vms Software

A history of technology

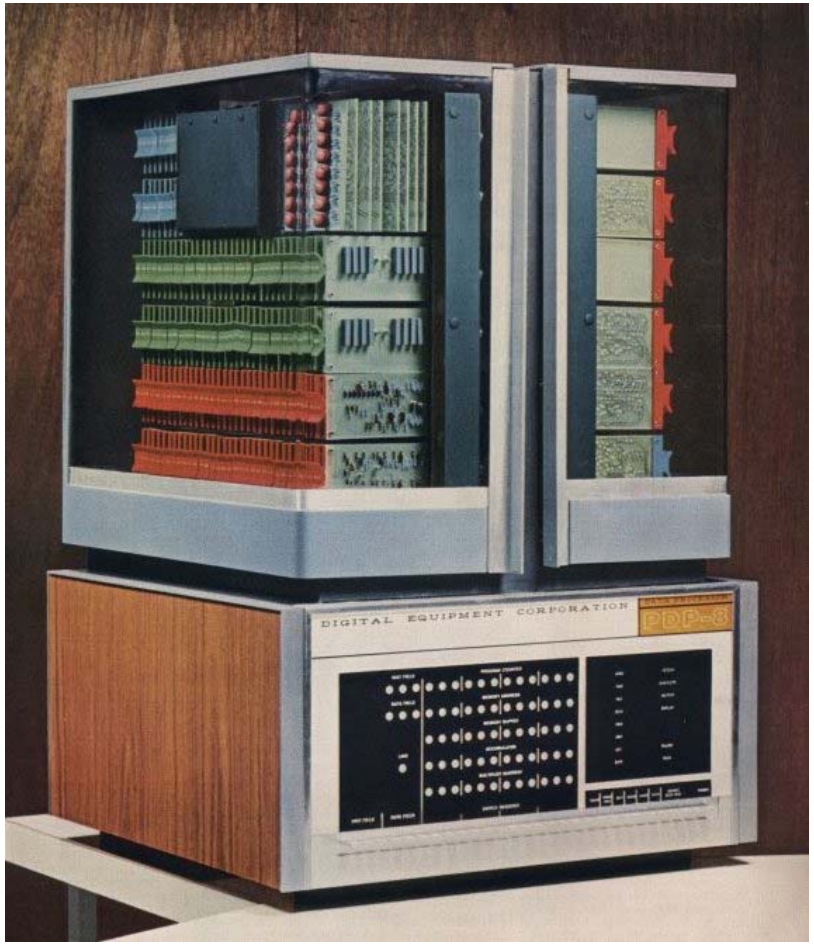
VMS: The First 40 Years

Andy Goldstein

Prehistory: DEC and the PDP-1

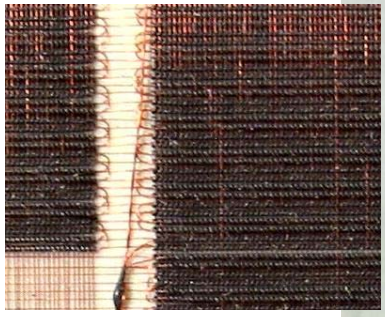
- Company founded in 1957
- PDP-1 – first computer in 1960
- First computer to cost less than $1Million

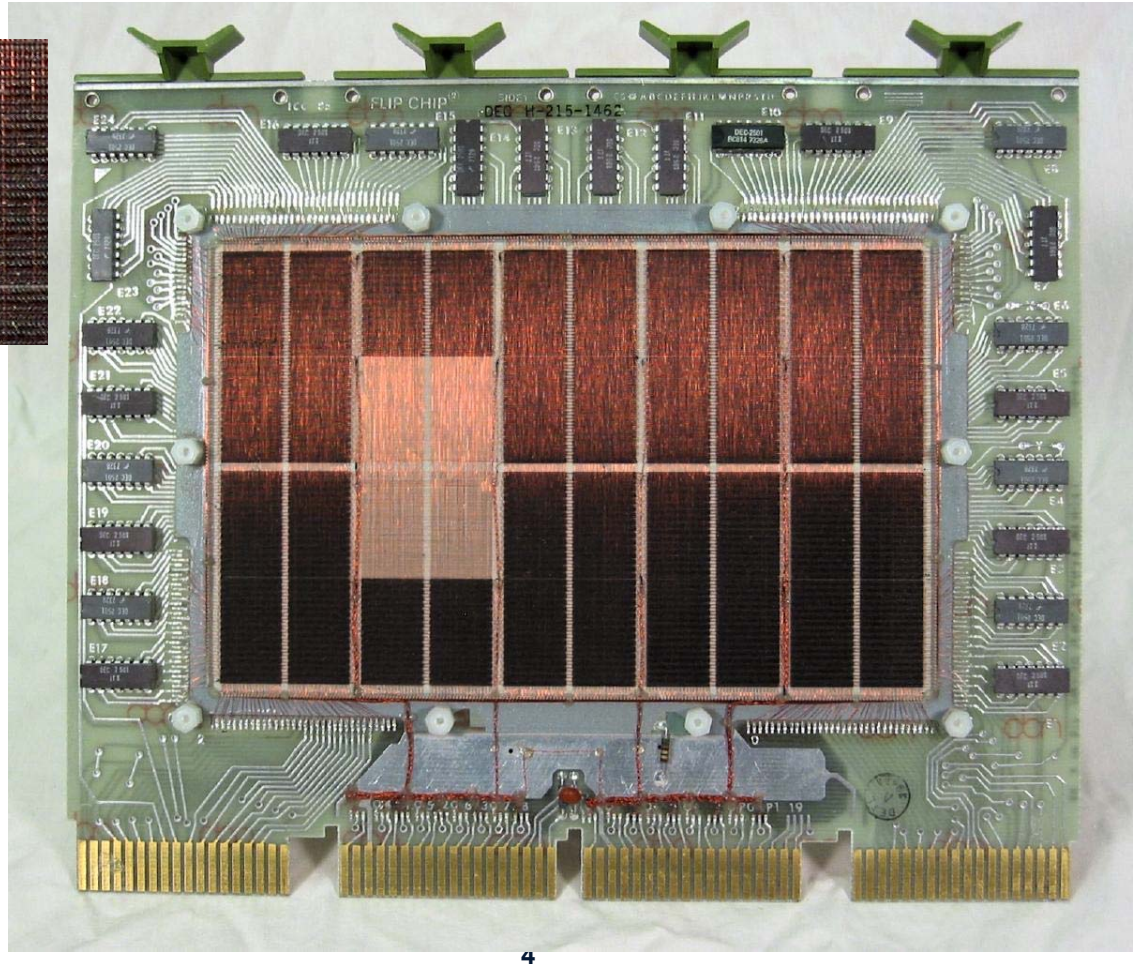
PDP-8 – The First Desktop Computer



- 12 bits
- 4K words memory
- Disk-based operating system

Memory – the Old Way

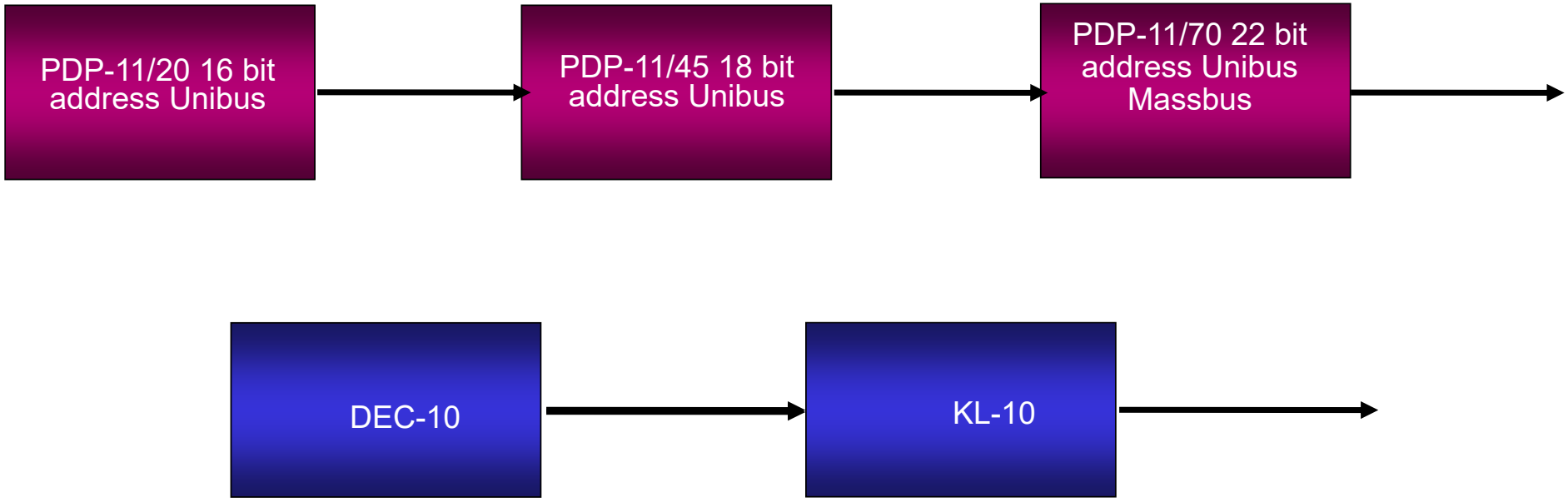




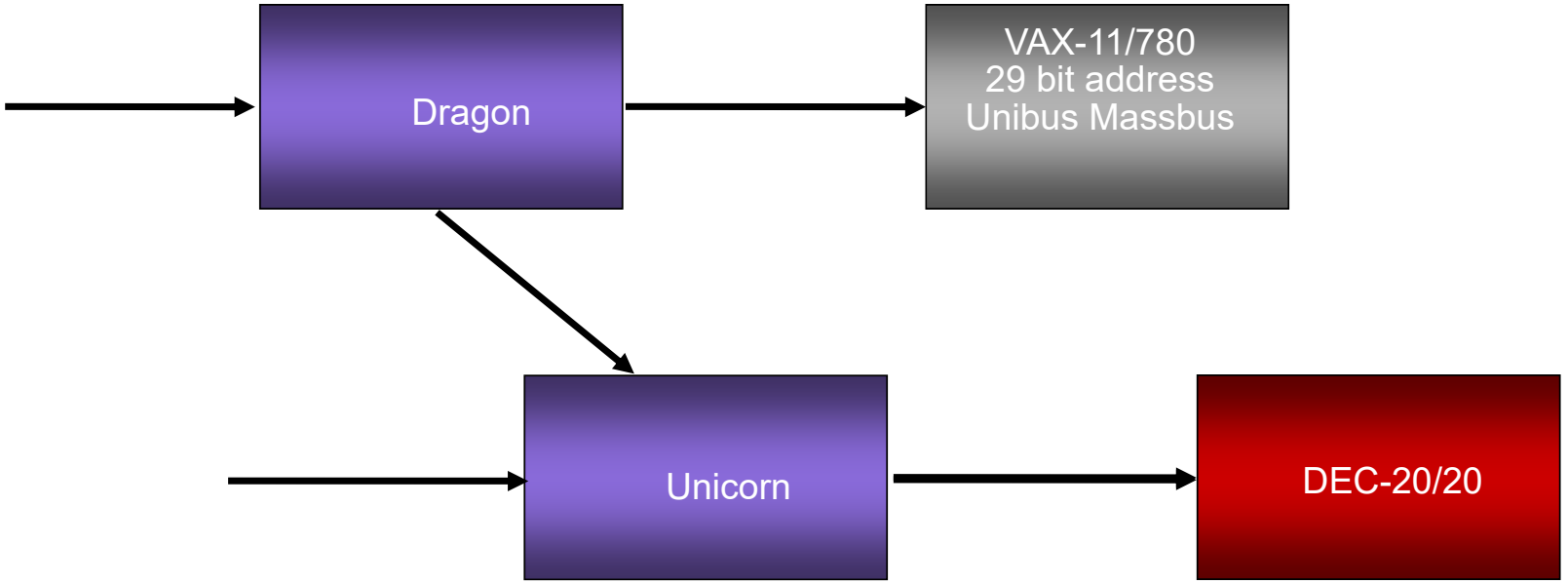
Beginning of the Byte Architecture

- PDP-11 - 16-bit architecture
- Multiple operating systems
- 1974: Should we build a 32-bit PDP-11?

Genealogy of the VAX



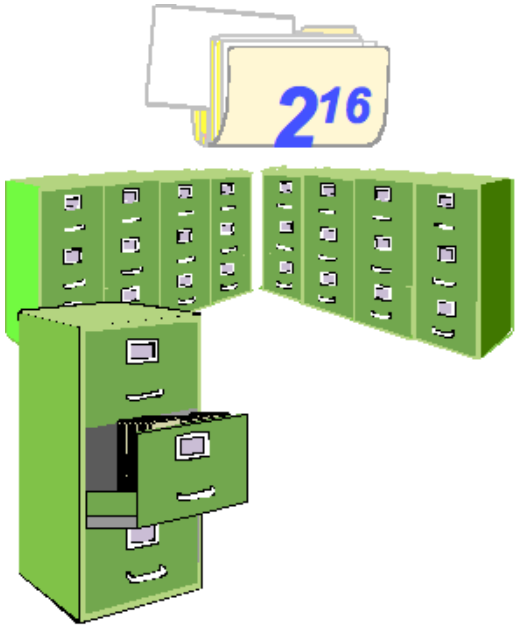
Genealogy (continued)



1975: STAR and STARLET goals

- April 1975: Gordon Bell says “Go”
- Integrated hardware and software design
- Expand addressing to 32 bit
- Highly scalable architecture
- One system, compatible tools

32 Bits – Do the Math



- Eliminates software “overlays”
- Critical software stays resident
- Improved performance
  - Programmer efficiency
  - Program execution

VAXA Committee

- Gordon Bell
- Peter Conklin
- Dave Cutler
- Bill Demmer
- Tom Hastings
- Richie Lary
- Dave Rogers
- Steve Rothman
- Bill Strecker, chief architect

Early Development

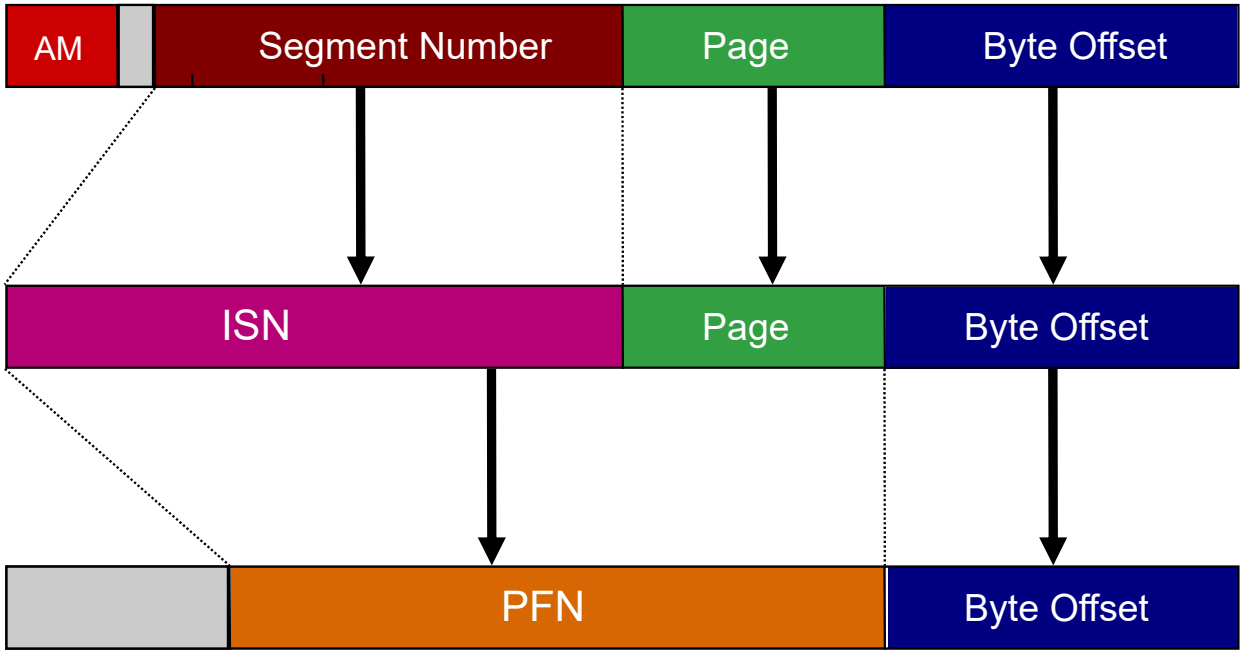
- Sept 1975 SRM Rev 1
- April 1976 April Task Force
- June-Aug Detailed software design
- Sept 1976 Hardware simulator and initial system kernel
- April 1977 DCL and file system
- June 1977 Breadboard and first VMS timesharing

Initial VMS Design Team

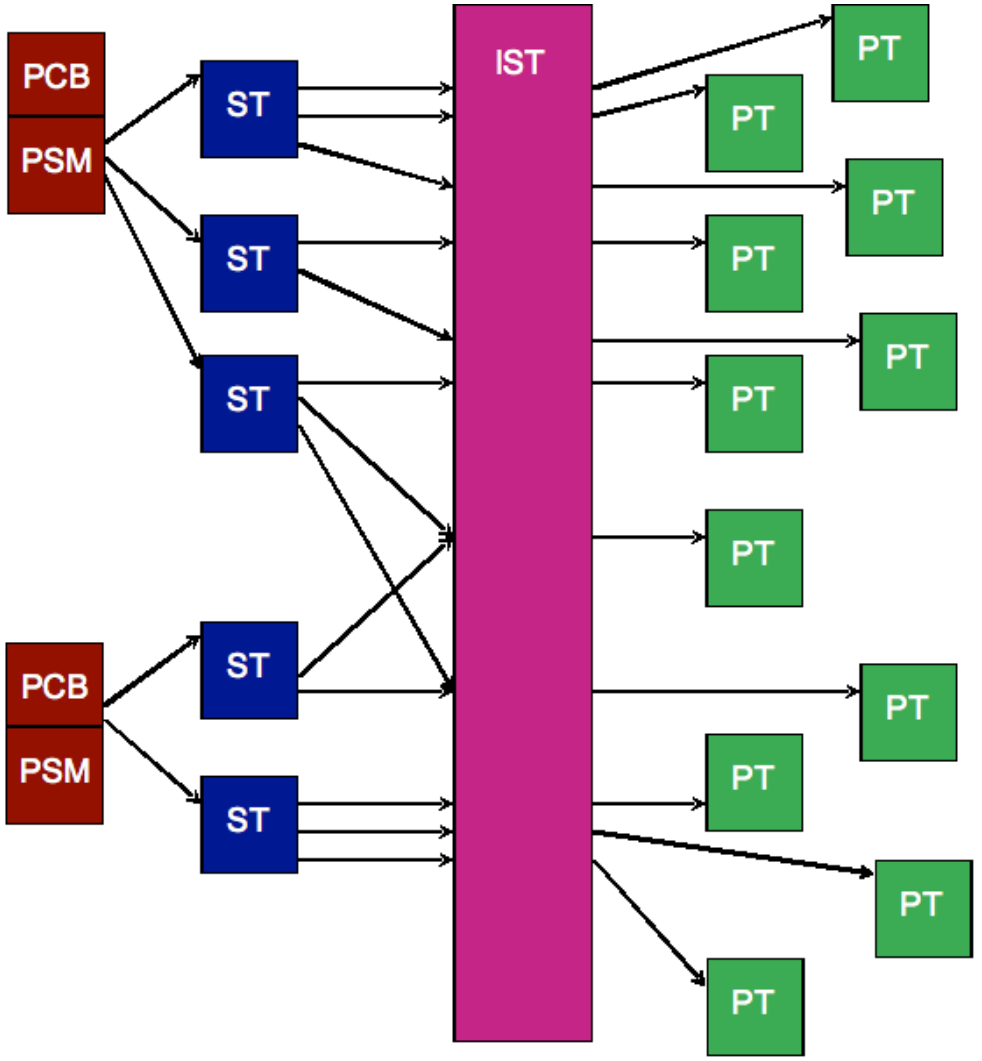
By November, 1975...

- Dave Cutler, project leader
- Andy Goldstein
- Roger Gourd, manager
- Roger Heinen
- Dick Hustvedt
- Hank Levy
- Peter Lipman
- Trev Porter

SRM V1 Memory Management

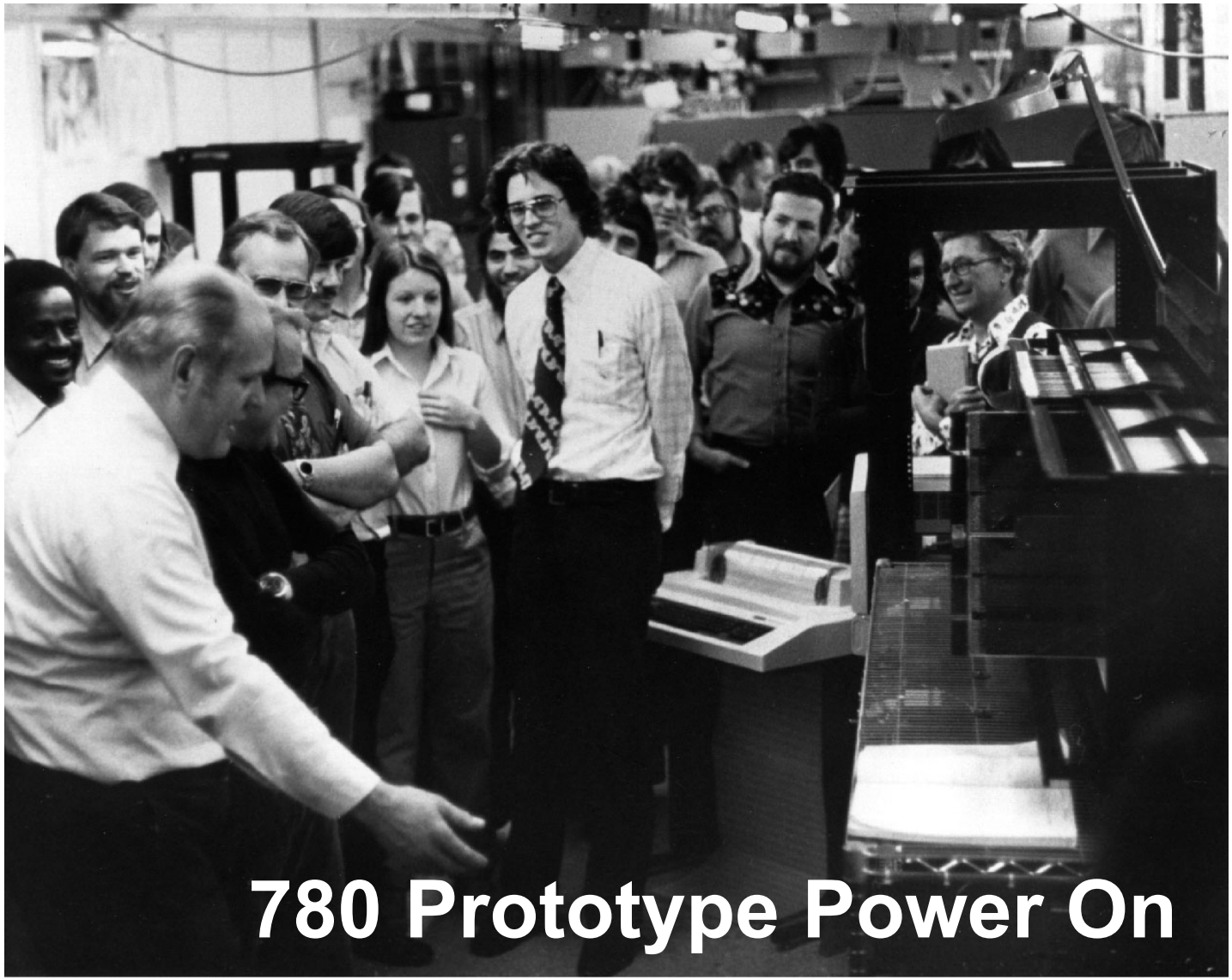


Address Translation



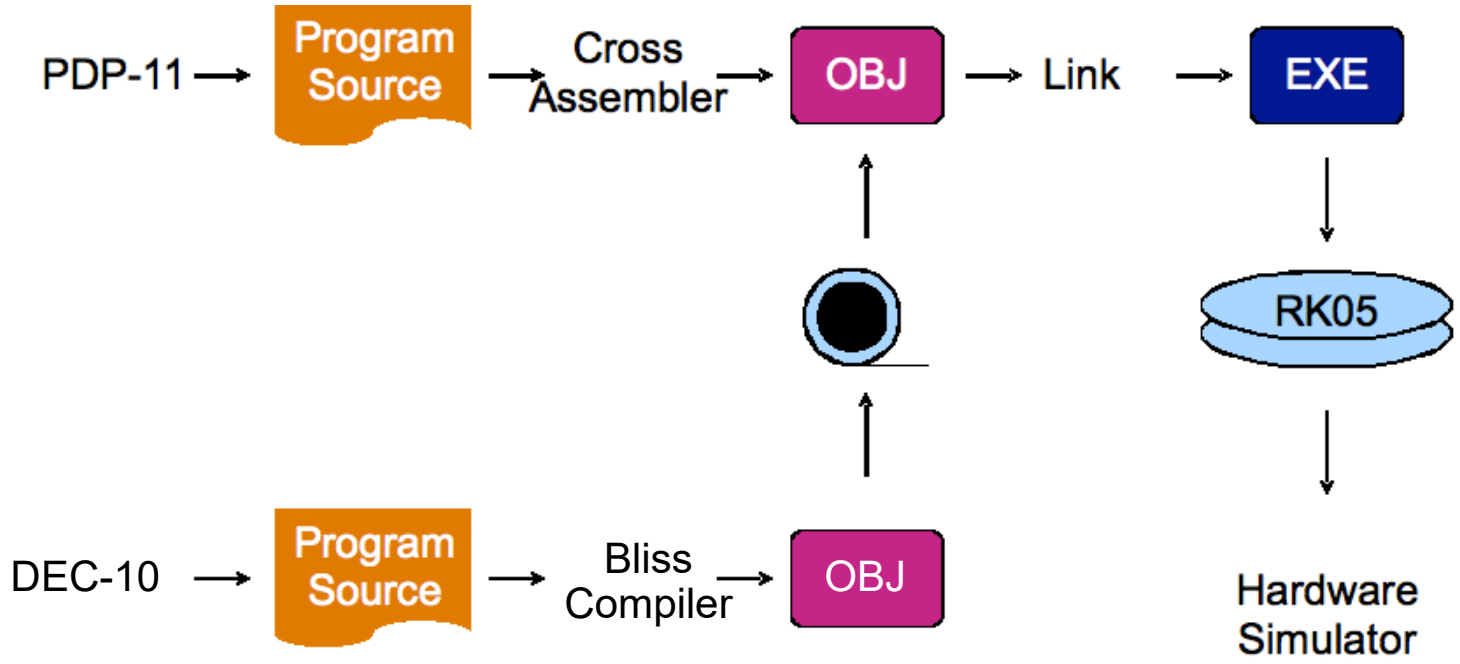
Early Development

- Sept 1975 SRM Rev 1
- April 1976 April Task Force
- June-Aug Detailed software design
- Sept 1976 Hardware simulator and initial system kernel
- April 1977 DCL and file system
- June 1977 Breadboard and first VMS timesharing



780 Prototype Power On

Program Development and Testing



Timesharing on the Prototype

- Prototype 780, 1MB memory
  - 2 RP06 + RK07
- VT52s in the offices
- Self-supporting
  - System builds
  - Bliss Compiler
  - “Eat our own dog food”

1977 1978 1979 1980 1981 1982...

- October 25, 1977
- VAX-11/780
- VMS V1.0 Announced

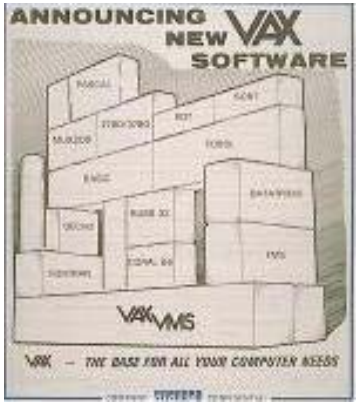
October 1977 Announcement

V1.0 Development Team

1977 **1978 1979** 1980 1981 1982...

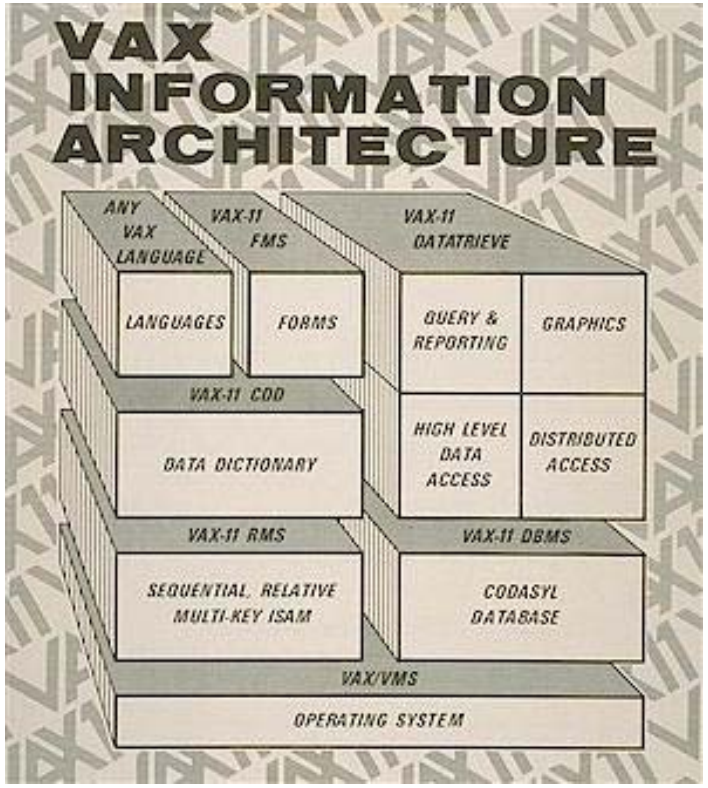
- VMS V1.0 Shipped
- DECnet Phase II
- FORTRAN IV
- Up to 64 MB Memory

1977 1978 1979 **1980** 1981 1982...



- DECnet Phase III & Ethernet
- VMS V2.0
- New Languages & Tools
- VAX-11/750

1977 1978 1979 1980 **1981** 1982...



- VAX Information Architecture
- Common Data Dictionary
- RMS and VAX-11 DBMS
- Datatrieve
- CALLable From Any VMS Programming Language

VMS Technology Highlights

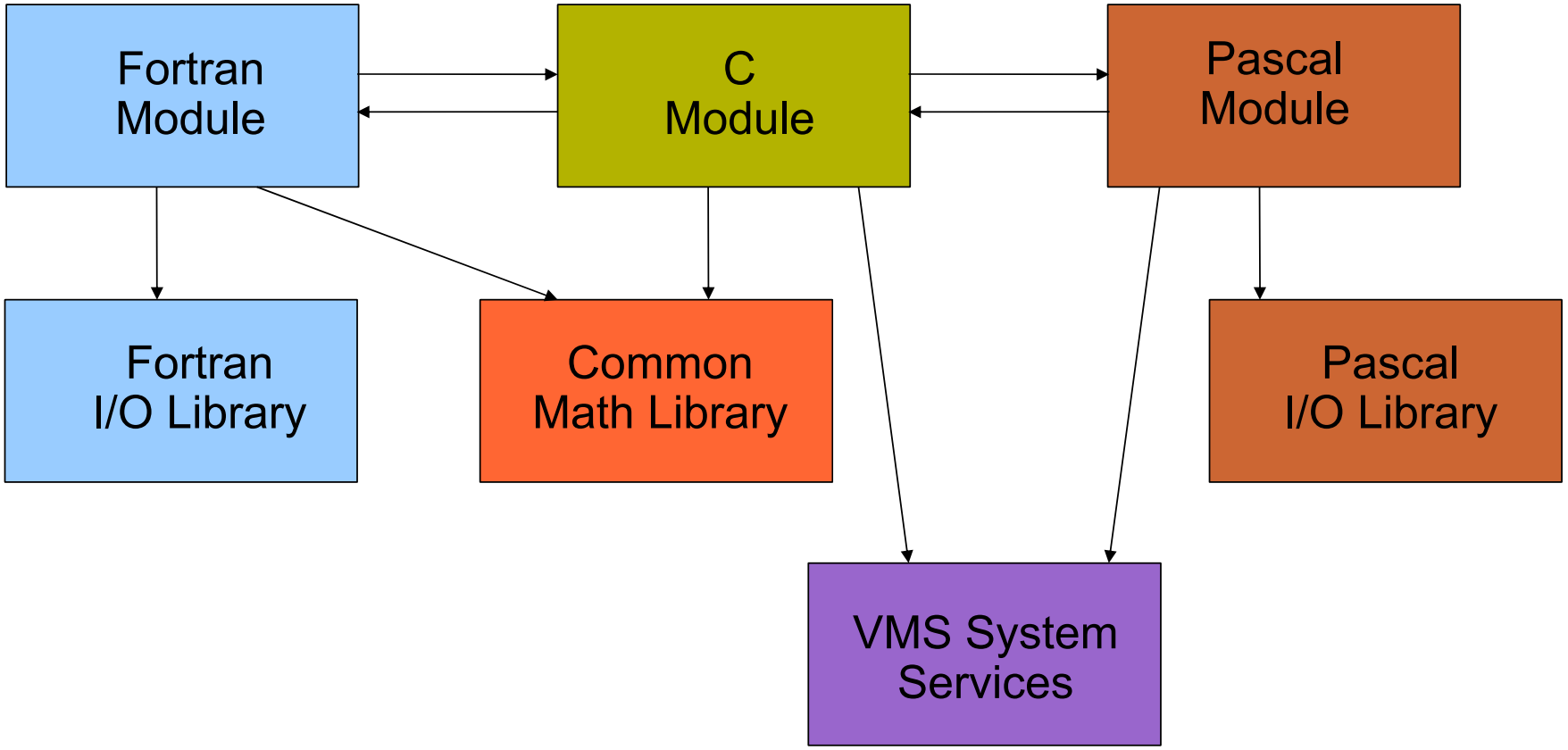
- OpenVMS Calling Standard
- VMSclusters
- Symmetric Multiprocessing
- The Alpha Port
- OpenVMS Galaxy
- The Itanium Port

OpenVMS Calling Standard

A common binary interface between software modules, regardless of language

- Argument list format
- Register conventions
- Descriptors
- Condition handling

OpenVMS Calling Standard



VAXes Get Small

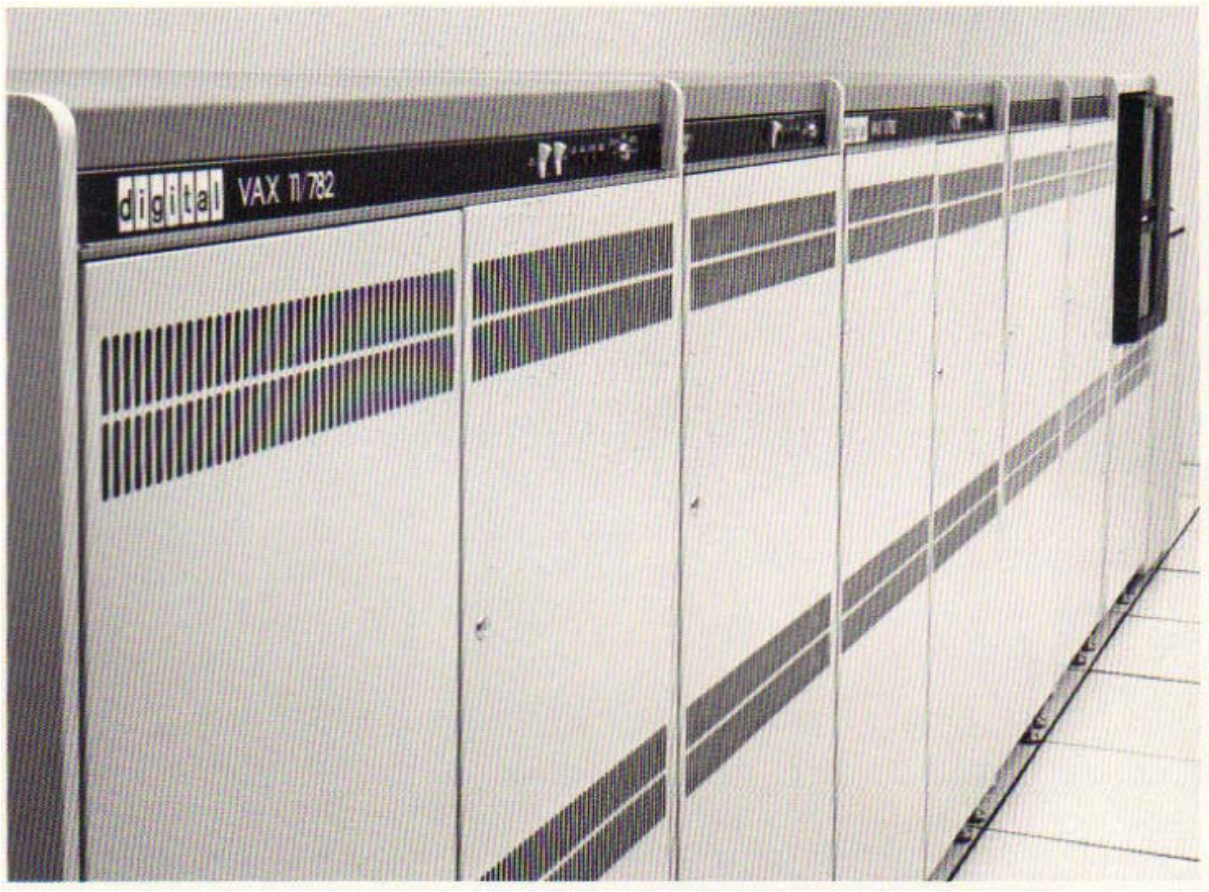
- MicroVAX
- VAXstation 2000
- and more
- CVAX Chip…
  When You Care Enough to Steal the Very Best!

DECwindows

VMS becomes a workstation

- Graphics device drivers
- Port of X-11 and OSF Motif
- Session manager menu items:
  - DCL shell script
- Existing character cell apps:
  - Partition into character cell UI and callable application logic
  - Add new windows UI

VAXes Get Big



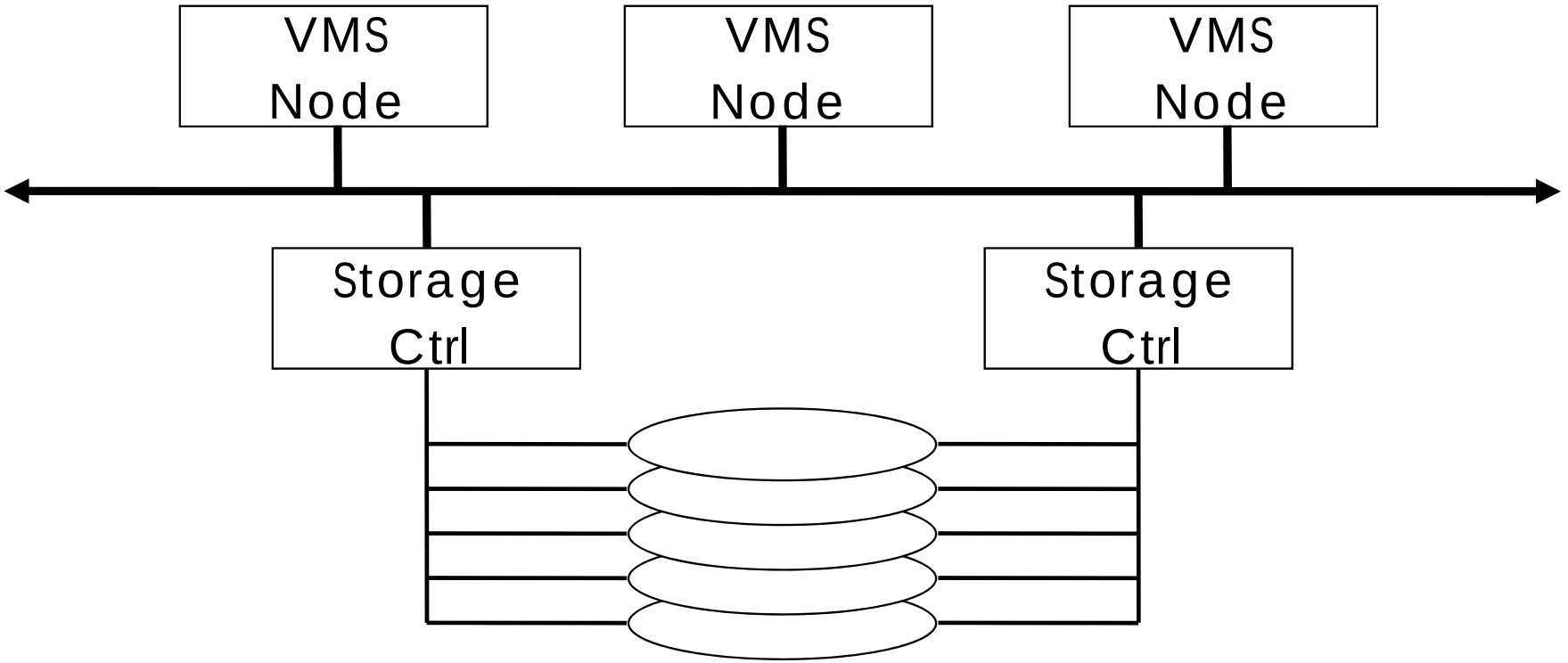
- MA-780 Shared Memory
  - Shared memory global sections and mailboxes
- VAX-11/782
- VAX-11/785
- VAX 8600, etc.

VMSClusters

Not like this!

VMSClusters

VMS Becomes a Distributed Operating System



SCS – Cluster Communications Architecture

Designed for high performance, low latency

- Fully connected LAN
- Virtual circuit with guaranteed response
- Direct memory block transfer
- Failures detected with timeout or “last gasp” datagram

Cluster Configuration

Fully automatic with no permanent master

- SCS connections are formed to all visible nodes
- Prospective member announces connectivity
- Coordinator node proposes new membership
- Other nodes verify and either accept or reject

Clusters: The Lock Manager

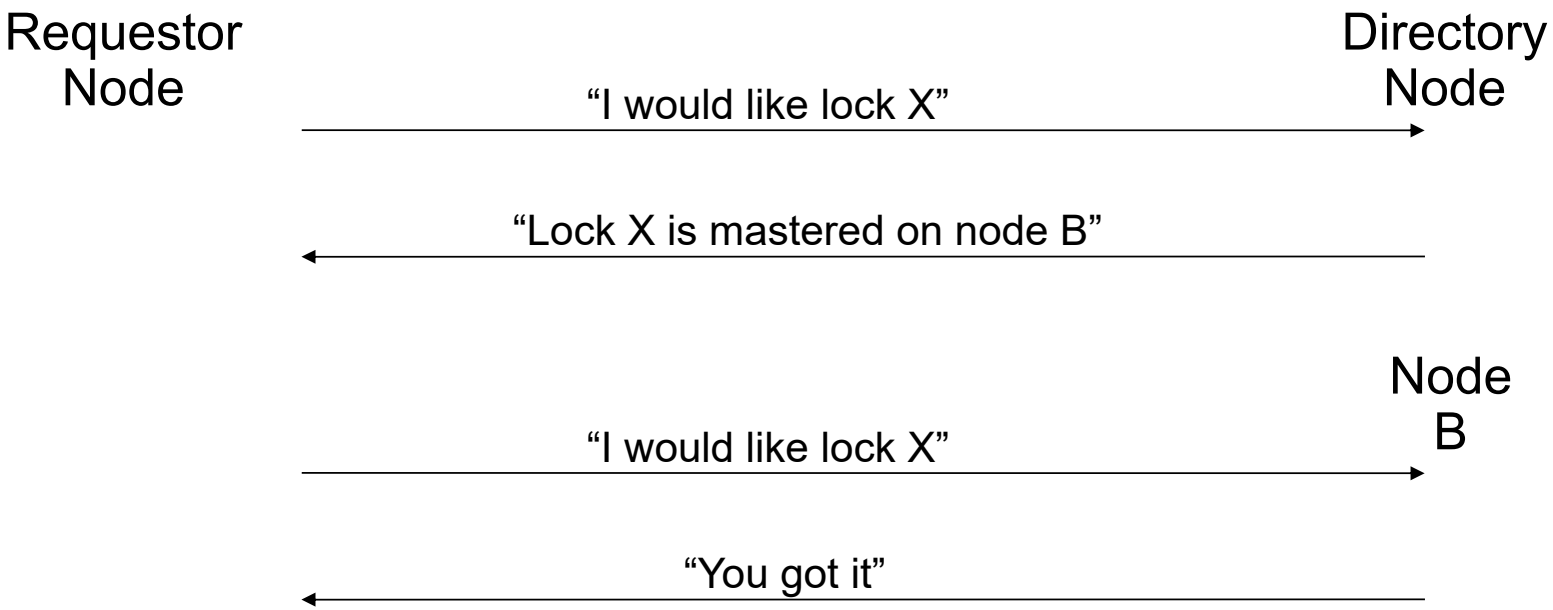
- Abstract named resources
- Lock modes to represent typical data access:
  - EX
  - PW
  - PR
  - CW
  - CR
  - NL

Clusters: The Lock Manager

Fully distributed implementation with no permanent master

- Distributed directory identifies master for a resource
- Lock ownership recorded by master and lock holders
- Master is the node with the most activity
- Automatic reconfiguration on node failure

Clusters: The Lock Manager



RMS and the Lock Manager

RMS Features

- Record-oriented I/O package
  - Sequential, direct, indexed
- Coherent shared write access with record locking
- Process local buffers with coherent cache management

Private locking implementation replaced with cluster lock manager

Before Clusters: File ACP

Server process intercepts complex file operations

- Open file context in system pool
- File metadata cache in process context
- Single thread operation provided implicit synchronization

Clusters: the File XQP

- Cluster implementation choices
  - Single server with failover
  - Multiple coordinated ACPs
- Server process converted to run in client process context
  - Cache moved to system pool
  - Simple threading package layered on AST mechanism
  - Explicit synchronization with lock manager

VAXes Get Bigger: Symmetric Multiprocessing

Original kernel synchronization designed for uniprocessor:

- IPL 24-31: clock, cpu errors
- IPL 16-23: I/O interrupts
- IPL 8-11: device driver threads
- IPL 8: scheduling, memory management, kernel-level messages, etc.
- IPL 4: I/O completion processing
- IPL 3: process rescheduling
- IPL 2: AST delivery
- IPL 0: process execution

Symmetric Multiprocessing

Implicit IPL synchronization replaced with explicit spinlocks

- Each IPL becomes a spinlock
- IPL 8 broken into functional areas
  - Memory Management
  - Scheduling
  - Cluster communications
  - File system
  - etc.
- Locking refined in subsequent releases

SMP Conversion

Brute force effort

- Entire kernel inspected for synchronization
- Aided by existing macros (DSBINT, ENBINT, SETIPL)
- Counters converted to interlocked instructions
- Spinlock rank design detects design deadlocks
- Debug and production locking macros

The VAX isn't big enough

- 32 bit address space vs application and real memory size
- VAX performance vs RISC machines
  - Instruction bit efficiency vs large instruction caches
  - Instruction decoding and pipelining
  - Firmware vs “direct wired” implementation

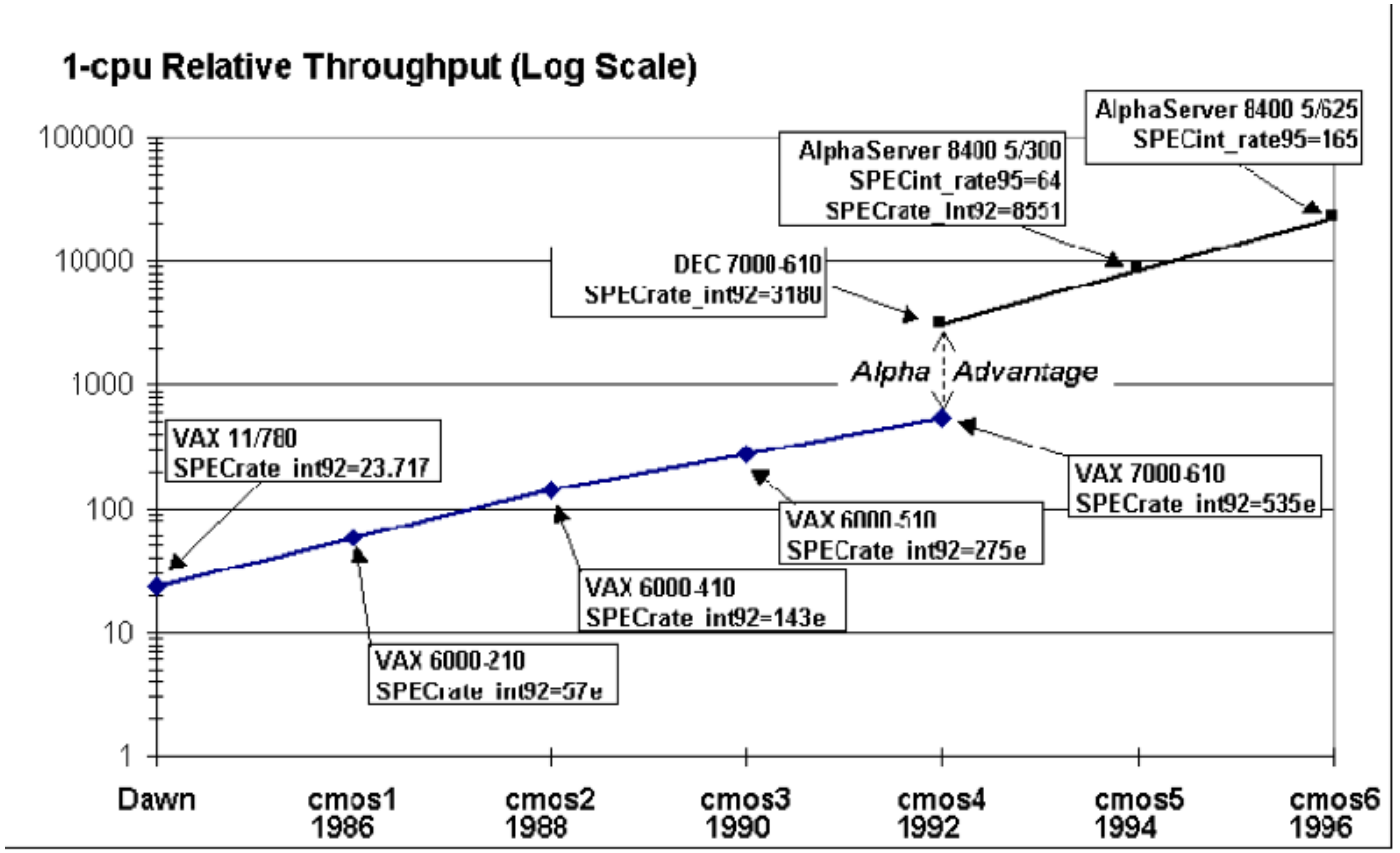
The address space... again

- VAX and VMS 32-bit addressing capability…
- Q: If VAX 32-bit addressing equates to 20 minutes of TV, what size multimedia can 64-bit manage?

The address space... again

- A: Every TV Show Ever Shown Since 1948!

The RISC Advantage



Port to Alpha

VMS and VAX were made for each other

- Privileged architecture (memory management, access modes, IPLs, etc.)
- Variable length CISC instructions, 32 bit architecture
- Most of VMS kernel in macro

Port to Alpha

Alpha is

- 64 bit architecture
- Fixed length RISC instruction

But…

- VAX-like privileged architecture
- Compatible datatypes

New calling standard

- Register arguments
- New stack & function descriptor format

Port to Alpha

Rewrite:

- CPU support
- Boot code
- Some drivers
- Low level memory management
- Exception handling
- Math RTL

Keep:

- All major interfaces

Port to Alpha

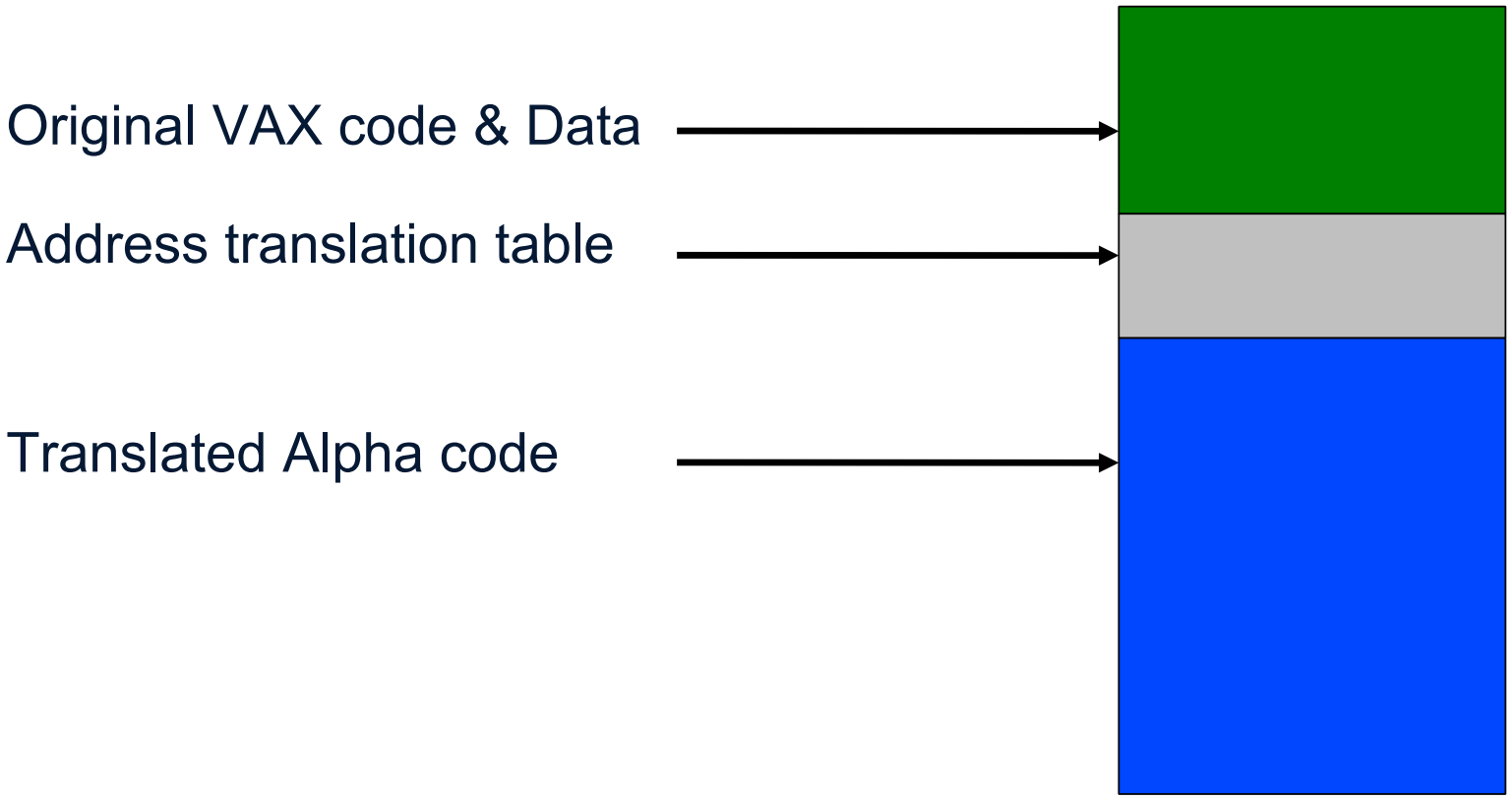
Compile everything else:

- Bliss & C
- Macro!
  - 32 bit vs 64 bit
  - Compilable macro
  - Atomicity issues
- Executable images!!

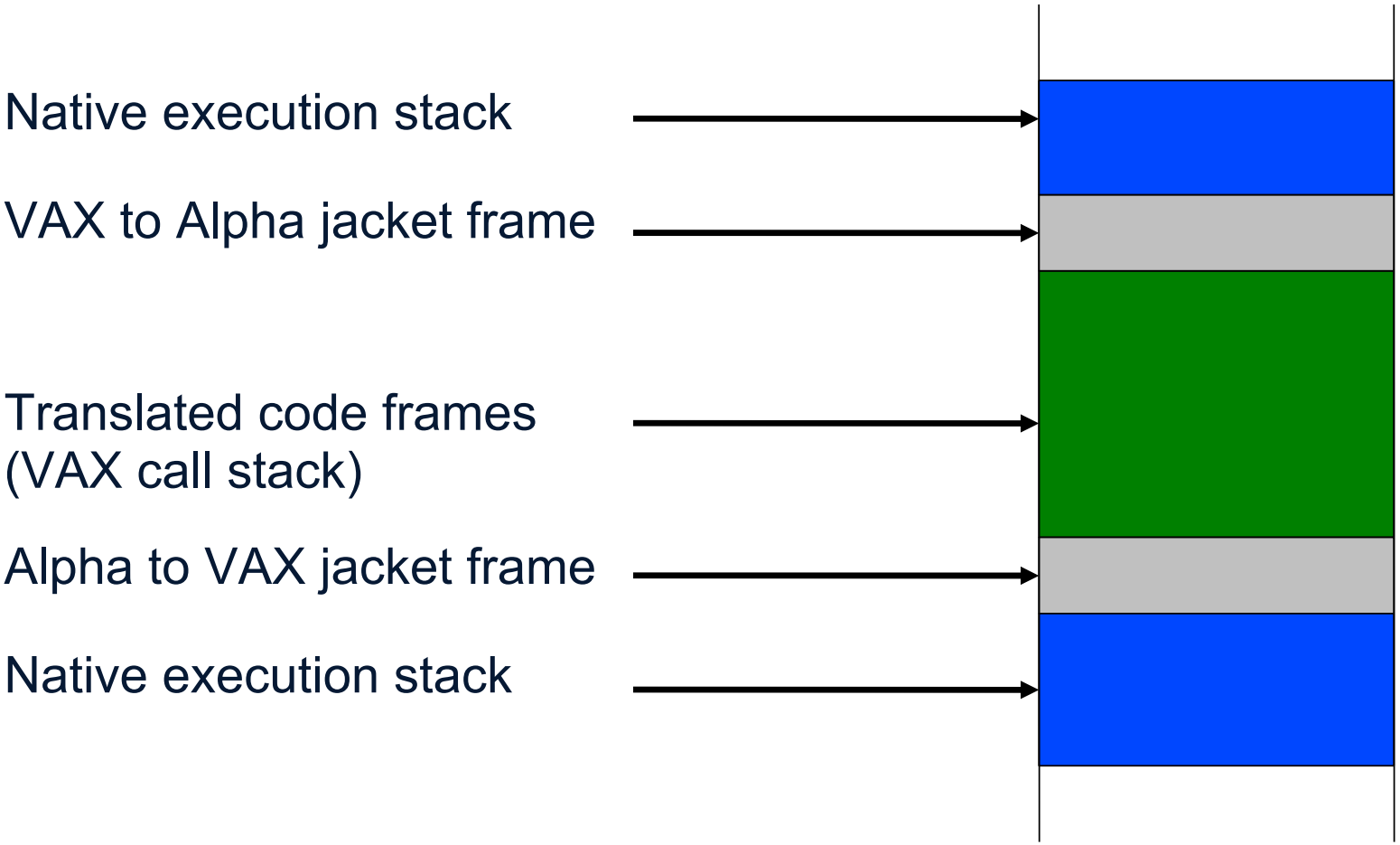
Result:

"It's really VMS. It even has the same bugs."
- early Alpha user

Binary Translation

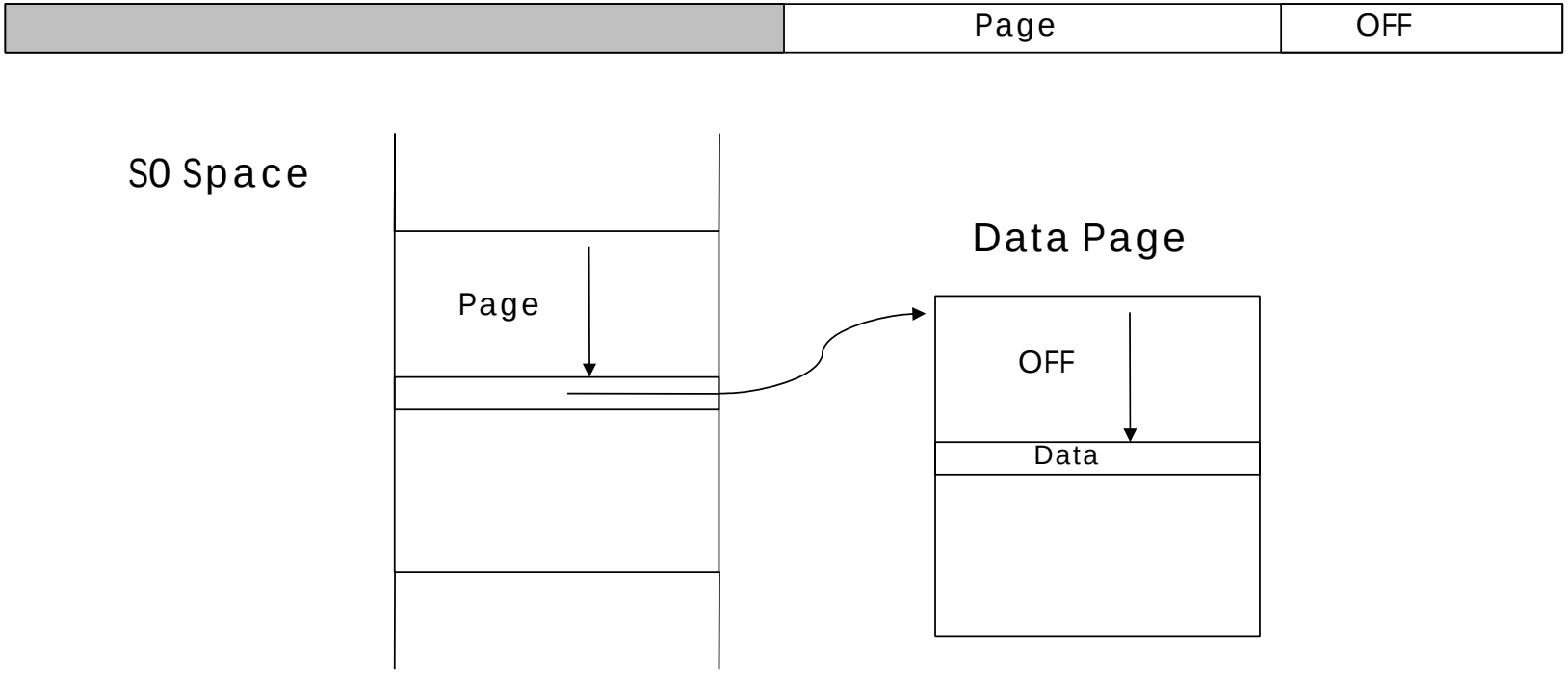


Translated Code Execution



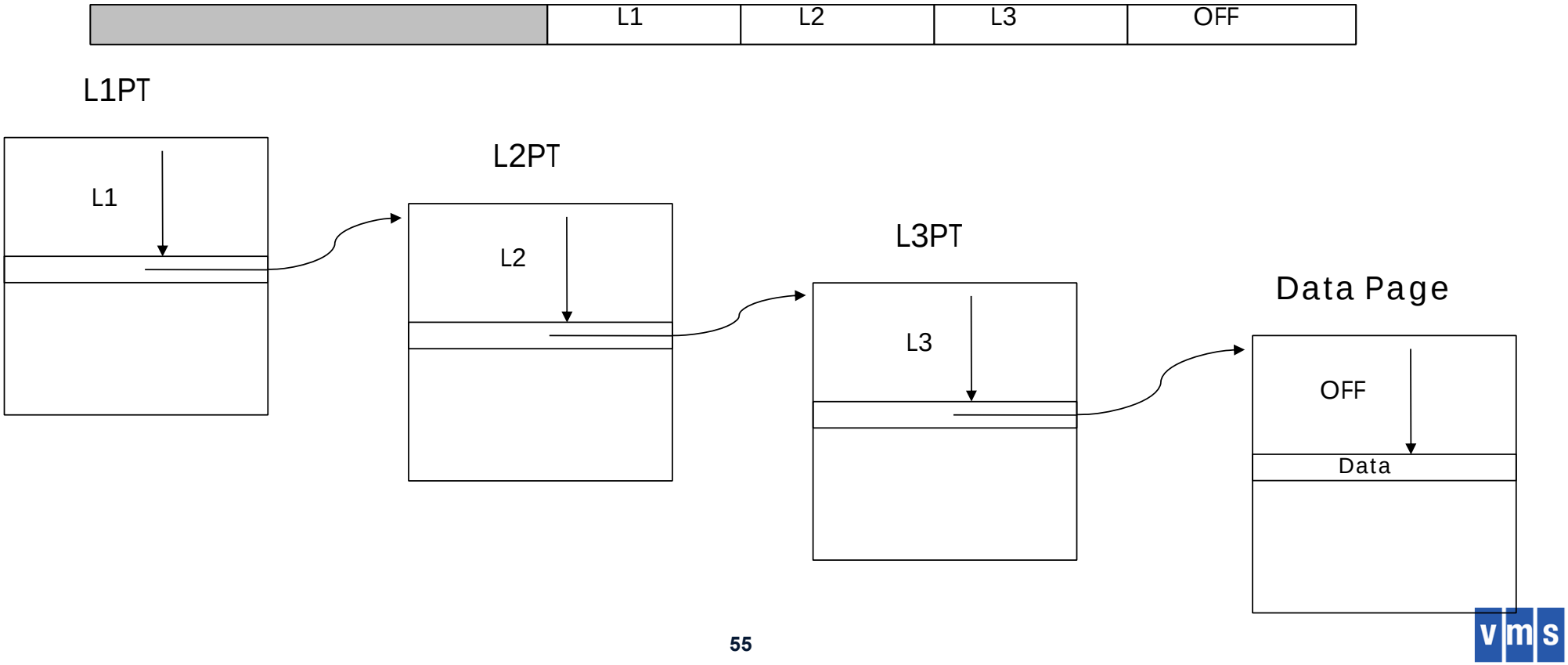
64 Bit Virtual Memory

- Original page table design



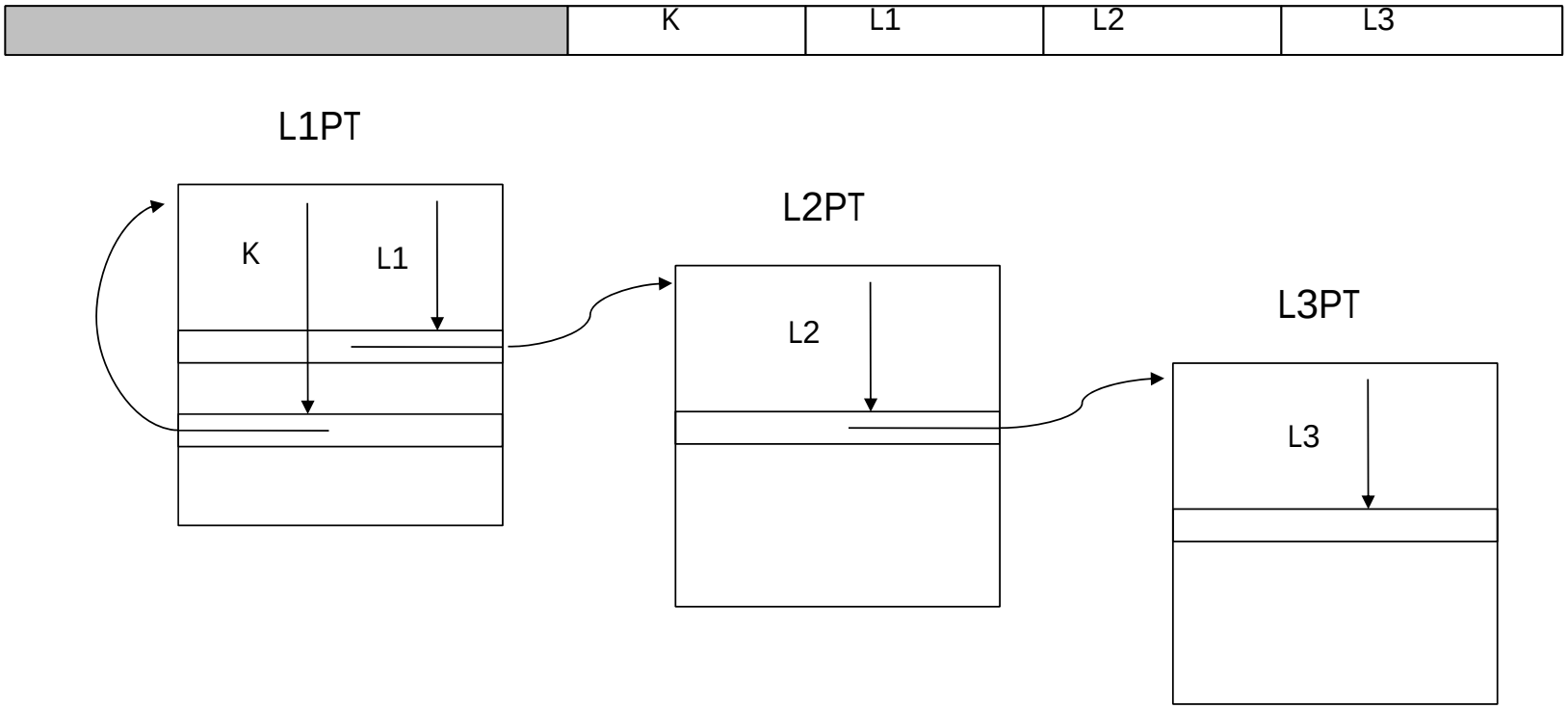
64 Bit Virtual Memory in VMS V7.0

- Extended virtual addressing

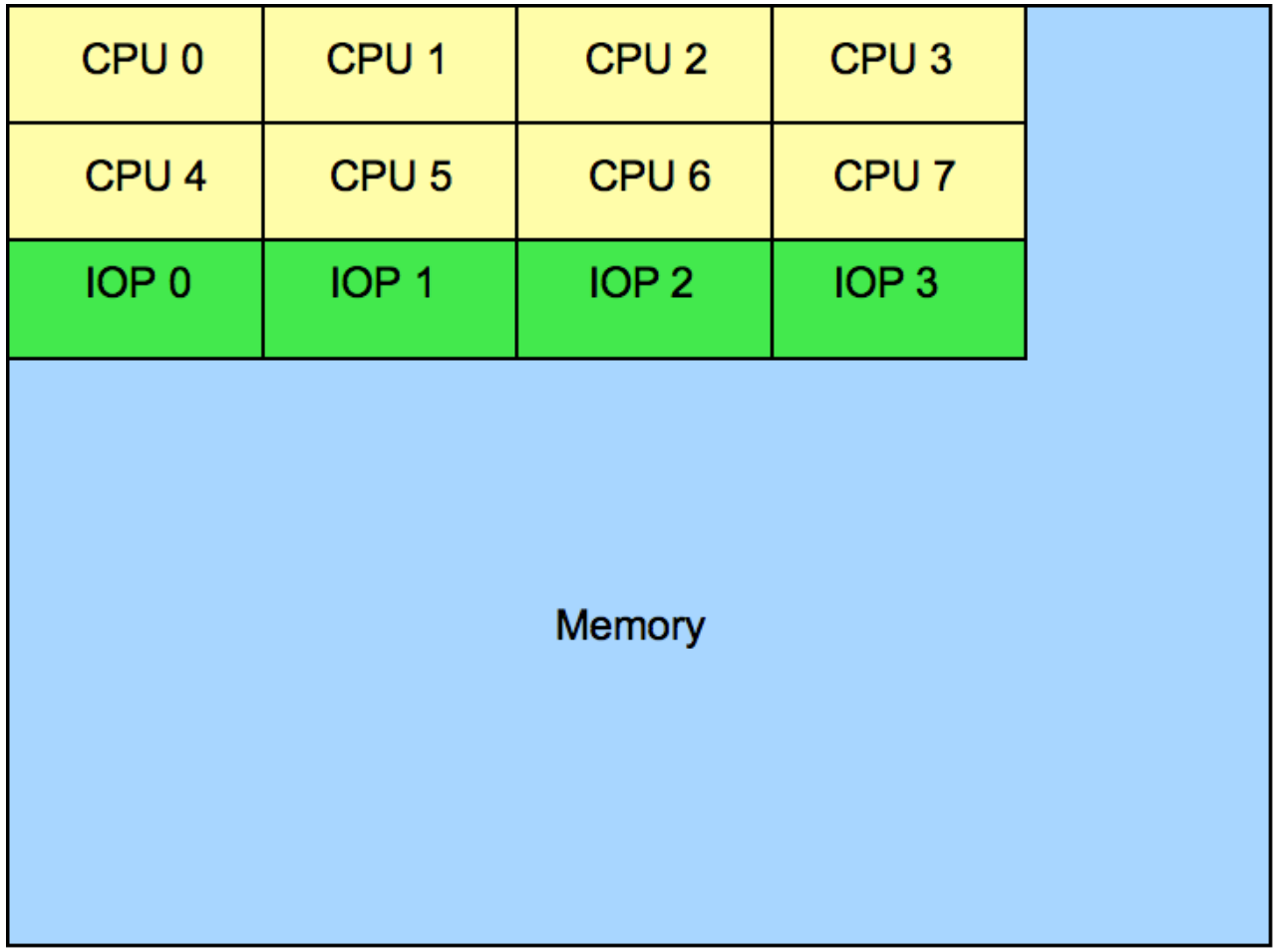


64 Bit Virtual Memory

- Page table reference



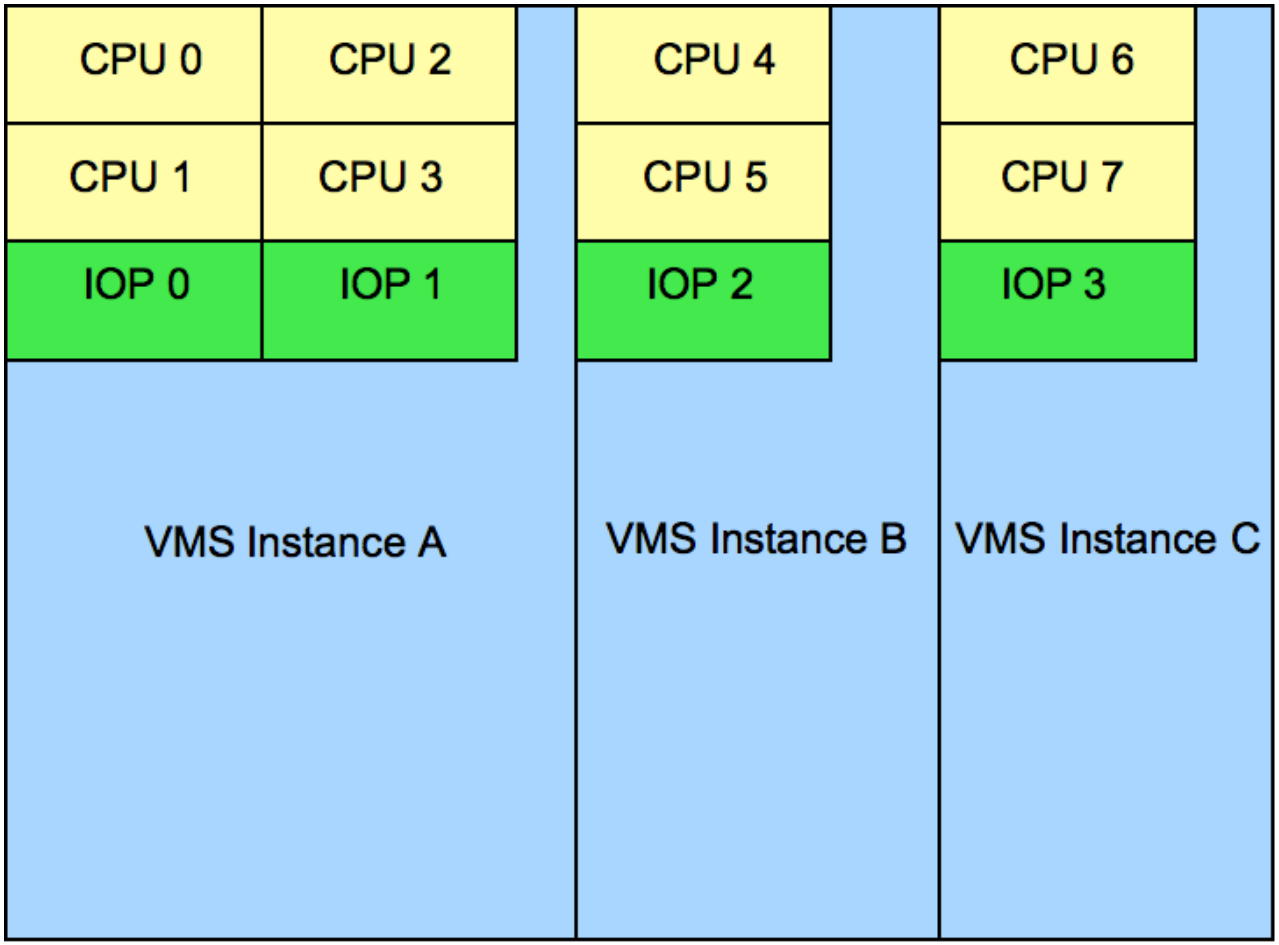
Alphas Get Even Bigger



AlphaServer GS1280

- 64 CPUs
- 64GB memory
- Pushing the limits of SMP scaling

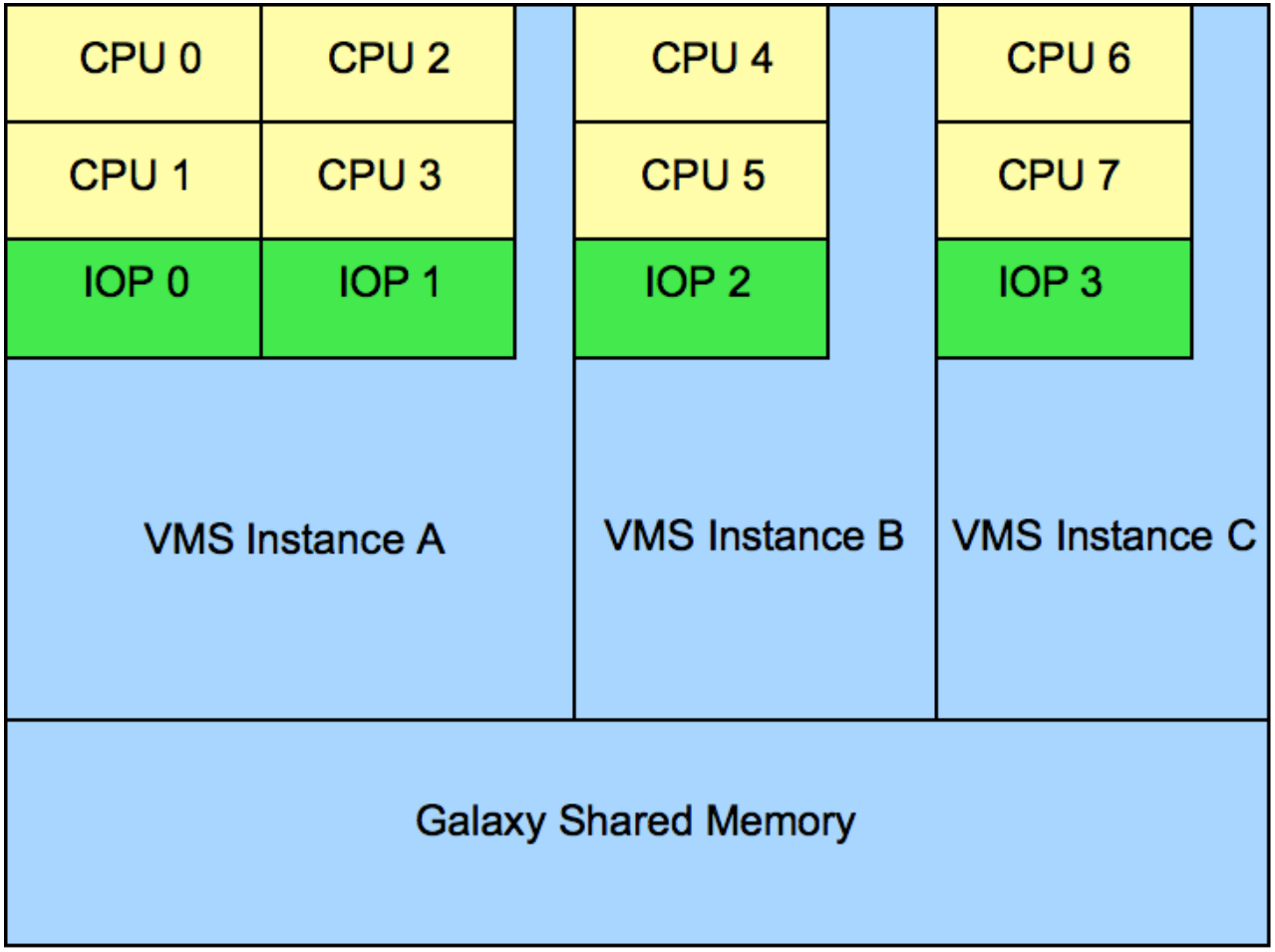
Hard Partitioning



Partitioning by hardware console

- Only allocated resources are visible to each instance

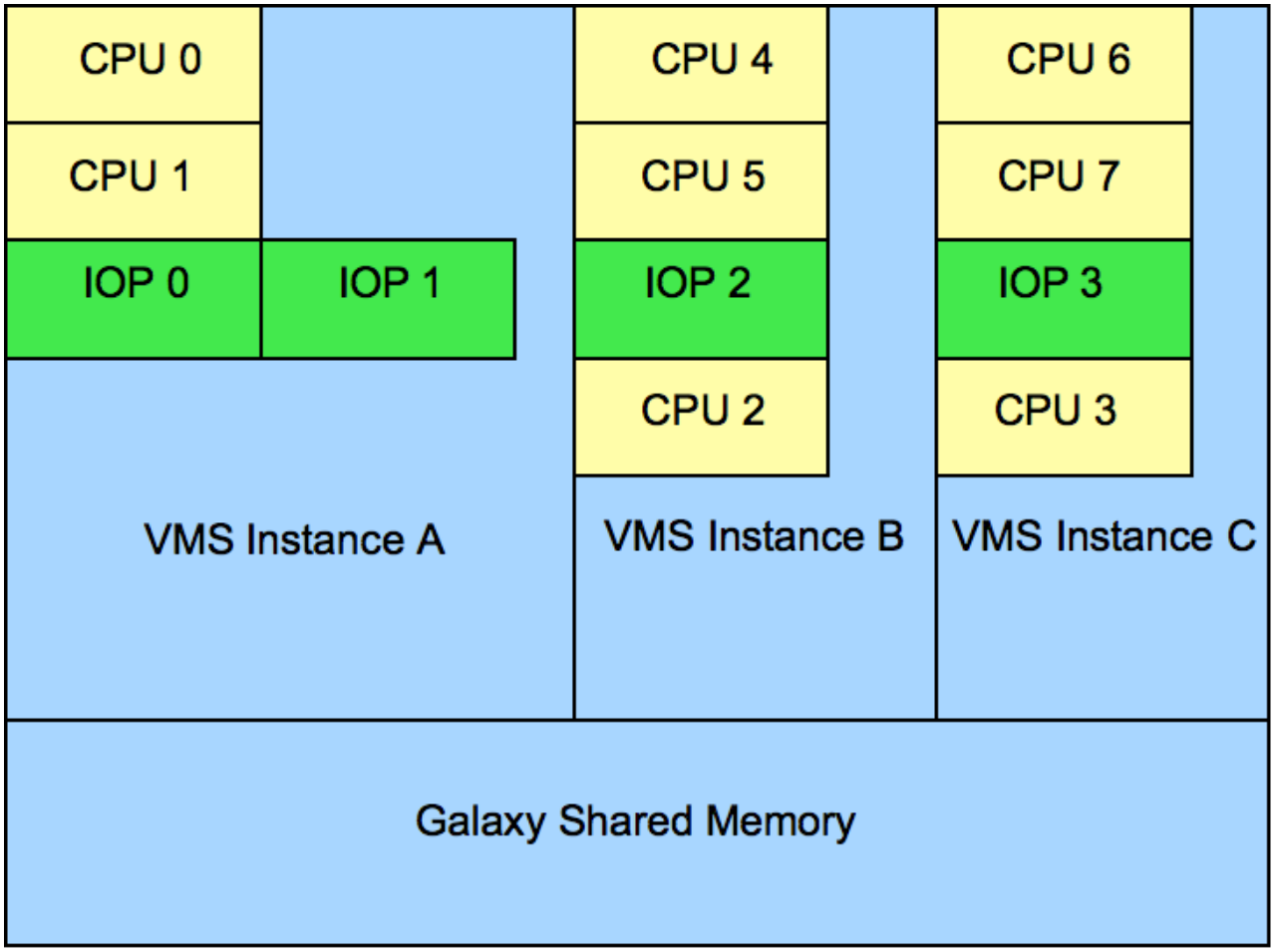
Soft Partitioning – OpenVMS Galaxy



VMS instances cooperate to partition the hardware

- CPU and IOP assignment
- Memory allocation
- Shared memory

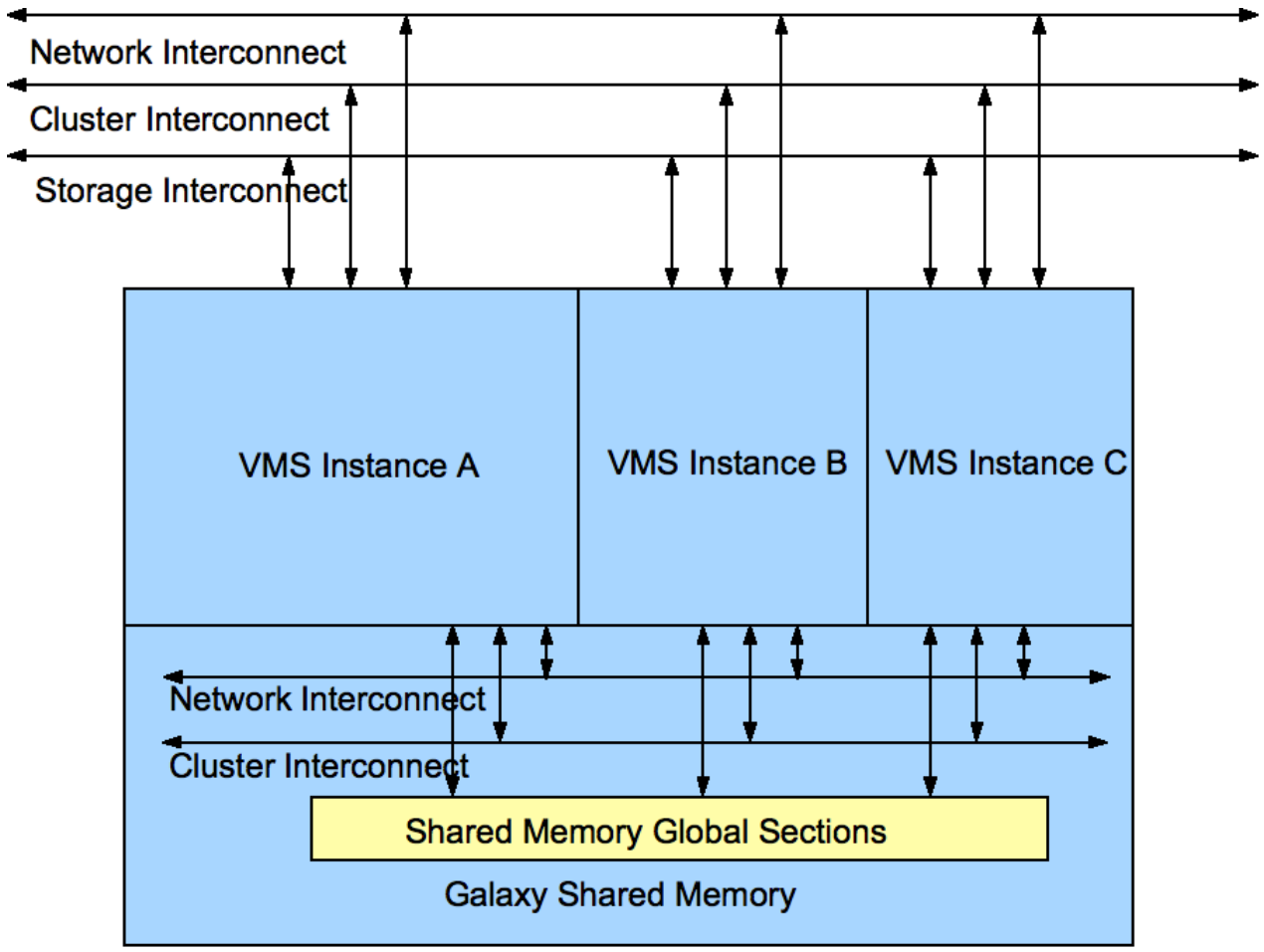
Soft Partitioning – OpenVMS Galaxy



VMS instances cooperate to partition the hardware

- Resources can be reallocated

Galaxy Cluster Architecture



Shared memory is the fastest interconnect

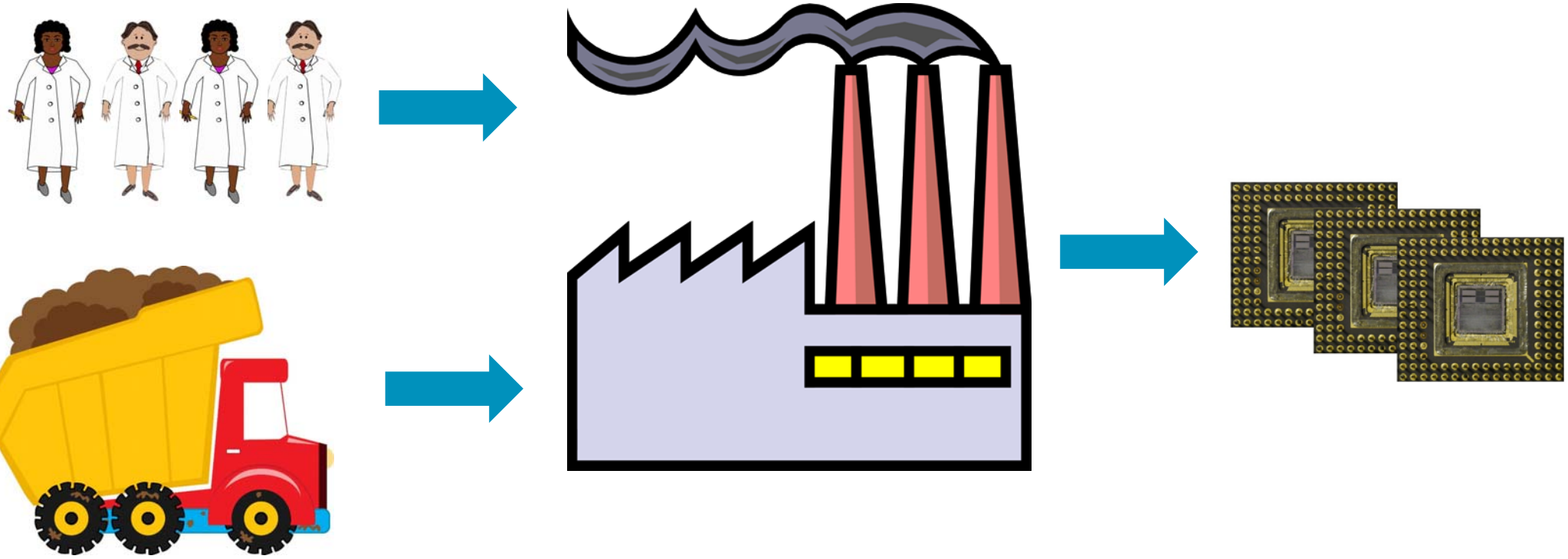
The Itanium Architecture

The next generation beyond RISC

- Explicit parallel execution
- Many more registers

Driven by chip development economics as much as technical factors

Chip Development Economics



Port to Itanium

- Another 64-bit architecture, but...
- Different register conventions
- Intel calling standard
- Different privileged architecture
  - No PALcode
  - Different console / boot procedure
  - Different interrupt architecture
  - Different synchronization primitives

Port to Itanium

- Fortunately...
- 4 access modes
- Compatible memory protection features
- Memory atomicity no worse than Alpha

Port to Itanium

- Rewrite
  - CPU support
  - Boot code
- New
  - Interrupt & exception delivery in software
  - Emulation of interlocked instructions (queues, etc.)
  - EFI partition on system disk
- Redesign
  - Calling standard and condition handling
  - Object and executable file format

Port to Itanium

- Recompile
- 95% of base OS code recompiled without change
  - Biggest problem was “IF ALPHA” conditionals
- Binary translator also available
  - Even supports VAX translated images!

Distributed Client-Server Computing

- Oh... VMS did that 35 years ago
- Keeping up with current technologies and tools (over time)
  - TCPIP
  - OSF DCE
  - Microsoft DCOM
  - Apache web server
  - OpenSSL, LDAP, Kerberos
  - ...

How to Build an Evolvable System

It begins at the beginning

- Start with a team of grownups
- Design with care
- Keep the team small
  - Initial VMS architecture came from 3 people
  - Entire VMS V1 team was 24 people
- Keep the pressure up
  - The first known “fact” about VMS was the schedule
  - Beware of creeping elegance

How to Build an Evolvable System

- Modularity
- Modularity
- Modularity

Modularity in VMS

- Dynamically loaded modules for all configuration dependent components
- Huge number of system models and devices supported over the life of the system
- Any VMS system disk will boot on any configuration of a particular architecture
- New hardware is supported with minimal effect on the rest of the system

Maintain Design Integrity

Causes of “software rot”

- Lack of design understanding
- Quick and dirty changes
- Changes that compromise the original design
- Functional extension without extending the original design
- Duplication of function
- Runaway complexity

Maintain Design Integrity

- Document the design
- Well defined stable interfaces
- “Firewall” major modules – validate inputs
- Clean house – rewrite “worn out” components

Ready For the Next Adventure…

- X86 port
- New file system
- Ongoing modernization
- …

Thank You

To learn more please contact us:

vmssoftware.com
info@vmssoftware.com
+1.978.451.0110

Divider with highlight…

Section header subtitle Arial 28pt