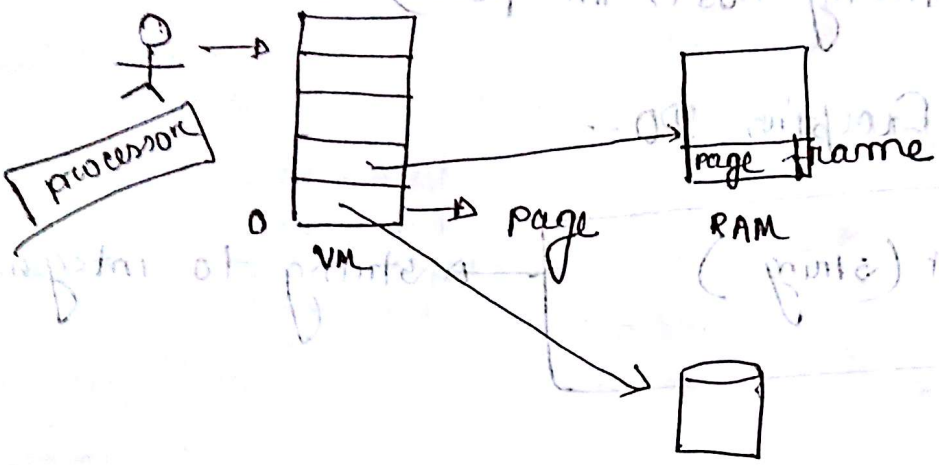
# Nachos - 3




* RAM এ আছে নাকি, disk এ আছে, RAM এর কোথায় আছে?

index

Page Table

disk এ নাকি RAM এ

| vpn | Valid | ppn |
|---|---|---|
| 0 | True | 0 |
| 1 | False | — |

page fault: RAM এ নাই

```
Exception Handler (which) {
    if (which == PageFaultException)
{ int addr = machine → ReadRegister (39);
    pagefaulthandler (add); ← disk থেকে RAM এ আনবে
}.
```

void pagefaultHandler(int addr) {

int vpn = addr / pageSize;

TranslationEntry *pte = &machine → pageTable[vpn];

pageTable এর entry এর pointer.

↳ বর্তমান process এর page table

pte → physicalPage = memoryManager → AllocPage(pte, currentThread → pid);

↳ at any cost একটা physical page দিবে, এটাকে page table এ save করতে হবে।

pte → valid = TRUE;

pte → dirty = ~~FALSE~~ FALSE;

pte → use = FALSE; → এখনো page টা access করি নাই, just নিয়ে আসছি

pte → readOnly = FALSE;

if (vpn's first time fault) {

code segment কোথায় আছে জানতে হবে, code segment এর base address.

cs = currentThread → space → noffH.code.virAddr;

css = ..... → code.size;

ids = ... → ... initData.virAddr;

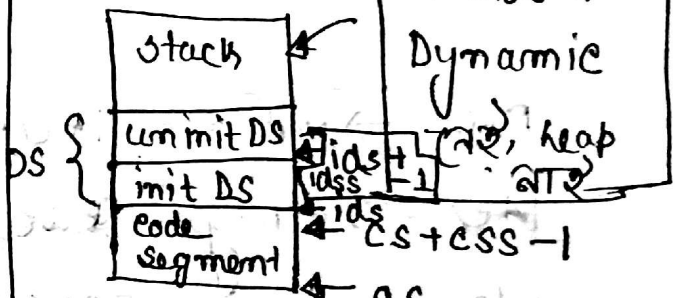
ids এর size 0 হলে, ids এ garbage value থাকে।

idss = → ... initData.size;

if (x >= cs && x <= cs + css - 1)
{
  in code segment.
}

disk → a collection of files, তাই page টা কোথায় আছে (disk এর) জানতে হবে, এটা আমরা → swap space → executable file.

page fault হওয়ার পর executable file এ থাকবে (let).



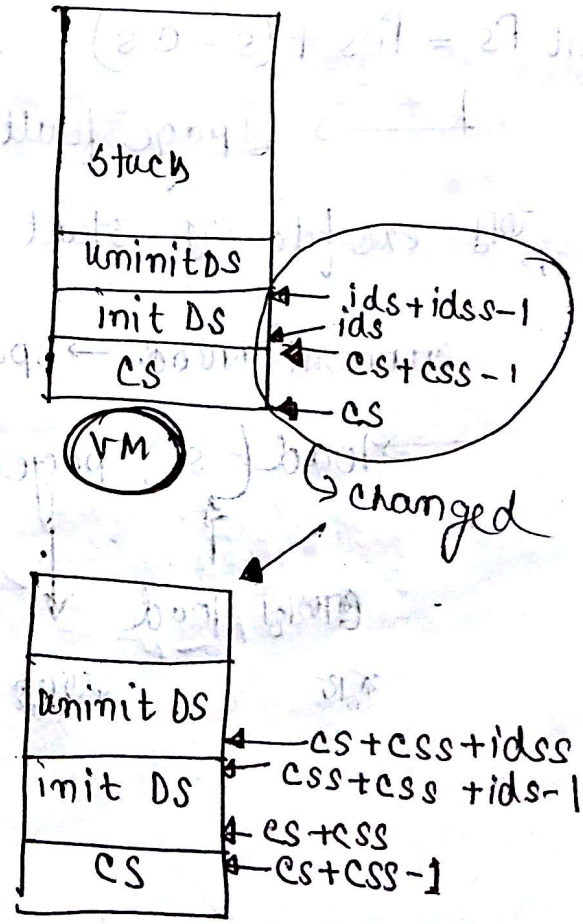

chrome এর memory layout.

```
int b=10;
int a;
int main()
{
}
```

Data segment এ global variable থাকে।

| initialized DS, b=10 |
|---|
| un " " , a |

∴ কোনো segment এর size 0 হলে তার তার attribute এর মান হবে garbage value যেমন, তাই তখন size check করে attribute use করতে হবে। Otherwise এভাবে করতে হবে



int s = vpn * PageSize;

int e = s + pageSize - 1;

↳ যেটার জন্য fault খেয়েছি, তার ending add (virtualk add)

// এটা case এর জন্য condition লিখতে হবে,

if (case 1) {

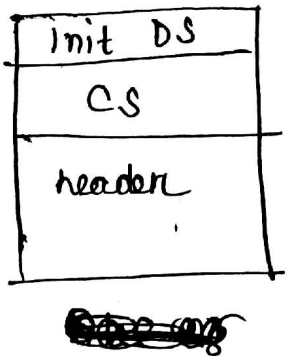
int fs = fcs + (s - cs);

↳ যে page fault হয়েছে, তার exe file এর start

current Thread → space

→ load( s , pagesize, fs)

কোথায় load করব. কতটুকু লোড করব। কোথা থেকে।




executable file

↳ এখানে uninitds নাই, তাই [redacted] চাইলে pagetable 0 করে দিব।


এখানেই 0.

segment, page size এর multiple না।

case 1: s → cs, e → cs

case 2: s → cs, e → init Data

case 3: s → cs, e → ainit/stack

else if (case 2)

{

int fs = fcs + (s - cs);

int size = (ce - s) + 1;

currentThread → space →

load (s, size, fs);

fs = fids;

size = pageSize - size;

currentThread → space →

load (ids, size, fs);

↳ init data segment

↳ start (VM ...)

}

else if (case 3)

{

int fs = fcs + (s - cs);

int size = (ce - s) + 1;

current Thread → space →

load (s, size, fs)

case 4: s → initData,

e → initData.

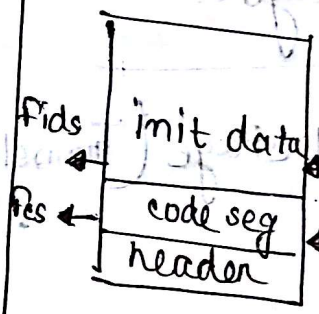
case 5: s → initData,

e → uninitData / stack.

case 6: s → uninitData/

stack

e → uninit / stack




noffH ...

address space ...

instance ...

current Thread →

space → noffH.

fs = fids.

size = idss;

currentThread → space → load. (ids, size, fs);

size = PageSize - (ce - s + 1) * idss;

currentThread → space → clear (cs + css + ids, size);

↑ কোথা থেকে clear করব;

↑ কতটুকু 0 করব.

---

Memory Manager:

→ int AllocPage (Translation *pte, int pid)

* এই page টা কোন process id নিয়েছে
* কোন virtual page এই page frame টা নিচ্ছে, (pte store করব)

Variables:

int * pidarr;

→ constructor এ ফিক্স numPhyPage

(TranslationEntry *)*ft;

save করা pte.

pt[0] = 0

↳ page free আছে।

আমরা constructor এ 0 করে দিব,

pt[0] ≠ 0 → page free নয়

```
int AllocPage (                    )
                ← blocks.
{
    for (int i=0; i<NumPhysPages; i++)
    {
        if (pt[i] == 0) {
            pt[i] = pte;
            pidarr[i] = pid;
            return i;
        }
    }

    int f = frameToReplace
Thread *t = processTable → Get (pidarr[f]);
```

↳ Thread এর pointer return করবে।

প্রতিটা process এর জন্য একটা করে file
↳ যেটা swap space হিসেবে কাজ করবে।

pidarr[f]

t → space → bs → Page Out

(ft[f]);

↳ pageframe টা সরান, যেটাকে swap space এ পাঠাবো।

```
ft[f] = pte;
pidarr[f] = pid;
return f;
}

void Release(int f){ ──→ SC-EXIT এ call হবে।
ft[f] = 0;
}
```

Backing store: ← এটা process টা

OpenFile * f;

BitMap * map; → কোন page swap space এ থাকবে তার জন্য। হবে.

Backing store (int numPage) → কোন process টা জন্য কত page

{ file তৈরী করবে file create করে

তারপর bit Map initialization;

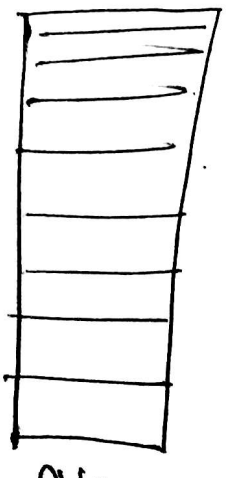
তার Initially সব 0;

}

void PageOut ← RAM থেকে swap space এ চলে যাবে,

(TranslationEntry *pte)

{



pte → valid = FALSE

if ( ! isPresent (pte) || pte → dirty)

→ swap space ও তাকে দিবা.

{

data copy করতে হবে, f → writeAt (pte → PhysicalPage * PageSize, PageSize, baseadd)

pte → virtual Page) * Page size ⇒ আগের same mapping; (file এর কত no offset এ?)

}

map → Mark (pte → virtual Page);

```
void PageIn (TranslationEntry * pte){

    f -> readAt (&machine -> mainMemory [pte -> physicalPage * PageSize], PageSize,
                 pte -> virtualPage * PageSize);

}

bool isPresent (TranslationEntry * pte)
{

}
```

---

Addrspace:

---

variables:

---

```
OpenFile * executable;
NoffHeader noffH;
BackingStore bs;
```

Constructor }

Executable ... ;

// NoffHeader noffH;

} to initialise;

void Load (int to, int size, int infileAddr)

{

executable → ReadAt (&(machine → MainMemory

[pageTable [to/pageSize]. physicalPage *

PageSize + to % PageSize], size, infileAddr);

}

Clear () { bzero }

LRU

int count = 0;

ReadMem, WriteMem (Translate ... [count++]).

Translate ... (

Translate () { ple → LRU = count++;

}