

# Construction d'un OS pour système embarqué (2014)

1. Objectif du module
2. Plateforme matérielle
3. Organisation du module
4. Cours
5. TME
6. Documentations annexes

## Objectif du module

L'objectif de ce module est d'analyser en détail le fonctionnement d'un système d'exploitation pour système embarqué. Le cours rappelle brièvement les principaux concepts des systèmes d'exploitation, puis détaille l'implémentation d'un système d'exploitation spécifique pour des architectures matérielles de type SOC (system on chip), c'est à dire constituées d'un ou plusieurs processeurs 32 bits, de mémoire embarquée, et de différents contrôleurs de périphériques mappés en mémoire. On vise donc des systèmes embarqués autonomes capables de contrôler un équipement industriel. Le cours est organisé autour de la réalisation pratique d'un OS embarqué multi-tâches compatible POSIX.

Les TME consistent à écrire, à partir de rien ou presque, les différents composants du système d'exploitation. La progression que vous allez suivre :

- Un noyau d'OS qui virtualise le processeur, sans communication entre les tâches, sans partage de devices mais avec une HAL.
- Ajout des communication intertâches, partage de devices: mécanisme de file d'attente et événements.
- user land : ABI kernel (liste des syscall) + pthread + dietlibc + compilation séparée.

## Plateforme matérielle

Le code est mis au point en simulation sur un SOC modélisé en SystemC en utilisant la plate-forme de modélisation SoCLib, et composé de 4 processeurs mips32, d'une mémoire, d'un timer, d'un multi-tty (terminal), d'un contrôleur video, d'un contrôleur de disque et d'un dma.

Seule la ligne d'interruption 0 des MIPS est utilisée. Pour les MIPS 1 à 3, la ligne d'interruption est branchée aux timers 1 à 3. Pour le MIPS 0, la ligne d'interruption est branchée sur l'ICU. L'ICU reçoit les interruptions provenant, dans cet ordre, du timer0, des 4 TTY, du DMA et de l'IO controller. Ce sont donc les lignes d'entrée de l'ICU de 0 (timer0) à 6 pour l'IO controller.

Périphériques	Segments dans la ROM	Segments dans la RAM
TIMER_BASE 0xd3200000	KTEXT_LMA_BASE 0xbf800000	RAM_BASE 0x7F400000
TIMER_SIZE 0x00000080	KTEXT_LMA_SIZE 0x00020000	RAM_SIZE 0x01000000
ICU_BASE 0xd2200000	KDATA_LMA_BASE 0xbf820000	KTEXT_BASE 0x80000000
ICU_SIZE 0x00000020	KDATA_LMA_SIZE 0x00020000	KDATA_BASE 0x80020000
DMA_BASE 0xd1200000	UTEXT_LMA_BASE 0xbf840000	KDATA_SIZE 0x003E0000
DMA_SIZE 0x00000014	UTEXT_LMA_SIZE 0x00060000	USR_TEXT_BASE 0x7F400000
TTY_BASE 0xd0200000	UDATA_LMA_BASE 0xbf8A0000	USR_DATA_BASE 0x7F460000
TTY_SIZE 0x00000040	UDATA_LMA_SIZE 0x00020000	USR_DATA_SIZE 0X00B9F000
BD_BASE 0xd5200000	BOOT_BASE 0xbfC00000	
BD_SIZE 0x20	BOOT_SIZE 0x00001000	

La figure et le tableau représentent l'architecture du SOC et le placement des mémoires et périphériques dans l'espace d'adressage physique.

## Organisation du module

Les cours ont lieu en salle 105 couloir 65-66, le mercredi de 8h30 à 10h30. Les TME ont lieu les vendredi de 8h45 à 13h00 en salle 65-66 406.

- Franck Wajsburt (franck.wajsburt @ lip6.fr)
- Ghassan Almaless (ghassan.almaless @ lip6.fr)

## Cours

- Introduction
- Libk
- Bootloader
- HAL partie 1
- HAL partie 2
- Gestion des threads par l'ordonnanceur

## TME

- Découverte de la plateforme
- Libk
- hal-cpu
- thread

## Documentations annexes

- ?Composants SoCLib: ?TTY, ?Dma, ?Timer, ?Icu, ?FrameBuffer, ?Bloc Device.
- Assembleur MIPS (module LI312)
- Architecture MIPS (module LI312)
- Abstraction Binary Interface
- Gestion de la pile