



Introduction

- Haiku : Operating System
 - **Not** GNU/Linux
 - Fast and lightweight
 - « Desktop » focussed
 - Soft realtime for multimedia
 - BeOS spirit
 - Integrated multithreaded GUI server (**not** X11)

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1);
len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm
volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec
%1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm
volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R0
0T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm
_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post
_scm m68k_vm_translation_map_init m68k_vm_translation_map_init post area m68k_vm_translation_map_init post scm m68k_vm_translatio
```



Supported Architectures

- X86 32bit 99%
 - PC (official platform)
- PowerPC 70%
 - Pegasos, Macintosh
- Motorola 680x0 20%
 - Atari Falcon (for fun)
- ARM 5%
 - Gumstix (target for Google Summer of Code)

```
static void sync_icache_030(addr_t address,size_t len){int l,off;char*p;uint32 cacr;off=(unsigned int)address&(CACHELINE-1);
len+=off;l=len;p=(char*)address-off;asm volatile("nop");asm volatile("movec %%cacr,%0":"=r"(cacr:));cacr|=0x00000004;/**/do{asm
volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec
%1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n":"=r"(p),"r"(cacr));p+= CACHELINE;} while((l-=CACHELINE)>0);asm
volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R0
0T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%srp\npmove (%0),%crp\n":"a"((uint64*)&entry));}struct m68k_vm
_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post
_scm,m68k_vm_translation_map_init_m68k_vm_translation_map_init_post_area,m68k_vm_translation_map_init_post_scm,m68k_vm_translatio
```




Google Summer of Code

- 3 months sponsored work on a Free Software project
- **6 projects** allocated this year
- ARM port is one of them

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char *p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Development tools

- Gcc4 (cross: arm-unknown-haiku)
 - No gcc2 BeOS binary compatibility unlike x86
- **U-Boot** (source code)
- Vim & nano
- **QEMU**
- Cafeine

```
static void sync_icache_030(addr_t address,size_t len){int l,off;char*p;uint32 cacr;off=(unsigned int)address&(CACHELINE-1);
len+=off;l=len;p=(char*)address-off;asm volatile("nop");asm volatile("movec %%cacr,%0":"=r"(cacr));cacr|=0x00000004;/**/do{asm
volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec
%1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n":"=r"(p),"r"(cacr));p+= CACHELINE;} while((l-=CACHELINE)>0);asm
volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R0
0T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%%srp\npmove (%0),%%crp\n":"a"((uint64*)&entry));}struct m68k_vm
_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post
_init,m68k_vm_translation_map_init_post_area,m68k_vm_translation_map_init_post_scm,m68k_vm_translation
```



Target platforms

- Gumstix verdex as QEMU machine
- Gumstix **Overo Water** (Cortex-A8) and **Tobi** boards
- Future ?
 - Armadeus
 - OpenMoko FreeRunner
 - iPhone
 - **<ad>Your product here</ad>**

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0": "=r"(cacr):); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n": "=r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n": "a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Technical choices

- Go straight for EABI, no need for old ABI compatibility
- SMP support:
 - Yes, but disabled at compile-time for now, and later on for speed depending on target cpu.
- MMU Required (no μ CHaiku)
- Use JFFS for early developments, then JFFS or ext2 + attribute layer (no BFS in U-Boot)

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```




Differences from Linux boot

- Custom 2nd stage bootloader which sets things up for the kernel (paging...)
- vmlinux (kernel) → haiku_loader
- initrd → kernel+modules.tgz
- kernel args → struct driver_settings

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Boot Loader

- `haiku_loader` : custom bootloader for our kernel and modules
- ELF or ubimage version loaded from U-Boot
- Uses U-Boot provided calls to
 - query for framebuffer, basic serial output, and tells the kernel,
 - locate volumes to load and start the kernel.

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Kernel TODO

- Handle data passed from bootloader
 - struct arch_kernel_args, platform_kernel_args
- ELF arch support (shared with haiku_loader)
- Assembler stubs:
 - atomic ops, system_time, timer, Irq, context switch, syscalls, floating point, kernel debugger support
- MMU

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char *p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr,%0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n"::"r"(p),"r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R00T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%%srp\npmove (%0),%%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post_scm,m68k_vm_translation_map_init_m68k_vm_translation_map_init_post_area,m68k_vm_translation_map_init_post_scm,m68k_vm_translatio
```



Atomic ops

- Inline assembler when possible, or irq protected
- `src/system/libroot/os/arch/arm/atomic.S`
- `atomic_set(vint32 *value, int32 newValue), atomic_get()`
- `atomic_test_and_set()`
- `atomic_add()`
- `atomic_and(), atomic_or()`
- `Atomic_*64()` 64bit versions

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char *p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R0_0T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translation
```




Interrupts and contexts

- AITC backend to implement:

- `void arch_int_{dis|en}able_io_interrupt(int irq);`
- `void arch_int_enable_io_interrupt(int irq);`
- `void arm_exception_entry(struct iframe *iframe);`
- `struct arm_cpu_exception_context *arm_get_cpu_exception_context(int cpu);`
- `void arm_set_current_cpu_exception_context(struct arm_cpu_exception_context *context);`

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char *p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R0_0T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Syscalls

- Not much choice on ARM: SWI
- Possibly Unknown Instructions, but dangerous for future cpu support

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Floating point

- Kernel uses floating point when not in interrupts
- Should be possible to emulate without trapping unknown opcodes for speed

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



MMU support

- ARM has optional functions on MMU
 - Need to abstract
- No support for MMU-less
 - But you can send a patch
 - But it's really worthless if you ask me ;-)

```
static void sync_icache_030(addr_t address,size_t len){int l,off;char*p;uint32 cacr;off=(unsigned int)address&(CACHELINE-1);
len+=off;l=len;p=(char*)address-off;asm volatile("nop");asm volatile("mvec %%cacr,%0":="r"(cacr));cacr|=0x00000004;/**/do{asm
volatile("mvec %0,%%caar\nmvec %1,%%cacr\naddq.l #4,%0\nmvec %0,%%caar\nmvec %1,%%cacr\naddq.l #4,%0\nmvec %0,%%caar\nmvec
%1,%%cacr\naddq.l #4,%0\nmvec %0,%%caar\nmvec %1,%%cacr\n":="r"(p),"r"(cacr));p+= CACHELINE;} while((l-=CACHELINE)>0);asm
volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R0
0T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%%srp\npmove (%0),%%crp\n":="a"((uint64*)&entry));}struct m68k_vm
_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post
_init,m68k_vm_translation_map_init,m68k_vm_translation_map_init_post_area,m68k_vm_translation_map_init_post_init,m68k_vm_translation
```



MMU prototypes

- `void arch_vm_aspace_swap(struct vm_address_space *from, struct vm_address_space *to);`
- `status_t arch_vm_translation_map_early_map(struct kernel_args *args, addr_t va, addr_t pa, uint8 attributes, addr_t (*get_free_page)(struct kernel_args *));`
- `void *arm_translation_map_get_pgdir(vm_translation_map *map);`
- `void arm_set_pgdir(void *rt);`

```
static void sync_icache_030(addr_t address, size_t len){int l,off;char*p;uint32 cacr;off=(unsigned int)address&(CACHELINE-1);len+=off;l=len;p=(char*)address-off;asm volatile("nop");asm volatile("movec %%cacr,%0":="r"(cacr:));cacr|=0x00000004;/**/do{asm volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n":="r"(p),"r"(cacr));p+= CACHELINE;} while((l-=CACHELINE)>0);asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R00T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%%srp\npmove (%0),%%crp\n":="a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post_scm,m68k_vm_translation_map_init_m68k_vm_translation_map_init_post_area,m68k_vm_translation_map_init_post_scm,m68k_vm_translatio
```



libroot

- glibc (antique version, work in progress to use BSD libc) + native calls + libm
- Floating point (need to copy from correct glibc version)
- Atomic ops (same as for kernel, but inline asm or syscalls)

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char *p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr,%0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0),%%srp\npmove (%0),%%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={ m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Current status (SoC midterm)

- Loader, kernel and most other components buildable (but stubbed)
- Loader loading... close to work
- Next:
 - Fill stubs in kernel and libroot
 - Drivers

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```



Future Possibilities

- Ethernet, USB, bluetooth and audio support
- Hardware accelerated graphics
- Optimized xattr enabled flash filesystem
- Custom hardware accelerated decoder media node using Cortex-A8 features
- GSM/UMTS support + people files + live queries
- `<ad>`[Your patch here!](#)`</ad>`

```
static void sync_icache_030(addr_t address,size_t len){int l,off;char*p;uint32 cacr;off=(unsigned int)address&(CACHELINE-1);
len+=off;l=len;p=(char*)address-off;asm volatile("nop");asm volatile("movec %%cacr,%0":"=r"(cacr:));cacr|=0x00000004;/**/do{asm
volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec
%1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n":"=r"(p),"r"(cacr));p+= CACHELINE;} while((l-=CACHELINE)>0);asm
volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R0
0T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%srrp\npmove (%0),%scrp\n":"a"((uint64*)&entry));}struct m68k_vm
_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post
_scm m68k_vm_translation_map_init m68k_vm_translation_map_init post area m68k_vm_translation_map_init post scm m68k_vm_translatio
```



So long, and thanks for all the fish!

- No focus shift.
- It's the current trend, let's precede it.
- Still a lot to do, but nothing can stop us.

```
static void sync_icache_030(addr_t address,size_t len){int l,off;char*p;uint32 cacr;off=(unsigned int)address&(CACHELINE-1);
len+=off;l=len;p=(char*)address-off;asm volatile("nop");asm volatile("movec %%cacr,%0":"=r"(cacr:));cacr|=0x00000004;/**/do{asm
volatile("movec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec
%1,%%cacr\naddq.l #4,%0\nmovec %0,%%caar\nmovec %1,%%cacr\n":"=r"(p),"r"(cacr));p+= CACHELINE;} while((l-=CACHELINE)>0);asm
volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry;*(uint64*)&entry=DFL_PAGEENT_VAL;entry.type=DT_R0
0T;entry.addr=TA_TO_PREA(((addr_t)rt));asm volatile("pmove (%0),%%srp\npmove (%0),%%crp\n":"a"((uint64*)&entry));}struct m68k_vm
_ops m68030_vm_ops={_m68k_translation_map_get_pgdir,m68k_vm_translation_map_init_map,m68k_vm_translation_map_init_kernel_map_post
_init,m68k_vm_translation_map_init_post_area,m68k_vm_translation_map_init_post_sem,m68k_vm_translatio
```



Questions?

```
static void sync_icache_030(addr_t address, size_t len){int l, off; char* p; uint32 cacr; off=(unsigned int)address&(CACHELINE-1); len+=off; l=len; p=(char*)address-off; asm volatile("nop"); asm volatile("movec %%cacr, %0"::"r"(cacr)); cacr|=0x00000004; /**/do{asm volatile("movec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\naddq.l #4, %0\nmovec %0, %%caar\nmovec %1, %%cacr\n"::"r"(p), "r"(cacr)); p+= CACHELINE;} while((l-=CACHELINE)>0); asm volatile("nop");}static void set_pgdir(void*rt){long_page_directory_entry entry; *(uint64*)&entry=DFL_PAGEENT_VAL; entry.type=DT_R00T; entry.addr=TA_TO_PREA(((addr_t)rt)); asm volatile("pmove (%0), %%srp\npmove (%0), %%crp\n"::"a"((uint64*)&entry));}struct m68k_vm_ops m68030_vm_ops={_m68k_translation_map_get_pgdir, m68k_vm_translation_map_init_map, m68k_vm_translation_map_init_kernel_map, post_scm_m68k_vm_translation_map_init, m68k_vm_translation_map_init_post_area, m68k_vm_translation_map_init_post_scm, m68k_vm_translatio
```