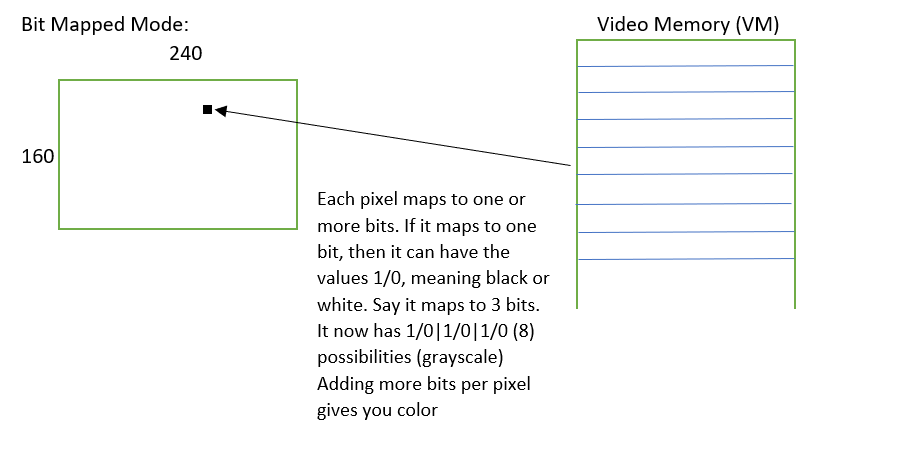
CS2261 Media Device Architecture – Lecture 2

General Overview:

* Quiz next week, Wednesday
* Majority of notes are uploaded under Resources on T-Square
* Breaks in class next week too, to stand up and stretch

Output:

* 240 x 160 pixel color video display screen (How many pixels?)
  + 38400 pixels
* 6 display modes
  + Start off with bit mapped mode
    - Each pixel corresponds directly to one or more bits in (video) memory (38400 units of memory)
    - Change pixel by changing corresponding memory unit (here we assume one “unit” is one bit
    - What would we have if each pixel mapped to a single bit?
      * Black or white
    - More bits per pixel. 2,3,…
      * 3 bits per pixel means a pixel can have how many values?



Lighting Up a Pixel: Getting a GBA to do things often involves the use of I/O registers

* Registers: Often thought of as a storage unit on a processor. A processor is like a conductor, it defines the flow of execution. It needs to store data to work, so data moves from memory to the processor, and the processor has these registers that it’ll use to store things when processing information.
  + In a GBA, register also means specific locations in memory, and server is quite overloaded in GBA

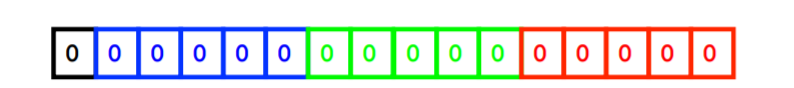
More Bits Per Pixel

* 3 bits per pixel means a pixel can have 8 values (grayscale display mode)
* For color, assign more bits per pixel
  + Group of bits represent how much red, green, and blue there is
* Grouping of bits to pixels represent display mode

Mode 3

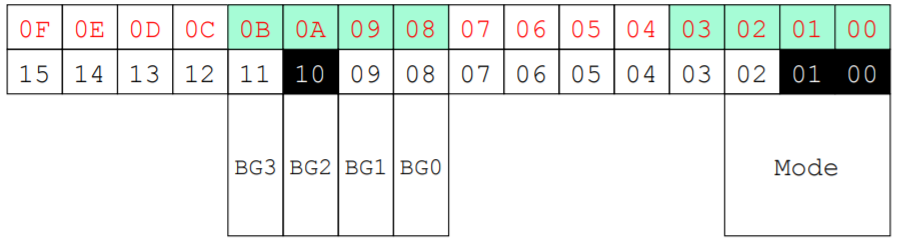
* • 16 bit quantity to describe color (216 different ones)
  + Assigned as a short int type (syntax: ‘short’)
    - The size of a short is 2 bytes, which is 16 bits
    - 16 bits per pixel, 216 possibilities
* For mode 3, GBA reserves block of memory that is 38400 shorts
  + We assign short values to places in memory, and those short values will then map to some coloration on the screen. Each pixel maps to 30400 shorts per pixel.
* Describing color in 16 bits:
  + 5 bits for blueness, 5 bits for greenness, and 5 bits for redness. These number bits 0-14. The last bit, bit 15 (the 16th bit) is always set to 0. It’s just holding a spot.
  + How many levels of each color can we have?
    - 2^5 = 32
  + How many colors?
    - 32x32x32 = 2^15= 32768

When all the bits are set to 0, they are off. Turning it on means setting it to 1:



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15th | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

* There is a register for assigning display mode, and its called the Display Control Register. It’s memory location is x4000000 (written in hexadecimal)
  + X4000000 is the address
  + Its 16 bits
  + This is a display of that register. Top row is hexadecimal representation, row below is integer representation
  + Designate mode and background prior to messing with pixels



* First three bits tell the GameBoy which mode it’s going to be in.
* Skip a few bits (3-7)
* Bits 8-11 represent backgrounds
* To set modes and backgrounds, set the 0’s there to 1’s
  + Have to give a value to the bits in this range. If we want BG2 and Mode3, we give it value 1027, which is 0100 0000 0011 in binary

Video Memory (in Mode 3)

* Starts at location 0x6000000 (address). Imagine memory as a block. The first section of the block at 0,0 coordinates on the screen corresponds to 0x6000000. We set a binary value at that address that represent the color we want. Recall that color consists of 16 bits, with the last bit always set to 0.
  + If we want the pixel at 0,0 to be white, the address 0x6000000 would hold the value 0111111111111111
    - All the colors on create white
    - 0111111111111111 in decimal is 32767
* Consists of 240 x 160 16-bit integers (short)

Video Memory (VM)



* At address x4000000, you set a value of 1027 to set the GBA to Mode 3
* Refer to slides and the posted code for how to light up a pixel by writing and executing a C program

C

* Like Java, C starts with the main function
* Its written as int main() {}
* On GBA, there is no OS system, so a return value (like in the int main()) is meaningless
* Your games never totally end, they instead continually loop
* Unsigned just means positive