

Expected / Projected / Actual Class Progression

Week 1 - 2/1

- Syllabus
- What's already assigned
- Install
- Questions
- Recording

Week 2 - 2/6 2/8

- Tuesday's Recording
- Thursday's Recording
- Apple Silicon
- Windows
- Intel Mac - get the distro, get QEMU, follow instructions for Windows except use your plain old terminal instead of WSL.
- Binary
- Powers of 2 up to 216
- Signed and Unsigned Integers
- 1's Complement and 2's Complement
- Registers
 - Integer Registers w & x
 - Why Have Registers
 - * Speed of Processors Relative to RAM
 - *Up to this point was Tuesday 2/6. Thursday's class follows.*
 - Special Registers
 - * Program Counter - pc
 - * Stack Pointer - sp

Week 3 - 2/13 2/15

- Tuesday's Recording
- Thursday's Recording

- Floating Point Registers ***h***, *s*, *d*, *v* & *q*
 - *h* are half floats - not used much - are least significant half of *s*'s
 - *s* are single precision values - least significant half of *d*'s
 - *d* are double precision values - are least significant half of *v*'s
 - *v*'s are a vector of something
 - *q*'s are a single 128 bit value
- Floating Point Construction
 - Floats / Doubles are approximations
 - Normalized scientific notation
 - * Sign
 - * Exponent
 - * Mantissa
 - Single Precision - how above are implemented
 - Double Precision - how above are implemented
- Why Have Registers (Continued)
 - Steps Needed to Execute an Instruction
 - Pipelined Execution
- Aside:
 - Bit fields in C/C++
 - Unions in C/C++
- **Above this was covered Tuesday. Below this covered Thursday.**
- Special Registers (other than the *really* special registers)
 - Frame Pointer - x29
 - Link Register - x30
- How linking works - what is an object file
- Assembly Language!
 - **bl** branch with link (x30)
 - **ret** return (uses x30)
 - **and** bitwise and
 - **cbnz** compare and branch if non-zero
 - **cmp** compare (is actually a subtraction)

- `b` unconditional branch
- `.p2align` power of 2 alignment
- `.text` what comes next is code
- `.global` add "I have ____" to object file TOC
- `str`, `stp`, `ldr`, `ldp` store to memory, load from memory
- `beq` branch if the previous `cmp` is zero
- `add` add two registers together and store result in a register
- `mov` copy a value into a register
- `.end` nothing else should come after this
- `.asciz` put an ASCII string with null terminator into memory

Week 4 - 2/20 2/22

- **Tuesday**
- Assembly Language
- **Thursday**
- Assembly Language

Week 5 - 2/27 2/29

Week 6 - 3/12 3/14

Week 7 - 3/19 3/21

Week 8 - 3/26 3/28

Week 9 - 4/2 4/4

Week 10 - 4/9 4/11

Week 11 - 4/16 4/18

Week 12 - 4/23 4/25

Week 13 - 4/30 5/2

Week 14 - 5/7 5/9