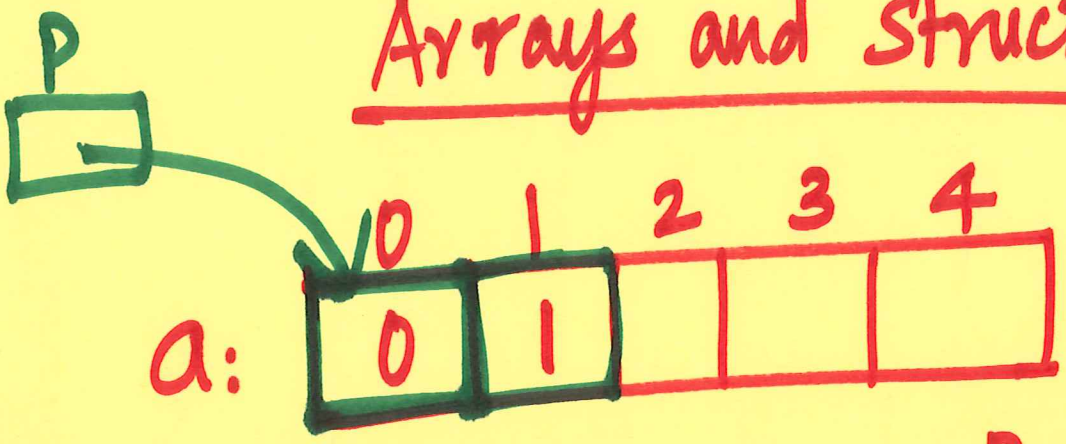
Arrays and Structs



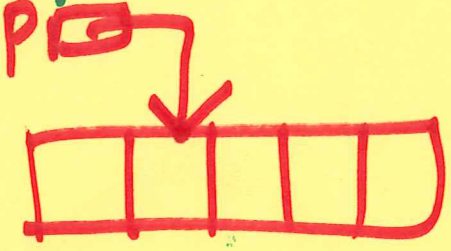

```
int *p;
```

```
p = &a[0];
(or)
p = a;
```

```
a[0] = 0;
a[1] = 1;
```

```
*(p+0) = 0;
*(p+1) = 1;
```

```
p = p+1;
```



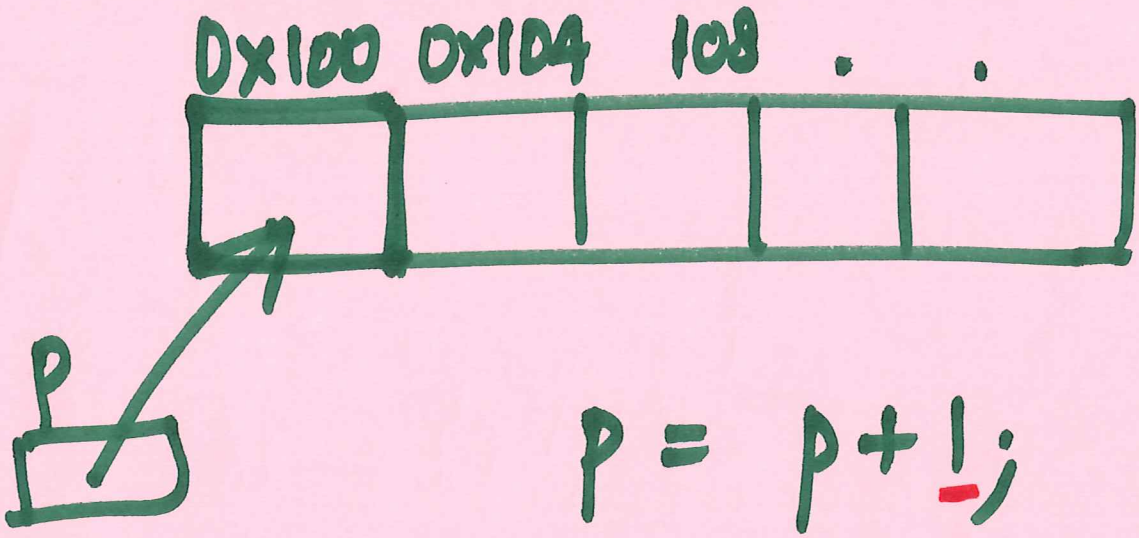
✓

```
*(a+1) = 1;
```

&a[0]

```
a = a+1;
```

compiler error.

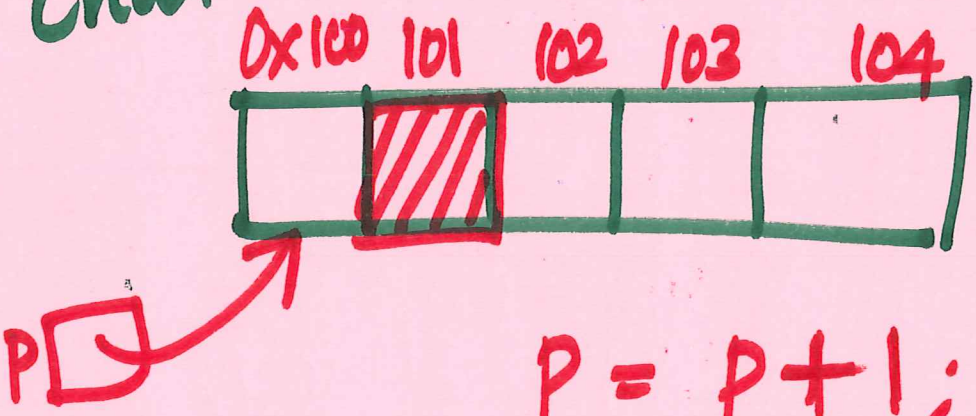


$p = p + 1;$

$p = 0x100 + 4$

$p = 0x104$

char



$p = p + 1;$

$p = 0x100 + 1;$

$p = 0x101$

short – scaling factor = 2.

data type	scaling fact
char	1
short	2
int	4

movl %eax, -8(%ebp, %edx, 4)

Imm(Rb, Ri, s)

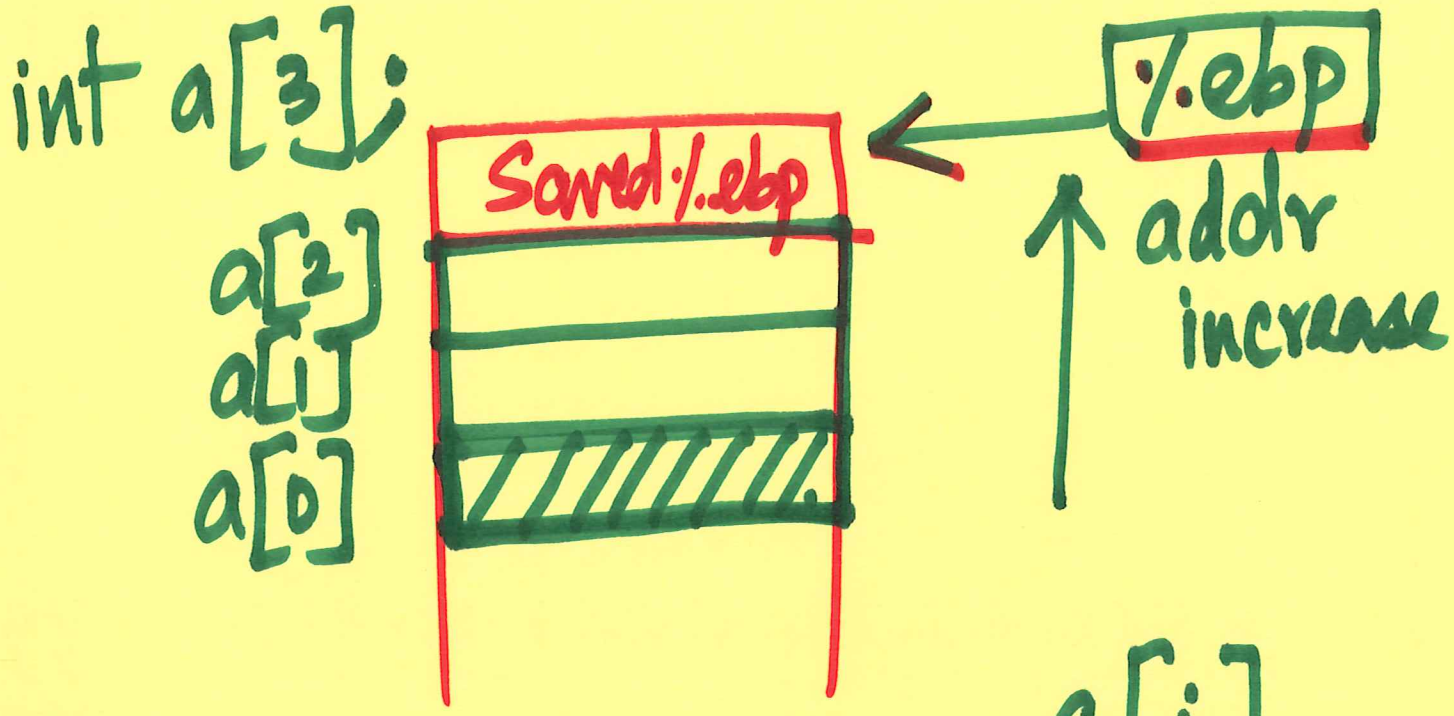
M[-8 + 0x100 + 4]

=

M[Imm + Rb + Ri · s]

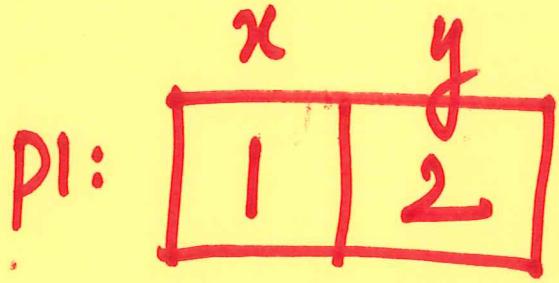
ebp 0x100

edx 1

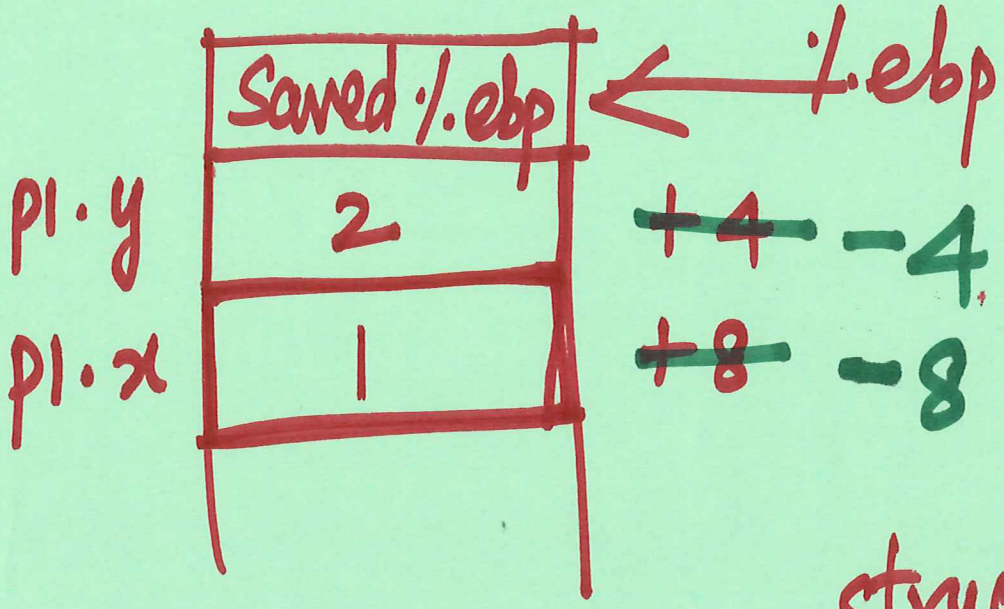


Structures

```
struct point
{   int x;
    int y;
};
```

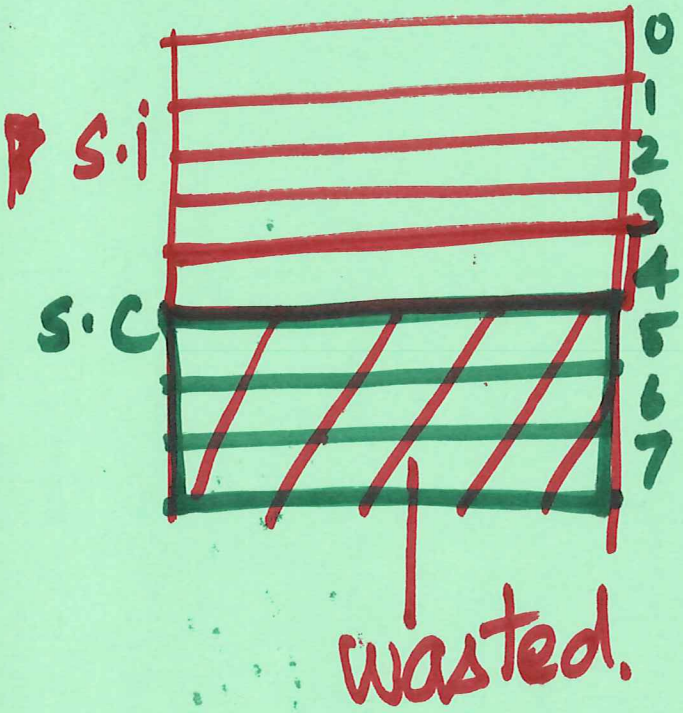


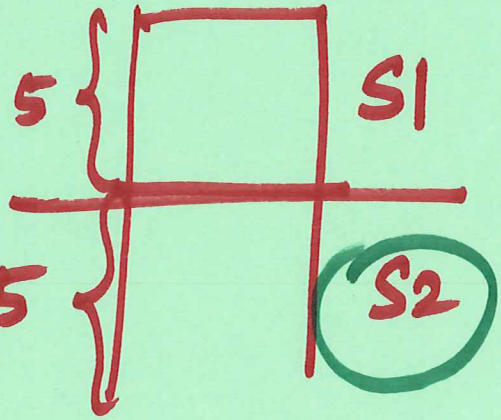

```
struct point p1; // local variable
p1.x = 1;
p1.y = 2;
```



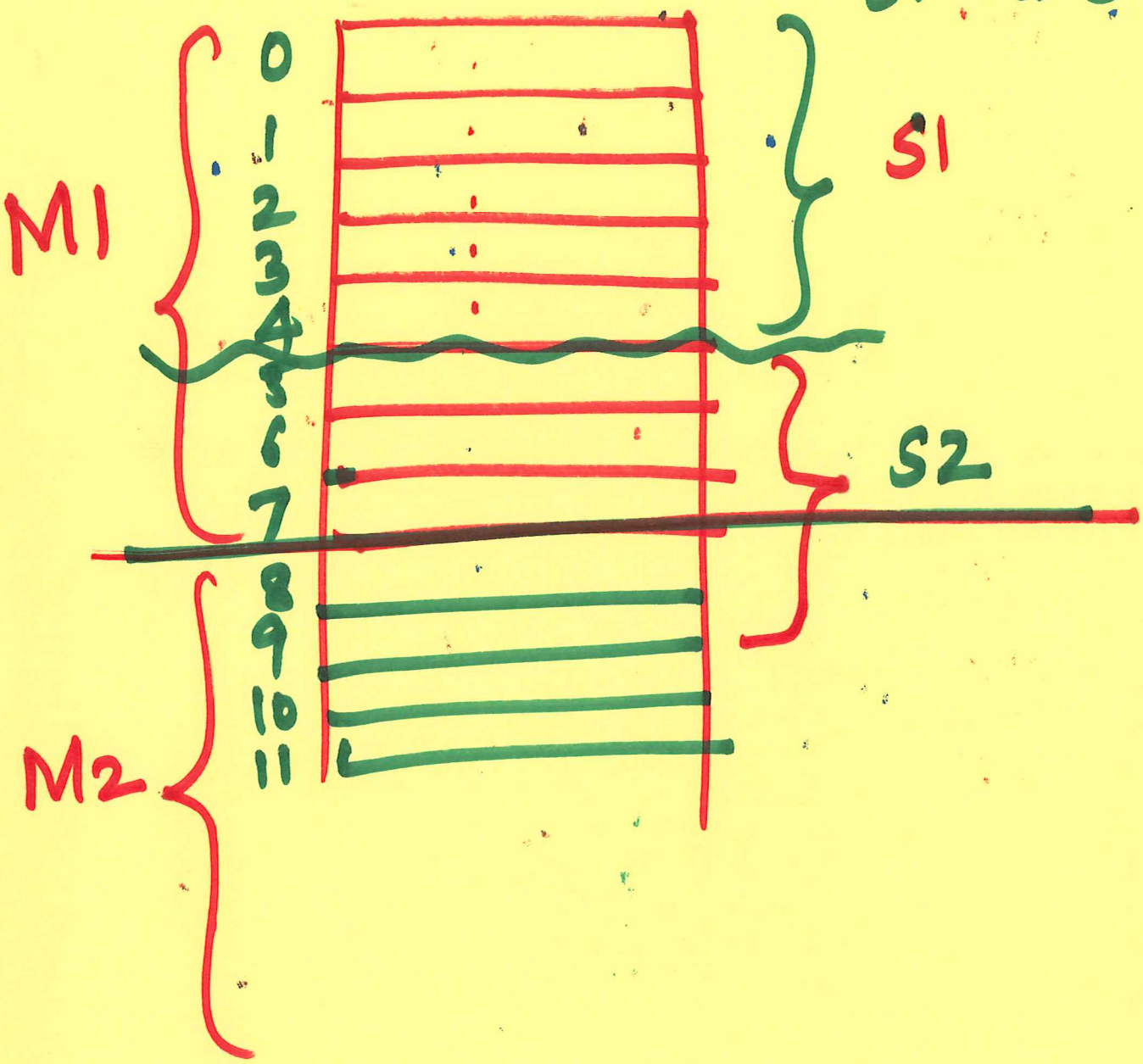
struct s
{
int i; — 4
char c; — 1
};

4 bytes





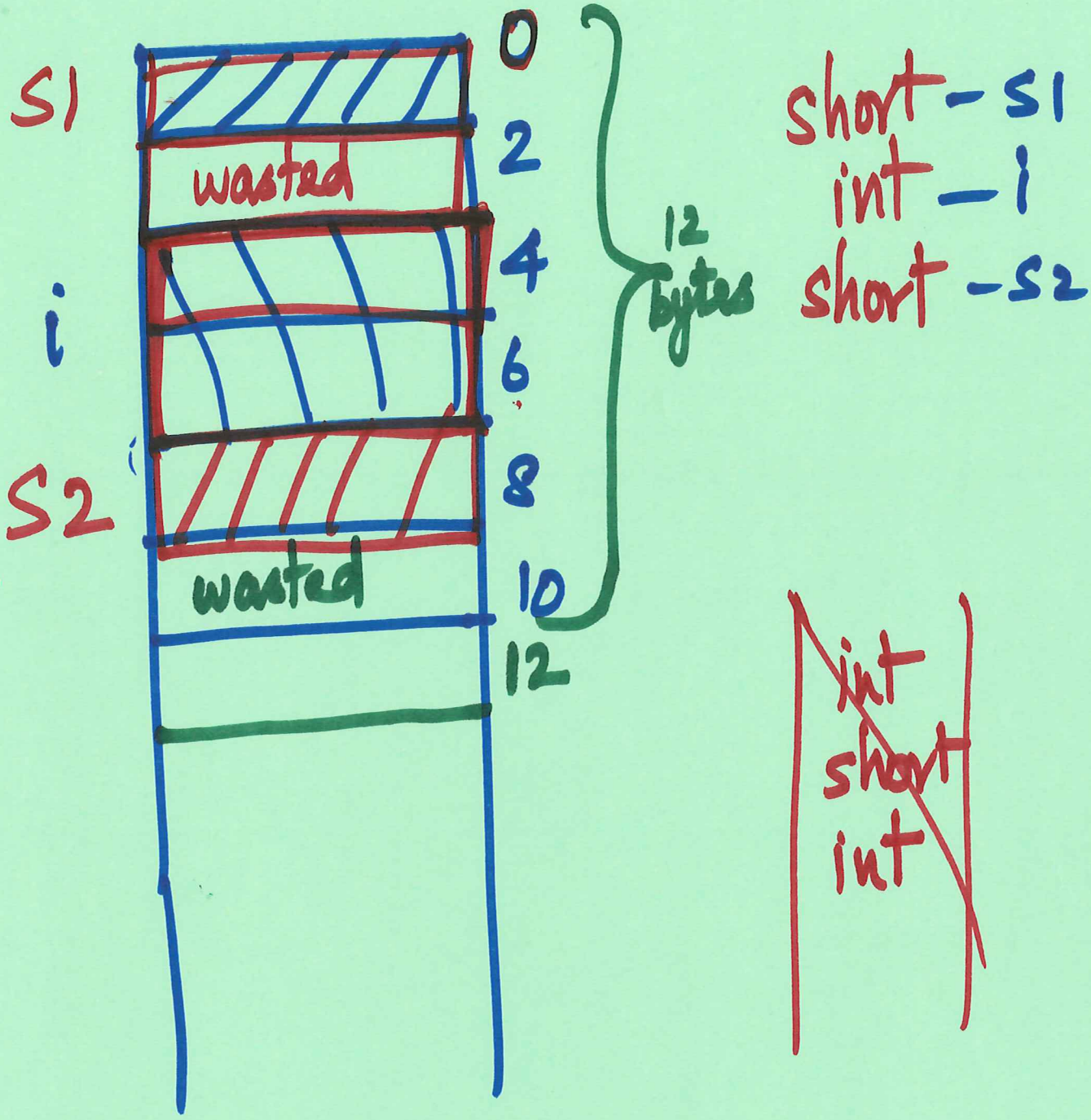
struct s s1;



leave

movl %.ebp, %.esp

popl %.ebp



int - address - div by 4.

short - " - div by 2.