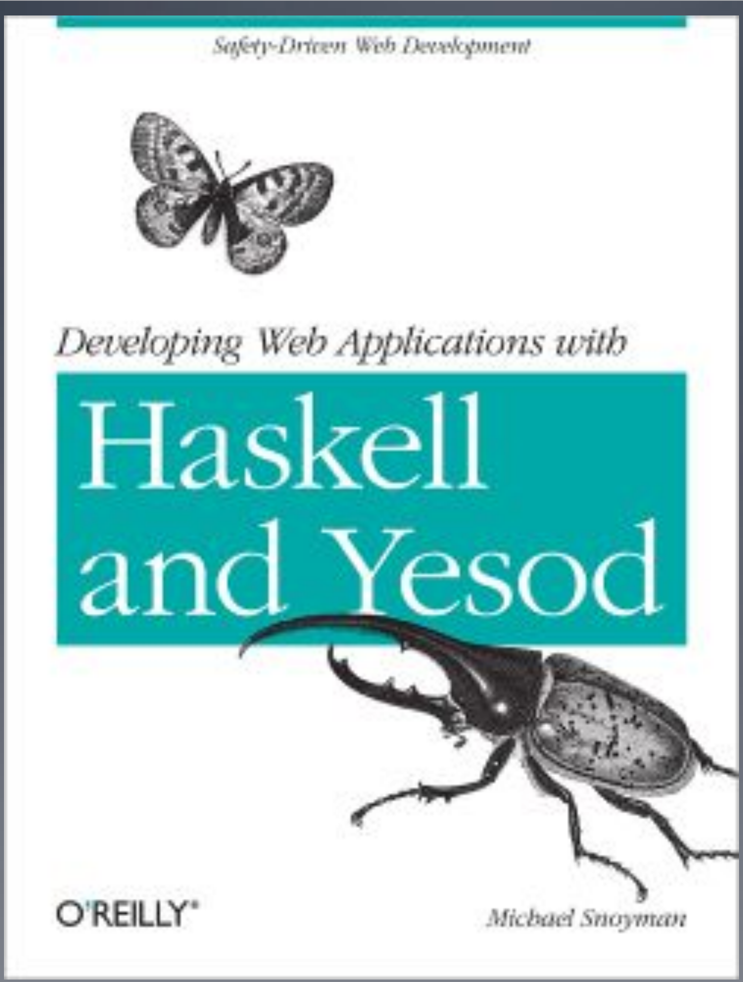
A measure of GHC's influence is the way that many of the ideas of purely functional, "typeful programming" have been carried into newer languages and language features. including C#, F#, Java Generics, LINQ, Perl 6, Python, and Visual Basic 9.0. Peyton Jones and Marlow have been visionary in the way that they have transitioned research into practice. They have been role models and leaders in creating the large and diverse Haskell community, and have made GHC an industrial-strength platform for commercial development as well as for research.

Why Haskell?

- Prove program will not crash (given constraints)
- Fast – industrial grade compiler
- Always deterministic
- Immutable variables
- “many cores” is the “killer app” for functional programming
- Haskell is the “purest”

Yesod

http://www.yesodweb.com/



Safety-Driven Web Development

Developing Web Applications with

Haskell and Yesod

O'REILLY

Michael Snoyman

~~Cleary~~ Curry

Haskell + :=:

http://www-ps.informatik.uni-kiel.de/currywiki/

Set Functions for Functional Logic Programming

The independence of the order of evaluation is an essential property of declarative languages. Consequently, our approach positively solves a long-standing problem. For this reason, we believe that our approach will eventually replace all the previously proposed approaches found in current implementations of functional logic languages.

Sergio Antoy
Computer Science Department
Portland State University
Portland, OR 97207, USA
antoy@cs.pdx.edu

Michael Hanus
Institute of Computer Science
Christian-Albrechts-University of Kiel
D-24098 Kiel, Germany
mh@informatik.uni-kiel.de

PPDP’09, September 7–9, 2009, Coimbra, Portugal, Copyright c 2009 ACM

DDC

The Disciplined Disciple Compiler (DDC)

Haskell + :=

http://disciple.ouroborus.net/

Disciple is a dialect of Haskell that uses strict evaluation as the default and supports destructive update of arbitrary data structures. Disciple includes region, effect and closure typing, and this extra information provides a handle on the operational behaviour of code that isn't available in other languages. Programs can be written in either a pure/functional or effectful/imperative style, and one of our goals is to provide both styles coherently in the same language. The two styles can be mixed safely, for example: when using laziness the type system guarantees that computations with visible side effects are not suspended. Many Haskell programs are also Disciple programs, or will run with minor changes. Our target applications are the ones that you always find yourself writing C programs for, because existing functional languages are too slow, use too much memory, or don't let you update the data that you need to.

Endnotes: Burroughs history

Narrative Description of the B5000 MCP

This was my first introduction to this innovative machine. Prof McCormick, then in charge of the Illiac 3 project, brought this manual to the MATH 395 class I was taking. *description*

http://bitsavers.org/pdf/burroughs/B5000_5500_5700/1023579_Narrative_Description_Of_B5500_MCP_Oct66.pdf

Books

Computer SystemOrganization, The B5700/B6700 Series, ELLIOTT I. ORGANICK

http://www.bitsavers.org/pdf/burroughs/B5000_5500_5700/Organick_B5700_B6700_1973.pdf
-this link may not work

ALGOL 60 implementation: the translation and use of ALGOL 60 programs on a computer, Brian Randell and L. J. Russel

Ben Dent, one of the designers of the B6500, said “Randell and Russell did what we did. It was all there.”
I haven’t read the book – but a member of the audience said he had enjoyed reading it.

Published papers

The Burroughs B6500/B7500 Stack Mechanism, by E. A. HAUCK and B. A. DENT, AFIPS SJCC 1968

http://bitsavers.trailing-edge.com/pdf/burroughs/B6500_6700/1035441_B6500_B7500_Stack_Mechanism_1968.pdf

Segment Sizes and Lifetimes in Algol 60 Programs, A.P. Batson and R.E. Brundage, University of Virginia

http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCYQFjAA&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.116.8206%26rep%3Drep1%26type%3Dpdf&ei=Bs5kUPGbJKXDyQGm24DICg&usg=AFQjCNGAc9s9EVT0RUONFZd1GJVCzSeVDw&sig2=iOu4P5pA0I2zxI3zO4Obrw

Endnotes: Burroughs history

Design of the B-5000 System, William Lonergan and Paul King, Product Planning, Burroughs, Datamation May 1961
I didn't know this article existed until I did this talk.

http://www.cs.berkeley.edu/~culler/courses/cs252-s05/papers/b5000.pdf

Oral History conference, Conducted by Bernard A. Galler and Robert F. Rosin, 6 September 1985,

http://conservancy.umn.edu/bitstream/107105/1/oh098b5c.pdf

Interview with Leroy Gluck, OH 248, 25 June 1987, Charles Babbage Institute, University of Minnesota

http://conservancy.umn.edu/bitstream/107342/1/oh248lrg.pdf

Interview with Bob Creech, OH 249, 25 June 1987

http://conservancy.umn.edu/bitstream/107759/1/oh249bac.pdf

Endnotes: Haskell information

- Historical context

Faith, Evolution, and Programming Languages

http://www.infoq.com/presentations/Faith-Evolution-Programming-Languages

Prof Philip Wadler discusses second-order quantification, from its inception in the symbolic logic of Frege through to the generic features introduced in Java 5, touching on aspects of faith and evolution.

- Why change to Haskell?

From Bartosz Milewski's blog. "Imperative programming is in my bloodstream. I've been a C++ programmer for most of my life. I wrote a book about C++ … Programmers are scared of concurrency, and rightly so. Managers are scared of concurrency even more. … We make mistakes in coding and we have systems for debugging and testing programs — there is no such system for concurrency bugs. I should know because I worked for a company that made state of the art data race detector. But this detector couldn't prove that a program was 100% data-race free. ... The company's web site had a list of major disasters caused by data races. Among them was the famous Therac-25 disaster that killed three patients and injured several others, and the Northeast Blackout of 2003 that affected 55 million people. The fears are well founded!"

http://fpcomplete.com/the-downfall-of-imperative-programming/

http://fpcomplete.com/ten-things-you-should-know-about-haskell-syntax/

Bibliography

- ***Miran Lipovača , Learn You a Haskell for Great Good! — an excellent Haskell tutorial***
  - http://learnyouahaskell.com/chapters
- ***Manuel Chakravarty, Data Parallelism in Haskell — a video presentation***
  - http://vimeo.com/28477220
- Data Parallel Haskell — a collection of papers.
  - http://research.microsoft.com/en-us/um/people/simonpj/papers/ndp/
- ***Repa — REgular PArallel arrays.***
  - http://repa.ouroborus.net/