Exam 2 preview

Glenn Bruns
CSUMB

Memory management summary

Main idea was virtual address space, and address translation (virtual → physical).

Main schemes:

- base-and-bounds addressing
- segmentation
- simple paging
- multi-level paging

We also saw:

- TLB (cache to deal with paging speed)
- Free-space mgmt (design of memory allocators)
- Swapping (use disk when not enough memory)

Key learning outcomes

For each memory management schemes, you need to be able to:

- explain the structure of a virtual address
- translate a virtual address to a physical address
- describe the size of each part of a virtual address given info about things like the size of a page
- explain how the scheme addresses performance and protection

Also, you need to be able to:

- do memory allocation with a free list, under various policies (which chunk to allocate from, where to put chunks in the list when freed, etc.)
- answer questions about how the TLB works; how to calculate average access time
- answer questions about how swapping works

What will be covered?

memory mgmt.

address spaces	(OSTEP 13)
C memory API	(14)
address translation	(15)
segmentation	(16)
free-space mgmt	(17)
paging	(18)
TLBs	(19)
advanced paging	(20)
swapping	(21,22)

bash

uniq, sort, diff, …
grep
regular expressions
pipes & redirection
sed
awk
make
bash customization

~ 20 questions on memory management
~ 13 questions on bash and proc. management

You are responsible for assigned material in text even if not discussed in class.

Exam format

50 minute exam

No notes allowed

Some multiple-choice and fill-in-the-blank

Bash, AWK, sed questions

- most will ask you to provide 1 or 2 lines of code

I expect you to understand OSTEP simulators used in lab and homework

How to prepare

Answer lab questions.

Answer homework problems.

Answer questions asked in lecture slides.

Peerwise.

Write down definitions or explanations for all concepts we've covered.

Write your own exam questions from lecture slides.

Do the EC homework on addresses if you haven't already

Flash cards

1. Look at card
2. Write down answer
3. Check answer

paging calculations

bash

awk

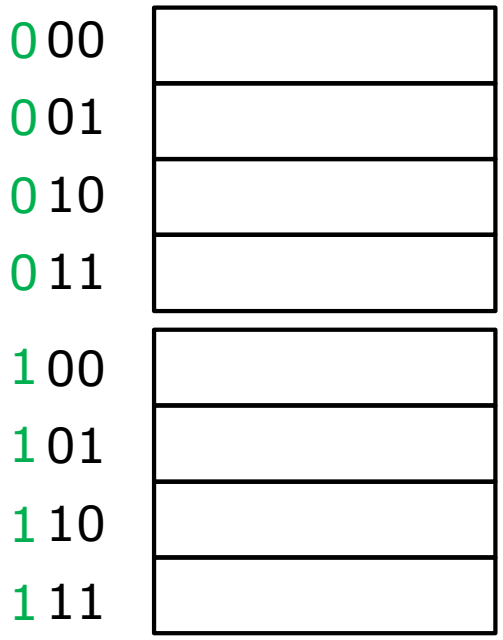
find

Write your own flash cards

Address spaces and bits

Two questions should be easy for you now:

1. If I have 2^n items, how many bits do I need to address them?
2. If I have n bits, how many items can I address?



Address spaces and bits

If there are 16 bits in a virtual address, and 64 pages, then how many bits in the offset part?

answer: 10

If there are 64 pages, then how many bits do we need in the VPN part of a virtual address? 64 = 2^6, so 6.

16 bits in address – 6 bits in VPN part = 10 bits in offset part

Address translation with base/bounds

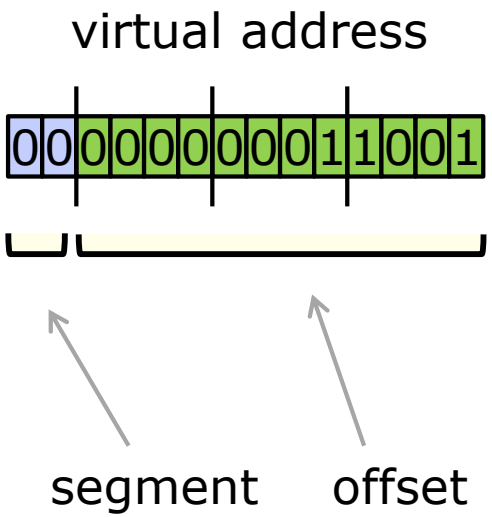
Suppose base is 16KB and bounds is 4KB

Virtual address	Physical address
0	16 KB
1 KB	?
3 KB	?
5 KB	?

In this scheme, a virtual address is unstructured: it has no "parts".

Address translation with segmentation

Segment	Base	Size	Grows Positive?
Code	32K	1K	1
Heap	34K	2K	1
Stack	28K	2K	0



Logical address: 25 (i.e. 0x0019)
- top two bits are 00, rest is 25
- 25 < 1K, so no error
- addr = 32K + 25 = 32768 + 25 = 32793

Logical address: 4202 (i.e. 0x106a)
- top two bits are 01, rest is 106
- 106 < 2K, so no error
- addr = 34K + 106 = 34816 + 106 = 34922

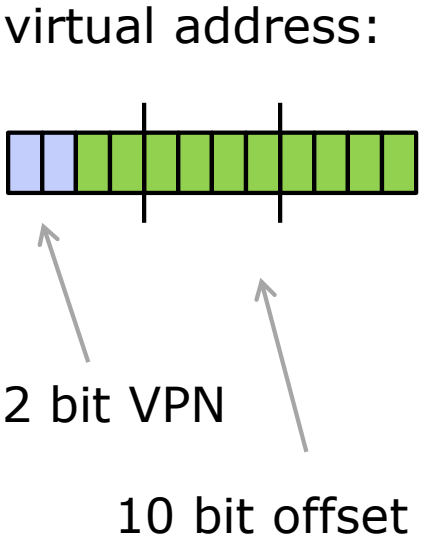
In this scheme, a virtual address has two parts: segment and offset

Note: the authors often write 32K to mean 32 * 1024

Address translation with paging

page table

	page frame number	valid	pro-tection	present	dirty	ref-erence
00	011	1	11	1	0	1
01	111	1	01	1	1	0
10	101	0	11	1	0	1
11	010	1	00	0	1	1



(how big is a page?)

0x031c → ?

0x09F2 → ?

Addr. trans. with multi-level paging

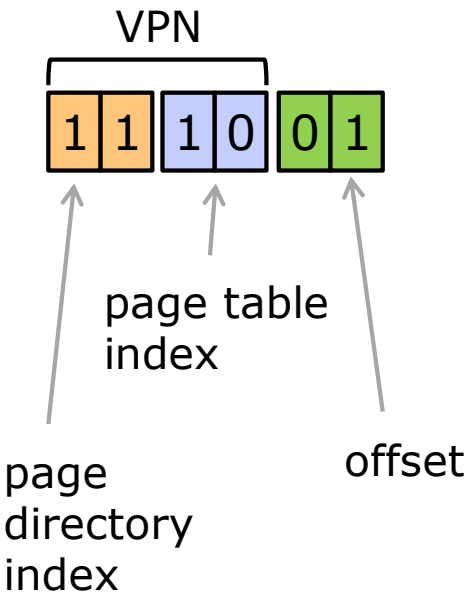
page directory

00	1	201
01	0	-
10	0	-
11	1	204

page table

00	12	PFN 201
01	13	
10	-	
11	100	
00	-	PFN 204
01	-	
10	88	
11	15	

virtual address:



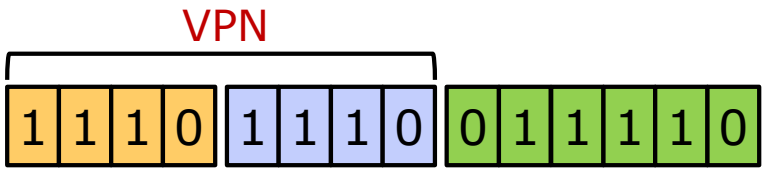
0x0039 → ?

0x000D → ?

0x001F → ?

Note: address of page directory stored in a register.

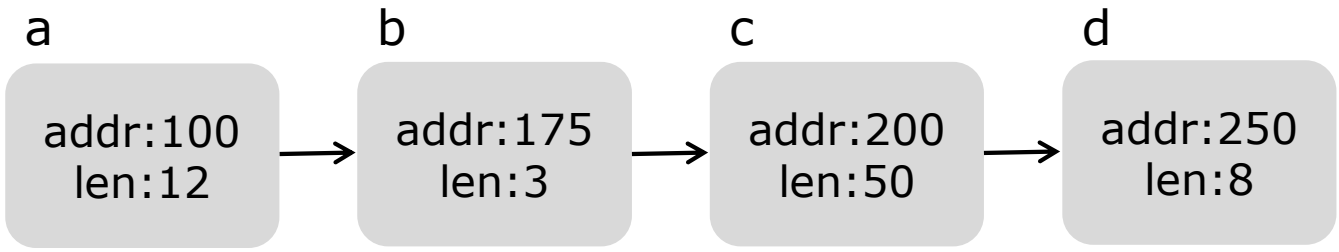
Addr. trans. with multi-level paging (II)



1111 1110 000000 ⟶ ?

Page Directory		Page of PT (@PFN:100)			Page of PT (@PFN:101)		
PFN	valid?	PFN	valid	prot	PFN	valid	prot
100	1	10	1	r-x	—	0	—
—	0	23	1	r-x	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	80	1	rw-	—	0	—
—	0	59	1	rw-	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	55	1	rw-
101	1	—	0	—	45	1	rw-

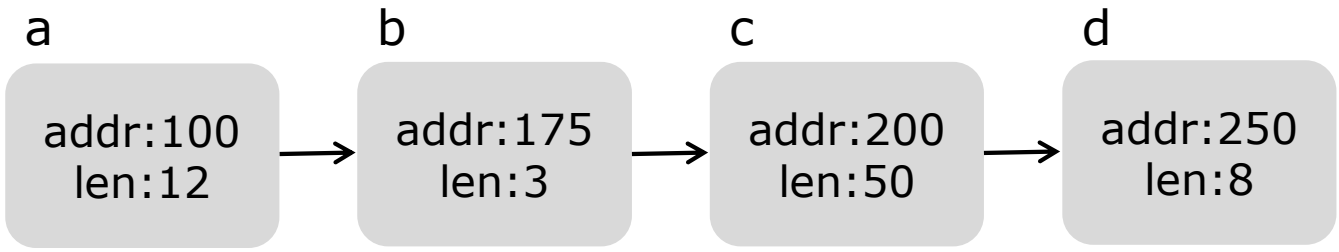
Free space management



Request is **malloc(6)**

- If best-fit policy, what does list look like after allocation?
- If worst-fit policy, what does list look like after allocation?
- If first-fit policy, what does list look like after allocation?

Free space management



Is coalescing of this free list possible?

Command Line

regular expressions

Find lines of foo.txt containing 'samu4883'

$ _____

answer:

```
grep samu4883 foo.txt
```

regular expressions

Find lines of foo.txt containing only 'samu4883'

$ _____

answer:

```
grep '^samu4883$' foo.txt
```

regular expressions

Find lines of foo.txt that start with a digit

$ _____

answer:

```
grep '^[0-9]' foo.txt

or

grep '^[[:digit:]]' foo.txt
```

regular expressions

Find lines of foo.txt that start with a non-digit

$ _____

answer:

```
grep '^[^0-9]' foo.txt
```

regular expressions

Suppose foo.txt contains only these lines:

```
123b
12b
012b34
```

What is the result of command

```
grep '[0-9][a-z]*b' foo.txt?
```

answer:

```
123b
12bb
012b34
```

regular expressions

Find lines of foo.txt containing a lower-case letter, followed by one or more numbers, and then another lower-case letter.

$ _____

answer:

```
egrep '[a-z][0-9]+[a-z]' foo.txt
```

or

```
grep '[a-z][0-9][0-9]*[a-z]' foo.txt
```

AWK

Print the second and fourth fields of foo.txt

$ _____

answer:

```
awk '{print $2,$4}' foo.txt
```

awk

Print every line of contacts.txt that contains ".edu"

$ _____

answer:

```
 awk '/\.edu/ { print $0 }' contacts.txt
or
 awk '/\.edu/ { print }' contacts.txt
or
 awk '/\.edu/' contacts.txt
```

awk

What does this awk script do?

```
{ total += $2 }

END { print total }
```

answer:

It prints the sum of the values in field $2.

awk

Command 'ps –ef' gives output like this:

```
brun1992 12234 12228  0 21:25 ?        00:00:00 sshd: brun1992@pts/2
brun1992 12235 12234  0 21:25 pts/2    00:00:00 -bash
brun1992 12263 12235  0 21:25 pts/2    00:00:00 ps -ef
root     25908  1816  0 17:46 ?        00:00:00 sshd: reye3973 [priv]
reye3973 25926 25908  0 17:46 ?        00:00:00 sshd: reye3973@pts/0
```

Write a bash script that would take this as input and produce:

```
brun1992 3
root 1
reye3973 1
```

bonus question:
How to use ps and this script from command line?

answer:

```
{
  cnt[$1]++
}
END {
  for (id in cnt) {
    print id, cnt[id]
  }
}
```

sed

Replace all occurrences of "22" in foo.txt with "33".

$ _____

answer:

```
 sed 's/22/33/g' foo.txt
or
 sed –i 's/22/33/g' foo.txt
```

Make

Suppose I enter 'make' at the command line.

Then I modify spec.txt, and enter 'make' again. What happens?.

```
test-out.txt: foo
	tester foo > test-out.txt

foo: foo.c baz.c
	gcc –o foo foo.c baz.c
baz.c: code-gen spec.txt
	./code-gen spec.txt > baz.c
code-gen: code-gen.c
	gcc –o code-gen code-gen.c
```

```
answer:
target is test-out.txt

dependencies:

test-out.txt → foo
foo → foo.c, baz.c
baz.c → code-gen, spec.txt
code-gen → code-gen.c
```

Bonus bash question

How can I find all Makefiles beneath the current working directory?

$ _____

answer:

```
find . -name Makefile
```

Address spaces

- virtual address space
- physical address space
- virtual address, physical address
- virtual address spaces and processes
- why do we need a virtual address spaces?
- what are issues in providing a virtual address space?

Base and bounds

- simple virtual memory space
- base and bounds addressing - mechanics
- base and bounds registers
- dealing with multiprogramming
- addressing efficiency, protection

Segmentation

- segmentation
- what are typical segments?
- what problems does it solve?
- mechanics of address translation
- how to achieve performance?
- how to achieve protection?
- special advantages of segmentation in protection?
- code sharing
- fine-grained segmentation
- problems with segmentation

Paging

- virtual page
- physical page
- virtual page number
- offset
- physical frame number
- page table
- mechanics of address translation
- calculating size of virtual address, pages, number of pages, etc.
- pros/cons of paging

Multi-level paging

- space problems with simple paging
- page directory
- layout of virtual memory addresses
- valid bit
- mechanics of address translation

In both simple and multi-level paging, the main thing is to map a VPN to a PFN.

The two approaches simply use different data structures.

Translation lookaside buffers

- translation lookaside buffer
- cache
- hit rate
- calculating average access time from hit rate
- locality principle
- temporal locality; spatial locality
- TLB and multiprogramming
- flushing the cache
- MMU

Free-space management

- memory allocation
- malloc(), free()
- free list
- best fit, worst fit, first fit
- memory fragmentation
- internal vs external fragmentation
- design goals
- mechanics of free list
- coalescing

When memory is allocated:
- best fit?
- worst fit?
- first fit?

When memory is freed:
- front of list?
- in size order?
- in address order?

Also, when to coalesce?

Swapping

- swapping
- swapping vs. paging
- paging with limited physical memory
- mechanics of swapping
- page fault
- page fault handler
- present bit
- swapping and caching
- page replacement policy
- policies: FIFO, LRU, random, optimal, LFU
- workload and simulation

C memory API

- memory allocation
- heap
- malloc()
- free()
- typical errors
- memory leak