N^{th} Prime Number

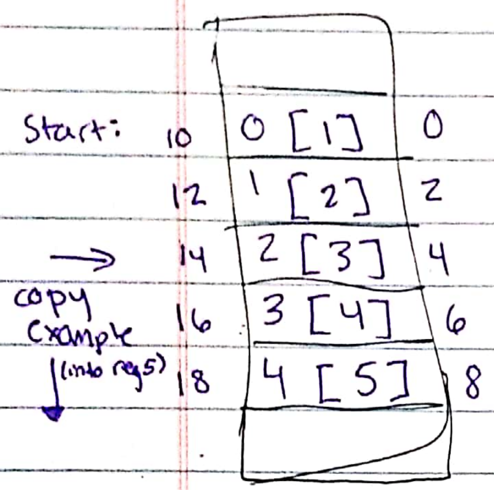
- Only Check odd numbers
- Only division by other odds
  - ↳ don't check 1

if In=1 return 2
~~else~~
j=1

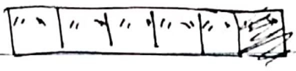
Loop: ~~while n>0~~
n-=1
if n==0
out=j
end
j+=2 //next odd int
~~x=3~~ x=1 // (1+2) (both already stored)
Div Loop jc=j x+=2 ~~check for overflow~~ if x>j
jc /= x
if remainder == 0 //not prime
exit div Loop
if

goto loop

Assembler (PDP-11) Arrays



Char



Char is an array [l...4] of type int

\# of elements

lower

$((4 - l) + 1) * \text{sizeof type}$

$5 * 1$ word

address (Char[i])

Start Location + (i-l) sizeoftype

Addressing Mode

Dynamic

Start: + offset

↳ put in registers for the sake of speed

"Indexed Mode Addressing"

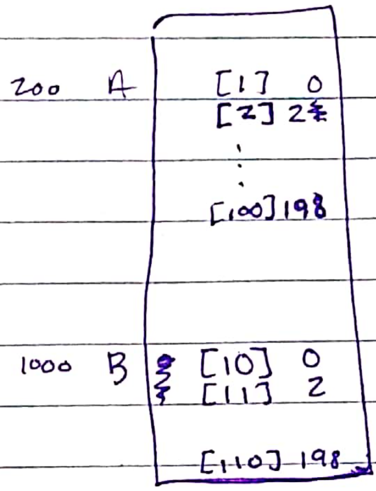
6, i

MOV Start(R1), R5 → MOV ~~[illegible]~~ #.4, R5

0001	110 001	000 101		0 01010
	6 1	0 5		10
	6 i			
	R1	R5		START

A is an array [1...100] of INTEGER
B is an array [10...110 of INTEGER

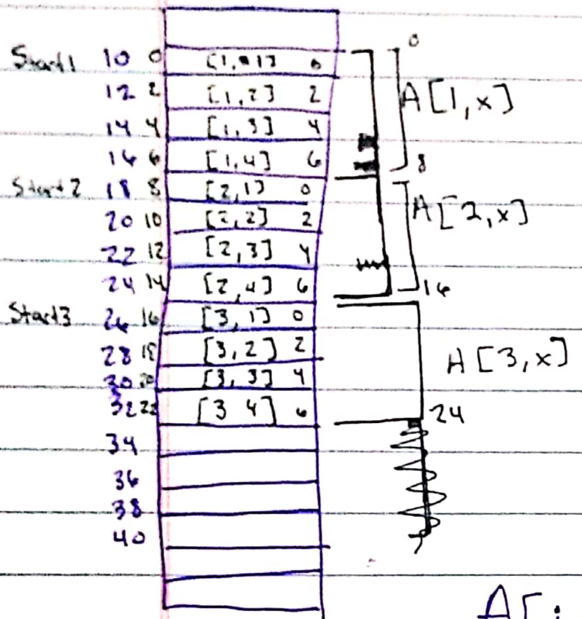
```
200.  A:    .BLKW  100.
1000. B:    .BLKW  100.

COPY:  MOV #198., R1
LOOP   MOV A(R1), B(R1)

       SUB #2., R1
       BGE LOOP
```



A is an array $[l_1 \ldots u_1, l_2 \ldots u_2]$ of type
($l_1 \ldots u_1$ = 1 … 3, $l_2 \ldots u_2$ = 1 … 4)

similar shape

A is an array$[l_1 \ldots u_1]$ of type array$[l_2 \ldots u_2]$
(1 … 3) (1 … 4)



$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{pmatrix}$$

ROW ORDER MAPPING

Row major order

$A[i,j]$

Total Size: $[u_1 - l_1 + 1] * [u_2 - l_2 + 1] *$ size of type (bytes)

$3 \quad * \quad 4 \quad * \quad 2$

24 bytes → 12 words of storage

$i \rightarrow (u_1 - l_1 + 1) *$ size of type $* \, i$

$j \rightarrow$ ~~…~~ $*$ size of type $* \, j$

$\rightarrow (u_1 - l_1 + 1) *$ size of type $* \, i +$ size of type $* \, j$

$\rightarrow$ size of type $* \, ((u_1 - l_1 + 1) * i + j)$

A[2,3]
→ offset 12
→ Addr 22

$\rightarrow (i - l_1)(u_2 - l_2 + 1)$ size of basetype
$+ (j-1) * 2$

Scanned with CamScanner

Ch 8 1abc, Ch 11 1, 6abc, Ch 12 1abc

1. a) $a^ib^j : i,j \ge 0$ and $i+j=5$

Regular strings a^*b^* with length of 5

-only 5 possible strings so regular

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

$p \ge 1$

so lets look at a^kb^k, where $y=a^p$, $|xy| \le k$

pump up such that $w=a^{k+p}b^k$. Since $p \ge 1$

at the smallest $w=a^{k+1}b^k$ which no longer fits

the rule of $i-j=5$. Not Regular.

$w=a^kba^k$ since $|xy| \le k$ we know that

y must fall under a^k so $y=a^p$ $p \ge 1$

So let's pump up $w=a^{k+p}ba^k \to w=a^{k+1}ba^k$

this is no longer a palindrome

$a^ib^j : 2i=3j+1$ ~~b~~, aabbbbbbb, aaaabbbbbbbbbbb

~~S → aaSbbb~~

~~S → b~~

i=0	X		
1	2=3j+1 X		
2	4=3j+1	j=1	aab✓
3	6=3j+1 X		
4	8=3j+1 X		
5	10=3j+1	j=3	aaaaabbb
6			

$\boxed{\text{aaa S bbb aab}}$

$a^i b^j : 2i \neq 3j+1$

i	j
0 →	✓ any
1 →	✓ any
2 →	any except 1
3 →	✓ any
4 →	✓ any
5 →	any except 3
6 →	✓ any
7 →	✓ any
8 →	any except 5
9 →	✓ any
10 →	✓ any

$S \to aaaSbb$

$S \to aaaX$ / extra a's

$S \to T$ / Terminate

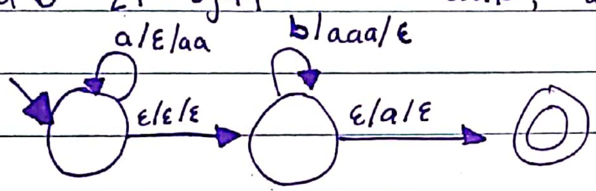
$X \to A \mid Ab$ / arb. more a's

$T \to A \mid B \mid aB \mid aabbB$

$A \to aA \mid \varepsilon$

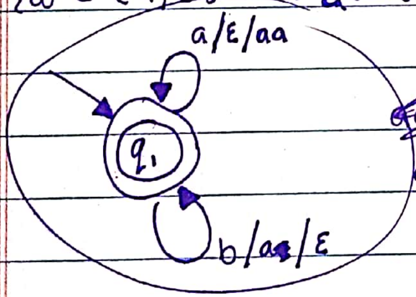
$B \to bB \mid \varepsilon$

$a^i b^j : 2i = 3j+1$ aab, aaaaa bbb



~~aaabbb~~ ~~aaaaaa~~

$\{w \in \{a,b\}^* : \#_a(w) = 2\#_b(w)\}$



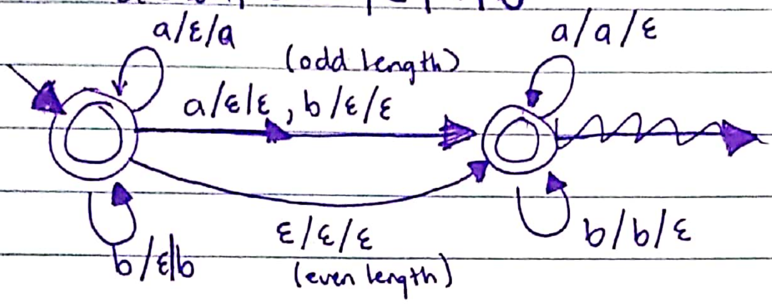
~~aab~~ ~~aaaa~~

aabaabb ~~aaaaa~~

$\{ w \in \{a,b\}^* : w = w^R \}$

all palindromes, both even and odd

$S \to aSa \mid bSb \mid \varepsilon \mid a \mid b$

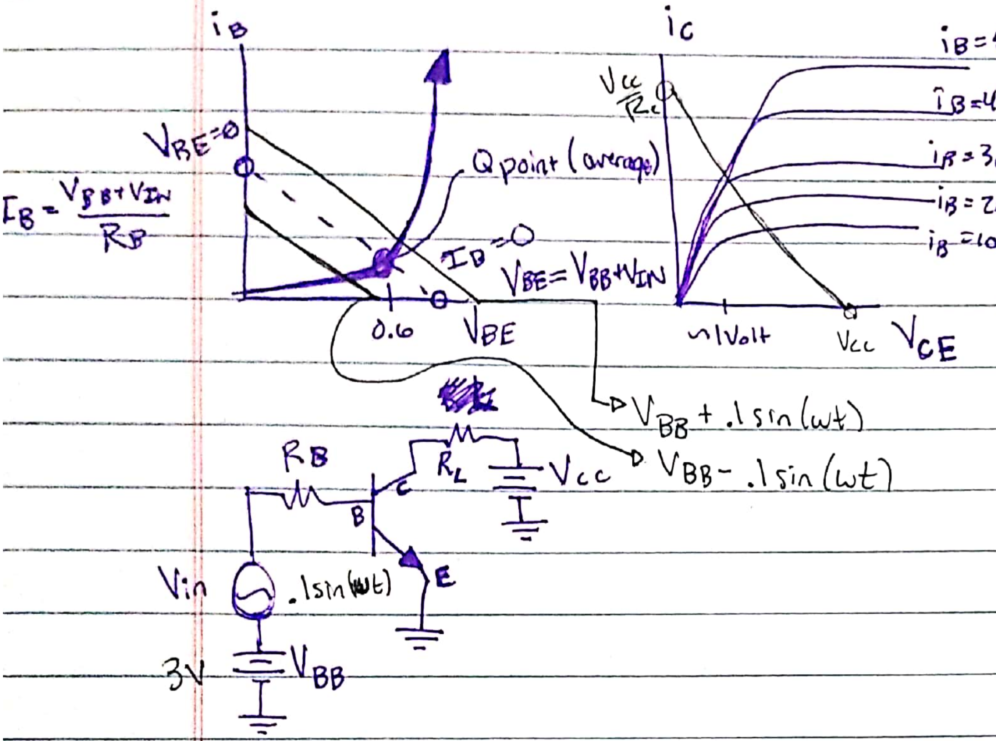


$w = a^k b^k$ even length palindrome

↳ $a^{k+1} b^k$ → no longer palindrome

$w = a^k b b^k$

↳ $a^{k+1} b b^k$



Loop #1: $\emptyset + V_{BB} + V_{in} - V_{RB} - V_D = \emptyset$

$$V_{BB} + V_{in} - I_B R_B - V_{BE} = 0$$

$$I_B = \frac{V_{BB}+V_{in}}{R_B} = \frac{.8V + \cancel{.2\sin(2000\pi t)}}{40k\Omega} \cancel{\approx .00052\ \text{Amps}}$$

$$V_{BE} = V_{BB} + V_{IN} = .8V + \cancel{\ldots} = \cancel{\ldots\ \text{Volts}} = .8$$

$V_{BB} + .2 = 1$ Volt

$V_{BB} - .2 = .6$ Volt

Loop 2: $V_{cc} - I_C R_c - V_{CE} = 0$

$I_C = 0 \quad V_{CE} = V_{cc}$

$V_{CE} = 0 \quad I_C = \frac{V_{cc}}{R_c}$

MOV src,dst

MOV B

1001	src	dst

src [ ] dst [ ]

N-HOB (7.)

CLRB

7	0

INCB

DECB

2's comp.

MAXINT → 01111111 $(2^8-1)-1$
→ +127

MININT → 10000000 $-(2^8-1)$
→ -127

NEGB

TSTB

ASRB

CARDINAL MAXINT → 11111111 → 255

ASLB

ADCB

SBCB

MOVB

CMPB

Strings

Linked lists? → overhead

Arrays? ~~....~~ → extra empty space

132 characters S1: .BLKW 133.

?	C1	C2	C3	.	.	. . . .	
0	1	2	3	4	5	. . .	132

LENGTH

3	A	B	C

COPY: MOV #132, R0
LOOP: MOVB S1(R0), S2(R0)
DEC R0
BGE LOOP

COPY: MOVB ~~#~~ S1, R0 → sign exchange causes this to break when length is >127.

LOOP: MOVB S1(R0), S2(R0)
DEC R0
BGE LOOP

~~TMP: .BLKW 1~~
MOVB S1, TMP
~~MOVB #0, TMP~~
CLRB TMP+1
MOV TMP, R0

BR DST

OPCODE	offset

BRANCHES to 2* offset
can't reach very far

JMP DST

OPCODE	DST DESC

ANY ADDRESSING MODES ARE ALLOWED
so any address is reachable

BEQ X // too far, so

BNE NEXT
JMP X
NEXT:

Addressing Mode Review   MOV ____, R0

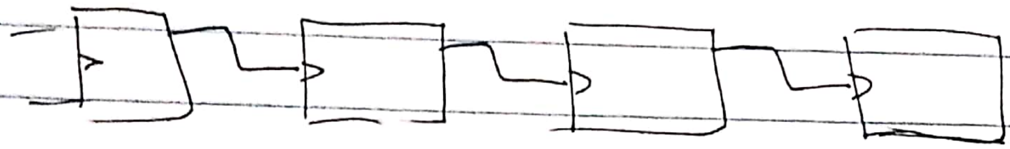
MOV 16., R0 → Relative Mode Addressing

Indirect Addressing   (4 of them)
Example (Post Office Box with mail forwarding)

MOV @4., R0   use the contents of 4 as the address
7,7,   and store the contents of this address
111 111   in register 0

MOV @#28., R0
Absolute Mode Addressing

1.



111
011
001
000

111
110
101
000
011
010

$11_{10} \rightarrow 1011_2$

$111 \rightarrow 7$
$110 \rightarrow 6$
$101 \rightarrow 5$
$100 \rightarrow 4$
$011 \rightarrow 3$
$010 \rightarrow 2$
$001 \rightarrow 1$
$000 \rightarrow 0$

.25 HZ .5Hz 1HZ

Switch case is not fair. It takes longer to get to the 4^{th} case than the 1^{st}, since it is ~~essentially~~ Pretty much a big if else if...

```
I:          .BLKW   1
TABLE:      .WORD   L10     1   0
            .WORD   L20     2   2
            .WORD   L30     3   4
            .WORD   L40     4   6
```

; Table is an array [1, 4] of ~~type~~ address

Table(I)

Turn Index into offset

```
Mov I,R0
Dec R0      ; bring down one
ASL R0      ; times 2 because addresses occupy words
-

JMP @TABLE(R0)
```

0 Register

1 Ind. Register

2,7 Immediate

~~3~~,7 Absolute

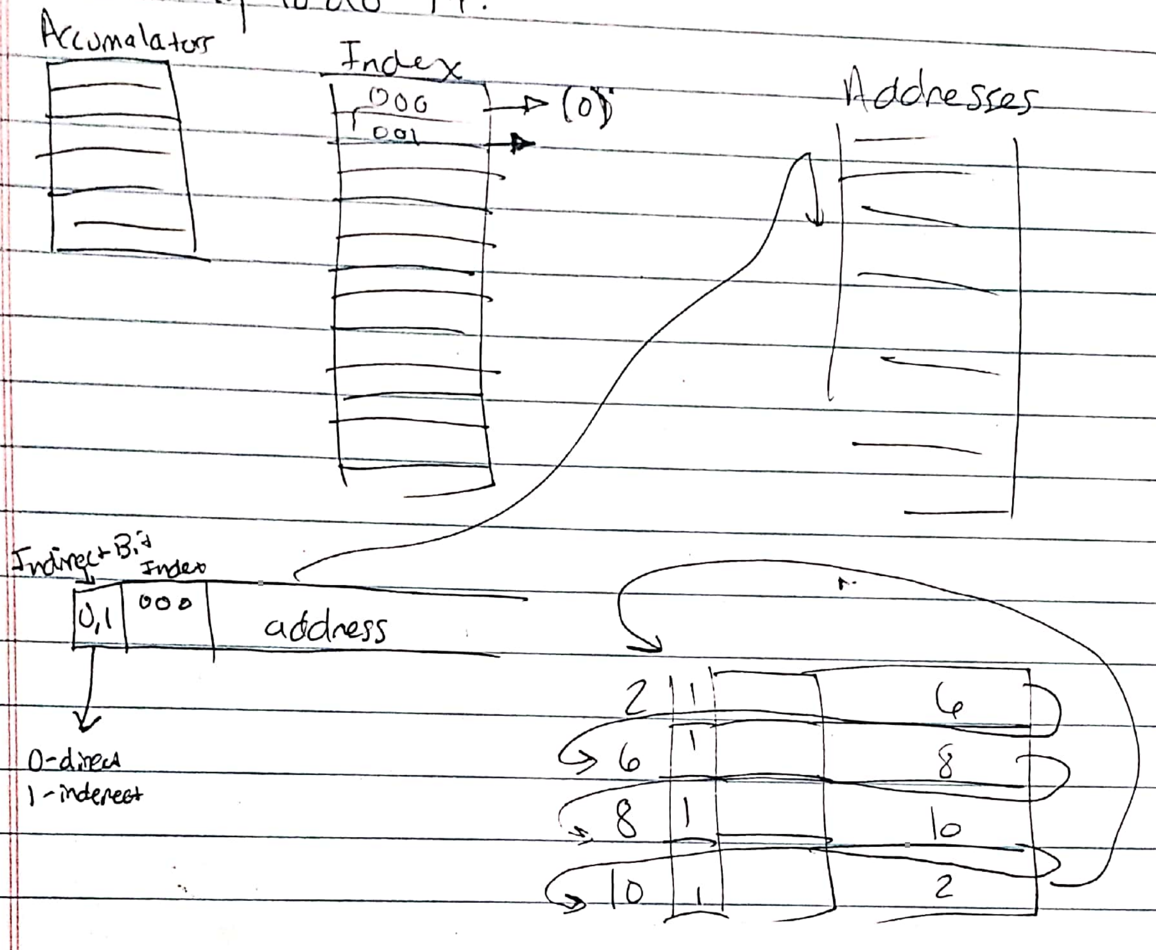
6,7 Relative

7,7

6,i Index

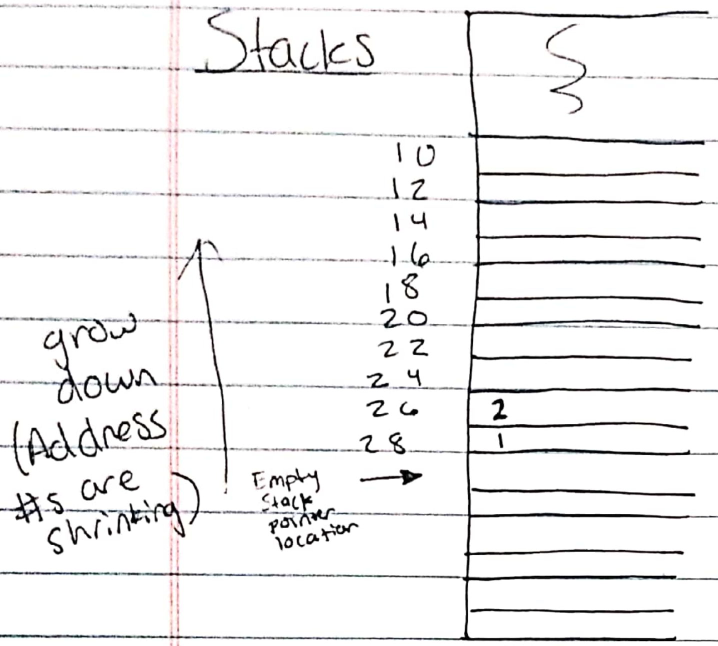
7,i Ind. Index

Other way to do it.



Indirect address Loop

Stacks



STACK: .BLKW 10.
EMPTY:

CREATE:
MOV #Empty, R0 ; Address
PUSH: CMP R0, #STACK
BEQ FULLSTACK
SUB #2., R0
MOV ITEM, @R0

Stackpointer
STACK + 10.*2bytes
10 + 20 = 30

POP: CMP R0, EMPTY
BEQ EMPTY STACK
MOV @R0, wherever
ADD #2, R0

ITEM	R0
1	~~30~~
2	~~28~~
	~~26~~
POP	~~26~~
2 goes wherever	28

→ 2 stays in memory, the stack just doesn't care about it.

Addressing Mode 4 – Auto decrement mode addressing

4,0 → Register 0

-(R0)

use the address inside the register and decrement ~~by 2 if word instruction~~ or 1 if byte instruction.

PUSH: MOV ITEM, -(R0)

Registers 6 and 7 only decrement by 2

POP: MOV ITEM, (R0)+

PC must be even. Register 6 is Stack Pointer (Special Stack)

Auto Dec
4,0 — -(R0) — decrement then use indirectly

MOV ITEM, -(R0)

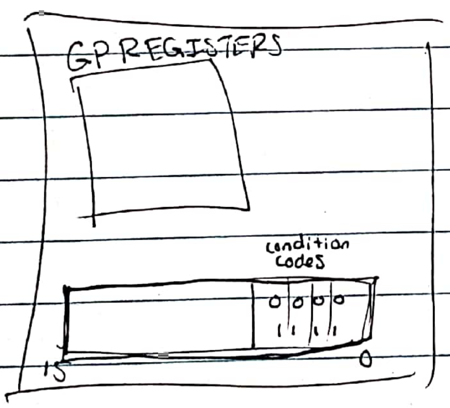
Auto Inc
2,0 — (R0)+ — use indirectly, then increment

mov +(R0), whatever

MOV #2, R0

0001	010 111	000000		0 ——— 010
	20.			22.

PC
~~20.~~
22.



clear / set

N Z V C

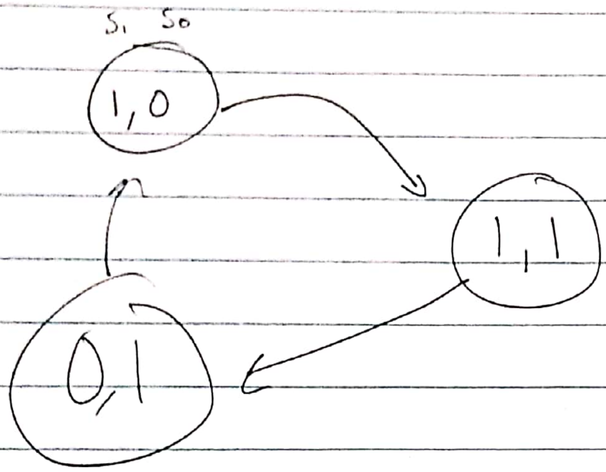
0000 0000 101 | 0/1 | | | | |

		clear
CLC →	0000 0000 101	0 0001
CLV	———	0 0010
CLZ	———	0 100
CLN	———	1000
CCC	———	1111
SEZ	———	1 0100
SEV	———	1
SEN	———	1
SEC	———	1
SCC	———	1 1111

NOP — 0000 0000 101 0 0000 — clear Nothing

No op

Time Delay or Padding



A \ B	1	0
1	0	1
0	1	?

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

A B

OLD	NEW
1 0	1 1
1 1	0 1
0 1	1 0
Q	$\bar{A} + \bar{B}$ A

A \ B	1	0
1	1	1
0	0	?

A

NEW ~~STATES~~ INPUTS

	J_1	K_1	J_2	K_2
1 0	1	0	1	0
1 1	0	1	1	0
0 1	1	0	0	1
	$A \oplus B$	AB	A	$\bar{A}B$

Scanned with CamScanner

$L = \{a^n b^{2n} a^{4n} : n \geq 0\}$

Not Context Free

push a's on to stack

pop b's $\rightarrow$ stack is empty, can't keep track of 4n for the second a

$w = a^k | b^{2k} | a^{4k}$ at least k symbols, is accepted by L

1 2 3

v, y

(1,1)	pump up once	$a^{k+p} b^{2k} a^{4k}$	$1 \leq p \leq k$	not accepted
(2,2)	——	$a^k b^{2k+p} a^{4k}$	——	not accepted
(3,3)	——	$a^k b^{2k} a^{4k+p}$	——	not accepted
(1,2)	——	$a^{k+p} b^{2k+q} \underbrace{a^{4k}}_{\text{not enough}}$	——	——
(1,3)	——	$a^{k+p} b^{2k} a^{4k+p}$	——	v and y are more than k apart
(2,3)	——	$\underbrace{a^k}_{\text{not enough}} b^{2k+p} a^{4k+q}$	——	not accepted

Span regions pumping up will get them out of order

$L = \{a^{n^2} : n \geq 0\}$

NCF can't keep track of n^2 with just a stack

a^{k^2}

1

Since we only have one region, v and y must fall under a^{k^2} so $\omega = a^{k^2+p}$ $1 \leq p \leq k$

if we pump up once $\omega = a^{k^2+1}$ $vy = a^p$

The next string after a^{k^2} is $a^{(k+1)^2}$ a^{k^2+p}

so $k^2 + \underbrace{2k+1}$

$p = 2k+1$ and ~~p~~

so too big $|vy| \leq k$

$L = \{ w \in \{a,b,c\}^* : \#_a(w) = \#_b(w) = \#_c(w) \}$

$L' = L \cap a^* b^* c^*$

$= \underbrace{a^n b^n c^n : n \geq 0}$

context free $\cap$ regular = context free

so must also be context free

~~z~~ $w = aabbcc$ not k length

$w = a^k | b^k | c^k$

① ② ③

(1,1) : a^{k+p} $1 \leq p \leq k$ ~~so~~ $a^{k+p} b^k c^k$ not accepted

~~(1,2)~~ :

(2,2) : $a^k b^{k+p} c^k$ $1 \leq p \leq k$

(3,3) : $a^k b^k c^{k+p}$

(1,2) : $a^{k+p} b^{k+q} c^k$

(2,3) : $a^k b^{k+p} c^{k+q}$

(1,3) : too far apart vy

cross Regions : if crossed regions, pump would take letters out of order

$L = \{a^n b^m a^n b^m a^n \ : n,m \geq 0\}$

can't keep track of n or m more than once

or ~~at the same time~~ in this order

$$\underset{1}{a^k} \,\Big|\, \underset{2}{b^k} \,\Big|\, \underset{3}{a^k} \,\Big|\, \underset{4}{b^k} \,\Big|\, \underset{5}{a^k}$$

(1,1): $a^{k+p} b^k a^k b^k a^k$

(2,2): $a^k b^{k+p} a^k b^k a^k$

(3,3): $a^k b^k a^{k+p} b^k a^k$

(4,4): $a^k b^k a^k b^{k+p} a^k$

(5,5): $a^k b^k a^k b^k a^{k+p}$

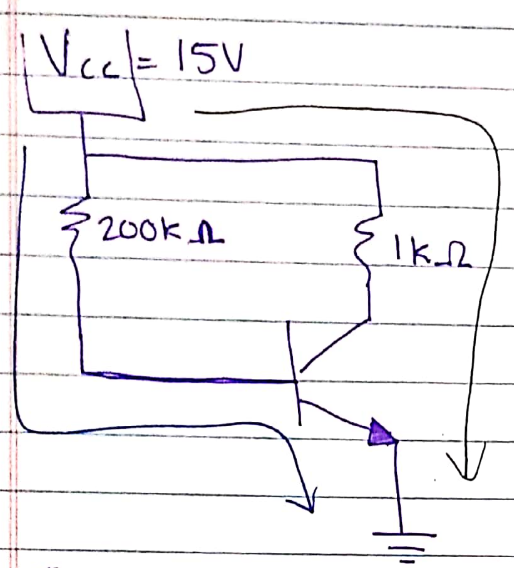
(1,2):

(2,3):

(3,4):

(4,5):

span regions:



Or assume saturation

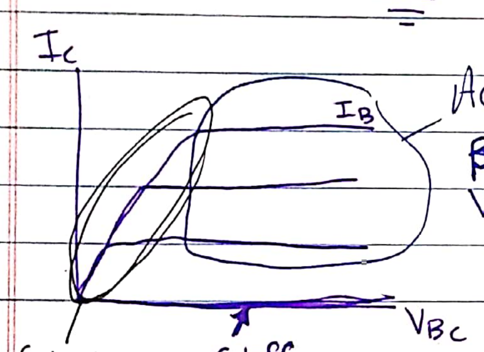
$15 - I_c(1000) - .2 = 0$

$I_c \cong 14.8\text{mA}$

$\beta = \frac{14.8\text{mA}}{72\mu A} \cong 200$

not to spec

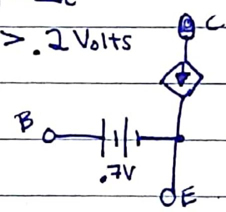
"proved" wrong



Active

$\beta I_B = I_c$

$V_{BE} > .2$ Volts



Saturation

$\beta I_B >> I_c$

(Usually used for switching)

V_{BC} is small

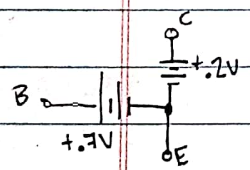
Cutoff

When $V_{BE} < 0.5V$

or

$I_B = 0$

Not cutoff



$I_B = \frac{15 - .7}{200k} \cong 72\mu A$

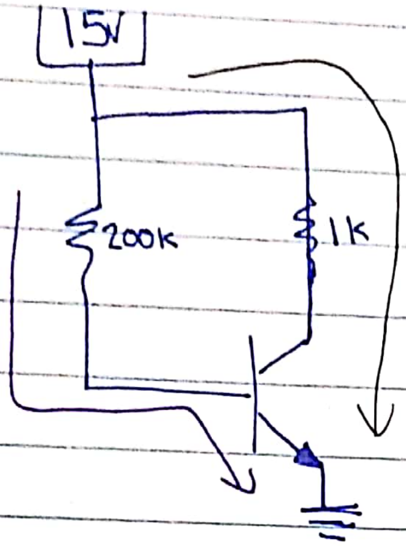
Assume Active → $I_c = 100(72\mu A)$

$= 7.2\text{mA}$

$15 - 1000(I_c) - V_{CE} = 0$

$15 - 7.2 = V_{CE} \cong 7.8$ Volts

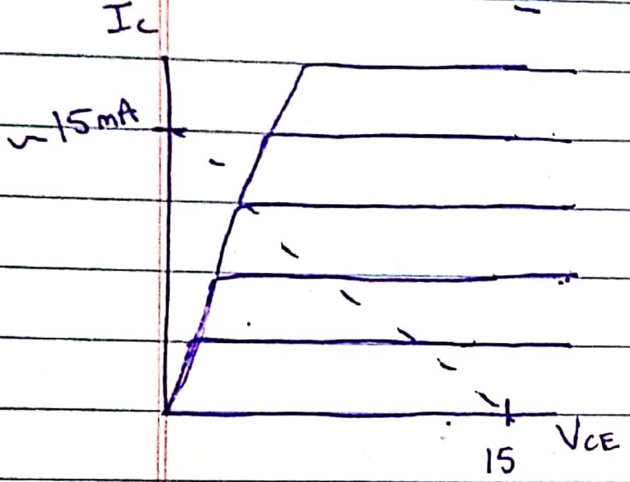
Makes sense for active region



$\beta = 300$

$15 - 200k(I_B) = V_{BE}$

Not in cutoff



__Assume ~~Active~~__

$15 - I_B(200k) \doteq .7$

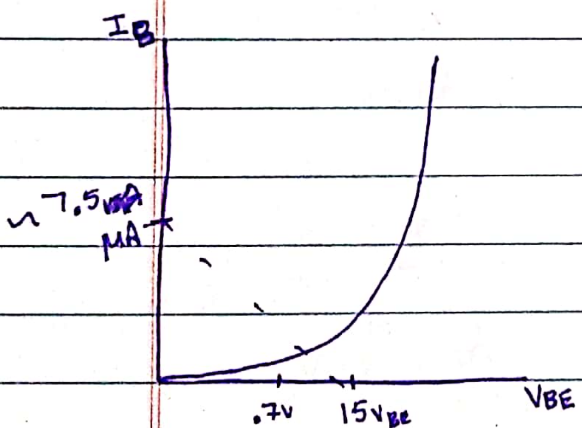
$I_B \doteq 72\mu A \qquad I_c = 300(72\mu A) \doteq 21.6mA$

$15 - 1k(I_c) = V_{CE} \rightarrow 1kI_c = 15 \quad I_c = \frac{15}{1000}$

$15 - 1000(21.6mA) = V_{CE} \qquad 15 = V_{CE}$

$15 - 21.6 = V_{CE} \doteq -6.6$

$\boxed{\text{So not active}}$



__Saturation__

$15 - I_c(1000) = V_{CE}$

$15 - I_c(1000) = .2$

$I_c = \frac{14.8}{1000} = 14.8mA$

$\beta = \frac{14.8}{72\mu A} \doteq 200$

$200 < 300$ ✓

$15 - I_B(200k) = V_{BE}$

$I_B = \frac{15}{200k}$ when $V_{BE} = 0$

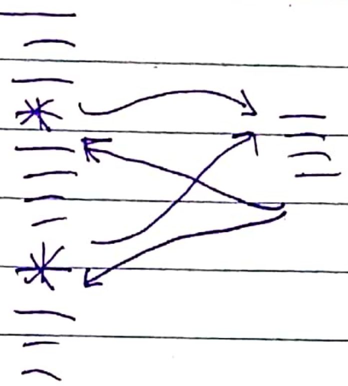
$V_{BE} = 15$ when $I_B = 0$

Scanned with CamScanner

"Subroutine" - function, method, procedures, etc.

- Open (Macros)
  - "Copy Paste"

- Closed
  - goto sub and come back



Linkage Problem

1. Get to Sub
2. Get back from Sub
3. share data

Java, C++, Curly Brackets

Newer higher level languages are "block structured"

Original higher ——— "non-block structured"

- Methods and Main Programs are compiled separately and Linked together by the Linker

MOV #RA1, SUB
JMP SUB+2

RA1 *

RA2 *

SUB: RA1 ← use more space here for data passing

JMP @ SUB

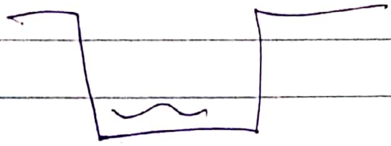
JSR R_i, dst

↓ (SP) ← (R_i)

(R_i) ← (PC)

(PC) ← dst

Assumes properly formed stack at SP



RTS R_i

(PC) ← (R_i)

(R_i) ← (SP)↑

TABLE:

	# of arguments
address A_1	
address A_2	
address A_3	

3

mov #ARGTBL, R5

JSR PC, ADDER

RA:

ITMP=5

~~I = ITMP+K~~

RETURN

```
ARGTBL  .WORD 3
        .WORD IA
        .WORD IB
        .WORD IC
IA:     .BLKW 1
IB:     .BLKW 1
IC:     .BLKW 1
```

```
ADDER: MOV @4(R5), ITMP
       MOV ITMP, @2(R5)
       ADD @6(R5), @2(R5)
       RTS PC
```

2 more pages for block structure version

Non-Restoring Division Example (5 bit)

$X = 15_{10} = 01111_2$ $N=5$

$Y = 4_{10} = 00100_2$

1. Make X double precison $01111 \rightarrow 00000\ 01111$

   Loop Counter $= N-1 \rightarrow 5-1=4$ $i=4$

   $00100 * 2^4$ $2^i * Y$ (shift left i times)

2. → X 00000 01111

   → Y 00010 00000 subtract

   11110 01111

3. if negative $q_i = 0$ else $q_i = 1$

   $i\text{--} = 1 \rightarrow 3 \rightarrow 2$

   $q_4 = 0$
   $q_3 = 0$
   $q_2 = 0$
   $q_1 = 1$
   $q_0 = 1$

4. if neg add $2^i * Y$ to reslt else sub $2^i * Y$

```
            11110  01111
          + 00001  00000    neg
            11111  01111    goto step 3
q3 = 0    + 00000  10000
            11111  11111
q2 = 0    + 00000  01000
q1 = 1 → 1: 00000  00111
          − 00000  00100    sub because last answer was non neg
q0 = 1      00000  00011    quotient cause i=-1
                            00011
```

All double Precision Arithm.

Adder (I, J, K)
Var ~~I~~ ITmp : INT
ITMP = J
I = ITMP + K
Return

Build a record (struct)

ITMP	Arg IA	Arg IB	Arg IC	Ret. Addr

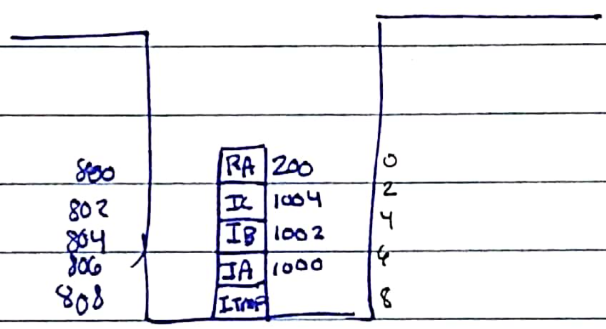
"Activation Record"
"Stack Frame"

Pointer to NORMALLY BUT LETS ~~KEEP~~ KEEP IT SIMPLE → Heap

Driver

Build, use, deconstruct

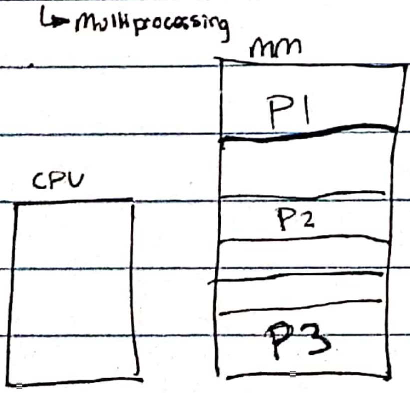
SUB #2, SP
MOV #IA, -(SP)
MOV #IA, -(SP)
MOV #IB, -(SP)
MOV #RA, -(SP)
JMP ADDER
RA: ADD #10, (SP)



To Allow for nested calls let's put our activation record on a stack

SUBROUTINES SUPPORT
1. REUSABLE
2. RECURSIVE
3. REENTRANT
   ↳ Multiprocessing

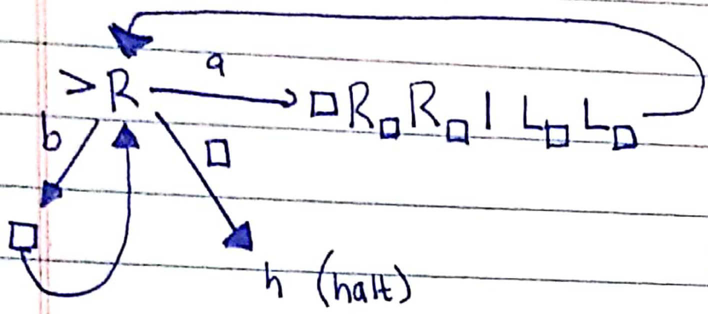
1000	2	IA :
1002	10	IB :
1004	20	IC :



ADDER: MOV @4(SP), 8(SP) COPY IB TO ITMP
MOV 8(SP), @6(SP) COPY ITMP TO IC
ADD @2(SP), @6(SP) ADD IA TO IC
JMP @(SP)

STATE INFORMATION
→ COPYING This to main memory we call a CONTEXT SWITCH

Turing Machines



□aba□

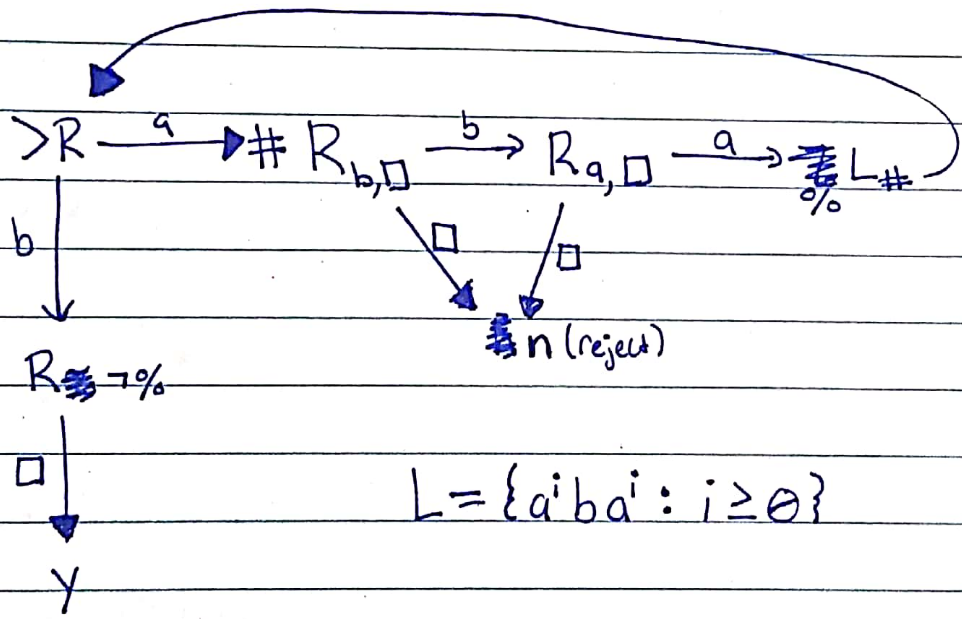
↓

□□ba□1

□□□a□1□1

□□□□□1□1□1

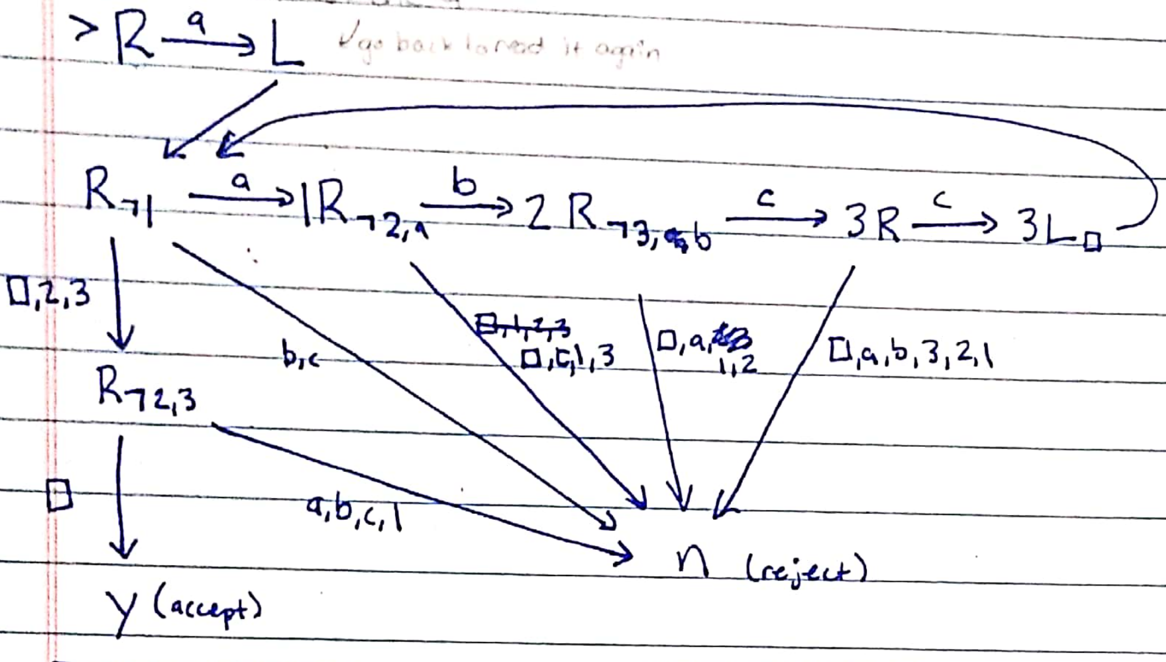
~~□□□□h1□1□1~~



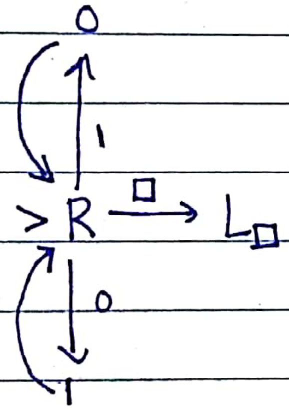
$L = \{a^i b a^i : i \geq 0\}$

$L = \{a^n b^n c^{2n} : n \geq 1\}$

at least one a



Compute the complement of a binary # on the tape

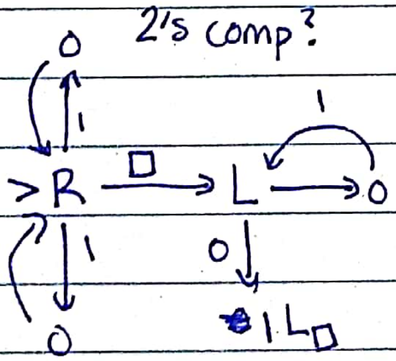


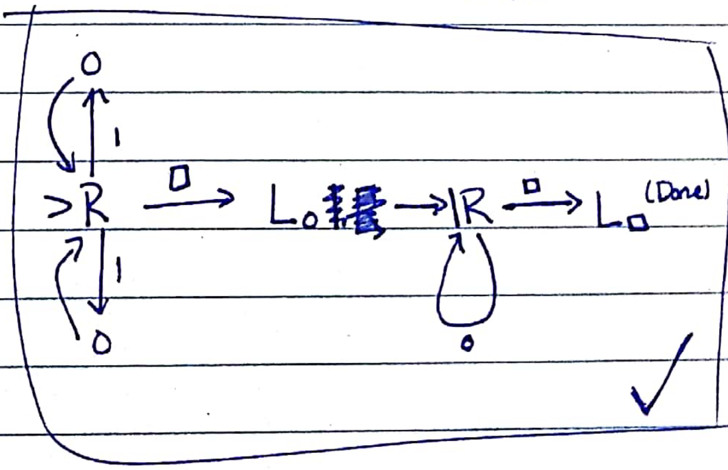
□ØXXØXXX□
□1001000□

~~100011~~
100100

Find the first 0 from ~~the right and add~~ flip to 1, flip all ~~bits to the right~~ to 0s

2's comp?





Double Binary #

$>R_\square 0 L_\square$ (shift left 1)

Halve # (no leading zero)

$$\begin{tikzcd} >R \arrow[r, "1"] \arrow[d, "0"'] & R \arrow[r, "{0,1}"] \arrow[d, "\square"'] & R_\square L \square L_\square \\ L & L0L \end{tikzcd}$$

Russian peasant Multiplication (2 binary #s)

$a \times b$

$2a \times \frac{b}{2}$ b even

$2a \times \frac{b-1}{2} + a$ b odd

$A>B$

e_2

A_1A_0 \ B_1B_0	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

$A_0\bar{B}_1\bar{B}_0$

$\bar{B}_0A_1A_0$

$A_1\bar{B}_1$

$A=B$

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

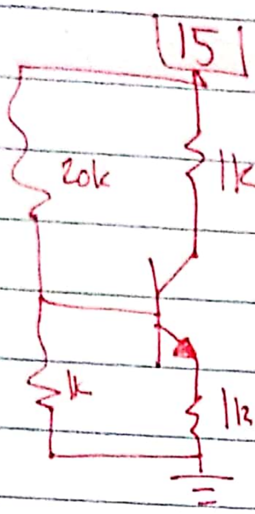
$A<B$

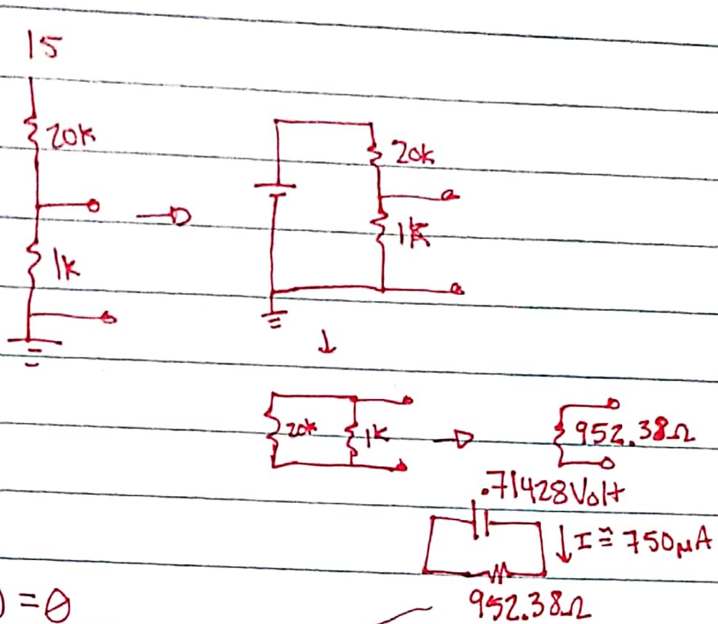
A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

~~B~~ $\bar{A}_1B_1$

$B_0\bar{A}_1\bar{A}_0$

$B_1B_0\bar{A}_0$





$15 - 20k(I_B + I_G) - 1k(I_G) = 0$

$-20k I_B - 20k I_G - 1k I_G = -15$

$-20k I_B - 21k I_G = -15 \rightarrow$

I_B	I_G	V_{CE}	I_c	
$-20k$	$-2k$	0	0	-15

$15 - 1k I_c - V_{CE} - 1k I_E = 0$

$-V_{CE} - 1k I_c - 1k(I_B + I_c) = -15$

$-1k I_B - V_{CE} - 2k I_c = -15 \rightarrow$ $-1k \quad 0 \quad -1 \quad -2k \mid -15$

$15 - 20k(I_G + I_B) - .7 - 1k I_E = 0$

$-20k I_B - 20k I_G - 1k(I_B + I_c) = 14.3$

$-20k I_B - 20k I_G - 1k I_B - 1k I_c = -14.3$

$-21k I_B - 20k I_G - 1k I_c = -14.3 \rightarrow$ $-21k \quad -20k \quad 0 \quad -1k \mid -14.3$

Week from thursday
Today is Oct 21st, Tuesday

Nth Prime Number

Write a program to find the Nth prime number. For our purposes, the first three primes are 2,3 and 5.

Do not use arrays or indexed mode addressing. This eliminates a class of algorithms such as sieves.

While the restriction above limits your ability to write a real efficient program, you should try to be as efficient as possible given these limitations.

```
;
;       Data Area
;
N:      .BLKW    1          ;IN   - 2's complement > 0
PRIME: .BLKW     1          ;OUT - 2's complement Nth prime or
                            ;         -32768. As error msg.

        .END
```

4.15. An *npn* transistor has $V_{BE} = 0.7\,\text{V}$ for $I_E = 10\,\text{mA}$. Find V_{BE} if $I_E = 1\,\text{mA}$. Repeat for $I_E = 1\,\mu\text{A}$. Assume a temperature of 300 K.

D4.16. Beta meter. Design a "β-meter" for measurement of the β of small-signal *npn* silicon transistors at room temperature. Assume that $v_{BE} = 0.7$ for the transistors to be measured. The following parts are available:

1. A 1-mA-full-scale meter having a resistance of 150 Ω.
2. Standard 5%-tolerance resistors. (See Appendix A.)
3. Potentiometers of 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ, and 1 MΩ.
4. A 4.7-V Zener diode.
5. Switches and mechanical components, as required.
6. A 9-V "transistor" battery.

The meter is to have switch-selectable full-scale values of $\beta_{FS} = 10, 100$, and 1000. Adjustments are to be provided that allow calibration of the meter, which is to provide accurate readings for battery voltages ranging from 7 to 9 V. Under reasonable operating conditions (including short-circuited test terminals), the battery drain should not exceed 5 mA.

Suggestions:

1. *Use a Zener diode regulator to obtain a constant source voltage for delivering base current to the transistor.*
2. *Use a switch and resistor network to apply a suitable base current to the transistor under test for each range of β to be measured.*
3. *Use the meter to measure the collector current with full scale corresponding to $\beta = 10, 100$, or 1000.*

4.17. Use SPICE to obtain output characteristics for an *npn* BJT having $I_s = 10^{-16}\,\text{A}$, $\beta = 200$, and $V_A = 25\,\text{V}$. Allow v_{CE} to range from 0 to 10 V, and let $i_B = 0, 10, 20, 30, 40$, and $50\,\mu\text{A}$.

4.18. Use SPICE to obtain input characteristics for an *npn* BJT having $I_s = 10^{-16}\,\text{A}$, $\beta = 200$, and $V_A = 25\,\text{V}$. Set $v_{CE} = 10\,\text{V}$, and allow v_{BE} to range from 0 to 0.7 V.

Section 4.2: Load-Line Analysis of a Common-Emitter Amplifier

4.19. What can cause distortion in BJT amplifiers?

4.20. Consider the circuit of Figure 4.10. Assume that $V_{CC} = 20\,\text{V}$, $V_{BB} = 0.8\,\text{V}$, $R_B = 40\,\text{k}\Omega$, and $R_C = 2\,\text{k}\Omega$. The input signal is a 0.2-V-peak, 1-kHz sinusoid given by $v_{in}(t) = 0.2\sin(2000\pi t)$. The common-emitter characteristics for the transistor are shown in Figure P4.20. Find the maximum, minimum, and Q-point values for v_{CE}. What is the approximate voltage gain for this circuit?

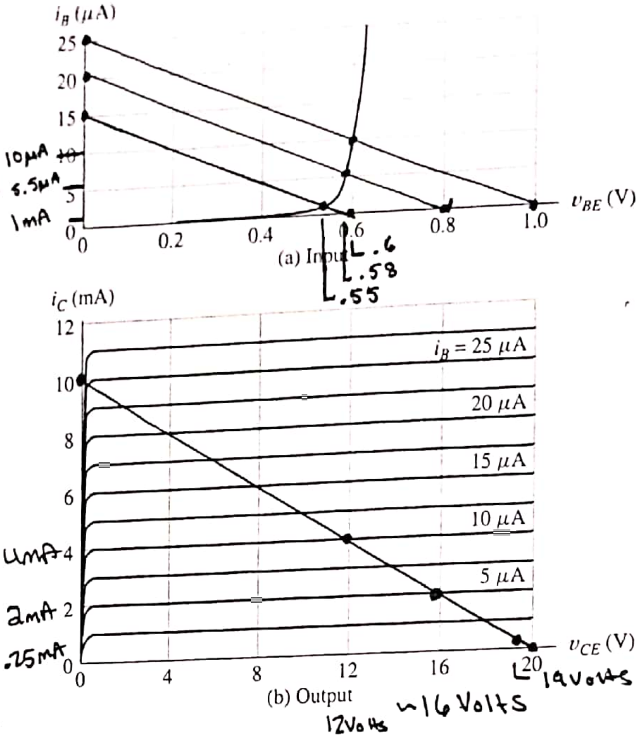


Figure P4.20

4.21. Repeat Problem 4.20 for $R_C = 10\,\text{k}\Omega$. What can you say about the waveform for $v_{CE}(t)$?

4.22. Repeat Problem 4.20 for $V_{BB} = 0.3\,\text{V}$. Why is the gain so small in magnitude?

Section 4.3: The *pnp* Bipolar Junction Transistor

4.23. Draw the circuit symbol for a *pnp* BJT. Label the terminals and the currents. Choose reference directions that agree with the true directions of the current for operation in the active region.

4.24. A certain *pnp* silicon transistor has $\beta = 100$ and $i_B = 0.1\,\text{mA}$. Sketch i_C against v_{CE} for v_{CE} ranging from 0 to $-5\,\text{V}$. Repeat for $\beta = 300$. Ignore second-order effects.

4.25. At a temperature of 30°C, a particular *pnp* transistor has $V_{BE} = -0.7\,\text{V}$ for $I_E = 2\,\text{mA}$. Estimate V_{BE} for $I_E = 0.1\,\text{mA}$ at a temperature of 180°C.

Section 4.4: Large-Signal DC Circuit Models

4.26. Draw the large-signal dc circuit model for a silicon *npn* transistor operating in the active region at room temperature. Include the constraints of any currents or voltages that guarantee