2a.) $w \in \{a,b\}^*$: every a in w is immediately preceded and followed by a b

$(b \cup bab)^*$

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

$\varepsilon \cup (a \cup b)^*(b \cup aa)$

c.) $w \in \{0,1\}^*$: $\exists y \in \{0,1\}^*$ ($|xy|$ is even)

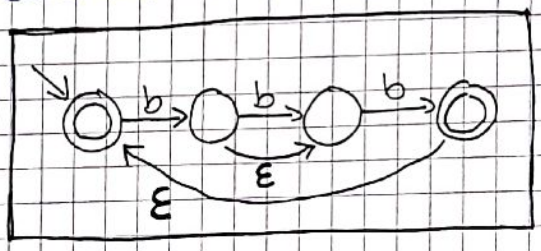
assuming $|xy|$ means $|w \cdot y|$, $\{0 \cup 1\}^*$

7.) Use Kleene's Theorm to construct FSm of the following

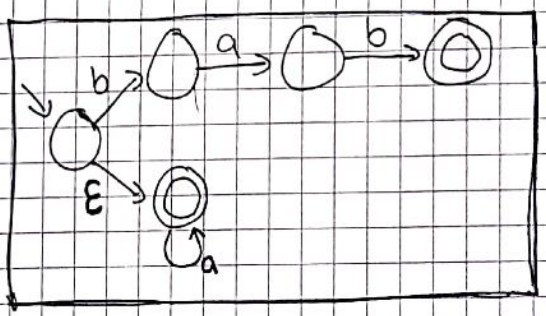
a) $(b(b \cup \varepsilon)b)^*$

b) $bab \cup a^*$

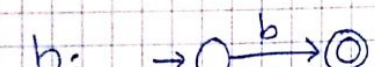
a)

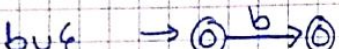


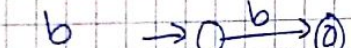
b)

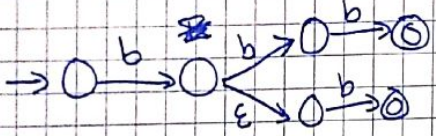


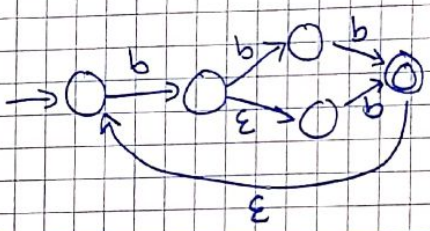
$(b(b\cup\varepsilon)b)^*$

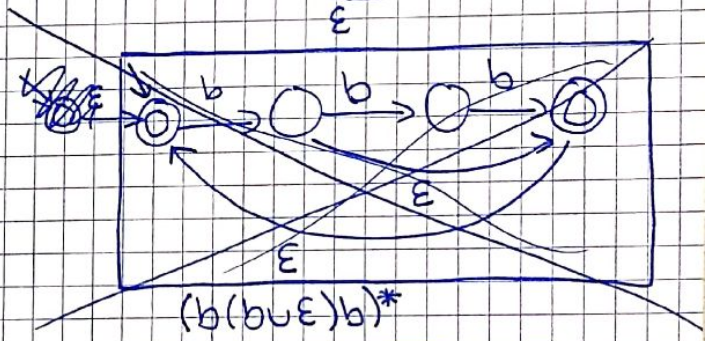
b: 

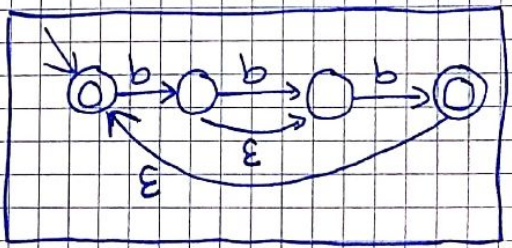
$b\cup\varepsilon$ 

b 

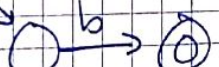


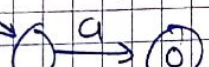


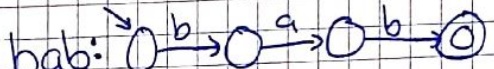


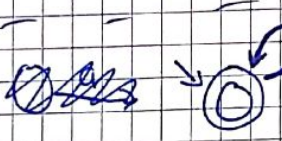


babua*

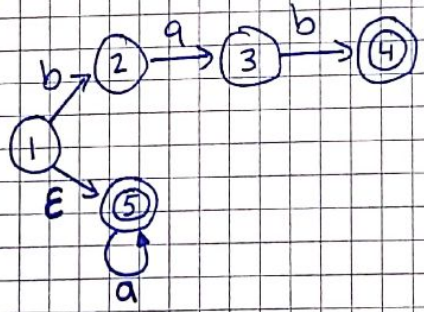
b: 

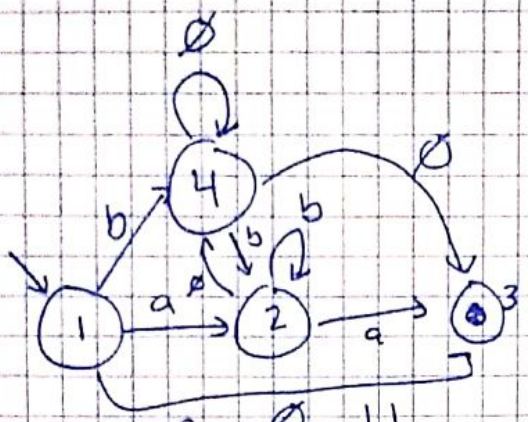
a: 

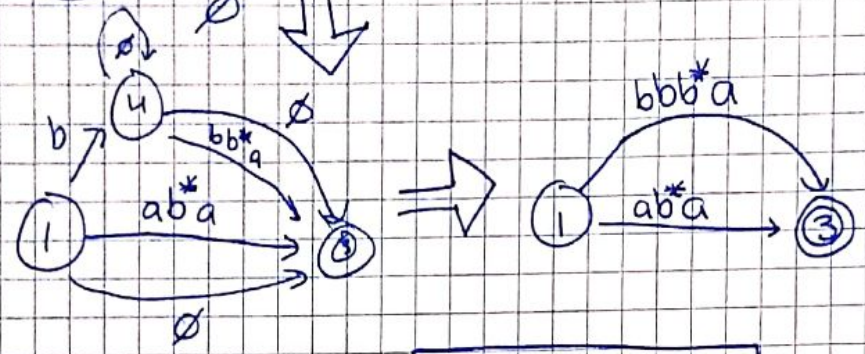
bab: 

a*: 

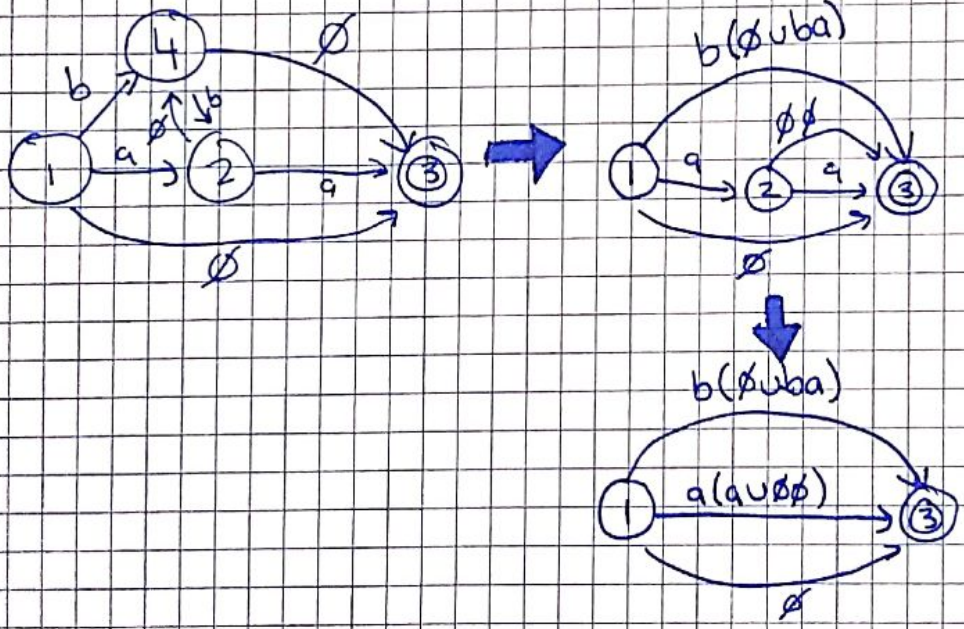
babua*







$$(bb \cup a)(b^*a)$$



$$\emptyset \cup a(a \cup \emptyset\emptyset) \cup b(\emptyset \cup ba)$$

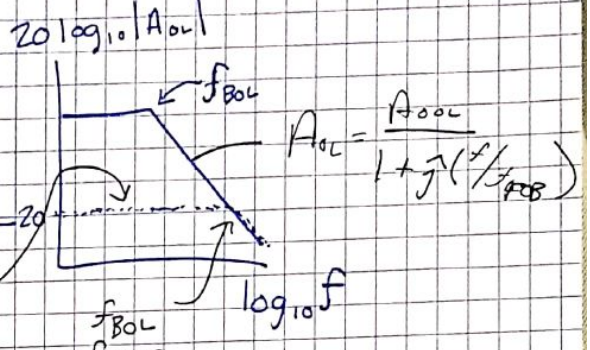
$20\log_{10}|A_V| = 110$

$\log_{10}|A_V| = \frac{110}{20}$

$10^{\frac{110}{20}} = |A_V| = \boxed{316227.76?}$

$\frac{\beta A_{OL} + 1}{A_{OL}}$

$20\log_{10}|-10| = 20dB$



$20\log_{10}(\quad)$

$A_{OCL} = \frac{A_{OOL}}{1 + \beta A_{OOL}}$

f_{BCL} of our selected gain

$f_{BCL} = f_{BOL}(1 + \beta A_{OOL})$

$V_o = i R_2 + \beta V_o$

$[V_S = V_i + \beta V_o] \rightarrow V_S = \frac{V_o}{A_{OL}} + \beta V_o$

$V_o = A_{OL} V_i$

$V_S = V_o\left(\beta + \frac{1}{A_{OL}}\right)$

Closed Loop Gain

$A_{CL} = V_o / V_S$

$A_{CL} = \frac{1}{\beta + \frac{1}{A_{OL}}} = \frac{A_{OL}}{\beta A_{OL} + 1}$

$\frac{\frac{A_{OOL}}{(1+\beta A_{OOL})} + j(f/f_{BOL})}{(1+\beta A_{OOL})}$

$= \frac{(A_{OOL}/(1+\beta A_{OOL}))}{1 + j(f/f_{BOL}(1+\beta A_{OOL}))} = A_{OCL}$

f_{BCL}

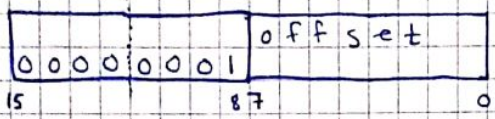
$A_{CL} = \frac{\frac{A_{OOL}}{1 + j(f/f_{BOL})}}{1 + \beta\left(\frac{A_{OOL}}{1 + j(f/f_{BOL})}\right)}$ $\quad 1 + j(f/f_{BOL})$

$= \frac{A_{OOL}}{1 + j(f/f_{BOL}) + \beta A_{OOL}}$

Branches

↳ Unconditional

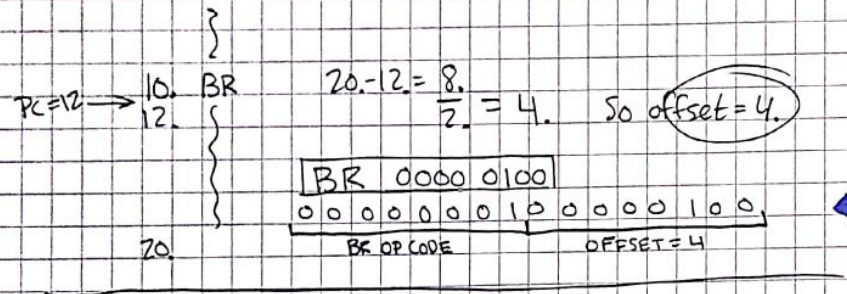
BR address

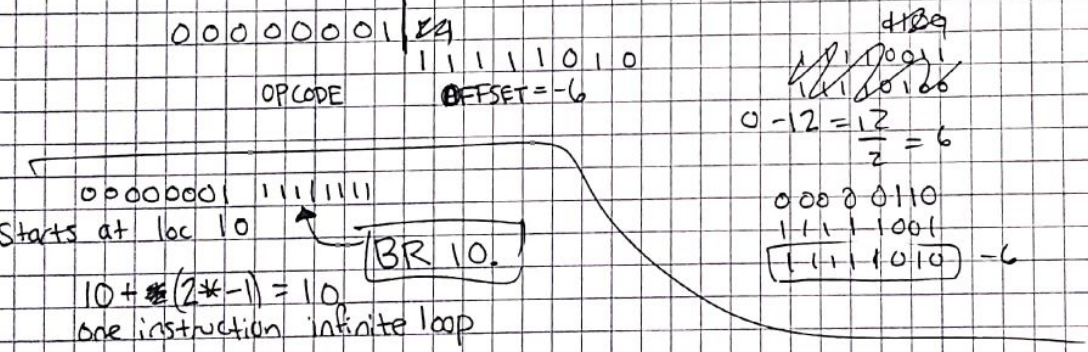


(PC) ← (PC) + 2*(offset)

↳ 8 bit 2's comp int.

Does NOT change condition codes





FORTRANESC

IF I Then 10 (Neg), 20 (Zero), 30 (Pos) ← labels

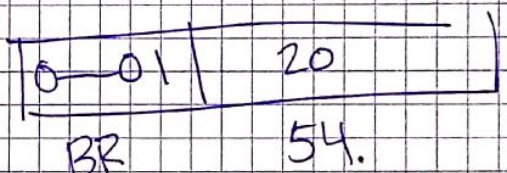
So to do >, <, = we can do

IF (I-J) then 10, 20, 30

I=J → 0 → GOTO 20
I>J → >0 → GOTO 30
I<J → <0 → GOTO 10

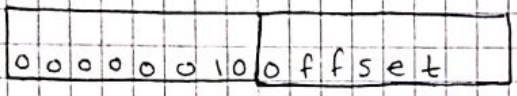
```
MOV I,IMP ; Copy I
SUB J,I   ; I-J
```

Z=1 means (I)=(J)
N=1 means (I)<(J)
N=0 means (I)≥(J)



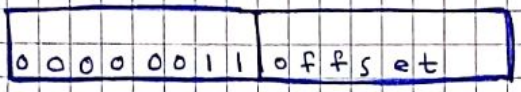
BRANCH NOT EQUAL

BNE address



(PC) ← (PC) + 2* offset if z=0

BEQ



(PC) ← (PC) + 2* offset if z=1

```
TEMP:   BLKW    2
    I1: BLKW    1       ;IN 2'S COMP
    I2: BLKW    1       ;IN 2'S COMP
    I3: BLKW    1       ;IN 2'S COMP

ISMT:   BLKW    1       ;OUT 2'S COMP SML INPUT
    ABOVE ALL OF THIS PUT YOUR INSTRUCTIONS

START   :
```

100dB

$20 \log_{10}|A_v| = 100dB$

$\log_{10}|A_v| = 5dB$

$A_v = 10^5$

$20 \log A_{OL} = 110$

$\log A_{OL} = \frac{11}{2}$

$A_{OL} \approx 316227.77$

$(0 \cup 10)^* 1^*$

0 :

1 :

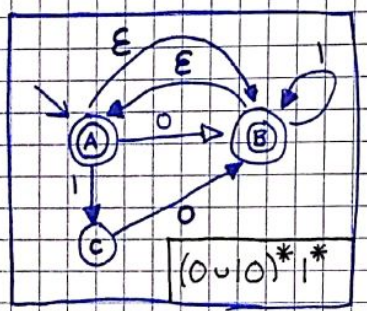
10 :

1^* :

$(0 \cup 10)$:

$(0 \cup 10)^*$:

$(0 \cup 10)^* 1^*$:



$$Y = D\overline{(C \cdot \overline{(A+B)})}$$

$$\bar{D} + \overline{(C \cdot \overline{(A+B)})}$$

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

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

$y = \bar{A}\bar{B} + AB\bar{C} + A\bar{B}$

~~$B(\bar{A} + AB\bar{C} + A)$~~

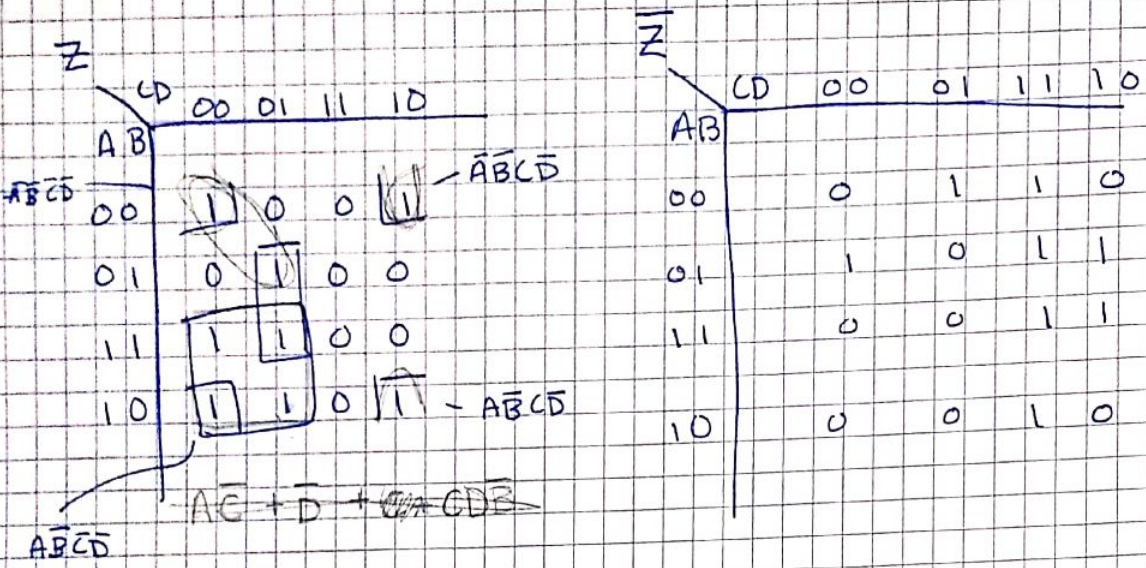
~~$\bar{B}(A + A\bar{C} + A)$~~

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

~~$B(C) = B\bar{C}$~~

~~$B(\bar{A} + AB\bar{C} + A)$~~

~~$B + B\bar{C}$~~



$A\bar{C} + \bar{D} + \cancel{A\bar{C}\bar{D}}$

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

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

$\bar{C} + C\bar{B}$

~~$\bar{C} + C\bar{B}$~~

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

~~$A\bar{C} + \bar{A}\bar{C} + C\bar{B}\bar{D}$~~

~~$\bar{C}(A+\bar{A}) + C\bar{B}\bar{D}$~~

~~$\bar{C} + C\bar{B}\bar{D}$~~

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

$Z = \bar{C}(A+B\bar{D}) + \bar{B}\bar{D}$

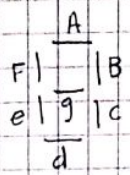
$\bar{Z} = \overline{\bar{C}(A+B\bar{D}) + \bar{B}\bar{D}}$

$C + \overline{(A+B\bar{D})} + B(B+D)$

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

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

ABCDEF	0	0 0 0 0
BC	1	0 0 0 1
ABDEG	2	0 0 1 0
ABCDG	3	0 0 1 1
BCFG	4	0 1 0 0
ACDFG	5	0 1 0 1
ACDEFG	6	0 1 1 0
ABC	7	0 1 1 1
ABCDEFG	8	1 0 0 0
ABCFG	9	1 0 0 1



C

CD / AB	00	01	11	10
00				
01				
11				
10				

$$C = \bar{C} + D + B$$

$$\bar{C} = \overline{\bar{C} + D + B}$$

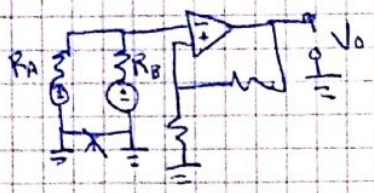
$$\bar{C} = C\bar{D}\bar{B}$$

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

1 1 1
0 or 1 or 1 = 1

1 1 1
1 0 0 = 0

22. Find V_o



$$V_o = -\frac{1}{C}Q = -V_c$$

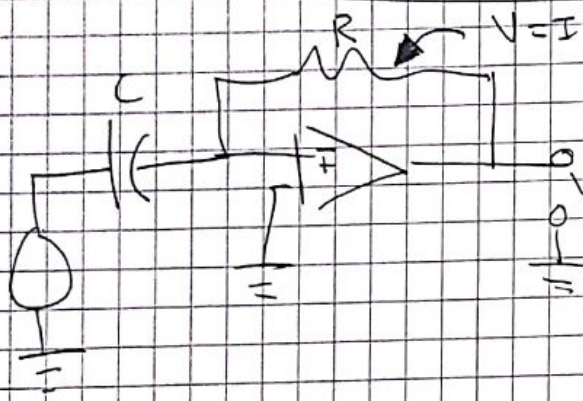
$$Q = CV$$

$$\Delta Q = I \qquad \frac{dQ}{dt} = I = \frac{V_{IN}}{R} \qquad dQ = \frac{V_{in}}{R}dt$$

$$Q = \int (V_{IN}/R)\,dt$$

$$V_o = -\frac{-1}{RC}\int_0^t V_{IN}\,dt$$

15 Minutes = 900 seconds



$$Q = CV$$

$$V_o = \frac{dQ}{dt}R$$

$$\frac{dQ}{dt} = \frac{dC\,dV}{dt}$$

So

$$V_o = R\left(\frac{dC\,dV}{dt}\right)$$

$$V_o = -CR\left(\frac{dV}{dt}\right)$$

M ÷ N → Quo, Rem

```
M:      .BLKW   1 ; INZ, sCOMP > 0
N:      .BLKW   1 ; INZ, sCOMP > 0
QUO:    .BLKW   1 ; OUTZ, sCOMP M÷N QUO
REM:    .BLKW   1 ; OUTZ, sCOMP M÷N REM
```

LAST Block

Start Program
Set Quo to 0
A: if M ≥ N → if M<N goto B
Quo++
M = M-N
goto A
B:
REM = M
End Program

Sub n, tmpm

N	T	N
<	=	>

```
START:  MOV  ZERO, QUO
        MOV  M, TMPM ; (Tmpm - M) (later)
S2      :    SUB  N, TMPM
             BLT  S3
             ADD  ONE, QUO
             BR   S2

S3      :    ADD  N, TMPM
             MOV  TMPM, REM
             HALT

ONE  :  .WORD  1
ZERO :  .WORD  0
TMPM :  .BLKW  1 ; Temp M
```

A B	x y
1 1	0 0
1 0	0 1
0 1	1 0
0 0	1 1

~~$S\bar{R} + \bar{S}\bar{R}$~~

$E = S\bar{R}$

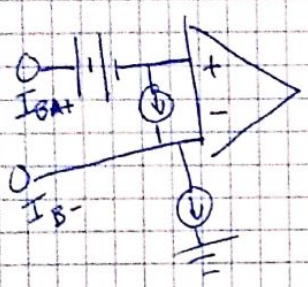
~~$S\bar{R} + R = S\bar{R}$~~

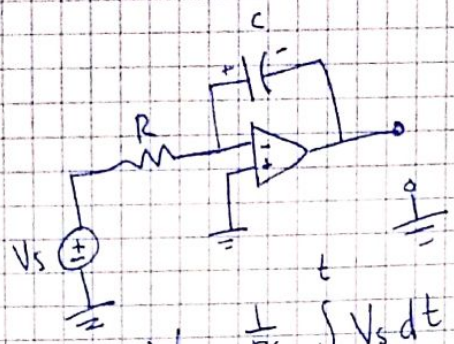
$E = S + R$

so

$E = S + \bar{R}$

Bias Currents





$$V_o = \frac{1}{RC} \int_0^t V_s \, dt$$

Digital Circuits
Homework 3
Ch2: 2, 4, 14, [illegible], 16, 24, 28, 33, 38, 40, 44

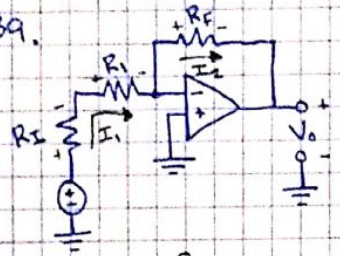
Adam Stammer

2.)

Electronics
Homework 4
Ch 2: 39, 47, 51, 75, 77

Adam Stammer

39.



$R_I = [1000\,\Omega, 1m\,\Omega]$ Design this circuit (R_1 and R_F) such that $A_V = -20 \pm 15\%$. Assume resistor tolerances of $\pm 5\%$.

Start by solving for V_o.

$I_1 = I_2 \rightarrow \frac{V_S - V_-}{R_I + R_1} = \frac{V_- - V_o}{R_F} \rightarrow$ $\boxed{V_o = -V_S\left(\frac{R_F}{R_1 + R_I}\right)}$ (A_V)

So, $\left(\frac{R_F}{R_1 + R_I}\right) = 20 \pm 15\%$. ~~Add in~~

$\left(\frac{R_F}{R_1 + 1000}\right) = 20(1.15)$ and $\left(\frac{R_F}{R_1 + 1m}\right) = 20(.85)$

now add in our resistor tolerances,

$$\left(\frac{R_F(1.05)}{R_1(.99) + 1000}\right) = 20(1.15), \quad \left(\frac{R_F(.99)}{R_1(1.05) + 1m}\right) = 20(.85)$$

With two equations and two unknowns we can solve via substitution.

$$R_F = 17(R_1(.99) + 1000)/1.05 \qquad R_1 = \left(\frac{R_F(.99)}{23} - 1m\right)/1.05$$

$$= \frac{17(.99)R_1 + 17{,}000}{1.05} \qquad = R_F\left(\frac{.99}{23(1.05)}\right) - \frac{1m}{1.05}$$

$R_F \approx 44502113.31106$

$\boxed{R_F \approx 44.502113\,m\Omega}$

$$R_1 = \left(\frac{17(.99)R_1 + 17000}{1.05}\right)\left(\frac{.99}{23(1.05)}\right) - \frac{1m}{1.05}$$

$$R_1 = (16.029R_1 + 16190.48)(0.040994) - \frac{1m}{1.05}$$

$$R_1 = .657098R_1 + 663.71 - \frac{1m}{1.05}$$

$$R_1 = \frac{663.71 - \frac{1m}{1.05}}{.34291}$$

$\boxed{R_1 \approx 2775414.08061 \approx 2.775414\,m\Omega}$

Checking this answer we expect $\left(\frac{R_F}{R_1 + 499500}\right)$ to equal our exact 20 gain goal. I actually found these resister values to achieve a $\pm 15.2\%$ tolerance around a gain of ≈ 13.9 which is significantly lower than our design requirements.

New Answer:

$\boxed{R_1 = 948595\,m\Omega \quad R_2 = 199.3005\,m\Omega}$

47.) DC Gain of 100
Cascading amps with gain of 10

$A_{OCL} = A_V = 100$ $\quad GBP = A_{OCL} \cdot F_{BOL} = 10^4$

$f_{BOL} = \frac{10^4}{100} = 10^4 Hz$

$A_{OL}(f) = \frac{A_{OOL}}{1+j\left(\frac{f}{f_{BOL}}\right)} = \frac{A_{OOL}}{1+j\left(\frac{f}{10^4}\right)}$ $\quad f_{BOL} = \frac{10^6}{10} = 10^5 Hz$ $\quad A_{OL}(f) \frac{A_{OOL}}{1+j\left(\frac{f}{10^5}\right)}$

$A_{OL}(f) = \left(\frac{A_{OOL}}{1+j\left(\frac{f}{10^5}\right)}\right)\left(\frac{A_{OOL}}{1+j\left(\frac{f}{10^5}\right)}\right)$

$= \left(\frac{10}{1+j\left(\frac{f}{10^5}\right)}\right)\left(\frac{10}{1+j\left(\frac{f}{10^5}\right)}\right) = \frac{100}{\left(1+j\left(\frac{f}{10^5}\right)\right)^2} = \frac{100}{1+2j\left(\frac{f}{10^5}\right)-\left(\frac{f}{10^5}\right)^2}$

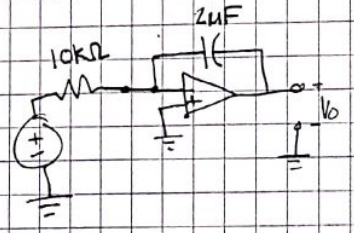
$= \frac{100}{\frac{f^2+10^5 10}{10^{10}} + 2j\left(\frac{f}{10^5}\right)}$ Complex conjugate to take further?

51.) Slew rate at 100kHz sin wave, amplitude of 5 Volts

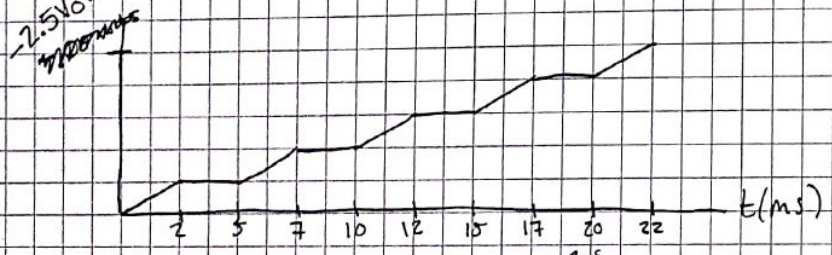
$SR = 2\pi f V_{OMAX} = 2\pi(100{,}000)(5) = 3.14 * 10^6 V/s$

$= 3.14 V/\mu s$

75.)



Sketch V_o over time (ms=0, ms=22)



What if $V_o = -10$?
↳ How many pulses have gone through?

$10/.5 = 20$ pulses

$\frac{1}{10k(2\mu)} \int_0^{2ms} 5$ Volts dt ~~.01 Volts per pulse~~ .5 Volts per pulse

~~10/.01 = 1000 pulses~~

Robotic Arm Displacement

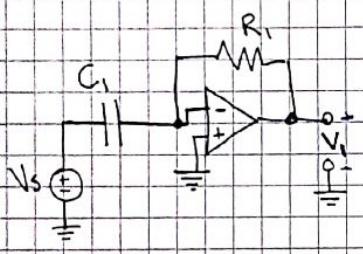
10cm → 1 Volt

Design Circuit such that 1m/s → 1Volt, and another circuit such that $1m/s^2$ → 1Volt

$$\frac{1\text{ Volt}}{.1\text{meter}} = \frac{x\text{ Volts}}{1\text{meter}} \qquad x = \frac{1}{.1} * 1 = 10$$

so 1 meter of displacement corresponds to 10 Volts.

Velocity is the derivative of displacement so let's use a differentiator



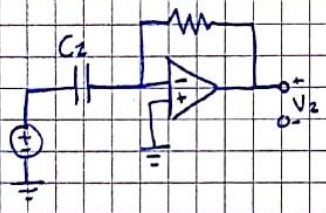
$$V_1 = R_1C_1\frac{d(V_s)}{dt} \qquad V_s = \text{displacement (in meters)} * \frac{\text{Volts}}{\text{meter}}$$

$$V_1 = R_1C_1\frac{d(10x)}{dt} \leftarrow V_s = x\left(\frac{10}{1}\right) = 10x$$

$$1m/s = R_1C_1\frac{d(10x)}{dt} \rightarrow 1 = R_1C_1 10$$

$$R_1C_1 = \frac{1}{10}$$

So let's choose $\boxed{C_1 = .1\mu F}$ → $R_1(.1*10^{-6}) = \frac{1}{10}$ → $\boxed{R_1 = \frac{\frac{1}{10}}{(.1*10^{-6})} = 1M\Omega}$



Now if we differentiate again we will have acceleration

a) $\{a^i b^j : i, j \geq 0 \text{ and } i+j=5\}$

This a regular language because there is a countable set of i and j that equal 5

0+5
1+4
2+3 and the reverses.

An FSM could be made to accept this language.

b) $\{a^i b^j : i, j \geq 0 \text{ and } i-j=5\}$

This we know is not a regular language because there is an uncountably infinite number of combinations of i and j such that $i-j=5$.

We just need the number of a's to be 5 more than the number of bs

Lets pump out $a^{k+5} b^k$. Since $|xy| \leq k$ y must equal some a^p

Pumping y out once we get $a^{k+5-p} b^k$ which is not accepted by our language since the number of a's is less than 5 more than the number of bs

c) $\{a^i b^j : i, j \geq 0 \text{ and } |i-j| \equiv_5 0\}$

This language is regular. $|i-j| \equiv_5 0$ only when $i \equiv_5 j \equiv 0$. We can count a and b by mod 5 in an FSM and accept when they are equal.

-2 * 4

1010
-1
1001

	4	①
-3	-8	② ✓
-4	-4	③
-5	0	④
-6	4	⑤
-7	-8	⑥ ✓
-8	-4	⑦
7	0	⑧
6	4	⑨
5	-8	⑩ ✓
4	-4	⑪
3	0	⑫
2	4	⑬
1	-8	⑭ ✓
0	-4	⑮

$14 \times 4 = 56$

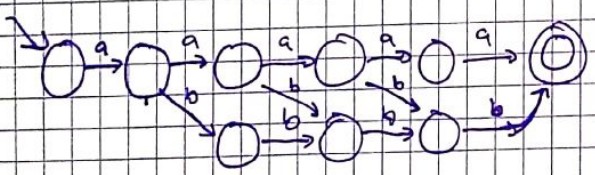
2^n where $n = 4 = 16$

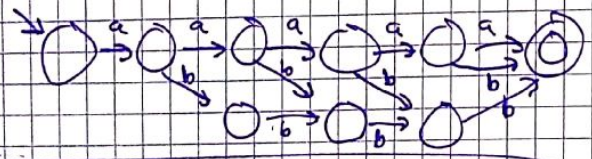
$56 \bmod 16 = 8\ (1000_2)$

$= -8$ in 2's comp.

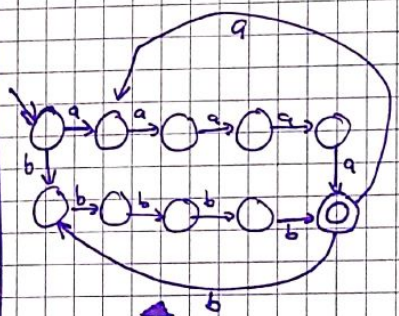
500
421
401
101

$\{a^i b^j : i, j \geq 0 \text{ and } i + j = 5\}$





No a will come after a b and the length of the string must be 5



$\{a^i b^j : i, j \geq 0 \text{ and } |i - j| \equiv_5 0\}$

A = B
A < B
A > B

SUB B, A ; (A)-(B)→(A)
B - -

- BLT
- BGT
→ Destructive

CMP src, dest
(src - dst) → only sets condition codes

See how the order of subtraction is different than that of compare

Register Mode Addressing (Mode 0)

CMP R0, R1
(R0) - (R1)

OP Code	SRC MODE		DST MODE	
	000	000	000	001
	↑ MODE	↑ REGISTER #		

1 word
1 read

SUB R0, R1
(R1) ← (R1) - (R0)

Rel mode addr. =

OP CODE	67	67	relA	relB

3 words
5 reads

	Clear Prod		3
LOOP:	ASRN		4
	BCC SKIP		1
	ADD M, PROD		6
SKIP:	ASL M		4
	BNE LOOP		1
	HALT	1	16*16 = 256 memory references

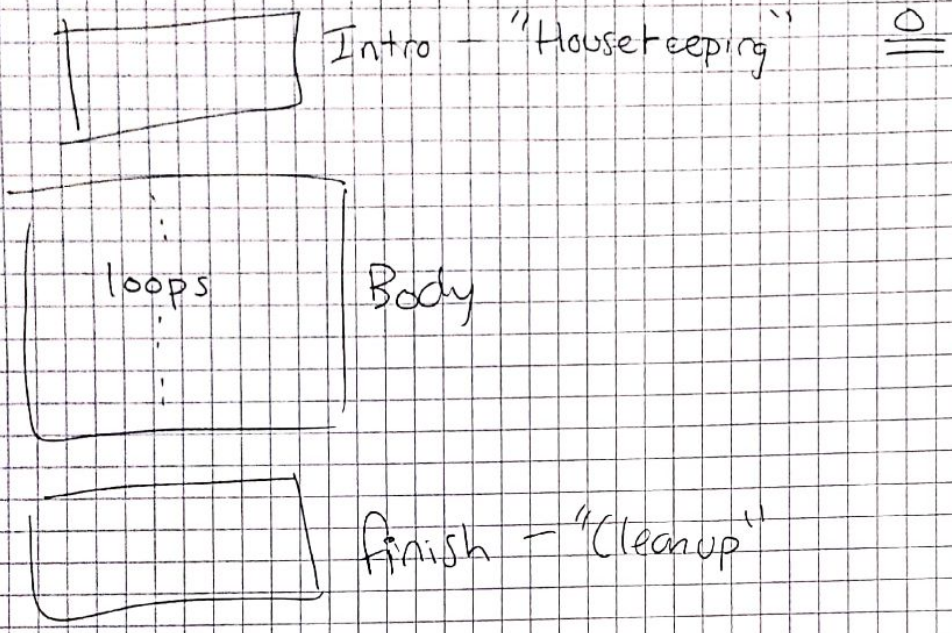
12 words of instructions

3+1+(16*16) = 260 memory references

M: .BLKW 1.
N: .BLKW 1.
RROD: .BLKW 1.

memory references instead of 5
is instead of 4 (or 3)

Computer Systems Exam #1 → Oct 10 Exam



~~Initialize to zero~~

ZERO: .WORD 0

MOV ZERO, A

~~# of REFS~~

~~HHH~~

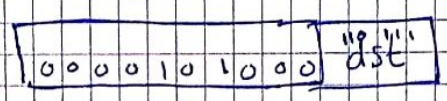
SPACE

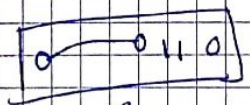
3 words (Instruction)
1 words (constant)
4 words

Use # of memory references a proxy measurement since memory references take so long

20.

CLR dst

10. 



12.

(dst) ← 0

N - 0
Z - 1
V - 0
C - 0

Only 3 memory references instead of 5
2 words instead of 4 (or 3)

```
ONE: .WORD 1
A:   ADD ONE, A
     SUB ONE, A
```

3 words 6 memory references

```
INC dst   ; increment dst by 1   (dst) ← (dst)+1.
DEC dst   ; decrement dst by 1   (dst) ← (dst)-1.
```

Same reg codes as add except C is not changed

2 words 4 memory references

MUL → $O(2^n)$ 2^n: max int
DIV → Both
↳ Div MaxInt / 1
↳ Mul Anything * Maxint

multiplying by hand is an $O(n)$ algorithm for multiplication

```
  12
  144
   36
  ---
  864
 4320
 ----
 5184
```

```
  00101
  00011
  -----
  00101
 0010100   ← bitshift left (A x 2^1)
 -------
  00111111  ✓
```

To split a number into its digits we could divide by 2 and take the remainder

```
ASL dst   ← 0
ASR dst   → C
N gets copy of HOB
V 1 on overflow (N ⊕ C)
```

Pure Shift → fill bit holes with 0 — Cardinal we want 0, 2's comp we want 1

Arithmatic Shift → Left shift still a 0, Right shift replicate existing high order bit

M 00011
N 00101
PROD

CLR B

CLP PROD

16x for loop

LOOP: ASR N
BCC SKIP
ADD M, PROD
→ could check for overflow
↳ BVS OVERFLOW
SKIP: ASL M

~~DEC I~~
BNE LOOP
~~SIXTEEN: .WORD 16.~~
I: .BLKW 1.
M: .BLKW 1.
N: .BLKW 1.

Decoder

$Y_3 = A_1 B_0$

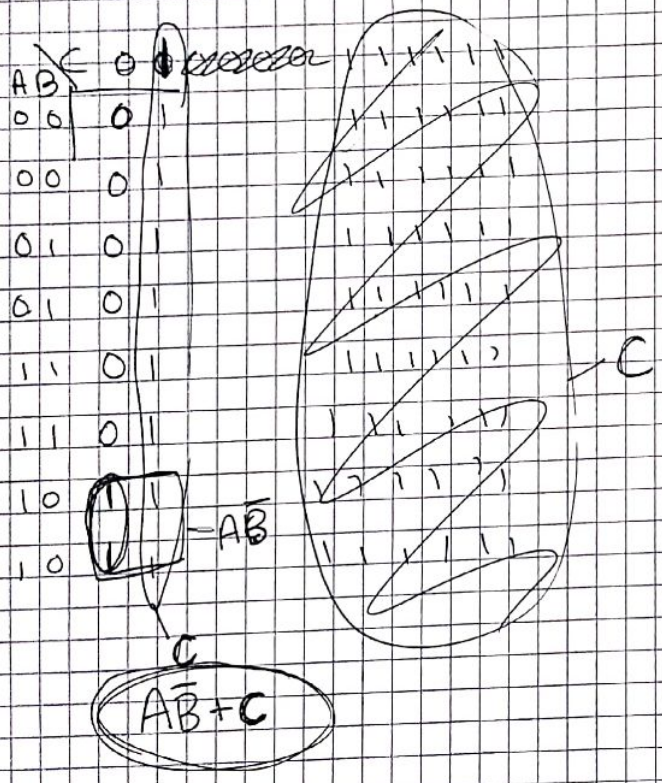
$Y_2 = A_1 \bar{A}_0$

$Y_1 = \bar{A}_1 A_0$

$Y_0 = \bar{A}_1 \bar{A}_0$

$Y_5 = A\bar{B}C + \bar{A}\bar{B}C$

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

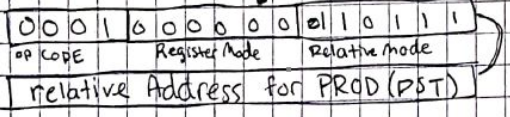


O(N)
n = # of bits

CLR R0

MOV M, R1
MOV N, R2

			words	mem refs
LOOP:	ASR	R2	1	1
	BCC	SKIP	1	1
	ADD	R1, R0	1	1
SKIP:	ASL	R1	1	1
	BNE	LOOP	1	1
			5 words	5 mr
	HALT		1	
	MOV	R0, PROD	2	



MORE REGISTERS WOULD NOT ONLY SLOW DOWN THE READING OF REGISTERS BUT ALSO REQUIRE ADDITIONAL BITS TO REFERENCE IN THE ADDRESSING MODE

$$30 = \frac{177828}{1+\left(\frac{R_1}{R_1+R_2}\right)177628}$$

$$A_{OCL} = \frac{A_{OOL}}{1+\beta A_{OOL}}$$

$$\frac{V_o - V_-}{30k} = \frac{V_s - 0}{10k}$$

$$A_{OCL} = \frac{A_{OOL}}{1+\beta A_{OOL}}$$

$$\frac{V_o - V_s}{30k} = \frac{V_s}{10k}$$

$$A_{OCL} =$$

$$V_o = \left(\frac{R_1+R_2}{R_1}\right)$$

$$10^{(105/20)} \cong 177827.941$$

$$A_{OOL} \cong 177828$$

$$\frac{V_o - V_s}{R_2} = \frac{V_s - 0}{R_1}$$

$$V_o = \frac{V_s R_2}{R_1} + V_s$$

$$= V_s\left(\frac{R_2}{R_1} + 1\right)$$

$$= V_s\left(\frac{R_2+R_1}{R_1}\right)$$

9 x 1k and 2 10k

$$\frac{1k + (29k)}{1k}$$

B

$$A_{vo} = 200 V_i = 200\left(\frac{100k}{102k}\right)V_s$$

$$V_i = \frac{R_i}{R_i + R_s}$$

$$V_L = \frac{R_L}{R_o + R_L} A_{vo} = \frac{R_L}{10 + R_L}\left(200\left(\frac{100}{102}\right)\right)V_s \quad \hookrightarrow .01$$

$$= \frac{R_L}{10 + R_L}\left(\frac{100}{102}\right)2$$

$$1.5 = \frac{R_L}{10 + R_L}\left(\frac{200}{102}\right)$$

$$1.5 = \frac{R_L\left(\frac{200}{102}\right)}{10 + R_L}$$

$$10 + R_L = \frac{R_L\left(\frac{200}{102}\right)}{1.5} = R_L\left(\frac{200}{153}\right)$$

$$10 = R_L\left(\frac{47}{153}\right)$$

$$R_L \cong 32.55$$

$$\cong 33\,\Omega$$

EIS → EXTENDED INSTRUCTION SET

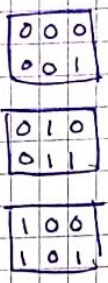
ACCUMALATOR BASED

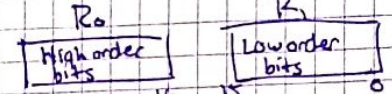
2 - 16 bit integers

Addition → N+1 digits

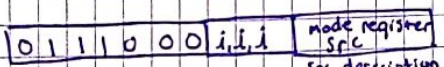
Multiplication → 2N digits

LD so lets pair registers together to store 32 bits





MUL



$(R_i, R_{i+1}) \leftarrow (src) * (R_i)$

32 bit or 16 bit | 16 bit 2's comp | 16 bit 2's comp

LD or the stated register # with 1. If you get the same #, a 16 bit answer will be stored there. Otherwise you get the register number + 1 and store a 32 bit product in those two registers.

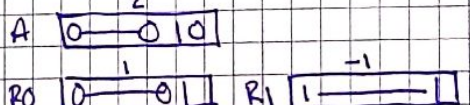
→ odd register → 16 bit stored
→ even register → 32 bits stored

Double Precision is considered default → condition codes based off of 32 bit result.

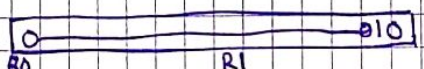
N - HOB
Z - all 32 bits 0?
V - 0 always
C - fits 16 bits C → 0
not fit 16 bits C → 1

32 bit → double precision
16 bit → single precision

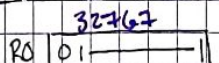
Examples



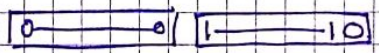
MUL A, R0



N-0 Z-0 V-0 C-0

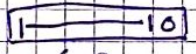


MUL A, R0



N-0 Z-0 V-0 C-1

MUL A, R1

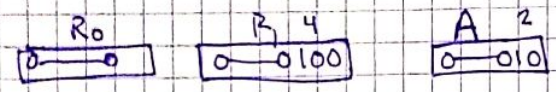


N-1 Z-0 V-0 C-0

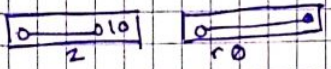
DIV A, Ri i must be even

DP ÷ SP = SP QUO
SP REM

N - HOB of 16 bit QUO
Z - QUO ZERO ?
V - quo fit in 16 bits or divide by zero
C - if div by zero

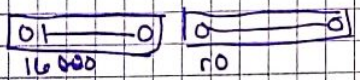


DIV A, R0

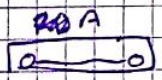


N-0 Z-0 V-0 C-0

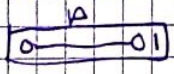
DIV A, R0



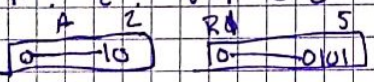
N-0 Z-0 V-0 C-0



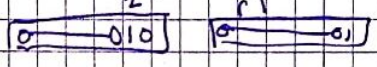
N? Z? V-1 C-1



N? Z? V-1 C-0



DIV A, R0 ;5/2=2r1



N-0 Z-0 V-0 C-0

MOV R1, R1
↳ changes the condition codes based on R1 (The remainder)

N-0 Z-0 V-0 C-0

Fib(x) = 1 if x <= 2

Fib(x) = Fib(x-1) + Fib(x-2) if x > 2

```
TMP:    .WORD   1   ; TMP COUNTER
I:      .BLKW   1   ; IN  2's comp > 0
FIBN:   .BLKW   1   ; OUT 2's comp FIB((I)) or
                    ;     -32768 as error message
```

```
        clear FIBN
        Copy I to TMPcount      <- CHECK INPUT: IF I < 1
                                                GOTO CRASH
        FibNew = 1   FibOld = 0
LOOP    Temp = FibNew
        FibNew += FibOld
        FibOld = Temp
                                <- ERROR CHECK
        TMPCOUNT -= 1
        IF TMPCOUNT == 0
          GOTO FIN
        ELSE GOTO LOOP

        IF N
          GOTO CRASH
        IF Z
          GOTO CRASH
        IF V
          GOTO CRASH

FIN:    FIBN = FibNew;
        HALT
CRASH:  FIBN = -32768.
        HALT
```

MUL
DIV

OPCODE	iii	SRC DESCRIPTION FIED

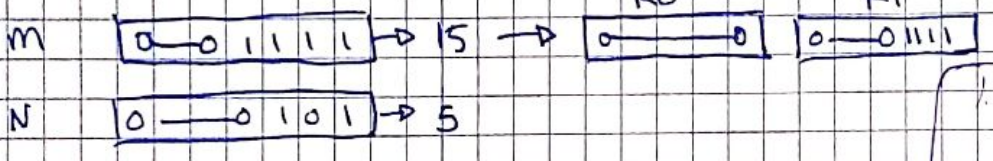
2's comp only

DP ← SP * SP

Even # register for multiply
↳ input will come even register
and product will span both registers
V=0 always (overflow impossible)

DP ÷ SP = SP Quotient Even Register
SP Remainder ODD Register

M ; Dividend
N ; divisor




```
MOV  ONE, R0
MUL  M, R0
```

or

```
CLR  R0
MOV  M, R1
BPL  SKIP
MOV  NEGONE, R0
SKIP:
```

SXT dst
(dst) ← 16 copies of N
N → same
Z → if all zeros
V → 0 always
C → Not changed

```
MOV  M, R1
SXT  R0
```

Euclid's Algorithm Example

M and N are 2's comp non zero

S1 Divide M by N set quo, rem

S2 if rem=0 you're done N is GCD

S3 M ← N
N ← R
GO to S1

Euclid's Algorithm Code

```
M:   .BLKW  1
N:   .BLKW  1
GCD: .BLKW  1

START: MOV  M, R0      ; we need M to be double precision
       MOV  N, R2      ; We
   S1: MOV  R0, R1
       SXT  R0         ; R0→R1 is M double precision

       DIV  R2, R0     ; R0→ Quo, R1→ Rem

   S2: MOV  R1, R1     ; set Z

       BEQ  DONE

   S3: MOV  R1, R3     ; copy remainder so its not lost
       MOV  R2, R0     ; M←N
       SXT  R0         ; make our new N doubleprecision
       MOV  R3, R2     ; N←R from previous copy
       BR   S1
```

16 bit 2's complement

100 ⋯ 00 → -32768
011 ⋯ 11 → 32767

Double Precision: -2^{31} to $2^{31}-1$

M: .BLKW 2
N: .BLKW 2

= (copy) MOV M,N HOB
MOV M+2,N+2 NO HOB

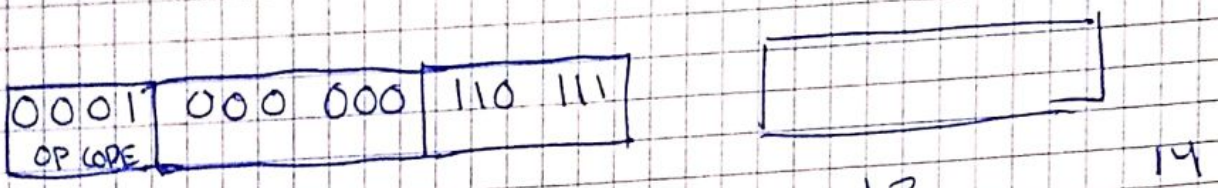
\+ ADD M+2,N+2
BCC SKIP
ADD ONE,N
SKIP: ADD M,N
* - BVS OVERFLOW

ADD CARRY ADC dst
SUB CARRY SBC dst

(dst) ← (dst) + (C)

- → but use SUB instead of ADD

location 10. Mov R0, 2



10 + 2 + 2 = 14

What do I add to 14 to get 2?

$14 - 2 = 12_{10} \rightarrow +1100_2$

1's → 10011

1 → 10100

- Find an advisor that does what you want to do
  → apply based on that, choose school based around that

- gradschoolshopper.com *

- where do I want to go? → where do I not want to go?

- invite people to see your stuff (poster)

- postback?
  → few years off

→ do you want to teach, research, or work in industry?

→ go somewhere you want to live
  ↳ it's not all about school

GRE
  ↳ prep courses
  → do it beforehand so applications have scores already

→ check deadlines
  ↳ ask for recommendation letters early

first paragraph
  ↳ personal story of why I want grad school and why I ~~want~~ would be good at it.

Be Purposeful!
  → Why this program?
  → Why?
  → What do you want to do?

Length of program? Support?

Don't pay for school
  → get paid
] PhD programs

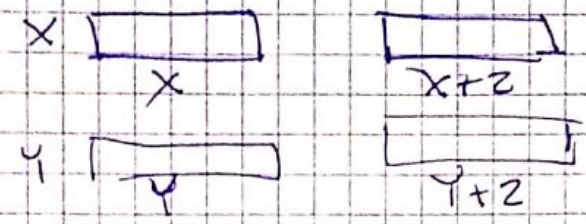
Fly out after acceptance (they should pay to bring you out)

Know ~~there~~ their research and talk to them about it

Ask about support and services
  - leave of absence
  - pregnancy
  - mental health
  - illness

Exam 1 Distro

60 2
50 6
40 5
30 3




```
ADD  X+2, Y+2
BCC  SKIP
ADD  ONE, Y
SKIP: ADD  X, Y
```

↓

```
ADD  X+2, Y+2
ADC  Y
ADD  X, Y
```

32 bit 2's comp but also works for cardinal

Overflow V bit

C bit

ADC → Add carry
SBC → Sub carry

(dst) ← (dst) ± (C)

≠ →
```
SUB  X+2, Y+2
BNE  →
SUB  X, Y   → [Not Equal]
BNE  →
```

= → [Equal]

(B-A) = 0 → equal → N=0
B>A
A>B → N=1

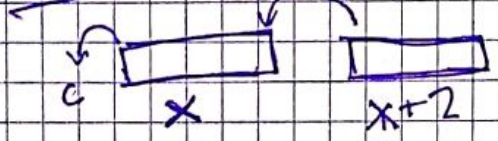
TST dst does not change dst

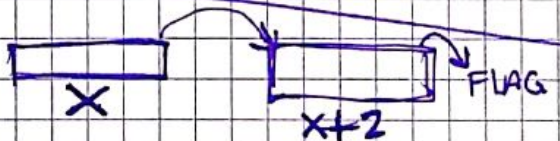
N - Negative MOB

Z - 1 if all zeros
0 otherwise

V - always zero

C - always zero





~~ASL X~~

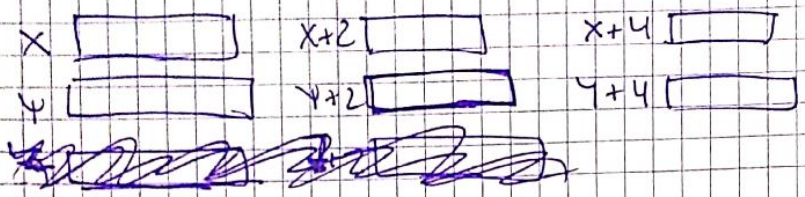
DP Left Shift

```
CLR FLAG
ASL X
ADC FLAG
ASL X+2
ADC X
```

DP Shift Right

```
CLR FLAG
ASL X+2
ADC FLAG
MOV X+2, X+2 → set N condition
BPL ITSZERO
SUB MININT, X+2
ITSZERO: ASL X
BCC DONE
ADD MININT, X+2
```

Triple Precision Addition



ADD X+4, Y+4
ADC X+2, Y+2
ADC X, Y

← carry the carry all the way to the left

ADD X+2, Y+2
ADC X, Y

check for overflow

ADD X, Y

CRAY → ERA → CDC Control Data Corp

CRAY RESEARCH → designed some of the fastest computers of his day

---

Time of Instructions

- different configurations of hardware create different times, so old machine with only 1 configuration

PDP-11/40

CORE (NON VOLATILE) .980μs MAJOR Cycle Time
↳ read and writeback → rowhammer exploit

Minor Cycle Time

LOAD	MOV	X, Ri	2.32μs		
STORE	MOV	Ri, X	2.58μs		
	ADD	X, Ri	2.54μs	Ri, Rj	1.07
	MUL	Ri, X	9.66μs		
	DIV		12.08μs		

Tues Oct. ~~22~~ 22nd?

Week from 15th

N Squared without Multiplication

A computational method for finding N squared is to sum the first N odd integers.

5 squared equals the sum of 1 + 3 + 5 + 7 + 9

Write a program to compute N squared by using this computational method.

```
;
;       Data
;
;
N:      .BLKW       1       ;IN     2's complement > 0
NSQ:    .BLKW       1       ;OUT    2's Compl.The Square of (N) or -32768. as error msg
        .END
```

~~J: .BLKW 1 ; current odd int~~

```
        J=1
        NSQ=0       ← if N>0 → if N<1 GOTO CRASH
        TEMPN=N
LOOP:   TEMPN-=1
        IF TEMPN≤0  GOTO FINISH
        ADD J,NSQ
        ADD TWO,J
        GOTO LOOP

CRASH:  NSQ=-32768
        END

FINISH: END
```

```
J:      .BLKW   1 ; current odd int (2's comp)
N:      .BLKW   1 ; IN   2's comp >0
NSQ:    .BLKW   1 ; OUT  2's comp. of N^2 or -32768 as error msg
```

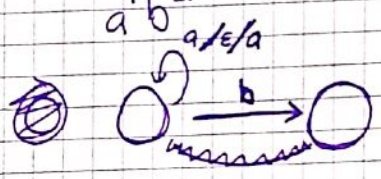
J : x 5

N : 2

NSQ : 4

TEMPN : 2 x 0 - 1

$4 = N^2$ ✓



J: x 3

N: 2

NSQ: x 4

TEMPN: x x 0 → Finish

$a^i b^i \; i \geq 0$

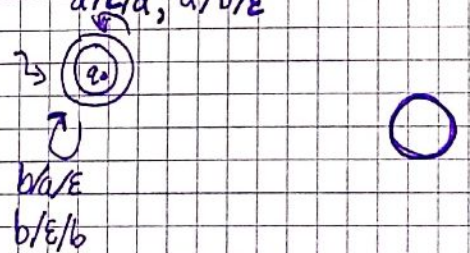
$\#_a(w) = \#_b(w)$

NSQ = 1

LOOP: TempN = TempN - 1
IF TEMPN < 1 FINISH
ADD TWO, J
Check for overflow
ADD J, NSQ
Check for overflow

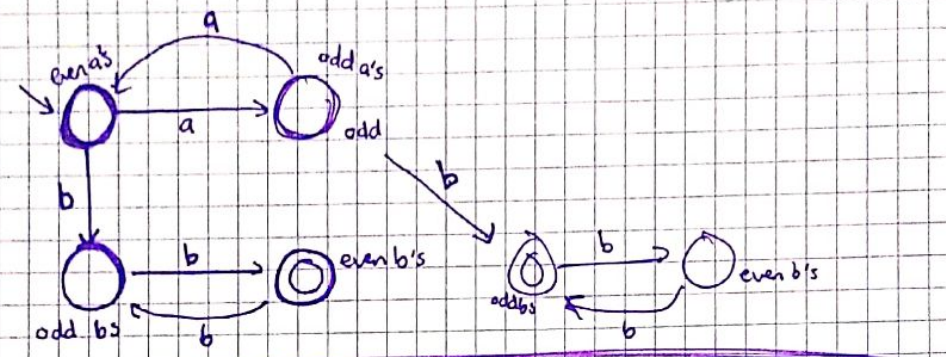
TEMPN: 76 100
N: 100 102
J: 102 104
NSQ: 76 104

$\#_a(w) = \#_b(w)$

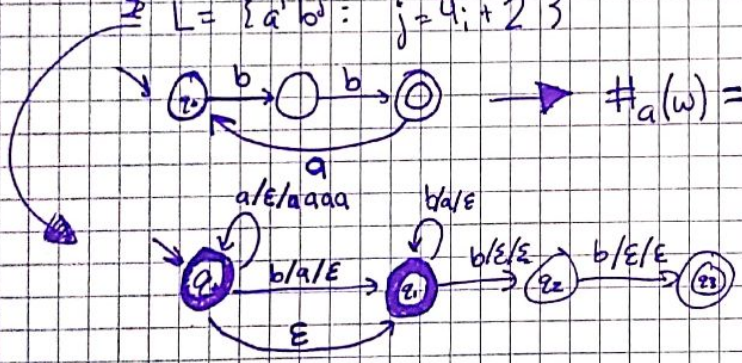


$a^i b^j$: $i \equiv_2 j$ (both odd or both even)

no need for stack

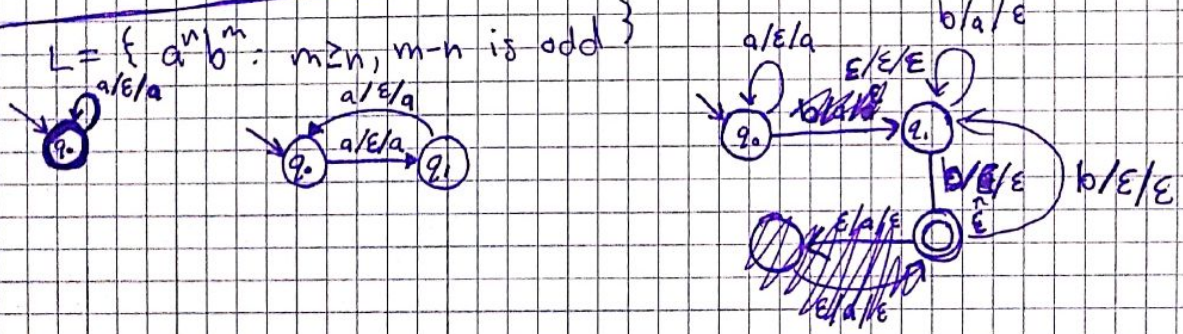


$L = \{a^i b^j : j = 4i + 2\}$



$\#_a(w) = 2\#_b(w)$

$L = \{a^n b^m : m \geq n,\ m-n \text{ is odd}\}$



MOV R0, R1

0001	000 000	000 001
OP CODE	R 0	R 1

SB	EXP	MANTISSA FRACTION
1	10000001	00——0

Excess 128

EXP-129

$1.00-0 * 2 = 1 \times 2^1 = -1$

12.5

12_{10}

1100_2

.5

2

1.0

.1

1100.1

$.11001 * 2^4$

0	10001100	10010——0

Exp

128+4=132

Mantissa

1001

10000000

01111000

128+4=132

10000100_2

10001100_2

6.2831853

2.3661952

$.7753975$

$$\frac{2.3661952 - .7753975}{6.2831853} \approx .253183317704$$

$$\frac{\left(\left(\frac{2\pi}{4}\right) + \sin^{-1}\left(\frac{14}{20}\right)\right) - \sin^{-1}\left(\frac{14}{20}\right)}{2\pi} = 25\%$$

$20\sin(x) = 14$

$\sin(x) = \frac{14}{20}$

$\sin^{-1}\left(\frac{14}{20}\right) = x$

$\frac{\frac{\pi}{2}}{2\pi}$

Digital Circuits
Ch3 prob. 2, 19, 20, 22, 26

2.)

100 TEMPN
102 N
104 J
106 NSQ

-32768	CRASH	✓
-32767	C	✓
-10000	C	✓
-170	C	✓
-100	C	✓
-2	C	✓
-1	C	✓
0	0	✓
1	1	✓
2	4	✓
3	11 → 9	✓
20		
100	620_8 →	
180	77720_8 →	
181	077771_8 →	
182	ERR	✓
183	—	✓
200	—	✓
10000	—	✓
32767	—	✓

Optimizing Compilers

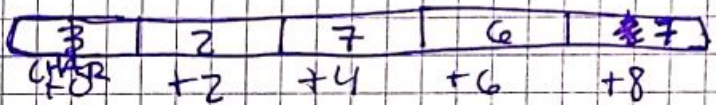
2*M → M+M
- more subtle changes to save time
- hard to debug

```
M:      .BLKW 1     ; non neg 2's comp int
CHARS:  .BLKW 5     ;

START:
        MOV  M, R1
        SXT  R0     ; (R0,R1) = (M)

        DIV  TEN, R0 ; R0 is quotient, remaind R1
        ADD  SIXTY, R1
        MOV  R1, CHAR+8.
        mov  R0, R1
        SXT  R0
        DIV  TEN, R0
        ADD  SIXTY, R1
        MOV  R1, CHAR+6.
        repeat for the next three characters
```

0 0111
0 0 0 0 7

SUBROUTINES
OPEN
CLOSED

```
TEN:    .WORD 10.; STORE 10 FOR LATER
SIXTY:  .WORD 60x; ASCII ZERO
```



```
START:  MOV #FIVE, R5
        MOV M, R0

LOOP:   MOV R0, R1
        SXT R0
        DIV TEN, R0
        ADD SIXTY, R1
#ABC:   MOV R1, CHAR+8
        MOV R0, R1
        SUB TWO, ABC+2
```

MOV R1, CHAR+8

Instruction Modification

0001 001 110 111 | CHAR+8

So subtract 2 from ↑ this word

0001 001 110 111 | CHAR+6

do it again

0001 001 110 111 | CHAR+4

Immediate mode addressing (2,7)

MOV #5, R5

01 27, 05 | 000 00 101

MOV Five, R5

01 67 05

FIVE RELATIVE

Student Travel on WSV website
↳ on Slack

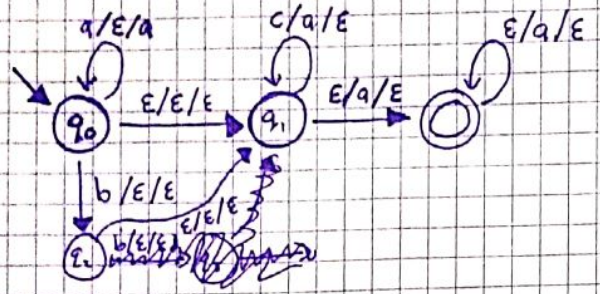
Outline Poster
↳ Be detailed
→ Things to say
→ Sections on Poster
→ Mostly not important that it is a poster yet

Profresional Skills on OSF

Leave Hawaii on Thursday the 9th

Return date → Friday the 10th

$L = \{a^i b^j c^k : i > k,\ 0 \le j < 3\}$



Exam Next Week

4 languages – context free & not regular

Show 3 things

- not regular (pumping lemma)
  - LD reverse of not regular → original not regular
- push down automata for it
- context free grammar for it