Laura Clark

Development Tools Product Marketing
Lisp Product Marketing Manager

Macintosh Allegro Common Lisp Update

Overview

What You Expect in a Powerful Lisp Environment . . .

- Full Common Lisp
- EMACS-style editor
- Inspector
- Debugging tools
- Lisp Listener

What You Want in a Macintosh Development Environment . . .

- Fast, compact compiler
- High-level access to the toolbox
- Object library defines windows, dialogs, and menus
- Interactive interface designer

Why Lisp?

- Interactive
- Dynamic
- Memory management
- Run-time error handling
- Object technology

Macintosh Allegro CL v.1.3.2

- Released in April 1990
- Interface designer with source
- Programmable grapher with source
- Windoids
- FFI support MPW 3.0 object files
- Color dialogs and menus

In the Future

- System 7.0 support
  - Virtual Memory
  - AppleEvents
- Ephemeral GC
- CLOS
- Steele second edition compatibility
- Small applications

Bill St. Clair

Macintosh Allegro Common Lisp
Lisp Hacker - ATG Cambridge

Macintosh Allegro Common Lisp Update

What’s new in MACL 2.0 Technical Details

MACL 2.0–New Features

- CLtL2 compatible (186 cleanup issues)
- CLOS replaces Object-Lisp
- New representation
  – 32-bit clean, VM, dynamic-extent
- Views replace Dialogs
- New inspector and debugger

Common Lisp Object System

Common Lisp

- CLOS is the standard
- Seamlessly integrated
  - Generic functions behave just like regular functions
  - Classes are part of the type system
- Featureful (and then some)

Object System

- Class/Instance paradigm
- Instance “slots” for local state
- Class “slots” for shared state
- Methods provide specialized behavior for generic functions
- Multiple inheritance

Object-Lisp

- Prototype/Instance paradigm
- Local variables for local state
- Inherited variables for shared state
- Object functions provide specialized behavior
- Multiple inheritance
- Object-Lisp is dead. Long live CLOS.

Object-Lisp => CLOS

- defobject + exist =>
  defclass + initialize-instance
- defobfun => defmethod (automatic)
- object-var => (slot-value ...)
- (ask instance (f x y)) => (f instance x y)
- usual-xxx => call-next-method

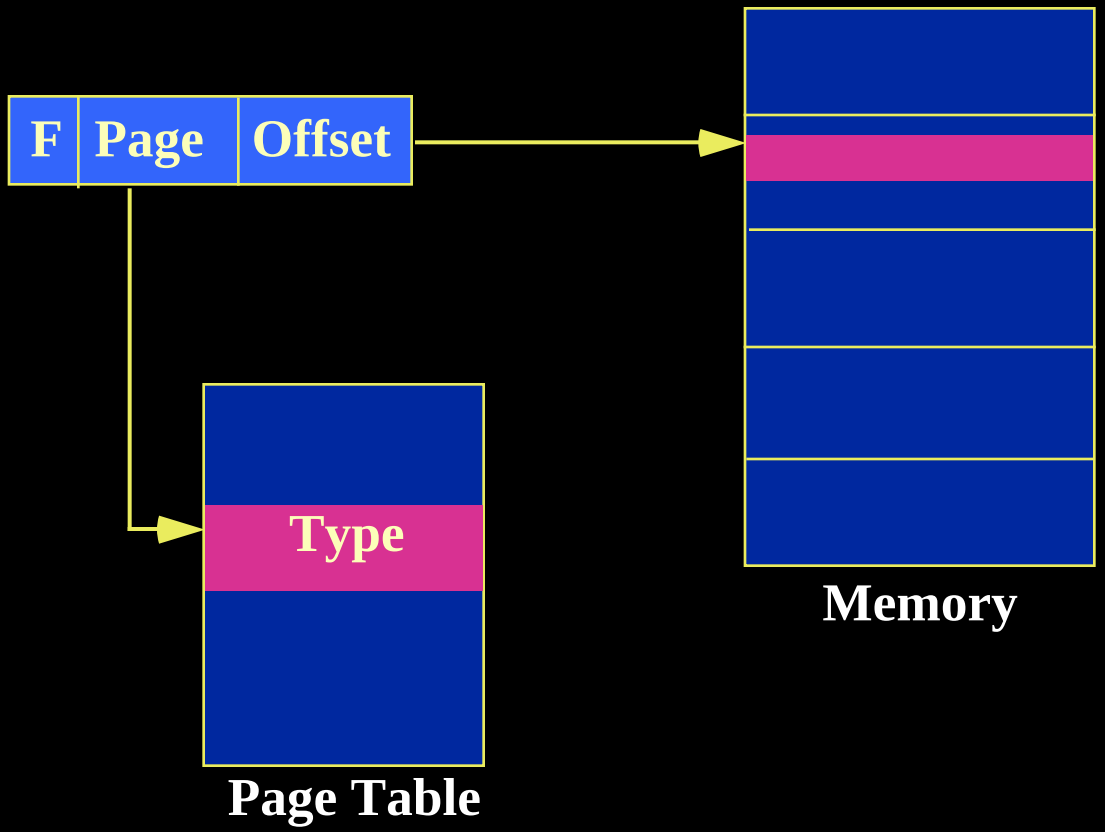
Statistics

	1.3 OL	2.0 CLOS	2.0 PCL	1.3	2.0
Function-call	8	3	9	2	1
Slot-value	21	11	41	1	1
Accessor	21	18	23	1	1
Make-instance	1.6	1	8-50		
Instance size	4N+8	N+3	N+6		
System size	16K	200K	1M		

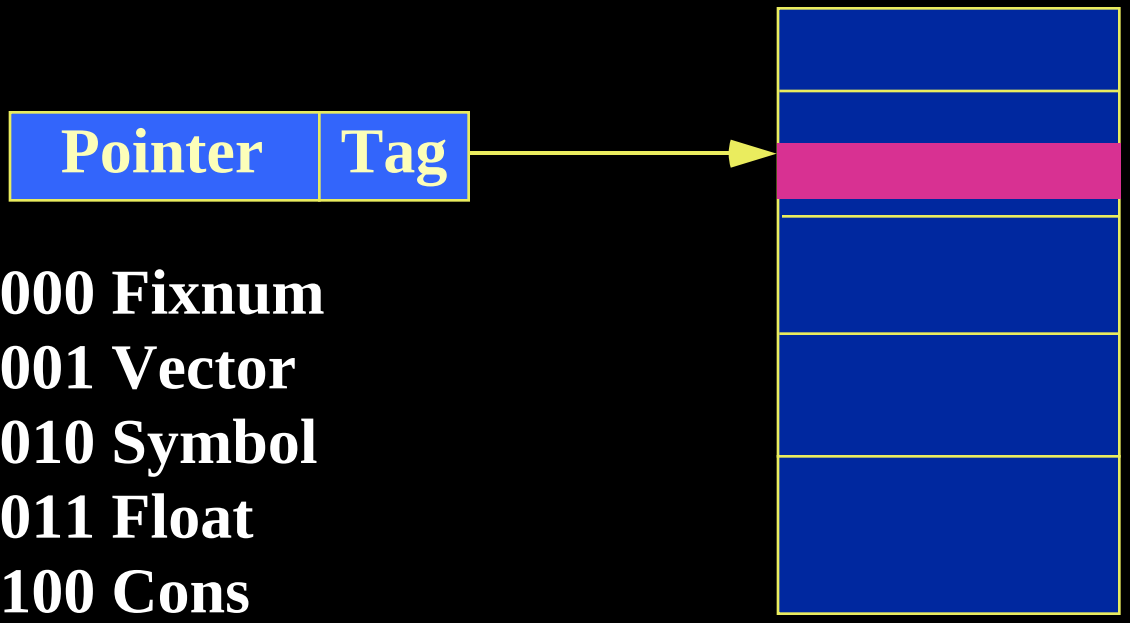
New Representation

- BBOP => low-bits tag
- Stack-consed (downward) closures
- Dynamic-extent
- Ephemeral gc
- Virtual memory: copying gc

BBOP (Big Bag of Pages)



Low-Bits Tag



000 Fixnum
001 Vector
010 Symbol
011 Float
100 Cons
101 Reserved
110 Lfun
111 Immediate

New Inspector and Debugger

- Inspected values can be modified
- Cut and paste of Lisp objects
- Return-from/restart frame
- Condition system restarts
  - Unbound variable
  - File open error

Matthew MacLaurin

SIAC / SQA
Lead Engineer–Automation

Macintosh Allegro Common Lisp Update

GATE: A Case Study

GATE

Generalized Automated
Test Environment

- A “user emulator” for software testing
- Written in Lisp, C, and Modula-2
- In use at Apple today

Project Features

- Emulates user actions–drag, type, etc.
- Understands Macintosh interface and applications
- Writes test scripts
- Checks test results
- Tests multiple CPUs simultaneously
- Handles target malfunction gracefully

Multi-Lingual Development

- Used existing toolkit written in C
- New utilities written in Modula/2
- Interface written in Object Lisp
- Main program logic in Common Lisp

Why Lisp?

- Project challenges
  - Few precedents
  - Moving target
  - Algorithmically nasty problem
- Lisp advantages
  - Development time
  - Environment

What Makes Sense in Lisp?

- Complex, large scale projects
- Dynamic projects
- Smart applications
- Knowledge-based systems
- Things that haven’t been done before

What Doesn’t Make Sense in Lisp?

- Device drivers
- Real-time–unless you’re very careful
- Applications which don’t stress functionality

My Favorite Features

- Multi-tasking
- Foreign function interface
- Interface designer
- Interactive environment

How to Lisp in Your Organization

- Think functionality
- Do your homework re: Lisp style
- Integrate existing code in C, Pascal

The power to be your best.