

Memory: C and x86 assembly

Loop Refresher

Optimized or unoptimized?

	sum:		mem ops
	.LFB2:		
	.loc 1 2 0		
	.LVL0:		
	.loc 1 4 0		
	movl \$0, %eax	eax == s	0
	.LVL1:		
int sum(int count)	testl %edi, %edi	???	0
{ int s = 0;	jle .L4	???	0
int i;	movl \$0, %eax	s = 0	0
for(i = 0; i < count; i++) {	movl \$0, %edx	i = 0	0
s += i;	.LVL2:		
}	.L5:		
return s;	.loc 1 5 0		
}	addl %edx, %eax	s += i	0
	.loc 1 4 0		
	addl \$1, %edx	i++	0
	cmpl %edi, %edx	i < count	0
	jne .L5	go again	0
	.L4:		
	.LVL3:		
	.loc 1 8 0		
	rep ; ret	Done	0

Loop Refresher

Optimized or
unoptimized?

Optimized

	sum:		mem ops
	.LFB2:		
	.loc 1 2 0		
	.LVL0:		
	.loc 1 4 0		
	movl \$0, %eax	eax == s	0
	.LVL1:		
int sum(int count)	testl %edi, %edi	???	0
{ int s = 0;	jle .L4	???	0
int i;	movl \$0, %eax	s = 0	0
for(i = 0; i < count; i++) {	movl \$0, %edx	i = 0	0
s += i;	.LVL2:		
}	.L5:		
return s;	.loc 1 5 0		
}	addl %edx, %eax	s += i	0
	.loc 1 4 0		
	addl \$1, %edx	i++	0
	cmpl %edi, %edx	i < count	0
	jne .L5	go again	0
	.L4:		
	.LVL3:		
	.loc 1 8 0		
	rep ; ret	Done	0

Array Access in a Loop

	sum:		
	.LFB2:		mem ops
int array[10];	.loc 1 5 0		
	.LVL0:		
int sum(int count)	.loc 1 8 0		
{	movl \$0, %eax	s = 0	0
int s = 0;	.LVL1:		
int i;	testl %edi, %edi	???	0
for(i = 0; i < count; i++) {	jle .L4	???	0
s += array[i];	movl \$0, %eax	s = 0	0
}	movl \$0, %ecx	i = 0; is in a 32 l	0
return s;	.LVL2:		
}	movl \$0, %edx	t1 = 0, this is an address (64 bits)	0
	.L5:		
	.loc 1 9 0		
	addl array(,%rdx,4), %eax	s += array[1]	1
	.loc 1 8 0		
	addl \$1, %ecx	i++	0
	addq \$1, %rdx	t1++	0
	cmpl %edi, %ecx	i < count	0
	jne .L5		0
	.L4:		
	.LVL3:		
	.loc 1 12 0		
	rep ; ret		0
	.LFE2:		
	.size sum, .-sum		
	.comm array,40,32	allocate 40 bytes for array aligned at 32 byte boundary	

access memory at
array + (long)i * 4

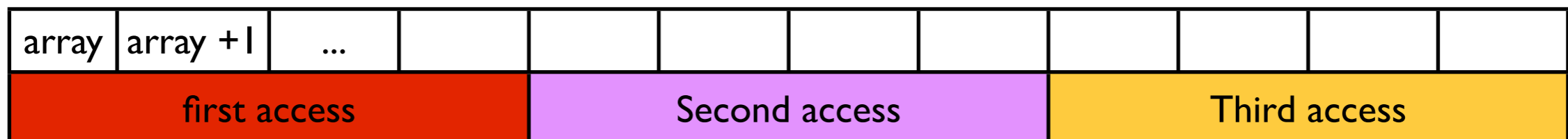
The array is
statically
declared here

Array Access in a Loop

	Assembly	Comment	mem ops
	sum:		
	.LFB2:		
int array[10];	.loc 1 5 0		
	.LVL0:		
int sum(int count)	.loc 1 8 0		
{	movl \$0, %eax	s = 0	0
int s = 0;	.LVL1:		
int i;	testl %edi, %edi	???	0
for(i = 0; i < count; i++) {	jle .L4	???	0
s += array[i];	movl \$0, %eax	s = 0	0
}	movl \$0, %ecx	i = 0; is in a 32 l	0
return s;	.LVL2:		
}	movl \$0, %edx	t1 = 0, this is an address (64 bits)	0
	.L5:		
	.loc 1 9 0		
	addl array(,%rdx,4), %eax	s += array[1]	1
	.loc 1 8 0		
	addl \$1, %ecx	i++	0
	addq \$1, %rdx	t1++	0
	cmpl %edi, %ecx	i < count	0
	jne .L5		0
	.L4:		
	.LVL3:		
	.loc 1 12 0		
	rep ; ret		0
	.LFE2:		
	.size sum, .-sum		
	.comm array,40,32	allocate 40 bytes for array aligned at 32 byte boundary	

Good Spatial
Locality

byte



Long long int instead

arrayLoop2.c			
	.globl sum		
	.type sum, @function		
	sum:		
	.LFB2:		
	.loc 1 5 0		
	.LVL0:		
	.loc 1 8 0		
	movl \$0, %eax	s = 0	
	.LVL1:		
long long int array[10];	testl %edi, %edi	???	
	jle .L4	???	
int sum(int count)	movl \$0, %eax	s = 0	
{	movl \$0, %edx	i = 0	
int s = 0;	.LVL2:		
long long int i;	movslq %edi,%rcx	cast count to a long long int	
for(i = 0; i < count; i++) {	.LVL3:		
s += array[i];	.L5:		
}	.loc 1 9 0		
return s;	addl array(,%rdx,8),%eax	now x8 instead x4	1
}	.loc 1 8 0		
	addq \$1, %rdx	i++	
	cmpq %rcx, %rdx	i < count	
	jne .L5		
	.LVL4:		
	.L4:		
	.LVL5:		
	.loc 1 12 0		
	rep ; ret		
	.LFE2:		
	.size sum, .-sum		
	.comm array,80,32	2x the bytes	

Structs

How big is aStruct?

24 bytes!

0	a	First
1	a	
2	a	
3	a	
4	b	Second
5	b	
6	b	
7	b	
8	c	Third
9	padding for	
10	padding for	
11	padding for	
12	padding for	
13	padding for	
14	padding for	
15	padding for	
16	d	Fourth
17	d	
18	d	
19	d	
20	d	
21	d	
22	d	
23	d	

Spatial locality?

Some good some bad

struct.c			mem ops
	.globl sum		
struct aStruct{	.type sum, @function		
int a;	sum:		
int b;	.LFB2:		
char c;	.loc 1 9 0		
long long int d;	.LVL0:	s == rdi	
};	.loc 1 13 0		
	movl (%rdi), %eax	t = 0; t += s->a	1
int sum(struct aStruct * s) {	addl 4(%rdi), %eax	t += s->b	1
	.LVL1:		
int t = 0;	movsbl 8(%rdi),%edx	cast s->c to long	1
t += s->a;	addl %edx, %eax	t += s->c	
t += s->b;	.LVL2:		
t += s->c;	addl 16(%rdi), %eax	t += s->d	1
t += s->d;	.loc 1 19 0		
	ret		
return t;			
}			

Structs

How big is aStruct?

24 bytes!

struct.c			mem ops
	.globl sum		
struct aStruct{	.type sum, @function		
int a;	sum:		
int b;	.LFB2:		
char c;	.loc 1 9 0		
long long int d;	.LVL0:	s == rdi	
};	.loc 1 13 0		
	movl (%rdi), %eax	t = 0; t += s->a	1
int sum(struct aStruct * s) {	addl 4(%rdi), %eax	t += s->b	1
	.LVL1:		
int t = 0;	movsbl 8(%rdi),%edx	cast s->c to long	1
t += s->a;	addl %edx, %eax	t += s->c	
t += s->b;	.LVL2:		
t += s->c;	addl 16(%rdi), %eax	t += s->d	1
t += s->d;	.loc 1 19 0		
	ret		
return t;			
}			

0	a	First
1	a	
2	a	
3	a	
4	b	Second
5	b	
6	b	
7	b	
8	c	Third
9	padding for	
10	padding for	
11	padding for	
12	padding for	
13	padding for	
14	padding for	
15	padding for	
16	d	Fourth
17	d	
18	d	
19	d	
20	d	
21	d	
22	d	
23	d	

Note the usefulness of the immediate for mem ops.

Spatial locality?
Some good some bad

2D Array

	sum:	
long long int array[10][10];	.LFB2:	
	.loc 1 5 0	
int sum(int x, int count)	.LVL0:	
{	.loc 1 8 0	
int s = 0;	movl \$0, %r8d	r8 == s
long long int i;	.LVL1:	
for(i = 0; i < count; i++) {	testl %esi, %esi	???
s += array[x][i];	jle .L4	???
}	movslq %edi, %rax	cast x to long long
return s;	leaq (%rax, %rax, 4), %rax	x = x + x*4
}	salq \$4, %rax	x *= 16, so x = 16x + x*64
	addq \$array, %rax	array+x
	movl \$0, %r8d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi, %rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += array[x][i]
	.loc 1 8 0	
	addq \$1, %rdx	i ++
	addq \$8, %rax	addr += 8
	cmpq %rcx, %rdx	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	
	.LFE2:	
	.size sum, .-sum	
	.comm array, 800, 32	

Step one entry in the array

The array is a contiguous chunk of 800 bytes

Good Spatial Locality



2D Array

	sum:	
long long int array[10][10];	.LFB2:	
	.loc 1 5 0	
int sum(int x, int count)	.LVL0:	
{	.loc 1 8 0	
int s = 0;	movl \$0, %r8d	r8 == s
long long int i;	.LVL1:	
for(i = 0; i < count; i++) {	testl %esi, %esi	???
s += array[x][i];	jle .L4	???
}	movslq %edi, %rax	cast x to long long
return s;	leaq (%rax, %rax, 4), %rax	x = x + x*4
}	salq \$4, %rax	x *= 16, so x = 16x + x*64
	addq \$array, %rax	array+x
	movl \$0, %r9d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi, %rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += array[x][i]
	.loc 1 8 0	
	addq \$1, %rdx	i ++
	addq \$8, %rax	addr += 8
	cmpq %rcx, %rdx	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	
	.LFE2:	
	.size sum, .-sum	
	.comm array, 800, 32	

Step one entry in the array

The array is a contiguous chunk of 800 bytes
Good Spatial Locality



2D Array

long long int array[10][10];	sum:	
	.LFB2:	
	.loc 1 5 0	
int sum(int x, int count)	.LVL0:	
{	.loc 1 8 0	
int s = 0;	movl \$0, %r8d	r8 == s
long long int i;	.LVL1:	
for(i = 0; i < count; i++) {	testl %esi, %esi	???
s += array[x][i];	jle .L4	???
}	movslq %edi, %rax	cast x to long long
return s;	leaq (%rax,%rax,4), %rax	x = x + x*4
}	salq \$4, %rax	x *= 16, so x = 16x + x*64
	addq \$array, %rax	array+x
	movl \$0, %r9d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi, %rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += array[x][i]
	.loc 1 8 0	
	addq \$1, %rdx	i ++
	addq \$8, %rax	addr += 8
	cmpq %rcx, %rdx	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	
	.LFE2:	
	.size sum, .-sum	
	.comm array,800,32	

$$x = (x + 4x) * 16$$

$$x = 16x + 64x$$

$$x *= 80$$

$$80 = 10 \times \text{sizeof}(\text{long long})$$

Step one entry in the array

The array is a contiguous chunk of 800 bytes

Good Spatial Locality

array + x*80

array + (x+10)*80



2D Array

long long int array[10][10];	sum:	
	.LFB2:	
	.loc 1 5 0	
int sum(int x, int count)	.LVL0:	
{	.loc 1 8 0	
int s = 0;	movl \$0, %r8d	r8 == s
long long int i;	.LVL1:	
for(i = 0; i < count; i++) {	testl %esi, %esi	???
s += array[x][i];	jle .L4	???
}	movslq %edi, %rax	cast x to long long
return s;	leaq (%rax,%rax,4), %rax	x = x + x*4
}	salq \$4, %rax	x *= 16, so x = 16x + x*64
	addq \$array, %rax	array+x
	movl \$0, %r8d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi, %rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += array[x][i]
	.loc 1 8 0	
	addq \$1, %rdx	i++
	addq \$8, %rax	addr += 8
	cmpq %rcx, %rax	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	
	.LFE2:	
	.size sum, .-sum	
	.comm array, 800, 32	

$$x = (x + 4x) * 16$$

$$x = 16x + 64x$$

$$x * = 80$$

$$80 = 10 \times \text{sizeof}(\text{long long})$$

Step one entry in the array

The array is a contiguous chunk of 800 bytes

Good Spatial Locality

array + x*80

array + (x+10)*80



2D Array

long long int array[10][10];	sum:	
	.LFB2:	
	.loc 1 5 0	
int sum(int x, int count)	.LVL0:	
{	.loc 1 8 0	
int s = 0;	movl \$0, %r8d	r8 == s
long long int i;	.LVL1:	
for(i = 0; i < count; i++) {	testl %esi, %esi	???
s += array[x][i];	jle .L4	???
}	movslq %edi, %rax	cast x to long long
return s;	leaq (%rax,%rax,4), %rax	x = x + x*4
}	salq \$4, %rax	x *= 16, so x = 16x + x*64
	addq \$array, %rax	array+x
	movl \$0, %r9d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi, %rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += array[x][i]
	.loc 1 8 0	
	addq \$1, %rdx	i++
	addq \$8, %rax	addr += 8
	cmpq %rcx, %rax	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	
	.LFE2:	
	.size sum, -sum	
	.comm array,800,32	

$$x = (x + 4x) * 16$$

$$x = 16x + 64x$$

$$x * = 80$$

$$80 = 10 \times \text{sizeof}(\text{long long})$$

Step one entry in the array

The array is a contiguous chunk of 800 bytes

Good Spatial Locality

array + x*80

array + (x+10)*80



2D Array #2

nestLoop2.c	.globl sum	
	.type sum, @function	
long long int array[5][5];	sum:	
	.LFB2:	
int sum(int x, int count)	.loc 1 5 0	
{	.LVL0:	
int s = 0;	.loc 1 8 0	
long long int i;	movl \$0, %r8d	s = 0
for(i = 0; i < count; i++) {	.LVL1:	
s += array[i][x];	testl %esi, %esi	???
}	jle .L4	???
return s;	movslq %edi,%rax	cast x to long long
}	leaq array(,%rax,8), %rax	t1 = x * 8 + array
	movl \$0, %r8d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi,%rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += *t1
	.loc 1 8 0	
	addq \$1, %rdx	i++
	addq \$40, %rax	addr += 5*8 (skip one row of the matrix)
	cmpq %rcx, %rdx	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	

The offset into one row of the array

Poor Spatial Locality



2D Array #2

nestLoop2.c	.globl sum	
	.type sum, @function	
long long int array[5][5];	sum:	
	.LFB2:	
int sum(int x, int count)	.loc 1 5 0	
{	.LVL0:	
int s = 0;	.loc 1 8 0	
long long int i;	movl \$0, %r8d	s = 0
for(i = 0; i < count; i++) {	.LVL1:	
s += array[i][x];	testl %esi, %esi	???
}	jle .L4	???
return s;	movslq %edi,%rax	cast x to long long
}	leaq array(,%rax,8), %rax	t1 = x * 8 + array
	movl \$0, %r8d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi,%rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += *t1
	.loc 1 8 0	
	addq \$1, %rdx	i++
	addq \$40, %rax	addr += 5*8 (skip one row of the matrix)
	cmpq %rcx, %rdx	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	

The offset into one row of the array

Poor Spatial Locality



2D Array #2

nestLoop2.c	.globl sum	
	.type sum, @function	
long long int array[5][5];	sum:	
	.LFB2:	
int sum(int x, int count)	.loc 1 5 0	
{	.LVL0:	
int s = 0;	.loc 1 8 0	
long long int i;	movl \$0, %r8d	s = 0
for(i = 0; i < count; i++) {	.LVL1:	
s += array[i][x];	testl %esi, %esi	???
}	jle .L4	???
return s;	movslq %edi, %rax	cast x to long long
}	leaq array(,%rax,8), %rax	t1 = x * 8 + array
	movl \$0, %r8d	s = 0
	movl \$0, %edx	i = 0
	.LVL2:	
	movslq %esi, %rcx	cast count to a long long int
	.LVL3:	
	.L5:	
	.loc 1 9 0	
	addl (%rax), %r8d	s += *t1
	.loc 1 8 0	
	addq \$1, %rdx	i++
	addq \$40, %rax	addr += 5*8 (skip one row of the matrix)
	cmpq %rcx, %rdx	
	jne .L5	
	.LVL4:	
	.L4:	
	.loc 1 12 0	
	movl %r8d, %eax	
	ret	

The offset into one row of the array

Poor Spatial Locality

