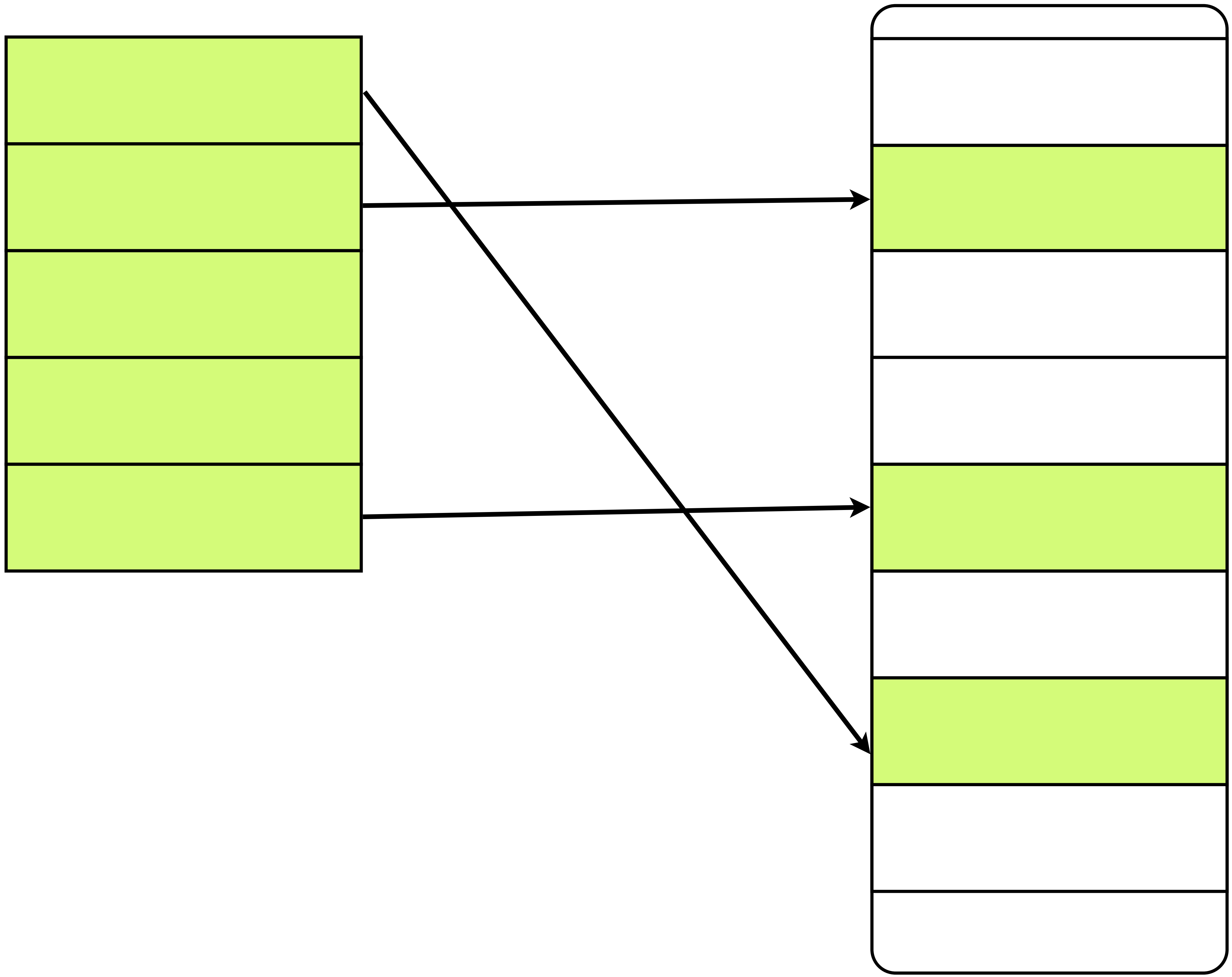
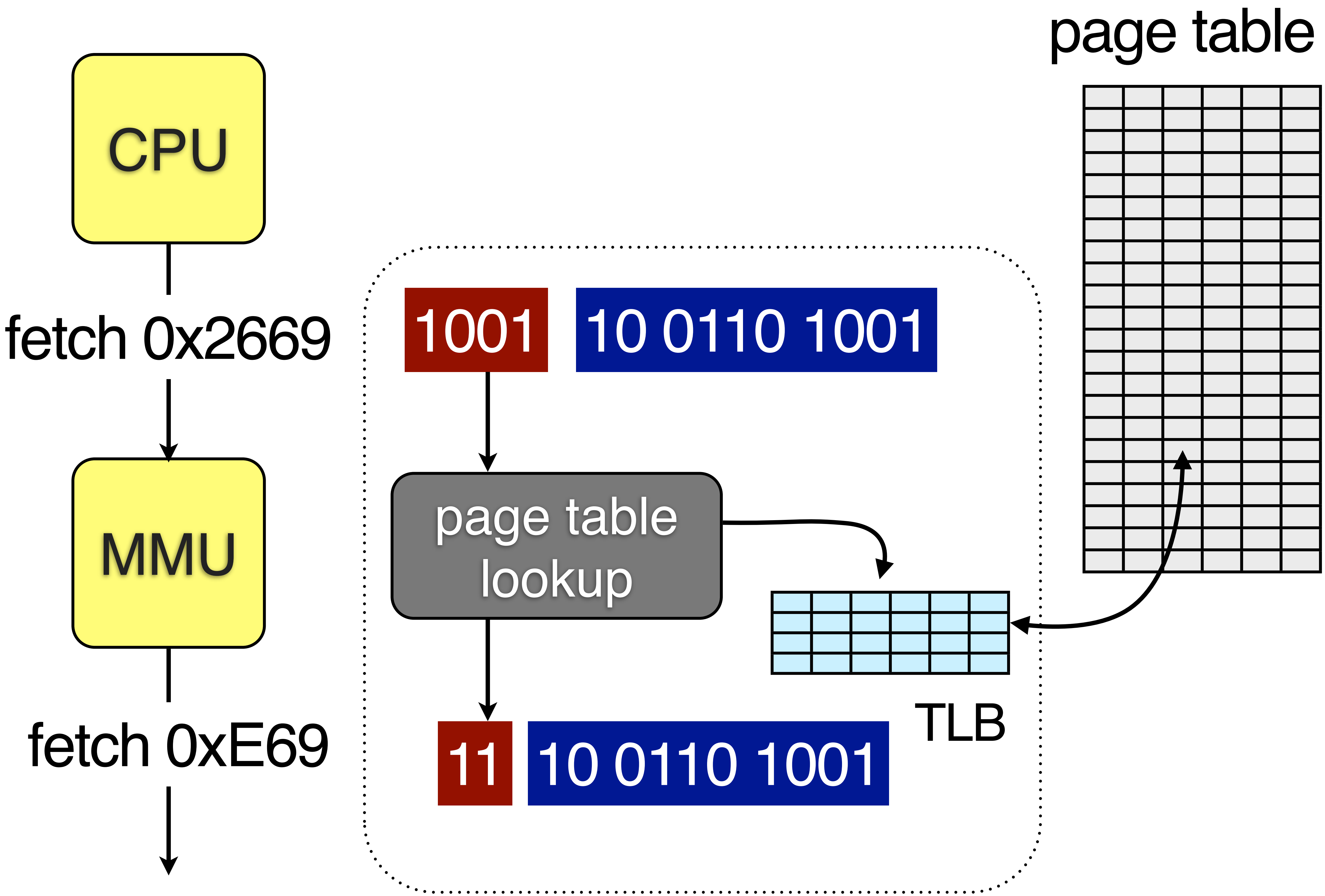
Lecture 9
Memory Management II

21 October, 2011

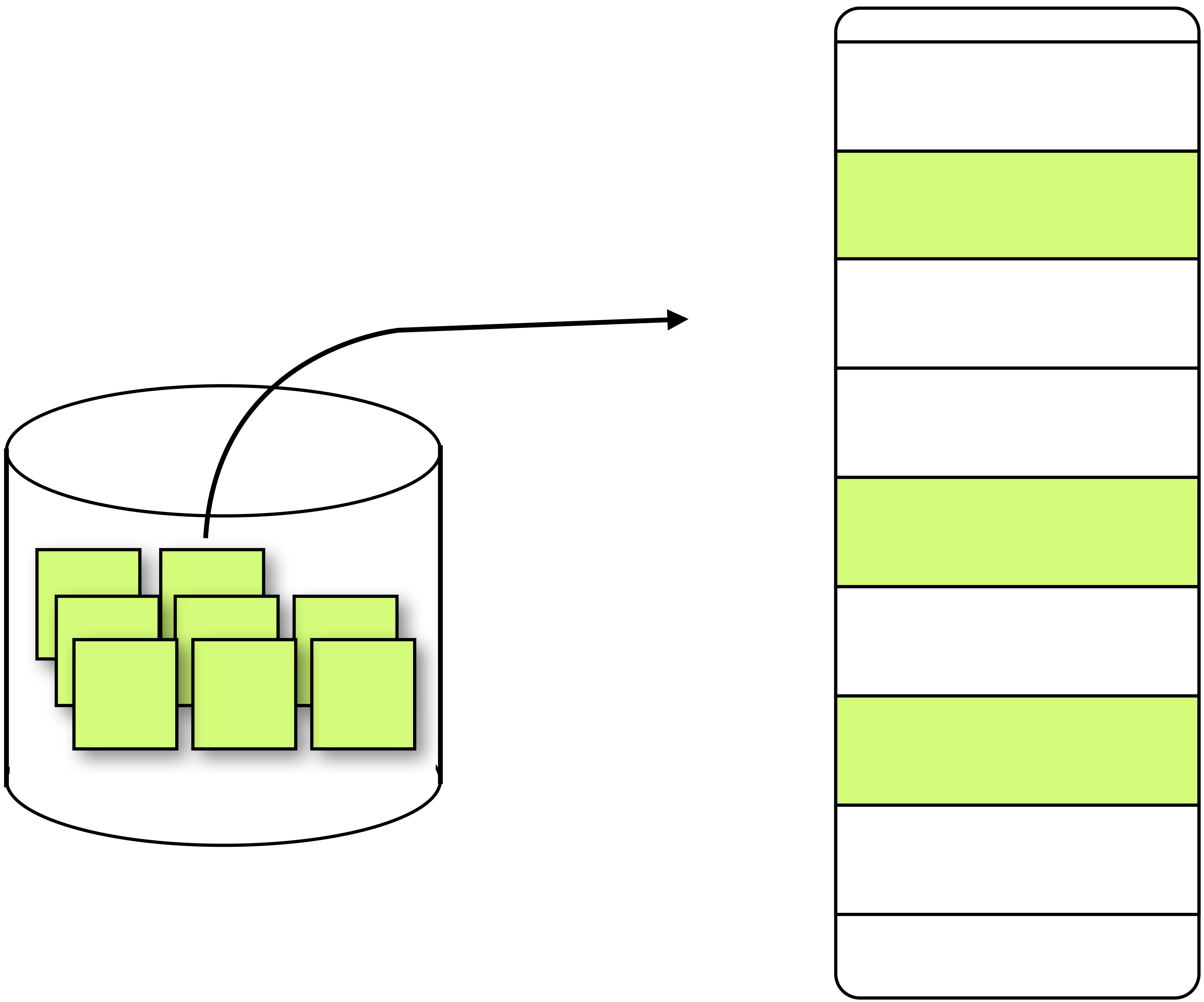
Address Space

Physical Memory




page table
CPU
fetch 0x2669
1001
10 0110 1001
MMU
page table
lookup
TLB
fetch 0xE69
11
10 0110 1001

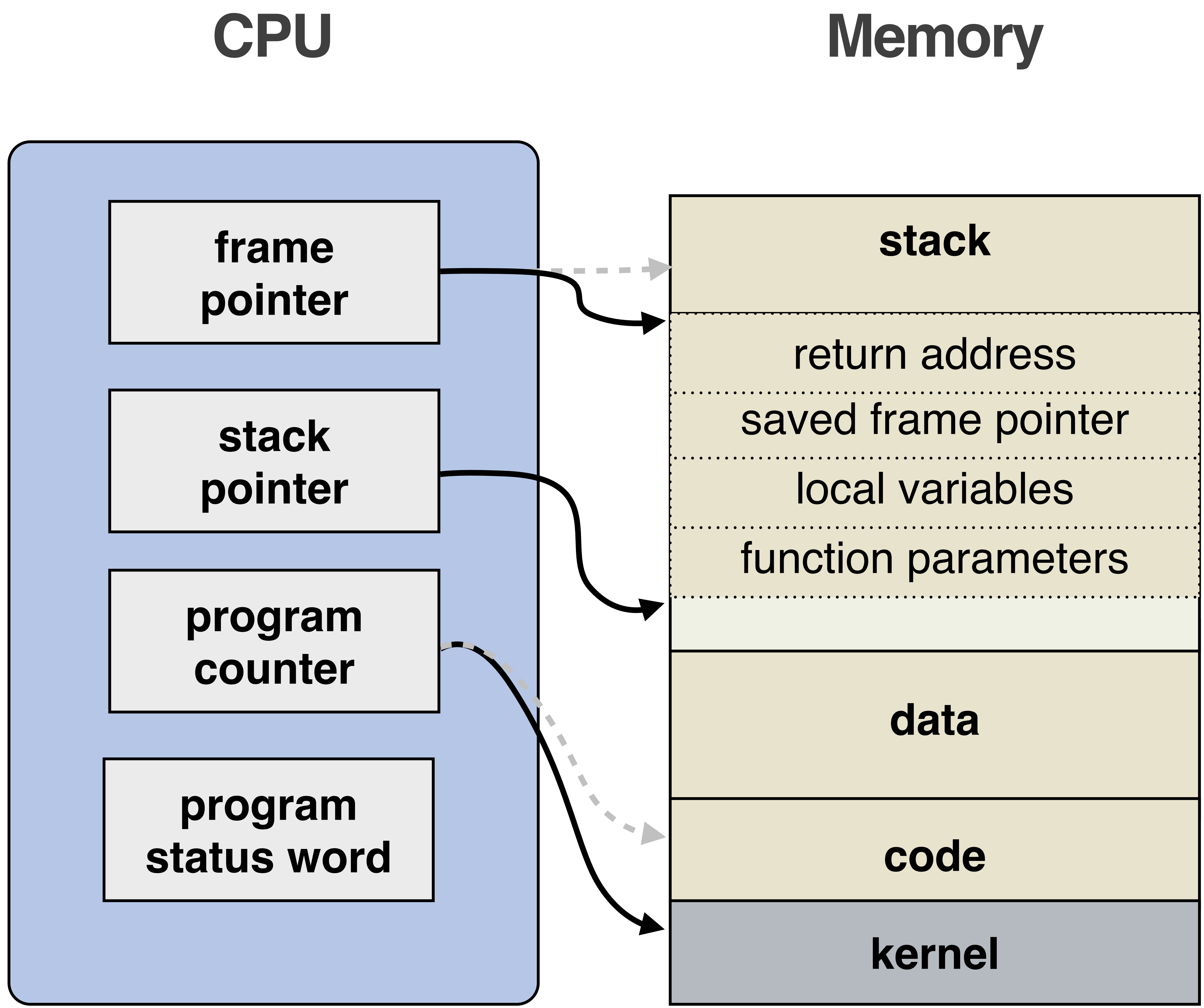
Physical Memory



What really happened during a page fault?

(i)

save current execution context and jump to a page fault handler


CPU
Memory
frame pointer
stack pointer
program counter
program status word
stack
return address
saved frame pointer
local variables
function parameters
data
code
kernel

(ii)

find out which page is needed and whether the memory access is valid

(iii)

determine which frame to load the page into

(iv)

if the frame to be replaced is dirty, block faulting process.

(v)

write dirty frame to disk and load the faulted page from disk (using DMA)

(vi)

upon disk interrupt, update page table, rewind any partially executed instruction, and mark process as ready

(vii)

when process is scheduled to run, restore execution context

Global vs. Local Replacement

Dirty
vs.
Clean
Pages

Locked
vs.
Unlocked
Pages

Demand Paging

vs.

Prepaging

Free when needed

vs.

Free in background

Swap Partition

vs.

Page File

major

vs.

minor

page fault

Page Replacement Algorithm

which page to evict from the physical memory?

Physical Memory

	1	3	0	12	9	6	4	18
M bit →	1	0	1	0	1	0	0	0

Page Requests

1, 4, 6, 0, 2, 3, 4, 4, 12, 10, 2, ...

Goals:

Maximize Hit Rate

Simple & Fast

Belady’s Algorithm:

evict the page which will be accessed furthest into the future

optimal

but

unrealizable

assuming that

pages that haven’t been referenced recently will not be reference in near future

Physical Memory

	1	3	0	12	9	6	4	18
M bit	1	0	1	0	1	0	0	0
R bit								

Page Requests

1, 4, 6, 0, 2, 3, 4, 4, 12, 10, 2, ...

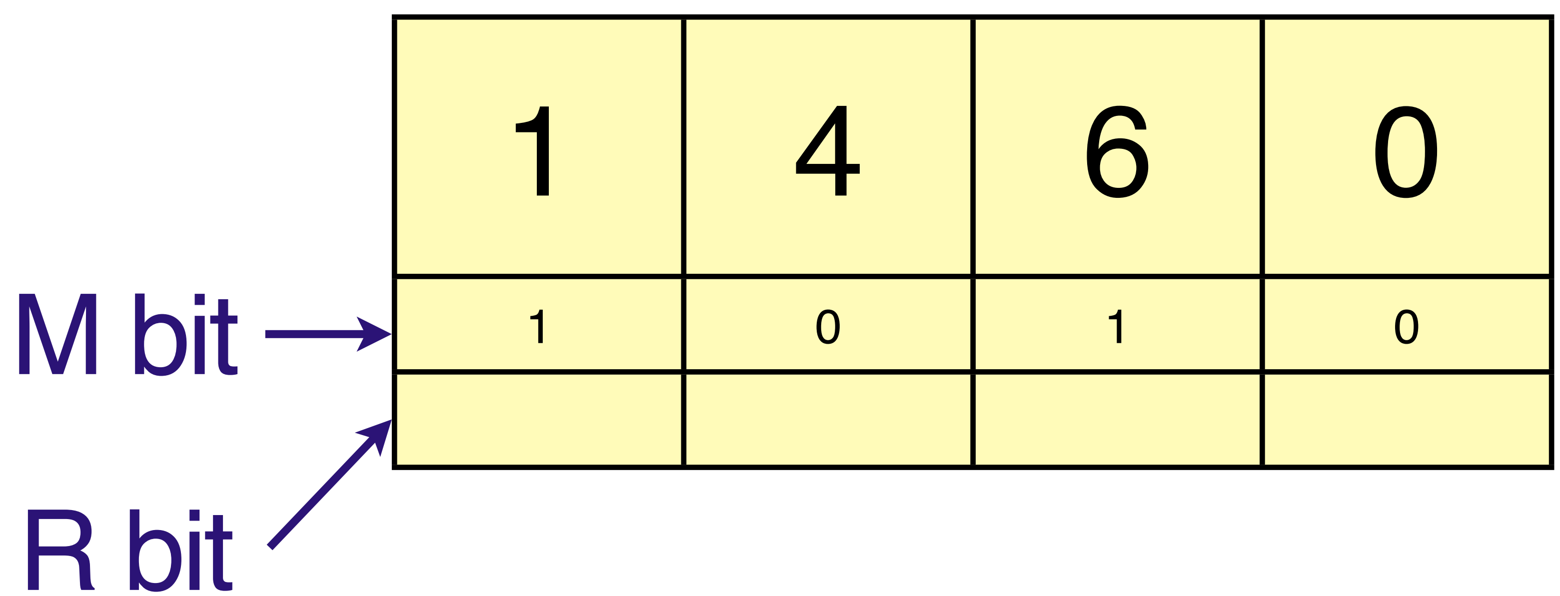
Not Recently Used (NRU)

evicts pages in the order

1. **R = 0, M = 0**
2. **R = 0, M = 1**
3. **R = 1, M = 0**
4. **R = 1, M = 1**

First-in-first-out (FIFO) evicts the page that stayed in RAM the longest

Physical Memory



Page Requests

1, 4, 6, 0, 2, 3, 4, 4, 12, 10, 2, ...

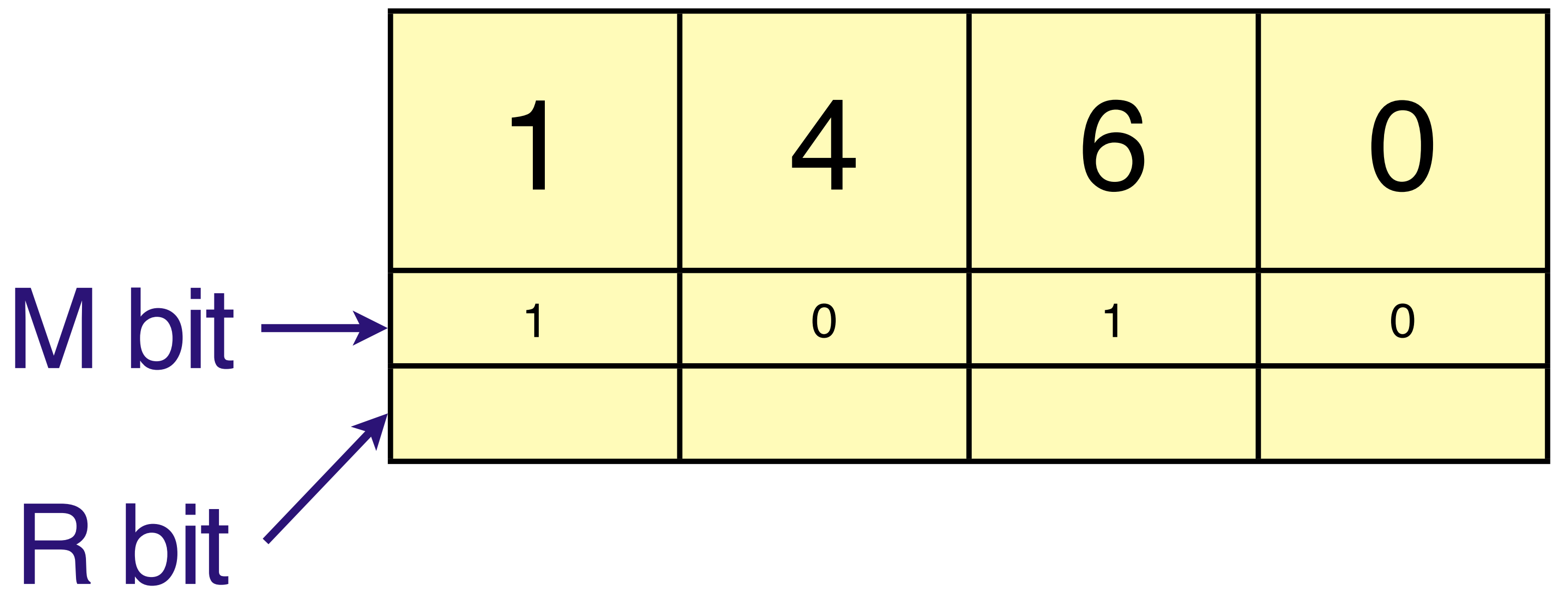
Second Chance

evicts the page with R=0

that stayed in RAM

the longest

Physical Memory



Page Requests

1, 4, 6, 0, 1, 2, 3, 4, 4, 12, 10, ...

```
p = pages[i]
while p.R is 1
    p.R = 0
    i = (i + 1) % N
    p = pages[i]
evict p
```

This is called the

Clock

Algorithm

Least Recently Used (LRU) evicts the page that hasn’t been used the longest time.

LRU implementation with linked list

on referencing frame k:

move k to back of list

on evict:

evict head of list

LRU implementation with counter

on referencing frame k:

frame.t = current time

on evict:

evict frame with min t

LRU implementation with matrix

on referencing frame k:

set all bits of row k to 1

set all bits of column k to 0

on evict:

evict frame with min row

LRU approximation with NFU

periodically

page.count += page.R

on evict:

evict frame with min count

LRU approximation with aging

periodically

```
page.count >>= 1
page.count |= (page.R << N)
```

on evict:

evict frame with min count

locality of reference: process refers to a small number of pages at a time.

working set of a process: set of pages accessed in last T seconds

```
while noone is evicted
    i = (i + 1) % N
    p = pages[i]
    if p.R is 1
        p.R = 0
    else
        if p last accessed > T ago
            if p is dirty
                schedule p’s write to disk
            else
                evict p
```

what if we have gone one round with evicting?

all pages are in working set.
evict any clean page.

what if all pages are dirty?

evict current page
(or any page)

This is called the

WSClock

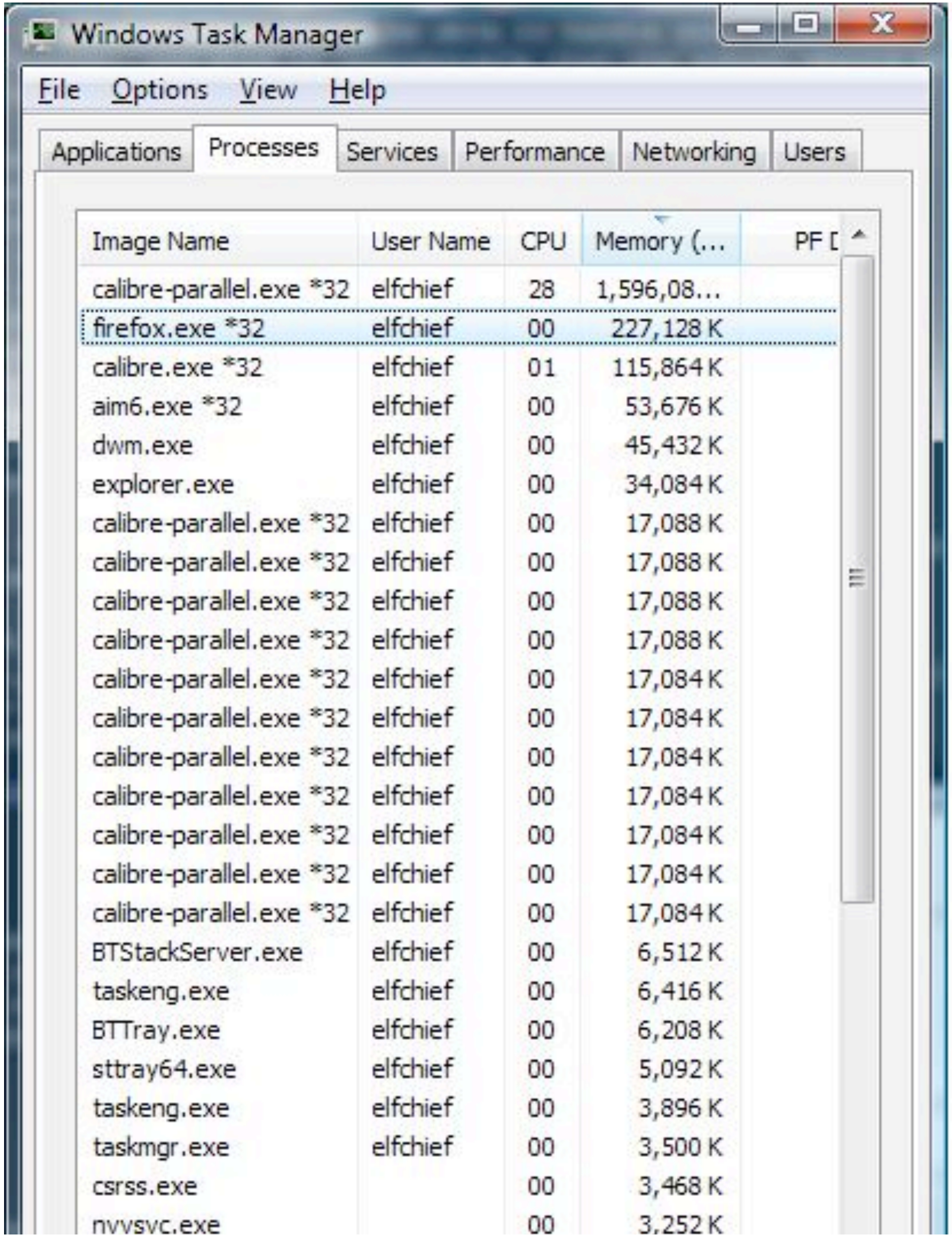
Algorithm

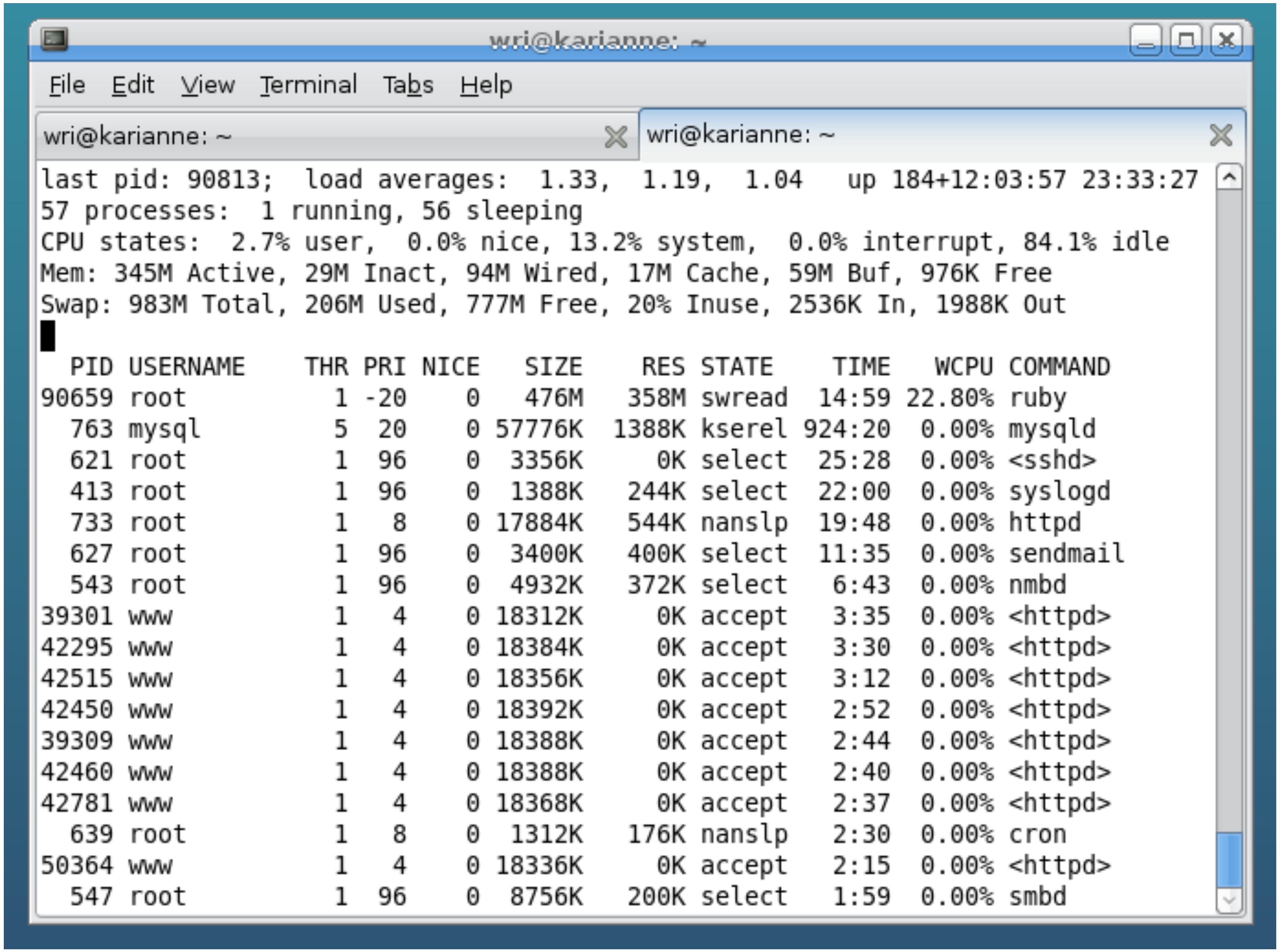
NRU
FIFO
SC
CLOCK
LRU
WSCLOCK

What happen when working set of all processes exceed the size of physical memory ?

Thrashing

CPU spend significant amount of time paging in/out, not doing real work


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Image Name User Name CPU Memory (... PF [
calibre-parallel.exe *32 elfchief 28 1,596,08...
firefox.exe *32 elfchief 00 227,128 K
calibre.exe *32 elfchief 01 115,864 K
aim6.exe *32 elfchief 00 53,676 K
dwm.exe elfchief 00 45,432 K
explorer.exe elfchief 00 34,084 K
calibre-parallel.exe *32 elfchief 00 17,088 K
calibre-parallel.exe *32 elfchief 00 17,088 K
calibre-parallel.exe *32 elfchief 00 17,088 K
calibre-parallel.exe *32 elfchief 00 17,088 K
calibre-parallel.exe *32 elfchief 00 17,084 K
calibre-parallel.exe *32 elfchief 00 17,084 K
calibre-parallel.exe *32 elfchief 00 17,084 K
calibre-parallel.exe *32 elfchief 00 17,084 K
calibre-parallel.exe *32 elfchief 00 17,084 K
calibre-parallel.exe *32 elfchief 00 17,084 K
calibre-parallel.exe *32 elfchief 00 17,084 K
BTStackServer.exe elfchief 00 6,512 K
taskeng.exe elfchief 00 6,416 K
BTTray.exe elfchief 00 6,208 K
sttray64.exe elfchief 00 5,092 K
taskeng.exe elfchief 00 3,896 K
taskmgr.exe elfchief 00 3,500 K
csrss.exe 00 3,468 K
nvvsvc.exe 00 3,252 K


wri@karianne: ~
File Edit View Terminal Tabs Help
wri@karianne: ~
wri@karianne: ~
last pid: 90813; load averages: 1.33, 1.19, 1.04   up 184+12:03:57 23:33:27
57 processes: 1 running, 56 sleeping
CPU states: 2.7% user, 0.0% nice, 13.2% system, 0.0% interrupt, 84.1% idle
Mem: 345M Active, 29M Inact, 94M Wired, 17M Cache, 59M Buf, 976K Free
Swap: 983M Total, 206M Used, 777M Free, 20% Inuse, 2536K In, 1988K Out
PID USERNAME THR PRI NICE SIZE RES STATE TIME WCPU COMMAND
90659 root 1 -20 0 476M 358M swread 14:59 22.80% ruby
763 mysql 5 20 0 57776K 1388K kserel 924:20 0.00% mysqld
621 root 1 96 0 3356K 0K select 25:28 0.00% <sshd>
413 root 1 96 0 1388K 244K select 22:00 0.00% syslogd
733 root 1 8 0 17884K 544K nanslp 19:48 0.00% httpd
627 root 1 96 0 3400K 400K select 11:35 0.00% sendmail
543 root 1 96 0 4932K 372K select 6:43 0.00% nmbd
39301 www 1 4 0 18312K 0K accept 3:35 0.00% <httpd>
42295 www 1 4 0 18384K 0K accept 3:30 0.00% <httpd>
42515 www 1 4 0 18356K 0K accept 3:12 0.00% <httpd>
42450 www 1 4 0 18392K 0K accept 2:52 0.00% <httpd>
39309 www 1 4 0 18388K 0K accept 2:44 0.00% <httpd>
42460 www 1 4 0 18388K 0K accept 2:40 0.00% <httpd>
42781 www 1 4 0 18368K 0K accept 2:37 0.00% <httpd>
639 root 1 8 0 1312K 176K nanslp 2:30 0.00% cron
50364 www 1 4 0 18336K 0K accept 2:15 0.00% <httpd>
547 root 1 96 0 8756K 200K select 1:59 0.00% smbd