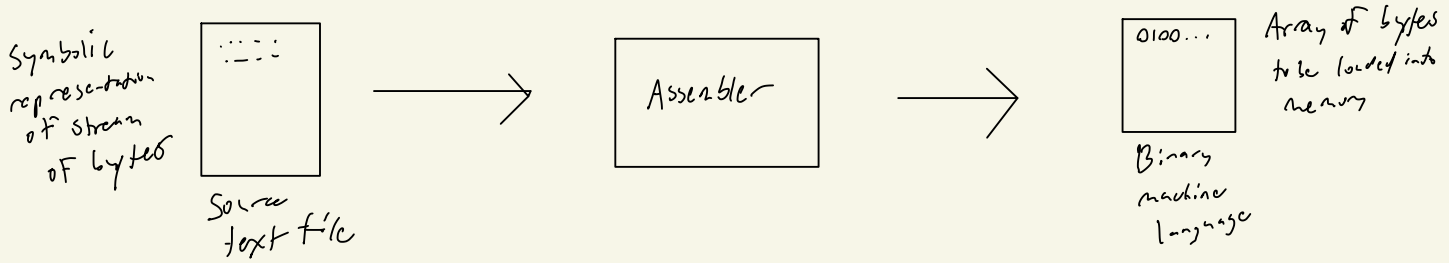

Chlo

Assembly Language



* this course uses UASM Assembly Language
  - main elements
    - comments
    - Symbols → constants
    - Labels → symbols for addressed instructions
    - Macros → instructions (expands into seq of bytes)

Assembling

1. Load predefined symbols into a symbol table
2. Read input line-by-line
   - expand macros
   - evaluate expressions

Pseudo instructions

- convenience macros that expand to one or more real instructions
- extend set of operations without adding to the ISA

USAM Expressions and Layout

- values can be written as expressions
  - ↳ assembler evaluates expressions, they are not translated to instructions to compute the value

Summary of Assembly Language

- low-level language, symbolic rep. of seq. of bytes
  - ↳ Abstracts: - bit level rep. of instructs.
    - addrs

Universality

- Universal: a set of Boolean gates is universal if we can implement any Boolean function using only gates from that set

Turing Machines

- FSM + infinite digital tape that can be read and written at each step
  - ↳ encode input as symbols on tape
    - ↳ FSM reads tape / writes symbols / changes state until it halts
      - ↳ answer encoded on tape
- TMs as integer functions: $y = TM_I[x]$

Church's Thesis

- Every discrete function computable by any realizable machine is computable by some Turing Machine:

  $f(x)$ computable $\iff$ for some y, all x $f(x) = T_y[x]$

Interpretation

- manipulate coded representations of computing machines, rather than the machines themselves

Turing Universality

- can emulate every other Turing machine
- to show that your computer is Turing Universal, demonstrate it can emulate some known UTM

✗ composition is key

Uncomputability

- no algorithm can compute $f(x)$ for arbitrary x in finite steps
- e.g.: **Halting function i**

$$f_H(y, j) = 1 \text{ if } T_y[j] \text{ halts}$$
$$0 \text{ otw}$$

↳ determines whether the y^{th} TM halts when given a tape containing j.