The Command Line

Assembly

Sizes of C Data Types in x86-64

C decleration	Size (bytes)	Intel data type	Assembly suffix
char	1	Byte	b
char	2	Word	w
int	4	Double word	l
long	8	Quad word	q
char *	8	Quad word	q
float	4	Single precision	s
double	8	Double precision	l

Instruction Data Types

MOV source, dest	Move data source -> dest
movb	Move 1 byte
movw	Move 2 bytes
movl	Move 4 bytes
movq	Move 8 bytes

x86-64 Integer Registers

%rax	%eax %ax %al	(function result)	%r8	%r8d	(fifth argument)
%rbx	$ebx %bx %bl		%r9	%r9d	(sixth argument)
%rcx	$ecx %xc %cl	(fourth argument)	%r10	%r10d	
%rdx	%edx %dx %dl	(third argument)	%r11	%r11d	
%rsi	%esi %si %sil	(second argument)	%r12	%r12d	
%rdi	%edi %di %dil	(first argument)	%r13	%r13d	
%rsp	%esp %sp % bsl	(stack pointer)	%r14	%r14d	
%rbp	%ebp %bp %bpl		%r15	%r15d	

Arguments

- 1. %rdi
- 2. %rsi
- 3. %rdx
- 4. %rcx
- 5. %rax
- 6. %rbx

return address is between the stack frames below rbp
our addresses live in 400000 space
0x83 0x05 0x40
is in big endian? and we have to translate it to
0x40 0x05 0x83
() = reference
cmp x y = subtract y - x
look at p and q
reading memory is two actions
leaq (%rsi, %rsi, 4) = rsi + 4rsi

The Memory Hierarchy

```
Latency (approximate)                                           Storage Type
~1 ns                   L1 Cache                               <-\
~10 ns                  L2 Cache                               <--- SRAM
~40 ns                  L3 Cache (shared)                      <-/
100-200 ns              Main memory                            DRAM
```

```
10_000 ns                Local Secondary Storage              HDD/SSD
                         Remote Secondary Storage             LAN
```

valid bit tells us we have something important there

tag is the id of the block

Look at disk access time math on the presentation, wee 11 lecture 2

nested arrarys

Typical question is about finding C for example Array[i][j] = $A+(i*C+j)*K$

discaccess

architecture questions

cache question

stack question

start translating addresses from the right to left

0x135

0001-0011-0101

set index - tag - block offset

0001 0011 01 01 TTTT TTTT SS BB

tags are shared if they belong to the same memory approximate

tag is not the unique for all cache only the set

the bigger the address the bigger chance it's a miss

block alrady in cache: write-through write-back

block not already in cache: write-allocate no-write-allocate

direct map = only one block per set

n-way = many blocks in one set

tag is an id for block in a set, tag is unique per set

shifting and arithmetic shifting