Systèmes Embarqués 1 & 2

Classes T-2/I-2 // 2018-2019

# p.06 – MMU - Memory Management Unit

*Exercices*

**Exercice 1**

Décrivez à l'aide d'un schéma l'architecture de la MMU du µP TI AM335x.

**Exercice 2**

Concevez une interface en langage C permettant d'accéder aux opérations pouvant être effectuées par la MMU du µP TI AM335x, soit

- Initialiser la MMU avec toutes des sections "faulty" par défaut
- Enclencher/déclencher la MMU
- Configurer des régions / sections

**Exercice 3**

Concevez une application permettant de tester la fonctionnalité de la MMU.

Quelques références

- MMU - 03_cortex_a_programmer_guide.pdf, chapitre 9
- MMU - 02_cortex_a8_technical_reference_manual.pdf, chapitre 6
- CP15 - 02_cortex_a8_technical_reference_manual.pdf, chapitre 3
  - 3.2.25 c1, Control Register
  - 3.2.31 c2, Translation Table Base Register 0
  - 3.2.33 c2, Translation Table Base Control Register
  - 3.2.34 c3, Domain Access Control Register
  - 3.2.58 c10, Memory Region Remap Registers

```
// 1st level translation table: device types
//   nG     = non-Global
//   S      = Shareable
//   APX-AP = Access Permissions
//   TEX    = Type EXtension
//   NX     = Never Execute
//   C      = Cacheable
//   B      = Bufferable
//                  0        nG         S        APX       TEX       AP        P
//                  NX       C          B
#define STRONG  ((0<<18) | (0<<17) | (0<<16) | (0<<15) | (0<<12) | (3<<10) | (0<<9) | \
                 (1<< 4) | (0<< 3) | (0<< 2) | (2))
#define NORMAL  ((0<<18) | (0<<17) | (0<<16) | (0<<15) | (5<<12) | (3<<10) | (0<<9) | \
                 (0<< 4) | (1<< 3) | (1<< 2) | (2))

// memory region configuration
struct region {
  char* region_name;        // name of the region in 1MiB section
  uint32_t virtual_addr;    // virtual address
  uint32_t physical_addr;   // physical adress
  uint32_t sz;              // size of the region in MiB
  uint32_t cfg;             // configuration flags of the section building the region
};

// list of memory regions in 1MiB sections with populated I/O devices and memories
static const struct region regions[] = {
  {"sram",      0x40200000, 0x40200000,   1, NORMAL},
  {"l3_ocmc",   0x40300000, 0x40300000,   1, NORMAL},
  {"l3f_cfg",   0x44000000, 0x44000000,   4, STRONG},
  {"l3s_cfg",   0x44800000, 0x44800000,   4, STRONG},
  {"l4_wkup",   0x44c00000, 0x44c00000,   4, STRONG},
  {"mcasp0",    0x46000000, 0x46000000,   4, STRONG},
  {"mcasp1",    0x46000000, 0x46000000,   4, STRONG},
  {"usb",       0x47400000, 0x47400000,   1, STRONG},
  {"mmchs2",    0x47800000, 0x47800000,   1, STRONG},
  {"l4_per",    0x48000000, 0x48000000,  16, STRONG},
  {"tpcc",      0x49000000, 0x49000000,   1, STRONG},
  {"tptc0",     0x49800000, 0x49800000,   1, STRONG},
  {"tptc1",     0x49900000, 0x49900000,   1, STRONG},
  {"tptc2",     0x49a00000, 0x49a00000,   1, STRONG},
  {"l4_fast",   0x4a000000, 0x4a000000,  16, STRONG},
  {"debug",     0x4b100000, 0x4b100000,   1, STRONG},
  {"emif0",     0x4c000000, 0x4c000000,  16, STRONG},
  {"gpmc",      0x50000000, 0x50000000,  16, STRONG},
  {"adc_tsc",   0x54c00000, 0x54c00000,   4, STRONG},
  {"sgx530",    0x56000000, 0x56000000,  16, STRONG},
  {"sdram",     0x80000000, 0x80000000, 512, NORMAL},
  {"duplicate", 0xfff00000, 0x80000000,   1, NORMAL}, // duplicate 1st MiB of sdram
};
```