*** this was recreated spilled stuff on original

Date

Physical Memory = List of frames
Page table = List of Pages
TLB table = List of Pages
LRU stuff = Stack of Pages
LRU Page = Stack of Read
Back store = File
Statistic = new static,

Frame
    title =
    DirtyBit
    Section = List of 256 bytes

Page
    title
    Valid
    frame*

Statistics
    Number of calls
    Number of hits
    Number of Page faults

High level view of what the main characters in this program

I understand that I could of just did Physical Memory without this and just do char PhysicalMem[65000], But I felt at the time it would just Be easier this way

this page and the next is just overviews of my thinking

Date

I used Bits from each

```
while read R   Number of calls
  Raw = Addresstxt read[ split()
  Address = Raw[0]
  Commad = Raw[1]
  Page = upper Bits
  offset = Lower Bits
  if TLBTabe[Pay].isvalid() o. T
      PhysicalMemory[TLB[Pay].frame]
      if Commad
      farame is dor
      Els
        Read
      tLB Hit
  el if PageT[Pay].isvalid
      Pa
      Fu
  Else
      Page Lower addres up
      read it
      Stor in frame
      updat tLB
      updat Pay
      PageFault+1 e1
  Get Results
```

Readin fm mem

writing / Readin algorithm

Date

HW3

overview

Physical Memory = [ ]

TLB = Dictionary

Pagetable = [ ]

while reading

addr Read in address

int Page, Offset

determine valid

if (TLBHit or Page table Hit)

{

convert to physical address

read into

}

Else // Page Fault

{

Page memory

read from m

Put into main

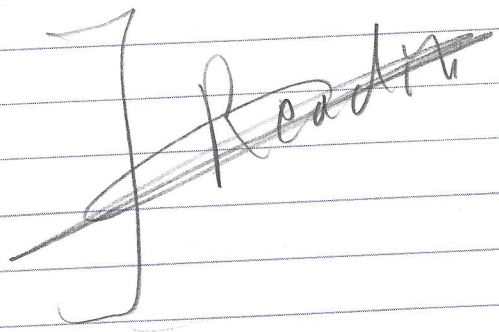
Put in TLB

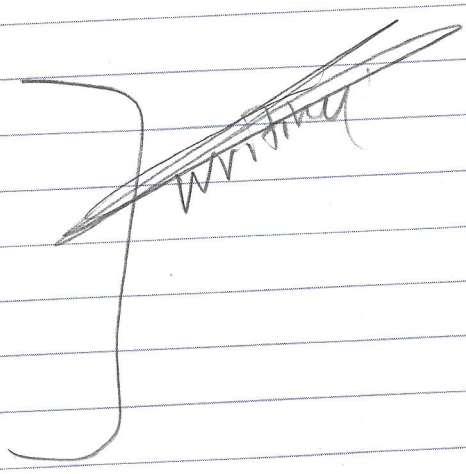
Put into Page

convert the Physical

Read in

read into





temp

pin 3 step | Push step2 pop | Broccoli Put ato | Step1 Find | Step2 | step3 | Date

0 1 2 (3) 4 5 6 7 8 9

0 3

~~30~~ 0 1 2 4 5 6 7 8 9

0 1 2 4 5 6 7 8 9 3
none

0 0 1 2 4 5 6 7 8 9

11 0 1 2 4 5 6 7 8 9

```
Pick (int I)
 int Index=0 ; temp
 for in J=0  J<Size ; J++
     if [J]=I
        index=J
        Break.
     temp=list(J)

 for  index <1  index--
    index[index] = index-1

 List[0]= temp

   return true
```

80
1 12 1 15

128 64 32 16 8 4 2 1
1000 0000

a CH queue or stack couldn't help me so I made my own thing for LRU

we also need a bit to say no good

Class LRU Page Array

Public

int ~~Numbers~~ [Size]
Bool: Page True

Int LRU SIze array ()

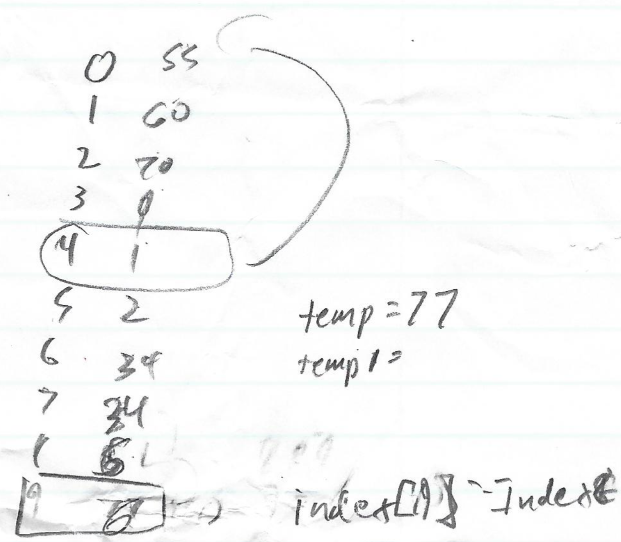
for in 255 -1
size[i]= 254-I;

254
:
0

~~Pick~~ Pop
Int ~~Pick~~ (In)
temp= first size-1;
in temp 1
for in i<255 and i>0
index(i)=index[i-1]
~~index[i-1]=temp[i]~~

Ind 0 = temp

Pick (int)
for int i in 255
If first[i]==int
Break
for in 3 >0 5



C: 11

Add to TLB

int T = LRU Table.pop

fitt[?] 66 wh 320

i = 662

0	-1	No
	-1	No
1		
2	-1	No
3	-1	No
4	-1	No
5	-1	No
6	-1	No
7	-1	No
8	-1	No
9	-1	No
10	-1	No
11	-1	No
12	-1	No
13	-1	No
14	-1	No
15	-1	No

316

16

t = 7

there was a Bug with the TLB LRU counter thing

Date

Page

Date

```
if TLB hit
    Stats NumbeOf TLB ++
    Value
    Put at the top of ~~[illegible]~~ (TLB) upper)

else if Page hit
    Value
    put on top of LRU (Page)
177 put on top of LRU (Page

Put on top oLRU ( pon List , upper), Size)
    for into size
        if   List i = upper

fi
```

Page

Date

else {
find free frame
while i < size
if physicalme[I], free
Break
I++
if i = 255
I = pop front
page[I].is invalid
set page to valid
prev popback[I]

Idea to handle page faults

Page

Date

1 fix the two intermediate queues in inter-departure
Processes
2 for the process output
3

128	64	32	16	8	4	2	1
1	1	0	0	0	0	0	0
1	0	1					
1	0	1	0	0	0	0	
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0

int Numbe
if Numb < 128
~~128~~ Numbe < 128 - 128
Else
return Numbe

working out the
Value>
get values function