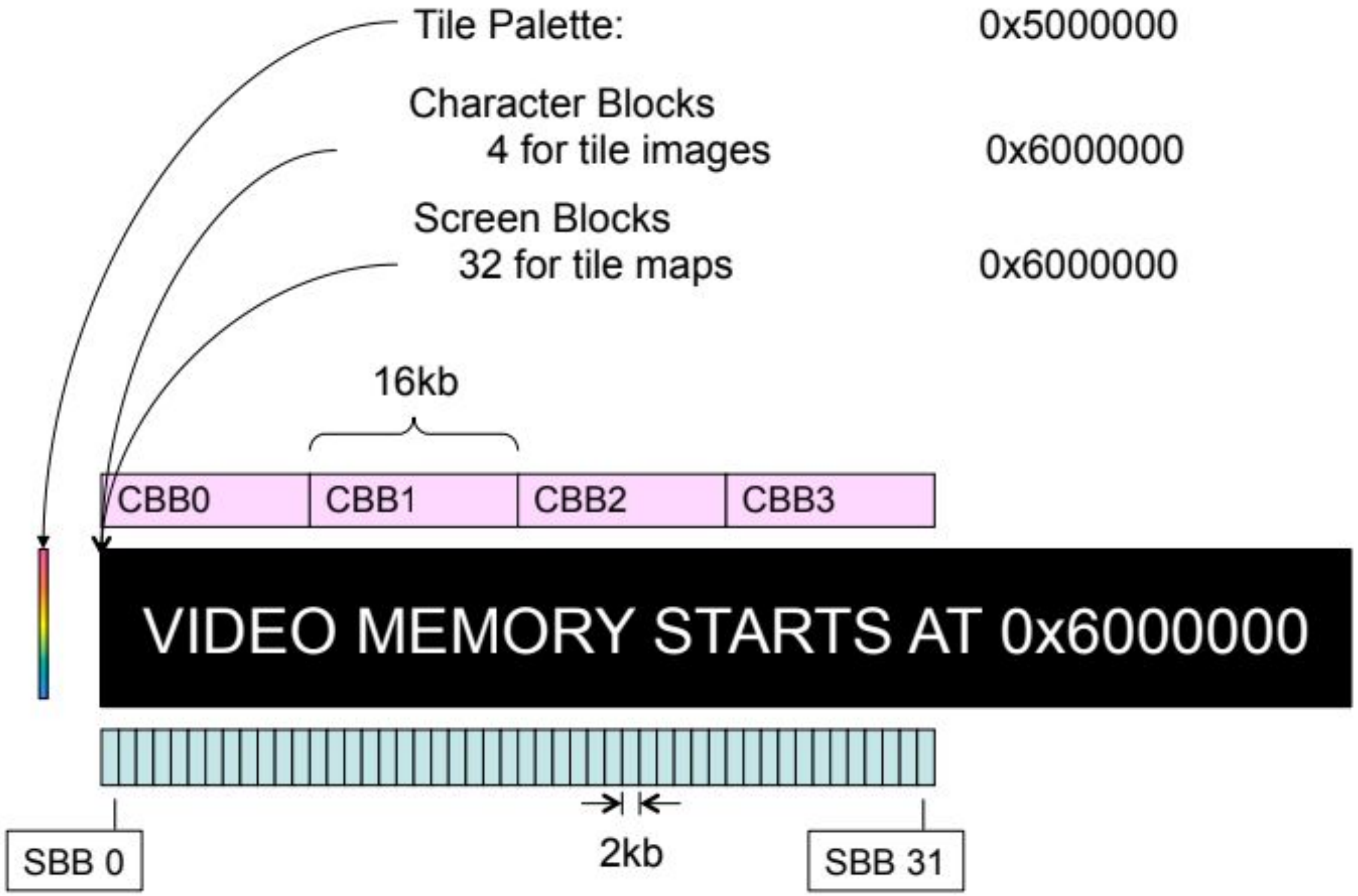
CS 2261: Media Device Architecture - Week 10

HANSEN – Oct. 22, 2018 – Georgia Institute of Technology

Overview

- Quick Quiz 3 Review
- More Mode 0
  - How to actually work with it!
  - Demo

Video Memory in Mode 0

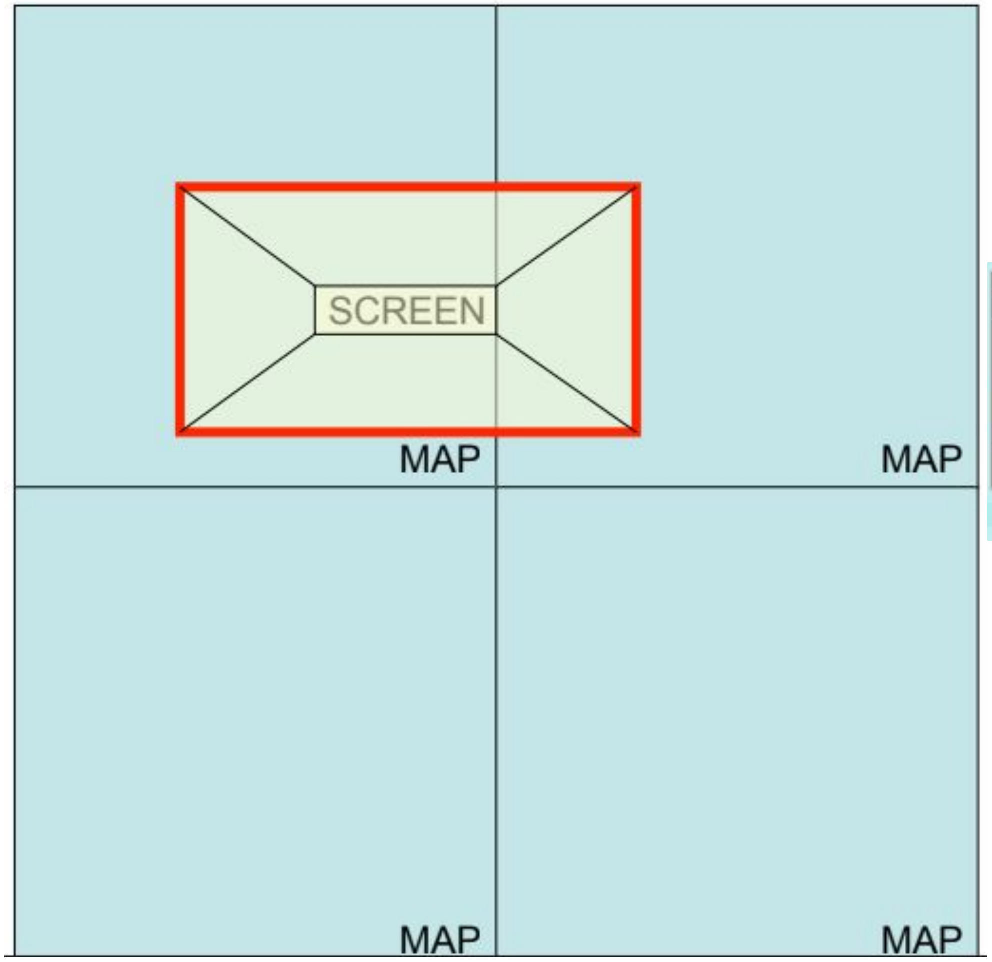


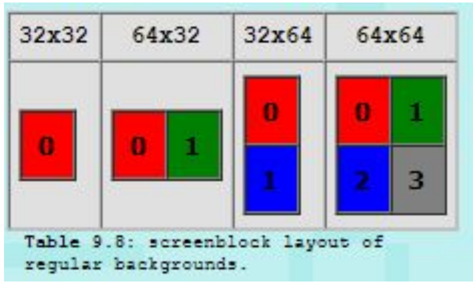
Getting the Tiles and Tilemap into Memory

- Tile images must start at the beginning of a character block
- Tilemaps must start at the beginning of a screen block
- How do we determine the correct addresses?
- How do we do any of this in C?

Larger sizes use more than one screenblock

64x64 example:





32x32	64x32	32x64	64x64
0	0 1	0 1	0 1 2 3

Table 9.8: screenblock layout of regular backgrounds.

Option 1

- Pre-calculate every screen and char block address and create a bunch of defines:

```
#define CHARBLOCK0 0x06000000
#define CHARBLOCK1 0x06004000
#define CHARBLOCK2 0x06008000
// etc.

#define SCREENBLOCK0 0x06000000
#define SCREENBLOCK1 0x06000800
#define SCREENBLOCK2 0x06001000
// etc.
```

- Then you can just use these as addresses with DMA

Option 2

- Let C do some of that work for you:
  - Each Screenblock is 2048 bytes, which you can carve into 1024 unsigned shorts

    ```
    typedef u16 SCREENBLOCK[1024];
    #define SB ((SCREENBLOCK*)0x06000000)
    ```
  - Now SB[0] will be 0x06000000,
    SB[1] will be 0x06000800, etc.
  - Also, you can manually modify a screenblock entry (if needed), at address SB[sb_num][sb_entry]

Option 2 (continued)

- Each charblock should contain tile information.
- For 4bpp tiles, you can define the following:

```
typedef struct { u16 data[16];  } TILE; // 32 bytes
```

- The for charblocks, you can define:

```
typedef TILE CHARBLOCK[512];
#define CB ((CHARBLOCK*)0x06000000)
```

  - Now CB[1] is correctly 512x16x2 (16KB) after CB[0].
  - Also CB[0][0], is the first tile in CB[0] (and can be copied to using struct copying for a single tile
    - You can also DMA in all the tiles at once.
  - Remember, CB[0] - CB[3] are the tile blocks for backgrounds (CB[4] and CB[5] are for sprites)

Working with Video Memory in Mode 0

```
// Some useful defines:
#define REG_BG0CNT *(volatile unsigned short*)0x4000008
#define REG_BG1CNT *(volatile unsigned short*)0x400000A
#define REG_BG2CNT *(volatile unsigned short*)0x400000C
#define REG_BG3CNT *(volatile unsigned short*)0x400000E

#define BG_SBB(num) ((num) << 8)
#define BG_CBB(num) ((num) << 2)
#define BG_COLOR256 (1 << 7)
                       // width x height
#define BG_SIZE0 0<<14 // 32 x 32 tiles
#define BG_SIZE1 1<<14 // 64 x 32 tiles
#define BG_SIZE2 2<<14 // 32 x 64 tiles
#define BG_SIZE3 3<<14 // 64 x 64 tiles

#define REG_BG0HOFS *(volatile unsigned short *)0x04000010
#define REG_BG0VOFS *(volatile unsigned short *)0x04000012
#define REG_BG1HOFS *(volatile unsigned short *)0x04000014
#define REG_BG1VOFS *(volatile unsigned short *)0x04000016
#define REG_BG2HOFS *(volatile unsigned short *)0x04000018
#define REG_BG2VOFS *(volatile unsigned short *)0x0400001A
#define REG_BG3HOFS *(volatile unsigned short *)0x0400001C
#define REG_BG3VOFS *(volatile unsigned short *)0x0400001E
```

Setting Registers

```
#define MODE0 0
#define BG0_ENABLE (1 << 8)

REG_DISPCTL = MODE0 | BG0_ENABLE;
REG_BG0CNT = BG_SIZE3 | BG_SBB(28) | BG_CBB(0) | BG_COLOR256;
```

Peek at Scrolling Code

```
while(1) {
  if(KEY_DOWN_NOW(BUTTON_LEFT)) hoff--;
  if(KEY_DOWN_NOW(BUTTON_RIGHT)) hoff++;
  waitForVblank();
  REG_BG0HOFS = hoff;
  REG_BG0VOFS = voff;
}

/* We can't do REG_BG0HOFS++ or REG_BG0HOFS-- because
   those registers are _WRITE_ONLY_. So we just sync
   them to two variables every frame. */
```