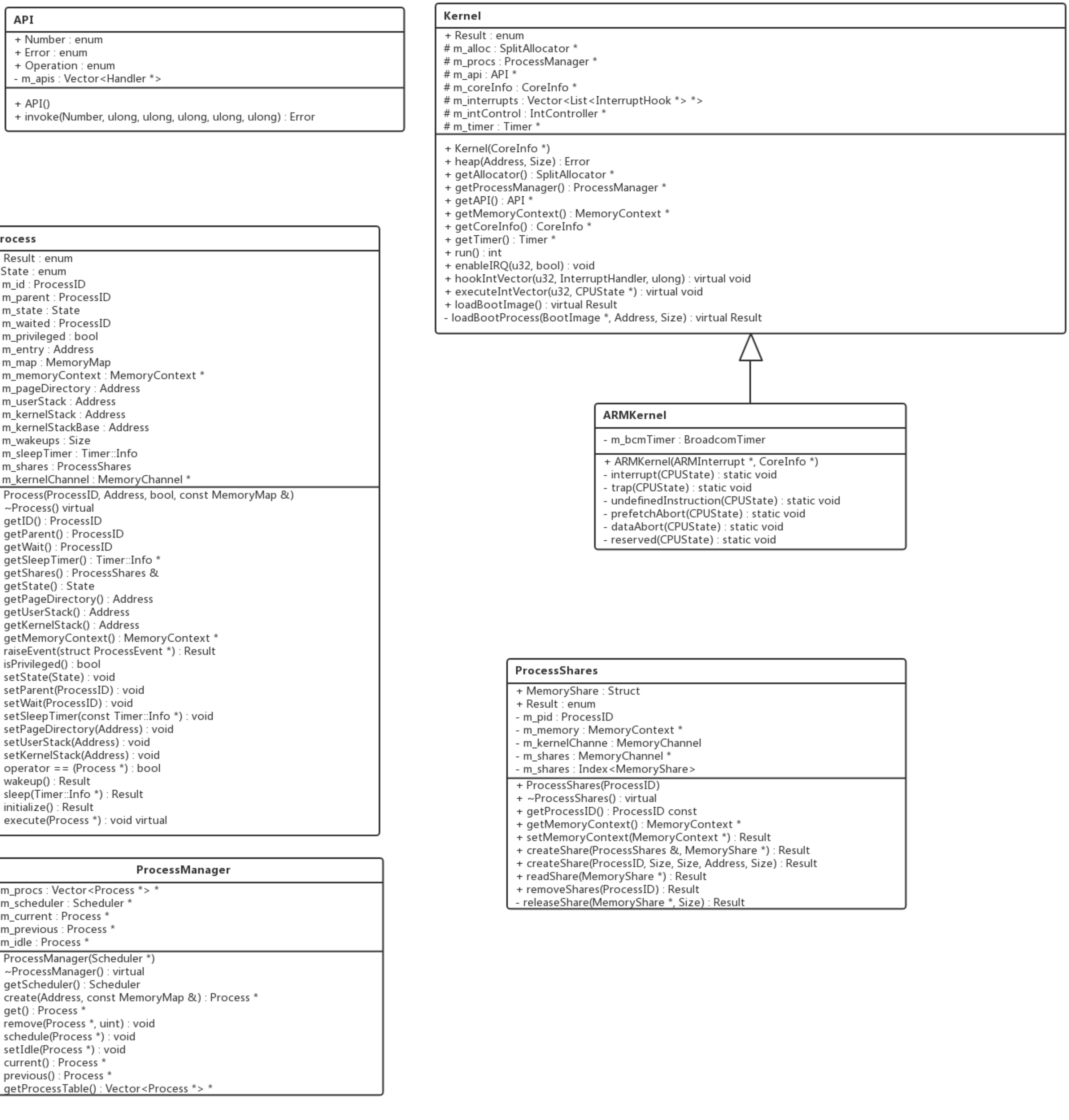
API
+ Number : enum
+ Error : enum
+ Operation : enum
- m_apis : Vector<Handler *>
+ API()
+ invoke(Number, ulong, ulong, ulong, ulong, ulong) : Error
Kernel
+ Result : enum
# m_alloc : SplitAllocator *
# m_procs : ProcessManager *
# m_api : API *
# m_coreInfo : CoreInfo *
# m_interrupts : Vector<List<InterruptHook *> *>
# m_intControl : IntController *
# m_timer : Timer *
+ Kernel(CoreInfo *)
+ heap(Address, Size) : Error
+ getAllocator() : SplitAllocator *
+ getProcessManager() : ProcessManager *
+ getAPI() : API *
+ getMemoryContext() : MemoryContext *
+ getCoreInfo() : CoreInfo *
+ getTimer() : Timer *
+ run() : int
+ enableIRQ(u32, bool) : void
+ hookIntVector(u32, InterruptHandler, ulong) : virtual void
+ executeIntVector(u32, CPUState *) : virtual void
+ loadBootImage() : virtual Result
- loadBootProcess(BootImage *, Address, Size) : virtual Result
ARMKernel
- m_bcmTimer : BroadcomTimer
+ ARMKernel(ARMInterrupt *, CoreInfo *)
- interrupt(CPUState) : static void
- trap(CPUState) : static void
- undefinedInstruction(CPUState) : static void
- prefetchAbort(CPUState) : static void
- dataAbort(CPUState) : static void
- reserved(CPUState) : static void
Process
+ Result : enum
+State : enum
m_id : ProcessID
m_parent : ProcessID
m_state : State
m_waited : ProcessID
m_privileged : bool
m_entry : Address
m_map : MemoryMap
m_memoryContext : MemoryContext *
m_pageDirectory : Address
m_userStack : Address
m_kernelStack : Address
m_kernelStackBase : Address
m_wakeups : Size
m_sleepTimer : Timer::Info
m_shares : ProcessShares
m_kernelChannel : MemoryChannel *
+ Process(ProcessID, Address, bool, const MemoryMap &)
+ ~Process() virtual
+ getID() : ProcessID
+ getParent() : ProcessID
+ getWait() : ProcessID
+ getSleepTimer() : Timer::Info *
+ getShares() : ProcessShares &
+ getState() : State
+ getPageDirectory() : Address
+ getUserStack() : Address
+ getKernelStack() : Address
+ getMemoryContext() : MemoryContext *
+ raiseEvent(struct ProcessEvent *) : Result
+ isPrivileged() : bool
+ setState(State) : void
+ setParent(ProcessID) : void
+ setWait(ProcessID) : void
+ setSleepTimer(const Timer::Info *) : void
+ setPageDirectory(Address) : void
+ setUserStack(Address) : void
+ setKernelStack(Address) : void
+ operator == (Process *) : bool
+ wakeup() : Result
+ sleep(Timer::Info *) : Result
+ initialize() : Result
+ execute(Process *) : void virtual
ProcessShares
+ MemoryShare : Struct
+ Result : enum
- m_pid : ProcessID
- m_memory : MemoryContext *
- m_kernelChanne : MemoryChannel
- m_shares : MemoryChannel *
- m_shares : Index<MemoryShare>
+ ProcessShares(ProcessID)
+ ~ProcessShares() : virtual
+ getProcessID() : ProcessID const
+ getMemoryContext() : MemoryContext *
+ setMemoryContext(MemoryContext *) : Result
+ createShare(ProcessShares &, MemoryShare *) : Result
+ createShare(ProcessID, Size, Size, Address, Size) : Result
+ readShare(MemoryShare *) : Result
+ removeShares(ProcessID) : Result
- releaseShare(MemoryShare *, Size) : Result
ProcessManager
- m_procs : Vector<Process *> *
- m_scheduler : Scheduler *
- m_current : Process *
- m_previous : Process *
- m_idle : Process *
+ ProcessManager(Scheduler *)
+ ~ProcessManager() : virtual
+ getScheduler() : Scheduler
+ create(Address, const MemoryMap &) : Process *
+ get() : Process *
+ remove(Process *, uint) : void
+ schedule(Process *) : void
+ setIdle(Process *) : void
+ current() : Process *
+ previous() : Process *
+ getProcessTable() : Vector<Process *> *