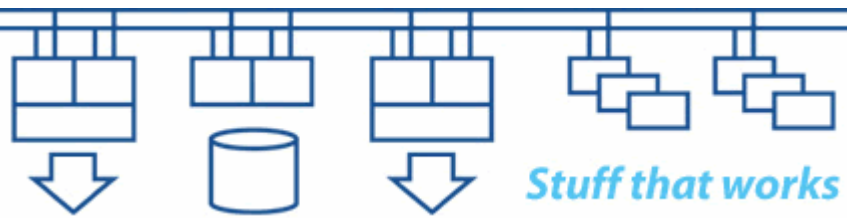
HPUG "Back to Basics" series

Hardware Platform Architecture

Colin Butcher

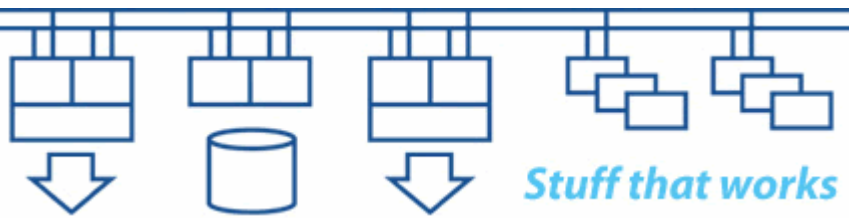
Technical Director, XDelta Limited



Seminar goal

Gain an understanding of how the different hardware components of a computer system work individually and how they interact with each other within a complete system.

This seminar is unlikely to answer all of your questions, so please be prepared to contribute and share your knowledge.

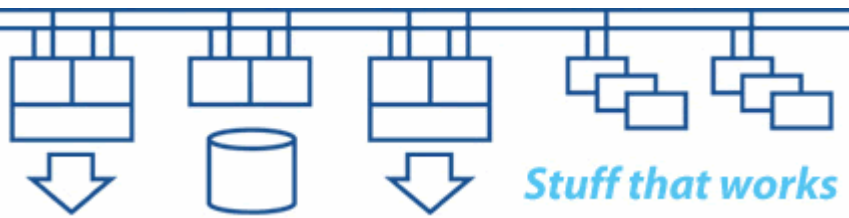


Topics (1)

- **Basic principles**
- **The functional elements in a computer system**
- **History and development**
- **What do we mean by “hardware”?**
- **CPUs (Central Processing Units)**
- **Memory subsystem**
- **IO subsystem**

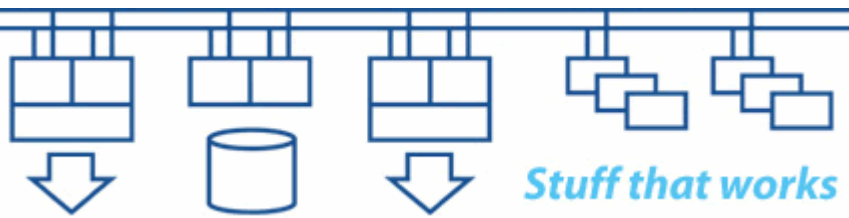
Topics (2)

- **Interconnects**
- **Bus structures**
- **Performance characteristics**
- **Environmental characteristics**
- **Software implications:**
  - **Operating systems**
  - **Applications**



History (1)

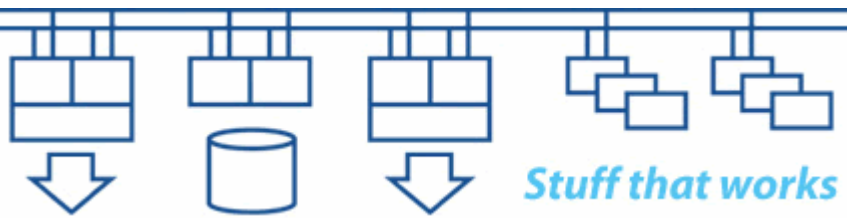
- **The stored program computer concept:**
  - **Babbage's difference engines**
- **Early days:**
  - **Bletchley Park, Turing & Colossus**
  - **Valves**
  - **Mercury delay lines**
  - **Paper tape**
  - **Core memory**



History (2)

Computers are only a tool to do a job:

- **Repeatable mathematics quickly (eg: generating tables for logarithms, gunnery etc.)**
- **Solving equations (eg: calculus)**
- **Warfare (eg: signals decoding – Colossus)**
- **Storing and processing data (eg: LEO)**
- **Control systems (eg: fly-by-wire aircraft)**
- **Mathematical modelling (eg: stress analysis)**
- **Data collection and analysis (eg: phamaceuticals)**



Binary arithmetic and Boolean logic

There are 10 kinds of people in the world –

those who understand binary arithmetic and those who don't.

Boolean logic:

- **basic gates: AND, OR, NOT**
- **derived gates: NAND, NOR, XOR, XNOR**

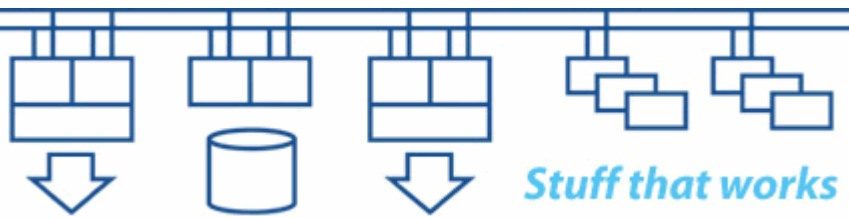
Logic gates – truth table

Input A	Input B	AND	NOT AND	OR	NOT OR	XOR	NOT XOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Basic principles of performance

- **Bandwidth – determines throughput**
- **Latency – determines response time**

- **Parallelism**
- **Synchronisation**
- **Serialisation**



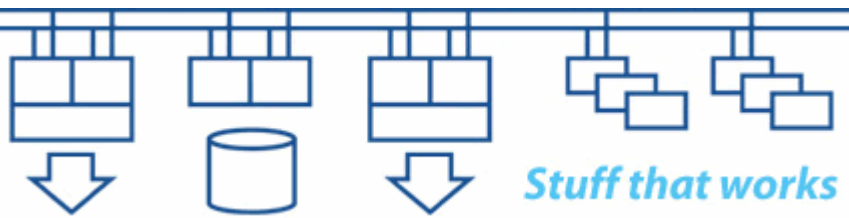
Basic limits to performance

- Speed of light
- Feature size in “chip”
- Signal degradation
- Power consumption
- Heat dissipation
- Chemical impurities
- Mechanical contacts
- Manufacturing process

Components of a computer system

- CPU (Central Processing Unit)
- Memory subsystem
- IO subsystem
- Storage subsystem

- Power
- Cooling
- EMI and RFI protection
- Mechanical protection

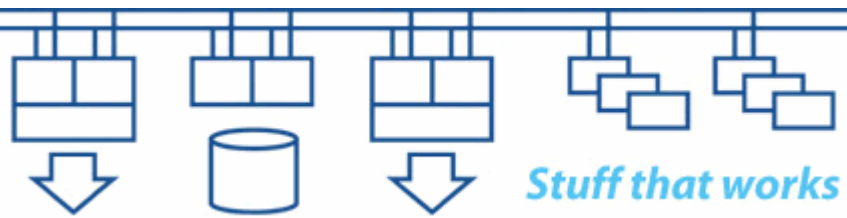


CPUs (1)

- **Processes instructions**
- **Manipulates data**
- **Instruction Set**
- **ALU (Arithmetic and Logic Unit)**
- **Microsequencer**
- **Protection mechanisms (processor modes)**
- **Clocking**
- **“bit width” (16bit, 32bit, 64bit …)**

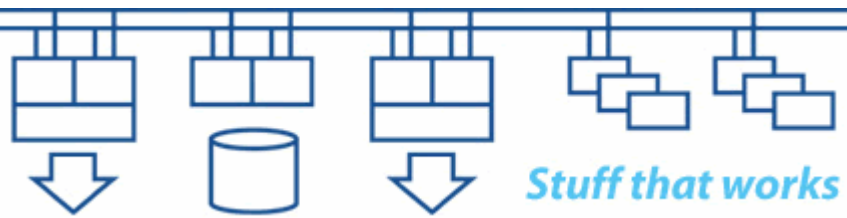
CPUs (2)

- **Performance:**
  - **Registers**
  - **Cache**
  - **Pipelines**
  - **Parallel execution**
- **CISC (Complex Instruction Set Computer)**
- **RISC (Reduced Instruction Set Computer)**
- **EPIC (Explicitly Parallel Instruction Computing)**
- **Compilers are the key to performance**



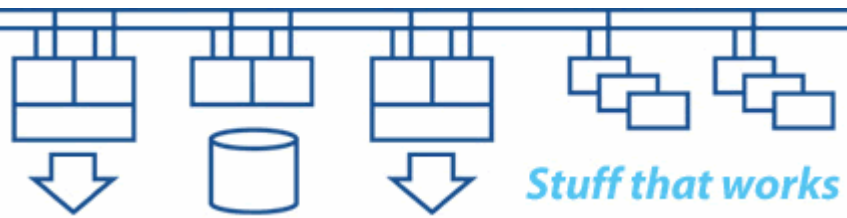
Memory subsystem

- Used to be small, slow and expensive, so tried hard to minimise memory usage
- Now plentiful, fast and relatively cheap, so now ignore it – to our peril!
- Can use memory to gain performance
- Cache and synchronisation
- Error detection and correction
- “Locality” to CPUs
- Memory interconnects to CPUs



IO (Input / Output) subsystem

- **Provides interface with outside world (physical, electrical, logical …)**
- **Connects to storage subsystems**
- **Access times (memory, cache etc.)**
- **“Locality” to CPUs**
- **IO devices operation (interrupts, registers)**
- **IO interconnects to CPU and memory subsystems**



Device detection (VAX, Alpha etc.)

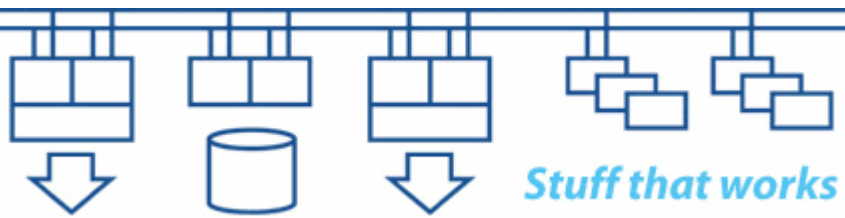
- **Console firmware**
- **Devices appear as a set of CSRs (Control and Status Registers) in physical memory - the IO space**
- **Devices have Interrupt Vectors which connect a device interrupt request to the device driver Interrupt Service Routine**
- **CSR addresses and contents indicate device type**
- **Operating system will scan IO space to find devices and set up device drivers to communicate with the hardware**

Device detection – Itanium2 (Integrity)

- **Itanium Processor Family architecture uses ACPI (Advanced Configuration and Power Interface) for device detection by firmware**
- **Devices appear as a set of CSRs (Control and Status Registers) in physical memory - the IO space**
- **Devices have Interrupt Vectors which connect a device interrupt request to the device driver Interrupt Service Routine. Device data obtained from ACPI data.**
- **ACPI data indicate device type**
- **Operating system will query ACPI data to find devices and set up device drivers to communicate with the hardware**

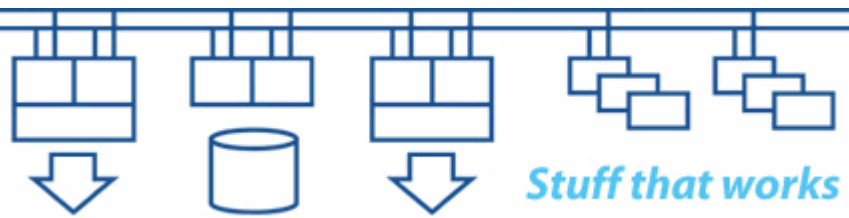
Storage subsystem

- **Lifetime requirements of data**
- **Access requirements for data**
- **Data integrity**
- **Data security**
- **On-line storage**
- **Near-line storage**
- **Off-line storage**
- **Storage interconnects to IO subsystem**



Discs – they're simple!

- **Data on rotating surfaces**
- **Read / Write heads move radially**
- **How to find data?**
- **How to record data?**

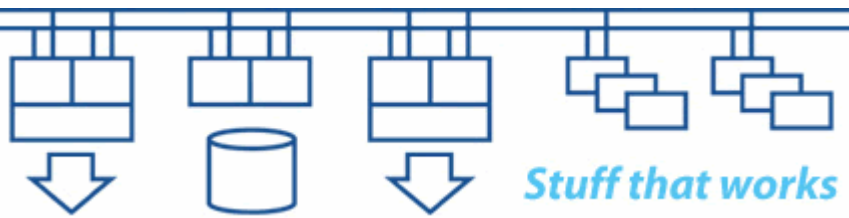


Tapes – they're simple too!

- **Data on moving surface**
- **Read / Write heads fixed (true for ½" tape)**
- **How to find data?**
- **How to record data?**

Part of the Operating System

- **Devices are “block structured”**
- **Volumes are “file structured”**
- **Data are “record structured”**
- **Operating System arbitrates access, e.g. Distributed Lock Manager in OpenVMS**



Useful aphorisms

- **Occam's Razor:**

  "Pluralitas non est ponenda sine neccesitate"

  "Entities should not be multiplied unnecessarily"

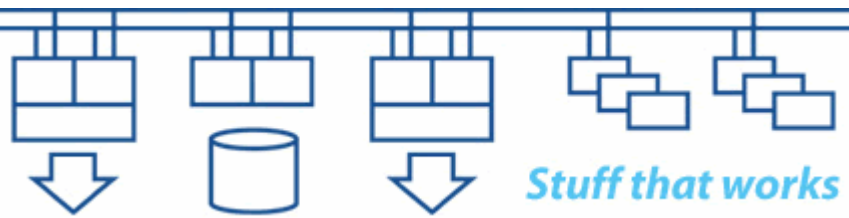
  "Keep it as simple as possible"

- **Hanlon's Razor:**

  "Never attribute to malice that which can be adequately explained by stupidity"

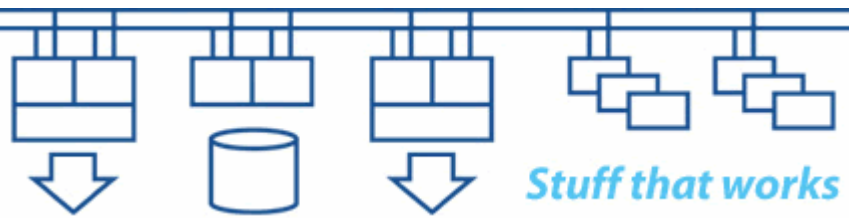
Multiprocessors

- Asymmetric Multiprocessing (attached processor)
- Symmetric Multiprocessing (SMP)
- Interconnections (switch, mesh, toroid etc.)
- Latency and Bandwidth
- Synchronisation and Serialisation
- IO processing
- Partitioning
- “Locality of resources” (eg: RADs in OpenVMS)



Systems - examples

- **Configuring systems for:**
  - **Cost**
  - **Performance**
  - **Availability**
  - **Maintainability**
  - **Safety**
  - **Simplicity**



Hardware summary - VAX

- **32bit CISC. Designed when memory was expensive, so instruction set tries to save space where possible (variable length instructions etc.)**
- **Includes specific instructions to assist when writing operating system software, such as providing guaranteed synchronisation of access to data structures.**
- **Early VAXes included true PDP11 compatibility.**
- **Support a wide range of IO devices and bus structures.**
- **“Bomb-proof” engineering, especially big VAXes.**

Hardware summary - Alpha

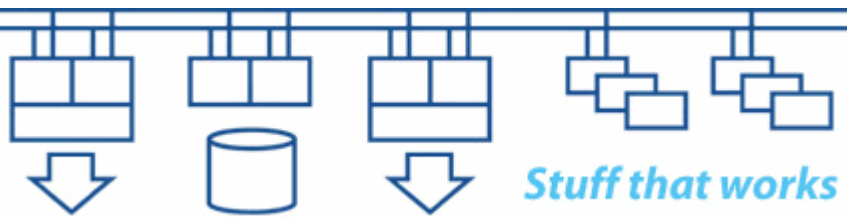
- **64bit RISC. Designed for performance.**
- **Best performance achieved with data aligned on 64bit boundaries (no need for bit masking in registers etc.).**
- **Needs synchronisation issues to be carefully considered and coded. What was a single "atomic instruction" on VAX can be multiple instructions on Alpha, so instruction flow through the CPU can be interrupted.**
- **"VAX like" console commands (SRM console).**
- **"PC like" BIOS (AlphaBIOS console).**
- **Predominantly PCI bus based.**

Hardware summary - Integrity

- **64bit EPIC. Designed when memory is plentiful and cheap.**
- **Itanium Processor Family architecture relies on compiler output to generate an efficient instruction and data flow through the CPU**
- **Needs synchronisation issues to be carefully considered and coded. What was a single "atomic instruction" on VAX can be multiple instructions on Integrity, so instruction flow through the CPU can be interrupted.**
- **Has both MP and BMC consoles. Uses EFI rather than BIOS. No "VAX like" or "Alpha like" console.**
- **PCI bus based (3.3volt only).**

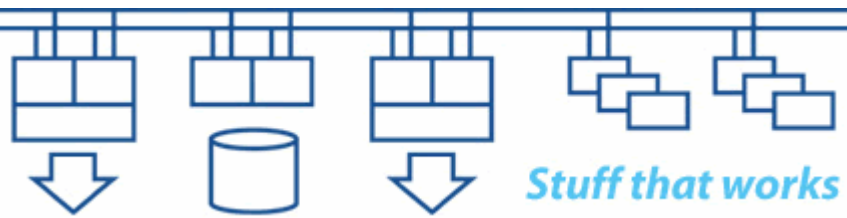
Software considerations

- Operating system design
- System configuration
- Performance tuning
- Application design – synchronisation
- Compilers generate code better than you can
- Portability
- Reliability
- Supportability



Question and Answers

Q & A



Contacts

Colin Butcher, XDelta Limited

Office:	**+44 117 904 8209**
Cellphone:	**+44 7768 857615**
E-mail:	**colin.butcher@xdelta.co.uk**
Web:	**http://www.xdelta.co.uk/**