Pre-MT CS

CALL:

C / Memory

*: dereferencer
&: referencer.

- ptr = malloc(size); allocates uninitialized block, size size @ pointer ptr in memory
- ptr = calloc(num_items, size); allocates num_items blocks of zeroed memory of size size @ pointer ptr.
- free(ptr); free memory allocated at ptr. do this if you'll never use that mem. again.
- realloc(ptr, new_size); Either
  - Expands or contracts memory allocated by ptr to new_size
  - Allocates a new block, copies the memory it can from old block, frees old block.

Alignment

- char: 1 byte, no alignment rules
- short: 2 bytes, aligned by half word
- int: 4 bytes, aligned by word
  void*

Stack vs. Heap Space

- Stack space: memory allocated on the call stack
  - most primitive data types
- heap space: memory allocated during execution
  - allocated memory
- static storage: global vars, string literals
- code storage: bytecode that comprises the code being run.

Struct:

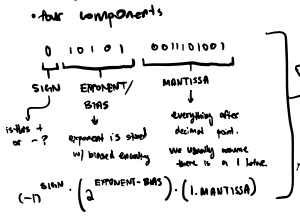
- multiple blocks of data in one structure.
- info is bunched adjacently & spaced according to alignment rules.

Unions:

- multiple data types stored in one block of memory.
- allocated to be the largest data type
- set block = 0 = set everything to 0.

Number Rep:

- **Unsigned**
  - representable range: $[0, 2^n - 1]$
  - just ignores sign, best for nonnegative-only metrics like length.
  - if we wanted signs...
- **Signed Magnitude**
  - Representable Range: $[-2^{n-1}+1, 2^{n-1}-1]$
  - one bit determining sign, rest computing ints. as normal
- **bias**
  - Representable Range: $[B, 2^n - 1 + B]$, usually $B > 0$.
  - allows us to represent neg. #'s as nonneg
  - would need to perform bias externally and it would work like unsigned.
- **2's complement**
  - Representable Range: $[-2^{n-1}, 2^{n-1} - 1]$
    - then, to repr. a negative number, flip all bits in original number & add 1 to the result.
- **Floating Point**
  - four components



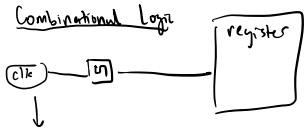
Denorm?
- Change behavior of 0b00...0 s.t. range extends to exactly 0.
  - instead of this being $1 \cdot 2^{BIAS}$, make it $0 \cdot 2^{BIAS+1}$ when exponent = 0
  - decreases precision but at least we can represent #'s < 1

∞
- If exponents are all 1's, then our value is ∞ only if mantissa = 0. Otherwise, is NaN.

Pre-Midterm (ASM)

Syntax:

Combinational Logic



clock repeatedly turns on & off. this is the clock cycle.

Registers only update once per clock cycle, when clk turns on. however, you must wait until after clk-to-q time to see updates.

What are limits to clock cycle?

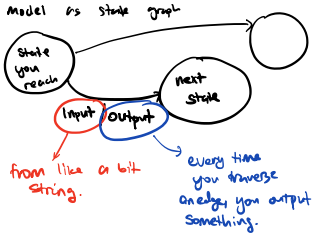
- each component in the circuit has a propagation time.
- setup time: amount of time before a clock interval that input must be stable } is a constant
- hold time: amt. of time after a clock interval that input must be stable. } depends on components.
- $t_{clk\text{-}to\text{-}q} + t_{shortest\text{-}comb\text{-}path} \geq t_{hold}$
- $t_{clk\text{-}to\text{-}q} + t_{longest\text{-}comb\text{-}path} + t_{setup} \leq t_{clock\text{-}period}$

SHORTEST comb. path is determined by finding least time btwn. two timed components. (NOT necessarily regs.!)

LONGEST comb. path is determined by finding the longest time btwn. two timed components

Finite States

- detect a bit sequence or smth
- model as state graph



Post - MT:

Pipeline; (data path in ref sheet)

- latency: delay to process operation
- throughput: # of operations done in one op.
- pipelining increases latency, but also throughput.

There are 5 stages:

- IF → instruction fetch. what's the instruction
- ID → instruction decode - figd the meaning of instruction
- EX → execute - do stuff w/ branch comp. ALU, etc.
- MEM → memory - put info into memory
- WB → write back - what to reinsert into the reg?

Must put registers btwn. stages

Parallelism: Can use the fact that CPU has multiple cores to do multiple things @ once

SIMD: does vector addition, in order to save time for array addition. should have a speed increase proportional to vector size.

from Intel AVX library.

Cache:

-