```
+---------------------------+
!   !   !   !   !   !   !   !
! d ! i ! g ! i ! t ! a ! l !
!   !   !   !   !   !   !   !
+---------------------------+
```

INTEROFFICE MEMORANDUM

FROM: Leslie Klein
DATE: July 1, 1988
DEPT: Technical Languages and Environments
LOC: ZK2-3/N30 EXT: 381-2055
NODE: TLE::KLEIN

TO: Ken Hobday
Al Simons
Rich Grove
Dick Wilson
Jeff Rudy
Tom Harris
Scott Davis
Dave Cutler
Celeste LaRock
Bob Travis
Jim Totton
Don MacLaren
Benn Schreiber
John Gilbert
Bev Schultz
Evan Suits
Chip Nylander
Bill Keating
Dena Yancey
Bill James

SUBJECT: June 30, 1988 Woods Meeting - Results and Action Items

Thank you for your participation in the June 30 woods meeting. The following is an overview of results and of the action items. If you own an action item, please let me know whether the due date is reasonable or not and feel free to propose a different one. You'll get a copy of the slides via internal mail within the next couple of weeks.

1 SUMMARY OF DECISIONS

We reached the following global decisions:

1. We will support Dave Cutler's Dual Environment proposal for the OSF operating system development strategy.

2. The GEM compiler strategy as presented by Rich Grove was ratified, with reservations around the priority to be attached to VAX GEM work (an action item on this appears below).

3. We will work from the strength of the PRISM software architectural work done to date, fixing the key pieces that were machine dependent or that need reevaluation in light of OSF (e.g. calling standard, debugger, ...).

2 GOALS

I heard some good program goals expressed. This may not be the complete list, but I'd like to capture them here:

1. We must sell in heterogeneous environments (mixed vendors and mixed VAX/RISC-of-the-month/other-Digital-platforms).

2. We need to make our software more portable and set concise portability goals.

3. We would still like people to buy our (Digital's) hardware.

4. We need to identify our points of control -- the features of our software offering that show our added value and that make our hardware the hardware of choice. (e.g. don't go for "lowest common denominator" in the name of portability -- OK and probably desirable to have extra capabilities on strategic platforms)

After the meeting, Chip Nylander gave me a sythesis of the strategies we're going after that I believe is very helpful:

1. We want to develop a software strategy that is responsive to the hardware requirements of the corporation as they evolve and change, and which insulates us to some degree from the shifting tides of hardware strategy.

2. We want to consciously position our interfaces and architectures into two "boxes" or "toolkits" -- the "universally portable", and the "proprietary but portable as sensible and required to strategic DEC platforms", and then concsciously position each product against one or the other of these toolkits.

3. We are free to migrate things from the "portable but DEC proprietary" toolkit to the "universal" toolkit over time as we decide to drive standards.

4. The "DEC proprietary" toolkit is one of the basis for added value, and one of the basis for being able to deploy our layered products on all the platforms that are or become important to the corporation.

5. We will have to have some control of the underlying operating environment (operating system) where the "DEC toolkit" is provided, in order to implement such capabilities as structured condition handling and record management with reliability and integrity.

3 ACTION ITEMS LIST

- o OWNERS: Don MacLaren, Chip Nylander, Calling Standard Team

  DELIVERABLE: Position paper

  DUE DATE: August 1, 1988

  ISSUE: What does it take to make the calling standard machine independent (if possible) and what are the tradeoffs? What would we alter in the current MICA/PRISM calling standard in order to achieve or partially achieve this?

- o OWNER: Jim Totton

  DELIVERABLE: Position paper

  DUE DATE: July 12, 1988

  ISSUE: What does IEEE FP mean for a MICA server in a VMS environment?

- o OWNER: Bill Keating

  DELIVERABLE: Announcement

  DUE DATE: July 15, 1988

  ACTION: Identify someone in SDT to own gathering OSF information and requirements.

- o OWNER: Scott Davis

  DELIVERABLE: Position Paper

  DUE DATE: August 1, 1988

  ACTION: Scott is the SDT driver relative to our higher level architectures and their positioning relative to OSF (calling standard, ARUS, RPC, ...) and relative to external standards (PCTE, POSIX, ...).

- o OWNERS: Bill Keating and Scott Davis

  DELIVERABLE: Announcement

  DUE DATE: July 15, 1988

  ACTION: Identify someone in SDT to own APA (Applications Portability Architecture).

o **OWNER:** Leslie Klein

  DELIVERABLE: Position paper

  DUE DATE: July 15, 1988

  ACTION: Peruse agreement with that new RISC third party relative to support for their making "VAX compatible languages". Decide what level and type of participation SDT should have here.

o **OWNERS:** Leslie Klein and Chip Nylander

  DELIVERABLE: Position paper

  DUE DATE: August 1, 1988

  ACTION: Determine priority and timing of VAX GEM -- esp. with reference to importance of vectorization to VAX C (Liz Freburger will help here).

o **OWNERS:** Jim Totton and Leslie Klein

  DELIVERABLE: Proposal

  DUE DATE: July 22, 1988

  ACTION: Develop a proposal back to NAC describing relationship of DDL to RPC with respect to stub compilers.

o **OWNER:** Jim Totton

  DELIVERABLE: Proposal

  DUE DATE: July 22, 1988

  ISSUE: Should VAX ULTRIX DECwindows components be retargeted to PMAX? (Needs to "penetrate the fog" around PMAX OS, incl. big-endian versus little-endian.)

o **OWNER:** Bill Keating

  DELIVERABLE: Written status statement

  ACTION: Needs to get the top-level strategic decisions on VAX ULTRIX futures -- esp. as to which members of the VAX hardware family will be supported, vector support, MP support.

o **OWNER:** Ken Hobday

  DELIVERABLE: Position paper

  DUE DATE: July 22, 1988

ISSUE: Do we want to drive our model of condition handling into OSF?

- OWNER: Chip Nylander

  DELIVERABLE: Position paper

  DUE DATE: July 15, 1988

  ACTION: Document why we need GEM compilers for that third party RISC vendor's machine. Also needs to reword the "straw horse" proposal from his presentation to match Dave Cutler's Dual Environments model.

- OWNERS: Al Simons, Ken Hobday, Jim Totton, Chip Nylander

  DELIVERABLE: Position paper

  DUE DATE: August 8, 1988

  ISSUE: ARUS - can we make it meet Keating's requirements/expectations?

- OWNER: Chip Nylander

  DUE DATE: ASAP

  ACTION: Get that third party RISC vendor's SRM and any other available materials (XOPEN standard, ...) that will help us in doing our new architectural work and compiler back end retargetting.

- OWNER: Leslie Klein

  DUE DATE: August 10, 1988

  DELIVERABLE: Position paper

  ISSUE: Who is the group that is going to support the MIPS/UNIX language products?

- OWNER: Leslie Klein (temporarily!)

  DELIVERABLE: Proposal

  DUE DATE: August 8, 1988

  ACTION: Define portability tactics (operational plans)

- OWNER: SDT

  DELIVERABLE: New Phase 0/1 plans for Dual Environment approach

ACTION: Follow Dave Cutler's "Get project back on track" recommendations--

- GET agreement on OSF implementation and product strategy and its relationship to the jelly bean UN*X.

- Assess current status and changes required by MIPS architecture and decide upon strategy.

- Define hardware platforms, OSF product, and layered product deliverables.

- Revise plans and set new schedules

AGENDA - JUNE 30, 1988

8:30-8:50 - Bill Keating

Overview, goal-setting

8:50-9:10 - Chip Nylander

CALLING STANDARD: How does this map to MIPS? Is this what we want on MIPS/MICA and MIPS/UNIX?

9:10-9:25 - Al Simons

RPC DIRECTIONS: How does this map to MIPS? to OSF? What is current status of VAX/VMS RPC? Effects relative to portability?

9:25-9:35 - Rich Grove

RISC-y VAX: What is it? What might it mean to us?

9:35-10:00 - Jim Totton

RTL DIRECTIONS: (esp. math library and ARUS) What do we do around floating point support? ARUS - completely transportable target - do we want to from day 0 constrain LP's to use ARUS interfaces? What would that mean and what would it accomplish? Effects of MIPS on ARUS directions?

10:00-10:15- BREAK

10:15-11:00- Scott Davis

AIA DIRECTIONS: the portability dimension. How does AIA help our software be more easily retargetted. What specific requirements does this put on AIA?

11:00-11:30- Liz Freburger/Dick Wilson

REWRITING OUR SOFTWARE IN C: Approaches they think are viable -- results of their contractor research to date. Benefits? Cost?

11:30-12:00- Rich Grove

COMPILER DIRECTIONS: Are our strategies/tactics still the right ones? How do they map over to MIPS? Should we do compilers for MIPS? Should VAX GEM have highest priority?

12:00-1:00 - LUNCH

1:00-1:30 - Liz Freburger

VAX ULTRIX: What is going to happen here and to what level should we participate?

1:30-3:00 - Chip Nylander

MIPS: how much do we put there when and how do we provide business justification for doing so?

PMAX/ULTRIX - do we want to step up to getting base DECwindows tookit/products here? When? Do we buy into the "Armando Stettner" system (PMAX/ULTRIX...) or concentrate on the "Dave Cutler" system (MIPSco/VMS+OSF)? Do we spearhead a "Common Software Architecture" for MIPSco or do we just eat the impact of the lack of such an architecture at the language and run-time level? Or do we confine ourselves to one system?

3:00-3:15 - BREAK

3:15-4:30 - Leslie Klein

PORTABILITY ISSUES: What is portability? How do we get a good multitarget development environment, and what is it?

Based on what we've discussed today, what should our goals for portable software be? Which targets? Do we do rewrites in C and to what extent? Do we redirect effort to getting AIA/ARUS/etc. base components done for more than one target now rather than later?

4:30-5:00 - Bill Keating

WRAPUP AND ACTION ITEMS

5:00 - ADJOURN

MEETING

SDT Technical Direction

- Get Act Together After Change.
- Salvage The PRISM Software & Architecture Plan To Degree Possible.
- Move To An Aggressive Plan.
- Drive Plan Within DEC.

Bill K - 1 of 5

__TODAY__

Review Major Technical/Architectural Components.

Move To More General Areas.

__Goals__

- Recapture Our Vision And Reshape It To New Environment.

- Identify Fundamental Agreements. Capture These Formally.

- Identify Hardspots / Issues. Plan To Solve These During July.

In General, Have Our Act Together By Mid-August.

Get Any Corporate Issues Identified.

Bill K. - 2 of 5

RULES:

No Time on Worrying The Past -
Only If It Helps Understand Our Future Direction.

Avoid Rat Holes.

Seek General Theme.
Help Pieces And Decisions To Fit.

Bill K -3 of 5

__My Themes Are__:

Leadership

Architecture

Main Areas: CASE, Core Applications

Implementations - (On DEC Hardware first)
(Key Targets, Priorities)

Our Products Must Concentrate On:

Capability	- (To Qualify Overall Leadership Candidate)
Integration	- (To Insure System Offering Is Indeed Leadership)
Portability	- (Position To Move Into SW Area)

I would like our products to be isolated (to reasonable degree) from HW and basic OS services.

Counting on MIPS here.

What This Means To Me Going Into This Meeting:

- Move all products to a generally available HW independent language.
- Achieve Generic RTL (and File Access) layer which can work on the interesting OS's.
- Isolate OS dependent capabilities. (I.e., Multi-Thread Architecture).
- Follow through on our FE / BE Language Strategy.
- ?? Understand Distribution of Stuff To "Less - Than - WorkStations" ??
- ?? Enlist Partners For Non-DEC HW Platforms??

Bill K-585

What is Software Architecture?

Interfaces and conventions agreed upon and utilized across multiple software products to produce a system with qualities required for market leadership and success.

Some software architecture can be implemented on multiple hardware platforms (rehosted) in order to make software products more portable. For example,

- Utility run–time interfaces
- Record management services
- Remote procedure call services
- Common Multithread Architecture
- X–windows
- DDIF

Some software architecture is not rehostable across hardware platforms. This is generally language implementation software architecture, including

- Object language
- Calling standard
- Language run–time interfaces

Note that most of this software architecture is NOT DIRECTLY VISIBLE to applications. Applications see the benefits.

Benefits of Good Software Architecture

- Consistency of software products
- Integration—multilanguage environment, common run–time environment, tool integration among tools and with languages, application integration, etc.
- Ease of development
- Reliability
- Evolvability
- Longevity
- Performance
- Completeness of solutions

Software Architecture at Digital

PDP-11

Layered product software architecture tended to be ad-hoc and/or accidental. Mostly provided by operating systems, and by emulation of PDP-11 operating systems on other PDP-11 operating systems.

Software Architecture at Digital

VAX

VAX software product architecture was deliberate and thought out.

However, did not have the benefit of experience, so some mistakes were made (VAX Calling Standard descriptor design) and some things took years to get right and stabilize (command definition and parsing interfaces).

This is an excellent software architecture on VAX/VMS, but most of the older architecture is VAX/VMS specific—we would probably not want to use it as the basis for future systems.

Software Architecture at Digital

VAX

In 1988, VAX product software architecture includes:

- Calling Standard
- Condition Handling
- Object Language
- RMS–32
- VMS Librarian, including interactive HELP
- Command definition and parsing (CLD and CLI$)
- Many VMS system services
- Message files, message handling, status codes
- Common Run Time Library
- DECwindows
- Compound Document Architecture
- Common Data Dictionary
- Application Integration Architecture (soon)
- Common Multithread Architecture (soon)
- Remote Procedure Calls (soon)

Software Architecture at Digital

PRISM

PRISM software product architecture was deliberate and thought out, for MULTIPLE OPERATING ENVIRONMENTS.

For the most part, ONLY BUILD LAYERED PRODUCTS ONCE.

Had the benefit of VAX experience, so mistakes were corrected.

Software Architecture at Digital

PRISM

PRISM Common Software Architecture included:

- Calling Standard
- Condition Handling
- Object Language
- Record Management System
- Librarian interfaces, including HELP
- Interactive HELP librarian
- Command definition and parsing
- Message files, message handling, status codes
- Application Runtime Services
- DECwindows
- Common Multithread Architecture
- Compound Document Architecture
- Application Integration Architecture (incrementally)
- Remote Procedure Calls (incrementally)
- Common Data Dictionary (later)

Software Architecture at Digital

MIPS

To recap,

- PDP–11: accidental and ad–hoc layered software architecture
- VAX: deliberate, thought–out software architecture for one operating environment
- PRISM: deliberate, thought–out software architecture for multiple operating environments, reflecting experience with VAX, providing benefits that include consistency of software products, integration, ease of development (only develop once), reliability, evolvability, longevity, performance, completeness of solutions.

MIPS is here. Now what?

Software Architecture at Digital

MIPS

OVERSIMPLIFYING, we will have to choose between investing in design and deployment of layered software architecture that we think is required to provide benefits that we think are important; or else take what's there with probable loss of the benefits of layered software architecture as we understand it.

Also must distinguish between rehostable software architecture (AIA, ARUS, CMA, etc.) and implementation software architecture (calling standard, object language, condition handling, etc).

PROPOSAL:

- Commit to defining and deploying rehostable software architecture as basis for portable SDT products.
- Evaluate need for new implementation software architecture on a platform-by-platform basis.

RISCy VAX

R. Grove
30 June 1988

RISCy VAX Program

An STF task force to find ways to keep VAX and VMS competitive for an extended lifetime

- Modernized VMS software
- Modernized compilers
- Identify preferred RISCy subset of VAX
- Possible incompatible extensions

Task force reports to be completed in a couple of months

Would affect VAX processors AFTER Aquarius and Rigel

RISCy VAX Architecture

First, identify a RISC-like subset of VAX

- MOVL for Load/Store, 3-operand register-to-register
- CPUs execute the RISCy subset fast
- Compilers target the fast RISCy subset
- Old/new programs OK on old/new machines

Second, consider incompatible extensions to gain more performance

- More registers (RISC needs more than VAX)
- Virtual address space extensions
- No longer backward compatible with classical VAX

RISCy VAX Compilers

Compiler modernization

Common code generator (GEM)
Reduce procedure overhead
Tailored code generation and scheduling

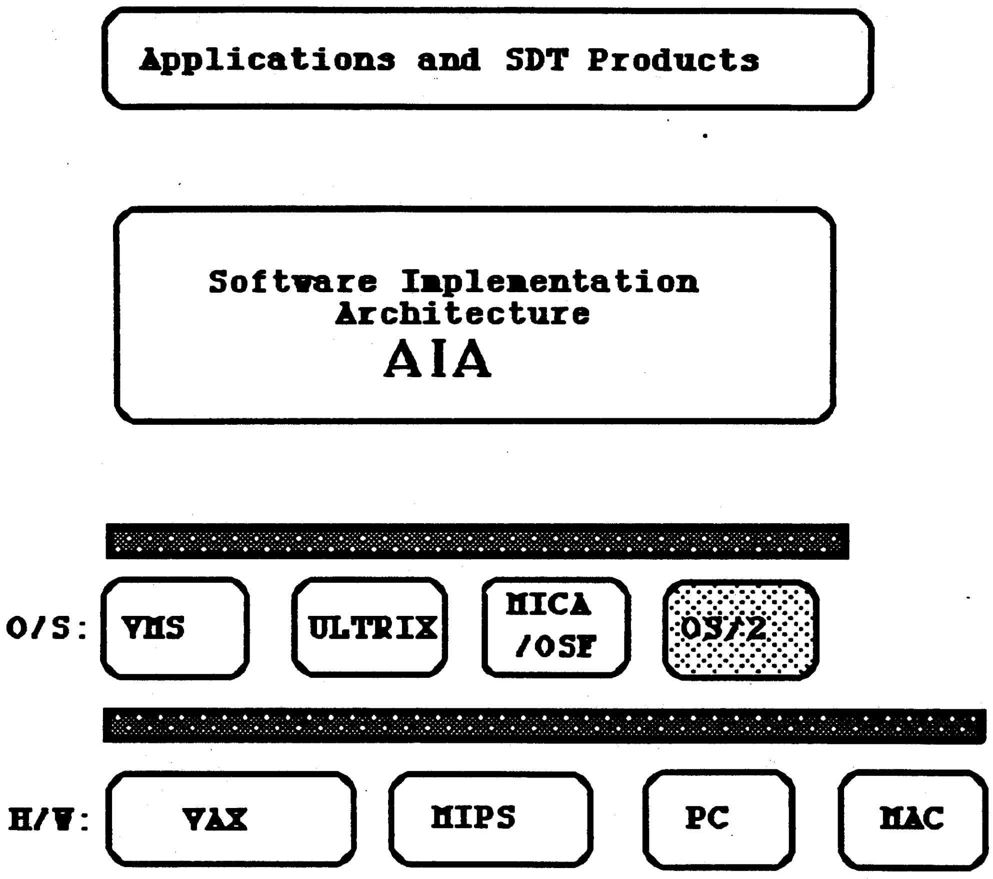
Possible compiler strategies

- Low level of effort: Some mods to a few compilers
FORTRAN, BLISS, VCG
- High level of effort: Use GEM
FORTRAN, C, Ada, Pascal, ...

Comments on RISCy VAX

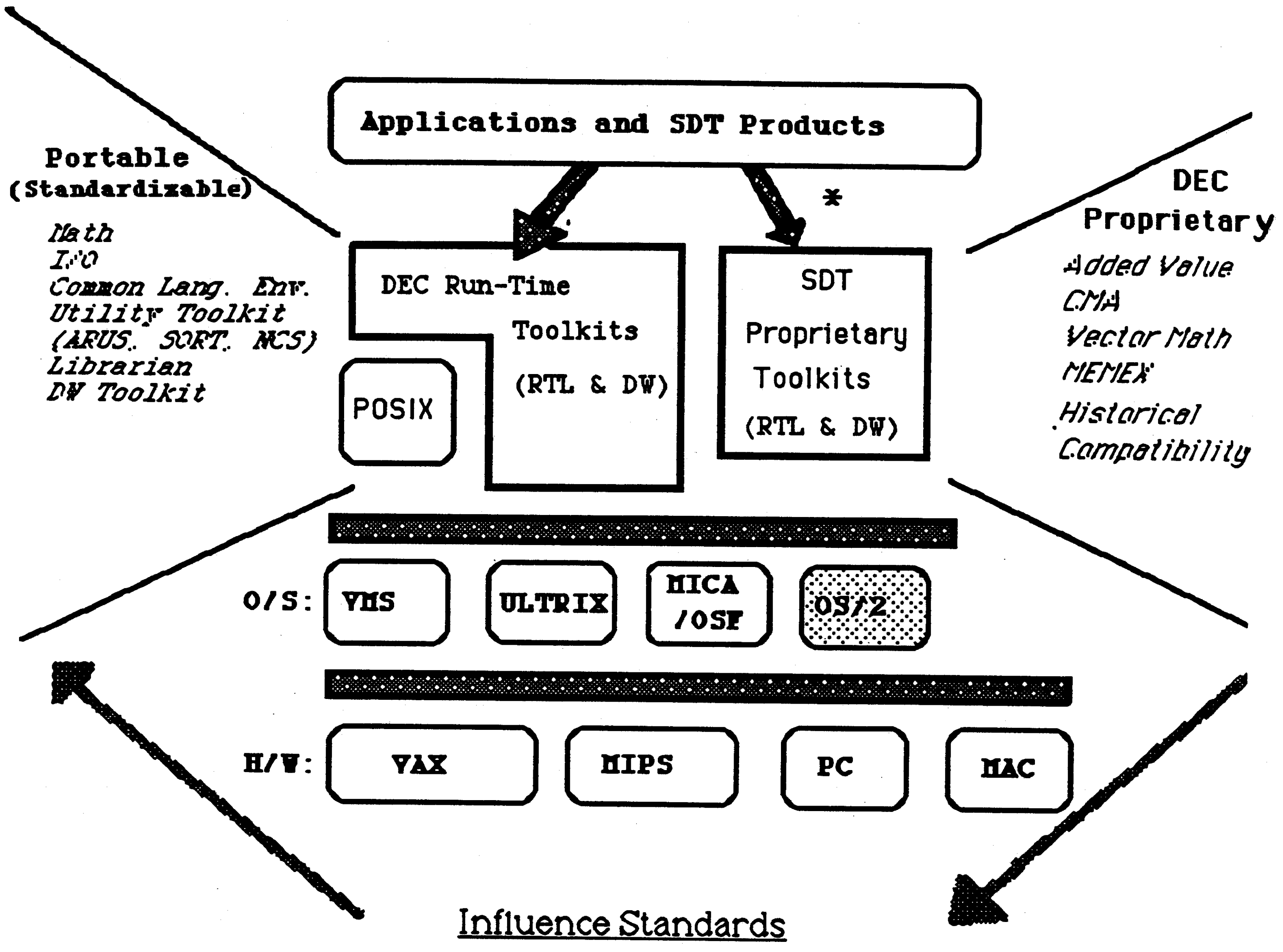
- SDT will need to participate in task force
- A RISCy VAX is neither RISC nor VAX
- RISCy VAX should NOT be the primary focus for our leading edge compiler technology. Real RISC machines will be the performance leaders, and that's where we need to compete and win.

A Model For Portability


Applications and SDT Products
Software Implementation Architecture
AIA
O/S:
VMS
ULTRIX
MICA /OSF
OS/2
H/W:
VAX
MIPS
PC
MAC

Jim T. – 1 of 3

A Model For Portability



Influence Standards

* Trade off against portability requirements

Jim T. - 2 of 3

Recommendations

- Define DEC Run-Time Toolkit Architecture—Portable
- Define SDT Run-Time Toolkit Architecture—Platform Dependent
- Implement in portable C except for critical components tuned to platform requirements.
- Participate and drive appropriate components into key industry standards (OSF, POSIX, X/OPEN, PCTE). Ongoing effort based on prioritized components.

Jim T. - 3 of 3

AIA and Portability

Where were we going?

- ARUS for common architected runtime on MICA and FLINT, move to VMS and VAX/ULTRIX

- Investigating common architected service/components for "AIA platforms"

    - Data dictionary

    - Naming

    - DB access

Determining architected services that might be available

- Graphics

- DSMS (?)

Scott D-1

AIA and Portability

Where are we?

-ARUS for MICA and FLINT blown away. [Well, what about MICA?]

-Corporate commitment to OSF seems to imply commitment to POSIX. [And what else? And when? And Who?]

-Work on common services/components needs a new plan.

Scott D, - 2

AIA and Portability

Where do we go from here?: APA

-Sort out OSF and what it means to portability

-In order to avoid perturbations in the future, go ahead full tilt with a common runtime environment for all applications: POSIX plus ARUS extensions.

-Be prepared to make ARUS run anywhere.

-Push ARUS into OSF, if we can; i.e., if we're serious about OSF.

ScottD,-3

BLISS to C Conversion Project

- Charter/Scope
- Strategy
- Status
- Issues

Dick W. - 1 of 5

BLISS to C Conversion Project

Charter/Scope

Purpose:

Position our current toolset/utilities into being portable to other architectures by replacing proprietary implementation language dependency (BLISS) with a broader based implementation language (C).

Scope:

The Following tools and utilities have been suggested for conversion consideration:

- VAXset
- SORT
- VAX Notes
- Project Manager
- CDD+
- Rdb

Non-goal:

Compiler Technology

Dick W. - 2 of 5

BLISS to C Conversion Project

Strategy

- To quickly convert our tools with minimal impact on engineering schedule
- Investigate vendors providing conversion tools/services
- Issue RFP to identified vendor candidates soliciting response for pilot conversion project (DTM)
- Evaluate responses
- Top two compete in DTM conversion
- Evaluate results
- Winner continues with other software (if quality from both is found comparable, possibly sub-divide the work between both)
- RFP spells out task, schedule and acceptance criteria. Also,defines input (BLISS language manual attachment) and output (Digital C Coding Standard attachment).

Dick W. - 3 of 5

BLISS to C Conversion Project

- Four candidates identified
  - LEXEME
  - RAPIDTECH
  - COMPASS
  - ISC
- RFP nearing completion
  - brief, 3 pages
  - BLISS language spec
  - C coding standard
  - sample of code
  - all covered by non-disclosure agreement
- Next Steps
  - Send to candidates by 07/8/88
  - their response due by 08/15/88
  - pilot project begin by 09/15/88
  - pilot project complete by 11/15/88
  - final vendor selection by 12/01/88

Dick W. - 4 of 5

BLISS to C Conversion Project

Issues

- Measuring portability of converted code
- Potential performance loss of converted code
- Code being converted is evolving dynamically
- What to reveal to vendors regarding retargets (e.g. VAX/ULTRIX)

Dick W. - 5 of 5

SDT Compiler Technology Strategy and Tactics

R. Grove

30 June 1988

World-Class Compiler Technology

"World-class" compiler technology is an essential ingredient for DEC

- Classical global scalar optimization
- Advanced scalar optimization
  - Loop unrolling
  - Code scheduling
- Interprocedural optimization
  - Inlining
  - Linkage Tailoring
  - Global register allocation
  - Compilation database
- Vectorization
- Parallel decomposition
- Multi-language code generator
- Industry-leadership language features
- Integrated tools environment

PRISM Compilers vs. MIPS

Global optimization

PRISM V1 compilers will have state of the art scalar optimization and loop unrolling

Target-specific code scheduling better than current MIPS

Interprocedural optimization

PRISM V1 compilers will do inlining, linkage tailoring, global register allocation. Equal to or better than MIPS.

Compiler architecture designed to support interprocedural database

Vectorization

PRISM V2 FORTRAN compiler

PRISM vectorizer based on VAX FORTRAN will equal or exceed industry leaders CONVEX and IBM

MIPS doesn't do it, doesn't believe in vectors

PRISM Compilers vs. MIPS (continued)

Parallel decomposition

- Automatic decomposition in PRISM V2 FORTRAN
- Decomposition builds on global optimizer and vector dependency analysis
- MIPS doesn't do it

Language features

- VAX languages are industry-leadership
- PRISM languages derived from VAX
  Designed and tested 100% VAX-compatible
- MIPS third-party front ends not 100% VAX-compatible

Tools and environment

- PRISM software architecture is the basis for future leadership products

What has Changed?

Changes as a result of MIPS decision

- MIPS ISP instead of PRISM, possibly others
- No vectors planned for MIPS in near term
- A much greater emphasis on parallel decomp and threads

The following are still true

- DEC will offer a comprehensive operating system (MICA/OSF) on a RISC processor (MIPS for now)
- DEC will define and implement a comprehensive software architecture on RISC processors
- DEC needs a full line of compilers

GEM targets

The following architectures are possible targets for some GEM-based compilers:

- VAX/VMS
- RISCy VAX
- MIPS (or others) running MICA/OSF
- MIPS running MIPS UNIX
- Future 64-bit machine (the future is not all that far away)

GEM added value for VAX/VMS

Replacing any existing VAX compiler by a GEM-based compiler is very difficult because of large customer base, compatibility issues, and reliability issues.

A GEM-based compiler must add substantial value such as vectors, decomp, interprocedural optimization, or major new tools

Opportunities for added value

- FORTRAN: Interprocedural
- Pascal: Vectors, decomp, interprocedural
- C: Vectors, better scalar code, C++ support
- Ada: Currently stretching limits of VCG. Improved optimization, robustness, retargetability, vectors
- BLISS: Modern optimization technology

GEM added value for others

RISCy VAX: An aggressive effort would use GEM-based compilers

- Common code generator
- Code scheduling and tailoring

MIPS UNIX systems - MIPS has a complete line of compilers now. There is relatively little that we could add:

- More VAX compatibility (but DEC is committing to help them solve this problem)
- VAX Ada - leadership Ada with optimization

SDT Compiler Strategy

Work with DECwest to execute the previous PRISM program on MIPS hardware with a greater emphasis on portability and retargetability.

Priorities for GEM work:

1. Multi-language retargetable compiler for FORTRAN, Pascal, BLISS, then Ada.
2. State of the art global optimization
3. Interprocedural optimization
4. Parallel decomp
5. Tools and environment

Design for future 64-bit and vector machines

VAX version as AD project initially

VAX RPC status

- Current status and schedule:
  - In first Field Test (March, 1988)
  - FT update in Fall '88
  - Working on a design based on the current VAX RPC spec.
  - Ship to SDC Q3 '89. (Current hope.)

- The V1 product:
  - A subset of the DEC RPC architecture.
  - No direct conflicts with the arch. Customers should be able to grow into the corp. arch. with little or no pain.
  - Not portable, highly VAX/VMS specific.

AL S. - 1 of 3

Effects of MIPS decision

- Effects on the corporate architecture
  - Architecture is targeted to multi-HW, multi-OS from inception.
  - Architecture is optimized for VAX hardware datatype formats.
  - We see *NO IMPACT* on the architecture.
- Effects on portability
  - Joint NAC/SDT proposal to provide common RPC tools is being put forward.
  - Components would need to be smarter sooner, but no unforseen or unsolvable problems due to MIPS.

AL 5. - 2 of 3

Effects of OSF announcement

- ~~We appear to have publicly endorsed~~ Apollo's RPC.
  SEEMS LIKE A CANDIDATE STARTING POINT.
- In several areas it is incompatible with the architecture that is emerging.
- Wasn't deemed adequate for our needs earlier, doubt that it has improved.
- We HOPE ~~are highly unlikely~~ to be able to influence it.
- The DEC RPC Architects need direction if a change is desired. We are going ahead as before.

- NAC DRIVING PROGRAM TO GATHER REQUIREMENTS, GAIN OSF MEMBERS BUY IN, AND INFLUENCE OSF RPC REQUIREMENTS.

AR S. - 3 of 3

VAX/ULTRIX Strategy

UEG's Position (Unofficial)

- MIPS platform is primary focus
- VAX platforms judiciously selected
- Wants SDT to focus on MIPs
- ULTRIX-32 will probably support VAX Vectors
- Decomposition seems to be a very low priority

Liz F. -1of2

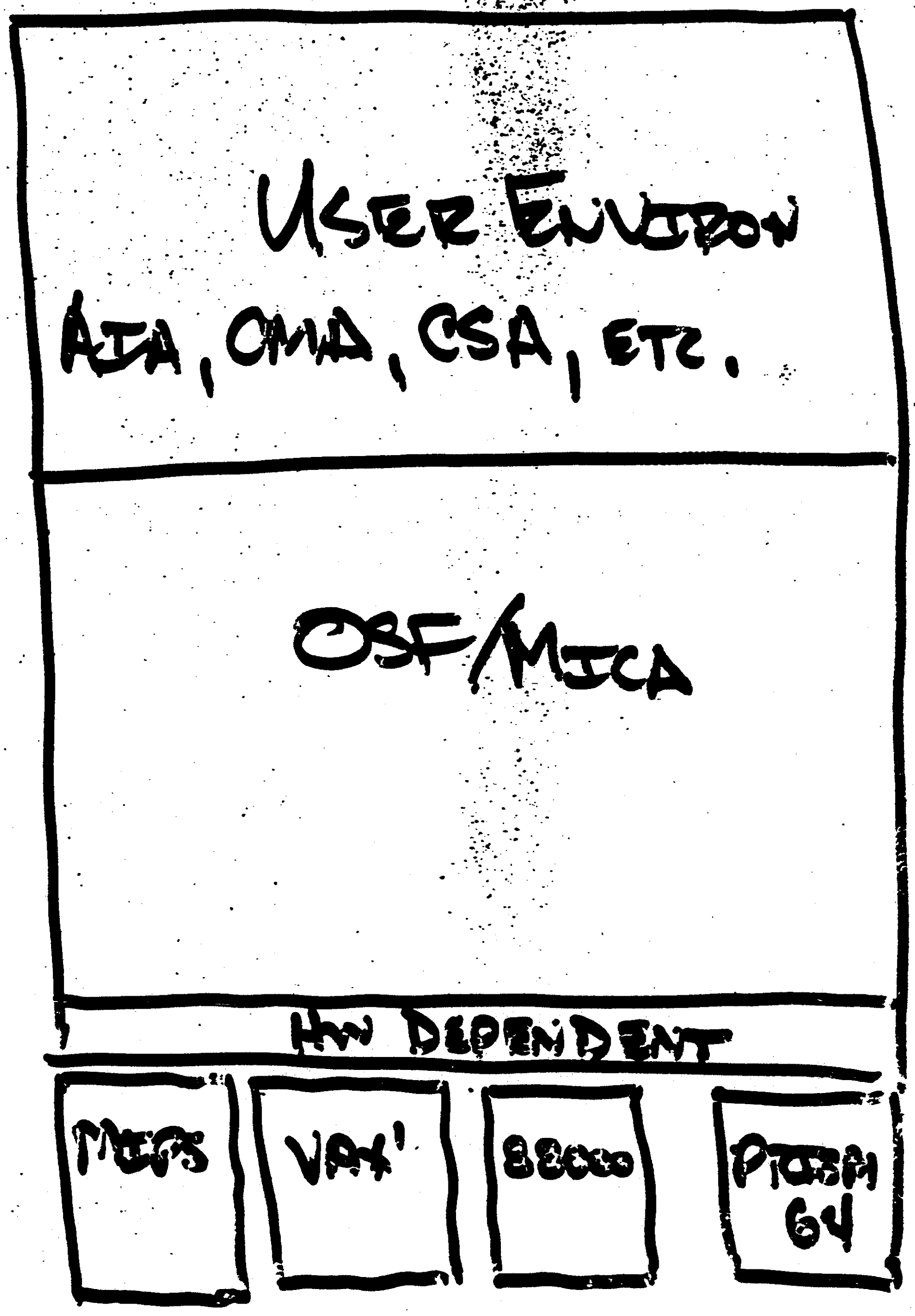
VAX/ULTRIX Strategy

SDT Product Issues

- UEG not committed to VCC as systems compiler for VAX/ULTRIX
- Funding for port of VAXset to VAX/ULTRIX uncertain
- Marketing will push for VAXset on all DEC platforms
- Marketing will push for VAX FORTRAN/ULTRIX and VAX C/ULTRIX VAX vector support

Liz F, 2 of 2

Another Model of Platforms



OSF Operating System Development Strategy

David N. Cutler
DECwest Engineering
June 27, 1988

OSF Vision/Goals

Produce a general purpose, OSF compliant, operating system for workstations and servers that:

- addresses the STF OS of the future requirements (e.g. portability, range, robustness, extensibility, etc.).
- provides all the components of the OSF Level 0 specification.
- exploits Digital's existing software technology.
- is integrated into the Digital Computing Environment.
- makes it easy to share layered products with VMS.
- provides at least source level migration for VMS applications.

OSF Level 0 Components

- Operating System - XOPEN, POSIX.
- Languages - C, Fortran, Pascal, Ada, Basic, Cobol, Lisp.
- User Interface - X Windows, X language bindings.
- Graphics libraries - GKS, PHIGS.
- Networking Services - Selected ARPA/BSD services, TCP, IP, SMTP, TELNET, FTP, Selected OSI protocols.
- Database management - SQL.

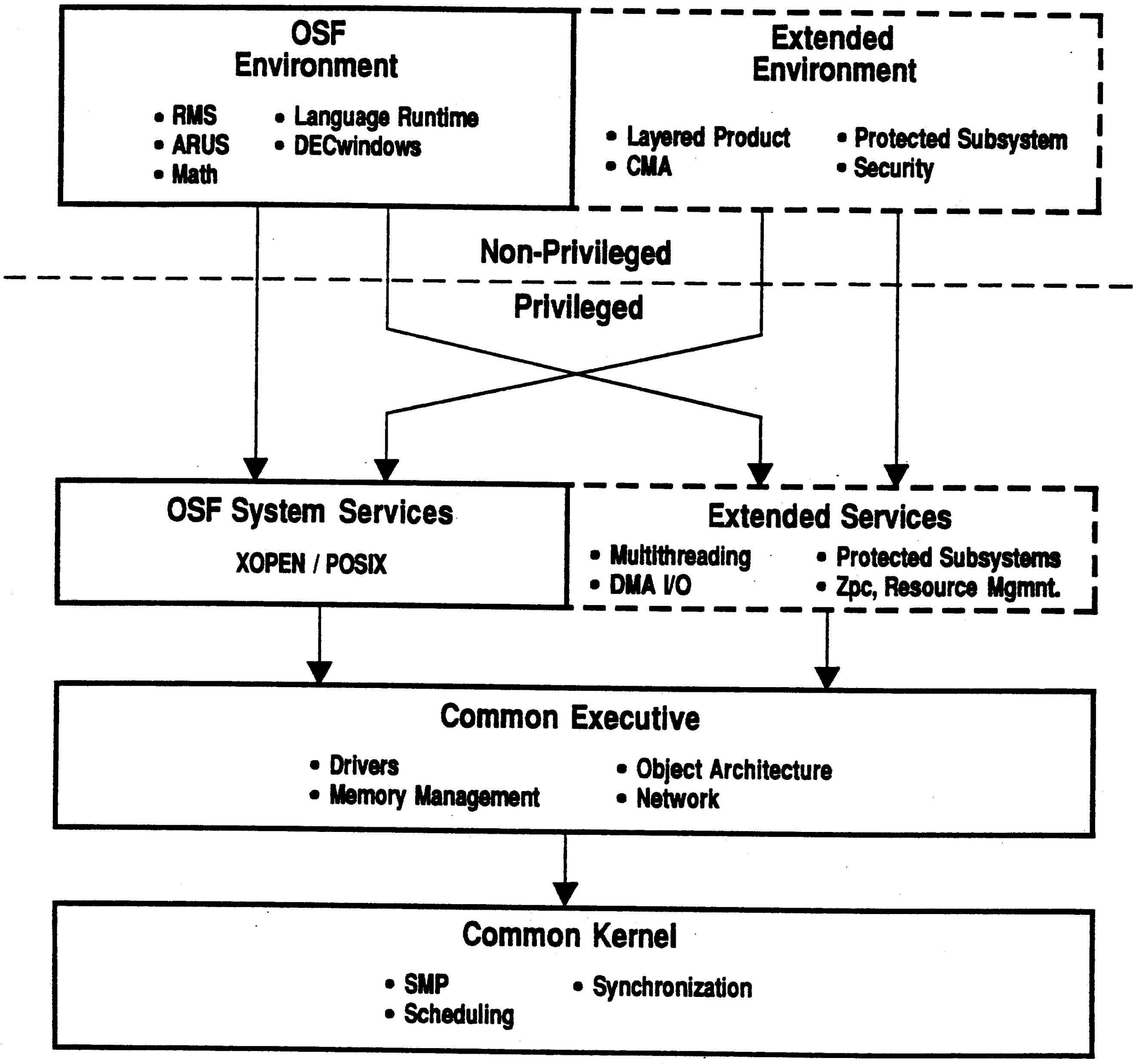
Opportunities for OSF Leadership in Base Systems Technology

- Multithreading (CMA).
- Enhanced security (ACL based protection) and protected subsystems.
- Shareable code libraries, named data segments, dynamic binding.
- Advanced file system capabilities.
- Integration of DDA and DDTA, two Phase Commit, resource management, recovery.
- High performance, high availability I/O - DMA non-buffered I/O, striping, shadowing.
- System and network management.

Opportunities for OSF Leadership in Program Environment Technology

- Uniform interlanguage calling and condition handling standard.
- Industrial strength, high quality, state-of-the-art VMS compatible compilers.
- Integrated program support environment and CASE tools.
- Record management, math library, and application runtime utility services (e.g. AIA capabilities).
- RPC capabilities for intra- and inter-system operation with VMS, other OSF systems, and UN*X.
- DECwindows, VMS migration, and layered product compatibility.
- Support of SQL and data management capabilities.

Single Environment Proposal



Advantages of Single Environment Proposal

- Absolute OSF compliance with extended features incorporated directly as system services.

- No tradeoffs made in any dimension for VMS, Mica, or compatibility with any other system.

- A single execution environment with the OSF case sensitivity, character set, filename, and parsing rules.

- Database, enhanced security, protected subsys-terms, availability, and other industrial grade features provided by extended OSF environment.

- Common language runtime environment (e.g. math library, RMS, ARUS, etc.), DECwindows, and AIA support.

- Multithreading available via the Common Multithread-ing Architecture.

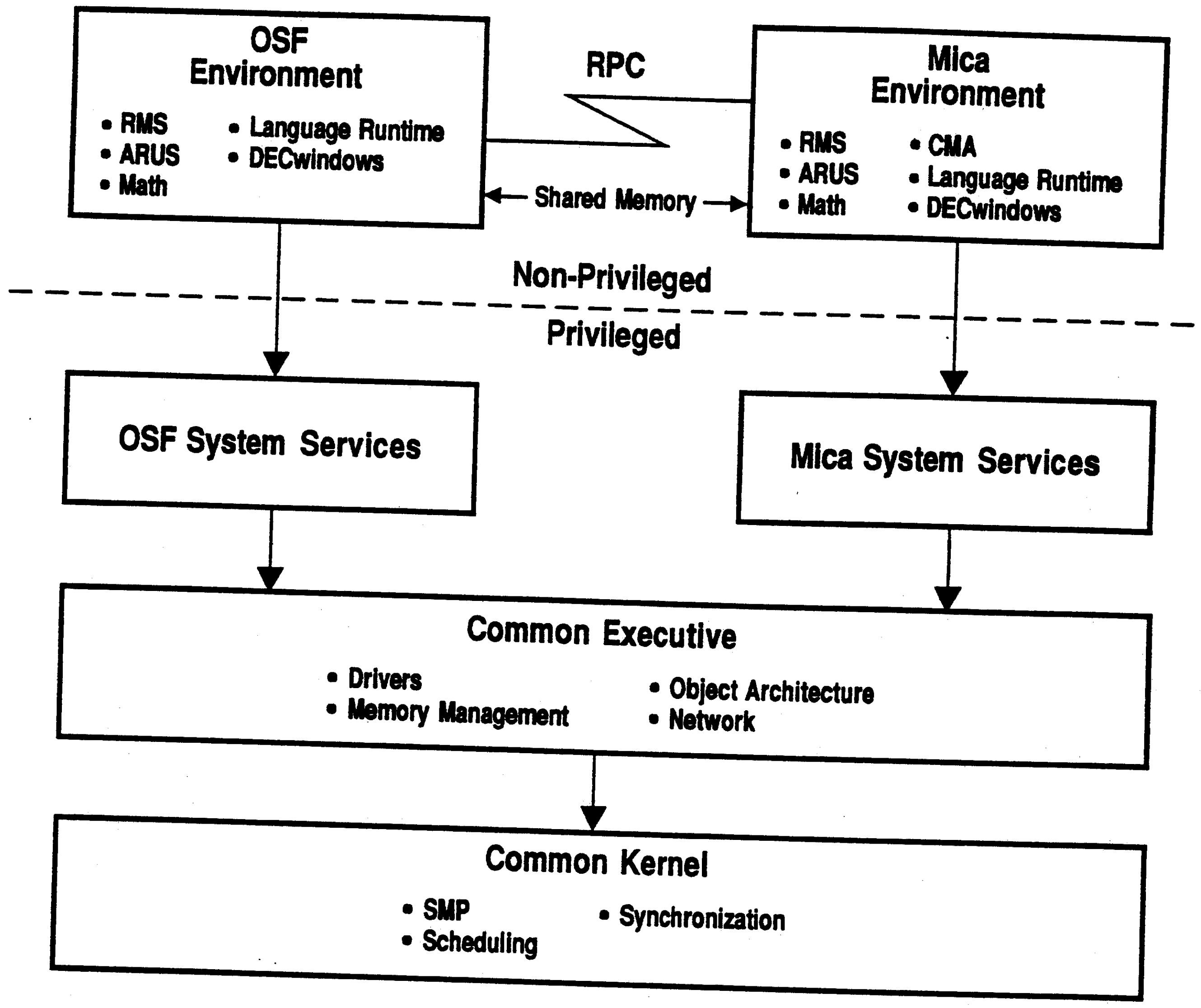
Disadvantages of Single Environment Proposal

- The extended OSF environment is not currently defined by any standard and will require extensive additions to the XOPEN and POSIX standards - an undertaking that should not be taken lightly.

- Incorporation of certain extended features may be difficult and require incompatible changes (e.g. ACL's on all objects, protected subsystem support with impersonation services, multithreading, etc.).

- VMS layered products and applications that do not use AIA interfaces will be more difficult to port.

Dual Environment Proposal



Advantages of Dual Environment Proposal

- Absolute OSF compliance without impacting the design and capabilities of Mica.
- No need to make tradeoffs in favor of either OSF or VMS - both can be accommodated equally.
- Database, enhanced security, availability, and other industrial grade features available to both environments.
- Provides largest number of VMS layered products earliest via high degree of compatiblity with Mica environment.
- Common language runtime environment (e.g. math library, RMS, ARUS, etc.), DECwindows, and AIA support.
- Mica protected subsystem functionality available to OSF programs as well as Mica programs - prefered method for adding functionality.

Disadvantages of Dual Environment Structure

- Programs must be executable from more than one environment which has implications on case sensitivity, character set, filename and command parsing.
- OSF capabilities may be perceived as second class when compared to those of Mica and therefore Digital's commitment to OSF may be questioned.
- New OSF functionality is added via Mica protected subsystems rather than pioneering and leading OSF standards activities which would incorporate these capabilites directly.

Open Questions

- What is the OSF operating system standard - XOPEN and POSIX?

- What about VAX ULTRIX compatibility, Berkeley UN*X compatibility, SVID compatibility?

- What should be done about UN*X concepts that compromise security (i.e. set UID and set GID)?

- What about Sun tools - Yellow Pages, RPC, etc.?

- What is the relationship of OSF to MIPS jelly bean UN*X? MIPS compilers to SDT compilers?

- Will an OSF system that is not pure UN*X be saleable in the UN*X market?

How Do We Get the Project Back on Track

- Get agreement on OSF implementation and product strategy and its relationship to MIPS UN*X.
- Assess current status and changes required by MIPS architecture and decided upon strategy.
- Define hardware platforms, OSF product, and layered product deliverables.
- Revise plans and set new schedules.

Assessing Current Status

- DECwest produces short analysis of significant architectural differences between MIPS and PRISM.
- DECwest produces short architectural description of OSF implementation and product strategy alternatives.
- SDT/DECwest analyze runtime software changes.
- SDT/DECwest develop code generation white paper.
- SDT generates calling standard white paper.
- DECwest analyzes the impact of the MIPS privileged architecture on the Mica memory management, condition handling, and multiprocessing capabilities.

Other Activities

- DECwest produces plan for MIPS development environment (development systems, communication tools, development tool changes, development compilers).

- SDT/DECwest define changes to:
  + Calling standard
  + RPC strategy

- DECwest evaluates impact on previous plans for seamless client-server communication and integration.

- DECwest evaluates the Mica I/O strategy to determine what VAX devices, if any, can be supported by MIPS architecture.

- SDT/DECwest Product Management and development rethink what, when, and how for compilers and layered products (e. g. SQL)

What to do about MIPS?

Preparing this presentation in absence of answers to many questions, had to take a very general approach.

Have made a few assumptions, asked some questions, and explored some possible answers.

Drew up a straw horse proposal to stimulate discussion and provide something to react to.

What to do about MIPS?

Assumptions

1. Assume (at least) four significant (overlapping) operating system audiences:
   a. UNIX (tm) purists
   b. OSF
   c. VMS migration/capability
   d. Industrial–strength operating environment: high performance, highly available, fault tolerant, multithreaded, shared servers, etc. (MICA)
2. Assume that Ultrix or MIPS UNIX will be A.
   Refer to this system as "UNIX".
3. Assume that MIPS–based hardware will be used to address at least one of B, C, and/or D.
   Refer to B, C, and D collectively as "industrial–strength OSF system".
4. Assume that Ultrix or MIPS UNIX will not be B, C, or D.
5. Then, we are faced with two operating environments on the MIPS–based hardware.

What to do about MIPS?

Questions

1. Will we define a Rehostable Software Architecture for portability, to which we target portable SDT products?

   Propose YES. This is an important strategic requirement for positioning our products for future requirements, opportunities, and portability.

2. Will Rehostable Software Architecture be vanilla UNIX?

   Propose NO. Application Integration Architecture, Digital added value, and benefits of more comprehensive software architecture (c.f. this morning's software architecture presentation) demand more than vanilla UNIX.
   Note that this will require investment in design and multiple deployments.

What to do about MIPS?

Questions

3. Do we build most layered products for industrial-strength OSF system?

   Propose YES. This will be a strategic systems, will require SDT layered products, represent an opportunity for SDT, and be a good first target for rehostable software architecture for portability.

What to do about MIPS?

Questions

4. Do we build versions of most layered products for UNIX?

   DON'T KNOW. Depends on answers to other questions below, and on business requirements.

   - Note that many third-party products are available for "vanilla UNIX", so should SDT invest in that system in addition to the OSF system?
   - PMAX people have argued strongly that absolutely no SDT products besides DECwindows client software and toolkit are required.
   - Will revisit this question after looking at some other questions.

5. Will we implement Rehostable Software Architecture on UNIX?

   DON'T KNOW. Depends on where we want to invest.

What to do about MIPS?

Questions

6. Will we take the non–portable software implementation architecture (calling standard, condition handling, object language, etc.) of MIPS as is, or will we apply our experience and goals to provide a better MIPS–specific implementation architecture for industrial–strength OSF system?

   NOT ENOUGH INFORMATION AVAILABLE. Need to evaluate the MIPS implementation architecture.

7. Should we build compilers for MIPS UNIXsystem?

   NO. Adequate compilers already exist.

8. Should we build compilers for MIPS OSF system?

   DON'T KNOW. Must answer questions about non–portable software implementation architecture, and must determine requirements of OSF system which existing MIPS compilers don't satisfy.

What to do about MIPS?

Questions

9. Revisited, when do we build a layered product for UNIX?

   Propose when business requires product (third party UNIX product are not adequate); AND in addition

   – all software architecture (portability architecture and non-portable implementation architecture) required by product is provided compatibly with industrial-strength OSF system; OR

   – if we are willing to build the software product twice; OR

   – if we can just recompile product from VAX Ultrix for MIPS UNIX.

Otherwise, don't build product for MIPS UNIX.

What to do about MIPS?

Trade–offs

Summarizing, the trade–offs include (but are probably not limited to)

- Costs of new software architecture design and deployment versus the benefits of such investment.
- Costs and liabilities of common software architecture versus cost of developing products twice.
- Costs of maintaining software for multiple software environments versus opportunities for selling products in both environments.

ALSO, we should treat the MIPS system at the FIRST of a possible SERIES of such targets, NOT as an isolated system.

We should develop a position and strategy for MIPS which applies to other such targets and which will help position us for future targets.

What to do about MIPS?

Straw Horse Proposal

- Keep current premise that SDT has at most two strategic targets.
- Those targets are currently MIPS OSF and VAX/VMS
- SDT long–term strategy is to converge to ONE "virtual target" by deploying rehostable software architecture on VAX/VMS, and other strategic targets.
- SDT deploys reshostable software architecture on MIPS OSF.
- SDT provides products on MIPS OSF.
- Products required are similar to those required for PRISM systems.
- SDT does not deploy portability architecture on MIPS UNIX. Let someone else do that if business requires it.
- SDT provides products on MIPS UNIX if business requires them AND if they can be targeted to a common software architecture.

What to do about MIPS?

Issues and Problems

- What to do about non-portable architecture on OSF system needs to be researched.
- What to do about compilers on OSF system needs to be researched.
- Have to deal with architectural incompatibilities with VAX. Floating point formats, etc.
- What do we do with the BLISS-based products?
- How different are UNIX and OSF, really? Will re-hostable software architecture go from OSF to UNIX "for free"?

DEFINITIONS

1. **Portability:**
   - Multiple hosts
   - recompile and run
   - requires that everything you plug into is on host (e.g. VAX instruction set, library routines, file system, dictionary ...)
   - requires great care in managing what you depend upon
   - requires avoiding dependency on any single operating environment, even indirectly (e.g. "memory is cheap")
2. **Retargetability**
   - Mutiple targets
   - Requires that everything you plug into is there (e.g. language RTL for generated code)
   - Requires that certain support is present on the host(s) to enable building software for the target and communicating with the target
   - May imply a certain structure/design for a software product (e.g. main/remote debugger split)

Lebvik 1 of 5

PRODUCT PORTABILITY LEVELS

1. Absolute—recompile it and it runs on the new host /target
2. Partial—well-defined modules need to be rewritten in well-defined ways. Performance tuning criteria well-understood. Can port the product in wall-clock time of 6-9 months.
3. Painful—product requires a rewrite that could take close to the same amount of effort it took to write it originally.

[illegible] 2of5

DEVELOPMENT ENVIRONMENT

SUPPORTING REHOSTING/RETARGETTING

OUR PRODUCTS

1. Portablity checking - compiler(s) and tools must support writing portable code (syntactic, semantic checks)
2. Cross-development tools to automate/support development processes: Including good DTM support, performance analysis support, configuration management support ...
3. Defined portable interfaces
4. Well-documented development methodologies, including a well-defined process for doing ports (probably includes "recompile and run" versions of base components to allow quick and dirty ports to get the testing environment in place)
5. Technology that supports retargetting our implementation language(s) quickly and rehosting the runtimes it/they depend on quickly

[illegible] K - 3 of 5

MAJOR STUMBLING BLOCKS

1. Implementation language
2. Run time support of all flavors
3. Operating environment (OS, etc.) assumptions reflected in product structure/design/packaging

[illegible]

QUESTIONS

1. How do we know when we are sufficiently "portable"?
2. What must always be ported (rehosted)? What needs to be retargetted?
3. What are the goals?
4. What are the "bounds" of portability? Can we come up with a reference model for what sorts of hardware and software environments we want to be positioned for, versus those that will never be of interest?

Oleshik–8 of