Principles of Programming Languages
Lecture 1

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgment: parts of these slides are based on material by Diane Horton & Eric Joanis @ UoT

References: Scheme by Dybvig
Concepts of PL by Sebesta
Prog. in Prolog by Clocksin and Mellish
PL Concepts and Constructs by Sethi
ML for the Working Prog. By Paulson
PL Pragmatics by Scott

Today

- Administrivia
- History of Programming Languages!
- Programming Languages Paradigms

Administrivia

- **Class web site:**
  - http://portal.utoronto.ca/
  - Course information sheet, grading, important dates, remark requests, discussion board, assignment submission, announcements,…

- **Three programming assignments (50%)**

- **Midterm on March 2nd (15%) and final worth 35%**

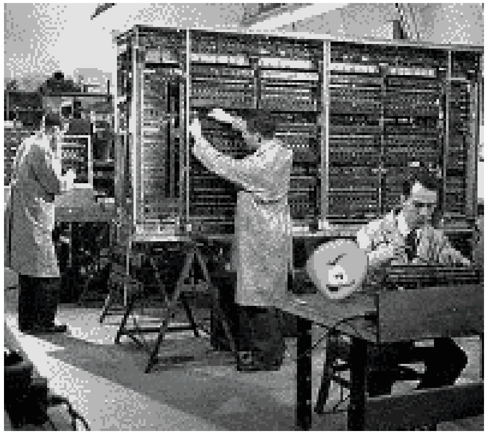
Course Contents

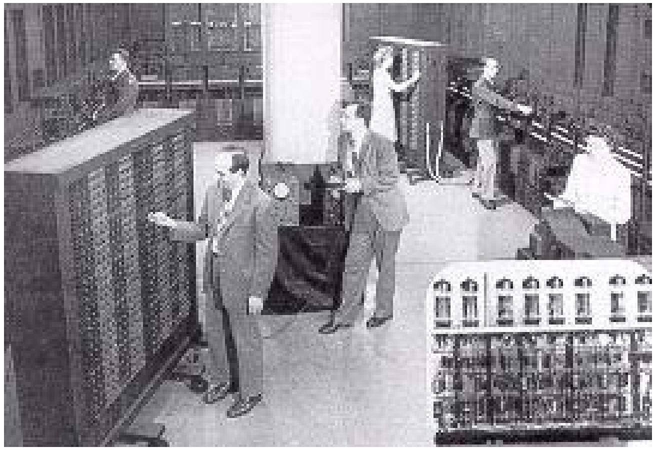
- **Programming Language Varieties ☺**
  - Logic Programming (Prolog)
  - Imperative Programming (Javascript)
  - Functional Programming (Scheme and ML)

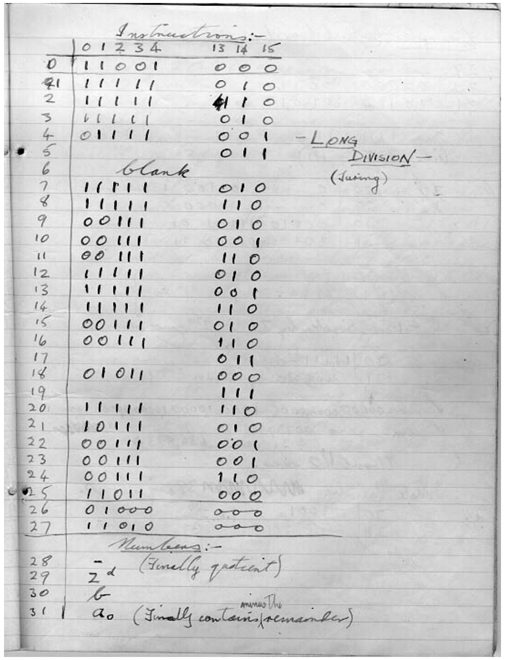
- **Programming Language Design**
  - Formal specification
  - Issues in designing a language

Introduction && PL History

PL History: programming then…







Instructions:—

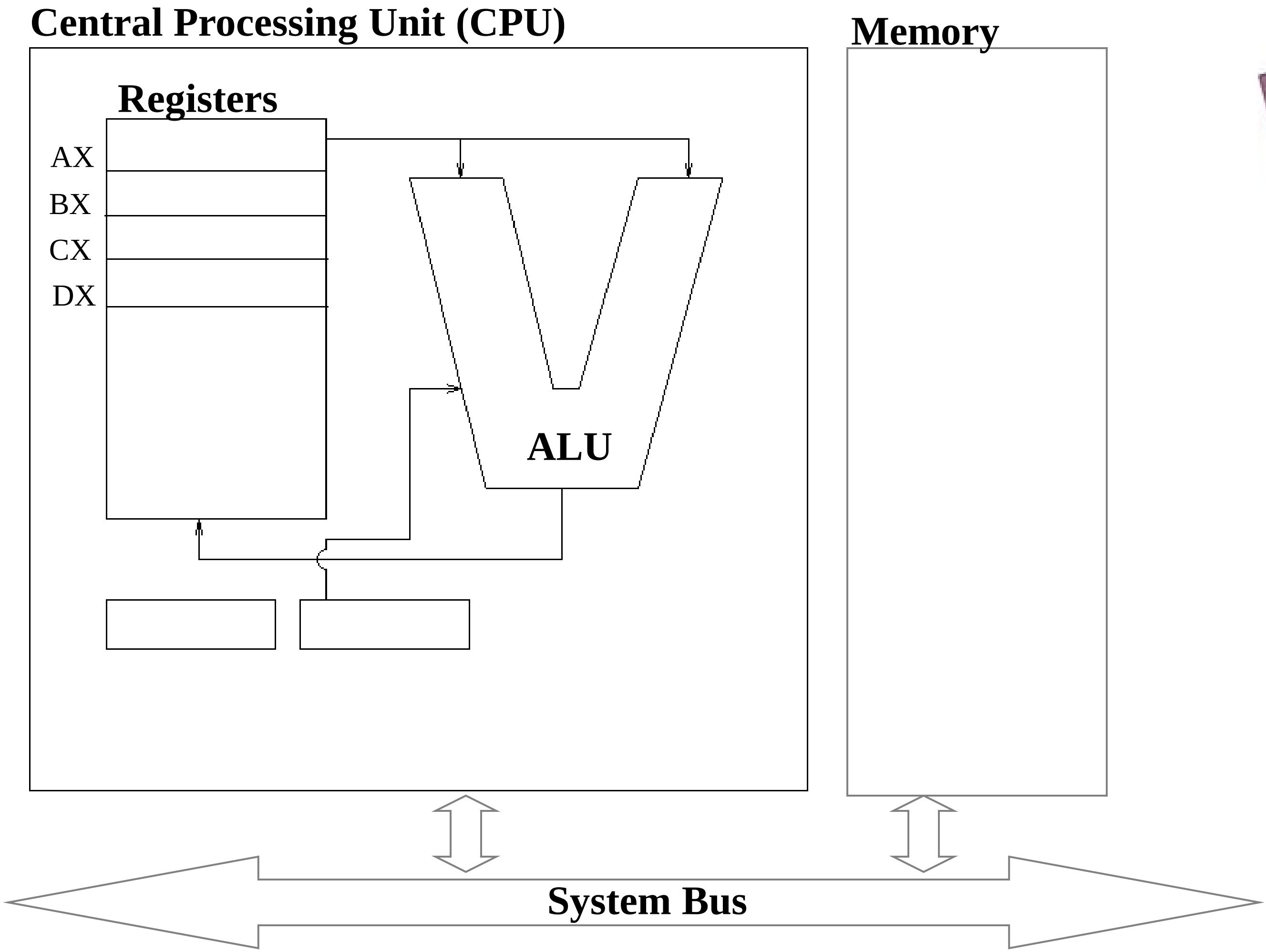
	0	1	2	3	4	13	14	15
0	1	1	0	0	1	0	0	0
1	1	1	1	1	1	0	1	0
2	1	1	1	1	1	1	1	0
3	1	1	1	1	1	0	1	0
4	0	1	1	1	1	0	0	1
5						0	1	1
6	blank							
7	1	1	1	1	1	0	1	0
8	1	1	1	1	1	1	1	0
9	0	0	1	1	1	0	1	0
10	0	0	1	1	1	0	0	1
11	0	0	1	1	1	1	1	0
12	1	1	1	1	1	0	1	0
13	1	1	1	1	1	0	0	1
14	1	1	1	1	1	1	1	0
15	0	0	1	1	1	0	1	0
16	0	0	1	1	1	1	1	0
17						0	1	1
18	0	1	0	1	1	0	0	0
19						1	1	1
20	1	1	1	1	1	1	1	0
21	1	0	1	1	1	0	1	0
22	0	0	1	1	1	0	0	1
23	0	0	1	1	1	0	0	1
24	0	0	1	1	1	1	1	0
25	1	1	0	1	1	0	0	0
26	0	1	0	0	0	0	0	0
27	1	1	0	1	0	0	0	0

—Long Division— (Turing)

Numbers:—

28 –
29 $\bar{z}^d$ (Finally quotient)
30 b
31 a_0 (Finally contains minus the remainder)

PL History: Von Neumann architecture



Central Processing Unit (CPU)

Registers

AX

BX

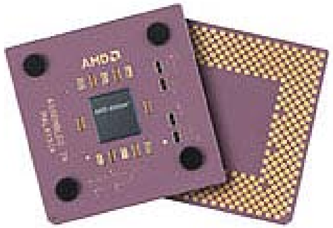
CX

DX

ALU

Memory

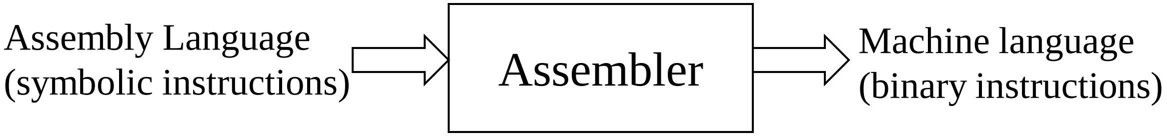
System Bus



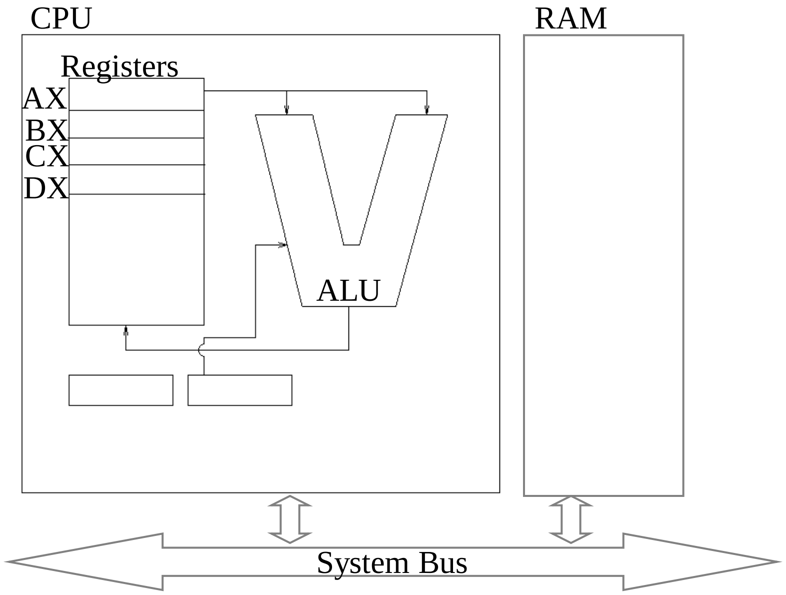
- **How to specify a program?**

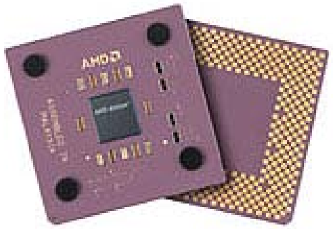
PL History: assembly language

- **Assembly language consist of a set of instructions that are in one-to-one corresponds with machine language**



- **Instructions:**
  - mov
  - add
  - Sub
  - Mul
  - int

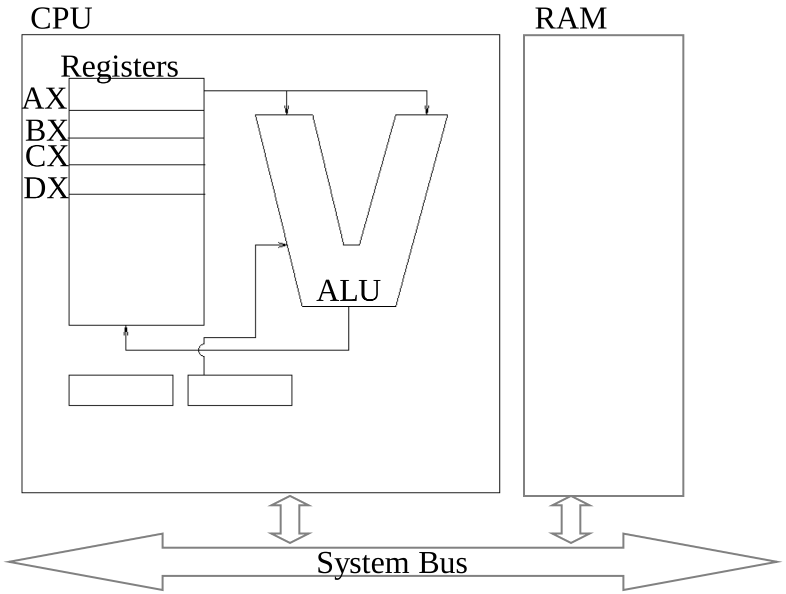




PL History: assembly language

- **Example 1:**
  - Adding 3 numbers (-3,-4 & 10) and multiply result by 6

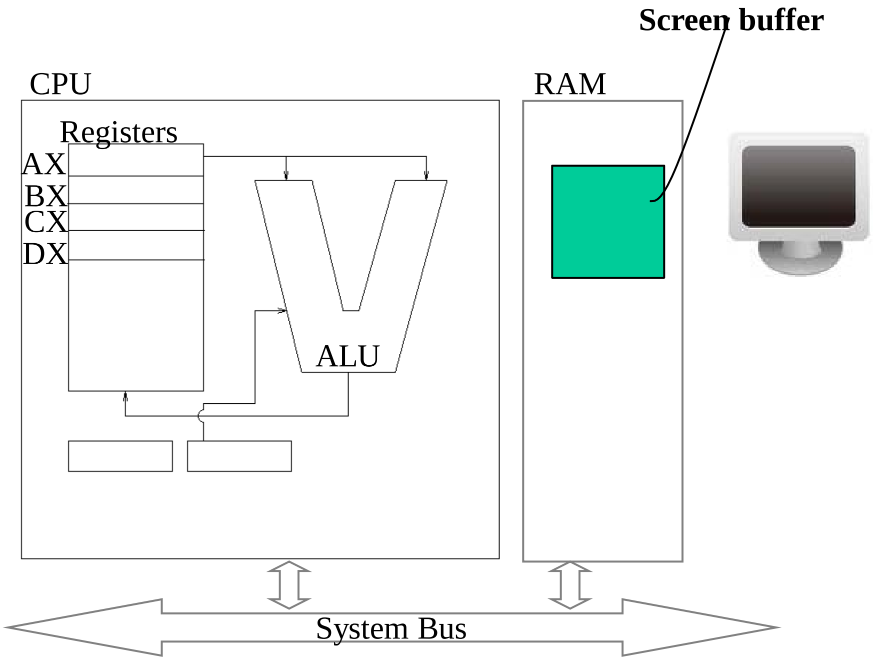
```
MOV  AX, -3
MOV  BX, -4
ADD  AX, BX
MOV  BX, 10
ADD  AX,BX
MUL  AX, 6
```



PL History: assembly language

- **Example 2:**
  - Displaying Hello World screen

```
MOV AH,02H
MOV DX,OFFSET "HELLO$"
INT 21H
MOV AH,02H
MOV DX,OFFSET "WORLD$"
INT 21H
```



PL History: assembly language

- **What’s the problem?**
  - Hard to Write (tedious, very detailed)
  - Hard to Read
  - Hard to Maintain (error-prone)
  - Not Portable (machine-specific)

PL History: what is a PL?

"a language intended for use by a person to express a process by which a computer can solve a problem"

-- Hope and Jipping

"a set of conventions for communicating an algorithm"

-- E. Horowitz

"the art of programming is the art of organizing complexity"

-- E. Dijkstra, 1972

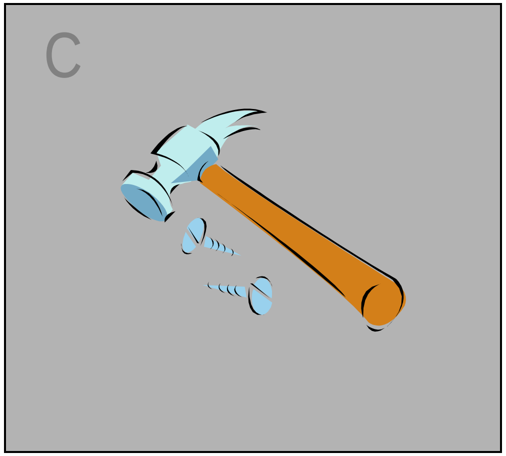
PL History: what is a PL?

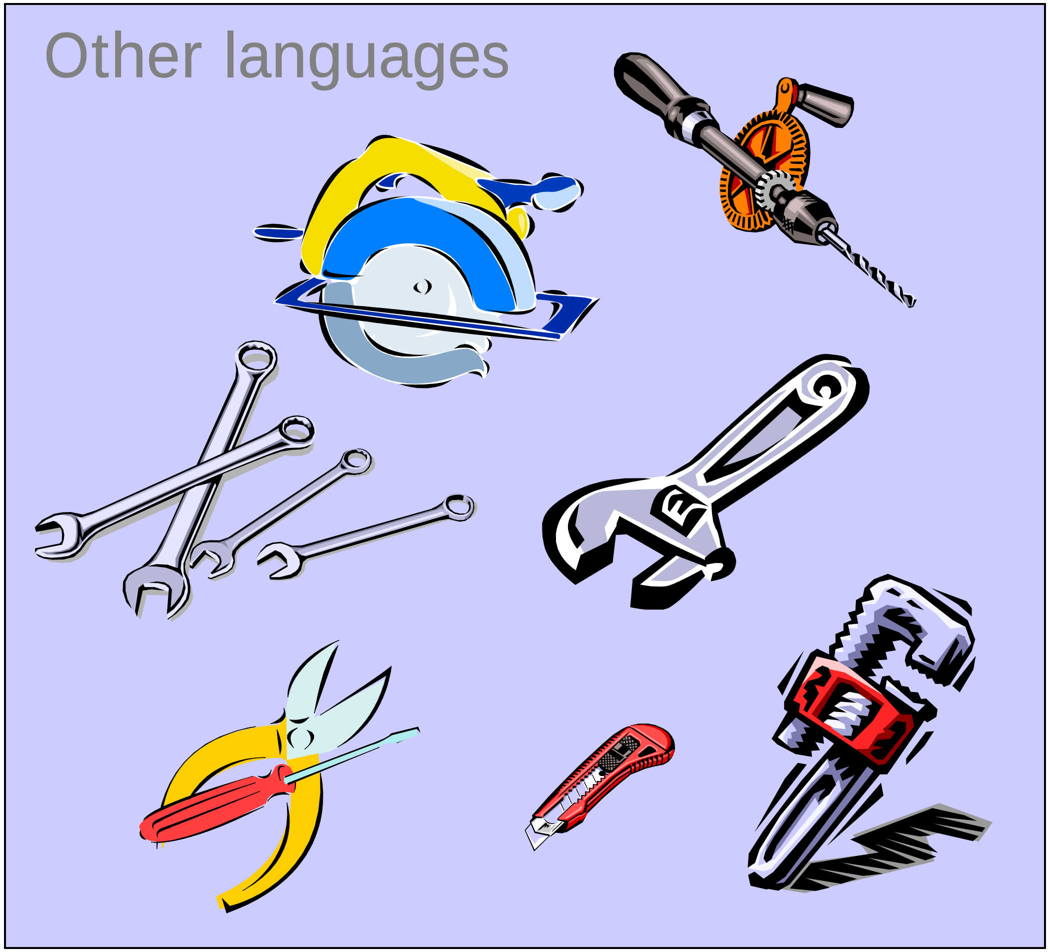
"The main idea is to treat a program as a piece of literature, addressed to human beings rather than to a computer."

Donald Knuth

http://www-cs-faculty.stanford.edu/~knuth/lp.html

PL History: PLs as toolsets





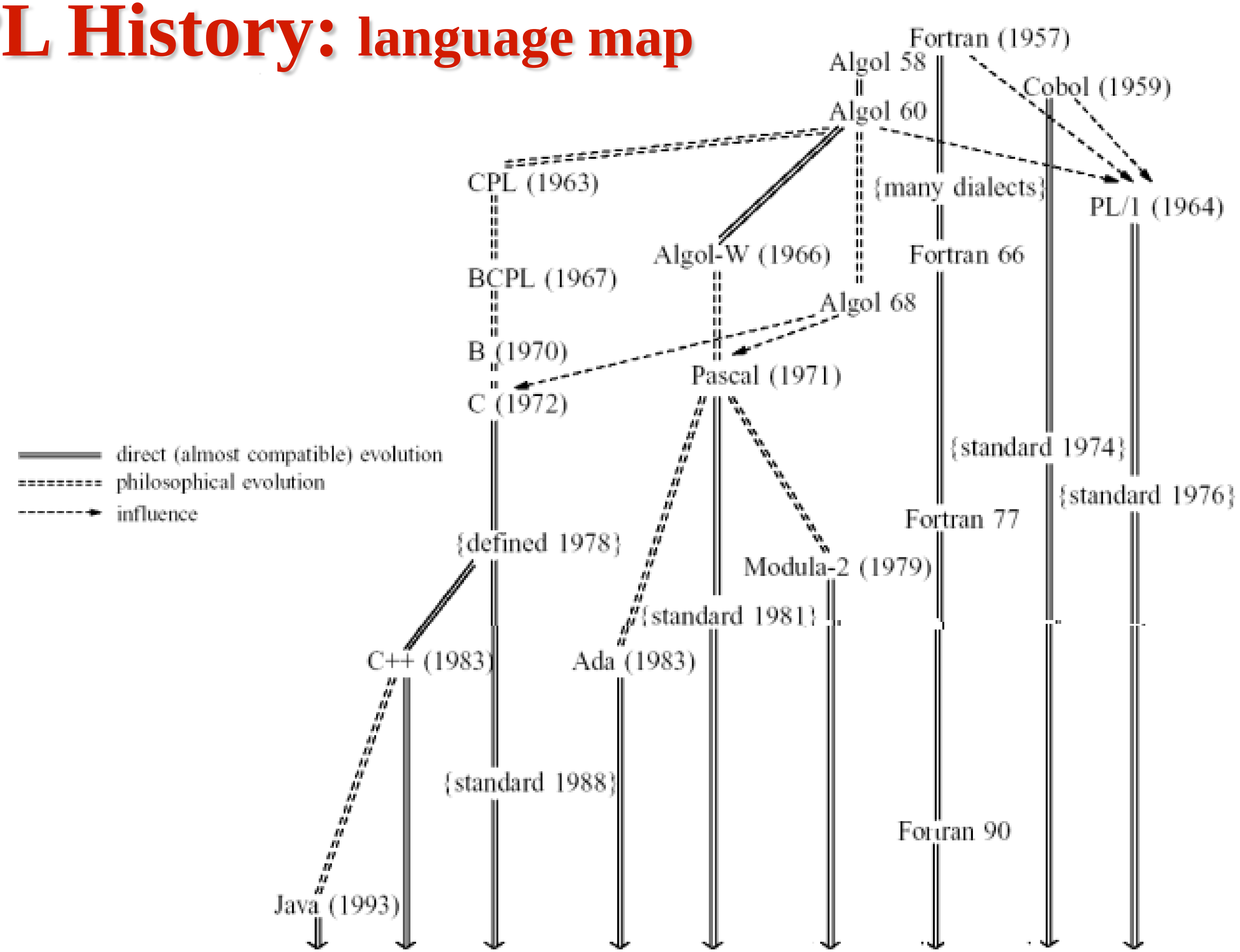
- ***Carpentry view:***

If all you have is a hammer, then everything looks like a nail!

Digression: "A hammer is more than just a hammer. It's a personal tool that you get used to and you form a loyalty with. It becomes an extension of yourself."

http://www.hammernet.com/romance.htm

PL History: language map



PL History: why are there so many PLs?

- We've learned better ways of doing things over time
- Socio-economic factors: proprietary interests, commercial advantage
- Orientation toward special purposes
- Orientation toward special hardware
- Different ideas about what is pleasant to use

PL History: successful/popular languages - why?

- **Easy to learn**
  - BASIC, Pascal, LOGO

- **Easy to express things; Easy use once fluent; ‘Powerful’**
  - C, Perl

- **Easy to implement**
  - Basic

- **Possible to compile to very good (fast/small) code**
  - Fortran

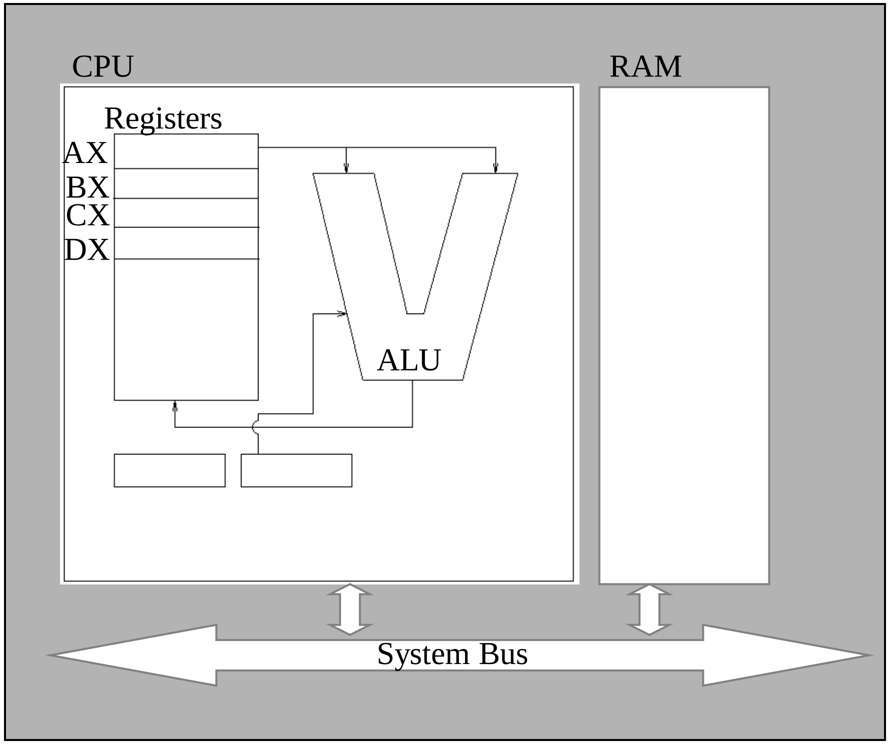
- **Backing of a powerful sponsor**
  - Ada, visual basic

- **Wide dissemination at minimal cost**
  - Pascal, java

PL Paradigms

PL Paradigms: imperative

- **Underlying notion of an abstract machine**
  - Von Neumann architecture
    - Store (memory)
    - Accumulator (ALU)
    - Load/store into memory
  - Key operation: assignment



PL Paradigms: imperative examples

Sum up twice each number from 1 to N.

Fortran

```
       SUM = 0
       DO 11 K=1,N
       SUM = SUM + 2 * K
11     CONTINUE
```

C

```
sum = 0;
for (k=1; k <= n; ++k)
   sum += 2*k;
```

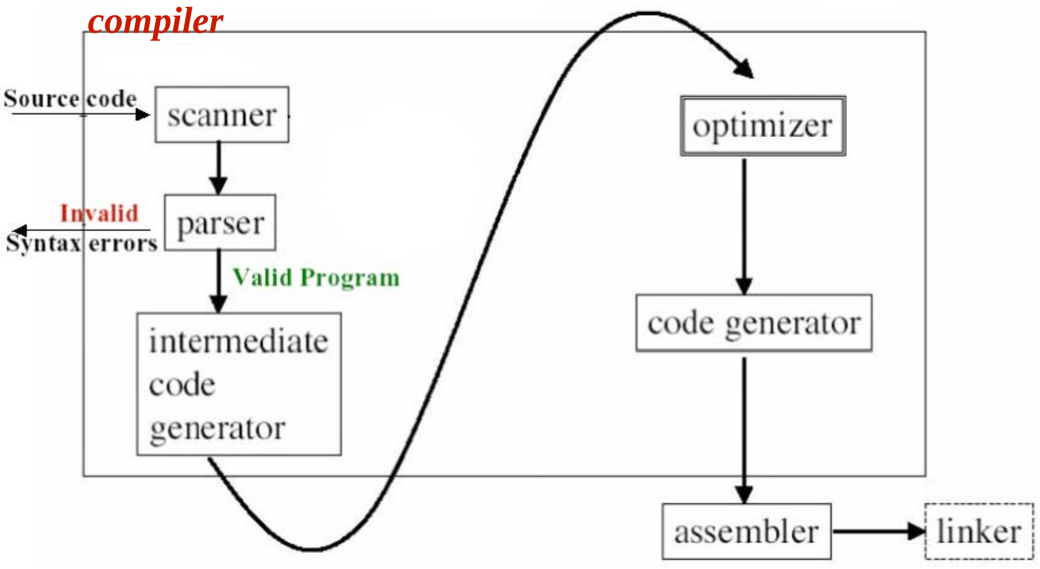
Pascal

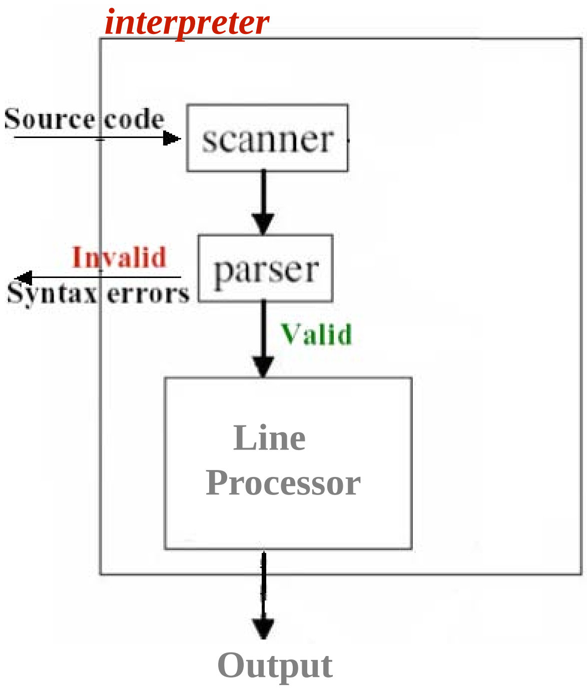
```
sum := 0;
for k:= 1 to n do
   sum := sum + 2 * k;
```

Compilation vs. Interpretation

- **Compilation**
  - Translation of a program written in a high-level PL into a form that is executable on the machine *(done by compiler)*

- **Interpretation**
  - A program is translated and executed one statement at a time *(done by interpreter).*





PL Paradigms: imperative vs. assembly

```
int main() {
  int nIndex,nSum;
  for( nIndex=0; nIndex<10;nIndex++)
      nSum =+ 2 * nIndex;
}
```

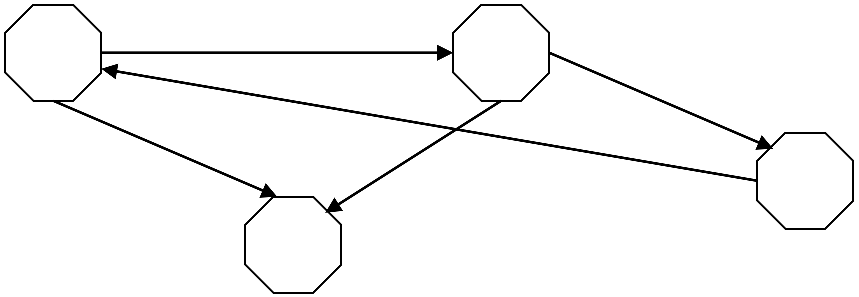
```
 .file   "foo.c"
                 .text
                 .p2align 4,,15
.globl main
                 .type   main, @function
main:            push   BP
                 mov    $9, AX
                 mov    SP, BP
                 sub    $8, SP
                 and    $-16, SP
                 .p2align 4,,15
.L6:             dec    AX
                 jns    .L6
                 mov    BP, SP
                 pop    BP
                 ret
                 .size   main, .-main
                 .ident  "GCC: (GNU) 3.3.1"
```

```
01010101010001
10101010101111
10101001010101
10010101001000
00000001101111
00000000000000
11111111100001
```

Try this: gcc -O2 -S -c foo.c

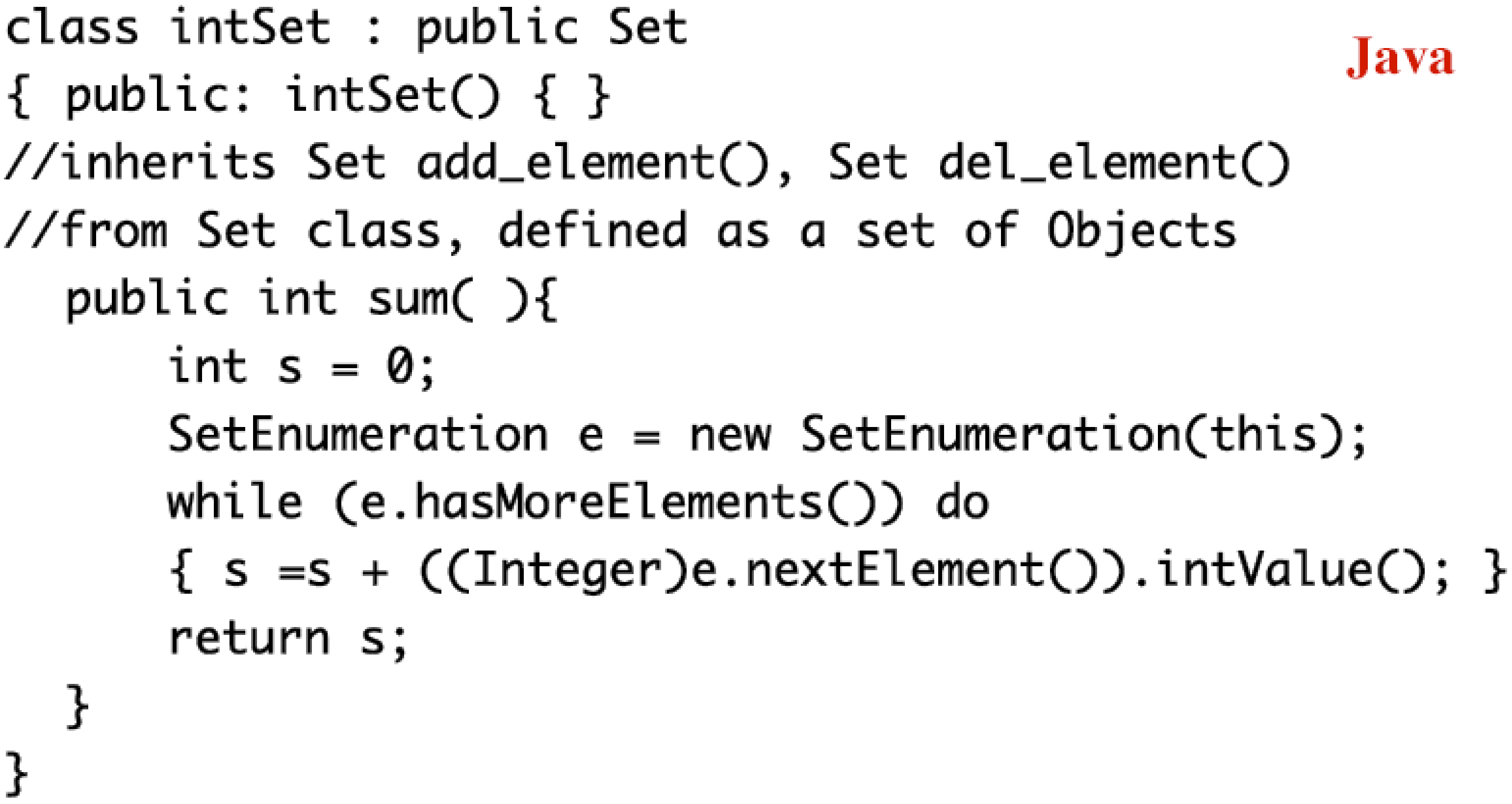
PL Paradigms: object oriented

- **Organizes a program to be operations on abstract representations of the data**
  - Objects with data abstraction and information hiding
    - Object implementation is hidden from user
  - Actions performed on objects (messages)
  - Key operation: message passing



PL Paradigms: object oriented example

Java



```
class intSet : public Set
{ public: intSet() { }
//inherits Set add_element(), Set del_element()
//from Set class, defined as a set of Objects
  public int sum( ){
      int s = 0;
      SetEnumeration e = new SetEnumeration(this);
      while (e.hasMoreElements()) do
      { s =s + ((Integer)e.nextElement()).intValue(); }
      return s;
  }
}
```

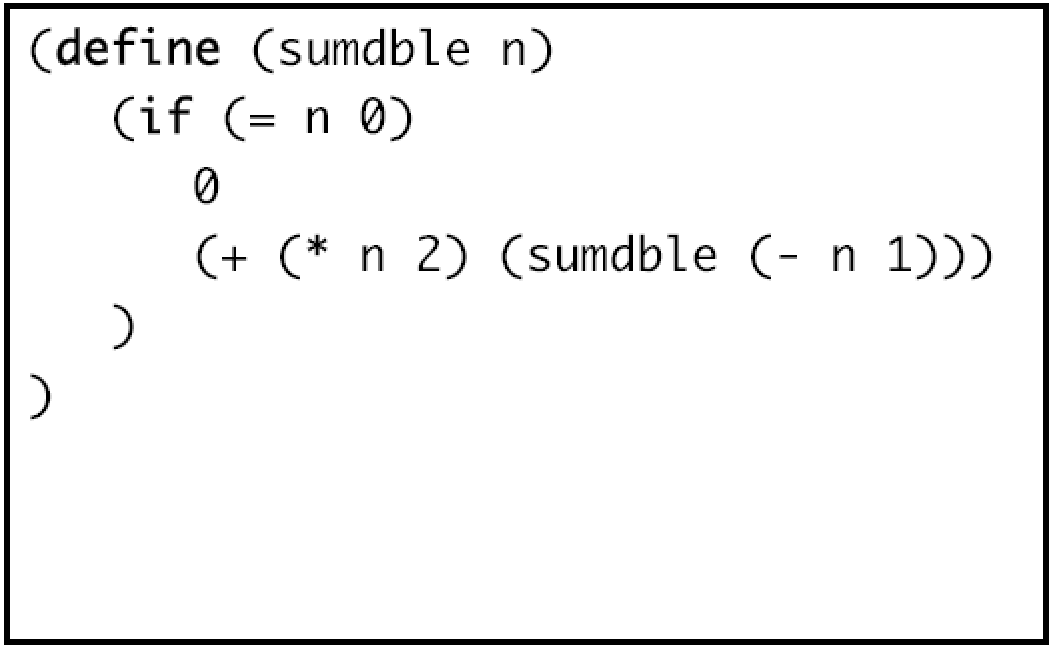
PL Paradigms: functional

- **Process of problem solution expressed as a sequence of operations on the data**
  - (Pure) value binding through parameter passing
  - No store accessible through names
  - No iteration
  - Key operation: function application (with recursion)

PL Paradigms: functional language

$$\begin{aligned}\int x\cos(x)\,dx &= \int u\,dv\\ &= uv - \int v\,du\\ &= x\sin(x) - \int \sin(x)\,dx\end{aligned}$$

Scheme



```
(define (sumdble n)
   (if (= n 0)
      0
      (+ (* n 2) (sumdble (- n 1)))
   )
)
```

PL Paradigms: logic

- **Program is a formal description of characteristics required of a problem solution**
  - Programs tell what should be not how to make it so
  - Solutions through a reasoning process called theorem proving

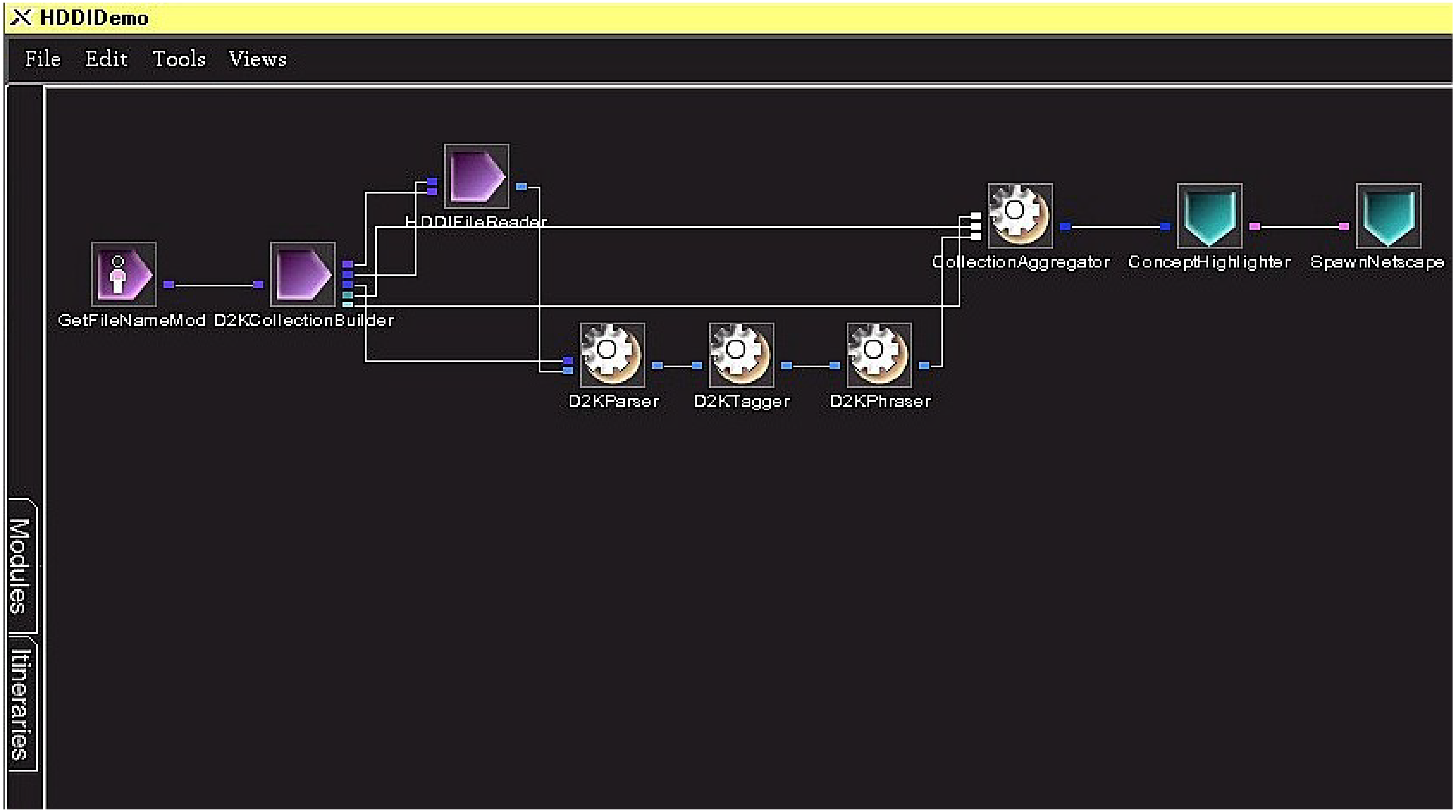
PL Paradigms: logic language example

```
sum(0,0).
sum(N,S) :- NN is N - 1,
            sum(NN, SS),
            S is N * 2 + SS.
```

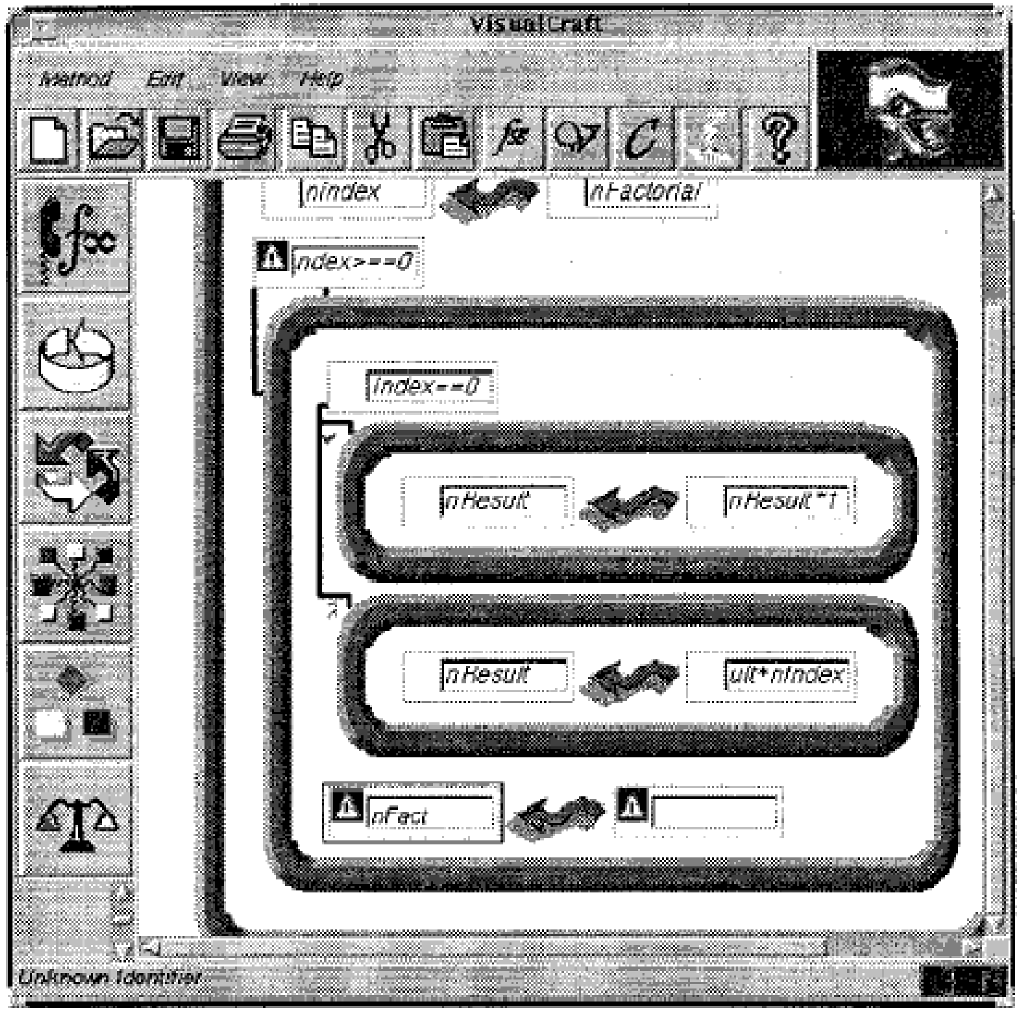
Prolog

```
?- sum(1,2).
yes
?- sum (2,4).
no
?-sum(20,S).
S = 420
?-sum (X,Y).
X = 0 = Y
```

PL Paradigms : visual languages



PL Paradigms : visual languages



nIndex

nFactorial

ndex>==0

Index==0

nResult

nResult*1

nResult

ult*nIndex

nFact

Unknown Identifier