The C Standard:

The C standard offers many library functions which are either infeasible for us to implement or outside the scope of the project. For example, floating point support in hardware is relatively new, and not needed for our project. Of the many functions provided by the C standard, the following should be provided in our development environment:

- ctype.h
- errno.h
- stdint.h/inttypes.h
- limits.h
- stdbool.h
- stddef.h
- stdlib.h (subset)
    - RNG functions.
    - (?) Memory allocator functions.

We may choose to provide the following as well:

- assert.h
- setjmp.h
- stdarg.h
- stdio.h
    - An argument could be made for UART, or a terminal mode for the GPU.
- stdlib.h (all)
    - Numeric conversion
    - Process control
- string.h

For most of these libraries, we need only find an open implementation for ARM (or for a generic platform) and rig it up to the specs of our kernel.

System call interface:

```
/* Controller interface and defines */

/* Buttons are active low, a 0 indicates a button is pressed. */
#define BTN_B_MASK (1 << 15)
#define BTN_Y_MASK (1 << 14)
#define BTN_SELECT_MASK (1 << 13)
…
#define BTN_R_MASK (1 << 4)

unsigned int get_button_state(void);
```

Hardware design notes:

Communication:

- The Cyclone 5 supports 64 different IRQs being sent from the FPGA to the HPS over two different IRQ lines. (CV Man p. 3425)
- The AXI bridge is complicated
    - I'm gonna start taking guesses as to how it works because the CV man is wildly unhelpful.
    - The ARM has a remappable address range from 0xC0000000 to 0xFC000000, which can be used to communicate over the HPS-to-FPGA AXI bridge.
    - I *think* what happens is the FPGA receives the AXI signals with the data, and we (i.e, the people implementing things on the FPGA) have to decide what to do with those signals. If that's the case, we can basically setup our own MMIO protocol.
    - Pages of note:
        - GPV register map: 498
        - CPU address space: 485
        - CPU MMIO: 55
        - HPS-to-FPGA signals: 629
- https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_54001.pdf

Audio:

- Uses I2S for PCM audio
    - Freq of I2S is the product of the samping freq, bit depth, and number of channels. For us, this value would be 512KHz assuming we select 32KHz.
    - Samples are encoded as 2's complement and MSB first.
    - The lowest word length we can use for the DE10-Nano is 16, though we could just shift the values the user gives us a word left. Optionally, we could dedicate more ram and give a better PCM. (ADV p. 141)
    - Lowest freq we can select is 32KHz. (ADV p. 141)
    - https://www.analog.com/media/en/technical-documentation/user-guides/ADV7513_Programming_Guide.pdf
    - https://en.wikipedia.org/wiki/I%C2%B2S

Controller input (SNES):

- SNES Controller has 7 pins: | (1) (2) (3) (4) | (5) (6) (7))
    - Pins 1 and 7 are 5v and ground, respectively.
    - Pins 5 and 6 are not used in the standard controller protocol.
    - Pin 2 is the clock
    - Pin 3 is the data latch, used to start polling
    - Pin 4 is the serial data output.
- SNES Protocol is pretty simple:
    - When the receiver wants to know whats on the controllers mind, its ends a 12us (600 clock cycles at 50MHz) high pulse on pin 3.
    - The controller sees this pulse and latches its current button state.

    - 6us after the pulse finishes, the the receiver sends 16 50% duty cycle pulses at 12us per full cycle.
    - At the rising edge, the controller sends a button state bit, at the falling edge the receiver samples it. Button state is active low for pressed. Last four states are not used (always high) for the standard controller.
    - Buttons are sent in the order: B, Y, Select, Start, Up, Down, Left, Right, A, X, L, R.
- GPIO provides us with a 5v and a ground. Output pins are 3.3V, but people have used arduinos to communicate with SNES controllers, and they have the same output voltage, so it should be fine. (DE10-Nano p. 28)
- An appropriate interface seems to just have a system call that asks the FPGA what the last known controller state was. Having the program poll for input is fine, its how most interfaces work. Games can't always react to input immediately, anyway.
- https://github.com/marcosassis/gamepaduino/wiki/SNES-controller-interface
- https://gamefaqs.gamespot.com/snes/916396-super-nintendo/faqs/5395