Architecture Porting

What to Expect?

- Porting Linux for a new Architecture

Architecture Branch

- 20+ architectures supported in Linux
- Each typically contains
  - Makefile, Kconfig
  - kernel (Arch specific Kernel code)
  - mm (Arch specific MM code)
  - boot (for specific bootable target)
  - mach-* (LSPs)
  - configs (default configs)

Porting to a New CPU

- Create a similar directory structure
- Do all the architecture specific implementations for
  - Kernel interfaces
  - Memory Management interfaces
- Create/Modify Makefile & Kconfig, as required
- LSPs & configs would be added as & when the porting for the specific board is done
- Significantly challenging, if totally afresh
- Typically, start from a closer reference

Pre-requisites & Assumptions

- System RAM is already initialized
- System Memory Map is already initialized
- Optionally, Serial port is configured
  - For early kernel boot messages
- Have a closest baseline Kernel

Starting Point

- Modify the Architecture specific code
  - For example
    - For PPC, it is under arch/ppc/platforms
    - For MPC5200, the file is lite5200.c
  - A typical set would include few structures & functions
  - Typical Functions
    - *show_cpuinfo – CPU info texts
    - *map_irq – Hardware specific interrupt logic routing
    - *setup_cpu – CPU specific init
    - *setup_arch – Architecture specific init
    - platform_init – Board-specific init

Recall the Startup Flow

- machine_init
  - Called from arch/<arch>/kernel/head*.S
  - Parameter Setup / Passing before calling machine_init
    - These are typically setup by the bootloader
  - Called before MMU init. So, calls from it are memory restricted
  - Code is in arch/<arch>/kernel/setup.c
  - Calls platform_init with the parameters as is
- platform_init
  - Sets up Board Information Structure (BIS)
  - If initrd, Start & End addresses of ramdisk image are saved
  - Store the Kernel Command Line parameters
  - This code should be taking care of “Early Variable Access”

Early Variable Access

- Kernel statically linked to well known, user configured base address
  - KERNELBASE (Typically, 0xC0000000)
- Kernel is relocated to RAM (usually to 0)
- After MMU is enabled, this works all fine
- But before that, it should be relocatable, and access to symbols should be fixed up
- And, hence calls for a particular way of for the early variable access
  - By subtracting the offset KERNELBASE

Setting up BIS

- Usual ways
  - struct bd_info (From U-boot)
    - Address in r3 on PPC
  - struct bi_record (Attempt to unify)
    - Found by looking for a special tag
- Contains stuff like
  - Command Line Arguments
  - Start & End address of initrd image
  - Machine Type
  - Memory Info
  - Optionally, many more hardware information
- May or may not be used by the LSP

Final Steps

- Other Generic Architecture specific Functions needed, could be
  - machine_restart
  - machine_power_off
  - machine_halt
- Modify the preocessor / architecture specific C file & header file as per
  - Hardware specifications
  - Schematics
  - Any other relevant hardware platform data
- Update the Kconfig & Makefile, as appropriate

What all have we learnt?

- Porting Linux for a new Architecture
  - In particular, for a new CPU

Any Queries?