* Day 1
* Bases
  + Decimal - radix of 10 - \*101
  + Binary - radix of 2 - \*21
  + Hex - radix of 16 - \*161
  + Fractional uses -1
* ALU - Part of the computer that actually performs arithmetic & logical operators on data

Control Signals -> -> Flags

Control Signals -> **ALU** -> Flags

Control Signals ->

Operand register -> -> Result Register

* Operands for the ALU are implemented through registers
* Flags can be set such as overflow
* Integer Representation
  + 37 -> 100101
  + 37.5 -> 100101.1
  + -1101.0101 -> -13.3125
  + - is the sign
  + . is radix point
  + 8-bit word
    - Word = Fixed size data representation
    - 1 -> bit
    - 4 -> nibble
    - 8 -> byte
  + Sign-magnitude
    - 1st bit is sign, other 7 are number
      * 1=-
      * 0=+
  + 1’s Complement
    - Flip all bits to make it -
  + 2’s complement
    - Flip all bits and add 1 to make it -
* 01/11/18
* Parts of the CPU
  + ALU
  + Control Unit
  + Internal Bus
  + Cache (Potentially)
* ARM
  + Advanced RISC Machine/Acorn RISC Machine
  + RISC
* Intel 8086 led to x86 chips
* Embedded Systems
* Microprocessor VS Microcontroller
* Microcontroller
  + Computer on a chip
  + Puts RAM and ROM on single chip
  + All parts of a computer
* Microcontrollers are designed for 1 task
  + Hundreds in a car
    - One per system
* 4 basic Functions of a computer
  + Storage
  + Movement
  + Processing

----------------------------------------------------------------------------------------------------------------------------

* Assembly
* Translated into machine language by an assembler NOT A COMPILER
* Assembler
  + Each input statement corresponds to a single machine instruction or fixed sequence of instructions
  + Software utility that takes an assembly program as input and produces Object Code as output
  + Object Code
    - Binary file
  + Can be done using 1-pass or 2-pass
    - 2-pass is more common
      * 1st pass assembler is concerned with lable definitions (construct symbol table)
      * 2nd pass, each instruction is translated to appropriate binary code
  + Turned into executable code by the linker
* Assembly language vs machine language
  + NOT THE SAME
  + Machine language is directly executable by the processor
    - Contains opcode, operand references, and other bits for execution(like flags)
  + Assembly makes greater use of symbolic names, assigning names to memory locations, and specific instruction location
    - Includes statements that are not executable, but act as instructions to assembler
* Assembly programming
  + Many different instruction sets
  + Many different assemblers
  + Have to change how you code for linux, os/x, windows...
  + Many different object file formats
    - He likes .asm
  + Have to know how libraries are linked, not all linkers work the same way
* We will use
  + NASM - Netwide Assembler
  + Free small, can use many object files
* 4 elements of assembly
  + Label
    - Address into which first byte of the object code will be loaded
  + Mnemonic
    - Name of operand or function
  + Operand
    - Immediate value, register value, memory value
  + Comment
    - Most use ; as start of comment
* 01/18/18
* Test
  + Simple mult choice, FTB …
  + Simple computation
    - Addressing modes
    - Homework things
    - Short answer
    - 4 of CPU
    - Top-level function of a computer
    - 4 of computer
  + Booth's algorithm
  + Assembly
* Top-Level view of computer function and interconnection
* 4 Major components of a machine
  + Bus
  + Memory
  + CPU
  + I/O
* Registers
  + MAR - Memory Address Register
    - Specifies address in memory for next read/write
  + MBR - Memory Buffer Register
    - Contains the data to be read/written from memory
  + I/O AR - I/O Address Register
    - Specifies a particular I/O Device
  + I/O BR - I/O Buffer Register
    - Used for exchange of data between I/O and CPU
  + 4 steps in instruction cycle
    - **Fetch** - PC
    - Decode - IR
    - **Execute** - Execution Unit
    - Storage - MAR, MBR, I/O AR, I/O BR
  + Fetch Cycle
    - PC holds address of next instruction
    - Increment PC
    - Fetched instruction loaded into IR
  + 4 Basic Functions of a computer
    - Data Processing
    - Data movement
    - Data storage
    - Data control
  + Interrupts - Mechanism by which other modules can disrupt the normal processing of the processor
    - Program - Condition during execution
      * Illegal machine instruction (arithmetic/memory over/underflow, division by zero…)
    - Timer - Generated by processor
    - I/O - Generated by I/O controller
    - Hardware Failure - Power failure, memory parity error
* Busses
  + System Bus - Major bus
  + 3 functional groups
    - Data line - moves data
    - Address line - source and destination definition
    - Control line - controls access to data and address line