CS2261 Media Device Architecture

* Lab 4 and HW3 due tonight
* Lab5 on Thursday
* Grades will be out soon

Notes:

* Direct Memory Access:
  + A super fast copy-paste circuit for your memory
  + Process:
    - Set the registers, processor is put to sleep, DMA does its copying at blinding speed, processor awoken and code execution continues
  + Needs three pieces of info: where to copy from, where to copy to, and how to do it
    - These make the source register, destination register and the control register
* DMA Source and Destination registers
  + They will hold and address, of where to copy from and to
  + They are pointers
    - So far we have only worked with int pointers and short pointers
  + But the DMA can copy anything, so it can’t be restricted by type
    - Have to say its an “anything” pointer
    - Void pointer
* Void pointer
  + The generic pointer type in C
  + It means the “address of something”
  + Ex:
    - Int a = 8;
    - Short b = 10;
    - Void \*c = &a;
    - Void \*d = &b;
  + However, since you don’t know what’s stored there, you cannot use pointer arithmetic on a void pointer
    - Pointer arithmetic requires knowing the type of the variables to work, it won’t know how many bits to jump
    - You also cannot dereference either without knowing what’s stored there
  + Solution: Casting.
    - void \*c = &a;
    - int b = \*c; // ERROR!!! CANNOT DO THIS
    - int b = \*((int\*)c); //this is okay
    - int b = \*((short\*)c);// **not** okay, will cut off info
* Constant and volatile pointers
  + You can also add const and volatile modifiers to pointers
    - Const int\*a;
    - Volatile short\*b;
  + These do not mean that the pointers are constant or volatile
    - They mean that what they are pointing to is constant or volatile
    - This is an important distinction. For example:
      * const int \*b = &a;
      * this tells you that a is supposedly a constant variable, and you cannot use b to change it.
        + A doesn’t necessarily have to be a constant pointer
  + DMA Source and destination registers are declared as const volatile void
    - Nothing can be const and vol, that doesn’t make sense so it doesn’t mean that in this case.
    - Instead, this means that whatever value the pointer is pointing to may be const or volatile or neither
    - Const volatile void\*
* DMA Control Register
  + The control register tells the dma how to do transferring
    - There are losts of diff options, compacted into a 32 bit register
  + Bits 0 – 15
    - How mnay chunks to copy
  + Bits 16 – 20 – do nothing
  + Bits 21-22 - destination adjustment
  + Bits 23 – 24 - source adjustment
  + Bits 25 – repeat (don’t worry about this)
  + Bit 26 – chunk size
  + Bit 27 - nothing
  + Bits 28-29 - timing mode (don’t worry about this)
  + Bit 30 – fire interrupt (don’t worry about this)
  + Bit 31 – start copying
* Chunk size and number
  + Bits 0 – 15
    - Control how many chunks to copy
    - Anywhere form 1 – 2^16-1
    - Giving it a 0 tells It to copy as much as possible
      * Don’t do this ever
  + Bit 26 – determines how big each copied chunk is
  + DMA is very fast, but doesn’t know what is it copying so it just does it. It copies the specified number of bits the specified number of times
  + number of transfers = number of items \* sizeof(item)/2
* Source and Destination Adjustment:
  + After each transfer, you can increment the addresses, decrement the addresses or hold them fixed
    - we going to use “source fixed destination incremented”
    - source incremented, destination decremented = reverse image
    - will later use source incremented and destination fixed for sound (music)
* GBA has four DMA channels
  + Some have special purposes
  + For general purpose transfers we use Channel 3 or else it might not work
  + All stored end-to-end in memory
* Easily Coding DMA:
  + Make a struct and declare a pointer to the starting address

typedef volatile struct {

const volatile void \*src;

volatile void \*dst;

unsigned int control;

}DMA;

DMA \*dma = (DMA \*) 0x4000000;

* DMA is good for fast contiguous copies, but not fast for a small amount of copies, can only copy contiguous memory, and copies blindly.
* Code for drawChar and drawString attached
  + The code wasn’t tested
* Text
  + Need to code out a text writer
  + Will use font.c
    - A .c file with a bunch of 1s and 0s
    - 1 means set this pixel, 0 means don’t
  + Represents a 6x8 pixel font with a complete ascii set
    - 6 columns \* 8 rows = 48 pixels a character