Global Varible

xmcPartitionTab

Declaration

```
//file core/kernel/setup.c
struct xmcPartition *xmcPartitionTab;
```

Description

An array of `xmcPartiton`. Length of this array is `xmcTab.noPartitions`. `xmcPartition` struct consists of the id of partition, number of virtual CPUs assigned to this partition, communication ports of the partition, consoleDev, etc.. It is the detailed representation of XtratuM partition transalted from XML files.

One of the attribute `xmcPartitionArch` is empty.

Initialization

Initialization is done using xmcparser and xml tools.

Functions

1. SetupPartitions

   xmcPartitionTab[e].noPhysicalMemoryAreas is used for print information about physical memory area.

2. CreatePartition

   Assign xmcPartition to a partition_t p, an element in partitionTab.

   For each virtual CPU, allocate one thread to partition, with flags cleared and timers allocated.

---

xmcMemRegTab

Declaration

```
//file core/kernel/setup.c
struct xmcMemoryRegion *xmcMemRegTab;
```

Description

An array of `xmcMemoryRegion`. The length of this array is `xmcTab.noRegions`, which is the number of regions of XtratuM. `xmcMemoryRegion` struct consists of the start address, size of the memory region and corresponding flags of it.

Initialization

Initialized by parser and xml tools.

Functions

1. PmmFindAnonymousPage

   Binary search for a certain address.

2. PmmFindPage

3. PmmFindArea

4. PmmResetPartition `page=&physPageTab[memArea->memoryRegionOffset][(addr-memRegion->startAddr)>>PAGE_SHIFT];` is used to find addr located page.

5. SetupPhysMM

   Create empty physical memory of size:
   `//e for one of current xmcTab.noRegions GET_MEMZ(physPageTab[e], sizeof(struct physPage)*(xmcMemRegTab[e].size/PAGE_SIZE))`
   and init spinlock for the area

---

xmcPhysMemAreaTab

Declaration

```
//file core/kernel/setup.c
struct xmcMemoryArea *xmcPhysMemAreaTab;
```

Description

An array of `xmcMemoryRegion`. The size of the array is the summ of `xmcPartitonTab[0~xmcTab.noPartitions-1].noPhysicalMemoryAreas`. Struct `xmcMemoryArea` consists of its starting address, mapped address, flags, as well as the size of this memory area.

This array is used mainly to record the memory size allocation. Array `memBlockData` is used to keep tracking the usage of memory of a `kDevice_t`.

Initialization

Initialized by parser and xml tools.

Functions

1. ReadMemBlock

   //file core/drivers/memblock.c

2. WriteMemBlock

3. SeekMemBlock

4. InitMemBlock

   VmMapPage virtual memory paging operation here

5. SetupVmMap

   //file core/kernel/arch/vmmap.c

   Set flags and initial value of `_ptdL3`

6. PmmFindPage & FindAddr & FindArea & ResetPartition

   //file core/kernel/mmu/physmm.c

   Uses xmcPhysMemAreaTab's memory region information.

7. SetupPageTable

   //file core/kernel/mmu/vmmap.c

   same as above

8. SetupPartitions

   //file setup.c

   same as above

---

xmcCommChannelTab

Declaration

```
//file core/kernel/setup.c
struct xmcCommChannel *xmcCommChannelTab;
```

Description

An array of `xmcCommChannel` with size of `xmcTab.noCommChannels`. Struct `xmcCommChannel` contains type and union of xm channel.

Initialization

Initialized by parser and xml tools.

Functions

It is used file core/objects/commports.c mainly (not considering ttnocports.c here). Only provide configuration details.

---

xmcCommPorts

Declaration

```
//file core/kernel/setup.c
struct xmcCommChannel *xmcCommChannelTab;
```

Description

Similar as above.

Initialization

Functions

xmcIoPortTab

//TODO

Declaration

```
//file core/kernel/setup.c
struct xmcIoPort *xmcIoPortTab;
```

Description

Initialization

Functions

xmcRsvMemTab

Declaration

```
//file core/kernel/setup.c
struct xmcRsvMem *xmcRsvMemTab;
```

Description

An array that keeps recording which memory region is reserved/used.

Initialization

Initialized by parser and xml tools.

Functions

1. InitRsvMem

   //file core/kernel/rsvmem.c

2. AllocRsvMem

   Use for-loop to iterat among memory obj one by one. If currrent memory's size is equal to required memory size, then mark the memory as used and return its address.

   This function is used by `GET_MEMA` and `GET_MEMAZ` functions, which are used for allocating thread, stack and page memory.

xmcBootPartTab

Declaration

```
//file core/kernel/setup.c
struct xmcBootPart *xmcBootPartTab;
```

Description

This array stores information about partition boot image address, entry point, details about the custom files. //TODO

Initialization

Initialized by parser and xml tools.

Functions

1. SetupLdr

   `xmcBootPartTab` provides image start address. The partition image mapping is setted in this function.

2. ResetPartition

   Reset page table using image handler with address `hdrPhysAddr`. Reset partition's thread (Warm and boot situation) with entryPoint in `xmcBootPartTab`.

---

xmcRswInfo

Declaration

```
//file core/kernel/setup.c
struct xmcRswInfo *xmcRswInfo;
```

Description

Not in use

Initialization

Functions

---

xmcDstIpvi

Declaration

```
//file core/kernel/setup.c
struct xmcRswInfo *xmcRswInfo;
```

Description

Ipvi stands for *Inter-Partition Virtual Interrupts*. The XM_raise_ipvi() hypercall generates an virtual interrupt to one or several partitions as speficied in the configuration file (XM CF).

Initialization

Initialized by parser and xml tools.

Functions

1. hypercall RaiseIpviSys

   This function is the implementation of XM_raise_ipvi() hypercall. The XM_raise_ipvi() hypercall generates an virtual interrupt to one or several partitions as speficied in the configuration file (XM CF). The link between the partition that generates the interrupt and the receiver partitions is specified in the channel section of the configuration file.

2. hypercall RaisePartitionIpviSys

   This function has similar implementation as above. However, the return values of them are different. Set irq pending if and only if `xmcDisIpvi[ipvi->disOffset + e] == partitionId`.

---

xmcStringTab

Declaration

```
//file core/kernel/setup.c
xm_s8_t *xmcStringTab;
```

Description

This array of string keeps the name of partitions and plans.

Initialization

Initialized by parser and xml tools.

Functions

1. hypercall GetGidByNameSys

   This function is the implementation of XM_get_gid_by_name. Returns in the identifier of an entity as defined in the configuration file by providing the entity name string.

2. SetupPct

   //file core/kernel/kthread.c

   Setup partiton ctrl requires the name of partition.

   Setup memory area requires the name of `xmcMemArea->nameoffset`.

3. file core/objects/commports.c

   Finding port is using the name of ports.

---

xmcVCpuTab

Declaration

```
//file core/kernel/setup.c
struct xmcVCpu *xmcVCpuTab;
```

Description

This array is used to store the cpuId for each partition.

Initialization

Initialized by parser and xml tools.

Functions

1. hypercall ResumeVCpuSys

   Use more informaiton about current partition’s smp vcpu. Send ipi to the cpu that is not running current thread.

2. hypercall RaisePartitionIpviSys

   Similar as above. SMP support

3. hypercall RaiseIpviSys

   Similar as above. SMP support

4. CreatePartition

5. SetupPct

   For index VCpu scheduling policy

6. ResetKThread

   Reset current thread only. If smp, then can keep scheduling on other cores.