LYNXLINE

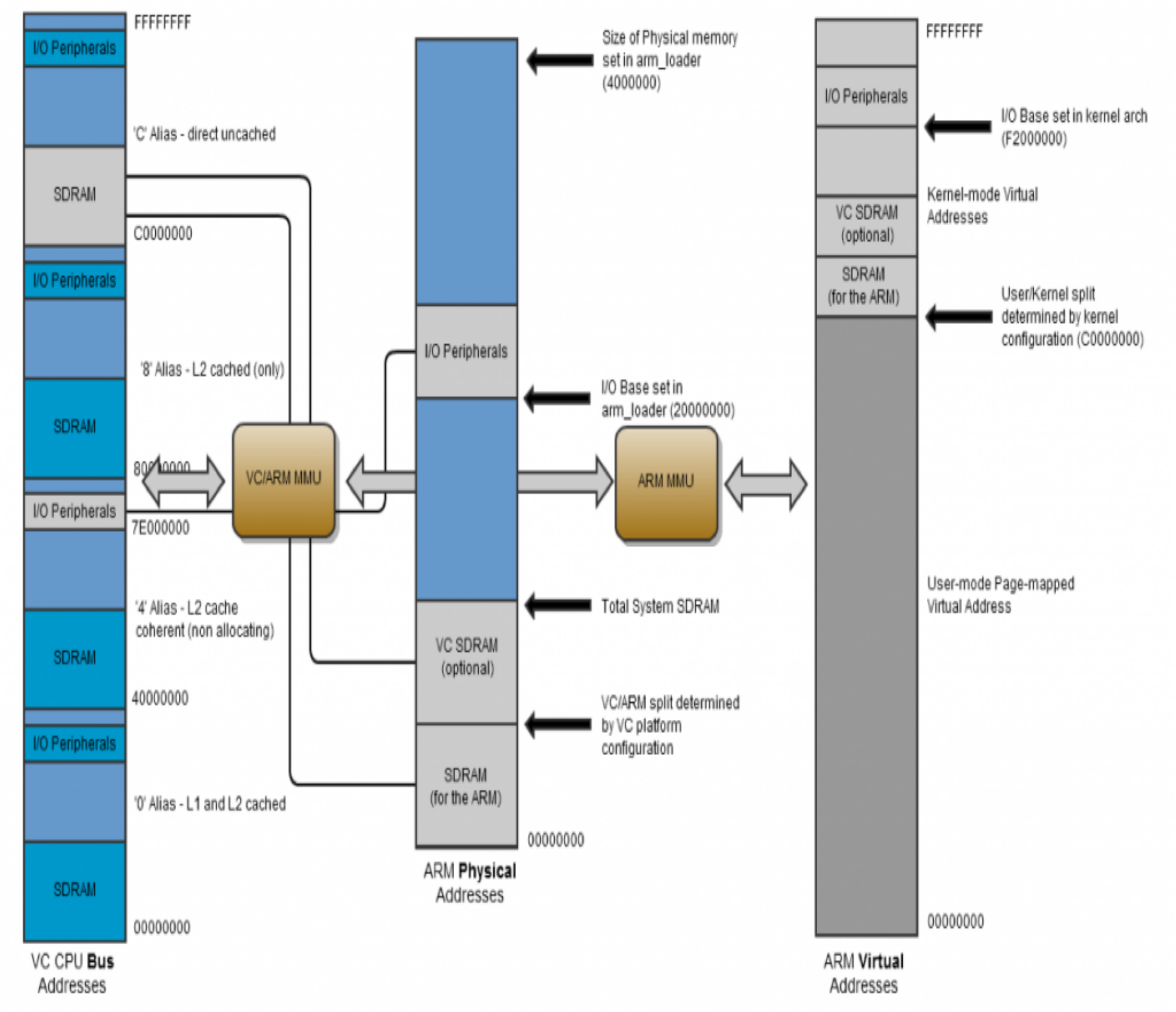
Professional Software Development Services

Home | Blogs | Projects | About | Services | Career | Contact Us

Lab 8, memory model

By ADMIN | *Published:* JANUARY 24, 2013

Now time to have a look at memory model. The BCM2835 gives next picture:


FFFFFFFF
I/O Peripherals
'C' Alias - direct uncached
SDRAM
C0000000
I/O Peripherals
'8' Alias - L2 cached (only)
SDRAM
80000000
I/O Peripherals
7E000000
'4' Alias - L2 cache coherent (non allocating)
SDRAM
40000000
I/O Peripherals
'0' Alias - L1 and L2 cached
SDRAM
00000000
VC CPU **Bus** Addresses
VC/ARM MMU
Size of Physical memory set in arm_loader (4000000)
I/O Peripherals
I/O Base set in arm_loader (20000000)
ARM MMU
Total System SDRAM
VC SDRAM (optional)
VC/ARM split determined by VC platform configuration
SDRAM (for the ARM)
00000000
ARM **Physical** Addresses
FFFFFFFF
I/O Peripherals
I/O Base set in kernel arch (F2000000)
Kernel-mode Virtual Addresses
VC SDRAM (optional)
SDRAM (for the ARM)
User/Kernel split determined by kernel configuration (C0000000)
User-mode Page-mapped Virtual Address
00000000
ARM **Virtual** Addresses

Actually when U-Boot pass the control, we have memory model equal to just physical addresses (middle column), so we just do not do anything we can perfectly sit with memory model equal with physical.

So then, our kernel is loaded at 0×8000, which is 32KB, and we are going to use the low 0-0×8000 memory part for kernel structures and kernel stack (which is not expected to be too much)

Now back to loader, we need to do some specific arm things before passing control: first we need to disable interrupts:

```
MOVW    $(PsrDirq|PsrDfiq|PsrMsvc), R1  /* SVC mode: interrupts disabled */
MOVW    R1, CPSR
```

Set address for stack: we decided to put *Mach* structure at address 0×2000 (8KB), so let's have a stack at MACHADDR+BY2PG (added a page 4KB) = 12KB, as result we will have stack area for kernel as 0×3000-0×8000 (12KB–32KB). It is 20KB, which is expected as enough of all kernel routines.

Categories

- Blog
- Boost
- C++
- Cryptography
- Embedding
- Hybrids
- Inferno OS
- MacAppStore
- Misc
- Models
- Projects
- PyQt
- PySide
- Qt
- QtSpeech
- Raspberry Pi
- Research
- Ru
- TogMeg
- Trac
- TTS
- Tutorial
- Undo
- Web

```
MOVW    $(MACHADDR+BY2PG-4),SP /* stack; 4 bytes for link */
```

Enable cycle counter:

```
MOVW    $1, R1
MCR     CpSC, 0, R1, C(CpSPM), C(CpSPMperf), CpSPMctl
```

Enable caches:

```
MRC     CpSC, 0, R0, C(CpCONTROL), C(0), CpMainctl
ORR     $(CpChv|CpCdcache|CpCicache), R0
MCR     CpSC, 0, R0, C(CpCONTROL), C(0), CpMainctl
MOVW    $0, R0
MCR     CpSC, 0, R0, C(CpCACHE), C(CpCACHEinvi), CpCACHEwait
```

(*these codes I took from 9pi, so may not good explain step-by-step, I would appreciate if pointed to much better description of all specifics*).
Resulting **load.s**:

```
#include "mem.h"
#include "armv6.h"

TEXT _start(SB), 1, $-4
    MOVW    $setR12(SB), R12     /* static base (SB) */
    MOVW    $(PsrDirq|PsrDfiq|PsrMsvc), R1 /* SVC: interrupts disabled */
    MOVW    R1, CPSR

    MOVW    $(MACHADDR+BY2PG-4),SP /* stack; 4 bytes for link */

    MOVW    $1, R1
    MCR     CpSC, 0, R1, C(CpSPM), C(CpSPMperf), CpSPMctl /* counter */

    MRC     CpSC, 0, R0, C(CpCONTROL), C(0), CpMainctl /* caches */
    ORR     $(CpChv|CpCdcache|CpCicache), R0
    MCR     CpSC, 0, R0, C(CpCONTROL), C(0), CpMainctl
    ISB

    BL      ,main(SB)
```

File armv6.h with all needed constants I took from 9pi sources.

FILES:
load.s
armv6.h

This entry was posted in *Blog*, *Inferno OS*, *Raspberry Pi*, *Research*. Bookmark the *permalink*. *Post a comment* or leave a trackback: *Trackback URL*.

« *About Boost Multi-index Containers*   *Lab 9, coding assembler part* »