CSCE 462

SOC:

- System On a Chip

Latency vs Throughput

- latency: time to complete one operation
- throughput: operations per second
- memory has both latency and throughput

GPIO

- General Purpose Input/Output

Raspberry Pi 3

- 1.2GHz 4-core processor
- 1GB RAM
- 40 board pins, 26 are GPIO
- WiFi, Ethernet

ARM

- link register: stores return address of function
  - means that leaf function calls are faster

ARM Assembly

- instruction format: `add{cond}{s} <dest>, <lhs>, <rhs>`
  - `s`, if present, means the instruction can change processor flags (e.g. over/underflow)
- `ldr`: load word (to register)
- `str`: store word
- 16 registers: `R0-R15`, and `CPSR`
  - `R0-R3`: arguments to function call, and return value

  - `R12 IP`: Intra procedural call
  - `R13 SP`: Stack Pointer
  - `R14 LR`: Link Register
    - stores return address
  - `R15 PC`: Program Counter
    - you can read/write to this directly, it's not special (though not a good idea)
  - `CPSR`: Current Program State Register:
    - stores flags about the state of the program: negative, zero, carry, overflow, underflow, privileged mode, etc...
    - can be modified directly (want to do read-modify-write state backup first)
  - `R4-R11` and `R13 SP` are callee preserved: function should push during prologue and pop in epilogue of function
  - callee must preserve `R14 LR` if it wishes to call subroutines
  - `R12 IP` is weird. Used for libraries as stack space?
- `push`/`pop` for accessing stack
  - you can push/pop multiple things at once, but you need to reverse at the end:
    ```
    push {ip, lr, sp}
    pop {sp, lr, ip}
    ```
- `bl`: branch and link (set link register). use to call subroutines
- `bx lr`: branch to link register, to return from subroutine
- constants: `mov r1, =label`
  - can do `ldr r0, =0x523` (translates to `PC` relative load)
- literal: `mov r0, #5`, `mov r2, #0x1C`
- memory access: `ldr r0 [r0]`, `str r0 [r1, #offset]`
  - memory access is sometimes done relative to `PC`

powers of two:

- 2^{10} is 1K
- 2^{20} is 1M
- 2^{30} is 1G

SysTick timer

- 24-bit timer that counts down at bus clock frequency
- interrupts CPU when it hits 0? TODO
- control registers (memory mapped)
  - `ctrl`: control
    - controls whether counts up/down? TODO

        * can have different sources (but is always internal on raspberry pi)
    - `reload`: value to start at
    - `current`: TODO

Finite State Machine (FSM)

- Mealy vs Moore
    - Moore
    - output on states
    - output based on current state only
    - Mealy
    - output on state transitions (edges of graph)
    - output based on state and current input
- FSM can be implemented like a linked list (or other ways)

interrupts

- in ARM, each source of hardware interrupt has one bit (in some control register?)
    - hardware sets that bit to interrupt the CPU
- can set other bit to enable/disable interrupts by source