FlingOS – TB1 – OS Dev Lecture Course

# Lecture 1 – Introduction, Booting

* Introduce course
* Explain course format: short lecture followed by workshops
* Announce sponsor (Imagination Technologies)
* Ensure everyone is of appropriate ability / motivation
* Explain x86 boot sequence
* Explain MIPS boot sequence
* Explain basic x86 boot ASM code
* Explain basic MIPS boot ASM code
* Q&A
* Workshop:
  + Get everyone set up with toolchains (x86 & MIPS)
  + Get people set up with virtual machines (for x86)
  + Get people set up with CI20s (for MIPS)
  + Compile & run first basic OS
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 2 – x86, MIPS and Assembly Language

* Welcome back
* Explain x86 architecture features
* Explain MIPS architecture features (with comparison to x86)
* Explain aspects of embedded vs. large-scale development (CI20 vs. PC)
* Explain / demo x86 assembly language
* Explain / demo MIPS assembly language
* Q&A
* Workshop:
  + Distribute CI20s
  + Get people working on using more x86/MIPS features
  + Get people working on writing more complex ASM code
  + Get people using online x86/MIPS ASM resources
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 3 – Initialisation (MIPS & x86)

* Welcome back
* Explain x86 initialisation sequence
  + Involves most of previous lecture’s architecture features
* Explain MIPS initialisation sequence
  + Involves some of previous lecture’s architecture features
* Demonstrate some of the pitfalls of incorrect initialisation
* “Look ahead” in design process/route to explain why some initialisation is necessary
* Q&A
* Workshop:
  + Distribute CI20s
  + Get people working on implementing initialisation sequences (x86 / MIPS)
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 4 – Compiling & Linking using C

* Welcome back
* Explain why we want to use C
* Explain why ASM is still necessary
* Explain what compiling is
* Explain the steps a compiler takes
* Example: GCC
* Explain what linking is
* Explain what dynamic linking is
* Explain what steps the linker takes
* Example: Ld
* Explain what a Makefile is
* Demonstrate cs-make, gcc and ld
* Demonstrate NASM and ld
* Explain what C standard library is
* Explain why we can’t use the C standard library
* Q&A
* Workshop:
  + Distribute CI20s
  + Get people working on writing basic C implementations (x86 / MIPS)
  + Get people linking ASM and C (x86 / MIPS)
  + Get people looking at the standard library and functions to increase their understanding (x86 / MIPS)
  + Re-collect all CI20s (and any other items lent out)
* Next time
  + C# but workshop still running even if people don’t want to do C#

# Lecture 5 – Compiling & Linking using C#

* Welcome back
* Warning: Windows & VS2013 only.
* Explain why we might want to use C#
* Explain what IL code is
* Explain MSBuild steps
* Explain FlingOS Compiler steps
* Explain plugs
* Demonstration / example
* Q&A
* Workshop:
  + **Allow this workshop to start alongside lecture so some people can carry on in C / ignore C#**
  + Distribute CI20s
  + Get people set up with VS / IDE
  + Get people set up with FlingOS Compiler
  + Get people writing their own basic C# OS (x86 / MIPS)
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 6 – Memory (MIPS & x86)

* Welcome back
* Explain types of memory
* Explain types of memory hardware
* Explain memory addressing systems (mention Isolated I/O)
* Explain x86 real-mode memory
* Explain virtual memory as a concept
* Explain segmentation
* Explain paging
* Explain x86 segmentation/paging combo setup
* Explain x86 segmentation configuration
* Explain x86 paging configuration
* Explain MIPS memory layout
* Explain MIPS memory configuration & pitfalls
* Q&A
* Workshop:
  + Distribute CI20s
  + Get people working on segmentation & paging configuration (x86)
  + Get people working with / investigating memory layout (MIPS)
  + Get people comparing performance (x86 real vs protected)
  + Get people comparing performance (MIPS cached vs uncached)
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 7 – Interrupts (MIPS & x86)

* Welcome back
* Explain what interrupts are
* Explain why interrupts are necessary
* Explain alternatives to interrupts
* Explain x86 real-mode vs protected-mode interrupts
* Explain x86 protected-mode interrupts configuration
* Explain MIPS interrupts configuration
* Explain devices, IRQs and timer
* Explain timer configuration (x86 and MIPS)
* Demonstration of timer interrupt (x86 and MIPS)
* Q&A
* Workshop:
  + Distribute CI20s
  + Get people working on interrupts setup (x86 / MIPS)
  + Get people working on timer interrupts (x86 / MIPS)
  + Get people researching / investigating use of timer (particularly multitasking)
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 8 – Video-output (x86-only)

* Welcome back
* **Warning: Video output only possible on x86 (at this stage)**
* Explain displays
* Explains graphics and how it is different
* Explain display controller
* Explain display memory
* Explain display buffers / layers
* Explain how to output characters (with colour)
* Explain strings
* Demonstration
* Q&A
* Workshop:
  + **Allow this workshop to start alongside lecture so some people can carry on with MIPS**
  + Distribute CI20s
  + Get people working on outputting characters
  + Get people working on various string-oriented method implementations
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 9 – Keyboard (MIPS & x86)

* Welcome back
* Explain devices (again)
* Explain keyboards and mice
* Explain PS/2 (x86)
* Explain USB emulation of PS/2 (x86)
* Explain scancodes (x86)
* Explain keycodes and keychars (x86)
* Explain IRQ handling (x86)
* Explain serial
* Explain reading from UART (MIPS)
* Explain character queueing / buffering and critical vs non-critical interrupts
* Demonstration: Typing characters straight to screen (x86) / serial echo (MIPS)
* Q&A
* Workshop:
  + Distribute CI20s
  + Get people working on implementing PS/2 driver (x86)
  + Get people working on reading from serial/UART (MIPS)
  + Get people working on console/shell interfaces (x86 / MIPS)
  + Get people investigating performance issues / IRQ spamming/flooding (x86 / MIPS)
  + Re-collect all CI20s (and any other items lent out)
* Next time

# Lecture 10 – Review

* Welcome back
* Final lecture
* Review what’s been achieved
* Any further questions / follow up
* Where to go next: FlingOS (videos, articles, codebase, community), online, OSDev.org, other?
* Feedback
* Buy your own: CI20, Intel Edison, RaspberyPi, Arduino, Microchip PIC32/PICKit
* Workshop:
  + Distribute CI20s
  + Anything people want to do really…
  + Re-collect all CI20s (and any other items lent out)
* Goodbye!