Structures

① Structures

② Data Alignment

Structures

```
struct rec {
    int i;
    int j;
    int a[2];
    int *p;
};
```

```
movl        (%rdi),%eax
movl        %eax,4(%rdi)
```

```
leaq        8(%rdi,%rsi,4),%rax
```

```
movl        4(%rdi),%eax
addl        (%rdi),%eax
cltq
leaq        8(%rdi,%rax,4),%rax
movq        %rax,16(%rdi)
```

0	4	8	16	24

r | i | j | a[0] | a[1] | p |

Example

```
void sp_init(struct prob *sp) {
    sp->s.x   = ___sp->s.y___;
    sp->p     = _&(sp->s.x)_;
    sp->next  = _____sp_____;
}
```

```
struct prob {
    int *p;
    struct {
        int x;
        int y;
    } s;
    struct prob *next;
};
```

```
sp_init:
    movl  12(%rdi),%eax
    movl  %eax,8(%rdi)
    leaq  8(%rdi),%rax
    movq  %rax,(%rdi)
    movq  %rdi,16(%rdi)
    ret
```

0		8	12	16		24

prob	p	s.x	s.y	next

Example
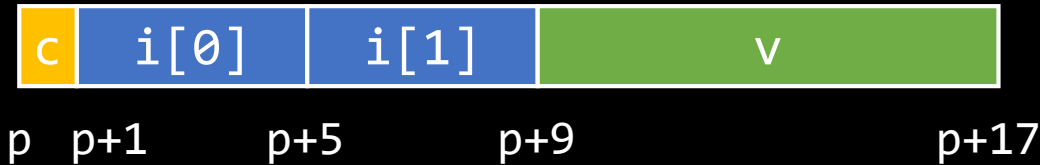
```
fun:
    movl    $0,%eax
    jmp     .L2
.L3:
    addq    (%rdi),%rax
    movq    8(%rdi),%rdi
.L2:
    testq %rdi,%rdi
    jne     .L3
    rep; ret
```

```
long fun(struct ELE *ptr)

long ret=0;
while(ptr) {
    ret += ptr->v;
    ptr = ptr->p;
}
return ret;
```

```
struct ELE {
    long v;
    struct ELE *p;
};
```
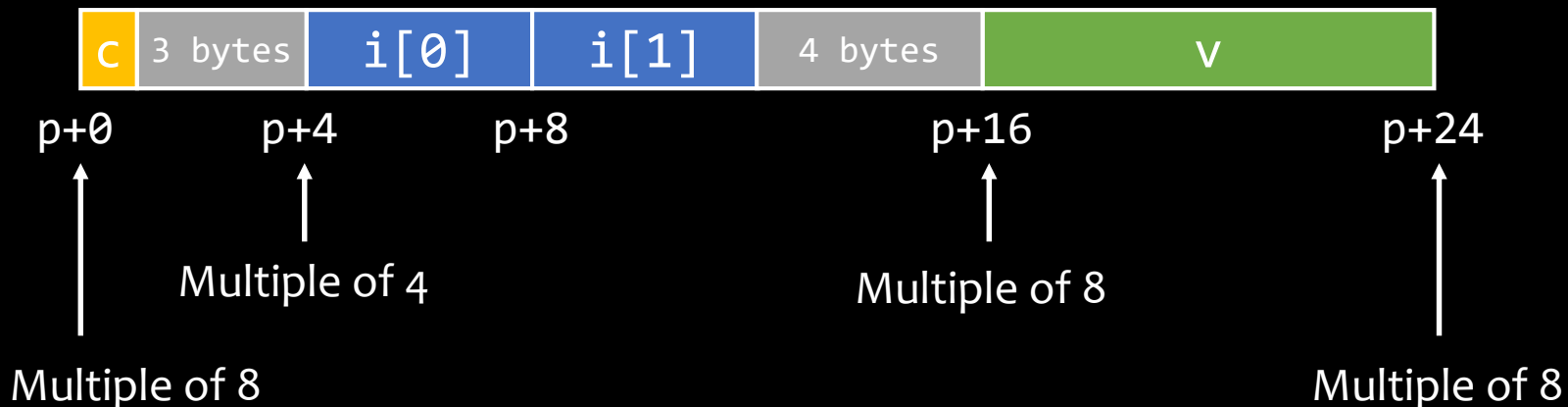
Data Alignment

```
c    i[0]    i[1]         v
```

p p+1   p+5   p+9   p+17

```
struct S1 {
    char c;
    int i[2];
    double v;
} *p;
```

① Primitive data type requires K bytes

② Address must be multiple of K

padding

```
c  3 bytes   i[0]    i[1]   4 bytes        v
```

p+0   p+4   p+8   p+16   p+24

Multiple of 4

Multiple of 8

Multiple of 8

Multiple of 8

Why?

c	i[0]	i[1]	v

1 byte: `char`
2 bytes: `short`
4 bytes: `int, float`
8 bytes: `double, long, char *`
16 bytes: `long double`

bad

good

c	3 bytes	i[0]	i[1]	4 bytes	v

Example

```
struct S1 {
    int i;
    char c;
    int j;
};
```

③ Overall structure must
be multiple of K

④ Put large data types first

```
struct S2 {
    int i;
    int j;
    char c;
};
```

```
struct S2 d[4];
```
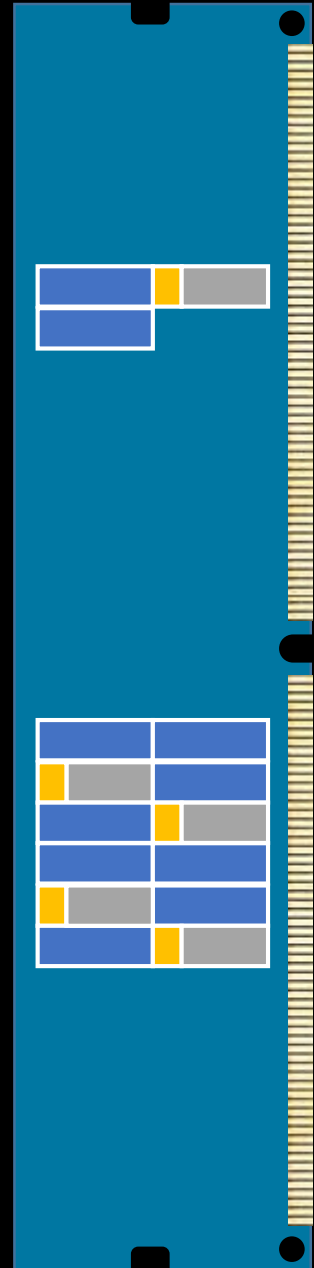
Saving space

```
struct S4 {
    char c;
    int i;
    char d;
} *p;
```

```
struct S5 {
    int i;
    char c;
    char d;
} *p;
```

`struct S4 d[8];`

④ Put large data types first

| c | 3 | i | d | 3 |

| i | c | d | 2 |

`struct S5 d[8];`
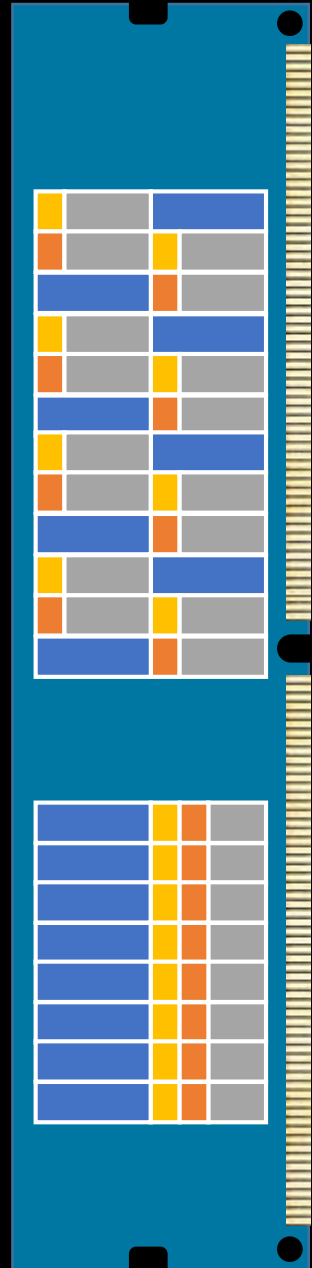
Example

```
struct P1{ int i; char c; int j; char d; };
```



```
struct P2{ int i; char c; char d; long j; };
```



```
struct P3{ short w[3]; char c[3]; };
```



```
struct P4{ short w[5]; char *c[3]; };
```



```
struct P5{ struct P3 a[2]; struct P2 t };
```

Summary

- Structures
- Data Alignment

Eric Steven Raymond

Author of "The Cathedral and the Bazaar"

" Smart data structures and dumb code works a lot better than the other way around. "