

CSO2 (CS3130)

themes

automating building software

- libraries, taking advantage of incremental compilation

sharing machines

- multiple users/programs on one system

parallelism and concurrency

- doing two+ things at once

under the hood of sockets

- layered design of networks

- implementing secure communication

under the hood of fast processors

- caching, (hidden) parallelism, avoiding idle time

themes

automating building software

libraries, taking advantage of incremental compilation

sharing machines

multiple users/programs on one system

parallelism and concurrency

doing two+ things at once

under the hood of sockets

layered design of networks

implementing secure communication

under the hood of fast processors

caching, (hidden) parallelism, avoiding idle time

make

```
$ ./foo.exe
```

```
...
```

```
...
```

```
$ edit readline.c
```

```
$ make
```

```
clang -g -O -Wall -c readline.c -o readline.o
```

```
ar rcs terminal.o readline.o libreadline.a
```

```
clang -o foo.exe foo.o foo-utility.o -L. -lreadline
```

```
$
```

themes

automating building software

libraries, taking advantage of incremental compilation

sharing machines

multiple users/programs on one system

parallelism and concurrency

doing two+ things at once

under the hood of sockets

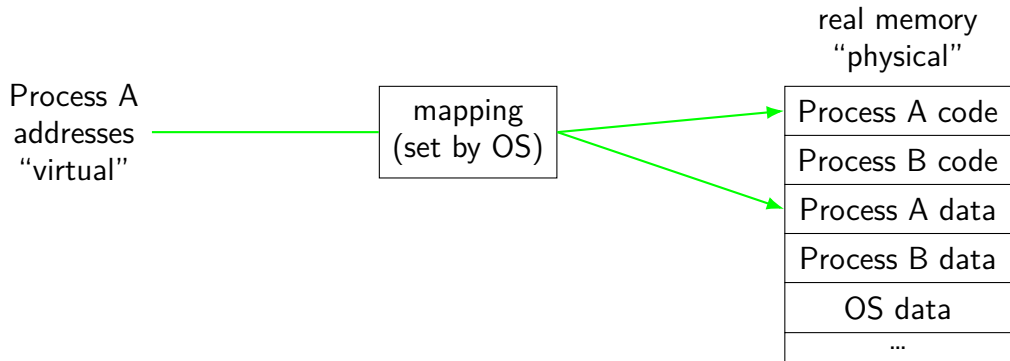
layered design of networks

implementing secure communication

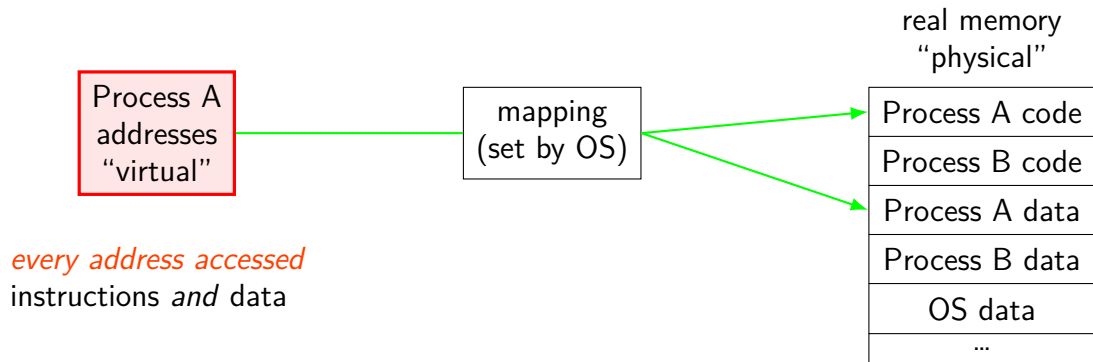
under the hood of fast processors

caching, (hidden) parallelism, avoiding idle time

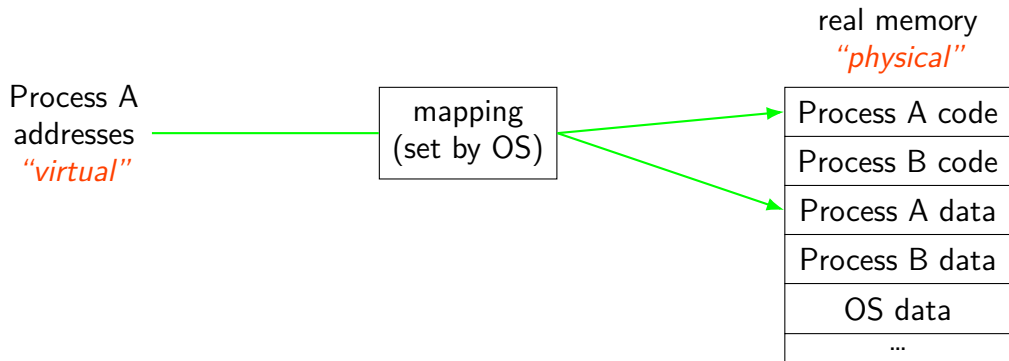
address translation



address translation

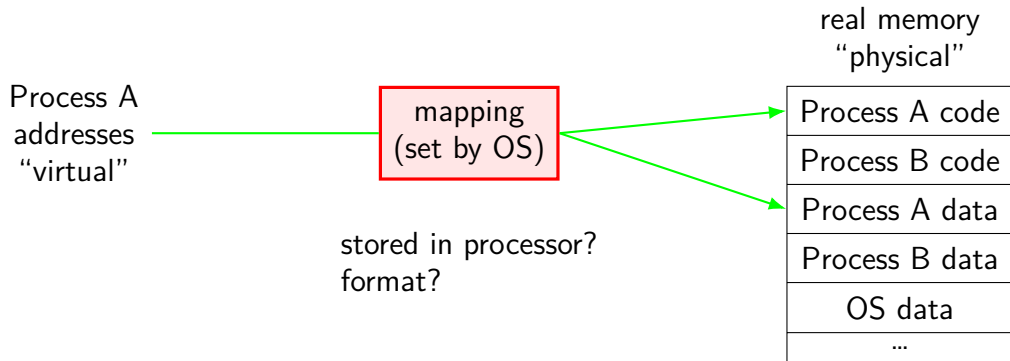


address translation



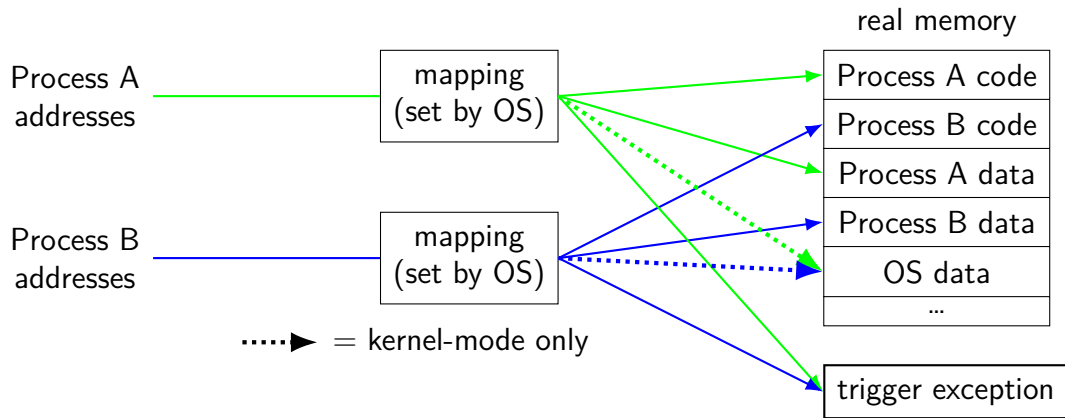
program addresses are 'virtual'
real addresses are 'physical'
can be *different sizes*!

address translation



address spaces

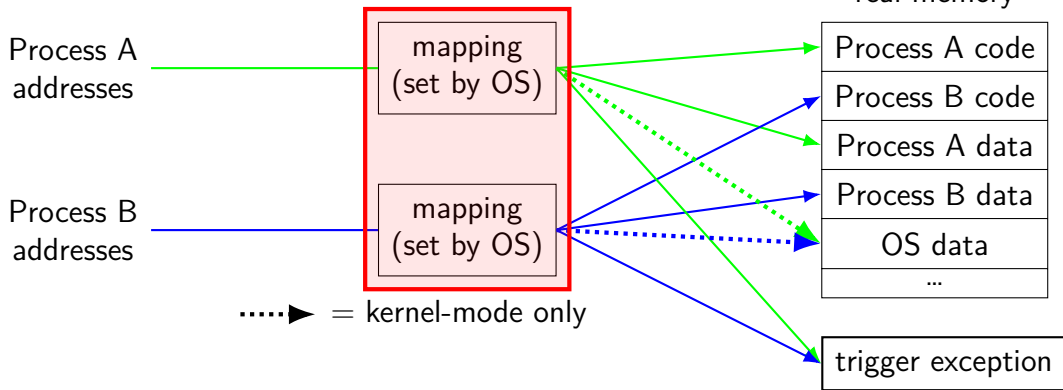
illusion of *dedicated memory*



address spaces

illusion of *dedicated memory*

chose one during context switch



themes

- automating building software

 - libraries, taking advantage of incremental compilation

- sharing machines*

 - multiple users/programs on one system

- parallelism and concurrency*

 - doing two+ things at once

- under the hood of sockets

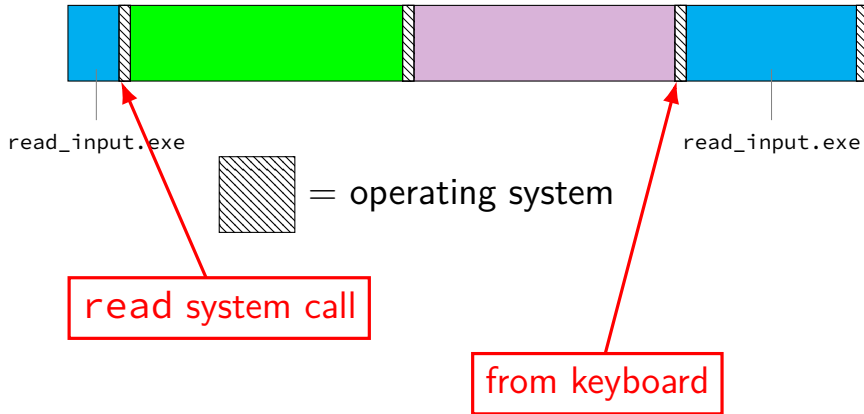
 - layered design of networks

 - implementing secure communication

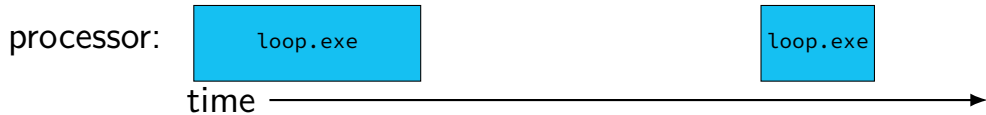
- under the hood of fast processors

 - caching, (hidden) parallelism, avoiding idle time

keyboard input timeline



time multiplexing



time multiplexing

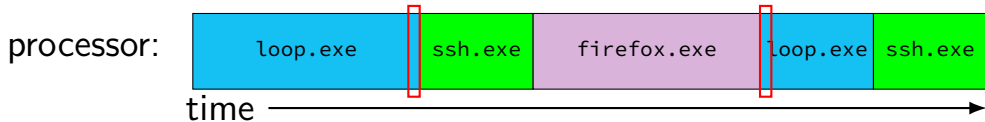


```
...  
loop: ...  
    ...  
    jmp loop  
loop: ...  
    ...
```

million cycle delay

```
    ...  
    jmp loop  
loop: ...
```

time multiplexing



```
...  
loop: ...  
    ...  
    jmp loop  
loop: ...  
    ...
```

million cycle delay

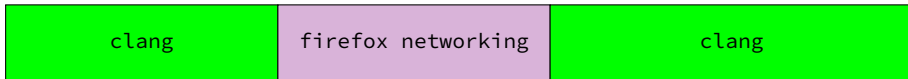
```
    ...  
    jmp loop  
loop: ...
```

multiple cores+threads

core 1:

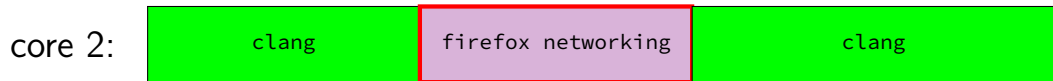


core 2:



multiple cores? each core still divided up

multiple cores+threads



one program with multiple *threads*

themes

automating building software

libraries, taking advantage of incremental compilation

sharing machines

multiple users/programs on one system

parallelism and concurrency

doing two+ things at once

under the hood of sockets

layered design of networks

implementing secure communication

under the hood of fast processors

caching, (hidden) parallelism, avoiding idle time

permissions

```
$ ls /u/other/secret  
ls: cannot open directory '/u/other/secret': Permission denied  
$ shutdown  
shutdown: Permission denied
```

themes

automating building software

libraries, taking advantage of incremental compilation

sharing machines

multiple users/programs on one system

parallelism and concurrency

doing two+ things at once

under the hood of sockets

layered design of networks

implementing secure communication

under the hood of fast processors

caching, (hidden) parallelism, avoiding idle time

layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

layers terminology

application	application-defined meanings	
transport	reach correct program, reliability/streams	segments/datagrams
network	reach correct machine (across networks)	packets
link	coordinate shared wire/radio	frames
physical	encode bits for wire/radio	

names and addresses

name	address
<i>logical identifier</i>	<i>location/how to locate</i>
variable counter	memory address 0x7FFF9430
DNS name www.virginia.edu	IPv4 address 128.143.22.36
DNS name mail.google.com	IPv4 address 216.58.217.69
DNS name mail.google.com	IPv6 address 2607:f8b0:4004:80b::2005
DNS name reiss-t3620.cs.virginia.edu	IPv4 address 128.143.67.91
DNS name reiss-t3620.cs.virginia.edu	MAC address 18:66:da:2e:7f:da
service name https	port number 443
service name ssh	port number 22

secure communication?

how do you know who your socket is to?

who can read what's on the socket?

what can you do to restrict this?

themes

automating building software

libraries, taking advantage of incremental compilation

sharing machines

multiple users/programs on one system

parallelism and concurrency

doing two+ things at once

under the hood of sockets

layered design of networks

implementing secure communication

under the hood of fast processors

caching, (hidden) parallelism, avoiding idle time

2004 CPU

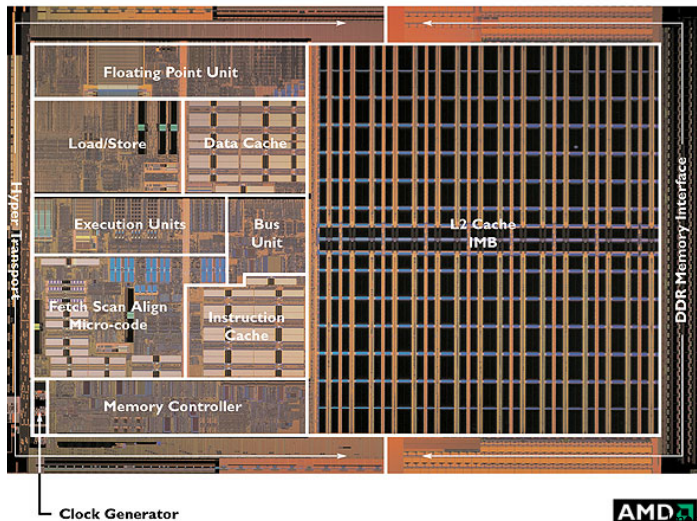


Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

2004 CPU

▲ Registers

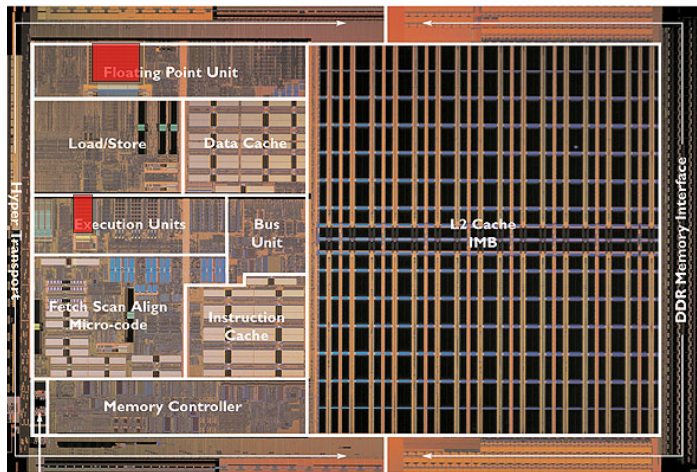


Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

2004 CPU

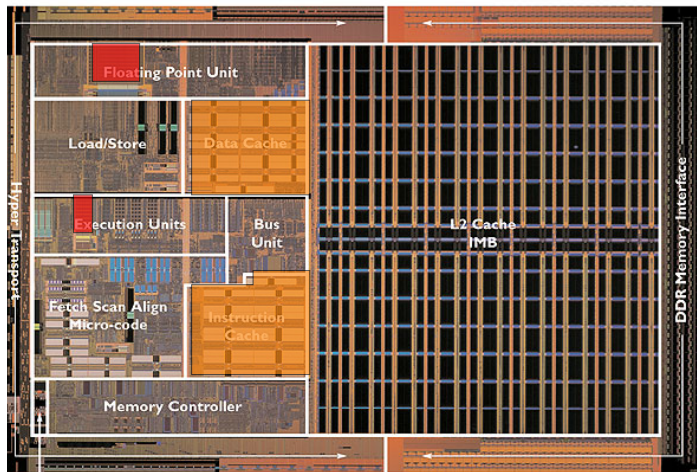
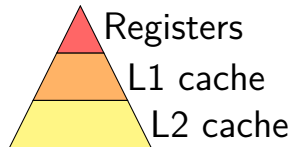
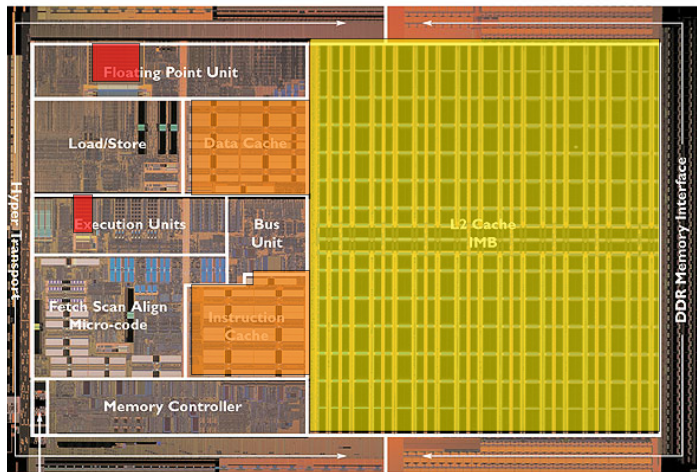


Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

2004 CPU



Clock Generator



Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

2004 CPU

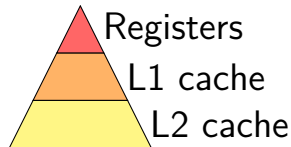
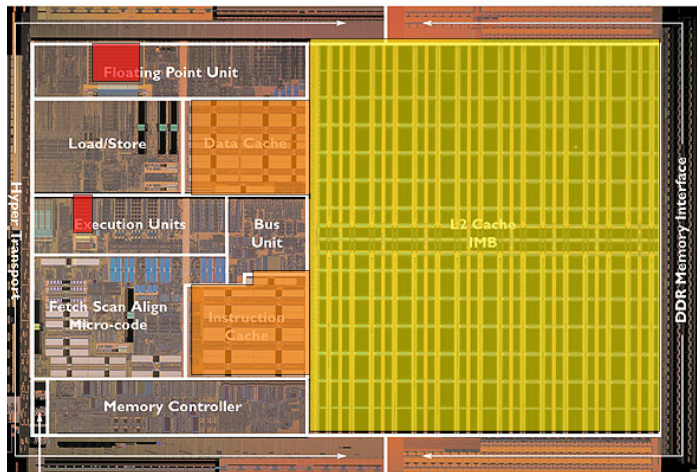


Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

2004 CPU

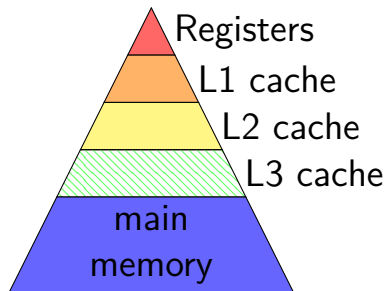
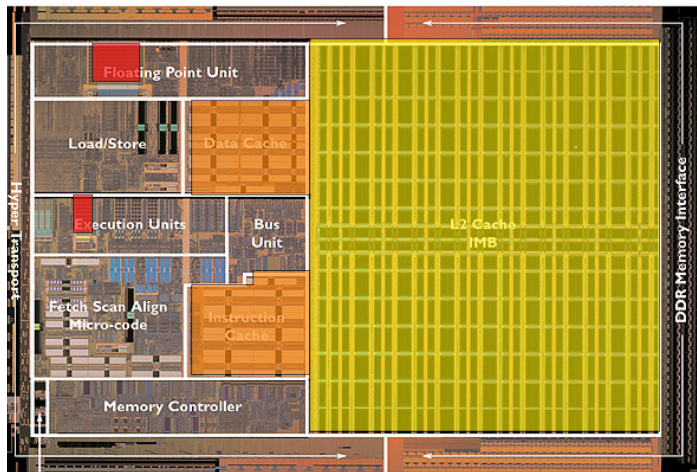


Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

2004 CPU

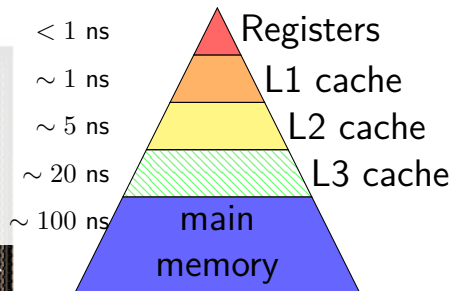
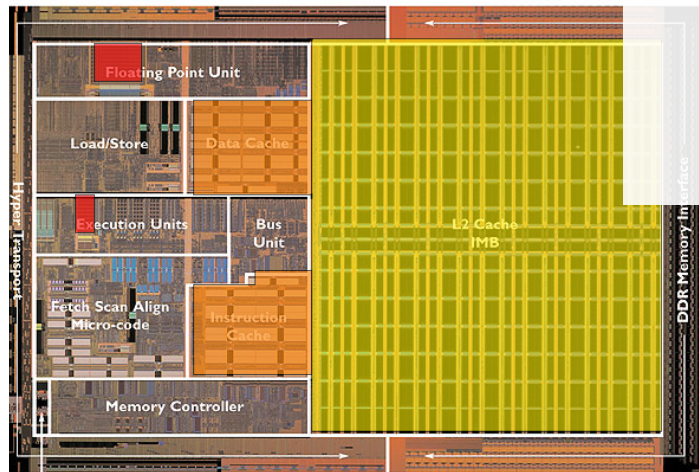


Image: approx 2004 AMD press image of Opteron die;
approx register location via chip-architect.org (Hans de Vries)

some performance examples

example1:

```
    movq $1000000000000, %rax
loop1:
    addq %rbx, %rcx
    decq %rax
    jge loop1
    ret
```

about 30B instructions

my desktop: approx 2.65 sec

example2:

```
    movq $1000000000000, %rax
loop2:
    addq %rbx, %rcx
    addq %r8, %r9
    decq %rax
    jge loop2
    ret
```

about 40B instructions

my desktop: approx 2.65 sec

some performance examples

example1:

```
    movq $1000000000000, %rax  
loop1:  
    addq %rbx, %rcx  
    decq %rax  
    jge loop1  
    ret
```

about 30B instructions

my desktop: approx *2.65 sec*

example2:

```
    movq $1000000000000, %rax  
loop2:  
    addq %rbx, %rcx  
    addq %r8, %r9  
    decq %rax  
    jge loop2  
    ret
```

about 40B instructions

my desktop: approx *2.65 sec*

C exercise

```
int array[4] = {10,20,30,40};  
int *p;  
p = &array[0];  
p += 2;  
p[1] += 1;
```

array =

- | | |
|-----------------------------|------------------|
| A. compile or runtime error | B. {10,20,30,41} |
| C. {10,20,32,41} | D. {10,21,30,40} |
| E. {12,21,30,40} | F. none of these |

C exercise (2)

```
int *array2[4]; int array1[4] = {10,20,30,40};
void mystery(int **p) {
    *p = &array1[2];
}
int main() {
    int **q;
    q = array2;
    mystery(q);
    array1[1] = *q;
    ...
}
```

array1 =

- | | |
|-----------------------------|------------------|
| A. compile or runtime error | B. {10,10,30,40} |
| C. {10,30,30,40} | D. {10,10,20,30} |
| E. {10,20,10,20} | F. none of these |

C exercise (2)

```
int *array2[4]; int array1[4] = {10,20,30,40};
void mystery(int **p) {
    *p = &array1[2];
}
int main() {
    int **q;
    q = array2;
    mystery(q);
    array1[1] = *q;
    ...
}
```

array1 =

- | | |
|-----------------------------|------------------|
| A. compile or runtime error | B. {10,10,30,40} |
| C. {10,30,30,40} | D. {10,10,20,30} |
| E. {10,20,10,20} | F. none of these |

some avenues for review

review CSO1 stuff

labs 9–12 (of last Fall)

<https://researcher111.github.io/uva-cso1-F23-DG/>

exercises we've used in the past:

implement strsep library function

implement conversion from dynamic array to linked list

some pointer stuff

0x040

0x038

0x030

0x028

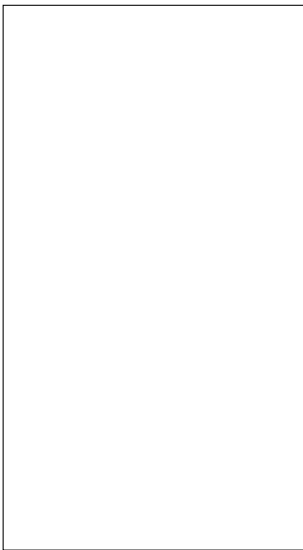
0x020

0x018

0x010

0x008

0x000



```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr = ???
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr = ???
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

~~*ptr = 0xAB;~~ runtime error

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr: 0x28
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

```
ptr = &single;
```

```
ptr = (int*) 0x28;    addr. of single
```

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr: 0x28
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

```
ptr = &single;  
ptr = (int*) 0x28;  addr. of single
```

~~ptr = 0x28; compile error~~

~~ptr = (int*) single;~~

pointer to unknown place

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
0x028	single: 0xFF
	ptr: 0x28
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;  
ptr = &single;
```

**ptr = 0xFF;*

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr: 0x2C
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

```
ptr = array;  
ptr = &array[0];  
ptr = (int*) 0x2C;
```

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr: 0x2C
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

```
ptr = array;  
ptr = &array[0];  
ptr = (int*) 0x2C;
```

~~ptr = array[0]; compile error~~

~~ptr = (int*) array[0];~~

pointer to unknown place

some pointer stuff

0x040	
0x038	array[2]: 0xFF
	array[1]: 0x45
0x030	array[0]: 0x12
	single: 0x78
0x028	ptr: 0x2C
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;  
ptr = &array[0];
```

```
ptr[2] = 0xFF;  
*(ptr + 2) = 0xFF;
```

```
int *temp1; temp1 = ptr + 2;  
*temp1 = 0xFF;
```

```
int *temp2; temp2 = &ptr[2];  
*temp2 = 0xFF;
```

some pointer stuff

0x040	
0x038	array[2]: 0x67
	array[1]: 0x45
0x030	array[0]: 0x12
0x028	single: ...
	ptr: 0x2C
0x020	
0x018	
0x010	
0x008	
0x000	

```
int array[3]={0x12,0x45,0x67};  
int single = 0x78;  
int *ptr;
```

```
void change_arg(int *x) {  
    *x = compute_some_value();  
}  
...  
change_arg(&single);
```

backup slides