Aug 27th, 2019

Computer Systems

08/02/10 → Symbols with multiple meanings based on context

Computers manipulate symbols to achieve contextually useful things

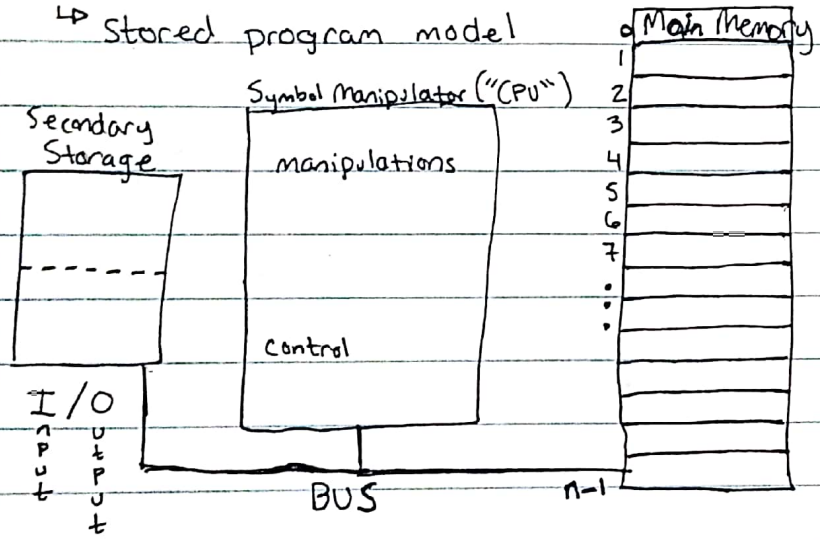
Any mutually exclusive unambiguous 2 state representation can be used as Symbols

→ Photons, electrons, punch cards, etc.

on or off, hole or no hole

John Van ____

↳ Stored program model



1. Fixed Sized Cells
2. Linear Organization
3. Address, addressable
   ↳ each cell has a unique address
   ↳ need to know the address of a cell to manipulate that cell
4. Symbol meaning in cells is determined by "context of reference"
5. Memory holds both data and instructions

"word" size → cell size (16, 32, 64, etc)

BUS can transfer one word at a time

↳ Van ____ bottleneck

10: BR 4

10 10+2=12 + 2(4)

12+8=20

Aug 27th

Computer Systems

Data → numbers, characters, boolean, (instructions)

↳ multiple ways to do any of these

↳ Primitive Types – architecture recommended representations

→ size (word size)

- - - - - - - - - -

Positional Representation of numbers

$123 \longrightarrow 3 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2$

$d_{n-1} \ldots d_2, d_1, d_0$

$\longrightarrow d_0 \cdot 10^0 + d_1 \cdot 10^1 + d_2 \cdot 10^2$

$d_{n-1} \ldots d_2, d_1, d_0 . d_{-1}, d_{-2}, d_{-3}$ Fractions

$0.123 \rightarrow 1 \cdot 10^{-1} + 2 \cdot 10^{-2} + 3 \cdot 10^{-3}$

Bin → Dec expected

Dec → Bin expected

$23_{10} \rightarrow ?_2$

base$\overline{)23_{10}}$

$2\overline{)23_{10}}$ 11 r 1 → d_0

$2\overline{)11_{10}}$ 5 r 1 → d_1

$2\overline{)5_{10}}$ 2 r 1 → d_2

$2\overline{)2}$ 1 r 0 → d_3

$2\overline{)1_{10}}$ 0 r 1 → d_4

10111 ✓

$8\overline{)23}$ 2 r 7 → d_0

$8\overline{)2}$ 0 r 2 → d_1

$27_8 = 23_{10} = 10111_2$

$0 \ldots$ maxINT

$0 \ldots 2^{\text{cell size}} - 1$

1 byte → $0 \ldots 2^8 - 1 = 255$

16 bits → $0 \ldots 2^{16} - 1 = 65535$

Aug 28th, 2019

Computation Theory

Language is our framework to work in

↳ a (possibly infinite) set of finite length strings over a finite length alphabet

String → possibly empty, finite sequence of symbols drawn from the alphabet Σ

→ ε is the empty string

→ Σ^* is the set of all possible string over an alphabet Σ

→ $|s|$ is the number of symbols in s

↳ $|\varepsilon| = 0$, $|1001001| = 7$

→ $\#_c(s)$ is the number of symbol c in s

↳ $\#_a(adam) = 2$

→ Concat → $x = ad$, $y = am$ $\quad xy = adam$

↳ $|xy| = |x| + |y|$

ε is the identity for concatenation of strings

$\forall_x (x\varepsilon = \varepsilon x = x)$

Concatenation is associative So:

$\forall_{s,t,w} ((st)w = s(tw))$

→ Replication ; repeated concatenation

$w^0 = \varepsilon$

$w^{i+1} = w^i w$

$a^3 = aaa$, $a^0 b^3 = bbb$

→ Reverse w^R

if $|w| = 0$ then $w^R = w = \varepsilon$

if $|w| \geq 1$ then

$\exists_{a \in \Sigma} (\exists_{u \in \Sigma^*} (w = ua))$

so define $w^R = a u^R$

- Relations of strings

Substrings → part of a string

proper substrings → a substring that is not the original string (doesn't include everything)

every string is a substring of itself

ε is a substring of every string

Prefix s is a prefix if

$\exists x \in \Sigma^* (t = sx)$

abba → $\underbrace{\varepsilon, a, ab, abb}_{\text{proper prefix}}$, abba

Suffix s is a suffix if

$\exists x \in \Sigma^* (t = xs)$

abba → $\underbrace{\varepsilon, a, ba, bba}_{\text{proper suffix}}$, abba

Defining a Language

finite or infinite set of strings over a finite alphabet Σ

ex. $\Sigma = \{a, b\}$

some Languages over Σ:

$\emptyset$, $\{\varepsilon\}$, $\{a, b\}$, $\{\varepsilon, a, b, aa, bb, aba\}$

Σ^* language contains every possible string combo from Σ

$L = \{x \in \{a, b\}^* : \text{all a's precede all b's}\}$

ε, a, aa, aabb, bb are in L

aba, ba, abc are not in L

$L = \{\} = \emptyset$

$L = \{\varepsilon\}$

Functions on Languages

Set operations
- Union
- Intersection
- Complement

Language Operations
- Concatenations
- kleene star (*)

if L_1 and L_2 are languages ove Σ

$L_1L_2 = \{w \in \Sigma^* : \exists s \in L_1 \ (\exists t \in L_2 (w = st))\}$

$L_1 = \{cat, dog\}$

$L_2 = \{apple, pear\}$

$L_1L_2 = \{catapple, catpear, dogapple, dogpear\}$

Language Concat

$L\{\varepsilon\} = \{\varepsilon\}L = L$

$\emptyset$ zero for concat

$L\emptyset = \emptyset L = \emptyset$

$L_1 = \{a^n : n \geq 0\}$

$L_2 = \{b^n : n \geq 0\}$

$L_1L_2 = \{a^nb^m : n, m \geq 0\}$

$L_1L_2 \neq \{a^nb^n : n \geq 0\}$

Kleene Star

$$L^* = \{\epsilon\} \cup \{w \in \Sigma^* : \exists k \geq 1 \; (\exists w_1, w_2 \ldots w_k \in L (w = w_1 w^2 \ldots w_k))\}$$

All possible combinations of the strings in a language

+ Operator

$$L^+ = LL^*$$

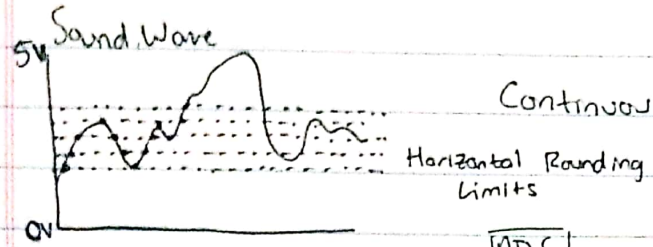
$$L^+ = L^* - \{\epsilon\} \text{ if } \epsilon \notin L$$

Decision Problems

↳ yes or no (true or false) answerable questions/problems

→ Given language L and string w, is w in L

Aug 28th, 2019
Electronics



Continuous Analog Signal

ADC

Analog to Digital Converters →

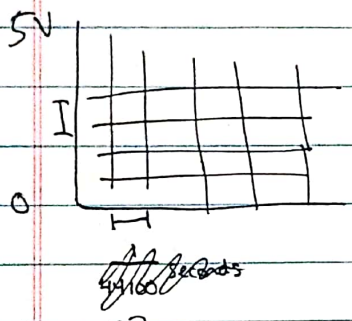
- Horizontal Line Width determined by bit length

10 bit ADC from 0 - 5 Volts

↳ $(5-0)/(2^{10})$ = Horizontal Line Width

Sample Frequency 44.1kHz (CD Audio Quality)
Sample rate

Bit Depth of CD 16-bit



44.1kHz @ 16 bits

$\frac{5-0}{2^{16}} = \frac{5}{65536} \approx 76.3\,\mu V$

$T = \frac{1}{f}$ $\frac{1}{44100} \approx 23\,\mu s$

$$V_{RMS} = \left[\frac{1}{T}\int_0^T V(t)^2 dt\right]^{1/2}$$

$$V_{RMS}^2 = \frac{1}{T}\int_0^T V(t)^2 dt$$

$$V(t) = V_0^2 \sin^2\left(\frac{2\pi}{T}t\right) \qquad f = \frac{1}{T}$$

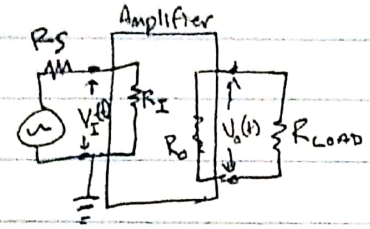
$$= V_0^2 \sin^2(2\pi f t)$$

$$V_{RMS}^2 = \frac{1}{T}\int_0^T V_0^2 \sin^2\left(\frac{2\pi}{T}t\right) dt$$

$$\frac{V_0^2}{T}\int_0^T \sin^2\left(\frac{2\pi}{T}t\right) dt$$

$$\frac{V_0^2}{T}\int_0^T \frac{1}{2}\left(1 - \cos\left(\frac{2\cdot 2\pi}{T}t\right)\right) dt$$

$$V_{RMS} = \frac{V_{PEAK}}{\sqrt{2}}$$



$$P_G = \frac{i_o V_o}{i_i V_i}$$

$$V_o(t) = V_I(t)\,A$$

↳ Amplification Factor

$5 - 1.25mA(1k\Omega) = 5 - 1.25 = 3.75$

$5\left(\frac{3}{4}\right) =$ ⟶

Current Gain $A_i = \frac{i_o}{i_i}$ $\qquad G = (A_v)^2 \frac{R_i}{R_L}$

Voltage Gain $V_i = \frac{V_o}{V_i}$

Power Gain P_o / P_i

C	0	0	0	0
A	0	1	0	1
B	0	0	1	1
	0	1	1	0

1	1	1	1
0	1	0	1
0	0	1	1
1	0	0	1

A	B	C_{IN}	C_{OUT}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Σ SOP Forms

$\bar{A}\,\bar{B}\,C$ $\quad \bar{A}\,B\,C$

$\bar{A}\,B\,\bar{C}$

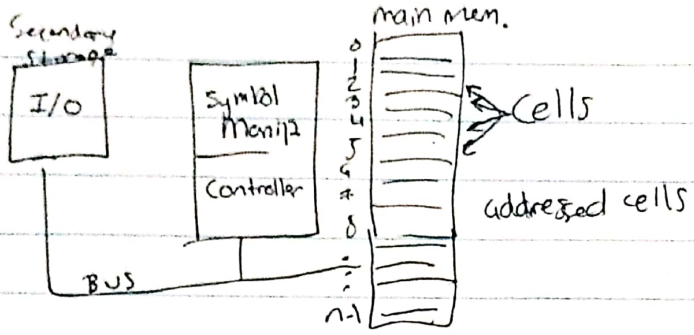
$A\,\bar{B}\,\bar{C}$

$A\,\bar{B}\,C$

$A\,B\,\bar{C}$

ABC $\quad A\,B\,C$

Aug. 29th, 2019
Computer Systems



$-3, -2, -1, 0, +1, +2, +3$ (Sign magnitude) representation

↳ write the absolute value and polarity separately

+1	−1	+1	−1
+2	−2	−2	+2
+3	−3	−1	+1

many steps to do addition

0 POS	0	0	0
1 NEG	0	1	+1
	1	0	0
	1	1	−1

two symbols for 0 and zero has a sign

↳ not good

Complement System

19	-1
18	-2
17	-3
16	-4
⋮	
03	+3
02	+2
01	+1
00	0

no longer Symetric but zero only has one

Cannot excede maxint

$-(2^{n-1}-1) \leftrightarrow +(2^{n-1}-1)$

EZ

Symmetric

Harder to manipulate

03	+3
18	-2
21	+1
% 20	✓
01	→ +1

01	+1
02	+2 ✓
03	+3
!	
19	-1
18	-2
37	-3
% 20	
17	→ -3 ✓

Aug 29th, 2019
Computer Systems

One's Complement

↳ Symetric
→ two representations for Ø
→ Ø is signed

0000	0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	0

```
   000
   0001   +1
   0010   +2
0) 0011   +3 ✓

   100
   1110   -1
   1101   -2
1) 1011   -4 ✗ END ROUND
   └──→ 1 ← CARRY
   1100   -3 ✓

   001
   0001   +1
   1101   -2
0) 1110   -1 ✓

   110
   1110   -1
   0010   +2
1) 0000      ERC
   └──→ 1
   0001   +1 ✓
```

$\bar{A}BC, A\bar{B}C, AB\bar{C}, ABC$

$A(\bar{B}C + B\bar{C} + BC) + \bar{A}(BC)$

$A(\bar{B}C + B(\bar{C}+C)) + \bar{A}(BC)$

$A(\bar{B}C + B) + \bar{A}BC$

2 ORs
3 ANDS
2 Nots

$A\bar{B}C + AB + \bar{A}BC$

3 ANDS, 2 NOTS, 1 OR

Aug 29th, 2019
Computer Systems

Two's Complement

↳ asymetric

```
  1 1 0
  1 1 1 1  -1
  1 1 1 0  -2
-------------
✗ 1 1 0 1  -3 ✓
```

```
  1 0 0
  1 1 0 0  -4
  1 1 0 1  -3
-------------
✗ 1 0 0 1  -7 ✓
```

```
  0 0 0
  0 0 1 0  +2
  1 1 0 1  -3
-------------
  1 1 1 1  -1 ✓
```

```
  1 1 1
  1 1 0 1  -2
  0 0 1 0  +3
-------------
✗ 0 0 0 1  +1 ✓
```

```
0 0 0 0  0
0 0 0 1  +1
0 0 1 0  +2
0 0 1 1  +3
0 1 0 0  +4
0 1 0 1  +5
0 1 1 0  +6
0 1 1 1  +7
1 0 0 0  -8
1 0 0 1  -7
1 0 1 0  -6
1 0 1 1  -5
1 1 0 0  -4
1 1 0 1  -3
1 1 1 0  -2
1 1 1 1  -1
```

$-(2^{n-1}) \leftrightarrow (2^{n-1}-1)$

∴ No complement has a sign bit! At best it's a sign indicater bit

Shortcut

0011 +3

↳ find 1's comp → 1100

→ add 1 → 1101 -3 ✓

BINARY CODED DECIMAL

0	0000
1	0001
⋮	⋮
8	1000
9	1001

used ~~a~~ rarely often as a means to avoid base 2 to base 10 conversion losses

BINARY	CARDINAL	SIGN MAG	1's Comp.	2's Comp
0000	0	+0	0	0
0001	1	+1	1	1
0010	2	+2	2	2
0011	3	+3	3	3
0100	4	+4	4	4
⋮				
1101	~~13~~	~~-5~~	-2	-3
1110	~~14~~	~~-6~~	-1	-2
1111	~~15~~	~~-7~~	-0	-1

ALL FINITE

↳ must stay in the bounds

Aug 29th, 19

Comp. SYS.

FLOATS/REALS

123.45

$2\overline{)123}$ 61 r1
$2\overline{)61}$ 30 r1
$2\overline{)30}$ 15 r0
$2\overline{)15}$ 7 r1
$2\overline{)7}$ 3 r1
$2\overline{)3}$ 1 r1
$2\overline{)1}$ 0 r1

1111011.0111001...

Floats are aprox.

.45 * 2

.9
.8
.6
.2
.4
.8
.6

$\bar{A} + A\bar{B} = \bar{A} + \bar{B}$

$\bar{A} + A\bar{B}$

$A(\bar{1} + \bar{B})$

$A(0 + \bar{B})$

$A(\bar{B})$

$A\bar{B}$

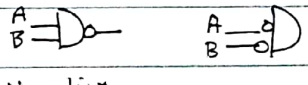
$\overline{A+B}$

	0	1
0	1	0
1	0	0

$\bar{A}\bar{B}$

	0	1
0	1	0
1	0	0

$!(A \,||\, B) \rightarrow (!A \,\&\, !B)$



Negative And

NAND

~~A B~~ A B $\bar{A}+\bar{B}$ $\bar{A}\bar{B}$ $\overline{A\bar{B}}$

$A\bar{B}$

B \ A	0	1
0	0	1
1	0	0

$\bar{A}+\bar{B}$

B \ A	0	1
0	1	1
1	1	0

$4a = 4$

~~4 4~~

$a = 1$

$6\overline{)147}$ 24 r 3 $6\overline{)24}$ 4 r 0 $6\overline{)4}$ 0 r 4

27
24

24 r 3 → 4 r 0 → 0 r 4

$6\overline{)147}$ $6\overline{)24}$ $6\overline{)4}$

12
27
24
3

$403_6 = 147_{10}$

$2\overline{)147}$ 73 r 1 $2\overline{)73}$ 36 r 1 $2\overline{)36}$ 18 r 0 $2\overline{)18}$ 9 r 0 $2\overline{)9}$ 4 r 1 $2\overline{)4}$ 2 r 0 $2\overline{)2}$ 1 r 0 $2\overline{)1}$ 0 r 1

10010011

10010011

.6
2
1.2
.4
.8
1.6
1.2
.4

.10011001100 11

$2\overline{)175}$
0 r 1
$2\overline{)1}$ r 0
$2\overline{)2}$ r 1
$2\overline{)5}$ r 0
$2\overline{)10}$ r 1
$2\overline{)21}$ r 1
$2\overline{)43}$ r 1
$2\overline{)87}$ r 1
$2\overline{)175}$

10101011_2

0xF5

10101111

0xAF

$$16\overline{)333}_{10} \quad 20 \text{ r } 13 \quad 32 \quad 13$$

$$16\overline{)20} \quad 1 \text{ r } 4$$

$$16\overline{)1} \quad 0 \text{ r } 1$$

$\boxed{0x14D}$

00010100

~~0~~ 10000111_2

111

10008111

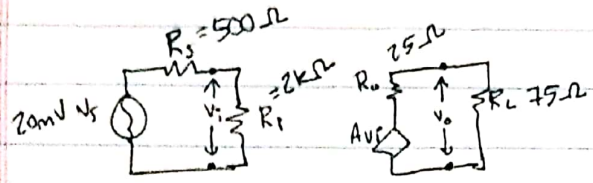
0000	0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
(1111)	-1

F

$$\begin{array}{r} 0001 \quad +1 \\ 1110 \quad -2 \\ \hline 0)\,1111 \quad -1 \end{array} \checkmark$$

Exercise 1.2

500
16
3000
5000
8000



$A = 500$

$V_s = 20mV$

$R_s = 500\Omega$

$R_i = 2k\Omega$

$R_o = 25\Omega$

$R_L = 75\Omega$

$$I_i = \frac{20mV}{500+2000\Omega} \approx \frac{1}{250} \approx 8\mu A$$

$$\therefore V_i = I_i R_i = (8_E - 6) \cdot 2000$$

$$V_i \approx 16mV$$

$$V_A = A_{V_i} = 500(16mV) = 8\ Volts$$

$$V_o = \left(\frac{75}{75+25}\right)(8V) = 6\ Volts$$

What $A_v = \frac{V_o}{V_{gi}} = ?$

$$A_v = \frac{6\ Volts}{16mV} \approx \frac{6}{.016} \approx 375$$

$A_{VS} = \frac{V_o}{V_s}$ $\quad A_i = ?$

$A_p = \frac{P_o}{P_s}$

Sep 3rd, 2019

Floating Point Representation

Computer Systems

↳ PDP-11 Based Rep

12.175_{10}

$2\overline{)12}$ 6 r 0
$2\overline{)6}$ 3 r 0
$2\overline{)3}$ 1 r 1
$2\overline{)1}$ 0 r 1

1100_2

.75 × 2 = 1.50
.50 × 2 = 1.00

$.11_2$

1100.11_2

Need to deal with both very large and very small numbers based on application

Significant digits and scale

↳ scientific notation

→ Normalization

$1100.11 * 2^0$
$110.011 * 2^1$
$11.0011 * 2^2$
$1.10011 * 2^3$
$.110011 * 2^4$

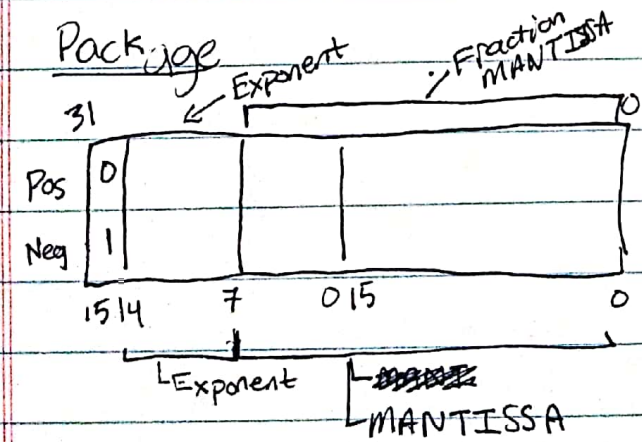
↳ normalize to a fraction with the leading decimal being a 1

0	10000100	10011000... 0
POS	EXP	MANTISSA

EXP: 4+128 = 132

→ Since the one is always there, why store it? We don't

Package



- POS/NEG
- Sig. Dig.
- exponent
  ↳ +127 — -128

256 1111 1111 +127
⋮
+2
+1
128 1000 0000 −0
−1
−2

exp+128

"Excess 128 Rep"

0 0000 0000 -128

Floats ARE APROX! Don't check for equality

$1.2 * 10^1$
$2.0 * 10^2$

10^3

10000001 129
10000010 130
100000011 259
−128
131 → 3 EXP ✓

exp1 + 128
exp2 + 128
exp1 + exp2 +

= ✓
± ✓
± CONSTANTS ✓

Take away one of the excess

All 0 exp → Rep for 0

+2 1000 0010 130
−1 0111 1111 127
256 256
−128
129 +1 ✓

$|A-B| < \varepsilon$

A	$+.11 * 2^1$	0 10000001 (1)100...0
B	$+.11 * 2^2$	0 10000010 (1)100...0
C	$+.111 * 2^2$	0 10000010 (1)110...0

View them as integers and they sort the same as they do as floats

With 16 bit words and 2 words / float
↳ ≈ 7 decimal places (single precision)
(Not PDP-11) 4 words ≈ 15 decimal places (double precision)

Booleans

All zeros is false

Anything other than zero will be true

Characters

Alphabet, Punctuation

↳ I/O contextual shapes

→ Culturally Sensative

N Characters

$\lceil \log_2 N \rceil$

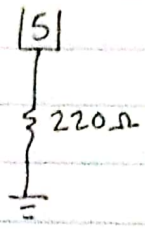
(EVERYONE ELSE) (IBM)

Standardized to ASCII and EBCDIC

Unit Record - Punch Card

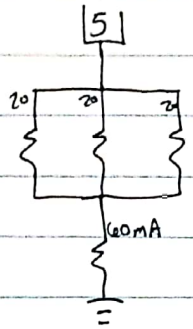
UNICODE - ISO10646

221



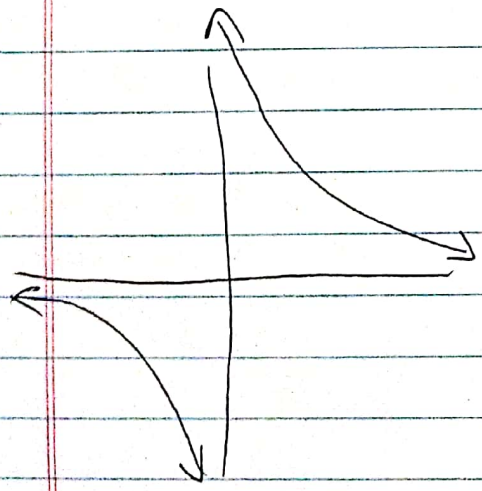
511

330



221Ω	9.8mA
511	4.5mA
680	3.5mA
330	6.9mA
909	2.7mA

.75kΩ



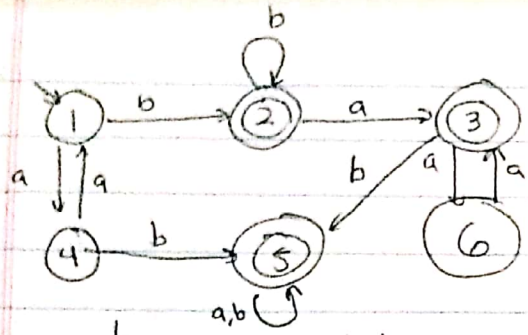
$f(x) = e^{x}$

$V = IR$

$\frac{I}{R} \quad \frac{1}{R}$

$\frac{1}{x} \qquad \frac{I}{\frac{1}{R}} = IR$

Pumpkin Drop Demo Ideas
Merch Ideas
Exp. Ideas
Long Term Projects (Semester)
External Events



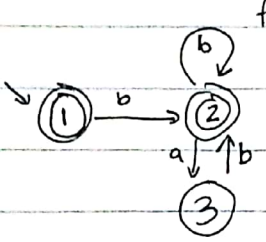
b, b_n, a

All strings of d b's with starting with or more a's followed by at l. one b

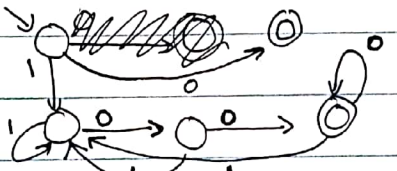
a, b a and b are accepted by this automata

a. $\{w \in \{a,b\}^* :$ every ~~aaa~~ a in w is immediately preceded and followed by b$\}$

~~start only with b~~



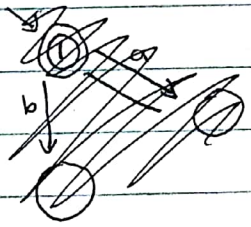
b. $\{w \in \{a,b\}^* :$ w corresponds to a binary encoding, without leading leading zeros, ~~of~~ of natural numbers that are evenly divisible by 4$\}$

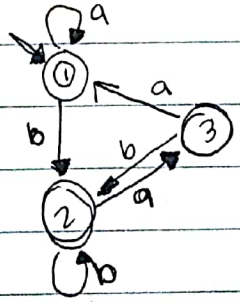


c. $\{w \in \{a,b\}^* :$ w does not end in ~~aa~~ ba$\}$

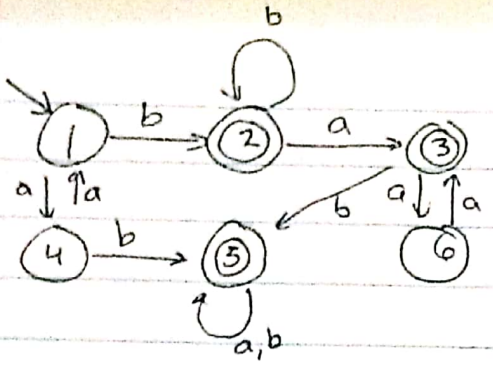
ends in b ✓

ends in a only if from an a





Even a's → at least 1 b → zero or odd a's

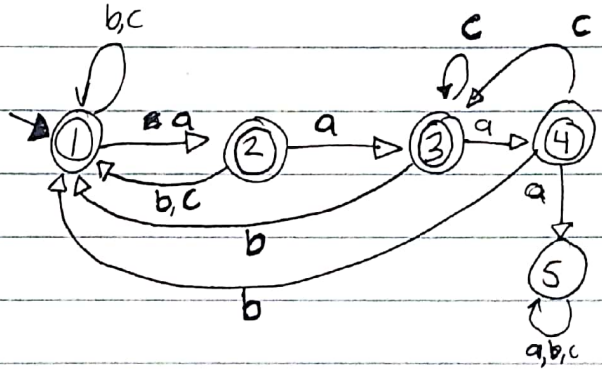


State 1 → even number of a's –

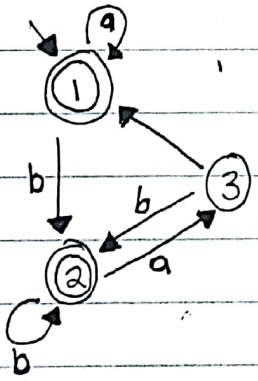
* State 4 → odd number of a's 5 → a b followed by anything

Any string that begins with an odd number of a's followed by a b followed by nothing or and combination of a's and b's

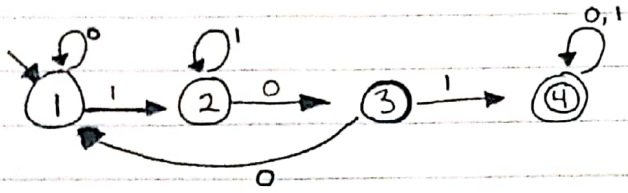
And!

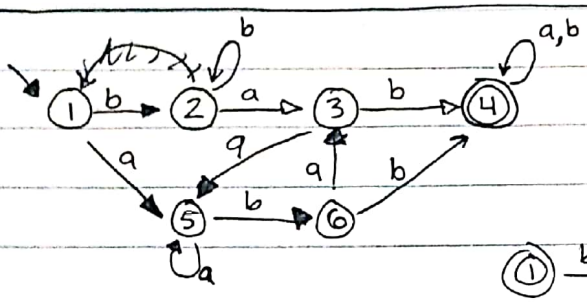


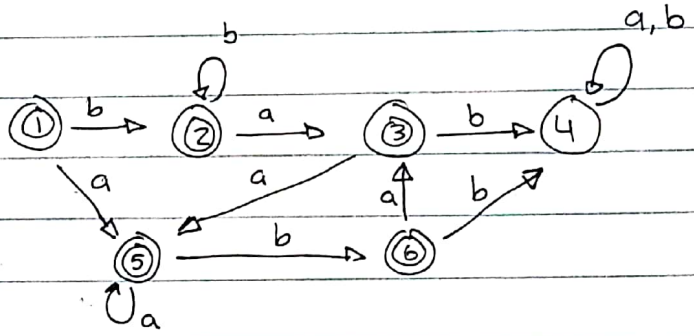
not end in ba



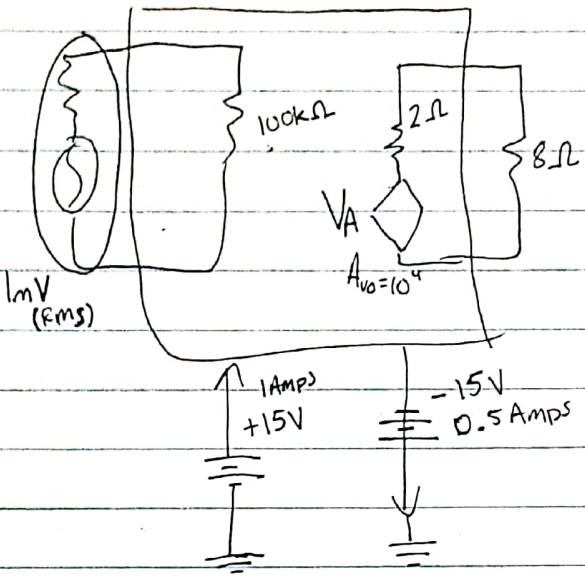
$\omega \in \{0,1\}^* : \omega$ has "101" as a substring







Amplifier efficiency



~~Power to R_L~~
~~Power from V_s~~

Value
Cost

Power from R_L

Power from V_s + 15W + 7.5W
(from + pin) (from - pin)

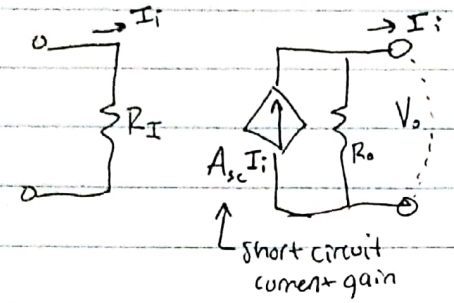
$$\frac{(1mV)^2}{100k\Omega} = P_I = 1*10^{-11} \text{ Watts}$$

$$1mV * 10^4 = 10 \text{ Volts}$$

$$V_o = 10\left(\frac{8}{8+2}\right) = 8 \text{ Volts}$$

$$\frac{V_o^2}{R_L} = P_o = \frac{64}{8} = 8 \text{ Watts}$$

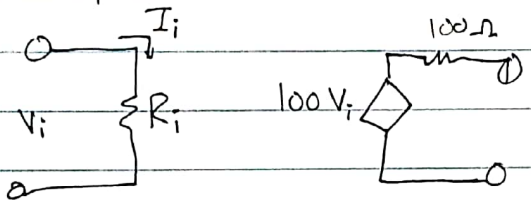
$$\text{Eff} \rightarrow \frac{8 \text{ Watts}}{22.5W + 10^{-11}W} \approx 35.5\%$$



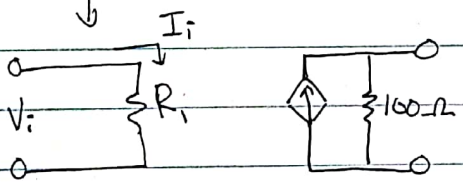
Short Circuit
Current Gain = A_{isc}

$$= \frac{i_{osc}}{i_i}$$

Example 1.5



Norton

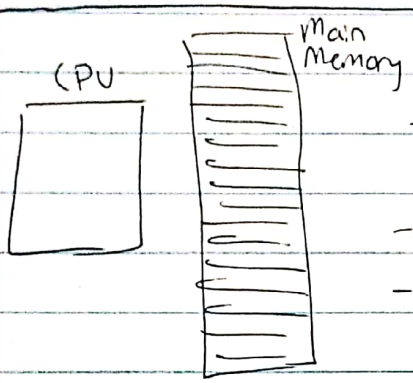


$$A_{osc} = \frac{i_{osc}}{i_i}$$

$$= \frac{100 v_i}{100\Omega} \left(\frac{1k}{V_i} \right) = 1k$$

Computer Systems

UNICODE (16 bits/char)
context (which language)
followed by 16 bit chars



each cell has an address
(On PDP-11, 16 bit addresses)
- Data
- Instructions

Instructions - bit pattern to say which operation we need to do
↳ operations code (op code)
→ add, subtract, multiply, Divide (N)
↳ number of operations determines size of opcode $\lceil \log_2 N \rceil$
→ 16 operations in 4 bit opcode
→ addresses (one output, two inputs) (M)
↳ operand # input, input, output $\lceil \log_2 M \rceil$
→ address of next instruction

4 address Instruction

OP CODE	INPUT ADDR 1	INPUT ADDR 2	OUTPUT ADDR	NEXT INSTR ADDR

Direct Access Memory - Read cells in any order
Random Access Memory - Read cells in any order and in the same amount of time regardless of last cell read

3 address instruction

→ Drop the Next INSTR ADDR to make a 3 address instruction

→ add a register to the CPU

↳ program counter (PC), Instr addr register

↳ store current instruction address in this register

↳ usually actually stores next cell

→ store next instruction directly after the current instruction

↳ Branches or jumps allow you to override the auto incrementation of the PC

---

2 address instruction

- the second input address is where the output is placed after the operation
- you lose the second input and need to copy it if you need it later

---

1 address instruction

- Load and Store
- additional "accumulator" in CPU
  - load one address
  - operate with another address
  - store with another address

---

0 address (stack address)

- make the accumulator a stack and do the op on the top two things on the stack

X = a + b * c
24 25 10 33

4AI

1. MUL b, c, X, (2)
2. ADD a, X, X, (3)

MUL 10, 33, 24, (2)
ADD 25, 24, 24, (3)

2 INST, 8 Addresses

3AI

2INST, ~~8~~ 6 addresses

2AI

1. MOV 33, 24
2. MUL 10, 24
3. ADD 25, 24

3 INSTR, 6 addresses (longer → better)

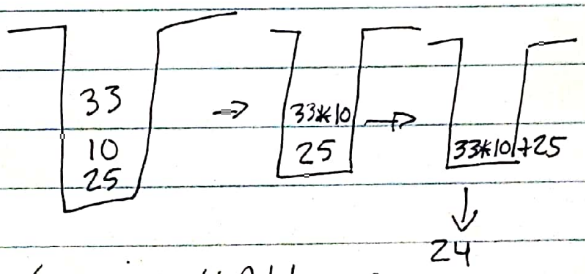
1A

1. LOAD 33
2. MUL 10
3. ADD 25
4. STORE 24

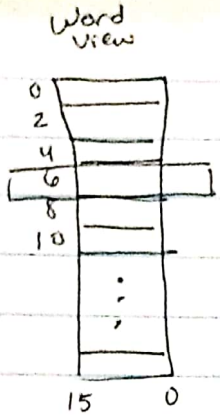
4 INSTR. 4 Addresses.

STACK

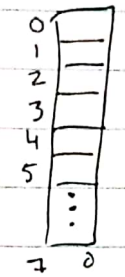
1. PUSH 25
2. PUSH 10
3. PUSH 33
4. MUL
5. ADD
6. POP 24

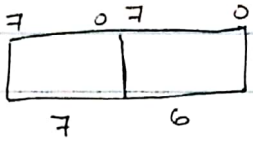


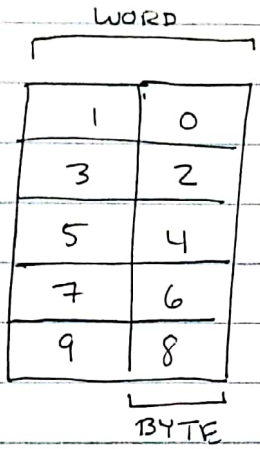
6 INSTR 4 Addresses



Word and byte addressable



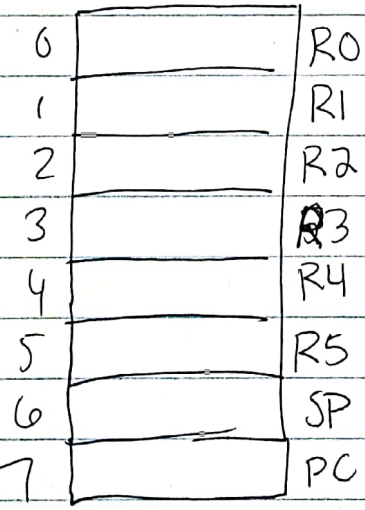




Registers are only word addressable 16 bit

BUS is 16 bits

Registers



Roles
- Accumulator
- Help form addresses

Processor Status Word (Processor Status Register)

- 16 bits → boolean status flags

N - negative Z - zero V overflow C - carry out of high order bit

~~(overflow flag)~~

$1000_2 = 4_{10}$

$101000_2 = 40_{10}$

$101000 \times 1000 = 1\,0100\,0000$ (256 128 64 32 16 8 4 2 1)

$101000 \quad 40_{10}$

$100 \quad 4_{10}$

$10100000 \quad 128_{10} + 32_{10} = 160_{10}$ ✓ (128 64 32 16 8 4 2 1)

C-AND Σ = XOR

$\Sigma = 1$ if $(A = 1, B = 0) + (A = 0, B = 1)$

if $(A != B)$

if $(A \cdot \bar{B}) + (\bar{A} \cdot B)$

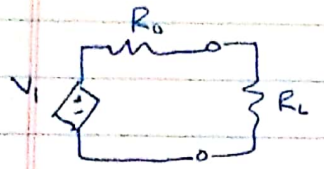
How

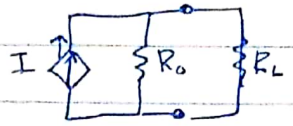
0

0

0

0





$V_i = 5$ Volts
$R_o = 100\,\Omega$
$R_L = 10\,\Omega$

$R_o = 100\,\Omega$
$R_L = 10\,\Omega$

$$V_o = V_i \left(\frac{R_L}{R_o + R_L} \right)$$

$$V_i = R_o I$$

$$5 = 100\,\Omega I$$

$$I = \frac{5V}{100\,\Omega} = \frac{1}{20}\text{ Amps} = .05\text{ Amps} = 50\text{ mA} \quad ?$$

$$V = IR$$

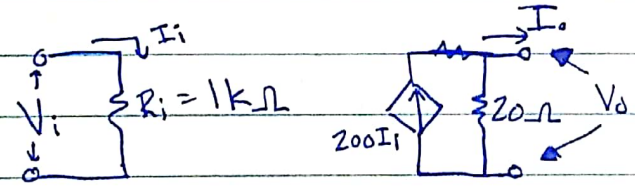
$$I = \frac{V}{R}$$

$$I = \bar{i}_o + \dot{i}_L = \frac{V_o}{R_o} + \frac{V_o}{R_L}$$

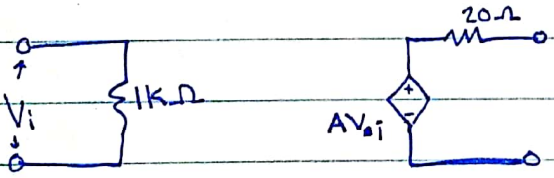
$$I = \frac{V_o(R_L + R_o)}{R_o R_L}$$

Ex 1.10

Current Amp



Voltage Amp



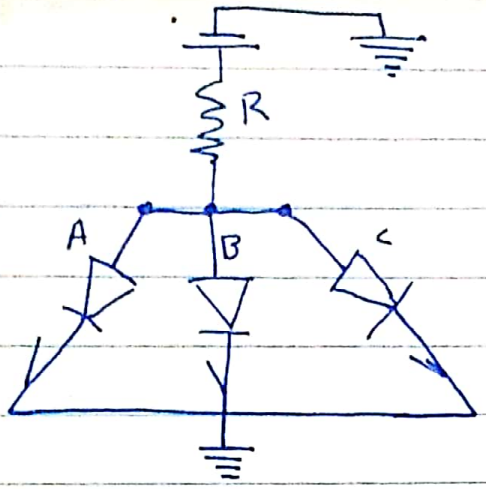
$$AV_{oi} = R_o I_o$$

$$AV_{oi} = R_o 200 I_i$$

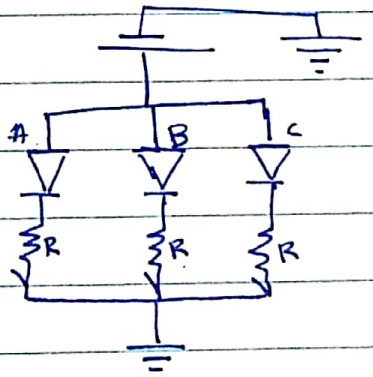
$$A_{ov} = \frac{R_o 200 I_i}{V_i}$$

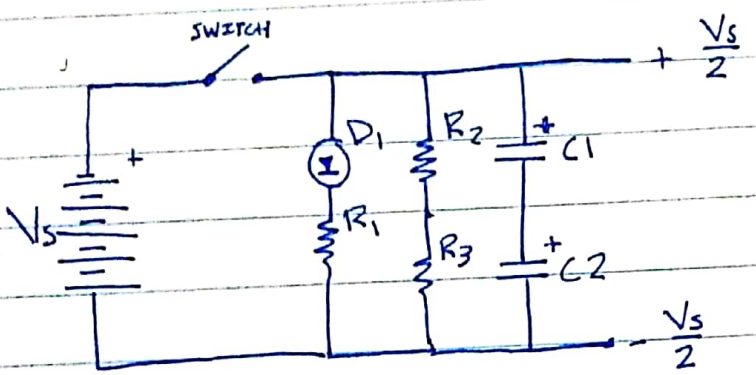
$$200 I_i = \frac{A_{ov} V_i}{R_o}$$

$$A_{ov} = \frac{200 I_i R_o}{V_i}$$



VS





0.0000011 10000000

.11 E5 101 10000101 128+4+1 = 133

S	EXP	MANTISSA
1	10000101	0000011
0	10000101	0000011

$\frac{1}{10B}$

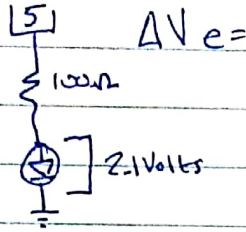
W.D

10.10011001

10.1001

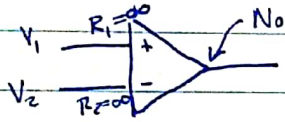
10.101D

[5] $\Delta V_e = \Delta_\mu = \frac{1234 e V_{nm}}{\lambda nm}$



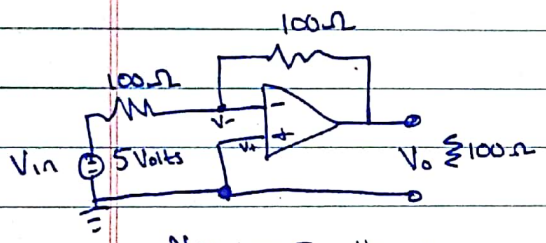
$2.1V \cdot e = \frac{1234}{\lambda}$

$\lambda = \frac{1234}{2.1} = 587 nm$



$V_o = A_{OL}(V_1 - V_2)$

No input current



$V_- = V_+$

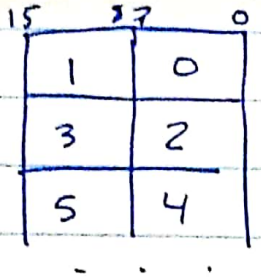
No Current in or out of V_- or V_+

Negative Feedback

$i_1 = i_2$ $\quad \frac{V_{in} - 0}{R_1} = \frac{0 - V_o}{R_2} \rightarrow \frac{V_{in}}{R_1} = \frac{-V_o}{R_2}$

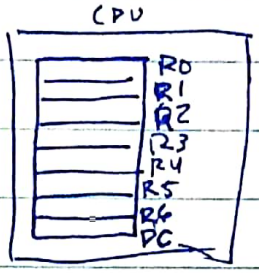
$V_o = \frac{-V_{in} R_2}{R_1}$

$$\boxed{V_o = -\frac{R_2}{R_1} V_{in}}$$



PDP-11 is word and byte addressable

Word addressable Registers



General Purpose Registers

Help form addresses
↳ some times index registers

Processor Status Word

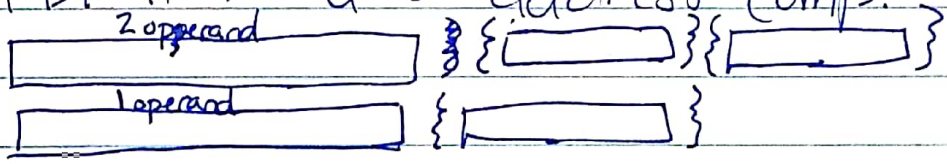
R7 → Program Counter

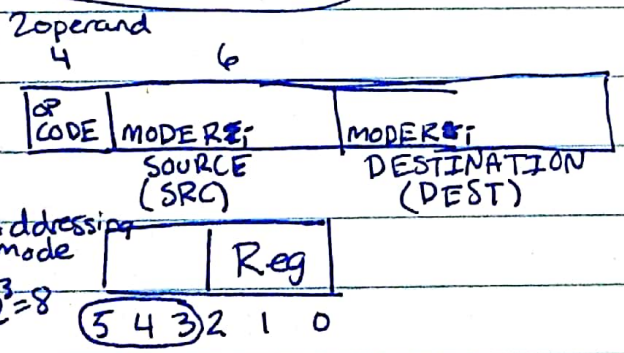
Register (16 bit) for cpu status

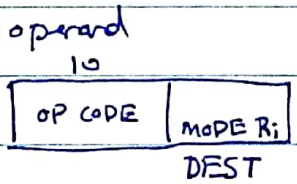
"N Z V C" boolean status bits (T-1 F-0)

- N Negative
- Z Zero
- V Overflow
- C Carry

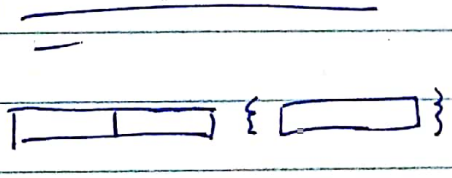
PDP-11 is a 2-address comp.







Extra words for extra info



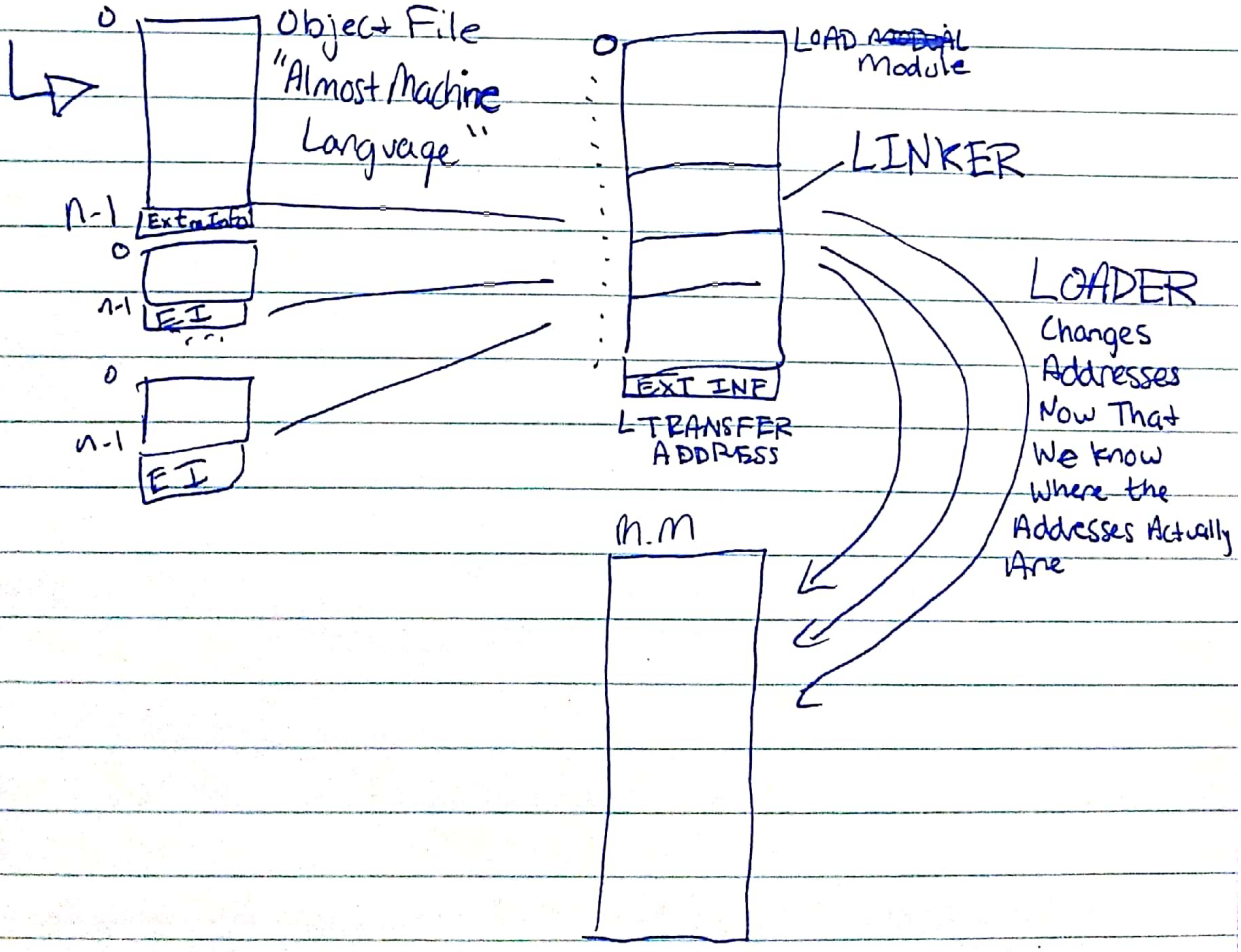
High Level Language: C/C++ A = B+C

Low Level Language: Assembler MOV 10,20

↓ substitution via "assembler"

As Low As You Can Go: Machine Language

Compile - "Translate" from higher level language to the Machine Language of the machine



HLL Variables

A = B + C

Cell of memory has address and contents. That's it.

Low No Variables!

ADD B, C

Addresses

HLL → contextual data types determine the operand type

+ → addition, concat, etc.

LLL → Operater decides the type of the operands

MACRO-11 TWO PASS MACRO ASSEMBLER (READ FILE TWICE)

{Label:} Operator {Operand(s)} {;comments}

Col1 ↳ Statement must be one line upto 127 chars (Col9, Col17, Col33)

↳ one statement per line

seperate with a comma

At least 1 space after operands

Use a fixed sized font so things line up

Operators

"ADD"

Directives → tell the software to do something. "Save 10 words of memory for me to do"

+ R0, R1, R2...

↳ Predefined Symbols

USER Defined Symbols MOV A, B

Symbol Tables

A-Z 0-9 . $

Only the first 6 chars count

10 → defaults to base 8

10. → . is directive to signify base 10

A

No operator precedence — <...> grouping directive